INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Masterarbeit

Schwachstellenmanagement in Hochschulnetzen am Beispiel des Münchner Wissenschaftsnetzes

Michael Steinke

INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Masterarbeit

Schwachstellenmanagement in Hochschulnetzen am Beispiel des Münchner Wissenschaftsnetzes

Michael Steinke

Aufgabensteller:	Priv.-Doz. Dr. Wolfgang Hommel
Betreuer:	Dipl.-Inform. Stefan Metzger Dipl.-Math. Felix von Eye
Abgabetermin:	16. Dezember 2015

Hiermit versichere ich, dass ich die vorliegende Masterarbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

München, den 16. Dezember 2015

. . . . . .
(Unterschrift des Kandidaten)

Zusammenfassung

Der beispiellose Umfang an in Hochschulnetzen gehaltenen und kommunizierten Daten macht eine gleichartig erschöpfende Ergreifung von Maßnahmen zum Erhalt der Informationssicherheit unbedingt notwendig. Hochschulrechenzentren als Betreiber von Hochschulnetzen setzen dabei üblicherweise auf eine Vielzahl technischer Lösungen wie *Intrusion Detection Systeme*, *Intrusion Prevention Systeme* oder *Firewalls* und übersehen den eigentlichen Aspekt von Angriffen, der diese erst möglich macht – nämlich Schwachstellen in Software und deren Konfiguration.

Mit potenziell mehreren zehntausend aktiven Nutzern und einer von zentraler Stelle aus unüberblickbaren Anzahl an im Netz angebotenen Diensten und betriebenen Geräten, steht die Planung und Realisierung eines geregelten Umgangs mit Schwachstellen vor enormen Herausforderungen, die weit über den Einsatz von auf dem Markt erhältlichen Schwachstellen-Scannern hinausgehen. Insbesondere in der Umgebung eines Hochschulnetzes gibt es keine konkreten Anforderungen und Überlegungen bezüglich des Inhalts eines Schwachstellenmanagements – Beschreibungen aus vorhandenen, einsehbaren Konzepten vermitteln diesbezüglich nur einen stark abstrahierten Eindruck.

Der Hauptteil der Arbeit beschreibt eine konkrete Ausarbeitung eines Konzepts eines Schwachstellenmanagements in Hochschulnetzen, das gleichzeitig ausreichend allgemein beschrieben ist, um in einer beliebigen Hochschulumgebung anwendbar zu sein. Die Grundlage des Konzeptinhalts bilden Charakteristika und Herausforderungen in der Hochschulumgebung, deren Zusammenstellung ebenso Teil dieser Arbeit ist. Dabei werden durchgehend die Aktivitäten der Identifikation, Klassifikation, Beseitigung und Abschwächung sowie der Prävention von Schwachstellen betrachtet. Das Konzept wird zudem mit Anforderungen abgeglichen, die auf Basis eines auf dem *Münchner Wissenschaftsnetz* basierenden Szenarios sowie den diesbezüglichen Erfahrungen am *Leibniz-Rechenzentrum* an ein Schwachstellenmanagement festgelegt werden. Auch werden bereits bestehende Konzepte und Lösungen zum Schwachstellenmanagement berücksichtigt, welche ebenfalls durch Abgleich der Erfüllung der Anforderungen auf ihre Eignung zur Umsetzung in Hochschulnetzen hin überprüft werden.

Die Arbeit umfasst außerdem die Beschreibung einer Implementierung der im Konzept beschriebenen technischen Komponenten des Schwachstellenmanagements sowie veranschaulichend, verschiedene Fallbeispiele im Rahmen eines exemplarischen Durchlaufs durch das Konzept.

Inhaltsverzeichnis

1 Einleitung

Der Informationssicherheit wird heutzutage mehr Bedeutung zugemessen denn je. Beinahe monatlich wird man von den Medien auf neue schwerwiegende Sicherheitsvorfälle in Unternehmen und Einrichtungen aufmerksam gemacht, die nicht selten mit dem Verlust oder der unberechtigten Offenlegung von Millionen von kritischen Daten in Verbindung stehen. In der letzten Zeit bekannt gewordene Vorfälle sind beispielsweise ein Cyberangriff auf die US-amerikanische Bank *JPMorgan Chase* im Sommer 2014, bei dem persönliche Daten von 76 Millionen Privatkunden und 7 Millionen Firmenkunden offengelegt wurden [Gie14] sowie der Angriff auf die Informationsinfrastruktur des *Deutschen Bundestages* im Mai 2015, bei dem es zu erheblichen Störungen im Betrieb und einer unberechtigten Offenlegung von Daten kam. [And15]

Aber auch Einrichtungen im Ausbildungssektor bleiben nicht von Datenpannen verschont – laut *Privacy Rights Clearinghouse* (PRC), einer gemeinnützigen Organisation mit dem Ziel, Personen bezüglich des Schutzes ihrer Privatsphäre zu informieren, traten in den Jahren 2005 bis 2014 alleine 747 veröffentlichte Verletzungen im Umgang mit Daten an Ausbildungseinrichtungen in den USA auf, und liegt damit hinter dem medizinischen Sektor auf dem zweiten Platz der häufigsten Datenpannen. [EDU14] Bei dieser Auswertung wurden gemeldete Vorfälle in sieben Industriebereichen betrachtet: Der Finanz- und Versicherungssektor, der Einzelhandelssektor und nicht in diese Kategorien passenden gewerbliche Unternehmen, genauso wie der Regierungs- und Militärbereich, das Gesundheitswesen, gemeinnützige Organisationen und der Ausbildungssektor.

Allgemein findet sich die Ursache von Vorfällen dieser Art in der Ausnutzung von Sicherheitslücken aufgrund technischer, organisatorischer oder auf menschliches Fehlverhalten zurückzuführender Schwachstellen. Typische Beispiele für solche Schwachstellen sind Implementierungs- und Konfigurationsfehler in Software, unzureichende Zugriffskontrollen oder auch fahrlässiger Umgang mit kritischen Daten.
Ein bewusstes Arbeiten mit Schwachstellen im Rahmen eines strukturierten Schwachstellenmanagements kann insofern zu einem besseren Verständnis für Sicherheit und Sicherheitslücken führen und gleichzeitig die Verwundbarkeit der Komponenten im Netz deutlich verringern.

Hochschulen, als Anbieter verschiedenster IT-Dienste wie einem Vorlesungsverzeichnis, Diensten zur Organisation des Prüfungs- und Tutoriumsbetriebes, der zur Verfügungstellung von E-Mail-Diensten und vieler weiterer, aber auch als Erhebungsstelle kritischer, oftmals besonders schutzwürdiger personenbezogener Daten, müssen sich daher besonders um ein hohes Sicherheitslevel bemühen.

1.1 Motivation

Zu den Hauptzielen des Informationssicherheitsmanagements in Hochschulen und Forschungseinrichtungen gehört nicht zuletzt die Minimierung von Risiken, um einen möglichen Schaden zu verhindern, bevor er entsteht beziehungsweise die Schadenshöhe zu verringern. Ein solcher Schaden kann sich beispielsweise durch materielle Verluste, Nachteile für Servicenutzer, einer Störung des Betriebs oder auch einer Beeinträchtigungen der Reputation einer Einrichtung manifestieren. Generell entstehen Risiken durch die Kombination vorhandener Schwachstellen und Bedrohungsereignissen, die diese potenziell ausnutzen oder auslösen können, wobei die Eintrittswahrscheinlichkeit der Ausnutzung und die dadurch entstehende Auswirkung bzw. Schadenshöhe die Risikohöhe bestimmen. Da es oft nicht möglich ist, ein Risiko durch die Beseitigung von Bedrohungen zu minimieren, ist der gängigere und oft einzig mögliche Ansatz, vorhandene Schwachstellen zu beheben, um somit möglichen Bedrohungen keinen Angriffspunkt zu bieten.

Insofern nimmt der Umgang mit und insbesondere die Behebung von Schwachstellen als proaktive Maßnahme eine entscheidende Rolle zur Gewährleistung der Informationssicherheit ein. Ergänzend zur Prävention stehen die Detektion und reaktive Maßnahmen zur Behandlung von sicherheitsrelevanten Ereignissen, welche nicht konkurrierende, sondern ergänzende und aufeinander aufbauende Funktionalitäten einnehmen, jedoch durch ein effektives Schwachstellenmanagement deutlich entlastet werden können. In gewisser Weise kann die Behebung von Schwachstellen auch als reaktive Maßnahme angesehen werden. Dies ist insbesondere der Fall, wenn Verantwortliche auf eine offene Schwachstelle erst durch ihre Ausnutzung aufmerksam gemacht werden.

Aus dem Lagebericht des Jahres 2014 [Bun14a] des *Bundesamtes für Sicherheit in der Informationstechnik* (BSI) geht hervor, dass – auch wenn die absolute Anzahl der Softwareschwachstellen kürzlich gesunken ist – sich die Anzahl der kritischen Softwareschwachstellen auf einem relativ gleichbleibend hohen Level mit leicht steigender Tendenz befindet (siehe Abbildung 1.1) und sich um rund 700 jährliche Vorkommen bewegt.

Adobe Flash Player	Adobe Reader	Apple OS X
Apple Quicktime	Apple Safari	Google Chrome
Linux Kernel	Microsoft Internet Explorer	Microsoft Office
Microsoft Windows	Mozilla Firefox	Mozilla Thunderbird
Oracle Java/ JRE		

Tabelle 1.1: Softwareprodukte, die bei der Auswertung in Abbildung 1.1 berücksichtigt wurden. [Bun14a]

Zu beachten ist, dass sich diese Auswertung lediglich auf 13 sehr häufig genutzte Anwendungen – unter anderem den *Adobe Flash Player*, *Oracle Java* und den beliebtesten Webbrowsern *Microsoft Internet Explorer*, *Mozilla Firefox* und *Google Chrome* – aber auch Betriebssysteme wie *Linux*, *Apple OS X* und *Microsoft Windows* bezieht. Alle ausgewerteten Produkte sind in Tabelle 1.1 zu finden. Die tatsächlich vorhandenen Schwachstellen, insbe-

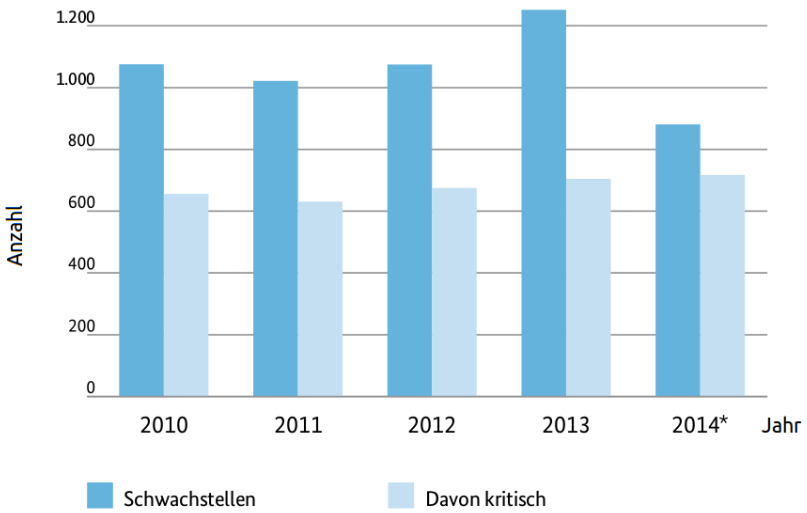


Abbildung 1.1: Anzahl der Softwareschwachstellen in 13 weit verbreiteten Anwendungen, angelehnt an Bundesamt für Sicherheit in der Informationstechnik – Die Lage der IT-Sicherheit in Deutschland 2014, S.13 (*Werte von 2014 ab September hochgerechnet) [Bun14a]

sondere in einem Hochschulnetz, wird diese Anzahl allein aufgrund des weiten Spektrums an eingesetzter Software deutlich übersteigen.

Abgesehen davon, dass die in Tabelle 1.1 aufgelisteten Produkte trotz hoher Nutzung nur einen minimalen Teil der Softwareprodukte auf dem Markt abdecken, liegt der Fokus der aufgelisteten Produkte ausschließlich auf Client-Anwendungen und schließt allein dadurch einen Großteil der Software in Hochschulnetzen aus: Als IT-Dienstanbieter ist insbesondere in Hochschulrechenzentren eine große Anzahl an Server-Software zu finden, welche wiederum viele weitere Tools, Softwareplugins und Softwarebibliotheken nutzen. Auch darf der Forschungsaspekt in Hochschulnetzen und den darin befindlichen Geräten und Nutzern, Wissenschaftler, Mitarbeiter und Studenten, welche Software jeglicher Versionen in jedem erdenklichen Teilbereich der Kunst und Forschung im Einsatz haben, nicht vergessen werden.

Ein strukturierter Umgang mit Schwachstellen soll aber nicht nur zu einer Verbesserung des Sicherheitsniveaus führen, sondern vor allem auch unter dem Aspekt einer höheren Effizienz zur Schonung von Ressourcen, insbesondere dem Einsparen von manuellem Aufwand für Mitarbeiter durch die Definition strukturierter Vorgänge, Zuständigkeiten und Verfahren beitragen.

1.2 Zielsetzung und Ausrichtung

Die Zielsetzung dieser Arbeit ist es, ein Konzept und eine exemplarische Umsetzung eines Schwachstellenmanagements auszuarbeiten, das die Anforderungen eines Hochschulnetzes und seiner Charakteristika erfüllt.

Damit das Schwachstellenmanagement diesen Anforderungen entspricht, werden zunächst die Charakteristika eines Hochschulnetzes anhand eines Vergleichs mehrerer Hochschulnetze ausgearbeitet sowie, daran angelehnt, die Herausforderungen, die bei der Umsetzung eines Schwachstellenmanagements in einer solchen Umgebung auftreten, verdeutlicht. Als Grundlage für das Konzept eines Schwachstellenmanagements in Hochschulnetzen dient – aufgrund seiner guten Beschreibung – das *Münchner Wissenschaftsnetz* und bewährte Maßnahmen im Umgang mit Schwachstellen, die sich am *Leibniz-Rechenzentrum* etabliert haben.

Der Hauptfokus richtet sich dabei auf bekannte Softwareschwachstellen, das heißt, Schwächen wie Implementierungsfehler oder Fehlkonfigurationen, deren Ausnutzung zu Schaden für Nutzer oder Einrichtungen führen kann. Die Kernfunktionen eines solchen Systems umfassen das Sammeln und eine effektive Organisation bekannter Schwachstellen aus unterschiedlichen Informationsquellen, die Priorisierung der Schwachstellen mit Hilfe eines geeigneten Bewertungssystems, geeignete Wege zur Meldung von Schwachstellen sowie Ergebnissen, Verfahren zur Detektion dokumentierter Schwachstellen, die Berichterstattung und Behandlung von Schwachstellen. Außerdem werden Möglichkeiten zur Prävention von Schwachstellen betrachtet.

Darüber hinaus soll innerhalb des Konzepts ein sinnvoller organisatorischer Rahmen erstellt werden, der die Maßnahmen und Abläufe zu einem strukturierten und effektiven Umgang mit Schwachstellen zusammenfasst. Dazu gehört unter anderem die Beachtung existierender Rollen und Einteilung von Verantwortlichkeiten sowie Verfahren innerhalb der einzelnen Aktivitäten des Schwachstellenmanagements.

Eine exemplarische Anwendung des Konzepts soll schließlich am Beispiel des *Münchner Wissenschaftsnetzes* vorgenommen werden, um das Zusammenspiel der einzelnen Phasen im Schwachstellenmanagement und in Verbindung mit anderen Prozessen zu verdeutlichen. Zum Inhalt der Arbeit gehört auch die Implementierung ausgewählter Teile des Konzepts, die bei der Veranschaulichung helfen und die organisatorischen Aspekte unterstützen sollen.

1.3 Aufbau der Arbeit

Die Grundlagen zu dieser Arbeit werden im folgenden Kapitel beschrieben. Dazu gehört die Festlegung des Fokus in Bezug auf betrachtete Schwachstellen und Aktivitäten, eine Beschreibung der Hochschulumgebung sowie notwendiges Wissen zum Thema IT-Service-Management.

Darauf aufbauend werden in Kapitel 3 Anforderungen an ein Schwachstellenmanagement in Hochschulnetzen gestellt, die im Verlauf der Arbeit als Leitbild und Möglichkeit zur Bewertung von Konzepten und Umsetzungen dienen. Die Anforderungen werden auf Basis zweier Szenarien, dem *Münchner Wissenschaftsnetz* (MWN) sowie dem aktuellen Umgang mit Schwachstellen am *Leibniz-Rechenzentrum*, dem Betreiber des MWN, erstellt und ebenfalls in diesem Kapitel beschrieben.

In Kapitel 4 werden bereits vorhandene Konzepte und Arbeiten im Bereich Schwachstellenmanagement erläutert und auf ihre Tauglichkeit bezüglich einer Realisierung in Hochschulnetzen hin überprüft.

Schließlich wird in Kapitel 5 ein Konzept für ein Schwachstellenmanagement beschrieben, das auf die Anforderungen und Herausforderungen in Hochschulnetzen angepasst ist.
Inwieweit das Konzept die aus Kapitel 3 gestellten Anforderungen erfüllt, wird am Ende des Kapitels ausgewertet.

In Kapitel 6 werden Implementierungsdetails von Kernelementen der technischen Komponenten des Schwachstellenmanagements gezeigt und erläutert.

Das Zusammenspiel des Konzepts wird in Kapitel 7 anhand einer exemplarischen Anwendung im Münchner Wissenschaftsnetz verdeutlicht, indem das Schwachstellenmanagementsystem beispielhaft umgesetzt und in Verbindung mit darin befindlichen technischen Komponenten aus Kapitel 6 in verschiedenen Szenarien durchlaufen wird.

In Kapitel 8 werden die Ergebnisse der Arbeit zusammengefasst und auf weiterführende wissenschaftliche Arbeiten und Fragestellungen hingewiesen.

Im Anhang sind zudem ein Glossar mit häufig verwendeten Begriffen im Zusammenhang mit Schwachstellenmanagement sowie weitere Dokumente, die im Verlauf der Arbeit erläutert werden.

2 Grundlagen

Das Spektrum an Schwachstellen, die es im Hochschulbereich gibt, ist riesig und unübersichtlich. Praktisch in jedem Organisationsbereich existieren unzählige Schwachstellen, deren Ausnutzung zu Schaden für die jeweilige Institution führen kann. Beispiele sind fehlende Zugangskontrollen zu Räumlichkeiten mit besonderem Sicherheitsbedarf wie Mitarbeiterbüros sowie Lager- und Serverräumen, Fehlverhalten von Mitarbeitern, Baumängel, organisatorische Schwächen oder Fehler in Software und Hardware der Computersysteme und darauf eingesetzten fehlkonfigurierten Anwendungen.

Dieses Kapitel soll eine Grundlage schaffen und den Rahmen festlegen, um das Verständnis dieser Arbeit zu erleichtern. Daher wird zunächst der Rahmen des Schwachstellenmanagements in Form der Art der betrachteten Schwachstellen, des durch das Schwachstellemanagement angestebten Ziels und der dazugehörigen Aktivitäten festgelegt. Um ein Grundverständnis für die einzelnen Ansatzpunkte im Umgang mit Schwachstellen zu schaffen, wird außerdem der Lebenszyklus von Schwachstellen beschrieben. Abschließend wird die Umgebung eines Hochschulnetzes erläutert, in der das Schwachstellenmanagement umgesetzt werden soll. Zum einen durch die Vermittlung der Beschaffenheit des Netzes und dessen Organisation, sowie zum anderen der Herausforderungen, die sich daraus für das Schwachstellenmanagement ergeben.

2.1 Schwachstellen in Computersystemen

> Eine **Schwachstelle** ist eine Schwäche in einem Informationssystem, in sicherheitsrelevanten Prozeduren eines Systems, internen Kontrollen oder der Implementierung, die durch eine Bedrohung ausgelöst oder ausgenutzt werden kann. [Kis13] Oft ist die Ausnutzung einer Schwachstelle mit einem daraus resultierenden potenziellen Schaden verbunden.

Die Definition von Schwächen ist in diesem Zusammenhang abhängig von der Art der betrachteten Schwachstellen. Mit Fokus auf Software, können Schwächen beispielsweise in Form von Fehlern im Quelltext oder dem Softwaredesign auftreten. Der Unterschied zwischen Schwächen und Schwachstellen liegt insofern in der Ausnutzbarkeit und der Möglichkeit, dass in irgendeiner Form Schaden oder ein ungewollter Zustand durch die Ausnutzung entsteht.

Da sich diese Arbeit auf Schwachstellen im Netz und insbesondere im Hochschulnetz bezieht, liegt der Fokus auf Schwachstellen, die seit Beginn der Nutzung von Computersystemen und Rechnernetzen zur Unterstützung der Geschäftsprozesse einer Organisation unzählige Male Ziel von Angriffen wurden: **Softwareschwachstellen und Fehlkonfiguration**.

In Abschnitt 2.3 werden einige bewährte Vorgehensweisen (engl. „Good Practices“) im Umgang mit Softwareschwachstellen und deren Charakteristika vorgestellt und anhand eines als *Heartbleed Bug* bekannt gewordenen Fehlers in *OpenSSL* [Ope15b], einer weltweit genutzten Implementierung der *Secure Socket Layer* (SSL) bzw. *Transport Layer Security* (TLS) Protokolle, erläutert. Aufgrund der großen Verbreitung von Webserver-Software, die OpenSSL verwenden, darunter der *Apache HTTP Server* und *nginx*, betraf die Schwachstelle zum Zeitpunkt ihrer Entdeckung potenziell mehr als 66% aller Websites im Internet. [Cod14]

Schwachstellen in der Konfiguration haben in der Regel die Eigenschaft, dass sie nicht so leicht erkennbar und definiert sind wie Softwareschwachstellen und vielmehr abhängig von der Umgebung sind, in der sie auftauchen. Es ist oft notwendig, beispielsweise die Rolle, Aufgabe und den Schutzbedarf der verarbeiteten Daten eines Computersystems festzulegen, um fehlerhafte oder auch ungewollte Zustände zu erkennen.

Beispielsweise ist die unverschlüsselte Übertragung einer öffentlichen Webpräsenz einer Fakultät über das Web (via *http*) eher als unbedenklich einzustufen. Werden hingegen personenbezogene oder allgemein sensible Daten unverschlüsselt über einen Webserver verbreitet, so ist das Fehlen einer angemessenen Verschlüsselung (beispielsweise unter der Nutzung von TLS) relativ klar als Schwachstelle in der Konfiguration erkennbar, da Personen ohne Leseberechtigung unter Umständen die Möglichkeit haben, die Daten beispielsweise über einen *Man-in-the-Middle-Angriff* einzusehen oder zu manipulieren.

Ein weiteres Beispiel ist die Identifikation von Fehlkonfigurationen anhand der Rolle eines Computersystems im Netz. So kann das Vorhandensein von Clientsoftware (z.B. Webbrowser, PDF-Reader oder Mediaplayer) auf Servern dazu führen, dass die Anzahl der Softwareschwachstellen auf derartigen Geräten unnötigerweise erhöht wird. Entsprechend kann die Existenz der Clientanwendungen als fehlerhafte Konfiguration des Systems angesehen werden.

Insofern ist die Erkennung und Behebung von Fehlkonfigurationen vor allem von einer guten Organisation und einer klaren Definition der IT-Dienste und IT-Infrastruktur abhängig, die diese realisieren. Eine allgemeine Orientierungshilfe, auch bei der Einschätzung von Softwareschwachstellen, bietet jedoch wieder die Beachtung der Aufrechterhaltung der drei Schutzziele der Informationssicherheit, die in Abschnitt 2.3.1 ausgeführt werden: *Vertraulichkeit*, *Integrität* sowie *Verfügbarkeit* von Informationen.

2.2 IT-Service-Management und Schwachstellen

In Hochschulen gibt es ähnlich wie in Unternehmen Zielvereinbarungen, deren Erreichung sie anstreben. Beispiele für solche Ziele im Hochschulumfeld sind die „Sicherung der guten wissenschaftlichen Praxis“, „Stärkung des Wissens- und Technologietransfers“ oder auch die „Steigerung der Effizienz in der Hochschulverwaltung“. [Bay15] Die durch die dazugehörigen Rechenzentren zur Verfügung gestellten Dienste und insbesondere das Kommunikationsnetz dienen der Unterstützung der Zielerreichung. Entsprechend zielen die Rechenzentren selbst unter anderem auf eine hohe Verfügbarkeit ihrer angebotenen IT-Dienste (z.B. E-Mail und Webdienste) ab, was durch ein effektives Management unterstützt und erreicht werden kann.

Management bezeichnet die Planung, Erstellung, den Betrieb und die Überprüfung von Aktivitäten mit dem Zweck, die Ziele eines Unternehmens zu erreichen. [Sch14a] Zur Umsetzung des Managements ist es notwendig, dass – entweder implizit oder explizit – ein Managementsystem existiert.

Ein Managementsystem ist eine Menge, zum Zweck einer Zielerreichung, in Wechselbeziehung stehender Elemente. Diese Elemente umfassen insbesondere Richtlinien (Vorgaben durch die Unternehmensleitung), Verantwortlichkeiten, Anleitungen, Verfahren und die zur Umsetzung des Managements benötigten Betriebsmittel wie erforderliches Personal, Hardware und Software. Eine weitere wichtige Komponente innerhalb eines Managementsystems sind Prozesse, das heißt in gegenseitiger Beziehung zueinander stehende Mittel und Aktivitäten, die Eingaben in Ergebnisse umwandeln. Dieser prozessorientierte Ansatz soll dazu führen, dass Abläufe, Verfahren sowie der Einsatz der Betriebsmittel standardisiert und die Verantwortlichkeiten festgelegt werden, wodurch sich eine höhere Zielorientierung, Wirksamkeit der beabsichtigten Aufgabe und Effizienz einstellen. [Sch14b] Der Anwendungsbereich eines Managementsystems kann sich dabei je nach Anforderungen und Absicht auf eine oder mehrere Aufgabenbereiche oder Organisationsgebiete erstrecken [Int14a] und legt allgemein den Rahmen des Managements fest.

Im IT-Service-Management ist die Betrachtung der Rollen ein wichtiger Aspekt der Verwaltung und insbesondere Klärung von Zuständigkeiten für IT-Dienstleistungen. Die wichtigste Unterscheidung besteht in der Trennung der Rollen des *Dienstanbieters*, des *Kunden* und des *Nutzers*.

> Der **Dienstanbieter** stellt einen IT-Dienst zur Verfügung und erfüllt je nach Absprache mit dem Kunden weitere Leistungen, wie die Wartung des Dienstes. Umfang und Dienstgüte (engl. „Quality of Service“) einer Dienstleistung wird üblicherweise in Form eines *Service Level Agreements* (SLA), einer Vereinbarung zwischen Dienstanbieter und Kunden, festgehalten.
>
> Der **Kunde** nimmt einen Dienst in Anspruch, mit der Absicht, die Erreichung seiner eigenen Ziele zu unterstützen und dadurch einen Mehrwert zu generieren.
>
> **Nutzer** bzw. Dienstanwender nutzen IT-Dienste zur Unterstützung oder Ermöglichung der Erfüllung ihrer Aufgaben. [Sch14b] [Sch14c]

In Hinblick auf das Management von Schwachstellen, gibt es einige Kernprozesse des IT-Service-Managements, die dafür als erweiterter Rahmen dienen. Um diese zu bestimmen, muss jedoch klar sein, worauf sich das in dieser Arbeit konzipierte Schwachstellenmanagement bezieht, beziehungsweise, welche Aktivitäten darunter verstanden werden.

> „Schwachstellenmanagement bezeichnet die zyklische Ausübung der **Identifikation**, **Klassifikation**, **Beseitigung und Abschwächung**“ [For09] sowie die **Prävention** von Schwachstellen.

Der Inhalt der jeweiligen Aktivität wird im folgenden Abschnitt anhand von Beispielen erläutert. Aufgrund des Fokus dieser Arbeit auf Netz- und Computersysteme, wird diese

Definition im Kontext des Managements von Softwareschwachstellen und Schwächen in der Konfiguration bezogen (vgl. Abschnitt 2.1). Bei der Umsetzung eines Schwachstellenmanagements (bzw. von Management generell) ist zu beachten, dass die Umsetzung der Aktivitäten im Allgemeinen nicht vorgegeben ist und je nach Umgebung bzw. Anwendungsbereich unterschiedlich realisiert werden kann. So ist es auch möglich, dass Aktivitäten begründet überhaupt nicht umgesetzt werden.

Kernprozesse, die für ein effektives Management von IT-Diensten gemäß der ISO/IEC-20000-Norm betrachtet werden müssen, finden sich auch oft im Hochschulumfeld wieder und werden in der Regel umgesetzt. Dazu gehören das *Change Management*, *Problem Management* und auch das *Information Security Management* sowie zehn weitere. [Sch14d]

> Das Ziel eines Schwachstellenmanagements ist die systematische Reduzierung von Schwachstellen auf den im Anwendungsbereich definierten Computersystemen zur Erreichung einer höheren Netzsicherheit.

Auch wenn Schwachstellenmanagement offenbar Teil des Information Security Managements ist und dem Erhalt der Vertraulichkeit, Integrität und Verfügbarkeit von Daten dient, so steht es auch mit den beiden anderen Kernprozessen in Verbindung. Das Change Management gewährleistet, „dass alle Veränderungen an Infrastruktur und Services in standardisierter Weise bewertet, autorisiert und implementiert werden, um unerwünschte negative Auswirkungen zu verhindern“ [Sch14d]. Die Ziele des Schwachstellenmanagements und dessen strukturierte Ausübung wirken dabei unterstützend. Beispielsweise ist es sinnvoll, sich bei der Aktualisierung einer Software über Schwachstellen in der Folgeversion zu informieren und diese gegebenenfalls zu beheben oder durch die Installation einer alternativen Version zu umgehen, damit keine offensichtlichen Angriffspunkte für Bedrohungsereignisse existieren. In jedem Fall ist es jedoch sinnvoll, ein Bewusstsein über vorhandene Schwachstellen zu schaffen.

Ähnlich unterstützend wirkt das Schwachstellenmanagement auch als Teil des Problem Managements, das „Störungen durch Analyse von möglichen Ursachen für Störungen und proaktive Maßnahmen“ [Sch14d] vermeiden soll. Da Schwachstellen potenziell Ursachen für Störungen sein können und im Rahmen des Schwachstellenmanagements diese proaktiv oder reaktiv behandelt werden, ist der mögliche Beitrag zum Problem Management gegeben.

Allgemein ist Schwachstellenmanagement auch ein entscheidender Teil des im Informationssicherheitsmanagement betrachteten Managements von Risiken, welche in Kombination mit Bedrohungen entstehen. Es ist auch oft der Teil des Risikomanagements, der erheblichen, wenn nicht sogar generell den einzig möglichen Einfluss auf die Behandlung von Risiken hat, da Bedrohungen (z.B. Angreifer oder Unfälle) in der Regel nicht behoben werden können, Schwachstellen hingegen schon.

2.3 Bewährte Maßnahmen im Umgang mit Softwareschwachstellen

In diesem Abschnitt wird der Lebenszyklus einer Softwareschwachstelle – von ihrer Entstehung bis hin zur Behebung und anschließenden präventiven Maßnahmen – erklärt und

die gängigsten Methoden innerhalb der Aktivitäten im Schwachstellenmanagement genannt. Die Aktivitäten umfassen grob die **Identifikation**, **Klassifikation**, **Beseitigung und Abschwächung**, [For09] sowie die **Prävention** von Schwachstellen. Vorab wird jedoch deren Entstehung sowie typische Wege zur Entdeckung und Veröffentlichung erläutert.

2.3.1 Entstehung einer Schwachstelle

Schwachstellen in Software bezeichnen allgemein eine Teilmenge von *Bugs*, also Fehler, in Programmen bzw. in deren Quelltext oder Konfiguration. Ob diese Bugs jeweils Schwachstellen sind, hängt davon ab, ob sie zum einen ausgenutzt werden können und ob ihre Ausnutzung zu einem ungewollten Zustand, also der Verletzung mindestens eines der drei Schutzziele der Informationssicherheit,

- **Vertraulichkeit** – „Schutz von Information vor unberechtigter Offenlegung“ [Sch14e]
- **Integrität** – Schutz von Information vor unberechtigter Modifikation, Einfügungen, Löschungen, Umordnung, Duplizierung oder Wiedereinspielung [Sch14e]
- **Verfügbarkeit** – „Sicherstellung der Zugänglichkeit und Nutzbarkeit von Informationen für berechtigte Entitäten“ [Sch14e]

führen kann. Das Auftreten von Softwarefehlern liegt in der Verantwortung des Entwicklers, der diese beispielsweise aus Unerfahrenheit, Unwissen, Leichtsinn sowie Termindruck und möglicherweise auch mit Absicht in den Quelltext oder der Konfiguration einfügt.

2.3.2 Entdeckung von Schwachstellen

Ein weiterer Schritt im Lebenszyklus einer Schwachstelle ist ihre Entdeckung in einem Softwareprodukt. Nach einem Bericht des *Bundesamtes für Sicherheit in der Informationstechnik* (BSI) [Bun12] hängen der Gefährdungsgrad und die unternommenen Schritte von der Art der Veröffentlichung der Schwachstelle durch den Entdecker ab. Die gängigsten Möglichkeiten der Bekanntmachung sind ein *Full Disclosure*, ein *Responsible Disclosure* oder auch keine Veröffentlichung (*Non Disclosure*) einer Schwachstelle.

Bei einem **Full Disclosure** wird die Schwachstelle mit Informationen zur Ausnutzung, falls diese vorhanden sind, komplett öffentlich gemacht. Ab dem Zeitpunkt der Veröffentlichung besteht infolgedessen ein hohes Risiko der Ausnutzung, da Betroffene in der Regel aus Mangel an notwendigem Fachwissen, aber besonders aus Mangel einer Lösung durch den Hersteller keine Möglichkeit der Behebung haben, Angreifer jedoch mit diesen Informationen gezielt einen *Exploit* (die Möglichkeit der Ausnutzung einer Schwachstelle) entwickeln können. Sobald ein Exploit für eine Schwachstelle existiert, steigt das Risiko der Ausnutzung deutlich und kann mit einer Verbreitung des Exploits weiter zunehmen. Der Hersteller gibt nach Bekanntwerden der Schwachstelle in der Regel ein *Advisory* heraus – eine Meldung, die von der Schwachstelle betroffene Systeme und Softwareprodukte auflistet und unter Umständen bereits eine Übergangslösung (einen *Temporary Fix*) beinhaltet, welche die Ausnutzung der Schwachstelle erschweren oder verhindern soll, bis die Schwachstelle durch einen *Patch*, einer Korrektur des Fehlers in der Software, geschlossen werden kann. Der Vorteil, der sich aus der kompletten Veröffentlichung ergibt, ist, dass Softwarehersteller aufgrund der öffentlichen Aufmerksamkeit tendentiell schneller einen Patch entwickeln, um

den Ruf des Unternehmens zu wahren. Der gravierendste Nachteil eines Full Disclosure ist, dass die Schwachstelle durch ihr Bekanntwerden gezielt ausgenutzt werden kann, noch bevor geeignete Möglichkeiten zur Behebung existieren.

Daher ist der gängigere Ansatz, ein **Responsible Disclosure** (auch *Coordinated Disclosure* genannt), die Meldung der Schwachstelle mit allen Informationen zunächst ausschließlich an den Hersteller der betroffenen Software und einer optionalen Meldung der Existenz der Schwachstelle ohne weitere Informationen an die Öffentlichkeit. Da der Hersteller durch eine ausschließliche Meldung einer Schwachstelle an ihn keinen konkreten Anlass zur Handlung sehen könnte, wird dem Hersteller durch den Entdecker häufig eine bestimmte Frist vorgegeben, in der er Zeit hat, einen Patch zu entwickeln und zu veröffentlichen, bis ein Full Disclosure der Schwachstelle vorgenommen wird. Dadurch wird er mit Hilfe des öffentlichen Drucks zum Handeln veranlasst und das Risiko einer Ausnutzung der Schwachstelle gleichzeitig relativ niedrig gehalten. [Bru07]
Letztendlich kommt es jedoch nicht nur auf den Hersteller an, ob eine Sicherheitslücke durch einen von ihm herausgegebenen Patch geschlossen wird, sondern vor allem auch auf betroffene Nutzer, die für die Installation des Patches verantwortlich sind.

Die letzte Möglichkeit besteht darin, dass der Entdecker der Schwachstelle weder die Öffentlichkeit noch den Hersteller des betroffenen Produkts informiert. In diesem Fall handelt es sich um einen sogenannten **Non Disclosure**, der nicht selten dazu führt, dass die Schwachstelle erst durch ihre Ausnutzung bekannt wird. Eine derartige Ausnutzung wird als *Zero-Day-Exploit* bezeichnet, wobei es oft unklar ist, wie lange eine solche Schwachstelle bereits bekannt ist und aktiv ausgenutzt wird. Daher birgt diese Art der Schwachstellenentdeckung das größte Gefahrenpotential für Softwarenutzer.

Im Falle der Schwachstelle, die als „Heartbleed Bug“ bekannt wurde, ist der Weg der Entdeckung und Veröffentlichung gut dokumentiert. Die Entdeckung fand in erster Instanz am 21. März 2014 durch Sicherheitsforscher der *Google Inc.* statt, woraufhin unternehmensintern ein Patch entwickelt wurde. Bis zur Meldung an den Entwickler, die *OpenSSL Software Foundation*, am 1. April wurde die Kenntnis über den Programmfehler an verschiedene Unternehmen und Personen verbreitet, jedoch nicht vollständig der Öffentlichkeit preisgegeben. Die Weitergabe an Informationen über den Fehler hat sich in den folgenden Tagen verstärkt und wurde beispielsweise in einigen privaten Mailinglisten verbreitet sowie an verschiedene Hersteller von Unix-Distributionen weitergegeben. Am 7. April wurde durch das OpenSSL-Team schließlich ein *Bug-Fix* (eine von dem Fehler behobene Version) sowie ein Advisory veröffentlicht. [Gru14] Die Vorgehensweise lässt sich insofern durch die Weitergabe der Informationen an teilweise unbekannte Personen und Organisationen nicht exakt einer der gängigen Veröffentlichungswege zuordnen, liegt durch den Ausschluss der Öffentlichkeit bis zur Existenz einer Möglichkeit zur Behebung jedoch näher an einem *Responsible Disclosure*.

Zu beachten ist, dass die genannten Varianten der Schwachstellenveröffentlichung nur grobe Möglichkeiten der Einteilung sind und es, wie am Beispiel des Heartbleed Bugs, keine klare Abgrenzung gibt. Am Ende hängt die Vorgehensweise der Veröffentlichung alleine vom Schwachstellenentdecker bzw. den Personen ab, die im Laufe der Veröffentlichung über die Schwachstelle informiert werden.

2.3.3 Identifikation von Schwachstellen

Die Identifikation von Schwachstellen ist einer der grundlegendsten und bedeutendsten Schritte im Schwachstellenmanagement. Der Zweck ist die Dokumentation der Schwachstelle in einer Art und Weise, dass insbesondere eine sinnvolle Kommunikation von Schwachstelleninformation ermöglicht wird und diese zur Weiternutzung ausgetauscht werden können. Dazu gehört in der Regel ein eindeutiger Bezeichner der jeweiligen Schwachstelle sowie weitere Informationen, um die Eigenschaften der Schwachstelle zu beschreiben, insbesondere betroffene Produkte, Möglichkeiten der Ausnutzung und das Schadenspotential.

Sobald Informationen über eine Schwachstelle in einem Softwareprodukt, entweder durch den Hersteller oder andere Personen, an die Öffentlichkeit gelangen, werden Informationen über sie, abhängig von betroffenen Softwareprodukten und Art der Schwachstellen, in öffentlichen oder nicht-öffentlichen (z.B. herstellerinterne) Listen dokumentiert. Eine der umfangreichsten dieser Schwachstellen-Listen ist die frei nutzbare *Common Vulnerabilities and Exposures* (CVE), die für eine hohe Anzahl an Hersteller und Produkten eine Sammlung an Schwachstelleninformation verwaltet. Unter den Produkten finden sich insbesondere Betriebssysteme, Serveranwendungen und Tools; eine Auflistung ist auf der offiziellen Website [MIT14d] hinterlegt.

Im Folgenden wird die Identifikation und Dokumentation von Schwachstellen sowie dazu notwendigen Informationen aufgrund der standardisierten Art und Weise und dem allgemein hohen Verwendungsgrad sowie Akzeptanz, am Beispiel der CVE durchlaufen.

Einen großen Vorteil, den standardisierte Schwachstellenlisten bieten, ist ein klares Datenschema und vor allem die Möglichkeit der eindeutigen Bestimmung einer Schwachstelle durch ein definiertes Namensschema. Im Falle der CVE besteht ein Eintrag aus einer eindeutigen Identifikationsnummer – der *CVE-ID*, dem Status des Eintrags, einer Beschreibung der Schwachstelle und Referenzen auf Informationen (z.B. Bestätigungen durch den Hersteller, Advisories und Patches) mit Relevanz für die Schwachstelle.

Die Vergabe von CVE-IDs geschieht durch sogenannte *CVE Numbering Authorities* (CNAs). Dazu gehört die *MITRE Coproration*, welche die Verwaltung der CVE übernimmt, diverse *Computer Emergency Response Teams* (CERTs) und eine Reihe von Softwareherstellern, die in der Regel nur CVE-IDs für jeweils eigene Produkte ausstellen. [MIT15c] Um eine CVE-ID als Nicht-CNA zu erhalten, muss der jeweilige Softwarehersteller oder ein CERT kontaktiert werden. Eine Alternative zur Erreichung einer Eintragung in die CVE ist die Veröffentlichung der Informationen über eine Schwachstelle in öffentlichen bekannten Quellen, wie z.B. der Schwachstellen-Mailingliste *Bugtraq*. [MIT15f]

Der Status kann die Werte „candidate“ und „entry“ annehmen und ist insofern notwendig, da Schwachstellenmeldungen in der Regel in die Liste eingetragen werden und anschließend evaluiert werden muss, ob es sich dabei tatsächlich um eine Schwachstelle handelt. Solange die Schwachstelle auf ihre tatsächliche Aufnahme in die CVE geprüft wird, hält der Eintrag den Status „candidate“; nach einer Aufnahme entsprechend „entry“. Die Überprüfung wird vom *CVE Editorial Board* vorgenommen, das sich aus einer Reihe von Organisationen im Bereich IT-Sicherheit zusammensetzt, darunter Unternehmen wie *Cisco Systems, Inc.*, wissenschaftliche Einrichtungen oder auch staatliche Einrichtungen wie das *National Institute of Standards and Technology* (NIST). [MIT15a]

Eine CVE-ID hat folgende Form:

$$CVE - YYYY - NNNN[N]^{\star} \tag{2.1}$$

Der Identifikator beginnt immer mit „CVE“, gefolgt von der ausgeschriebenen Jahreszahl (z.B. „2015“) der Registrierung des CVE Eintrages (in der Regel ungleich dem Zeitpunkt der Entdeckung der Schwachstelle) und einer mindestens vierstelligen beliebigen natürlichen Zahl. Die variable Stelligkeit der Zahl, die das Ende einer CVE-ID bildet, hat den Grund, dass die Anzahl der dokumentierbaren Schwachstellen pro Jahr auf diese Weise nicht begrenzt ist, typischerweise geht die natürliche Zahl jedoch nicht über sechs Stellen hinaus. CVE-Einträge werden in aufsteigender Reihenfolge registriert, jedoch nicht unbedingt in der gleichen Reihenfolge mit Informationen über Schwachstellen versehen, da die Einträge vorab an viele Softwarehersteller und Organisationen vergeben werden. Zu beachten ist außerdem, dass zum einen jede CVE-ID eindeutig einer Schwachstelle zugewiesen ist und darüber hinaus nicht zwei Einträge dieselbe Schwachstelle beschreiben dürfen. [MIT15b] [MIT15e]

Die CVE gilt als eine der umfangreichsten Quellen für Datenbanken von Softwareschwachstellen. Beispielsweise wird die *National Vulnerability Database* (NVD) [Nat15a], eine vom *National Institute of Standards and Technology* (NIST) betriebene und öffentlich angebotene Softwareschwachstellen-Datenbank, mit Schwachstelleninformation aus der CVE gespeist. Darüber hinaus liefert die NVD weitere Informationen wie die in folgendem Abschnitt erläuterten „Base Metrics“ der jeweiligen Schwachstelle im Rahmen des *Common Vulnerability Scoring System* Version 2 (CVSSv2) [For07] und Referenzen zu darauf bezogenen Advisories sowie Lösungen. [Nat15c] Des Weiteren stellt das NIST eine Suchfunktion für Schwachstellen in seiner Datenbank sowie Referenzen zu relevanten Einträgen in der *Common Weakness Enumeration* (CWE) und der *Common Platform Enumeration* (CPE) zur Verfügung. Die CPE dient der Identifikation insbesondere von Softwareprodukten und bietet unter anderem ein zu diesem Zweck maschinenlesbares Format [MIT13], das in der NVD genutzt wird, um von der jeweiligen Schwachstelle betroffene Softwareanwendungen zu beschreiben.

Neben der NVD gibt es noch weitere öffentlich zugängliche Quellen, die Informationen zu Schwachstellen anbieten. Neben weiteren Datenbanken wie der *Open Source Vulnerability Database* (OSVDB) – einer Datenbank, die Schwachstelleninformation aus öffentlichen Quellen bezieht – stellen in der Regel Computer Emergency Response Teams, wie das DFN-CERT (siehe Glossar 9), das CERT-BUND (dem CERT des Bundesamtes für Sicherheit in der Informationstechnik) oder auch das US-CERT – einem CERT des US Department of Homeland Security – derartige Informationen ihren Nutzern zur Verfügung. Oft werden Schwachstellen auch durch den Hersteller selbst gemeldet, wie beispielsweise durch die *Microsoft Security Bulletins* [Mic15]. Dabei werden üblicherweise jeweils hersteller- bzw. listenspezifische Bezeichner und Beschreibungen verwendet, jedoch in der Regel – falls vorhanden – auch auf den entsprechenden Eintrag in der CVE verwiesen.

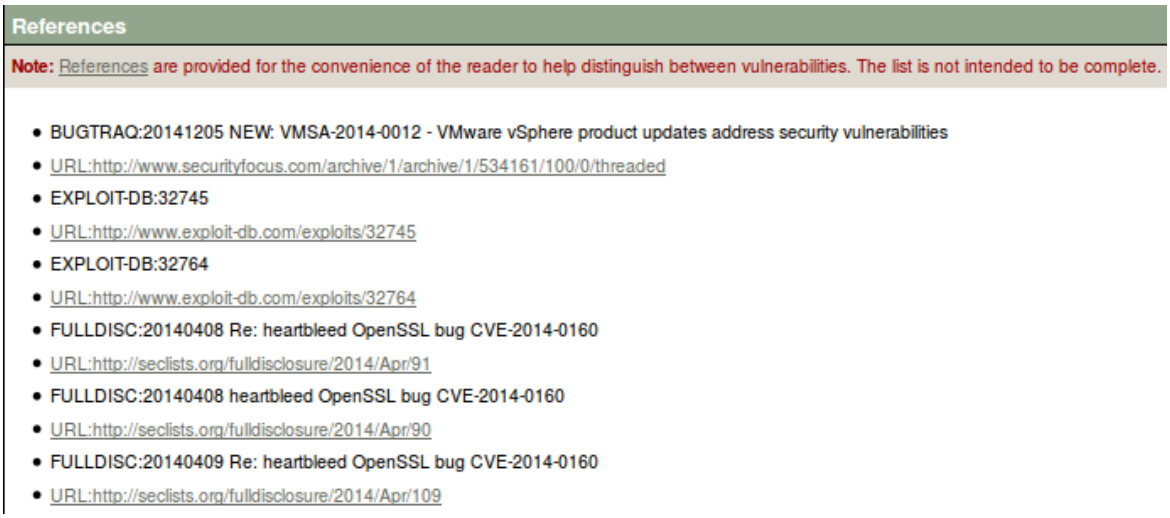
Bezeichner wie „Heartbleed Bug“ entstehen in seltenen Fällen inoffiziell durch den Gebrauch, beispielsweise bei Diskussionen in Foren. Im Falle des Heartbleed Bugs wurden bei der Dokumentation in der CVE zwei offizielle Einträge für dieselbe Schwachstelle genutzt: Zum einen ein Eintrag mit der CVE-ID „CVE-2014-0160“ und zum anderen der Eintrag mit

der ID „CVE-2014-0346“. [Cod14] Duplikate in der CVE entstehen aufgrund einer Entdeckung und Meldung derselben Schwachstelle durch unabhängige Personen (in diesem Fall zunächst durch Mitarbeiter der Unternehmen *Google Inc.* und wenig später unabhängig davon durch Mitarbeiter des Unternehmens *Codenomicon*). Bei der Aufdeckung von Duplikaten werden diese, abhängig von ihrer Verbreitung bzw. allgemeinen Benutzung, der Quelle, dem Zeitpunkt des Eintrages oder der Nummer in der CVE-ID [MIT11], in ihrem Beschreibungstext als „REJECT“ gekennzeichnet und sollten nicht weiter benutzt werden. So auch der Eintrag mit der CVE-ID „CVE-2014-0346“.

Die Beschreibung der Schwachstelle im offiziell zugewiesenen Eintrag mit der CVE-ID „CVE-2014-0160“ ist wie folgt:

> „The (1) TLS and (2) DTLS implementations in OpenSSL 1.0.1 before 1.0.1g do not properly handle Heartbeat Extension packets, which allows remote attackers to obtain sensitive information from process memory via crafted packets that trigger a buffer over-read, as demonstrated by reading private keys, related to d1_both.c and t1_lib.c, aka the Heartbleed bug.“ [MIT14b]

In diesem Fall wird aus der Beschreibung ersichtlich, welche Versionen der Software betroffen sind, nämlich alle Implementationen der Version 1.0.1 bis 1.0.1g, sowie die Angriffsmöglichkeit, die der Bug bietet: Ein Auslesen des Speichers, ausgelöst durch Netzwerkpakete durch einen Angreifer. In den für den Bug aufgelisteten Referenzen finden sich beispielsweise Beschreibungen, Bestätigungen und Patches verschiedener Quellen. Ein Beispiel


References

Note: References are provided for the convenience of the reader to help distinguish between vulnerabilities. The list is not intended to be complete.

- BUGTRAQ:20141205 NEW: VMSA-2014-0012 - VMware vSphere product updates address security vulnerabilities
- URL:http://www.securityfocus.com/archive/1/archive/1/534161/100/0/threaded
- EXPLOIT-DB:32745
- URL:http://www.exploit-db.com/exploits/32745
- EXPLOIT-DB:32764
- URL:http://www.exploit-db.com/exploits/32764
- FULLDISC:20140408 Re: heartbleed OpenSSL bug CVE-2014-0160
- URL:http://seclists.org/fulldisclosure/2014/Apr/91
- FULLDISC:20140408 heartbleed OpenSSL bug CVE-2014-0160
- URL:http://seclists.org/fulldisclosure/2014/Apr/90
- FULLDISC:20140409 Re: heartbleed OpenSSL bug CVE-2014-0160
- URL:http://seclists.org/fulldisclosure/2014/Apr/109

Abbildung 2.1: Teilliste der Referenzen für CVE-2014-0160 (Heartbleed Bug) [MIT14b]

für die Auflistung von Referenzen für den Heartbleed Bug ist in Abbildung 2.1 zu sehen. Die Form der Referenzen ist genauso standardisiert wie die CVE-Einträge selbst und bestehen aus der Referenzquelle (im Beispiel: „*BUGTRAQ*“ [Sec15], „*EXPLOIT-DB*“ [Exp15] und „*FULLDISC*“ [Gor15a]) und einem eindeutigen Identifikator innerhalb der Quelle. [MIT15d]

2.3.4 Klassifikation von Schwachstellen

Die Klassifikation von Schwachstellen ist ein weiterer Hauptaspekt des Schwachstellenmanagements und liefert einen erheblichen Beitrag zum Erreichen insbesondere von Effizienz bei der Beseitigung von Schwachstellen. Denn zum einen ist nicht jede Schwachstelle relevant für eine bestimmte IT-Umgebung und muss infolgedessen nicht behandelt werden, und zum anderen können durch eine aus der Bewertung abgeleiteten Schwachstellenpriorisierung organisatorische Abläufe optimiert werden.

Die Klassifikation von Schwachstellen baut auf der Identifikation von Schwachstellen auf und ist insbesondere von den darin dokumentierten Informationen abhängig. Die Bewertung kann aufgrund verschiedener und teilweise auch umgebungs- und zeitabhängiger Kriterien basieren. Einen guten, jedoch nicht unbedingt in jeder Umgebung und für jeden Zweck bedeutsamen oder vollständigen Überblick über diese bewertungsrelevanten Kriterien bietet beispielsweise das standardisierte *Common Vulnerability Scoring System* (CVSS). Im Verlauf des Abschnitts soll deshalb anhand des CVSS ein Eindruck vermittelt werden, welche Aspekte einen Einfluss auf die Kritikalität von Schwachstellen haben. Die „Kritikalität“ einer Schwachstelle bezeichnet das Gefahrenpotenzial das von ihr ausgeht. Dieses setzt sich allgemein aus Faktoren der Ausnutzbarkeit sowie der Auswirkungen einer Ausnutzung zusammen. Je einfacher die Ausnutzung einer Schwachstelle umgesetzt werden kann und je höher das von ihr ausgehende Schadenspotenzial ist, desto höher ist die Kritikalität einer Schwachstelle.

Das in der NVD eingesetzte CVSS Version 2 zur Bewertung von Schwachstellen ist eines der am verbreitetsten, akzeptiertesten und wohl-dokumentiertesten Bewertungssysteme sowie zur freien Nutzung herausgegeben. Das *Forum of Incident Response and Security Teams* (FIRST), der Herausgeber des CVSS, hat bereits eine Version 3 veröffentlicht; dieses findet jedoch zum aktuellen Stand (November 2015) noch kaum Anwendung, wird aber im Laufe des Abschnitts ebenfalls erläutert.

Der CVSS Version 2 berücksichtigt bei der Bewertung einer Schwachstelle folgende drei Aspekte (vgl. Abbildung 2.2):

Die Base Metrics
Die „Base Metrics“ drücken zeit- und umgebungsunabhängige Charakteristika einer Schwachstelle aus. Darin betrachtet wird die Ausnutzbarkeit der Schwachstelle, berechnet durch die Art des Zugangs (lokal oder über das Netz), die Komplexität bzw. Möglichkeit der Ausnutzung (z.B. in Form von notwendigem Fachwissen, der Anzahl betroffener Systeme oder auch erforderlichen Nutzerrechten) sowie der Notwendigkeit einer Authentifizierung als Vorbedingung der Ausnutzung.

Des Weiteren fließen die Auswirkungen einer erfolgreichen Ausnutzung auf die drei Hauptziele der Informationssicherheit, der Vertraulichkeit, Integrität und Verfügbarkeit in diese Bewertung mit ein.

Die Temporal Metrics
Der zweite Aspekt betrifft zeitliche Veränderungen in Bezug auf die Schwachstelle, die „Temporal Metrics“, deren Wert sich im Verlauf der Zeit ändern kann. Ein wichtiger Gesichtspunkt dieses Wertes ist die Wahrscheinlichkeit einer Ausnutzung – welche

abhängig von der Verfügbarkeit eines Exploits ist: Die Schwachstelle ist beispielsweise weniger kritisch, wenn sie nicht durch einen Exploit ausnutzbar ist oder es höchstens einen theoretischen Ansatz der Ausnutzung gibt. Andernfalls ist eine Schwachstelle als deutlich kritischer zu bewerten, falls die Ausnutzung automatisiert durchgeführt oder gar ohne Exploit, sondern manuell ausgenutzt werden kann.

Des Weiteren werden die „Temporal Metrics“ durch die Behebbarkeit der Schwachstelle beeinflusst. Diese kann nicht bekannt oder nicht verfügbar sein sowie in Form eines *Workarounds* – einer inoffiziellen Lösung, die nicht durch den Hersteller herausgegeben, sondern häufig in Foren oder Communities durch Nutzer und Interessierte entwickelt wird – eines *Temporary Fix* (eine offizielle aber nicht-dauerhafte Möglichkeit der Behebung) oder eines *Offical Fix* (einer endgültigen Lösung, z.B. einem Patch) vorliegen.

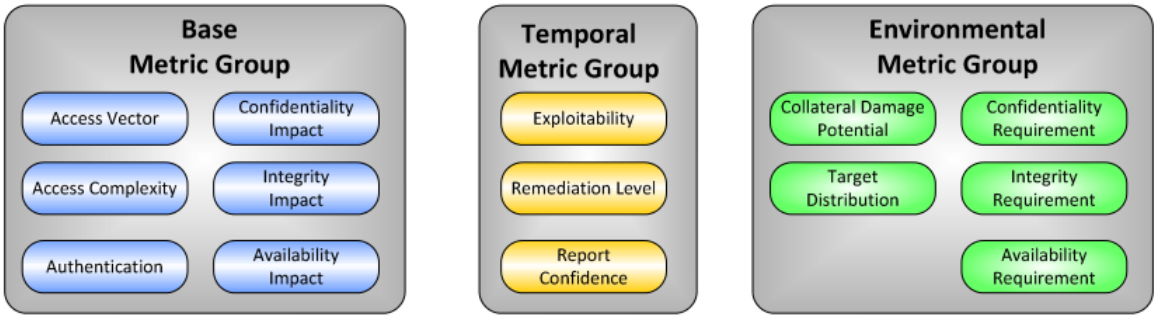


Abbildung 2.2: CVSS Version 2 Metriken und Einflussgrößen [For07]

Der letzte zeitliche Faktor ist die Glaubwürdigkeit der Existenz einer Schwachstelle (*Report Confidence*). Beispielsweise geht von einer Schwachstelle weniger Ausnutzungspotential aus, wenn die Informationen über sie aus einer unbestätigten Quelle stammen, als wenn ihre Existenz mehrfach bewiesen oder durch den Hersteller des betroffenen Produktes sogar offiziell bestätigt wurde.

Die Environmental Metrics
Der dritte Wert des CVSSv2 betrifft die Relevanz innerhalb der IT-Umgebung, in der die Schwachstelle auftritt, den „Environmental Metrics“. Darin betrachtete Größen sind das Schadenspotential – zum einen für Assets aber auch im Sinne wirtschaftlicher Aspekte, der Anteil der von der Schwachstelle betroffenen Systeme und der jeweilige Einfluss der Verletzung von Vertraulichkeit, Integrität und Verfügbarkeit auf die Organisation oder Einzelpersonen innerhalb der Organisation. Ein weiterer bedeutsamer Faktor ist die Verfügbarkeit möglicher Ziele innerhalb der IT-Umgebung. Der Wert der „Environmental Metrics“ fällt entsprechend höher aus, wenn die Anzahl möglicher Zielsysteme höher ist. Vor allem dieser Wert kann für Schwachstellen im Hochschulnetz entscheidend sein, da so die spezielle Umgebung mit in die Bewertung einfließen kann.

Die Werte der „Base Metrics“, „Temporal Metrics“ und „Environmental Metrics“ stehen für sich, liegen jeweils in einem Intervall zwischen 0.0 und 10.0 und werden entsprechend als *BaseScore*, *TemporalScore* und *EnvironmentalScore* bezeichnet. Zu beachten ist jedoch, dass der BaseScore eine obere Schranke des TemporalScores bildet und der TemporalScore

den EnvironmentalScore nach oben beschränkt. [For07] Anhand dieser Werte wird allgemein die Kritikalität und die Wahrscheinlichkeit der Ausnutzung einer Schwachstelle abgeleitet, wobei erst die Betrachtung der einzelnen Metriken ein genaues Bild über das Verhältnis der beiden Faktoren liefert.

Metrik	**Wert**
Access Vector	Network
Access Complexity	Low
Authentication	None
Confidentiality Impact	Partial
Integrity Impact	None
Availability Impact	None

Tabelle 2.1: Werte der CVSSv2 Base Metrics des Heartbleed Bugs [Cod14]

Auch wenn der Heartbleed Bug allgemein aufgrund seiner hohen Verbreitung und der möglichen Offenlegung von Daten mit besonderem Schutzbedarf (z.B. Passwörter) als äußerst schwerwiegend eingestuft wird, erreicht er anhand seiner betrachteten Charakteristika (siehe Tabelle 2.1) nur einen moderaten BaseScore von 5.0/10.0. Denn obwohl die Ausnutzbarkeit (berechnet aus *Access Vector*, *Access Complexity* und *Authentication*) den maximalen Wert von 10.0 erreicht, so bietet die Schwachstelle lediglich die Möglichkeit einer nichtvollständigen Verletzung der Vertraulichkeit, das heißt, dass einige mehr oder weniger durch Zufall bestimmte, im Arbeitsspeicher befindliche Daten, jedoch nicht beliebige Daten des betroffenen Systems ausgelesen werden können. Des Weiteren hat der Angreifer keine Kontrolle über die spezifischen Daten, die offengelegt werden. Die Auswirkung einer Ausnutzung wird unabhängig von den Daten daher lediglich mit 2.9/10.0 bewertet. Der eher niedrige *BaseScore*-Wert berechnet sich durch die in [For07] ausführlich erklärte Formel.

> Ein Problem der Bewertung durch das CVSSv2 zeigt sich demnach bereits hier: Die Schwachstellen werden unabhängig von der Art der durch sie betroffenen Daten eingestuft. Dieser Aspekt ist ebenso wenig in den Temporal noch in den Environmental Metrics explizit vorgesehen. Folglich ist der CVSS zur Bewertung unvollständig, wodurch das CVSS nicht ausschließlich als Kriterium zur Festlegung der Kritikalität einer Schwachstelle verwendet werden sollte.

FIRST, der Herausgeber des CVSS, hat Mitte des Jahres 2015 eine neue Version des Standards herausgegeben, den CVSS Version 3 (CVSSv3) [For15b]. Dieser zielt darauf ab, die Anwendbarkeit des Standards weiter zu verbessern. Das Schema der dort relevanten Metriken ist in Abbildung 2.3 dargestellt.

In den „Base Metrics“ wird bei der Beschreibung der Ausnutzbarkeit im nun genannten *Attack Vector* (ehemals *Access Vector*) neben der lokalen Ausnutzung bzw. der Ausnutzung über das Netz auch die Möglichkeit in Betracht gezogen, dass der Angreifer physischen Zugriff auf das Gerät haben kann. Die *Access Complexity* wurde in *Attack Complexity* umbenannt und hat anstelle von drei möglichen Bewertungen nur noch zwei:

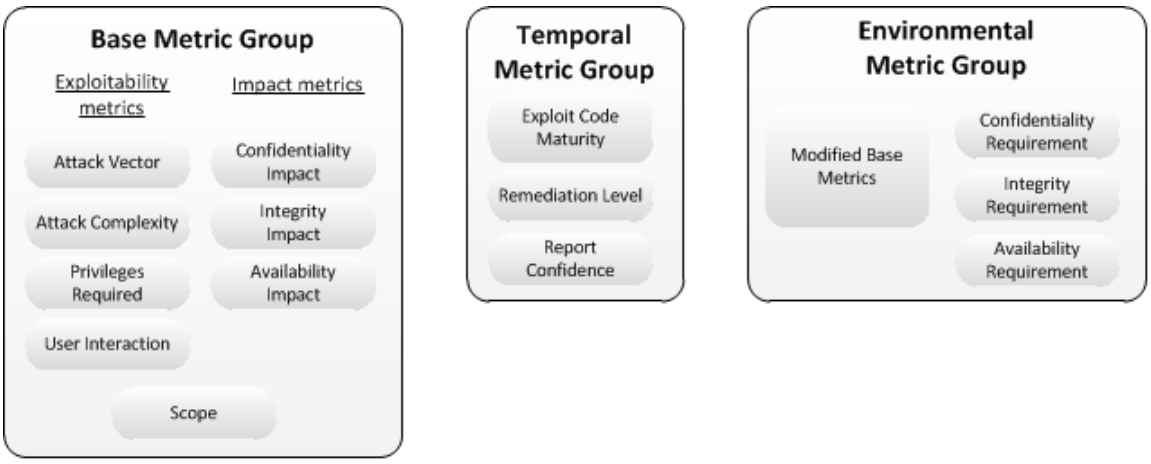


Abbildung 2.3: Metriken des CVSS Version 3 [For15b]

„Low“ – Es existieren keine weiteren Behinderungen an der Ausnutzung

„High“ – Die Schwachstelle ist nicht ohne weiteres ausnutzbar, sondern ein Angreifer muss zunächst Aufwand betreiben und beispielsweise weitere Informationen über das Zielsystem erlangen, die Umgebung des Zielsystems anpassen oder auf dem gleichen logischen Netzpfad liegen.

Anstelle der Metrik bezüglich der Notwendigkeit einer Authentifizierung, wird in Version 3 genereller die Notwendigkeit des Besitzes von weiteren Privilegien für das Zielsystem betrachtet: Dabei unterschieden wird, ob ein möglicher Angriff mit oder ohne Authentifizierung erfolgen kann, ob dessen Erfolg vom Besitz geringfügiger Rechte (in der Regel Nutzerrechten mit Lese- und Schreibberechtigungen, lediglich für Dateien im Besitz des Nutzers) oder ob für einen erfolgreichen Angriff weitgreifendere Rechte auf dem betroffenen System (beispielsweise Administratorrechte) notwendig sind. Die beiden letzteren Fälle sind entsprechend auch von einer erfolgreichen Authentifizierung abhängig.

Ein neuer Aspekt, der in die Bewertung mit einfließt, ist die mögliche Unverzichtbarkeit einer Handlung durch einen Nutzer ($\neq$ Angreifer), wodurch ein Angriff erschwert werden könnte. Außerdem ist die *Scope*-Metrik neu, welche sich auf den Autorisierungskontext bezieht und beschreibt, ob eine Schwachstelle in einem Produkt, das in einem größeren Rahmen (z.B. einem Computersystem) bestimmte Rechte zugewiesen bekommen hat, Einfluss auf Ressourcen in anderen Autorisierungskontexten (z.B. einem weiteren Computersystem) hat. Ein weiteres Beispiel wäre auch ungewollte bzw. unautorisierte Lese- und Schreibzugriffe eines Java-Applets auf Daten außerhalb der Java-Sandbox im Webbrowser. Die Metriken der Auswirkung auf Vertraulichkeit, Integrität sowie Verfügbarkeit sind genau wie der größte Teil der „Temporal Metrics“ gleich geblieben. Es wurde lediglich die CVSSv2 Metrik *Exploitability* in *Exploit Code Maturity* umbenannt, sowie die Werte der *Report Confidence* angepasst. Diese haben sich weg von der Glaubwürdigkeit und Anzahl der Quellen, hin zu Detailgrad der Beschreibung und Reproduzierbarkeit des die Schwachstelle betreffenden Verhaltens verändert.

Bei der Bewertung der Schwachstelle in Bezug auf die Umgebung, in der sie sich befindet,

wurde allgemein das Schadenspotenzial sowie die Anzahl betroffener Systeme aus der Bewertung ausgeschlossen. Stattdessen können im einzelnen alle Faktoren der „Base Metrics“ jeweils für die betrachtete IT-Umgebung angepasst werden.

Genau wie in der Version 2 des CVSS stehen die Werte der einzelnen Gruppen für sich und lediglich die Angabe der „Base Metrics“ ist unverzichtbare für eine Bewertung. Der TemporalScore bleibt weiterhin nach oben durch den Wert des BaseScore begrenzt, hingegen bleibt der EnvironmentalScore wie in der Version 2 abhängig vom TemporalScore, kann aber nun über diesen hinausgehen.

Aufgrund der weiteren Faktoren, die im CVSSv3 betrachtet werden, hat sich die Formel zur Berechnung der Werte geändert, wodurch Schwachstellen nicht dieselben Werte für CVSSv2-Scores und CVSSv3-Scores einnehmen müssen, sondern sich potenziell unterscheiden: Im Falle des Heartbleed Bugs errechnet sich so ein CVSSv3 BaseScore von 7.5/10.0, welcher die tatsächliche Kritikalität der Schwachstelle besser beschreibt als die Bewertung anhand der Version 2 des CVSS. Bei der Berechnung des BaseScores sind mit Ausnahme der Auswirkungen auf die Vertraulichkeit, welche im CVSS Version 3 als *high* angegeben ist, alle bisherigen Base Metrics gleich geblieben. Für die Metrik des Scopes wurde der Wert *unchanged* verwendet. [For15a]

Ein Vorteil, den das CVSS mit sich bringt, ist eine hohe Transparenz und Nachvollziehbarkeit der Bewertung. Zu diesem Zweck werden die einzelnen Metriken und ihre jeweiligen Werte, die zur Berechnung verwendet werden, als „Vektorstring“ dargestellt, um die Metriken beispielsweise in Anwendungen weiterverarbeiten zu können. Eine Darstellung der Base Metrics des Heartbleed Bugs aus Tabelle 2.1 in CVSS Version 2 als Vektorstring sieht wie folgt aus:

$$AV:N/AC:L/Au:N/C:P/I:N/A:N \quad \text{[Nat15d]} \tag{2.2}$$

Auch wenn das Grundprinzip der Darstellung in der Version 3 des CVSS gleich geblieben ist, so müssen die Abkürzungen dennoch den vorgenommenen Änderungen angepasst werden. Außerdem ist in der Version 3 ein „CVSS:<VERSION>/“ Präfix neu hinzugekommen. Der Abschnitt <VERSION> bezeichnet dabei die Versionsnummer des Standards. Insofern hat die Schreibweise der Base Metrics am Beispiel des Heartbleed Bugs folgende Form:

$$CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N \quad \text{[For15c]} \tag{2.3}$$

Analog kann der String auch in beiden Versionen um die Werte der „Temporal Metrics“ sowie der „Environmental Metrics“ erweitert werden.

Das CVSS ist zwar eines der verbreitetsten, jedoch nicht das einzige Bewertungssystem für Schwachstellen. Im Gegensatz zu vielen anderen zeichnet es sich jedoch durch eine sehr hohe Transparenz aus.

2.3.5 Ausnutzung, Detektion, Beseitigung und Abschwächung

Es gibt zahllose Angriffsmöglichkeiten auf Schwachstellen, die abhängig von der Art und den Charakteristika der jeweiligen Schwachstelle zum Erfolg führen können. Laut Definition einer Schwachstelle im Rahmen der CVE [MIT15e] würde ein erfolgreicher Angriff über eine Softwareschwachstelle darin resultieren, dass dem Angreifer Zugang zu einem System oder einem Netzwerk gewährt wird. Die Ziele, die der Angreifer damit verfolgt, können genauso

zahlreich sein und reichen von einer gutmütigen Absicht zur Verbesserung der allgemeinen Sicherheit eines Systems bis hin zur unrechtmäßigen Bereicherung. Weiterhin bekannte Arten von Fehlern in Software, die zu einer Schwachstelle führen können, sind beispielsweise Pufferüberläufe (engl. „Buffer Overflows“), Cross-Site-Scripting oder SQL-Injections (weitere mögliche Ursachen sind in der CWE gelistet).

Die größte Gefahr besteht aufgrund der weltweiten Erreichbarkeit für Systeme, die direkt an das Internet angeschlossen und deren Dienste aus dem Internet nutzbar sind. Prinzipiell kann – insofern nicht weitere Maßnahmen zur Sicherung angewendet werden – jeder Rechner, der auch mit dem Internet verbunden ist, auf diese Weise nach Schwachstellen auf einem „Opfersystem“ suchen und diese unter Umständen ausnutzen. Dabei finden sich solche Schwachstellen nicht nur in den unmittelbaren Anwendungen, die als Basis für IT-Dienste dienen (z.B.: Webserver und Datenbanksysteme), sondern auch zum großen Teil in Betriebssystemen. Um Clients und Server diesen Gefahren nicht auszusetzen, haben sich weitere Maßnahmen zu deren Schutz etabliert. Dazu gehören beispielsweise die Nutzung von privaten, nicht über das Internet erreichbaren Adressbereichen, die Zugriffskontrolle durch Firewalls und Gateways sowie die Einführung von Authentifizierungs- und Autorisierungsmechanismen. Dabei ist zu beachten, dass vor allem technische Komponenten wie Firewalls selbst nicht frei von Schwachstellen sind und genauso kompromittiert werden können.

Ein wesentlicher Bestandteil der Beseitigung und Abschwächung von Schwachstellen ist deren Detektion auf den jeweiligen zu schützenden Computersystemen. Diese muss in irgendeiner Art und Weise der Beseitigung vorhergehen, da unentdeckte Schwachstellen in der Regel nicht behoben werden können. Allgemein erfolgt die Detektion durch den Abgleich mit Hilfe der im Voraus dokumentierten Schwachstelleninformationen.

Zur Detektion von Schwachstellen gibt es mehrere Möglichkeiten: Zum einen kann mit sogenannten „Penetration Tests“ überprüft geprüft werden, ob es ausnutzbare Schwachstellen in einem System gibt, zum anderen kann man mit dem notwendigen Fachwissen direkt die Quelle derartiger Schwachstellen durchsuchen: Den Quelltext sowie die Konfiguration von Software. Ein Penetration Test ist der direkte Versuch einer Ausnutzung von Schwachstellen. Dies kann entweder manuell geschehen – beispielsweise durch das Testen von Eingaben – aber vor allem auch durch Softwaretools, die dazu automatisierte Abläufe anbieten. Typische automatisierte Anwendungen sind das Knacken und Testen von Passwörtern mittels spezieller Algorithmen oder *Brute-Force-Angriffen* oder die Auswertung des Verhaltens des „Opfersystems“ bei direktem Ansprechen über das Netz.

Penetration Tests werden einerseits von Angreifern mit Hinblick auf eine Ausnutzung der Schwachstellen durchgeführt, als auch andererseits in einem sicheren Rahmen und im Auftrag des Eigentümers der getesteten Computersysteme in Hinblick auf eine Aufdeckung und Schließung der Schwachstellen, bevor sie ausgenutzt werden.

Eine der einfachsten sowie gleichzeitig zuverlässigsten Methoden ist der gezielte Vergleich betroffener Software aus Schwachstellendokumentationen mit auf Computersystemen eingesetzter Software.

Nach der Entdeckung oder dem Erfahren einer Schwachstelle sollte zur Verhinderung einer böswilligen Ausnutzung, je nach Gefährdungsgrad, möglichst schnell gehandelt und die Schwachstelle geschlossen werden. Zur Beseitigung bzw. Abschwächung von Schwachstellen gibt es eine Vielzahl an Möglichkeiten; eine Auswahl sieht wie folgt aus:

- Die **Installation von Patches** ist inzwischen eines der bewährtesten und für den Anwender einfachsten Verfahren zur Schließung von Schwachstellen. Patches kommen in der Regel in Form von selbstinstallierenden bzw. ausführbaren Dateien, werden durch den Hersteller herausgegeben und beheben den entsprechenden Fehler im Quelltext. Als eine sich bewährte Verfahrensweise im Umgang mit Patches, hat sich die Ausbringung bzw. Installation im größeren Umfang in vordefinierten Wartungsfenstern erwiesen, um dadurch mögliche negative Auswirkung kontrollierbarer zu erkennen und zu beheben.

  Ein Nachteil, der daraus resultiert, ist, dass Patches in der Regel nicht unmittelbar nach deren Veröffentlichung installiert werden, wodurch Schwachstellen über eine gewisse Zeitspanne ausnutzbar bleiben.

- Ein **Workaround** ist eine inoffizielle Lösung, die häufig von Anwendern bzw. Betroffenen der Schwachstelle selbst entwickelt und über verschiedene Kommunikationskanäle wie Foren und Mailinglisten verbreitet werden. Workarounds beheben in der Regel nicht den Fehler im Quelltext, sondern führen lediglich dazu, dass dieser nicht ausgenutzt werden kann. Infolgedessen sind sie (ihrem Namen entsprechend) nur als Übergangslösung geeignet.

  Nicht selten sind Workarounds auch in Form von Anpassungen der Konfiguration anzutreffen, beispielsweise durch die Einrichtung weiterer Sicherheitsmechanismen wie Zugriffskontrollen. Diese können sich zum Beispiel auch in Form von technischen Lösungen manifestieren, zum Beispiel durch den Einsatz von Firewalls und *Access Control Lists* (ACLs).

- Das **Ersetzen des betroffenen Softwareprodukts** durch alternative Produkte ist ebenfalls eine mögliche Maßnahme. Beispielsweise könnte ein HTTP-Service bei Bekanntwerden einer Schwachstelle durch einen anderen ersetzt werden (z.B. *Apache HTTP Server* durch *nginx*). Diese Möglichkeit ist jedoch nicht immer gegeben, falls spezielle, eventuell selbstentwickelte Software betroffen ist (siehe auch 2.5) oder es keine Alternative gibt.

- Eine der Maßnahmen, die in der Regel nur getroffen werden, falls keine der oben genannten Maßnahmen möglich ist oder zum Erfolg führt, ist die **Außerbetriebnahme der Software** (und somit unter Umständen auch des Dienstes) oder des **kompletten Gerätes**.

- Eine weitere Maßnahme zur Abschwächung kann auch in Form der **Überwachung einer Schwachstelle** auf ihre Ausnutzung erfolgen. So wird zwar im Allgemeinen nicht die Ausnutzbarkeit an sich verhindert, jedoch kann durch eine unmittelbare Erkennung der Ausnutzung der Schaden durch Anwendung weiterer Maßnahmen minimiert werden.

Worauf auf jeden Fall geachtet werden muss, sind mögliche Beeinträchtigungen, die in Folge von Maßnahmen zur Schwachstellenbeseitigung auftreten können. Beispielweise kann es vorkommen, dass durch die Installation eines Patches weitere Schwachstellen eingebracht werden oder die Funktionalität einer Anwendung gestört wird. Im Falle des Heartbleed Bugs wurde zeitnah nach Bekanntwerden, eine gepatchte Version veröffentlicht, aus welcher der Fehler im Quelltext entfernt wurde.

2.3.6 Prävention von Schwachstellen

Ein Aspekt der in bestehenden Lösungen und Standards eher weniger Aufmerksamkeit bekommt, jedoch mindestens genau so effektiv wie die Beseitigung von Schwachstellen auf Computersystemen sein kann, ist die Prävention von Schwachstellen. Ein entscheidender Vorteil der Miteinbeziehung dieses Aspektes ist, dass Schwachstellen im Idealfall durch geeignete organisatorische sowie technische präventive Maßnahmen überhaupt nicht entstehen und die Gefahr einer Ausnutzung minimal gehalten wird.

Die Beseitigung von Schwachstellen – beschrieben im vorhergehenden Abschnitt – wird oft als das endgültige Ziel im Schwachstellenmanagement angesehen, wobei die vorhergehenden Aktivitäten, die Identifikation und Klassifikation, lediglich dazu hinführen bzw. eine effiziente Organisation unterstützen sollen. Die Prävention kann diesen Ansatz jedoch auch weiterführen und im Sinne eines „Lernens aus Schwachstellen“ angesehen werden, um den Aufwand jeder der anderen Aktivitäten mittel- und langfristig zu verringern oder auch effizienzsteigernd zu wirken:
Beispielsweise können Vergleiche von Schwachstellenstatistiken zweier gleichartiger Softwareprodukte, wie zweier HTTP Server mit gleichem Funktionsumfang, dazu führen, dass das Produkt mit weniger bekannten Schwachstellen als Nutzungsstandard innerhalb einer Organisation festgelegt wird. In diesem Fall könnte der Aufwand, der vom zuständigen Personal zur Beseitigung der Schwachstellen betrieben werden muss, verringert werden. Als präventive Maßnahme zur Steigerung der Effizienz der Identifikation, Klassifikation und Detektion von Schwachstellen kann jedoch im Beispiel auch der HTTP Server ausgewählt werden, dessen Schwachstellen ausführlicher dokumentiert sind.

Andererseits ist der effektivste Ansatzpunkt bei der Prävention von Schwachstellen, der Software-Entwickler selbst. Schwachstellen, die von vornherein in Software-Quelltext und Standardkonfiguration vermieden werden können bzw. nicht existieren, können entsprechend in der Theorie auch nicht ausgenutzt werden. In der Praxis ist es allgemein jedoch nicht möglich, Schwachstellen komplett aus Software fernzuhalten. Insbesondere für die Anwender und Nutzer von Software besteht diese Möglichkeit in der Regel nicht.

Die genannten Praktiken und Vorgehensweisen entsprechen allgemein einsetzbaren Maßnahmen mit Softwareschwachstellen. Bei der Umsetzung eines gemanagten Umgangs mit Schwachstellen in einer bestimmten Umgebung, in diesem Fall einem Hochschulnetz, ist es jedoch notwendig abzuwägen, welche dieser Vorgehensweisen anwendbar und sinnvoll sind.

Die Ausarbeitung der Charakteristika von Hochschulnetzen sowie der Betrachtung der darin auftretenden technischen sowie organisatorischen Herausforderungen sind daher essentiell für die Konzepterstellung eines Schwachstellenmanagements. Die Charakteristika von Hochschulnetzen werden im folgenden Abschnitt, die darauf aufbauenden Herausforderungen in Abschnitt 2.5 behandelt.

2.4 Charakteristika von Hochschulnetzen

Die Netzstruktur der jeweiligen Hochschule ist in der Regel von ihrem spezifischen Aufbau und ihrer Organisation abhängig. Üblicherweise sind Hochschulen in mehrere Fakultäten,

diese wiederum in Institute und Lehrstühle unterteilt. Da es jedoch eine Vielzahl an unterschiedlichen Hochschulen gibt, deren Struktur sich deutlich unterscheiden kann, ist es kaum möglich, ein prototypisches Hochschulnetz vorzugeben, das die Vorgaben aller Hochschulen erfüllen kann. Aspekte wie die Anzahl der zur Verfügung stehenden Administratoren und Netznutzer, die verwendete Netzarchitektur, der Art und Anzahl der bereitgestellten Dienstleistungen oder auch geographische Aspekte können die Organisation des Hochschulnetzes stark beeinflussen.

Aus diesem Grund werden im Folgenden allgemeine Charakteristika von Hochschulnetzen anhand mehrerer Netzkonzepte von verschiedenen deutschen Hochschulen erörtert. Zu den betrachteten Hochschulnetzen gehört das vom Leibniz-Rechenzentrum der Bayerischen Akademie der Wissenschaften betriebene Münchner Wissenschaftsnetz [WH15], das Hochschulnetz der Ruhr-Universität Bochum zum Stand des letzten veröffentlichten Netzkonzeptes im Jahr 2009, das Hochschulnetz der Martin-Luther-Universität Halle-Wittenberg zum Stand des letzten veröffentlichten Netzkonzeptes im Jahr 2010 und die Hochschulnetze der Otto-Friedrich-Universität Bamberg und der Universität Hamburg.

In Bezug auf die relativ alten Netzkonzepte der Ruhr-Universität Bochum sowie der Martin-Luther-Universität Halle-Wittenberg ergab die Nachfrage nach einer neueren Version, dass dies bereits die aktuellsten Dokumentationen sind.

Das **Münchner Wissenschaftsnetz** wird in Abschnitt 3.1.1 genauer beschrieben und dient als Szenario zur Identifikation der Anforderungen an ein Schwachstellenmanagement in Hochschulnetzen.

Zum Stand des Netzkonzepts der **Ruhr-Universität Bochum**, waren an der Hochschule ca. 33.000 Studenten eingeschrieben und rund 4.800 Mitarbeiter beschäftigt. Als Campus-Universität zeichnet sie sich dadurch aus, dass ein Großteil der Gebäude über ein geographisch eher kleines Gebiet verteilt ist und eine Vernetzung der Einrichtungen entsprechend gut untereinander durch die Nutzung von Glasfaserleitungen gewährleistet werden kann. Das Dienstangebot umfasst Standarddienste wie E-Mail, DNS, WWW und FTP und können von allen Angehörigen der Universität genutzt werden. Hochschulspezifische Dienste sind unter anderem E-Learning, Informationsdienste und ein Online-Kursangebot. Netznutzern stehen mehrere Zugangsmöglichkeiten zum Hochschulnetz zur Verfügung, insbesondere an den Arbeitsplätzen, Rechnerräumen, von den Hörsälen aus, von vorkonfigurierten Arbeitsplätzen aus sowie von Studentenwohnheimen und auch über eine mittels VPN gesicherte Leitung von Zuhause aus. Die Authentifizierung geschieht über eine Kombination aus Nutzerkennung und Passwort. Das Gesamtnetz ist in mehrere IP-Subnetze unterteilt, welche beispielsweise Lehrstühlen zugewiesen sind, die diese nach Belieben Geräten zuordnen können. Die Verwaltung der IP-Subnetze geschieht mit Hilfe von jeweils lokal bestimmten „Netzbetreuern“. [Ruh09]

An der **Martin-Luther-Universität Halle-Wittenberg** waren bei Veröffentlichung des Netzkonzeptes rund 17.500 Studenten und 2.000 Mitarbeiter eingetragen, wobei die Gesamtzahl der Netznutzer bei rund 29.000 liegt. Auch wenn die Universität in zwei größere, durch die Saale getrennte Bereiche eingeteilt ist, existiert ein Glasfasernetz, das die meisten Standorte verbindet. Die meisten Dienste werden zentral vom Rechenzentrum angeboten, darunter Basisdienste wie Backup und Archiv, E-Mail, WWW aber auch weitere, wie ein auf LDAP basierendes Identitätsmanagement, ein Web-Content-Managementsystem sowie eine Lehr-

und Lernplattform. Der Internetzugang erfolgt über das vom DFN betriebene X-WiN, der technischen Basis des deutschen Forschungsnetzes [Deu14]. Das Hochschulnetz selbst ist in ca. 400 Subnetze eingeteilt, wobei die IP-Adressvergabe durch das Rechenzentrum selber durchgeführt wird und auch hier größtenteils zentrale DNS-Server zum Einsatz kommen. Zur Erleichterung der Organisation des Netzes wurde allgemein darauf geachtet, Server und Daten weg von einer Dezentralisierung in Richtung zentraler Unterbringung in das Rechenzentrum zu integrieren, welches für den größten Teil der Organisation zuständig ist. Auch an der Martin-Luther-Universität sind Zugangsmöglichkeiten zum Netz leicht nutzbar und in großer Zahl vorhanden. [Mar10]

Die **Universität Bamberg** zeichnet sich durch rund 14.000 Studierende und 2.000 Mitarbeiter [Uni15b] sowie mehrere über die Stadt Bamberg verteilte Standorte der universitären Einrichtungen aus. Die Basisdienste werden zentral vom Rechenzentrum angeboten, allerdings gibt es auch Dienste, die durch dezentral installierte Server realisiert werden. Zudem sind Dienste der Bibliothek und der Universitätsverwaltung zentral realisiert. In das Dienstangebot ist zudem *Eduroam* integriert. Generell wird die komplette Infrastruktur durch das Rechenzentrum betrieben, wobei sich Zuständigkeiten und Verantwortlichkeiten auf mehrere Abteilungen aufteilen, die sich beispielsweise für den Betrieb des Netzes, der Client-PCs und Server unterscheiden. Zur Netznutzung ist eine Authentifizierung durch eine gültige Kombination aus Benutzerkennung und Passwort erforderlich und der Zugang ähnlich wie bei den vorher beschrieben Universitäten sowohl auf dem Universitätsgelände über WLAN, PC-Räume sowie über das Internet mittels VPN-Verbindung möglich. [Uni15c]

Die **Universität Hamburg** zählt mit ca. 42.000 Studierenden und rund 12.000 Mitarbeitern [Uni15d] zu den größten Universitäten Deutschlands. Die Einrichtungen und Standorte sind über das gesamte Stadtgebiet und auch außerhalb der Stadt verteilt. Das Hochschulnetz basiert auf einem öffentlichen IPv4 „Class-B“ Netz und ist in ca. 430 Subnetze unterteilt, die den jeweiligen Einrichtungen zugeordnet sind. Generell wird von der Nutzung von privaten IP-Adressen abgesehen. [Net14]

2.4.1 Beschaffenheit und Organisation

Hochschulnetze verfolgen in erster Linie den Zweck, die Forschung und Lehre in den Hochschulen und Instituten sowie die Ausbildung von Studenten zu unterstützen und zu erleichtern. Allgemein dienen sie der Kommunikation – zum einen intern, aber auch als Zugangspunkt zu größeren Netzen wie dem deutschlandweiten *Deutschen Forschungsnetz* (DFN), dem europaweiten Forschungsnetz *GÉANT* oder dem Internet – aber auch als Basis für viele bereitgestellte IT-Dienste.

2.4.1.1 Verwaltung des Netzes

Für die Planung, Bereitstellung und den Betrieb des Hochschulnetzes ist allgemein das jeweilige Rechenzentrum zuständig, wobei es üblich ist, das Gesamtnetz je nach Funktionalität, Lokalität oder Struktureinheit in mehrere Subnetze zu unterteilen. Unmittelbare Vorteile, die sich daraus ergeben, sind eine Entlastung des Netzes, da subnetzinterne Kommunikation in Verbindung mit geeigneten Switches nicht in das Gesamtnetz weitergeleitet wird und eine erhöhte Sicherheit durch effektivere Zugriffskontrollmöglichkeiten sowie ein gezielteres Routing. Organisatorische Vorteile können sich durch eine derartige Strukturierung insofern erge-

ben, da Zuständigkeitsbereiche – sowohl rechenzentrumsintern aber auch an Mitarbeiter der einzelnen Institutionen – effektiv jeweiligen Rollen und Personen zugeordnet werden können. An vielen Hochschulen werden aus diesem Grund die Verantwortung bzw. Teilaufgaben bei der Verwaltung von Subnetzen an mehrere, typischerweise außerhalb des Rechenzentrums beschäftigte, Mitarbeiter (in der Regel sogenannte *Netzbetreuer* oder auch *Netzverantwortliche*) vergeben, die beispielsweise die Verwaltung der Namens- und Adressräume sowie die Dokumentation der Endgeräte übernehmen und die Mitarbeiter des Rechenzentrums bei Problemen in ihrem Subnetz unterstützen beziehungsweise auch als erste Ansprechpartner bei Störungen, Fragen und für Benachrichtigungen gelten.

Eine Alternative zu diesem Ansatz ist die zentrale Verwaltung der Subnetze, die jedoch bei einer sehr hohen Anzahl verschiedener Institutionen nur mit ausreichend zur Verfügung stehendem Personal im Rechenzentrum bewältigt werden kann. Generell wird jedoch darauf geachtet, dass die Netzkomponenten möglichst fernwartbar und die Verwaltung vom Arbeitsplatz der jeweiligen zuständigen Personen erfolgen kann – beispielsweise durch geeignete Managementsysteme über das zu diesem Zweck entwickelte *Simple Network Management Protocol* (SNMP) oder dem Einsatz spezieller herstellerspezifischer Lösungen.

2.4.1.2 IT-Dienste im Hochschulnetz

Allgemein ist eines der herausragendsten Charakteristika, dass das Hochschulnetz selbst eine IT-Dienstleistung an dessen Nutzer ist. Der Dienstleister, das Hochschulrechenzentrum, sowie Kunden und Dienstnutzer – Wissenschaftler und Mitarbeiter der einzelnen Einrichtungen sowie Studenten und Forscher aus dem internationalen Raum – befinden sich in der Regel im gleichen Netz.

In Hochschulen gibt es oft ein weites Spektrum an angebotenen IT-Dienstleistungen, zu welchen neben klassischen Internet-Diensten auch viele hochschulspezifische Dienste gehören. Als grundlegendste Dienstkomponenten für die effiziente Nutzung eines Netzes gelten DHCP (*Dynamic Host Configuration Protocol*) und DNS (*Domain Name System*), die inzwischen seit längerer Zeit nicht nur in Hochschulnetzen zur Mindestausstattung gehören. Durch den Einsatz von DHCP-Servern fällt für Nutzer bei der Einbindung von Endgeräten ein essentieller Teil der Konfiguration weg, da die Endgeräte bei ihrer Kopplung ans Netz automatisch eine IP-Adresse zugewiesen bekommen; statische Adressen müssen entsprechend eingetragen werden. Insbesondere in einer Hochschulumgebung mit mehreren tausend potenziellen Netz-Nutzern, die je nach Bedarf Clientgeräte (z.B. Smartphones, Tablet-PCs, Notebooks) in das Netz einbinden oder vom Netz trennen, ist die Möglichkeit einer einfachen Benutzbarkeit des Netzes von Bedeutung. Dazu gehört auch die Verwendung von für Menschen verständlichen Bezeichnern für Geräte und Dienste im Netz, die durch den Einsatz von Nameserver realisiert werden. In der Regel werden Name- und DHCP-Server zentral vom Rechenzentrum betrieben, wobei diese Dienste je nach Konzept auch von einzelnen Institutionen innerhalb des Hochschulnetzes angeboten werden können. Des Weiteren stellen Rechenzentren im Allgemeinen zentral E-Mail- und Webdienste sowie Datei-, Backup- und Archivierungssysteme zur Verfügung. Darauf aufbauend finden sich in Hochschulnetzen einige Dienste zum allgemeinen Management und der Organisation der Tätigkeiten von Studenten und Mitarbeitern; darunter beispielsweise zentrale IT-gestützte Dienste zur Unterstützung des Vorlesungs- und Übungsbetriebs, zur Verwaltung von Noten und Prüfungen und E-Learning Systeme.

Weitere Nutzer und Anbieter von Diensten sind jedoch oft auch, zwar separate, aber uni-

versitätsnahe Einrichtungen, die in das Hochschulnetz eingegliedert sind; mögliche Beispiele sind wissenschaftliche Einrichtungen, Bibliotheken, Studentenwohnheime, Museen und zur Hochschule gehörende Kliniken. Je nach Angehörigkeit einer Organisation oder eines Instituts haben die entsprechenden Nutzer im Hochschulnetz bestimmte Rollen und Berechtigungen. Beispielsweise sind administrative Dienste im Hochschulnetz Mitarbeitern des Rechenzentrums bzw. Netzadministratoren oder auch, falls vorhanden, Netzbetreuern vorbehalten, wobei es auch hier wieder Einschränkungen in Bezug auf Funktionalität oder den Umfang der zur Verfügung gestellten Daten gibt.

Nicht selten kommen innerhalb von Hochschulnetzen private IP-Adressen zum Einsatz. Im Gegensatz zu öffentlichen IP-Adressen können diese nicht direkt aus dem Internet heraus geroutet werden, sondern Anfragen müssen erst über Techniken wie *Network Address Translation* (NAT) entsprechend verarbeitet werden.

2.4.1.3 Heterogenität im Hochschulnetz

Die Art der mit dem Hochschulnetz verbundenen Endgeräte ist vielfältig: Aufgrund der unterschiedlichen Aufgaben, Vorlieben und Erfahrungen der Netznutzer gibt es sehr häufig viele verschiedene Gerätetypen und Gerätehersteller sowie eine große Bandbreite an genutzten Betriebssystemen und Softwareprodukten. Neben den „Standardgeräten“ zur Erledigung der jeweiligen Aufgaben der Mitarbeiter für Wissenschaft und Betrieb sowie der Studenten, nämlich in der Regel immobile Rechner wie Desktop-PCs und Servergeräte, haben sich nun auch seit längerem Notebooks aufgrund ihres flexibleren Einsatzes am Arbeitsplatz sowie auch als Möglichkeit zur Erleichterung der Telearbeit etabliert.

Relativ neu in den letzten Jahren hinzugekommen ist der verstärkte Gebrauch von Smartphones, Tablet-PCs und weiteren persönlichen Geräten, die in das Netz integriert werden können, darunter zum Beispiel Smartwatches – computergestützte Uhren. Neben herkömmlichen Betriebssystemen wie Microsoft Windows und populären Linux- und UNIX-Distributionen wie beispielsweise *openSUSE*, *Ubuntu*, *Solaris* oder Derivate der *Berkeley Software Distribution* (BSD) [Bod15] ist der Einsatz von Betriebssystemen im Hochschulnetz tendenziell durch eine sehr hohe Vielfalt und aus Sicht des Betreibers, dem Rechenzentrum, zum Großteil durch starke Intransparenz geprägt, da netzweite Kontrollen und Vorschriften, alleine durch die Einbindung privater Geräte durch Mitarbeiter, Gäste und Studenten, nicht möglich sind. Diese Tendenz wurde in den letzten Jahren durch eine steigende Beliebtheit an mobilen Geräten und Betriebssystemen für mobile Geräte – z.B. Apples *iOS* oder das auf Linux basierende *Android* – sowie, speziell in diesem Bereich, dem Einsatz alternativer Prozessorarchitekturen weiter verstärkt. Die leistungsstarken Prozessoren auf *CISC*-Architektur („Complex Instruction Set Computer“) der Arbeitsplatzrechner wurden in mobilen Geräten beispielsweise durch stromsparendere Prozessoren auf *RISC*-Basis („Reduced Instruction Set Computer“) ersetzt. Infolgedessen wird Software auch für eine hohe Zahl an Prozessorarchitekturen entwickelt, wodurch die Softwarelandschaft im Hochschulnetz weiter an Umfang gewinnt.

Entsprechend vielfältig fällt die Auswahl an Softwareanwendungen aus, die sich in einem Hochschulnetz finden lassen. Diese reichen von Produkten großer Hersteller über alle Arten von Open-Source-Produkten, bis hin zu selbstentwickelten Anwendungen für wissenschaftliche Zwecke sowie den privaten Gebrauch.

2.4.1.4 Zugangsmöglichkeiten zum Netz

Die Zugangsmöglichkeiten zum Netz sind generell mit Absicht zahlreich und unkompliziert gestaltet: Hochschulnetze zielen in erster Linie meist nicht auf eine hochsichere Infrastruktur ab, sondern auf eine möglichst verlässliche Verfügbarkeit der Dienste. Angehörige einer Hochschule haben oft mehrere Möglichkeiten, ihre Geräte in das Netz zu integrieren. Neben einem Wireless LAN Zugang ist das Angebot von Rechnerräumen inzwischen zum Standard geworden. Zum einen befinden sich dort bereits zur Nutzung vorkonfigurierte Rechner, zum anderen Netzdosen, über die private Geräte mit dem Netz verbunden werden können. Für Mitarbeiter und Studenten hat sich jedoch auch die Möglichkeit etabliert, aus anderen Netzen über einen VPN-Tunnel auf das Hochschulnetz zugreifen zu können. In jedem Fall muss sich der Nutzer aber – üblicherweise durch eine Kombination aus netzweit gültiger Nutzerkennung und Passwort – authentifizieren. Die Anzahl der Nutzer kann je nach Größe der betrachteten Hochschule bzw. je nach Anzahl und Art der an das Hochschulnetz angeschlossenen Einrichtungen stark variieren.

Durch die mit Absicht so einfach wie möglich gehaltene Nutzbarkeit des Hochschulnetzes, die vielen Rollen in einer Hochschulumgebung und die Etablierung mobiler Geräte verhält sich das Netz bezüglich seiner Größe (erfasst anhand der Anzahl der verbundenen Geräte) entsprechend hochdynamisch.

2.4.2 Mandantenfähigkeit im Hochschulnetz

Rechenzentrumsexterne Nutzer in einem Hochschulnetz können allgemein verschiedene Rollen annehmen, die möglicherweise aus Sicht des Rechenzentrums nicht ganz transparent oder komplett unbekannt sind. Ein Grund dafür ist, dass die Zuständigkeit des Rechenzentrums – je nach beispielsweise Struktur, Größe und Aufbau des Netzes – lediglich „bis zur Datendose“ in den jeweiligen Instituten reicht und die Institute für ihre interne Netzinfrastruktur insofern selbst zuständig sind.

Die Mandantenfähigkeit hat daher auch im Hochschulbereich und den darin angebotenen IT-Diensten eine hohe Relevanz. Als Mandanten werden in diesem Fall Kunden beziehungsweise Nutzer des gleichen Dienstes bezeichnet, die alle auf dieselbe Anwendungsinstanz und eine zentrale Benutzeroberfläche zugreifen. Die bearbeiteten Daten, Konfigurationsmöglichkeiten und der zur Verfügung gestellte Funktionsumfang muss jedoch den bestimmten Nutzercharakteristika, insbesondere den aufgrund der unterschiedlichen Zuständigkeiten resultierenden Berechtigungen, angepasst sein. Die Realisierung erfolgt in der Regel durch eine Trennung der Daten pro Nutzer, die untereinander isoliert gehalten werden [KMK12].

Vor allem in einem Hochschulnetz, mit Angehörigen vieler verschiedener Einrichtungen und Inhaber verschiedener Rollen und Berechtigungen ist das Vorhandensein mandantenfähiger Anwendungen eine Notwendigkeit. Beispielsweise muss ein Dienst zur Vorlesungsverwaltung unterschiedliche Informationen und Funktionen für Lehrstuhl-Mitarbeiter, die eine Vorlesung anbieten, als für Studenten, die sich über Vorlesungen informieren und diese besuchen wollen, zur Verfügung stellen.

Potentielle Mandanten mit netzadministrativen Aufgaben im Hochschulumfeld sind insbesondere Netzadministratoren, Netzbetreuer bzw. Netzverantwortliche der Subnetze von den

jeweiligen Institutionen und Forschungsbereichen sowie darin befindliche Systemadministratoren und Einzelnutzer mit Verantwortung für ihre eigenen Computersysteme.

Dabei kann eine Hierarchie entstehen, um Verantwortlichkeiten für Abläufe und Verwaltungsaufwand auf verschiedene Personen zu verteilen. Im Beispiel wäre ein möglicher Ansatz, dass Systemadministratoren und Einzelnutzer den Verantwortlichen des Subnetzes, in dem ihre Rechner liegen, bei Problembehebungen in Bezug auf ihre Systeme unterstützen. Je nach Größe eines Subnetzes ist es auch möglich und sinnvoll, diese in weitere organisatorische Bereiche zu unterteilen und einzeln durch weitere (Sub-)Netzbetreuer verwalten zu lassen, die ihren übergeordneten Netzbetreuer bei seinen Aufgaben unterstützen. Die Spitze der Hierarchie bildet das für das Hochschulnetz verantwortliche Rechenzentrum als Dienstleister und die dort beschäftigten Netzadministratoren.

Da jede der Rollen eine andere Aufgabe und in Bezug zur Anzahl der relevanten Computersysteme einen anderen Umfang hat, muss jeder Rolle eine Sicht auf die Schwachstellendaten und Informationen über die Systeme bereitgestellt werden, die diese erfüllt: Sinnvoll ist es auch, Rollen mit der Überwachung des aktuellen und fortlaufenden Zustandes eines Teilnetzes oder auch letztendlich des Gesamtnetzes zu betrauen, um eine ganzheitliche Sicht zu bekommen.

2.5 Herausforderungen im Schwachstellenmanagement in Hochschulnetzen

Herausforderungen, die das Schwachstellenmanagement im Hochschulumfeld betreffen, resultieren aus verschiedenen Gründen. So sind es weniger technische Hürden, sondern vielmehr die hochschulnetzspezifischen organisatorischen Aspekte sowie zum Teil auch rechtliche Rahmenbedingungen und das allgemeine Nutzerverhalten im Netz, die zu erschwerten Bedingungen bei einem kontrollierten Umgang mit Schwachstellen führen können. Zudem gibt es in einem für die Erfüllung der Aufgaben der Mitarbeiter und Studenten aufgebauten Netz Unterschiede, die beispielsweise in Unternehmensnetzen oder Heimnetzen nicht oder in einem nur geringeren Umfang auftreten.

Eine organisatorische und aktivitätsunabhängige Herausforderung, die Hochschulnetze betrifft, ist, dass das Schwachstellenmanagement aufgrund der Zusammenarbeit und Abhängigkeit der Einrichtungen vom Rechenzentrum, organisations- bzw. institutionsübergreifend angewendet wird. In einem derartig besonderen Umfeld ist es von großer Bedeutung, dass Rollen und Zuständigkeiten klar definiert sind, da es prinzipiell bei mehreren zusammenarbeitenden Institutionen keine klare Hierarchie bzw. Rangordnung gibt, sondern mehrere.

2.5.1 Herausforderungen an die Identifikation

Ein grundlegendes Problem, das organisatorisch bereits vor der Umsetzung eines Schwachstellenmanagements anzusehen ist, ist die Abhängigkeit von der Einschätzung des Anwenders der Identifikation von Computersystemen als Assets – in diesem Fall Geräte, die im Rahmen des Schwachstellenmanagements als schützenswert angesehen werden. Oft werden Computersysteme von Nutzern nicht als solche realisiert und bleiben bei der Betrachtung außen vor. Typische Gerätegruppen, die diesem Problem nicht selten ausgesetzt sind, sind mobile

Geräte und Netzkomponenten wie Firewalls, Router und generell Appliances (physikalische oder virtuelle Geräte, die auf eine bestimmte Anwendung spezialisiert sind) und insbesondere auch „Testsysteme“ für Soft- und Hardware. Der Nutzer muss sich jedoch bewusst werden, dass jedes ans Hochschulnetz angeschlossene Gerät ein weiteres Risiko darstellt.

Ein großer Teil der Herausforderungen, die auf dem Weg zu einem Schwachstellenmanagement im Hochschulnetz berücksichtigt werden müssen, begründen sich jedoch auch auf durch die Struktur des Netzes zurückzuführende Gegebenheiten. Generell ein nicht unerhebliches Problem bei der Behandlung von Schwachstellen ist die hohe Heterogenität der Geräte und Softwareprodukte, die sich unter anderem aus den verschiedenen Dienstangeboten im Netz sowie den unterschiedlichen Aufgaben der Institutsmitarbeiter und den vielen verschiedenen Rollen ergibt. Da verschiedene Gerätetypen und Betriebssysteme teilweise für einen bestimmten Zweck entworfen wurden, bietet es sich an, diese Vorteile zu nutzen und zur Erfüllung bestimmter Aufgaben einzusetzen. Beispielsweise sind einige kommerzielle Softwareprodukte in Funktionsumfang, Benutzerfreundlichkeit und insbesondere dem geleisteten Support noch immer ihren auf Open-Source-Basis entwickelten Konkurrenten überlegen, wodurch auf Computersystemen – Clients als auch Servern – je nach Zweck Plattformen eingesetzt werden, die diese Produkte unterstützen. Auch ist es möglich, dass es teilweise für bewährte Produkte gar keine Alternativen gibt.

Das führt dazu, dass im Schwachstellenmanagement eine große Anzahl und Vielfalt an unterschiedlichen Architekturen und Software berücksichtigt werden muss.

Das NIST beschreibt in seinem *Guide to Enterprise Patch Management Technologies* [Mur13] darüber hinaus verschiedene Geräte und Lösungen, die das Schwachstellen- und Patchmanagement erschweren und vor allem im Hochschulumfeld verstärkt vorkommen: Im Besonderen nicht-gemanagte Geräte, spezielle (teilweise selbstentwickelte) Lösungen, mobile Geräte und virtuelle Maschinen:

- **Nicht-gemanagte Geräte**
  Der Einsatz nicht-gemanagter Geräte, also Hosts, bei denen insbesondere Patches nicht an zentraler Stelle z.B. durch das Hochschulrechenzentrum verwaltet werden, führt dazu, dass die Behebung von Schwachstellen mittels Patches durch den Nutzer durchgeführt werden muss. Oft fehlt diesem jedoch das dazu notwendige Fachwissen und die Motivation, Schwachstellen zu beheben. Da Schwachstellen je nach vorhandenem Verständnis entweder gar nicht in dem Bewusstsein des Nutzers vorhanden sind oder sehr theoretisch und ungefährlich erscheinen und ihr Vorhandensein an sich nicht zu Schaden führt, sondern sie zunächst ausgenutzt bzw. ausgelöst werden müssen, wird ihre Beseitigung nicht selten als überflüssig erachtet.

- **Spezielle Hard- und Softwarekomponenten**
  Auch spezielle – von der Standardkonfiguration abweichende – Hard- und vor allem Softwarelösungen sind ein Bestandteil des Hochschulnetzes. Aufgrund der vielen verschiedenen organisatorischen Notwendigkeiten durch die zahlreichen Aufgaben, die ein Hochschulnetz unterstützen soll – als groben Umriss: Die Unterstützung der Tätigkeiten der Mitarbeiter, Wissenschaftler und Studenten in Hinblick auf eine Steigerung der Effizienz – können kommerzielle Standardprodukte oft nicht alle Anforderungen erfüllen. Folglich wird maßgeschneiderte Software häufig durch Angehörige der

Hochschulen und Rechenzentren selbst, oder in Verbindung mit studentischen Arbeiten konzipiert und entwickelt. Auch durch die an Hochschulen betriebene Forschung werden in vielen Bereichen Anwendungen entwickelt, die Schwachstellen beinhalten, da auch die Entwickler und insbesondere auch Wissenschaftler sowie Studenten in Studienrichtungen weit fernab der Informationssicherheit, wenig Verständnis für sichere Programmierung haben. Die Pflege spezieller und natürlich auch selbstentwickelter Software muss jedoch im Gegensatz zu kommerziellen Produkten in der Regel von den Hochschulen selbst erledigt werden, was durch knappe Ressourcen, besonders budget- und personaltechnisch, nicht in ausreichendem Ausmaß bewältigt werden kann. Schwachstellen in speziellen Komponenten bleiben daher lange erhalten. Eine Erweiterung des Problems ergibt sich bei der Wiederverwendung von Programmcode und Bibliotheken, da hier Schwachstellen in mehrere Softwareanwendungen verteilt werden können, wodurch sich das Ausnutzungsrisiko aber auch der Behebungsaufwand vergrößert, da jede Software mit dem betroffenen Programmcode berücksichtigt werden muss.

- **Mobile Geräte**
  Die Möglichkeit der Einbindung von privaten Geräten in das Netz (umgangssprachlich „Bring Your Own Device“, BYOD), hat dazu geführt, dass auch immer mehr mobile Geräte wie Smartphones und Tablet-PCs einen Platz im Hochschulnetz finden. Die Probleme, die diese Geräte mit sich bringen, entstehen vor allem durch ihr Patchkonzept und eine Ausrichtung auf eine möglichst hohe Benutzungsfreundlichkeit. Üblicherweise gibt es bei solchen Geräten nicht die Möglichkeit, sie durch den Betreiber des Hochschulnetzes zu managen, sondern nur durch den Anbieter des Gerätes. Standardmäßig besitzen Nutzer von Betriebssystemen mobiler Geräte, wie *Android* oder *iOS* auf ihnen zudem keine Administratorrechte, wodurch die Möglichkeiten der Schwachstellensuche weiter eingeschränkt werden und die Behebung komplett dem Geräteanbieter überlassen werden muss.

- **Virtuelle Maschinen**
  Ein weiterer Komponententyp, der sich innerhalb von Kommunikationsnetzen etabliert hat, sind virtuelle Maschinen. Sie ermöglichen die Simulation mehrerer Rechner durch eine physikalische Maschine. Virtuelle Maschinen bringen einige Vorteile mit sich: Beispielsweise nimmt die Installation neuer virtueller Rechner an sich keinen weiteren Platz in Serverräumen ein, solange der als Basis dienende physikalische Rechner genug Ressourcen zur Verfügung hat. Sie sind zudem in der Regel einfach in Betrieb zu nehmen, da vorkonfigurierte Images (Abbilder von Speichermedien) leicht eingesetzt werden können. Ein weiterer Vorteil ist eine einfache Möglichkeit zur Erstellung von *Snapshots*, der Sicherung des Systemzustandes zu einem bestimmten Zeitpunkt, so dass dieser ohne weiteren Aufwand wiederhergestellt werden kann. Dabei muss beachtet werden, dass das System in Bezug auf vorhandene Schwachstellen bei Wiedereinspielung eines Snapshots den Zustand zum Zeitpunkt der Erstellung des Snapshots annimmt.

  Diese Vorteile machen virtuelle Maschinen vor allem auch für Tests von Softwareprodukten oder bestimmten Konfigurationen beliebt, da Testmaschinen in kürzester Zeit eingerichtet sind und die Möglichkeit besteht, Änderungen im Fehlerfall schnell wieder rückgängig zu machen – wodurch sie sich aber auch nachteilig im Hinblick auf

das Management von Schwachstellen auswirken können: Nicht selten ist es der Fall, dass Netznutzer selbst weitere virtuelle Maschinen im Einsatz haben – in der Regel als Clientsysteme – wodurch sich die Anzahl der Systeme im Netz noch dynamischer verhält.

Schlimmer sind jedoch virtuelle Maschinen, die im Rechenzentrum selber oder in Einrichtungen im Hochschulnetz als „Testserver“ genutzt werden, möglicherweise durchgehend erreichbar (und somit angreifbar) sind und aufgrund ihrer einfachen und schnellen Installation oft zu diesem Zweck genutzt werden. Oft besteht die Gefahr, dass der „offizielle“, also von der Leitung des Rechenzentrums festgelegte Weg der Anforderung einer virtuellen Maschine umgangen und die Mitarbeiter selbst virtuelle Maschinen einrichten und betreiben. Da sie keinen räumlichen Platz wegnehmen, verschwinden sie oft aus dem Bewusstsein des Verantwortlichen und werden schlichtweg vergessen. Entsprechend kommt es vor, dass sich das Risiko durch derartig im Einsatz befindliche virtuelle Maschinen erhöht, da ihre Wartung – vor allem die Installation von Sicherheitspatches – übersehen wird. Generell ist es so relativ schwierig, an zentraler Stelle einen netzweiten Überblick über betriebene virtuelle Maschinen zu behalten.

Daraus resultiert, dass bei der Identifikation von Schwachstellen ein großes Spektrum an Software und Konfigurationsmöglichkeiten berücksichtigt werden muss. Die Quellen für Schwachstelleninformation müssen entsprechend zahlreich sein, wodurch sich die Organisation der vielen verschiedenen Daten und Datenformate als Herausforderung erweisen kann.

2.5.2 Herausforderungen an die Klassifikation

Bei der Klassifikation von Schwachstellen kann vor allem die hohe Zahl an Nutzern sowie Verantwortlichkeiten herausfordernd auf die Umsetzung eines Schwachstellenmanagements wirken. Prinzipiell ist es letztendlich von der verantwortlichen Person abhängig, wie eine Schwachstelle bewertet und die Beseitigung als notwendig angesehen wird. Bei vielen involvierten Rollen gibt es daher in der Regel bei ungemanagten Abläufen keine einheitliche Vorgehensweise.

2.5.3 Herausforderungen an die Beseitigung, Abschwächung und Prävention

Ein Aspekt, der als Herausforderung für die Detektion von Schwachstellen als Teil der Beseitigung (siehe Abschnitt 2.3.5) auf Assets gilt, ist die Einbindung privater Rechner in das Hochschulnetz. Diese erschweren eine gezielte und einheitliche Suchmethodik nach Schwachstellen nach einem bestimmtem Softwareprodukt, da auf diesen Computersystemen beliebige Software installiert sein kann, ohne dass der Netzbetreiber die Möglichkeit hat, einen Überblick über das Softwareinventar im Netz zu bekommen.

Aber auch technische Herausforderungen für die Detektion von Schwachstellen folgen aus der hohen Vielfalt an Geräten und Betriebssystemen. So müssen beispielsweise Softwaretools (z.B. Softwareagenten), die zur Identifikation auf Systemen eingesetzt werden sollen, für ein großes Spektrum an Hardwarearchitekturen und Betriebssystemen entwickelt werden, um das Schwachstellenmanagement für einen uneingeschränkten Anwendungsbereich nutzbar zu machen.

Eine weitere technische Herausforderungen, die nicht nur, jedoch auch in Hochschulnetzen anzutreffen ist, ist die Nutzung privater IP-Adressen in Subnetzen innerhalb des Hochschulnetzes. Dadurch ist eine Detektion, insbesondere das *Scannen* nach Schwachstellen auf Assets über das Netz teilweise ohne weitere Vorkehrungen nicht möglich, da diese Geräte in der Regel nur innerhalb desselben internen Netzes sichtbar sind.

Die nicht selten hohe Anzahl an Studenten (und Mitarbeitern), die das Hochschulnetz nutzen und auch private Geräte entweder auf dem Hochschulgelände, oder über VPN aus dem Internet einbinden, ist des Weiteren eine Herausforderung bei der Beseitigung und Abschwächung sowie der Prävention von Schwachstellen.

Da die Nutzung der Dienste im Hochschulnetz möglichst gut erreichbar sein und die Nutzer im Idealfall von überall aus bei ihren Tätigkeiten unterstützen soll, ist es anders als in vielen Unternehmensnetzen nicht sinnvoll bzw. nicht möglich, bestimmte Produkte – sowohl Computersysteme als auch Software – präventiv im Netz zu verbieten, wodurch beispielsweise der Anwendungsbereich eingeschränkt werden und die Komplexität des Schwachstellenmanagements verringert werden könnte.

Hochschulen achten generell stärker auf den Erhalt der Handlungsfreiheit der Netznutzer als beispielsweise Unternehmen. Wissenschaftler, Mitarbeiter, Dozenten sowie Studenten muss die Möglichkeit gegeben werden, sich aus verschiedenen Quellen Wissen und Meinungen anzueignen – andere Ansätze würden alleine den in Abschnitt 2.2 genannten Zielen der Hochschulen widersprechen. Die genauen Rechte und Pflichen der Nutzer im Hochschulnetz werden jedoch in der Regel in den Nutzungsbestimmungen festgelegt, welchen die einzelnen Nutzer vor Nutzung des Netzes zustimmen müssen.

Da anders als beispielsweise im Unternehmen, nicht alle Geräte und Softwareprodukte im Hochschulnetz durch das Rechenzentrum bzw. die Hochschule beschafft werden, sondern insbesondere private Geräte einen großen Teil des Netzes ausmachen, lassen sich einige Maßnahmen, die sich im Unternehmen zur effektiven Prävention von Schwachstellen etabliert haben, nicht umsetzen.

Solche typischen Maßnahmen sind beispielsweise ein *Whitelisting* oder auch *Blacklisting* von Software und Geräten und können zum einen die Anzahl an Schwachstellen und zum anderen Detektions- sowie Behebungsaufwand stark reduzieren. Whitelisting bezeichnet die ausdrückliche, im Vorhinein festgelegte Nutzungserlaubnis von Produkten, wobei Soft- und Hardware außerhalb dieser Festlegung nicht von Mitarbeitern genutzt werden dürfen. Analog kann diese Maßnahme auch, je nach Umfang und erwartetem Evaluationsaufwand ungeeigneter Produkte als Blacklisting umgesetzt werden – einer Auflistung ausdrücklich zur Nutzung nicht-freigegebener Produkte.

Aus Sicherheitsgründen werden den meisten Mitarbeitern und somit Netznutzern in Unternehmensnetzen keine Administratorrechte auf den vom Unternehmen bereitgestellten Rechnern ermöglicht, wodurch Konfigurationsmöglichkeiten und die Installation beliebiger Software eingeschränkt wird. Auch wenn diese Maßnahme in Hochschulnetzen an den von der Hochschule zur Verfügung gestellten Rechnern teilweise durchgesetzt werden kann, besteht diese Möglichkeit nicht bei den privaten Rechnern der Nutzer.

Des Weiteren ist es aufgrund der hohen Nutzerzahl, verschiedenen Nutzergruppen und der Gewährung einer hohen Freiheit der Nutzer nicht möglich, netzumfassende Sicherheitsrichtlinien durchzusetzen. Diese könnten beispielsweise analog zu Richtlinien einer erforderlichen

Passwortstärke oder der Einrichtung einer Bildschirmsperre mit Authentifizierungsmechanismus – also organisatorischen Schwachstellen – auch für die Prävention von Softwareschwachstellen genutzt werden.

Da die Zuständigkeit des Rechenzentrums im Netz oft an der Datendose der Institutionen und Einrichtungen endet, ist – als Kontrast zur Erlaubnis von Softwareprodukten durch Softwarewhitelisting und -Blacklisting – im Übrigen eine netzweite Zwangsinstallation von Software ebenfalls nicht ausführbar. Beispielsweise ist ein Sammeln von Schwachstelleninformationen durch Softwareagenten daher nicht ohne Zustimmung der jeweiligen Intitutionen möglich. Generell wird daher, je nach technischer Umsetzung (z.B. der Verwendung von privaten IP-Adressen innerhalb eines Instituts), ein gewisser Teil des Hochschulnetzes nur auf freiwilliger Basis durch ein Schwachstellenmanagement abgedeckt sein.

Somit sind die Möglichkeiten des Rechenzentrums bei der netzweiten Beseitigung von Schwachstellen teilweise stark eingeschränkt bzw. auch gar nicht möglich.

2.5.4 Generelle Herausforderungen in Hochschulnetzen

Ein Problem, das sich Hochschul- und Unternehmensnetze teilen, ist das Verhalten der Netznutzer. Die regelmäßigen Meldungen und Erinnerungen bezüglich der Existenz neuer Updates und Patches für Betriebssysteme und Software erscheinen dem Nutzer in der Regel als intransparente und sinnlose Belästigung, wodurch sie möglicherweise ignoriert werden – wenn nicht sogar mit einer Deaktivierung des automatischen Update-Dienstes einhergehen. Aufgrund der allgemein größeren Freiheit in Bezug auf die Auswahl der Geräte und Software sowie der teilweise Vollzugriffsrechte von Nutzern im Hochschulnetz und der nur begrenzten Zuständigkeit des Rechenzentrums, ist dieses Problem im Hochschulumfeld noch kritischer und schwieriger lösbar. Weitere Aspekte wie Fachwissen über einen professionellen Umgang in Bezug auf die Administration von Computersystemen spielen bei der Identifikation und vor allem der Behebung von Schwachstellen eine größere Rolle, da diese ausschlaggebend für eine erfolgreiche Beseitigung bzw. Abschwächung sein kann. Aufgrund der gemischten Forschungs- und Studienschwerpunkte der Nutzer ist es wahrscheinlich, dass ein derartiges Fachwissen nicht überall vorhanden ist und sie im Umgang mit Schwachstellen unterstützt werden müssen.

Auch die Effektivität des Reportings, dem Melden von Schwachstellen an den jeweiligen Nutzer, ist abhängig von seinen Gewohnheiten und seinem alltäglichen Verhalten. Heutzutage haben die meisten Menschen beispielsweise mehr als nur eine E-Mail-Adresse: In der Regel mindestens eine private E-Mail-Adresse und als Mitglieder von Organisationen, bestimmten Arbeitsgruppen und automatisch bei der Inanspruchnahme gewisser Angebote wie z.B. Webhosting oder bei der Mitgliedschaft in Sozialen Netzwerken bekommen sie weitere E-Mail-Adressen, die sie nutzen können. Typischerweise ist das als Mitglied einer Hochschule in derselben Weise üblich und Mitarbeiter und Studenten bekommen auch hier teilweise weitere E-Mail-Adressen zugewiesen. Somit ist die Abfragehäufigkeit der entsprechenden Postfächer bei der Berichterstattung über E-Mails ein wichtiges Kriterium. Ähnlich verhält es sich auch bei anderen Medien wie Websites oder Ticketsystemen.

2.5.5 Abgrenzung zu Unternehmensnetzen

Grundsätzlich sind die auffälligsten Herausforderungen in Hochschulnetzen, die sie von Unternehmensnetzen unterscheiden, die netzweite, größere Freiheit der Nutzer und die gleichzeitig geringeren Kontroll- und Handlungsmöglichkeiten des Netzbetreibers. Während Netzadministratoren im Unternehmen oft Vollzugriff auf die Geräte und sogar mehr Rechte als der Nutzer selbst besitzen, ist das im Hochschulnetz nur auf einen eher geringeren Teil beschränkt – nämlich zum Rechenzentrum gehörende Geräte. Der direkte Einflussbereich des Hochschulrechenzentrums ist insofern deutlich geringer, wodurch ein erfolgreiches netzweites Schwachstellenmanagement nicht ohne entsprechend vorhandene Motivation der einzelnen Institute und Organisationen möglich ist. Die Verantwortung für die Gewährleistung von netzweiter Informationssicherheit liegt im Hochschulumfeld also nicht nur beim Netzbetreiber, sondern zum großen Teil auch bei den einzelnen Nutzern.

3 Anforderungen an ein Schwachstellenmanagement in Hochschulnetzen

In diesem Kapitel werden Anforderungen an ein Schwachstellenmanagement in Hochschulnetzen definiert. Die Grundlage der Anforderungen bildet das in Abschnitt 3.1.1 beschriebene Szenario des *Münchner Wissenschaftsnetzes* (MWN) sowie gängige Praktiken und Maßnahmen im Umgang mit Schwachstellen, welche aktuell (Stand November 2015) im MWN-betreibenden Hochschulrechenzentrum, dem *Leibniz-Rechenzentrum* (Abschnitt 3.1.2) unternommen werden.

Die Erläuterung der Form der Anforderungen wird schließlich im Abschnitt 3.2, die eigentliche Auflistung und Beschreibung der Anforderungen in Abschnitt 3.3 und eine Zusammenfassung in tabellarischer Form wird in Abschnitt 3.4 vorgenommen.

3.1 Szenarien zur Anforderungsanalyse

Das Szenario des Münchner Wissenschaftsnetzes (MWN) basiert auf vom *Leibniz-Rechenzentrum der Bayerischen Akademie der Wissenschaften* veröffentlichten Netzkonzepts (Stand 2015)[WH15][Lei15]. Dagegen dieses Szenario mit Erfahrungen aus der Praxis ergänzend, wurde das Szenario zum aktuellen Umgang mit Schwachstellen innerhalb des MWN am Leibniz-Rechenzentrum mittels einer Umfrage in Erfahrung gebracht. Das Umfrageformular ist in Anhang 1 zu finden.

3.1.1 Das Münchner Wissenschaftsnetz (MWN)

Das *Münchner Wissenschaftsnetz* ist ein Hochschulnetz, das mehrere, über ein großes geographisches Gebiet (innerhalb und außerhalb Münchens) verteilte, staatliche Hochschulen und deren Institute, wissenschaftliche Einrichtungen und Studentenwohnheime verbindet. Es wurde aufgrund seiner guten Dokumentation als Szenario zur Ermittlung der Anforderungen an ein Schwachstellenmanagement in Hochschulnetzen ausgewählt.

Für eine bessere Verständlichkeit der daraus abgeleiteten Anforderungen ist das Szenario in die verschiedenen Sichten des IT-Service-Managements unterteilt: Die des Dienstanbieters, des Kunden und Dienstnutzers, die jeweils unterschiedliche Anforderungen an die Dienste stellen. Da ein Hochschulnetz Angehörige jede dieser Rollen hat, ist es im Hinblick auf ein netzweites Schwachstellenmanagement wichtig, dass alle berücksichtigt werden. Eine Besonderheit von Hochschulnetzen, die auch ein Charakteristikum des MWNs ausmacht, ist, dass Dienstanbieter, Kunden sowie Nutzer alle in demselben Netz sind.

3.1.1.1 Der Dienstanbieter im MWN – das Leibniz-Rechenzentrum

Das zur *Bayerischen Akademie der Wissenschaften* gehörende *Leibniz-Rechenzentrum* (LRZ) ist für den Betrieb, die Bereitstellung sowie die Planung des Netzes verantwortlich. Die in den jeweiligen Einrichtungen angeschlossenen Endgeräte befinden sich jedoch nicht mehr im Zuständigkeitsbereich des Rechenzentrums, sondern werden durch den jeweiligen Kunden verwaltet.

Des Weiteren ist das Leibniz-Rechenzentrum Betreiber einer der wenigen Höchstleistungsrechner in Deutschland – dem *SuperMUC*.

Das Hauptziel des Angebots des Hochschulnetzes als Dienstleistung ist die Gewährleistung der Verfügbarkeit für seine Nutzer – zum einen des Netzes und dazugehöriger Dienstkomponenten selbst, aber andererseits auch der darin angebotenen IT-Dienste.

Zu diesem Zweck betreibt das Leibniz-Rechenzentrum verschiedene Systeme und Lösungen zur Aufrechterhaltung eines gewissen Sicherheitsstandards im Netz. Dazu gehören die Nutzung verschiedener Lösungen zur Detektion von und Reaktion auf Sicherheitsvorfälle, insbesondere IDS- sowie IPS-Lösungen (auf Basis von Suricata) sowie einem System zur Verwaltung von sicherheitsrelevanten Informationen und Vorfällen (engl. *Security Information & Event Management*, SIEM). Weitere getroffene Maßnahmen sind beispielsweise die Umsetzung von Mechanismen zur Zugangs- und Zugriffskontrolle (beispielsweise private IP-Adressen, VLANs sowie der Einsatz von Firewalls) in Kombination mit einem netzweiten Identitätsmanagement.

Andererseits soll den Netznutzern vor allem in Hinblick auf die Erfüllung ihrer wissenschaftlichen und alltäglichen Aufgaben und Tätigkeiten möglichst viel Handlungsfreiheit gewährleistet werden, auch in Bezug auf die Einhaltung von §4 des Hochschulrahmengesetzes, der Freiheit von Kunst und Wissenschaft, Forschung, Lehre und Studium [Bun15b].

Ziel des MWN und der darin angebotenen Dienstleistungen ist vor allem die Ermöglichung und Unterstützung der Netznutzer bei der Ausübung ihrer Aufgaben und Tätigkeiten.

Das Dienstangebot reicht von grundlegenden Diensten (bzw. Dienstkomponenten) zum effizienten Betrieb und einer einfachen Nutzung des Netzes, beispielsweise durch den Einsatz von zentralen DHCP- und DNS-Servern (welche Dienstkomponenten des Dienstangebotes des MWN darstellen), bis hin zur Organisation und dem Housing sowie Hosting von Domains und Servern.

Als mittlerweile unverzichtbarer Dienst zur Erfüllung der Aufgaben der Nutzer im MWN und weltweit hat sich die Verwendung von E-Mail erwiesen. Auch im MWN wird daher jedem Nutzer mindestens ein E-Mail-Postfach zur Verfügung gestellt.

Weitere Dienste zur Unterstützung der Tätigkeiten umfassen beispielsweise öffentliche und interne Webserver und Webservices, ein Archiv- und Backupsystem, das von rund 9.000 aus 450 Einrichtungen stammenden Rechnern genutzt wird, verschiedene Verzeichnisdienste auf Basis von LDAP oder Microsofts *Active Directory*, zentral angebotene Bibliotheksdienste, aber auch E-Learning-Systeme und Dienste zur Verwaltung des Hochschulbetriebs, insbesondere zur Planung von Veranstaltungen, Vorlesungen und des Übungsbetriebs.

Das Dienstangebot im MWN beinhaltet auch den redundant ausgelegten Zugang zum Internet – zum einen als Teil des Deutschen Forschungsnetzes (DFN) über das X-WiN, zum anderen als Backup-Leitung über einen weiteren Internet Service Provider.

Eine Herausforderung, die nicht nur in Unternehmen auftaucht sondern auch besonders staatliche Einrichtungen wie das Leibniz-Rechenzentrum betrifft, ist die allgemein auftretende Ressourcenknappheit. Zum einen in Bezug auf zur Verfügung stehendem Budget und zum anderen in Bezug auf zur Verfügung stehendem Personal. So kommen nur wenige Dutzend Mitarbeiter im LRZ mit der Aufgabe der Bereitstellung und des Betriebs des MWN auf weit über 120.000 Netznutzer.

Nicht zuletzt deswegen und im Sinne eines effektiven und effizienten Ressourceneinsatzes, einer besseren Zielorientierung sowie Qualitätsicherung der Angebote, wird im Rechenzentrum nicht nur für das Dienstmanagement, sondern auch allgemein zur Unterstützung der Geschäftsprozesse, ein strukturiertes IT-Service-Management umgesetzt.

3.1.1.2 Die Kunden im Münchner Wissenschaftsnetz

Die Kunden des Leibniz-Rechenzentrums, welche die Dienste innerhalb des Münchner Wissenschaftsnetzes in Anspruch nehmen, sind Hochschulen wie die *Ludwig-Maximilians-Universität München* (LMU) und die *Technische Universität München* (TUM) sowie Hochschulkliniken, Museen, Bibliotheken und allgemein wissenschaftliche Einrichtungen.

Der Kunde ist mitverantwortlich bei der Verwaltung des Netzes – inbesondere den ihm zugewiesenen Subnetzen – bzw. Unterstützung des Rechenzentrums, da die zentrale Verwaltung von den im MWN befindlichen rund 5000 IPv4 und IPv6 Netzen schwer bis unmöglich ist.

Zu diesem Zweck wurde innerhalb des Münchner Wissenschaftsnetzes die Rolle des Netzverantwortlichen eingeführt, dessen Aufgaben die Verwaltung der jeweiligen Namens- und IP-Adressräume, die Dokumentation der angeschlossenen Netze sowie Endgeräte und die Unterstützung des Rechenzentrums bei seinen Hauptaufgaben – die Planung, der Betrieb und auch die Erweiterung – mit Bezug zu dem jeweiligen Subnetz, umfassen. Weitere Aufgaben mit besonderer Relevanz in Hinblick auf den Umgang mit Schwachstellen ist die Mitarbeit bei der Fehlerbehebung innerhalb der eigenen Kundensubnetze und die Unterstützung des Rechenzentrums bei der Verhinderung von Missbrauch im Netz.

Da die Kunden im MWN vorwiegend Einrichtungen und Personen ohne Tätigkeiten in der System- sowie Netzadministration sind, haben eingesetzte Netzverantwortliche teilweise entsprechend kaum Erfahrung oder Fachwissen in diesem Bereich und müssen bei ihren Tätigkeiten und Aufgaben in dieser Rolle unterstützt werden. Zur Erleichterung ihrer Tätigkeiten gibt es ein mandantenfähiges, erweiterbares Tool, das die Netzverantwortlichen netzweit über eine Weboberfläche aufrufen können und einen Überblick über die in ihrem Subnetzen befindlichen Geräte bekommen können. Zu den bereitgestellten Informationen gehören beispielsweise die IP- und MAC-Adresse und der Bezeichner der verwendeten Datendose.

Des Weiteren besteht allgemein die Möglichkeit, dass die im vorherigen Abschnitt genannten Dienstleistungen des Rechenzentrums der Kommunikationsebene auch von den einzelnen Institutionen als Dienst erbracht werden können.

3.1.1.3 Die Nutzer des Münchner Wissenschaftsnetz und seiner Dienste

Zu den Nutzern im Hochschulnetz gehören vor allem Studenten, Mitarbeiter der Institutionen und Wissenschaftler innerhalb und außerhalb Deutschlands, beispielsweise im Rahmen von Konferenzen oder gemeinsamer, interuniversitärer Projekte.

Nutzungsberechtigung des Netzes und der Dienste hat jeder Nutzer mit einer gültigen Kennung, von denen ca. 130.000 aktiv sind. Dabei ist zu beachten, dass ein Nutzer mehrere Kennungen haben kann bzw. es auch Funktionskennungen für bestimmte Aufgaben gibt.

Die Geräte der Nutzer können generell durch eine Vielzahl an Zugangsmöglichkeiten in das MWN integriert werden. So wird in 250 an das MWN angeschlossenen Einrichtungen WLAN mit Hilfe von ca. 3.000 Accesspoints angeboten, über die sich mobile Geräte verbinden können. Außerdem gibt es rund 350 vorkonfigurierte Datendosen, über die weitere Geräte angeschlossen werden können. Eine Nutzung des Internetzugangs über das MWN kann außerdem über Eduroam erfolgen. Wie an vielen Hochschulen üblich, können Geräte auch von außerhalb, beispielsweise über das Internet, mittels VPN in das Netz integriert werden, wodurch der uneingeschränkte Zugang zu Diensten innerhalb des Hochschulnetzes ermöglicht wird. Wöchentlich registrieren sich auf diese Weise 20.000 Nutzergeräte über eine VPN-Verbindung im Netz. Auch haben Nutzer die Möglichkeit einen der zahlreichen, innerhalb der Einrichtungen gelegenen Rechnerräume mit den darin vorkonfigurierten Rechnern zu verwenden.

Die Größe des Netzes ist entsprechend umfangreich und umfasst im Maximum rund 180.000 gleichzeitig mit dem Netz verbundene Endgeräte, die sich in ungefähr 5.000 Server, 80.000 Arbeitsplatzrechner, und 100.000 über das angebotene Wireless LAN verbundene Rechner und mobile Endgeräte aufteilen.

Aufgrund der hohen Anzahl an Personen im Netz und der relativ großen Freiheit der Nutzer sowie der vielen Forschungs- und Interessensgebiete, befinden sich im Hochschulnetz zahlreiche verschiedene Geräte, Geräterollen und Softwareprodukte im Einsatz. Beispiele hier sind Client- und Servergeräte unterschiedlichster Hersteller und Architekturen, Drucker, Geräte der Netzinfrastruktur wie Router, Switches und verschiedene Appliances, Notebooks, Tablet-PCs sowie Smartphones. Auch im Münchner Wissenschaftsnetz ist es möglich, private Geräte unkompliziert zu integrieren.

Ein ähnliches Bild ergibt sich bei der Betrachtung der Software, die im MWN genutzt wird. Prinzipiell gibt es keine netzweiten Vorgaben bezüglich der Benutzung von speziellen Produkten für Softwaretypen (z.B. Webbrowser), wodurch es bei einer derart hohen Anzahl an Nutzern praktisch nicht möglich ist, lediglich einen Überblick darüber zu bekommen. Im Allgemeinen findet sich im MWN jede gängige Art von Software wieder: Von kommerziellen und freien Produkten bis Open-Source-Software sowie selbstentwickelten Programmen und Softwaresystemen, die der Verwaltung oder auch der Forschung, beispielsweise in Form von Simulationen, dienen. Gleiches gilt auch in Bezug auf die Betriebssysteme im Netz. Auch wenn die Variation dieser auf den zum Rechenzentrum gehörenden Rechnern relativ gut einschränkbar ist, so ist aus Sicht des LRZ – da es nur bis zur Datendose der jeweiligen Einrichtungen zuständig ist – eine Kontrolle jenseits der Datendosen nicht möglich. Aber selbst innerhalb des Rechenzentrums ist die Softwarelandschaft nicht komplett kontrollierbar: Da die Mitarbeiter des LRZ in der Regel auf Anforderung administrative Rechte auf

ihren Arbeitsplatzrechnern bekommen, steht ihnen die Installation und Nutzung beliebiger Softwareprodukte praktisch frei.

3.1.2 Aktueller Umgang mit Schwachstellen am Leibniz-Rechenzentrum

Im Münchner Wissenschaftsnetz werden eine Vielzahl an IT-Diensten genutzt, die Angehörige der Hochschulen und Institutionen bei der Erfüllung ihrer Tätigkeiten unterstützen sollen oder diese erst ermöglichen. Das Leibniz-Rechenzentrum als Betreiber des größten Teils dieser Dienste – andere werden auch durch die Institutionen selber angeboten, jedoch dienen die von LRZ angebotenen Dienste immer als Grundlage (vgl. vorheriger Abschnitt) – muss im Sinne seiner Kunden die Sicherheit der Dienste kontrollieren und gewährleisten, wobei die Behebung von Schwachstellen ein Kernelement dabei einnimmt.

> Das vom Leibniz-Rechenzentrum angestrebte Ziel ist insbesondere der Schutz des Netzes und allgemein der darin angebotenen Dienste und Daten.

Die zu diesem Zweck am LRZ unternommenen Tätigkeiten im Umgang mit Schwachstellen wurden mit Hilfe einer für diesen Zweck entwickelten freiwillig auszufüllenden Umfrage erfasst, welche im Anhang 1 dieser Arbeit beiliegt.

3.1.2.1 Die Erfassung der Tätigkeiten

Die Umfrage richtet sich vor allem an Servicebetreiber und Administratoren, aber auch an jeden anderen Mitarbeiter, der eine oder mehrere der folgenden Aufgaben des Schwachstellenmanagements ausführt:

1. Die **Identifikation**,
2. die **Klassifikation** und insbesondere **Bewertung**,
3. die **Beseitigung** sowie **Abschwächung** von Schwachstellen (vgl. Abschnitt 2.2)

Maßnahmen zur Prävention von Schwachstellen wurden nicht berücksichtigt, da diese zum Zeitpunkt der Umfrage noch nicht ausgearbeitet waren.

Bei der Betrachtung der Auswertung ist zu beachten, dass sich die prozentualen Angaben auf die Anzahl der Personen beziehen, die auf die jeweilige Frage eine Antwort gegeben haben, bzw. diejenigen Dienstbetreiber nicht berücksichtigt wurden, welche die Frage betreffende Aktivität nicht ausführen. Die Summe aller Antworten auf eine Frage kann über 100% hinausgehen, da die Teilnehmer bei jeder Frage die Möglichkeit hatten, eine Auswahl mehrerer Antworten anzugeben.

Gängige Antwortmöglichkeiten innerhalb des Fragebogens sind bereits vorgegeben und mit Kontrollkästchen zum Ankreuzen versehen, um zum einen ähnliche Antworten für eine einfachere Auswertung zu standardisieren und zum anderen eine benutzerfreundlichere Bearbeitung anbieten zu können. Des Weiteren ist am Ende des Fragebogens speziell weiterer Platz für Freitext vorgesehen, um den Umfrageteilnehmern die Gelegenheit zu geben, gezielt Anforderungen und persönliche Wünsche an ein Schwachstellenmanagement zu nennen.

3.1.2.2 Die Teilnehmer der Umfrage

Insgesamt haben an der Umfrage 14 Dienstbetreiber der wesentlichsten Dienste des Leibniz-Rechenzentrums aus den drei operativen der vier verschiedenen Abteilungen und mehreren Gruppen teilgenommen. Die vierte Abteilung hat verwaltende Aufgaben und insofern keinen Bezug zu einem Schwachstellenmanagement. Da sich die Behandlung von Schwachstellen durch Mitarbeiter für denselben Dienst oft zum Großteil überschneiden, decken einzelne bearbeitete Fragebögen teilweise die Tätigkeit von mehreren Verantwortlichen eines Dienstes ab. Die für die Umfrage relevanten Dienste wurden anhand ihrer Bedeutsamkeit zur Unterstützung der Tätigkeiten und Aufgaben der Nutzer im MWN ausgewählt.

Die Dienste innerhalb des MWN, die durch die bearbeiteten Fragebögen der Teilnehmer abgedeckt werden, sind unter anderem Web-Dienste, der Betrieb der Datenbanksysteme, der FTP- und Lizenzserver, IT-Bibliotheksdienste, der Betrieb der Linux-Server und Gateways sowie der Server des Identity Managements, diverser Netzkomponenten wie Router und den Server Load Balancern, den Backup- und Archivsystemen sowie des Patchmanagements der Desktoprechner innerhalb des LRZ.

3.1.2.3 Organisatorischer Rahmen der Schwachstellenbehandlung

Die Auswertung der Umfrage ergab, dass sich acht der 14 Teilnehmer regelmäßig mit der Behandlung von Schwachstellen beschäftigen. Sechs der Teilnehmer kümmern sich dagegen spontan bzw. bei Bedarf darum.

Dabei ist die Häufigkeit, mit der sie sich um mindestens eine der Tätigkeiten des Schwachstellenmanagements befassen, sehr unterschiedlich: Viele der Teilnehmer befassen sich mehrmals im Monat, zwei der Teilnehmer sogar täglich damit; die wenigsten – lediglich gut ein Fünftel – ein Mal oder weniger im Monat. Auch die Suche und Behebung von Schwachstellen als Reaktion auf einen Sicherheitsvorfall wird von mehr als 70% der Dienstbetreiber vorgenommen, außerdem gaben 35% der Teilnehmer an, sich vor der Inbetriebnahme neuer Geräte und Software mit den dafür relevanten Schwachstellen auseinanderzusetzen und ein weiteres Fünftel, dies auch bei größeren Konfigurationsänderungen zu tun.

Bei der Betrachtung des Umfangs der Computersysteme, auf denen die Teilnehmer nach Schwachstellen suchen und diese beseitigen, gaben alle an, dass sie die von ihnen betreuten bzw. administrierten Computersysteme, insbesondere Server, berücksichtigen. Die Gewährleistung der Aktualität der *Windows*-, *openSUSE*- sowie *MAC*-Arbeitsplatzrechner werden durch ein jeweiliges Patchmanagementsystem sichergestellt. Dennoch gaben weitere 35% der Teilnehmer an, dass sie sich zusätzlich auf ihren Arbeitsplatzrechnern um die Behebung von Schwachstellen kümmern. Es werden jedoch auch Rechner außerhalb des LRZ bei der Schwachstellensuche miteinbezogen, zumal zwei der Teilnehmer je nach Bedarf das komplette MWN, insbesondere bei besonders kritischen Schwachstellen wie *Poodle* oder dem *Heartbleed Bug*, scannen.

Nur in wenigen Fällen wird die Suche und Behebung von Schwachstellen für Software und Computersysteme allein von Einzelpersonen durchgeführt. Lediglich drei der Dienstbetreiber gaben an, dass dabei ausschließlich sie selbst beteiligt sind. In der Regel gibt es mehrere Personen, die sich damit befassen und sich bezüglich der Aufgabenverteilung beziehungsweise der Beeinträchtigung der Systeme absprechen. Auch gaben fünf Teilnehmer an,

dass sie betroffene Personen, insbesondere Kunden, bezüglich möglicher Beeinträchtigungen informieren. Die Behebung der Schwachstelle hängt in der Regel davon ab, wer für das System verantwortlich ist und wer durch eine Störungen betroffen sein kann: So gaben 57% der Teilnehmer an, dass sie bei der Detektion von Schwachstellen in Software und auf Assets, die zur Behebung der Schwachstelle zuständigen Personen informieren, ansonsten selbst die Behandlung übernehmen.

3.1.2.4 Quellen für Schwachstelleninformation und Bewertung von Schwachstellen

Ein zentrales Thema in Bezug auf Schwachstellenmanagement, ist auch die Wahl der Bezugsquellen für Schwachstelleninformationen. Eine der umfangreichsten Quellen, die *Common Vulnerabilities and Exposures* Auflistung, ist in Abschnitt 2.3.3 erläutert und speist viele frei nutzbare Schwachstellen-Datenbanken. Aufgrund der geringeren Nutzerfreundlichkeit durch fehlenden Funktionsumfang ist sie für den direkten Gebrauch durch den Endnutzer jedoch ungeeignet.

Die meistgenutzten Quellen der LRZ-Mitarbeiter sind in Abbildung 3.1 illustriert. Demnach bezieht der größte Teil der Dienstbetreiber (knapp 60%) im LRZ Schwachstelleninformationen aus CERT-Meldungen. Innerhalb des LRZ wird die Aufarbeitung von Schwachstelleninformationen manuell und auf freiwilliger Basis durch einen Mitarbeiter angeboten und regelmäßig an Interessierte per E-Mail geschickt. Diese Schwachstelleninformationen stammen ursprünglich aus dem *DFN-CERT Portal* [DFN15] und werden von acht der Befragten genutzt. Die Dienstbetreiber suchen jedoch auch selbst zusätzlich auf dem Portal oder informieren sich über die Sicherheitsmeldungen des *US-CERT* [Uni15a].

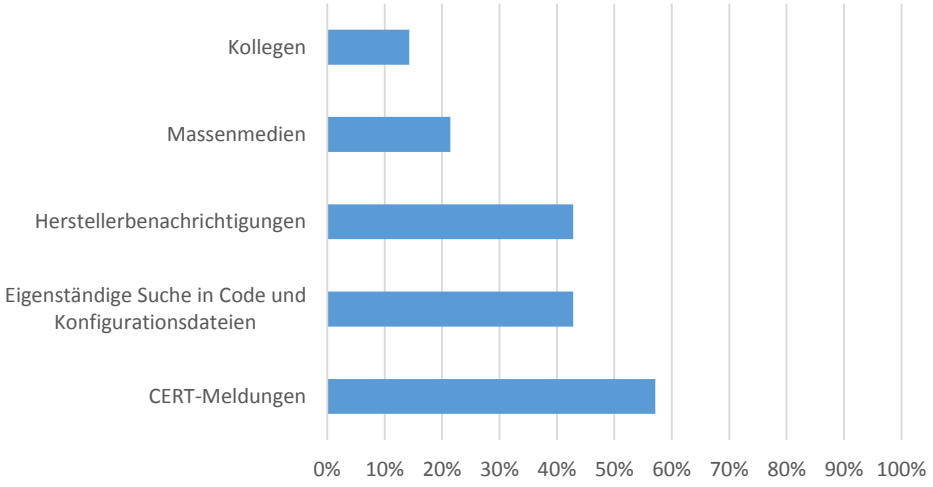


Abbildung 3.1: Meistgenutzte Informationsquellen für Schwachstellen

Eine weitere wichtige Quelle für Schwachstellen sind Herstellerbenachrichtigungen. Diese unterscheiden sich nicht grundlegend von Quellen mit allgemeinerem Bezug, bieten jedoch den Vorteil, dass die Benachrichtigungen nicht mehr, oder nur in geringerem Umfang gefiltert werden müssen. Je nach Produkt kann es jedoch sein, dass die vom Hersteller gemeldeten Schwachstellen nicht in öffentlich zugänglichen Quellen angeboten werden, da deren Umfang

beispielsweise die Art der Schwachstelle oder des Produktes bzw. der Hersteller nicht umfassen. Genauso wichtig wird auch die Suche nach Schwachstellen in Konfigurationsdateien oder Quelltext angesehen.

Außerdem nicht zu vernächlässigen sind Massenmedien als Quelle für Schwachstelleninformationen, insbesondere Online-Zeitungen, über welche sich rund jeder fünfte Dienstbetreiber informiert. Des Weiteren gaben 14% der Teilnehmer an, dass ihnen Informationen von Kollegen dabei helfen, über Schwachstellen informiert zu bleiben. Weitere Quellen, die von einzelnen Teilnehmern genannt wurden, sind Hinweise durch Dienstnutzer, diverse Mailinglisten wie *Full Disclosure* oder auch von Softwaredistributoren (z.B. *SUSE-Linux*).

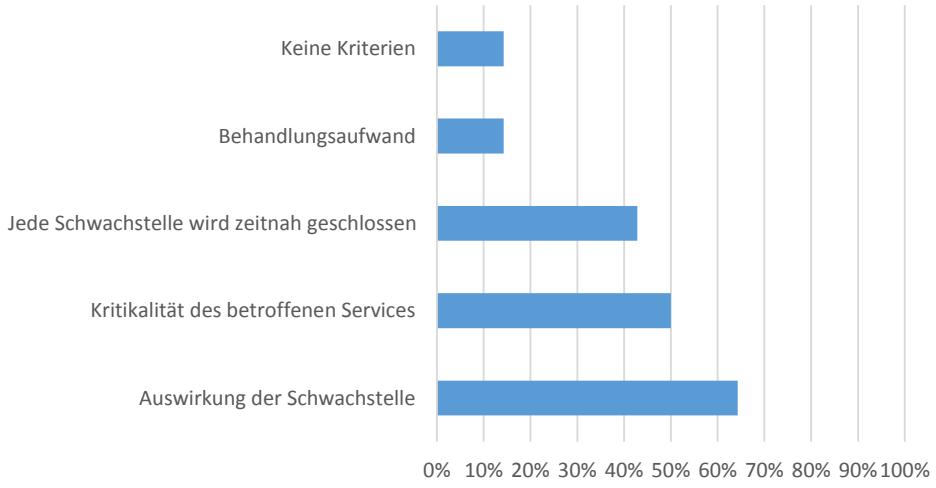


Abbildung 3.2: Kriterien zur Beseitigung von Schwachstellen

Nach dem Erfassen des Zustands über aktuelle Schwachstellen kommt üblicherweise der Schritt der Einstufung und der Bewertung hinsichtlich ihrer Relevanz, Priorität und dem daraus resultierenden Handlungsbedarf. Am LRZ gibt es für diesen Zweck keine standardisierte Vorgehensweise, sondern die Einschätzung hängt von der Beurteilung und den Kriterien der einzelnen Mitarbeiter ab. Eine Zusammenfassung der von den Teilnehmern zur Schwachstellenbewertung herangezogenen Kriterien zeigt Abbildung 3.2: Mit über 60% der Nutzung durch die Dienstbetreiber ist das am meisten herangezogene Kriterium die Kritikalität der jeweiligen Schwachstelle. Hier unterstützend kann auch der CVSS (siehe Abschnitt 2.3.4) oder andere Bewertungssysteme wirken. Auch die Kritikalität der betroffenen IT-Dienste wird von der Hälfte der Teilnehmer in Betracht gezogen. Mehr als 40% gaben an, dass sie Schwachstellen möglichst zeitnah ohne weitere Bewertung schließen. Dabei ist zu beachten, dass auch hier die Auswahl mehrerer Kriterien möglich war und beispielsweise das Verhältnis von Kritikalität der Schwachstelle zu Behandlungsaufwand auch zur Einschätzung herangezogen wird.

Lediglich jeweils 14% der Dienstbetreiber berücksichtigen den Aufwand zur Beseitigung von Schwachstellen oder hatten keine Kriterien bezüglich der Einschätzung der Notwendigkeit einer Behandlung bzw. zur Priorisierung von Schwachstellen.

3.1.2.5 Detektion von Schwachstellen auf Computersystemen

Ein grundlegender Schritt, der darüber entscheiden kann, ob ein Schwachstellenmanagement Erfolg hat oder nicht, ist die Nutzung eines Asset-Inventars, in dem alle im Rahmen des Schwachstellenmanagements betrachteten Computersysteme aufgelistet sind und notwendige Informationen hält.

Von den insgesamt 14 Umfrageteilnehmern gaben elf an, dass sie in irgendeiner Form ein Asset-Inventar haben und nutzen, es jedoch kein zentrales Inventar zu diesem Zweck gibt. Es existiert jedoch ein gut verwaltetes Tool, der „LRZ Monitor“ zur Inventarisierung der Linux-Server, das von einigen Mitarbeitern genutzt wird und viele Informationen über die Systeme bereitstellt, welche im Laufe des nächsten Absatzes umrissen werden. Weitere zwei gaben an, dass sie eine Datenbank zu diesem Zweck verwenden und sechs Teilnehmer nutzen eine Textdatei zur Inventarisierung. Weitere Mitarbeiter nutzen verschiedene Dokumentationen und eine sich noch im Aufbau befindliche *Configuration Management Database* (CMDB), in der die Betriebsmittel des LRZ aufgelistet werden. Im Rahmen des am LRZ betriebenen zentralen Patchmanagements der Arbeitsplatzrechner, gibt es einen *Windows Server Update Services*-Dienst (WSUS), dessen Verwaltungskonsole ein Asset-Inventar über die dort verwalteten Windows-Rechner führt. Drei der Umfrageteilnehmer verwenden kein Asset-Inventar.

Die Informationen, die in den Asset-Inventaren gehalten werden, sind allgemein relativ ähnlich: Ein Anteil von 65% der Befragten halten die IP-Adresse ihrer Systeme als wichtige Information, gefolgt von dem zuständigen Besitzer (engl. „owner“, im Sinne von „Verantwortlicher“) und einer Beschreibung des Computersystems mit jeweils acht Personen, die diese Auswahl getroffen haben. Jeweils knapp 30% der Umfrageteilnehmer dokumentieren außerdem noch weitere Informationen über das Betriebssystem, die Kernelversion und jeweils knapp 15% nutzen zusätzlich Zeitstempel zur Dokumentation des Zeitpunktes der letzten Detektion von Schwachstellen auf dem System, der installierten Softwarepakete und Versionen sowie die MAC-Adresse des Systems im Netz. Weitere Informationen, die insbesondere innerhalb des „LRZ Monitor“ Verwendung finden, sind eine Auflistung der dafür zuständigen Systemadministratoren, betroffenen Serviceadministratoren, der Systemnutzer und der Netzkonfiguration mit beispielsweise einer Auflistung offener Ports.

Die Anzahl der im Schwachstellenmanagement berücksichtigten Computersysteme variiert stark. Einige Befragte behandeln Schwachstellen auf knapp einem Dutzend Rechner und andere auf mehreren hundert bis im Maximum 1200 Rechnern bzw. separat 1300 *VHosts* (im Web-Bereich), die jedoch nicht nur durch eine Person, sondern dann durch das ganze jeweilige Team administriert werden. Wie bereits im Abschnitt 3.1.2.3 erwähnt, wird auch das gesamte MWN bei Bedarf, d.h. in der Regel bei Meldung besonders kritischer Schwachstellen, mit Netzwerkscannern auf diese überprüft.

Eine Detektion von Schwachstellen auf Computersystemen kann je nach betrachteter Schwachstelle durch eine Vielzahl an Methoden erfolgen. Die gängigsten und am meisten genutzten sind der Abgleich der Software und ihrer Version mit den aus der Schwachstellenmeldung genannten betroffenen Versionen. Des Weiteren oft verwendet werden Korrelationstests, insbesondere für die (Netzwerk-)Konfiguration und der Zustand des Systems. Dazu kann beispielsweise ein Scan der offenen Ports helfen, herauszufinden, ob ein Rechner seinen darauf betriebenen Diensten entsprechend konfiguriert ist.

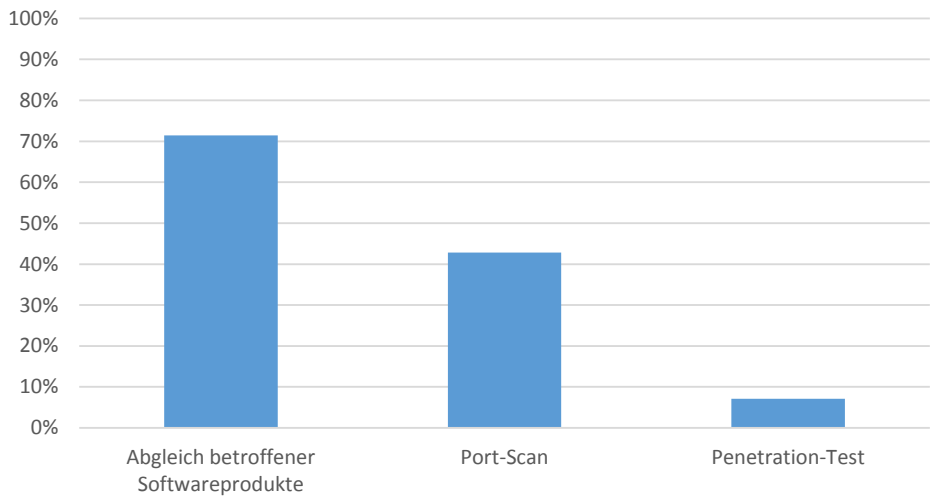


Abbildung 3.3: Methodik zur Schwachstellendetektion

Abbildung 3.3 illustriert den Anteil der zur Detektion genutzten Methoden unter den Befragten. Gut zu erkennen ist, dass die wohl zuverlässigste Methode, die Detektion anhand des Abgleichs betroffener Softwareprodukte, am meisten genutzt wird. Auch Portscans werden zur Detektion von Schwachstellen herangezogen. Am wenigsten werden Penetrationtests eingesetzt, nämlich nur durch weniger als zehn Prozent der Teilnehmer. Weitere Vorgehensweisen, die im Rahmen der Umfrage genannt wurden, sind die Prüfung der Konfigurationsdateien auf den Systemen selbst, die Untersuchung von Anwendungen (insbesondere Webanwendungen) und des dazugehörigen Quelltextes, Tests auf die Funktionalität von Sicherheitsmechanismen (hier insbesondere Firewallfunktionalität) und das Heranziehen weiterer Korrelationstests.

Um die genannten Tests durchzuführen, werden am LRZ einige Softwaretools verwendet. Obwohl es jedoch bereits einige Tools zur Unterstützung des Schwachstellenmanagement auf dem Markt gibt, sowohl kommerzielle als auch Open-Source Produkte (z.B. *OpenVAS* oder *Nessus*), benutzt keiner der Umfrageteilnehmer derartige Software.

Auch in Bezug auf die verwendeten Softwaretools zur Detektion von Schwachstellen auf Assets gibt es keine vorgeschriebenen bzw. einheitlichen Lösungen. Für die Detektion über das Netz hat sich der Netzwerkscanner *nmap* [Gor15b] etabliert. Durch die Möglichkeit, seine Funktionalität mit Hilfe von *Lua*-Skripten zu erweitern und je nach Charakteristik einer Schwachstelle zu suchen, wird er auch eingesetzt, um gezielt die Ausnutzbarkeit von Schwachstellen zu testen. Auf der offiziellen Website [Gor15c] gibt es eine Sammlung an Skripten, die zu diesem Zweck verwendet werden können, darunter beispielsweise auch Tests auf den Heartbleed Bug [Cod14]. Ein weiterer Einsatzzweck im LRZ ist ein Plausibilitätstest der IT-Dienste durch eine Prüfung auf offene Ports. In Verbindung mit *nmap*, aber auch anderen Netzwerkscannern, wird das dafür am LRZ konzipierte Softwaretool *Dr. Portscan* [Fel13] eingesetzt. Dieser bietet die Möglichkeit, den Ist- bzw. Soll-Zustand zu überprüfen, indem Veränderungen in Ergebnissen von Portscans von verschiedenen Zeitpunkten ausgewertet werden.

Aus Systemsicht gibt es am LRZ eine große Vielfalt an Software, welche die Systeme überwacht und bei der Feststellung möglicher Schwachstellen hilft. Darunter sind einige Tools, die – vor allem auf Linux-Systemen – den aktuellen Systemzustand erfassen und melden. Beispielsweise helfen *netstat* und *ss* dabei, die Netzwerkverbindungen eines Rechners zu überwachen und Statistiken zu liefern. Des Weiteren werden mehrmals täglich von den Mitarbeitern selbst-entwickelte Skripte ausgeführt, die die Konfiguration der Systeme überprüfen. Auch teilweise im Einsatz befindet sich der *Simple Event Correlator* [Vaa15], eine leichtgewichtige Lösung zur Analyse von und Reaktion auf Ereignisse in Log-Dateien.

Ebenso ist *Nagios*, ein Tool zur Überwachung (engl. „Monitoring“) der IT-Infrastruktur und IT-Diensten, stellenweise im Einsatz.

3.1.2.6 Behebung von Schwachstellen

Die Maßnahmen, die ein Dienstbetreiber am LRZ zur Behebung von Schwachstellen treffen kann, bleiben in erster Linie ihm selbst überlassen. Das weitere Vorgehen nach der Detektion einer Schwachstelle hängt von vielen Faktoren ab, wie beispielsweise ihrer Kritikalität, der vorhandenen Patchmöglichkeiten oder auch vom betroffenen Computersystem. Eine Auflistung der am häufigsten getroffenen Maßnahmen, die von den Umfrageteilnehmern in Betracht gezogen werden, ist in Abbildung 3.4 illustriert.

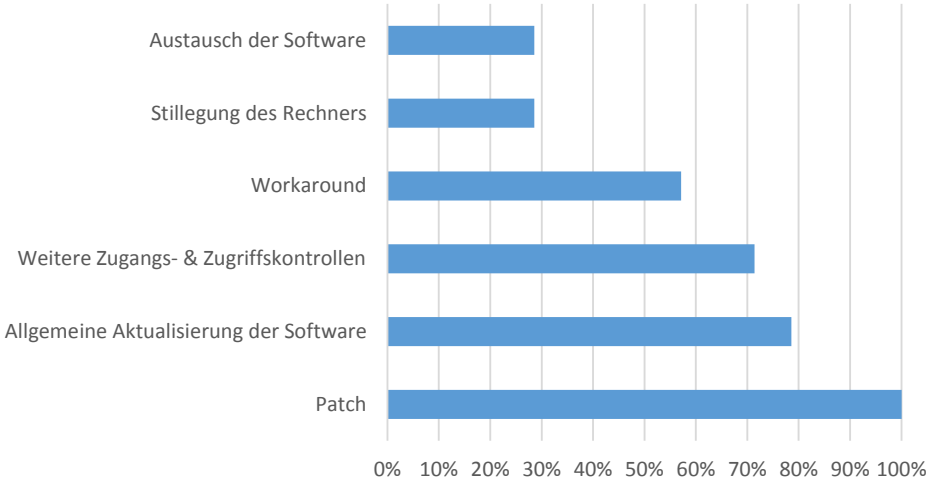


Abbildung 3.4: Maßnahmen zur Behebung von Schwachstellen im LRZ

Gut zu erkennen ist, dass alle Teilnehmer die Installation eines Patches vornehmen oder vornehmen würden, sofern einer zur Verfügung steht. Ähnlich viele Teilnehmern erwägen eine komplette Aktualisierung der betroffenen Software, auch wenn das teilweise zu Kompatibilitätsproblemen mit anderen Softwarepaketen führen kann, die die betroffenen Softwareprodukte verwenden.

Hingegen gut 70% der Dienstbetreiber würden weitere Zugangs- und Zugriffskontrollen einrichten. Gemeint ist beispielsweise die Einrichtung einer ACL (*Access Control List*) zur Einschränkung des Zugriffs oder auch von Authentifizierungsmechanismen.

Knapp 60% der Teilnehmern würden die Anwendung eines Workarounds, also einer inoffiziellen und oftmals nicht dauerhaften Lösung zur Abschwächung in Betracht ziehen.

Die Stillegung des Rechners wird von den meisten Befragten dagegen nicht als Mittel zur Beseitigung bzw. Abschwächung von Schwachstellen gesehen, genauso der Austausch der betroffenen Software. Da in Hochschulnetzen oft spezielle und selbstentwickelte Software zum Einsatz kommt (vgl. Abschnitt 2.5), ist der Austausch oft auch nicht möglich, da ein Ersatz bzw. Konkurrenzprodukte auf dem Markt nicht verfügbar sind.

Weitere Maßnahmen, die vereinzelt in Betracht gezogen werden, sind der Umzug der von Schwachstellen betroffenen Systeme in einen gesicherten Netzabschnitt bzw. der Umzug hinter Netzfilter und insbesondere bei betroffenen Webanwendungen die allgemeine Entziehung der Schreibberechtigungen.

3.1.2.7 Die Meldung detektierter Schwachstellen an Betroffene

Die Art und Weise der Meldung von Schwachstellen kann ausschlaggebend dafür sein, ob eine Schwachstelle tatsächlich behandelt wird oder offen bleibt – insbesondere, wenn die Behandlung nicht im Zuständigkeitsbereich des LRZ liegt und es so keinen Einfluss darauf hat.

Lediglich zwei der Dienstbetreiber geben an, dass sie einen Schwachstellenscan ankündigen, bevor sie ihn durchführen. Die Antworten bezogen sich auf Scans über das Netzwerk (Portscans und Schwachstellenscans mittels *nmap*) und haben das gesamte Münchner Wissenschaftsnetz betroffen. Fünf Befragte geben an, die Detektion von Schwachstellen nicht anzukündigen. Der Grund dafür liegt vor allem darin, dass derartige Eingriffe und Maßnahmen teilweise bereits in Verträgen mit den jeweiligen Kunden geregelt sind. Die restlichen Befragten enthielten sich bei der Beantwortung. Die Ursache dafür ist jedoch, dass sie keine Kundensysteme, sondern nur zum Rechenzentrum gehörige Computersysteme betreuen.

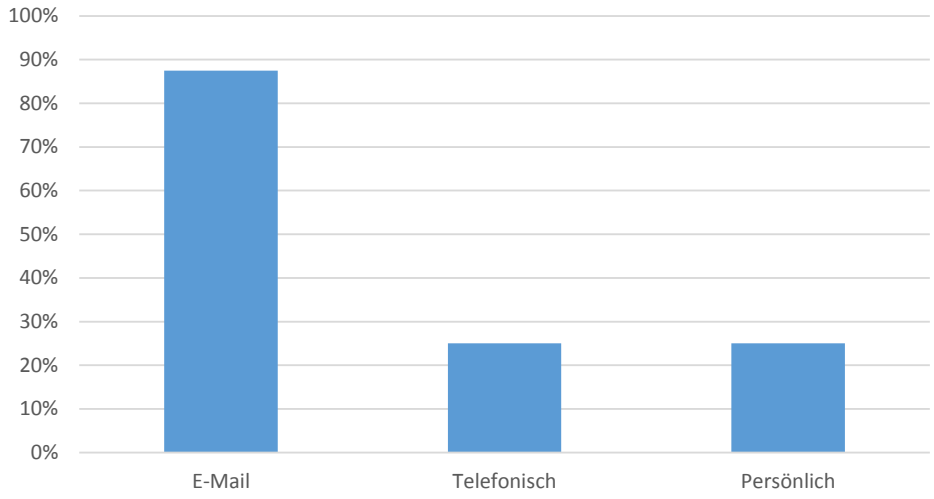


Abbildung 3.5: Meldewege nach der Detektion von Schwachstellen auf Kunden- und Nutzersystemen

Der bevorzugte Meldeweg der Teilnehmer war mit knapp 90% Nutzungshäufigkeit am höchsten: Die Benachrichtigung per E-Mail. Jeweils 25% der Teilnehmer benachrichtigen Betroffene des Weiteren telefonisch bzw. persönlich.

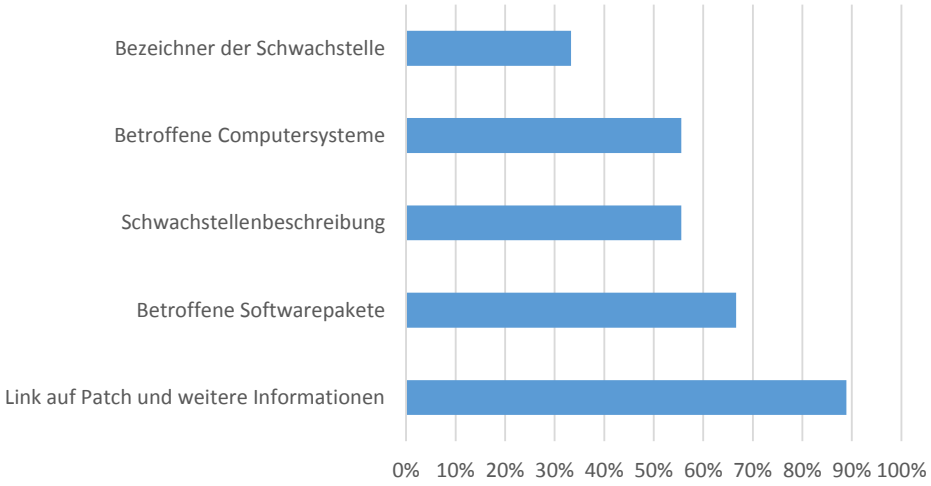


Abbildung 3.6: Meldung von Schwachstellen-Informationen

Ein weiterer bedeutsamer Faktor bei der Unterstützung der Behebung von Schwachstellen, sind die Art sowie die in der Meldung zur Verfügung gestellten Informationen. Ein Überblick der von den Befragten bereitgestellten Informationen ist in Abbildung 3.6 zu sehen. Am häufigsten wird ein Verweis zu Patches oder weiteren Informationen zur Behebung der Schwachstelle kommuniziert. Als weitere wichtige Information sehen die Umfrageteilnehmer die konkreten, von der Schwachstelle betroffenen Softwarepakete an sowie den Bezeichner der betroffenen Computersysteme und die Beschreibung der Schwachstelle.

Als weniger wichtig wird der Name bzw. Bezeichner der Schwachstelle selbst angesehen, den nur knapp über 30% der Befragten an die Betroffenen kommunizieren. Weitere vereinzelt genannte Informationen waren Auswirkungen auf den jeweiligen Dienst, den die von der Schwachstelle betroffene Infrastruktur haben kann sowie konkrete Nennung von Methoden zur Schwachstellenbeseitigung, insbesondere auf Linux-Systemen – falls möglich – durch genaue Beschreibung der auszuführenden *UNIX*-Befehle.

Bei der Befragung der Unterstützung der Kunden und Nutzer bei der Behandlung von Schwachstellen gab lediglich einer an, dass eine Unterstützung aus zeitlichen Gründen nicht möglich ist. Ein Anteil von 90% der bei dieser Aktivität beteiligten Dienstbetreiber leisten bei der Behebung Unterstützung, wobei mit 70% der Antworten die Unterstützung per E-Mail am gängigsten ist, gefolgt von einer telefonischen Unterstützung mit 50%. Jeweils 30% der Teilnehmer gab an, dass sie zum einen auch über die *Secure Shell* (SSH), als auch zum anderen direkt persönlich beim Kunden anwesend bei der Behebung unterstützen.

3.1.2.8 Wünsche und Anforderungen der Umfrageteilnehmer

Im letzten Abschnitt der Umfrage, wurde den Teilnehmern die Möglichkeit gegeben, Anforderungen und Wünsche an ein zukünftiges Schwachstellenmanagement zu äußern. Diese Möglichkeit wurde von fünf der Teilnehmer genutzt.

Aus der Abteilung „Benutzernahe Dienste und Systeme“ kamen zwei Anforderungen bzw. Wünsche, die im Rahmen eines Schwachstellenmanagements berücksichtigt werden sollten. Zum einen die Bereitstellung einer zentralen und einheitlichen Asset-Datenbank, welche die Computersysteme verwalten soll, die bei der Behandlung von Schwachstellen miteinbezogen werden. Zum anderen wurde von drei der Teilnehmer aus dieser Abteilung der Wunsch nach dem Aufbau einer Gruppe am LRZ für Penetrationtests geäußert, welche entsprechend interne sowie von Kunden genutzte Computersysteme professionell nach Schwachstellen überprüfen können. Ein ähnlicher Wunsch wurde aus der Gruppe „Hochleistungssysteme“ geäußert: Generell ein Team zur Suche nach Schwachstellen und technische Unterstützung bei der Beseitigung.

Weitere Anforderungen wurden aus der Abteilung „Kommunikationsnetze“ gestellt, wonach die Zuständigkeiten, Rollen, sowie Computersysteme mit Fähigkeiten zur Detektion von Schwachstellen klar definiert sein sollen. Des Weiteren sollen die organisatorischen Abläufe und Verfahren festgelegt werden.

3.2 Erläuterung der Anforderungen an ein Schwachstellenmanagement

In den folgenden Abschnitten werden die Anforderungen an ein Schwachstellenmanagement in Hochschulnetzen festgelegt. Diese sind, aus Gründen der Übersicht, aber vor allem der besseren Anwendbarkeit, nach den verschiedenen Aktivitäten des Schwachstellenmanagements (vgl. Abschnitt 2.2) gegliedert und als Zusammenfassung in tabellarischer Form am Ende des Kapitels aufgelistet. Daher ergibt sich jeweils eine eigene Kategorie für allgemeine Anforderungen an ein Schwachstellenmanagement, Anforderungen an die Identifikation, Klassifikation, Beseitigung und Prävention von Schwachstellen sowie der in diesem Rahmen notwendigen Berichterstattung.

Neben der Unterscheidung der Kategorien erfolgt die grobe Unterteilung in funktionale und nicht-funktionale Anforderungen: Funktionale Anforderungen beschreiben Funktionalitäten, die in einem Schwachstellenmanagement zur Verfügung stehen müssen bzw. sollten. Demgegenüber bezeichnen nicht-funktionale Anforderungen qualitative Anforderungen an ein Schwachstellenmanagement.

Des Weiteren erfolgt eine Unterscheidung zwischen technischen und organisatorischen Anforderungen, die sich in der Struktur des folgenden Abschnittes widerspiegelt, jedoch nicht im Namensschema der Identifikatoren der Anforderungen auftaucht, um diese nicht unübersichtlicher zu gestalten. Durch diese Kategorisierung ist eine sinnvolle Gliederung des in Kapitel 5 erstellten Konzepts möglich sowie leichter verständlich.

Eine Anforderung ist ein Tripel, bestehend aus einem `Identifikator`, der `Anforderungsbeschreibung` sowie einer `Gewichtung` in Bezug auf die Notwendigkeit der Erfüllung der jeweiligen Anforderung für die Effektivität und Effizienz des Schwachstellenmanagements.

Der Identifikator der Anforderung dient vor allem im Verlauf der Arbeit als Möglichkeit für eine übersichtliche Kurzreferenzierung, damit der Abgleich zwischen Ist- und Soll- Zustand bei Konzepten eines Schwachstellenmanagements in Hochschulnetzen einfach durchgeführt werden kann. Das Namensschema der Anforderungsidentifikatoren ist wie folgt:

$$< FUNK >< TYP >< NUM > \tag{3.1}$$

Die Trennung der funktionalen bzw. nicht-funktionalen Anforderungen werden durch das `<FUNK>` Segment in der Anforderungsbezeichnung ausgedrückt. Entsprechend gibt es zwei verschiedene Werte, die `<FUNK>` annehmen kann:

F – Eine funktionale Anforderung

N – Eine nicht-funktionale Anforderung

Das `<TYP>` Segment beschreibt die jeweilige Kategorie, zu der die Anforderung gehört. So soll bei Benennung einer Anforderung im weiteren Verlauf der Arbeit schnell erkennbar sein, welcher Aktivität die Anforderung angehört. Die Werte von `<TYP>` sind:

A – Eine allgemeine Anforderung an ein Schwachstellenmanagement

I – Eine Anforderung an die Aktivität der Identifikation von Schwachstellen

K – Eine Anforderung, welche die Klassifikation von Schwachstellen betrifft

B – Eine Anforderung an die Beseitigung und Abschwächung von Schwachstellen

P – Eine Anforderung an die Prävention von Schwachstellen

Allgemeine Anforderungen sind aktivitätsunabhängige Anforderungen, deren Umsetzung das gesamte Schwachstellenmanagement betreffen. Darunter fallen insbesondere management-spezifische Anforderungen sowie Anforderungen an die Berichterstattung, welche sich generell über alle Aktivitäten im Schwachstellenmanagement erstrecken.

Das `<NUM>` Segment ist eine bei „1“ beginnende und mit jeder weiteren Anforderung innerhalb der jeweiligen Aktivität fortlaufende Nummer, welche eine eindeutige Bezeichnung garantiert. Das heißt, dass `<NUM>` für Anforderungen in jeder Aktivität wieder mit 1 beginnt.

Beispielsweise bezeichnet der Identifikator **FK2** die zweite Anforderung an die Umsetzung der Aktivität der Klassifikation im Schwachstellenmanagement in Hochschulnetzen. Dem voranstehendem „F“ ist zu entnehmen, dass es sich um eine funktionale Anforderung handelt.

Die Anforderungsbeschreibung dient dem leichteren Verständnis der Anforderung.

Da es Anforderungen gibt, von deren unbedingter Umsetzung der Erfolg des Schwachstellenmanagements abhängig ist, und wiederum andere mehr als empfohlene, aber nicht zwingend zu erfüllende Aspekte gelten, werden die einzelnen Anforderungen mit einer Gewichtung versehen, welche folgende Werte annehmen kann:

erforderlich – Die Erfüllung von Anforderungen mit der Gewichtung *erforderlich* ist zwingend notwendig für ein effektives und effizientes Schwachstellenmanagement.

empfohlen – Die Erfüllung von Anforderungen mit der Gewichtung *empfohlen* ist hilfreich und in der Regel effizienzfördernd, betrifft grundsätzlich jedoch nicht die Wirksamkeit des Schwachstellenmanagements.

Eine ausführliche Beschreibung der jeweiligen Anforderungen wird in den folgenden Abschnitten vorgenommen.

3.3 Analyse der Anforderungen

Die Anforderungen an ein Schwachstellenmanagement in Hochschulnetzen resultieren aus mehreren Quellen: Zum einen liefert die Definition von Schwachstellenmanagement Aspekte, deren Erfüllung obligatorisch sind, zum anderen muss es Anforderungen entsprechen, die aus dem Managementhintergrund abgeleitet werden, um die durch das Management angestrebten Ziele zu erreichen (siehe Abschnitt 2.2). Weitere Anforderungen ergeben sich aus den Charakteristika (vgl. Abschnitt 2.4), Herausforderungen (vgl. Abschnitt 2.5) und in erster Linie den betrachteten Szenarien – dem Münchner Wissenschaftsnetz sowie aktuellen Maßnahmen und Abläufe, die durch das LRZ zur Beseitigung von Schwachstellen bereits getroffen werden.

3.3.1 Technische Anforderungen

In diesem Abschnitt werden die technischen Anforderungen an ein Schwachstellenmanagement in Hochschulnetzen erklärt.

3.3.1.1 Allgemeine Anforderungen

NA1 `Existenz eines Asset-Inventars`
Eine der grundlegendsten allgemeinen Anforderungen ergibt sich zum einen aus der Größe von Hochschulnetzen, die im Szenario des MWN thematisiert wurde: Die Existenz eines für das Schwachstellenmanagement nutzbaren Asset-Inventars. In der Regel finden sich in einem Hochschulnetz mehrere tausend bis zehntausend Computersysteme, die unter Aspekten der Netzsicherheit im Schwachstellenmanagement berücksichtigt werden sollten, insbesondere ist daher die Möglichkeit der Dokumentation von den im Anwendungsbereich abgedeckten Assets unbedingt notwendig.

NA2 `Existenz und Zentralisierung einer Schwachstellen-Dokumentation`
Die im Szenario beschriebene Größe des Netzes – in diesem Fall durch die Anzahl der Nutzer gegeben – sowie Unterschiedlichkeit der Rollen und Tätigkeiten der Personen im Hochschulnetz, sowie das teilweise geringe Fachwissen der Netznutzer führt zu der Notwendigkeit der Existenz einer zentralen Schwachstellen-Dokumentation. Auch hier – ähnlich dem Zweck des Asset-Inventars – ist das Ziel, die darin enthaltenen Informationen netzweit zu standardisieren und zentral zugänglich zu machen. Diese Anforderung ist insofern als allgemein und aktivitätsübergreifend anzusehen, da sie Abläufe und Daten aus allen Aktivitäten beeinflusst.

NA3 `Zentralisierung des Asset-Inventars`
Um zum einen ein einheitliches Datenformat und ausreichenden Informationsgehalt garantieren zu können sowie für einen organisatorisch einfacheren Umgang, sollte das

Asset-Inventar zentral verwaltet und angeboten werden, da ansonsten die Gefahr besteht, dass Systeme übersehen werden bzw. schlecht dokumentiert sind. Diese Anforderung wird durch die im LRZ durchgeführte Umfrage verstärkt: Hier ist noch kein zentrales Asset-Inventar vorhanden, sondern die Dienstbetreiber pflegen in der Regel eigene Lösungen wie Textdateien oder dienstspezifische Datenbanken. Viele äußerten jedoch den Wunsch nach einer zentralen, umfassenderen Lösung. Da die Zentralisierung des Asset-Inventars unter Umständen nicht immer umsetzbar oder von Kunden gewünscht ist sowie ein Fehlen der Erfüllung dieser Anforderung nicht zwangsläufig die Wirksamkeit des Schwachstellenmanagements beeinflusst, ist sie als empfohlen, jedoch nicht als erforderlich anzusehen. Genauso kann ein Asset-Inventar dezentral vom jeweiligen Kunden betrieben werden.

FA4 `Funktionen zum Aufbau und Pflege des Asset-Inventars`
Damit das Asset-Inventar allgemein im Schwachstellenmanagement akzeptiert wird und einen Nutzen bringt, muss es den involvierten Rollen ein gewisses Angebot an Funktionalität zur Verfügung stellen. Dazu gehört das Eintragen neuer Computersysteme in einem einheitlichen Datenformat, eine Suchfunktion nach Assets anhand eines Identifikators oder auch anhand bestimmter Attribute der Assets (z.B. der Rolle des Assets) und enstprechend auch eine Möglichkeit der Dateneinsicht. Zum Mindestfunktionsumfang gehört auch das Entfernen von Assets, falls diese beispielsweise nicht mehr betrieben werden, sowie die Möglichkeit der Änderung der Attribute.

FA5 `Zugriff auf Informationen und Funktionen über eine API`
Da sich die Planung und Umsetzung von Dienstleistungen im Hochschulumfeld oft an Standards zum IT-Service-Management orientiert und entsprechende Systeme zum *Incident & Service Request Management* oder dem *Change Management* bereits implementiert sind, sollten Informationen aus dem Schwachstellenmanagement über eine Programmierschnittstelle (API, „Application Programming Interface“) für den automatisierten Zugriff bereitgestellt werden.

Da solche Lösungen jedoch nicht zwingend in jeder Hochschule und jedem Institut existieren und eine API allgemein nicht zwangsläufig erfolgsentscheidend ist, ist die Umsetzung nicht notwendigerweise erforderlich.

FA6 `Automatischer Erhalt der Aktualität des Asset-Inventars`
Das Asset-Inventar muss auch stets aktuell gehalten werden können, damit die Aktivitäten des Schwachstellenmanagements effektiv durchgeführt werden. Da das Hochschulumfeld, wie in Abschnitt 3.1.1 beschrieben, eine hochdynamische Umgebung ist, kommt es sehr häufig vor, dass sich Eigenschaften von Geräten ändern oder neue Assets in das Netz integriert oder auch wieder entfernt werden. Eine dieser Eigenschaften ist beispielsweise die IP-Adresse von Systemen, die diese dynamisch durch DHCP zugewiesen bekommen.

NA7 `Automatisierung der Aktivitäten`
Die große Zahl der Nutzer bzw. das Verhältnis der Nutzerzahl zu dem im Netz Komponenten-betreibendem Personal im Hochschulnetz sowie die allgemein begrenzten finanziellen und personellen Ressourcen im Hochschulumfeld führen zu einem weiteren fundamentalen Aspekt einer effektiven und insbesondere effizienten Umsetzung des Schwachstellenmanagements. In der Regel kommen wenige Dutzend Mitarbeiter im

Rechenzentrum auf nicht selten mehrere tausend Nutzer im Netz. Eine komplette manuelle Durchführung oder Unterstützung bei der Identifikation, Klassifikation, Beseitigung und Abschwächung sowie der Prävention ist daher in der Praxis nicht möglich. Zur Bewältigung dieser Tätigkeiten ist es daher notwendig, dass diese automatisiert ablaufen und der verbleibende manuelle Aufwand möglichst gering bleibt.

NA8 `Unterstützung manueller Tätigkeiten`
Dennoch ist es in der Regel nicht möglich, die Abläufe innerhalb der Aktivitäten komplett zu automatisieren. Insbesondere bei zentralen Punkten im Schwachstellenmanagement, wie beispielsweise der eigentlichen Beseitigung von Schwachstellen auf Assets, wird aufgrund der hohen Vielfalt an Schwachstellentypen sowie dazu möglicher Maßnahmen zur Schwachstellenbehebung nicht komplett automatisiert möglich sein, sondern immer wieder von Entscheidungen abhängen, die durch Personen getroffen werden müssen. Da es im Hochschulumfeld auch oder vor allem Nutzer gibt, deren Tätigkeitsbereich weit fernab der Informatik, System- und Netzadministration liegt und diese entsprechend wenig Erfahrung und Fachwissen in diesem Bereich besitzen, müssen diese Entscheidungen und manuellen Vorgänge zumindest technisch unterstützt (beispielsweise durch eine Bereitstellung von Dokumentationen) werden, um die Effektivität der getroffnen Maßnahmen zu sichern.

NA9 `Plattformunabhängigkeit der Aktivitäten`
Wie aus dem Szenario des MWN sowie der Umfrage am LRZ zum Umgang mit Schwachstellen hervorging, gibt es eine große Vielfalt an Geräten sowie Betriebssystemen. Im LRZ wird die Beseitigung von Schwachstellen bereits davon unabhängig auf einer großen Bandbreite an Computer- und Betriebssystemen betrieben. Daher ist es wichtig, dass auch die im Schwachstellenmanagement umgesetzten Aktivitäten nicht nur auf bestimmte Softwareprodukte angewendet werden können, sondern vollständig plattformunabhängig möglich ist. Andernfalls würde zwangsweise eine Einschränkung des Anwendungsbereichs die Folge sein, wodurch das Schwachstellenmanagement an Effektivität verliert.

FA10 `Verbinden mit weiteren sicherheitsrelevanten Informationen`
Für eine umfangreichere Zielerreichung im Schwachstellenmanagement – der Verbesserung des Sicherheitsniveaus im Hochschulnetz – sollten Ergebnisse und Erfahrungen aus den Aktivitäten mit anderen vor allem technischen Komponenten des Informationssicherheitsmanagements verknüpft werden. Im MWN gibt es bereits viele technische Lösungen zur Sicherung des Netzes, darunter *Intrusion Detection-* und *Intrusion Prevention Systeme*, oder auch Lösungen zum Management von Informationen mit Bezug zur Informationssicherheit (SIEM, *Security Information & Event Management*). In Kombination mit Daten dieser Systeme können so kompliziertere Muster (beispielsweise Angriffsmuster in Verbindung mit der Ausnutzung von Schwachstellen) detektiert oder Abläufe im Schwachstellenmanagement unterstützt werden. Diese Anforderung hat im Allgemeinen jedoch keine Auswirkungen auf die Wirksamkeit, da das Schwachstellenmanagement auch ohne derartige Funktionalitäten durchführbar ist.

FA11 `Exportierbarkeit der Daten`
Um eine effektive Möglichkeit der Korrelation der Daten und auch die Wiederverwendbarkeit dieser Informationen zu gewährleisten, sollten die im Schwachstellenma-

nagement erlangten Informationen exportierbar sein. So können diese Daten auch in weiteren Systemen und Lösungen genutzt werden. Auch diese Funktionalität beeinflusst nicht die Effektivität des Schwachstellenmanagements an sich und ist daher als empfohlen eingestuft.

NA12 `Präsentation der Daten`
Generell hat die Umfrage in Bezug auf die Verwaltung der Assets und der Haltung der dazugehörigen Informationen gezeigt, dass Lösungen mit einer benutzerfreundlichen Darstellung der Daten bevorzugt werden. Dienstbetreiber, die bereits solche Lösungen benutzen (z.B. „LRZ-Monitor“) schätzen die Wichtigkeit einer zentralen Quelle zur Informationshaltung über teilweise mehrere hunderte Computersysteme mit einer übersichtlichen Darstellung. Dienstbetreiber ohne diese Möglichkeit äußerten jedoch teilweise den Wunsch nach einer derartigen Darstellung der Informationen. Die Umsetzung dieser Anforderung ist jedoch als Empfehlung und nicht als Notwendigkeit anzusehen, da, auch wenn die Bewältigung des Schwachstellenmanagements vor allem im großen Umfang von einer entsprechend übersichtlichen Präsentation abhängig ist, hingegen ein Schwachstellenmanagement mit Anwendungsbereich im kleinen Maßstab in der Regel auch mit einer einfachen Darstellungsform der Informationen umsetzbar bleibt.

NA13 `Mandantenfähigkeit`
Im folgenden Abschnitt wird die allgemeine organisatorische Anforderung gestellt, dass die Aktivitäten an dezentralen Stellen ausgeführt werden müssen. Aus der Notwendigkeit einer Dezentralisierung der Tätigkeiten folgt eine weitere Anforderung an die technische Lösung zur Unterstützung des Schwachstellenmanagements: Der Einsatz mandantenfähiger Zugriffsmöglichkeiten (vgl. Abschnitt 2.4.2). In Verbindung mit der erforderlichen Dezentralisierung der Tätigkeiten ist auch diese Anforderung als erforderlich zu erachten.

NA14 `Institutsübergreifende Benutzerverwaltung`
Aufgrund der Umsetzung im hochschulnetzweiten Rahmen und infolgedessen möglichen Miteinbeziehung aller Einrichtungen muss es im Schwachstellenmanagement eine technische Lösung zur Verwaltung einer Vielzahl von Benutzern aus unterschiedlichen Instituten geben. Diese beinhaltet Mechanismen zur Authentifizierung sowie Autorisierung bezüglich vordefinierten Verantwortlichkeiten.

3.3.1.2 Identifikation

FI1–4 `Funktionsumfang der Schwachstellen-Dokumentation`
Durch die zentrale Schwachstellen-Dokumentation soll insbesondere eine Kommunikation von Schwachstelleninformation ermöglicht werden, die den Sinn einer netzweiten Identifizierung erfüllt. Daraus ergeben sich beispielsweise auch weitere organisatorische Anforderungen, die in Abschnitt 3.3.2.2 erklärt werden. Die im Schwachstellenmanagement involierten Rollen müssen dafür jedoch auch eine Zugriffsmöglichkeit auf die Schwachstellen-Dokumentation haben. Diese ist rollenabhängig und umfasst im Mindesten die Eintragung und Beschreibung neuer Schwachstellen, die Suche nach Einträgen sowie deren Einsicht und die Möglichkeit, Einträge zu editieren. Insbesondere die Möglichkeit der manuellen Eintragung von Schwachstellen muss aufgrund von

häufig im Hochschulnetz genutzten speziellen und selbstentwickelten Softwareprodukten berücksichtigt werden. Die Möglichkeit des Editierens von Schwachstellen-Informationen ergibt sich aus den Charakteristika von Schwachstellen. Da im Lebenszyklus einer Schwachstelle der zur Verfügung stehende Informationsgehalt nicht immer gleich ist und typischerweise in einem gewissen Umfang weiter anwächst, muss auch die Dokumentation daran anpassbar sein.

FI5 `Automatischer Erhalt der Aktualität der Schwachstellen-Dokumentation`
Als organisatorische Anforderung `NI8` wird in Abschnitt 3.3.2.2 die Aktualität der Schwachstellen-Dokumentation gefordert. Da die Eintragung von Schwachstellen für ein derartig großes Spektrum insbesondere bei den im Hochschulumfeld stark begrenzten Ressourcen manuell nicht durchgeführt werden kann, ist eine weitestgehende Automatisierung des Erhalts der Aktualität der Schwachstellen-Dokumentation eine unbedingt notwendige Anforderung. Dazu müssen beispielsweise Fremdquellen automatisiert abgefragt werden.

NI6 `Flexible Festlegung der Quellen der Schwachstellen-Dokumentation`
Die Umfrage innerhalb des LRZ ergab, dass die Dienstbetreiber viele verschiedene Quellen zum Informieren über Schwachstellen nutzen bzw. nutzen müssen. Grund dafür sind unter anderem die vielfältigen und zahlreichen genutzten Softwareprodukte. Vorteile, die sich daraus ergeben, sind zum einen eine höhere Abdeckung an Informationen über Schwachstellen für viele Produkte und zum anderen eine höhere Abdeckung der Schwachstelleninformation pro Softwareprodukt. Insofern muss es in der Schwachstellen-Dokumentation die Möglichkeit geben, die Quellen für Schwachstelleninformation flexibel anzupassen. Auch muss die Quellenauswahl unabhängig von den unterschiedlichen Kommunikationsmöglichkeiten (z.B. als Datei aus dem Web, per E-Mail Benachrichtigung oder im persönlichen Gespräch) integrierbar sein.

FI7 `Filtern relevanter Schwachstellen`
Einige der in der Umfrage genannten Quellen für den Bezug von Informationen über Schwachstellen decken ein sehr weites Feld an Softwareprodukten ab. Da in der Regel nur eine Teilmenge dieser Softwareprodukte im Hochschulumfeld tatsächlich zum Einsatz kommt, muss es möglich sein, beim Import von Schwachstelleninformationen diese anhand geeigneter Kriterien zu filtern, da die Dokumentation irrelevanter Schwachstellen im Zweifelsfall keinen Nutzen, sondern lediglich Mehraufwand mit sich bringt. Auch kann so der Inhalt der Schwachstellen-Dokumentation in Abstimmung auf den Anwendungsbereich kontrolliert werden.

3.3.1.3 Klassifikation

FK1 `Automatisierung der Klassifikation`
Genauso wie die Automatisierung der Identifikation von Schwachstellen, ist aus gleichem Grund – der begrenzten Ressourcen im Hochschulumfeld – eine Automatisierung innerhalb der Aktivität der Klassifikation als unbedingt erforderlich für die Wirksamkeit des Schwachstellenmanagements anzusehen; insbesondere in Verbindung mit einem umfangreichen Anwendungsbereich. Zentraler Aspekt innerhalb der Klassifikation ist die Bewertung von Schwachstellen. In den meisten Fällen wird bei der Identifikation von Schwachstellen durch Fremdquellen (z.B. dem Hersteller oder der CVE bzw.

NVD) bereits eine Bewertung der Schwachstellen mit angegeben. Derartige Chancen müssen genutzt werden, um die Klassifikation von Schwachstellen im Hochschulumfeld umsetzbar zu machen.

FK2 `Manuelle Klassifikation`
Es ist allerdings nicht immer der Fall, dass eine Bewertung bei einer Schwachstellenbeschreibung in Fremdquellen mit angegeben wird. Des Weiteren besteht ein Teil der Softwarelandschaft im Hochschulnetz aus selbstentwickelten Anwendungen, die nicht selten geschäftskritische Dienste realisieren oder unterstützen und für die es üblicherweise keine externen Schwachstellen-Dokumentationen gibt. Daher muss die Bewertung auch manuell durch Personen in einer dafür vorgesehene Rolle möglich sein.

3.3.1.4 Beiseitigung und Abschwächung

FB1 `Verfahren zur Detektion von Schwachstellen`
Wie bereits im bisherigen Verlauf der Arbeit beschrieben, ist die Detektion von Schwachstellen ein wesentlicher Bestandteil innerhalb der Aktivität der Beseitigung und Abschwächung. Ohne Verfahren zur Detektion von Schwachstellen ist in der Regel die eigentliche Behebung nicht möglich. Daher muss es in der Aktivität der Beseitigung und Abschwächung innerhalb des Schwachstellenmanagements allgemein die Möglichkeit zur Nutzung und Anpassung von Verfahren für die Detektion geben.

FB2 `Automatisierung der Detektion`
Die Detektion von Schwachstellen auf Assets ist eine der Aufgaben im Schwachstellenmanagement, die eine starke Eignung zur Automatisierung aufweisen. Die eigentliche Detektion erfolgt in der Regel rein technisch und benötigt bei angemessener Umsetzung nicht notwenigerweise menschliche Unterstützung. Daher muss die Durchführung zur Schonung der Ressourcen im Hochschulumfeld sowie einer Steigerung der Effizienz unbedingt automatisiert erfolgen.

FB3 `Detektion aus Systemsicht`
Die Vorgehensweise der Detektion lässt sich grob in zwei Kategorien einteilen: Zum einen die Detektion aus Systemsicht und zum anderen die Detektion aus Netzsicht. Die Systemsicht ist mit der Sicht des jeweiligen Systemadministrators gleichzusetzen und umfasst alle Verfahren zur Schwachstellendetektion, die auch aus systemadministrativer Perspektive auffindbar sind. Die Möglichkeiten sind in der Regel sehr umfangreich und zuverlässig. Daher sollten Methodiken zur Detektion von Schwachstellen aus Systemsicht Anwendung finden. Die Umsetzung einer Detektion ist nicht obligatorisch, da die Wahl des jeweiligen Verfahrens auch abhängig vom festgelegten Anwendungsbereich des Schwachstellenmanagements ist und es weitere effektive Verfahren zur Detektion gibt.

FB4 `Detektion aus Netzsicht`
Auf der anderen Seite steht die Detektion von Schwachstellen aus Netzsicht. Dazu gehören üblicherweise Scans und Penetrationstests, die über das Netz erfolgen. Entsprechend sind derartige Methoden als detektierende Methoden aus Sicht des Angreifers zu sehen. Der Vorteil, der sich daraus ergibt, ist die Abdeckung der Möglichkeiten

zur Detektion von Schwachstellen aus Angreifersicht. Daher sollte in einem Schwachstellenmanagement in Hochschulnetzen eine solche Sicht implementiert werden. Genau wie bei der Anforderung einer Detektion aus Systemsicht, ist auch die Detektion aus Netzsicht nicht obligatorisch. Entscheidend ist lediglich, dass es eine zuverlässige Möglichkeit zur Schwachstellendetektion gibt.

FB5 `Unterstützung bei der Behebung durch Computersysteme`
Die Anforderung der Unterstützung bei der Behebung ist ein Spezialfall der im Abschnitt 3.3.1.1 bereits genannten `Unterstützung manueller Tätigkeiten`. Wie bereits erwähnt, ist insbesondere die Behebung von Schwachstellen üblicherweise nicht in vollem Umfang automatisiert möglich. Insbesondere Personen mit geringem Fachwissen müssen daher bei Entscheidungen der Maßnahmenergreifung bei der Behebung von Schwachstellen unterstützt werden.

3.3.2 Organisatorische Anforderungen

In diesem Abschnitt werden die organisatorischen Anforderungen an ein Schwachstellenmanagement aufgelistet und erläutert. Die Anforderungen sind genau wie bei der Auflistung der technischen Anforderungen abhängig von ihrer Zugehörigkeit zu einer Aktivität und analog zum Namensschema zu einer Anforderungskategorie gruppiert.

3.3.2.1 Allgemeine Anforderungen

Allgemeine organisatorische Anforderungen an ein Schwachstellenmanagement in Hochschulnetzen beziehen sich vor allem auf Anforderungen, die einen wirksamen sowie effizienten Umgang mit Schwachstellen ermöglichen sollen. Viele davon resultieren weniger aus dem Szenario des MWN, sondern vielmehr aus Anforderungen aus dem IT-Service-Management sowie aus praktischen Erfahrungen im Umgang mit Schwachstellen am LRZ.

NA15 `Flexibilität des Anwendungsbereichs`
Eine der Grundvoraussetzungen für jegliches Management ist ein klar festgelegter Anwendungsbereich, in welchem insbesondere die Aktivitäten eines Managementprozesses sowie das Ausmaß der Umsetzung innerhalb von Organisationsbereichen festgelegt wird. Da, wie bereits in Abschnitt 2.4 erwähnt, die Struktur und Organisation von Hochschulnetzen allgemein nicht standardisiert ist und diese sich stark voneinander unterscheiden können, kann auch der Anwendungsbereich nicht allgemeingültig festgelegt sein. Selbst innerhalb eines Hochschulnetzes kann es je nach Umsetzung innerhalb der jeweiligen Institutionen mehrere Anwendungsbereiche geben. So auch im MWN mit einem Dienstleister – dem LRZ – und einer großen Anzahl an Kunden. Daher ist es unbedingt notwendig, dass der Anwendungsbereich flexibel festgelegt werden kann.

NA16 `Zweckdienlichkeit des Anwendungsbereichs`
Gleichzeitig muss der Anwendungsbereich so gewählt werden, dass das Ziel des Schwachstellenmanagements erfüllt werden kann: Die Erreichung einer höheren Netzsicherheit durch die Reduktion der Schwachstellen auf den im Anwendungsbereich vorhandenen Assets.

NA17 `Definition der Aktivitäten`
Ein zentraler Bestandteil beim effektiven Management ist die Festlegung der im Pro-

zess vorgesehenen Aktivitäten. Die Aktivitäten wurden bereits in Abschnitt 2.2 beschrieben: Die Identifikation, Klassifikation, Beseitigung und Abschwächung sowie die Prävention von Schwachstellen. Die Aktivitäten geben prinzipiell nur einen groben Rahmen des Schwachstellenmanagements vor und müssen auch nicht alle erfüllt werden. Der Inhalt der Aktivitäten muss jedoch an die Umgebung angepasst sein.

NA18 `Ausübungsintervall des Aktivitätenzyklus`
Wie in Abschnitt 2.3 beschrieben, bauen die Aktivitäten aufeinander auf und bilden den Inhalt eines zyklischen Prozesses des Schwachstellenmanagements. Sinnvollerweise muss dieser Zyklus, je nach Inhalt der Aktivitäten, definiert sein und in regelmäßigen und ausreichend kurzen Abständen durchlaufen werden.

NA19 `Definition von Verfahren`
Im Management ist es auch notwendig, Verfahren innerhalb der Aktivitäten zu definieren, um zum einen die Wirksamkeit und Effizienz gewährleisten, als auch Zuständigkeiten festlegen zu können.

NA20 `Definition von Rollen und Verantwortlichkeiten`
Zur Klärung der Zuständigkeiten müssen Rollen auf Ebene von Einzelpersonen im Managementprozess definiert werden. Diesen werden schließlich Verantwortlichkeiten übertragen, für deren Erfüllung sie zuständig sind.

NA21 `Prüfung durch Leistungsindikatoren`
Da der Zweck von Managementprozessen insbesondere einer Erreichung der Effektivität und Effizienz folgt, muss fortlaufend überprüft werden, ob dies noch der Fall ist. Auch die Prüfung nach Konformität mit Richtlinien und Vorgaben muss kontrolliert werden. Die Prüfung erfolgt anhand sogenannter „Key Performance Indicators“ (KPI), bzw. Leistungsindikatoren. Die Leistungsindikatoren müssen so definiert sein, dass die genannten Aspekte durch sie überprüft werden können.

NA22 `Dezentralisierung der Aktivitäten`
Aufgrund der Struktur von Hochschulnetzen, insbesondere dem als Szenario genutzten MWN, ist es notwendig, die Aktivitäten zu dezentralisieren. Zum einen, da das LRZ und seine Kunden im selben Netz sind und die Zuständigkeit des LRZ nur „bis zur Datendose“ reicht. Dies traf ebenfalls für einige der in Abschnitt 2.4 betrachteten Hochschulumgebungen zu. Insofern sind Hochschulrechenzentren üblicherweise nicht für Assets in Kundennetzen verantwortlich und haben in der Regel keine Möglichkeit, Maßnahmen für diese Systeme zu ergreifen. Zum anderen kann so der Ressourcenmangel, zumindest teilweise, kompensiert werden. Zu beachten ist, dass jeder Organisationsbereich die Aktivitäten des Schwachstellenmanagements, abhängig von den Anforderung der jeweiligen Umgebung, umsetzen muss.

NA23 `Schnittstellen zu anderen Prozessen`
Da die Aktivitäten und Geschäftsprozesse im Hochschulbereich und auch am LRZ gemanagt sind, sollte ein Zusammenhang bzw. geeignete Schnittstellen zwischen dem Schwachstellenmanagement und vorhandenen, geeigneten Managementprozessen wie beispielsweise dem Changemanagement oder dem Risikomangement hergestellt werden (vgl. Abschnitt 2.2). Diese Maßnahme kann zu einer Steigerung der Effizienz beitragen, ist jedoch nicht ausschlaggebend für ein funktionierendes Schwachstellenmanagement.

NA24 `Identifikation von Assets`
Um eine effektive Kommunikation zu gewährleisten, ist es notwendig, dass Assets innerhalb des Schwachstellenmanagements eindeutig identifiziert werden können.

NA25 `Qualität der Asset-Beschreibung`
Um ein wirksames und effizientes Schwachstellenmanagement realisieren zu können, ist es erforderlich, dass Assets ausreichend beschrieben sind. Inbesondere für die Wiederverwendbarkeit in automatisierten Abläufen muss das Asset-Inventar genug Informationen halten, um die im Anwendungsbereich festgelegten Aktivitäten ausführen zu können.

NA26 `Definition eines Asset-Verantwortlichen`
Wie bereits im vorhergehenden Abschnitt als allgemeine technische Anforderung festgelegt, muss es zur Sicherstellung der Effektivität und Effizienz des Schwachstellenmanagements ein Asset-Inventar geben (Anforderung `NA1`). Auch wurde die Notwendigkeit sowie Probleme einer Automatisierung der Abläufe erläutert. Da die vollständige Automatisierung nicht immer möglich ist und es Abläufe gibt, die notwendigerweise manuell durchgeführt werden müssen, muss jedem Asset ein Verantwortlicher (engl. „Owner“) zugewiesen sein.

NA27 `Nutzung geeigneter Meldewege`
Die Meldung von Schwachstellen und diesbezüglichen Informationen ist ein wichtiger Aspekt beim Aufbau eines Schwachstellenmanagements, da in diesem in der Regel eine Vielzahl an Personen involviert sind. Auch im MWN mit einigen zehntausend Nutzern ist das der Fall. Um den Informationsfluss kontrollierbar zu machen, benötigt es – vor allem bei der hohen Anzahl verschiedener Parteien in Hochschulnetzen wie des MWN – vorgegebene Meldewege. So kann die Zuverlässigkeit eines Informationsflusses innerhalb eines entsprechend großen Organisationsbereichs wie einem Hochschulnetz sichergestellt werden.

FA28 `Erstellung von Berichten`
Um einen Überblick über den aktuellen Zustand von Schwachstellen auf Assets zu bekommen, muss dieser durch die Erstellung von Berichten und Kommunikation an die für das Schwachstellenmanagement oder die Informationssicherheit verantwortlichen Personen übermittelt werden.

3.3.2.2 Identifikation

NI8 `Aktualität der Schwachstelleninformation`
Da die im MWN eingesetzten Softwareprodukte nicht vorgegeben sind bzw. durch die Art der Organisation und der hohen Anzahl an Nutzern überhaupt nicht einschränkbar ist, muss auch die Schwachstellen-Dokumentation potenziell Schwachstellen für eine große Anzahl an Softwareprodukten halten. Dazu kommt, dass nicht nur die eingesetzte Software variiert, sondern auch die verwendeten Versionen. Viele Nutzer halten ihre Systeme auf dem neuesten Stand, wodurch auch einige Schwachstellen beseitigt werden, aber auch neue Schwachstellen in die Umgebung hinzukommen können. Daher muss für ein effektives Schwachstellenmanagement und eine effektive Identifikation der Schwachstellen darauf geachtet werden, dass vorhandene Daten aktuell sind.

NI9 `Identifikation und Kommunikation von Schwachstellen`
Im Schwachstellenmanagement bilden Informationen über Schwachstellen die Grundlage zur Ausführung der darin umgesetzten Aktivitäten. Insofern müssen Schwachstellen regelmäßig identifiziert werden sowie allgemein die Möglichkeit zur Identifikation und Dokumentation bestehen. Ein wichtiger Bestandteil des Schwachstellenmanagements ist die Kommunikation von Schwachstellen. Zu diesem Zweck ist es unbedingt notwendig, den Schwachstellen jeweils genau einen Bezeichner zuzuweisen, der die Schwachstelle eindeutig identifiziert.

NI10 `Qualität der Schwachstellenbeschreibung`
Eine Anforderung, die nicht nur in Hochschulnetzen von Bedeutung ist, sondern generell zur Ermöglichung der Beseitigung der Schwachstellen unumgänglich ist, ist die Qualität der Beschreibung der jeweiligen Schwachstelle. Aufgrund der hohen Vielfalt an Software und Möglichkeiten zur Ausnutzung sowie Beseitigung von Schwachstellen, müssen von der Schwachstelle betroffene Produkte, die Art und Weise der Ausnutzbarkeit sowie die Folgen einer erfolgreichen Ausnutzung aus der Beschreibung hervorgehen.

3.3.2.3 Klassifikation

NK3 `Einheitlichkeit und Qualität der Schwachstellenbewertung`
Eine Auffälligkeit, die sich aus der Umfrage am LRZ ergab, ist, dass die Kriterien zur Behebung von Schwachstellen am LRZ nicht festgelegt und von der persönlichen Einschätzung der Mitarbeiter abhängig sind. So kann es je nach bearbeitendem Mitarbeiter vorkommen, dass dieselbe Schwachstelle auf gleichartigen Systemen unterschiedlich behandelt wird. Insbesondere bei entsprechend großem Anwendungsbereich (z.B. einem im MWN netzweiten Schwachstellenmanagement) kann sich diese Problematik weiter verstärken. Dies führt zu der Anforderung nach einem schwachstellen- und umgebungsabhängigen und insbesondere netzweit einheitlichen Bewertungssystem.

Anhand des Bewertungssystems muss ersichtlich sein, ob eine Schwachstelle behoben werden muss. Zudem muss die Kritikalität zweier Schwachstellen vergleichbar sein, um klare und wiederholbare Aussagen über die Priorisierung der Behebung treffen zu können.

NK4 `Berücksichtigung der Umgebung`
Die Klassifikation muss zum einen an den Einsatzbereich des Schwachstellenmanagements angepasst sein sowie zum anderen an das jeweilige Asset, das von einer Schwachstelle betroffen ist. Diese Anforderung ergibt sich vor allem aufgrund der unterschiedlichen Computersysteme im Münchner Wissenschaftsnetz und allgemein in Hochschulnetzen. Dabei müssen die Verfahren innerhalb der Klassifikation auf alle Typen und Rollen von Assets (z.B. Arbeitsplatzrechner, Server, Router, Switch und Drucker) anwendbar sein.

NK5 `Kritikalität von Assets`
Aus der Anforderung der Berücksichtigung der Umgebung ergibt sich insofern eine weitere Anforderung an das Asset-Inventar: Darin muss die Kritikalität eines jeden Assets entnehmbar sein. Die Bestimmung der Kritikalität ist konzeptabhängig und kann sich beispielsweise aus der Rolle eines Assets und den darauf verarbeiteten bzw. gespeicherten Daten ergeben.

NK6 `Kriterium der Schwachstellenakzeptanz`
Im Schwachstellenmanagement in Hochschulnetzen muss auch ein Kriterium zur Akzeptanz von Schwachstellen definiert sein. Durch die beschränkten Ressourcen, die dem Rechenzentrum und Instituten zur Verfügung stehen, können nicht alle Schwachstellen behoben werden bzw. das Verhältnis des Aufwands zur Behebung nicht angemessen sein. Das Schwachstellenakzeptanzkriterium führt dazu, dass – je nach Definition des Kriteriums – weniger kritische Schwachstellen akzeptiert werden und kritischeren Schwachstellen dadurch mehr Aufmerksamkeit geschenkt werden kann.

3.3.2.4 Beseitigung und Abschwächung

NB6 `Zentralisierung der Methoden und Daten der Detektion`
Wie bereits im Szenario des MWN beschrieben, gibt es im Hochschulnetz nicht selten mehrere Parteien. Zum einen das Rechenzentrum, welches Dienstleistungen anbietet und zum anderen Kunden und Nutzer, die diese Dienstleistungen in Anspruch nehmen. Insofern – und aus Mangel an Ressourcen – werden die Aktivitäten im Hochschulnetz bei einem großen Anwendungsbereich vom Rechenzentrum sowie den einzelnen Instituten umgesetzt. Bei Verwendung mehrerer Methoden der Detektion sollten daher die Ergebnisse zentral zusammenlaufen, um diese zu kombinieren und einen größeren Zusammenhang zu ermöglichen. Eine Nicht-Erfüllung kann zu mehr Aufwand bei der Verarbeitung der Daten führen, beeinflusst jedoch nicht die Wirksamkeit des Schwachstellenmanagements.

NB7 `Definition von Maschinen zur Detektion`
Aus Aspekten der Sicherheit und Verwaltung sollten Computersysteme mit den notwendigen technischen Möglichkeiten zur Identifikation ausgestattet und als Maschinen zu Zwecken der Detektion von Schwachstellen über das Netz definiert werden. Durch die Bestimmung der Rechner ist es einerseits leichter möglich, nicht nur die Scan-Funktionalität konsistent zu halten, sondern diese auch entsprechend zu warten. Andererseits kann so die Konfiguration auf den Zielsystemen und innerhalb der Zielnetze leichter auf die Scans abgestimmt werden, beispielsweise durch eine Freischaltung des Zugriffs dieser Systeme in Form entsprechender Einträge in *Access Control Lists* oder Firewall-Regeln. Auch diese Anforderung zielt vor allem auf eine höhere Effizienz im Schwachstellenmanagement ab. Ein Schwachstellenmanagement ohne definierte Maschinen zur Detektion kann ebenso wirksam sein.

NB8 `Zeitnahe Beseitigung und Abschwächung`
Die zeitliche Komponente bei der Behandlung von Schwachstellen nimmt eine wichtige Rolle in der Sicherstellung der Informationssicherheit ein. IT-Dienste, die im MWN angeboten werden und teilweise auch über das Internet erreichbar sind, werden in der Regel von mehreren zehntausend Nutzern in Anspruch genommen und besitzen zum Teil eine hohe Relevanz bei der Erfüllung ihrer Aufgaben. Dienste wie E-Mail oder DNS, aber insbesondere auch Dienste zur Unterstützung der Organisation haben einen großen Einfluss auf den Alltag der Nutzer. Insofern muss darauf geachtet werden, dass Schwachstellen nicht nur regelmäßig und in kurzen Abständen erfasst und detektiert, sondern auch tatsächlich abgeschwächt bzw. beseitigt werden, um die Sicherheit der Assets zu erhöhen. Offene Schwachstellen – dem Dienstbetreiber bekannt oder unbekannt – bedeuten immer eine potenzielle Gefahr für die Assets, darauf liegenden Daten

sowie für die Verfügbarkeit der durch die Assets realisierten IT-Dienstleistungen.

NB9 `Prüfung einer Maßnahme auf Wirksamkeit`
Aus dem gleichen Grund, der eine zeitnahe Beseitigung und Abschwächung von Schwachstellen (`NA8`) erforderlich macht, muss auch die Wirksamkeit einer Maßnahme überprüft werden: Maßnahmen, die Schwachstellen vermeintlich beheben, verringern im schlechtesten Fall weder die Wahrscheinlichkeit noch die Auswirkung einer Ausnutzung, lassen die Bedrohung der Schwachstelle jedoch aus dem Bewusstsein der Asset- und Dienstbetreiber verschwinden. Daher muss die Wirksamkeit von Maßnahmen zur Beseitigung und Abschwächung von Schwachstellen überprüft werden.

NB10 `Berücksichtigung von Risiken einer Behebung`
Durch das hohe Ausmaß an Schadenspotential, das durch Ausfälle von Assets ausgelöst werden kann – beispielsweise durch eine Einschränkung der Aufgaben der Netznutzer bei Beeinträchtigungen der Verfügbarkeit oder auch Auswirkungen durch den Verlust der Vertraulichkeit oder Integrität von Informationen (insbesondere personenbezogenen Daten) – muss nicht nur berücksichtigt werden, dass die Ausnutzung von Schwachstellen zu derartigen negativen Auswirkungen führen kann, sondern auch Gegenmaßnahmen zur Behebung der Schwachstellen. Bei der Anwendung einer Maßnahme, beispielsweise der Installation eines Patches oder dem Austausch der Software, müssen genauso mögliche Risiken und Beeinträchtigungen in Betracht gezogen werden.

NB11 `Priorisierung kritischer Schwachstellen`
Eine weitere Anforderung an ein Schwachstellenmanagement, die einen hohen Beitrag zur Informationssicherheit leisten kann, ist die Notwendigkeit einer bevorzugten Behebung kritischer Schwachstellen bei der Behebung. Da sich – abhängig von der jeweiligen Klassifikation bzw. Bewertung – kritische Schwachstellen gegenüber weniger kritischen Schwachstellen entweder leichter ausnutzen lassen oder mehr Schadenspotential aufweisen, sind sie unbedingt mit höherer Priorität zu beseitigen, da von ihnen ein größeres Risiko ausgeht.

NB12 `Priorisierung kritischer Assets`
Genauso müssen auch kritische Assets bei der Behebung von Schwachstellen bevorzugt behandelt werden. Die Kritikalität eines Assets hängt von mehreren, in der Regel umgebungs- und nutzungsabhängigen Faktoren ab, sowie vom Schutzbedarf der darauf verarbeiteten Daten.

NB13 `Möglichkeit einer effektiven Behebung`
Die Dienstbetreiber am Leibniz-Rechenzentrum haben im Moment die Möglichkeit (organisatorisch sowie technisch), selbst über die Anwendung von Maßnahmen zur Beseitigung von Schwachstellen zu entscheiden. Das hat vor allem den Vorteil, dass Schwachstellen in der Regel relativ schnell nach Identifikation geschlossen werden, da beispielsweise nicht auf die Existenz eines Patches gewartet werden muss, sondern auch Workarounds und Temporal Fixes (Definition und Unterschiede siehe Abschnitt 2.3.2) ebenfalls als akzeptable – wenn auch nur temporäre – Lösungen gelten. Teilweise besteht überhaupt nicht die Möglichkeit einer Beseitigung von Schwachstellen nach einem vorgeschriebenem Verfahren.

3.3.2.5 Prävention

NP1 `Risikoeinschätzung vor Einsatz von Software`
Da Schwachstellen auf Computersystemen durch den Einsatz von Software auftreten, setzt die Prävention zeitlich davor an. Insofern muss es vor jedem Einsatz von Software eine Einschätzung darauf vorhandener Schwachstellen geben.

3.4 Tabellarische Kurzzusammenfassung der Anforderungen

In diesem Abschnitt befindet sich eine in die Aktivitäten des Schwachstellenmanagements aufgeteilte Liste der Anforderungen in Tabellenform, die in Abschnitt 3.3 erläutert wurden.

Die Gliederung erfolgt zu besseren Übersichtlichkeit einzeln nach Aktivität im Schwachstellenmanagement

3.4.1 Allgemeine Anforderungen an ein Schwachstellenmanagement

In Tabelle 3.1 sind die allgemeinen Anforderungen an ein Schwachstellenmanagement zusammengefasst. Technische Anforderungen werden in Abschnitt 3.3.1.1, organisatorische in Abschnitt 3.3.2.1 erläutert.

ID	Anforderungsbeschreibung	Gewichtung
	technisch	
NA1	Existenz eines Asset-Inventars	erforderlich
NA2	Existenz und Zentralisierung einer Schwachstellen-Dokumentation	erforderlich
NA3	Zentralisierung des Asset-Inventars	empfohlen
FA4	Funktionen zum Aufbau und Pflege des Asset-Inventars	erforderlich
FA5	Zugriff auf Informationen und Funktionen über eine API	empfohlen
NA6	Automatischer Erhalt der Aktualität des Asset-Inventars	erforderlich
NA7	Automatisierung der Aktivitäten	erforderlich
NA8	Unterstützung manueller Tätigkeiten	erforderlich
NA9	Plattformunabhängigkeit der Aktivitäten	erforderlich
FA10	Verbinden mit weiteren sicherheitsrelevanten Informationen	empfohlen
FA11	Exportierbarkeit der Daten	empfohlen
NA12	Präsentation der Daten	empfohlen
NA13	Mandantenfähigkeit	erforderlich
NA14	Institutsübergreifende Benutzerverwaltung	erforderlich
	organisatorisch	
NA15	Flexibilität des Anwendungsbereichs	erforderlich
NA16	Zweckdienlichkeit des Anwendungsbereichs	erforderlich
NA17	Definition der Aktivitäten	erforderlich
NA18	Ausübungsintervall des Aktivitätenzyklus	erforderlich
NA19	Definition von Verfahren	erforderlich
NA20	Definition von Rollen und Verantwortlichkeiten	erforderlich
NA21	Prüfung durch Leistungsindikatoren	erforderlich
NA22	Dezentralisierung der Aktivitäten	erforderlich
NA23	Schnittstellen zu anderen Prozessen	empfohlen

NA24	Identifikation von Assets	erforderlich
NA25	Qualität der Asset-Beschreibung	erforderlich
NA26	Definition eines Asset-Verantwortlichen	erforderlich
NA27	Nutzung geeigneter Meldewege	erforderlich
FA28	Erstellung von Berichten	erforderlich

Tabelle 3.1: Allgemeine Anforderungen an ein Schwachstellenmanagement in Hochschulnetzen

3.4.2 Anforderungen an die Identifikation von Schwachstellen

In diesem Abschnitt bzw. Tabelle 3.2 werden Anforderungen an die Identifikation im Schwachstellenmanagement zusammengefasst. Die Erläuterung der technischen Anforderungen findet sich in Abschnitt 3.3.1.2, die der organisatorischen Anforderungen in Abschnitt 3.3.2.2

ID	**Anforderungsbeschreibung**	**Gewichtung**
	technisch	
FI1	(Manuelles) Hinzufügen neuer Einträge in die Schwachstellen-Dokumentation	erforderlich
FI2	Zugriff auf Einträge in der Schwachstellen-Dokumentation	erforderlich
FI3	Editieren von Einträgen in der Schwachstellen-Dokumentation	erforderlich
FI4	Suchfunktionalität innerhalb der Schwachstellen-Dokumentation	erforderlich
FI5	Automatischer Erhalt der Aktualität der Schwachstellen-Dokumentation	erforderlich
NI6	Dynamische Festlegung der Quellen der Schwachstellen-Dokumentation	erforderlich
FI7	Filtern relevanter Schwachstellen	erforderlich
	organisatorisch	
NI8	Aktualität der Schwachstellen-Dokumentation	erforderlich
NI9	Identifikation und Kommunikation von Schwachstellen	erforderlich
NI10	Qualität der Schwachstellenbeschreibung	erforderlich

Tabelle 3.2: Anforderung an die Identifikation von Schwachstellen

3.4.3 Anforderungen an die Klassifikation von Schwachstellen

Die Zusammenfassung der Anforderungen an die Klassifikation wird in Tabelle 3.3 vorgenommen. Die Erläuterung wird in Abschnitt 3.3.1.3 für technische bzw. Abschnitt 3.3.2.3 für organisatorische Anforderungen vorgenommen.

ID	**Anforderungsbeschreibung**	**Gewichtung**
	technisch	
FK1	Automatisierung der Klassifikation	erforderlich
FK2	Manuelle Klassifikation	erforderlich
	organisatorisch	
NK3	Einheitlichkeit und Qualität der Schwachstellenbewertung	erforderlich

NK4	Berücksichtigung der Umgebung	erforderlich
NK5	Kritikalität von Assets	erforderlich
NK6	Kriterium der Schwachstellenakzeptanz	erforderlich

Tabelle 3.3: Anforderung an die Klassifikation von Schwachstellen

3.4.4 Anforderungen an die Beseitigung und Abschwächung von Schwachstellen

In diesem Abschnitt befindet sich die Zusammenfassung der Anforderungen an die Aktivität der Beseitigung und Abschwächung. Die Anforderungen werden in Abschnitt 3.3.1.4 sowie Abschnitt 3.3.2.4 erklärt.

ID	Anforderungsbeschreibung	Gewichtung
	technisch	
FB1	Verfahren zur Detektion von Schwachstellen	erforderlich
FB2	Automatisierung der Detektion	erforderlich
FB3	Detektion aus Systemsicht	empfohlen
FB4	Detektion aus Netzsicht	empfohlen
FB5	Unterstützung bei der Behebung durch Computersysteme	erforderlich
	organisatorisch	
NB6	Zentralisierung der Methoden und Daten der Detektion	empfohlen
NB7	Definition von Maschinen zur Detektion	empfohlen
NB8	Zeitnahe Beseitigung und Abschwächung	erforderlich
NB9	Prüfung einer Maßnahme auf Wirksamkeit	erforderlich
NB10	Berücksichtigung von Risiken	erforderlich
NB11	Priorisierung kritischer Schwachstellen	erforderlich
NB12	Priorisierung kritischer Assets	erforderlich
NB13	Möglichkeit einer effektiven Behebung	erforderlich

Tabelle 3.4: Anforderung an die Beseitigung und Abschwächung von Schwachstellen

3.4.5 Anforderungen an die Schwachstellenprävention

Die Zusammenfassung der Anforderungen an die Schwachstellenprävention wird in Tabelle 3.5 vorgenommen. Die Erläuterung findet sich in Abschnitt 3.3.2.5.

ID	Anforderungsbeschreibung	Gewichtung
	organisatorisch	
NP1	Risikoeinschätzung vor Einsatz von Software	erforderlich

Tabelle 3.5: Anforderung an die Prävention von Schwachstellen

4 Bestehende Arbeiten zum Schwachstellenmanagement in Hochschulnetzen

Die Behandlung von Schwachstellen in Netzen und Computersystemen gilt in vielen Standards zur Umsetzung eines Informationssicherheitsmanagements als wichtiger Teilaspekt, auch wenn genaue Anleitungen, Vorgehensweisen und die konkrete Umsetzung allgemein nicht oder nur auf einer sehr abstrakten Ebene erläutert werden. Der Hauptgrund für solche ungenauen Angaben zum Aufbau und Inhalt eines Schwachstellenmanagements ist, dass der Umfang der betrachteten Schwachstellen und die im Management unternommenen Maßnahmen abhängig von der jeweiligen Definition des Rahmens und den Anforderungen ist, die eine Organisation an ein solches System stellt. Auch gibt es einige konkretere Überlegungen über den Inhalt und die Tätigkeiten von Schwachstellenmanagement. Nicht selten wurden diese jedoch bereits vor einigen Jahren verfasst und treffen nicht mehr exakt die heutigen Anforderungen. Andererseits hat sich die grobe Struktur nur wenig geändert, wodurch die grundlegenden Maßnahmen noch immer verwendbar sind.

In diesem Kapitel wird auf einige Arbeiten zum Schwachstellenmanagement eingegangen, die im Anschluss im Hinblick auf ihre Eignung zur Umsetzung eines Schwachstellenmanagements in Hochschulnetzen anhand ihrer Erfüllung der im vorherigen Abschnitt festgelegten Anforderungen überprüft werden. Der Aufbau des Kapitels richtet sich nach der Art der betrachteten Arbeiten. Zum einen gibt es Standards und „Good Practices“, die mehr Anforderungen an ein Schwachstellenmanagement stellen als ein konkretes Konzept zu beschreiben und im folgenden Abschnitt betrachtet werden. Zum anderen sind in Abschnitt 4.2 Konzepte und bereits allgemein vorhandene Lösungen beschrieben. In Abschnitt 4.3 werden zudem relevante wissenschaftliche Arbeiten erläutert.

Eine Übersicht der Erfüllung der Anforderungen wird im letzten Abschnitt des Kapitels in Tabelle 4.2 bereitgestellt.

4.1 Standards und Good Practices

In diesem Abschnitt wird der Inhalt von Standards und „Good Practices“ bezüglich Schwachstellenmanagement hin überprüft. Zu ersterer Kategorie wird insbesondere die *ISO/IEC 27000-Reihe* betrachtet, zu letzterer Kategorie Dokumente im Rahmen der *Special Publications* Reihe des *National Institute of Standards and Technology* (NIST) sowie eine durch das SANS-Institut veröffentliche Arbeit. Der vom *Bundesamt für Sicherheit in der Informationstechnik* (BSI) veröffentlichte *IT-Grundschutz* stellt eine Überscheidung aus Standards und „Good Practices“ dar.

In Abschnitt 4.1.5 wird auf Veröffentlichungen von Herstellern von Softwarelösungen zum Schwachstellenmanagement eingegangen.

4.1.1 ISO/IEC 27000-Reihe

In dem Standard *ISO/IEC 27001* – „Information technology - Security techniques - Information security management systems - Requirements“ [Int08] der *International Organization for Standardization* und *International Electrotechnical Commission* werden Schwachstellen zum einen insbesondere im Risikomanagement erwähnt, in dem sie ein Teil der Identifizierung von Risiken ausmachen.

Außerdem wird in Punkt *A.12.6* der darin aufgezählten Maßnahmenziele und Maßnahmen direkt das Thema Schwachstellenmanagement adressiert. Darin wird unter dem Punkt *A.12.6.1* das Maßnahmenziel „Kontrolle technischer Schwachstellen“ betrachtet, das durch das rechtzeitige Einholen von Informationen über derartige Schwachstellen, deren Bewertung und Behandlung erfüllt werden soll.

Eine Umsetzung der Maßnahme ist in dem damit in Verbindung stehenden Dokument *ISO/IEC 27002* – „Informationstechnik - IT-Sicherheitsverfahren - Leitfaden für das Informationssicherheits-Management“ [Int14b] beschrieben: Demnach soll die Umsetzung einer zeitnahen Anwendung von Gegenmaßnahmen zur Behebung einer Schwachstelle dienen (vgl. Anforderung `NB8`).

Dazu müssen zunächst alle Assets mit aktueller Konfiguration erfasst werden. Auch die Festlegung von Aufgaben und Verantwortlichkeiten zur Überwachung und Bewertung von Schwachstellen und der Installation von Softwarepatches muss vorgenommen werden. Somit entspricht dieser Schritt den Anforderungen `NA19` und `NA20` aus dem vorherigen Kapitel.

Quellen zur „Feststellung relevanter technischer Schwachstellen und deren nachhaltiger Bewusstmachung“ [Int14b] müssen ebenfalls identifiziert und ein Zeitplan erstellt werden, der die Behandlung von Schwachstellen beschreibt.

Im Standard wird auch auf eine Notwendigkeit zur Bewertung der Kritikalität von Schwachstellen hingewiesen. Eine konkrete Umsetzung, beispielsweise in Form eines Bewertungssystems für Schwachstellen, wird nicht beschrieben.

Bevor Patches als Maßnahme zur Behebung einer Schwachstelle angewendet werden, müssen diese jeweils auf damit verbundene Risiken überprüft werden. Dazu gehört eine Überprüfung auf Wirksamkeit (d.h. der Schließung der Schwachstelle) sowie durch die Installation unerwünschte Effekte (Anforderungen `NB9` und `NB10`). Auch muss eine Zurückstellung von Patches abgewogen und bewertet werden.

Im Standard sind außerdem beispielsweise eine Außerbetriebnahme eines Dienstes, die Überwachung des Assets oder auch der Einsatz weiterer Maßnahmen zur Zugriffssteuerung vorgesehen, falls eine Schwachstelle nicht durch einen Patch behoben werden kann. In jedem Fall fordert der Standard, dass Assets mit hohem Risiko bevorzugt behandelt (Anforderungen `NB12`, `NB13`) sowie angewendeten Maßnahmen in einem Audit-Protokoll vermerkt werden. In der Umsetzung muss es zudem Pläne für detektierte, jedoch nicht behebbare Schwachstellen geben.

Zudem wird gefordert, dass der Prozess überwacht sowie auf seine Effektivität und Effizienz hin bewertet wird (Anforderung `NA21`). Konkrete Leistungsindikatoren werden nicht angegeben.

Genau wie in Anforderung `FA10` wird festgelegt, dass die Daten aus dem Schwachstellenmanagement mit „Aktivitäten im Umgang mit Sicherheitsvorfällen“ (in der Regel dem *Security Information & Event Management*) abgeglichen werden sollen.

Die Beschreibung des Schwachstellenmanagements im Standard *ISO/IEC 27001* sowie *ISO/IEC 27002* ist sehr oberflächlich gehalten und umreißt den groben Rahmen mit allgemeinen Vorschlägen zum Inhalt. Insofern geht sie nicht weit über die Anforderungsbeschreibung hinaus und ist nicht als konkret realisierbares Konzept nutzbar.

4.1.2 BSI IT-Grundschutz

Der IT-Grundschutz des BSI ist eine Sammlung von Standards mit technischen als auch organisatorischen Maßnahmen zur Erreichung von Sicherheit allgemein und insbesondere Informationssicherheit, sowie einem geregeltem Management in Organisationen.

4.1.2.1 BSI IT-Grundschutz-Standards

In den IT-Grundschutz-Standards des BSI, welche Empfehlungen zu Methoden, Prozessen, Verfahren, Vorgehensweisen sowie Maßnahmen zur Verbesserung der Informationssicherheit bieten, findet sich der Umgang mit Schwächen und Schwachstellen an Kernpunkten wieder [Bun15a]. Demnach ist die Beseitigung von Schwächen die abschließende der grundlegenden Phasen im Lebenszyklus des Informationssicherheits-Prozesses.

Angelehnt an das PDCA-Modell („Plan“, „Do“, „Check“, „Act“) beginnt der Prozess zur Informationssicherheit mit der Planung – der Ausarbeitung einer Sicherheitsstrategie, geht über die Phasen der Umsetzung des vorher ausgearbeiteten Plans durch die Erstellung eines Sicherheitskonzepts und der Erfolgskontrolle bzw. der Zielerreichung, bis zur Handlungsphase, der Verbesserung und Beseitigung der Schwächen.

Im Rahmen des zum Informationssicherheitsmanagement gehörenden Risikomanagements, sind weiterhin die Identifikation und Beseitigung von Schwachstellen essentielle Schritte zur Verringerung des Sicherheitsrisikos für die jeweilige Organisation [Bun08a]. Weitere Umsetzungshilfen eines Informationssicherheitsmanagementsystems stellt das BSI im *BSI-Standard 100-2* [Bun08b] sowie Empfehlungen zur Risikoanalyse auf der Basis von IT-Grundschutz im *BSI-Standard 100-3* [Bun08c] zur Verfügung.

4.1.2.2 BSI IT-Grundschutz-Kataloge

Der Umgang mit Schwachstellen wird nicht nur in den Standards des IT-Grundschutzes erwähnt, sondern ebenfalls in den darin enthaltenen Katalogen.

So wird im Gefährdungskatalog in den Unterpunkten *G 0.28* [Bun11] sowie *G 4.22* [Bun14b] darauf hingewiesen, dass Sicherheitslücken insbesondere auch durch Schwachstellen in Software auftreten. Mehr als den eigentlichen Hinweis und Beispiele umfasst die Beschreibung hingegen nicht.

Jedoch auch ein weiterer Punkt, ein sogenannter „Baustein“ im IT-Grundschutz mit dem Bezeichner *G 1.14* – „Patch- und Änderungsmanagement“ [Bun09], weist einen Bezug zum Schwachstellenmanagement auf. In diesem wird beschrieben, dass eine Umsetzung eines Patch- und Änderungsmanagements insbesondere zur Vermeidung von Störungen und Sicherheitslücken führen soll. Zu diesem Zweck sollen „alle Änderungen an Hard- und Softwareständen sowie deren Konfiguration“ [Bun09] über dieses gesteuert werden.

Abgesehen davon, dass es sich dabei nicht um ein Schwachstellenmanagement handelt und dieser Aspekt in keinem Baustein des BSI-Grundschutz für sich adressiert wird, fehlen Aspekte wie die Identifikation, Bewertung, Priorisierung und Detektion von Softwareschwachstellen komplett.

4.1.2.3 Bewertung des BSI IT-Grundschutz

Auch wenn in den Katalogen und Standards des IT-Grundschutz grob die Notwendigkeit der Behebung von Schwachstellen beschrieben wird, so ist jedoch keine konkrete Beschreibung einer Umsetzung eines Schwachstellenmanagements enthalten.

Des Weiteren ist ein Schwachstellenmanagement als solches darin nicht vorgesehen, sondern mehr als Aspekt in anderen Prozessen, wie dem Management von Patches oder Änderungen, integriert.

Die Beschreibung geht zudem nicht über die Erwähnung einzelner Aktivitäten (vor allem Identifikation und Behandlung) (Anforderung `NA17`) sowie der Zusammenhänge zur Informationssicherheit hinaus und ist als konkreter Umsetzungsplan ungeeignet.

4.1.3 NIST Special Publications 800

Die *Special Publications* (SP) des NIST des US-Handelsministeriums umfasst drei Serien an Dokumenten und Empfehlungen zum Thema Informationssicherheit : Die *SP 500* „Computer Systems Technology“, welche generellere Informationen zum Thema *Information Technology* beinhaltet, die *SP 800* „Computer Security“-Reihe mit Dokumenten rund um Informationssicherheit sowie die dritte Reihe mit praktischen Leitfäden zur Umsetzung dieser, der *SP 1800* „NIST Cybersecurity Practice Guide“, welche erst im Jahr 2015 entstanden ist. [Nat15b]

Die höchste Relevanz für das Management von Software- und Konfigurationsschwachstellen bietet die *SP 800* Reihe, in der insbesondere drei Dokumente näher auf das Thema eingehen und in den folgenden Abschnitten beschrieben werden.

4.1.3.1 Assessing Security and Privacy Controls in Federal Information Systems and Organizations

Der Inhalt dieser Special Publication (*SP 800-53 Revision 4*) [Nat13] befasst sich mit der Auswahl von Maßnahmen allgemein zum Schutz von Assets vor einer Vielzahl von Bedrohungen und insbesondere dem Schutz vor Angriffen auf Computersysteme.

Wie auch in den IT-Grundschutz-Standards des BSI, wird die Beseitigung von Schwachstellen als einer der zentralen Aspekte zur Erreichung von Informationssicherheit angesehen. Der in dem Dokument veröffentlichte Katalog enthält mehrere Maßnahmen mit Bezug zum Schwachstellenmanagement, beispielsweise die Verknüpfung von Informationen über Schwachstellen mit auftretenden Sicherheitsvorfällen (`IR-6 (2)`), die Bedrohungs- und Schwachstellenanalyse bei der Entwicklung von Informationssystemen, Systemkomponenten oder Diensten durch den Entwickler (`SA-11 (2)`, `SA-15 (7)`), die Wiederverwendung von Informationen über Schwachstellen sowie Bedrohungen (`SA-15 (8)`) und die Schulung von Softwareentwicklern bezüglich Schwachstellen und deren Ausnutzung (`AT-3 (3)`).

Ein grundlegender Aspekt der Informationssicherheit, die Gefahrenanalyse oder Risikobewertung, wird in dem Dokument als eigene Kategorie an Maßnahmen zum Erhalt der Informationssicherheit eingestuft, in welcher auch hier die Identifikation, Klassifikation und Behebung von Schwachstellen von besonderer Bedeutung sind.

Darin befasst sich Maßnahme `RA-5` mit dem „Vulnerability Scanning“, also allgemein der Detektion von Schwachstellen, welche in vordefinierten Zeitintervallen bzw. als Folge einer

Identifikation neuer Schwachstellen durchgeführt werden soll. Die Autoren schreiben, dass diese außerdem unter der Verwendung der Standardisierung von Informationen wie Plattformen (bzw. Betriebssystemen), Softwarefehlern und Fehlkonfiguration, aber auch Checklisten und Prozeduren für Tests sowie der Schwachstellenbewertung automatisiert ablaufen soll.

Es folgt die Analyse der Detektionsberichte, die Behebung der Schwachstellen sowie der Bereitstellung von Informationen aus dem Prozess der Detektion zur Erleichterung der Behebung von gleichen oder ähnlichen Schwachstellen in weiteren Computersystemen.

Weitere Anforderungen, die von den Autoren gestellt werden, sind die Aktualisierbarkeit der Detektionssoftware und insbesondere angewendeter Mechanismen zur Detektion, um neu entdeckte Schwachstellen ebenfalls detektieren zu können. Ein zusätzlicher Aspekt, der berücksichtigt werden muss, ist die Einrichtung eines privilegierten Zugangs auf Assets für die Detektion von Schwachstellen, um diese zu vereinfachen. Allgemein muss jedoch auch auf der anderen Seite festgelegt werden, welche Informationen eines Computersystems durch Angreifer einsehbar und nützlich für eine Kompromittierung sein könnten.

Ein wesentlicher Bestandteil der Detektion von Schwachstellen ist zudem die automatische Erstellung von Berichten, um Ergebnisse über eine gewisse Zeitspanne vergleichen zu können sowie zur Ermittlung, ob eine Schwachstelle in der Vergangenheit bereits auf einem Asset ausgenutzt wurde. Auch eine Korrelation von Daten aus verschiedenen Werkzeugen zur Detektion muss vorgenommen werden, um Zusammenhänge der Ausnutzung mehrerer Schwachstellen bzw. Assets erkennen zu können.

Eine weitere Maßnahme mit starkem Zusammenhang zum Schwachstellenmanagement wird in `SI-2` „Flaw Remediation“ beschrieben – die Identifikation, Meldung sowie Behebung von Fehlern auf Computersystemen, welche zur Dokumentation mit dem Prozess des Konfigurationsmanagements verknüpft werden soll. Eine Überprüfung von Softwareupdates, die zur Behebung von Fehlern installiert wurden, muss auf ihre Wirksamkeit sowie mögliche unerwünschte Nebeneffekte durchgeführt werden.

Auch wenn das Dokument einige Maßnahmen nennt, die wesentliche Bestandteile der Aktivitäten im Schwachstellenmanagement ausmachen, ist der Inhalt dennoch stark abstrahiert und eher vergleichbar mit einem Anforderungskatalog als mit einem Konzept zur konkreten Realisierung eines Schwachstellenmanagements. Insbesondere Rollen und Verantwortlichkeiten als auch konkrete Umsetzungen technischer Lösungen in Verbindung mit Verfahrensweisen bezüglich eines Schwachstellenmanagements fehlen hier komplett.

Als Abgleich der Anforderungen kann festgestellt werden, dass Maßnahme `IR-6 (2)` der Anforderung zum Verbinden mit weiteren sicherheitsrelevanten Informationen (`FA10`) in dieser Arbeit entspricht. Maßnahme `RA-5`, „Vulnerability Scanning“, erfüllt die Anforderungen `FB1` und `FB2` – einer automatisierten Detektion. Durch die Forderung nach der Beseitigung von Fehlern auf Computersystemen und deren Tests in Maßnahme `SI-2`, „Flaw Remediation“, wird die in dieser Arbeit gestellte Anforderung der Berücksichtigung von Risiken (`NB9`) und Überprüfung der Wirksamkeit (`NB8`) bei der Beseitigung und Abschwächung erfüllt.

4.1.3.2 Creating a Patch and Vulnerability Management Program

Im Bereich Patch- und Schwachstellenmanagement ist die inzwischen zurückgezogene *Special Publication 800-40 Version 2* – „Creating a Patch and Vulnerability Management Program“ [Pet05] eine der grundlegendsten Veröffentlichungen des NIST. Darin wird allgemein eine

systematische Herangehensweise beschrieben, wie eine Organisation ein Patch- und Schwachstellenmanagement umsetzen sollte. Demnach soll die Verantwortung dafür an eine Gruppe von Personen, der *Patch and Vulnerability Group* (PVG), übertragen werden, die mit den in diesem Bereich insgesamt elf erforderlichen Aufgaben betraut werden:

1. Erstellung eines Inventars an Ressourcen
2. Die Überwachung von Quellen für Schwachstellen sowie Behebungsmöglichkeiten (hier insbesondere Patches)
3. Priorisierung der Behebung
4. Erstellen einer Datenbank mit Behebungsmöglichkeiten
5. Testen der Maßnahmen
6. Beaufsichtigung der Behebung von Schwachstellen
7. Weitergabe von Informationen über Schwachstellen und Maßnahmen an Administratoren
8. Automatisierte Installation von Patches
9. Konfiguration einer automatischen Aktualisierung von Anwendungen
10. Überprüfung der Behebung von Schwachstellen durch Detektion aus Netz- und Systemsicht
11. Schulung von Administratoren bezüglich der Behebung von Schwachstellen

Die andere in dem Dokument erwähnte Gruppe wird durch Systemadministratoren gestellt. Diese führen nicht-automatisierte Behebungen aus und werden von der PVG unterstützt. In dem Dokument werden zu jeder der Tätigkeiten Umsetzungsvorschläge und Hinweise beschrieben, die ein Verständnis für den Umgang mit Schwachstellen fördern und im Folgenden zusammengefasst sind. Aufgrund der Einteilung des Schwachstellenmanagements in mehrere Aufgaben kann Anforderung `NA17`, die Definition von Aktivitäten, erfüllt werden.

So weisen die Autoren darauf hin, dass die Bestimmung von Assets auf einer geeigneten Abstraktionsebene, d.h. weder zu feingranular noch grobgranular (jeweils für Hardware und Software), sowie automatisiert geschehen sollte. Zudem wird eine Liste an Attributen vorgegeben, durch die ein Asset beschrieben werden kann. Auch wird darauf hingewiesen, dass alle Assets erfasst und zur Unterstützung der Behandlung von Schwachstellen priorisiert werden müssen. Systemadministratoren müssen zudem Zugriff auf die Dokumentation ihrer eigenen Systeme haben.

Bezogen auf die im vorherigen Kapitel identifizierten Anforderungen an ein Schwachstellenmanagement in Hochschulnetzen werden hier die Punkte `NA1`, `NA24` und `NA25` sowie teilweise ebenfalls `FA4` erfüllt. Anforderung `NA15` ist teilweise erfüllt, da alle Assets berücksichtigt werden – weitere Angaben über den Anwendungsbereich fehlen.

Die Überwachung von Quellen für Schwachstellen, Gegenmaßnahmen und Bedrohungen muss durch die PVG überwacht werden. Auf welche Art und Weise (d.h. manuell oder automatisch) dies geschieht wird nicht näher erläutert.

Die Priorisierung der Behebung erfolgt auf Basis der Signifikanz einer Schwachstelle in Verbindung mit einer Analyse verwundbarer Systeme. Inwiefern zum einen Schwachstellen, zum anderen Assets bewertet werden, wird nicht ausgeführt. Bei der Priorisierung spielt zudem die Lage über aktuelle Viren sowie eine Risikoabwägung bezüglich einer Installation eines Patches im Vergleich zur Akzeptanz einer Schwachstelle ohne das Treffen von Maßnahmen zur Behebung hinein. Diese Beschreibung, insbesondere einer Priorisierung, erfüllt die Anforderungen `NB11` sowie `NB12`.

Die Erstellung einer Datenbank für Gegenmaßnahmen zu einer Schwachstelle ist laut den Autoren vor allem sinnvoll in Bezug auf Assets, die nicht von Patch-Management Tools unterstützt werden. Sie enthält Anweisungen zum Entfernen von Schwachstellen und üblicherweise eine Kopie jedes Patches. Diese Maßnahme kann als Unterstützung bei der Beseitigung und Abschwächung betrachtet werden und erfüllt daher Anforderung `FB5`.

Das Testen von Gegenmaßnahmen umfasst die Überprüfung von Patches auf ihre Integrität und einem Virenscan der jeweiligen Datei, die den Patch darstellt. Tests auf negative Auswirkungen müssen laut Autoren auf Systemen außerhalb der produktiven Umgebung vorgenommen werden. Außerdem muss geprüft werden, ob es Abhängigkeiten der Patches untereinander gibt und die Software auch mit Patch korrekt läuft. Dabei muss das PVG auch Nutzererfahrungen miteinbeziehen.

Im weiteren Verlauf der Arbeit gehen die Autoren insbesondere auf Patches als Gegenmaßnahme zur Behebung einer Schwachstelle ein, darunter Hinweise zum Einsatz der Maßnahmen, welche im Rahmen eines Patch Managements automatisiert geschehen sollte und die Verteilung von Informationen über Schwachstellen und Gegenmaßnahmen an die Systemadministratoren mittels geeigneter Software des Patch-Managements oder beispielsweise auch per E-Mail. Schließlich wird in der Arbeit zudem auf Möglichkeiten der Überprüfung der Wirksamkeit von Patches eingegangen: Diese kann zum einen über eine Detektion noch offener Schwachstellen sowie der Kontrolle von „Patch Logs“ oder beispielsweise durch den direkten Versuch der Ausnutzung einer Schwachstelle, die durch einen Patch geschlossen werden sollte, erfolgen. Insofern werden hier (einschließlich des vorherigen Absatzes) die Anforderungen zur Überprüfung von Maßnahmen zur Behebung von Schwachstellen, `NB9` und `NB10` erfüllt.
Im Punkt der Detektion von Schwachstellen wird in der Arbeit grob über Möglichkeiten eingegangen, so auch über netzwerk- und systembasierte Detektion. Auch wenn diese Punkte sehr ungenau beschrieben sind, würden hier die Anforderungen `FB3` und `FB4` erfüllt werden.

Die Autoren nennen zudem Metriken für das Patch- und Schwachstellenmanagement, wodurch Anforderung `NA21` erfüllt werden kann. Diese können in drei Kategorien eingeteilt werden:

- Als erstes die Messung der **Anfälligkeit der Systeme für einen Angriff**. Darin betrachtete Metriken sind die Anzahl installierter Patches pro System, die Anzahl detektierter Schwachstellen pro System sowie die Anzahl der Dienste, die ein System zur Verfügung stellt und ein mögliches Risiko darstellen.

- Die zweite Kategorie an Metriken bezieht sich auf die **Reaktionszeit bei der Behandlung**. Die Reaktionszeit bezieht sich auf die Identifikation von Schwachstellen und Patches, d.h. wie lange es dauert, bis eine Schwachstelle bekannt ist, auf die Anwendung eines Patches und auf die Anwendung weiterer Gegenmaßnahmen.

- Die letzte Kategorie umfasst Metriken zur **Bestimmung von Kosten**, bezogen auf die PVG, die Unterstützung der Systemadministratoren und eingesetzten Softwarelösungen im Patch- und Schwachstellenmanagement.

Die Beschreibung der Umsetzung des Schwachstellenmanagements im Dokument wird in dieser Arbeit in deutlich konkreterem und umfassenderem Umfang vorgenommen, als in dem im vorherigen Abschnitt beschriebenen *SP 800-53 Revision 4*, ist jedoch teilweise nicht im Hochschulumfeld umsetzbar. Beispielsweise wird in dem Dokument für eine umsetzbare Durchführung von Tests von Maßnahmen zur Behebung von Schwachstellen empfohlen, dass die Konfiguration von Computersystemen standardisiert wird. In einem Hochschulnetz ist diese Voraussetzung alleine aufgrund des Einsatzes privater Geräte in der Praxis nicht umsetzbar.

Auf Seiten verwendbarer Aspekte der Arbeit hingegen steht allgemein die Bestimmung einer für das Schwachstellenmanagement zuständigen Gruppe. In einem Konzept müssen jedoch Verantwortlichkeiten deutlich klarer, d.h. auf Ebene von Rollen für Einzelpersonen, festgelegt sein. Auch wenn die Aktivitäten im Allgemeinen durch die elf Aufgaben der PVG gut dokumentiert sind, so beziehen sie sich jedoch tendenziell stärker auf ein Patch-Management als auf ein Schwachstellenmanagement.

Hingegen können beschriebene Metriken auch direkt für ein Schwachstellenmanagement genutzt werden, falls die übliche Maßnahme der Installation eines Patches um weitere Maßnahmen erweitert wird. So kann in einem Schwachstellenmanagement die Reaktionszeit von der Detektion einer Schwachstelle bis zum Schließen der Schwachstelle betrachtet werden.

4.1.3.3 Guide to Enterprise Patch Management Technologies

Die Special Publication 800-40 Version 2 (vorheriger Abschnitt) wurde durch NISTs *Special Publication 800-40 Revision 3* – „Guide to Enterprise Patch Management Technologies“ [Mur13] abgelöst. Darin wird die Behandlung von Schwachstellen jedoch als Teil eines Patchmanagements angesehen, da, neben beispielsweise der Funktionserweiterung von Anwendungen, eine Schwachstelle ein Grund zur Herausgabe und Installation eines Patches sein kann. Insofern nennt dieses Dokument vor allem Herausforderungen und Probleme der Behandlung von Schwachstellen durch Patches.

Laut der Veröffentlichung ist das Ziel des Patchmanagements die Identifikation, Aneignung, Installation und die Überprüfung von Patches und deckt daher nur einen Teil der im Schwachstellenmanagement auszuführenden Aktivität der **Beseitigung und Abschwächung** ab. Da im Schwachstellenmanagement zudem nur ein Teil der Behebung von Schwachstellen durch Patches erfolgt – andere beispielsweise durch die Außerbetriebnahme von Software oder die Implementierung zusätzlicher Sicherheitsmaßnahmen – ist der Bereich, den das Patchmanagement im Schwachstellenmanagement umsetzt, nochmals geringer.

Aufgrund dessen kann das Dokument, in Verbindung mit der darin eher allgemeineren, unkonkreten Beschreibung von Lösungsansätzen, ebenfalls nicht für die Implementierung eines Schwachstellenmanagements benutzt werden.

4.1.3.4 Gesamtbewertung der Special Publications 800

Die oben beschriebenen Dokumente der *Special Publications 800* mit Bezug zum Schwachstellenmanagement als Ganzes gesehen, bilden eine gute Grundlage, um ein Verständnis der Behandlung von Schwachstellen im Rahmen des Informationssicherheitsmanagements zu bekommen. Die Beschreibung der Aktivitäten im Schwachstellenmanagement gehen deutlich über die in den *BSI IT-Grundschutz-Standards* hinaus und liefern einige Hinweise, die bei der Umsetzung von Schwachstellenmanagement zu beachten sind.

4.1.4 Vulnerability Management: Tools, Challenges and Best Practices

Bezüglich des Ablaufs eines Schwachstellenmanagements hat das *SysAdmins, Networking and Security* (SANS) Institut im Jahr 2003 in seinem „Vulnerability Management: Tools, Challenges and Best Practices“ [Cat03] einen Aktivitätenzyklus veröffentlicht. Dieser bezieht sich jedoch auf eine heutzutage eingeschränkte bzw. veraltete Definition des Schwachstellenmanagments, da darin lediglich das Finden, die Bewertung und Behebung von Schwachstellen auf Servern und Arbeitsplatzrechnern beschrieben wird, jedoch weitere Gerätegruppen wie Netzkomponenten, mobile und private Geräte außer Acht gelassen werden. Bereits die Erfüllung zweier grundlegender, im vorherigen Kapitel ausgearbeiteter Anforderungen – die „Flexibilität des Anwendungsbereichs“ (`NA14`) sowie die „Zweckdienlichkeit des Anwendungsbereich“ (`NA15`) – ist daher nicht gegeben.

Insofern ist diese Arbeit ein praktisches Beispiel dafür, wie sich der Geltungsbereich von Prozessen im Laufe der Zeit ändern kann. Die Autorin schlägt sechs konkrete Aktivitäten vor, die sie im Schwachstellenmanagement sieht:

1. **Pflege eines Asset-Inventars**
   Demnach soll es ein Asset-Inventar geben, dessen Einträge über das *Change Management* aktuell gehalten werden. Insofern sind Anforderungen `NA1`, `NA6` und `NA23` erfüllt. Zudem sollen Einträge mit Hilfe eines Schemas zur Nummerierung dokumentiert werden (Anforderung `NA24`).

2. **Management des Flusses relevanter Informationen in das Unternehmen**
   Für die Informationsbeschaffung bezüglich Schwachstellen, Computerviren und -Würmern ist ein in der Organisation ansässiges CSIRT („Computer Security Incident Response Team“) verantwortlich. Da ebenfalls Informationen über Schadsoftware dokumentiert wird, verschwimmt das Schwachstellenmanagement an diesem Punkt mit dem Risikomanagement, da Schadsoftware keine Schwachstellen, sondern mehr Bedrohungen darstellen. Anforderung `NI9` wird hier jedoch teilweise erfüllt, da Schwachstellen keinen eindeutigen Identifikator bekommen, wodurch die Kommunikation erschwert wird.

3. **Bewertung des Risikos von Schwachstellen**
   Schwachstellen müssen laut der Autorin anhand von vier verschiedenen Kriterien bewertet werden: Anhand von Bedrohungen, Konsequenzen einer Ausnutzung, der Häu-

figkeit des Auftretens der Bedrohung und einer Einschätzung der Eintrittswahrscheinlichkeit. Ein konkretes Bewertungsschema ist jedoch nicht angegeben.

4. **Identifikation bzw. Detektion angreifbarer Assets**
   Neben der Aufzählung einiger Softwareprodukte zur Durchführung der Detektion (beispielsweise *nmap* oder *Nessus*) ist die Beschreibung der Durchführung relativ ungenau gehalten. Angesprochen wird die Auswahl einer zuverlässigen Lösung zur Detektion mit möglichst wenig False-Positives und möglichst geringer Auswirkung auf die Verfügbarkeit des Netzes. Vor einer Detektion auf einem Asset muss zudem die Erlaubnis des Managements (in diesem Fall im Sinne der Geschäftsleitung) eingeholt werden. Anhand der Beschreibung kann hier Anforderung `FB1` teilweise erfüllt werden. Der unerfüllte Teil bezieht sich darauf, dass in der Arbeit nicht erläutert wird, wie Verfahren zur Detektion angepasst und erweitert bzw. aktuell gehalten werden.

5. **Die Meldung und Behebung der Schwachstellen**
   Die Umsetzung der Aktivität zur Meldung und Behebung von Schwachstellen wird nicht näher ausgeführt. Vielmehr wird die Notwendigkeit einer Meldung erläutert – damit die Administratoren der Assets die Sicherheitslage ihrer Assets einschätzen können. Es wird jedoch festgelegt, dass der Fokus zur Behebung insbesondere auf Schwachstellen und Assets mit hoher Kritikalität liegen muss. Somit sind die Anforderungen `NB11` und `NB12` erfüllt.

6. **Erstellung eines Planes zur Reaktion auf Schwachstellen**
   Der Plan bezieht sich vor allem auf die Erstellung des organisatorischen Rahmens. Dabei geht es um die Einteilung von Verantwortlichkeiten zur Durchführung der genannten Aktivitäten, welche laut Autorin verständlich dokumentiert werden müssen. Des Weiteren müssen genaue Vorgehensweisen festgelegt werden. Dies entspricht den Anforderungen `NA19` und `NA20`, die in dieser Arbeit gestellt wurden. Eine konkrete Umsetzung wird jedoch nicht beschrieben, wodurch diese Anforderungen als teilweise erfüllt anzusehen sind.

Auch wenn der Anwendungsbereich eingeschränkt ist, sind diese Schritte auf den heutigen Fokus von Schwachstellenmanagement noch anwendbar.

Die beschriebenen Lösungen und „Best-Practices“ stellen jedoch allgemein mehr Hinweise bei der Umsetzung zur Verfügung jedoch keine ganzheitliche Umsetzungsmöglichkeit an sich.

4.1.5 Weitere Arbeiten

In diesem Abschnitt werden weitere Arbeiten mit Bezug zum Schwachstellenmanagement betrachtet. Insbesondere Whitepaper, die durch Hersteller von Softwarelösungen zum Schwachstellenmanagement veröffentlicht werden.

Der Großteil dieser Whitepaper befasst sich jedoch in der Regel konkret mit dem Funktionsumfang der jeweiligen, vom Hersteller angebotenen Softwarelösung. Auf diese wird in dieser Arbeit nicht eingegangen.

Ein ausführlicheres, auch den organisatorischen Rahmen betrachtendes Whitepaper ist das von **Tenable Network Security** herausgegebene „Implementierung eines effektiven

Schwachstellenmanagement-Systems“ [Ten13a], das im Jahr 2013 als Thema eines „Webinars“ [Ten13b] vorgestellt wurde. Darin werden zunächst Herausforderungen in der Umsetzung angesprochen, darunter die Vielfalt an Schwachstelle, die durch zahlreiche stetige Veränderungen des Netzes auftreten, eine unzureichende Detektion von Schwachstellen in zu großen Abständen, eine hohe Dynamik in Netzen sowie das unaufhörliche Auftauchen neuer Schwachstellen. In dem Whitepaper wird daher ein Verknüpfen eines kontinuierlichen Ansatzes mit dem Asset- und Patchmanagement sowie dem Incident und Service Request Management gefordert, um Schwachstellen effektiv zu vermeiden (Anforderung `NA23`).

Bei der Umsetzung des Schwachstellen-Managements gibt es laut Tenable Network Security mehrere Disziplinen, die berücksichtigt werden müssen:

Zunächst die Identifikation und das Management von Assets. Demnach müssen Assets in der gesamten IT-Umgebung im Netz identifiziert und Beziehungen sowie Abhängigkeiten untereinander festgestellt werden. Laut der Arbeit kann dies manuell in statischen Netzen oder durch eine Abfrage der an das Netz angeschlossenen und eine IP-Adresse zugewiesenen Geräte erfolgen. In der Arbeit wird jedoch nicht erwähnt, dass eine Identifikation anhand der IP-Adresse in der Regel nicht eindeutig ist, insbesondere bei einer dynamischen Adressvergabe.

Die nächste Disziplin ist demnach die Identifikation von Schwachstellen. Dabei sollen diese für jedes Asset identifiziert und ihre Kritikalität ermittelt werden.

Gemäß der Arbeit muss eine kontinuierliche Durchführung des Schwachstellenmanagements, insbesondere eine regelmäßige Ausführung der Detektion von Schwachstellen erfolgen. Diese Beschreibung stimmt teilweise mit Anforderung `NA18` überein, da weitere Teilaktivitäten jenseits der Detektion nicht berücksichtigt werden.

Die Risikobewertung liefert die Basis für die weitere Vorgehensweise. Hier wird vor allem der „Wert des Assets“ [Ten13a] betrachtet. Eine konkrete Vorgehensweise zur Bewertung wird nicht vorgeschlagen.

Des Weiteren soll, wie bereits erwähnt, das Schwachstellenmanagement mit einigen weiteren Managementprozessen verknüpft werden. So auch das Change Management, um mögliche Sicherheitsprobleme früher identifizieren und behandeln zu können, Patch Management zur Sicherstellung einer ordnungsgemäßen Installation von Patches, Mitigation Management generell zur Behandlung von Schwachstellen und insbesondere der Identifikation von Maßnahmen zur Behebung sowie dem Incident Management zwecks einer möglichst schnellen Reaktion auf Schwachstellen.

Dabei wird als Schlüssel zur Umsetzung die Automatisierung des Prozesses genannt (Anforderung `NA7`), wobei auch hier konkrete Umsetzungsmöglichkeiten nicht beschrieben werden.

Zusätzlich werden in dem Whitepaper verschiedene Leistungsindikatoren (KPIs) im Schwachstellenmanagement erwähnt: Die Anzahl der Schwachstellen pro Hersteller und Produkte, das Alter einer Schwachstelle (wobei nicht erwähnt wird ab welchem Startzeitpunkt, d.h. beispielsweise dem Zeitpunkt der Veröffentlichung der Schwachstelle oder der Identifikation im Schwachstellenmanagement), der Anteil auf Schwachstellen gescannter Assets und die Anzahl an Schwachstellen in einem bestimmten Zeitraum.

Jedoch ist es schwierig anhand genannter KPIs die Effektivität bzw. Effizienz einzuschätzen. Die KPIs betrachten vor allem die Identifikation von Schwachstellen, wobei Aspekte der Be-

seitigung und Abschwächung, beispielsweise die durchschnittliche Dauer von der Detektion bis zur Behebung einer Schwachstelle, komplett fehlen. Insofern wird Anforderung NA21 nur als teilweise erfüllt angesehen.

4.2 Konzepte und Lösungen

Als Kontrast zu den in den vorhergehenden Abschnitten auf ihre Übereinstimmung mit den Anforderungen überprüften Standards und „Good Practices“ setzt sich dieser Abschnitt mit verschiedenen konzeptuellen Arbeiten auseinander. Eine besondere Rolle darin nimmt jedoch die *National Vulnerability Database* (NVD) in Verbindung mit dem *Security Content Automation Protocol* (SCAP) ein. Dabei handelt es sich nicht um eine konzeptionelle Arbeit, sondern um ein „Content Repository“ dessen Inhalt sich durch eine standardisierte Beschreibung auszeichnet.

4.2.1 Concepts and Successes in Vulnerability Management

In der Arbeit [Jos04] des SANS Instituts nimmt der Autor eine Kategorisierung des Schwachstellenmanagements in Bezug auf die Vorgehensweise zur Behandlung von Bedrohungen vor und weist auf die Vorteile hin, die eine Realisierung des Schwachstellenmanagements mit sich bringen. Demnach unterscheidet der Autor bei der Methodik zwischen

- „Firefighting“ – der Reaktion als Folge einer Ausnutzung einer Schwachstelle und Wiederherstellung des Betriebs
- „Reactive“ – der Reaktion während eines durchgeführten Angriffs zur Schadensminimierung
- „Proactive“ – dem Handeln zum Zweck einer Verhinderung der Ausnutzung von Schwächen, sowie
- „Architecture“ – dem Ansetzen bei der Softwareentwicklung, um Schwachstellen in der Implementierung zu reduzieren.

Gemäß dem Autor ist ein Schwachstellenmanagement ein proaktives Handeln, das aus vier Aktivitäten besteht: Die ersten beiden sind beschrieben als das Zusammenstellen eines Inventars an Hardware-Assets und Softwarekomponenten sowie der Bewertung (eng. „assess“) durch Aufbauen einer Datenbasis an Schwachstellen und deren Abgleich mit dem im Inventar gelisteten Komponenten zur Identifikation betroffener Assets und deren Priorisierung.

Die dritte Phase bezeichnet das Abschwächen (engl. „mitigate“) der Schwachstellen durch eine Behebung auf Basis der zuvor festgelegten Priorisierung. Dazu gehört auch die Sicherstellung, dass Maßnahmen zur Abschwächung keine neuen Schwachstellen einführen. Genannte Maßnahmen zur Behebung einer Schwachstelle sind die Installation von Patches, die Außerbetriebnahme von Software oder Einschränkung des Zugriffs auf das Asset. Daher werden durch die Beschreibung die Anforderungen NB9 und NB13 erfüllt.

Die letzte Phase ist die Berichterstattung bzw. das Reporting. Dieses bezieht sich in der Arbeit jedoch lediglich auf die Berichterstattung an Personen, die „Entscheidungen innerhalb eines Unternehmens treffen“ [Jos04]; d.h. vor allem an das Top-Level-Management.

Durch Definition der Aktivitäten erfüllt das Konzept an diesem Punkt die Anforderung NA17. Der Anwendungsbereich bezieht sich wie üblich auf Softwareschwachstellen, schränkt jedoch nicht in Bezug auf bestimmte Geräte oder Systeme ein. Auch fordert das Konzept ein Asset-Inventar und eine Basis an Schwachstellen, wodurch Anforderungen NA1 und NA2 erfüllt sind.

Bezüglich einer Umsetzung eines Schwachstellenmanagements wird in der Arbeit auf die Detektion von Schwachstellen sowie eine Priorisierung eingegangen. Bei der Detektion wird auch hier zwischen den üblichen Kategorien unterschieden: Detektion aus Netz- sowie Systemsicht (Anforderungen FB3 und FB4). Während die Netzsicht vor allem grob *Penetration Testing* beschreibt, schreibt der Autor, dass der größte Unterschied zur Systemsicht unterschiedliche Rechte sind und in der Systemsicht zudem mehr Informationen bereitstehen.

Die Priorisierung erfolgt ausschließlich auf Basis der Verletzung einer der drei Schutzziele der Informationssicherheit (Vertraulichkeit, Integrität und Verfügbarkeit) in Bezug auf ein Asset und einer Einschätzung („Ja“ bzw. „Nein“) des Einflusses einer Bedrohung auf das Asset. In diesem Ansatz werden jedoch wichtige Charakteristika von Schwachstellen – die Wahrscheinlichkeit einer Ausnutzung sowie die einzeln resultierende Auswirkung der Ausnutzung der Schwachstelle auf die Ziele in der Informationssicherheit – komplett außer Acht gelassen, wodurch dieser Ansatz wenig über die Ausnutzung einer Schwachstelle auf einem Asset aussagt, sondern mehr eine Bewertung von Assets darstellt. Des Weiteren ist kein Bewertungsschema in Form von konkreten Werten oder Vorgehensweisen angegeben, wodurch eine Priorisierung nicht ersichtlich ist.

Insofern wird das Bewertungskonzept nicht in Bezug auf Schwachstellen anerkannt, da es mehr die Kritikalität von Assets beschreibt. Folglich erfüllt dieses Konzept nur die Anforderung zur Berücksichtigung der Umgebung (NK4) und die Kritikalität der Assets (Anforderung NK5). In Kombination mit der durchaus unkonkreten Beschreibung kann die Arbeit nicht als Konzept zur Umsetzung eines Schwachstellenmanagements verwendet werden.

4.2.2 Implementing a Vulnerability Management Process

In der Arbeit [Tom13] mit dem Titel „Implementing a Vulnerability Management Process“ beschreibt der Autor konzeptionell die Umsetzung eines Schwachstellenmanagements. Schwachstellenmanagement an sich wird demnach als die Identifikation von Schwachstellen und Risiken und eine darauf aufbauende Behebung bzw. Akzeptanz der Schwachstellen definiert. Da diese Definition die Aktivitäten im Schwachstellenmanagement festlegt, ist Anforderung NA17 erfüllt.
Als Ziel des Schwachstellenmanagements nennt der Autor die Detektion in ausreichend kurzen Abständen und rechtzeitige Behebung von Schwachstellen. Somit ist auch Anforderung NB8 erfüllt.

Im Prozess sind schließlich vier verschiedene Rollen vorgesehen:

- Der **Security Officer**, der die Verantwortung über den Gesamtprozess innehat sowie für die Planung und Umsetzung verantwortlich ist,
- der **Vulnerability Engineer**, dessen Verantwortung in der Konfiguration der Schwachstellen-Scanner und zeitlichen Planung der Ausführung eines Scans liegt,

- der **Asset Owner**, dessen Verantwortlichkeit in der Entscheidung über eine Behebung von Schwachstellen auf seinem Asset liegt sowie
- der **IT System Engineer**, welcher für die Implementierung von Maßnahmen zur Behebung von Schwachstellen zuständig ist.

Durch die konkrete Festlegung von Rollen und Verantwortlichkeiten, kann die Anforderung `NA20` aus dem letzten Kapitel erfüllt werden.

Im Schwachstellenmanagement-Prozesses beschreibt der Autor fünf verschiedene Phasen:

- Vorbereitung („Preparation“)
- Schwachstellen-Scan („Vulnerability scan“)
- Definition von Maßnahmen zur Behebung („Define remediating actions“)
- Umsetzung der Behebungsmaßnahmen („Implement remediating actions“)
- Kontrollprüfung („Rescan“)

In der **Vorbereitungs**-Phase soll demnach der Anwendungsbereich durch den Security Officer festgelegt werden. Darin enthalten sind die Computersysteme, die im Schwachstellenmanagement als Assets behandelt werden. Auch wird darin die Art der Detektion festgelegt. In der Arbeit wird jedoch ausschließlich die Detektion über das Netz betrachtet – die Festlegung im Anwendungsbereich bezieht sich auf Scans von außerhalb des Netzes sowie Scans von innerhalb. Insofern ist hier lediglich die Anforderung der Detektion aus Netzsicht erfüllt (`FB4`). Aufgrund des flexiblen Anwendungsbereichs wird zudem Anforderung `NA15` erfüllt.

Nach Auswahl der Assets folgt das Informieren der Asset Owner und die Planung der Durchführung von Scans. Auch diesbezüglich kann der Anwendungsbereich die betrachteten Schwachstellen einschränken und beispielsweise Schwachstellen mit geringem Risiko ignorieren. Insofern ist Anforderung `NK6`, die Definition eines Kriteriums zur Akzeptanz von Schwachstellen erfüllt.

Die Phase des **Schwachstellen-Scans** dient insbesondere zur Erstellung von Berichten an Asset Owner über vorhandene Schwachstellen auf ihren Assets, generellere Berichte an den Security Officer und Berichte mit technischen Informationen über Schwachstellen an den IT System Engineer. Für die Durchführung ist der Vulnerability Engineer zuständig.

Die Phase zur **Definition von Maßnahmen zur Behebung** adressiert an die Analyse von Schwachstellen und Risiken durch – mit Ausnahme des Vulnerability Engineers – alle oben genannten Rollen. Schließlich ist es der Asset Owner, der Maßnahmen zur Behebung von Schwachstellen erstellt und Informationen an den IT System Engineer weitergibt, damit dieser die Maßnahme umsetzen kann. Die Maßnahmen sind insofern nicht auf einen bestimmten Typ beschränkt (z.B. Patch), wodurch Anforderung `NB13` erfüllt ist.

In der Phase der **Umsetzung der Behebungsmaßnahmen** werden diese entsprechend angewendet. Die Behebung erfolgt in dafür vorgesehenen Zeitfenstern und wird durch den Security Officer verfolgt.

Die letzte Phase, die **Kontrollprüfung** beinhaltet die Anwendung desselben Verfahrens zur Detektion der jeweiligen Schwachstelle zur Kontrolle auf eine wirksame Behebung der Schwachstelle (Anforderung `NB9`). Die Entscheidung über die Wirksamkeit wird insbesondere durch den Security Officer getroffen, welcher im negativen Fall die Behebung veranlasst.

Aufgrund der Beschreibung von Verfahren ist zudem Anforderung `NA19` sowie der Beschreibung von Meldewegen zudem `NA27` erfüllt. Durch die starke Abhängigkeit der Abläufe in jeder Phase von der zentralen Rolle des Security Officers scheint das Konzept jedoch nur auf einen eher klein gefassten Anwendungsbereich anwendbar zu sein. Bei potenziell mehreren tausend Assets kann der Aufwand zur Entscheidung in einzelnen Phasen nicht zentral durch eine Person bewältigt werden. Im Falle einer Aufteilung des in diesem Konzept beschriebenen Schwachstellenmanagements auf verschiedene Bereich innerhalb eines Hochschulnetzes, beispielsweise durch eine jeweils separate Umsetzung pro Institut, fehlt wiederum eine übergeordnete Instanz zum Herstellen eines möglichen netzweiten Zusammenhangs.

4.2.3 National Vulnerability Database (NVD) und SCAP

Einen praktischen Ansatz bei der Unterstützung von Schwachstellenmanagement bietet die *National Vulnerability Database*, eine durch das NIST angebotene Dokumentation von Schwachstellen, beschrieben auf Basis der CVE, CWE sowie Checklisteneinträgen der CPE und OVAL. Allgemein basiert die NVD auf Inhalten, die konform zum *Security Content Automation Protocol* (SCAP) sind.

4.2.3.1 Security Content Automation Protocol (SCAP)

Das *Security Content Automation Protocol* (SCAP) umfasst mehrere Spezifikationen zur Benennung und Kommunikation von Schwachstellen und allgemein sicherheitsrelevanten Informationen. Diese Spezifikationen gliedern sich in verschiedene Sprachen, Formate, Auflistungen, Bewertungssysteme sowie eines Modells zum Erhalt der Integrität. Die für die Kommunikation von Schwachstellen relevantesten Spezifikationen umfassen das *Extensible Configuration Checklist Description Format* (XCCDF), die *Open Vulnerability and Assessment Language* (OVAL), die *Common Platform Enumeration* (CPE), *Common Configuration Enumeration* (CCE) sowie CVE und das CVSS.

Das Ziel, das durch die mit Hilfe von SCAP umsetzbare Standardisierung der Kommunikation sicherheitsrelevanter Informationen erreicht werden soll, ist eine darauf aufbauende Automatisierbarkeit von Aktivitäten – beispielsweise der Überwachung der Konfiguration und Bestimmung des Zustands eines Systems unter sicherheitsrelevanten Aspekten sowie des Erhalts der Konformität zu Maßnahmen der Informationssicherheit, wie sie in 4.1.3.1 beschrieben sind. [Ste12]

Oben genannte Elemente mit besonderer Relevanz zum Schwachstellenmanagement werden im Folgenden beschrieben.

Das **Extensible Configuration Checklist Description Format** ist eine Sprache zur „Erstellung von sicherheitsrelevanten Checklisten, Benchmarks und damit in Verbindung stehenden Dokumenten“ [Nat14], wodurch das Ziel der Automatisierbarkeit der Definition

und Durchführung von Tests mit Bezug zur Informationssicherheit sowie von Konfigurationsanalysen verfolgt wird. Diese Möglichkeiten umfassen beispielsweise die Sicherstellung der Konformität von Systemen zu Richtlinien, ein einheitliches Management von sicherheitsrelevanten Tests und Audits sowie – aufgrund der standardisierten Darstellung – die Zusammenstellung von Regeln innerhalb einer Checkliste oder einem Benchmark aus verschiedenen Quellen. [Dav12].

Die genaue Beschreibung der XCCDF Spezifikation wird in [Dav12] behandelt.

Die Bedeutung der **Open Vulnerability and Assessment Language** (OVAL) bei der Behandlung von Schwachstellen resultiert aus ihrer Möglichkeit der Standardisierung des Bewertungsprozesses. Dies umfasst das *OVAL Definition* Schema, zur Beschreibung eines Systemzustands, das *OVAL System Characteristics* Schema zur Beschreibung von Informationen über ein System und das *OVAL Results* Schema, um die Ergebnisse einer Bewertung darzustellen. Dadurch wird OVAL als Standard in einem weiten Feld an Produkten – auch Software zur Unterstützung von Schwachstellenmanagement – akzeptiert und integriert. So kann mit Hilfe von in OVAL beschriebenen Tests beispielsweise anhand betroffener Betriebssysteme und Softwareversionen überprüft werden, ob eine Schwachstelle in einem Softwareprodukt oder der Konfiguration sowie ein geeigneter Patch auf einem System vorhanden ist [MIT14a]. Diese Möglichkeit kann in einem Konzept genutzt werden, um Verfahren zur Detektion zu erstellen. Im *OVAL Repository*, einem zentralen Anlaufpunkt für Mitglieder der *OVAL Community*, stellt die *MITRE Corporation* eine öffentliche Quelle an OVAL Definitionen dieser Tests zur Verfügung. [MIT14e]

Die **Common Vulnerabilities and Exposures** (CVE) Auflistung ist eine der umfangreichsten Listen von Schwachstellen überhaupt. Aktuell (Stand September 2015) umfasst der Umfang der frei nutzbaren und auf der Website der CVE zur Verfügung gestellten Datensammlung mehr als 80.000 Einträge. Zu beachten ist, dass nicht alle der 80.000 Einträge vollständige Dokumentationen von Schwachstellen beinhalten, sondern darin ebenfalls reservierte sowie abgelehnte Einträge enthalten sind.

Der Inhalt der CVE und der Aufbau der einzelnen Schwachstelleneinträge wird ausführlich in Abschnitt 2.3.3 beschrieben.

Des Weiteren enthält SCAP neben der Standardisierung von Schwachstellen, Schwächen und Plattformen ebenso eine standardisierte Auflistung von Aussagen über Konfigurationen mit Bezug zur Informationssicherheit in der **Common Configuration Enumeration** (CCE).

Ein Eintrag der CCE setzt sich aus einer eindeutigen CCE-ID, einer für Menschen lesbaren Beschreibung, möglichen Parametern zur Konfiguration, technischen Möglichkeiten zur Implementierung sowie Referenzen – beispielsweise auf Sicherheitsrichtlinien oder weitere Informationen – zusammen. Die Beschreibung ist dabei möglichst allgemein gehalten, wie am Beispiel des Eintrages mit der CCE-ID „CCE-10090-9“ in Tabelle 4.1 zu sehen ist. [MIT12]

Die **Common Weakness Enumeration** (CWE) ist, ähnlich der CVE, eine Auflistung von Schwächen in Softwareprodukten mit standardisierter Beschreibung. Auf diese Weise kann die CWE als Möglichkeit zu Klassifizierung von Schwachstellen nach Art der Schwäche genutzt werden.

Abkürzung	Charakteristikum
CCE ID	CCE-10090-9
CCE Description	The 'Do not allow passwords to be saved' setting should be configured correctly.
CCE Parameter	enabled/disabled
CCE Technical Mechanisms	(1) GPO: Computer Configuration\Administrative Templates\Windows Components\Remote Desktop Services\Remote Desktop Session Host\Security\Do not allow passwords to be saved (2) Registry Key: `HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows NT\Terminal Services\DisablePasswordSaving`

Tabelle 4.1: Beispielbeschreibung des CCE-Eintrages mit ID CCE-10090-9 (entnommen aus `http://cce.mitre.org/lists/data/downloads/cce-win7-5.20120521.xls`)

Inhalte eines Eintrags in der CWE umfassen beispielsweise einen eindeutigen Identifikator, eine Beschreibung der Schwäche, Beschreibung einer Ausnutzungsmöglichkeit, die Wahrscheinlichkeit und mögliche Maßnahmen zur Abschwächung einer Ausnutzung, die Anzahl der Einträge in der CVE vom Typ der Schwäche und weitere Attribute. [MIT14f]

Analog zur Standardisierung des Formats von Schwachstellen in der CVE sowie der Beschreibung von Konfigurationen durch die CCE, dient die **Common Platform Enumeration** (CPE) zur Standardisierung des Formats zur Beschreibung von Plattformen, Geräten sowie Softwareanwendungen. [MIT13]
Das durch die CPE definierte Beschreibungsschema umfasst unter anderem den Typ des beschriebenen Objekts (Softwareanwendung, Betriebssystem, Gerät), den Hersteller sowie die Produktbezeichnung und Version, das installierte Update und sechs weitere Felder. Die genaue Spezifikation findet sich in [Bra11].

Das **Common Vulnerability Scoring System** (CVSS) ist ein weit verbreitetes, gut dokumentiertes und transparentes Bewertungssystem für Schwachstellen. Eine detailierte Beschreibung des CVSS in der Version 2 und 3 ist in Abschnitt 2.3.4 zu finden.

4.2.3.2 National Vulnerability Database

Als „Content Repository“ von Inhalten des SCAP basiert die NVD zunächst vor allem auf den Daten der CVE und nutzt zur Identifikation entsprechend genauso die darin enthaltenen Informationen sowie das in der CVE definierte Namensschema für Identifikatoren. Das Hauptmerkmal, das die NVD von der CVE unterscheidet, ist die Bereitstellung ergänzender, im vorherigen Abschnitt erläuterter Informationen aus SCAP sowie zusätzlicher Funktionen. [Nat15c]

Wie am Beispiel des Eintrages des Heartbleed Bugs in der NVD zu sehen ist (vgl. Abbildung 4.1), ist neben generellen Informationen wie dem Datum der Veröffentlichung, der

letzten vorgenommenen Revision, Quelle der Schwachstellenmeldung und der zur CVE identischen Beschreibung, zusätzlich die Bewertung des BaseScores im CVSS vorhanden. Des Weiteren sind die einzelnen Base Metrics in Form des bereits in Abschnitt 2.3.4 gezeigten Vektorstrings sowie in ausgeschriebener Form einsehbar. Zum aktuellen Stand (November 2015) verwendet die NVD noch die Version 2 des CVSS.

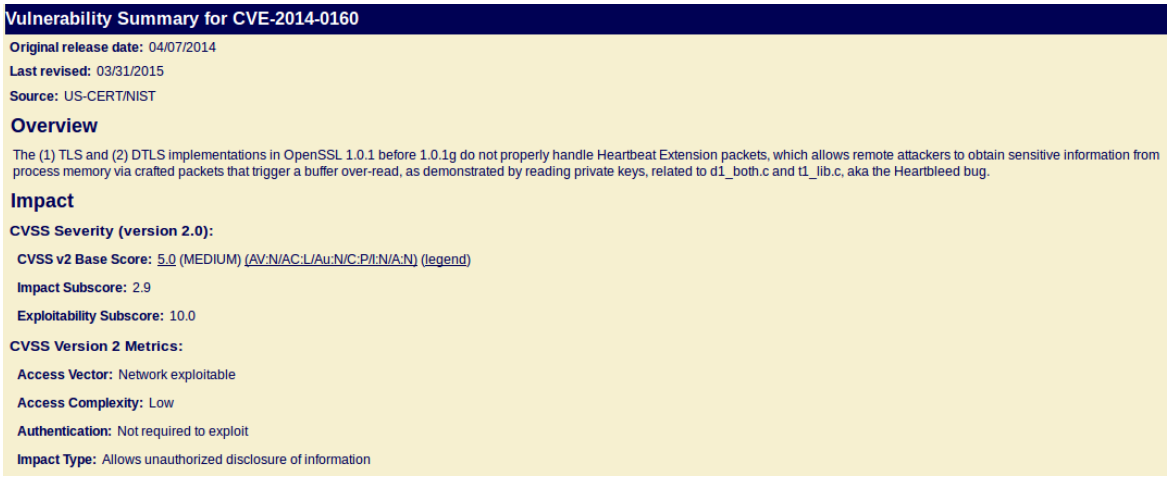


Abbildung 4.1: Teil der Schwachstellenbeschreibung des Heartbleed Bugs in der NVD, welche unter anderem die Base-Metriken des CVSSv2 zeigen [Nat15d]

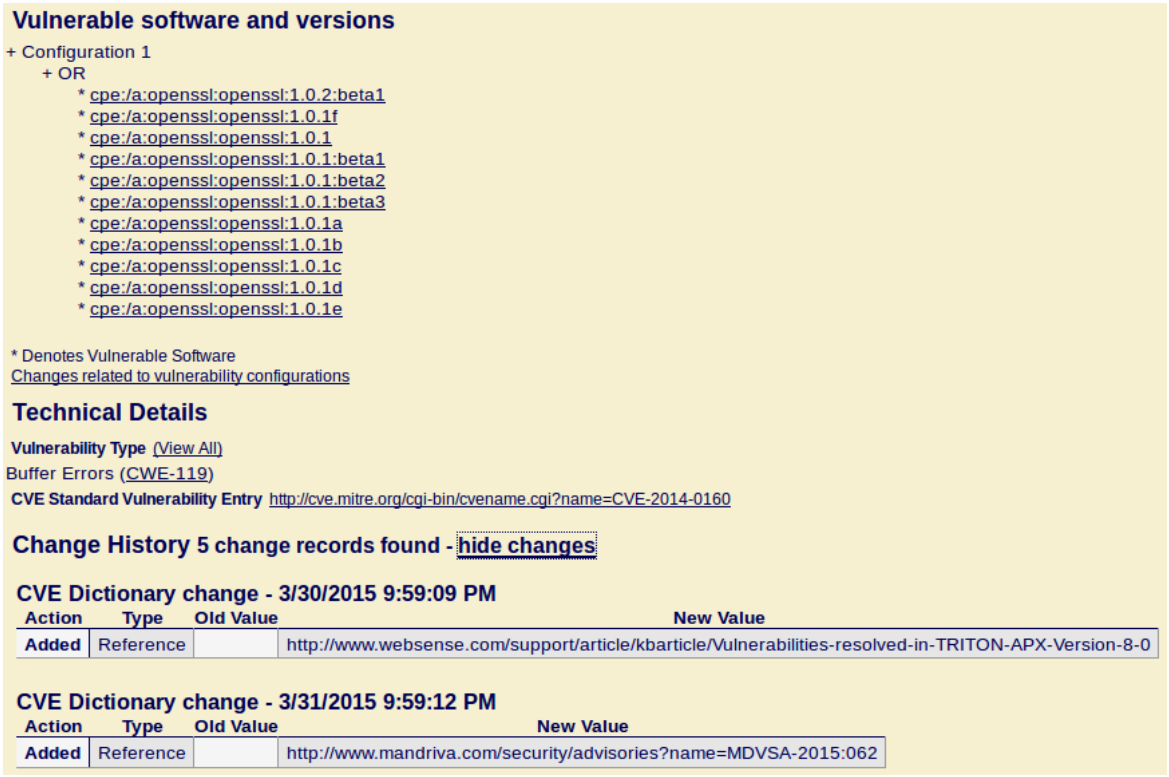
Weitere Angaben, die bereits in Abschnitt 2.3.3 erwähnt wurden, sind insbesondere eine Verknüpfung mit der CPE zur Erleichterung der Identifikation (bzw. einer Möglichkeit der automatisierten Verarbeitung) betroffener Softwareprodukte und Softwareversionen. Die relevanten Einträge aus der CPE sind in Form von OVAL Checks mittels logischen *Und*- bzw. *Oder*-Operatoren verknüpft. Ein Beispiel wird in Abbildung 4.2 gezeigt. Darin aufgelistet sind die verschiedenen, vom Heartbleed Bug betroffenen Versionen von *OpenSSL*, welche mit einem logischen *Oder*-Operator verknüpft sind.

Wie des Weiteren in Abbildung 4.2 zu sehen ist, klassifiziert die NVD eine Schwachstelle anhand von Einträgen der CWE. In diesem Fall wird der Heartbleed Bug als „Buffer Error“ bezeichnet.

4.2.3.3 Bewertung NVD und SCAP

Die NVD in Verbindung mit SCAP ist zunächst kein vollständiges Konzept für die Umsetzung eines Schwachstellenmanagements im Hochschulnetz und kann daher nicht komplett unter den im vorherigen Kapitel anhand der Szenarien aufgestellten Anforderungen an ein solches bewertet werden. Jedoch bieten die NVD und SCAP im Gegensatz zu den weitestgehend abstrahierten Ansätzen der vorherigen Arbeiten eine konkrete Umsetzung zumindest von Teilaspekten im Schwachstellenmanagement.

Da die NVD eine Schwachstellen-Datenbank ist und zentral angeboten wird, erfüllt sie die Anforderung `NA2`. Der Inhalt der NVD wird allgemein durch Schwachstellen plattformunabhängiger Softwareprodukte dargestellt. Insofern ist Anforderung `NA9` teilweise erfüllt, da



Vulnerable software and versions

+ Configuration 1
  + OR
    * cpe:/a:openssl:openssl:1.0.2:beta1
    * cpe:/a:openssl:openssl:1.0.1f
    * cpe:/a:openssl:openssl:1.0.1
    * cpe:/a:openssl:openssl:1.0.1:beta1
    * cpe:/a:openssl:openssl:1.0.1:beta2
    * cpe:/a:openssl:openssl:1.0.1:beta3
    * cpe:/a:openssl:openssl:1.0.1a
    * cpe:/a:openssl:openssl:1.0.1b
    * cpe:/a:openssl:openssl:1.0.1c
    * cpe:/a:openssl:openssl:1.0.1d
    * cpe:/a:openssl:openssl:1.0.1e

* Denotes Vulnerable Software
Changes related to vulnerability configurations

Technical Details

Vulnerability Type (View All)
Buffer Errors (CWE-119)
CVE Standard Vulnerability Entry http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-0160

Change History 5 change records found - hide changes

CVE Dictionary change - 3/30/2015 9:59:09 PM

Action	Type	Old Value	New Value
Added	Reference		http://www.websense.com/support/article/kbarticle/Vulnerabilities-resolved-in-TRITON-APX-Version-8-0

CVE Dictionary change - 3/31/2015 9:59:12 PM

Action	Type	Old Value	New Value
Added	Reference		http://www.mandriva.com/security/advisories?name=MDVSA-2015:062

Abbildung 4.2: Teil der Schwachstellenbeschreibung des Heartbleed Bugs in der NVD, welche Informationen aus der CPE, CWE und eine Änderungshistorie zeigen [Nat15d]

die Aktivität der Beseitigung und Abschwächung sowie der Prävention nicht berücksichtigt ist.

Der Zugriff auf die Einträge in der NVD ist über eine durch eine Webschnittstelle (`https://web.nvd.nist.gov/view/vuln/search`) realisierte Suchfunktion möglich. Die Einträge können jedoch auch direkt über eine URL aufgerufen werden, wobei der Eintrag durch die jeweilige CVE-ID in den Parametern auswählbar ist. Am Beispiel des Heartbleed Bugs beschreibt die URL `https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2014-0160` den entsprechenden Eintrag. Insofern sind die Anforderungen `FI2` und `FI4` erfüllt. Auch ist somit die Anforderung einer geeigneten Präsentation der Daten `NA12` erfüllt.

Die Aktualität der Einträge entspricht der Aktualität der CVE. Dadurch, dass viele Softwarehersteller als *CVE Numbering Authority* (CNA) fungieren [MIT15c] (vgl. Abschnitt 2.3.3) und generell Schwachstellen durch Nutzer weltweit gemeldet werden, ist die Aktualität der CVE gegeben. Insbesondere da einige große Softwarehersteller sowie CERTs als *CVE Numbering Authority* (CNA) fungieren (vgl. Abschnitt 2.3.3), ist die fortlaufende Pflege der CVE gewährleistet. Insofern erfüllt die NVD ebenfalls Anforderung `NI8`. In eingeschränktem Umfang ist es zudem möglich, Einträge manuell hinzuzufügen (`FI1`), beispielsweise über die Meldung an eine CNA oder Veröffentlichung einer gefundenen Schwachstelle. Eine Garantie auf Eintragung gibt es jedoch nicht, wodurch die Anforderung nur teilweise erfüllt wird.

Da bei der NVD keine weiteren manuellen Tätigkeiten durch die Rollen im Schwachstellenmanagement im Hochschulnetz zum Erhalt der Aktualität der Schwachstelleninformationen

notwendig sind, kann das Kriterium einer automatisierten Aktualität (FI5) erfüllt werden. Aufgrund der Verwendung des Identifikators und der Beschreibung aus der CVE, welche beide vorgegeben sind, können auch die Anforderungen NI9 und NI10 ebenfalls erfüllt werden.

Aus der Aktivität der Klassifikation erfüllt die NVD in Verbindung mit SCAP den Aspekt der Automatisierung (Anforderung FK1), da wie bei der Identifikation keine manuelle Bearbeitung notwendig ist. Die Einheitlichkeit und Qualität der Schwachstellenbewertung (Anforderung NK3) ist trotz Schwächen des CVSS Version 2 (siehe Abschnitt 2.3.4) erfüllt, da anhand des in der NVD bereitgestellten BaseScores eine Priorisierung effektiv gewährleistet ist.

Bezüglich der Aktivität der Beseitigung und Abschwächung bietet der von der NVD angebotene Informationsumfang eine Teilerfüllung der Anfoderung FB1, der Existenz von Verfahren zur Detektion: Durch die in Form von OVAL Checks aufgelisteten Einträge der CPE ist es möglich, eine Schwachstelle anhand betroffener Softwareprodukte zu detektieren.

Die Anforderung zur Unterstützung manueller Tätigkeiten NA8 kann ebenfalls als teilweise angesehen werden, da zu jeder Schwachstelle in der Regel Referenzen angegeben sind, welche nicht selten auch Maßnahmen zur Behebung beinhalten. Da der Inhalt der Referenzen nicht erweiterbar ist, ist die Anforderung nicht komplett erfüllt.

4.2.3.4 Ähnliche Lösungen

Datenbanken für Schwachstellen gibt es in verschiedenen Ausführungen, die im Prinzip der NVD in einigen Aspekten sehr ähnlich sind. Die NVD bietet im Gegensatz zu den meisten anderen Datenbanken jedoch den Vorteil, dass die in ihr gehaltenen Schwachstellen einem Standard folgend einheitlich und transparent beschrieben sind.

Üblicherweise halten die meisten Software-Hersteller selbst jeweils einen Bestand dokumentierter Schwachstellen ihrer Softwareprodukte, so beispielsweise auch Microsoft in seinem *Microsoft Security Bulletins* [Mic15], in der Einträge von Schwachstellen prinzipiell sehr ähnlich zu denen in der NVD aufgebaut sind. Schwachstellen besitzen ebenfalls einen quellenspezifischen Identifikator, eine Beschreibung, eine Bewertung [Mic11] sowie eine Auflistung betroffener Softwareprodukte. Dabei wird, falls vorhanden, ebenfalls auf einen Patch verwiesen, der die Schwachstelle schließt.

Eine weitere ähnliche Lösung wird beispielsweise in Form der *Open Source Vulnerability Database* (OSVDB) [Jak15a] angeboten. In dieser wird jeder Schwachstelle ebenfalls eine eindeutige ID zugewiesen sowie eine Bewertung mittels BaseScores des CVSS Version 2.

Der Name der Datenbank leitet sich dabei nicht aus der Art der dokumentierten Schwachstellen ab, sondern aus den Quellen, durch die an Informationen von Schwachstellen gelangt wird: Frei zugänglichen sowie nutzbaren Quellen. Beispiele für derartige Quellen sind freie Mailinglisten mit dem Hintergrund der Informationssicherheit (beispielsweise die *Full Disclosure Mailingliste* [Gor15a]) oder auch Meldungen, die durch Softwarehersteller selbst veröffentlicht werden. [Jak15b]

Ein Nachteil im Gegensatz zur NVD ist die Darstellung der betroffenen Softwareprodukte. Diese werden lediglich in der Beschreibung der Schwachstelle in Freitextformat, und nicht beispielsweise gemäß CPE-Einträgen in einem standardisiertem, maschinenlesbaren Format, angegeben.

Durch den ähnlichen Aufbau der Schwachstellen-Datenbanken fällt die Erfüllung der Anforderungen immer relativ ähnlich aus. Solche Lösungen bieten keinerlei managementspezifische Angaben über Rollen, Verantwortlichkeiten und Verfahren und wirken beispielsweise in der Detektion durch die von ihnen zur Verfügung gestellten Informationen eher unterstützend als ausführend.

Auch wenn der Grad an Erfüllung der Anforderungen in der Regel durch derartige Schwachstellen-Datenbanken sehr gering ist (vgl. vorheriger Abschnitt) und diese keine Umsetzung eines Schwachstellenmanagements darstellen, so ist es dennoch sinnvoll, die darin angebotenen Informationen zu nutzen. So können diese mittels technischer Umsetzungen zum Import der Daten verwendet werden, um trotz der hohen Anzahl täglich identifizierter Schwachstellen diese in ein Schwachstellenmanagement zu integrieren.

4.3 Wissenschaftliche Arbeiten

In der Veröffentlichung „Supporting Vulnerability Awareness in Autonomic Networks and Systems with OVAL“ [BBF11] beschreiben die Autoren eine Möglichkeit zur Vermeidung von Schwachstellen in autonomen Systemen bzw. Netzen. Die darin beschriebene Lösung setzt am Zeitpunkt des Einbringens der Schwachstellen im Netz an – bei Änderung der Systemkonfiguration. Dabei wird unter Verwendung von *CFEngine*, einer Agenten-basierten Softwarelösung zur „Verwaltung und Konfiguration von Computernetzen“ [ram06], durch Regelsätze sichergestellt, dass die Konfiguration der Systeme in keinen ungünstigen Zustand gerät. Der Kern der Arbeit handelt von einer automatischen Generierung von Regel für CFEngine. Zu diesem Zweck werden diese aus Schwachstelleninformationen, beschrieben durch OVAL und bezogen aus dem *OVAL Repository*, übersetzt und dem *CFEngine Server* im Netz zur Verfügung gestellt. Über diesen werden die Regeln an die Agenten verteilt und mit der Konfiguration abgeglichen.

In [WG09] stellen die Autoren eine Ontologie zur strukturierten Beschreibung einer Schwachstelle vor. Diese stellt die Zusammenhänge unter anderem von Schwachstellen zu Softwareprodukten, Angreifern, der Angriffsmethodik, den Folgen eines Angriffs und Gegemaßnahmen dar, um eine bessere Unterstützung der Kommunikation von Softwarelösungen zu erreichen. Das dabei erstellte Datenschema kann ebenfalls als Grundlage bzw. Übersicht für das im Konzept zu entwickelnde Datenschema dienen.

Die inzwischen nicht mehr aktuelle Arbeit [THSC04] behandelt die Automatisierung von Schwachstellenmanagement über ein entwickeltes Schema zur Repräsentation von Schwachstellen auf Basis von XML sowie der Verwendung von Web-Diensten. Das Datenschema besteht aus vier verschiedenen Teilbereichen: Der Beurteilung einer Schwachstelle, der Möglichkeit zur Überprüfung auf Betroffenheit von einer Schwachstelle, die Behebung und Ausnutzung einer Schwachstelle.

Zur Automatisierung haben die Autoren einen Rahmen entwickelt, dessen Kern eine Schwachstellen-Datenbank darstellt und Inhalt gemäß dem vorher erläuterten Repräsentationsschema formatiert ist. Daraus können Informationen über Schwachstellen von Clients abgerufen werden. Eine weitere Komponente, der „System Manager“ fragt Informationen bezüglich des Softwareinventars von Clients ab und sendet diese an einen weiteren

Web-Dienst „Scanner“. Dieser liefert nach Abgleich mit der Schwachstellen-Datenbank alle Schwachstellen zurück, von denen der Client betroffen ist. Der „System Manager“ führt daraufhin automatisch Maßnahmen zur Behebung der Schwachstellen aus. Als Beispiele sind das Herunterladen von Patches oder Anzeigen einer Meldung angeben, wobei nähere Informationen über die Vorgehensweise fehlen. Zudem existiert ein Web-Dienst zur Ausführung von Penetration-Tests.

Die Anzahl öffentlicher, verwendbarer wissenschaftlicher Arbeiten zum Schwachstellenmanagement ist allgemein sehr gering. Eine Gemeinsamkeit zu den in den vorherigen Abschnitten betrachteten Arbeiten ist jedoch die starke Abstraktion und fehlende oder ungenaue Beschreibung einer Umsetzungsmöglichkeit.

4.4 Zusammenfassung der erfüllten Anforderungen

In diesem Abschnitt wird eine zur Übersichtlichkeit vorhandener Konzepte und Lösungen beitragende tabellarische Zusammenfassung der Erfüllung der im vorherigen Kapitel gestellten Anforderungen an ein Schwachstellenmanagement gezeigt (siehe Tabelle 4.2).

Zur leichteren Unterscheidbarkeit der Gewichtung einer Anforderung, sind „empfohlene“ Anforderungen kursiv hervorgehoben.

Die in der ersten Zeile genannten Abkürzungen entsprechen den im folgenden beschriebenen Arbeiten:

NIST	NIST Special Publications 800
K1	Vulnerability Management: Tools, Challenges and Best Practices
K2	Concepts and Successes in Vulnerability Management
K3	Implementing a Vulnerability Maangement Process
NVD	National Vulnerability Database und SCAP

Da der Standard *ISO/IEC 27001* und der *BSI IT-Grundschutz*, aufgrund der darin jeweils sehr abstrakten Beschreibung des Umgangs mit Schwachstellen, die in dieser Arbeit gestellten Anforderungen nur in einem sehr geringem Umfang erfüllen, werden diese nicht aufgelistet. Genauso und aus gleichem Grund die durch Hersteller von technischen Schwachstellenmanagementsystemen veröffentlichten „Good Practices“ aus Abschnitt 4.1.5 sowie wissenschaftliche Arbeiten aus dem vorherigen Abschnitt.

Der Grad der Erfüllung wird durch drei verschiedene Zustände ausgedrückt: Zum einen erfüllt (✓), teilweise erfüllt (+) und nicht erfüllt (✗).

		NIST	K1	K2	K3	NVD
Allgemeine Anforderungen						
NA1	Existenz eines Asset-Inventars	✓	✓	✓	✗	✗
NA2	Existenz und Zentralisierung einer Schwachstellen-Dokumentation	✗	✗	✓	✗	✓
NA3	*Zentralisierung des Asset-Inventars*	✗	✗	✗	✗	✗
FA4	Funktionsumfang des Asset-Inventars	+	✗	✗	✗	✗
FA5	*Zugriff auf Informationen und Funktionen über eine API*	✗	✗	✗	✗	✗
FA6	Aktualität des Asset-Inventars	✗	✓	✗	✗	✗
NA7	Automatisierung der Aktivitäten	✗	✗	✗	✗	✗
NA8	Unterstützung manueller Tätigkeiten	✗	✗	✗	✗	+
NA9	Plattformunabhängigkeit der Aktivitäten	✗	+	✗	✗	+
FA10	*Verbinden mit weiteren sicherheitsrelevanten Informationen*	✓	✗	✗	✗	✗
FA11	*Exportierbarkeit der Daten*	✗	✗	✗	✗	✗
NA12	*Präsentation der Daten*	✗	✗	✗	✗	✓
NA13	Mandantenfähigkeit	✗	✗	✗	✗	✗
NA14	Institutsübergreifende Benutzerverwaltung	✗	✗	✗	✗	✗
NA15	Flexibilität des Anwendungsbereichs	+	✗	✗	✓	✗
NA16	Zweckdienlichkeit des Anwendungsbereichs	✗	✗	✗	✗	✗
NA17	Definition der Aktivitäten	✓	✗	✓	✓	✗
NA18	Ausübungsintervall des Aktivitätenzyklus	✗	✗	✗	✗	✗
NA19	Definition von Verfahren	✗	+	✗	✓	✗
NA20	Definition von Rollen und Verantwortlichkeiten	✗	+	✗	✓	✗
NA21	Prüfung durch Leistungsindikatoren	✓	✗	✗	✗	✗

NA22	Dezentralisierung der Aktivitäten	✗	✗	✗	✗	✗
NA23	Schnittstellen zu anderen Prozessen	✗	✓	✗	✗	✗
NA24	Identifikation von Assets	✓	✓	✗	✗	✗
NA25	Qualität der Asset-Beschreibung	✓	✗	✗	✗	✗
NA26	Definition eines Asset-Verantwortlichen	✗	✗	✗	✗	✗
NA27	Nutzung geeigneter Meldewege	✗	✗	✗	✓	✗
FA28	Erstellung von Berichten	✗	✗	✗	✓	✗
Identifikation						
FI1	(Manuelles) Hinzufügen neuer Einträge in die Schwachstellen-Dokumentation	✗	✗	✗	✗	+
FI2	Zugriff auf Einträge in der Schwachstellen-Dokumentation	✗	✗	✗	✗	✓
FI3	Editieren von Einträgen in der Schwachstellen-Dokumentation	✗	✗	✗	✗	✗
FI4	Suchfunktionalität innerhalb der Schwachstellen-Dokumentation	✗	✗	✗	✗	✓
FI5	Automatischer Erhalt der Aktualität der Schwachstellen-Dokumentation	✗	✗	✗	✗	✓
NI6	Dynamische Festlegung der Quellen der Schwachstellen-Dokumentation	✗	✗	✗	✗	✗
FI7	Filtern relevanter Schwachstellen	✗	✗	✗	✗	✗
NI8	Aktualität der Schwachstellen-Dokumentation	✗	✗	✗	✗	✓
NI9	Identifikation und Kommunikation von Schwachstellen	+	✗	✗	✗	✓
NI10	Qualität der Schwachstellenbeschreibung	✗	✗	✗	✗	✓
Klassifikation						
FK1	Automatisierung der Klassifikation	✗	✗	✗	✗	✓
FK2	Manuelle Klassifikation	✗	✗	✗	✗	✗

NK3	Einheitlichkeit und Qualität der Schwachstellenbewertung	✗	✗	✗	✗	✓
NK4	Berücksichtigung der Umgebung	✗	✗	✗	✗	✗
NK5	Kritikalität von Assets	✗	✗	✗	✗	✗
NK6	Kriterium der Schwachstellenakzeptanz	✗	✗	✗	✓	✗
Beseitigung und Abschwächung						
FB1	Verfahren zur Detektion von Schwachstellen	✓	+	✗	✗	+
FB2	Automatisierung der Detektion	✓	✗	✗	✗	✗
FB3	*Detektion aus Systemsicht*	✓	✗	✓	✗	✗
FB4	*Detektion aus Netzsicht*	✓	✗	✓	✓	✗
FB5	Unterstützung bei der Behebung durch Computersysteme	✓	✗	✗	✗	✗
NB6	*Zentralisierung der Methoden und Daten der Detektion*	✗	✗	✗	✗	✗
NB7	*Definition von Maschinen zur Detektion*	✗	✗	✗	✗	✗
NB8	Zeitnahe Beseitigung und Abschwächung	✓	✗	✗	✓	✗
NB9	Prüfung einer Maßnahme auf Wirksamkeit	✓	✗	✓	✓	✗
NB10	Berücksichtigung von Risiken	✓	✗	✗	✗	✗
NB11	Priorisierung kritischer Schwachstellen	✓	✓	✗	✗	✗
NB12	Priorisierung kritischer Assets	✓	✓	✗	✗	✗
NB13	Möglichkeit einer effektiven Behebung	✗	✗	✓	✓	✗
Prävention						
NP1	Einsatz von Software unter Berücksichtigung der Risiken durch Schwachstellen	✗	✗	✗	✗	✗

Tabelle 4.2: Erfüllung der Anforderung durch die ausgewählten Lösungen

5 Konzept eines Schwachstellenmanagements in Hochschulnetzen

In diesem Kapitel wird ein Konzept für ein Schwachstellenmanagement beschrieben, das auf die im bisherigen Verlauf der Arbeit genannten Herausforderungen in Hochschulnetzen angepasst ist und die daraus abgeleiteten Anforderungen in einem derartigen Umfeld erfüllt.

In dem Konzept ist die Ausführung jeder der vorher im Abschnitt 2.2 definierten Aktivitäten im Schwachstellenmanagement vorgesehen, welche auch, unter anderem, grob die Struktur dieses Kapitels vorgeben. Zunächst wird jedoch in Abschnitt 5.1, der genaue Anwendungsbereich festgelegt, den das entwickelte Konzept abdeckt. Weitere essentielle Managementkomponenten bilden die darin befindlichen Verantwortlichkeiten der einzelnen Rollen, welche in Abschnitt 5.2 beschrieben werden, sowie genaue Verfahren und Abläufe, die durch einen in Abschnitt 5.4 erläuterten – und als organisatorisches Kernelement dienenden – Aktivitätenzyklus verdeutlicht werden. Der Inhalt der einzelnen Aktivitäten wird in den Abschnitten 5.5 bis 5.8 bestimmt und ist jeweils in essentielle Teilaktivitäten aufgeteilt. In Abschnitt 5.9 wird ein nebenläufiger Prozess zur kontinuierlichen Verbesserung des Schwachstellenmanagements beschrieben.

Im Konzept wird dabei nicht nur auf organisatorische Aspekte eingegangen, sondern genauso technische Umsetzungsmöglichkeiten innerhalb der einzelnen Aktivitäten betrachtet. Eine Beschreibung technischer Komponenten im hier konzipierten Schwachstellenmanagement findet sich in Abschnitt 5.3.

Varianten der Umsetzung und potenzielle Abweichungen vom Konzept werden in Abschnitt 5.10 erläutert.

Das Ziel des Schwachstellenmanagements ist umgebungsunabhängig in Abschnitt 2.2 definiert: Die systematische Reduzierung von Schwachstellen auf den im Anwendungsbereich definierten Computersystemen zur Erreichung einer höheren Netzsicherheit. Die Hochschulumgebung wirkt sich insofern nicht auf die Zielsetzung des Schwachstellenmanagements aus, sondern vor allem auf die Umsetzung der darin betrachteten Aktivitäten, Rollen und Verantwortlichkeiten sowie Verfahrensweisen.

Da das Ziel der Sicherung des Netzes insbesondere vom Netzbetreiber, etwa dem Rechenzentrum angestrebt wird, steht es im Mittelpunkt dieses Konzeptes. Daher wird davon ausgegangen, dass die netzweite Umsetzung des Schwachstellenmanagements vom Rechenzentrum ausgehen muss und die Umsetzung beim Kunden auf dem vom Rechenzentrum realisierten Schwachstellenmanagement aufbaut.

Die Überprüfung auf Erfüllung des Konzepts der in Kapitel 3 gestellten Anforderungen an ein Schwachstellenmanagement in Hochschulnetzen wird in Abschnitt 5.11 vorgenommen.

5.1 Der Anwendungsbereich des Schwachstellenmanagements

Der Anwendungsbereich von Managementprozessen beschreibt den Inhalt und Umfang der Umsetzung des Managements. Dieser kann nach Organisation und Abteilung, jedoch auch nach anderen organisatorischen und technischen Aspekten oder je nach bestimmten Typen technischer Komponenten ausgewählt werden.

Das in diesem Kapitel ausgearbeitete Konzept beschreibt die Anwendung eines Schwachstellenmanagements durch die Miteinbeziehung jeder der im IT-Service-Management vorhandenen Rollen:

- Dem IT-Dienstleister – das Hochschulrechenzentrum,
- den Kunden – Personen und Einrichtungen, die IT-Dienstleistungen durch das Hochschulrechenzentrum in Anspruch nehmen, sowie
- den Nutzern – Personen, die den in Anspruch genommenen Dienst nutzen; hier vor allem Wissenschaftler, Mitarbeiter der Institutionen und Studenten.

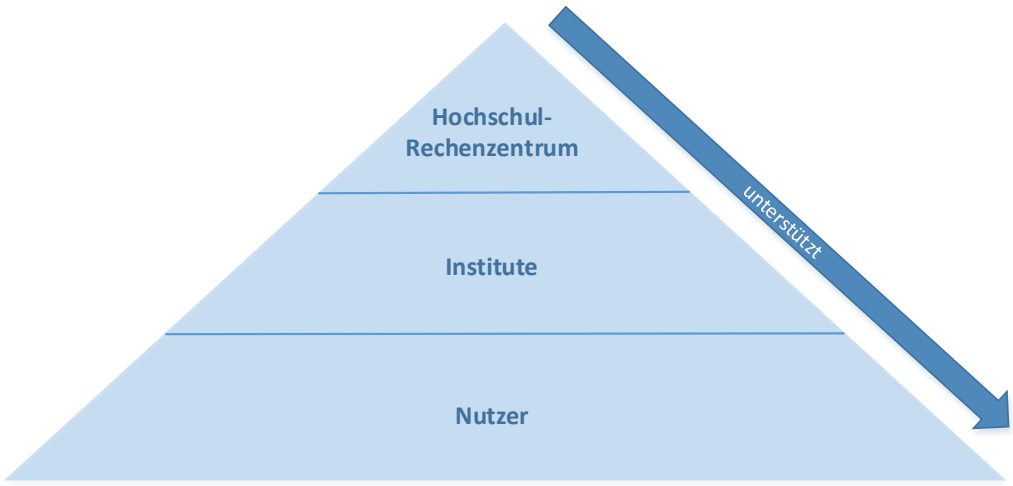


Abbildung 5.1: Hierarchisches Unterstützungsschema in Bezug auf Organisationen und Personen im Hochschulnetz bei der Realisierung des Schwachstellenmanagements.

Im Gesamtbild des Schwachstellenmanagements ergibt sich auf diese Weise eine Hierarchie der bestimmten Verantwortlichkeiten, worin das Hochschulrechenzentrum bzw. der Netzbetreiber als Wurzel die generellste Stelle bildet, gefolgt von den jeweiligen Instituten und auf unterster Ebene die Nutzer der jeweiligen Institute. Das Hochschulrechenzentrum unterstützt entsprechend die Institute und Nutzer bei der Behandlung von Schwachstellen (vgl. Abbildung 5.1).

Das Konzept kann zudem unabhängig vom Typ der als Assets ausgewählten Computersysteme angewendet werden; das heißt im Fokus liegen beispielsweise sowohl Server, Arbeitsplatzrechner, private und mobile Geräte sowie Drucker, als auch Netzkomponenten wie

Router, Switches, Gateways und beliebige weitere Computersysteme. Genauso wird der Anwendungsbereich des Konzepts in Bezug auf die betrachtete Software nicht eingeschränkt, sondern kann unabhängig von Hersteller, Produkt, Version und Betriebssystem eingesetzt werden. Lediglich die Ausrichtung der betrachteten Schwachstellen ist auf Schwachstellen in Software sowie der Konfiguration festgelegt (Softwareschwachstellen und Fehlkonfiguration).

Das Schwachstellenmanagement wird auch in Bezug auf bestimmte Netzbereiche so wenig wie möglich eingeschränkt. Ziel ist es, die Aktivitäten im gesamten Hochschulnetz anzuwenden, um die Netzsicherheit nicht nur in bestimmten Bereichen, sondern im gesamten Hochschulnetz zu verbessern.

Die Definition des Anwendungsbereichs für dieses Konzept dient in erster Linie zur Übersicht des Umfangs des Schwachstellenmanagements, der durch das Konzept abgedeckt wird. Bei der Realisierung eines Schwachstellenmanagements in Hochschulnetzen gilt jedoch weiterhin, dass der Anwendungsbereich flexibel eingeschränkt oder ausgeweitet werden kann. Varianten der Umsetzung des Schwachstellenmanagements in Hochschulnetzen sind in Abschnitt 5.10 beschrieben.

5.2 Rollen und Verantwortlichkeiten

In einem Schwachstellenmanagement gibt es zwei Komponenten, bei denen eine Bestimmung von Rollen sinnvoll ist: Zum einen müssen organisatorische Rollen für Personen vorgesehen sein, die ihren Verantwortlichkeiten entsprechend Aufgaben ausführen, und zum anderen müssen – davon unabhängig – technische Rollen für Computersysteme bestimmt werden, um beispielsweise ihren Zweck innerhalb des Schwachstellenmanagements zu beschreiben oder deren Kritikalität festzulegen.

Wie bereits im vorhergehenden Abschnitt beschrieben, schränkt das Konzept das Schwachstellenmanagement nicht auf eine der im vorhergehenden Abschnitt genannten Rollen im IT-Service-Management ein. Um eine klare Struktur in die Verantwortlichkeiten und Verfahrensweisen zu bekommen, reicht die Granularität dieser Rollen jedoch nicht aus, sondern kann lediglich als Einteilung des Zuständigkeitsbereichs angesehen werden, um zum einen eine übersichtliche Struktur der Rollen erstellen zu können, als auch die Aktivitäten separat für jede der Zielgruppen festzulegen. Die Notwendigkeit für letzteres besteht aus dem Grund, da das Schwachstellenmanagement im Endeffekt beim Kunden und den Nutzern durch den IT-Dienstleister unterstützt werden muss bzw. die Möglichkeiten der Maßnahmenergreifung und Kontrolle durch den IT-Dienstleister im Vergleich zu einer rechenzentrumsinternen Lösung stark eingeschränkt sind (vgl. Abschnitt 2.5).

> Der Verantwortungs- und **Zuständigkeitsbereich des Rechenzentrums** innerhalb des Schwachstellenmanagement ist im Kern durch alle zum Rechenzentrum gehörigen und durch das Rechenzentrum administrierte Computersysteme festgelegt. Als Dienstanbieter besteht seine Zuständigkeit außerdem darin, die Kunden und Nutzer durch das Angebot geeigneter Dienste bei der Realisierung eines Schwachstellenmanagements in deren Verantwortungsbereich zu unterstützen und einen hochschulnetzweiten Zusammenhang der Situation in Bezug auf Schwachstellen herzustellen.

Da, wie in Abschnitt 2.4.2 beschrieben, die Zuständigkeit des Rechenzentrums oft „an der Datendose“ des Kunden endet, ist der Einflussbereich des Rechenzentrums begrenzt. Das heißt folglich, dass der organisatorische Teil der Umsetzung eines Schwachstellenmanagements im gesamten Hochschulnetz zwar zum größten Teil, jedoch nicht alleine vom Rechenzentrum abhängig ist. Das Rechenzentrum kann die Anwendung beim Kunden unterstützen und größtenteils organisieren, letztendlich ist jedoch der Kunde bei Computersystemen, die nicht explizit durch das Rechenzentrum administriert werden, selbst verantwortlich.

> Der **Zuständigkeitsbereich des Kunden** (in Form einer Institution oder Person) ist durch die von ihm administrierten Computersysteme vorgegeben. Außerdem gilt der Kunde als Anlaufstelle der in seinem Bereich tätigen Nutzer bei der Unterstützung des Schwachstellenmanagements.

Zu beachten ist, dass der Anwendungsbereich, den der Kunde innerhalb seiner Einrichtung festlegt, von dem vom Rechenzentrum definierten Anwendungsbereich abweichen kann, ohne dass das Rechenzentrum Einfluss darauf ausüben kann.

Beispielsweise könnte ein Kunde nicht alle von ihm administrierten Computersysteme in seinem Schwachstellenmanagement betrachten, sondern nur Server auf Linux-Basis. Auch muss er nicht alle Aktivitäten des Schwachstellenmanagements notwendigerweise umsetzen, sondern könnte beispielsweise identifizierte und klassifizierte Schwachstelleninformationen, die von Dienstleister angeboten werden nutzen und auf seinen Assets selbst lediglich die Beseitigung und Abschwächung durchführen.

Aus diesem Grund wird in diesem Konzept davon ausgegangen, dass Anwender in Bezug auf Computersysteme und den im Schwachstellenmanagement ausgeübten Aktivitäten keinerlei Einschränkungen machen, da sich das Konzept im Nachhinein leicht auf einen kleineren Anwendungsbereich begrenzen lässt.

Die Nutzer von IT-Diensten im Hochschulnetz sind in der Regel Angehörige der Institutionen, also insbesondere wissenschaftliche Mitarbeiter und Studenten sowie Mitarbeiter zur Unterstützung des Betriebs der Einrichtungen.

> Der **Zuständigkeitsbereich der Nutzer** in Bezug auf das Schwachstellenmanagement ist abhängig von der ihm darin zugeteilten Rolle. Allgemein ist der Nutzer jedoch für eine Gerätegruppe zuständig, auf die Institutionen in der Regel keinen Einfluss haben und die trotzdem einen erheblichen Teil des Netzes ausmachen – jeweils eigene, private Geräte. Nutzer erfüllen außerdem abhängig von ihrer Rolle im Schwachstellenmanagement verschiedene Aufgaben (siehe folgende Abschnitte).

Nachdem der allgemeine Zuständigkeitsbereich der drei übergeordneten Rollen bei der Erbringung und Inanspruchnahme eines IT-Dienstes festgelegt ist, werden einzelne Rollen bei der Umsetzung eines Schwachstellenmanagements definiert, welche die einzelnen Aufgaben ausführen. Die in den folgenden Abschnitten definierten Rollen dienen insbesondere der Unterscheidung der Tätigkeiten und müssen nicht notwendigerweise auf jeweils verschiedene Personen verteilt werden.

Ein wichtiger Aspekt, der so umgesetzt werden kann ist die Funktionstrennung (engl. „Segregation of Duties“), um auch daraus entstehende Risiken wie böswillige Handlungen zu vermeiden.

Es ist jedoch genauso möglich und sinnvoll, Personen mehrere Rollen zuzuteilen oder beispielsweise im Falle einer Einschränkung des Umfangs des Schwachstellenmanagements, bestimmte Rollen und damit verbundene Tätigkeiten nicht in der Umsetzung vorzusehen.

Zu beachten ist auch, dass allgemein – abgesehen von der Rolle des Chief Vulnerability Managers, des Schwachstellenmanagers sowie der Quellenverantwortlichen – jede Rolle beliebig oft institutionsübergreifend im gesamten Hochschulnetz vergeben werden kann. So kann die Umsetzung der in den folgenden Abschnitten beschriebenen Aktivitäten und Teilaktivitäten auf möglichst viele Personen verteilt und von allen Personen genutzt werden (beispielsweise Informationen über Schwachstellen), wodurch auch Institute mit wenig Ressourcen die Möglichkeit haben, ein vollständiges Schwachstellenmanagement umzusetzen.

5.2.1 Aktivitätenübergreifende Rollen

Rollen im Schwachstellenmanagement, deren Verantwortungsbereich sich über mehrere Aktivitäten hinaus erstreckt, haben insbesondere verwaltende und koordinierende Aufgaben zu erfüllen. Eine Ausnahme davon bildet die Rolle des *Asset-Verantwortlichen*, dem insbesondere unterstützende, jedoch nicht weniger essentielle Aufgaben zukommen, sowie des *Quellenverantwortlichen*, die beide durchaus technische Tätigkeiten ausführen.

5.2.1.1 Chief Vulnerability Manager

Der *Chief Vulnerability Manager* – angelehnt an die Rolle des *Chief Information Security Officers* – verfügt im Hochschulnetz über die generellste organisatorische Sicht und den Überblick über die Verbreitung und den Grad der Umsetzung des Schwachstellenmanagements. Er ist entsprechend für den Dienst des Schwachstellenmanagements, allgemein für den auf Schwachstellen bezogenen Zustand des Gesamtnetzes und der Einschätzung daraus resultierender Zusammenhänge für die Informationssicherheit verantwortlich. Informationen aus den einzelnen Instituten werden allgemein über die dort jeweils tätigen Schwachstellenmanager an den Chief Vulnerability Manager gemeldet.

Die Rolle des Chief Vulnerability Manager wird hochschulnetzweit genau einer Person zugeordnet, wobei es aufgrund der Zentralität des netzbetreibenden Hochschulrechenzentrums und ihm als Ausgangspunkt des Schwachstellenmanagements sinnvoll ist, diese Rolle darin zu vergeben.

Zu den Aufgaben des Chief Vulnerability Managers gehören organisatorische Tätigkeiten zur Verwaltung globaler Strukturen. Insbesondere ist er so als übergeordnete Instanz für die Führung von globalen White- und Blacklisten von relevanten Softwareprodukten, welche im Verlauf des Kapitels näher erläutert werden, verantwortlich. Auch ist es die einzige Rolle mit der Zuständigkeit zur Festlegung eines Wertes der *Temporären Dringlichkeit* für Schwachstellen (siehe Abschnitt 5.6.1.1), Standardwerten für das Updateintervall von Fremdquellen und dem Detektionsintervall von Assets sowie dem *Schwachstellenakzeptanzkriterium* (Abschnitt

5.6.4). Darüber hinaus fällt die Prüfung der Eignung von Abbildungen von Bewertungssystemen auf das hochschulnetzweite Bewertungssystem (Abschnitt 5.6.1.2), welche durch Quellenverantwortliche entwickelt werden, in seinen Verantwortungsbereich.

Zudem gibt er bezüglich der Implementierung von *Detektionsverfahren* (siehe Abschnitt 5.3.1.1) verwendbare Skript- und Programmiersprachen hochschulnetzweit vor. Beispielsweise kann er festlegen, dass Implementierungen ausschließlich in Perl realisiert werden. Dazu gehört auch die Festlegung der Laufzeitumgebungen und Software (z.B. *nmap* und *OpenVAS*), die von Detektionsverfahren auf Detektionssystemen genutzt werden können (vgl. Abschnitt 5.6.4).

5.2.1.2 Schwachstellenmanager

Die Anzahl der im Hochschulnetz erforderlichen *Schwachstellenmanager* ist direkt abhängig von der Anzahl der das Schwachstellenmanagement umsetzenden Institute bzw. unabhängigen organisatorischen Einheiten. In jedem Institut gibt es entsprechend genau einen Schwachstellenmanager, dessen Verantwortlichkeit in der Verteilung der weiteren Rollen – mit Ausnahme der Rolle des Chief Vulnerability Managers sowie des Quellenverantwortlichen – liegt. Des Weiteren legt der Schwachstellenmanager den Anwendungsbereich und somit die relevanten Softwareprodukte, Schwachstellen, Fremdquellen und Assets für das ihm angehörige Institut fest und ist dafür verantwortlich, den aktuellen Zustand über offene Schwachstellen sowie Leistungsparameter im Schwachstellenmanagement in seinem Bereich (idR. das jeweilige Institut) zu kennen und aufgrund dessen eine kontinuierliche Verbesserung (Abschnitt 5.9) des Managements vorzunehmen. Der aktuelle Zustand wird in regelmäßigen, eher größeren Abständen (z.B. halbjährlich) an den Chief Vulnerability Manager gemeldet.

Auch stellt der Schwachstellenmanager sicher, dass die in dem ihm angehörigen Institut umgesetzte Variante des Schwachstellenmanagements zweckdienlich ist und funktioniert, und nimmt bei Bedarf Anpassungen am Anwendungsbereich oder der Rollenverteilung vor.

5.2.1.3 Asset-Verantwortlicher

Die Verwaltung von Assets ist im Management allgemein ein zentraler Punkt. Damit ein sinnvolles Schwachstellenmanagement realisiert werden kann, ist es daher notwendig, dass eine aktuelle Übersicht über vorhandene Assets gepflegt wird. Diese Aufgabe kommt den *Asset-Verantwortlichen* zu, welche zum einen dafür zuständig sind, dass Assets ausreichend dokumentiert und inventarisiert werden und zum anderen dass die darüber gehalteten Informationen aktuell sind (siehe Abschnitt 5.5.4). Weiterhin ist der Asset-Verantwortliche für die Dokumentation einer angemessenen Einschätzung der Kritikalität seines Assets zuständig (Abschnitt 5.6.3) und unterstützt den Behebungsverantwortlichen bei der Behebung von Schwachstellen auf seinen Systemen. In der Regel ist der Asset-Verantwortliche gleichzusetzen mit der im Management üblichen Rolle des *Asset-Owners*.

5.2.1.4 Quellenverantwortlicher

Im Schwachstellenmanagement gibt es einen oder mehrere *Quellenverantwortliche*, welche ausschließlich, genau wie die Schwachstellen-Dokumentation (vgl Abschnitt 5.3.1), dem Hochschulrechenzentrum angehören. Er ist für das Hinzufügen, Implementieren, für Anpassungen

sowie das Entfernen von Fremdquellen zum Import von Schwachstelleninformationen in der Schwachstellen-Dokumentation verantwortlich (siehe Abschnitt 5.5.1).

Der Aspekt, der den Quellenverantwortlichen aktivitätenübergreifend macht, ist seine Verantwortlichkeit zur Entwicklung von anwendbaren Abbildungen der Bewertungssysteme in Fremdquellen auf das im Hochschulnetz benutzte Bewertungssystem (Abschnitt 5.6.1.2).

5.2.1.5 Nutzer

Die Rolle des *Nutzers* ist eine an alle Nutzer des Hochschulnetzes vergebene Rolle. Mit dieser sind standardmäßig festgelegte Berechtigungen wie beispielsweise die Einsicht von Informationen über Schwachstellen und weitere beliebige, vom Dienstleister zur Verfügung gestellte Dienste verbunden.

5.2.2 Rollen in der Identifikation

Die Rollen in der Identifikation von Schwachstellen sind entsprechend den groben – für diesen Zweck notwendigen – Schritten gewählt. Zum einen gibt es die Rolle des *Schwachstellen-Melders*, durch welchen Schwachstellen bekannt werden, und zum anderen eine Rolle zur Ersterfassung und allgemeiner Dokumentation der Schwachstellen, dem *Schwachstellen-Dokumentator* sowie zur Verifikation gemeldeter Schwachstellen in Form des *Schwachstellen-Prüfers*.

5.2.2.1 Schwachstellen-Melder

Die Rolle des *Schwachstellen-Melders* unterscheidet sich in einem Aspekt von den anderen Rollen im Schwachstellenmanagement: Schwachstellen-Melder werden nicht explizit durch die einzelnen Einrichtungen ernannt, sondern Personen bekommen diese Rolle implizit zugewiesen, indem sie die Existenz von bzw. Informationen zu Schwachstellen in einem Softwareprodukt melden (Abschnitt 5.5.2). Prinzipiell kann also jedes Mitglied des Hochschulnetzes die Rolle des Schwachstellenmelders annehmen.

5.2.2.2 Schwachstellen-Dokumentator

Die Hauptaufgabe des *Schwachstellen-Dokumentators* liegt in der Dokumentation neuer Schwachstellen. Dazu nehmen Personen in dieser Rolle Meldungen von Schwachstellen-Meldern entgegen, führen für jede Meldung eine Plausibilitätprüfung durch und dokumentieren die jeweilige Schwachstelle (Abschnitt 5.5.2). Dabei müssen sie zudem kontrollieren, ob der Informationsgehalt für eine Dokumentation ausreichend ist und im Zweifelsfall weitere Informationen anfordern oder selbst einholen.

5.2.2.3 Schwachstellen-Prüfer

Der *Schwachstellen-Prüfer* beurteilt die Existenz einer Schwachstelle in Softwareprodukten. Dazu ist es in der Regel notwendig, dass Personen, die diese Rolle einnehmen, ein gutes technisches Verständnis sowie Grundlagenwissen über Schwachstellen besitzen, um diese in der Praxis nachvollziehen zu können. Die Prüfung einer Schwachstelle ist in Abschnitt 5.5.3 beschrieben.

Pro jeweiligem betriebenen Dienst im Hochschulnetz gibt es mindestens einen dafür zuständigen Schwachstellen-Prüfer, welcher den Dienst kennt. So wird sichergestellt, dass Schwachstellen, die explizit mit Bezug auf die Implementierung oder Konfiguration einer der Dienste gemeldet werden, von Personen überprüft werden können, die notwendige Kenntnisse sowie Befugnisse zur Überprüfung haben.

5.2.3 Rollen in der Klassifikation

Die Klassifikation von Schwachstellen kann anhand mehrerer Kriterien vorgenommen werden: Zum einen ist ein essentieller Bestandteil der Klassifikation die Bewertung anhand eines geeigneten Bewertungsschemas, zum anderen können Schwachstellen anhand der von ihnen betroffenen Softwareprodukte kategorisiert werden. Insofern ergeben sich zwei sinnvolle Rollen in der Aktivität der Klassifikation – die Rolle des *Schwachstellen-Bewerters* und die Rolle des *Kategorisierungsverantwortlichen*. Die Kategorisierung hat insbesondere praktische Gründe und trägt zur Erhöhung der Effizienz im Umgang mit Schwachstellen bei, worauf im Abschnitt 5.6.2 näher eingegangen wird.

5.2.3.1 Schwachstellen-Bewerter

Die manuelle Klassifikation von Schwachstellen wird in Form von Personen in der Rolle des *Schwachstellen-Bewerters* durchgeführt. Dieser muss außerdem zunächst feststellen, ob die dokumentierten Informationen über die jeweiligen zu bewertenden Schwachstellen für eine Bewertung ausreichen. Die Beschreibung der Teilaktivität der Bewertung von Schwachstellen ist in Abschnitt 5.6.1 erläutert.

5.2.3.2 Kategorisierungsverantwortlicher

Eine weitere Aufgabe, die in die Klassifikation von Schwachstellen eingeordnet wird, ist die Kategorisierung. Diese betrifft insbesondere die Extraktion relevanter Informationen aus der Beschreibung einer Schwachstelle und wird durch den *Kategorisierungsverantwortlichen* vorgenommen. Derartige Attribute einer Schwachstelle umfassen insbesondere den Hersteller des von der Schwachstelle betroffenen Softwareprodukts sowie das Softwareprodukt selbst inklusive betroffener Versionen (siehe Abschnitt 5.6.2).

Auch können Kategorisierungsverantwortliche bei assetspezifischen Schwachstellen – das heißt Schwachstellen, die allgemein weder auf alle Assets angewendet noch nach Softwareprodukten kategorisiert werden können (siehe Beispiel in Abschnitt 7.5) – eine Schwachstelle einem bestimmten Asset zuweisen.

5.2.4 Rollen in der Beseitigung und Abschwächung

Die Beseitigung und Abschwächung setzt sich grob aus den Teilaktivitäten der Detektion, dessen Erfüllung in der Verantwortung des *Detektionsverantwortlichen* und der Behebung, dessen Erfüllung in der Verantwortung des *Behebungsverantwortlichen* liegt, zusammen, zudem die Kontrollprüfung von Schwachstellen auf Assets. Die Kontrollprüfung entspricht einer erneuten automatischen oder manuellen Detektion und ist insofern durch den Detektionsverantwortlichen abgedeckt.

5.2.4.1 Detektionsverantwortlicher

Die Aufgabe bzw. Verantwortlichkeit des *Detektionsverantwortlichen* besteht in erster Linie aus der Ermöglichung der Detektierbarkeit von Schwachstellen und deren Freigabe zur Gewährleistung der Sicherheit (Abschnitt 5.7.1), sowie der anschließenden Anwendung bzw. Ausführung von Verfahren zur Detektion von Schwachstellen auf den Assets, für die er verantwortlich ist (Abschnitt 5.7.2).

Da jedem Asset mindestens ein Detektionsverantwortlicher zugewiesen ist, ist genau einer davon zur Definition klarer Verantwortlichkeiten pro Asset als „hauptverantwortlich“ ausgezeichnet, welcher neben der Durchführungsverantwortung gleichzeitig im Falle einer Nicht-Durchführung rechenschaftspflichtig ist. Die Eigenschaft der *Hauptverantwortlichkeit* kann durch den Schwachstellenmanager des gleichen Instituts bei Bedarf an einen anderen Detektionsverantwortlichen übertragen werden. Dies ist sinnvoll, falls beispielsweise geplante oder spontane längere Abwesenheiten (z.B. durch Urlaub oder im Krankheitsfall) von Personen in dieser Rolle absehbar sind.

Hauptverantwortliche Detektionsverantwortliche können zudem für ihre Assets die Zeitintervalle festlegen, in denen eine automatische Detektion von Schwachstellen auf ihren Assets durchgeführt wird.

Im Schwachstellenmanagement haben Detektionsverantwortliche zudem die Möglichkeit, Schwachstellentickets (siehe Abschnitt 5.3.1.1) manuell zu öffnen, beispielsweise für kritische, jedoch nicht-detektierbare Schwachstellen. Dies gilt jedoch nur in Bezug auf Assets, für die sie verantwortlich sind. Zudem führen sie falls notwendig eine manuelle Kontrollprüfung und Eskalation für nicht-automatisch detektierbare Schwachstellen durch.

5.2.4.2 Behebungsverantwortlicher

Die Aufgabe des *Behebungsverantwortlichen* besteht in der Erstellung, Dokumentation und Anwendung von Maßnahmen zur Beseitigung von Schwachstellen auf Assets. Dazu gehört auch die Kontrolle auf Wirksamkeit der durchgeführten Maßnahmen sowie die Feststellung von bzw. Kontrolle auf unerwünschte Nebeneffekte, die durch die Behebung auftreten (Abschnitt 5.7.3).

Ebenso wie bei Detektionsverantwortlichen ist auch genau ein Behebungsverantwortlicher pro Asset als „hauptverantwortlich“ ausgezeichnet und entsprechend rechenschaftspflichtig für die Beseitigung und Abschwächung von Schwachstellen auf seinem Asset. Die Eigenschaft *hauptverantwortlich* kann ebenfalls durch einen Schwachstellenmanager des gleichen Instituts bei Bedarf an einen anderen Behebungsverantwortlichen des Assets übertragen werden.

Es ist sinnvoll, die Anzahl der Behebungsverantwortlichen (als auch Detektionsverantwortlichen) pro Asset auf den jeweiligen Bedarf und Wartungsaufwand anzupassen, d.h. weder zu vielen noch zu wenigen Personen diese Rolle und Verantwortlichkeit zuzuweisen. Beispielsweise benötigt ein Hochleistungsrechner (z.B. SuperMUC) in der Regel mehr Wartungspersonal als ein Arbeitsplatz-PC.

Die jeweiligen Behebungsverantwortlichen eines Assets müssen für die Beseitigung oder Abschwächung von Schwachstellen dafür notwendiges Fachwissen und technische Möglichkeiten mitbringen sowie entsprechende Berechtigungen haben, um ihre Aufgaben effektiv erfüllen zu können.

5.2.5 Rollen in der Prävention

In der Prävention gibt es ausschließlich die Rolle des *Präventionsverantwortlichen*, dessen Verantwortungsbereich die Ausarbeitung und Ergreifung von Maßnahmen zur Prävention von Schwachstellen umfasst (vgl. Abschnitt 5.8).

5.3 Technische Komponenten im Schwachstellenmanagement

Technische Komponenten und Funktionen bilden in den Anforderungen, welche in Kapitel 3 an ein Schwachstellenmanagement in Hochschulnetzen getroffen wurden, einen wesentlichen Aspekt bei der Realisierung und Zielerreichung. Insbesondere durch die dadurch ermöglichte Automatisierbarkeit von Abläufen und die Unterstützung von Personal, jedoch auch durch die Standardisierung von relevanten Informationen, wird die Umsetzung teilweise erst ermöglicht.

Im Schwachstellenmanagement gibt es neben den Computersystemen, die als Assets im Schwachstellenmanagement angesehen werden, fünf verschiedene Rollen für Computer- und Softwaresysteme, welche in den folgenden Abschnitten erläutert werden.

Für die Installation und Wartung der Schwachstellen-Dokumentation, Asset- und Benutzerverwaltung, des Steuerungssystems sowie der Detektionssysteme sind allgemein am Rechenzentrum bzw. im Hochschulnetz für derartige Änderungen bereits vorhandene, verantwortliche Gruppen und Personen zuständig.

5.3.1 Die Schwachstellen-Dokumentation

Die Schwachstellen-Dokumentation ist eine der zentralen Komponenten im Schwachstellenmanagement und allgemein in allen Aktivitäten notwendig. Der Zweck, der durch sie verfolgt wird, ist die Sammlung und Vereinheitlichung von Informationen über im Hochschulnetz relevante Schwachstellen und die Möglichkeit der eindeutigen Identifizierbarkeit von Schwachstellen. Somit kann eine effektive Kommunikation derartiger Informationen gewährleistet werden. Aus dem zentralen Angebot der Schwachstellen-Dokumentation im Hochschulnetz folgen einige Vorteile, die ein effektives Schwachstellen-Management teilweise erst ermöglichen. Zum einen kann der Verwaltungsaufwand auf eine Vielzahl involvierter Personen und Rollen aufgeteilt werden, zum anderen garantieren die zentrale Sammlung und ein zentraler Zugriff die standardisierte Dokumentation von Informationen, wodurch eine Automatisierbarkeit der Abläufe erreicht werden kann.

Durch die Bereitstellung eines sinnvollen Funktionsumfangs, welcher im Verlauf des Abschnitts erklärt wird, ist es möglich, die im Schwachstellenmanagement involvierten Personen und Rollen zu entlasten.

Insofern entspricht die Schwachstellen-Dokumentation einem System aus einem als Datenbasis liegenden Datenbanksystem, einem darüberliegenden Softwaresystem zur Bereitstellung eines Funktionsumfangs sowie einer geeigneten *Application Programming Interface* (API), worüber die Informationen und Funktionen anderen Systemen zur Verfügung gestellt werden.

5.3.1.1 Informationen der Schwachstellen-Dokumentation

Die in der Schwachstellen-Dokumentation erfassten Informationen dienen als Grundlage für Verfahren in allen anderen Aktivitäten. Die generelle Unterteilung der Informationstypen, welche in der Schwachstellen-Dokumentation verwaltet werden, ist wie folgt:

- **Softwareprodukte**
  Der Informationstyp des Softwareprodukts hat im Schwachstellenmanagement insbesondere eine unterstützende Rolle und dient der Standardisierung der Werte bei der Kategorisierung von Schwachstellen (Abschnitt 5.6.2) und Assets (Abschnitt 5.6.3.2), um Eingaben als Freitext zu vermeiden. So soll zum einen die vollständige Beschreibung eines Softwareprodukts sowie eine gleichbleibende Bezeichnung erreicht werden.

  Relevante Attribute eines Softwareprodukts umfassen den `Hersteller`, die `Bezeichnung` des Softwareprodukts sowie die exakte `Version`. Daneben, zur eindeutigen Identifizierung im Schwachstellenmanagement, besitzt jedes Softwareprodukt einen `Identifikator`.

- **Schwachstellen-Fremdquellen**
  Fremdquellen dienen im Schwachstellenmanagement als Grundlage des automatisierten Imports von Schwachstellen. Prinzipiell kann jede Informationsquelle als Fremdquelle genutzt werden, die in irgendeiner Form eine technische Möglichkeit zur automatischen Abfrage der Daten erlaubt. Beispiele für Schwachstellen-Fremdquellen sind die CVE, die NVD, Mailinglisten mit Schwachstelleninformationen, wie *Bugtraq* oder auch generell Meldungen von Softwareherstellern, beispielsweise die *Ubuntu Security Notices*.

  Unverzichtbare Informationen zur Beschreibung einer Schwachstellen-Fremdquelle sind `Verbindungsinformationen` (beispielsweise eine URL), ein für die Quelle und ihre Pflege zuständiger `Quellenverantwortlicher`, Angaben über Softwareprodukte zu den im folgenden Abschnitt beschriebenen lokalen `Black- und Whitelisten` zur Filterung der importierten Schwachstellen sowie weitere tatsächlich adressierbare `Referenzen` auf Detailinformationen, beispielsweise dem Ort der quellenspezifischen Bewertung einer Schwachstelle. Des Weiteren ist zur einfacheren Identifikation einer Quelle ein eindeutiger `Titel` (z.B. „CVE“) und, falls die Notwendigkeit besteht, `Zugangsdaten` für die Fremdquelle (z.B. Benutzer in Kombination mit Passwort) sowie eine Angabe des `Intervalls` zur automatischen Aktualisierung in Tagen in der Beschreibung enthalten.

  Generell gibt es im Schwachstellenmanagement verschiedene Arten von Quellen, die abhängig von den darin ausgeführten Teil-Aktivitäten sind: Quellen zur Identifikation, Bewertung, Kategorisierung und Detektion. Die Teilaktivitäten werden im Laufe des Kapitels erläutert.

- **Meldungstickets**
  Meldungstickets sind im Schwachstellenmanagement ein Mittel zur Meldung von Schwachstellen durch einen Schwachstellen-Melder an die für die Überprüfung und Dokumentation zuständigen Rollen im Schwachstellenmanagement (siehe insbesondere Abschnitt

5.5.2 und die darauf folgenden Abschnitte). Sie sollen vor allem durch die sie darstellenden Werte Informationen zur Meldung standardisieren und die Bearbeitung dadurch vereinfachen.

Meldungstickets müssen eindeutig identifizierbar sein, um von bearbeitenden Rollen wiedergefunden werden zu können, falls dies notwendig ist. Insofern hat jedes Meldungsticket einen eindeutigen `Identifikator`. Da die Rolle des `Schwachstellen-Melders` generell jedes Mitglied des Hochschulnetzes annehmen kann, und dieser für Nachfragen erreichbar sein muss, sind dessen `Kontaktdaten` ebenfalls Bestandteil des Tickets. Des Weiteren, um seinen Zweck zu erfüllen, mögliche Schwachstellen so konkret wie möglich melden zu können, eine Beschreibung der Schwachstelle, bestehend aus `Softwareprodukt` und `Hersteller` oder gegebenenfalls alternativ den betroffenen `Dienst` im Hochschulnetz sowie das `Vorgehen bzw. die Verhaltensweise`, die zu einer Ausnutzung führen kann. Darüber hinaus hält jeder Eintrag ein Datum zur Protokollierung des `Zeitpunktes der Erstellung` des Tickets sowie ein weiteres Datum über den `Zeitpunkt des Abschlusses`.

Schließlich nach Zuweisung des Tickets an einen Schwachstellen-Dokumentator, wird dieser als `Bearbeiter` vermerkt. Ein weiteres Feld zeigt auf den Identifikator des aus der Meldung resultierenden `Schwachstellen`-Eintrags in der Schwachstellen-Dokumentation, damit diese bei einer Bearbeitung in Zusammenhang gebracht und beispielsweise die Kontaktdaten des Schwachstellen-Melders wiedergefunden werden können.

- **Schwachstellen**
  Schwachstellen wurden im Verlauf der Arbeit und insbesondere in Abschnitt 2.3 bereits ausführlich erläutert. Vor allem um die Anforderungen aus Kapitel 3 zu erfüllen, ist es erforderlich, dass jede Schwachstelle durch bestimmte Informationen beschrieben wird.

  Die grundlegendste Information ist ein hochschulnetzweit für jede Schwachstelle eindeutiger `Identifikator`, durch dessen Vorhandensein insbesondere eine Kommunikation von Schwachstelleninformation ermöglicht wird.

  Daneben ist eine `Beschreibung` jeder Schwachstelle ein weiterer wichtiger Aspekt, der zum einen das Verständnis für die Schwachstelle garantiert und zum anderen als Basis für eine weitere Datenverarbeitung dient. Zur Erfüllung dieses Zwecks muss die Beschreibung einige Kriterien erfüllen: So muss aus der Beschreibung das Softwareprodukt in seiner betroffenen Version und dessen Hersteller eindeutig hervorgehen, damit angemessene Methoden zur Detektion entwickelt werden können. Dabei unterstützend und in der Beschreibung notwendigerweise zu integrieren ist auch die Ausnutzungsmöglichkeit der Schwachstelle sowie die Folgen einer Ausnutzung.

  Eine weitere wichtige Information ist die `Bewertung` der jeweiligen Schwachstelle. Diese ergibt sich alleine aus den Charakteristika der Schwachstelle ohne Miteinbeziehung von umgebungsabhängigen Faktoren. Eine Beschreibung des im Konzept verwendeten Bewertungssystems zur Bestimmung der Kritikalität von Schwachstellen findet sich in Abschnitt 5.6.1.1. Ist die Bewertung nicht vorhanden, nimmt sie den Wert *NA* (engl. „not available“) an.

  Auch eine Liste an `Referenzen` zu Informationen aus anderen Quellen sind Bestandteil der Dokumentation einer Schwachstelle. So kann der Aufwand der Beschreibung von

Schwachstellen deutlich reduziert werden, da weitere Informationen oder auch Maßnahmen zur Beseitigung oder Abschwächung der jeweiligen Schwachstelle aus Fremdquellen leicht in die Beschreibung integriert werden können.

Da die Schwachstelleninformationen aus mehreren Quellen importiert werden, hält die Schwachstellen-Dokumentation außerdem zwei weitere Daten pro Schwachstelle: Zum einen die `Informationsquelle`, aus welcher die Schwachstelle stammt, sowie den `quellenspezifischen Identifikator`. Diese Informationen können dazu verwendet werden, Duplikate aus derselben Quelle automatisch zu erkennen und außerdem zu Zwecken der Filterung und Suche von Schwachstellen.

Da die Suche und Filterung von Informationen ein wichtiger Aspekt ist, stellt die Schwachstellen-Dokumentation eine Möglichkeit bereit, um derartige Funktionen zu unterstützen. Dazu wird eine Liste der auch in der Beschreibung vorkommenden betroffenen `Softwareprodukte` mit `Hersteller` und `Softwareversion` separat abgelegt, um die Komplexität der Suche deutlich zu vereinfachen (siehe Abschnitt 5.6.2).

Da teilweise lediglich die Implementierung einer Version eines Softwareprodukts auf bestimmten Plattformen (z.B. Linux, Apple OS X) von einer bestimmten Schwachstelle betroffen ist oder diese möglicherweise in Verwendung mit anderen Softwareprodukten auftritt, wird betroffene Software als Kette von (mindestens einem) Softwareprodukt angegeben, die durch eine logische *Und*-Beziehung miteinander verknüpft sind.

Beispielsweise besteht eine Kette aus den Elementen „Microsoft Windows“ und „Mozilla Firefox“, wodurch die Schwachstelle nur bei kombiniertem Auftreten dieser beiden Softwareprodukte auf einem Asset auftritt, hingegen beispielsweise nicht in einer Linux-Umgebung. Derartige Ketten von *Und*-Verknüpfungen können – falls notwendig – wiederum mit logischen *Oder*-Operatoren verknüpft werden.

Durch die Angabe des `Status` wird die Stufe der Verarbeitung der jeweiligen Schwachstelle verdeutlicht. Dieser hat den Wert *check*, falls die Plausibilität der Informationen bestätigt, die Existenz der Schwachstelle jedoch noch überprüft werden muss, *reject*, falls die Existenz nicht bestätigt werden konnte oder sich als Duplikat herausgestellt hat oder aus einem anderen Grund nicht mehr relevant ist, *assess*, falls die Schwachstelle bestätigt, jedoch noch keine Bewertung zugewiesen bekommen hat und *incomplete*, im Falle, dass die Schwachstellenbeschreibung zur Bewertung nicht ausreichend ist. Der Status *categorize* sagt aus, dass die Schwachstelle kategorisiert werden muss und *detect*, dass die Schwachstelle bestätigt, bewertet sowie kategorisiert, jedoch noch nicht detektierbar ist. Ist eine Schwachstelle bestätigt, bewertet und durch mindestens ein dokumentiertes und implementiertes Detektionsverfahren detektierbar, bekommt sie den Status *complete*. Die im Verlauf des Schwachstellenmanagementprozesses üblichen, anhand der Reihenfolge der Aktivitäten festgelegten Übergänge sind in Abbildung 5.2 illustriert. Der Verlauf kann davon jedoch abweichen, beispielsweise, falls eine Schwachstelle durch den Import bereits kategorisiert, jedoch nicht bewertet ist. In diesem Fall wird immer der – bezogen auf die Reihenfolge der Teilaktivitäten – kleinste zutreffende Status eingetragen ($check < reject < incomplete < assess < categorize < detect < complete$) .

Ein Status bezüglich der Prävention von Schwachstellen ist nicht vorgesehen, da die Erstellung einer Maßnahme dazu nicht für jede Schwachstelle, gemessen am notwendigen

Aufwand, sinnvoll ist.

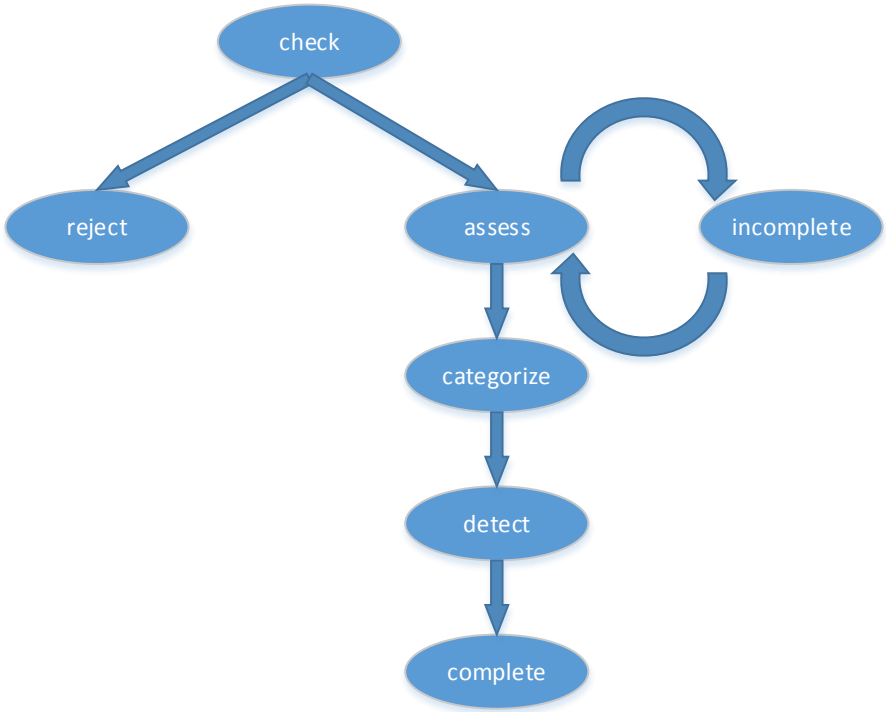


Abbildung 5.2: Übliche Übergänge des Status einer Schwachstelle. Der Status zeigt an, in welchem Verarbeitungsstadium sich der Schwachstelleneintrag in der Schwachstellen-Dokumentation befindet

Weitere Informationen sind ein Zeitstempel, aus dem die `letzte Änderung` des jeweiligen Schwachstelleneintrages ersichtlich wird, sowie die Speicherung des `Editors`, der die letzte manuelle Änderung an der `Beschreibung` der Schwachstelle vorgenommen hat. Änderungen, die durch die Bewertung oder Kategorisierung entstehen, werden beim Setzen des Editor-Feldes nicht berücksichtigt. So kann bei Unklarheiten in der Schwachstellenbeschreibung Kontakt zu dessen letztem Editor hergestellt werden.

Ein weiterer Zeitstempel beschreibt das Datum der `Veröffentlichung` der Schwachstelle. Dieses kann beispielsweise zu Zwecken der Filterung von Einträgen genutzt werden.

- **Maßnahmen**
  Den letzten Informationstyp bilden die Maßnahmen zur Beseitigung, Abschwächung sowie der Prävention von Schwachstellen. Die Dokumentation von Maßnahmen soll die im Schwachstellenmanagement involvierten Personen bei der Behebung unterstützen. So können insbesondere Personen mit weniger notwendigem Fachwissen auf bereits vorhandene Lösungen zurückgreifen und diese umsetzen. Zentraler Aspekt der Dokumentation von Maßnahmen ist daher ihre einfache Wiederverwendbarkeit.

Auch Maßnahmen können durch einen eindeutigen `Identifikator` identifiziert werden und beschreiben im Wesentlichen ein Tupel aus einem `Schwachstellenidentifikator` und einer `Maßnahmenbeschreibung`. Zudem wird zur Unterstützung von Such- sowie Filterfunktionen ein `Maßnahmentyp` gespeichert, welcher die Art der Maßnahme kategorisieren soll (beispielsweise „Patch“, siehe Abschnitt 5.7.3). Genau wie bei Schwachstellen und Detektionsergebnissen hält die Dokumentation einer Maßnahme den `Verfasser` der Maßnahme sowie `Referenzen` auf Fremdquellen, beispielsweise für weitere Erläuterungen, fest.

- **Schwachstellentickets**
  Ein Schwachstellenticket stellt allgemein die Verknüpfung zwischen `Schwachstellen` und `Assets` dar und dient als Hinweis zur Notwendigkeit einer Bearbeitung offener Schwachstellen an Behebungsverantwortliche.

  Zum Zweck einer eindeutigen Identifizierbarkeit eines Eintrages bekommt, genau wie jedes Asset und jede Schwachstelle, auch jedes Schwachstellenticket einen eindeutigen `Identifikator`. Außerdem enthält ein Eintrag Informationen über den `Zeitpunkt der Ticketerstellung` sowie den `Zeitpunkt der Behebung` der Schwachstelle von dem jeweiligen Asset. Bei Abschluss eines Schwachstellentickets wird eine Referenz auf die angewendete `Maßnahme` dokumentiert. Auch ist ein Feld zur Beschreibung eines eventuell angewendeten `Workarounds` sinnvoll.

  Um die Beseitigung bzw. Abschwächung der durch die Tickets dargestellten offenen Schwachstellen, abhängig von ihrer Kritikalität, priorisieren zu können, enthält ein Schwachstellenticket außerdem den Wert der in der Klassifikation ermittelten `Behebungsnotwendigkeit` (vgl. Abschnitt 5.6.4).

  Zudem wird das `Eskalationslevel` dokumentiert.

- **Detektionsverfahren**
  Im Gegensatz zu den weiteren Informationstypen in der Schwachstellen-Dokumentation dient die Dokumentation eines Detektionsverfahrens in erster Linie nicht informativen Zwecken, sondern vor allem der praktischen Weiterverwendung in der Aktivität der Beseitigung und Abschwächung von Schwachstellen. So sind Detektionsverfahren dafür essentielle und unverzichtbare Komponenten und beschreiben allgemein eine Möglichkeit zur Detektion einer Schwachstelle auf Assets.

  Für jede Schwachstelle gibt es kein, ein oder mehrere Detektionsverfahren, wodurch diese Zuordnung in der Dokumentation eindeutig ersichtlich sein muss. Daher besteht es grundlegend aus einem für den jeweiligen Eintrag eindeutigen `Identifikator` sowie der durch das Verfahren auf Assets detektierbaren `Schwachstelle`. Da es wie in Abschnitt 5.7.2 beschrieben, zwei verschiedene Arten der Detektionsverfahren gibt – Detektionsverfahren aus Systemsicht sowie aus Netzsicht – ist genau dieser `Typ` Teil der Dokumentation und kann die Werte *system* bzw. *netz* zugewiesen bekommen, sowie eine kurze `Beschreibung` der Vorgehensweise. Da sich Verfahren für bestimmte Betriebssysteme unterscheiden können, müssen auch diese in Form eines dafür vorgesehenen Feldes `Plattform` angegeben werden, wobei die Kombination der am allgemeinsten zutreffenden Werte eingetragen werden muss.

  Beispiele für die Liste unterstützter Plattformen sind:

[*plattformunabhängig*]

[*Microsoft Windows 8.1*, *Linux*, *BSD*]

[*FreeBSD 10.2*, *Arch Linux 2015.10.01*]

[*Microsoft Windows*, *Arch Linux 2015.10.01*, *Debian Linux*]

Auch wird der `Editor` bzw. Ersteller des Detektionsverfahrens erfasst.

Kernstück eines Detektionsverfahrens ist die konkrete `Implementierung` in einer dafür geeigneten und in den weiteren Komponenten im Schwachstellenmanagement, insbesondere den Detektionssystemen sowie Detektionsagenten, benutzbaren Skript- bzw. Programmiersprache. Die Implementierung zeichnet sich dadurch aus, dass sie, abgestimmt mit dem angegebenen Typ, in der Lage ist, eine Schwachstelle entweder aus Systemsicht, d.h. durch Ausführung auf dem jeweiligen Asset, oder aus Netzsicht, also ausgeführt auf einem Detektionssystem, zu erkennen und entsprechend einen booleschen Rückgabewert liefert: Einerseits *wahr*, falls das Asset durch die jeweilige Schwachstelle betroffen ist, andernfalls *falsch*. Die Implementierung liegt als Quelldatei sowie als ausführbare Datei vor.

Zur Erreichung der Benutzbarkeit ist die Implementierung des Weiteren genau zu dokumentieren, mit Angaben über die verwendete `Programmiersprache`, notwendige `Module bzw. Softwarebibliotheken`, der genutzte `Compiler bzw. Interpreter` sowie eine exakte Dokumentation über die `Eingaben` (beispielsweise die IP-Adresse des Assets) zur Ausführung des Programms.

Da Detektionsverfahren nicht nur durch ihren Ersteller getestet, sondern auch durch einen weiteren Detektionsverantwortlichen abgenommen werden müssen (vgl. Abschnitt 5.7.1), wird der `Begutachter` und der `Freigabestatus` ebenfalls in dem Eintrag dokumentiert.

Diese Beschreibung stellt den „Standardfall“ eines Detektionsverfahrens dar. Zusätzlich gibt es noch eine weitere Art von Detektionsverfahren, die nicht direkt zur Detektion einer Schwachstelle eingesetzt werden, sondern eine Ausgabe liefern, die von Personen interpretiert werden müssen. Ein Beispiel für ein derartiges Verfahren ist ein *Portscan*, der für ein Zielhost eine Liste offener Ports zurückliefert. Entsprechend ist der Rückgabewert dieser Art von Detektionsverfahren nicht *true* bzw. *false*, sondern beliebig wählbar. Ein weiterer Unterschied ist, dass diese Detektionsverfahren in der Regel keine Referenz auf eine bestimmte Schwachstelle haben. Ein Anwendungsfall ist in Abschnitt 6.5.2 (Funktion „`detectionDryRun`“) erklärt.

5.3.1.2 Funktionen der Schwachstellen-Dokumentation

Der grundlegende Unterschied der Schwachstellen-Dokumentation zu einem einfachen Datenbanksystem ergibt sich aus dem Funktionsumfang, den sie mit sich bringt. Diese Funktionen umfassen Möglichkeiten zur Erstellung, Änderung sowie zum Löschen von Datensätzen der im vorhergehenden Abschnitt erwähnten Datentypen sowie zur Unterstützung der Organisation und allgemein von Abläufen im Schwachstellenmanagement:

- **Manuelle Dokumentation von Schwachstellen**
  Die Nutzung des vollen Funktionsumfangs der technischen Lösung zur Unterstützung

eines Schwachstellenmanagements ist besonders im Hochschulumfeld von größerer Bedeutung als in Unternehmen. Hochschulnahe Institute sind oft treibende Kräfte bei der Entwicklung neuer, in der Regel quelloffener Software und zugleich Einsatzbereich spezieller (eventuell prototypischer) Software zur Verwaltung oder auch zu Forschungszwecken (z.B. Simulationen). Die Möglichkeit der Dokumentation von Schwachstellen derartiger Softwareprodukte ist daher notwendig und in der technischen Komponente der Schwachstellen-Dokumentation vorgesehen.

- **Import von Schwachstelleninformation aus Fremdquellen**
  Auch wenn die Möglichkeit der manuellen Dokumentation ein Bestandteil des Konzepts ist, so ist es zugleich notwendig, bereits existierende, gut dokumentierte Schwachstellen automatisiert auszulesen. Das Konzept sieht insofern die Verwendung von Schwachstelleninformation aus einer oder mehreren Quellen vor, welche automatisiert abgerufen werden. Derartige Quellen können beispielsweise in Form von Schwachstellen-Listen wie die CVE, Hersteller- bzw. CERT-Benachrichtigungen oder öffentlichen Mailinglisten wie *Bugtraq* vorliegen.

  Aufgrund der unterschiedlichen Charakteristika der Fremdquellen – beispielsweise dem unterschiedlichen Datenformat, in dem die Schwachstelleninformationen angeboten werden, die unterschiedlichen Aktualisierungsintervalle der Quellen oder auch die Art der Informationsübertragung (z.B. http, E-Mail) – erfolgt der Import von Schwachstelleninformationen über ein modular aufgebautes und erweiterbares Softwaresystem, welches die jeweiligen Informationen aus den Quellen abruft und entsprechend auf das in der Schwachstellen-Dokumentation vorgegebene Format abbildet.

  Die Quellen für Schwachstelleninformationen können dabei prinzipiell, je nach Anforderungen der jeweiligen Hochschulumgebung, frei gewählt und in die Schwachstellen-Dokumentation integriert werden. Dabei müssen auch zwei inhaltlich verschiedene Arten von Schwachstellenquellen unterschieden und berücksichtigt werden: Zum einen Quellen, welche ausschließlich aktuelle und neue Informationen bereitstellen, sowie zum anderen Quellen, die ausschließlich den kompletten Inhalt zur Verfügung stellen. Insbesondere bei letzterem besteht die Gefahr der erneuten Eintragung bereits erfasster Schwachstellen, wodurch entsprechende Maßnahmen zur Vermeidung von Duplikaten vorgesehen sind. Die Vorgehensweise zur Vermeidung redundanter Schwachstelleneinträge wird im Verlauf dieses Abschnitts näher beschrieben.

  Das Abrufen der Fremdquellen geschieht entweder automatisiert in vorgegebenen Intervallen oder auch durch ein manuelles Auslösen durch den Chief Vulnerability Manager. Dabei sind zudem grundlegend zwei unterschiedliche Arten des automatischen Imports bzw. der Ablaufkoodination (engl. „Scheduling“) zu beachten:

  Einerseits können Informationen über einen „Push“- und andererseits über einen „Pull“-Ansatz eingeholt werden. Bei der Umsetzung der Informationssammlung mit Hilfe des **„Push“-Ansatzes** wird der Import als Reaktion auf das Ereignis der Aktualisierung bzw. dem Hinzufügen neuer Schwachstellen durch die Fremdquelle ausgelöst. Ein Beispiel ist hier die Nutzung von Mailinglisten, wodurch das Importieren der in der Mail befindlichen Informationen durch ihr Eintreffen ausgelöst wird.

  Beim zweiten Ansatz, dem **„Pull“-Ansatz**, wird das Ereignis zum Importieren von Schwachstelleninformationen durch die Schwachstellen-Dokumentation selbst ausgelöst, wobei die Fremdquelle nicht unbedingt neue Informationen halten muss. Vielmehr wird

die Überprüfung des Inhalts von Fremdquellen auf neue Schwachstellen in geeignet-definierten Zeitintervallen ausgeführt. Diese Zeitintervalle sind abhängig von dem Aktualisierungsintervall der jeweiligen Fremdquelle und werden daran angepasst, um zum einen unnötig hohe Systemlast zu vermeiden (d.h. die Überprüfung erfolgt nicht in zu kurzen Zeitintervallen) und zum anderen die Aktualität der Schwachstellen-Dokumentation zu gewährleisten (d.h. zu lange Zeitintervalle werden ebenfalls vermieden). Bei unklaren oder unregelmäßigen Aktualisierungsintervallen einer Fremdquelle wird jedoch die Überprüfung in kurzen Zeitabständen bevorzugt, da andernfalls die Sicherstellung der Aktualität der Schwachstellen-Dokumentation verletzt werden kann. Die CVE ist ein typisches Beispiel einer Fremdquelle, die mit Hilfe des „Pull"-Ansatzes abgefragt wird.

Unabhängig vom Ansatz des Importierens von Schwachstellen bekommen diese je nach Verarbeitungsstadium einen jeweiligen Status zugewiesen (vgl. vorherigen Abschnitt). Sind die importierten Schwachstellen lediglich beschrieben, jedoch nicht bewertet, so bekommen sie den Status *assess*. Liefert die Fremdquelle oder eine weitere Quelle eine Bewertung zu den importierten Schwachstellen, so bekommen diese den Status *categorize*. Ist hingegen bereits eine Kategorisierung möglich (vgl. Abschnitt 5.6.2), beispielsweise wie in der NVD durch verknüpfte Einträge in der CPE, so haben die betroffenen Schwachstellen den Status *detect*. Beim automatisierten Import von implementierten Detektionsverfahren aus Fremdquellen hingegen ist zu beachten, dass diese vor deren Einsatz unbedingt auf ihre Konformität zur Beschreibung von Detektionsverfahren im Konzept sowie auf ihre Eignung zur Detektion und möglichen negativen Effekten geprüft werden oder ob es sich gar um Malware handelt. Insofern ist der Import von Detektionsverfahren üblicherweise nicht vollautomatisiert möglich, im Konzept jedoch auch nicht ausgeschlossen.

Neben der Vorgehensweise beim Importieren von Schwachstelleninformationen ist insbesondere auch der **Umfang der Schwachstelleninformationen** genauer zu betrachten. Einige Quellen enthalten Informationen über Schwachstellen vieler verschiedener Softwareprodukte, darunter auch zum großen Teil im Hochschulumfeld nicht-relevante oder im Anwendungsbereich nicht-definierte Schwachstellen. Nachteile, die sich aus der Integration derartiger Schwachstellen in die Schwachstellen-Dokumentation eines Hochschulnetzes ergeben, sind die unnötige Belegung von Speicherplatz in den Datenbanken, die auf lange Sicht und mit zunehmendem Inhalt zu erwartende Verlangsamung der Datenbankzugriffe sowie die Schaffung von weiterem unnötigem technischem sowie organisatorischem Aufwands, da jede Schwachstelle in weiteren organisatorischen Prozessen (z.B. der Klassifikation und Beseitigung bzw. Abschwächung) sinnvollerweise miteinbezogen werden muss, um sie nutzbar zu machen. Die lediglich Speicherung der Informationen ohne Weiterverarbeitung bringt zugleich keinen Nutzen. Eine Lösung zur Vermeidung dieses Problems ist das bereits im Schritt des Importierens angewendete Aussortieren nicht betrachteter Schwachstellen. Ansätze zur Filterung werden im Laufe des Abschnitts erläutert.

- **Aktualisierung von Schwachstelleninformation**
  Da sich Schwachstellenbeschreibungen nicht selten im Verlauf der Zeit ändern bzw. weitere Informationen in Fremdquellen hinzugefügt werden, ist die Erkennung von

Änderungen von Einträgen in Fremdquellen sowie eine automatische Aktualisierung in der Schwachstellendokumentation Teil des Funktionsumfangs. Hier ist es wichtig, dass der Schwachstelleneintrag nach der Aktualisierung seinen eindeutigen Identifikator behält, da weitere Informationen (Referenzen, Maßnahmen, Schwachstellentickets, Detektionsverfahren und Verantwortlichkeiten) im Schwachstellenmanagement in der Regel auf diesen referenzieren.

Die Aktualisierung von Änderungen aus Fremdquellen in der Schwachstellen-Dokumentation kann dabei auf verschiedene Weisen realisiert werden. Insbesondere kann die Schwachstelle als Ganzes oder nur Teile der Beschreibung aktualisiert werden, die sich geändert haben, wobei sich aus ersterem Ansatz lediglich ein möglicher Vorteil einer einfacheren Implementierung ergibt, aus letzterem eine effizientere Ausführung. Der resultierende Eintrag in der Schwachstellen-Dokumentation enthält hingegen unabhängig vom verwendeten Verfahren die gleichen Informationen.

Als problematischer erweist sich eine Aktualisierung in Bezug auf möglicherweise erneut durchzuführenden Verarbeitungsschritten. Beispielsweise kann eine automatische Aktualisierung der Schwachstellenbeschreibung eines bereits dokumentierten Eintrags mit Status *complete* zu einer abweichenden Bewertung oder Kategorisierung führen und insofern auch Einfluss auf Detektionsverfahren haben, sodass der Status nicht mehr korrekt ist. Eine Lösung des Problems ist in der Regel jedoch abhängig von der Darstellung und Standardisierung der Informationen auf Seite der Fremdquellen:

Bietet eine Fremdquelle die Informationen zu Schwachstellen lediglich in Form von Freitext an, ist es üblicherweise nicht automatisiert möglich, beispielsweise die Korrektheit der Bewertung oder Kategorisierung zu beurteilen. Ist es hingegen möglich, den Typ der aktualisierten Informationen automatisch zu bestimmten (beispielsweise eine Bewertung oder ein weiteres von der Schwachstelle betroffenes Softwareprodukt), so können auch weitere Abhängigkeiten von dem geänderten Datum feingranularer bestimmt werden. Kann beispielsweise automatisch erkannt werden, dass sich bei einer Aktualisierung eines Schwachstelleneintrages ein von der Schwachstelle betroffenes Softwareprodukt geändert hat, kann davon ausgegangen werden, dass mit dem Eintrag verknüpfte Detektionsverfahren mit hoher Wahrscheinlichkeit angepasst werden müssen.

- **Vermeidung redundanter Schwachstelleneinträge**
  Methoden zur Vermeidung doppelter Schwachstelleneinträge gehören in Hinblick auf die Ermöglichung der effektiven Kommunikation von Schwachstelleninformationen zur Grundfunktionalität der Schwachstellen-Dokumentation. Dabei muss darauf geachtet werden, dass Schwachstellen aus einer Fremdquelle jeweils auf ihre Existenz in der Schwachstellen-Dokumentation überprüft werden, um diese nicht doppelt zu dokumentieren. Weitaus komplizierter kann die Erkennung von jeweils einem Eintrag in zwei oder mehr unterschiedlichen Fremdquellen sein, welche die gleiche Schwachstelle beschreiben.

  Die **Vermeidung einer doppelten Dokumentation von Schwachstellen aus genau einer Fremdquelle** betrifft in erster Linie Fremdquellen, welche nicht nur neu-hinzugefügte, sondern auch bereits früher veröffentlichte Schwachstellen erneut, bzw. im „Gesamtpaket“ anbieten, wie beispielsweise die CVE auf ihrer Download-

Site (`https://cve.mitre.org/data/downloads/`). Dabei gibt es zwei verschiedene, effektive Ansätze, die dieses Problem lösen können, beide jedoch eine Speicherung eines Identifikators der jeweiligen Fremdquelle sowie des fremdquelleninternen Identifikators der jeweiligen Schwachstelle benötigen. Der Identifikator für die Fremdquelle muss beim Hinzufügen der Quelle definiert werden und kann beispielsweise eine beliebige Zahl oder eine Zeichenkette sein, wobei die Wahl einer sinnvollen Zeichenkette durch ihre leichtere Verständlichkeit für Personen von Vorteil ist. Der fremdquellenspezifische Identifikator der jeweiligen Schwachstelle ist in der Regel in der Fremdquelle definiert. Ein Identifikator für die CVE kann beispielsweise „cve“ sein; der fremdquellenspezifische Schwachstellenidentifikator ist folglich die jeweilige CVE-Nummer, z.B. „CVE-2014-0160“.

Insofern besteht eine Möglichkeit darin, die Schwachstelle mit dem aktuellsten fremdquellenspezifischen Identifikator aus der Schwachstellen-Dokumentation zu erfassen und alle Schwachstellen aus der jeweiligen Fremdquelle mit einem aktuelleren Identifikator hinzuzufügen.

Hätte beispielsweise der letzte, aus dem *Microsoft Security Bulletin* importierte Schwachstelleneintrag den Fremdquellenidentifikator „MS15-098“, so müssten aufgrund der pro Eintrag hochzählenden Bulletin-Nummer bei folgendem Importieren lediglich Einträge ab dem Identifikator „MS15-099“ berücksichtigt werden.

Dabei besteht jedoch die Gefahr, dass Änderungen an bereits erfassten Schwachstelleneinträgen in der Fremdquelle übersehen werden und somit die Aktualität der Schwachstellen-Dokumentation nicht immer erfüllt ist. Auch ist dieser Ansatz bei Quellen wie der CVE ungeeignet, da, wie in Abschnitt 2.3.3, die CVE-ID nicht von der Reihenfolge der Beschreibung der Schwachstellen abhängig ist, sondern von der Reihenfolge der Verteilung der IDs an entsprechende Einrichtungen und Softwarehersteller.

Daher besteht die zweite Möglichkeit darin, jeden Eintrag der Fremdquelle zu überprüfen und bei Differenzen, die zwischen Fremdquelle und der Schwachstellen-Dokumentation festgestellt werden, den entsprechend anhand der fremdquellenspezifischen Schwachstellen-ID identifizierten Eintrag in der Schwachstellen-Dokumentation zu aktualisieren. In der Schwachstellen-Dokumentation nicht-existierende Einträge werden entsprechend hinzugefügt. Dabei abzugleichen sind insbesondere der Beschreibungstext der Schwachstelle in der hochschulinternen Dokumentation sowie der jeweiligen Schwachstelle – falls vorhanden – die Referenzen und – ebenfalls, falls vorhanden – das Datum der letzten Änderung eines Eintrages.

Die **Vermeidung einer doppelten Dokumentation von Schwachstellen aus mehreren Quellen** kann hingegen nicht alleine durch den Abgleich der Beschreibungstexte erfolgen, da die Einträge in unterschiedlichen Quellen in der Regel unterschiedlich genau und vor allem in einem anderem Wortlaut beschrieben sind. Es besteht jedoch die Möglichkeit – falls vorhanden – identische Schwachstellen in mehreren Quellen anhand von Referenzen in den Fremdquellen zu erkennen. Die Erkennung kann beispielsweise durch Überprüfung des Vorhandenseins von Referenzen der Schwachstelle auf den jeweils anderen Eintrag der zu überprüfenden Fremdquelle erfolgen oder durch

die Kontrolle auf identische Referenzen zu einer dritten Fremdquelle. Haben beispielsweise beide zu vergleichende Einträge eine Referenz auf denselben CVE-Eintrag, so kann davon ausgegangen werden, dass es sich um dieselbe Schwachstelle handelt.

Als Ursache doppelter Einträge einer Schwachstelle kann genauso sich überschneidende automatisierte Dokumentation und eine manuelle Dokumentation angesehen werden. Dabei muss man den zeitlichen Aspekt, d.h. welcher Eintrag zuerst in der Dokumentation vorhanden war, berücksichtigen.

Im Falle, dass die manuelle Dokumentation derselben Schwachstelle nach der automatisierten Eintragung aus einer Fremdquelle geschieht, ist die Erkennung insbesondere durch die auswertenden Schwachstellen-Dokumentatoren vorzunehmen. Eine Beschreibung dazu findet sich in Abschnitt 5.5.2.

Der andere Fall ist, dass die manuelle Dokumentation vor der automatisierten Dokumentation vorgenommen wurde. In diesem Fall ist die automatische Erkennung des Duplikats essentiell von den angegebenen Daten durch den Schwachstellen-Melder abhängig. Die Erkennung kann allgemein analog zum oben beschriebenen Fall zur Vermeidung doppelter Einträge aus mehreren Quellen ablaufen – auf Basis gleicher Referenzen.

Eine weitere Herausforderung besteht jedoch in der **Auswahl der Informationen aus zwei Fremdquellen**, die in der Schwachstellen-Dokumentation übernommen werden. Da diese in der Regel als Freitext vorliegt und die Qualität der Beschreibung einer Schwachstelle von vielen Faktoren abhängig ist, kann dieser Schritt nicht automatisiert und zugleich komplett verlässlich erfolgen. Jedoch ist die manuelle Überprüfung jeder Schwachstellenbeschreibung und deren Abgleich für ein effizientes Schwachstellenmanagement üblicherweise durch die hohe Anzahl an Schwachstellen zu aufwändig und daher nicht sinnvoll. Eine mögliche Lösung ist die Integration mehrerer Beschreibungen einer Schwachstelle in die Schwachstellen-Dokumentation, wodurch es jedoch wahrscheinlich ist, dass sich die gespeicherten Texte stark ähneln, ohne dass entsprechend mehr Informationen zur Verfügung stehen. Daher ist es sinnvoller, lediglich die Schwachstellenbeschreibung aus einer Quelle anhand verschiedener Charakteristika oder Vorgehensweisen auszuwählen und auf identische Einträge in anderen Fremdquellen zu referenzieren.

Derartige Charakteristika sind beispielsweise große Unterschiede in der Länge des Beschreibungstextes, das Datum der letzten Änderung des jeweiligen Eintrages der Fremdquelle oder in der Praxis auch Erfahrungen, die mit der jeweiligen Fremdquelle gemacht wurden („welche Quelle bietet in der Regel mehr Informationen?“). In diesem Konzept wird letzteres Kriterium – die in der Praxis mit einer Fremdquelle gemachten Erfahrungen – in Form einer Möglichkeit zur Priorisierung von Schwachstellenquellen gewählt, wobei die Priorisierung von genannten Erfahrungswerten im Zusammenhang mit dem Schwachstellenmanagement basiert.

Im Konzept ist nicht vorgesehen, dass zwei Fremdquellen identische Priorität haben. Insofern ist die Wahl der Quelle immer eindeutig.

Eine weitere Möglichkeit zur Vermeidung doppelter Einträge stellt das im folgenden Abschnitt erklärte fremdquellenspezifische White- bzw. Blacklisting dar, mit der

Möglichkeit, Duplikate proaktiv durch die Nicht-Beachtung von Schwachstellen bestimmter Softwareprodukte aus bestimmten Quellen zu vermeiden. Beispielsweise könnten alle Schwachstellen für *Mozilla Firefox* in der CVE ignoriert werden und ausschließlich durch Benachrichtigungen vom Hersteller direkt importiert werden.

- **Filtern irrelevanter Schwachstellen**
  Das Filtern nach relevanten Schwachstellen beim Import von Schwachstelleninformation aus Fremdquellen dient vor allem der Steigerung der Effizienz sowie der Vermeidung von unnötigem Aufwand. Der Schritt erfolgt direkt nach dem Auslesen der Fremdquelle. So kann vermieden werden, dass für das im Hochschulnetz umgesetzte Schwachstellenmanagement irrelevante Schwachstellen dokumentiert werden.

> Dabei ist zu beachten, dass die Wahl relevanter Schwachstellen insbesondere von dem jeweiligen Anwendungsbereich der einzelnen Institutionen abhängig ist. Falls die IT-Umgebung beispielsweise ausschließlich aus Linux-Rechnern besteht, werden Schwachstellen für andere Betriebssysteme und deren plattformabhängige Software zur Vermeidung unnötigem Aufwands nicht im Schwachstellenmanagement betrachtet.

Die Filterung der Schwachstelleninformationen aus einer Fremdquelle wird anhand von Softwareprodukten vorgenommen, wobei Betriebssysteme als in diese Kategorie fallend betrachtet werden. Bei der Vorgehensweise zur Filterung werden dabei zwei verschiedene Ansätze verfolgt. Der erste Ansatz ist ein **Whitelisting**-Verfahren, in dem Softwareprodukte mit Relevanz für das Schwachstellenmanagement im jeweiligen Hochschulnetz aufgelistet werden. Die jeweiligen Daten aus den Fremdquellen werden daraufhin auf ihre Zugehörigkeit zu einem Softwareprodukt automatisch ausgewertet und im Falle eines positiven Ergebnisses zur Eintragung in die Schwachstellendokumentation freigegeben. Dabei gibt es mehrere unterschiedliche Wege, um die Relevanz von Schwachstellen herauszufinden:

1. Die erste Möglichkeit zur Erkennung der Zugehörigkeit einer Schwachstelle ist anhand ihrer **Quelle**. Werden Schwachstelleninformationen beispielsweise durch eine Mailingliste des Herstellers eines relevanten Softwareprodukts importiert (z.B. von den *Ubuntu Security Notices* [Can15]), so kann man im Regelfall davon ausgehen, dass alle Schwachstellen das jeweilige Softwareprodukt betreffen. Ist dies nicht der Fall, so wird eine der folgenden Möglichkeiten genutzt.
2. Da die **Beschreibung einer Schwachstelle** in der Praxis oft als Freitext vorliegt, wird als weitere Möglichkeit die Suche nach Begriffen in Betracht gezogen. Vorkommnisse der Bezeichner von Softwareprodukten werden als Indiz der Zugehörigkeit der jeweiligen Schwachstelle zum entsprechenden Softwareprodukt angerechnet. Diese Methodik des Aussortierens ist jedoch potenziell stark unzuverlässig.
3. Die letzte Möglichkeit besteht in der Nutzung weiterer von der Beschreibung **separierter Informationen** in der Fremdquelle. Beispielsweise können die in der

NVD zu Verfügung gestellten Einträge aus der CPE ausgewertet werden (vgl. Abbildung 4.2). Diese Möglichkeit wird jedoch in der Regel nicht bzw. nicht immer in Fremdquellen angeboten, erleichtert die Identifikation betroffener Softwareprodukte jedoch enorm.

Die andere, dem Whitelisting entgegengesetzte Möglichkeit wird durch ein **Blacklisting** von Softwareprodukten erreicht. Das heißt, dass Schwachstellen der in der Blacklist aufgelisteten Softwareprodukte nicht automatisch in die Schwachstellen-Dokumentation eingetragen werden. Die Vorgehensweise ist ähnlich wie im Whitelisting, mit dem Unterschied, dass bei einer positiven Zuordnung einer Schwachstelle zu einem Produkt in der Blacklist, keine Eintragung in die Schwachstellen-Dokumentation erfolgt.

Eine Erweiterung der beiden Ansätze erfolgt durch die Betrachtung eines globalen White- bzw. Blacklistings, mit Vorgaben für alle Fremdquellen sowie einem lokalen, also fremdquellenspezifischen White- bzw. Blacklistings. Die globale Vorgehensweise dient der Auswahl der Schwachstellen für Softwareprodukte, die generell bei der Umsetzung des Schwachstellenmanagements eine Rolle spielen, wohingegen die lokale Vorgehensweise insbesondere zur Sicherstellung der Qualität der Schwachstelleninformation dient sowie eine weitere Maßnahme zur Vermeidung doppelter Einträge darstellt.

Neben der Filterung von Schwachstellen aufgrund des betroffenen Softwareprodukts besteht genauso bei Bedarf die Möglichkeit, Schwachstellen aufgrund ihrer Kritikalität zu filtern. Können aufgrund sehr knappen Personals beispielsweise lediglich kritische Schwachstellen behoben werden, so wird die Aussortierung von Schwachstellen mit einer Bewertung *gering* bis *hoch* in Betracht gezogen.

Auch ist eine Filterung beispielsweise anhand des Datums der Veröffentlichung oder (falls vorhanden) des Typs der Schwäche (vgl. *Common Weakness Enumeration*) möglich. Des weiteren können mehrere Filter in Kombination in Form einer *Und-* als auch *Oder*-Verknüpfung der Kriterien zur Filterung angewendet werden. Beispielsweise „Eine Schwachstelle ist relevant, falls sie innerhalb der letzten fünf Jahre dokumentiert wurde und mindestens eine mittlere Kritikalität aufweist“.

- **Festlegen des Status von Schwachstellen**
  Der Status jeder in der Schwachstellen-Dokumentation eingetragenen Schwachstelle kann separat durch verschiedene Rollen gesetzt werden. Er dient vor allem organisatorischen Zwecken und unterstützt die Abläufe im Schwachstellenmanagement.

- **Bewerten von Schwachstellen**
  Genauso kann die Bewertung von Schwachstellen manuell festgelegt werden. Zudem bietet die Schwachstellen-Dokumentation die Nutzung von bereits in Fremdquellen dokumentierten Bewertungen. Dazu ist im Konzept vorgesehen, eine Abbildung des jeweiligen Bewertungssystems auf das im Hochschulumfeld genutzte Bewertungssystem vorzunehmen. Die Beschreibung einer Abbildung findet sich in Abschnitt 5.6.1.2.

- **Einsehen dokumentierter Schwachstellen**
  Neben der Ermöglichung einer strukturierten Organisation der Schwachstelleninformationen und der Unterstützung der im Schwachstellenmanagement beteiligten Personen bei ihren Aufgaben, ist die Einsicht von Schwachstelleninformation der Haupt-

zweck der Schwachstellen-Dokumentation. Dadurch wird es den Nutzern allgemein ermöglicht, sich über Schwachstellen im Hochschulnetz zu informieren. Da potenziell mehrere zehntausend Schwachstelleneinträge in der Datenbank auf lange Sicht nicht unwahrscheinlich sind, gibt es eine Suchfunktion, die das Auffinden unterstützt.

Diese bietet die **Suche nach Schwachstellen** anhand eines bestimmten `Identifikators`, eines `Fremdquellenbezeichners`, des `fremdquellenspezifischen Schwachstellenidentifikators`, anhand des `Status`, des `Herstellers` des betroffenen Softwareprodukts sowie des `Softwareprodukts` selbst. Außerdem die Suche durch eine Angabe eines gewünschten Intervalls, anhand der Bewertung sowie dem Zeitpunkt der Einfügung der Schwachstelle in die Datenbank. Des Weiteren, für unspezifischere Suchanfragen, ist eine Suche mit Hilfe von Platzhaltern (engl. „wildcard“) auf Basis des `Beschreibungstextes` vorhanden.

Prinzipiell können die in der Schwachstellen-Dokumentation gespeicherten Schwachstellen und Maßnahmen von allen Nutzern des Hochschulnetzes eingesehen werden, um ihre Systeme selbst sicherer zu machen. Hier ist jedoch zu beachten, dass dies nicht für assetspezifische Schwachstellen gilt, sondern diese nur von Personen eingesehen werden können, die abgesehen von der Rolle *Nutzer* mindestens eine weitere Rolle zugewiesen bekommen haben (vgl. 5.2). Diese Einschränkung dient ebenfalls der Sicherheit im Hochschulnetz, da assetspezifische Schwachstellen gezielt und einfacher für Angriffe genutzt werden können.

- **Erstellung und Abschluss von Meldungstickets**
  Bei der Erstellung der Meldungstickets erfüllt die Schwachstellen-Dokumentation eine Kontrollfunktion zur Prüfung auf Vollständigkeit der angelegten Daten. Sie gewährleistet jedoch auch die Zugänglichkeit der Daten und ist für eine automatische Zuweisung an einen entsprechenden Bearbeiter in der Rolle des Schwachstellen-Dokumentators zuständig. Die Zuweisung erfolgt durch die zufällige Auswahl innerhalb aller Schwachstellen-Dokumentatoren in Kombination mit einer Berücksichtigung offener Meldungstickets pro Person. Eine Zuweisung an Schwachstellen-Dokumentatoren, denen kein offenes Meldungsticket zugewiesen ist, wird bevorzugt vorgenommen.

- **Erstellung von Schwachstellentickets**
  Die Erstellung von Schwachstellentickets kann automatisiert oder auch manuell erfolgen. Eine automatische Erstellung geschieht auf Basis einer vorgenommenen Detektion von Schwachstellen auf einem Asset. Im Falle eines positiven Ergebnisses, d.h. einer erkannten Schwachstelle auf einem Asset, wird bei zutreffender Behebungsnotwendigkeit, welche aufgrund der Kritikalität des Assets sowie der jeweiligen Schwachstelle festgestellt wird, ein Schwachstellenticket erstellt. Die Bewertung der Kritikalität einer Schwachstelle ist in Abschnitt 5.6.1.1, die Bewertung von Assets in Abschnitt 5.6.3.1 sowie die Bewertung der Behebungsnotwendigkeit in Abschnitt 5.6.4 beschrieben.

  Da nicht alle Schwachstellen automatisiert detektiert werden können, können Schwachstellentickets außerdem manuell durch Detektionsverantwortliche erstellt werden.

- **Automatische und manuelle Eskalation von Schwachstellentickets**
  Wie später in Abschnitt 5.7.4 beschrieben, wird im Schwachstellenmanagement das

Mittel der Eskalation von Schwachstellentickets genutzt, um die Behebung von Schwachstellen zu unterstützen bzw. auf unter Umständen unberücksichtigte Schwachstellen auf Systemen hinzuweisen. Die Schwachstellen-Dokumentation bietet dafür die Kontrolle mittels vordefinierter Kriterien der Eskalation, bei deren Erfüllung automatisch eskaliert wird.

- **Manuelle Eskalation der Verantwortung zur Verifikation**
  Eine Eskalation der Verantwortung der Verifikation (Abschnitt 5.5.3) ist vor allem bei durch Nutzer gemeldeten Schwachstellen in Form eines Meldungstickets mit Bezug auf einen Dienst im Hochschulnetz notwendig. So kann bei ungeeigneter automatischer Zuweisung der Verantwortung durch die Schwachstellen-Dokumentation der jeweilige Schwachstellen-Prüfer die Verantwortung an einen zufällig gewählten, mit dem Dienst in Verbindung stehenden Schwachstellen-Prüfer, weiterleiten. Eine geeignete Person ist über die in der Asset- und Benutzerverwaltung hinterlegten Informationen über einen Nutzer identifizierbar.

- **Einsehen und Abschließen von Schwachstellentickets**
  Die Einsicht von Schwachstellentickets ist ein grundlegender Aspekt bei der Beseitigung von Schwachstellen. Dabei wird insbesondere das betroffene Asset, die betroffene Schwachstelle sowie das Eskalationslevel ersichtlich. Des Weiteren das Erstellungsdatum sowie, ob das Ticket bereits gelöst oder noch offen ist. Der Abschluss des Schwachstellentickets kann lediglich durch einen Behebungsverantwortlichen des betroffenen Assets erfolgen.

- **Dokumentation und Einsicht von Maßnahmen**
  Die eigentliche Ausführung einer Maßnahme ist im Allgemeinen immer mit persönlichem Aufwand verbunden. Die Dokumentation bekannter und neuer Maßnahmen zur Beseitigung von Schwachstellen dient der Aufwandsreduzierung sowie der Schaffung von Effizienz durch die Einsicht und Verwendung von bereits dokumentierten Maßnahmen.

- **Meldung von Schwachstelleninformation**
  Als proaktive Maßnahme zur Steigerung eines Bewusstseins für Schwachstellen und deren Relevanz für die Informationssicherheit sowie anfälliger Softwareprodukte besteht eine automatisch-ablaufende Funktionalität der Schwachstellen-Dokumentation darin, die Nutzer des Schwachstellenmanagements über aktuelle Gefahren zu informieren. Diese Berichte dienen des Weiteren auch zur Einschätzung des aktuellen Zustands einzelner Einheiten im Hochschulnetz (d.h. Institute, Hochschulen, Lehrstühle, dem Rechenzentrum) oder dem Gesamtnetzzustand und dem Vergleich verschiedener Einheiten.

- **Dokumentation von Detektionsverfahren**
  Die Dokumentation von Detektionsverfahren wird in Abschnitt 5.7.1 erläutert. Die Aufgabe der Schwachstellen-Dokumentation ist es, Detektionsverfahren zu sichern und im Schwachstellenmanagement zugänglich zu machen.

- **Archivierung von Schwachstellen**
  Eine Archivierung von Schwachstellen leistet einen wichtigen Beitrag zum Erhalt der Effektivität und Effizienz von organisatorischen und insbesondere technischen Abläufen

(z.B. Suchanfangen). Sehr große Datenmengen, im Schwachstellenmanagement vor allem Schwachstellenbeschreibungen und damit verbundene Daten, können dazu führen, dass einerseits Speicherplatz weiträumig belegt wird und andererseits Zugriffe auf die Datenbank erheblich länger dauern. Durch eine Archivierung und dem damit verbundenen Ausschließen von Einträgen aus Datenbankzugriffen, soll eine allgemeine technische Verlangsamung des Systems verhindert werden.

Die Archivierung kann je nach verarbeiteten Datenmengen in mehr oder weniger großen Zeitabständen stattfinden und monatlich genauso wie im Abstand mehrerer Jahre erfolgen. Kriterien zur Auswahl der Einträge, die archiviert werden, sind beispielsweise der Grad der Verarbeitung einer Schwachstelle oder auch die Dauer, die eine Schwachstelle in der Schwachstellen-Dokumentation liegt. Auch eine Kombination ist möglich und sinnvoll, da beispielsweise nicht-detektierbare Schwachstellen (Status $\neq$ *complete*), die seit mehreren Jahren dokumentiert sind, womöglich keine Relevanz im jeweiligen Hochschulnetz haben.

- **Aufräumen der Schwachstellen-Dokumentation**
  Das Aufräumen der Schwachstellen-Dokumentation ist eine notwendige Funktion insbesondere zur Vermeidung der Haltung unnötiger Informationen und insofern zur Reduktion der Komplexität von Suchanfragen sowie Schaffung verfügbaren Speicherplatzes. Dies soll durch ein automatisiertes oder manuell ausgelöstes Löschen unnötiger Einträge erfolgen, welche entweder nicht mehr gebraucht werden oder sich im Verlauf der Überprüfung der Plausibilität bzw. Existenz nicht bestätigen (Status *reject*).

  Beim Aufräumen werden ebenfalls alte Schwachstellentickets miteinbezogen und falls sinnvoll ebenfalls aus der Schwachstellen-Dokumentation entfernt.

- **Zugriff über ein Application Programming Interface (API)**
  Die API dient in erster Linie der Kommunikation der technischen Komponenten untereinander. Ausschließlich das Steuerungssystem kann so Daten der Schwachstellen-Dokumentation auslesen und schreiben.

5.3.2 Die Asset- und Benutzerverwaltung

Die Asset- und Benutzerverwaltung ist neben der Schwachstellen-Dokumentation die zweite Quelle an essentiellen Informationen im Schwachstellenmanagement. Auch wenn die Zentralisierung dieser Informationen sinnvoll ist – insbesondere zum Zweck einer Sicherstellung der gleichartigen Beschreibung von Assets – so ist die Umsetzung beispielsweise durch die Haltung hochschulweiter vertraulicher und personenbezogener Informationen nicht immer möglich und durch jeden Kunden gewünscht.

Daher sieht das Konzept zunächst eine zentrale Lösung einer Asset- und Benutzerverwaltung vor, die jedoch potenziell um weitere, jeweils dezentral verwaltete Asset- und Benutzerverwaltungen erweitert werden kann.

Ähnlich wie die Schwachstellen-Dokumentation ist auch die Asset- und Benutzerverwaltung ein System, das auf Basis eines Datenbanksystems weitere Funktionen bereitstellt.

5.3.2.1 Informationen der Asset- und Benutzerverwaltung

Wie der Name bereits aussagt, hält die Asset- und Benutzerverwaltung vor allem Informationen über Assets und Nutzer. Eine ausführliche Beschreibung wird in den folgenden Punkten vorgenommen.

- **Assets**
  Informationen der Assets und Schwachstellen sind im Schwachstellenmanagement die einzigen unverzichtbaren Daten und werden prinzipiell in allen Aktivitäten benötigt. Ein Asset bezeichnet ein Computersystem, dessen Behebung von Schwachstellen im Rahmen des Schwachstellenmanagements organisiert werden soll. Entsprechend muss es beschrieben werden.

  Ein Asset hat genau wie eine Schwachstelle im Hochschulnetz einen eindeutigen `Identifikator`. Des Weiteren besitzen Assets als Computersysteme, falls vorhanden – d.h. falls es in das Hochschulnetz integrierte Geräte sind, einen `netzspezifischen Identifikator`, welcher in Form einer IPv4- bzw. IPv6-Adresse oder einem im Netz eindeutigen Hostname (bspw. „dnsserver01“) hinterlegt ist.

  Auch Teil eines Assets ist eine `Beschreibung` aus welcher der Verwendungszweck des Assets für jeweilige Nutzer ersichtlich ist. Falls das Asset die Basis für eine IT-Dienstleistung oder andere Systeme (z.B. ein Rechnercluster bzw. verteiltes System) bildet, muss der Gesamtzusammenhang in der Beschreibung dargestellt werden. Der dazugehörige `Dienst` wird zudem in einem separaten Feld dokumentiert. Genauso die `Kritikalität` eines Assets. Ein Bewertungssystem zur Beschreibung der Kritikalität von Assets ist in Abschnitt 5.6.3.1 beschrieben. Diese ergibt sich insbesondere aus der Art der auf dem Asset gespeicherten bzw. verarbeiteten Daten, der Anzahl und Art betroffener Nutzer sowie der Abhängigkeit von IT-Dienstleistungen von dem Asset. Des Weiteren ist für eine optimale Nutzung von Informationen das Asset betreibende `Institut`, der `Gerätetyp` sowie dessen exakte Bezeichnung (inklusive Version) des `Betriebssystems` gespeichert.

  Auch die für das Asset verantwortlichen Nutzer müssen ersichtlich sein: Zum einen ein eindeutiger Asset-Verantwortlicher, mindestens ein Detektionsverantwortlicher sowie mindestens ein Behebungsverantwortlicher. Um klare Verantwortlichkeiten zu schaffen wird jeweils ein Detektionsverantwortlicher sowie Behebungsverantwortlicher als hauptverantwortlich ausgezeichnet.

  Des Weiteren enthält ein Eintrag zwei Zeitstempel, einer der den Zeitpunkt der `letzten Änderung` der Daten des Assets zeigt, um die Suche nach nicht aktuellen Einträgen zu erleichtern sowie ein weiterer zur Dokumentation der letzten durchgeführten `Detektion`.

  Auch enthält die Beschreibung eines Assets eine Liste der später in der Teilaktivität der Kategorisierung identifizierten und auf dem jeweiligen Asset eingesetzten bzw. installierten `Softwareprodukte`.

  Jedes Asset hält außerdem einen Wert, der die Zeitspanne in Tagen beschreibt, in dem auf dem Asset automatisch Schwachstellen detektiert werden – das `Detektionsintervall`. Wird das Detektionsintervall nicht explizit bei Anlegen eines Assets gesetzt, so

bekommt es einen konfigurierbaren Standardwert zugewiesen. Der Standardwert wird je nach Bedarf gesetzt (beispielsweise 7 Tage).

- **Nutzer**
  Eine Information, die genau wie Assets weniger schwachstellenmanagementspezifisch, sondern viel mehr allgemein managementspezifisch ist, stellen beteiligte Nutzer in den Managementprozessen dar. Deren Dokumentation ist essenziell für die Festlegung organisatorischer Abläufe und daher auch in einem definierten Umgang mit Schwachstellen unverzichtbar.

  Im Schwachstellenmanagement sind vor allem drei Aspekte notwendig: Zum einen die Identifikation von Personen, welche in das Schwachstellenmanagement involviert sind und Kontaktmöglichkeiten, sowie die dem Nutzer zugeordneten Aufgaben und Zuständigkeiten.

  Die Identifikation wird anhand zweier Attribute festgelegt: Einerseits bekommt jeder Nutzer einen im Hochschulnetz sowie im Schwachstellenmanagement eindeutigen `Identifikator` zugewiesen und zum anderen muss der volle `Name` des Nutzers definiert sein.

  Die Zuständigkeit eines Nutzers ist anhand der Angabe der Zugehörigkeit zu einem `Dienst`, `Institut` sowie einer oder mehrerer `Rollen` möglich. Die Angabe über das Institut hilft zudem bei der Identifikation eines Nutzers. Außerdem ist ein `Kontakt`, beispielsweise in Form einer oder mehrerer E-Mail-Adressen hinterlegbar.

5.3.2.2 Funktionen der Asset- und Benutzerverwaltung

Die Funktionen dienen allgemein dem Hinzufügen, Ändern und Entfernen von Assets und Nutzern. Zudem existiert eine von der Asset- und Benutzerverwaltung angebotene API zum Zugriff durch das Steuerungssystem.

- **Anlegen, Einsehen, Bearbeiten und Entfernen von Assets**
  Assets können angelegt, bearbeitet und ebenfalls wieder aus der Asset- und Benutzerverwaltung entfernt werden. Das Bearbeiten ist notwendig, da Assets möglicherweise ihre Konfiguration oder andere Attribute (beispielsweise die Netz-ID) ändern und insofern auch im Schwachstellenmanagement aktuell gehalten werden müssen. Das Entfernen von Assets kann beispielsweise bei der Einschränkung des Anwendungsbereichs notwendig sein. Die Informationen über Assets müssen zudem auslesbar sein, um sie im Schwachstellenmanagement verwenden zu können.

- **Anlegen, Einsehen, Bearbeiten und Entfernen von Nutzern**
  Auch Nutzer im Schwachstellenmanagement können sich häufig ändern. Insofern gibt es auch hier die Möglichkeit, neue Benutzer hinzuzufügen, zu bearbeiten und zu entfernen. Zudem kann auf die Nutzerdaten lesend zugegriffen werden.

- **Zugriff über eine API**
  Die Asset- und Benutzerverwaltung ist genau wie die Schwachstellen-Dokumentation über eine API und ausschließlich durch das Steuerungssystem nutzbar.

5.3.3 Detektionssysteme

Detektionssysteme bezeichnen Computersysteme, die zum Zweck der Detektion von Schwachstellen auf Assets eingesetzt werden. Sie erfüllen insofern zwei Hauptaufgaben, zum einen die Detektion aus Netzsicht und zum anderen dienen sie als Komponente zur Kommunikation mit den auf Assets eingesetzten Detektionsagenten (vgl. Abschnitt 5.3.4). Bei beiden Aufgaben ergibt sich eine Problematik, die private Netze mit sich bringen: Dass Assets innerhalb dieser Netze nicht von außerhalb direkt adressierbar und demnach nicht erreichbar sind.

Auch wenn die Möglichkeit besteht, dieses Problem für die Kommunikation mit Detektionsagenten beispielsweise durch Port-Forwarding zu lösen, ist dieser Ansatz für eine Vielzahl an Assets in dem jeweiligen privaten Netz aufgrund des hohen Konfigurations- und Dokumentationsaufwands nicht geeignet und vor allem in Netzen mit dynamischer Adressvergabe (DHCP) schwierig. Das Problem der Unmöglichkeit der Adressierung von Assets in privaten Netzen verhindert hingegen die Detektion aus Netzsicht von außerhalb des Netzes im Allgemeinen komplett.

Eine einfache Lösung bietet die Einrichtung von einem oder mehreren Detektionssystemen innerhalb der privaten Netze, welche wiederum mit Port-Forwarding deutlich effizienter ansprechbar sind und die Konfiguration deutlich einfacher zu warten ist. So können Detektionssysteme in privaten Netzen von außerhalb angesprochen werden, welche die Kommunikation mit den Detektionsagenten innerhalb des privaten Netzes übernehmen und gleichzeitig die Detektion aus Netzsicht durchführen können. Die Identifikation des richtigen Detektionssystems erfolgt anhand der im Asset gespeicherten Instituts-ID sowie Netzwerk-ID. Assets in privaten Netzen müssen als Netz-ID eine IP-Adresse angeben, um das Netz identifizieren zu können. Eine Identifikation aufgrund des Hostnamens ist nicht möglich.

Die Ausführung eines Detektionsdurchlaufs sowie der Anstoß zur Aktualisierung von Detektionsverfahren wird durch das Steuerungssystem durchgeführt und zurückgelieferte Daten (vor allem Informationen über detektierte Schwachstellen) weiterverarbeitet. Die Wartung von Detektionssystemen ist über das Netz möglich und wird vom Rechenzentrum aus durchgeführt. Um die Komplexität der Ausbringung von Detektionsverfahren über das Netz an Detektionssysteme zu vereinfachen, haben diese – angefangen beim Betriebssystem, auf denen sie eingerichtet sind – eine netzweit einheitliche Konfiguration.

5.3.3.1 Informationen im Zusammenhang mit Detektionssystemen

Um Verfahren zur Detektion von Schwachstellen auf Assets durchzuführen, benötigen Detektionssysteme die im Folgenden beschriebenen Informationen aus der Schwachstellen-Dokumentation sowie der Asset- und Benutzerverwaltung. Sie selbst haben keinen Zugriff auf diese Komponenten, sondern werden über das Steuerungssystem verwaltet und angesteuert.

- **Verbindungsinformationen**
  Verbindungsinformationen eines Assets wie eine IP-Adresse oder ein eindeutiger Hostname werden zum einen zur Realisierung der Detektion über das Netz als auch zur Abfrage der Detektionsagenten benötigt.

- **Implementierte Detektionsverfahren**
  Die in der Schwachstellen-Dokumentation abgelegten Detektionsverfahren beschreiben

für Detektionssysteme eindeutige Anweisungen zur Detektion von Schwachstellen auf Assets. Die Implementierungen der Detektionsverfahren sind auf den Detektionssystemen abgelegt und durch diese ausführbar.

- **Schwachstellen**
  Ein Detektionssystem benötigt Informationen über die Schwachstellen, die detektiert werden sollen. So muss mindestens der Identifikator einer Schwachstelle bekannt sein, damit Detektionssysteme zum einen ein angemessenes Detektionsverfahren ausführen sowie im Anschluss ein positives Detektionsergebnis eindeutig einer Schwachstelle zuordnen können.

5.3.3.2 Funktionen von Detektionssystemen

Der von den Detektionssystemen unterstützte Funktionsumfang richtet sich nach der Aufgabe, die es im Schwachstellenmanagement einnimmt und zielt auf eine Erfüllung dieser sowie einer – aufgrund der potenziell hohen Verteilung im Hochschulnetz – möglichst guten Wartbarkeit von zentraler Stelle ab.

- **Aktualisierung der Detektionsverfahren**
  Um eine Vielzahl von parallel betriebenen Detektionssystemen zu verwalten und hinsichtlich ihrer Möglichkeiten zur Detektion aktuell und konsistent zu halten, müssen diese und insbesondere die Organisation von Detektionsverfahren über das Netz wartbar sein. Dazu zählt im Allgemeinen das Hinzufügen und Entfernen von Detektionsverfahren auf dem jeweiligen Gerät. Einzelne Änderungen bzw. Anpassungen an Detektionsverfahren an sich sind nicht in Detektionssystemen vorgesehen, da diese zentral in den Dokumenten der Schwachstellen-Dokumentation erfolgen.

  Da wie oben beschrieben, Detektionssysteme eine einheitliche Konfiguration haben, ist die Filterung der Detektionsverfahren über das Netz (insbesondere nach Betriebssystem) nicht notwendig, sondern es werden alle Detektionsverfahren auf allen Detektionssystemen eingesetzt.

  Des Weiteren findet die Aktualisierung der Detektionsagenten über Detektionssysteme statt.

- **Durchführen eines Detektionsdurchlaufs**
  Allgemein sind Detektionssysteme für die Steuerung und Ausführung der Detektion auf einem Asset zuständig. Zentrale Rolle spielt dabei die Detektion über das Netz, die sie selbst durchführen sowie die Beauftragung der Detektionsagenten auf den relevanten Assets. Ein Detektionsagent liefert das Ergebnis der Detektion aus Systemsicht an die Detektionssysteme zurück – genauer den Identifikator der auf dem Asset gefundenen Schwachstellen – und wird auf den Detektionssystemen mit dem Ergebnis der Detektion aus Netzsicht kombiniert (vgl. Abschnitt 5.7.2.2). Dies ist vor allem zur Erkennung und dem Zusammenfassen von doppelt detektierten Schwachstellen notwendig, damit pro detektierter Schwachstelle immer nur ein Schwachstellenticket erstellt wird (vgl. Abschnitt 5.7.2).

- **Abfragen des Softwareinventars auf einem Asset**
  Für die Kategorisierung sowie Möglichkeit zur Filterung von Detektionsverfahren nach

relevanten Schwachstellen spielt das Softwareinventar eines Assets eine wichtige Rolle. Detektionssysteme können Funktionen zur Abfrage der installierten Software auf einem Asset aufrufen und entweder zur Kategorisierung oder Filterung an das Steuerungssystem weitergeben

5.3.4 Detektionsagenten

Ein Detektionsagent ist im Gegensatz zu den anderen technischen Komponenten im Schwachstellenmanagement kein Computersystem, sondern ein reines Softwaresystem, das auf den jeweiligen Assets zum Einsatz kommt. Es muss daher (abhängig vom festgelegten Anwendungsbereich des Schwachstellenmanagements) entsprechend plattformunabhängig implementierbar sein, da Assets im Hochschulumfeld bezüglich des Gerätetyps und verwendeter Software, inklusive Betriebssystem im Allgemeinen zunächst keinen Einschränkungen unterworfen sind. Die für den Einsatz notwendigen Bibliotheken und Softwareprodukte werden exakt dokumentiert und für Nutzer öffentlich zugänglich gemacht, um den Betrieb des Detektionsagenten auf einem Asset sicherzustellen.

Informationen, die ein Detektionsagent kennen muss, umfassen prinzipiell lediglich den in der Asset- und Benutzerverwaltung hinterlegten `Identifikator` des Assets, auf dem der Detektionsagent eingesetzt wird, um die von ihm gesammelten Daten dem entsprechendem Asset eindeutig zuweisen zu können. Gesteuert wird ein Detektionsagent ausschließlich durch ein Detektionssystem, das mit der Detektion von Schwachstellen auf dem Asset durch das Steuerungssystem beauftragt wird.

Wie bereits erwähnt ist ein Problem, das bei diesem Kommunikationsweg („Detektionssystem steuert Detektionsagenten“) insbesondere auch Detektionsagenten betrifft, deren Einsatz in privaten Netzen. Somit sind Detektionsagenten von außerhalb des Netzes nicht erreichbar und können zunächst weder detektieren noch aktualisiert werden, da sie ausschließlich über Detektionssysteme gesteuert werden. Eine alternative Lösung ist die Steuerung des Agenten direkt vom Asset aus, sodass Asset-Nutzer die Detektion und Aktualisierung direkt ausführen können. Eine weitere Möglichkeit ist ein lokaler Scheduler pro Asset, der in vordefinierten Zeitabständen eine Detektion und Aktualisierung durchführt. So würde die Kommunikation von innerhalb des privaten Netzes nach außen stattfinden, was in der Regel deutlich unkomplizierter ist.

Ein Problem, das bei der Ausführung direkt vom Asset aus auftreten kann, ist unter Umständen eine Manipulation des Detektionsagenten zur Vortäuschung einer anderen Identität. Insbesondere in Hochschulnetzen besteht dafür ein erhöhtes Risiko, da, wie in Abschnitt 2.5.4 erwähnt, Netznutzer oft administrative Rechte auf den von ihnen genutzten Rechnern, und vor allem auf privaten Geräten, haben. Insbesondere aus privaten Netzen heraus ist die Prüfung auf die Identität eines Assets, beispielsweise anhand der Netz-ID ein Problem, das schwierig zu lösen ist, da alle Assets aus demselben lokalen Netz nach außen die gleiche IP-Adresse aufweisen. Insofern ist die Möglichkeit der Ausführung einer Detektion nicht im Konzept vorgesehen.

Neben der **Detektion** von Schwachstellen auf Assets umfasst der Aufgaben- und Funktionsbereich von Detektionsagenten die Unterstützung der **Dokumentation** des jeweiligen Assets, auf dem der Detektionsagent eingesetzt wird. Insbesondere zur Unterstützung der

Kategorisierung von Assets, ist die Detektion eingesetzter Software auf dem Asset, eine wichtige Funktion.

Ein Detektionsagent ist zudem dynamisch um Detektionsverfahren erweiterbar. Die Aktualisierung der Detektionsverfahren in Detektionsagenten wird von Detektionssystemen vorgenommen. Um die Konfiguration von Detektionsagenten vor dem Zugriff und vor Änderungen durch unberechtigte Personen zu schützen, wird dieser durch entsprechende Besitz- und Zugriffsberechtigungen auf dem jeweiligen Asset gesichert. Dazu kann beispielsweise ein weiteres Benutzerkonto auf Betriebssystemebene angelegt werden, auf das ausschließlich Detektionsverantwortliche zugreifen können.

Die (Erst-)Einrichtung eines Detektionsagenten auf einem Asset entspricht der Installation eines Softwarepakets und muss vor allem für eine Vielzahl von Assets entsprechend geplant und durchgeführt werden. Die Verantwortung für die jeweilige Einrichtung des Detektionsagenten liegt bei dem jeweiligen Asset-Verantwortlichen. Gründe dafür sind vor allem, dass die Einrichtung ohne Zustimmung des Asset-Verantwortlichen unter Umständen nicht möglich ist oder es durch den Einsatz des Detektionsagenten zu unerwünschten Effekten kommen kann (beispielsweise durch inkompatible Software). Dem Asset-Verantwortlichen muss jedoch eine genaue Anleitung zur Installation mit einer Beschreibung aller benötigten Softwarepakete (z.B. eine bestimmte Perl-Umgebung) und einer Beschreibung der Konfiguration des Agenten und des Assets zur Verfügung gestellt werden.

5.3.5 Das Steuerungssystem

Das Steuerungssystem bildet eine organisatorisch zentrale Komponente im Schwachstellenmanagement das Personen, abhängig von ihrer Zugehörigkeit zu einem bestimmten Institut sowie der Rolle, die sie innehaben (vgl. Abschnitt 5.2), den Zugriff auf bestimmte Funktionen und Daten der Schwachstellen-Dokumentation, Asset- und Benutzerverwaltung oder auch in der Detektion ermöglicht.

Für den Nutzerzugriff stellt ein Steuerungssystem eine mandantenfähige Oberfläche in Form eines Webportals zum Schwachstellenmanagement zur Verfügung, für dessen Nutzung eine Authentifizierung notwendig ist.

Generell geht die Organisation der Informationen der Schwachstellen-Dokumentation sowie Asset- und Benutzerverwaltung vom Steuerungssystem aus und stellt die Verbindung beider Komponenten dar. Daher hat das Steuerungssystem auf alle Informationen und Funktionen der beiden Komponenten Zugriff. Weitere Funktionen, die vom Steuerungssystem übernommen werden, sind im Folgenden aufgelistet:

- **Automatische Zuweisung von Verantwortlichkeiten**
  Eine wichtige Funktion zur Unterstützung des organisatorischen Ablaufs des Schwachstellenmanagements ist die eindeutige Zuweisung von Verantwortungen bei der Verarbeitung von Schwachstellen. Die Stufe der Verarbeitung der jeweiligen Schwachstelle wird durch ihren Status ausgedrückt. So weist das Steuerungssystem automatisch bei Änderung des Status einer Schwachstelle, der Erstellung eines Meldungstickets oder auch eines Schwachstellentickets – abhängig von seiner Implementierung – die Verantwortung zu einer der manuell durchzuführenden Teilaktivitäten (Ersterfassung, Verifikation, Bewertung, Kategorisierung, Detektion, Behebung und Eskalation) eindeutig einer Person in der entsprechend dafür verantwortlichen Rolle zu.

- **Scheduling automatischer Prozesse**
  Die Schwachstellen-Dokumentation bietet die Möglichkeit, automatische Abläufe durch eine Zeitablaufsteuerung zu planen und auszuführen. Automatisierte Prozesse sind beispielsweise der Import von Schwachstellen und die geplante Detektion auf Assets.

- **Ausführen einer Detektion von Schwachstellen auf Assets**
  Die Ausführung eines Detektionslaufs geht vom Steuerungssystem aus. Dieses erhält beispielsweise durch Nutzereingaben oder den Scheduler die Aufforderung zur Detektion einer Liste von Schwachstellen auf bestimmten Assets. Die Anfrage zum Starten eines Detektionslaufs wird vom Steuerungssystem – je nach Anzahl an Assets, auf denen Schwachstellen detektiert werden – an ein oder mehrere Detektionssysteme weitergegeben.

  Zuvor nimmt das Steuerungssystem jedoch eine Filterung nach relevanten Schwachstellen vor, welche unter anderem anhand der Behebungsnotwendigkeit im Abgleich zum Schwachstellenakzeptanzkriterium bestimmt werden. Weitere Faktoren, die Einfluss auf die Relevanz einer Schwachstelle zur Detektion haben, sind allgemein die Detektierbarkeit einer Schwachstelle sowie die Softwarelandschaft auf den Zielassets.

  Als für die Detektion auf einem bestimmten Asset irrelevante Schwachstellen werden solche angesehen, deren Behebungsnotwendigkeit zu gering ist ($\leq$ Schwachstellenakzeptanzkriterium), zu denen es kein geeignetes Detektionsverfahren für das Asset gibt sowie von der Schwachstelle betroffene Produkte nicht auf dem Asset installiert sind.

- **Zugriff über ein Application Programming Interface (API)**
  Die API des Steuerungssystems bildet die Schnittstelle zu Komponenten außerhalb des Schwachstellenmanagements. So können beispielsweise die CMDB, Lösungen des *Security Information & Event Management* (SIEM), Daten aus dem Identitätsmanagement (insbesondere Nutzer und Rechte) oder weitere Systeme auf die Daten und Funktionen des Steuerungssystems zugreifen. Zur Sicherung vor unautorisiertem Zugriff wird die API durch geeignete Mechanismen der Authentifizierung und Autorisierung geschützt.

5.3.6 Kommunikationsmodell der Komponenten

Die Kommunikation der Komponenten untereinander ist klar festgelegt und in Abbildung 5.3 illustriert. Die Steuerung der Komponenten im Schwachstellenmanagement geht von zentraler Stelle, dem Steuerungssystem, aus. Von diesem aus können zum einen Nutzer die Informationen aus dem Schwachstellenmanagement nutzen und Funktionen zur Erfüllung ihrer Aufgaben ausführen. Zum anderen werden automatische Abläufe wie geplante Detektionsläufe oder der geplante Import von Schwachstellen aus Fremdquellen über den Scheduler ausgeführt.

Das Steuerungssystem ist die einzige Komponente, die auf die Schwachstellen-Dokumentation sowie die Asset- und Benutzerverwaltung zugreift und somit die Konsistenz der Daten zwischen beiden Komponenten untereinander sicherstellt. Beispielsweise werden Nutzer aus der Asset- und Benutzerverwaltung ebenfalls als Attribute mehrerer Daten in der Schwachstellen-Dokumentation benötigt.

Als zentrale Stelle ist das Steuerungssystem auch die einzige Komponente, welche auf die Detektionssysteme Zugriff hat und diese steuert.

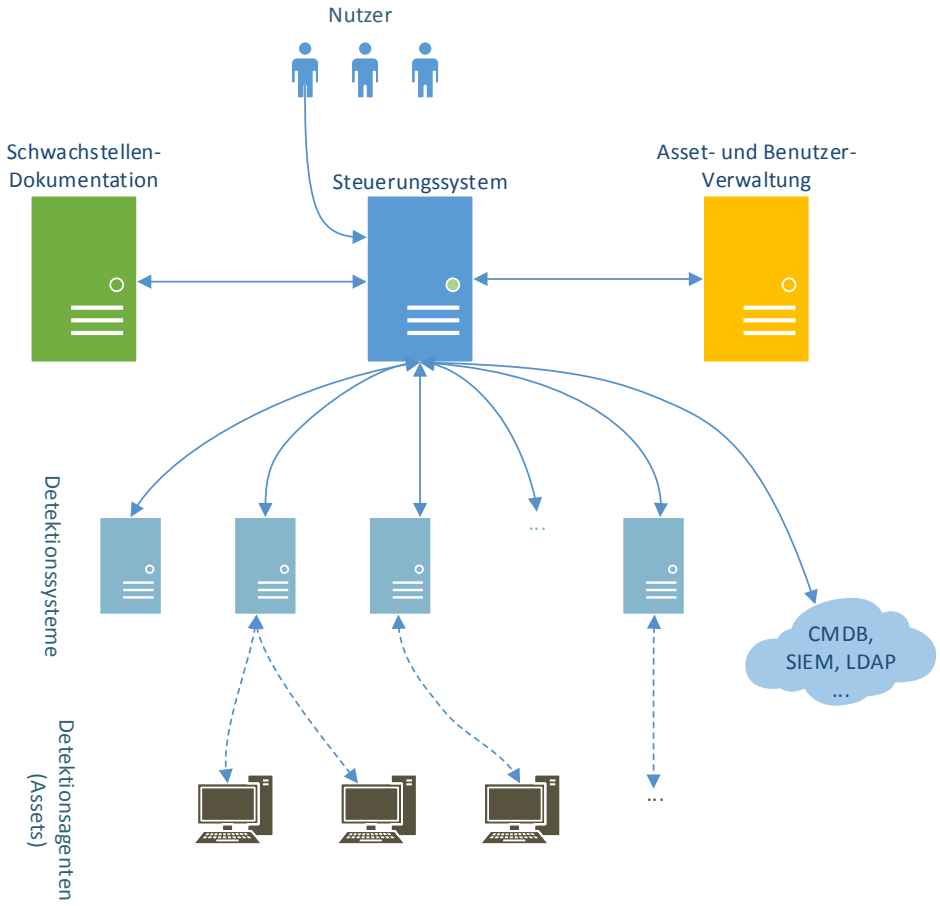


Abbildung 5.3: Geräte, Nutzer und ihre Kommunikation im Schwachstellenmanagement (gestrichelte Linien bezeichnen dynamisch zugewiesene Verbindungen)

Wird auf einem Steuerungssystem die Funktion zur Ausführung der Detektion auf einem Asset aufgerufen, dann wählt das Steuerungssystem ein geeignetes Detektionssystem aus und ruft auf diesem die Funktion zur Detektion auf dem jeweiligen Asset auf. Das Detektionssystem spricht dafür zum einen den Detektionsagenten des Assets an und ruft bei diesem die Funktion zur Detektion auf, zum anderen führen Detektionssysteme selbst auf demselben Asset die Detektion aus Netzsicht durch.

Auch Zugriffe von Komponenten außerhalb des Schwachstellenmanagements greifen ausschließlich über das Steuerungssystem zu. Beispiele für derartige Komponenten sind eine *Configuration Management Database* (CMDB) aus dem Configuration Management, Komponenten des *Security Information and Event Managements* (SIEM) oder auch eines *Lightweight Directory Access Protocol* (LDAP) Dienstes zur Anbindung des Schwachstellenmanagements an ein Identitätsmanagement.

5.4 Der Prozess des Schwachstellenmanagements

Wie bereits im Verlauf der Arbeit erklärt, handelt es sich hier um die im Schwachstellenmanagement betrachteten Aktivitäten um die Identifikation, Klassifikation, Beseitigung und Abschwächung sowie die Prävention von Schwachstellen. Der Prozess des Schwachstellenmanagements ist – wie in Managementprozessen üblich – durch zyklische Wiederholungen charakterisiert, das heißt, dass die Ausübung kein Ende findet, sondern erneut durchlaufen wird.

Abbildung 5.4 illustriert den Prozess des Schwachstellenmanagements mit den erwähnten Aktivitäten sowie untereinander möglichen organisatorischen Übergängen im Idealfall. Wichtige Ein- und Ausgaben der jeweiligen Aktivitäten werden in den einzelnen zugehörigen Abschnitten erklärt und in den dazugehörigen Abbildungen 5.5, 5.6, 5.7 zusammengefasst.

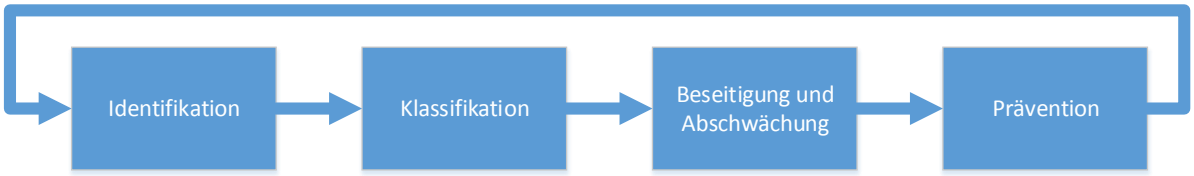


Abbildung 5.4: Organisatorischer Aktivitätenzyklus im Schwachstellenmanagement

Das grundlegende Ziel im Schwachstellenmanagement ist letztendlich die Reduzierung von Schwachstellen auf den an das Netz angeschlossenen Systemen, wodurch die Aktivitäten der Beseitigung und Abschwächung, aber auch der Prävention von Schwachstellen im Vordergrund stehen. Die Erreichung einer effektiven sowie effizienten Durchführung kann in der Praxis jedoch nicht ohne die vorherige Identifikation sowie der Klassifikation von Elementen im Schwachstellenmanagement geschehen. In der Aktivität der Identifikation wird entsprechend durch die Erfassung des aktuellen Zustands über Schwachstellen sowie Assets die Basis für ein funktionierendes Management geschaffen.

Direkt auf die Identifikation folgt die Aktivität der Klassifikation, in der durch die Bewertung von Schwachstellen und Assets zum einen eine effiziente Bearbeitung, vor allem aber auch die Effektivität in einem großen Umfang der Umsetzung ermöglicht wird, da nicht-relevante Elemente (beispielsweise Schwachstellen mit sehr geringer Auswirkung) aussortiert und im weiterem Verlauf nicht berücksichtigt werden, wodurch die Komplexität der Ausführung stark verringert wird.

Aufgrund der durchgeführten Klassifikation werden in den mit höherem manuellen Aufwand verbundenen Aktivitäten der Beiseitigung und Abschwächung sowie der Prävention

von Schwachstellen lediglich relevante Schwachstellen auf relevanten Assets betrachtet und die Durchführung mit geeignet organisierten Daten unterstützt.

Die Beseitigung und Abschwächung vorhandener Schwachstellen auf Assets ist, aufgrund der akuten Gefahr einer potenziellen Ausnutzung, aus organisatorischer Sicht, der Prävention von Schwachstellen zu bevorzugen.

5.5 Identifikation von Schwachstellen

Im Prozess des Schwachstellenmanagements ist die Identifikation von Schwachstellen, organisatorisch betrachtet, die erste auszuführende Aktivität zur Schaffung einer Grundlage an Daten und Kommunikation für die weiteren Aktivitäten.

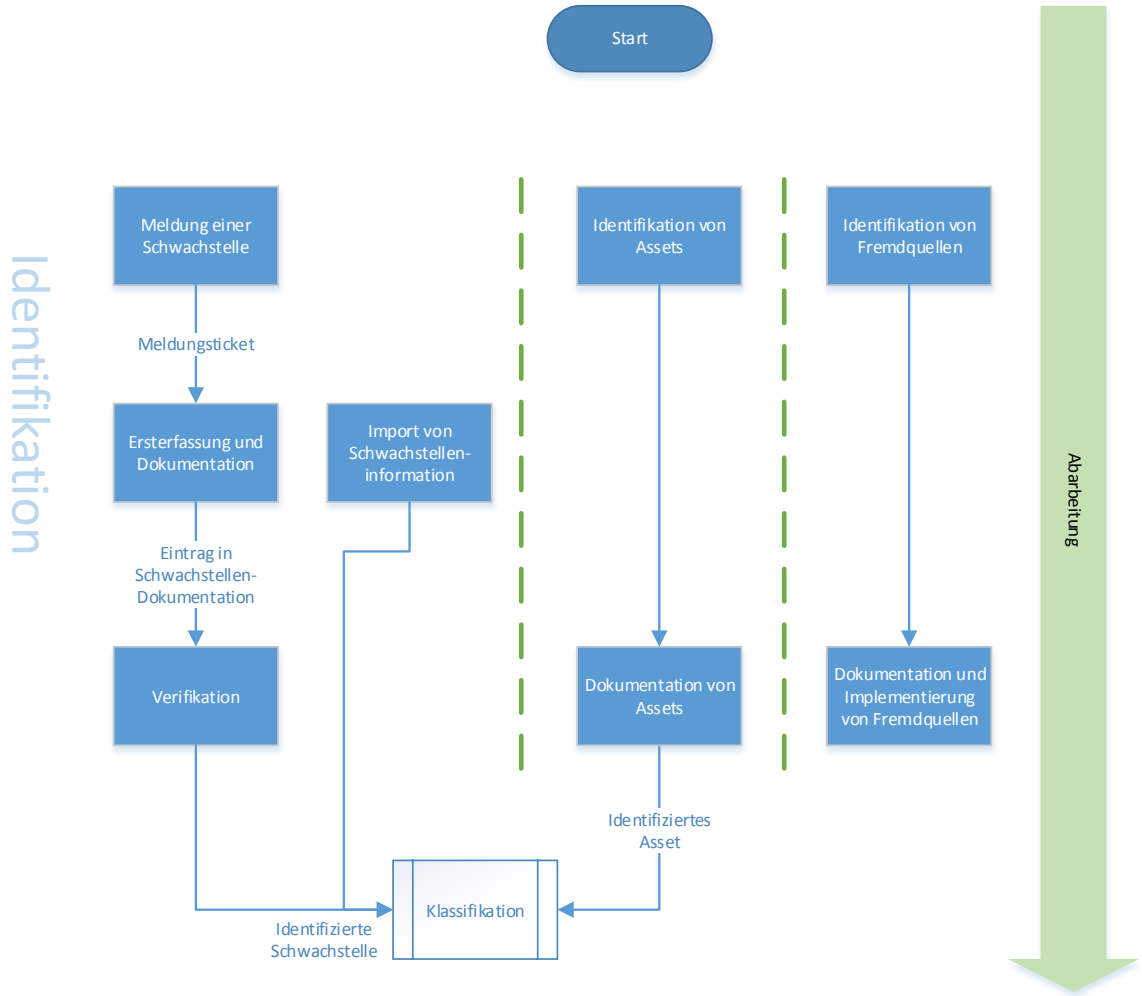


Abbildung 5.5: Teilaktivitäten in der Identifikation von Schwachstellen

Die Teilaktivitäten in der Identifikation sind in Abbildung 5.5 illustriert und gliedern sich grob in drei verschiedene Gruppen:

Zum einen die Identifikation von **Schwachstellen**, wobei die konkrete Vorgehensweise davon abhängt, ob eine Schwachstelle manuell dokumentiert wird oder eine Dokumentation bereits existiert und lediglich aus einer Fremdquelle importiert werden muss. Der wesentliche Unterschied zwischen den Vorgehensweisen liegt in der Notwendigkeit der Verifikation auf die Existenz einer durch Nutzer gemeldeten Schwachstelle, wohingegen importierte Schwachstellen aus Fremdquellen bis auf Weiteres als bestätigt angesehen und somit zur Vermeidung des damit verbundenen hohen Aufwands nicht genauer überprüft werden.

Der zweite Aspekt der Identifikation betrifft relevante **Assets**. Diese werden zunächst identifiziert und zur Sicherstellung der Nutzbarkeit durch entsprechende in Abschnitt 5.3.2.1 beschriebene Attribute und Charakteristika dokumentiert.

Der dritte Aspekt besteht aus der Identifikation von **Fremdquellen** und anschließender Dokumentation sowie Nutzung durch die Implementierung eines angemessenen Moduls zum Import der darin angebotenen Daten.

Ziel ist es, Informationen über Schwachstellen aus verschiedenen Quellen sowie Assets derart zu organisieren, dass sie in der Aktivität der Klassifikation bewertet werden können und im gesamten Schwachstellenmanagement eindeutig identifizierbar sind.

5.5.1 Behandlung von Fremdquellen

Je nach Definition des Umfangs des Anwendungsbereichs spielen Fremdquellen im Schwachstellenmanagement eine fundamentale Rolle bei der Erreichung einer effektiven und effizienten Umsetzung.

Das Ziel der Nutzung von Informationen aus Fremdquellen zur Dokumentation von relevanten Schwachstellen ist in erster Linie die Sicherstellung der Aktualität über den Zustand von Schwachstellen in Softwareprodukten und insofern eine essentielle Maßnahme zum Erhalt von Informationssicherheit.

Zudem ist die Nutzung von Fremdquellen ausschlaggebend zur Automatisierung der Identifikation und in die Klassifikation eingeordnete Teilaktivität der Bewertung und Kategorisierung von Schwachstellen, wodurch letztendlich eine erhebliche Entlastung des im Schwachstellenmanagement tätigen Personals realisiert wird.

5.5.1.1 Hinzufügen einer neuen Quelle

Das Hinzufügen einer neuen Fremdquelle für den Import von Schwachstelleninformationen ist zentral durch Quellenverantwortliche durchzuführen. Erforderlich wird dies dadurch, dass aufgrund der in der Praxis verschiedenen Formate, Informationen und Kommunikationswege in der Regel eine Erweiterung des Funktionsumfangs der Schwachstellen-Dokumentation in Form einer Implementierung der jeweiligen Module notwendig ist. Bei der Entwicklung eines weiteren Moduls für den Import von Schwachstellen aus Fremdquellen in die hochschulnetzweite Datenbank sind vor allem die Bestandteile der Kommunikation mit der Fremdquelle (beispielsweise über HTTP) zur Extraktion der benötigten Daten, z.B. durch einen Parser, sowie – falls vorhanden – zur Bewertung der Schwachstellen mittels einer Abbildung des vorhandenen Bewertungsschemas in das hochschulnetzweite Bewertungsschema notwendig.

Die Entscheidung und Beantragung der Hinzufügung einer Fremdquelle zur zentralen Schwachstellen-Dokumentation wird jeweils durch den verantwortlichen Schwachstellenmanager des jeweiligen Instituts bzw. organisatorischen Bereichs vorgenommen:

Der Antrag wird über das Webportal des Steuerungssystems an alle der im Hochschulrechenzentrum tätigen Quellenverantwortlichen gestellt und beinhaltet

- die Quelle mit Verbindungsinformationen (beispielsweise eine URL und falls notwendig Daten zur Authentifizierung)
- eine quellenspezifische lokale White- bzw. Blacklist an relevanten Softwareprodukten,
- falls notwendig und vorhanden, weitere Quellen zur Bewertung und Kategorisierung der Schwachstellen als auch Quellen für Detektionsverfahren
- sowie ein geeignetes Intervall zur Ausführung eines automatischen Updates der Quelle in Tagen.

Die Quellenverantwortlichen koordinieren je nach Aufwand der Dokumentation und Implementierung die Zuständigkeiten der Aufgaben untereinander. Die den Antrag bearbeitenden Quellenverantwortlichen entscheiden schließlich über die Eignung der Anbindung der neuen Fremdquelle sowie die Konformität zu globalen White- und Blacklisten von Softwareprodukten. Bei identifizierten Inkonsistenzen lokaler Black- und Whitelisten mit globalen Black- und Whitelisten wird die Notwendigkeit entsprechender Einträge mit dem Schwachstellenmanager des beantragenden Instituts abgeklärt. Bei unbedingter Notwendigkeit der geforderten lokalen Black- und Whitelisten wird durch den bearbeitenden Quellenverantwortlichen die Anpassung der globalen Listen bei dem Verantwortlichen, dem Chief Vulnerability Manager, beantragt, welcher über eine Anpassung entscheidet.

Mögliche Resultate sind die Anpassung der globalen oder auch lokalen White- und Blacklisten oder die Ablehnung des Antrags zum Hinzufügen der jeweiligen Fremdquelle.

Bei Hinzufügen jeder neuen Fremdquelle wird nicht nur der den Antrag stellende Schwachstellenmanager diesbezüglich informiert, sondern alle Schwachstellenmanager im Hochschulnetz, da die Fremdquelle hochschulnetzweit genutzt wird und eine geeignete Quelle für weitere Schwachstellen sein kann.

5.5.1.2 Dokumentation und Implementierung

Fremdquellen müssen üblicherweise jeweils einzeln implementiert werden. Als Grundlage dazu dienen die in der Schwachstellen-Dokumentation abgelegten Informationen über Fremdquellen, die bei der Implementierung verwendet werden sowie Funktionen, die durch die Schwachstellen-Dokumentation bereitgestellt werden (z.B. zum Hinzufügen einer Schwachstellen). Die Zuständigkeit dafür liegt bei den bearbeitenden Quellenverantwortlichen. Der Inhalt, der aus einer Fremdquelle importiert wird, umfasst mindestens die Beschreibung der Schwachstelle und wenn möglich ebenfalls eine Bewertung der Schwachstelle, welche auf einen Wert des Bewertungssystems des Hochschulnetzes abgebildet werden kann (vgl. Abschnitt 5.6.1.2) sowie betroffene Softwareprodukte und zusätzlich möglicherweise eine Quelle für Detektionsverfahren.

Durch die Implementierung durchläuft die importierte Schwachstelle die Aktivitäten der Identifikation und möglicherweise zusätzlich der Klassifikation daher nicht manuell, sondern automatisch.

5.5.2 Meldung und Ersterfassung von Schwachstellen

Die manuelle Meldung einer neuen Schwachstelle kann durch jeden Nutzer im Hochschulnetz erfolgen, welchem durch die Meldung implizit die Rolle des Schwachstellen-Melders zugewiesen wird. Die Meldung erfolgt an eine zentrale Stelle zur Verteilung der Meldungen an die im Schwachstellenmanagement tätigen Schwachstellen-Dokumentatoren, welche die Meldung zunächst auf ihre Plausibilität hin überprüfen und die Meldung im negativen Fall verwerfen bzw. nicht weiter verfolgen.

Der bearbeitende Schwachstellen-Dokumentator überprüft auch, ob die gemeldete Schwachstelle bereits in der Schwachstellen-Dokumentation eingetragen und somit ein Duplikat ist. Die Prüfung wird anhand des Abgleichs mit ähnlichen Schwachstellen gemacht. Ähnliche Schwachstellen teilen sich zumindest entweder das betroffene Softwareprodukt sowie dessen Hersteller oder den betroffenen IT-Dienst. Die Version des Softwareprodukts kann ebenfalls ein Anhaltspunkt für die Ähnlichkeit sein. Dabei muss jedoch beachtet werden, dass sich Schwachstellen in Software über mehrere Versionen hinweg erstrecken können.

Die Meldung erfolgt in Form einer zentralen Liste aus `Meldungstickets`, wobei alle Einträge durch potenziell sämtliche Schwachstellen-Dokumentatoren bearbeitet werden.

Falls die Meldung als plausibel eingestuft wird, ist die Überprüfung des Informationsumfangs der Meldung durch den Schwachstellen-Dokumentator vorzunehmen. Informationen, die bei der Meldung ersichtlich sein müssen, sind

- das betroffene **Softwareprodukt**, der **Hersteller** des Softwareprodukts, betroffene **Versionen**,

  oder der **Dienst** im Hochschulnetz, in dem die Schwachstelle identifiziert wurde,

- das exakte **Vorgehen**, welches zur Ausnutzung führt bzw. führen kann sowie

- aus der Ausnutzung **hervorgerufenes Verhalten**.

Sie dienen der anschließenden Verifikation der Existenz der Schwachstelle sowie allgemein der Dokumentation für die Anwendung einer nachvollziehbaren Klassifikation, Beseitigung bzw. Abschwächung sowie zur Erstellung von Maßnahmen zur Prävention der Schwachstelle.

Bei Fehlen einer der genannten verpflichtenden Kombination von Informationen und dennoch plausibler Einschätzung der Schwachstelle ist die Nachfrage nach diesen Informationen beim Schwachstellen-Melder durch den bearbeitenden Schwachstellen-Dokumentator vorzunehmen. Diese Nachfrage erfolgt über die im Kontaktdaten-Feld hinterlegten Informationen des jeweiligen Meldungstickets und umfasst die konkrete Nachfrage über fehlende Informationen sowie eine Kontaktinformation (beispielsweise eine E-Mail-Adresse), an die der Schwachstellen-Melder antworten kann.

Ist andernfalls ausreichend Information zur weiteren Verarbeitung der Schwachstelle vorhanden, so erstellt der Schwachstellen-Dokumentator einen neuen Eintrag in der Schwachstellen-Dokumentation mit dem Wert *check* für den Status zur Signalisierung an den Schwachstellen-Prüfer, dass die Schwachstelle plausibel wirkt, die Existenz jedoch weder widerlegt noch bestätigt ist. Bei Erstellung des Eintrages der Schwachstelle wird im Meldungsticket dessen Identifikator hinterlegt.

Beim Eintragen der `Schwachstellenbeschreibung` werden genau die oben genannten Informationen als Freitext so verfasst, dass diese für weitere Personen direkt und so konkret wie möglich ersichtlich sind.

Falls eine Schwachstellenmeldung bereits einem Eintrag ähnelt oder mit diesem bis auf die betroffene Version des Softwareprodukts identisch ist, so wird der ähnliche, ältere Schwachstelleneintrag in den Referenzen des neuen Eintrags vermerkt, so dass der Schwachstellen-Prüfer im nächsten Schritt eine Prüfung der Relation beider Schwachstellen zueinander vornimmt. Die Zuweisung der Verantwortlichkeit erfolgt automatisch durch das Steuerungssystem.

5.5.3 Verifikation der Existenz von Schwachstellen

Der Schritt der Verifikation von Schwachstellen folgt in der Regel auf die manuelle Meldung einer Schwachstelle durch einen Schwachstellen-Melder und die Plausibilitätsprüfung durch einen Schwachstellen-Dokumentator. Zu diesem Zeitpunkt gibt es in der Schwachstellen-Dokumentation bereits einen Eintrag einer Schwachstelle mit grundlegenden Informationen zur Identifikation (siehe vorheriger Abschnitt) und dem Status *check*.

Die Verifikation der Existenz einer potenziellen Schwachstelle wird durch einen Schwachstellen-Prüfer vorgenommen. Falls es sich bei dem von der Schwachstelle betroffenen Softwareprodukt um eine Komponente eines im Hochschulnetz angebotenen Dienstes handelt oder die Software eine Eigenentwicklung eines Instituts im Hochschulnetz ist, nimmt die Rolle des Schwachstellen-Prüfers eine Person ein, die entsprechend technisches Fachwissen zum Dienst und der betroffenen Komponente besitzt. Zu diesem Zweck kann ein Schwachstellen-Prüfer die Verantwortung an eine geeignetere Person in dieser Rolle eskalieren. Andernfalls wird die Überprüfung einem beliebigen Schwachstellen-Prüfer zugewiesen.

Falls durch die Referenz auf einen weiteren Eintrag in der Schwachstellen-Dokumentation, auf die Ähnlichkeit zu diesem hingewiesen wird, erfolgt zunächst die Überprüfung auf die Gleichheit beider Schwachstellen. Falls sie sich als eindeutig identisch erweisen, wird der alte Eintrag gegebenenfalls um weitere, neue Informationen aus dem neu-gemeldeten Eintrag erweitert (beispielsweise eine weitere betroffene Softwareversion). Der neu-gemeldete Eintrag wird auf Status *reject* gesetzt.

Sind keine möglichen Duplikate genannt oder die Gleichheit ähnlicher Schwachstellen bestätigt sich nicht, so wird die Überprüfung durch das Nachvollziehen des vom Schwachstellen-Melders mitgeteilten Verfahrens zur Ausnutzung der jeweiligen Schwachstelle durchgeführt.

Dabei muss darauf geachtet werden, die Durchführung mit möglicherweise dadurch beeinträchtigten Personen oder Asset-Verantwortlichen abzusprechen.

Falls die Überprüfung erfolgreich durchgeführt sowie das beschriebene Verhalten beobachtet werden kann, oder die Vorgehensweise auch ohne Ausführung plausibel erscheint, ist die Ausnutzbarkeit der potenziellen Schwachstelle bestätigt. Um die potenzielle Schwachstelle komplett zu bestätigen, fehlt ab hier der Aspekt des Nutzens der Ausnutzung, generell durch die Verletzung einer oder mehrerer der Ziele der Informationssicherheit (Vertraulichkeit, Integrität und Verfügbarkeit) oder dem Entstehen von Schaden durch die Ausnutzung. Kann die Analyse des Schwachstellen-Prüfers auch letzteren Aspekt bestätigen, gilt der gemeldete Eintrag als Schwachstelle und muss insofern als solche gekennzeichnet werden, wird also von

seinem aktuellen Status *check* auf *assess* gesetzt, da der nächste Schritt aus der Bewertung der Schwachstelle besteht. Die Bewertung bzw. Festlegung der Priorität wird in Abschnitt 5.6.1 beschrieben.

Falls entweder das Verfahren zur Ausnutzung der Schwachstelle nicht erfolgreich repliziert und nachvollzogen oder im Laufe der Analyse durch den Schwachstellen-Melder keine Beeinträchtigung der drei Ziele der Informationssicherheit festgestellt werden kann, so ist die Existenz der potenziellen Schwachstelle als nicht-bestätigt anzusehen und wird daher durch Setzen des Status auf *reject* entsprechend gekennzeichnet.

5.5.4 Identifikation von Assets

Die Identifikation von Assets mit Relevanz für das Schwachstellenmanagement wird insbesondere durch den für das jeweilige Institut potenziell unterschiedlichen Anwendungsbereich festgelegt, wodurch der Schwachstellenmanager im Endeffekt für die Auswahl der Assets verantwortlich ist.

Die Dokumentation jedes im Anwendungsbereich liegenden Assets ist ein essentieller Faktor, von dem die Wirksamkeit des Schwachstellenmanagements effektiv abhängig ist, da ein Überblick über Assets eine der Grundvoraussetzungen für ein funktionierendes Management ist. Die Verantwortung für die Dokumentation sowie Sicherstellung der Aktualität in der Asset- und Benutzerverwaltung für das eigene Asset liegt bei dem jeweiligen Asset-Verantwortlichen.

Zum Erhalt der Aktualität eines Asset-Eintrages muss der jeweilige Asset-Verantwortliche bei Änderung der Konfiguration in Bezug auf eine der in Abschnitt 5.3.2.1 beschriebenen, gehalteten Informationen seines Assets Änderungen direkt in die Asset- und Benutzerverwaltung einpflegen. Auch muss er durch regelmäßige Kontrollen die Aktualität bestätigen. Die Dokumentation von Assets wird von den darauf installierten Detektionsagenten unterstützt, wodurch Daten bei Änderungen automatisiert in die Asset- und Benutzerverwaltung eintragen werden. Auch wird die Aktualität durch Anbindung der CMDB gewährleistet. Diese aktualisiert bei Änderungen, die in ihr vorgenommen werden ebenfalls die Daten in der Asset- und Benutzerverwaltung durch den automatisierten Zugriff über die API des Steuerungssystems.

Die Teilaktivität der Identifikation von Assets dient als Voraussetzung für die Klassifikation von Assets, die im Anschluss folgt.

5.6 Klassifikation von Schwachstellen

Die Klassifikation von Schwachstellen setzt sich aus mehreren Bestandteilen zusammen. Kernelemente bilden die Bewertung von Schwachstellen mittels eines geeigneten, den Anforderungen des Hochschulumfelds entsprechenden Bewertungssystems sowie einer Risikobewertung von Assets. Beide Bewertungen zusammen bilden die Grundlage zur Ermittlung der Notwendigkeit einer Behandlung einer Schwachstelle auf einem bestimmten Asset, welche zum einen – abgeglichen an dem vorher definierten **Schwachstellenakzeptanzkriterium** (siehe Abschnitt 5.6.4) – die Relevanz einer Schwachstelle auf dem jeweiligen Asset für die

Beseitigung und Abschwächung festlegt und zum anderen die Basis der Behandlungspriorisierung darstellt.

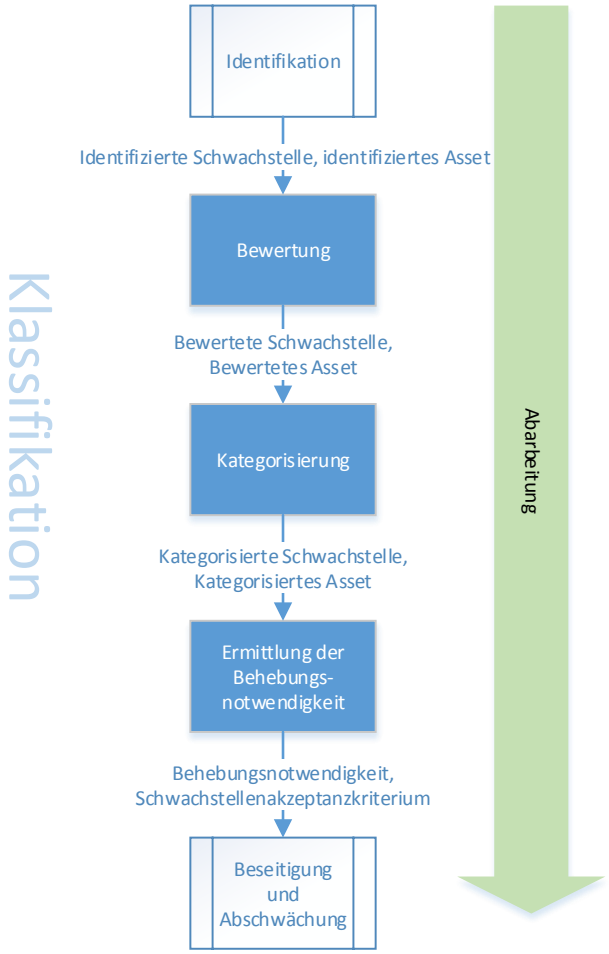


Abbildung 5.6: Teilaktivitäten in der Klassifikation von Schwachstellen

Ein weiterer Aspekt in der Aktivität der Klassifikation ist die Kategorisierung von Schwachstellen anhand betroffener Softwareprodukte. Der Zweck der Kategorisierung ist vor allem eine Verbesserung der Verwaltbarkeit sowie zur Unterstützung technischer Funktionen wie der Suche und Filterung von Informationen.

Ausgaben der Klassifikation sind insofern bewertete, kategorisierte Schwachstelleneinträge, nach Risiko eingestufte Assets, sowie eine Bewertung der Notwendigkeit zur Behandlung einer bestimmten Schwachstelle auf einem konkreten Asset.

Alle Teilaktivitäten und für die Verarbeitung notwendige Informationen sind in Abbildung 5.6 illustriert und werden in den folgenden Abschnitten beschrieben.

5.6.1 Bewertung von Schwachstellen

Die Zuweisung eines Wertes zur Beschreibung der Kritikalität einer Schwachstelle wird direkt nach ihrer Beschreibung und Verifikation vorgenommen. Die Zuständigkeit für die manuelle Bewertung von Schwachstellen liegt bei den Schwachstellen-Bewertern. Zu diesem Zweck wird im folgenden Abschnitt ein Bewertungssystem für ein Schwachstellenmanagement in Hochschulnetzen beschrieben.

Zu beachten ist, dass eine manuelle Bewertung zur Reduzierung des Aufwands und daraus folgender Entlastung der Ressourcen im Hochschulnetz, insbesondere Mitarbeiter, weitestgehend vermieden werden muss und nur für nicht-automatisiert bewertbare Einträge durchgeführt wird.

Daher werden bereits vorhandene Bewertungen von Schwachstellen aus Fremdquellen, falls vorhanden, weitergenutzt und zum Erhalt der Einheitlichkeit der Bewertung auf das im Schwachstellenmanagement genutzte Bewertungssystem abgebildet. Verantwortlich für die Suche und Wahl der Fremdquellen für die automatisierte Bewertung von Schwachstellen sowie die Entwicklung einer Abbildung des Bewertungssystems (vgl. Abschnitt 5.6.1.2) sind die Quellenverantwortlichen. Neu-entwickelte Abbildungen eines Bewertungssystems auf das im Hochschulnetz genutzte Bewertungssystem werden exakt dokumentiert und durch den Chief Vulnerability Manager auf ihre Eignung geprüft. Im positiven Falle kann die Abbildung allgemein im gesamten Schwachstellenmanagement angewendet werden. Andernfalls muss diese überarbeitet oder im Falle einer unbedingten Nicht-Eignung, die manuelle Bewertung der betroffnenen Schwachstellen in Betracht gezogen werden.

Die manuelle Bewertung von Schwachstellen wird über ein geeignetes, zentral durch das Steuerungssystem angebotenes Webportal unterstützt.

Kann eine Schwachstelle aufgrund von zu wenig Information in der Beschreibung nicht bewertet werden, dann wird dem entsprechenden Schwachstelleneintrag in der Schwachstellen-Dokumentation der Status *incomplete* zugewiesen. In diesem Fall kann durch Absprache des aktuell bewertenden Schwachstellen-Bewerters mit dem für die Verifikation zuständigen Schwachstellen-Prüfer die fehlende Information identifiziert und dokumentiert werden. Falls notwendig, wird für diesen Zweck des Weiteren beim Schwachstellen-Melder nachgefragt, dessen Kontaktdaten (beispielsweise in Form einer E-Mail-Adresse) im dazugehörigen `Meldungsticket` vorliegen.

Ist eine Schwachstelle erfolgreich bewertet, wird der Status ihres Eintrages auf *categorize* gesetzt, um zu verdeutlichen, dass der nächste Schritt aus der Kategorisierung des Eintrages besteht (Abschnitt 5.6.2).

5.6.1.1 Bewertungssystem für Schwachstellen

Ein Bewertungssystem für Schwachstellen im Hochschulnetz dient im Allgemeinen zwei verschiedenen Anwendungen: Einerseits als Bestandteil bei der Ermittlung der Behebungsnotwendigkeit einer Schwachstelle und andererseits für die Einschätzung zur Notwendigkeit der Dokumentation der Schwachstelle.

Aufgrund der umfangreichen und ausführlichen Dokumentation des *Common Vulnerability Scoring System* ist das in diesem Abschnitt beschriebene System als diesbezügliche Vereinfachung zu verstehen, mit dem Ziel der Verwendung einer Teilmenge der darin beschriebenen

Charakteristika von Schwachstellen zur Erreichung einer praxisorientierten Anwendbarkeit im Schwachstellenmanagementprozess.

Gleichzeitig ist durch die Anlehnung an das weit verbreitete CVSS die Möglichkeit einer Abbildung (engl. „mapping“) des CVSS direkt sowie potenziell weiterer Bewertungssysteme auf das Konzeptsystem möglich. Eine Vorgehensweise zur Abbildung eines Bewertungssystems auf das Konzeptsystem wird im auf diesen folgenden Abschnitt erläutert.

Eine der grundlegenden Herausforderungen bei der Entwicklung eines Bewertungssystems für Schwachstellen stellt die Granularität der Unterscheidbarkeit der Kritikalität von Schwachstellen dar. Dabei ist die Vorgehensweise grundlegend unterschiedlich. Bewertungssysteme wie das aktuelle CVSS Version 3 können die Kritikalität einer Schwachstelle anhand verschiedener Aspekte (hier: den Kerncharakteristika, zeitliche sowie umgebungsbezogene Charakteristika) auf eine Punktzahl zwischen 0.0 bis 10.0 mit einer Genauigkeit auf eine Nachkommastelle unterscheiden. Als Gegensatz dazu wird beispielsweise durch das Bewertungssystem in den Microsoft Security Bulletins lediglich eine Unterscheidung von vier Kategorien angeboten – „niedrig“, über „mittel“ und „hoch“ bis „kritisch“. [Mic11]

Der Vorteil einer feingranularen Unterscheidung der Kritikalität besteht aus einer im Endeffekt besseren Priorisierbarkeit von Schwachstellen, wohingegen der Nachteil darin besteht, dass stark feingranulare Bewertungsschemata zum einen bei Anwendern und Nutzern in der Praxis eher zur Verwirrung beitragen und sich weiterhin die Abbildung bereits in Fremdquellen verwendeter, grobgranularer Bewertungssysteme auf ein feingranulares System als schwierig erweist.

Ein Beispiel zu letzterem Fall ist die in der offiziellen Spezifikation [For15b] beschriebene Abbildung eines CVSS Version 3-Wertes auf ein fünfstufiges, qualitatives Schema und wird in Tabelle 5.1 verdeutlicht.

CVSS Score	**Bewertung**
0.0	None
0.1 – 3.9	Low
4.0 – 6.9	Medium
7.0 – 8.9	High
9.0 – 10.0	Critical

Tabelle 5.1: Abbildung des CVSSv3 Scores auf ein fünfstufiges Bewertungsschema

Hingegen ist die Abbildung in die andere Richtung, d.h. von einem Wert des fünfstufigen Systems zurück auf einen CVSS-Score, nicht eindeutig möglich, da ohne weitere Informationen Beschreibungsdetails fehlen und lediglich ein Intervall bekannt ist, in dem das Ergebnis liegt.

Da der automatische Import von Schwachstellen und deren Bewertung in der Schwachstellen-Dokumentation mittels einer geeigneten Abbildung ein wesentlicher Bestandteil des Konzeptes ist, bildet das in dieser Arbeit beschriebene Bewertungssystem die im Laufe des Abschnitts genannten Charakteristika von Schwachstellen auf vier mögliche Werte der Kritikalität einer Schwachstelle ab:

niedrig, **mittel**, **hoch**, **kritisch**

Dabei ist zu beachten, dass „niedrig“ die geringste Kritikalitätsstufe darstellt und „kritisch“ die höchste. Des Weiteren bestimmt diese Bewertung nicht die Behebungsnotwendigkeit bzw. die Bearbeitungspriorität, sondern stellt lediglich einen Teil dieser Bestimmung dar.

Als Generalisierung zum Common Vulnerability Scoring System Version 3 und einer daraus angestrebten resultierenden Erleichterung der Anwendbarkeit des Systems im Hochschulumfeld, werden nicht alle im CVSSv3 verwendete Charakteristika von Schwachstellen bei der Bewertung betrachtet, sondern lediglich eine Teilmenge, welche als Kernelemente anzusehen sind.

Die Bestimmung der Kritikalität einer Schwachstelle ist – angelehnt an die Vorgehensweise zur Risikobestimmung im Informationssicherheitsmanagement – abhängig von Werten zweier Gruppen: Zum einen Faktoren zur Bestimmung der Wahrscheinlichkeit der Ausnutzung einer Schwachstelle sowie zum anderen Faktoren zur Beschreibung der Auswirkungen als Folge einer Ausnutzung der Schwachstelle.

Demnach werden zur Bestimmung der **Wahrscheinlichkeit** einer Ausnutzung der jeweiligen Schwachstelle aus den Base Metrics die Charakteristika des „Access Vectors“ sowie der „Attack Complexity“ zur Bewertung einer Schwachstelle genutzt. Zusätzlich wird als zeitlicher Faktor die *Bekanntheit der Ausnutzung der Schwachstelle* verwendet.

Faktoren zur Bestimmung der **Auswirkung** in Folge einer Ausnutzung einer Schwachstelle sind wie im CVSSv3 der Einfluss auf den Erhalt der Ziele in der Informationssicherheit, der *Vertraulichkeit*, *Integrität* und *Verfügbarkeit*.

Eine Kompensation der weiteren Charakteristika erfolgt durch den Faktor der *Temporären Dringlichkeit*.

Auch die Werte, die jedem einzelnen Charakteristikum zur Bestimmung der Kritikalität zugewiesen werden können, haben sich im Vergleich zur Genauigkeit im CVSS vereinfacht, da die zuständigen Personen in der Rolle des Schwachstellen-Bewerters im Hochschulumfeld möglicherweise keine Experten im Umgang mit Schwachstellen sind und entsprechend diese bei einer hohen Zahl an sich ähnelnden Antwortmöglichkeiten nicht in der Lage sind, eine Schwachstelle komplett korrekt zu bewerten. Auch muss – neben der Fehlertoleranz – der Aufwand, welcher mit einer derartigen Bewertung verbunden ist, bei der Wahl eines passenden Bewertungssystems beachtet und weitestgehend minimiert werden. Daher wird bei

der Bewertung ein gröberes, jedoch zur Korrektheit der Bewertung beitragendes Spektrum an Antwortmöglichkeiten einem feingranulareren vorgezogen. Die möglichen Werte zu den jeweiligen Charakteristika sind in Tabelle 5.2 zusammengefasst.

Charakteristikum	**mögliche Werte**
Möglichkeit der Ausnutzung über das Netz (*Access Vector*)	nein, ja
Aufwand zur Ausnutzung (*Attack Complexity*)	hoch, gering
Bekanntheit von Fällen der Ausnutzung	nein, ja
Beeinträchtigung der Vertraulichkeit	nein, ja
Beeinträchtigung der Integrität	nein, ja
Beeinträchtigung der Verfügbarkeit	nein, ja
Temporäre Dringlichkeit	NA, gering, mittel, hoch, kritisch

Tabelle 5.2: Charakteristika und mögliche Werte bei der Bewertung von Schwachstellen

- Das Charakteristikum der **Möglichkeit der Ausnutzung über das Netz** trägt erheblich zur Ausnutzbarkeit einer Schwachstelle bei. Im Falle, dass die zu bewertende Schwachstelle über das Netz ausgenutzt werden kann, ist *ja* anzugeben, sonst, falls die Möglichkeit zur Ausnutzung der Schwachstelle ausschließlich lokal erfolgen kann, ist das Kriterium *nein* zu bewerten.

- Der **Aufwand zur Ausnutzung** der Schwachstelle ist genauso lediglich anhand zweier Werte festzulegen: *gering*, im Falle, dass ein Angreifer zur Ausnutzung der Schwachstelle keine weiteren Voraussetzungen benötigt und die Ausnutzung wiederholbar ist, als auch falls ein Exploit für die Schwachstelle existiert. Andernfalls ist dieses Charakteristikum mit dem Wert *hoch* zu bewerten, beispielsweise falls die Ausnutzung der Schwachstelle einen hohen Vorbereitungsaufwand erfordert (z.B. Aufklärungsaufwand des Opfersystems oder Aufwand zur Manipulation der IT-Umgebung) [For15b]. Auch wenn die Notwendigkeit weiterer Rechte bzw. einer erfolgreichen Authentifizierung im CVSSv3 separiert und unter dem Aspekt *Privileges Required* aufgelistet wird, so wird dieser Aspekt in diesem Konzept aus Gründen der Vereinfachung und Anwendbarkeit des Bewertungssystems zum Faktor des zu betreibenden Aufwands einer Ausnutzung zugerechnet. Werden für die Ausnutzung einer Schwachstelle also weitere Nutzer- oder Administratorrechte bzw. eine erfolgreiche Authentifizierung benötigt, ist dieser Faktor ebenfalls mit *hoch* zu bewerten.

- Das letzte Kriterium zur Bestimmung der Wahrscheinlichkeit einer Ausnutzung einer Schwachstelle ist das öffentliche **Wissen über Fälle einer Ausnutzung** der Schwachstelle. Bei Zutreffen ist der Aspekt mit *ja* zu beantworten, ansonsten mit *nein*. Da, wie bereits beschrieben, die Pflege zeitabhängiger Faktoren sehr aufwendig sein kann, wird das Setzen des Wertes durch Verknüpfung des Schwachstellenmanagements mit dem *Security Information & Event Management* realisiert. Wird in diesem ein Ereignis gemeldet, das mit der Ausnutzung einer Schwachstelle in Verbindung steht, so wird (falls nicht bereits getan) dieser Faktor mit *ja* bewertet. Dies passiert je nach technischen Möglichkeiten im SIEM-Prozess automatisch durch Setzen des Wertes über die API

oder andernfalls manuell über einen Nutzer in der Rolle des Schwachstellen-Bewerters, der in beiden Management-Prozessen tätig ist.

Ein analoges Vorgehen ist auch bei den Charakteristika von Schwachstellen zur Bestimmung der Auswirkungen einer Ausnutzung gegeben. Die Bestimmung einer Beeinträchtigung des jeweiligen Schutzziels in der Informationssicherheit ist einerseits mit *ja* anzugeben, falls das jeweilige Schutzziel bei einer Ausnutzung der Schwachstelle nicht gewährleistet werden kann, oder zum anderen mit *nein*, falls das jeweilige Schutzziel nicht beeinträchtigt wird:

- **Vertraulichkeit** – „Schutz von Information vor unberechtigter Offenlegung“ [Sch14e]
- **Integrität** – Schutz von Information vor unberechtigter Modifikation, Einfügungen, Löschungen, Umordnung, Duplizierung oder Wiedereinspielung [Sch14e]
- **Verfügbarkeit** – „Sicherstellung der Zugänglichkeit und Nutzbarkeit von Informationen für berechtigte Entitäten“ [Sch14e]

Auch hier werden zur Vereinfachung nicht wie im CVSS drei Werte zur Beschreibung des Einflusses auf das jeweilige Schutzziel verwendet (im CVSS: None, Partial, Complete), sondern, wie beschrieben, lediglich zwei, um den Aufwand der manuellen Bewertung zu minimieren. Auch wenn dadurch möglicherweise eine ungenauere Bewertung resultiert, wird die Anwendbarkeit erhöht.

Das Element der **Temporären Dringlichkeit** dient als korrigierender, zeitlich begrenzter Faktor zur Anpassung der Priorität einer Schwachstelle. Da sich zeitliche Charakteristika von Schwachstellen als sehr pflegeintensiv erweisen, da sie sich mit der Zeit ändern und aktuell gehalten werden müssten, ist es in einem Hochschulumfeld schwierig, diese mit begrenztem Personal zu verwalten. Eine Lösung ist ein Sonderkriterium, durch das die Kritikalität der Schwachstelle für einen gewissen konfigurierbaren Zeitraum manuell gesetzt werden kann.

Die Festlegung der Temporären Dringlichkeit ist dem Chief Vulnerability Manager vorbehalten. Mögliche Werte sind *NA* („not available“), falls die Temporäre Dringlichkeit nicht beachtet werden soll, oder der entsprechende Wert der Kritikalität (*gering*, *mittel*, *hoch*, *kritisch*), falls die Kritikalität manuell festgelegt ist. Da die Herausgabe von Schwachstellenpatches durch Softwarehersteller in der Praxis selten länger als einen Monat dauert (z.B. Microsofts „Patchday“) und betroffene Software in der Regel in einer ähnlich großen Zeitspanne ersetzt ist, wird die Temporäre Dringlichkeit nach einer Dauer von zwei Monaten auf *NA* zurückgesetzt. Ist diese Zeitdauer nicht ausreichend oder zu lang, kann der Wert manuell erneut korrigiert werden.

Schwachstellenmanager haben keinen Einfluss auf den Wert der Temporäre Dringlichkeit einer Schwachstelle, da sie als netzweites Mittel zur Festlegung einer vom Bewertungsschema abweichenden Bewertung für Schwachstellen dient und im Konzept für Einzelfälle vorgesehen ist. Ein Beispiel für einen hierfür relevanten Einzelfall ist der Heartbleed Bug, dessen Bewertung im CVSS Version 2 lediglich einen BaseScore von 5.0 erreicht, die Schwachstelle abhängig von durch über ihre Ausnutzung ausgelesenen Daten und ihrem hohen Verbreitungsgrad jedoch praktisch ein deutlich höheres Schadenspotenzial aufweist (vgl. Abschnitt 2.3.4).

Faktoren die im CVSSv3 zur Evaluation der Kritikalität einer Schwachstelle herangezogen werden, in diesem Konzept jedoch außen vor gelassen werden, sind die „User Interaction“ sowie der „Scope“ aus der Base Metric Group sowie die komplette Temporal Metric Group.

Die „User Interaction“ ist aufgrund ihrer inhaltlichen Überschneidung mit dem Charakteristikum des Ausnutzungsaufwands nicht bei der Bewertung mit eingeflossen. Durch den Aspekt der Wiederholbarkeit, der durch eine Abhängigkeit von Nutzereingaben nicht immer gegeben ist, sind derartige Fälle Teil des Aufwands zur Ausnutzung einer Schwachstelle.

Der im CVSS vertretene Aspekt des „Scope“ einer Schwachstelle (siehe Abschnitt 5.6.1) ist für die Umsetzung eines Schwachstellenmanagements im Hochschulumfeld weniger geeignet, da die Einschätzung dieser Metrik in der Regel ein gutes technisches Verständnis und Fachwissen voraussetzt, das nicht alle Nutzer im Schwachstellenmanagement mitbringen. Insofern wird dieses Kriterium nicht bei der Bewertung von Schwachstellen herangezogen.

Die Vorgehensweise zur Berechnung der Ausnutzungswahrscheinlichkeit ist in Formel 5.1 dargestellt, wobei die Bedeutung der einzelnen Abkürzungen und Werte ihrer Auswahlmöglichkeiten in Tabelle 5.3 definiert sind.

$$W_S = likelihood(N, L, B) \tag{5.1}$$

Die Berechnung der Auswirkung hingegen wird durch die in Formel 5.2 gezeigte und im Verlauf des Kapitels definierte Funktion *impact* beschrieben.

$$A_S = impact(C, I, A) \tag{5.2}$$

Abkürzung	**Charakteristikum**
K	Kritikalität der Schwachstelle
W_S	Wahrscheinlichkeit einer Ausnutzung der Schwachstelle
A_S	Auswirkung der Ausnutzung
N	Möglichkeit der Ausnutzung über das Netz
L	Aufwand zur Ausnutzung
B	Bekanntheit von Fällen der Ausnutzung
C	Beeinträchtigung der Vertraulichkeit
I	Beeinträchtigung der Integrität
A	Beeinträchtigung der Verfügbarkeit
T	Temporäre Dringlichkeit

Tabelle 5.3: Abkürzungen der einzelnen Charakteristika zur Bewertung von Schwachstellen

Die Funktion zur Berechnung der Wahrscheinlichkeit der Ausnutzung der bewerteten Schwachstelle anhand der gegebenen Parameter ist in folgendem, an Pascal angelehnten Pseudocode, beschrieben (Listing 5.1).

Listing 5.1: Funktion zur Berechnung der Ausnutzungswahrscheinlichkeit einer Schwachstelle

```
1  type
2    Likelihood = (gering, mittel, hoch, kritisch);
3
4  function likelihood(L, N, B) : Likelihood;
5  begin
6    if (L and N and B)
7      then result := kritisch
8    else if ((L and N) or (N and B))
9      then result := hoch
10   else if ((L and B) or N)
11     then result := mittel
12   else result := gering;
13 end;
```

Die Abkürzungen der Parameter der Funktion sind gleichbedeutend mit denen aus Tabelle 5.3, stellen jedoch boolesche Werte dar. Falls Parameter `L` den Wert `true` hat, entspricht das dem Wert *gering* des Aufwands zur Ausnutzung der Schwachstelle, andernfalls *hoch*. Ähnliches gilt für die Werte der Parameter `N` sowie `B`, wobei der Wert `true` gleichbedeutend mit *ja* und ein `false` gleichbedeutend mit *nein* ist (siehe Tabelle 5.2).

Sind alle drei Aspekte erfüllt (ein geringer Aufwand der Ausnutzung, die Ausnutzbarkeit über das Netz sowie die Bekanntheit von Fällen der Ausnutzung) ist die Wahrscheinlichkeit der Ausnutzung der Schwachstelle als **kritisch** zu bewerten.

Entsprechend etwas weniger wahrscheinlich ist die Ausnutzung der Schwachstelle, falls entweder keine Fälle bekannt sind, die Schwachstelle theoretisch jedoch leicht und zudem über das Netz ausnutzbar ist, oder die Komplexität der Ausnutzung der Schwachstelle hoch ist, die Möglichkeit einer Ausnutzung über das Netz besteht und zugleich Fälle der Ausnutzung bekannt sind. Beide Fälle werden als **hoch** eingestuft.

Die Kombination aus einer leicht ausnutzbaren Schwachstelle mit bereits bekannten Fällen der Ausnutzung ist als **mittel** zu bewerten, da hier ein wichtiger Aspekt fehlt, der die Wahrscheinlichkeit einer Ausnutzung der Schwachstelle stark erhöht: Die Ausnutzbarkeit über das Netz. Gleiches gilt, falls die Ausnutzung über das Netz als einziger Faktor gegeben ist, da dadurch potenziell eine hohe Anzahl an Bedrohungen existieren.

Ist ausschließlich entweder eine einfache Ausnutzbarkeit oder die Bekanntheit von Fällen zur Bestimmung der Ausnutzungswahrscheinlichkeit erfüllt, kann diese als **gering** eingestuft werden, da so immer die Möglichkeit zur Ausnutzung über das Netz fehlt und somit eine Ausnutzung als unwahrscheinlich zu betrachten ist.

Die Funktion zur Berechnung der Auswirkungen der Ausnutzung der Schwachstelle ist in Listing 5.2 in Pseudocode definiert.

Listing 5.2: Funktion zur Abbildung der drei Schutzziele auf einen Wert zur Bestimmung der Auswirkungen einer Ausnutzung einer Schwachstelle

```
1
2  type
3    Impact = (NA, mittel, hoch, kritisch);
4
5
6  function impact(C, I, A) : Impact;
7  begin
8    if (C and I and A)
9      then result := kritisch
10   else if ((C and I) or (C and A) or (I and A))
11     then result := hoch
12   else if (C or I or A)
13     then result := mittel
14   else result := NA;
15 end;
```

Die Auswirkung wird anhand der Anzahl der beeinträchtigten Ziele der Informationssicherheit ermittelt: Sind alle drei Ziele, das heißt die Vertraulichkeit, Integrität und die Verfügbarkeit betroffen, ist die Auswirkung der Schwachstelle mit **kritisch** bewertet. Ist eine Kombination aus zwei Zielen der Informationssicherheit betroffen, ist das Schadenspotenzial durch die Ausnutzung der Schwachstelle **hoch**. Wird entweder die Vertraulichkeit oder die Integrität oder die Verfügbarkeit von Daten durch die Schwachstelle verletzt, ist die Auswirkung mit **mittel** eingestuft.

Ein Charakteristikum, das eine Schwäche von einer Schwachstelle unterscheidet, ist generell die Ausnutzbarkeit und die damit verbundene Entstehung von Schaden. Ist kein Ziel der Informationssicherheit durch die Ausnutzung der Schwachstelle verletzt, ist kein Schaden zu erwarten (**NA**, „not available“). In diesem Fall ist die Schwäche auch keine Schwachstelle und wird somit aus der Schwachstellen-Dokumentation komplett entfernt.

Die Kritikalität einer Schwachstelle ergibt sich aus der Kombination der Werte der Wahrscheinlichkeit ihrer Ausnutzung sowie dem dadurch entstehenden Schadenspotenzial, oder durch den Wert, den die *Temporäre Dringlichkeit* vorgibt, falls er festgelegt ist:

$$K_S = \begin{cases} T, & T \neq NA \\ criticality(W_S, A_S), & sonst \end{cases} \tag{5.3}$$

Die Funktion zur Berechnung der Kritikalität wird durch Tabelle 5.4 dargestellt.

Ist die Auswirkung **kritisch**, so ist auch – mit Ausnahme einer geringen Eintrittswahrscheinlichkeit – die Kritikalität einer Schwachstelle als kritisch anzusehen, da die Verletzung aller Ziele der Informationssicherheit als schwerwiegend einzustufen ist. Bei geringer Wahrscheinlichkeit einer Ausnutzung bleibt das Risiko hoch, da eine einzelne erfolgreiche Ausnutzung zu sehr großem Schaden führen kann.

Auswirkung A_S	*Eintrittswahrscheinlichkeit* W_S			
	gering	**mittel**	**hoch**	**kritisch**
mittel	gering	mittel	hoch	hoch
hoch	mittel	mittel	hoch	kritisch
kritisch	hoch	kritisch	kritisch	kritisch

Tabelle 5.4: Tabelle zur Berechnung der Kritikalität einer Schwachstelle, bzw. Definition der in Formel 5.3 verwendeten Funktion *criticality*

Die Kritikalität bei einer als **mittel** sowie **hoch** eingestuften Auswirkung wird anhand der Wahrscheinlichkeit einer Ausnutzung festgelegt: Falls der Wert der Eintrittswahrscheinlichkeit kleiner ist als der Wert der Auswirkung, so wird der direkt kleinere Wert nach dem Wert der Ausnutzung gewählt. Stimmt der Wert der Ausnutzungswahrscheinlichkeit mit dem Wert der Auswirkung überein, so wird dieser Wert für die Kritikalität der Schwachstelle festgelegt. Im letzten Fall, wenn die Eintrittswahrscheinlichkeit einer Ausnutzung einen höheren Wert als die Auswirkung hat, wird der zur Auswirkung nächst-höhere Wert gewählt.

Die Tabelle, welche die Kritikalität der Schwachstelle beschreibenden Funktion definiert, ist in dieser Form empfehlend und kann an praktische Erfahrungen in der Umsetzung angepasst werden. Die Berechnung geschieht an zentraler Stelle in der Schwachstellen-Dokumentation und muss entsprechend bei Anpassungen nur dort geändert werden.

5.6.1.2 Abbildung weiterer Bewertungssysteme auf das Konzeptsystem

Alleine im Rahmen der „Bulletins“ (`https://www.us-cert.gov/ncas/bulletins`) des *United States Computer Emergency Readiness Teams*, bzw. US-CERT, werden wöchentlich potenziell über einhundert Schwachstellenmeldungen veröffentlicht, wobei dies in der Regel nur ein Bruchteil der tatsächlich gefundenen Schwachstellen darstellt. Bei möglicherweise mehreren hunderten Schwachstellen, die automatisiert in die Schwachstellen-Dokumentation eingetragen werden, ist eine manuelle Bewertung jeder einzelnen Schwachstelle aufgrund des Ressourcenmangels in der Praxis nicht möglich.

Insofern ist es notwendig, die Bewertung der jeweiligen Fremdquelle zu verwenden und im Hinblick auf den Erhalt eines gleichartigen Bewertungsschemas und einer Vergleichbarkeit der Schwachstellen die fremdquellenspezifische Bewertung auf das im Hochschulnetz eingesetzte Bewertungssystem abzubilden.

Die Möglichkeiten zur Abbildung sind je nach Detailgrad der Beschreibung und in Betracht gezogenen Aspekte bei der Bewertung von Schwachstellen durch das fremdquellenspezifische Bewertungssystem mehr oder weniger vielfältig und genau. Beispielsweise könnte die Abbildung des CVSS BaseScore auf das im vorhergehenden Abschnitt beschriebene Bewertungssystem auf drei Arten erfolgen:

Die **erste Möglichkeit** besteht darin – da das Bewertungssystem im Konzept auf den Base Metrics des CVSS Version 3 basiert – die einzelnen Werte der Faktoren (d.h. Access

Vector, Attack Complexity, usw.) durch die Vektorstringdarstellung (vgl. Abschnitt 2.3.4) zu erfassen und als Bewertungsgrundlage zu nutzen.

Folgendes Kurzbeispiel in Tabelle 5.5 zeigt die Durchführung anhand der POODLE („Padding Oracle On Downgraded Legacy Encryption“) Schwachstelle (CVE-2014-3566 [MIT14c]), wobei genutzte Abkürzungen denen aus dem vorhergehenden Abschnitt (Tabelle 5.3) gleichen. Die Bewertungsgundlage sind die in der NVD unter der URL `https://nvd.nist.gov/cvss.cfm?version=2&name=CVE-2014-3566&vector=%28AV:N/AC:M/Au:N/C:P/I:N/A:N%29` dokumentierten Base Metrics des CVSS Version 2.

Abkürzung	**Bewertung**	**Erläuterung**
N	**ja**	POODLE ist über das Netz ausnutzbar
L	**gering**	Es ist keine Authentifizierung erforderlich. Außerdem existieren Exploits zur Ausnutzung der Schwachstelle
B	**ja**	Fälle der Ausnutzung können nicht aus dem CVSSv2 Base Metrics ausgelesen werden. Aufgrund dem in der Base Metrics als *medium* eingestuften Access Complexity wird angenommen, dass Fälle existieren.
C	**ja**	POODLE beeinträchtigt die Vertraulichkeit laut Base Metrics
I	**nein**	POODLE hat keine Auswirkungen auf die Integrität
A	**nein**	POODLE hat keine Auswirkungen auf die Verfügbarkeit
T	**NA**	Die Temporäre Dringlichkeit ist nicht gesetzt

Tabelle 5.5: Bewertung der POODLE Schwachstelle (CVE-2014-3566) durch das Konzept-Bewertungssystem anhand der Base Metrics

Aus den Werten ergibt sich laut der Definition zur Berechnung aus vorherigem Abschnitt eine Wahrscheinlichkeit der Ausnutzung von

$$W_{POODLE} = likelihood(ja, gering, ja) = kritisch \tag{5.4}$$

sowie eine Auswirkung der Ausnutzung von

$$A_{POODLE} = impact(ja, nein, nein) = mittel \tag{5.5}$$

Die Gesamtbewertung der Schwachstelle wird durch die Funktion $K_S = criticality(W_S, A_S)$ berechnet (vgl. Tabelle 5.4), da die *Temporäre Dringlichkeit* nicht gesetzt ist:

$$K_{POODLE} = criticality(kritisch, mittel) = hoch \tag{5.6}$$

So ist die Abbildung eher kompliziert, da beispielsweise Faktoren wie die *Bekanntheit der Ausnutzung* der Schwachstelle nicht unbedingt in anderen Bewertungssystemen vorkommen und anhand der anderen Faktoren geschätzt werden müssen. Auch kann es sein, dass lediglich der CVSS-BaseScore in einer Fremdquelle bereitgestellt wird, jedoch nicht der dazugehörige Vektorstring.

Der Vorteil hingegen ist eine hohe Genauigkeit und entsprechend eine gute Korrektheit der Bewertung.

Die **zweite Möglichkeit** zur Abbildung des CVSS BaseScores auf das Bewertungssystem des Konzeptes besteht in der direkten Übertragung der Kritikalitätseinschätzung aus Tabelle 5.1. Da der CVSS fünf Kritikalitätsstufen hat (*None*, *Low*, *Medium*, *High*, *Critical*), und *None* lediglich den BaseScore 0.0 abdeckt, ist die Eingliederung der Stufe `None` in `Low` sinnvoll, um das vierstufige System aus dem Konzept abzudecken. Auf diese Weise würde eine Abbildung vom BaseScore auf die Kritikalitätsstufen wie in Tabelle 5.6 aussehen.

CVSS BaseScore	**Bewertung**
0.0 – 3.9	gering
4.0 – 6.9	mittel
7.0 – 8.9	hoch
9.0 – 10.0	kritisch

Tabelle 5.6: Abbildung des CVSSv2 Scores auf die vier Stufen des Konzept-Bewertungssystems

Mit einem BaseScore von 4.3 wird der POODLE Bug daher im Unterschied zur ersteren Möglichkeit, als *mittel* eingestuft.

Die **dritte Möglichkeit** der Abbildung des CVSS auf das Konzept-Bewertungssystem zeichnet sich durch die Zuweisung anderer Werte des BaseScores für die jeweilige Kritikalitätsstufe aus, d.h. einer Anpassung der Werte in der Linken Spalte der Tabelle 5.6.

Prinzipiell ist jede der gezeigten Varianten möglich und kann beim Importieren von Schwachstelleninformationen eingesetzt werden. Die beiden letzteren Methoden zur Abbildung haben den Vorteil einer geringen Komplexität und lediglich die Abhängigkeit von einer Zahl, dem BaseScore, können jedoch zu ungenauen oder unangemessenen Einschätzungen führen.

Wichtig ist, dass, zum Erhalt der Konsistenz, es nicht mehrere Abbildungen von einem konkreten Bewertungssystem auf das im Schwachstellenmanagement genutzte Bewertungssystem geben darf sowie die Abbildung aller im vorhergehenden Abschnitt beschriebenen Faktoren zur Bestimmung der Kritikalität von Schwachstellen bestmöglich berücksichtigt.

Die jeweilige Definition einer anwendbaren Abbildung ist daher von der Wahl der Fremdquellen und den darin zur Verfügung gestellten Informationen zur Bewertung einer Schwachstelle abhängig. Die Verantwortlichkeit zur Entwicklung einer Abbildung liegt bei den Quellenverantwortlichen.

5.6.2 Kategorisierung von Schwachstellen

Im Gegensatz zur Bewertung von Schwachstellen trägt die Kategorisierung weder zur Ermittlung der Kritikalität einer Schwachstelle auf einem Asset, noch zur Priorisierung bei. Das Ziel der Kategorisierung ist im Besonderen die Kompensation der Ineffizienz der alleinigen Beschreibung von Schwachstellen durch Freitext und der daraus folgenden Unterstützung der Identifizierbarkeit von Schwachstellen. Konkret geht es bei der Kategorisierung von Schwachstellen um die Bestimmung und Dokumentation der betroffenen Softwareprodukte in den betroffenen Versionen sowie der Hersteller des Softwareprodukts.

Üblicherweise gibt es zwei verschiedene Ansätze, um diese Informationen einer Schwachstelle zu identifizieren. Zum einen sind diese Informationen in der Regel Teil der Schwachstellenbeschreibung, da es sich um grundlegende Charakteristika einer Schwachstelle handelt. So ist es insbesondere von der jeweiligen Quelle abhängig, ob diese Informationen lediglich in einer als Freitext vorhandenen Beschreibung oder maschinenlesbar vorliegen. Im Falle, dass betroffene Softwareprodukte automatisch abgefragt werden können, ist dieser Schritt nicht relevant, andernfalls liegt die Aufgabe der Bestimmung bei den im Hochschulnetz tätigen Kategorisierungsverantwortlichen.

Je nach Definition der jeweiligen Schwachstelle kann es sein, dass eine Kategorisierung nach Softwareprodukten nicht sinnvoll ist, insbesondere bei stark assetspezifischen Schwachstellen, die nicht allgemein auf alle Assets und Softwareprodukte angewendet werden können (vgl. Beispiel in Abschnitt 7.5). In diesem Fall wird die Kategorisierung anhand der betroffenen Assets vorgenommen, d.h. Kategorisierungsverantwortliche weisen Schwachstellen direkt bestimmten Assets zu, wodurch die Schwachstellen-Dokumentation die jeweiligen Detektionsverantwortlichen des Assets informiert. Diese können anschließend bei Bedarf manuell ein Schwachstellenticket für die Schwachstelle auf dem Asset eröffnen.

Ist die Kategorisierung des Schwachstelleneintrages vorgenommen worden, wird ihm der Status *detect* zugewiesen. Insofern wird angezeigt, dass die Schwachstelle identifiziert und klassifiziert ist, jedoch noch nicht detektierbar.

5.6.3 Klassifikation von Assets

Für die Klassifikation von Assets ist, genauso wie für die Dokumentation und Sicherstellung der Aktualität der das Asset beschreibenden Informationen in der Asset- und Benutzerverwaltung, der Asset-Verantwortliche zuständig. Bei Aktualisierung der Informationen über ein Asset muss dieser auch überprüfen, ob die vergebene Bewertung noch aktuell ist und der Beschreibung des Assets entspricht.

Die Bewertung von Assets ist in der Regel Teil des Risikomanagements und insofern nicht Fokus dieser Arbeit. Eine Auflistung einiger der Kriterien sowie beispielhafte Werte zur Einschätzung von Assets sind im folgenden Abschnitt beschrieben. Außerdem in Abschnitt 5.6.3.2 Möglichkeiten zur Kategorisierung von Assets.

5.6.3.1 Bewertung der Kritikalität von Assets

Eine Einschätzung der Kritikalität von Assets ist in der Klassifikation von ähnlich großer Bedeutung, wie die Bestimmung der Kritikalität von Schwachstellen.

Da die Bewertung von Assets, wie bereits im vorhergehenden Abschnitt beschrieben, in der Regel Teil des Risikomanagements ist und in unterschiedlichen Umgebungen durch unterschiedliche Charakteristika bestimmt wird, werden in diesem Abschnitt lediglich Aspekte aufgelistet, die die Assetbewertung beeinflussen können.

Üblicherweise bestimmt sich die Kritikalität eines Assets in erster Linie durch die Rolle, die es in Bezug auf die Erbringung geschäftskritischer Prozesse spielt, den aktuell angewendeten Maßnahmen, die zum Schutz eines Angriffs auf das jeweilige Asset umgesetzt sind sowie den darauf verarbeiteten und gespeicherten Daten und der Anzahl betroffener Nutzer. Ein weiteres Kriterium ist die Möglichkeit zur Nutzung des Assets als „Einfallstor“ zur Kompromittierung weiterer Assets (siehe Tabelle 5.7).

Charakteristikum	**mögliche Werte**
Abhängigkeit von kritischen IT-Dienstleistungen	nein, ja
Erreichbarkeit aus dem Internet	nein, ja
Speicherung bzw. Verarbeitung kritischer Daten	nein, ja
Anzahl betroffener Nutzer	0-20, 21-100, 101-1000, >1000
Das Asset als Einfallstor ins Netz	unwahrscheinlich, möglich

Tabelle 5.7: Charakteristika und beispielhafte Werte zur Bestimmung der Kritikalität eines Assets

Diese Aspekte können jedoch auch noch deutlich feingranularer gestaltet sein und beispielsweise nicht nur die Anzahl betroffener Nutzer bei einem Vorfall betrachten, sondern auch die Rollen der Nutzer, die auf das Asset angewiesen sind. Die Umsetzung ist dabei den einzelnen Instituten überlassen. Dabei besteht genauso die Möglichkeit, dass Institute in der Praxis grundsätzlich keine Risikoeinschätzung ihrer Assets vornehmen, wodurch im Endeffekt die Bewertung der Schwachstelle alleine als Kriterium der Behebungsnotwendigkeit dient.

Ähnlich wie bei Schwachstellen, ist auch die Granularität der Einschätzung von Assets prinzipiell nicht vorgegeben, sondern kann von einem zweistelligen Bewertungssystem in der Form *unkritisch* und *kritisch* bis hin zu einer Bewertung in Form von Zahlen mit mehreren Kommastellen Genauigkeit erfolgen.

Daher wird auch hier als beispielhaftes Bewertungsschema ein Mittelweg zwischen Einfachheit und Unterscheidbarkeit der Bewertungen gewählt und die Wahl der gleichen Werte wie für die Bewertung von Schwachstellen genommen:

niedrig, **mittel**, **hoch**, **kritisch**

Falls keine Bewertung vorliegt nimmt die Kritikalität den Wert **NA** ein, was einer nicht vorhandenen Einschätzung entspricht. Diese werden im Abschnitt 5.6.4 zur Berechnung der Notwendigkeit der Behebung einer Schwachstelle verwendet.

5.6.3.2 Kategorisierung von Assets

Ähnlich wie die Kategorisierung von Schwachstellen ist auch in der Teilaktivität der Kategorisierung von Assets ein wesentlicher Aspekt, bestimmte Attribute eines Assets herauszuarbeiten, um diese insbesondere in der Aktivität der Beseitigung und Abschwächung nutzen und Abläufe möglichst effizient gestalten zu können. In diesem Fall ist es das Ziel der Kategorisierung, auf den jeweiligen Assets aktuell eingesetzte Softwareprodukte zu ermitteln, mit dem Ziel der Unterstützung einer effektiven und effizienten Detektion von Schwachstellen auf Assets. Eine Beschreibung diesbezüglich findet sich in Abschnitt 5.7.2.

Die Verantwortung für diese Teilaktivität trägt der jeweilige Asset-Verantwortlicher des Assets.

Herausforderungen bei der Kategorisierung von Assets entstehen vor allem durch mögliche stark dynamische Veränderungen in der Softwareumgebung eines Assets oder auch allgemein, durch die Unvollständigkeit der Beschreibung der Softwareumgebung. Lösungen zur Überwindung der Herausforderungen sind einerseits durch ein akkurates Änderungs- und Konfigurationsmanagement als auch direkt mit Hilfe der Detektionsagenten auf den Assets.

Erstere Möglichkeit, die Kategorisierung in Verbindung mit entsprechend ausgelegtem **Änderungs- und Konfigurationsmanagement** (*Change Management* bzw. *Configuration Management*), ist eine zuverlässige Möglichkeit, um den Überblick über die auf den Assets eingesetzte Software zu erhalten. Die im Konfigurationsmanagement eingesetzte *Configuration Management Database* (CMDB), eine Komponente, die eine logische Darstellung aller *Configuration Items* (CI) und deren Relation zueinander hält, liefert in der Regel bereits genau die hier gewünschten Informationen, da – je nach Granulatität und Definition eines CI in der betrachteten Umgebung – Software und Hardware ein wesentlicher Bestandteil der Konfiguration einer Umgebung sind.

Um manuellen Aufwand durch Personen im Schwachstellenmanagement, hier insbesondere der dafür zuständigen Asset-Verantwortlichen zu vermeiden, wird die Asset- und Benutzerverwaltung automatisiert bei Änderungen in der CMDB über die vom Steuerungssystem zur Verfügung gestellte API aktualisiert. Assets, die in der Asset- und Benutzerverwaltung, jedoch nicht in der CMDB vorhanden sind, werden entweder in die CMDB eingepflegt, manuell in der Asset- und Benutzerverwaltung des Schwachstellenmanagements gepflegt und/ oder mit Hilfe einer zweiten Möglichkeit aktuell gehalten: mit Hilfe zusätzlicher Funktionalität in Detektionsagenten.

Die Kategorisierung durch **Detektionsagenten** ist vor allem unterstützend und mehr Teil eines Änderungsmanagements – kann dieses also üblicherweise nicht komplett ersetzen. Das dadurch verfolgte Ziel ist es, die Softwarekonfiguration des jeweiligen Assets, auf dem ein Detektionsagent läuft, bei Bedarf möglichst aktuell und vollständig auszulesen. Dabei kann man zwei verschiedene Arten von Software unterscheiden: Zum einen Software, die auf dem Asset installiert ist und zum anderen Software, die auf dem Asset eingesetzt und ausgeführt wird. Die Unterscheidung kann die Komplexität weiter verringern, da auf dem Asset eingesetzte (laufende) Software in der Regel nur eine kleine Teilmenge der installieren Software ist und als Status darüber angesehen werden kann, welche Schwachstellen auf einem Asset zu einem bestimmten Zeitpunkt ausnutzbar sind, da Software- und Konfigurationsschwachstellen nur tatsächlich eingesetzte Produkte betreffen.

Hinweise auf allgemein installierte Software auf einem Asset liefern beispielsweise darauf eingesetzte Paketmanager (in auf Debian-basierenden Distributionen beispielsweise *dpkg*, bzw. Auflisten von Software mit *dpkg -l*) oder auch bestimmte Verzeichnisse, wie unter Microsoft Windows beispielsweise `C:\Program Files` bzw. `C:\Program Files (x86)` und bestimmte Schlüssel in der Windows-Registrierungsdatenbank (*Registry*), zum Beispiel unter `Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall`.

Hinweise auf Software, die auf dem Asset eingesetzt bzw. ausgeführt wird, liefern die darauf laufenden Prozesse – unter UNIX-Derivaten sowie unix-ähnlichen Betriebssystem beispielsweise mit Hilfe des Kommandos *ps*, unter Windows durch das Programm *tasklist*.

Da die Kombination beider Möglichkeiten die besten Ergebnisse verspricht bzw. das aktuellste und vollständigste Bild der Konfiguration von Assets bietet, ist diese im Konzept vorgesehen. Bei sich widersprechenden Angaben, ist davon auszugehen, dass Informationen, die über Detektionsagenten erfasst werden, aktueller sind, als Daten in der Asset- und Benutzerverwaltung bzw. der CMDB und werden deshalb entsprechend bevorzugt.

5.6.4 Ermittlung der Behebungsnotwendigkeit und Priorität

Der Wert der Behebungsnotwendigkeit ergibt sich einzeln jeweils für eine Kombination der Bewertung einer Schwachstelle mit der Kritikalitätseinschätzung eines Assets. Das Ziel, das damit verfolgt wird, ist insbesondere ein einheitliches Schema anzubieten, anhand dessen die Behebungsnotwendigkeit gleichbleibend transparent und für identische Charakteristika der Schwachstellen und Assets immer wiederholbar ist, um hochschulnetzweit eine konsistente Bearbeitung zu ermöglichen.

In Tabelle 5.8 ist eine beispielhafte Beschreibung einer Funktion zur Ermittlung der Behebungsnotwendigkeit gezeigt.

		Schwachstellenbewertung			
		gering	**mittel**	**hoch**	**kritisch**
Asset-Kritikalität	**NA**	gering	mittel	hoch	kritisch
	gering	gering	mittel	mittel	mittel
	mittel	gering	mittel	hoch	hoch
	hoch	mittel	hoch	hoch	kritisch
	kritisch	hoch	kritisch	kritisch	kritisch

Tabelle 5.8: Tabelle zur Berechnung der Behebungsnotwendigkeit einer Schwachstelle auf einem Asset

Der Aufbau ist dabei ähnlich wie der zur Ermittlung der Kritikalität einer Schwachstelle, welche in Tabelle 5.4 gezeigt wurde. Die Eingabe der Funktion umfasst die Bewertung der jeweiligen Schwachstelle, sowie die Kritikalität des Assets, auf dem die Schwachstelle liegt.

Die beispielhafte Berechnung der Behebungsnotwendigkeit erfolgt derart, dass der Wert der Asset-Kritikalität als Ausgangspunkt dient. Ist die Bewertung der Schwachstelle geringer, so ergibt sich der zur Asset-Kritikalität nächst-geringere Wert als Behebungsnotwendigkeit. Bei gleichem Wert der Schwachstellen-Bewertung sowie Asset-Kritikalität ergibt sich dieser Wert als Behebungsnotwendigkeit und im letzten Fall, dass die Bewertung der Schwachstelle größer ist als die der Kritikalität des Assets, ist der nächst-höhere Wert zur Asset-Kritikalität die ermittelte Behebungsnotwendigkeit.

Eine Ausnahme bildet die Ermittlung der Behebungsnotwendigkeit bei Assets mit Kritikalität *kritisch*: Aufgrund der hohen Geschäftskritikalität, die ein solches Asset in der Regel einnimmt, werden zudem Schwachstellen mit einer Bewertung von *mittel* sowie *hoch* genauso mit einer Behebungsnotwendigkeit *kritisch* klassifiziert.
Eine weitere Ausnahme bildet der Fall, dass die Kritikalität eines Assets nicht angegeben bzw. nicht bekannt ist (*NA*). In diesem Fall wird die Wert der Kritikalität der Schwachstelle als Wert der Behebungsnotwendigkeit benutzt.

> Zur Ermittlung der Behebungsnotwendigkeit gehört zudem die Bestimmung des **Schwachstellenakzeptanzkriteriums**. Dieses kann die gleichen Werte wie die Behebungsnotwendigkeit annehmen (*gering*, *mittel*, *hoch* und *kritisch*) und beschreibt die höchste Bewertung der Behebungsnotwendigkeit von Schwachstellentickets, die akzeptiert, d.h. nicht bearbeitet werden. Im Falle, dass alle Schwachstellen auf allen Assets behandelt werden, hat das Schwachstellenakzeptanzkriterium den Wert *NA* (engl. „not available“).

Ist das Schwachstellenakzeptanzkriterium beispielsweise auf *mittel* gesetzt, so werden ausschließlich alle Schwachstellen detektiert, deren jeweils resultierendes Schwachstellenticket eine Behebungsnotwendigkeit von *hoch* oder *kritisch* aufweist. Die Berechnung der Behebungsnotwendigkeit relevanter Schwachstellen auf einem Asset erfolgt jedoch schon vor der Detektion, sodass nach einer Schwachstellen auf Assets die ein Schwachstellenticket mit einer Behebungsnotwendigkeit ergeben würden, die geringer als das Schwachstellenakzeptanzkriterium ist, gar nicht detektiert werden müssen und somit die Erstellung unnötiger Schwachstellentickets verhindert wird.

5.7 Beseitigung und Abschwächung von Schwachstellen

Die wichtigsten Eingaben in der Aktivität der Beseitigung und Abschwächung von Schwachstellen kommen aus der Identifikation – die Information über Schwachstellen sowie Assets – als auch aus der Klassifikation – die Kritikalitätseinschätzung der jeweiligen Schwachstellen sowie der Einschätzung zur Notwendigkeit einer Beseitigung bzw. Abschwächung der Schwachstelle auf dem jeweiligen Asset. Eine Illustration der Teilaktivitäten sowie damit in Verbindung stehenden Ein- und Ausgaben ist in Abbildung 5.7 dargestellt.

Eine wesentliche Grundvoraussetzung für eine Behebung einer Schwachstelle ist die Erkennung bzw. Detektion einer relevanten Schwachstelle auf einem Asset. Die Ermittlung der Relevanz einer Schwachstelle geschieht vorab in der Aktivität der Klassifikation. Aus diesem Grund wird die Erstellung von Detektionsverfahren und deren Ausführung in der Aktivität der Beseitigung und Abschwächung eingeordnet.

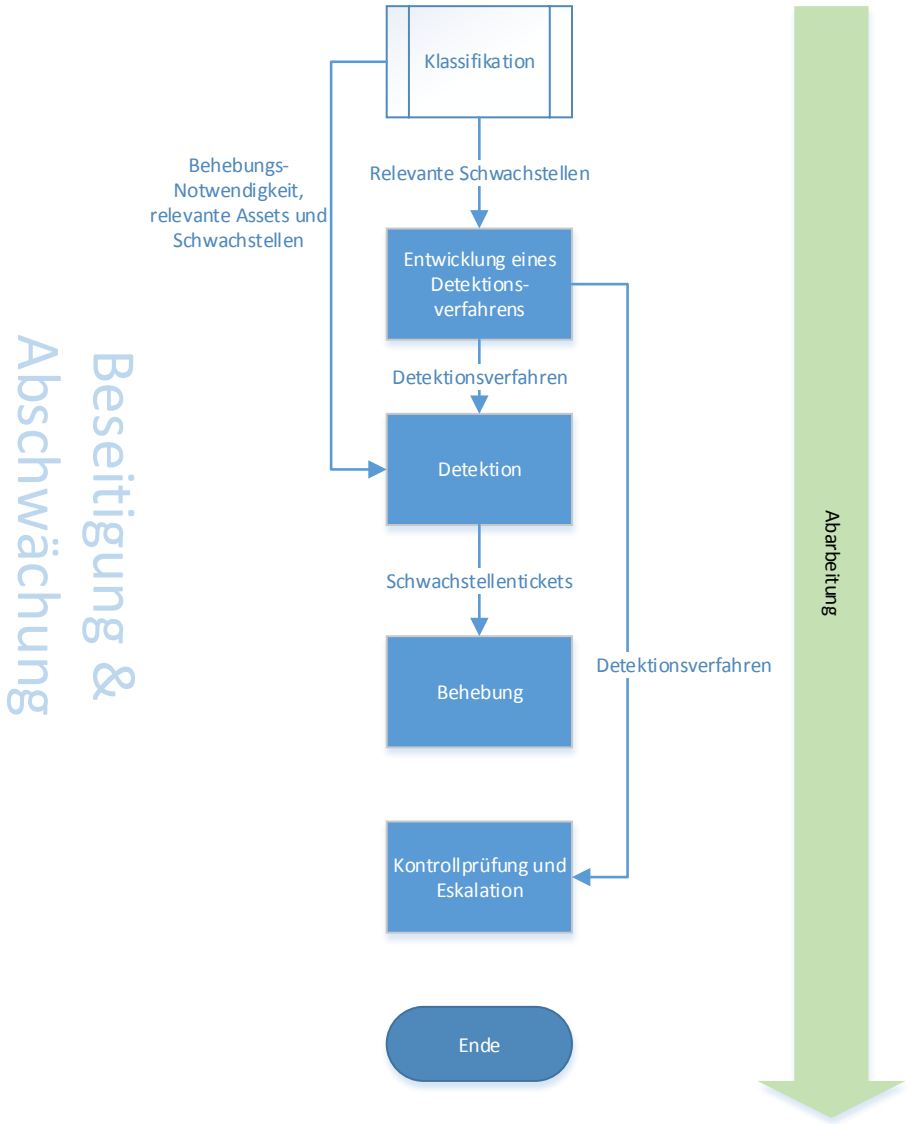


Abbildung 5.7: Teilaktivitäten in der Beseitigung und Abschwächung

5.7.1 Erstellung von Detektionsverfahren

Die Entwicklung von Verfahren zur Detektion von Schwachstellen auf Assets ist eine der zwei Hauptaufgaben des Detektionsverantwortlichen. Im Gegensatz zur eigentlichen Durchführung einer Detektion ist dieser Schritt jedoch zunächst nicht abhängig von einem bestimmten ihm zugewiesenen Asset, sondern vielmehr von allen im Anwendungsbereich definierten Plattformen (d.h. Betriebssystemen).

Da es wegen der hohen Unterschiedlichkeit der Ausprägung von Schwachstellen nicht möglich ist, die Vorgehensweise zur Erstellung von Detektionsverfahren einheitlich zu definieren, ist eine Lösung insbesondere abhängig von den Kenntnissen des jeweiligen Detektionsverantwortlichen. In diesem Konzept für ein Schwachstellenmanagement in Hochschulnetzen wird jedoch der Rahmen eines Detektionsverfahrens vorgegeben, welcher in Abschnitt

5.3.1.1 festgelegt ist, sowie die im Folgenden beschriebenen Richtlinien zur Erstellung eines Detektionsverfahrens:

- Da die Detektion von Schwachstellen über das Netz allgemein einfacher zu kontrollieren und zentral steuerbar ist, ist die Entwicklung von zuverlässigen Verfahren zur Detektion über das Netz vorzuziehen. Erweist sich das Verfahren aus Netzsicht als unzuverlässig und gleichzeitig ein Verfahren aus Systemsicht als deutlich erfolgreicher, so ist letzteres bevorzugt zu entwickeln. Eine Kombination der Anwendung jeweils eines (oder mehrerer) Verfahrens beider Sichten ist genauso möglich und als am erfolgversprechendsten zu betrachten.

- Es ist sinnvoll und teilweise notwendig, auf bereits existierende Möglichkeiten zur Detektion einer Schwachstelle zurückzugreifen. Beispielsweise kann die Funktionalität von Schwachstellenscannern wie *OpenVAS* oder *nmap* genutzt und ebenfalls als Detektionsverfahren verwendet werden. Dazu wird der eigentliche Netzwerkscanner in die jeweilige Komponente (entweder Detektionssystem für Verfahren aus Netzsicht oder Detektionsagenten aus Systemsicht) integriert. Die Implementationen der Detektionsverfahren nutzen den Funktionsumfang des jeweiligen Softwaretools und geben wie in Abschnitt 5.3.1.1 festgelegt, entweder *true*, *false* oder eine konkret definierte Ausgabe (beispielsweise eine Liste offener Ports) zurück. Ein Beispiel zur Nutzung von *nmap* zur Detektion des *Heartbleed Bugs* ist in Abschnitt 7.6 gezeigt.

- Implementierungen von Verfahren zur Detektion von Schwachstellen auf Assets aus Systemsicht werden in Programmier- bzw. Skriptsprachen entwickelt, die durch entsprechende Compiler und Interpreter plattformübergreifend angewendet werden können, da diese auf den Assets direkt zur Ausführung kommen und einen entsprechend weiten Umfang an Betriebssystemen abdecken müssen. Bekannte Beispiele hierfür sind *Perl*, *Java*, *C* und viele weitere.

  Je nach Umsetzung ist es notwendig, verwendbare Skript- und Programmiersprachen hochschulnetzweit vorzugeben, damit auch die Softwareumgebung von Detektionssystemen überschaubar bleibt und weniger aufwändig ist. Derartige Entscheidungen müssen durch den Chief Vulnerability Manager vorgenommen werden.

- Vorteile, die sich aus der Nutzung von Scriptsprachen und Interpretern ergeben, sind eine einfachere Wartbarkeit sowie Verteilung der Programme auf die Assets, auf denen Detektionsagenten eingesetzt werden, da der Schritt des Kompilierens wegfällt bzw. bei Ausführung (durch einen Just-In-Time-Compiler) auf den Assets geschieht. Demgegenüber müssen traditionelle Programme vorab für verschiedene Plattformen kompiliert und daraufhin an die Agenten verteilt werden. Der größte Vorteil der Verwendung von Programmiersprachen ist in der Regel eine höhere Systemnähe der Programme. Falls diese nicht genutzt werden kann oder nicht zur Detektion der jeweiligen Schwachstelle benötigt wird, ist die Nutzung von Skriptsprachen und Interpretern traditionellen Programmen vorzuziehen.

- Bei der Erstellung eines Detektionsverfahrens sollen auch Fremdquellen herangezogen werden. Eine öffentliche Quelle an Skripten zur Detektion von einigen Schwachstellen findet sich beispielsweise unter `https://nmap.org/nsedoc/scripts/`.

- Bei der Detektion müssen alle der in Abschnitt 5.3.1.1 definierten Attribute eines Detektionsverfahrens beschrieben werden.

Entwickelte Detektionsverfahren müssen anschließend auf ihre Wirksamkeit sowie auf unerwünschte Effekte (beispielsweise ein Ausfall des Zielassets) bezüglich einer Beeinträchtigung der drei Ziele der Informationssicherheit – ebenfalls durch den dafür zuständigen Detektionsverantwortlichen – getestet werden. Dazu gibt es eine Testumgebung bestehend aus einem Detektionssystem und Maschinen der verschiedenen Plattformen mit jeweilig einsetzbarem Detektionsagent für Testzwecke. Bei den Tests muss auch die Verzeichnisstruktur der Detektionssysteme und Agenten berücksichtigt werden. Verweise auf andere Dateien und Programme müssen auch auf den eigentlichen Geräten immer noch korrekt sein.

Sobald das entwickelte Verfahren als zweckdienlich für die Detektion einer Schwachstelle auf Assets eingestuft wird und keine unerwünschten Nebeneffekte festgestellt wurden, erfolgt die Dokumentation und Eintragung in die Schwachstellen-Dokumentation. Des Weiteren werden die entsprechenden Komponenten, Detektionssysteme sowie Detektionsagenten, je nach Art der Implementierung um das Detektionsverfahren erweitert.

Implementierungen aus Systemsicht werden den Detektionsagenten zur Verfügung gestellt, Implementierungen aus Netzsicht den Detektionssystemen.

Bevor die Implementierung des Detektionsverfahrens jedoch in den Funktionsumfang der verschiedenen Detektionssysteme bzw. Detektionsagenten aufgenommen wird, muss dieses durch einen weiteren, zufällig ausgewählten Detektionsverantwortlichen begutachtet und freigegeben werden, um böswillige oder versehentliche Nutzung des Schwachstellenmanagements zur Verbreitung von Schadsoftware zu verhindern. Dies kann je nach Komplexität des Programms lediglich durch Einsicht des Quelltextes geschehen oder in Kombination mit weiteren praktischen Testdurchläufen. Der Nutzer, der die Implementierung freigegeben hat, wird ebenfalls dokumentiert. Nach der Begutachtung mit positivem Ergebnis sowie allen durchgeführten Teilaktivitäten (Verifikation, Bewertung, Kategorisierung, Ermöglichung der Detektion), wird der jeweiligen Schwachstelle der Status *complete* zugewiesen. Ab diesem Zeitpunkt kann das Detektionsverfahren auf Detektionssysteme, oder auf den für die freigegebenen Plattformen jeweiligen Detektionsagenten eingesetzt werden.

Die Bereitstellung eines Detektionsverfahrens an Detektionssysteme bzw. Detektionsagenten ist abhängig von deren Implementierung. Eine mögliche Verfahrensweise ist im nächsten Kapitel beschrieben.

5.7.2 Detektion von Schwachstellen auf Assets

Um Schwachstellen auf Assets detektieren zu können, sind für diese Teilaktivität einige Eingaben erforderlich, die Produkte der vorhergehenenden Aktivitäten der Identifikation, Klassifikation als auch der Aktivität der Beseitigung und Abschwächung von Schwachstellen sind. Zunächst müssen Schwachstellen und Assets bekannt sein, auf denen die Detektion durchzuführen ist, wobei in der Regel nicht alle Schwachstellen für ein bestimmtes Asset relevant sind und irrelevante Schwachstellen zur Schonung von Ressourcen von der Detektion ausgeschlossen werden (siehe folgender Abschnitt). Schließlich können lediglich Schwachstellen

detektiert werden, zu denen mindestens ein Detektionsverfahren definiert ist. Die Kombination mehrerer Verfahren ist genauso möglich und kann die Verlässlichkeit der Detektion erhöhen.

Die Verantwortung für die Detektion von Schwachstellen auf Assets liegt ausschließlich bei den für ein Asset zuständigen, jeweiligen Detektionsverantwortlichen. Die Sicherstellung der Einhaltung der Zuständigkeiten und Einschränkung des Detektionsverantwortlichen in dem Umfang der für ihn notwendigen Informationen, d.h. relevanter Assets, wird mit Hilfe entsprechender in der Asset- und Benutzerverwaltung verwalteter Rechte der Form „*Detektionsverantwortlicher X darf Schwachstellen auf Asset Y detektieren*“, sowie geeigneter mandantenfähiger Nutzerschnittstellen gewährleistet. Diese Einschränkung betrifft die Einholung von Ergebnissen einer Detektion über dafür vorgesehene Detektionssysteme sowie Detektionsagenten.

Eine weitere Möglichkeit bietet eine weiterführende Einschränkung der Erlaubnis auf bestimmte Schwachstellen, das heißt, die explizite Bestimmung von Privilegien der Form „*Detektionsverantwortlicher X darf Schwachstelle Z auf Asset Y detektieren*“. Der Vorteil, der sich aus letzterem Ansatz ergibt, ist eine feingranularere Unterscheidung von Zuständigkeiten, so dass ein Detektionsverantwortlicher beispielsweise auf einem Asset mit Linux Betriebssystem nur die Schwachstellen detektieren darf, die den Linux-Kernel betreffen. Analog dazu würde auch die Möglichkeit bestehen, Rechteeinschränkungen der Form „*Detektionsverantwortlicher X darf Schwachstelle Z auf Asset Y nicht detektieren*“ vorzunehmen.

Ähnliches Vorgehen kann auch auf Basis von Software und vor allem Betriebssystemen erfolgen. Ein genereller Ansatz ist eine Erlaubnis in Form „*Detektionsverantwortlicher X darf Schwachstelle in Softwareprodukt Y auf Asset Z detektieren*“, wobei hier auch Betriebssysteme allgemein als Softwareprodukte angesehen werden. Dieser Ansatz ist insofern sinnvoll, da in Hochschulrechenzentren Verantwortlichkeiten bestimmter organisatorischer Gruppen und Abteilungen beispielsweise nach Art von Betriebssystemen getrennt sind.

In der Regel entsteht durch derartige Ansätze insbesondere bei einer hohen Anzahl registrierter Schwachstelleneinträge ein erheblicher organisatorischer Mehraufwand. Sie sind somit nicht im Konzept vorgesehen, sondern ausschließlich der erste Ansatz, die Zuständigkeit bestimmter Detektionsverantwortlicher auf bestimmten, ihnen zugewiesenen Assets.

5.7.2.1 Auswahl der zu detektierenden Schwachstellen

In der Schwachstellen-Dokumentation werden alle Schwachstellen, die für das gesamte Hochschulnetz relevant sind, dokumentiert. Die Bandbreite der Art der Schwachstellen umfasst in der Regel mehrere Gerätetypen, Betriebssysteme als auch Server- und Anwendersoftware. Da diese Bandbreite üblicherweise nicht auf jedes Asset zutrifft, sondern ein Asset, d.h. Computersystem, vielmehr in seinem aktuell laufenden Zustand nur von einer Teilmenge dieser Schwachstellen betroffen ist, ist es sinnvoll, die Detektion auf diese Teilmenge zu beschränken.

Beispielsweise wird ein Webserver auf Linux-Basis betrachtet, dessen installierte Softwarepakete inklusive Softwareversion bekannt und aktuell dokumentiert sind. Ein Detektionsversuch von Schwachstellen im Windows-Kernel wäre daher erwartungsgemäß mit einem negativen Ergebnis und gleichzeitig unnötigem, vermeidbarem Aufwand verbunden. Im Idealfall

werden entsprechend ausschließlich Schwachstellen in den auf dem Webserver installierten Softwarepaketen bzw. einer Fehlkonfiguration der Software detektiert.

Notwendig wird diese Reduzierung vor allem durch die hohe Anzahl der Geräte im Hochschulnetz. Bei potenziell mehreren hundert bzw. auch tausend Assets, die in dieser Teilaktivität gleichzeitig auf Schwachstellen überprüft werden, kann die Laufzeit mit jeder weiteren zu detektierenden Schwachstelle bzw. mit jedem weiteren Asset erheblich steigen und weit fernab einer durch das Schwachstellenmanagement angestrebten Anwendbarkeit sein.

Die Auswahl der zu detektierenden Schwachstellen wird mit Hilfe der Informationen der in der Aktivität der Klassifikation durchgeführten Kategorisierung durchgeführt. Durch die Informationen kann zum einen eine Schwachstelle einem bestimmten Softwareprodukt zugewiesen werden und andererseits mit auf Assets installierter Software abgeglichen werden. Die Detektion muss daher lediglich für die Schnittmenge aller detektierbaren Schwachstellen in bestimmten Softwareprodukten mit installierter Software auf einem Asset ausgeführt werden.

Eine weitere Einschränkung erfolgt in Bezug auf Schwachstellen mit einem bestimmten berechneten Wert der Behebungsnotwendigkeit auf einem Asset. Je nach Umfeld und zur Verfügung stehenden Ressourcen kann der Chief Vulnerability Manager die Auswahl relevanter Schwachstellen durch Festlegen des Schwachstellenakzeptanzkriteriums beeinflussen. Schwachstellen auf Assets, deren Behebungsnotwendigkeit (errechnet aus Kritikalität der Schwachstelle in Kombination mit der Kritikalität des jeweiligen Assets) kleiner oder gleich groß ist, wie der Wert des Schwachstellenakzeptanzkriteriums, werden nicht behoben und insofern auch nicht detektiert.

Das Filtern relevanter Schwachstellen zur Detektion auf einem Asset findet an zentraler Stelle im Steuerungssystem statt.

5.7.2.2 Verfahrensweise zur Durchführung der Detektion

Die Detektion von Schwachstellen auf Assets wird entweder direkt durch einen Detektionsverantwortlichen oder in vordefinierten Zeitintervallen bzw. an festgelegten Zeitpunkten ausgeführt.

Mögliche Zeitintervalle sind wöchentlich genauso wie monatlich und sind insbesondere abhängig von der Umsetzungsmöglichkeit im jeweiligen Umfeld. Die Ausführung einer automatischen Detektion ist auch, je nach Anzahl der Assets, gestaffelt auszuführen, um zum einen technische Komponenten (Steuerungssystem und Detektionssysteme) als auch das Personal nicht zu überfordern.

Des Weiteren ist es sinnvoll, die Detektion von Schwachstellen bei bestimmten Ereignissen, beispielsweise einer Änderung der Konfiguration eines Assets und insbesondere bei der Installation neuer Software und Updates, durchzuführen, um über den Zustand der darin befindlichen Schwachstellen informiert zu sein und diese zu beheben.

Zugriff auf den Aufruf zur Ausführung der Detektion auf den ihm zugeordneten Assets hat der jeweilige Detektionsverantwortliche über ein entsprechend mandantenfähiges und vom Steuerungssystem zur Verfügung gestelltes Webportal als Benutzeroberfläche.

Herausforderungen in der Detektion von Schwachstellen in Netzen ergeben sich allgemein insbesondere aus der Netzstruktur bzw. technischen Umsetzung der Einteilung der jeweiligen

institutseigenen Netzbereiche. Zu beachten sind neben öffentlichen auch private Netzbereiche, das heißt, Adressbereiche innerhalb der durch CIDR Notation ausgedrückten Netze `10/8`, `172.16/12` sowie `192.168/16`. [Yak96] Letztere bringen das Problem mit sich, dass Geräte aus einem privaten Netzbereich nur durch Detektionssysteme adressierbar sind, die im selben privaten Netz liegen und entsprechend nur durch solche über das Netz detektierbar sind.

Bei öffentlich adressierten Geräten tritt dieses Problem in der Regel nicht auf bzw. kann durch eine angepasste Konfiguration (beispielsweise in Form von Regeln für Paketfilter im Netz) umgangen werden. Die Konfigurierbarkeit ist daher ein Grund zur ausdrücklichen Bestimmung von Detektionssystemen. So kann ein Beitrag zur Sicherheit geleistet werden, da die Zugriffskontrolle von Assets speziell auf diese Maschinen – Detektionssysteme – angepasst werden kann, um die Detektion über das Netz zu ermöglichen. Gleichzeitig können Schwachstellenscans ausgehend von anderen Maschinen geblockt werden.

Eine andere, zunächst von der Art der Detektion unabhängige Herausforderung betrifft den möglichen Einfluss auf die Verfügbarkeit oder lediglich die Befürchtung der Beeinträchtigung, beispielsweise durch die Asset-Verantwortlichen. Die Folge ist eine teilweise absichtlich herbeigeführte Zugriffseinschränkung, im Besonderen für Detektionssysteme, wodurch Schwachstellen auf dem betroffenen Asset möglicherweise nicht mehr erkannt und infolgedessen nicht beseitigt werden können. Daher muss bei Anwendung von Detektionsverfahren sichergestellt werden, dass diese selbst keine negativen Auswirkungen auf Zielassets haben und ein damit verbundenes Misstrauen von Seiten der Asset- und Dienstbetreiber in die Detektion entsteht. Eine Lösung stellt beispielsweise auch die Miteinbeziehung dieser Personen durch Vergabe entsprechender Rollen dar.

Wie bereits im Verlauf der Arbeit beschrieben, gibt es zwei verschiedene Ansatzpunkte bei der Detektion von Schwachstellen, welche – auch in Bezug auf die genannten Herausforderungen – vorteilhaft eingesetzt und kombiniert werden können.

- Die **Detektion aus Netzsicht** spiegelt prinzipiell die Sicht eines Angreifers auf Schwachstellen wider, der keinen lokalen bzw. physischen Zugriff auf das Asset hat. Die Vorgehensweise bei der Detektion über das Netz entspricht üblicherweise einer Provokation (durch aktives Anfragen) oder Beobachtung (z.B. durch Analyse des Netzverkehrs) eines bestimmten vom Asset ausgeführten Verhaltens, das sich je nach Schwachstelle unterschiedlich äußern kann, beispielsweise durch Senden einer bestimmten Antwort auf eine Anfrage, durch Zeigen bestimmter Eigenschaften (z.B. eine unsichere Verschlüsselung), oder einem Ausfall des Assets („Denial of Service“). Ein Hervorrufen eines Denial of Service muss aufgrund möglicher Beeinträchtigung von Diensten zur Detektion vermieden werden oder nur im Bewusstsein der Risiken und einer entsprechenden Verhältnismäßigkeit zu den Folgen einer Nicht-Detektierbarkeit der Schwachstelle angewendet werden. Oft wird in diesem Verfahren die Ausnutzung einer Schwachstelle zur Detektion daher direkt simuliert.

  Neben der Beobachtung eines Verhaltens bei Ausnutzung einer Schwachstelle werden auch weitere, generell über das Netz auslesbare Informationen zur Detektion verwendet. So ist es je nach Implementierung einer Software beispielsweise möglich, die Version auszulesen und für die Detektion von Schwachstellen zu verwenden.

  Die Detektion über das Netz erfolgt über dafür vorgesehene und für diese Aufgabe konfigurierte Computersysteme, den sogenannten Detektionssystemen. Zur Verteilung

der Detektion und einer Ermöglichung der parallelen Detektion auf mehreren Systemen, im Hinblick auf eine Reduzierung der Laufzeit, muss eine ausreichende Anzahl an Detektionssystemen im Hochschulnetz zur Verfügung gestellt werden, welche alle – bezogen auf den Umfang der detektierbaren Schwachstellen und Systemkonfiguration – eine gleichartige Konfiguration aufweisen und einzeln mit der Ausführung einer Detektion von Schwachstellen auf bestimmten Assets beauftragt werden können. Die Aufteilung der Detektion auf mehreren Assets innerhalb eines Detektionsdurchlaufs wird durch das ausführende Steuerungssystem vorgenommen und kann durch unterschiedliche Vorgehensweisen realisiert werden. Beispielsweise eine auf Basis der Anzahl der Assets gleichverteilte vorgenommene Zuweisung, sodass jedes Detektionssystem eine eigene Warteschlange (engl. „queue“) hat, oder auch eine zentrale Warteschlange im Steuerungssystem mit direkter Zuweisung an freie Detektionssysteme. Aufgrund der einfacheren Aufteilung ist im Konzept eine zentrale Warteschlange und Zuweisung bei Bedarf vorgesehen.

Das oben erwähnte Problem der Detektion in privaten Netzen kann zum einen derart gelöst werden, dass in den jeweiligen Instituten bzw. privaten Netzen ein Detektionssystem direkt integriert wird, durch das die Detektion von Schwachstellen auf Assets innerhalb desselben Netzes stattfindet. Zur Vermeidung potenzieller Kosten werden diese beispielsweise auch durch Virtualisierung und die Bereitstellung eines dafür ausgelegten virtuellen Datenträgers realisiert. Zur **Wartbarkeit** von zentraler Stelle – dem Rechenzentrum – sowie der Möglichkeit zur Ausführung einer Detektion muss jedoch der Zugriff auf die Detektionssysteme gewährleistet werden – beispielsweise durch entsprechende Port-Forwarding-Regeln.

Die Dokumentation von Detektionssystemen muss unbedingt vorgenommen werden, damit das Steuerungssystem auch für die Detektion in privaten Netzen die Zuweisung korrekt an das entsprechende Detektionssystem weiterleiten kann und beispielsweise keines zur Detektion in öffentlichen Netzen auswählt.

Eine Evaluation der Detektion über das Netz in großen Netzen, insbesondere mit Fokus auf die Detektion der POODLE Schwachstelle [MIT14c] wurde in [Mäu15] vorgenommen. Ausgewertet wurde unter anderem der Netzwerkscanner *nmap*, zu dem beispielsweise im Münchner Wissenschaftsnetz eine Auswertung durch das Tool *Dr. Portscan* [Fel13] vorgenommen wird.

- Die **Detektion aus Systemsicht** bietet in der Regel deutlich umfassendere Möglichkeiten, macht jedoch die Ausführung einer für diesen Zweck entwickelten Anwendung (den Detektionsagenten) mit – in der Regel privilegierten – Nutzerrechten auf dem Asset selber notwendig. Wie bereits bei der Kategorisierung von Assets (Abschnitt 5.6.3.2) erwähnt, sieht das Konzept in Detektionsagenten eine Funktion zum Auslesen installierter Software vor, um insbesondere auszuführende Verfahren für die Detektion über das Netz einzuschränken und unnötigen Aufwand zu minimieren (vgl. vorherigen Abschnitt). Diese Informationen sind auch direkt zur Detektion von Schwachstellen vorgesehen, indem nicht nur Softwareprodukte abgeglichen werden, sondern das konkrete, auf dem Asset installierte Softwarepaket mit dem von einer Schwachstelle betroffenem Softwareprodukt. Weitere Mechanismen zur Detektion können je nach Schwachstelle verwendet werden, insofern sie die in Abschnitt 5.7.1 beschriebenen Kriterien erfüllen und insbesondere keine negativen Auswirkungen mit deren Ausführung

verbunden sind. Beispiele für negative Auswirkungen sind ein ungewollter Systemabsturz genauso wie eine Beeinträchtigung der drei Schutzziele der Informationssicherheit durch den Detektionsagenten selbst.

Der Ablauf nach einem durch einen Detektionsverantwortlichen ausgeführten Detektionsdurchlauf ist in Abbildung 5.8 illustriert und auch für die automatische Ausführung, abgesehen vom Auslöser – in diesem Fall der Detektionsverantwortliche, gleich.

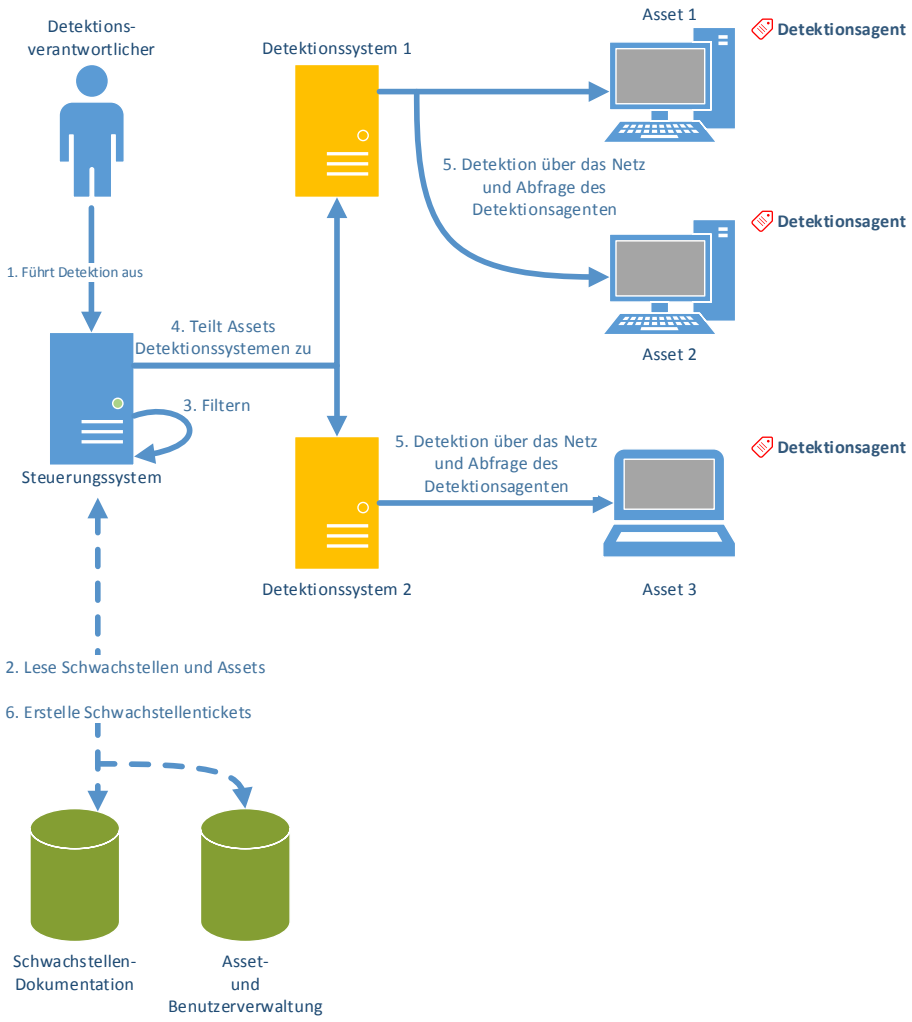


Abbildung 5.8: Übliche Durchführung eines Detektionsdurchlaufs

Zunächst wird durch das angesprochene Steuerungssystem überprüft, ob der ausführende Detektionsverantwortliche die notwendigen Rechte zur Detektion von Schwachstellen auf dem von ihm gewünschten Assets besitzt. Dies ist erforderlich, da die Detektion möglicherweise ausführliche, kritische Informationen an Angreifer liefern kann. Gleichzeitig wird durch die mandantenfähige Schnittstelle sichergestellt, dass derartige Daten nicht von unberechtigten

Personen eingesehen werden können. Dabei kann der Detektionsverantwortliche sowohl einzelne Assets anhand ihrer IP-Adresse im Netz auswählen, als auch gesamte Netzbereiche eingeben, in denen alle Assets, für die der Detektionsverantwortliche zuständig ist und bei der Detektion behandelt werden.

Die Detektion wird zur Vermeidung unnötigen Aufwands jedoch nur auf Schwachstellen angewendet, die in Bezug auf das betrachtete Asset, eine höhere Behebungsnotwendigkeit haben, als durch das **Schwachstellenakzeptanzkriterium** festgelegt ist (vgl. Abschnitt 5.6.4). Das Aussortieren irrelevanter Schwachstellen erfolgt bereits auf dem Steuerungssystem (auch betreffend der Auswahl von Schwachstellen zur Detektion durch Detektionsagenten). Ebenfalls aussortiert werden Schwachstellen, zu denen kein Detektionsverfahren dokumentiert ist und die infolgedessen nicht detektierbar sind, als auch durch die Filterung anhand der Kategorisierung von Schwachstellen im Abgleich mit dem Softwareinventar des jeweiligen Assets.

Schließlich weist das Steuerungssystem die Durchführung der Detektion der jeweiligen Assets je nach ihrer Anzahl einem bis mehreren Detektionssystemen zu. Diese führen Tests zur Detektion von Schwachstellen über das Netz aus und stoßen ebenfalls die Durchführung von Tests auf Assets durch Detektionsagenten an.

Die Implementierung eines Protokolls zur Anfrage einer Detektion auf Detektionagenten durch Detektionssysteme könnte wie folgt aussehen: Nachdem das Steuerungssystem alle irrelevanten Schwachstellen pro Asset anhand der auf dem jeweiligen Asset installierten Software sowie des Abgleichs der Behebungsnotwendigkeit einer Schwachstelle auf einem Asset mit dem Schwachstellenakzeptanzkriterium aussortiert hat, sendet es eine Liste an Identifikatoren der übrig gebliebenen relevanten Schwachstellen an das jeweilige Detektionssystem.
Dieses führt zum einen die Detektion aus Netzsicht für die identifizierten Schwachstellen aus und leitet die Liste relevanter Schwachstellen an den entsprechenden Detektionsagenten auf dem Asset weiter.
Der Detektionsagent erstellt ebenfalls eine Liste der Identifikatoren von auf dem Asset detektierten Schwachstellen und schickt diese über das Netz zurück an das Detektionssystem, welches die kombinierte Liste detektierter Schwachstellen aus beiden Sichten an das Steuerungssystem weiterleitet.
Ein derartiger Informationsaustausch kann beispielsweise über das HTTP bzw. HTTPS Protokoll realisiert werden.

Falls sowohl die Detektion über das Netz als auch die Detektion über Verfahren aus Systemsicht jeweils dieselbe Schwachstelle erkennen, werden die Daten zu einem einzigen positiven Detektionsergebnis zusammengefasst, um doppelte Einträge resultierender Schwachstellentickets zu vermeiden.

Da ein Asset immer noch komplett durch ein Detektionssystem behandelt wird, ist der Vergleich von detektierten Schwachstellen mehrerer Detektionssysteme untereinander nicht notwendig, da diese keine Duplikate enthalten. Schließlich liefern die Detektionssysteme ihr Ergebnis an das Steuerungssystem zurück, welches die Weiterverarbeitung der Informationen aus der Detektion übernimmt.
Falls es noch kein Schwachstellenticket für die gefundenen Schwachstellen gibt, wird jeweils eines erstellt. In diesem Fall weist das Steuerungssystem die Verantwortung zur Bearbeitung

der Schwachstellentickets dem hauptverantwortlichen Behebungsverantwortlichen des Assets zu. Dieser hat die Möglichkeit, die Schwachstellentickets bei Bedarf auf weitere für das Asset definierte Behebungsverantwortliche aufzuteilen.

Ist dagegen ein offenes Schwachstellenticket bereits für eine gefundene Schwachstelle vorhanden, wird dieser Eintrag entweder manuell durch den Detektionsverantwortlichen oder automatisch anhand verschiedener, in Abschnitt 5.7.4 beschriebener Kriterien eskaliert – bzw. nicht eskaliert, falls diese Kriterien nicht erfüllt sind und zugleich der Detektionsverantwortliche keine Eskalation vornimmt.

Auch kann der Fall auftreten, dass ein bereits geschlossenes Schwachstellenticket für die jeweilige, erneut detektierte Schwachstelle existiert, woraufhin für die erneut detektierte, identische Schwachstelle ein neues Ticket durch das Steuerungssystem erstellt wird.

Dadurch besteht jedoch die Gefahr, dass, aufgrund des bereits existierenden geschlossenen Schwachstellentickets derselben Schwachstelle auf demselben Asset, die erneute Erstellung des Schwachstellentickets eine Fehleinschätzung ist und die Schwachstelle tatsächlich bereits von den Behebungsverantwortlichen geschlossen wurde. Dies kann beispielsweise durch Verwendung eines ungeeigneten Verfahrens zur Detektion passieren, das die Schwachstelle als **False-Positive** detektiert. Bei Häufungen solcher Fehleinschätzungen durch Detektionsverfahren müssen die das Schwachstellenticket bearbeitenden Behebungsverantwortlichen einen Detektionsverantwortlichen informieren, welcher das jeweilige Detektionsverfahren überprüft.

Ein Problem bilden Schwachstellen, die nicht detektierbar, jedoch trotzdem auf einem Asset bekannt sind. In diesem Fall ist es nicht möglich, ein Schwachstellenticket automatisiert über den gerade beschriebenen Fall zu erstellen, da eine Automatik die Identifikation betroffener Assets durch das Fehlen eines Detektionsverfahrens nicht durchführen kann. Für derartige Fälle haben Detektionsverantwortliche die Möglichkeit, ein Schwachstellenticket manuell für ein oder mehrere Assets zu erstellen.

Ein **alternativer Durchlauf** zur Lösung des weiter vorne angesprochenen Problems von Detektionsagenten in privaten Netzen (vgl. Abschnitt 5.3.4) gestaltet sich durch das Initiieren der Detektion vom Detektionsagenten. In diesem Fall führt der Detektionsagent eine Detektion aus Systemsicht aus und schickt die Liste der detektierten Schwachstellen direkt an ein vordefiniertes Detektionssystem. Dieses schickt die Liste an das Steuerungssystem weiter, welches ab hier wie oben beschrieben weiter vorgeht: Durchführung der Prüfung auf die Existenz eines dazugehörigen Schwachstellentickets für jede detektierte Schwachstelle und, falls notwendig, die Erstellung eines solchen. Dieser Weg kann auch zur Kontrollprüfung und Eskalation genutzt werden (siehe Abschnitt 5.7.4). Herausforderungen, die dabei auftreten, sind, dass eine derartige Durchführung nicht zentral über das Steuerungssystem überwacht werden kann und lokal vom Asset ausgelöst werden muss. Dem Detektionsagenten muss außerdem mindestens ein Detektionssystem bekannt sein, an das er die Anfrage einer Detektion stellen kann. Bei Änderungen der Konfiguration zentraler technischer Komponenten kann insofern unter Umständen nicht gewährleistet werden, dass Assets stets aktuelle Informationen halten.

Wie bereits in Abschnitt 5.3.4 beschrieben, besteht jedoch vor allem hier die Gefahr der Manipulation, wodurch dieser Ansatz nicht im Konzept vorgesehen ist.

5.7.3 Behebung von Schwachstellen

Die Behebung (d.h. Beseitigung bzw. Abschwächung) von Schwachstellen auf Assets wird auf Basis vorhandener Schwachstellentickets vorgenommen, die in der Teilaktivität der Detektion (siehe vorheriger Abschnitt) erstellt wurden.

Die Abarbeitungsreihenfolge der Schwachstellentickets ist entsprechend des darin mitgelieferten Wertes der Behebungsnotwendigkeit auszuführen; das heißt Schwachstellentickets mit einer Behebungsnotwendigkeit *kritisch* müssen vorrangig vor allen anderen behandelt werden, gefolgt von Tickets mit Behebungsnotwendigkeit *hoch*, gefolgt von *mittel* und letztendlich *gering*.

In Hinblick auf eine möglichst schnelle und aufwandsarme Beseitigung bzw. Abschwächung der Schwachstelle kontrolliert der bearbeitende Behebungsverantwortliche, ob in der Schwachstellen-Dokumentation bereits eine entsprechende Maßnahme zur Lösung dokumentiert ist.

Im negativen Falle (es ist noch keine Maßnahme dokumentiert) besteht der nächste Schritt darin, in externen Quellen nach einer geeigneten Maßnahme zu suchen. Dazu sinnvolle Quellen stellen oft die in der Schwachstellen-Dokumentation importierten bzw. manuell eingetragenen Referenzen dar.

Erst im Fall, dass weder eine geeignete Maßnahme in der Schwachstellen-Dokumentation noch in Fremdquellen beschrieben ist, entwickelt der bearbeitende Behebungsverantwortliche selbst eine Lösung.

Alle noch nicht in der Schwachstellen-Dokumentation beschriebenen Maßnahmen, die zur Lösung der Schwachstelle im Hochschulnetz Anwendung finden, müssen als Maßnahme durch den Entwickler (in der Regel ein Behebungsverantwortlicher) dokumentiert werden.

Falls keine Lösung zur Beseitigung der Schwachstelle angewendet werden kann, ist mindestens die Anwendung eines entsprechenden Workarounds bzw. Temporary Fix notwendig, der bei Implementierung ebenfalls als Maßnahme dokumentiert und im Schwachstellenticket als angewendeter Workaround referenziert wird. Ein Workaround bzw. Temporary Fix unterscheidet sich von einer Lösung in der Regel dadurch, dass die Schwachstelle selbst nicht direkt geschlossen werden kann und lediglich weitere Maßnahmen zur Verhinderung der Ausnutzung angewendet werden (beispielsweise die Einführung weiterer Sicherheitsmaßnahmen wie den Schutz des Assets durch eine Firewall oder Implementierung einer zusätzlichen Authentifizierung).

Beispielsweise besteht ein möglicher Workaround für den Heartbleed Bug darin, die OpenSSL-Pakete ohne das betroffene Modul zu kompilieren. Ob eine Maßnahme ein Workaround ist oder die Schwachstelle letztendlich dadurch als behoben angesehen wird, ist jedoch oft abhängig vom konkret betroffenen Asset. Beispielsweise ist die Kompilierung der OpenSSL-Pakete ohne das betroffene Modul eine dauerhafte Lösung für ein Asset, das dieses Modul nicht benötigt, andererseits ein Workaround auf einem Asset, dessen Diensterbringung direkt von dem Modul abhängt und durch die Maßnahme eingeschränkt wird.

Diese Entscheidung ist durch den bestimmten Behebungsverantwortlichen zu treffen. Zudem muss generell abgewogen werden, ob die Anwendung einer Maßnahme in dafür vorgesehenen durch Wartungsintervalle definierte Zeitfenster geschieht bzw. vorgezogen wird. Letzteres ist vor allem bei besonders kritischen Schwachstellen sinnvoll.

> Zu beachten ist, dass ein Schwachstellenticket durch die Anwendung eines Workaround nicht geschlossen ist, sondern weiterhin offen bleibt, bis eine Lösung der Schwachstelle an sich erfolgt ist. Dabei ist es nicht von Bedeutung, ob die Lösung offiziell oder selbst entwickelt ist.

Ein Beispiel einer Maßnahme, die letztendlich für jede Schwachstelle in Betracht gezogen werden kann und die Schwachstelle beseitigt, ist die Außerbetriebnahme der betroffenen Software bzw. des Assets.

Auf jede Anwendung einer Maßnahme zur Beseitigung und Abschwächung folgt außerdem eine Überprüfung, zum einen auf die Wirksamkeit der Maßnahme und zum anderen auf mögliche negative Auswirkungen durch die Maßnahme. Dabei ist es nicht maßgeblich, ob es sich bei der Maßnahme um ein Workaround, ein Temporary Fix oder eine Maßnahme zur Beseitigung der Schwachstelle handelt.

Bei der Organisation der Daten in der Schwachstellen-Dokumentation tritt in dieser Teilaktivität eine Besonderheit im Bezug auf assetspezifische Schwachstellen auf. Assetspezifische Schwachstellen unterscheiden sich von anderen Schwachstellen dadurch, dass sie meist nur auf ein bestimmtes bzw. mehrere, jedoch immer vorher bekannte Assets beschrieben sind. Die Bezeichnung der Assets findet sich in der Regel in der Schwachstellenbeschreibung wieder. Nach der Behebung der Schwachstelle werden assetspezifische Schwachstellen in der Regel nicht mehr benötigt, da derartige Schwachstellen Einzelfälle sind. Falls der Eintrag nicht mehr gebraucht wird, kann sie manuell nach Behebung der Schwachstelle auf den betroffenen Assets mit Status *reject* versehen werden. So wird sie automatisch bei dem nächsten automatischen Aufräumen der Schwachstellen-Dokumentation entfernt.

Eine weitere empfohlene Vorgehensweise ist die Automatisierung der Behebung von Schwachstellen. Diese wird entweder durch Agenten auf Assets ausgeführt oder über das Netz mittels einer Möglichkeit zur Fernwartung wie beispielsweise der *Secure Shell* (SSH).

Eine große Herausforderung einer Automatisierbarkeit der Behebung als Teil des Schwachstellenmanagements stellt zum einen eine starke Heterogenität an Betriebssystemen und allgemein an Software in Netzen dar, da nicht überall dieselbe Lösung angewendet werden kann sowie andererseits die Vielfalt der Lösungen und deren (möglicherweise negativen) Auswirkungen. Aus diesem Grund kann in der Praxis oft lediglich nur eine gewisse Teilmenge an Schwachstellen auf Assets automatisch behoben werden – üblicherweise offizielle Patches, die zur automatischen Installation freigegeben sind.

Die in Abschnitt 5.3.1.1 beschriebene Möglichkeit zur Kategorisierung einer Maßnahme bietet die Grundlage einer automatisierten Installation von beispielsweise Patches. Eine Maßnahme, die als „freigegebener Patch“ und gemäß der anwendbaren Plattform kategorisiert ist, könnte durch eine Erweiterung der Funktionalität der Detektionsagenten automatisch angewendet werden.

Ein Problem, das vor allem aufgrund der organisatorischen Struktur im Hochschulnetz auftritt, ist die Bestimmung eines Patches, der für die automatische Installation freigegeben wird. Da Maßnahmen im Konzept im gesamten Hochschulnetz Anwendung finden sollen, ist es schwierig, die Anwendbarkeit eines Patches auf Basis von einem Institut bzw. einer Person (demjenigen, der den Patch am Ende zur automatischen Installation freigibt), für das

komplette Hochschulnetz zu bestimmen, da keine Person eines Instituts die Anforderungen und Softwareumgebung eines jeden anderen Instituts kennt.

Daher ist die automatische Installation von Patches zunächst nicht im Konzept für ein hochschulnetzweites Schwachstellenmanagement vorgesehen. Des Weiteren fällt diese Art der Behebung eher in den Bereich *hochschulweites „Patchmanagement“*.

Falls in der Umgebung (z.B. das Rechenzentrum oder ein Institut), in der das Schwachstellenmanagement umgesetzt wird, ein **Change Management** umgesetzt wird, werden insbesondere Änderungen an *Configuration Items* (CIs), die im Schwachstellenmanagement als Assets gelten, unbedingt über dieses abgewickelt. Aufgrund der Inhalte des Change Managements sind so üblicherweise Tests bezüglich der Wirksamkeit und möglicher Nebeneffekte von Änderungen (insbesondere Patches und Softwareupdates) bereits miteinbezogen. Ein weiterer Vorteil ist, dass durch die Abwicklung über das Change Management die Konsistenz der *Configuration Management Database* (CMDB) bezüglich der Softwareumgebung von Assets erhalten bleibt.

Ein Nachteil ist, dass die Abarbeitung über das Change Management die Behebung von Schwachstellen möglicherweise verzögert. Wirksame Maßnahmen wie Patches müssen daher – zumindest institutsweit – nach ihrer Prüfung vorautorisiert werden. Ein Testen der Maßnahmen auf ihre Wirksamkeit und negativen Auswirkungen muss jedoch trotzdem vorgenommen werden.

Bei Anwendung einer die Schwachstelle behebender Maßnahme wird das jeweilige Schwachstellenticket vom bearbeitenden Behebungsverantwortlichen abgeschlossen. Dabei ist außerdem die konkret angewendete Maßnahme zu dokumentieren.

5.7.4 Kontrollprüfung und Eskalation

Generell entspricht die Kontrollprüfung einer Ausführung der Detektion von Schwachstellen, mit dem Unterschied, dass keine neuen Schwachstellentickets angelegt werden, sondern im Falle ihrer **Existenz**, **Nicht-Abgeschlossenheit** und einer **positiven Bestätigung** durch die Detektion kontrolliert wird, ob Schwachstellentickets bearbeitet wurden oder nicht. Dadurch soll verhindert werden, dass eine detektierte Schwachstelle über längere Zeit nicht behoben wird oder gar trotz Hinweis in Form eines Schwachstellentickets ignoriert wird. Ein Mittel, um dies zu erreichen, besteht in der Eskalation von Schwachstellentickets.

Die **Eskalation eines Schwachstellentickets** ist angelehnt an die Eskalation von Tickets im *Incident & Service Request Management* Prozess und besteht grundlegend aus den Möglichkeiten der funktionalen Eskalation sowie der hierarchischen Eskalation.

Ersteres, die funktionale Eskalation, entspricht grob der Weiterreichung der Aufgabe – in diesem Fall der Behebung einer Schwachstelle – an eine Stelle mit notwendigen technischen Möglichkeiten und Berechtigungen sowie fachlichem Wissen. In diesem Fall ist im Konzept eine automatische funktionale Eskalation zum jeweiligen *Asset-Verantwortlichen* vorgesehen, da dieser letztendlich die Gesamtverantwortung für das Asset hat.

Eine weitere Möglichkeit der Eskalation ist in Form eines „Service Requests“ über den Prozess des *Incident & Service Request Managements* (ISRM) gegeben. Dies ist unabhängig von einer Kontrollprüfung, sondern manuell durchführbar und beispielsweise der Fall, wenn eine Schwachstelle durch weitere Maßnahmen beseitigt oder abgeschwächt werden kann, die

nicht in den Verantwortungsbereich eines Behebungsverantwortlichen des jeweiligen Assets fallen. Ein Beispiel hierfür ist das Vorschalten einer Firewall vor ein von einer Schwachstelle betroffenes Asset zur Einschränkung der Erreichbarkeit über das Netz. In diesem Fall gehen Behebungsverantwortliche den dafür vorgeschriebenen Meldeweg, oft in Form einer Ticketerstellung im ISRM.

Eine hierarchische Eskalation hingegen bezeichnet das „Informieren oder Einbeziehen höherer Management-Ebenen zur Unterstützung“ [Mar15]. Da die Kenntnis über den aktuellen Zustand des Schwachstellenmanagements im jeweiligen Bereich bzw. Institut zur Verantwortung des Schwachstellenmanagers gehört, muss dieser in besonderen Fällen in Kenntnis gesetzt werden, um das Entwickeln von Maßnahmen zur Schließung der jeweiligen Schwachstelle zu unterstützen. Diese Eskalation erfolgt manuell durch den bearbeitenden Behebungsverantwortlichen oder automatisch als Folge einer Kontrollprüfung und Erfüllung unten genannter Kriterien.

Die Kriterien zur Eskalation eines Schwachstellentickets müssen in solcher Weise gewählt werden, dass einerseits Schwachstellen nicht zu lange unbehoben und Assets bzw. Computersysteme angreifbar bleiben und zum anderen nicht jede Entscheidung durch den Schwachstellenmanager abhängig gemacht wird. Daher betrifft eine automatische Eskalation insbesondere Schwachstellen mit gewisser Kombination aus hoher Behebungsnotwendigkeit sowie einer bestimmten Dauer in ungelöstem Zustand.

Das Konzept sieht vor, dass insbesondere Schwachstellentickets mit einer als *hoch* und vor allem *kritisch* eingestuften Behebungsnotwendigkeit relativ zeitnah eskaliert werden.

Behebungsnotwendigkeit	**Zeitspanne im ungelösten Zustand**	**Meldung an**	**Eskalationslevel**
kritisch	3 Tage	Asset-Verantwortlicher	1
	7 Tage	Schwachstellenmanager	2
hoch	7 Tage	Asset-Verantwortlicher	1
	14 Tage	Schwachstellenmanager	2

Tabelle 5.9: Kriterien bei automatischer Eskalation

Da Schwachstellentickets mit kritischer Behebungsnotwendigkeit aufgrund der Kombination einer Schwachstelle mit mindestens hoher Kritikalität auf einem Asset mit mindestens hoher Risikoeinschätzung üblicherweise ein sehr hohes Schadenspotenzial haben und möglichst zeitnah behoben werden sollten, erfolgt eine automatische Eskalation bei relativ geringer Dauer im ungelösten Zustand. Im Konzept ist vorgesehen, dass derartig eingestufte Schwachstellentickets in diesem Fall innerhalb von drei Tagen im ungelösten Zustand an den Asset-Verantwortlichen des jeweiligen Assets eskaliert werden, um mögliche Auswirkungen einer Ausnutzung der Schwachstelle auf diesem genauer einzuschätzen und die Behebungsverantwortlichen bei der Behebung zu unterstützen. Eine Kommunikation mit den zuständigen Behebungsverantwortlichen ist zur Klärung der Problematik und Lösungsfindung sehr empfehlenswert.

Eine hierarchische Eskalation in Form einer informativen Meldung über den entsprechenden Eintrag erfolgt an den Schwachstellenmanager nach sieben Tagen. Dieser muss schließlich die Lösung aus Sicht einer höheren Managementebene koordinieren und kann ein Team zur Erstellung einer Maßnahme zur Lösung der Schwachstelle zusammenstellen.

Die Eskalation von Schwachstellentickets mit einer als *hoch* eingeschätzten Behebungsnotwendigkeit geschieht genauso zunächst an den Asset-Verantwortlichen und darauf an den zuständigen Schwachstellenmanager. Aufgrund der allgemein weniger dringlichen Behebungsnotwendigkeit ist es nicht notwendig, die Eskalation bei weniger oder gleichgroßer Zeitspanne des Schwachstellentickets im ungelösten Zustand durchzuführen und erfolgt daher später. Entsprechend sieht das Konzept vor, ungelöste Schwachstellentickets mit Behebungsnotwendigkeit *hoch* nach sieben Tagen ohne Lösung an den Asset-Verantwortlichen zu eskalieren und nach einer Zeitspanne von 14 Tagen an den zuständigen Schwachstellenmanager.

Schwachstellentickets mit einer Behebungsnotwendigkeit von *gering* oder *mittel* werden aufgrund ihrer geringeren Kritikalität und einer Ersparnis an Aufwand nicht automatisch gemeldet. Der zuständige Behebungsverantwortliche hat jedoch die Möglichkeit, diese bei erforderlicher Unterstützung manuell an den Asset-Verantwortlichen zu eskalieren oder im Extremfall dem Schwachstellenmanager zu melden. Ein Grund für eine Meldung ist üblicherweise die Unbehebbarkeit der Schwachstelle.

Ähnliches gilt für Schwachstellentickets für Schwachstellen, die aufgrund des Fehlens eines Detektionsverfahrens nicht detektierbar sind und somit nicht automatisch überprüft werden können. In diesem Fall werden nach Ablauf der Zeitspanne in ungelöstem Zustand anstatt einer Durchführung der Kontrollprüfung die Detektionsverantwortlichen des betroffenen Assets über das Webportal informiert.
Diese müssen sich auf irgendeine Art und Weise darüber informieren, ob die im Schwachstellenticket notierte Schwachstelle noch immer auf dem Asset vorhanden ist – beispielsweise über den manuellen Abgleich der betroffenen Softwarepakete. Im positiven Fall gibt ein Detektionsverantwortlicher im Webportal an, dass die Schwachstelle noch offen ist, woraufhin das Steuerungssystem den entsprechenden Mechanismus zur Eskalation vornimmt.

Eine Zusammenfassung des Eskalationsmechanismus ist in Tabelle 5.9 beschrieben.

Die Zeitspanne, nach der eine Kontrollprüfung folgt, kann je nach Bedarf gewählt werden. Aus Gründen der Effizienz ist für die Kontrollprüfung kein festes Intervall vorgesehen, in der kontinuierlich geprüft wird, sondern die Zeitspanne ist angepasst an den Eskalationsmechanismus. So liest das Steuerungssystem das Datum der Eröffnung jedes Schwachstellentickets aus und vergleicht den Abstand mit Zeitspannen, in denen automatisch eskaliert wird. In diesem Fall erfolgt die Kontrollprüfung für ein Schwachstellenticket mit Behebungsnotwendigkeit *kritisch* exakt drei Tage resp. sieben Tabe nach der Eröffnung, für ein Schwachstellenticket mit Behebungsnotwendigkeit *hoch* nach sieben bzw. 14 Tagen.

5.8 Prävention von Schwachstellen

In der Prävention ist das verfolgte Ziel ähnlich wie in der Beseitigung und Abschwächung das Auftreten von Schwachstellen auf Systemen im Netz generell zu reduzieren. Anders als

in der Beseitigung und Abschwächung, in der Schwachstellen nach deren Detektion behoben werden, setzt die Prävention in einem Netz von Anwendern vor dem Einsatz von Software auf Assets an. So fällt, verglichen mit der Aktivität der Beseitigung und Abschwächung, die Teilaktivität der Detektion in der Prävention weg. Gleiches gilt für die Kontrollprüfung, da diese durch eine Detektion realisiert wird und entsprechend in die Aktivität der Beseitigung und Abschwächung eingeordnet wird. Entsprechend sind es direkt Maßnahmen, die in dieser Aktivität identifiziert und umgesetzt werden.

Im Konzept setzt die Prävention von Schwachstellen ebenfalls an einem IT-Service-Management-Prozess an, über den auch im Hochschulumfeld durch kontrolliertes Vorgehen Änderungen an der Konfiguration der Umgebung geplant und umgesetzt werden; dem Change Management. Da einer der Hauptaspekte darin die Implementierung auf eine Art und Weise ist, dass unerwünschte negative Auswirkungen verhindert werden, besteht ohnehin eine Überschneidung zwischen Change Management und Schwachstellenmanagement, da Schwächen zu derartigen unerwünschten Nebenwirkungen führen können.

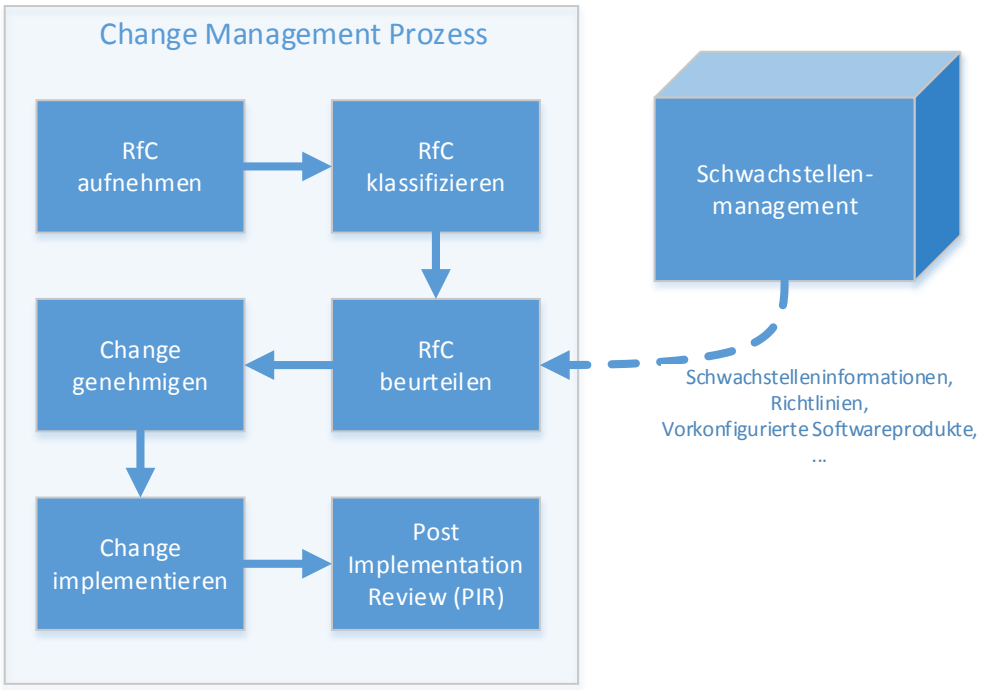


Abbildung 5.9: Nutzung von Schwachstelleninformationen im Change Management Prozess

Die Aktivitäten im Change Management umfassen die Aufnahme, Klassifikation und Beurteilung von Änderungsanfragen (engl. „Request for Change“, RfC) sowie die Genehmigung und Implementierung von Änderungen (engl. „Change“) als auch die Überprüfung der Änderung nach der Implementierung. Eine Möglichkeit zur Prävention von Schwachstellen besteht darin, Informationen über Schwachstellen aus dem Schwachstellenmanagement in der Beurteilung von RfCs zu verwenden. Eine Veranschaulichung ist in Abbildung 5.9 illustriert. So kann beispielsweise bereits vor dem Einsatz einer Software oder eines Upda-

tes überprüft werden, ob und in welchem Umfang Schwachstellen für ein Produkt bekannt sind. Infolgedessen können vor Installation eines Softwarepakets etwaige Risiken abgewogen werden.

Da die Prävention generell unabhängig von einem bestimmten Asset ist, ist entsprechend die Behebungsnotwendigkeit kein Kriterium zur Bestimmung der Notwendigkeit zur Entwicklung und Ergreifung einer präventiven Maßnahme.

Entsprechend wird ein praktischerer Ansatz gewählt und die in der Schwachstellen-Dokumentation hochschulnetzweit am häufigsten detektierten Schwachstellen in einem bestimmten, konfigurierbaren zeitlichen Abstand (z.B. monatlich, halbjährlich,...) zur Ergreifung präventiver Maßnahmen ausgewählt. Eine positive Detektion ist an einem erstellten Schwachstellenticket zu erkennen. Auch die Anzahl der schließlich behandelten Schwachstellen kann variabel getroffen werden. Beispielsweise werden zu den zehn häufigsten Schwachstellen im Quartal Maßnahmen zur Prävention entwickelt.

Maßnahmen im Rahmen der Prävention des Schwachstellenmanagements resultieren auch in Form von institutsweiten Richtlinien, beispielsweise durch eine Liste mit aus informationssicherheitstechnischer Sicht empfehlenswerten sowie nicht-empfehlenswerten Softwareprodukten – kategorisiert nach Art der Anwendung (z.B. Webservice, Datenbanksystem, Laufzeitumgebung und weitere). Derartige Übersichtslisten und Richtlinien werden beispielsweise monatlich anhand von Informationen aus Schwachstellenstatistiken erstellt.

Weitere beispielhafte Inhalte solcher Richtlinien können durch Checklisten bezüglich der Softwarelandschaft in Assets nach Typ beschrieben sein. Beispielsweise stellen installierte Client-Anwendungen auf Serversystemen in der Regel ein unnötiges Risiko dar, welches vermieden werden kann.

Eine weitere Maßnahme zur Prävention von Schwachstellen ist die Schulung von Nutzern im Hochschulnetz – hier vor allem Dienstbetreiber, Asset-Owner und Administratoren, jedoch auch Dienstnutzer. Inhalte der Schulungen gehen vom Umgang mit Schwachstellen bei ihrer Entdeckung (d.h. Meldung an das Schwachstellenmanagementsystem in Form der Eröffnung eines Meldungstickets) bis hin zum konkreten Aufzeigen von Schwachstellen im Alltag, wie z.B. durch Demonstration der Ausnutzung von Schwachstellen wie einer *SQL-Injection* und resultierender Auswirkungen. Auch kann das allgemeine Informieren über Schwachstellen in diese Kategorie fallen – beispielsweise per E-Mail oder einen abonnierbaren RSS („Really Simple Syndication“) Feed.

Im Konzept ist als präventive Maßnahme zur Schulung von Nutzern im Mindesten die Bereitstellung von Schwachstelleninformationen und Erstellung von Berichten über aktuell stark verbreitete Schwachstellen (beispielsweise anhand erstellter Schwachstellentickets) vorgesehen.

Ebenfalls ein wichtiger Ansatzpunkt ist der Entwickler von Software, durch den im allgemeinen Schwachstellen erst in Software auftreten. Diese müssen gleichermaßen durch geeignete Schulungen bezüglich Schwachstellen sensibilisiert werden.

Zur Vermeidung von Konfigurationsschwachstellen ist im Konzept das Angebot von vorkonfigurierten Softwarepaketen vorgesehen, die Nutzern des Hochschulnetzes zur Verfügung gestellt werden. Beispielsweise werden Konfigurationsdateien für Webserver wie den *Apache HTTP Server* angeboten, in welchen die Sicherung der Kommunikation durch TLS mit geeigneten Ciphersuites bereits aktiviert ist.

Für die Entwicklung und Ausarbeitung von Maßnahmen sind allgemein Personen in der Rolle des Präventionsverantwortlichen zuständig.

5.9 Kontinuierliche Verbesserung

Eine kontinuierliche Verbesserung der Abläufe ist einer der zentralen Aspekte im Management allgemein und zielt darauf ab, Abläufe im Management durch geeignete Leistungsindikatoren (KPIs) zu messen und durch darauf aufbauende Anpassungen im Managementprozess die Effektivität – beispielsweise institutsweit durch Ausweitung des Anwendungsbereichs – und die Effizienz des Managements zu erhöhen.

Die Kontinuierliche Verbesserung ist nicht direkt Teil des Prozesses des Schwachstellenmanagements (siehe Abbildung 5.4), sondern mehr als nebenläufiger Prozess zu erachten, welcher ebenso in regelmäßigen Abständen durchlaufen werden muss. Die Durchlaufintervalle sind üblicherweise deutlich größer als die des Schwachstellenmanagementprozesses, da die Durchführung der Aktivitäten auf Informationen, den Leistungsindikatoren, aus dem Prozess des Schwachstellenmanagements basieren und dort zunächst generiert werden.

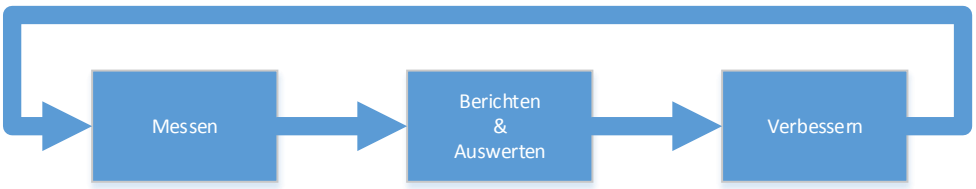


Abbildung 5.10: Allgemeiner Ablauf der kontinuierlichen Verbesserung

Die Aktivitäten in der kontinuierlichen Verbesserung bestehen allgemein aus der Messung, der Evaluierung und letztendlich der Verbesserung von Abläufen und Elementen des Schwachstellenmanagements (vgl. Abbildung 5.10).

Leistungsindikatoren dienen als Hinweis auf die Effektivität und Effizienz des Managements, in diesem Fall des Schwachstellenmanagements, und müssen klar definiert sein. Als Kriterien der Eignung von Leistungsindikatoren werden in der Regel die Erfüllung von Spezifität, Messbarkeit, Erreichbarkeit, Relevanz sowie zeitliche Terminierbarkeit herangezogen. [Sch14f]

Als mögliche Leistungsindikatoren im Schwachstellenmanagement werden in diesem Konzept folgende Aspekte betrachtet:

- Die **Durchschnittliche Dauer zur Abschließung eines Schwachstellentickets**, welche auf verschiedenen Detailebenen betrachtet werden kann. Sinnvolle Betrachtungen sind auf Ebene des gesamten, hochschulweiten Schwachstellenmanagements sowie auf Ebene der einzelnen Institute.

  Dieser Leistungsindikator eignet sich gut, um die Effektivität des Schwachstellenmanagements innerhalb der einzelnen Institute sowie der Institute untereinander zu vergleichen. Auch dient es als geeignetes Kriterium zur Kontrolle der Effektivität des

Schwachstellenmanagements im gesamten Hochschulnetz sowie einzelner Institute im Verlauf der Zeit.

- Das Verhältnis der Anzahl **geöffneter zu geschlossener Schwachstellentickets** lässt auf die Effektivität der Behebung im Schwachstellenmanagement schließen. Diese wird instituts- und netzweit betrachtet.

- Die **Anzahl automatisch eskalierter Schwachstellentickets** – ebenfalls netzweit und institutsweit – wodurch die Eignung des Eskalationsmechanismus evaluiert werden kann.

  Werden beispielsweise in einer gewissen Zeitspanne zu viele Schwachstellentickets an den jeweiligen Schwachstellenmanager eskaliert, müssen die Eskalationskriterien unter Umständen angepasst werden.

- Die **Anzahl der akzeptierten Schwachstellentickets**, d.h. Schwachstellentickets mit einer geringeren oder gleich großen Behebungsnotwendigkeit als der vom *Schwachstellenakzeptanzkriterium* festgelegte Wert. Dieser Leistungsindikator lässt auf die geeignete Festlegung des Schwachstellenakzeptanzkriteriums schließen. Da das Schwachstellenakzeptanzkriterium ein netzweiter Faktor ist, ist vor allem der Gesamtüberblick notwendig. Auch kann dieser Faktor bei der Einschätzung der Eignung des verwendeten Bewertungssystems helfen.

  Falls beispielsweise alle Schwachstellentickets akzeptiert werden, obwohl das Schwachstellenakzeptanzkriterium auf *gering* festgelegt ist, kann davon ausgegangen werden, dass das Bewertungssystem unter Umständen nicht geeignet ist und ersetzt werden sollte.

Die institutsinterne Auswertung bzw. Ausführung des Prozesses der kontinuierlichen Verbesserung in Bezug auf das Schwachstellenmanagement ist Aufgabe des jeweils eingesetzten Schwachstellenmanagers. Da die Anpassung netzweiter Strukturen wie dem Schwachstellenakzeptanzkriterium oder dem Bewertungssystem nicht durch die Schwachstellenmanager geändert werden kann, umfassen Verbesserungsmaßnahmen insbesondere Anpassungen in der Rollenverteilung und den Abläufen.

Die hochschulnetzweite Auswertung der Leistungsparameter ist Teil des Aufgabenspektrums des Chief Vulnerability Managers. Als Informationsgrundlage dienen ihm zusammengefasste Berichte der einzelnen Institute, die er durch den jeweils tätigen Schwachstellenmanager in halbjährlichem (oder ähnlich großem) Abstand erhält. Der Zeitraum dafür kann je nach Bedarf vom Chief Vulnerability Manager angepasst werden. Deutlich kürzere Abstände sind in der Regel aufgrund der eher langen Dauer eines kompletten Durchlaufs durch den Schwachstellenmanagementprozess nicht sinnvoll, da unter Umständen nicht ausreichend viele Informationen zum Vergleich mehrerer Institute vorhanden sind.

Die gelieferten Berichte umfassen stets ausschließlich Daten seit dem letzten Bericht. Verbesserungsmaßnahmen, die durch den Chief Vulnerability Manager vorgenommen werden, betreffen in der Regel Anpassungen des Bewertungssystems oder des Schwachstellenakzeptanzkriteriums, falls netzweit eine entsprechende Problematik (siehe oben) erkannt wird und eine Anpassung eine Verbesserungsaussicht bietet.

Zur Kommunikation vorgenommener, wirksamer Verbesserungsmaßnahmen sowie der Beratung über mögliche zukünftige Maßnahmen ist eine jährliche Absprache des Chief Vulnerability Managers sowie aller Schwachstellenmanager vorgesehen.

Die im Rahmen des Prozesses der kontinuierlichen Verbesserung erstellten Berichte werden zudem an Personen in der Rolle des *Chief Information Security Officers* (CISO), oder Personen in gleichbedeutender Funktion, in den jeweiligen Instituten kommuniziert, da sie hohe Relevanz für die Informationssicherheit haben.

5.10 Varianten der Umsetzungen des Konzepts

Wie bereits im Abschnitt 5.1 beschrieben, ist der Anwendungsbereich des Schwachstellenmanagements und infolgedessen das Konzept in Bezug auf die umzusetzenden Aktivitäten, Arten von Schwachstellen, Assets als auch technischen Komponenten zur Unterstützung beliebig einschränkbar und genauso (generell mit Ausnahme der Aktivitäten) erweiterbar.

Betreffend der Umsetzung des Schwachstellenmanagements gibt es nicht nur deswegen, sondern insbesondere auch durch die Art der Unterstützung der Kunden und Nutzer im Hochschulnetz verschiedene Varianten.

5.10.1 Varianten des Aktivitätenzyklus

Die in Abschnitt 5.4 beschriebene sequenzielle Reihenfolge zur Abarbeitung der Aktivitäten ist empfehlend und stellt einen idealen, vollständigen Durchlauf durch den Prozess dar. In der Praxis kann es jedoch unter Umständen häufig passieren, dass beispielsweise nach der Behebung einer Schwachstelle die Aktivität der Prävention übersprungen und mit der Identifikation fortgefahren wird.

Andererseits kann auch die Aktivität der Identifikation und Klassifikation übersprungen werden, falls beispielsweise die Suche nach neuen Schwachstellen ausgesetzt, die Beseitigung und Abschwächung der dokumentierten Schwachstellen auf vorhandenen Assets jedoch fortgeführt wird. Gleiches gilt auch, falls vorhandene Schwachstellen neu klassifiziert werden, zum Beispiel durch Setzen des Wertes der *Temporären Dringlichkeit* für eine oder mehrere Schwachstellen durch den Chief Vulnerability Manager und einer darauf folgenden Detektion sowie Behebung dieser Schwachstellen.

Je nach Anwendungsbereich besteht auch die Möglichkeit, eine Aktivität oder Teilaktivität komplett wegzulassen. Ist es beispielsweise aufgrund ausreichend vorhandener personeller Ressourcen möglich, alle identifizierten Schwachstellen innerhalb des Anwendungsbereichs zeitnah zu beheben, ist zumindest die Bewertung von Schwachstellen überflüssig, da eine Priorisierung und Filterung relevanter Schwachstellen nicht unbedingt notwendig ist.

5.10.2 Varianten des Schwachstellenmanagements als Dienstleistung

Im Konzept wird das Hochschulrechenzentrum als zentrale Einrichtung im Schwachstellenmanagement angesehen. Der Vorteil, der sich daraus ergibt, ist vor allem, aufgrund seiner Rolle als Netzbetreiber, technische Komponenten wie die Schwachstellen-Dokumentation und die Asset- und Benutzerverwaltung ebenfalls betreiben und deren Nutzung als Dienstleistung an andere Einrichtungen im Hochschulnetz anbieten zu können. So wird garantiert, dass Informationen hochschulnetzweit gleichartig durch eine zentrale Schnittstelle beschrieben werden. Ein anderer wichtiger Aspekt, der dadurch erfüllt wird, ist die gemeinsame

Erbringung der Informationen, wodurch insbesondere der Aufwand manueller Tätigkeiten auf Personen im gesamten Hochschulnetz verteilt wird.

Generell kann für alle Nutzer im Hochschulnetz ein Dienst angeboten werden, der beispielsweise per E-Mail oder RSS-Feed neueste Schwachstellen meldet, für die sich ein Nutzer interessiert. Dazu kann ein Nutzer eine Liste von Softwareprodukten angeben, über deren Zustand bezüglich Schwachstellen er informiert werden will. Das System sucht entsprechende Einträge in der Schwachstellen-Dokumentation anhand ihrer Kategorisierung heraus und verteilt diese. Eine zusätzliche Möglichkeit zum Filtern kann anhand der Kritikalität einer Schwachstelle erfolgen.

Auch können nicht nur Informationen aus dem Schwachstellenmanagement als Dienstleistung angeboten werden, sondern vor allem auch technische Systeme zur Detektion von Schwachstellen. Insbesondere die durch Detektionssysteme bereitgestellten Möglichkeiten können beispielsweise auf Geräten angewendet werden, die nicht direkt als Asset in der Asset- und Benutzerverwaltung deklariert sind.
So kann jedem Nutzer im Hochschulnetz über eine speziell angepasste Benutzeroberfläche auf dem Steuerungssystem die Funktion angeboten werden, Schwachstellen auf dem Gerät, mit dem der Nutzer auf die Oberfläche zugreift, zu detektieren. Ergebnisse gefundener Schwachstellen werden entsprechend nicht in Form von Schwachstellentickets dokumentiert, sondern direkt auf der Weboberfläche mit – falls vorhanden – dazu passenden Maßnahmen zur Behebung der Schwachstelle angezeigt. Zusätzlich kann der Detektionsagent als Paket mit einfacher Möglichkeit zur Installation im Netz zur Verfügung gestellt, durch Nutzer heruntergeladen und auf ihren Systemen einsetzen werden. Allgemein nutzbare Angebote der Detektion sollten zum Erhalt der Einfachheit plattformunabhängig implementiert sein, damit sie von allen Netznutzern anwendbar sind.

Auf diese Weise wird ebenfalls die in Abschnitt 5.2 erwähnte Verantwortlichkeit der Nutzer zur Behebung von Schwachstellen auf den von ihnen in das Hochschulnetz integrierten privaten Geräten mit notwendigen Informationen und technischen Möglichkeiten zur Erkennung unterstützt. Ein Beispiel wird in Abschnitt 7.7 erläutert.

5.11 Erfüllung der Anforderungen

Im Folgenden werden die aus Kapitel 3 Abschnitt 3.2 gestellten Anforderungen an ein Schwachstellenmanagement im Hochschulnetz mit dem Konzept abgeglichen. Eine Erläuterung ist in den folgenden Abschnitten zu finden.

5.11.1 Zusammenfassung der Erfüllung

Die Zusammenfassung der Erfüllung der Anforderungen an ein Schwachstellenmanagement in Hochschulnetzen ist in Tabelle 5.10 gezeigt. Eine Erfüllung der Anforderung wird durch ein grünes Häkchen (✓) ausgedrückt, eine Nicht-Erfüllung durch ein rotes Kreuz (✗).
Für eine bessere Unterscheidbarkeit der Gewichtung werden Anforderungen mit einer Gewichtung „empfohlen“ kursiv hervorgehoben.

ID	Beschreibung	Erfüllung
	Allgemeine Anforderungen	
NA1	Existenz eines Asset-Inventars	✓
NA2	Existenz und Zentralisierung einer Schwachstellen-Dokumentation	✓
NA3	*Zentralisierung des Asset-Inventars*	✓
FA4	Funktionsumfang des Asset-Inventars	✓
FA5	*Zugriff auf Informationen und Funktionen über eine API*	✓
FA6	Aktualität des Asset-Inventars	✓
NA7	Automatisierung der Aktivitäten	✓
NA8	Unterstützung manueller Tätigkeiten	✓
NA9	Plattformunabhängigkeit der Aktivitäten	✓
FA10	*Verbinden mit weiteren sicherheitsrelevanten Informationen*	✓
FA11	*Exportierbarkeit der Daten*	✓
NA12	*Präsentation der Daten*	✗
NA13	Mandantenfähigkeit	✓
NA14	Institutsübergreifende Benutzerverwaltung	✓
NA15	Flexibilität des Anwendungsbereichs	✓
NA16	Zweckdienlichkeit des Anwendungsbereichs	✓
NA17	Definition der Aktivitäten	✓
NA18	Ausübungsintervall des Aktivitätenzyklus	✓
NA19	Definition von Verfahren	✓
NA20	Definition von Rollen und Verantwortlichkeiten	✓
NA21	Prüfung durch Leistungsindikatoren	✓
NA22	Dezentralisierung der Aktivitäten	✓
NA23	Schnittstellen zu anderen Prozessen	✓
NA24	Identifikation von Assets	✓
NA25	Qualität der Asset-Beschreibung	✓
NA26	Definition eines Asset-Verantwortlichen	✓
NA27	Nutzung geeigneter Meldewege	✓
FA28	Erstellung von Berichten	✓
	Identifikation	
FI1	(Manuelles) Hinzufügen neuer Einträge in die Schwachstellen-Dokumentation	✓
FI2	Zugriff auf Einträge in der Schwachstellen-Dokumentation	✓
FI3	Editieren von Einträgen in der Schwachstellen-Dokumentation	✓
FI4	Suchfunktionalität innerhalb der Schwachstellen-Dokumentation	✓
FI5	Automatischer Erhalt der Aktualität der Schwachstellen-Dokumentation	✓

NI6	Dynamische Festlegung der Quellen der Schwachstellen-Dokumentation	✓
FI7	Filtern relevanter Schwachstellen	✓
NI8	Aktualität der Schwachstellen-Dokumentation	✓
NI9	Eindeutiger Bezeichner einer Schwachstelle	✓
NI10	Qualität der Schwachstellenbeschreibung	✓
Klassifikation		
FK1	Automatisierung der Klassifikation	✓
FK2	Manuelle Klassifikation	✓
NK3	Einheitlichkeit und Qualität der Schwachstellenbewertung	✓
NK4	Berücksichtigung der Umgebung	✓
NK5	Kritikalität von Assets	✓
NK6	Kriterium der Schwachstellenakzeptanz	✓
Beseitigung und Abschwächung		
FB1	Verfahren zur Detektion von Schwachstellen	✓
FB2	Automatisierung der Detektion	✓
FB3	*Detektion aus Systemsicht*	✓
FB4	*Detektion aus Netzsicht*	✓
FB5	Unterstützung bei der Behebung durch Computersysteme	✓
NB6	*Zentralisierung der Methoden und Daten der Detektion*	✓
NB7	*Definition von Maschinen zur Detektion*	✓
NB8	Zeitnahe Beseitigung und Abschwächung	✓
NB9	Prüfung einer Maßnahme auf Wirksamkeit	✓
NB10	Berücksichtigung von Risiken	✓
NB11	Priorisierung kritischer Schwachstellen	✓
NB12	Priorisierung kritischer Assets	✓
FB13	Möglichkeit einer effektiven Behebung	✓
Prävention		
NP1	Risikoeinschätzung vor Einsatz von Software	✓

Tabelle 5.10: Erfüllung der Anforderung durch die beschriebenen Lösungen

5.11.2 Erläuterung der Bewertung

In den folgenden Abschnitten wird zur Einordnung der Anforderungen der Zusammenhang zum Konzept hergestellt.

5.11.2.1 Erfüllte Anforderungen

Bezüglich des Inhalts eines Managementprozesses sind alle notwendigen Aspekte festgelegt. Das Konzept beinhaltet Rollen für Einzelpersonen im Hochschulnetz und darauf aufbauend die Beschreibung von Verantwortlichkeiten für Verfahren bzw. Teilaktivitäten innerhalb der festgelegten Aktivitäten im Schwachstellenmanagement. Die Aktivitäten umfassen

die Identifikation, Klassifikation, Beseitigung und Abschwächung sowie die Prävention von Schwachstellen, deren Ausübung durch das im Konzept beschriebene Rollenmodell netzweit stattfindet und in ausreichend kurzen Zeitintervallen durchlaufen wird. Die eigentliche Umsetzung ist auch abhängig von der Wahl des Anwendungsbereichs, welcher im Konzept, bezogen auf im Hochschulnetz befindliche Institute und Organisationen sowie Typen der Assets, möglichst generell gehalten ist. Daher sind die Anforderungen NA9, NA15 bis NA18, NA19 und NA29 sowie NA22 erfüllt.

Die Kommunikation der Rollen untereinander ist mittels definierter Meldewege vorzunehmen, wodurch Anforderung NA27 ebenfalls erfüllt ist.

Das Konzept sieht eine möglichst weitgreifende Automatisierung der Tätigkeiten vor, hier vor allem durch den Import von Informationen aus Fremdquellen. Die darin enthaltenen Informationen werden ausgewertet und können je nach Informationsumfang die Identifikation und Klassifikation bis hin zur Detektion komplett automatisiert durchführen. Die Automatisierung der Detektion aus Netz- und Systemsicht wird durch ein entsprechend dafür vorgesehenes System aus einer Zeitablaufsteuerung (einem „Scheduler“), den notwendigen zentralen Informationen (Detektionsverfahren, Assets und Schwachstellen) und technischen Komponenten realisiert. Daher können auch Anforderungen NA7, FK1, FB1 bis FB4, NB6 und NB7 vom Konzept erfüllt werden. Jeder der automatisch ausgeführten Schritte kann jedoch auch manuell bzw. mittels technischer Unterstützung ausgeführt werden (Anforderungen NA8 und FK2).

Auch die Unterstützung der Behebung wird durch eine in der Schwachstellen-Dokumentation verwaltete Sammlung an Maßnahmen realisiert, die durch Personen im Hochschulnetz erweitert und eingesehen werden können (Anforderung FB5). Die Maßnahmen zur Behebung sind keinen Einschränkungen unterlegen, sofern sie eine Schwachstelle effektiv schließen (Anforderung FB13). Das Konzept sieht jedoch auch vor, dass Maßnahmen bezüglich möglicher Risiken und ihrer Wirksamkeit überprüft werden müssen (Anfoderungen NB10 und NB9).

Im Konzept ist jeweils eine zentrale Schwachstellen-Dokumentation sowie eine zentrale Asset- und Benutzerverwaltung, beide mit notwendigem Funktionsumfang zum Zugriff sowie Funktionen zur Suche nach Daten, angeboten, die prinzipiell im gesamten Hochschulnetz genutzt werden können. Der Zugriff für Nutzer erfolgt primär über eine dafür vorgesehene mandantenfähige Nutzeroberfläche. Somit erfüllt das Konzept die Anforderungen NA1 bis FA4, NA14 und FI1 bis FI4.

Diese dienen neben weiteren Komponenten der technischen Unterstützung im Schwachstellenmanagement und vor allem als zentrale Stellen zur Haltung der Daten. Sowohl die Attribute zur Beschreibung einer Schwachstelle als auch zur Beschreibung eines Assets sind somit im gesamten Hochschulnetz einheitlich mit jeweils einem netzweit eindeutigen Identifikator, wodurch die Anforderungen NA24 bis NA26 sowie NI9 und NI10 erfüllt werden.

Die Aktualität der Schwachstellen-Dokumentation wird durch die Möglichkeit des automatisierten Imports von Schwachstellen aus beliebigen Fremdquellen sichergestellt. Die aus den Fremdquellen importierten Daten können auf Basis beliebiger Kriterien wie dem Datum einer Schwachstelle oder anhand der Bewertung einer Schwachstelle ausgefiltert und somit entsprechend dem Anwendungsbereich angepasst werden. Somit werden Anforderungen FI5 bis NI8 erfüllt.

Die Aktualität der Asset- und Benutzerverwaltung wird mittels der Kombination zweier Ansätze sichergestellt: Zum einen durch die Zuweisung der Verantwortung dafür an den

Asset-Verantwortlichen (für die Aktualität seiner Assets) und der Regelung, dass es für jedes Asset genau einen Verantwortlichen geben muss. Zum anderen automatisiert durch den Import mittels einer API der Asset- und Benutzerverwaltung (und ebenfalls der Schwachstellen-Dokumentation), welche über die API des Steuerungssystems genutzt werden kann, hier insbesondere von der CMDB und Verzeichnisdiensten wie LDAP. Insofern können auch Anforderungen `FA5` und `FA6` erfüllt werden. Über die API sind Daten im Schwachstellenmanagement zudem exportierbar (Anforderung `FA11`).

Die Bewertung von Schwachstellen erfolgt im Konzept zum einen unter Berücksichtigung der Aspekte der Eintrittswahrscheinlichkeit und Auswirkung der Ausnutzung einer Schwachstelle, sowie durch die Miteinbeziehung der Kritikalität von Assets. Das Bewertungssystem ist dabei möglichst praktisch und unkompliziert gehalten. Entsprechend sind Anforderungen `NK3`, `NK4` und `NK5` erfüllt.

Die zeitnahe Behebung von Schwachstellen wird durch einen Eskalationsmechanismus unterstützt, durch den noch offene, detektierte Schwachstellen einer konfigurierbaren Kritikalität und Dauer im offenen Zustand gemeldet werden. Neben der durch das Bewertungssystem möglichen Priorisierung wird zudem ein Schwachstellenakzeptanzkriterium eingeführt, das beliebig netzweit gesetzt werden kann und den Fokus der Behebungsverantwortlichen auf relevante Schwachstellen lenkt. Dadurch können die Anforderungen `NK6`, `NB8`, `NB11` und `NB12` erfüllt werden.

Zur Prävention von Schwachstellen sieht das Konzept die Erstellung von Maßnahmen als Reaktion auf Schwachstellen vor. Zudem wird durch Verbindung mit dem Change Management ermöglicht, die Risiken einer Anpassung der Konfiguration eines Computersystems in Bezug auf Softwareprodukte anhand vorhandener Schwachstellen einzuschätzen. Somit sind Anforderungen `NA23` und `NP1` erfüllt.

Anforderung `FA10`, die Verknüpfung der Daten im Schwachstellenmanagement mit weiteren sicherheitsrelevanten Daten ist erfüllt, da Ergebnisse aus dem *Security Information & Event Management* bei der Bewertung von Schwachstellen genutzt werden. Auf Auftreten eines Sicherheits-Ereignisses betreffend der Ausnutzung einer Schwachstelle wird der Wert der *Bekanntheit von Fällen einer Ausnutzung* auf *ja* gesetzt, falls dies nicht bereits der Fall ist.

Im Rahmen der kontinuierlichen Verbesserung werden zudem Leistungsindikatoren vorgeschlagen, durch die die Effektivität und Effizienz des Schwachstellenmanagements beurteilt werden kann. Die Beurteilung wird in Form von Berichten an die institutsweit dienstverantwortlichen Personen in der Rolle des Schwachstellenmanagers sowie den netzweit dienstverantwortlichen Chief Vulnerability Manager und den *Chief Information Security Officer* (CISO) kommuniziert und durch diese für Anpassungen des Dienstes genutzt. Insofern sind die Anfoderungen `NA21` sowie `FA28` erfüllt.

5.11.2.2 Unerfüllte Anforderungen

Im Konzept wird die Präsentation bzw. Darstellung der Daten (Anforderung `NA12`) nicht weiter konkretisiert, da diese zwar wünschenswert, die Ausarbeitung jedoch keinen Kernaspekt für ein Funktionieren des Schwachstellenmanagements darstellt und ebenfalls bei der konkreten Umsetzung realisiert werden kann.

6 Implementierung

In diesem Kapitel wird die Implementierung der einzelnen technischen Komponenten, deren Kommunikation sowie der Datenhaltung beschrieben. Eine skizzierte Darstellung des Gesamtbildes der Komponenten (inklusive Nutzer) und ihrer Kommunikation untereinander ist in Abbildung 5.3 des vorherigen Kapitels dargestellt.

Die zentrale Komponente wird durch das Steuerungssystem dargestellt, welche automatische Abläufe steuert und Daten der anderen Komponenten zusammenführt. Es stellt außerdem die einzige Zugriffsmöglichkeit für Nutzer über ein von ihr angebotenes Webportal zur Steuerung dar. Detektionssysteme sind so implementiert, dass sie, genau wie die auf den Assets eingesetzten Detektionsagenten, in variabler Anzahl eingesetzt werden können.

> Der Inhalt aller Konfigurationsdateien ist beispielhaft, wodurch insbesondere die URLs der Maschinen und Informationen zur Authentifizierung nicht tatsächlich vorhanden sind.

6.1 Kommunikation der Komponenten

Zum Austausch von Informationen ist zwischen Komponenten eine geeignete Kommunikation untereinander über das Netz unbedingt notwendig. Zu diesem Zweck hat jede technische Komponente eine REST-Schnittstelle, über die verschiedene Funktionen aufgerufen werden können. Die Gemeinsamkeit der APIs aller Komponenten ist die Realisierung über einen Webservice und das HTTPS-Protokoll, durch welche die API über eine pro System bestimmte URL adressierbar ist.

Der Datenaustausch – sowohl als Eingabeparameter einer Funktion, als auch als Ergebnis – findet über Objekte in der *Javascript Object Notation* (JSON) statt. Ein Beispiel eines in JSON dargestellten Datensatzes aus der Tabelle *softwareprodukte* ist in Listing 6.1 verdeutlicht.

Listing 6.1: Beispielrückgabe eines Softwareprodukts über die API

```
1 {
2     "ID": 23,
3     "HERSTELLER": "oracle",
4     "BEZEICHNER": "jre",
5     "VERSION": "1.6.0:update101"
6 }
```

Da jedoch bei der Eingabe von Parametern zusätzlich die Funktion identifiziert werden muss, die auf dem entsprechend angefragten Gerät ausgeführt wird, gibt es ausschließlich

beim **Funktionsaufruf** einen zusätzlichen Schlüssel „FUNCTION“, welcher den Identifikator der aufzurufenden Funktion beschreibt, sowie einen Schlüssel „PARAM“ mit dem Eingabeparameter – ebenfalls ein JSON-Objekt. Diese Schlüssel sind bei Senden eines Ergebnisdatensatzes nicht notwendig.

Würde beispielsweise das Hinzufügen des in Listing 6.1 gezeigten Softwareproduktes in die Schwachstellen-Dokumentation mittels der Funktion „addSoftwareprodukt“ angestrebt werden, würde sich die Darstellung wie in Listing 6.2 ändern.

Listing 6.2: Aufruf der Funktion „addSoftwareprodukt“ der Schwachstellen-Dokumentation mit Parameter

```
1 {
2     "FUNCTION": "addSoftwareprodukt",
3     "PARAM": {
4         "ID": 23,
5         "HERSTELLER": "oracle",
6         "BEZEICHNER": "jre",
7         "VERSION": "1.6.0:update101"
8     }
9 }
```

Die JSON-Objekte werden über die Methode *GET* gesendet und entsprechend zur Übertragung als Parameter in die URL enkodiert, wodurch das aus Listing 6.2 gezeigte Objekt die in Listing 6.3 gezeigte Form erhält und übertragen wird.

Listing 6.3: Zur Übertragung an die REST-Schnittstelle der Schwachstellen-Dokumentation enkodiertes JSON-Objekt aus Listing 6.2

```
1 http://<URL>/<PATH>?FUNCTION=addSoftwareprodukt&PARAM%5BID%5D=23&
    PARAM%5BHERSTELLER%5D=oracle&PARAM%5BBEZEICHNER%5D=jre&PARAM%5
    BVERSION%5D=1.6.0%3Aupdate101
```

Auf Serverseite wird diese Darstellung wieder dekodiert, wodurch es wieder als JSON-Objekt behandelt wird.

Seit der *Java Standard Edition 6* ist ein Webservice in Java einfach durch die im Paket `com.sun.net.httpserver` mitgelieferte API implementierbar. Dazu wird mittels der darin enthaltenen Klasse `HttpServer` ein Webserver gestartet und durch Aufruf der Funktion `createContext(String arg0, HttpHandler arg1)` ein Kontext erstellt; so auch der Kontext „*/api*“ durch `createContext(''/api'', new APIHandler())`, wobei `APIHandler` die Java-Schnittstelle `HttpHandler` implementiert und den Aufruf des Kontextes durch einen Client behandelt. Durch den Aufruf von `setAuthenticator(Authenticator arg0)` ist zudem eine Authentifizierung bei der Nutzung der API notwendig. Diese wird innerhalb der überschriebenen Methode `checkCredentials` der Instanz der Klasse `BasicAuthenticator` behandelt. Der komplette Auszug ist in Listing 6.4 zusammengestellt.

Listing 6.4: Erstellen eines HTTP-Dienstes in Java

```
1  HttpServer fService;
2  //[...]
3  public void start(port)
4  {
5    try
6    {
7      fService = HttpServer.create(new InetSocketAddress(port));
8      HttpContext c = fService.createContext("/api",
9                                             new APIHandler());
10
11     c.setAuthenticator(new BasicAuthenticator("api"){
12
13       @Override
14       public boolean checkCredentials(String user, String pwd) {
15         //[...]
16       }
17     });
18
19     fService.setExecutor(null);
20     fService.start();
21     System.out.println("HTTP Service running on port "+port);
22   }
23   catch (IOException ioe)
24   {
25     ioe.printStackTrace();
26   }
27 }
28
29 class APIHandler implements HttpHandler
30 {
31   // Die Behandlung bei Aufruf des Kontextes /api erfolgt hier
32   public void handle(HttpExchange t)
33   {
34     //[...]
35   }
36 }
```

Zur Behandlung von JSON-Objekten, das heißt der Serialisierung von Java-Klassen und Deserialisierung von JSON-Objekten, gibt es mehrere Softwarebibliotheken für Java. In der Implementierung wird „Gson“ (`https://github.com/google/gson`) genutzt, ein gut gepflegtes und dokumentiertes Open-Source-Projekt.

6.2 Die Konfiguration im Schwachstellenmanagement

Die Konfiguration essentieller Informationen, die im Schwachstellenmanagement im Konzept (vorheriges Kapitel) identifiziert wurden, werden zentral durch eine Datenbank angeboten. Weitere Konfigurationen der einzelnen Komponenten sind teilweise auf Dateien in diesen ausgelagert und werden in den entsprechenden Abschnitten der Komponenten erklärt.

6.2.1 Konfiguration des Datenbankservers

Eine Darstellung des Datenbankschemas zur effizienten Organisation der Daten und Abläufe ist im Anhang 2 zu finden und wird so von einem MySQL Datenbanksystem angeboten. Auf die darin zu sehenden Tabellen und Inhalte wird im Verlauf des Kapitels oft verwiesen. Der Zugriff auf den Server ist mittels „Access Control Lists“ (ACL) auf die Maschinen der Schwachstellen-Dokumentation und der Asset- und Benutzerverwaltung eingeschränkt und zusätzlich durch die vom Datenbanksystem angebotenen Mechanismen zur Authentifizierung und Autorisierung auf Benutzerebene geschützt.

6.2.2 Erläuterung des Datenbankschemas

In der Datenbank sind die bereits im Konzept beschriebenen Informationen der Schwachstellen-Dokumentation, d.h. *Fremdquellen*, *Softwareprodukte*, *Meldungstickets*, *Schwachstellen*, *Schwachstellentickets*, *Detektionsverfahren* sowie *Maßnahmen* enthalten, als auch Informationen aus der Asset- und Benutzerverwaltung (*Assets* und *Nutzer*) mit den im Konzept beschriebenen Attributen. Der Wert des Schwachstellenakzeptanzkriteriums, des Standardwertes für Updateintervalle aus Fremdquellen, des Standardwertes für das Detektionsintervall von Assets, das Intervall zum Zurücksetzen des Wertes der Temporären Dringlichkeit sowie das maximale Alter von Schwachstellentickets, nach dessen Überschreiten sie aus der Schwachstellen-Dokumentation entfernt werden, sind in einer gesonderten Tabelle *vulmgmt_config* dokumentiert. Diese enthält zudem den Wert des Intervalls, in dem die Funktion zum Aufräumen der Schwachstellen-Dokumentation ausgeführt wird und den Zeitstempel der letzten Ausführung.

Weitere Tabellen wie *bewertungsquellen*, *kategorisierungsquellen*, *detektionsquellen*, *institut*, *dienste*, *geraetetypen*, *betriebssystem*, *plattformen*, *compiler* und *rollen* dienen der Standardisierung von Antwortmöglichkeiten. Eine weitere Tabelle, die diese Funktion erfüllt, ist zusätzlich die bereits oben genannte Tabelle *softwareprodukte*, welche jedoch aufgrund ihrer Relevanz und Zentralität im Schwachstellenmanagement speziell als Information in der Schwachstellen-Dokumentation genannt wird.

Das Ziel, das damit verfolgt wird, ist, Anwendern bei Eingaben eine Auswahl an Antwortmöglichkeiten vorgeben zu können, um netzweit eine gleichartige Notation realisieren zu können und das Auftreten verschiedener Bezeichnungen wie beispielsweise „SLES“ und „SUSE Linux Enterprise Server“ für dasselbe Softwareprodukt zu vermeiden.

Eine weitere Gruppierung von Tabellen dient der Zuweisung von Verantwortlichkeiten im Schwachstellenmanagement. Die Tabelle *ist_verantwortlich* beschreibt die Zuweisung der Verantwortlichkeit zur Weiterverarbeitung einer Schwachstelle an einen Nutzer im Schwachstellenmanagement. Die Zuweisung ist im vorherigen Kapitel beschrieben (Abschnitt 5.3.1.2).

Die durchzuführende Teilaktivität ist anhand des Status der jeweils zugewiesenen Schwachstelle erkennbar.

Durch die Tabelle *rollenverteilung* werden Nutzern eine oder mehrere der Rollen im Schwachstellenmanagement zugewiesen, wobei eine Ausnahme von den mit bestimmten Assets verknüpften Rollen bilden: Die Zuweisung eines Detektionsverantwortlichen wird über einen Eintrag in der Tabelle *ist_detektionsverantwortlich*, die Zuweisung eines Behebungsverantwortlichen über einen Eintrag in der Tabelle *ist_behebungsverantwortlich* vorgenommen.

In der Beispielimplementierung des Datenschemas gibt es eine globale Blacklist für die Filterung der aus Fremdquellen importierten Schwachstellen nach Softwareprodukten, realisiert durch Tabelle *blacklist_global* sowie jeweils pro Fremdquelle eine lokale Blacklist, umgesetzt in Form der Tabelle *blacklist_lokal*.

Die restlichen Tabellen *referenzen*, *softwareinventar*, *ist_asset_spezifisch*, *abhaengig_von* und *dienstmap* kompensieren hingegen das Fehlen von Arrays als Datentypen in relationalen Datenbanksystemen.

Eine semantisch besondere Bedeutung haben die beiden Tabellen zur Zuweisung einer Schwachstelle zu den von ihnen betroffenen Softwareprodukten. Da – wie im vorherigen Kapitel in Abschnitt 5.3.1.1 beschrieben – Schwachstellen teilweise durch den Einsatz einer Kombination von Softwareprodukten auftreten (beispielsweise nur in einer Implementierung eines Softwareprodukts für die Linux-Plattform, auf anderen Plattformen hingegen nicht), können diese Beziehungen durch die Tabellen *ist_betroffenes_produkt_kette* sowie *ist_betroffenes_produkt* dargestellt werden.

Die Tabelle *ist_betroffenes_produkt_kette* entspricht einer logischen *Und*-Verknüpfung der damit in Verbindung gebrachten Softwareprodukte. Die Tabelle *ist_betroffenes_produkt* hingegen stellt eine logische *Oder*-Verknüpfung der gebildeten Ketten dar.

Die Datenbank hält abgesehen von der Implementierung eines Detektionsverfahrens und der Konfiguration der einzelnen technischen Komponenten alle Informationen im Schwachstellenmanagement. Tabellen, die Informationen der Asset- und Benutzerverwaltung halten, sind *institut*, *geraetetypen*, *betriebssystem*, *assets*, *softwareinventar*, *ist_behebungsverantwortlich*, *ist_detektionsverantwortlich*, *nutzer*, *dienst*, *dienstmap*, *rollen* und *rollenverteilung* (in der Abbildung in Anhang 2 orange markiert).

Alle weiteren Tabellen werden von der Schwachstellen-Dokumentation verwaltet.

6.3 Die Schwachstellen-Dokumentation

Die Schwachstellen-Dokumentation besteht im Kern aus einer API als Schnittstelle zur Datenbank und Bereitstellung des Funktionsumfangs, sowie einer variablen Anzahl an Importmodulen, welche den Import von Schwachstellen aus Fremdquellen realisieren. Zudem hält sie die Implementierung der Detektionsverfahren, welche in der im vorherigen Abschnitt beschriebenen Datenbank dokumentiert sind.

Eine Illustration des Aufbaus der Schwachstellen-Dokumentation sowie der Zugriff der Komponenten innerhalb ist in Abbildung 6.1 gezeigt.

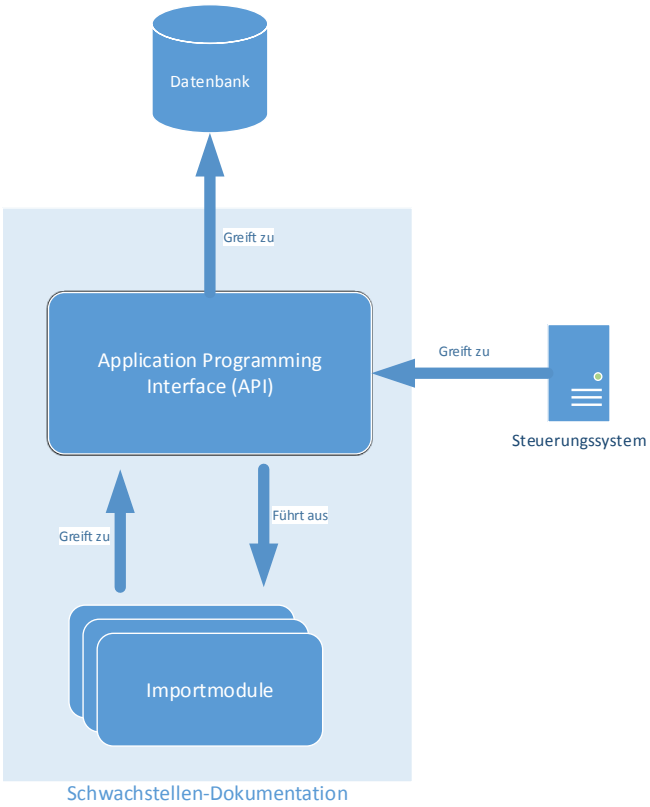


Abbildung 6.1: Aufbau der Schwachstellen-Dokumentation

6.3.1 Verzeichnisstruktur

Die Software der Schwachstellen-Dokumentation wird auf einem Linux-System installiert, wobei als Installationsverzeichnis `/srv/vuldoku/` verwendet wird. Das Verzeichnis selbst gliedert sich in folgende Unterverzeichnisse:

- **bin/**
  In diesem Verzeichnis liegt die API der Schwachstellen-Dokumentation als Java-Klasse *schdoku.jar*

- **conf/**
  Das `conf` Verzeichnis beinhaltet eine Datei *doku.conf*, welche allgemein Parameter für die Konfiguration der Schwachstellen-Dokumentation liefert: Den Port, auf dem die REST-API erreichbar ist sowie Verbindungsinformationen zur MySQL-Datenbank. Der Inhalt von *doku.conf* ist in Listing 6.5 gezeigt.

Listing 6.5: Inhalt der Datei *doku.conf*

```
port            8080

database        jdbc:mysql://vulmgmtdb.lrz.de:3306/
dbname          vulmgmtdatabase
dbuser          vulmgmtuser
dbpassw         #20DFA_34!K0z
```

In einer weiteren Konfigurationsdatei *filter.conf* sind Regeln zum Filtern der Einträge in der Schwachstellen-Dokumentation beschrieben. Da die Daten in einer relationalen Datenbank gehalten werden, sind die Regeln konkrete Aussagen in SQL. Ein beispielhafter Inhalt der Datei ist in Listing 6.6 angegeben.

Listing 6.6: Inhalt der Datei *filter.conf*

```
UPDATE `schwachstellen`
SET `STATUS` = 'reject'
WHERE `EDITOR` = 0
AND `VEROEFFENTLICHT` < NOW() - INTERVAL 5 YEAR
AND `BEWERTUNG` IN ('NA', 'gering');

UPDATE `schwachstellen`
SET `STATUS` = 'reject'
WHERE `EDITOR`= 0
AND `ID` =
   (SELECT a.`SCHWACHSTELLE`
    FROM `ist_betroffenes_produkt` a,
         `ist_betroffenes_produkt_kette` b
    WHERE b.`ID`= a.`SCHWACHSTELLE`
    AND b.`SOFTWAREPRODUKT` IN
       (SELECT `SOFTWAREPRODUKT`
        FROM `blacklist_global`));

//...
```

Im Beispiel wird der Status aller Schwachstellen auf *reject* gesetzt, die entweder älter als 5 Jahre sind und eine Bewertung von *NA* bzw. *gering* haben, oder alle Schwachstellen, von denen betroffene Softwareprodukte in der globalen Blacklist eingetragen sind. Dabei werden nur Schwachstelleneinträge betrachtet, bei denen das Feld `EDITOR` den Wert *0* hat, also von keinem Nutzer manuell eingetragen oder geändert wurden.

Die Datei kann um beliebig viele zusätzliche Filter in Form von SQL Ausdrücken erweitert werden

- **import/**
  Dieses Verzeichnis beinhaltet die Skripte zur Ausführung der Importmodule. Die Da-

teinamen entsprechen dem Identifikator des jeweiligen Eintrags in der Tabelle *fremdquellen* in der Datenbank.

- **res/import/**
  Hier liegen die eigentlichen Implementierungen der Importmodule in jeweils einem eigenen Unterverzeichnis. Das Schema zur Namensgebung der Unterverzeichnisse ist „import$< ID >$“, wobei $< ID >$ der Identifikator der entsprechenden Fremdquelle ist.

- **res/import/tmp/**
  Ein Verzeichnis, das von Importmodulen zur Zwischenspeicherung von Daten genutzt werden kann, z.B. Quelldateien für Schwachstelleninformation.

- **res/detect/**
  In diesem Verzeichnis sind alle ausführbaren Implementierungen der Detektionsverfahren abgelegt. Die Dateien sind nach dem dazugehörigen Identifikator in der Tabelle *schwachstellen* sowie *detektionsverfahren* benannt (z.B. „18_832“).

- **res/detect_src/**
  In diesem Verzeichnis liegen die Quelltextdateien der Implementierung in einem jeweils pro Detektionsverfahren angelegten Unterordner. Die Unterordner haben den gleichen Namen (Identifikator) wie das implementierte Detektionsverfahren.

Ein Beispiel für ein im Verzeichnis `/srv/vuldoku/import` liegendes Skript „1“ ist in Listing 6.7 zu sehen. Es verwendet das Unix-Tool *wget* zum Herunterladen der CVE Quelldatei und speichert sie im Verzeichnis `/src/vuldoku/res/import/tmp` zwischen. Das Programm wartet bis der Download beendet ist (mittels *wait*) und ruft danach die Implementierung des Importmoduls auf.

Listing 6.7: Skript „1“ zum Starten des Imports aus der CVE

```
1 #! /bin/bash
2
3 wget -O "../res/import/tmp/q1.csv" "https://cve.mitre.org/data/
    downloads/allitems.csv" &
4 wait
5
6 java -cp ./res/import/import1/import1.jar quellen.cve.Import "../
    tmp/q1.csv"
```

6.3.2 Die API der Schwachstellen-Dokumentation

Die API der Schwachstellen-Dokumentation ist ihr Kernelement und dient als einzige Schnittstelle auf die Tabellen der Schwachstellen-Dokumentation. Sie stellt dem Steuerungssystem und den Importmodulen eine Reihe an Funktionen zum Zugriff auf die Daten zur Verfügung mit gleichzeitiger Bewahrung der Konsistenz der Daten.

Die API ist zunächst in Form eines Java-Projekts als Teil des Pakets `/srv/vuldoku/bin/schdoku.jar` implementiert und kann von den lokal im Verzeichnis `/srv/vuldoku/res/import` liegenden Importmodulen eingebunden werden. Auf diese „Low Level“ API ist zudem

ein HTTPS Service aufgesetzt, welcher auf dem in der Datei `conf/doku.conf` angegebenen Port lauscht (vgl. Listing 6.5) und dadurch eine REST-Schnittstelle anbietet. Die Funktionen der API können so vom Steuerungssystem aus durch den Aufruf von bestimmten Pfaden in der URL der Schwachstellen-Dokumentation verwendet werden.

Das Steuerungssystem des Schwachstellenmanagements ist die einzige Komponente, die abgesehen von Importmodulen auf die API der Schwachstellen-Dokumentation zugreifen kann. Die Zugriffsbeschränkung wird durch eine ACL in Verbindung mit einer Authentifizierung über eine Abfrage einer Kombination aus Benutzer und Passwort umgesetzt.

Die API der Schwachstellen-Dokumentation ist über die URL `https://vulmgmtdoku.lrz.de/api/` erreichbar. Ihr Funktionsumfang stellt sich wie folgt dar:

Hinzufügen und Auslesen eines Meldungstickets (`addNotification, getNotification`)
Das Hinzufügen eines Meldungstickets wird durch Anlegen eines Datensatzes in der Tabelle *meldungstickets* der Datenbank vorgenommen. Abgesehen von den Feldern `ID` und `ERSTELLT`, welche bei Eintragung automatisch gesetzt werden, sowie das Feld `GESCHLOSSEN`, das bei Erstellung leer bleibt, erwartet die Funktion Eingaben für alle weiteren Felder. Falls keine Angabe für das Feld `KONTAKT` angegeben wird, bleibt es ebenfalls leer.

Das Auslesen eines Meldungstickets gibt alle Felder in Form eines JSON-Objekts zurück und erwartet als Parameter den Identifikator des jeweiligen Meldungstickets.

Hinzufügen einer Schwachstelle (`addVulnerability`)
Beim Hinzufügen von Schwachstellen werden in der Regel Datensätze in mehreren Tabellen der Datenbank eingefügt. Diese Funktion stellt die Möglichkeit der Eintragung einer Schwachstelle anhand grundlegender Daten zur Verfügung. Sie erwartet mindestens eine `BESCHREIBUNG` in Textform und die `ID` des Editors. Die `BEWERTUNG` wird genauso wie `TEMP_DRINGLICHKEIT` zunächst automatisch auf *NA* gesetzt, der `STATUS` auf *check*. Die Funktion trägt zudem das aktuelle Datum der letzten Änderung automatisch ein.

Hinzufügen einer Referenz (`addReference`)
Die Erstellung und Zuweisung von Referenzen wird in der Tabelle *referenzen* dokumentiert. Die Zuweisung wird durch einen Verweis mittels der `ID` auf eine Schwachstelle oder Maßnahme umgesetzt.

Bewerten einer Schwachstelle (`assessVulnerability`)
Diese Funktion erwartet als Parameter zum einen die Faktoren zur Beschreibung der Ausnutzungswahrscheinlichkeit, sowie zum anderen die Faktoren zur Beschreibung der Auswirkung einer Ausnutzung. Die Funktion ist wie in Abschnitt 5.6.1.1 des vorhergehenden Kapitels umgesetzt. Eine Implementierung der Funktionen ist in Listing 6.8 zu sehen.

Falls das Feld `BEWERTUNG` auf *NA* gesetzt ist, bekommt der Schwachstelleneintrag den Status *assess* zugewiesen – andernfalls *categorize*.

Listing 6.8: Implementierung der Berechnung der Kritikalität einer Schwachstelle in Java

```
1   public static Likelihood ssLikelihood(boolean aufwand,
        boolean netz, boolean bekannteFaelle)
2   {
3     if (!aufwand && netz && bekannteFaelle)
4       return Likelihood.kritisch;
5     else if ((!aufwand && netz) || (netz && bekannteFaelle))
6       return Likelihood.hoch;
7     else if ((!aufwand && bekannteFaelle) || netz)
8       return Likelihood.gering;
9     else return Likelihood.gering;
10  }
11
12  public static Impact ssImpact(boolean vertraulichkeit,
        boolean integritaet, boolean verfuegbarkeit)
13  {
14    if (vertraulichkeit && integritaet && verfuegbarkeit)
15      return Impact.kritisch;
16    else if ((vertraulichkeit && integritaet)
17          || (vertraulichkeit && verfuegbarkeit)
18          || (integritaet && verfuegbarkeit))
19      return Impact.hoch;
20    else if (vertraulichkeit
21          || integritaet
22          || verfuegbarkeit)
23      return Impact.mittel;
24    else return Impact.NA;
25  }
26
27  public static SSCriticality ssCriticality(Likelihood l,
        Impact i)
28  {
29   if (i == Impact.kritisch)
30   {
31     if (l == Likelihood.gering)
32       return SSCriticality.hoch;
33     else return SSCriticality.kritisch;
34   }
35   if (i == Impact.hoch)
36   {
37     if (l == Likelihood.gering
38      || l == Likelihood.mittel)
39       return SSCriticality.mittel;
40     else if (l == Likelihood.hoch)
41       return SSCriticality.hoch;
42     else return SSCriticality.kritisch;
```

```
43    }
44    if (i == Impact.mittel)
45    {
46      if (l == Likelihood.gering)
47        return SSCriticality.gering;
48      else if (l == Likelihood.mittel)
49        return SSCriticality.mittel;
50      else return SSCriticality.hoch;
51    }
52    else return SSCriticality.NA;
53  }
```

Die Rückgabetypen `Likelihood`, `Impact` und `SSCriticality` sind Aufzählungen (engl. „enumerations“) und enthalten jeweils die Werte *NA*, *gering*, *mittel* und *hoch*. Der durch die Funktion *ssCriticality* berechnete Wert wird in das Feld `BEWERTUNG` der Tabelle *schwachstellen* eingetragen. Zur Identifikation des betrachteten Datensatzes benötigt die Funktion zudem die `ID` der jeweiligen Schwachstelle.

Hinzufügen einer Kette von einer Schwachstelle betroffenen Softwareprodukten (`addAffectedSoftware`)
Diese Funktion entspricht dem Hinzufügen eines oder mehrerer miteinander verknüpfter Softwareprodukte in die Tabelle *ist_betroffenes_produkt_kette*, und genau eines Eintrags mit Zeiger auf das Anfangselement der Kette in der Tabelle *ist_betroffenes_produkt*. Das heißt, dass immer nur genau eine Kombination von Softwareprodukten hinzugefügt wird, deren Einsatz zusammen zum Vorkommen der Schwachstelle führt. Insofern benötigt die Funktion als Parameter eine Liste an Softwareprodukten sowie den Identifikator der dazugehörigen Schwachstelle.

Bevor die eingegebenen Softwareprodukte in die Tabelle *softwareprodukte* eingetragen werden, wird zunächst überprüft, ob diese bereits existieren. Ist das der Fall, so wird die `ID` des existierenden Eintrages verwendet, andernfalls ein neuer Eintrag erstellt und die `ID` von diesem in die Liste *ist_betroffenes_produkt_kette* eingetragen.

Der Status einer Schwachstelle bei Eintragen von Softwareprodukten wird (falls die Schwachstelle ebenfalls bereits bewertet ist) automatisch auf *detect* gesetzt.

Auslesen einer Schwachstelle (`getVulnerability`)
Anders als die Dokumentation einer Schwachstelle, die in mehrere Funktionen aufgeteilt ist, werden die Informationen einer Schwachstelle komplett mit gesamtem Inhalt ausgelesen. Das heißt, alle dazugehörigen Informationen aus der Tabelle *schwachstellen*, *referenzen*, *ist_betroffenes_produkt*, *ist_betroffenes_produkt_kette* und *softwareprodukte*. Die Rückgabe ist beispielhaft in Listing 6.9 verdeutlicht.

Listing 6.9: Schema der Rückgabe bei Abfrage einer Schwachstelle im JSON-Format

```
{
    "ID": 0,
    "BESCHREIBUNG": "",
    "BEWERTUNG": "",
    "TEMP_DRINGLICHKEIT": ""
    "FREMDQUELLE": "","
    "FREMDQUELLENID": "",
    "EDITOR": 0,
    "STATUS": "",
    "GELOEST": "",
    "GEAENDERT": "",
    "REFERENZEN": [
      {"ID": 0,
       "QUELLE": "",
       "QUELLENID": "",
       "SCHWACHSTELLE": 0,
       "MASSNAHME": 0,
       "EDITOR": 0},
       ...
    ]
    "SOFTWAREPRODUKTE": [
      {"ID": 0,
       "HERSTELLER": "",
       "BEZEICHNER": "",
       "VERSION": ""},
      ...
    ]
}
```

Erstellung eines Detektionsverfahrens (`addDetectionMechanism`)
Die Funktion zur Erstellung eines Detektionsverfahrens erwartet mit Ausnahme des Feldes `ID` und `FREIGEGEBEN`, Eingaben für alle Felder der Tabelle *detektionsverfahren*, zudem Angaben zu den Feldern der Tabellen *compiler*, *abhaengig_von* sowie *plattformen*, die jeweils mehrere Einträge mit Referenz auf dasselbe Detektionsverfahren beinhalten können.

Des Weiteren ist ein Upload der Implementierung und Quelldateien über HTTPS Teil der Funktion. In Folge des Uploads werden diese Dateien wie oben beschrieben in die Verzeichnisstruktur der Schwachstellen-Dokumentation einsortiert.

Abrufen von Detektionsverfahren (`getDetectionMechanism`)
Die Dokumentation wird ähnlich wie am Beispiel des Abrufs einer Schwachstelle in JSON übertragen, die Implementierungen werden als gepackte der Datei *update.7z* gesendet. Beides geschieht über HTTPS.

Freigeben von Detektionsverfahren (`enableDetectionMechanism`)
Die Funktion zur Freigabe eines Detektionsverfahren benötigt als Eingabe lediglich den Identifikator des jeweiligen Eintrags in der Tabelle *detektionsverfahren*, in welchem der Wert des Feldes `FREIGEGEBEN` auf *true* gesetzt wird.

Erstellung von Schwachstellentickets (`addVulerabilityTicket`)
Zur Erstellung eines Schwachstellentickets benötigt die API als Parameter nur das jeweilige Asset, seine Kritikalität und die Schwachstelle, für die ein Schwachstellenticket erstellt wird. Beides wird in die Tabelle *schwachstellentickets* in das entsprechende Feld eingetragen. Das Feld `ERSTELLT` enthält einen Zeitstempel über den Zeitpunkt der Erstellung, das Feld `GESCHLOSSEN` bleibt leer. Das Feld `BEHEBUNGSNOTWENDIGKEIT` wird automatisch bei Erstellung durch die Schwachstellen-Dokumentation berechnet. Die dafür vorgesehene Implementierung der Funktion ist in Listing 6.10 gezeigt. Die Kritikalität der jeweiligen Schwachstelle wird aus der Tabelle *schwachstellen* geholt, die Kritikalität des Assets aus *assets*. Das Feld `ESCALL` (das Eskalationslevel) ist bei Erstellung auf „*0*“ gesetzt.

Listing 6.10: Implementierung der Funktion zur Berechnung der Behebungsnotwendigkeit in Java

```
1  public static Behebungsnotwendigkeit behebungsnotwendigkeit(
       SSCriticality sscrit, AssetCriticality acrit)
2  {
3    if (acrit == AssetCriticality.kritisch)
4    {
5      if (sscrit == SSCriticality.gering)
6        return Behebungsnotwendigkeit.hoch;
7      else return Behebungsnotwendigkeit.kritisch;
8    }
9    else if (acrit == AssetCriticality.hoch)
10   {
11     if (sscrit == SSCriticality.gering
12      || sscrit == SSCriticality.mittel)
13       return Behebungsnotwendigkeit.mittel;
14     else if (sscrit == SSCriticality.hoch)
15       return Behebungsnotwendigkeit.hoch;
16     else return Behebungsnotwendigkeit.kritisch;
17   }
18   else if (acrit == AssetCriticality.mittel)
19   {
20     if (sscrit == SSCriticality.gering)
21       return Behebungsnotwendigkeit.gering;
22     else if (sscrit == SSCriticality.mittel)
23       return Behebungsnotwendigkeit.mittel;
24     else return Behebungsnotwendigkeit.hoch;
25   }
26   else if (acrit == AssetCriticality.gering)
```

```
27      {
28        if (sscrit == SSCriticality.gering)
29          return Behebungsnotwendigkeit.gering;
30        else return Behebungsnotwendigkeit.mittel;
31      }
32      else //acrit == NA
33      {
34        return Behebungsnotwendigkeit.valueOf(sscrit.name());
35      }
36    }
```

Die Klassen `Behebungsnotwendigkeit`, `AssetCriticality` und `SSCriticality` sind Auflistungen mit den Werten *NA*, *gering*, *mittel*, *hoch* und *kritisch*.

Setzen des Eskalationslevels eines Schwachstellentickets (`setVulTicketEscalation`)
Das Eskalationslevel wird durch Setzen des Feldes `ESKALL` in der Tabelle *schwachstellentickets* festgelegt.

Schließen eines Schwachstellentickets (`closeVulTicket`)
Die Funktion erwartet als Parameter den Identifikator der Maßnahme, die zum Abschluss des Tickets angewendet wurde. Dieser wird in das Feld `MASSNAHME` der Tabelle *schwachstellentickets* eingetragen, sowie der Zeitstempel des Feldes `GESHCLOSSEN` auf die aktuelle Systemzeit gesetzt.

Erstellung einer Maßnahme (`addRemedy`)
Zur Erstellung einer Maßnahme wird die `ID` der dazugehörigen Schwachstelle, eine Beschreibung der Maßnahme, der Editor sowie optional ein Typ der Maßnahme erwartet. Damit wird ein neuer Eintrag in der Tabelle *massnahmen* erstellt. Für die Erstellung von Referenzen für Maßnahmen wird die oben beschriebene Funktion zum Hinzufügen einer neuen Referenz verwendet.

Setzen des Schwachstellenakzeptanzkriteriums (`setVulnerabilityAcceptance`)
Der Wert des Schwachstellenkriteriums wird durch Setzen des Feldes `SCHWACHSTELLENAKZEPTANZKRITERIUM` des einzigen Datensatzes in der Tabelle *vulmgmt_config* festgelegt. Ein Beispiel eines Aufrufs der Funktion ist in Listing 6.11 gezeigt.

Listing 6.11: Beispieleingabe zum Setzen des Schwachstellenakzeptanzkriteriums

```
1 {
2     "FUNCTION": "setVulerabilityAcceptance",
3     "PARAM": {
4         "SCHWACHSTELLENAKZEPTANZKRITERIUM": "gering"
5     }
6 }
```

Setzen des Standardwertes von Updateintervallen (Fremdquellen) (`setVulSourceUpdateInterval`)
Der Standardwert des Intervalls der automatischen Aktualisierung von Fremdquellen wird im Feld `STD_UPDATEINTERVALL` in der Tabelle *vulmgmt_config* festgelegt.

Setzen des Standardwertes von Detektionsintervallen (Assets) (`setAssetDetectionInterval`)
Der Standardwert des Intervalls der automatischen Detektion von Assets wird über das Feld `STD_DETEKTIONSINTERVALL` der Tabelle *vulmgmt_config* beschrieben. Da die Tabelle Teil der Schwachstellen-Dokumentation und nicht Teil der Asset- und Benutzerverwaltung ist, ist auch die Funktion Teil der Schwachstellen-Dokumentation.

Setzen und Holen des Intervalls zum Zurücksetzen der Temporären Dringlichkeit (`setTempUrgencyInterval, getTempUrgencyInterval`)
Als Eingabe erwartet die Funktion einen Integer-Wert, der die Anzahl der Tage festlegt, nach deren Ablauf die Temporäre Dringlichkeit automatisch zurückgesetzt wird. Der Wert wird in das Feld `TEMP_DRINGLICHKEIT_RESETINTERVALL` der Tabelle *vulmgmt_config* eingetragen.

Anpassung globaler und lokaler Blacklisten (`addGlobalBlacklistEntry, addLocalBlacklistEntry, deleteGlobalBlacklistEntry, deleteLocalBlacklistEntry`)
Die globale Blackliste wird durch Erstellen von Datensätzen in der Tabelle *blacklist_global* umgesetzt und bildet prinzipiell lediglich einen Zeiger (mittels der Eingabe einer `ID` eines Softwareprodukts) auf die Tabelle *softwareprodukte*. Das gleiche Vorgehen wird ebenso beim Setzen einer lokalen Blacklist in der Tabelle *blacklist_lokal* durchgeführt. Einziger weiterer Parameter ist ein Zeiger auf einen Eintrag der Tabelle *fremdquellen*.

Anlegen einer Fremdquelle (`addVulnerabilitySource`)
Das Anlegen einer Fremdquelle wird über die Erstellung eines Eintrags in der Tabelle *fremdquellen* erreicht. Prinzipiell sind alle Felder außer der `ID` bei Erstellung anzugeben. Eine weiterführende Dokumentation von Quellen der Klassifikation (Tabellen *bewertungsquellen* und *kategorisierungsquellen*) sowie für Detektionsverfahren (Tabelle *detektionsquellen*) sind optional. Andererseits können diese auch unabhängig von einem Eintrag in der Tabelle *fremdquellen* angegeben werden, sind dann jedoch nicht nicht Teil einer Fremdquelle.

Setzen und Holen des letzten Updatezeitpunktes einer Fremdquelle (`setVulSourceUpdateTS, getVulSourceUpdateTS`)
Als Eingabe erfordert diese Funktion lediglich den Identifikator einer Fremdquelle. Resultat ist das Setzen des Feldes `ZPLIMPORT` auf den aktuellen Zeitpunkt.

Abfragen der Fremdquellendokumentation (`getVulnerabilitySource`)
Die Informationen der Fremdquellen können insbesondere zum Zweck der Konfiguration der Importmodule über die API angerufen werden. Zurückgeliefert wird eine Antwort in JSON-Format (vgl. Beispiel in Listing 6.12)

Listing 6.12: Beispiel der Rückgabe bei Abfrage einer Fremdquelle

```
{
    "ID": 0,
    "QUELLE": "https://cve.mitre.org/data/downloads/allitems.
        txt",
    "BENUTZER": "",
    "PASSWORT": "",
    "BEWERTUNGSQUELLE": "https://web.nvd.nist.gov/view/vuln/
        detail?vulnId=",
    "BBENUTZER": "",
    "BPASSWORT": "",
    "KATEGORIEQUELLE": "https://web.nvd.nist.gov/view/vuln/
        detail?vulnId=",
    "KBENUTZER": "",
    "KPASSWORT": "",
    "DETEKTIONQUELLE": "",
    "DBENUTZER": "",
    "DPASSWORT": ""
}
```

Priorisierung von Fremdquellen (`exchangeSourcePriority`)
Wie in Abschnitt 5.3.1.2 beschrieben, ist die Priorisierung von Fremdquellen eine Möglichkeit zur Selektion von Informationen aus verschiedenen Quellen für dieselbe Schwachstelle. Die Priorität einer Fremdquelle wird durch ihren Identifikator in der Tabelle *fremdquellen* festgelegt. Je kleiner der Wert eines Identifikators ist, desto höher ist die Priorität einer Fremdquelle. Das Setzen erfolgt durch den Austausch der Werte zweier Einträge untereinander. Daher benötigt die Funktion als Parameter genau zwei IDs von Einträgen aus der Tabelle *fremdquellen*. Resultat ist der Austausch der IDs für die jeweiligen Einträge sowie eine Aktualisierung aller Daten, die auf die jeweiligen Einträge gezeigt haben (Datensätze aus den Tabellen *blacklist_lokal* und *schwachstellen*).

Zuweisung von Verantwortlichkeiten (`setNotificationResponsibility`, `setVulResponsibility`)
Die Darstellung einer Zuweisung in der Datenbank für Schwachstellen und Meldungstickets wird durch die Schwachstellen-Dokumentation übernommen. Die Zuweisung eines Meldungstickets geschieht durch das Eintragen der `ID` eines Nutzers in das Feld `BEARBEITER` des entsprechenden Eintrags des Meldungstickets.

Die Verantwortung für die Bearbeitung von Schwachstellen wird hingegen über eine Verknüpfung eines Schwachstelleneintrages mit einem Nutzer über die jeweilige `ID` in der Tabelle *ist_verantwortlich* zugewiesen. So gibt es hier die Möglichkeit der Zuweisung einer Schwachstelle an mehrere Personen genauso wie mehrere Schwachstellen an genau eine Person.

Zu beachten ist, dass die Logik der Zuweisung nicht in der Schwachstellen-Dokumentation liegt, sondern als verbindende Komponente zwischen ihr und der Asset- und

Benutzerverwaltung im Steuerungssystem liegt.

Hinzufügen und Abfragen von Softwareprodukten (`addSoftwareproduct, getSoftwareproduct`)
Softwareprodukte werden durch Erstellung eines Eintrags in der Tabelle *softwareprodukte* hinzugefügt, wobei die Funktion als Parameter den Hersteller, die Produktbezeichnung sowie die Version erwartet. Die Abfrage eines Softwareproduktes erwartet dieselben Parameter und liefert bei einem Treffer die komplette Beschreibung inklusive `ID` zurück (vgl. Beispiel in Listing 6.1).

Filtern von Schwachstellen (`filterVulnerabilities`)
Die Funktion erwartet keine weiteren Parameter. Bei Aufruf liest sie die Befehle zum Filtern aus der Datei *filter.conf* (vgl. Abschnitt 6.3.1, Listing 6.6) aus und führt diese aus. Vor der Ausführung überprüft sie jedoch, ob sich der Effekt lediglich auf das Feld `STATUS` der Tabelle *schwachstellen* beschränkt, um möglichen Schaden bei fehlerhaften Befehlen zu verringern. Ist dies nicht der Fall, wird der entsprechende SQL-Ausdruck ignoriert.

Import aus einer bestimmten Quelle (`import`)
Die Funktion zum Import aus einer bestimmten Quelle entspricht lediglich dem Ausführen des quellenspezifischen Skripts, das im Verzeichnis `import/` existieren muss. Die Funktion erwartet als Eingabeparameter die `ID` der Fremdquelle, welche gleichzeitig der eindeutige Name des auszuführenden Skripts ist. Nach jedem Durchlauf wird die Funktion `filterVulnerabilities` zum Filtern der Schwachstellen in der Schwachstellen-Dokumentation ausgeführt.

Setzen der Temporären Dringlichkeit (`setTemporaryUrgency`)
Die Funktion zum Setzen der Temporären Dringlichkeit benötigt als Eingabeparameter den Identifikator der Schwachstelle in der Tabelle *schwachstellen*, für den der Wert festgelegt werden soll, sowie den Wert (NA, gering, mittel, hoch, kritisch), auf den sie gesetzt werden soll. Bei Ausführung wird außerdem der dazugehörige Zeitstempel `TEMP_DRINGLICHKEITTS` gesetzt, um das Datum des automatischen Zurücksetzens der Temporären Dringlichkeit bestimmen zu können.

Aufräumen der Datenbank (`cleanVulnerabilityDoc`)
Die Funktion erwartet keine weiteren Parameter und hat die Aufgabe, Schwachstellen mit Wert *reject* des Feldes `STATUS` zu entfernen. Dazu gehören auch darauf referenzierende Datensätze in den Tabellen *referenzen*, *ist_betroffenes_produkt_kette*, *ist_betroffenes_produkt*, *ist_verantwortlich* und *ist_asset_spezifisch*.

Vor dem Entfernen ruft sie die Funktion zum Filtern von Schwachstellen `filterVulnerabilities` auf.

In einem Durchlauf wird zudem die Temporäre Dringlichkeit von Schwachstellen zurückgesetzt, falls diese ungleich *NA* sind und das aktuelle Datum größer oder gleich dem berechneten Datum aus `TEMP_DRINGLICHKEIT`+`TEMP_DRINGLICHKEIT_RESETINTERVALL` aus den Tabellen *schwachstellen* bzw. *vulmgmt_config* ist.

Des Weiteren werden alte Schwachstellentickets aus der Tabelle *schwachstellentickets* gelöscht, falls ihr Alter in Tagen größer als der Wert `SCHWTICKET_MAX_INTERVALL` in der Tabelle *vulmgmt_config* ist.

Nach Durchführung wird das Datum `LAST_CLEANUP` zur Bestimmung des letzten Aufräumens in Tabelle *vulmgmt_config* auf die aktuelle Systemzeit gesetzt.

Abbildung einer Schwachstellenbewertung auf das Bewertungssystem des Hochschulnetzes (`mapCriticality`)
Jede Abbildung eines Bewertungssystems aus einer Fremdquelle (beispielsweise CVSS oder das *Microsoft Security Bulletin Ranking*) auf das Bewertungssystem des Schwachstellenmanagements im Hochschulnetz muss als Implementierung in der Schwachstellen-Dokumentation vorliegen. Andernfalls ist ein Bewertungssystem nicht abbildbar. Alle Abbildungen befinden sich in einer Java-Klasse *BewertungsAbbildung.java* (siehe Abbildung 6.13) als Funktionen, welche von der API aufgerufen werden. Die Funktion zur Abbildung in der API erwartet als Parameter einen Identifikator der Abbildung und der hierfür notwendigen Parameter.

Listing 6.13: Auszug aus der Datei *BerwertungsAbbildung.java mit Beispiel der Abbildung des CVSS BaseScores*

```
1
2  //SSCriticality = ENUM (NA, gering, mittel, hoch);
3
4   public SSCriticality mapCriticality(String identifier,
       Object[] params)
5   {
6     if (identifier.equals("cvssBase"))
7     {
8       return mapCVSSBaseScore((float) params[0]);
9     }
10    //...
11  }
12
13  public static SSCriticality mapCVSSBaseScore(float baseScore
       )
14  {
15    if (baseScore >= 0.0 && baseScore <= 3.9)
16      return SSCriticality.gering;
17    else if (baseScore >= 4.0 && baseScore <= 6.9)
18      return SSCriticality.mittel;
19    else if (baseScore >= 7.0 && baseScore <= 8.9)
20      return SSCriticality.hoch;
21    else if (baseScore >= 9.0 && baseScore <= 10.0)
22      return SSCriticality.kritisch;
23    else return SSCriticality.NA;
24  }
25
26  //...
```

Die API ruft schließlich die in der Datei *BewertungsAbbildung.java* befindliche Funktion `callAbbildung` auf, welche für jede Abbildung (mittels Hinzufügen einer wie im Listing gezeigten `if`-Abfrage) angepasst werden muss.

Suchfunktionen der Schwachstellen-Dokumentation
Die Schwachstellen-Dokumentation bietet über die API Funktionen zur Suche verschiedener Informationen an. Insbesondere Meldungstickets, Schwachstellen, Detektionsverfahren, Schwachstellentickets und Fremdquellen können anhand ihrer ID oder der jeweiligen Attribute gesucht werden.

Neben den oben genannten Funktionen bietet die Schwachstellen-Dokumentation allgemein das Hinzufügen, Entfernen von Einträgen in den Tabellen und gezielte Änderungen der Werte der Felder an.

6.3.3 Das Importmodul

Ein Importmodul besteht im Allgemeinen aus einer oder mehreren Komponenten zum **Abruf** von Schwachstelleninformationen sowie mindestens einem **Parser** zur Extraktion der Informationen aus den Quellen und realisiert den Import aus einer Fremdquelle. Importmodule können allgemein abhängig vom jeweils implementierenden Quellenverantwortlichen in unterschiedlichen Programmiersprachen implementiert sein, da die REST-API unabhängig von verwendeten Plattform und Programmiersprache genutzt werden kann. Da die API jedoch zusätzlich als Java-Bibliothek vorliegt, ist die Verwendung von Java und eine Einbindung der API als effizienter anzusehen, da Kommunikation über den HTTP-Dienst nicht notwendig ist.

Die Konfiguration einer Fremdquelle ist in den Tabellen *fremdquellen*, *bewertungsquellen*, *kategorisierungsquellen* und *detektionsquellen* hinterlegt. Zur Wiederverwendbarkeit und Vermeidung von redundanten Einträgen sind diese jeweils getrennt und nicht in einer einzigen Tabelle zusammengefasst. Beispielsweise kann so, falls notwendig, dieselbe Konfiguration für eine Kategorisierungsquelle in mehreren Fremdquellen genutzt werden.

Die Tabelle *fremdquellen* beschreibt das gesamte Objekt einer Fremdquelle mit insbesondere einer tatsächlich adressierbaren Referenz (z.B. eine URL) auf die Quelle zur Identifikation von Schwachstellen. Dazu, falls notwendig, können Werte zur Authentifizierung – Benutzername und Passwort – angegeben werden. Da je nach Quelle möglicherweise weitere Informationen für die Authentifizierung notwendig sind, muss berücksichtigt werden, dass die Tabelle um weitere Felder erweitert werden könnte. Alternativ hat jedes Importmodul je nach Bedarf zusätzlich eine weitere eigene Konfigurationsdatei in ihrem Projektverzeichnis, welche jedoch nicht zur Dokumentation von Quellen genutzt wird, da sie insofern nicht für andere Importmodule nutzbar ist.

Die Tabelle *fremdquellen* hält außerdem Referenzen auf jeweils einen Eintrag in der Tabelle *bewertungsquellen*, zur Erfassung der Konfiguration einer Quelle zur Bewertung von Schwachstellen, *kategorisierungsquellen*, zur Dokumentation einer Quelle zur Kategorisierung, sowie auf *detektionsquellen* zur Konfiguration von Quellen für Detektionsverfahren. Die Referenz auf eine der genannten Quellen ist optional und lediglich die Angabe einer Quelle zur Identifikation von Schwachstellen ist erforderlich.

In der Tabelle *fremdquellen* ist zudem ein Parameter zur Beschreibung des Intervalls in Tagen anzugeben (Feld: `UPDATEINTERVALL`), in der die Informationen aus der jeweiligen Fremdquelle automatisch abgerufen werden. Andernfalls wird der konfigurierbare Standardwert dafür übernommen, welcher in der Tabelle *vulmgmt_config* liegt. Durch Wert `UPDATEINTERVALL` – in Kombination mit dem ebenfalls hinterlegten Zeitstempel der letzten Durchführung eines automatischen Imports (Feld: `ZPLIMPORT`) – kann durch den Scheduler im Steuerungssystem der Zeitpunkt der nächsten Durchführung berechnet werden.

Alle der in den folgenden Abschnitten beschriebenen Operationen auf der Datenbank werden über die API der Schwachstellen-Dokumentation abgewickelt.

6.3.3.1 Identifikation

In der Regel ist es notwendig, für jeweils eine bestimmte Quelle ein dazu geeignetes Importmodul zu implementieren und die Daten der jeweiligen Fremdquelle derart abzubilden, dass sie gemäß des im Hochschulnetz genutzten Schemas zur Beschreibung von Schwachstellen in die Datenbank eingetragen werden können. Insofern haben Importmodule eine beliebige, quellenabhängige Eingabe, die derart verarbeitet wird, dass damit mindestens ein Eintrag mit folgenden ausgefüllten Feldern erstellt wird:

In der Tabelle *schwachstellen* muss mindestens das Feld `BESCHREIBUNG` gesetzt werden, damit eine Fremdquelle einen Sinn erfüllt (die automatische Dokumentation von Schwachstellen). Es enthält die allgemeine Beschreibung der Schwachstelle in Textform. Gleiches gilt für das Feld `FREMDQUELLE`, welches auf den Identifikator des entsprechenden Konfigurationseintrags in der Tabelle *fremdquellen* zeigt. Das Feld `EDITOR` wird beim automatischen Import unabhängig von der jeweilig genutzten Fremdquelle auf einen fiktiven Nutzer mit ID=0 gesetzt, welcher die Automatik selber beschreibt. Das Feld `FREMDQUELLENID` ist abhängig davon, ob die für den Import genutzte Fremdquelle einen eigenen Identifikator für die Schwachstellen hat. Ist dieser existent und auslesbar, wird er eingetragen, andernfalls bleibt das Feld leer. Dem Feld `TEMP_DRINGLICHKEIT` wird bei Erstellung eines Eintrags der Standardwert „NA“ zugewiesen.

Zudem werden, falls vorhanden, existierende Referenzen angelegt. Eine Referenz besteht nach Vorbild der in der CVE beschriebenen Referenzen aus einer `QUELLE` sowie einer `QUELLENID`, welche beide in Textform anzugeben sind. In das Feld `EDITOR` wird automatisch der Wert „0“ – der ID des Nutzers „Automatik“ – und in das Feld `SCHWACHSTELLE` wird die ID des betreffenden Eintrags in der Tabelle *schwachstellen* eingetragen.

6.3.3.2 Bewertung

Falls die Fremdquelle zudem eine Möglichkeit zur Bewertung der Schwachstelle bietet, wird das Feld `BEWERTUNG` in der Tabelle *schwachstellen* ebenfalls ausgefüllt. Dies geschieht jedoch über eine separate Funktion der API der Schwachstellen-Dokumentation, welche nicht direkt die Bewertung aus der Fremdquelle an sich, sondern eine Abbildung davon auf einen Wert des Bewertungssystems im Hochschulnetz einträgt (siehe folgender Abschnitt).

6.3.3.3 Kategorisierung

In die automatische Kategorisierung einer Schwachstelle sind die Tabellen *ist_betroffenes_produkt_kette* und *ist_betroffenes_produkt* sowie *softwareprodukte* involviert. Mit diesen Tabellen können boolesche Aussagen in disjunktiver Normalform über das Auftreten einer Schwachstelle in Softwareprodukten getroffen werden. So wird das Auftreten der Schwachstelle in Abhängigkeit von der Nutzung mehrerer Kombinationen von Softwareprodukten (bestehend aus einem oder mehreren Elementen) beschrieben. Zu diesem Zweck stellt die Tabelle *ist_betroffenes_produkt_kette* eine einfach-verkettete Liste dar, in der ein Verweis in einem Eintrag auf einen anderen derselben Tabelle als logische *Und*-Verknüpfung interpretiert wird. Diese Listen werden in der Tabelle *ist_betroffenes_produkt* referenziert und stellen eine logische *Oder*-Verknüpfung dar.

Bevor neue Einträge in der Tabelle *softwareprodukte* angelegt werden, wird über die API zunächst gesucht, ob das Softwareprodukt bereits existiert und beschrieben ist. Falls es existiert, wird der Identifikator des existierenden Eintrages zur Erstellung eines neuen Datensatzes in der Tabelle *ist_betroffenes_produkt_kette* verwendet. Andernfalls wird ein neuer Eintrag in der Tabelle *softwareprodukte* erstellt und auf diesen verwiesen.

Ab diesem Punkt der Verarbeitung findet die Filterung irrelevanter Schwachstellen anhand betroffener Softwareprodukte statt. Ein Importmodul muss abgleichen, ob das von einer Schwachstelle betroffene Softwareprodukt in der globalen Blacklist bzw. der zur Fremdquelle dazugehörigen lokalen Blacklist enthalten ist. Falls dies zutrifft, wird das Feld *STATUS* der Schwachstelle auf *reject* gesetzt, wodurch die komplette Schwachstellenbeschreibung bei der nächsten Ausführung der Aufräumfunktion der Schwachstellen-Dokumentation aus der Datenbank entfernt wird.

6.4 Die Asset- und Benutzerverwaltung

Die Asset- und Benutzerverwaltung besteht ähnlich wie die Schwachstellen-Dokumentation aus einer API mit Funktionen zum Anlegen von Assets und Nutzern in Rollen sowie zur Abfrage dieser Informationen. Auch sie ist als Java-Anwendung umgesetzt und nutzt JDBC zum Zugriff auf die MySQL-Datenbank.

6.4.1 Verzeichnisstruktur

Auch die Asset- und Benutzerverwaltung ist auf einem Linux-System eingerichtet und befindet sich im Installationsverzeichnis `/srv/vulanman/`. Die Unterverzeichnisse sind wie folgt organisiert:

- **bin/**
  In diesem Verzeichnis liegt die Implementierung des Datenbankzugriffs sowie der API.
- **conf/**
  Im `conf/` Verzeichnis liegt die Konfigurationsdatei *anman.conf*.

Der Aufbau und Inhalt der Konfigurationsdatei *anman.conf* ist in diesem Fall vom Inhalt her identisch mit der Konfigurationdatei der Schwachstellen-Dokumentation (*doku.conf*, Vergleiche Listing 6.5), da sie zum einen die gleichen Konfigurationsparameter benötigen und sich in diesem Fall die Datenbank teilen. Außerdem wird der Webservice zur Bereitstellung der API ebenfalls über Port 8080 betrieben.

6.4.2 Die API der Asset- und Benutzerverwaltung

Die API der Asset- und Benutzerverwaltung ist als REST-Schnittstelle auf HTTPS-Basis realisiert, wird also durch einen von ihr erbrachten Webservice angeboten (vgl. Abschnitt 6.1).

Das Steuerungssystem ist die einzige Komponente mit direktem Zugriff auf die API der Asset- und Benutzerverwaltung. Wie bei der Schwachstellen-Dokumentation wird die Zugriffsbeschränkung mittels ACL und Authentifizierung realisiert.

Die API ist unter der URL `https://vulmgmtanman.lrz.de/api/` erreichbar; Funktionen, die durch sie bereitgestellt werden, umfassen:

Anlegen eines Assets (`addAsset`)
Das Anlegen eines Assets erwartet alle Informationen (abgesehen von der `ID`) aus den Tabellen *assets*, *institut*, *geraetetypen* und *betriebssystem*. Die Parameter werden in der URL der REST-Schnittstelle übergeben.

Anpassen des Softwareinventars eines Assets (`setAssetSoftwareList`)
Das Softwareinventar eines Assets – in der Datenbank als Verknüpfung von Softwareprodukten und Assets dargestellt – wird über die Tabelle *softwareinventar* umgesetzt. Als Eingabeparameter erwartet die Funktion eine `ID` des Assets, für welches das Softwareinventar aktualisiert wird, sowie eine `ID` des jeweiligen Softwareprodukts, verweisend auf Einträge der Tabelle *softwareprodukte* in der Schwachstellen-Dokumentation.

Anlegen eines Nutzers (`addUser`)
Ein Nutzer wird grundlegend durch die Felder der Tabelle *nutzer* dargestellt. Angaben zu `NAME`, `KONTAKT` und `INSTITUT` sind unbedingt notwendig. Ebenfalls notwendig ist die Angabe mindestens einer Rolle über die entsprechende `ID`. Falls die `ID` der Rolle eines Detektionsverantwortlichen oder Behebungsverantwortlichen angegeben wird, muss zudem die `ID` des entsprechenden Assets mit übergeben werden. Optionale Angaben betreffen die `ID`s der Dienste, für die der Nutzer verantwortlich ist.

Rollen werden über die Tabelle *rollenverteilung* zugewiesen, Dienste über *dienstmap*. Die Verantwortlichkeit für die Detektion eines Nutzers für ein Asset wird über die `ID`s in *ist_detektionsverantwortlich* eingetragen, die Verantwortlichkeit für die Behebung in *ist_behebungsverantwortlich*. Beim Anlegen eines Nutzers wird zudem darauf geachtet, dass es genau einen Hauptverantwortlichen für die Detektion und Behebung gibt (Feld `HAUPTVERANTWORTLICH`). Wird bei bereits existierendem Hauptverantwortlichen die Hauptverantwortlichkeit bei Anlegen eines neuen Nutzers gesetzt, wird das Attribut bei bisherigem Hauptverantwortlichen in der Tabelle auf *false* gesetzt.

Abrufen eines Assets (`getAsset`)
Die Funktion erwartet als Parameter die `ID` des spezifischen Assets. Zurückgeliefert werden alle Informationen, inklusive verantwortliche Nutzer im JSON-Format. Ein Beispiel ist in Listing 6.14 dargestellt.

Listing 6.14: Beispiel der Rückgabe bei Abfrage einer Fremdquelle

```
{
    "ID": 38,
    "NETZID": "192.168.99.3",
    "BESCHREIBUNG": "Mitarbeiter-PC von Hermann Hesse",
    "KRITKIALITAET": "gering",
    "INSTITUT": {
        "ID": 1,
        "BEZEICHNUNG": "Leibniz-Rechenzentrum"
    },
    "TYP": {
        "ID": 9,
        "BEZEICHNUNG": "Arbeitsplatz-PC"
    },
    "BETRIEBSSYSTEM": {
        "ID": 5,
        "BEZEICHNUNG": "Linux"
    },
    "VERANTWORTLICH": {
        "ID": 3,
        "NAME": "Hermann Hesse",
        "KONTAKT": "hhesse@domain.xy",
        "INSTITUT": {
            "ID": 1,
            "BEZEICHNUNG": "Leibniz-Rechenzentrum"
        },
        "DIENST": [
            {
                "ID": 29,
                "BEZEICHNUNG": "Webhosting"
            }
        ],
        "ROLLEN": [
            {
                "ID": 4,
                "BEZEICHNUNG": "Assetverantwortlicher"
            }
        ]
    },
    "GEAENDERT": "2015-11-11  15:43:00 UTC",
    "DETEKTIONSINTERVALL": 7,
    "ZPLDETEKT": "2015-11-07  12:06:00 UTC",
    "SOFTWAREINVENTAR": [

    ],
    "BEHEBUNGSVERANTWORTLICH": [
```

```
46              {
47                  "NUTZER": {
48                      ...
49                  },
50                  "HAUPTVERANTWORTLICH": "true"
51              },
52              ...
53          ],
54          "DETEKTIONSVERANTWORTLICH": [
55              {
56                  "NUTZER": {
57                      ...
58                  },
59                  "HAUPTVERANTWORTLICH": "true"
60              },
61              ...
62          ]
63  }
```

Das Beispiel wurde bei den Schlüsseln `BEHEBUNGSVERANTWORTLICH` und `DETEKTIONSVERANTWORTLICH` gekürzt, da ein Nutzer bereits als Wert bei Schlüssel `VERANTWORTLICH` dargestellt ist.

Abrufen eines Nutzers (`getUser`)
Das Abfragen eines Nutzers ist genauso wie im Falle eines Assets anhand der `ID` möglich. Die Rückgabe ist ebenfalls im JSON-Format und bereits als Wert des Schlüssels `VERANTWORTLICH` in Listing 6.14 dargestellt.

Setzen des Datums der letzten Detektion eines Assets (`setAssetDetectionTS`)
Als Eingabe erwartet die Funktion die `ID` des Assets, bei dem das Datum der letzten automatischen Detektion gesetzt werden soll. Es wird im Feld `ZPLDETEKT` in der Tabelle *assets* gehalten und auf einen zur Ausführung der Funktion aktuellen Zeitstempel gesetzt.

Weiterhin stellt die Asset- und Benutzerverwaltung allgemein Funktionen zum Hinzufügen, Entfernen und Ändern von Einträgen in den Tabellen zur Verfügung.

6.5 Das Steuerungssystem

Das Steuerungssystem hat die Aufgabe der Koordination der Daten aus der Schwachstellen-Dokumentation und der Asset- und Benutzerverwaltung, da diese wie im Datenbankschema zu sehen ist, stark untereinander vernetzt sind.

Des Weiteren als wichtiger Teil des Schwachstellenmanagements umfasst ein Steuerungssystem einen Scheduler zur Ausführung automatischer Abläufe – beispielsweise den Import von Schwachstellen aus Fremdquellen, dem Aufräumen der Datenbank und der automatischen Ausführung von Detektionsläufen.

Auch eine API ist Bestandteil des Steuerungssystems, die in diesem Fall nicht der Kommunikation der Komponenten im Schwachstellenmanagement untereinander, sondern als generelle Schnittstelle für Komponenten außerhalb des Schwachstellenmanagements dient, beispielsweise der CMDB oder Softwarelösungen des *Security Information & Event Managements* (SIEM). Die API bietet den kompletten Funktionsumfang des Kerns des Steuerungssystems an, d.h. ebenfalls alle Funktionen der Schwachstellen-Dokumentation und Asset- und Benutzerverwaltung, und wird daher in diesem Abschnitt nicht mehr einzeln im Detail beschrieben.

Eine Schnittstelle für Nutzer bietet sie in Form eines Webportals an, das über einen Webbrowser aufgerufen und bedient wird.

Die zentrale Komponente zur Verwaltung der Daten der Schwachstellen-Dokumentation und Asset- und Benutzerverwaltung ist in einem Kernmodul zusammengefasst. Eine Illustration des Steuerungssystems sowie Zusammenhänge mit den weiteren technischen Komponenten ist in Abbildung 6.2 gezeigt.

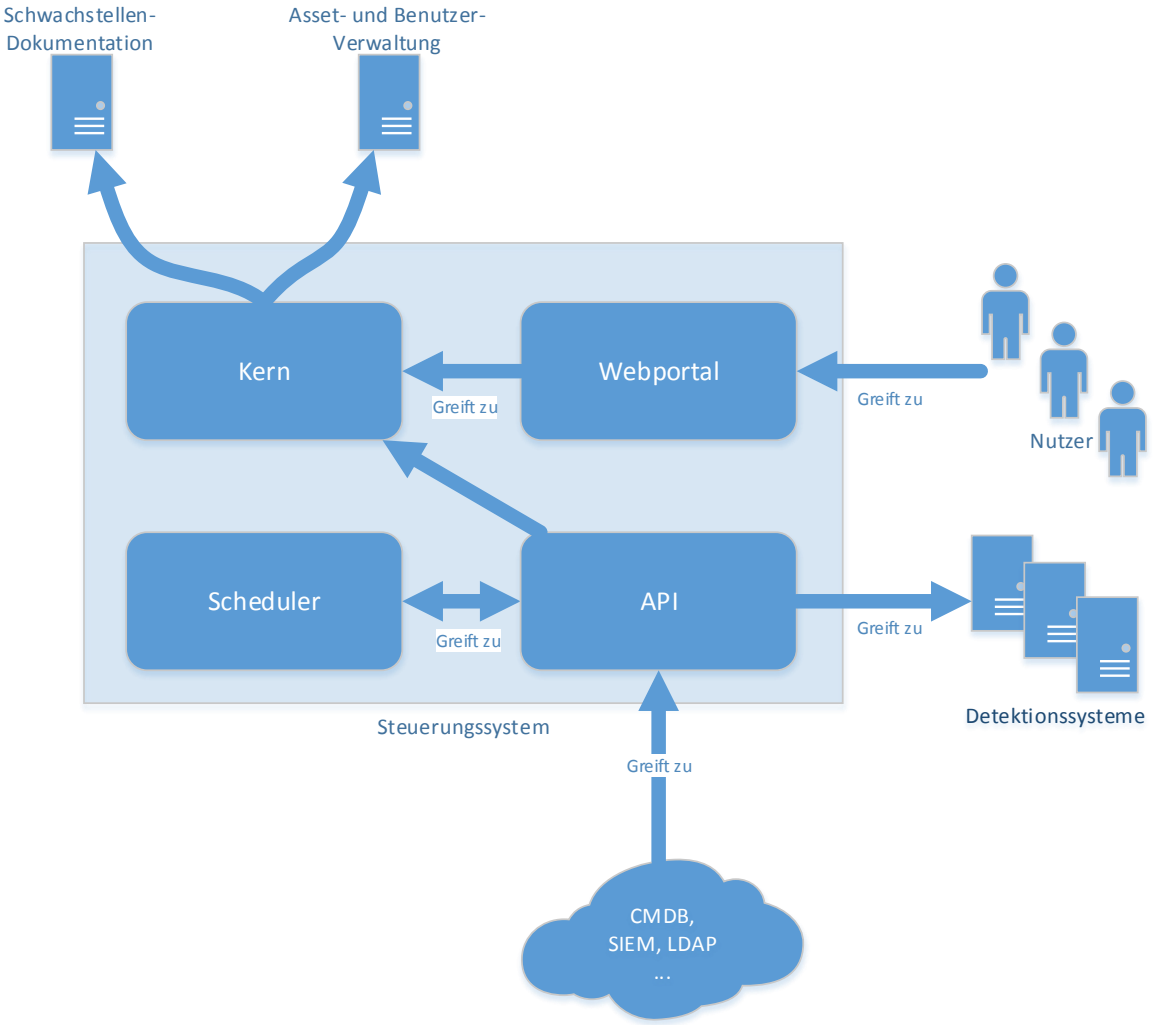


Abbildung 6.2: Komponenten des Steuerungssystems und Schnittstellen

6.5.1 Verzeichnisstruktur

Das Steuerungssystem ist auf einer mit Linux betriebenen Maschine eingerichtet und im Verzeichnis `/srv/vulste/` installiert. Die Verzeichnisstruktur des Projektverzeichnisses ist wie folgt organisiert:

- **bin/**
  In diesem Verzeichnis liegt die Implementierung des Steuerungssystems in Form des Java-Projektes *stesys.jar*. Die Datei beinhaltet den Kern und die API des Steuerungssystems.

- **conf/**
  Das `conf`-Verzeichnis enthält die Konfigurationsdatei *stesys.conf* des Steuerungssystems. Darin beschrieben sind der Port, auf dem die API angeboten wird und Verbindungsinformationen zu den APIs der Schwachstellen-Dokumentation und der Asset- und Benutzerverwaltung (URL sowie Zugangsdaten). Ein beispielhafter Inhalt der Datei *stesys.conf* ist in Listing 6.15 dargestellt. Des Weiteren sind Angaben über die vom Scheduler verwendeten Zugangsdaten („scheduser“ und „schedpassw“), mit denen er sich an der API authentifiziert, in der Datei enthalten. In der Regel hat der Scheduler mit diesen Zugangsdaten freien Zugriff auf alle Funktionen des Steuerungssystems, diese können jedoch je nach Bedarf eingeschränkt werden.

Listing 6.15: Inhalt der Datei *stesys.conf*

```
1  port          8080
2  scheduser     scheduler
3  schedpassw    l27_mf#!Gn
4
5  dokuapi       https://vulmgmtdoku.lrz.de/api/
6  dokuuser      vuldokuuser
7  dokupassw     r!_239lBi12Ja!
8
9  anmanapi      https://vulmgmtanman.lrz.de/api/
10 anmanuser     vulanmanuser
11 anmanpassw    MAi26Juiee1.0!
```

Neben der allgemeinen Konfigurationsdatei ist in dem Verzeichnis eine weitere Konfigurationsdatei *detektsys.conf* abgelegt. Diese enthält Verbindungsinformationen und Zugangsdaten über Detektionssysteme mit öffentlichen IP-Adressen sowie Detektionssysteme, die innerhalb eines privaten Netzes liegen. Bei letzteren wird die öffentliche IP-Adresse des Netzes angegeben, der beispielsweise mittels Port-Forwarding auf das Detektionssystem verweist.

Ein Beispielinhalt der Datei *detektsys.conf* ist in Listing 6.16 gezeigt.

Listing 6.16: Inhalt der Datei *detektsys.conf*

```
1  //HOST                          //USER        //PASSW
2
3  [PUBLIC]
4  detektsys1.lrz.de:8080          duser1       9wdjsa234j
5  detektsys2.lrz.de:8080          duser2       38ssjKl23!
6  detektsys3.lrz.de:8080          duser3       2WZlsa_54h
7  //[...]
8
9  [INSTITUTE ID=34]
10 129.187.33.112:54000            iuser34      o3iWhr!#12
11
12 [INSTITUTE ID=112]
13 129.187.23.9:44344              iuser112     o3iWhr!#12
```

Detektionssysteme mit öffentlichen Adressen werden unter der Kategorie „[PUBLIC]“ aufgelistet, solche in privaten Netzen innerhalb einer institutsspezifischen Kategorie.

Darüber hinaus ist noch eine weitere Konfigurationsdatei zur Beschreibung der Eskalation in dem Ordner enhalten: *escal.conf*.

Listing 6.17: Inhalt der Datei *escal.conf*

```
1  // LEVEL      // BEHEBUNGSNOTW      // TAGE OFFEN
2
3  1             kritisch             3
4  1             hoch                 7
5  2             kritisch             7
6  2             hoch                 14
```

Eine Zeile entspricht einem Datensatz. Spalte zwei („Behebungsnotwendigkeit“) sowie Spalte drei („Tage offen“) geben die Bedingungen an, wann das jeweilige Eskalationslevel (erste Spalte) festgelegt wird. In der ersten Spalte wird der Wert des resultierenden Eskalationslevels beschrieben.

Die letzte Konfigurationsdatei *sched.conf* betrifft die Konfiguration des Schedulers. Sie beinhaltet die Regeln zur Festlegung des Zeitpunktes bei Erstellung eines automatischen Eintrags. Ein Beispiel ist in Listing 6.18 gezeigt.

Listing 6.18: Inhalt der Datei *sched.conf*

```
1  // T-START    // T-END      // DIST
2  09:00         18:00         5
```

Der Inhalt der Datei legt fest, in welchem Zeitintervall der Scheduler Einträge zur automatischen Ausführung anlegt (`T-START` bis `T-END`), in diesem Fall neun Uhr in

der Früh bis 18 Uhr. Das Feld `MINDIST` legt fest, in welchem Zeitabstand Einträge angelegt werden – in diesem Fall alle 5 Minuten. Das heißt, der erste Eintrag wird um 09:00 Uhr, der zweite um 09:05 Uhr, der dritte um 09:10 Uhr, usw. angelegt.

- **tmp/**
  Das Verzeichnis `tmp` wird vom Steuerungssystem als temporäre Ablage für Implementierungen von Detektionsverfahren genutzt, die im weiteren Verlauf zur Aktualisierung an Detektionssysteme gesendet werden.

- **sched/**
  In diesem Verzeichnis befinden sich die Datei, die durch den Cron-Daemon täglich ausgeführt wird (*cron.pl*) sowie der „Ausführungsplan“ *scheduler_exec*. Beide Dateien werden in Abschnitt 6.5.4 näher erläutert.

6.5.2 Der Kern

Eine der Hauptaufgaben des Kerns eines Steuerungssystems ist die Authentifizierung und Autorisierung zur Zugriffskontrolle auf Daten und Funktionen. Außerdem werden bereits hier, abhängig vom angemeldeten Nutzer und seinen Verantwortlichkeiten, die Ergebnisse des Zugriffs auf Informationen gefiltert, um deren unberechtigte Einsicht zu verhindern. Aufbauend auf diesen Prinzipien kann der Kern prinzipiell alle Funktionen der API der Schwachstellen-Dokumentation und Asset- und Benutzerverwaltung ausführen. Diese werden insofern nicht mehr im Einzelnen in diesem Abschnitt besprochen.

Bei Funktionsaufrufen der Schwachstellen-Dokumentation oder Asset- und Benutzerverwaltung, die als Parameter einen Nutzer erwarten (beispielsweise einen Editor der Funktion `addRemedy` oder `addDetectionMechanism`), wird die ID des angemeldeten Nutzers automatisch als Eingabe übergeben.

Des Weiteren bietet der Kern zusätzlich folgenden Funktionsumfang an:

Authentifizieren eines Nutzers (`authenticate`)
Die Authentifizierung eines Nutzers im Schwachstellenmanagement erfolgt über den Zugriff auf einen *Lightweight Directory Access Protocol* (LDAP) Dienst. Da das Steuerungssystem in Java implementiert ist, wird das für diesen Zweck verwendbare *Java Naming and Directoy Interface* (JNDI) genutzt.

Setzen und Auslesen der Assetspezifität(`setAssetSpec, getAssetSpec`)
Der Zweck der Funktion *setAssetSpecificity* ist es, eine Schwachstelle einem bestimmten Asset in der Tabelle *ist_asset_spezifisch* zuzuweisen (siehe Abschnitt 5.6.2). Als Parameter erwartet sie den Identifikator der Schwachstelle sowie den Identifikator des Assets.

Die Funktion *getAssetSpecificity* liefert mit Parameter eines Identifikators einer Schwachstelle alle Assets, die der Schwachstelle zugewiesen wurden, hingegen mit Identifikator eines Assets, alle Schwachstellen, die dem Asset zugewiesen wurden. Ein zusätzlicher Parameter gibt an, von welchem Typ der Identifikator ist.

Manuelle Eskalation der Verantwortung zur Verifikation (`escalateVerification`)
Die Verantwortung der Verifikation von Schwachstellen ($\neq$ Schwachstellentickets) kann

über diese Funktion an einen Schwachstellen-Prüfer innerhalb eines Dienstes zugewiesen werden. Als Eingabe erfordert die Funktion den Identifikator der Schwachstelle sowie sowie den Identifikator eines Dienstes. Falls es einen Schwachstellen-Prüfer innerhalb des Dienstes gibt (auslesbar aus der Tabellen *dienste* und *dienstmap* über die Asset- und Benutzerverwaltung), wird die Verantwortung der Verifikation diesem zugewiesen, andernfalls erfolgt keine Zuweisung.

Automatische Zuweisung von Verantwortlichkeiten (`setResponsibilities`)
Die Zuweisung von Verantwortlichkeiten erfolgt, ausgelöst von einem Ereignis, insbesondere bei Erstellen eines Meldungstickets, dem Ändern des Status einer Schwachstelle und der Erstellung eines Schwachstellentickets.

Bei Erstellung eines Meldungstickets fragt die Funktion zunächst alle Nutzer in der Rolle des Schwachstellen-Dokumentators aus der Asset- und Benutzerverwaltung ab und vergleicht die zurückgelieferte Liste mit dem Feld `BEARBEITER` der Liste offener Meldungstickets aus der Schwachstellen-Dokumentation. Schwachstellen-Dokumentatoren, denen aktuell kein weiteres Meldungsticket zugewiesen ist, werden bevorzugt in der Einteilung der Verantwortlichkeit des neuen Meldungstickets ausgewählt. Falls alle Schwachstellen-Dokumentatoren bereits ein Meldungsticket zugewiesen haben, erfolgt die Auswahl per Zufall aus allen Schwachstellen-Dokumentatoren.

Dieses Vorgehen wird auch bei der Auswahl der Verantwortlichkeit für die weiteren Teilaktivitäten bei Schwachstellen genutzt. Die Auswahl erfolgt jedoch entsprechend innerhalb der Gruppe an Nutzern in der jeweiligen Rolle – beispielsweise bei Ändern des Status einer Schwachstelle zu *categorize* wird innerhalb aller Kategorisierungsverantwortlichen ausgewählt.

Bei der Zuweisung der Verantwortlichkeit bei der Bearbeitung von Schwachstellen gibt es jedoch einen Spezialfall: Falls einer Person die Verantwortung zur Bearbeitung einer Teilaktivität zugewiesen wurde und die Person die Teilaktivität erfüllt sowie zudem die Rolle innehat, die für die Bearbeitung der nächsten Teilaktivität verantwortlich ist, wird der Person automatisch die Verantwortung der nächsten Teilaktivität zugewiesen.

Hat beispielsweise Person A die Rolle des Schwachstellen-Bewerters und des Kategorisierungsverantwortlichen inne, wird ihr nach jeder Bewertung einer Schwachstelle, die Verantwortlichkeit zur Kategorisierung derselben ebenfalls zugewiesen.

Ein weiterer Spezialfall ergibt sich, falls eine Schwachstelle einem bestimmten Asset zugewiesen wurde (über die Funktion *setAssetSpecificity*). In diesem Fall wird die Verantwortlichkeit der Schwachstelle bereits bei Status „detect“ – aufgrund der klaren Verbindung der Schwachstelle zum Asset – allen Detektionsverantwortlichen des jeweiligen Assets zugewiesen.

Eine Ausnahme bildet die Zuweisung von Schwachstellentickets. Diese erfolgt immer an den hauptverantwortlichen Behebungsverantwortlichen eines Assets.

Nutzer, denen im Schwachstellenmanagement eine Verantwortlichkeit zugewiesen wurde, werden wahlweise per E-Mail benachrichtigt. Die Benachrichtigung kann von den Nutzern über das Webportal aktiviert oder deaktiviert werden und wird in Form des Feldes `MAILBENACHRICHTIGUNG` gespeichert.

Versenden einer E-Mail Benachrichtigung (`notificateUser`)
Die Funktion zur Benachrichtigung eines Nutzers wird vor allem nach Zuweisung der Verantwortlichkeit einer Teilaktivität (z.B. der Bewertung einer Schwachstelle) genutzt. Als Eingabe erwartet sie lediglich den Identifikator des Nutzers. Nach Aufruf liest die Funktion aus der Asset- und Benutzerverwaltung die Informationen über den entsprechenden Nutzer aus und verschickt die Benachrichtigung an die eingetragene Kontakt-E-Mail-Adresse.

Berechnen der Behebungsnotwendigkeit
Da die Behebungsnotwendigkeit aus der Kritikalität einer Schwachstelle und der Kritikalität eines Assets errechnet wird, erfolgt sie im Steuerungssystem. Als Eingabeparameter erwartet die Funktion die `ID` der Schwachstelle sowie die `ID` des Assets und holt sich beide Werte der Kritikalität über die API der jeweiligen Komponente. Die Implementierung der Behebungsnotwendigkeit in Java ist in Listing 6.19 abgebildet.

Listing 6.19: Eingabe der Funktion *detect* auf API-Ebene

```
1  public static Behebungsnotwendigkeit behebungsnotwendigkeit(
       SSCriticality sscrit, AssetCriticality acrit)
2  {
3    if (acrit == AssetCriticality.kritisch)
4    {
5      if (sscrit == SSCriticality.gering)
6        return Behebungsnotwendigkeit.hoch;
7      else return Behebungsnotwendigkeit.kritisch;
8    }
9    if (acrit == AssetCriticality.hoch)
10   {
11     if (sscrit == SSCriticality.gering
12      || sscrit == SSCriticality.mittel)
13       return Behebungsnotwendigkeit.mittel;
14     else if (sscrit == SSCriticality.hoch)
15       return Behebungsnotwendigkeit.hoch;
16     else return Behebungsnotwendigkeit.kritisch;
17   }
18   if (acrit == AssetCriticality.mittel)
19   {
20     if (sscrit == SSCriticality.gering)
21       return Behebungsnotwendigkeit.gering;
22     else if (sscrit == SSCriticality.mittel)
23       return Behebungsnotwendigkeit.mittel;
24     else return Behebungsnotwendigkeit.hoch;
25   }
26   else //acrit == gering
27   {
28     if (sscrit == SSCriticality.gering)
29       return Behebungsnotwendigkeit.gering;
30     else return Behebungsnotwendigkeit.mittel;
```

```
31     }
32 }
```

Die Klasse `SSCriticality` beschreibt die Kritikalität der Schwachstelle und `AssetCriticality` die Kritikalität des Assets. Beide sind als Auflistungen realisiert und haben die Werte *NA*, *gering*, *mittel*, *hoch* und *kritisch*.

Detektion von Schwachstellen (`detect`)
Die Funktion kann auf zwei unterschiedliche Arten aufgerufen werden: Zum einen mit Eingabe einer Liste von `ID`s von Einträgen, die in der Tabelle *assets* der Asset- und Benutzerverwaltung enthalten sind. In diesem Fall werden Schwachstellen auf allen Assets mit der übergebenen `ID` detektiert, die im Verantwortungsbereich des ausführenden Nutzers liegen.

Die zweite Möglichkeit besteht aus Eingabe eines Netzbereichs (dieser kann auch nur eine einzige IP-Adresse umfassen) durch einen Start- und einen Endwert. Bei dieser Möglichkeit werden Schwachstellen auf allen Assets detektiert, deren Netz-ID im Netzbereich liegt und der Ausführende die entsprechenden Rechte besitzt. Jeweils ein Beispiel der Eingabe beider Möglichkeiten ist in Listing 6.20 dargestellt.

Beide Funktionsaufrufe erwarten im Übrigen einen Parameter, der eine boolesche Aussage darüber trifft, ob das Feld `ZPLDETEKT` auf die aktuelle Zeit gesetzt werden soll.

Listing 6.20: Eingabe der Funktion *detect* auf API-Ebene

```
1  // Möglichkeit 1
2  {
3      "FUNCTION": "detect",
4      "PARAM": {
5          "ASSETS": [12, 34, 66, 231, 45],
6          "SCHWACHSTELLEN": [22, 33],
7          "SETZPLDETEKT": "FALSE"
8      }
9  }
10
11 // Möglichkeit 2
12 {
13     "FUNCTION": "detect",
14     "PARAM": {
15         "START": "192.168.1.12",
16         "END": "192.168.13.0",
17         "VULID": [],
18         "SETZPLDETEKT": "TRUE"
19     }
20 }
```

Bei beiden Möglichkeiten wird eine Liste von `ID`s von Schwachstellen übergeben, die detektiert werden sollen. Ist die Liste leer, erfolgt die Detektion aller Schwachstellen in der Schwachstellen-Dokumentation.

Das Steuerungssystem nimmt in jedem Fall eine Selektion der relevanten Schwachstellen für jedes Asset anhand der Kombination dreier Filterungsmechanismen vor.

Zum einen werden zunächst alle Schwachstellen ausgefiltert, zu denen kein Detektionsverfahren beschrieben ist, bzw. die nicht detektierbar sind (d.h. es gibt keinen Eintrag in der Tabelle *detektionsverfahren*, der auf die entsprechende Schwachstelle verweist).

Des Weiteren fragt es einerseits aus der Schwachstellen-Dokumentation das Schwachstellenakzeptanzkriterium ab und vergleicht damit die sich ergebende Behebungsnotwendigkeit der Kombinationen aus Asset und allen zu detektierenden Schwachstellen.

Bei der dritten Methode fragt es die Softwarekonfiguration des jeweiligen Assets über ein Detektionssystem sowie aus der Asset- und Benutzerverwaltung ab und filtert Schwachstellen aus, deren betroffene Softwareprodukte nicht in der Softwarekonfiguration des Assets vorkommen. Da in der Regel die Bezeichnung von Softwareprodukten nicht universell standardisiert ist und sich daher von Quelle zu Quelle unterscheiden kann, wird der Abgleich mit Hilfe eines geeigneten Verfahrens durchgeführt, um die Ähnlichkeit zu schätzen.

Im Falle dieser Implementierung überprüft die Funktion für alle Einträge der Software auf dem jeweiligen Asset sowie allen Einträgen von der Schwachstelle betroffener Softwareprodukte, ob der Bezeichner des einen Softwareprodukts ein Teilstring innerhalb des Bezeichners des anderen Softwareproduktes vorkommt und umgekehrt. Falls eines davon zutrifft, wird die Schwachstelle nicht von der Detektion ausgeschlossen, andernfalls schon.

Zur Erhöhung der Zuverlässigkeit wird die Groß- und Kleinschreibung vernachlässigt. Zudem werden ausschließlich die Bezeichner der Softwareprodukte verglichen, Hersteller und Versionen hingegen nicht. Dies ist insofern sinnvoll, da teilweise Angaben über Hersteller nicht angegeben sind (beispielsweise bei Abfrage über Paketmanager wie *dpkg* oder *rpm*) und sich die Schreibweise der Version bei gleicher Bedeutung teilweise stark unterscheidet (z.B. „1.1.45“ und „1_1_45“).

> Bei der Filterung anhand des Bezeichners der Softwareprodukte ist unbedigt zu beachten, dass das oben beschriebene Verfahren im Prinzip eine Liste der Schwachstellen aufbaut, die auf jeden Fall in der Detektion betrachtet werden müssen. So kann es jedoch vorkommen, dass Schwachstellen, die das Asset möglicherweise betreffen, von der Detektion ausgeschlossen werden. Insofern wäre der Aufbau einer Liste von Schwachstellen, die auf keinen Fall auf dem Asset auftreten können, deutlich sinnvoller, jedoch auch deutlich komplizierter auf Basis von Softwareprodukten zu berechnen. In letzterem Fall würden alle Schwachstellen in der Detektion berücksichtigt werden, die nicht in der Liste sind (d.h. das Komplement der berechneten Liste bezüglich der zu detektierenden Schwachstellen). Damit das beschriebene Verfahren zuverlässig funktionieren kann, muss sichergestellt werden, dass eine Schwachstelle vollständig kategorisiert und das komplette Softwareinventar des jeweiligen Assets bekannt ist. Andernfalls würden bei der Filterung sehr viele False-Positives und eine unvollständige Whitelist resultieren.

Es wird angenommen, dass das gesamte Softwareinventar eines Assets ermittelt werden

kann und auch die Liste betroffener Softwareprodukte von einer Schwachstelle komplett ist. Andernfalls ist die Filterung von Schwachstellen anhand der Softwareprodukte zu deaktivieren. Eine Beeinträchtigung der Effektivität der Detektion folgt nicht aus dem Entfernen dieser Filtermethode.

Nach der Filterung wird mit der resultierenden Liste relevanter Schwachstellen zusammen mit der Asset-ID die Funktion zur Detektion (`detect`) auf dem zur Durchführung ausgewählten Detektionssystem aufgerufen. Die Auswahl des Detektionssystems wird wie folgt durchgeführt: Falls ein Asset in einem privaten Netz ist, wählt das Steuerungssystem anhand des durch das Feld `INSTITUT` und `NETZID` der Tabelle *assets* ein passendes Detektionssystem aus der Konfigurationsdatei *detektsys.conf* aus und weist diesem das Asset zu. Bei mehreren Detektionssystemen wird zufällig eines ausgewählt.

Hat das Asset hingegen eine öffentliche IP-Adresse, wird ein beliebiges Detektionssystem aus *detektsys.conf* gewählt.

Behandlung von Detektionsergebnissen (`processDetectionResult`)
Das Steuerungssystem nimmt Ergebnisse einer Detektion durch diese Funktion entgegen. Als Parameter erwartet die Funktion die `ID` des Assets, auf dem die Detektion durchgeführt wurde sowie eine Liste von Identifikatoren positiv detektierter Schwachstellen.

Durch Auslesen der Einträge in der Tabelle *schwachstellentickets* aus der Schwachstellen-Dokumentation prüft die Funktion, ob ein entsprechendes Schwachstellenticket bereits existiert. Dies geschieht durch Abgleich der Felder `SCHWACHSTELLE` und `ASSET` aller Schwachstellentickets in der Datenbank mit der Kombination aus Asset-ID und aller Einträge der Liste der detektierten Schwachstellen. Falls keines existiert, wird ein neues Schwachstellenticket angelegt.

Falls bereits ein Schwachstellenticket für eine Kombination aus Schwachstelle und Asset existiert, werden die Kriterien zur Eskalation des Tickets aus der Konfigurationsdatei *escal.conf* überprüft und gegebenenfalls eine Eskalation vorgenommen.

Detektion im Probelauf-Modus (`detectionDryRun`)
Eine Detektion im Probelauf-Modus arbeitet prinzipiell wie die oben beschriebene Funktion zur Detektion, nur dass auf Basis der zurückgelieferten Ergebnisse keine Schwachstellentickets erstellt werden, sondern die konkrete Ausgabe eines Detektionsverfahrens. Die Funktion erwartet als Parameter jedoch den Identifikator eines Detektionsverfahrens, das ausgeführt wird, genauso wie einen Netzidentifikator des Computersystems, auf dem die Detektion durchgeführt wird. Die Funktion liefert die konkrete Ausgabe des ausgeführten Detektionsverfahrens zurück (beispielsweise *true* oder eine Liste offener Ports).

Diese Funktion ist vor allem zum Zweck der Detektion als Dienstleistung für Nutzer außerhalb des Schwachstellenmanagements sinnvoll (vgl. Abschnitt 5.10.2), da deren Computersysteme in der Regel nicht als Assets in der Asset- und Schwachstellen-Dokumentation geführt werden.

Aktualisierung der Detektionssysteme und Detektionsagenten (`updateDetectionMechanisms`)
Die Aktualisierung erfolgt für alle Komponenten. Dazu lädt das Steuerungssystem

die auf der Schwachstellen-Dokumentation liegenden Implementierungen über die API und das HTTPS Protokoll herunter und verteilt diese an alle Detektionssysteme in der Datei *detektsys.conf*.

Die Verteilung wird mittels *Secure Copy* (scp) durchgeführt und automatisch in das `tmp/` Verzeichnis des Detektionssystems kopiert. Die Detektionsverfahren sind in ein mit *7z* komprimiertes Paket zusammengefasst. Im Anschluss daran wird die Funktion zur Durchführung der Aktualisierung auf dem jeweiligen Detektionssystem aufgerufen. Eine schematische Darstellung der Aktualisierung von Detektionsverfahren, aus-

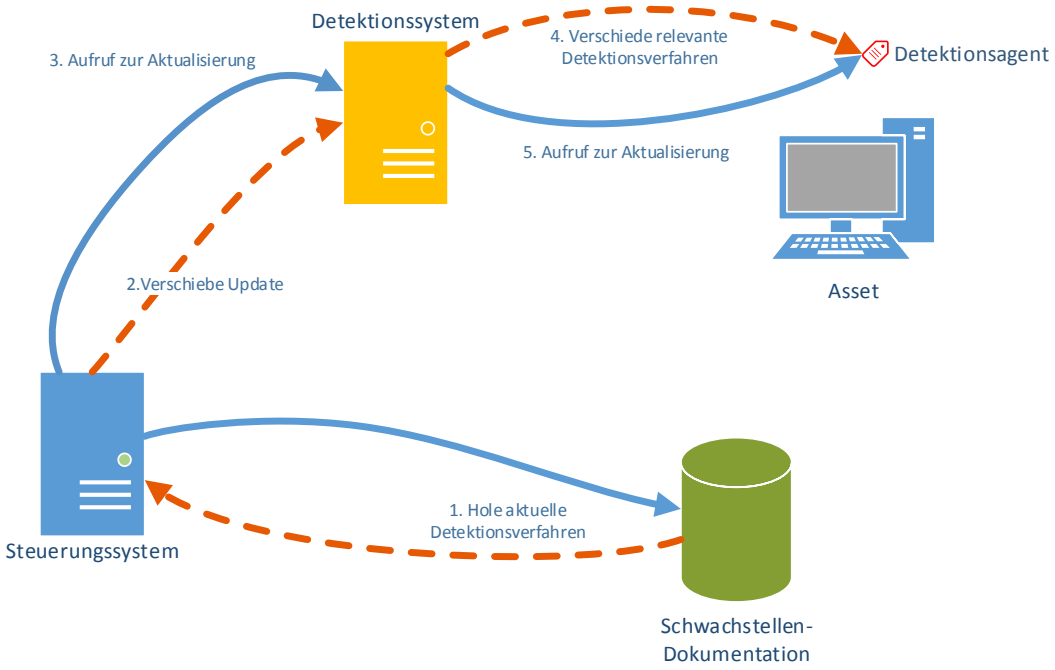


Abbildung 6.3: Aktualisierung der Detektionsverfahren auf den Detektionssystemen sowie Detektionsagenten

gehend vom Steuerungssystem, ist in Abbildung 6.3 illustriert. Blaue, durchgängige Pfeile symbolisieren den Funktionsaufruf, rote gestrichelte Pfeile die Übertragung der Implementierung der Detektionsverfahren.

Hinzufügen und Entfernen eines Befehls an den Scheduler (`addScheduleEntry`) Dem Scheduler können Einträge über geplante Aktionen (z.B. Detektion, Import aus Fremdquellen) zu einem bestimmten Zeitpunkt hinzugefügt oder entfernt werden. Dies geschieht durch Bearbeiten der in der Datei *sched_exec* gehaltenen Liste (siehe Abschnitt 6.5.4). Vor der Bearbeitung der Liste wird jedoch überprüft, ob der ausführende Nutzer die dafür erforderlichen Rechte – zum einen zum Bearbeiten allgemein, als auch zur Ausführung des jeweiligen Befehls anhand der von ihm eingenommenen Rollen – hat. Als Eingabe benötigt die Funktion zum Hinzufügen einen bestimmten Zeitpunkt

der Ausführung sowie den exakten Befehl an die API des Steuerungssystems.

6.5.3 Das Webportal

Das Webportal dient als Schnittstelle der technischen Komponenten zu den darin tätigen Nutzern. Dem Nutzer werden nur die für ihn wichtigen Informationen angezeigt. Je nach Rollen, die er erfüllt, werden ihm eine oder mehrere Registerkarten jeweils für eine der Teilaktivitäten angezeigt: Ersterfassung von Schwachstellen und Dokumentation, Überprüfung, Bewertung und Kategorisierung von Schwachstellen sowie Erstellung von Detektionsverfahren, Detektion, Behebung und Kontrollprüfung. Innerhalb der Registerkarten ist jeweils eine Liste der Schwachstellen (bzw. Tickets), die dem Nutzer zugewiesen wurden.

Darüber hinaus können alle Nutzer rollenunabhängig Meldungstickets über das Portal erstellen und bereits dokumentierte Schwachstellen und dazugehörige Massnahmen einsehen. Der Umfang der einsehbaren Schwachstellen kann über eine Konfigurationsdatei im Webportal zusätzlich beschränkt werden, beispielsweise falls die Existenz einer besonders kritischen, asset-spezifische Schwachstelle (siehe Beispiel im folgenden Kapitel) nicht bekannt werden soll. Assetspezifische Schwachstellen sind in der Tabelle *ist_asset_spezifisch* der Schwachstellen-Dokumentation dokumentiert.

6.5.4 Der Scheduler

Der Scheduler besteht aus drei Bestandteilen: Zum einen ein Skript (*sched/cron.pl*), das täglich über den in der Regel auf UNIX-Systemen standardmäßig installierten *Cron-Daemon* ausgeführt wird, sowie einer Anwendung zum Auslesen der in der Datenbank festgelegten Werte für Aktualisierungsintervalle in der Tabelle *fremdquellen*, Detektionsintervalle aus der Tabelle *assets* sowie dem Zeitpunkt der Erstellung, das Schwachstellenticket und die Behebungsnotwendigkeit von Schwachstellentickets (Tabelle *schwachstellentickets*). Die dritte Komponente bildet eine Konfigurationsdatei (*sched/sched_exec*) – der eigentliche Ausführungsplan.

Die Datei *sched_exec* hat das in Listing 6.21 gezeigte Format (inklusive Beispiele). Eine Zeile entspricht der Ausführung eines oder mehrerer Befehle zu einem bestimmten Zeitpunkt.

Listing 6.21: Inhalt der Datei *sched_exec*

```
1 //TIMESTAMP                    //API CALLS
2
3 // Detektion mehrerer Assets
4 2015-11-14  15:45:00 UTC      {"FUNCTION": "detect", "PARAM": {"
    ASSETS": [12, 34, 66, 231, 45],"SCHWACHSTELLEN": [22, 33], "
    SETZPLDETEKT": "TRUE"}}
5
6 // Import aus Fremdquelle und Setzen des Feldes ZPLUPDATE
7 2015-11-19  09:00:00 UTC      {"FUNCTION": "import", "PARAM": {"
    FREMDQUELLE": 13}}, {"FUNCTION": "SETZPLUPDATE", "PARAM": {}}
8
9 //[..]
```

Die Datei *cron.pl* wird minütlich vom Cron-Daemon aufgerufen und gleicht die Systemzeit mit dem in der Datei *sched_exec* aufgelisteten Zeitstempel des jeweiligen Aufrufs ab. Falls die Uhrzeit übereinstimmt, werden die in der zweiten Spalte stehenden – ein oder mehrere als JSON-Objekt dargestellte – Befehle an die API des Steuerungssystems gesendet und dort ausgeführt. Einträge, deren Zeitstempel bereits abgelaufen ist, werden automatisch entfernt.

Die erwähnte Java-Anwendung wird einmal pro Tag vom Cron-Daemon aufgerufen und liest zum einen aus der Tabelle *fremdquellen* die Felder `ZPLIMPORT` und `UPDATEINTERVALL` sowie zum anderen aus der Tabelle *assets* die Felder `ZPLDETEKT` und `DETEKTIONSINTERVALL` aus. Pro Eintrag in *fremdquellen* wird überprüft, ob das aktuelle Datum größer oder gleich dem Zeitpunkt $ZPLIMPORT + UPDATEINTERVALL$ ist. Falls ja, erstellt die Anwendung einen Eintrag in der Datei *sched_exec* mit dem Befehl zur Aktualisierung der jeweiligen Quelle. Der Parameter zum Setzen des Feldes `ZPLIMPORT` ist dabei auf *TRUE* gesetzt (bei manueller Ausführung einer Detektion hingegen immer auf *FALSE*). Ein analoges Vorgehen wird auch für jeden Eintrag in *assets* umgesetzt.

Des Weiteren liest die Anwendung das Intervall zum Aufräumen der Datenbank `CLEANUP_INTERVALL` aus Tabelle *vulmgmt_config* sowie den Zeitpunkt der letzten Durchführung `LAST_CLEANUP`. Falls $LAST\_CLEANUP + CLEANUP\_INTERVALL$ größer oder gleich der aktuellen Systemzeit ist, wird ein Eintrag zum Ausführen der Funktion zum Aufräumen (`cleanVulnerabilityDoc`) der Datenbank erstellt.

Auch liest die Anwendung den Zeitpunkt der Erstellung offener Schwachstellentickets (Tabelle *schwachstellentickets* Feld `ERSTELLT`) aus und gleicht die Daten mit den Regeln der Eskalation in Tabelle *conf/escal.conf* (vgl. Listing 6.17) ab. Treffen alle Kriterien zu, erstellt die Anwendung einen Eintrag in der Datei *sched_exec* zur Ausführung einer automatischen Kontrollprüfung in Form eines Aufrufs zur Detektion der im Schwachstellenticket angegebenen Schwachstelle auf einem Asset.

Die Zeitpunkte der automatisch angelegten Einträge hängen vom Inhalt der Datei *sched.conf* sowie dem Zeitpunkt des zuletzt angelegten Assets ab.

6.6 Detektionssysteme

Im Schwachstellenmanagement kann es in der Regel eine beliebige Anzahl von Detektionssystemen geben. Zur einfacheren Wartbarkeit sind alle Detektionssysteme gleichartig für Systeme auf Linux-Basis implementiert und konfiguriert. Ein Detektionssystem wird in zwei verschiedenen Formen angeboten: Zum einen als Softwarepaket zur Installation auf beliebigen Linux-Maschinen und zum anderen vorkonfiguriert auf einem virtuellen Datenträger. So kann es einfach, auch innerhalb von Instituten bzw. privaten Netzen, als virtuelle Maschine zum Einsatz kommen.

6.6.1 Verzeichnisstruktur

Das Detektionssystem ist im Verzeichnis `/srv/vulnetd/` installiert und teilt sich in folgende weitere Unterverzeichnisse auf:

- **bin/**
  Hier liegt die Implementierung des Kernmoduls des Detektionssystems als Datei *ndetekt.jar*.

- **conf/**
  Das Verzeichnis `conf/` enthält die Konfigurationsdatei *ndetekt.conf* des Detektionssystems. In dieser ist neben dem Port zur Kommunikation mit der eigenen API, auch der Pfad zur API in der URL der Detektionsagenten sowie der allgemeine Port und die Zugangsdaten aller Detektionsagenten angegeben, darüber hinaus die URL zum Aufrufen der Funktionen des Steuerungssystems in Verbindung mit den notwendigen Zugangsdaten. Eine beispielhafte Form der Datei *ndetekt.conf* ist in Listing 6.22 gezeigt.

Listing 6.22: Inhalt der Datei *ndetekt.conf*

```
1  port            8080
2
3  stesysapi       https://vulstesys.lrz.de/api
4  stesysuser      steuser
5  stesyspassw     mNiB26JB!Mar#
6
7  agentpath       api
8  agentport       49999
9  agentuser       agent007
10 agentpassw      G01d3nEy#!
```

- **detekt/**
  In diesem Verzeichnis liegen die Detektionsverfahren aus Netzsicht in Form von ausführbaren Anwendungen. Das Namensschema der Dateien lautet wie folgt:

$$< SSID > \_ < DID > \tag{6.1}$$

  Hier bezeichnet $< DID >$ die ID des Detektionsverfahrens in der Schwachstellen-Dokumentation und $< SSID >$ den Identifikator der dadurch detektierbaren Schwachstelle. Auf diese Weise kann es mehrere Detektionsverfahren für eine Schwachstelle geben, die dennoch – in Kombination mit dem Identifikator des jeweiligen Detektionsverfahrens – eine eindeutige Bezeichnung haben. So detektieren beispielsweise zwei Dateien „23_223“ und „23_419“ dieselbe Schwachstelle durch ein jeweils unterschiedliches Verfahren.

- **res/**
  Im `res/` Verzeichnis können weitere Dateien und Anwendungen abgelegt werden, die für die Ausführung von Detektionsverfahren notwendig sind. Beispielsweise kann hier ein Netzwerkscanner wie *nmap* abgelegt werden, auf den die Detektionsverfahren im Ordner `detekt/` zugreifen und dessen Funktionsumfang verwenden können.

- **tmp/**
  Bei einer Aktualisierung der Detektionsverfahren werden diese temporär im Ordner `tmp/` gespeichert. Zunächst gepackt als `7z` komprimierte Datei mit dem Namen *update.7z* und schließlich entpackt im dadurch angelegten Ordner `tmp/update/`. Dieser enthält wiederum die Verzeichnisse `netz/` sowie `system/`, in denen die eigentlichen aktualisierten Detektionsverfahren des jeweiligen Typs liegen.

6.6.2 Kern und API eines Detektionssystems

Im Detektionssystem bildet die API eine Komponente des Kerns, wodurch dessen Funktionen auch von anderen Systemen aus ausführbar werden. Prinzipiell haben Detektionssysteme drei Aufgabenbereiche, die durch den Kern realisiert werden. Einerseits die Ausführung von Detektionsverfahren aus Netzsicht und die Kommunikation mit den Detektionsagenten sowie als drittes die Durchführung der Aktualisierung – sowohl auf dem eigenen System sowie der vom Steuerungssystem dynamisch zugewiesenen Detektionsagenten.

Aktualisierung der Detektionsverfahren (`update`)
Die Aktualisierung wird ausgehend von dem im `tmp/` Verzeichnis liegenden und vorher durch das Steuerungssystem übertragenen Pakets *update.7z* mit Detektionsverfahren durchgeführt. Da alle Detektionssysteme eine einheitliche Konfiguration haben, können alle Detektionsverfahren aus den durch das Entpacken der vorhandenen Datei *update.7z* entstandenen Verzeichnissen `tmp/update/netz/` installiert werden. Die Installation erfolgt durch Löschen aller vorhanden Detektionsverfahren im Ordner `detekt/` und verschieben aller neuen Detektionsverfahren in dasselbe. Zudem kann ein Verzeichnis `tmp/update/netz/res/` für weitere Dateien vorhanden sein, die zur Ausführung von Detektionsverfahren notwendig sind. Diese werden in das Verzeichnis `res/` kopiert.

Die Aktualisierung der Detektionsagenten hingegen kann aufgrund der Heterogenität der Assets in der Regel nicht einheitlich für alle Detektionsagenten ausgeführt werden. Grundvoraussetzung für ein funktionierendes Detektionsverfahren aus Systemsicht ist das Vorhandensein der notwendigen Plattform (insbesondere Laufzeitumgebungen wie z.B. Perl- oder Java-Implementierungen). In dieser Implementierung wird angenommen, dass die notwendige Plattform auf allen Assets mit einem Detektionsagenten vorhanden ist. Die Anfrage einer Aktualisierung hat die in Listing 6.23 beispielhaft beschriebene Form.

Listing 6.23: Aufruf der Aktualisierung von Detektionsverfahren

```
1  {
2      "FUNCTION": "update",
3      "PARAM": {
4          "ASSET": {
5              "ASSETID": 12,
6              "NETZID": "192.168.34.12"
7              "BETRIEBSSYSTEM": {
8                  "HERSTELLER": "SUSE Linux GmbH",
9                  "BEZEICHNER": "SUSE Linux Enterprise Server",
10                 "VERSION": "11"
11             }
12         },
13         "DETEKTIONSVERFAHREN": [
14             {
15                 "ID": "23_291",
16                 "TYP": "system",
17                 "PLATTFORM": [
18                     {
```

```
19                              "HERSTELLER": "Microsoft",
20                              "BEZEICHNER": "Windows",
21                              "VERSION": "7"
22                          },
23                          {
24                              "HERSTELLER": "Microsoft",
25                              "BEZEICHNER": "Windows",
26                              "VERSION": "8"
27                          },
28                          {
29                              "HERSTELLER": "Microsoft",
30                              "BEZEICHNER": "Windows",
31                              "VERSION": "8.1"
32                          }
33                          ...
34                      ]
35                  }
36                  ...
37              ]
38      }
39  }
```

Für jede Aktualisierung der Detektionsverfahren der Systemsicht (d.h. von Detektionsagenten) wird eine Dokumentation des Assets aus der Asset- und Benutzerverwaltung (Tabelle *assets*) übergeben, auf dem der Detektionsagent aktualisiert werden soll, sowie die Dokumentation für jedes Detektionsverfahren – insbesondere Betriebssysteme und Plattformen (aus der Schwachstellen-Dokumentation, Tabelle *plattformen*), für die das Verfahren anwendbar ist. Wären die Laufzeitumgebungen auf den Assets nicht, wie angenommen, einheitlich, müsste zudem der Inhalt der Tabelle *abhaengig_von* mit dem Softwareinventar des jeweiligen Assets abgeglichen werden.

Das Detektionssystem entscheidet anhand der Eingabe und Übereinstimmung des Betriebssystem des Assets und der Plattform der Detektionsverfahren, welche davon auf dem jeweiligen Asset installiert werden. Das Detektionssystem installiert die geeigneten Detektionsverfahren durch Ausführung der Funktion zum Löschen aller Detektionsverfahren auf dem Detektionsagenten (`removeAllDetectMechanisms`) und anschließendem Kopieren der aktualisierten Detektionsverfahren in das Verzeichnis `detect/` des entsprechenden Detektionsagenten über das *Secure Copy*-Protokoll.

Ausführen der Detektion (`detect`)
Da die Filterung von Schwachstellen mit einer möglichen Relevanz für das Asset bereits auf dem Steuerungssystem vorgenommen wird, erwartet die Funktion als Eingabe eine Identifikation aller Schwachstellen, die im Detektionsdurchlauf überprüft werden sollen. Zudem benötigt die Detektion die ID und Netz-ID des Assets, um die Detektion durchführen und dem Steuerungssystem zu einem Asset zuweisbare Ergebnisse zurückliefern zu können.

Listing 6.24: Aufruf der Detektion einer Liste von Schwachstellen auf einem Asset in Detektionssystemen

```
{
    "FUNCTION": "detect",
    "PARAM": {
        "ASSET": {
            "ID": 14,
            "NETZID": "asset33.lrz.de"
        },
        "SCHWACHSTELLENIDS": [
            17, 34, 37, 41, 44, 58, 65, 67, 118
        ]
    }
}
```

Das Detektionssystem führt auf den in Listing 6.24 gezeigten Beispielaufruf auf dem Asset mit der ID=14 und Hostnamen *asset33.lrz.de* für die Schwachstellen in `SCHWACHSTELLENIDS` aufgelisteten Schwachstellen einen Detektionsdurchlauf aus. Dazu gleicht es die aufgelisteten Identifikatoren einzeln mit den Namen der Dateien in Verzeichnis `detekt/` ab. Falls eine der IDs auf den ersten Teil des Namens $< SSID >$ passt, wird dieses ausgeführt.

Neben der Ausführung der Detektionsverfahren über das Netz ruft das Detektionssystem auch über die Adresse `https://asset33.lrz.de:49999/api/` die Funktion des Detektionsagenten zur Ausführung der Detektionsverfahren aus Systemsicht auf (`detect`, siehe folgender Abschnitt). Der Port „49999“ sowie Kontext „api/“ der API des Detektionsagenten sind in der Konfigurationdatei *ndetekt.conf* dokumentiert und für alle Detektionsagenten einheitlich.

Bearbeitung von Detektionsergebnissen (`processDetectionResult`)
Die Funktion erwartet als Eingabe die ID des jeweiligen Assets, auf dem die Detektion ausgeführt wurde, und eine Liste der detektierten Schwachstellen, die zum einen aus der Durchführung der Detektion aus Netzsicht sowie von Detektionsagenten kommen. Ihre Verarbeitung zielt auf ein Zusammenfassen doppelter Einträge (die beispielsweise aus Netz- und Systemsicht oder mehreren Verfahren aus Netzsicht detektiert wurden) und den Aufruf der gleichnamigen Funktion zur Bearbeitung von Detektionsergebnissen auf dem Steuerungssystem ab.

Detektion im Probelauf-Modus (`detectionDryRun`)
Die Funktion funktioniert ähnlich wie oben beschriebene Funktionen zur Detektion. Als Parameter erwartet sie einen Identifikator eines Detektionsverfahrens und einen Netzidentifikator des Assets, auf dem Schwachstellen detektiert werden sollen. Ausgeführt werden die Verfahren im Ordner `detect/` in diesem Fall aufgrund des letzten Teil des Namens: $< DID >$. Die Rückgabe besteht aus der Ausgabe des ausgeführten Detektionsverfahrens.

Abfrage des Softwareinventars eines Assets (`getSoftwareList`)
Die Funktion ruft prinzipiell lediglich dieselbe Methode auf dem spezifizierten Detektionsagenten auf und liefert das Ergebnis zurück. Als Parameter benötigt sie dazu die ID und Netz-ID des Assets.

6.7 Detektionsagenten

Der einzige Zweck, der durch Detektionsagenten erfüllt wird, ist die Unterstützung der Detektion von Schwachstellen auf genau dem Asset, auf dem es installiert ist. Zu beachten ist, dass die Softwareumgebung auf allen Assets in Hochschulnetzen üblicherweise nicht vorgegeben bzw. standardisiert werden kann, wodurch Detektionsagenten auf einer Vielzahl an Betriebssystemen einsetzbar sein müssen.

In diesem Fall wurde die Implementierung des Detektionsagenten in Java *JRE 1.8* von Oracle umgesetzt, einer Java-Implementierung, die für *Microsoft Windows*, *Linux*, *Solaris* sowie *Apple OS X* verfügbar ist.

Genau wie alle vorher beschriebenen Komponenten besitzen Detektionsagenten eine wie in Abschnitt 6.1 beschriebene API zur Kommunikation mit den Detektionssystemen (vgl. Abbildung 5.3).

6.7.1 Verzeichnisstruktur

Die Verzeichnisstruktur wird anhand eines Assets mit Linux erläutert. Im Allgemeinen ist die Wahl des Installationsverzeichnisses des Detektionsagenten für die Funktionalität nicht von Bedeutung und lediglich die Verzeichnisstruktur innerhalb des Installationsverzeichnisses entscheidend.

Auf einem Linux-System (auf anderen Systemen analog), wird ein neuer Benutzer „vulagent“ für die Verwaltung des Detektionsagenten angelegt, um sicherzustellen, dass dieser nicht durch Nutzer mit Standardberechtigungen (d.h. ohne Administratorrechte) manipuliert werden kann. Der Benutzer „vulagent“ hat hingegen erweiterte Berechtigungen, um effektive Detektionsverfahren ausführen zu können.

Die Installation des Detektionsagenten erfolgt in ein Unterverzeichnis `agent/` des „Home“=Verzeichnis des Benutzers, d.h. üblicherweise `/home/vulagent/agent/`, welches sich wiederum in folgende Unterverzeichnisse aufgliedert:

- **bin/**
  Im `bin/` Verzeichnis liegt die eigentliche Implementierung des Detektionsagenten, die ausführbare Datei *sdetekt.jar*.

- **conf/**
  In diesem Verzeichnis liegt die Konfigurationsdatei *sdetekt.conf* des Detektionsagenten, welche die Asset-ID (Feld `ID` des Eintrags in der Tabelle *assets*), den Port und Kontext, auf dem die REST-Schnittstelle der API lauscht und den Benutzernamen und der SHA256-Hash des Passwort zur Nutzung der API enthält. Da wie in der Konfigurationsdatei des Detektionssystems gezeigt, das gemeinsame Passwort aller Detektionsagenten „G01d3nEy#!“ lautet, hat abgelegter SHA256-Hash zur Authentifizierung die in Listing 6.25 gezeigte Form. Die Authentifizierung auf den Detektionsagenten erfolgt durch Berechnung des SHA256-Hashs der Eingabe des Passworts und Abgleich mit dem in der Datei *sdetekt.conf* hinterlegten Wert.

Listing 6.25: Beispielhafter Inhalt der Konfigurationsdatei *sdetekt.conf*

```
1 assetid       12
2 port          49999
3 context       api
4 user          agent007
5 passw         4c4fc39353f3aa5650404e7b6612560c
6               db442a445112eeaf99541499e022dcc2
```

- **detekt/**
  Im Verzeichnis `detekt/` liegen die ausführbaren Implementierungen der Detektionsverfahren. Diese sind genau wie auf den Detektionssystemen gemäß dem Namensschema

  $$< SSID > \_ < DID > \tag{6.2}$$

  benannt, d.h. dem Identifikator der Schwachstelle, die sie detektieren ($< SSID >$) in Kombination der ID des Detektionsverfahrens, das sie in der Schwachstellen-Dokumentation beschreibt ($< DID >$).

- **res/**
  Falls erforderlich, können weitere Dateien und Ordner, die für die Ausführung von Detektionsverfahren notwendig sind (beispielsweise Software-Bibliotheken oder weitere Anwendungen) hier abgelegt werden. Die Implementierungen der Detektionsverfahren im Ordner `detekt/` können auf diese Dateien zugreifen.

6.7.2 Kern und API der Detektionsagenten

Zur Erfüllung ihres Zwecks stellen Detektionsagenten – implementiert im Kern und zugreifbar über ihre API – eine Reihe an Funktionen zur Verfügung, welche üblicherweise ausschließlich durch Detektionssysteme über das Netz aufgerufen werden. Die im Konzept im vorherigen Kapitel beschriebene Funktion zur automatisierten Behebung von Schwachstellen durch Detektionsagenten (insbesondere die Installation von Patches) wird zunächst nicht näher in der Implementierung ausgeführt, da es sich dabei insbesondere um eine Aktivität des Patchmanagements handelt. Eine Möglichkeit zur Erweiterung der Detektionsagenten um diese Funktionalität muss jedoch vorgesehen werden.

Folgende Funktionen sind ausführbar:

Ausführen der Detektion (Systemsicht) (`detect`)
Zum Ausführen eines Detektionslaufs aus Systemsicht erwartet die Funktion eine Liste an Identifikatoren von Schwachstellen, die auf dem Asset detektiert werden sollen. Ein beispielhafter Aufruf der Funktion ist in Listing 6.26 gezeigt.

Listing 6.26: Beispielhafter Aufruf der Funktion zur Detektion auf einem Detektionsagenten

```
1 {
2     "FUNCTION": "detect",
3     "PARAM": [12, 23, 54]
4 }
```

Nach Aufruf der Funktion führt der Detektionsagent alle Detektionsverfahren im Verzeichnis `detekt/` aus, deren Dateinamenpräfix (< *SSID* >) mit dem einer der Identifikatoren der Schwachstellen übereinstimmt, die bei Aufruf der Funktion übergeben wurden. Am Beispiel des Aufrufs aus Listing 6.26 würden also alle Detektionsverfahren ausgeführt werden, die auf die Namensschemata *12_**, *23_** und *54_** passen, wobei „*“ als Wildcard dient.

Wie bereits im Konzept beschrieben liefert jedes Detektionsverfahren eine Antwort *true* zurück, falls es eine Schwachstelle positiv detektiert hat, andernfalls *false*. Falls mehrere Detektionsverfahren dieselbe Schwachstelle positiv detektieren, wird das Ergebnis mittels logischer *Oder*-Operation zusammengefasst.

Die Liste der detektierten Schwachstellen wird in Kombination mit dem Identifikator des Assets an die Funktion zur Bearbeitung des Detektionsergebnisses an das aufrufende Detektionssystem zurückgesendet.

Detektion im Probelauf-Modus (`detectionDryRun`) Die Funktion erwartet als Parameter den Identifikator eines Detektionsverfahrens, das ausgeführt werden soll. Die Durchführung erfolgt analog zur gleichnamigen Funktion auf Detektionssystemen und liefert die Ausgabe des ausgeführten Detektionsverfahrens zurück.

Abfrage des Softwareinventars des Assets (`getSoftwareList`)
Die Funktion zur Abfrage des Softwareinventars des Assets, auf dem der Detektionsagent läuft, erwartet keine weiteren Parameter. Die Rückgabe besteht aus der ID des Assets sowie einer Liste von Softwareprodukten, deren Beschreibung (mit Ausnahme einer ID) gemäß dem Schema der in der Datenbank vorhandenen Tabelle *softwareprodukte* entsprechen. Eine beispielhafte Antwort bei Aufruf der Funktion ist in Listing 6.27 gezeigt.

Listing 6.27: Rückgabe einer beispielhaften Abfrage des Softwareinventars auf einem Detektionsagenten auf Grundlage der Rückgabe des *RPM Package Managers*

```
1  {
2      "ASSETID": 1,
3      "SOFTWARE": [
4          {
5              "HERSTELLER": "",
6              "BEZEICHNER": "screen",
7              "VERSION": "4.2.1"
8          },
9          ...
10     ]
11 }
```

Die Umsetzung der Abfrage der Software ist stark von der Plattform abhängig, auf der das Detektionssystem läuft. Und selbst auf gleicher Plattform gibt es je nach Version verschiedene Vorgehensweisen: Beispielsweise bei der Umsetzung durch Abfrage eines

Paketmanagers kann sich die Implementierung selbst auf zwei Plattformen mit demselben Linux-Kernel stark unterscheiden: So nutzen *Debian*-Distributionen üblicherweise *Debian Package* (dpkg), SUSE-Distributionen hingegen in der Regel den *RPM Package Manager* (rpm). In Listing 6.28 ist die Implementierung zum Auslesen der installierten Softwarepakete über den *RPM Package Manager* gezeigt.

Listing 6.28: Implementierung der Abfrage installierter Software unter Benutzung des *RPM Package Managers*

```
1
2  class Agent {
3    String  fAgentID;
4
5    class SoftwareResponse {
6      String              ASSETID;
7      Softwareprodukt[] SOFTWARE;
8
9      public SoftwareResponse {
10     }
11
12   }
13
14
15   class Softwareprodukt {
16     String HERSTELLER;
17     String BEZEICHNER;
18     String VERISON;
19
20     public Softwareprodukt {
21     }
22
23   }
24
25   //...
26
27   public String getSoftware()
28   {
29     SoftwareResponse     result  = new SoftwareResponse();
30     ArrayList<Softwareprodukt>  swlist  = new ArrayList<>();
31
32     result.ASSETID = fAsset;
33
34     Runtime r = Runtime.getRuntime();
35     try
36     {
37       // Den RPM Package Manager mit formatierter Ausgabe
38       // aufrufen.
```

```
39          // Beispielausgabe (eine Zeile):
40          //    "screen####4.2.1"
41          Process p = r.exec("rpm -qa --qf '%{NAME}####%{VERSION}'
                ");
42
43          // Ausgabe des RPM über den Scanner zeilenweise auslesen
44          Scanner s = new Scanner(p.getInputStream());
45
46          while (s.hasNext())
47          {
48            try
49            {
50                String line = s.nextLine();
51
52                // Zeile an der Zeichenfolge "####" in
53                // Arrayelemente auftrennen
54                // und neues Objekt der Klasse Softwareprodukt mit
55                // den Informationen erstellen
56                String[] cols = line.split("####");
57                Softwareprodukt sw = new Softwareprodukt();
58                sw.HERSTELLER = "";
59                sw.BEZEICHNER = cols[1];
60                sw.VERSION    = cols[2];
61
62                // In Liste der Softwareprodukte hinzufügen
63                swlist.add(sw);
64              }
65            }
66            catch (StringIndexOutOfBoundsException sioobe)
67            {
68              sioobe.printStackTrace();
69            }
70          }
71        }
72        catch (IOException ioe)
73        {
74          ioe.printStackTrace();
75        }
76
77        result.ASSETID = result.toArray(new Softwareprodukt[result
              .size()]);
78
79        Gson gson = new Gson();
80
81        // Objekt result der Klasse SoftwareResponse in ein JSON-
              Objekt "konvertieren" und zurückgeben
82        return gson.toJson(result);
```

```
83      }
84
85    //...
86  }
```

Die Funktion gibt ein serialisiertes JSON-Objekt auf Basis der Klasse `SoftwareResponse` zurück. Dieses enthält die ID des Assets sowie eine Liste an Objekten der Klasse `Softwareprodukt`.

Als Quelle der Informationen dient die formatierte Ausgabe aller installierter Softwarepakete aus dem *RPM Package Manager*, wodurch das Auslesen bzw. Trennen der Informationen des Bezeichners der Software sowie der Version der Software im weiteren Verlauf leichter umgesetzt werden kann. Der Ausgabe-Stream des Packetmanagers wird direkt an ein Objekt der Klasse `Scanner` übergeben, wodurch die zeilenweise Bearbeitung ermöglicht wird.

Eine Zeile hat die Form „%{$NAME$}####%{$VERSION$}“, wobei %{$NAME$} der Bezeichner des Softwareprodukts und %{$VERSION$} entsprechend die Version darstellt. Insofern muss eine Zeile zur Isolation dieser lediglich an der Zeichenfolge „####“ aufgetrennt werden und mit den Einzelinformationen ein Objekt der Klasse `Softwareprodukt` erstellt werden.

Die resultierende Liste an Softwareprodukten wird der Variable `SOFTWARE` im Rückgabeobjekt der Klasse `SoftwareResponse` übergeben und mittels des Gson-Objekts `gson` in ein JSON-Objekt serialisiert, welches die Form der in Listing 6.27 beispielhaft gezeigten Zeichenkette hat. Dieses wird so per HTTPS als Antwort an das anfragende Detektionssystem zurückgeschickt.

Entfernen der Detektionsverfahren (`removeAllDetectMechanisms`)
Diese Funktion erwartet keine weiteren Parameter. Ihr Zweck ist es, alle im Verzeichnis `detekt/` enthaltenen Dateien (d.h. Implementierungen von Detektionsverfahren) zu löschen. Die Funktion wird üblicherweise vom Detektionssystem aufgerufen, das die Aktualisierung des jeweiligen Detektionsagenten vornimmt, bevor es die neuen Detektionsverfahren mittels *Secure Copy* (scp) über das Netz in den Ordner verschiebt.

7 Exemplarische Anwendung des Konzepts

In diesem Abschnitt wird ein Durchlauf durch eine Variante des Schwachstellenmanagements im Umfeld des *Münchner Wissenschaftsnetzes* (vgl. Abschnitt 3.1.1) anhand zweier konkreter Schwachstellen sowie einem Asset in Form eines Webservers am *Leibniz-Rechenzentrum* durchgeführt.

Eine der beiden Schwachstellen betrifft die Konfiguration eines Webservers und wird durch einen Nutzer im MWN gemeldet. Sie soll als Beispiel zur Verdeutlichung von Verantwortlichkeiten für manuelle Verfahren dienen, die von der Identifikation bis zur Prävention durchlaufen werden (Abschnitt 7.5).

Die andere Schwachstelle – aufgetreten in verschiedenen Java-Implementierungen – wird aus einer Fremdquelle importiert und anhand Daten aus einer weiteren Fremdquelle automatisch über eine Abbildung bewertet sowie kategorisiert (Abschnitt 7.4).

Zusätzlich werden zwei weitere von dem Konzept aus Kapitel 5 abgedeckten Fälle beschrieben: Zum einen die Integration von Funktionen von Schwachstellen-Scannern in das Konzept (Abschnitt 7.6), sowie die Detektion als Dienst im MWN, bereitgestellt durch das LRZ (Abschnitt 7.7).

Der folgende Abschnitt beschreibt den Rahmen des Schwachstellenmanagements im MWN, gefolgt von einer Übersicht über Rollen in Abschnitt 7.2, der in den darauf folgenden Fallbeispielen auftretenden Personen.

7.1 Schwachstellenmanagement im Münchner Wissenschaftsnetz

Der Anwendungsbereich auf Dienstleisterebene und entsprechend die Umsetzung des Schwachstellenmanagements beschränkt sich auf mehrere hundert bis tausend Server auf Linux-Basis, die im Leibniz-Rechenzentrum (LRZ) betrieben werden, da diese den Großteil der Serverlandschaft bilden und beispielsweise Server auf Windows-Basis vergleichsweise wenig im Einsatz sind und zudem durch ein zentrales, ebenfalls vom LRZ betriebenes Patchmanagementsystem (Windows Server Update Services, WSUS) abgedeckt sind.

Des Weiteren wird das Schwachstellenmanagement als Dienstleistung an Institute und Organisationen im Münchner Wissenschaftsnetz zur Unterstützung der Behebung von Schwachstellen angeboten, wobei Leistungen von Seiten des LRZ die Bereitstellung von Daten über Schwachstellen und Maßnahmen sowie die Nutzung einer mandantenfähigen Schnittstelle für Nutzer umfassen, die als Webportal über Steuerungssysteme MWN-weit erreichbar ist. Ebenfalls im Dienstangebot enthalten ist die Nutzung zentraler und institutsinterner Detektionssysteme und deren Management (d.h. Konfiguration und Sicherstellung der Aktualität) sowie zentral durch das LRZ verwaltete Detektionsagenten. Darüber hinaus wird die Asset- und Benutzerverwaltung sowie das im Schwachstellenmanagement genutzte Steuerungssystem zur Nutzung an den Kunden angeboten.

Im Hochschulnetz wurden daher durch das LRZ zur Unterstützung des Schwachstellenmanagements sowie der Ermöglichung des Angebots des Schwachstellenmanagements als Dienstleistung für Kunden, einige technische Komponenten zentral im LRZ installiert: Jeweils eine Schwachstellen-Dokumentation sowie eine Asset- und Benutzerverwaltung, ein gemeinsam genutztes, zentrales Steuerungssystem sowie ein Dutzend Detektionssysteme, um Detektionsläufe effektiv parallelisieren zu können. Jeder als Asset deklarierte Linux-Server führt zudem eine Instanz des Detektionsagenten aus.

Vorgaben durch den Chief Vulnerability Manager betreffen folgende Punkte:

- Das **Schwachstellenakzeptanzkriterium** ist auf den Wert *gering* festgelegt, es werden also alle Schwachstellen detektiert und behoben, deren Behebungsnotwendigkeit auf einem Asset auf *mittel* oder höher ist.
- Der Standardwert des **Updateintervalls** von Fremdquellen ist auf sieben Tage festgelegt, zur **automatischen Detektion** auf zehn Tage.
- Aufgrund möglicher Abweichungen des Anwendungsbereichs von Kunden ist die **globale Blacklist** an Softwareprodukten für Schwachstellen, die in Fremdquellen ignoriert werden sollen, leer, d.h. es gibt keine Einschränkungen.
- **Detektionsverfahren** werden für eine bessere Kontrollierbarkeit und Übersichtlichkeit der Softwareumgebung auf Assets und Detektionssystemen in Perl implementiert. Zudem ist durch die Implementierung in Perl die Plattformunabhängigkeit, zumindest in Bezug auf die Lauffähigkeit der Programme, gewährleistet. Insofern haben alle Systeme, auf denen Detektionsagenten betrieben werden, eine gleichartige, kompatible Version einer Perl-Laufzeitumgebung.

7.2 Übersicht der Rollen und Akteure

Im Szenario werden zur Verdeutlichung des Durchlaufs Namen fiktiver Personen verwendet. Eine Liste der in der Beschreibung vorkommenden Namen und ID, der zugeteilten Rollen, des zugehörigen Instituts sowie – falls spezifiziert – der Verantwortlichkeit für spezielle Dienste findet sich in Tabelle 7.1. Bei assetspezifischen Rollen (Asset-Verantwortlicher, Detektionsverantwortlicher sowie Behebungsverantwortlicher) wird das dazugehörige Asset hinter der Rollenbezeichnung in Klammern angegeben.

Die Nutzer im Schwachstellenmanagement werden automatisch täglich mittels einer selbstentwickelten Anwendung eingetragen. Diese liest dazu automatisch alle Einträge eines bereits im Hochschulnetz existierenden LDAP-Verzeichnisdienstes aus und aktualisiert über die API des Steuerungssystems die Liste der Nutzer im Schwachstellenmanagement. Auf diese Weise neu-angelegte Nutzereinträge bekommen zunächst standardmäßig lediglich die Rolle *Nutzer* und damit verbundene Berechtigungen zugewiesen – das Einsehen von nicht-assetspezifischen Schwachstellen, Melden von Schwachstellen sowie der Nutzung eines *Scan-Selfservice* (siehe Abschnitt 7.7).

Weitere Berechtigungen können erst durch Schwachstellenmanager desselben Instituts des Nutzers bzw. durch den Chief Vulnerability Manager für institutsübergreifende Rollen wie dem Quellenverantwortlichen vergeben werden.

Person	ID	Institut	Rollen	Dienst
Richard Spiegel	1	LRZ [1]	Chief Vulnerability Manager	
Hermann Baum	7	LRZ	Quellenverantwortlicher	
Martin Wolke	15	LRZ	Detektionsverantwortlicher (webserver01.lrz.de)	
Joachim Holzmann	17	LRZ	Schwachstellen-Dokumentator, Schwachstellen-Prüfer	
Jens Schubert	18	LRZ	Kategorisierungsverantwortlicher	
Robert Zweig	33	LRZ	Schwachstellenmanager	
Julia Gitternetz	43	LRZ	Schwachstellen-Bewerter	
Joseph Wand	50	LRZ	Behebungsverantwortlicher (webserver01.lrz.de)	
Karl Blatt	55	LRZ	Schwachstellen-Prüfer	Schulung und Kurse
Herbert Grün	65	IPP [2]	Detektionsverantwortlicher	
Günter Reinhard	93	IFI [3]	Detektionsverantwortlicher	
Maria Mustermann	223	LRZ	Asset-Verantwortlicher (webserver01.lrz.de), Behebungsverantwortlicher (webserver01.lrz.de)	Webhosting
Reinhold Maurer	22321	LMU [4]	Nutzer	
Johannes Schwarz	100283	TUM [5]	Nutzer	
...				

[1] Leibniz-Rechenzentrum
[2] Max-Planck-Institut für Plasmaphysik
[3] Institut für Informatik an der Ludwig-Maximilians-Universität
[4] Ludwig-Maximilians-Universität
[5] Technische Universität München

Tabelle 7.1: Übersicht über Personen und Rollen

7.3 Assets und Fremdquellen

Eine Grundlage für das Schwachstellenmanagement bildet die Bestimmung der darin betrachteten Assets und Quellen für den Import von Schwachstelleninformationen. In den folgenden Unterabschnitten wird ein Durchlauf eines Assets für die Aktivitäten der Identifikation und Klassifikation beschrieben. Die Aktivität der Beseitigung und Abschwächung wird darauf aufbauend in Verbindung mit Schwachstellen betrachtet.

7.3.1 Identifikation und Dokumentation von Assets

In den jeweiligen Instituten und auch im Leibniz-Rechenzentrum ist der jeweils zuständige Schwachstellenmanager für die Identifikation relevanter Assets und Schwachstellen-Fremdquellen verantwortlich. Im Falle des LRZ nimmt diese Rolle der Nutzer Robert Zweig ein.

Unter der Auswahl dieser Assets befindet sich auch ein Webserver, der verschiedene Webanwendungen zur Verfügung stellt sowie bereits in der CMDB des Hochschulrechenzentrums dokumentiert ist. Alle darauf betriebenen Anwendungen werden über virtuelle Hosts

(„vhosts“) realisiert. Robert Zweig greift daher auf eine Funktion in der CMDB zurück, die es ihm nach einer Authentifizierung erlaubt, das in der CMDB ausgewählte *Configuration Item* (den Webserver) über die API des Steuerungssystems in die Asset- und Benutzerverwaltung einzutragen. Die Auswahl des Assets für das Schwachstellenmanagement wird in der CMDB hinterlegt, sodass bei jeder Änderung eines Attributs des Webservers (beispielsweise der Netzidentifikator) die Daten in der Asset- und Benutzerverwaltung über die API des Steuerungssystems aktuell gehalten werden.

Robert Zweig informiert daher den in der CMDB eingetragenen Asset-Owner, Maria Mustermann, und teilt dieser im Schwachstellenmanagement die Rolle des Asset-Verantwortlichen und zugleich die Rolle des dafür zuständigen hauptverantwortlichen Behebungsverantwortlichen für den Webserver zu. Maria Mustermann ist zudem Teil des Teams, das zuständig für den Dienst *Webhosting* ist.

Als zweiten Behebungsverantwortlichen wählt er Joseph Wand aus, der Maria Mustermann bei der Behebung von Schwachstellen auf dem Asset unterstützen soll. Da es sich bei dem Webserver um ein weltweit über das Internet erreichbares und so leicht angreifbares Asset handelt und er eine entsprechend hohe Kritikalität hat, sollte die Behebung von Schwachstellen zeitnah durchgeführt werden. Dies wird durch die Bestimmung mehrerer Behebungsverantwortlicher erreicht.

Die Rolleneinteilung erfolgt über das Webportal des Steuerungssystems, das Robert Zweig entsprechend seiner Zugehörigkeit zum Institut (dem LRZ) als auch seiner Rolle, die er innehat, die für ihn relevanten Informationen und Funktionen aus der Schwachstellen-Dokumentation und der Asset- und Benutzerverwaltung über die mandantenfähige Oberfläche anzeigt. Darunter fällt auch die Zuweisung von Asset-Verantwortlichen, Behebungsverantwortlichen und Detektionsverantwortlichen zu einem bestimmten Asset. Robert Zweig weist dem Asset des Weiteren einen hauptverantwortlichen Detektionsverantwortlichen, Martin Wolke, zu.

Zunächst greift Maria Mustermann über das Webportal auf die für sie bestimmten Funktionen der Schwachstellen-Dokumentation und Asset- und Benutzerverwaltung zu – darunter insbesondere Möglichkeiten zur Dokumentation und Einsicht von Assets und Schwachstellentickets, Schwachstellen und Maßnahmen. Auch wenn beim Import des Assets aus der CMDB, die Robert Zweig vorgenommen hat, bereits alle notwendigen Attribute gesetzt sind, kontrolliert Maria Mustermann diese Informationen, nachdem die Sicherstellung der Aktualität der Daten für das Asset in ihren Verantwortungsbereich fällt. Die entsprechende Dokumentation in der Asset- und Benutzerverwaltung ist in Tabelle 7.2 beschrieben. Des Weiteren gehört zur Beschreibung des Assets die Dokumentation des aktuellen Softwareinventars.

7.3.2 Kategorisierung von Assets

Die Kategorisierung des betrachteten Webservers, d.h. Erfassung des Softwareinventars (vgl. Tabelle 7.2) erfolgt aus einer Kombination der Daten aus der CMDB sowie des auf dem Asset installierten und ausgeführten Detektionsagenten.

Ebenso wie die Attribute eines als Asset deklarierten CIs in der CMDB (siehe vorheriger Abschnitt), wird dieses auch bei Änderungen des Softwareinventars des Assets automatisch von der CMDB über Zugriff auf die API des Steuerungssystems in der Asset- und Benutzerverwaltung aktuell gehalten.

Feld	Wert
ID	3
NetzID	webserver01.lrz.de
Beschreibung	Bereitstellung verschiedener Webanwendungen, darunter die offizielle Webpräsenz des LRZ und Websites zur Kursplanung
Kritikalität	hoch
Institut	Leibniz-Rechenzentrum
Typ	Webserver
Betriebssystem	SUSE Linux Enterprise Server 11
Geändert	2015-10-30 13:02:22
Letzte autom. Detektion	
Detektionsintervall	10 Tage
Verantwortlich	Maria Mustermann
Dienst	Webhosting

Tabelle 7.2: Beschreibung eines Webservers im Schwachstellenmanagement

7.3.3 Implementierung von Fremdquellen

Auch ist es der Schwachstellenmanager, der die relevanten Schwachstellen-Fremdquellen identifiziert. In diesem Fall entscheidet er sich aufgrund der hohen Vielfalt an Schwachstellen und der guten Dokumentation für die CVE in Kombination mit der NVD als Quelle für die Base Metrics der CVSSv2 Bewertung und beauftragt Hermann Baum, einen der Quellenverantwortlichen, mit der Implementierung des Datenimports und einer geeigneten Abbildung der in der NVD zur Verfügung gestellten Parameter zur Bewertung – in diesem Fall der Base Metrics – auf das im Hochschulnetz genutzte Bewertungssystem. Einen Vorschlag zur Abbildung meldet der Quellenverantwortliche an den Chief Vulnerability Manager, welcher schließlich vor dem aktiven Einsatz die Eignung der Abbildung einschätzen und seine Zustimmung geben muss.

Als Abbildung der Base Metrics auf das im Konzept beschriebene System wird der BaseScore als Grundlage verwendet und wie in Tabelle 5.6 des Konzeptkapitels auf die Kritikalitätsstufen *gering*, *mittel*, *hoch* und *kritisch* abgebildet.

Die Schwachstellen-Dokumentation wird sowohl um die Implementierung der Fremdquelle als auch um die Abbildung der Base Metrics des CVSS Version 2 durch Hermann Baum erweitert.

Die Implementierung umfasst Funktionen zum automatischen Import und der Dokumentation der Schwachstellen aus der CVE, die Anwendung der Abbildung des des Bewertungssystems aus Fremdquellen auf das hochschulnetzweite Bewertungssystem und Mechanismen zur Vermeidung von Duplikaten.

Die konkrete Quelle für die Schwachstellen bildet die von der MITRE Corporation bereitgestellte Datei im CSV-Format `https://cve.mitre.org/data/downloads/allitems.csv`, sowie eine Quelle zur Bewertung und Kategorisierung in Form der NVD `https://web.nvd.nist.gov/view/vuln/detail?vulnId=`, welcher über die URL die CVE Nummer der jeweiligen Schwachstelle übergeben und das in HTTP zurückgelieferte Ergebnis entsprechend geparst wird (siehe allgemeine Beschreibung einer Implementierung in Abschnitt 6.3.3).

Da die CVE eine Pull-Quelle ist, d.h. in bestimmten Zeitabständen abgefragt wird, und die zur Verfügung gestellten Daten in der Regel im Abstand weniger Tage aktualisiert werden, wird das Abfrageintervall der automatischen Aktualisierung vom zuständigen Quellenverantwortlichen auf 3 Tage festgelegt.

7.4 Erstes Fallbeispiel

Ab hier sind bis dahin notwendige Nutzer und Rollen, Fremdquellen sowie Assets definiert, wodurch nun der Aktivitätenzyklus aus Sicht von Schwachstellen betrachtet wird und entweder automatisiert oder manuell durchlaufen werden kann.

In diesem Fall wird eine Schwachstelle betrachtet, die über die CVE identifiziert und mit Hilfe von Daten aus der NVD klassifiziert wird. Der automatisierte Import deckt insofern bereits zwei der Aktivitäten im Schwachstellenmanagement ab.

7.4.1 Importieren von Schwachstellen

Im Szenario wird eine automatische Ausführung des Imports aus den im Abschnitt 7.3 genannten Quellen – der CVE sowie der NVD – durch den Scheduler im Steuerungssystem ausgeführt, wodurch mehrere neue Schwachstellen dokumentiert werden. Darunter befindet sich auch die Schwachstelle mit der CVE-ID „CVE-2015-4872“, deren Beschreibung in der Schwachstellen-Dokumentation in Tabelle 7.3 sowie deren Referenzen aus der CVE in Tabelle 7.4 gezeigt werden. Letztere Referenz ist jedoch nicht aus der CVE importiert , sondern ein Verweis auf die Fremdquelle für die Bewertung und Kategorisierung – die NVD. In Tabelle 7.5 sind die in der Schwachstellen-Dokumentation hinterlegten und aus der NVD importierten, von der Schwachstelle betroffenen Softwareprodukte, zusammengefasst.

Feld	**Wert**
ID	6939
Beschreibung	Unspecified vulnerability in Oracle Java SE 6u101, 7u85, and 8u60; Java SE Embedded 8u51; and JRockit R28.3.7 allows remote attackers to affect integrity via unknown vectors related to Security.
Bewertung	mittel
Temporäre Dringlichkeit	NA
Temporäre Dringlichkeit Zeitstempel	
Fremdquelle	CVE
Fremdquellen ID	CVE-2015-4872
Status	detect
Geändert	2015-10-31 16:07:52
Editor	0
Assetspezifisch	

Tabelle 7.3: Beschreibung der Schwachstelle mit der CVE-ID „CVE-2015-4872“ in der Schwachstellen-Dokumentation

Da die NVD bereits sowohl ausreichend Informationen zur Bewertung (der BaseScore des CVSS Version 2) sowie zur Kategorisierung (der Auszug relevanter Einträge aus der CPE) liefert, ist der Status der Schwachstelle bereits automatisch auf *detect* gesetzt. Dadurch, ausgelöst durch die Änderung des Status der Schwachstelle, weist das Steuerungssystem die Verantwortlichkeit zur Erstellung eines Detektionsverfahrens automatisch einem Detektionsverantwortlichen im Hochschulnetz zu.

Durch ausreichend Informationen aus Fremdquellen kann auf diese Weise die Aktivität der Identifikation sowie der Klassifikation der Schwachstelle bereits automatisiert innerhalb des Imports abgearbeitet werden, sodass die Schwachstelle praktisch in der Aktivität der Beseitigung und Abschwächung, aber auch bereits in der Aktivität der Prävention sinnvoll verwendet werden kann.

Quelle	Quellen-ID
CONFIRM	`http://www.oracle.com/technetwork/topics/security/alerts-086861.html`
NVD	`https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2015-4872`

Tabelle 7.4: Referenzen der Schwachstelle mit der CVE-ID „CVE-2015-4872“

Hersteller	Bezeichner	Version
oracle	jre	1.6.0:update_101
oracle	jdk	1.6.0:update_101
oracle	jre	1.7.0:update_85
oracle	jdk	1.7.0:update_85
oracle	jre	1.8.0:update_60
oracle	jre	1.8.0:update_60
oracle	jre	1.8.0:update_51
oracle	jre	1.8.0:update_51
oracle	jrockit	r28.3.7

Tabelle 7.5: Von der Schwachstelle „CVE-2015-4872“ betroffene Softwareprodukte [Nat15e]

7.4.2 Beseitigung und Abschwächung

Zu diesem Zeitpunkt ist die Schwachstelle bereits automatisch identifiziert, bewertet und kategorisiert. Da keine zuverlässige Bezugsquelle von Detektionsverfahren angegeben ist, erfolgt die Teilaktivität der Entwicklung eines Detektionsverfahrens für die Schwachstelle manuell.

7.4.2.1 Erstellung eines Detektionsverfahrens

Auch wenn jede Schwachstelle einem Detektionsverantwortlichen zugewiesen wird, so kann allgemein jedoch jeder Detektionsverantwortliche ein Detektionsverfahren für jede Schwachstelle erstellen. Alle nicht-detektierbaren Schwachstellen werden entsprechend auf dem Portal

des Steuerungssystems für alle Detektionsverantwortlichen aufgelistet, sodass diese auch andere Einträge bearbeiten können. Detektionsverantwortliche können zudem auch für bereits detektierbare Schwachstellen weitere, ergänzende Detektionsverfahren erstellen.

Das Steuerungssystem hat die Verantwortlichkeit der Erstellung eines Detektionsverfahrens dem Detektionsverantwortlichen Günter Reinhard zugewiesen. Dieser wird über die Zuweisung per E-Mail informiert und kann die Schwachstelle auf dem Webportal des Steuerungssystems einsehen.

Als Detektionsverfahren entscheidet sich Günter Reinhard für einen systembasierten Ansatz der Detektion durch Abgleich der installierten Softwareprodukte auf dem Asset mit den von der Schwachstelle betroffenen Softwareprodukten, welche zum einen aus der Schwachstellenbeschreibung auslesbar, als auch bereits in der Teilaktivität der Kategorisierung für die Schwachstelle exakt dokumentiert wurden. Aufgrund der Vorgaben durch den Chief Vulnerability Manager sind Implementierungen von Detektionsverfahren in Perl zu realisieren.

Um eine möglichst schnelle Detektion der Schwachstelle zu gewährleisten, plant Günter Reinhard, zunächst in einer ersten Version des Detektionsverfahrens, die Detektion für die im LRZ zu schützenden SUSE-Linux Server zu ermöglichen. Eine Erweiterung der Implementierung um die Funktionalität zur Detektion auf Windows-Systemen sowie weiteren Linux-Distributionen plant er, in den darauf folgenden Tagen durchzuführen.

Die Abfrage der installierten Softwarepakete setzt Günter Reinhard in Form eines Aufrufs des *RPM Package Managers* mit den Argumenten „-qa“ um. Ein Auszug der Ausgabe bei Ausführung auf einem System mit *SUSE Linux Enterprise Server 11* ist in Listing 7.1 zu sehen.

Listing 7.1: Beispiel einer Teilausgabe bei Ausführung des Befehls „rpm -qa“

```
1  ...
2  pesign-0.99-0.31.12
3  brocade-firmware-3.1.2.1-0.11.4
4  facter-1.6.18-0.3.1
5  libcroco-0_6-3-32bit-0.6.1-120.16
6  libnsssharedhelper0-1.0.10-0.7.33
7  xorg-x11-libXfixes-devel-7.4-1.16.2
8  perl-Tk-804.028-50.24
9  libdb-4_5-devel-4.5.20-95.39
10 libesd0-32bit-0.2.41-3.1.41
11 libmcrypt-2.5.8-43.21
12 startup-notification-32bit-0.9-60.13
13 obex-data-server-0.4.6-2.7.90
14 ...
```

Den Abgleich mit von der Schwachstelle betroffenen Softwarepaketen realisiert er durch eine Textsuche der Ausgabe mit Hilfe der regulären Ausdrücke in Perl ohne Beachtung der Groß- und Kleinschreibung. Falls mindestens eines der Pakete in der Auflistung vorkommt, gibt das implementierte Detektionsverfahren „true“ in dem Standardausgabekanal (STDOUT) aus, andernfalls „false“. Die Implementierung in Perl ist in Listing 7.2 gezeigt.

Listing 7.2: Beispielimplementierung zur Detektion der Schwachstelle mit ID 6939 aus Systemsicht auf einer SLES-Plattform

```
#! /usr/bin/perl


# Abfragen des Befehl "rpm -qa" und Speichern der Ausgabe in $out
my $out = `rpm -qa`;

# Auftrennen der einzelnen Einträge am Zeilenumbruch
my @lines = split(/\n/, $out);

# Standard Annahme ist, dass das Asset nicht von der Schwachstelle
# betroffen ist
my $result = "false";

# Abgleich der einzelnen Einträge der Ausgabe:
# Falls einer der Zeilen der Ausgabe von "rpm -qa" mit einem der in
# Stringform dargestellen Softwareprodukte beginnt, ist das
# jeweilige Paket installiert.
# Eine Ausnahme bildet die Abfrage des Pakets jrockit in
# betroffnener Version r28.3.7:
# Falls eine Zeile beide Stichworte erfüllt, ist das Paket als
# installiert anzusehen.
for my $i(0..$#lines)
{
        if ($lines[$i] =~ /^jre1\.6\.0_101/i # case-insensitive
        || $lines[$i] =~ /^jdk1\.6\.0_101/i
        || $lines[$i] =~ /^jre1\.7\.0_85/i
        || $lines[$i] =~ /^jdk1\.7\.0_85/i
        || $lines[$i] =~ /^jre1\.8\.0_51/i
        || $lines[$i] =~ /^jdk1\.8\.0_51/i
        || $lines[$i] =~ /^jre1\.8\.0_60/i
        || $lines[$i] =~ /^jdk1\.8\.0_60/i
        || (    $lines[$i] =~ /jrockit/i
            && $lines[$i] =~ /r28\.3\.7/i))
        {
                # $result wird auf "true" gesetzt, falls
                # mindestens eines der von der Schwachstelle
                # 6939 betroffenen Softwareprodukte auf
                # dem System installiert ist.
                $result = "true";
        }
}
# Ausgabe des Ergebnisses
print $result;
```

Nach der eigentlichen Implementierung testet Günter Reinhard diese auf ihre Funktions- und Zweckmäßigkeit. Auch wenn ihm kein Computersystem mit einem von der Schwachstelle betroffenen Softwarepaket zur Verfügung steht, nutzt er ein für derartige Zwecke vorhandenes, identisches Testsystem und speichert die Ausgabe der Abfrage der installierten Softwarepakete durch Ausführung der Befehle „rpm −qa > out.txt“ in einer Textdatei.

Da auf dem Testsystem weder die JRE noch das JDK oder JRockit installiert sind, findet sich in der gespeicherten Ausgabe kein entsprechender Eintrag. Daher lädt er aus dem Archiv von Oracles Java Implementierungen (`http://www.oracle.com/technetwork/java/javase/downloads/java-archive-downloads-javase7-521261.html`) das von der Schwachstelle betroffene *.rpm*-Paket der JRE 1.7.0 Update 85 herunter und installiert es auf dem Testsystem, wobei bei Ausführung von „rpm −qa“ nun der in Listing 7.3 gezeigte Eintrag in der Ausgabe auftaucht.

Listing 7.3: Relevante Ausgabe von „rpm -qa“

```
1 jre1.7.0_85-fcs.x86_64
```

Eine manuelle Ausführung des Programms bestätigt die Funktionsfähigkeit und seine Zweckdienlichkeit zur Detektion der Schwachstelle durch Abgleich der installierten Softwarepakete auf dem Asset mit von der Schwachstelle betroffenen Softwarepaketen. Aufgrund der einfachen Implementierung und Nutzung des vorinstallierten Programms *rpm* zur Detektion, das für die Auflistung installierter Pakete lediglich Benutzerrechte erfordert, sowie des Testlaufs, bei dem keine Probleme aufgetreten sind, sieht Günter Reinhard das Detektionsverfahren als sicher und anwendbar an. Nach erfolgreichem Test deinstalliert Günter Reinhard das von der Schwachstelle betroffene Softwarepaket auf dem Testsystem wieder.

Insofern fährt Günter Reinhard mit der Dokumentation des Detektionsverfahrens fort und erstellt über das Webportal einen Eintrag in der Schwachstellen-Dokumentation. Der von ihm erstellte Eintrag ist in Tabelle 7.6 dargestellt.

Günter Reinhard verfasst eine kurze Beschreibung der Schwachstelle, in der er die Vorgehensweise zur Detektion erläutert, gibt Abhängigkeiten bezüglich weiterer Softwareprodukte und Bibliotheken an, in diesen Fall lediglich den „RPM Package Manager“ und den Compiler bzw. Interpreter, mit dem er das Detektionsverfahren getestet hat. Zudem listet er Plattformen auf, auf denen er das Detektionsverfahren erfolgreich getestet hat (SLES 11) und somit die Funktionsfähigkeit bestätigen kann.

Die Datei mit der Implementierung wird von Günter Reinhard über das Portal des Steuerungssystems zur Ablage hochgeladen und innerhalb der Schwachstellen-Dokumentation gespeichert, wodurch die Dokumentation des Detektionsverfahrens vollständig ist und das Steuerungssystem automatisch die Verantwortung zur Begutachtung des Eintrages an einen Detektionsverantwortlichen überträgt, in diesem Fall an Herbert Grün, welcher eine Benachrichtigung darüber per E-Mail bekommt.

Bei Öffnen des Portals des Schwachstellenmanagements wird Herbert Grün der Eintrag über das von Günter Reinhard erstellte Detektionsverfahren mitsamt der Implementierung angezeigt. Aufgrund des kurzen Quelltextes kann Herbert Grün bereits durch Nachvollziehen

Feld	Wert
ID	21
Schwachstelle	6939
Beschreibung	Detektion per Abgleich der installierten Softwarepakete auf dem Asset mit Softarepaketen, die von der Schwachstelle betroffen sind. Das Programm wird auf einem Asset mit geeigneter Plattform ausgeführt und benötigt keine weiteren Parameter als Eingabe.
Typ	system
Editor-ID	93
Abhängig von	RPM Package Manager
Compiler/ Interpreter	Perl 5.18.2
Plattformen	SUSE Linux Enterprise Server 11
Abnahme durch	
Freigegeben	

Tabelle 7.6: Dokumentation des Detektionsverfahrens zur Schwachstelle mit ID=6939

des Quelltextes eine Beeinträchtigung bzw. böswillige Absicht ausschließen und gibt das Detektionsverfahren über das Portal frei, wodurch die dazugehörige Schwachstelle automatisch den Status *complete* bekommt.

Die Aktualisierung von Detektionsagenten sowie Detektionssystemen wird (durch Abstimmung des Schedulers des Steuerungssystems) in den wöchentlichen Wartungszeiten angestoßen, wodurch Steuerungssysteme sowohl Implementierungen von Detektionsverfahren aus Netzsicht sowie aus Systemsicht auf alle Detektionssysteme über das Netz kopieren. Detektionsverfahren aus Netzsicht dienen der Aktualisierung der Detektionssysteme selber, hingegen Detektionsverfahren aus Systemsicht werden von den Detektionssystemen an Detektionsagenten mit geeigneter Plattform über das *Secure Copy*-Protokoll (scp) übertragen.

Vor dem Kopieren der Dateien werden, wie im der Funktion `update` des Detektionssystems in Abschnitt 6.6.2 beschrieben, zunächst alle alten Detektionsverfahren sowohl auf Detektionssystemen als auch Detektionsagenten entfernt.

7.4.2.2 Detektion der Schwachstelle

Das Detektionsintervall des Webservers wurde durch den hauptverantwortlichen Detektionsverantwortlichen, Martin Wolke, nicht geändert, ist also auf den Standardwert von zehn Tagen festgelegt. Nach Ablauf der zehn Tage seit dem letzten automatischen Detektionsdurchlauf, wird demnach – ausgelöst durch den Scheduler des Steuerungssystems – der aktuelle Durchlauf gestartet. Eine Illustration des Ablaufs der Detektion mit den entsprechenden Funktionsaufrufen auf jeweiligen Komponenten ist in Abbildung 7.1 gezeigt.

Bevor das Steuerungssystem die Durchführung der Detektion startet und an eines der Detektionssysteme weitergibt, werden alle Schwachstellen, die nicht relevant sind, aussortiert. Dazu berechnet das Steuerungssystem aus der Kritikalität jeder detektierbaren Schwachstelle sowie der des Assets, die sich ergebende Behebungsnotwendigkeit und gleicht sie mit dem

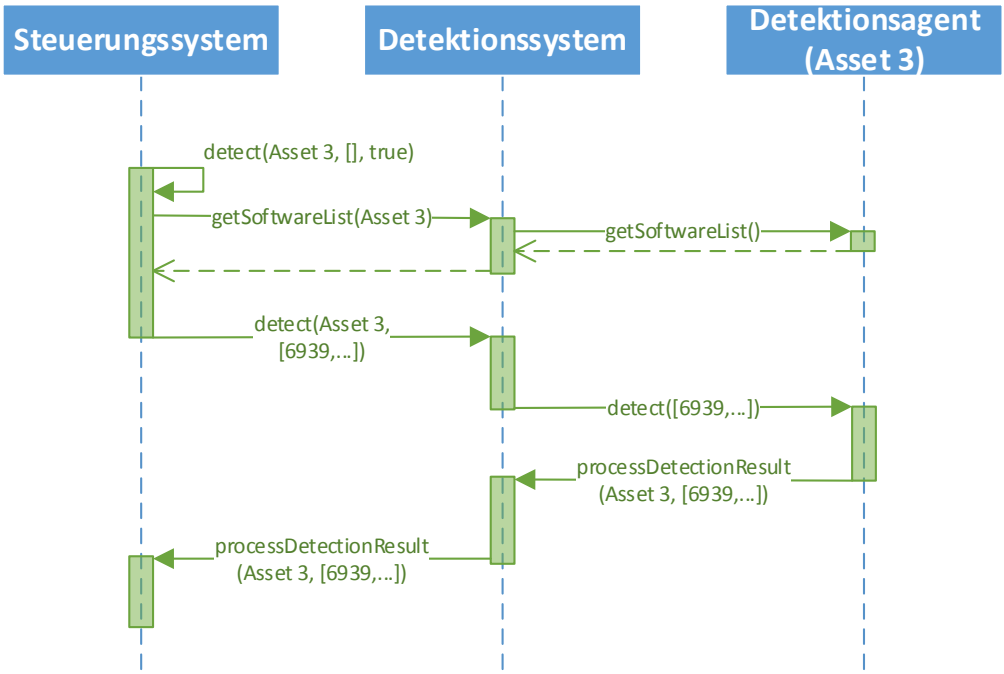


Abbildung 7.1: Sequenzdiagramm der Durchführung der Detektion

Schwachstellenakzeptanzkriterium ab. Da das Schwachstellenakzeptanzkriterium auf *gering* gesetzt ist, werden alle Schwachstellen mit Behebungsnotwendigkeit *mittel*, *hoch* oder *kritisch* auf einem Asset detektiert. Zusätzlich wird ein Abgleich der von der Schwachstelle betroffenen Software mit der auf dem Asset installierten Software durchgeführt. Da die resultierende Behebungsnotwendigkeit der Schwachstelle auf dem Webserver einen Wert von *hoch* hat, ist sie kritischer als der vom Schwachstellenakzeptanzkriterium festgelegte Wert und somit diesbezüglich relevant.

Anhand des Beispiels der betrachteten Schwachstelle ergibt die Auswertung von Seiten des Steuerungssystems, dass das Softwareinventar des Webservers ein JRE beinhaltet. Insofern ist die Schwachstelle bezogen auf die Detektion auf dem Webserver relevant.

Das Steuerungssystem wählt nach der Filterung relevanter Schwachstellen zufällig eines der Detektionssysteme aus, da der Webserver selbst auch unter einer öffentlichen IP-Adressen angesprochen werden kann, und ruft auf diesem die in Listing 7.4 gezeigte Funktion auf.

Listing 7.4: Aufruf der Funktion zur Detektion relevanter Schwachstellen auf dem Webserver auf dem Detektionssystem

```
{
    "FUNCTION": "detect",
    "PARAM": {
        "ASSET": {
            "ID": 3,
            "NETZID": "webserver01.lrz.de"
        },
        "SCHWACHSTELLENIDS": [
            ..., 6939, ...
        ]
    }
}
```

Infolgedessen führt das Detektionssystem zutreffende Detektionsverfahren für die im Funktionsaufruf aufgelisteten Schwachstellen aus und ruft gleichzeitig die in Listing 7.5 gezeigte Funktion zur Detektion auf dem unter der Adresse „webserver01.lrz.de“ ansprechbaren Detektionsagenten auf. Dieser führt ebenfalls das entsprechende Detektionsverfahren aus.

Listing 7.5: Aufruf der Funktion zur Detektion auf dem Detektionsagenten auf der Maschine des Webservers

```
{
    "FUNCTION": "detect",
    "PARAM": [..., 6939, ...]
}
```

Da die Implementierung des Detektionsverfahrens (in diesem Fall auf dem Detektionsagenten die ausführbare Perl-Datei *6939_21*) per Mustererkennung einen Treffer des Eintrages „jre1.8.0_51“ verzeichnet, gibt es „true“ zurück, wodurch der Detektionsagent den Identifikator der Schwachstelle *6939* in die Ergebnisliste mit aufnimmt. Insgesamt meldet ein Detektionsagent durch, den in Kurzform dargestellten, Funktionsaufruf *processDetectionResult(3, [6939,...])* eine Liste der Identifikatoren aller auf dem Asset positiv detektierten Schwachstellen an das Detektionssystem zurück. Dieses gleicht das Ergebnis aus Systemsicht mit dem Ergebnis der Detektion aus Netzsicht ab und entfernt doppelte Einträge. Im Anschluss daran ruft das Detektionssystem die Funktion *processDetectionResult(3, [6939,...])* auf dem Steuerungssystem auf, welches – da noch kein Schwachstellenticket für Asset mit ID=3 und der Schwachstelle (ID=6939) existiert – ein Schwachstellenticket erstellt (siehe Tabelle 7.7)

Das Schwachstellenticket stellt den Zusammenhang zwischen einer Schwachstelle und einem Asset her, ausgedrückt durch die jeweiligen Identifikatoren im Schwachstellenmanagement. Die Behebungsnotwendigkeit wird automatisch zentral durch das Steuerungssystem bei Erstellung des Schwachstellentickets aus der Kritikalität der jeweiligen Schwachstelle

Feld	Wert
ID	35
Schwachstelle-ID	6939
Asset-ID	3
Behebungsnotwendigkeit	hoch
Workaround	
Maßnahme	
Erstellt	2015-11-03 15:15:26
Geschlossen	
Eskalationslevel	0

Tabelle 7.7: Schwachstellenticket der Schwachstelle mit ID=6939 auf Webserver mit ID=3

(*mittel*) sowie der Kritikalität des betroffenen Assets (*hoch*) errechnet. So ergibt sich eine Behebungsnotwendigkeit des Schwachstellentickets mit dem Wert *hoch*.

Die Verantwortung zur Behebung des Schwachstellentickets wird bei Erstellung automatisch durch das Steuerungssystem an den hauptverantwortlichen Behebungsverantwortlichen des Webservers zugewiesen, Maria Mustermann.

7.4.2.3 Behebung und Kontrollprüfung

Das Fallbeispiel überschneidet sich ab hier mit dem Fallbeispiel im folgenden Abschnitt. Insofern wird die Behebung sowie Kontrollprüfung und Eskalation dieser Schwachstelle in Abschnitt 7.5.3.2 beschrieben.

7.4.3 Prävention

In diesem Szenario ist als präventive Maßnahme die Nutzung der Informationen aus dem Schwachstellenmanagement im Change Management vorgesehen.

In diesem Fall will Reinhold Maurer von der Ludwig-Maximilians-Universität (LMU) ein neues im Rahmen einer studentischen Arbeit entwickeltes Softwaresystem zur Verwaltung von Klausurergebnissen produktiv führen. Dieses basiert auf Java und kann pro Veranstaltung Statistiken inklusive Diagramme der Notenverteilung erstellen und darüber hinaus die durchschnittlich erreichte Note aller Veranstaltungen und aller Klausuren eines Lehrstuhl errechnen.

Die Produktivführung wird auch an der LMU über ein institutsweites Change Management abgewickelt, in dem Reinhold Maurer zunächst zwei „Request for Changes“ (RfC) erstellt – einen bezüglich der Installation der erwähnten Java-Anwendung, den anderen bezüglich der Plattform, welche er auf dem entsprechenden Server in Form der *Java Runtime Edition 1.7.0 Update 85* realisieren will. Beide RfCs werden innerhalb weniger Tage autorisiert.

Da auch in der LMU seit Einführung des Schwachstellenmanagements festgelegt ist, dass das Testen von geplanten Änderungen im Change Management ebenfalls die Berücksichtigung von Schwachstellen miteinbezieht, meldet er sich auf dem Webportal des Steuerungssystems an. Reinhold Maurer kann als Netznutzer alle Schwachstellen (inklusive Maßnahmen) einsehen, die nicht assetspezifisch sind. Durch Nutzung der Suchfunktion der Schwachstellen-

Dokumentation und der Suche nach der JRE 1.7.0 Update 85, wird ihm unter anderen die importierte Schwachstelle mit ID=6939 angezeigt.

Da bereits eine Maßnahme zur Behebung der Schwachstelle existiert, sieht Reinhold Maurer diese ein und erfährt, dass es sich um ein Softwareupdate handelt. Insofern beauftragt er vor Durchführung einer Installation der JRE 1.7.0 Update 85, den Studenten, der die Java Anwendung zur Verwaltung von Klausurergebnissen entwickelt hat, mit dem Test auf Funktionsfähigkeit der Anwendung unter der neuesten Version des JRE 1.8.0 (Update 66).

Nachdem alle Funktionen ebenfalls in der neuesten Version funktionieren, schließt Reinhold Maurer den Change zur Installation der JRE 1.7.0 Update 85 und eröffnet einen RfC zur Installation der Version JRE 1.8.0 Update 66.

Da hier aktuell deutlich weniger Schwachstellen bekannt sind wird die Installation dieser Version durchgeführt, wodurch Reinhold Maurer die geplante Anwendung zur Verwaltung von Klausurergebnissen in Betrieb nehmen kann.

7.5 Zweites Fallbeispiel

In diesem Fallbeispiel wird der Durchlauf durch das System anhand einer manuell gemeldeten Konfigurationsschwachstelle bis zur Teilaktivität der Detektion durchlaufen. Ab der Teilaktivität der Behebung werden die beiden Schwachstellen dieses Fallbeispiels und dem des vorherigen Abschnitts zusammen betrachtet.

7.5.1 Identifikation

Auf eine generell nicht in der CVE oder sonstigen Quelle zu findende Schwachstelle wird in diesem Beispiel eine Mitarbeiterin eines Instituts im Münchner Wissenschaftsnetz aufmerksam, als sie sich über eine vom LRZ zur Verfügung gestellte Website für einen der vom LRZ angebotenen Kurse anmelden will. Die Anmeldung zum Kurs erfordert die Eingabe einiger persönlicher Daten wie Name, Anschrift sowie Angaben zur Zugehörigkeit zu einem der Institute – personenbezogene Daten, die bei Abschicken des Web-Anmeldeformulars unverschlüsselt vom Webclient an den Webserver im LRZ übertragen werden.

Auch wenn die Mitarbeiterin lediglich die Rolle des *Nutzer* im Schwachstellenmanagement im Hochschulnetz einnimmt, so hat sie erfahren, dass das LRZ eine Weboberfläche – ähnlich zur Meldung von Vorfällen („Incidents“) sowie Anfragen zu Diensten („Service Requests“) – anbietet und findet den auf die Weboberfläche zeigenden Link auf der Webpräsenz des LRZ. Nach Anmeldung auf dem Webportal bekommt sie über ihre Nutzerkennung und das dazugehörige Passwort, aufgrund dessen, dass sie keine weitere Rolle im Schwachstellenmanagement innehat, lediglich das Formular zur Meldung einer Schwachstelle, die Funktion zum Suchen von Schwachstellen und den Scan-Selfservice angezeigt.

Auf der Weboberfläche zur Meldung einer Schwachstelle werden vier Freitext-Felder angezeigt: Zum einen ein Feld zur Beschreibung des betroffenen Produktes inklusive Hersteller und genauer Version oder alternativ der von der Schwachstelle betroffene Dienst. Das zweite Feld dient zur Beschreibung des Vorgehens zur Ausnutzung bzw. Auslösung der Schwachstelle und das dritte Feld zur Beschreibung des hervorgerufenen Verhaltens. Das vierte Feld

dient zur Angabe einer E-Mail-Adresse, über die bei Bedarf der Kontakt zum Schwachstellen-Melder für Nachfragen hergestellt wird.

Da die Schwachstellen-Melderin nicht sicher weiß, wie der Dienst bezeichnet wird, gibt sie im ersten Feld, der Beschreibung des betroffenen Softwareproduktes die URL zur Anmeldung bei genanntem Kurs an. Bei der Beschreibung der Vorgehensweise gibt sie an, dass sie auf der Seite der Kursanmeldung ihre Daten in die vorgegebenen Felder eingegeben und darauf den „Abschicken“ Button angeklickt hat. Im Feld des hervorgerufenen Verhaltens gibt die Mitarbeiterin an, dass die Daten offenbar über das HTTP Protokoll übertragen werden und die Verbindung zwischen ihrem Browser und dem Webserver entsprechend nicht verschlüsselt ist. Als Kontakt gibt sie eine von ihr oft genutzte E-Mail-Adresse an.

Die Meldung wird vom Steuerungssystem in die Schwachstellen-Dokumentation mit einem aktuellen Zeitstempel zur Öffnung des Tickets eingetragen. Bei Öffnung des Meldungstickets wird es automatisch durch das Steuerungssystem einem der in der Asset- und Benutzerverwaltung eingetragenen Schwachstellen-Dokumentatoren zugewiesen. Gleichzeitig wird an den jeweiligen Schwachstellen-Dokumentator über die hinterlegte E-Mail-Adresse eine Nachricht als Hinweis über die Zuweisung gesendet.

Das Meldungsticket wird an den Schwachstellen-Dokumentator und zugleich Schwachstellen-Prüfer, Joachim Holzmann, weitergeleitet, welcher die Benachrichtigung sieht und sich über das Steuerungssystem auf dem Webportal mit seiner Nutzerkennung anmeldet. Auf dem Webportal wird das offene Meldungsticket angezeigt und kann von ihm eingesehen werden. Nach der Prüfung auf Plausibilität, d.h. dass der Service bzw. das Softwareprodukt existiert sowie dass die Schwachstellenbeschreibung plausibel wirkt, eröffnet er nach Prüfung auf Duplikate mit Hilfe der Suchfunktion in der Schwachstellen-Dokumentation einen neuen Eintrag in der Schwachstellen-Dokumentation, welcher in Tabelle 7.8 dargestellt ist.

Feld	**Wert**
ID	7211
Beschreibung	Unverschlüsselte Übertragung von personenbezogenen Daten bei Kursanmeldung auf www.lrz.de/programmieren–mit–fortran/.
Bewertung	NA
Temporäre Dringlichkeit	NA
Temporäre Dringlichkeit Zeitstempel	
Fremdquelle	
Fremdquelle_ID	
Status	check
Geändert	2015-11-02 09:32:20
Editor-ID	17
Assetspezifisch	

Tabelle 7.8: Eintrag der gemeldeten Schwachstelle in der Schwachstellen-Dokumentation

Der einzige Wert, den Joachim Holzmann bei der Eintragung der Schwachstelle zunächst angeben muss, ist die Schwachstellenbeschreibung. Die übrigen Werte werden automatisch

eingetragen. Wichtig ist insbesondere das Feld „Status“, das bei Eintragung automatisch auf *check* gesetzt wird und signalisiert, dass die Schwachstelle noch nicht auf ihre tatsächliche Existenz überprüft wurde.

Aufgrund dessen, dass Joachim Holzmann nicht nur Schwachstellen-Dokumentator, sondern zugleich Schwachstellen-Prüfer ist, wird ihm als Ersterfasser die Eintragung und Notwendigkeit einer Überprüfung gemeldet. Da es sich um eine Schwachstelle in der Konfiguration eines Dienstes des LRZ handelt, eskaliert er die Verantwortung für die Verifikation der Schwachstelle funktional an einen der dem Dienst angehörenden Schwachstellen-Prüfer, Karl Blatt. Aus Sicht von Joachim Holzmann erfolgt die Eskalation in Form der Zuweisung der Verantwortung an einen registrierten Dienst. In diesem Fall den Dienst „Schulung und Kurse“, wodurch das Steuerungssystem die Verantwortung innerhalb der zum Dienst gehörenden Schwachstellen-Prüfer zuteilt.

Karl Blatt erhält die Benachrichtigung über die durchzuführende Verifikation der Schwachstelle und überprüft sie, indem er die von der Nutzerin beschriebene Vorgehensweise nachvollzieht. Dazu ruft er die von ihr angegebene URL `www.lrz.de/programmieren-mit-fortran/` auf und überprüft, ob die Daten über HTTP ohne Verwendung von TLS übertragen werden, was hier tatsächlich der Fall ist. Daher erweitert er die Beschreibung der Schwachstelle in der Schwachstellen-Dokumentation um die generelle Möglichkeit und Folgen der Ausnutzung der Schwachstelle – auch, um ausreichend Informationen für die Bewertung der Schwachstelle bereitzustellen. Da es sich um eine assetspezifische Schwachstelle handelt, d.h. in diesem Fall genau einen Webserver betrifft, trägt Karl Blatt als einer der Dienstbetreiber zusätzlich den Namen des betroffenen Servers ein.

> „Unsichere Übertragung von schutzbedürftigen, personenbezogenen Daten bei Kursanmeldung auf www.lrz.de/programmieren–mit–fortran/ auf Asset mit ID=3 (webserver01.lrz.de). Bei Abschicken der Anmeldung über den Button ‚Abschicken‘ werden die vom Nutzer eingegebenen Daten ohne Verwendung von TLS an den Webserver gesendet, wodurch diese von einem Angreifer über einen ‚Man-In-The-Middle‘ Angriff unberechtigt ausgelesen oder manipuliert werden können.“

Bei der Anpassung der Schwachstellen-Beschreibung wird gleichzeitig Karl Blatt als letzter Editor eingetragen. Über die Oberfläche im Steuerungssystem bestätigt Karl Blatt die Existenz der Schwachstelle, wodurch sich ihr Status auf *assess* ändert.

7.5.2 Klassifikation

Nachdem die Schwachstelle identifiziert ist und auf ihre Existenz überprüft wurde, sind die Teilaktivitäten der Identifikation abgeschlossen. Ab diesem Zeitpunkt erfolgt die Weiterverarbeitung der Schwachstelle in der Aktivität der Klassifikation.

7.5.2.1 Bewertung der Schwachstelle

Mit Setzen des Status der Schwachstelle der unverschlüsselten Übertragung personenbezogener Daten mit der ID *7211* (siehe Tabelle 7.8) auf *assess*, weist die Schwachstellen-Dokumentation den Eintrag automatisch einem der Schwachstellen-Bewerter im Schwachstellenmanagement zu, welcher den Eintrag ebenfalls auf der für ihn angepassten Oberfläche

auf den Steuerungssystemen einsehen und bewerten kann. Der Eintrag wird Julia Gitternetz zugewiesen, welche über die Zuweisung per E-Mail in Kenntnis gesetzt wird, woraufhin sie sich über das Webportal anmeldet. Bei Auswählen des Eintrages erscheint ein Webformular, das die im Konzeptkapitel in Abschnitt 5.6.1.1 in Tabelle 5.2 aufgelisteten Charakteristika mit den vorgegebenen Antworten ausfüllbar anzeigt.

Julia Gitternetz bewertet die Möglichkeit einer Ausnutzung über das Netz mit *ja*, da dies ein unbedingtes Charakteristikum eines Man-In-The-Middle-Angriffs, insbesondere bei der Verbindung zwischen Client und Server aus dem Internet in das MWN, ist. Den Aufwand der Ausnutzung schätzt sie als *gering* ein, da im ungünstigsten Fall lediglich ein Netzwerk-Sniffer wie *tcpdump* oder *wireshark* notwendig ist, um die übertragenen Daten abzugreifen. Da Julia Gitternetz im Web recherchiert und unter anderem im vom *Open Web Application Security Project* (OWASP) veröffentlichten Artikel „*OWASP TOP 10 – 2013 The Ten Most Critical Web Application Security Risks*“ [Ope15a] im Punkt A6, das Fehlen von SSL (bzw. TLS) als häufige Ursache für die Offenlegung sensibler Daten genannt ist, bewertet sie die Bekanntheit von Fällen der Ausnutzung mit *ja*. Aus der Schwachstellenbeschreibung gehen die Folgen einer Ausnutzung hervor: Die Beeinträchtigung der Vertraulichkeit und Integrität der übertragenen Daten, weshalb sie diese Aspekte mit *ja* bewertet, eine Beeinträchtigung der Verfügbarkeit mit *nein*.

Nach Bestätigung der Bewertung der Schwachstelle über die Weboberfläche des Portals ergibt sich aus den Daten und den in Abschnitt 5.6.1.1 definierten Funktionen zur Berechnung der Kritikalität einer Schwachstelle die Bewertung *kritisch*, welche in dem Eintrag vermerkt wird. Durch den Vorgang der Bewertung wird der Status der Schwachstelle auf *categorize* gesetzt.

7.5.2.2 Kategorisierung der Schwachstelle

Durch die Änderung des Status der Schwachstelle von *assess* zu *categorize*, weist das Steuerungssystem den Eintrag automatisch an einen der Kategorisierungsverantwortlichen zu. In diesem Fall an Jens Schubert, der ebenso wie die für die vorher durchgeführten Teilaktivitäten verantwortlichen Personen über E-Mail und als Benachrichtigung im Webportal informiert wird.

Da die Schwachstelle mit der ID *7211* die konkrete Konfiguration eines Webservers und vielmehr eine bestimmte Funktion – die Anmeldung zu einem Kurs – betrifft, ist die Kategorisierung genau dieses Eintrages in der Schwachstellen-Dokumentation nicht sinnvoll, da die Schwachstellenbeschreibung spezifisch auf genau ein einzelnes Asset zutrifft.

Eine Möglichkeit ist die Verallgemeinerung der Schwachstelle, indem die konkrete URL in der Schwachstellenbeschreibung herausgenommen wird und die verallgemeinerte Schwachstelle erneut manuell über ein Meldungsticket in den Prozess eingegeben wird. Ein beispielhafter Beschreibungstext für einen verallgemeinerten Eintrag dieser Schwachstelle könnte wie folgt lauten:

> „Übertragung schutzbedürftiger, personenbezogener Daten über das Netz ohne Verwendung von Mechanismen zur Sicherstellung der Vertraulichkeit und Integrität, wodurch ein Angreifer diese über einen Man-In-The-Middle-Angriff beeinträchtigen kann.“

Je nach Zweckdienlichkeit in der IT-Umgebung ist eine derartige Verallgemeinerung sinnvoll und notwendig; beispielsweise, falls diese Schwachstelle in der Konfiguration von Diensten weit verbreitet ist. Auf der anderen Seite ist diese Schwachstelle sehr allgemein und dadurch entsprechend schwierig zu kategorisieren, da sie prinzipiell auf jeden Datenaustausch mit schutzbedürftigen Daten angewendet werden und somit ein weites Spektrum an Softwareprodukten (z.B. Webserver, Datenbankserver, Verzeichnisdienste, ...) betreffen kann. In diesem exemplarischen Durchlauf wird die Dokumentation der Verallgemeinerung der Schwachstelle deshalb nicht durchgeführt.

Da das spezifische, betroffene Asset bereits eindeutig in der Schwachstellenbeschreibung identifiziert werden kann, kategorisiert der dafür Verantwortliche, Jens Schubert, die Schwachstelle nicht aufgrund betroffener Softwareprodukte, sondern ordnet sie als assetspezifisch konkret dem Webserver (ID=3) zu.

Nach Abschluss der Kategorisierung wird der Status der Schwachstelle auf *detect* geändert, da sie kategorisiert, jedoch nicht detektierbar ist. Die Einteilung der Verantwortung zur Erstellung eines Detektionsverfahrens wird üblicherweise automatisch vom Steuerungssystem einem zufällig gewählten Detektionsverantwortlichen zugewiesen, welcher für die Implementierung und Dokumentation verantwortlich ist. Da es sich in diesem Fall jedoch um eine assetspezifische Schwachstelle des Webservers handelt wird die Schwachstelle allen Detektionsverantwortlichen des Webservers im Webportal gemeldet. Die zu diesem Zeitpunkt zur Schwachstelle dokumentierten Informationen sind in Tabelle 7.9 zusammengefasst.

7.5.3 Beseitigung und Abschwächung

Bis hier ist die Konfigurationsschwachstelle in der Anwendung zur Kursanmeldung identifiziert, verifiziert, bewertet und kategorisiert. In dieser Aktivität wird allgemein die Detektion der Schwachstelle behandelt sowie, zusammen mit der Schwachstelle aus dem vorherigen Fallbeispiel in den Java-Implementierung, eine gemeinsame Behebung und Eskalation.

7.5.3.1 Erstellung eines Detektionsverfahrens und Detektion

Über die assetspezifische Schwachstelle, betreffend die fehlende Sicherung der Übertragung von schutzbedürftigen personenbezogenen Daten, wird der einzige Detektionsverantwortliche des betroffenen Assets informiert, Martin Wolke. Dieser muss entscheiden, ob die Entwicklung eines Detektionsverfahrens für die Schwachstelle sinnvoll ist und dem Aufwand entsprechend einen Nutzen bringt.

Da das betroffene Asset bereits in der Schwachstellenbeschreibung genannt und insofern dadurch detektierbar ist, entscheidet sich Martin Wolke gegen die Entwicklung eines Detektionsverfahrens und generiert manuell ein Schwachstellenticket mit Bezug zum betroffenen Asset. Das generierte Schwachstellenticket ist in Tabelle 7.10 zusammengefasst. Die Verantwortung zur Behebung wird Maria Mustermann als hauptverantwortlichen Behebungsverantwortlichen des Assets zugewiesen.

Aus der Kritikalität des Assets (*hoch*) sowie der Kritikalität der Schwachstelle (*kritisch*) folgt eine Behebungsnotwendigkeit von *kritisch*.

Feld	Wert
ID	7211
Beschreibung	Unsichere Übertragung von schutzbedürftigen, personenbezogenen Daten bei Kursanmeldung auf www.lrz.de/programmieren–mit–fortran/ auf Asset mit ID=3 (webserver01.lrz.de). Bei Abschicken der Anmeldung über den Button „Abschicken“ werden die vom Nutzer eingegebenen Daten ohne Verwendung von TLS an den Webserver gesendet, wodurch diese von einem Angreifer über ein „Man-In-The-Middle“ Angriff unberechtigt ausgelesen oder manipuliert werden können.
Bewertung	kritisch
Temporäre Dringlichkeit	NA
Fremdquelle	
Fremdquelle_ID	
Status	detect
Gelöst	false
Geändert	2015-11-03 13:14:31
Editor-ID	55
Assetspezifisch	3

Tabelle 7.9: Eintrag der gemeldeten Schwachstelle in der Schwachstellen-Dokumentation nach der Bewertung

7.5.3.2 Behebung der Schwachstellen

Im Szenario wurden die beiden Schwachstellen des aktuellen sowie vorherigen Fallbeispiels (Abschnitt 7.4) relativ zeitgleich bearbeitet und vor allem auf dem Webserver detektiert. Beide generierten Schwachstellentickets wurden Maria Mustermann als hauptverantwortliche Behebungsverantwortliche zugewiesen und werden daher bei ihrer nächsten Anmeldung auf dem Webportal zusammen aufgelistet. Zusätzlich kann sie die Schwachstelle sowie damit verknüpfte Referenzen, betroffene Software und eine Liste bereits vorhandener Maßnahmen zur Behebung der jeweiligen Schwachstelle im Webportal einsehen, wobei in diesem Fall jeweils noch keine Maßnahme zur Behebung dokumentiert ist. Eine beispielhafte Ansicht des Webportals ist in Abbildung 7.2 illustriert.

Die Schwachstellentickets sind nach jeweiliger Behebungsnotwendigkeit geordnet, das heißt, da das Schwachstellenticket mit ID=49, der unsicheren Übertragung schutzbedürftiger personenbezogener Daten mit *kritisch*, das Schwachstellenticket bezüglich der Java-Implementierungen (ID=35) hingegen mit *hoch* bewertet ist, werden die Tickets absteigend ihrer Kritikalität sortiert angezeigt, da die Bearbeitungsreihenfolge anhand dieser festgelegt ist.

Maria Mustermann fängt insofern mit der Ausarbeitung einer Maßnahme zur Behebung der am höchsten priorisierten Schwachstelle an. Da zu der manuell gemeldeten Schwachstelle die Lösung bereits in der Schwachstellenbeschreibung angedeutet wird – die Verwendung von TLS – sie jedoch kein Dienstbetreiber für den Dienst „Kurse und Schulung“ ist, erstellt sie in diesem Fall ein Service Request in Form eines Tickets des *Incident & Service Request Prozesses* und hat folglich die Bearbeitung funktional an die Verantwortlichen eskaliert.

Feld	Wert
ID	49
Schwachstelle-ID	7211
Asset-ID	3
Behebungsnotwendigkeit	kritisch
Workaround	
Maßnahme	
Erstellt	2015-11-04 18:04:59
Geschlossen	
Eskalationslevel	0

Tabelle 7.10: Schwachstellenticket der Schwachstelle in der Konfiguration (ID=7211) des Webservers (ID=3)

Als Asset- und Behebungsverantwortliche sowie ihrer Zugehörigkeit zum Dienst „Webhosting“ hat sie hingegen die Verantwortlichkeit und die Möglichkeit, das zweite Schwachstellenticket auf dem Asset selbst zu bearbeiten. Da auch hierzu keine Maßnahme dokumentiert ist, vergleicht sie zunächst auf Basis der Kategorisierung der Schwachstelle, welche Softwareprodukte betroffen sind sowie aus der Konfiguration des Webservers, welches Softwarepaket installiert ist – in diesem Fall die JRE von Oracle in der Version 1.8.0 Update 51.

Maria Mustermann informiert sich daraufhin über aktuelle Versionen der „Java Runtime Edition“ auf der Website des Herstellers und sieht, dass Oracle bereits JRE 1.8.0 Update 66 veröffentlicht hat.

Da im Leibniz-Rechenzentrum Änderungen an der Konfiguration von Assets und insbesondere *Configuration Items* über das Change Management abgewickelt werden, erstellt Maria Mustermann einen RfC bezüglich der Installation der Version 66 des JRE 1.8.0, wodurch die Bearbeitung im Prozess des Change Managements fortgesetzt wird.

Im Change Management Prozess wird der RfC bei Erfassung zunächst klassifiziert. Dazu gibt es drei verschiedene Klassifizierungen: Zum einen ein „Standard Change“, dessen Umsetzung standardmäßig autorisiert ist, ein „Minor Change“, dessen Auswirkungen und Risiko eher gering einzuschätzen ist sowie ein „Major Change“, wobei es sich um eine Änderung mit großer Auswirkungen handelt, von dem oft mehrere Dienste betroffen sind.

Die Klassifikation von RfCs wird durch den Ersteller, d.h. in diesem Fall Maria Mustermann, vorgenommen, welche die Installation des Updates der Java Runtime Edition als „Minor Change“ einstuft. Ein „Minor Change“ muss zunächst durch einen Dienstverantwortlichen autorisiert werden, bevor er umgesetzt werden kann. Da Maria Mustermann dem Dienst *Webhosting* angehört, autorisiert sie ihn direkt nach Erstellung.

Da der Webserver nicht ausschließlich von Maria Mustermann und der von ihr betriebenen Anwendung – der Webpräsenz des LRZ – genutzt wird, müssen in der auf die Autorisierung folgenden Testphase ebenfalls alle weiteren Nutzer des Assets miteinbezogen werden. Die Tests zielen darauf ab, in Erfahrung zu bringen, ob alle auf dem Webserver betriebenen Anwendungen nach dem Update immer noch funktionieren.

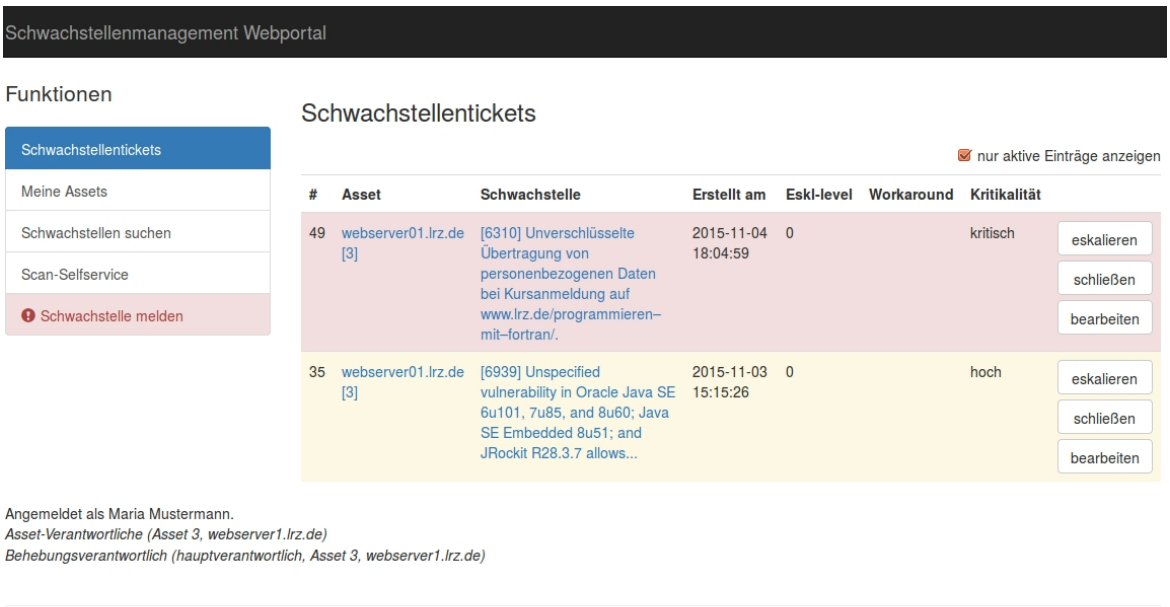


Abbildung 7.2: Beispielhafte Ansicht offener Schwachstellentickets auf dem Webportal

Daher benachrichtigt Maria Mustermann alle fünf weiteren Nutzer des Webservers, dass eine Aktualisierung der Java Runtime Edition geplant ist und sie eine Rückmeldung bezüglich Abhängigkeiten und möglichen Beeinträchtigungen benötigt. Während Maria Mustermann auf die Antworten wartet, nutzt sie die Zeit zur Dokumentation der Maßnahme: Sie erstellt entsprechend über das Webportal des Steuerungssystems einen Eintrag in der Schwachstellen-Dokumentation. Die Beschreibung der Maßnahme hält Maria Mustermann aufgrund dessen, dass mehrere Softwareprodukte betroffen sind – verschiedene Versionen der *JRE*, des *JDK* und zudem *JRockit* – eher allgemein:

„Aktualisierung der von der Schwachstelle betroffenen Java-Implementierungen.“

Bei der Wahl des Typs der Maßnahme gibt Maria Mustermann „Softwareupdate“ an, damit – falls weitere Maßnahmen von Behebungsverantwortlichen angelegt werden, diese zwecks einer besseren Übersichtlichkeit schneller bei deren Einsicht bei der Behebung einer bestimmten Art zugeordnet werden können.

Des Weiteren, da Maria Mustermann bereits eine URL zum Download einer neueren Softwareversion herausgesucht hat, gibt sie diese als Referenz für die Maßnahme an. Die Zusammenfassung des Inhalts der Maßnahme ist in Tabelle 7.11 dargestellt sowie die zur Maßnahme gehörigen Referenzen in Tabelle 7.12 gezeigt.

Da sich inzwischen alle der fünf weiteren Nutzer des Assets zurückgemeldet haben und vier davon ihre Anwendung komplett unabhängig einer JRE betreiben, beschränken sich die Tests auf die Webpräsenz des LRZ sowie eine Instanz eines *vhosts*, dessen betriebene Anwendung eine JRE benötigt.

Zu diesem Zweck erstellt Maria Mustermann eine Kopie des Webservers als Testsystem, bei dem sie die Aktualisierung der JRE 1.8.0 auf Update 66 durchführt. Nach der Installation

Feld	Wert
ID	11211
Schwachstelle	6939
Beschreibung	Aktualisierung der von der Schwachstelle betroffenen Java-Implementierung.
Typ	Softwareupdate
Editor	223

Tabelle 7.11: Beschreibung der Maßnahme zur Behebung der Schwachstelle in der Java-Implementierung (ID=6939)

Quelle	Quellen-ID
ORACLE	`http://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html`
ORACLE	`http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html`
ORACLE	`http://www.oracle.com/technetwork/java/javase/downloads/java-archive-downloads-javase7-521261.html`
ORACLE	`http://www.oracle.com/technetwork/java/javase/downloads/java-archive-downloads-javase6-419409.html`
ORACLE	`http://www.oracle.com/technetwork/java/javase/downloads/java-archive-downloads-jrockit-2192437.html`

Tabelle 7.12: Referenzen der Maßnahme mit ID=11211

prüft sie, ob die vom Testsystem zur Verfügung gestellte Webpräsenz des LRZ und alle darin implementierten Funktionen weiterhin funktionieren.

Zudem benachrichtigt sie den Betreiber der anderen Webanwendung, diese ebenfalls auf ein Funktionieren mit der aktualisierten Java-Version zu testen damit unerwünschte Auswirkungen ausgeschlossen werden können.

Da es keine Probleme gibt, weder bei der Webpräsenz des LRZ noch bei der weiteren Anwendung, die Java verwendet, wird die Aktualisierung der Version der JRE im Allgemeinen genehmigt und auch auf dem produktiven Webserver durch Maria Mustermann installiert.

Nach der Installation des Updates folgt ein weiterer Test, in diesem Fall nicht mehr auf mögliche negative Auswirkungen, sondern auf die Wirksamkeit der Maßnahme. Dazu führt Maria Mustermann auf ihrem Asset – dem Webserver – das Programm „rpm −qa | grep jre“ aus, welches alle installierten Versionen anzeigt. Die Ausgabe ist in Listing 7.6 gezeigt.

Listing 7.6: Relevanter Auszug der Ausgabe von „rpm -qa“

```
1 jre1.8.0_66-1.8.0_66-fcs.x86_64
```

Im Anschluss ruft sie das Webportal des Steuerungssystems auf und schließt das entsprechende Schwachstellenticket mit ID=49 (Tabelle 7.7) mit Angabe über die getroffene Maßnahme ab, wodurch im Feld „Geschlossen“ das aktuelle Datum eingetragen wird.

Das geschlossene Schwachstellenticket ist in Tabelle 7.13 gezeigt, wodurch der Eintrag aus der Liste aktiver (d.h. noch offener) Schwachstellentickets auf dem Webportal entfernt wird.

Feld	**Wert**
ID	35
Schwachstelle-ID	6939
Asset-ID	3
Behebungsnotwendigkeit	hoch
Workaround	
Maßnahme	11211
Erstellt	2015-11-03 15:15:26
Geschlossen	2015-11-06 10:49:59
Eskalationslevel	0

Tabelle 7.13: Schwachstellenticket der Schwachstelle mit ID=6939 auf Webserver mit ID=3 im abgeschlossenen Zustand

Von der, bezüglich der Priorisierung, ersteren Schwachstelle mit ID=7211 hingegen, der ungesicherten Kommunikation des Webservers über HTTP mit Clients bei der Kursanmeldung, hat Maria Mustermann noch keine Rückmeldung auf den Service Request erhalten.

7.5.4 Kontrollprüfung und Eskalation

Da es lediglich für die Schwachstelle in der Java-Implementierung (ID=6939) ein Detektionsverfahren gibt, kann auch nur diese automatisch detektiert werden. Die Schwachstelle in der Konfiguration des Webservers (ID=7211) muss hingegen manuell kontrolliert werden.

Nachdem daher am 07.11.2015 (d.h. drei Tage nach Erstellung) das Schwachstellenticket (ID=49, siehe Tabelle 7.10) der Konfigurationsschwachstelle in der Kommunikation zwischen Server und Clients noch immer offen ist und die Kriterien zur Eskalation in der Datei *escal.conf* erfüllt (Kritikalität ist *kritisch* und das Schwachstellenticket ist drei Tage in ungelöstem Zustand), wird der zuständige Detektionsverantwortliche des Assets, Martin Wolke, im Webportal informiert.

Dieser sieht die Schwachstelle erneut ein (vgl. Tabelle 7.9) und ruft mit seinem Webbrowser die in der Schwachstellenbeschreibung hinterlegte URL auf. Bei der Überprüfung stellt er fest, dass die Schwachstelle immer noch offen ist, woraufhin er dies ebenfalls im Webportal angibt. Aufgrund dessen wird das Schwachstellenticket durch das Steuerungssystem automatisch eskaliert – in diesem Fall an den Asset-Verantwortlichen, Maria Mustermann. Durch die Eskalation wird das Ekalationslevel des Schwachstellentickets zudem auf „1“ gesetzt.

Daraufhin fragt Maria Mustermann über das Ticketsystem des *Incident & Service Request Prozesses* (ISRP) für den von ihr geöffneten Service Request den aktuellen Stand der Bearbeitung nach, da sie bisher noch keine Antwort erhalten hat. Kurze Zeit später bekommt sie die Antwort, dass die Anfrage derzeit noch in Bearbeitung sei und sie informiert werde, sobald die Konfiguration des Webservers geändert ist.

Zwei Tage später bekommt Maria Mustermann schließlich die Benachrichtigung über die Einrichtung von TLS. Nach Aufruf der in der Schwachstellenbeschreibung hinterlegten URL,

`www.lrz.de/programmieren-mit-fortran/` zur Überprüfung der Effektivität der unternommenen Maßnahme stellt sie fest, dass die Übertragung wie im ISRP-Ticket gemeldet, nun über TLS stattfindet. Insofern schließt sie das dazugehörige Schwachstellenticket.

Die automatisierte Kontrollprüfung bezüglich des Schwachstellentickets (ID=35) der Schwachstelle in der Java-Implementierung wird nicht mehr durchgeführt, da das Schwachstellenticket bereits geschlossen ist.

7.5.5 Prävention

Maßnahmen zur Prävention für assetspezifische Schwachstellen sind in der Regel schwieriger zu ergreifen, da sie oft nicht auf weiteren Assets auftreten. Im hier gezeigten Beispiel könnten geeignete Maßnahmen jedoch beispielsweise in Form von Richtlinien zur Sicherung der Kommunikation der Dienste oder auch Schulungen diesbezüglich folgen.

7.6 Drittes Fallbeispiel – Detektion des Heartbleed Bugs

In diesem Fallbeispiel wird insbesondere auf die als *Heartbleed Bug* bekannt gewordene Schwachstelle mit CVE-ID „CVE-2014-0160“ und die Nutzung eines bereits im Netzwerk-Scanner *nmap* vorhandenen Tests zur Detektion der Schwachstelle über das Netz eingegangen. Das Ziel ist das Aufzeigen einer Möglichkeit zur Integration von bereits existierenden Schwachstellen-Scannern in die technischen Komponenten des Konzepts. Dieses Fallbeispiel verdeutlicht daher nicht die organisatorischen Abläufe, wodurch die Identifikation sowie Klassifikation nicht weiter betrachtet werden und Informationen zur Schwachstelle ähnlich wie im ersten Fallbeispiel (Abschnitt 7.4) automatisch aus der CVE sowie NVD importiert wurden.

Es resultiert der in Tabelle 7.14 beispielhaft erstellte Eintrag. Aufgrund der Kritikalität der Schwachstelle und der Fehlbeurteilung aus der NVD mit einem BaseScore von 5.0, ist die Temporäre Dringlichkeit der Schwachstelle auf *kritisch* gesetzt. Die Schwachstelle ist gemäß dem Inhalt der Beschreibung vollständig kategorisiert.

In diesem Beispiel wird angenommen, dass die Installation von nmap zur Standardkonfiguration jedes Detektionssystems gehört und insofern lediglich benötigte Skripte der *Nmap Scripting Engine* (NSE) hinzugefügt werden müssen, um sie nutzbar zu machen. Des Weiteren wird angenommen, dass alle dafür notwendigen Bibliotheken („tls.lua“ und „stdnse.lua“, jeweils aus `https://nmap.org/nsedoc/lib/`), ebenfalls bereits auf allen Detektionssystemen installiert sind.

Die Implementierung des Detektionsverfahrens für den Heartbleed Bug wird von einem beliebigen Detektionsverantwortlichen durchgeführt.

Der Detektionsverantwortliche lädt zunächst von der offiziellen Website (`https://svn.nmap.org/nmap/scripts/ssl-heartbleed.nse`) das LUA-Skript herunter, das im nmap durch die NSE ausgeführt wird, und legt es in einem Unterverzeichnis `res/nmapscripts/` des Implementierungsverzeichnisses in der Schwachstellen-Dokumentation ab. Dieses wird bei Aktualisierung der Detektionsverfahren in den Ordner `/srv/vulnetd/res/nmapscripts` jedes Detektionssystems abgelegt.

Laut Beschreibung [Gor15d] wird das Skript in nmap durch Ausführung des Programms mit Parametern „−p 443 −−script ssl-heartbleed <target>“ ausgeführt, wobei <target> die

Feld	Wert
ID	8819
Beschreibung	The (1) TLS and (2) DTLS implementations in OpenSSL 1.0.1 before 1.0.1g do not properly handle Heartbeat Extension packets, which allows remote attackers to obtain sensitive information from process memory via crafted packets that trigger a buffer over-read, as demonstrated by reading private keys, related to d1_both.c and t1_lib.c, aka the Heartbleed bug.
Bewertung	mittel
Temporäre Dringlichkeit	kritisch
Temporäre Dringlichkeit Zeitstempel	2015-11-26 16:02:25
Fremdquelle	CVE
Fremdquellen ID	CVE-2015-0160
Status	detect
Geändert	2015-11-24 14:51:12
Editor	0

Tabelle 7.14: Beschreibung der Heartbleed Bugs in der Schwachstellen-Dokumentation

Netz-ID des Assets ist. Angepasst an den Pfad des Skripts auf den Detektionssystemen und ohne Beschränkung auf Port 443 lautet der Befehl zur Ausführung daher „nmap −−script /srv/vulnetd/res/nmapscripts/ssl-heartbleed <target>“.

Der Detektionsverantwortliche erstellt nun das eigentliche Detektionsverfahren, das in Listing 7.7 gezeigt ist und gemäß dem Szenario in Perl implementiert wird.

Listing 7.7: Implementierung des Detektionsverfahrens bzgl. des Heartbleed Bugs unter Verwendung von *nmap*

```
1  #! /usr/bin/perl
2
3  # Übergabe der IP-Adresse des Zielassets
4  #
5  my $netid = $ARGV[0];
6
7  # Ausführen von nmap und Auslesen der Ausgabe
8  #
9  my $output = `nmap --script /srv/vulnetd/res/nmapscripts/ssl-
       heartbleed $netid`;
10
11 # Wenn in der Ausgabe das Wort "VULNERABLE" vorkommt, ist
12 # ein Asset von Heartbleed betroffen
13 #
14 if ($output =~ /VULNERABLE/)
15 {
16         print "true";
17 }
18 else
19 {
20         print "false";
21 }
```

Im Wesentlichen benötigt die Implementierung als Eingabe die Netz-ID des Assets in Form einer IP-Adresse oder einem Hostnamen, auf dem der Heartbleed Bug detektiert werden soll. Mit dieser Netz-ID wird nmap aufgerufen, welches wiederum das im Verzeichnis `/srv/vulnetd/res/nmapscripts/` liegende LUA-Skript „ssl-heartbleed.nse“ aufruft und gemäß dem Inhalt des Skripts das Ergebnis auf der Konsole ausgibt. Falls das Asset von Heartbleed betroffen ist, enthält die Ausgabe die Zeichenkette „VULNERABLE“, dessen Vorkommen im Perl Skript überprüft wird. Bei Auftreten gibt das Skript wie im Konzept vorgegeben „true“ zurück, andernfalls „false“.

Eine beispielhafte Dokumentation des Detektionsverfahrens ist in Tabelle 7.15 vorgenommen.

Das Detektionsverfahren aus Listing 7.7 wird wie bereits beschrieben entsprechend der ID der Schwachstelle in Kombination mit dem Identifikator der Dokumentation des Detektionsverfahrens in der Schwachstellen-Dokumentation in der Datei „8819_221“ abgespeichert.

Die Aktualisierung erfolgt wie im vorherigen Kapitel beschrieben, d.h. da alle Detektionssysteme gleichartig sind, wird das Detektionsverfahren nach seiner Freigabe in das Verzeichnis `detect/` und das nmap-Skript „ssl-heartbleed.nse“ in das Verzeichnis `res/nmapscripts/` auf alle Detektionssysteme verteilt.

Feld	Wert
ID	221
Schwachstelle	8819
Beschreibung	Detektion des Heartbleed Bugs unter Verwendung von nmap und des Skripts 'ssl-heartbleed.nse'. Als ersten und einzigen Parameter erwartet das Detektionsverfahren eine IP-Adresse oder den Hostnamen des Assets, auf dem die Detektion durchgeführt wird.
Typ	netz
Editor-ID	192
Abhängig von	nmap, ssl-heartbleed.nse, tls.lua, stdnse.lua
Compiler/ Interpreter	Perl 5.18.2
Plattformen	plattformunabhängig
Abnahme durch	
Freigegeben	

Tabelle 7.15: Dokumentation des Detektionsverfahrens bzgl. des Heartbleed Bugs

7.7 Viertes Fallbeispiel – Detektion als Dienstangebot

In diesem Fallbeispiel wird eine durch das Konzept unterstützte Umsetzung des Schwachstellenmanagements in Form eines Dienstangebots an alle Nutzer des Hochschulnetzes bzw. in diesem Fall des in Abschnitt 7.1 beschriebenen Szenarios erläutert.

7.7.1 Ausgangslage

Wie bereits in Abschnitt 6.5.3 beschrieben haben Nutzer, die keine weitere Rolle im Schwachstellenmanagement einnehmen die Möglichkeit, über das Webportal Meldungstickets zu erstellen sowie Informationen über nicht-assetspezifische Schwachstellen einzusehen. Die Anmeldung auf dem Webportal erfolgt über eine Kombination einer netzspezifischen Nutzerkennung sowie dem vom Nutzer netzweit festgelegten Passwort. Die Authentifizierung erfolgt durch die Abfrage der Korrektheit der Angaben über den im Hochschulnetz dafür vorgesehenen LDAP-Server. Die Rollen, die ein Nutzer innehat werden über die Asset- und Benutzerverwaltung abgefragt.

Als zusätzliche Funktion wird auf dem Webportal ein Dienst angeboten, durch den Nutzer die Möglichkeit haben über verschiedene Verfahren Schwachstellen auf ihrem Rechner zu detektieren. Dabei werden bereits zwei Detektionsverfahren angeboten: Zum einen ein einfacher Portscan und zum anderen ein Verfahren zur Detektion akzeptierter SSL-Cipher Suites mit einer automatischen Beurteilung der jeweiligen Cipher Suite bezüglich ihrer Sicherheit. Beide Verfahren werden bei der Ausführung ausschließlich auf die IP-Adresse des Gerätes ausgeführt, von dem aus der jeweilige Nutzer das Webportal aufgerufen hat.

7.7.2 Hinzufügen eines Dienstes für Nutzer

Aufgrund der hohen Kritikalität und Verbreitung des Heartbleed Bugs sowie einer hohen Anzahl davon potenziell betroffener Maschinen (z.B. FTP- und HTTP-Server) im Hoch-

schulnetz, entscheidet sich der Chief Vulnerability Manager, die für Nutzer bereitgestellten Funktionen um ein Verfahren zur Detektion des Heartbleed Bugs zu erweitern.

Der Chief Vulnerability Manager, Richard Spiegel, beauftragt einen Studenten mit der Implementierung der Erweiterung der Weboberfläche. Die Vorgaben sind wie folgt:

1. Zur Erhöhung der Sicherheit bzw. als Hindernis einer böswilligen Nutzung des Dienstes, wird ausschließlich der Host hinter der IP-Adresse bei der Detektion berücksichtigt, der das Webportal aufruft. Zusätzlich ist der Dienst nur mit einer gültigen Authentifizierung über die netzspezifische Nutzerkennung aufrufbar.

2. Der Nutzer muss keine weiteren Eingaben tätigen.

3. Das Ergebnis der Detektion wird direkt im Webportal angezeigt und es wird kein Schwachstellenticket erstellt.

Der Student setzt die Implementierung unter Verwendung der im Abschnitt 6.5.2 beschriebenen Funktion `detectionDryRun` des Steuerungssystems um, welche die Ausgabe des Detektionsverfahrens des Heartbleed Bugs (vgl. Abschitt 7.6) zurückgibt. In diesem Fall entweder *true*, falls die Detektion positiv ausfällt bzw. *false*, falls die Detektion negativ ausfällt. Als Parameter wird der Identifikator des Detektionsverfahrens, in diesem Fall mit Wert 221 (vgl. Tabelle 7.15 des vorherigen Fallbeispiels), und die IP-Adresse des Gerätes, mit dem das Webportal aufgerufen wurde (auslesbar über den HTTP-Header), übergeben.

Der Student entwickelt die Funktion so, dass bei Rückgabe *true* von `detectionDryRun` dem Nutzer der Text „Ihr Rechner ist von Heartbleed betroffen!“, sowie ein Link auf den Eintrag in der Schwachstellen-Dokumentation und dazugehörigen vorhandenen Maßnahmen angezeigt wird.
Falls die Funktion hingegen *false* zurückgibt, wird im Webportal der Text „Ihr Rechner ist nicht von Heartbleed betroffen!“ angezeigt.

Zur Bekanntmachung der Funktionserweiterung des Dienstes wird zum einen jeder Nutzer, der sich auf dem Webportal anmeldet, auf diese hingewiesen. Zum anderen wird an jeden Nutzer, der sich ebenfalls durch das Schwachstellenmanagement am LRZ per E-Mail automatisch über aktuelle Schwachstellen informieren lässt, die Benachrichtigung auf diesem Weg verschickt.

7.7.3 Nutzung des Dienstes

Ein Mitarbeiter an der Fakultät für Wirtschaftswissenschaften der *Technischen Universität München*, Johannes Schwarz, ist Betreiber eines öffentlich erreichbaren FTP-Servers auf *Debian*-Basis mit Anwendung *ProFTPD* zur Verwaltung von Vorlesungsmaterialien. Er nimmt keine Rolle im netzweiten Schwachstellenmanagement ein und das von ihm betriebene Gerät stellt kein Asset im Schwachstellenmanagement dar, da der Leiter des Instituts diese Dienstleistung des LRZ nicht in Anspruch nehmen will, da er (ungerechtfertigterweise) mehr Aufwand befürchtet.

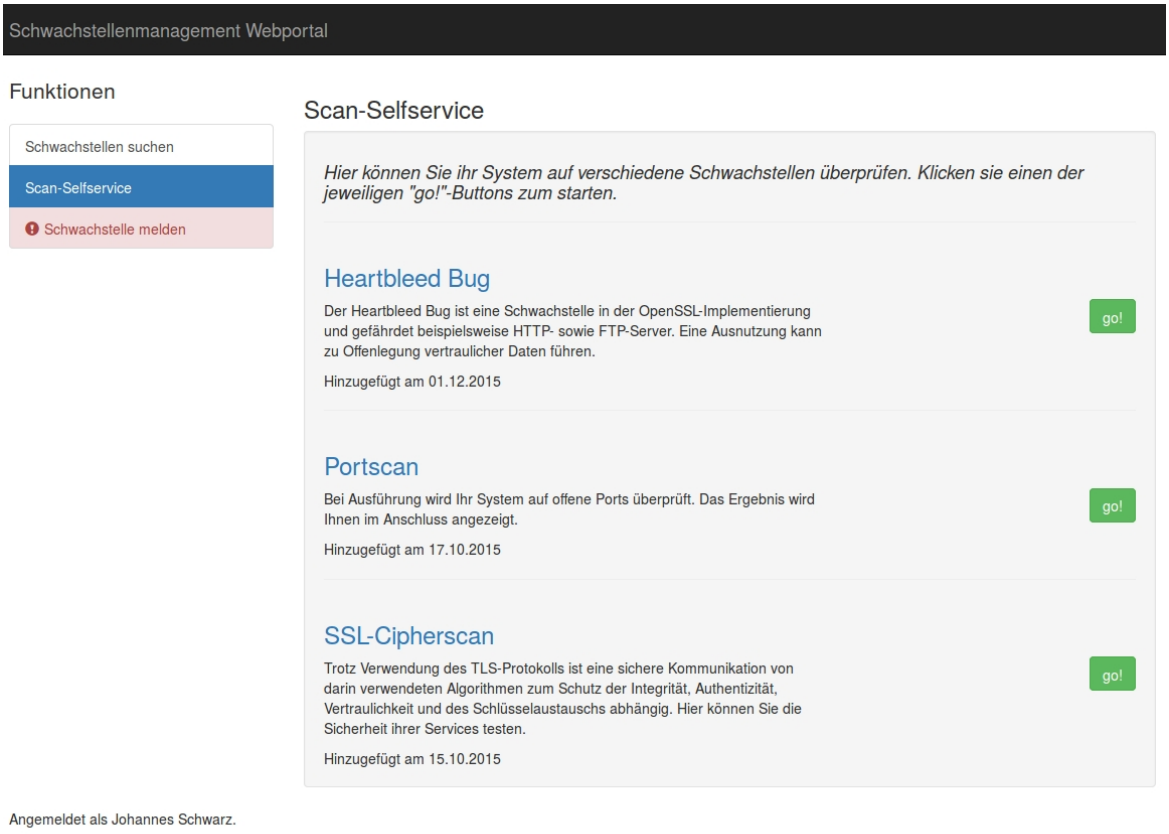


Abbildung 7.3: Beispielhafte Weboberfläche eines Nutzers ohne weitere Rollen im Schwachstellenmanagement. Gezeigt sind für ihn bereitgestellte Funktionen zur Detektion

Jedoch nimmt Johannes Schwarz den Benachrichtigungs-Dienst via E-Mail des LRZ über aktuell gefährdende Schwachstellen in Anspruch. Über diesen erfährt er, dass nun ebenfalls die Detektion des Heartbleed Bugs für Netznutzer auf ihren Geräten zur Verfügung steht.

Johannes Schwarz ruft daher, ausgehend von seinem FTP-Server, das Webportal des Schwachstellenmanagements auf und meldet sich mit seiner Kennung im Netz an. Nachdem er in der Funktionsliste die Registerkarte „Scan-Selfservice“ aufruft, gelangt er zu einer Übersichtsseite mit allen angebotenen Detektionsverfahren, die von allen Netznutzern verwendet werden können. Eine beispielhafte Ansicht der Weboberfläche ist in Abbildung 7.3 gezeigt.

Nachdem Johannes Schwarz die Detektion des Heartbleed Bugs startet, wird der Auftrag vom Steuerungssystem an ein geeignetes Detektionssystem zugewiesen, welches die Ausführung des in Listing 7.7 gezeigten Detektionsverfahrens unter Verwendung von *nmap* durchführt. Dieses liefert *true* zurück, da die Detektion des Heartbleed Bugs positiv ausfällt.

Johannes Schwarz wird insofern die in Abbildung 7.4 illustrierte Oberfläche angezeigt,

wodurch ihm seine Betroffenheit von der Schwachstelle, eine Referenz auf die Beschreibung des dazugehörigen Schwachstelleneintrags sowie bereits dokumentierte Maßnahmen mit Referenzen auf eine Detailbeschreibung, mitgeteilt werden.

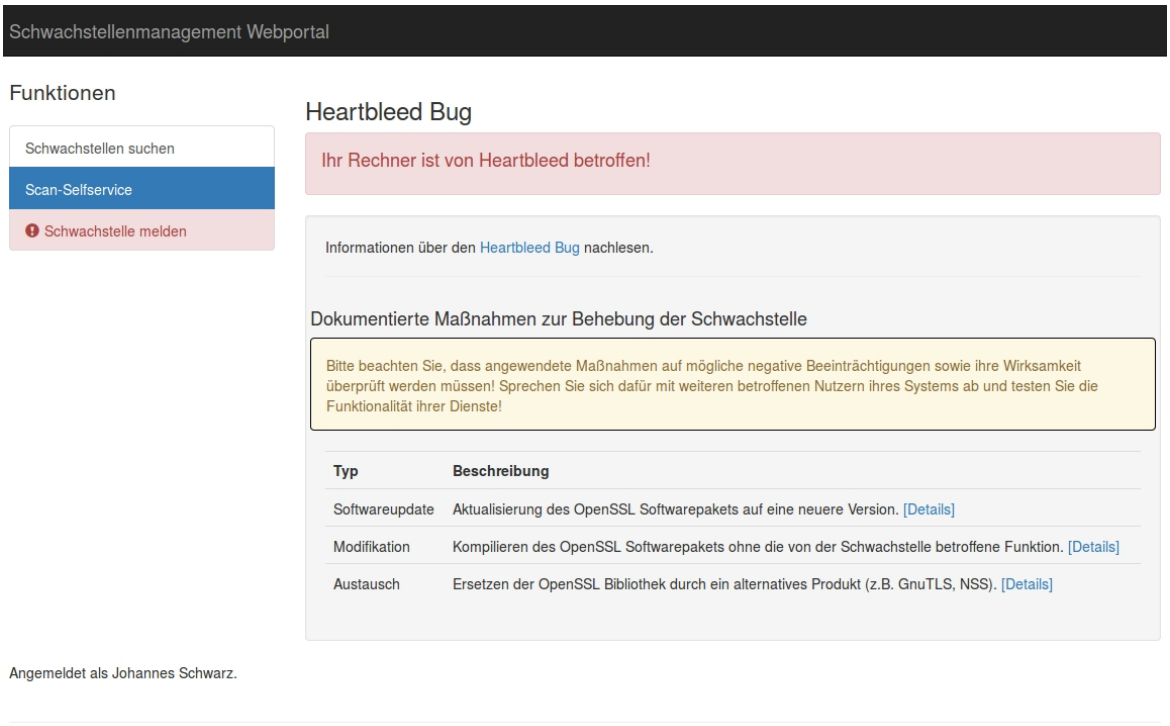


Abbildung 7.4: Auf dem Webportal angezeigte Oberfläche bei positiver Detektion des Heartbleed Bugs

Da das von der Schwachstelle betroffene Gerät auf Wunsch des Kunden nicht durch das Schwachstellenmanagement abgedeckt ist, ist die Verantwortlichkeit zur Behebung der Schwachstelle nicht eindeutig geklärt.

Johannes Schwarz hat nun potenziell die Möglichkeiten, das Ergebnis zu ignorieren und die Schwachstelle bestehen zu lassen, die Schwachstelle als Betreiber des Dienstes selbst zu beheben, oder die Informationen an eine im Institut dafür vorgesehene Person weiterzugeben.

8 Zusammenfassung und Ausblick

In diesem Abschnitt wird zum einen eine Zusammenfassung der Ergebnisse der Arbeit vorgenommen sowie auf zukünftige wissenschaftliche Arbeiten eingegangen, die zu einer Steigerung der Effizienz im Schwachstellenmanagement in Hochschulnetzen beitragen können.

8.1 Zusammenfassung der Arbeit

Zur Ermittlung der Anforderungen wurde in dieser Arbeit zunächst der Inhalt von Schwachstellenmanagement in einer Hochschulumgebung in Form der darin betrachteten Aktivitäten, sowie die Art der betrachteten Schwachstellen geklärt. Um mögliche Herausforderungen in diesem Umfeld zu identifizieren wurden typische Charakteristika eines Hochschulnetzes anhand des Abgleichs mehrerer Netzkonzepte bestimmt.

Um den Inhalt eines Schwachstellenmanagements an die Hochschulumgebung anzupassen, bestand ein wichtiger Teil der Arbeit darin, Anforderungen an ein solches zu stellen. Die Anforderungen wurden basierend auf zwei Szenarien erstellt:
Zum einen ein Szenario auf Basis der Beschaffenheit eines charakteristischen Hochschulnetzes, dem *Münchner Wissenschaftsnetz*, und zum anderen den zum Zeitpunkt der Verfassung der Arbeit aktuellen Maßnahmen zum Umgang mit Schwachstellen am *Leibniz-Rechenzentrum*, welche durch eine zu diesem Zweck entwickelte Umfrage – vor allem gerichtet an die Dienstbetreiber der wichtigsten Dienste – in Erfahrung gebracht wurden (vgl. Anhang 1).

Eine Begutachtung und Abgleich mehrerer Standards, Konzepte und vorhandener Lösungen zum Schwachstellenmanagement (insbesondere in einer Hochschulumgebung) ergab, dass bei weitem nicht alle Anforderungen erfüllt werden können und die konzeptionelle Beschreibung der Arbeiten weitestgehend abstrakt bleiben, wodurch eine direkte Umsetzung ausschließlich anhand dieser nicht möglich ist. Teilweise ergaben sich jedoch sinnvolle Aspekte, die in eine Umsetzung integrierbar sein müssen – beispielsweise Informationen aus existierenden Schwachstellen-Datenbanken.

Daher besteht der zentrale Teil der Arbeit in der Ausarbeitung eines Konzeptes, das die Anforderungen eines Schwachstellenmanagements in Hochschulnetzen erfüllt und sowohl die Organisation als auch notwendige technische Komponenten berücksichtigt.

Die **technischen Komponenten** gliedern sich grob in drei Kategorien: Zum einen Komponenten zur Verwaltung der Informationen im Schwachstellenmanagement – die Schwachstellen-Dokumentation sowie Asset- und Benutzerverwaltung – zum anderen eine Komponente zur Verwaltung der Abläufe, dem Zugriff auf das System durch Nutzer und Bereitstellung von Informationen und Funktionen für Komponenten außerhalb des Schwachstellenmanagements über eine universell nutzbare API – das Steuerungssystem. Die dritte Kategorie umfasst Komponenten zur Bereitstellung von Funktionen zur Detektion von Schwachstellen

und deren Ausführung – Detektionssysteme, zur Detektion aus Netzsicht sowie Detektionsagenten auf Assets zur Detektion aus Systemsicht. Die Anzahl der Detektionssysteme und Detektionsagenten kann dynamisch an die Größe des Netzes angepasst werden, damit ein Detektionsdurchlauf auch auf vielen Systemen zeitnah durchgeführt werden kann. Dazu können insbesondere Detektionsagenten zentral vom Dienstanbieter, dem Rechenzentrum, oder dezentral von einzelnen Instituten betrieben und in privaten Netzen eingesetzt werden. Die Wartung der Software der Detektionssysteme erfolgt in jedem Fall von zentraler Stelle durch das Rechenzentrum, um die Gleichartigkeit dieser zu erhalten und Kunden eine zuverlässige IT-Dienstleistung bieten zu können.

Die Durchführung der Aktivitäten kann im Konzept sowohl größtenteils **automatisiert** als auch **manuell** erfolgen, wobei eine automatisierte Vorgehensweise üblicherweise, abhängig von der umgesetzten Variante des Schwachstellenmanagements, den größeren Teil an Schwachstellen behandelt. Darin ist die flexible Kombination von Informationen über Schwachstellen aus mehreren verschiedenen Quellen vorgesehen, die durch geeignete, im Konzept beschriebene Mechanismen – insbesondere zur Vermeidung irrelevanter Einträge, Duplikate und dem Erhalt der Aktualität – verwaltet werden.

Zur Erreichung einer gleichartigen Einstufung von Schwachstellen, ist im Konzept zudem ein **hochschulnetzweit einsetzbares Bewertungssystem** beschrieben worden. Dieses berücksichtigt einerseits elementare Charakteristika einer Schwachstelle – die Wahrscheinlichkeit und die Auswirkung einer Ausnutzung – und ist zum Erhalt der Aktualität der Beschreibung an das *Security Information & Event Management* gekoppelt, andererseits zur Einsatzfähigkeit in einem Hochschulumfeld mit stark heterogenen Benutzergruppen möglichst einfach gehalten. Korrigierend wirkt hier der Faktor der **Temporären Dringlichkeit**. Dieser wird pro Schwachstelle hochschulnetzweit festgelegt und nach einer konfigurierbaren zeitlichen Dauer automatisch zurückgesetzt. Wesentlicher Bestandteil zur Nutzbarkeit des Bewertungssystems ist auch die Verwendung bereits vorhandener Bewertungen von Schwachstellen durch geeignete **Abbildungen auf das im Hochschulnetz genutzte Bewertungssystem**. So kann beispielsweise eine Bewertung durch das weit verbreitete CVSS und beliebige andere Bewertungssysteme wie Microsofts *Security Bulletin Severity Rating* oder Qualys *Security Levels* verwendet werden. Die **Notwendigkeit einer Behebung** einer Schwachstelle wird abhängig von ihrer Kritikalität umgebungsabhängig für jedes Asset separat ermittelt.

Im Konzept ist zudem ein System zum **Einsatz beliebiger Verfahren zur Detektion von Schwachstellen** enthalten. Dieses setzt sich aus einer hochschulnetzweiten Standardisierung der Beschreibung und Implementierung von Detektionsverfahren sowie beschriebenen Mechanismen zur Verteilung auf Detektionssystemen und Detektionsagenten sowie zur Ausführung und Interpretation der Ergebnisse zusammen. Auf diese Weise können auch beliebige Funktionen bereits existierender Schwachstellen-Scanner wie *nmap* oder *OpenVAS*, als auch selbst entwickelte Verfahren im Schwachstellenmanagement genutzt und gleichartig behandelt werden. Bei der Detektion von Schwachstellen auf Assets wird die **Menge der zur Detektion relevanten Schwachstellen** über die Anwendung mehrerer **Filter** optimiert. Dabei wird das hochschulnetzweite **Schwachstellenakzeptanzkriterium** als zentrales Element zur Kontrolle gesehen, durch das der Umfang der zu behandelnden Schwachstellen anhand ihrer Kritikalität gesteuert werden kann.

Um eine organisierte Behebung der Schwachstellen zu unterstützen, wurde im Konzept ein einfaches **Ticketsystem** beschrieben, das die zuständigen Rollen bei einer, je nach Behebungsnotwendigkeit, geeigneten **Priorisierung der Behebung** unterstützt und angewende-

te Maßnahmen dokumentiert. Auch kann ein bereits vorhandenes Ticketsystem, insbesondere das des *Incident & Service Request Managements*, in die Bearbeitung integriert werden, um fehlende Verantwortlichkeiten zu kompensieren und trotzdem eine angemessene Behebung von Schwachstellen zu garantieren. Darauf aufbauend werden abhängig von der Kritikalität einer Schwachstelle und der Dauer eines Tickets im offenen Zustand konfigurierbare Mechanismen zur funktionalen sowie hierarchischen **Eskalation der Behebung** beschrieben und angewendet.

Auch wurde im Konzept die in vorhergehenden Arbeiten in der Regel vernachlässigte **Prävention von Schwachstellen** thematisiert. Im Konzept ist diesbezüglich beispielsweise eine enge Verknüpfung des Schwachstellenmanagements mit dem Change Management vorgesehen.

Für manuell durchzuführende Aktivitäten – generell im Falle, dass eine Automatisierung nicht möglich ist – wurden im Konzept **Verfahren**, **Rollen** und dazugehörige **Verantwortlichkeiten** definiert. Zudem erfolgt immer eine Unterstützung durch die oben beschriebenen technischen Komponenten in Verbindung mit einer **zentralen und mandantenfähigen Zugriffsmöglichkeit** über ein vom Steuerungssystem angebotenes **Webportal**, wodurch beispielsweise ebenfalls der organisatorisch essentielle Aspekt der Benachrichtigung von Ereignissen über vordefinierte Meldewege erfolgt.

Für jede Teilaktivität (Ersterfassung, Verifikation, Bewertung, Kategorisierung, Detektion, Behebung und Eskalation) ist eine konkrete Rolle definiert, die in einer Umsetzung entweder jeweils einzeln oder in Kombination an eine Person vergeben werden kann.

Die Verwaltung auf Ebene einzelner Institute wird hier durch genau eine pro Institut vergebene Rolle verantwortet, dem Schwachstellen-Manager. Die netzweite Verantwortung für den Prozess und eine angemessene Konfiguration liegt bei der pro Hochschulnetz genau einmal vergebene Rolle des Chief Vulnerability Managers.

Ein Faktor, der sich bei der Umsetzung als herausfordernd erwies, ist die hochgradig heterogene Umgebung einerseits in Bezug auf Gerätetypen und andererseits vor allem auf die Softwarelandschaft, die potenziell vom Schwachstellenmanagement abgedeckt werden muss. Das Schwachstellenmanagement muss daher hinsichtlich der Assets weitestgehend **plattformunabhängig** wirksam und im gesamten Netz anwendbar zu sein. Dies wird zum einen durch einen angemessenen Anwendungsbereich, geeignete Quellen für Schwachstelleninformationen und plattformunabhängige technische Komponenten (insbesondere Detektionsagenten) realisiert.

Eine weitere Herausforderung ist die oft stark eingeschränkte Zuständigkeit des Rechenzentrums, die üblicherweise lediglich „bis zur Datendose“ reicht. Somit wird die Verantwortung der Umsetzung, die allgemein auf freiwilliger Basis stattfindet, zwangsläufig auf eine Vielzahl voneinander unabhängiger Institute verteilt. Zum einen soll unerfahrenen institutszugehörigen Nutzern eine Realisierung eines Schwachstellenmanagements ermöglicht und zum anderen vermieden werden, dass Institute jeweils eine eigene Lösung ohne größeren Zusammenhang und Nutzen umsetzen.

Gelöst wird diese Problematik durch die **netzweite Bereitstellung der technischen Komponenten** mit von zentraler Stelle angebotenen Funktionen zur einheitlichen Dokumentation von Schwachstellen und damit in Verbindung stehenden Informationen als Dienstleistung von Seiten des Hochschulrechenzentrums. Somit werden Informationen im gesam-

ten Hochschulnetz von allen Nutzern im Schwachstellenmanagement, unabhängig ihrer Zugehörigkeit zu einem Institut kollektiv für alle Nutzer des gesamten Netzes erarbeitet. Informationen über Schwachstellen und vor allem auch Maßnahmen zur Behebung von Schwachstellen stehen infolgedessen allen Nutzern zur Verfügung.

Durch das Angebot der technischen Komponenten wird, in Verbindung mit einer **einheitlichen Benutzerschnittstelle** zum System, ebenfalls die **einheitliche Beschreibung der Daten** gewährleistet. Insofern haben die Nutzer so die Möglichkeit, ihre privaten Geräte in das Schwachstellenmanagement zu integrieren.

Herausforderungen wie die in Abschnitt 2.5.1 im Hochschulnetz anzutreffenden nicht-gemanagten Geräte, spezielle Hard- und Softwarelösungen, mobile Geräte und virtuelle Maschinen sind vor allem über die Wahl eines geeigneten Anwendungsbereichs in Bezug auf betrachtete Assets und Schwachstellen lösbar.

Die in Abschnitt 2.5.4 genannten generellen Herausforderung bezüglich des Verhaltens von Netznutzern, wie einer Gleichgültigkeit gegenüber Schwachstellen oder einer eingeschränkten Erreichbarkeit wegen unzureichender Teilnahme am Prozess (in diesem Fall zum Beispiel das direkte Einloggen auf dem Webportal und Überprüfung konkret vorhandener, zugewiesener Verantwortlichkeiten zur Ausführung einer Teilaktivität), können durch geeignete Schulung der Nutzer und auch beispielsweise die automatische Eskalation der Behebungsverantwortung bewältigt werden.

Die Beschreibung der Implementierung findet sich in Kapitel 6. Darin wurden alle wichtigen Komponenten – die Schwachstellen-Dokumentation, Asset- und Benutzerverwaltung, das Steuerungssystem, Detektionssysteme und Detektionsagenten – und deren Kommunikation untereinander berücksichtigt. Alle im Konzept beschriebenen Informationen wurden zudem durch ein Datenbankschema dargestellt, das im Anhang 2 beiliegt.

Zur Veranschaulichung des Durchlaufs durch das Konzept wurden zwei Schwachstellen als Fallbeispiele genutzt und die jeweils einzelnen Stationen von ihrer Identifikation bis zur Beseitigung und Abschwächung sowie Prävention am Beispiel der Umgebung des Münchner Wissenschaftsnetzes und insbesondere des Leibniz-Rechenzentrums durchlaufen. Zusätzlich wird ein Fallbeispiel zur Integration bereits vorhandener Verfahren zur Detektion einer Schwachstelle unter Verwendung von *nmap* anhand des Heartbleed Bugs verdeutlicht sowie darüber hinaus ein Fallbeispiel zur Veranschaulichung der Detektion als Dienstleistung für Netznutzer beschrieben.

8.2 Weiterführende Arbeiten

In dieser Arbeit wurde durch das Konzept gezeigt, dass ein Schwachstellenmanagement in einer Hochschulumgebung, bestehend aus verschiedenen Instituten, unter Berücksichtigung der darin auftretenden Herausforderungen realisiert werden kann. Trotzdem gibt es Probleme, die mehr allgemein auftreten und weniger mit der Umgebung des Hochschulnetzes zusammenhängen. Deren Lösung kann vor allem zu einer Steigerung der Effizienz beitragen.

Eine dieser Herausforderungen hängt vor allen mit einer stark **heterogenen Beschreibung von Schwachstellen** zusammen. Eine Art der Beschreibung, die üblicherweise alle

Quellen für Schwachstelleninformationen gemein haben, liegt in Freitextform vor. Die Formulierung, der Inhalt und Detailgrad hängt jedoch stark von der jeweiligen Quelle bzw. der Person ab, die die Schwachstelle beschreibt, wodurch eine automatisierte Extraktion relevanter Informationen – insbesondere betroffene Softwareprodukte – in der Regel nicht möglich ist.

Als ein Schlüsselelement bezüglich der Automatisierung des Schwachstellenmanagements haben sich Softwareprodukte und im Besonderen die Art der Beschreibung zu ihrer eindeutigen Identifikation erwiesen. In Form der *Common Platform Enumeration* (CPE) existiert bereits ein gut dokumentiertes und zugleich einfach anwendbares Namensschema zu diesem Zweck (vgl. Abschnitt 4.2.3.1), und wird bereits in der *National Vulnerability Database* als Elemente in OVAL-Regeln verwendet, um betroffene Softwareprodukte einer Schwachstelle eindeutig zu identifizieren.

Das in der CPE verwendete Namensschema wird jedoch nicht flächendeckend genutzt, wodurch beispielsweise das Filtern von Schwachstellen vor der Detektion stark erschwert wird, da, auch wenn beispielsweise zwei Einträge „SUSE Linux Enterprise Server 11.0“ und „SLES 11_0“ dasselbe Softwareprodukte beschreiben, die Gleichheit aufgrund des stark unterschiedlichen Formats im Regelfall nicht automatisiert erkannt werden kann. Als Lösungsansatz kann hier die Verwendung einer Abstandsfunktion wie beispielsweise die Levenshteindistanz dienen, oder, wie im Konzept verwendet, der Abgleich der Bezeichnung des jeweils einen Softwareproduktes als Teilzeichenkette in der Bezeichnung des jeweils anderen Softwareprodukts. Die Verlässlichkeit hängt jedoch stark von der Unterschiedlichkeit der Bezeichnung ab.

Jedoch auch bei einem Abgleich zweier Bezeichner von Softwareprodukten über eine geeignete Funktion besteht dennoch das Problem, dass diese Art der Filterung tendenziell ungeeignet ist, da bei dieser Vorgehensweise der Ausgangspunkt eine leere Liste ist. Schwachstellen würden bei Nicht-Zutreffen des entsprechend genutzten Kriteriums zur Ermittlung der Ähnlichkeit als irrelevant betrachtet werden, wodurch eine hohe Zahl an False-Negatives entsteht, die dazu führen kann, dass Schwachstellen in bestimmten Softwareprodukten nicht detektiert werden. Verstärkt wird diese Problematik durch die Ungewissheit über eine Vollständigkeit, zum einen in Bezug auf die Liste der Softwareprodukte, die von einer bestimmten Schwachstelle betroffen sind, als auch in Bezug auf das Softwareinventar eines Assets.

Eine zuverlässigere Lösung bietet der Ansatz, zunächst alle Schwachstellen als relevant zu betrachten und Schwachstellen, die für eine Detektion als sicher irrelevant eingestuft werden, aus dieser Liste zu entfernen. So beeinflussen False-Negatives in keiner Weise die Wirksamkeit, sondern lediglich die Effizienz, da im ungünstigsten Fall alle dokumentierten Schwachstellen für die Detektion relevant sind. Auch das im vorherigen Absatz beschriebene Problem, das eine ungewisse Vollständigkeit der Dokumentation einer Schwachstelle bezüglich betroffener und Assets bezüglich installierter Softwareprodukte mit sich bringt, fällt bei diesem Ansatz weg. Im Zweifelsfall werden dann schließlich wieder alle Schwachstellen bei einer Detektion in Betracht gezogen. Eine Metrik für diese Vorgehensweise zu bestimmen ist jedoch erheblich schwieriger, da wie in dem oben gezeigten Beispiel bezüglich der Beschreibung des SUSE Linux Enterprise Servers, die Formate im Einzelfall extrem stark variieren können und somit eine gemeinsame Semantik der Beschreibungen schwer bis nicht ermittelbar ist.

Eine allgemein einheitliche Beschreibung von Softwareprodukten, zum einen in der Schwachstellen-Dokumentation als auch im Softwareinventar der Assets in Kombination mit einer garantierten Vollständigkeit dieser Daten, würde jedoch den Schritt des Filterns von Schwachstellen sowie die Detektion im klassischen Sinne aus Netz- und Systemsicht überflüssig werden lassen. Dann könnte auf einem Asset installierte Software mit von Schwachstellen betroffenen Softwareprodukten direkt an zentraler Stelle (z.B. dem Steuerungssystem im Konzept) abgeglichen werden und ohne jeglichen Zugriff auf Detektionssysteme und Agenten eine Detektion stattfinden lassen. Insofern ist die Vereinheitlichung der Bezeichnung von Softwareprodukten der Schlüssel zur Automatisierung im Schwachstellenmanagement.

Eine weitere Problematik stellt die **automatisierte Behebung** von Schwachstellen auf Assets im Schwachstellenmanagement dar. Anders als im Patchmanagement steht hier neben der Installation von Softwarepatches ein weites Spektrum an möglichen Maßnahmen zur Behebung zur Verfügung. Da üblicherweise Patches die geeignetste Lösung zur Behebung einer Schwachstelle sind und die Einrichtung bzw. Installation für jeden Patch in der Regel gleichartig ist, ist eine Automatisierung diesbezüglich gut möglich (und wird von Softwareherstellern allgemein flächendeckend umgesetzt). Problematischer sind Schwachstellen, zu denen es kein Softwarepatch gibt, insbesondere Konfigurationsschwachstellen. Maßnahmen wie die Ausweitung der Zugriffskontrolle oder Außerbetriebnahme der Software sind praktisch immer unter Betrachtung eines größeren Zusammenhangs vorzunehmen. Beispielsweise muss bekannt sein, ob das Asset bzw. das darauf von einer Schwachstelle betroffene Softwareprodukt einen Dienst unterstützt oder anbietet und welche Folgen die Ergreifung der Maßnahme hat. Nebenbei müssen Patches ebenfalls auf ihre Wirksamkeit und negative Auswirkungen überprüft werden, wodurch auch hier manueller Aufwand entsteht.

Insofern muss die automatisierte Behebung von Schwachstellen ein Teil zukünftiger Forschungen bleiben, bzw. ausgeweitet werden.

Eine weitere Herausforderung hängt mit der im Konzept erwähnten **Integration dezentraler Asset- und Benutzerverwaltungen** zusammen. Da Institute unter Umständen Detaildaten über ihre Assets und Mitarbeiter bzw. Nutzer nicht in eine vom Rechenzentrum angebotene Asset- und Benutzerverwaltung integrieren wollen, ist es sinnvoll, Datenquellen miteinzubeziehen, die direkt von den Instituten verwaltet und deren Daten nicht im Rechenzentrum zwischengespeichert werden.

Die zu untersuchende Herausforderung liegt dabei in der Kombination der Daten mit denen aus der Schwachstellen-Dokumentation. Dabei müssen insbesondere die Einhaltung der Anforderung eines netzweit eindeutigen Identifikators für Assets und auch die Sicherstellung der Aktualität der Daten beachtet werden. Ein weiterer wichtiger Aspekt, der bei einer derartigen Lösung genauer untersucht werden muss, ist eine netzweite Rollen- und Rechteverwaltung, um sicherzustellen, dass Personen nicht unberechtigterweise sensible Daten einsehen können – beispielsweise assetspezifische Schwachstellen.

Eine ebenfalls für das Schwachstellenmanagement hilfreiche weiterführende Arbeit ist die Erstellung eines **Bewertungssystems zum Ausdrücken des Zustandes** eines Assets in Bezug auf Schwachstellen. Das Ziel eines derartigen Bewertungssystems wäre es, anhand eines dadurch generierten Wertes (beispielsweise einer Punktzahl) den Schweregrad eines Asset bezüglich darauf existenter Schwachstellen einschätzen zu können. So kann zum einen ebenfalls unter Umständen, je nach Situation, eine geeignetere Priorisierung der Behebung

stattfinden oder zwei Assets direkt miteinander verglichen werden. Beispielsweise könnten die Anzahl der auf dem Asset existenter Schwachstellen sowie deren jeweilige Kritikalität ein Faktor zur Berechnung dieser Einschätzung sein.

Das in der Arbeit beschriebene Konzept bietet die Möglichkeit, in einer seiner Varianten in einem beliebigen Hochschulnetz realisiert zu werden. Eine Erweiterung zu diesem Ansatz kann ein **hochschulnetzübergreifendes Schwachstellenmanagement** bieten. So können daraus resultierende Vorteile, wie die gemeinsame Erarbeitung und Nutzung von Daten im Netz, entsprechend ausgeweitet werden, sodass gleichartig beschriebene Informationen über Schwachstellen, Detektionsverfahren und insbesondere Maßnahmen zur Behebung aus Instituten verschiedener Hochschulnetze erarbeitet und ebenfalls auf die Ausdehnung des gesamten Bereichs (beliebig vieler Hochschulumgebungen) verwendet werden können.

Dabei kann als Teil des Lösungsansatzes das in dieser Arbeit beschriebene Konzept in den einzelnen Hochschulnetzen umgesetzt werden, wobei die Bereitstellung von Daten und technischen Komponenten auf eine Art und Weise dass alle Hochschulnetze sie nutzen können als auch eine geeignete Rollen- und Rechteverwaltung herausfordernd wirken.

Beispielsweise kann eine Schwachstellen-Dokumentation sowie Detektionssysteme von übergeordneter Stelle wie in Deutschland dem Betreiber des *Deutschen Forschungsnetzes*, dem *DFN-Verein*, angeboten werden.

Auch sind insbesondere Aspekte zu betrachten, die in diesem Konzept durch die Rolle des Chief Vulnerability Managers institutionsübergreifend übernommen werden. Vor allem die Festlegung eines Wertes für essentielle Parameter wie dem Schwachstellenakzeptanzkriterium, Inhalt globaler Blacklisten oder Programmiersprachen zur Implementierung von Detektionsverfahren. Zu betrachten ist insbesondere der Rahmen, in dem diese festgelegt werden und die Instanz bzw. Rolle, welcher die Verantwortung zum Festlegen zukommt. Beispielsweise würde der Beschluss eines Wertes für das Schwachstellenakzeptanzkriterium auf Ebene des jeweiligen Hochschulnetzes mehr Sinn machen als auf hochschulnetzübergreifender Ebene, da jeder Hochschulumgebung ein unterschiedlicher Umfang an Resourcen zur Verfügung steht.

Andererseits ist eine hochschulnetzübergreifende Festlegung der Programmiersprache für Detektionsverfahren sinnvoll, da so die Wiederverwendbarkeit in anderen Umgebungen vereinfacht wird.

9 Glossar

In diesem Kapitel ist die Bedeutung in der Arbeit oft auftauchender Begriffe gesammelt. Das Glossar ist alphabetisch geordnet.

- **Anwendungsbereich**
  Der Anwendungsbereich, auch Geltungsbereich oder engl. „Scope“ genannt, definiert den Rahmen und Umfang der Umsetzung eines Managements. In diesem Fall das Management der Behandlung von Schwachstellen.

- **Asset**
  Gemäß der Definition des *Bundesamtes für Sicherheit in der Informationstechnik* sind Assets (engl. „Werte“) alles, was wichtig für eine Institution ist.[Bun13] In dieser Arbeit bezeichnet ein Asset ein Computersystem, das im Rahmen des Schwachstellenmanagements als schützenswert angesehen wird.

- **Bedrohung**
  Laut dem *Bundesamt für Sicherheit in der Informationstechnik* ist eine Bedrohung ein Umstand oder Ereignis, das Schaden verursachen kann. Dieser Schaden kann sich allgemein beispielsweise auf den Ruf eines Unternehmens, auf Assets, Mitarbeiter oder Informationen beziehen.[Bun13][Kis13] Im Zusammenhang mit Schwachstellenmanagement bezeichnet eine Bedrohung allgemein die Möglichkeit des versehentlichen Auslösens bzw. beabsichtigter Ausnutzung einer Schwachstelle.[Dep15]

- **Detektionsverfahren**
  Ein Detektionsverfahren ist eine Kombination aus technisch ausführbarer Möglichkeit zur Detektion einer bestimmten Schwachstelle (üblicherweise in Form einer Implementierung) und der dazugehörigen Dokumentation.

- **Exploit**
  Ein Exploit ist generell eine Möglichkeit zur Ausnutzung einer Schwachstelle. Oft in Form einer Anwendung bzw. eines Programms, das einem Angreifer die automatische Ausnutzung erlaubt.

- **Fremdquelle**
  Bezogen auf das Schwachstellenmanagement in einer definierten Umgebung (dem Hochschulnetz), beschreibt eine Fremdquelle eine Quelle an Informationen über Schwachstellen, die außerhalb der Umgebung liegt.

- **Patch**
  Ein Patch bzw. Softwarepatch ist „eine Aktualisierung eines Betriebssystems, einer Anwendung oder andersartiger Software, die speziell darauf abzielt, Probleme in der Software zu korrigieren“. [Kis13] In dieser Arbeit stellen besagte Probleme üblicherweise eine Schwachstelle im Quelltext des Programms dar.

- **Penetration Testing**
  Penetration Testing (auch „Pentest“) beschreibt den Versuch der Ausnutzung von Schwachstellen, um diese aufzudecken oder auszunutzen und „Sicherheitsmaßnahmen eines Informationssystems zu umgehen“ [Kis13].

- **Schwachstelle**
  Eine Schwachstelle ist eine Schwäche in einem Informationssystem, in sicherheitsrelevanten Prozeduren eines Systems, internen Kontrollen oder der Implementierung, die durch eine Bedrohung ausgelöst oder ausgenutzt werden kann. [Kis13] Oft ist die Ausnutzung einer Schwachstelle mit einem daraus resultierenden potenziellen Schaden verbunden.

- **Schwachstellenakzeptanzkriterium**
  Das Schwachstellenakzeptanzkriterium beschreibt den höchsten Wert eines Systems zur Bewertung der Kritikalität einer Schwachstelle, die (in der Regel aus Resourcenmangel) nicht behoben, sondern akzeptiert wird. Für alle Schwachstellen mit einer kritischeren Bewertung hingegen ist eine Behebung vorgesehen.

- **Schwachstellenmanagement**
  „Schwachstellenmanagement bezeichnet die zyklische Ausübung der Identifikation, Klassifikation, Beseitigung und Abschwächung“ [For09] sowie die Prävention von Schwachstellen.

- **Temporary Fix**
  Ein Temporary Fix ist eine offizielle, jedoch nicht dauerhafte Möglichkeit, zur Beseitigung oder Abschwächung einer Schwachstelle.

- **Workaround**
  Ein Workaround ist eine inoffizielle Lösung zur Beseitigung oder Abschwächung einer Schwachstelle, die nicht durch den Hersteller herausgegeben, sondern häufig in Foren oder Communities durch Nutzer und Interessierte entwickelt wird.

Abbildungsverzeichnis

Literaturverzeichnis

[And15] Andreas, Wilkens: *Hacker haben bei Cyber-Angriff auf Bundestag Daten abgezogen*, Mai 2015. http://www.heise.de/newsticker/meldung/Hacker-haben-bei-Cyber-Angriff-auf-Bundestag-Daten-abgezogen-2671071.html (abgerufen am 04.06.2015).

[Bay15] Bayerisches Staatsministerium für Bildung und Kultus, Wissenschaft und Kunst: *Zielvereinbarungen*, Juni 2015. http://www.km.bayern.de/studenten/wissenschaftspolitik/zielvereinbarungen.html (abgerufen am 19.06.2015).

[BBF11] Barrère, Martín, Rémi Badonnel und Olivier Festor: *Supporting vulnerability awareness in autonomic networks and systems with OVAL*. In: *Proceedings of the 7th International Conference on Network and Services Management*, Seiten 37–45. International Federation for Information Processing, 2011.

[Bod15] Bodnar, Ladislav: *DistroWatch Page Hit Ranking*, Juli 2015. http://distrowatch.com/dwres.php?resource=popularity (abgerufen am 12.07.2015).

[Bra11] Brant A. Cheikes, David Waltermire, Karen Scarfone: *Common Platform Enumeration: Naming Specification Version 2.3*, August 2011. http://csrc.nist.gov/publications/nistir/ir7695/NISTIR-7695-CPE-Naming.pdf (abgerufen am 07.11.2015).

[Bru07] Bruce, Schneier: *Schneier: Full Disclosure of Security Vulnerabilities a 'Damned Good Idea'*, Januar 2007. https://www.schneier.com/essays/archives/2007/01/schneier_full_disclo.html (abgerufen am 03.08.2015).

[Bun08a] Bundesamt für Sicherheit in der Informationstechnik: *BSI-Standard 100-1 Managementsysteme für Informationssicherheit (ISMS) Version 1.5*, 2008. https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/ITGrundschutzstandards/standard_1001_pdf.pdf?__blob=publicationFile (abgerufen am 02.07.2015).

[Bun08b] Bundesamt für Sicherheit in der Informationstechnik: *BSI-Standard 100-2 IT-Grundschutz-Vorgehensweise Version 2.0*, 2008. https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/ITGrundschutzstandards/standard_1002_pdf.pdf?__blob=publicationFile (abgerufen am 02.07.2015).

[Bun08c] Bundesamt für Sicherheit in der Informationstechnik: *BSI-Standard 100-3 Risikoanalyse auf der Basis von IT-Grundschutz Version 2.5*, 2008. https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/

ITGrundschutzstandards/standard_1002_pdf.pdf?__blob=publicationFile (abgerufen am 02.07.2015).

[Bun09] Bundesamt für Sicherheit in der Informationstechnik: *G 4.22 Software-Schwachstellen oder -Fehler*, 2009. https://www.bsi.bund.de/DE/Themen/ITGrundschutz/ITGrundschutzKataloge/Inhalt/_content/g/g04/g04022.html?nn=6605008 (abgerufen am 25.11.2015).

[Bun11] Bundesamt für Sicherheit in der Informationstechnik: *G 0.28 Software-Schwachstellen oder -Fehler*, 2011. https://www.bsi.bund.de/DE/Themen/ITGrundschutz/ITGrundschutzKataloge/Inhalt/_content/g/g00/g00028.html?nn=6604996 (abgerufen am 25.11.2015).

[Bun12] Bundesamt für Sicherheit in der Informationstechnik: *Lebenszyklus einer Schwachstelle*, März 2012. https://www.allianz-fuer-cybersicherheit.de/ACS/DE/_downloads/materialien/sensibilisierung/BSI-CS_027.pdf?__blob=publicationFile (abgerufen am 08.07.2015).

[Bun13] Bundesamt für Sicherheit in der Informationstechnik: *4 Glossar und Begriffsdefinitionen*, 2013. https://www.bsi.bund.de/DE/Themen/ITGrundschutz/ITGrundschutzKataloge/Inhalt/Glossar/glossar_node.html (abgerufen am 16.06.2015).

[Bun14a] Bundesamt für Sicherheit in der Informationstechnik: *Die Lage der IT-Sicherheit in Deutschland 2014*, Dezember 2014. https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/Lageberichte/Lagebericht2014.pdf?__blob=publicationFile (abgerufen am 05.06.2015).

[Bun14b] Bundesamt für Sicherheit in der Informationstechnik: *G 4.22 Software-Schwachstellen oder -Fehler*, 2014. https://www.bsi.bund.de/DE/Themen/ITGrundschutz/ITGrundschutzKataloge/Inhalt/_content/g/g04/g04022.html?nn=6605008 (abgerufen am 25.11.2015).

[Bun15a] Bundesamt für Sicherheit in der Informationstechnik: *IT-Grundschutz-Standards*, 2015. https://www.bsi.bund.de/DE/Themen/ITGrundschutz/ITGrundschutzStandards/ITGrundschutzStandards_node.html (abgerufen am 02.07.2015).

[Bun15b] Bundesministerium der Justiz und für Verbraucherschutz: *Hochschulrahmengesetz (HRG) §4 Freiheit von Kunst und Wissenschaft, Forschung, Lehre und Studium*, 2015. http://www.gesetze-im-internet.de/hrg/__4.html (abgerufen am 28.06.2015).

[Can15] Canonical Ltd.: *Ubuntu Security Notices*, September 2015. http://www.ubuntu.com/usn/ (abgerufen am 28.09.2015).

[Cat03] Cathleen, Brackin: *Vulnerability Management: Tools, Challenges and Best Practices*, Oktober 2003. http://www.sans.org/reading-room/whitepapers/threats/vulnerability-management-tools-challenges-practices-1267 (abgerufen am 28.07.2015).

[Cod14] Codenomicon: *The Heartbleed Bug*, April 2014. http://heartbleed.com/ (abgerufen am 22.07.2015).

[Dav12] David Waltermire, Charles Schmidt, Karen Scarfone, Neal Ziring: *Specification for the Extensible Configuration Checklist Description Format (XCCDF) Version 1.2*, März 2012. http://csrc.nist.gov/publications/nistir/ir7275-rev4/nistir-7275r4_updated-march-2012_clean.pdf (abgerufen am 08.11.2015).

[Dep15] Department of Homeland Security: *CDM Vulnerability Management (VUL) Capability*, 2015. https://www.us-cert.gov/sites/default/files/cdm_files/Intro_to_VUL.pdf (abgerufen am 16.06.2015).

[Deu14] Deutsches Forschungsnetz: *Das Wissenschaftsnetz X-WiN*, September 2014. https://www.dfn.de/xwin/ (abgerufen am 26.07.2015).

[DFN15] DFN-CERT Team: *DFN-CERT Portal Schwachstellen*, 2015. https://portal.cert.dfn.de/adv/ (abgerufen am 05.08.2015).

[EDU14] EDUCAUSE, Grama: *Just in Time Research: Data Breaches in Higher Education*, Mai 2014. https://net.educause.edu/ir/library/pdf/ECP1402.pdf (abgerufen am 05.06.2015).

[Exp15] Exploit Database: *Offensive Security Exploit Database Archive*, 2015. https://www.exploit-db.com/ (abgerufen am 06.08.2015).

[Fel13] Felix von Eye, Wolfgang Hommel, Stefan Metzger: *Dr. Portscan: Ein Werkzeug für die automatisierte Portscan-Auswertung in komplexen Netzinfrastrukturen*, 2013. https://git.lrz.de/?p=DrPortScan.git;a=blob;f=documents/DFN-CERT-Workshop-2013.pdf (abgerufen am 06.08.2015).

[For07] Forum of Incident Response and Security Teams: *A Complete Guide to the Common Vulnerability Scoring System Version 2.0*, Juni 2007. https://www.first.org/cvss/cvss-v2-guide.pdf (abgerufen am 13.07.2015).

[For09] Foreman, Park: *Vulnerability Management*. CRC Press, New York, 1 Auflage, 2009.

[For15a] Forum of Incident Response and Security Teams: *Common Vulnerability Scoring System v3.0: Examples*, 2015. https://www.first.org/cvss/examples (abgerufen am 19.11.2015).

[For15b] Forum of Incident Response and Security Teams: *Common Vulnerability Scoring System v3.0: Specification Document*, 2015. https://www.first.org/cvss/cvss-v30-specification-v1.7.pdf (abgerufen am 13.12.2015).

[For15c] Forum of Incident Response and Security Teams: *Common Vulnerability Scoring System Version 3.0 Calculator*, 2015. https://www.first.org/cvss/calculator/3.0#CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N (abgerufen am 30.07.2015).

[Gie14] Gieselmann: *US-Bericht: Über 80 Millionen Konten bei JPMorgan von Hacker-Angriff betroffen*, Oktober 2014. `http://www.heise.de/newsticker/meldung/US-Bericht-Ueber-80-Millionen-Konten-bei-JPMorgan-von-Hacker-Angriff-betroffen-2411422.html` (abgerufen am 04.06.2015).

[Gor15a] Gordon Lyon: *Full Disclosure Mailing List*, 2015. `http://seclists.org/fulldisclosure/` (abgerufen am 05.08.2015).

[Gor15b] Gordon Lyon: *NMAP.ORG*, 2015. `https://nmap.org/` (abgerufen am 06.08.2015).

[Gor15c] Gordon Lyon: *NMAP.ORG*, 2015. `https://nmap.org/nsedoc/scripts/` (abgerufen am 06.08.2015).

[Gor15d] Gordon Lyon: *NMAP.ORG*, 2015. `https://nmap.org/nsedoc/scripts/ssl-heartbleed.html` (abgerufen am 25.11.2015).

[Gru14] Grubb, Ben: *Heartbleed disclosure timeline: who knew what and when*, April 2014. `http://www.smh.com.au/it-pro/security-it/heartbleed-disclosure-timeline-who-knew-what-and-when-20140415-zqurk.html` (abgerufen am 22.07.2015).

[Int08] International Organization for Standardization, International Electrotechnical Commission: *Information technology - Security techniques - Information security management systems - Requirements*. International Organization for Standardization, International Electrotechnical Commission, 2008. ISO/IEC 27000:2005.

[Int14a] International Organization for Standardization, International Electrotechnical Commission: *Information Technology - Security Techniques - Information security management systems - Overview and Vocabulary*. International Organization for Standardization, International Electrotechnical Commission, 2014. ISO/IEC 27000:2014.

[Int14b] International Organization for Standardization, International Electrotechnical Commission: *Informationstechnik - IT-Sicherheitsverfahren - Leitfaden für das Informationssicherheits-Management*. International Organization for Standardization, International Electrotechnical Commission, 2014. ISO/IEC 27000:2014-02.

[Jak15a] Jake Kouns, Brian Martin: *Open Source Vulnerability Database*, September 2015. `http://osvdb.org/` (abgerufen am 07.09.2015).

[Jak15b] Jake Kouns, Brian Martin: *Open Source Vulnerability Database*, September 2015. `http://osvdb.org/faq` (abgerufen am 11.09.2015).

[Jos04] Joseph Shimanek: *Concepts and Successes in Vulnerability Management*, 2004. `http://www.giac.org/paper/gsec/3579/concepts-successes-vulnerability-management/105821` (abgerufen am 25.11.2015).

[Kis13] Kissel: *Glossary of Key Information Security Terms*, Mai 2013. http://nvlpubs.nist.gov/nistpubs/ir/2013/NIST.IR.7298r2.pdf (abgerufen am 16.06.2015).

[KMK12] Krebs, Rouven, Christof Momm und Samuel Kounev: *Architectural Concerns in Multi-tenant SaaS Applications.* CLOSER, 12:426–431, 2012.

[Lei15] Leibniz-Rechenzentrum der Bayerischen Akademie der Wissenschaften: *Überblick über das Münchner Wissenschaftsnetz (MWN)*, Juni 2015. https://www.lrz.de/services/netz/mwn-ueberblick/ (abgerufen am 19.06.2015).

[Mar10] Martin-Luther-Universtität Halle-Wittenberg: *Netzkonzept*, Mai 2010. http://wcms.itz.uni-halle.de/download.php?down=19093&elem=2453984 (abgerufen am 22.06.2015).

[Mar15] Markus Fürst: *Hierarchische Eskalation*, Oktober 2015. http://www.openadvice.de/service-management-gmbh/wissen/itilwissen/itil-glossar/itil-glossar/hierarchische-eskalation (abgerufen am 21.10.2015).

[Mic11] Microsoft Corporation: *Bewertungssystem für Security Bulletins des Microsoft Security Response Center (aktualisiert im Dezember 2011)*, Dezember 2011. https://technet.microsoft.com/de-de/security/gg309177.aspx (abgerufen am 28.07.2015).

[Mic15] Microsoft Corporation: *Microsoft Security Bulletin*, 2015. https://technet.microsoft.com/security/bulletin/ (abgerufen am 14.07.2015).

[MIT11] MITRE Corporation: *Handling Duplicate Public CVE Identifiers*, Oktober 2011. https://cve.mitre.org/cve/editorial_policies/duplicates.html (abgerufen am 22.07.2015).

[MIT12] MITRE Corporation: *Common Configuration Enumeration – CCE™*, Februar 2012. http://makingsecuritymeasurable.mitre.org/docs/cce-intro-handout.pdf (abgerufen am 08.11.2015).

[MIT13] MITRE Corporation: *About CPE - Archive*, März 2013. http://cpe.mitre.org/about/ (abgerufen am 13.07.2015).

[MIT14a] MITRE Corporation: *About OVAL*, Mai 2014. https://oval.mitre.org/about/ (abgerufen am 18.06.2015).

[MIT14b] MITRE Corporation: *CVE-2014-0160*, 2014. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-0160 (abgerufen am 22.07.2015).

[MIT14c] MITRE Corporation: *CVE-2014-3566*, 2014. http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-3566 (abgerufen am 27.10.2015).

[MIT14d] MITRE Corporation: *CVE Data Sources and Coverage*, Februar 2014. https://cve.mitre.org/cve/data_sources_product_coverage.html (abgerufen am 09.07.2015).

[MIT14e] MITRE Corporation: *Frequently Asked Questions*, Juli 2014. https://oval.mitre.org/about/faqs.html (abgerufen am 03.07.2015).

[MIT14f] MITRE Corporation: *Frequently Asked Questions (FAQ)*, Juli 2014. http://cwe.mitre.org/about/faq.html (abgerufen am 13.07.2015).

[MIT15a] MITRE Corporation: *CVE Editorial Board*, April 2015. https://cve.mitre.org/community/board/ (abgerufen am 07.08.2015).

[MIT15b] MITRE Corporation: *CVE-ID Syntax Change*, Juli 2015. https://cve.mitre.org/cve/identifiers/syntaxchange.html (abgerufen am 10.07.2015).

[MIT15c] MITRE Corporation: *CVE Numbering Authorities*, Februar 2015. https://cve.mitre.org/cve/cna.html (abgerufen am 07.08.2015).

[MIT15d] MITRE Corporation: *CVE Reference Key/Maps*, August 2015. http://cve.mitre.org/data/refs/index.html (abgerufen am 03.08.2015).

[MIT15e] MITRE Corporation: *Frequently Asked Questions*, Juli 2015. https://cve.mitre.org/about/faqs.html (abgerufen am 10.07.2015).

[MIT15f] MITRE Corporation: *Request a CVE Identifier*, Februar 2015. https://cve.mitre.org/cve/request_id.html (abgerufen am 07.08.2015).

[Mur13] Murugiah Souppaya, Karen Scarfone: *Guide to Enterprise Patch Management Technologies*, Juli 2013. http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-40r3.pdf (abgerufen am 02.07.2015).

[Mäu15] Mäurer, Nils: *Efficient scans in a research network*. Bachelorarbeit, Technische Universität München, München, Februar 2015.

[Nat13] National Institute of Standards and Technology: *Security and Privacy Controls for Federal Information Systems and Organizations*, April 2013. http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r4.pdf (abgerufen am 15.06.2015).

[Nat14] National Institute of Standards and Technology: *XCCDF – The Extensible Configuration Checklist Description Format*, Januar 2014. http://scap.nist.gov/specifications/xccdf/ (abgerufen am 08.11.2015).

[Nat15a] National Institute of Standards and Technology: *National Vulnerability Database*, Juli 2015. https://nvd.nist.gov/home.cfm (abgerufen am 12.07.2015).

[Nat15b] National Institute of Standards and Technology: *NIST Special Publications (SP)*, Oktober 2015. http://csrc.nist.gov/publications/PubsSPs.html (abgerufen am 07.11.2015).

[Nat15c] National Institute of Standards and Technology: *NVD Frequently Asked Questions*, Juli 2015. https://nvd.nist.gov/faq (abgerufen am 12.07.2015).

[Nat15d] National Institute of Standards and Technology: *Vulnerability Summary for CVE-2014-0160*, 2015. https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2014-0160 (abgerufen am 30.07.2015).

[Nat15e] National Institute of Standards and Technology: *Vulnerability Summary for CVE-2015-4872*, 2015. https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2015-4872 (abgerufen am 01.11.2015).

[Net14] Netzgruppe Universität Hamburg: *Informationen zum Netzkonzept*, Dezember 2014. https://www.rrz.uni-hamburg.de/services/netz/netzkonzept.html (abgerufen am 24.06.2015).

[Ope15a] Open Web Application Security Project: *OWASP TOP 10 – 2013 The Ten Most Critical Web Application Security Risks*, 2015. http://owasptop10.googlecode.com/files/OWASP%20Top%2010%20-%202013.pdf (abgerufen am 02.11.2015).

[Ope15b] OpenSSL Software Foundation: *Welcome to the OpenSSL Project*, 2015. https://www.openssl.org/ (abgerufen am 22.07.2015).

[Pet05] Peter Mell, Tiffany Bergeron, David Henning: *Creating a Patch and Vulnerability Management Program*, November 2005. http://csrc.nist.gov/publications/nistpubs/800-40-Ver2/SP800-40v2.pdf (abgerufen am 02.07.2015).

[ram06] ramprasadb: *cfengine*, Juli 2006. http://www.gnu.org/software/cfengine/ (abgerufen am 06.12.2015).

[Ruh09] Ruhr-Universität Bochum: *NETZ-Konzept Vernetzung Netzentwicklungsplan Netzbetrieb und Management für die Ruhr-Universität Bochum*, März 2009. https://noc.rub.de/web/_media/netzkonzept/vernetzungskonzept-2009.pdf (abgerufen am 22.06.2015).

[Sch14a] Schaaf, Thomas: *IT-Management*, 2014. http://www.nm.ifi.lmu.de/teaching/Vorlesungen/2014ss/itmgmt/Downloads/Skript/2.11_ITSM_COBIT_155-175.pdf (abgerufen am 19.06.2015).

[Sch14b] Schaaf, Thomas: *IT-Management*, 2014. http://www.nm.ifi.lmu.de/teaching/Vorlesungen/2014ss/itmgmt/Downloads/Skript/2.2_ITSM_Prozessorientierung_42-50.pdf (abgerufen am 19.06.2015).

[Sch14c] Schaaf, Thomas: *IT-Management*, 2014. http://www.nm.ifi.lmu.de/teaching/Vorlesungen/2014ss/itmgmt/Downloads/Skript/2.1_ITSM_Dienstbegriff_30-41.pdf (abgerufen am 10.08.2015).

[Sch14d] Schaaf, Thomas: *IT-Management*, 2014. http://www.nm.ifi.lmu.de/teaching/Vorlesungen/2014ss/itmgmt/Downloads/Skript/2.3_ITSM_Kernprozesse_51-60.pdf (abgerufen am 24.07.2015).

[Sch14e] Schaaf, Thomas: *IT-Management*, 2014. http://www.nm.ifi.lmu.de/teaching/Vorlesungen/2014ss/itmgmt/Downloads/Skript/4_ISM_326-370.pdf (abgerufen am 03.08.2015).

[Sch14f] Schaaf, Thomas: *IT-Management*, 2014. http://www.nm.ifi.lmu.de/teaching/Vorlesungen/2014ss/itmgmt/Downloads/Skript/2.4_ITSM_Prozesskontrolle_61-73.pdf (abgerufen am 21.10.2015).

[Sec15] SecurityFocus: *SecurityFocus*, 2015. http://www.securityfocus.com/ (abgerufen am 06.08.2015).

[Ste12] Stephen Quinn, Karen Scarfone, David Waltermire: *Guide to Adopting and Using the Security Content Automation Protocol (SCAP) Version 1.2 (Draft)*, Januar 2012. http://csrc.nist.gov/publications/drafts/800-117-R1/Draft-SP800-117-r1.pdf (abgerufen am 02.07.2015).

[Ten13a] Tenable Network Security: *Implementierung eines effektiven Schwachstellenmanamgent-Systems*, 2013. http://documents.sysob.com/tenable/tenable_wp_Implementing_DE.pdf (abgerufen am 05.12.2015).

[Ten13b] Tenable Network Security: *Implementing an Effective Vulnerability Management Program*, 2013. https://www.brighttalk.com/webcast/7927/80093?utm_campaign=webcasts-search-results-feed&utm_content=tenable+implementing+vulnerability+management&utm_source=brighttalk-portal&utm_medium=web&utm_term= (abgerufen am 05.12.2015).

[THSC04] Tian, HT, LS Huang, JL Shan und GL Chen: *Automated vulnerability management through web services*. In: *Grid and Cooperative Computing*, Seiten 1067–1070. Springer, 2004.

[Tom13] Tom Palmaers: *Implementing a Vulnerability Management Process*, 2013. https://www.sans.org/reading-room/whitepapers/threats/implementing-vulnerability-management-process-34180 (abgerufen am 03.12.2015).

[Uni15a] United States Computer Emergency Readiness Team: *National Cyber Awareness System*, 2015. https://www.us-cert.gov/ncas (abgerufen am 05.08.2015).

[Uni15b] Universität Bamberg: *Die Universität Bamberg in Zahlen (2012-2014)*, Mai 2015. http://www.uni-bamberg.de/kommunikation/die-universitaet-bamberg-in-zahlen/ (abgerufen am 27.07.2015).

[Uni15c] Universität Bamberg: *Protokolle, Strukturen und Konzepte*, Juni 2015. http://www.uni-bamberg.de/?id=76689 (abgerufen am 22.06.2015).

[Uni15d] Universität Hamburg: *Zahlen und Fakten*, Juni 2015. https://www.uni-hamburg.de/uhh/fakten.html (abgerufen am 27.07.2015).

[Vaa15] Vaarandi, Risto: *SEC - simple event correlator*, 2015. http://simple-evcorr.github.io/ (abgerufen am 06.08.2015).

[WG09] Wang, Ju An und Minzhe Guo: *OVM: an ontology for vulnerability management*. In: *Proceedings of the 5th Annual Workshop on Cyber Security and Information Intelligence Research: Cyber Security and Information Intelligence Challenges and Strategies*, Seite 34. ACM, 2009.

[WH15] Wolfgang Hommel, Helmut Reiser: *Das Münchner Wissenschaftsnetz (MWN) Konzepte, Dienste, Infrastruktur und Management*, Mai 2015. `https://www.lrz.de/services/netz/mwn-netzkonzept/mwn-netzkonzept-2015.pdf` (abgerufen am 19.06.2015).

[Yak96] Yakov Rekhter et al.: *Address Allocation for Private Internets*. RFC 1918 (Best Current Practice), Februar 1996.

Anhang

1 Umfragebogen

Umfrage: Der Umgang mit Schwachstellen im Münchner Wissenschaftsnetz am Leibniz-Rechenzentrum

im Rahmen einer Masterarbeit

„Schwachstellenmanagement in Hochschulnetzen am Beispiel des Münchner Wissenschaftsnetzes“

Umfragesteller: Michael Steinke (Michael.Steinke@lrz.de)

Fokus Softwareschwachstellen und Fehlkonfiguration

→ Fehler im Design, Implementation oder Kontrollen, durch deren Ausnutzung Schaden entstehen kann

---

Ziel der Umfrage?

Das Ziel dieser Umfrage ist es, den aktuellen Umgang mit Schwachstellen in Software und Konfiguration innerhalb des Münchner Wissenschaftsnetzes am LRZ zu ermitteln, um darauf aufbauend **organisatorische** sowie **technische** Anforderungen an ein Schwachstellenmanagementsystem erstellen zu können.

Was ist mit Schwachstellenmanagement gemeint?

Schwachstellenmanagement ist die zyklische Ausübung der Identifikation, Bewertung, Beseitigung und Abschwächung von Schwachstellen.

Identifikation	Dokumentation von Schwachstellen
Bewertung	Einstufung der Kritikalität einer Schwachstelle
Beseitigung und Abschwächung	Detektion von Schwachstellen und Maßnahmen zur Verhinderung der Ausnutzung einer Schwachstelle

An wen richtet sich diese Umfrage?

Diese Umfrage richtet sich an Mitarbeiter des Leibniz-Rechenzentrums, die eine oder mehrere Tätigkeiten (Identifikation, Bewertung, Beseitigung und Abschwächung) des Schwachstellenmanagements ausüben.

Name _____ (optional)

Abteilung/ Gruppe _____

Organisatorisches…

Wann befassen Sie sich mit der Behandlung von Schwachstellen?

- ☐ regelmäßig/ geplant
- ☐ spontan

Wie oft befassen Sie sich geschätzt mit der Behandlung von Schwachstellen?

- ☐ Einmal pro Monat
- ☐ Mehrmals im Monat
- ☐ Als Reaktion auf ein Security Event
- ☐ ca. _____ Mal pro _____
- ☐ Vor der Inbetriebnahme neuer Geräte/ Software
- ☐ Bei größeren Konfigurationsänderungen
- ☐ _____

Auf welchen Systemen suchen Sie nach Schwachstellen?

- ☐ Auf meinem Arbeitsplatzrechner
- ☐ Auf den Computersystemen, die ich administriere/ betreue
- ☐ Je nach Bedarf im gesamten Münchner Wissenschaftsnetz (z.B. durch Scan-Tools)
- ☐ Als Netzverantwortlicher in meinen Subnetzen:

Subnetze:_____

- ☐ Für einen oder mehrere Netzverantwortliche in ihren Subnetzen:

Subnetze:_____

- ☐ Nur auf Servern, die als Basis für IT-Dienste im MWN dienen
- ☐ _____

Wer ist beteiligt?

☐ Ich spreche mich mit Kollegen ab, bevor ich auf Computersystemen nach Schwachstellen suche

☐ bzgl. Aufteilung der Tätigkeit

☐ bzgl. möglicher Beeinträchtigungen der Systeme

☐ bzgl. _____

☐ In der Regel nur ich

Bei Identifikation einer Schwachstelle auf einem System

☐ informiere ich den Systemadministrator/ Netzverantwortlichen/ Owner

☐ behandle ich die Schwachstelle in der Regel selber

☐ _____

Sich über Schwachstellen informieren

Aus welchen Quellen erfahren Sie von neuen Schwachstellen?

☐ National Vulnerability Database (NISTs NVD)

☐ DFN-CERT als E-Mail Benachrichtigung

☐ Ich suche selber auf dem DFN-CERT Portal

☐ US-CERT

☐ Aus der Zeitung

☐ Meine Tools holen ihre Schwachstelleninformationen selber (z.B. OpenVAS)

☐ Ich suche selber nach Fehlern (z.B. in Konfigurationsdateien, Quelltext)

☐ Andere

_____

_____

Wonach entscheiden Sie, ob eine Schwachstelle relevant ist, bzw. die Notwendigkeit einer Behandlung besteht?

- ☐ Nach Auswirkung der Schwachstelle bzw. falls vorhanden, anhand des CVSS-BaseScores
- ☐ Nach Kritikalität des betroffenen Services
- ☐ Nach Behandlungsaufwand
- ☐ Jede Schwachstelle sollte zeitnah geschlossen werden
- ☐ Ich habe keine Kriterien bzgl. der Behandlungsnotwendigkeit – besser irgendwas schließen als gar nichts!
- ☐ Andere

_____

_____

Suche nach Schwachstellen auf Computersystemen/ Assets

Haben Sie ein Asset-Inventar, d.h. eine Übersicht über die Geräte, die Sie scannen wollen?

- ☐ Ja, in Form einer
  - ☐ Datenbank
  - ☐ Textdatei
  - ☐ Excel-Tabelle
  - ☐ _____
- ☐ Nein

Welche Informationen hält Ihr Asset-Inventar über die einzelnen Geräte?

- ☐ IP-Adresse
- ☐ Owner
- ☐ Beschreibung des Systems (z.B. „Server für Service XY“)
- ☐ Zeitstempel: Zuletzt auf Schwachstellen gescannt
- ☐ _____
- ☐ _____
- ☐ _____
- ☐ _____

Wie viele Systeme betrachten Sie bei der Behandlung von Schwachstellen?

☐ _____ Stück

☐ ungefähr _____ Stück

☐ alle innerhalb der Subnetze _____

Welche Methoden nutzen Sie zum Finden von Schwachstellen auf Computersystemen?

☐ Anhand eines Abgleichs der Version des Softwarepakets mit betroffenen Versionen

☐ Penetration Test

☐ Port Scan (z.B. zum Prüfen der Verwendung von http (unverschlüsselt!) oder in Verbindung der Rolle der Maschine: z.B. Ein Datenbankserver sollte in der Regel Port 3306 offen haben und nicht Port 80)

☐ _____

☐ _____

☐ _____

Welche Softwaretools verwenden Sie?

☐ nmap → Parameter: _____

☐ OpenVAS

☐ Nessus

☐ Secunia PSI

☐ Valgrind (Memcheck, Addrcheck, …)

☐ _____

☐ _____

☐ _____

☐ _____

Behandlung von identifizierten Schwachstellen

Welche Maßnahmen zur Beseitigung von Schwachstellen ergreifen Sie bzw. ziehen Sie in Betracht?

- ☐ Installation des jeweiligen Patches (falls verfügbar)
- ☐ Allgemeine Aktualisierung der Software (bzw. neuere SW-Version verwenden, falls vorhanden)
- ☐ Je nach Schwachstelle: Installation weiterer Sicherheitsmechanismen (z.B. Autorisierung, Authentifizierung, Zugangs- und Zugriffskontrolle)
- ☐ Anwendung eines Workarounds (z.B. aus einem Schwachstellen-Advisory)
- ☐ Austausch der Software
- ☐ Stilllegung des Rechners
- ☐ _____
- ☐ _____
- ☐ _____
- ☐ _____

Reporting – Im Falle, dass Sie Rechner anderer Personen scannen: wie melden Sie Schwachstellen an Betroffene?

Kündigen Sie Schwachstellenscans „fremder“ Systeme beim jeweiligen Administrator/ Netzverantwortlichen im Voraus an?

☐ ja ☐ nein

Falls Sie Schwachstellen auf Rechnern gefunden haben, die Sie nicht administrieren, wie informieren Sie die jeweiligen Owner?

- ☐ über E-Mail
- ☐ telefonisch
- ☐ Ticket
- ☐ _____

Welche Informationen über gefundene Schwachstellen stellen Sie zur Verfügung?

- ☐ Name/ Bezeichner der Schwachstelle (z.B. „CVE-2014-160“ oder „Heartbleed Bug“)
- ☐ Schwachstellenbeschreibung
- ☐ CVSS-Score
- ☐ Betroffene Softwarepakete
- ☐ Identifier (z.B. IPv4 Adresse, Hostname) des betroffenen Geräts
- ☐ evtl. Link auf/ Informationen zu Patch oder Workaround
- ☐ _____

Wie behandeln Sie Rückfragen bzw. Supportanfragen der Betroffenen?

- ☐ Ich biete ihnen in der Regel Unterstützung bei der Behebung der Schwachstellen
  - ☐ per E-Mail
  - ☐ telefonisch
  - ☐ Remote Desktop
  - ☐ SSH
  - ☐ _____
- ☐ Aus zeitlichen Gründen kann ich keine Unterstützung bieten, sondern sie nur informieren
- ☐ Es kommen keine Rückfragen
- ☐ Es kommen in der Regel zu viele Rückfragen für eine Unterstützung

Anforderungen/ Wünsche an ein Schwachstellenmanagementsystem

Ein Schwachstellenmanagementsystem umfasst insbesondere in Wechselwirkung stehende Richtlinien, Verantwortlichkeiten, Anleitungen, Verfahren und Betriebsmittel (Personal, Hardware, Software, …), die das Schwachstellenmanagement realisieren.

Anforderungen beziehen sich auf die Elemente des Systems und können organisatorisch oder auch technisch sein:

Beispiele für organisatorische Anforderungen:

- Die Suche nach Schwachstellen auf Computersystemen muss dezentral z.B. durch Netzverantwortliche möglich sein.
- Die Suche muss möglichst einfach durchführbar sein.
- Automatische Scans nur zu bestimmten Uhrzeiten.
- Unterstützung bei der Behandlung von Schwachstellen durch …
- Angebot einer Schwachstellenhistorie pro Gerät
- Darstellung der Scanergebnisse insofern, dass …
- …

Beispiele für technische Anforderungen:

- Automatische Inventarisierung neuer Schwachstellen
- Möglichkeit der automatischen Überprüfung auf Schwachstellen anhand der Softwareversion
- Möglichkeit eines Schwachstellenscans über das Netzwerk
- Scannen mit Hilfe von …
- …

Falls Sie Anforderungen an ein Schwachstellenmanagementsystem im Hochschulbereich haben, können Sie diese hier angeben:

_____

_____

_____

_____

_____

_____

_____

_____

2 Datenbankschema des Konzepts

In diesem Abschnitt wird das Datenbankschema der Implementierung aus Kapitel 6 gezeigt. Die Erklärung zu den Tabellen findet sich in Abschnitt 6.2.2. Tabellen mit oranger Markierung werden von der Asset- und Benutzerverwaltung organisiert, die restlichen durch die Schwachstellen-Dokumentation.


fremdquellen
ID: INT (PK)
TITEL: VARCHAR
QUELLENVERANTWORTLICHER: INT
QUELLE: VARCHAR
BENUTZER: VARCHAR
PASSWORT: VARCHAR
BEWERTUNGSQUELLE: INT
KATEGORISIERUNGSQUELLE: INT
DETEKTIONSQUELLE: INT
UPDATEINTERVALL: INT
ZPLIMPORT: TIMESTAMP
bewertungsquellen
ID: INT (PK)
REFERENZ: VARCHAR
USER: VARCHAR
PASSW: VARCHAR
kategorisierungsquellen
ID: INT (PK)
REFERENZ: VARCHAR
USER: VARCHAR
PASSW: VARCHAR
detektionsquellen
ID: INT (PK)
REFERENZ: VARCHAR
USER: VARCHAR
PASSW: VARCHAR
blacklist_global
ID: INT (PK)
SOFTWAREPRODUKT: INT
blacklist_lokal
ID: INT (PK)
SOFTWAREPRODUKT: INT
FREMDQUELLE: INT
softwareprodukte
ID: INT (PK)
HERSTELLER: VARCHAR
BEZEICHNER: VARCHAR
VERSION: VARCHAR
compiler
ID: INT (PK)
SOFTWAREPRODUKT: INT
DETEKTIONSVERFAHREN: INT
abhaengig_von
ID: INT (PK)
DETEKTIONSVERFAHREN: INT
SOFTWAREPRODUKT: INT
detektionsverfahren
ID: INT (PK)
SCHWACHSTELLE: INT
BESCHREIBUNG: VARCHAR
TYP: ENUM(netz, system)
PROGRAMMIERSPRACHE: VARCHAR
EDITOR: INT
ABNAHME_DURCH: INT
FREIGEGEBEN: BOOLEAN
plattformen
ID: INT (PK)
DETEKTIONSVERFAHREN: INT
SOFTWAREPRODUKT: INT
massnahmen
ID: INT (PK)
SCHWACHSTELLE: INT
BESCHREIBUNG: VARCHAR
TYP: VARCHAR
EDITOR: INT
schwachstellen
ID: INT (PK)
BESCHREIBUNG: VARCHAR
BEWERTUNG: ENUM(NA, gering, mittel, hoch, kritisch)
TEMP_DRINGLICHKEIT: ENUM(NA, gering, mittel, hoch, kritisch)
FREMDQUELLE: INT
FREMDQUELLENID: VARCHAR
EDITOR: INT
STATUS: ENUM(check, reject, assess, incomplete, categorize, detect, complete)
VEROEFFENTLICHT: TIMESTAMP
GEAENDERT: TIMESTAMP
TEMP_DRINGLICHKEITTS: TIMESTAMP
referenzen
ID: INT (PK)
QUELLE: VARCHAR
QUELLENID: VARCHAR
SCHWACHSTELLE: INT
MASSNAHME: INT
EDITOR: INT
ist_betroffenes_produkt_kette
ID: INT (PK)
SOFTWAREPRODUKT: INT
NAECHSTES_ELEMENT: INT
ist_betroffenes_produkt
SCHWACHSTELLE: INT
SWP_KETTE: INT
ist_verantwortlich
SCHWACHSTELLE: INT
NUTZER: INT
ist_asset_spezifisch
SCHWACHSTELLE: INT
ASSET: INT
meldungstickets
ID: INT (PK)
KONTAKT: VARCHAR
SOFTWAREPRODUKT_DIENST: VARCHAR
HERSTELLER: VARCHAR
VORGEHEN: VARCHAR
SCHWACHSTELLE: INT
BEARBEITER: INT
ERSTELLT: TIMESTAMP
GESCHLOSSEN: TIMESTAMP
vulmgmt_config
SCHWACHSTELLENAKZEPTANZKRIERIUM: ENUM(NA, gering, mittel, hoch, kritisch)
STD_UPDATEINTERVALL: INT
STD_DETEKTIONSINTERVALL: INT
TEMP_DRINGLICHKEIT_RESETINTERVALL: INT
CLEANUP_INTERVALL: INT
LAST_CLEANUP: TIMESTAMP
SCHWTICKET_MAX_ALTER: INT
schwachstellentickets
ID: INT (PK)
SCHWACHSTELLE: INT
ASSET: INT
ERSTELLT: TIMESTAMP
GESCHLOSSEN: TIMESTAMP
WORKAROUND: INT
MASSNAHME: INT
BEHEBUNGSNOTWENDIGKEIT: ENUM(gering, mittel, hoch, kritisch)
ESKALATIONSLEVEL: ENUM(0,1,2)
assets
ID: INT (PK)
NETZID: VARCHAR
BESCHREIBUNG: VARCHAR
KRITIKALITAET: ENUM(gering, mittel, hoch, kritisch)
INSTITUT: INT
TYP: INT
BETRIEBSSYSTEM: INT
VERANTWORTLICHER: INT
GEAENDERT: TIMESTAMP
DETEKTIONSINTERVALL: INT
ZPLDETEKT: TIMESTAMP
DIENST: INT
softwareinventar
ASSET: INT
SOFTWAREPRODUKT: INT
institut
ID: INT (PK)
BEZEICHNUNG: VARCHAR
nutzer
ID: INT (PK)
NAME: VARCHAR
KONTAKT: VARCHAR
INSTITUT: INT
MAILBENACHRICHTIGUNG: BOOLEAN
dienstmap
NUTZER: INT
DIENST: INT
rollenverteilung
NUTZER: INT
ROLLE: INT
rollen
ID: INT (PK)
BEZEICHNUNG: VARCHAR
dienst
ID: INT (PK)
BEZEICHNUNG: VARCHAR
geraetetypen
ID: INT (PK)
BEZEICHNUNG: VARCHAR
betriebssystem
ID: INT (PK)
BEZEICHNUNG: VARCHAR
ist_behebungsverantwortlich
ASSET: INT
NUTZER: INT
HAUPTVERANTWORTLICH: BOOLEAN
ist_detektionsverantwortlich
ASSET: INT
NUTZER: INT
HAUPTVERANTWORTLICH: BOOLEAN