INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Bachelorarbeit

Konzeption und Implementierung einer webbasierten Security Incident Management Software

Aaron Schweiger

INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Bachelorarbeit

Konzeption und Implementierung einer webbasierten Security Incident Management Software

Aaron Schweiger

Aufgabensteller:	Priv. Doz. Dr. H. Reiser
Betreuer:	Stefan Metzger Daniela Pöhn
Abgabetermin:	01. Juli 2014

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

München, den 1. Juli 2014

.....
(Unterschrift des Kandidaten)

Abstract

Die Zunahme der Cyberkriminalität erfordert neben der Verbesserung von proaktiven Schutzvorkehrungen auch eine Optimierung reaktiver Maßnahmen, um auf sicherheitsrelevante Vorfälle schnellst- und bestmöglich reagieren zu können. Hierbei ist eine standardisierte Vorgehensweise anhand eines Security Incident Response (SIR) Prozesses in der Praxis üblich.

Im Rahmen dieser Arbeit wird eine webbasierte Information Security Incident Managementsoftware für das Leibniz-Rechenzentrum (LRZ) konzipiert und implementiert, die alle Phasen des SIR-Prozesses digital abbildet. Die Webanwendung ermöglicht eine standardisierte Erfassung und eine workfloworientierte Bearbeitung von Sicherheitsvorfällen. In diesem Zusammenhang stellt die Software u.a. einen Priorisierungsautomatismus bereit, lässt eine vereinfachte Behandlung von Standard Security Incidents zu und sieht die Erfassung von Post Incident Review Ergebnissen vor. Das System führt automatisch eine feingranulare Dokumentation aller Prozessschritte inklusive der Kommunikationsvorgänge sowie eine Messung der Prozessperformance durch und vereinfacht die Meldung von Security Incidents an die Allianz für Cyber-Sicherheit. Export-, Filter- und Auswertungsfunktionen runden das Spektrum des Managementwerkzeugs ab.

Die Software ermöglicht dem LRZ-CSIRT nicht nur eine prozesskonforme Vorfallbearbeitung, sondern erleichtert auch die Identifikation von Security Trends durch die Auswertung von KPIs und trägt letztlich zu einem noch professionelleren und zugleich effizienteren Umgang mit Sicherheitsvorfällen bei.

Inhaltsverzeichnis

1. Einleitung

Die allgegenwärtige Verfügbarkeit des Internets und die weite Verbreitung von Smartphones und Tablets haben das klassische Nutzerverhalten im Sinne einer weitestgehend lokalen, Schreibtisch zentrierten Webnutzung mittels Desktopcomputer oder Laptop stark verändert. Der Ausbau des mobilen Highspeed-Internets und die steigende Zahl öffentlicher WLAN-Hotspots in Bahnhöfen, Cafés und Hotels beschleunigen den Trend des *Ubiquitous Computings* [Wei91]. Die heutige Gesellschaft ist immer und überall online, sogar über Endgerätegrenzen hinweg. Digitale Geräte werden „simultan und nahtlos in das Alltagshandeln integriert“ [Pip12]. Einen bedeutenden Anteil an dieser Entwicklung trägt das *Cloud Computing*. Speicherdienste wie Dropbox oder Google Drive bieten eine zentrale Datenhaltung in Rechenzentren und ermöglichen dadurch den Nutzern eine standort- und geräteunabhängige Zugriffsmöglichkeit auf deren persönliche Dokumente, Fotos und Musik. Zu den Cloud-Dienstleistern zählen ebenfalls die E-Mail-Provider und die Anbieter von Webapplikationen (Software-as-a-Service). Auch in Unternehmen verliert das Desktop-Paradigma an Bedeutung. Im Rahmen von Bring Your Own Device (BYOD)-Policies bieten immer mehr Firmen ihren Mitarbeitern an, private Geräte auch am Arbeitsplatz zu nutzen. [BIT13]

1.1. Motivation

Das rasante Voranschreiten der Technologietrends hat nicht nur Veränderungen im Nutzerverhalten zur Folge, sondern auch die Notwendigkeit, den Sicherheits- und Datenschutzaspekten einen besonders hohen Stellenwert beizumessen.

Dass selbst große Softwarehersteller nicht ausreichend gegen Sicherheitsangriffe geschützt sind, zeigt der im Oktober 2013 bekannt gewordene Fall bei Adobe. Hacker sind in das Adobe-Netzwerk eingedrungen und haben sich nicht nur Zugriff auf Sourcecode verschafft [Ark13a], sondern auch persönliche Daten und Kreditkartendaten von 2,9 Millionen Kunden kopiert [Ark13b]. Darüber hinaus wurden rund 127 Millionen Benutzername-Passwort-Kombinationen ausgelesen. Der durch diesen Information Security Incident entstandene Schaden ist enorm und nicht zu beziffern.

Die Folgen des in den Medien [Hei13] [Fra13] [Stü13] [Klu13] ausführlich diskutierten Hacks wären deutlich geringer ausgefallen, wenn Adobe die Passwörter nicht mit einem symmetrischen Verschlüsselungsverfahren, sondern in gehashter Form gespeichert hätte. Durch die Rekonstruierbarkeit der verschlüsselten Passwörter sind auch Accounts bei weiteren Internetdiensten in Gefahr, bei denen die gleichen Benutzername-Passwort-Kombinationen verwendet wurden.

Information Security Incidents treten häufig in kleineren Dimensionen auf, beispielsweise in Form von gehackten virtuellen Maschinen, externen SSH-Attacken und mit Malware infizierten Computern. Die Bearbeitung derartiger Sicherheitsvorfälle ist Aufgabe des Information Security Incident Managements.

Sicherheitsvorfälle können zum Ausfall oder zur Beeinträchtigung der Qualität der jeweiligen IT-Services führen. Die schnellstmögliche Wiederherstellung der betroffenen Dienste ist ein wesentliches Ziel des Information Security Incident Managements, das ebenfalls zur Minimierung der Ausfallzeit und Begrenzung des entstandenen Schadens beiträgt. Liegen Incidents im Bereich des Abgreifens von sensiblen unternehmensinternen Informationen oder im Bereich des Identitätsdiebstahls durch das Erlangen von personenbezogenen Daten und Passwörtern, so zielt das Information Security Incident Management auf die Schadenbegrenzung, Beweismittelsicherung und Kooperation mit den Strafverfolgungsbehörden ab. Die Möglichkeiten der IT-Forensik können zur Aufklärung von Sicherheitsvorfällen beitragen. Dabei wird eine „streng methodisch vorgenommene Datenanalyse auf Datenträgern und in Computernetzen“ [Bun11] durchgeführt, um Spuren zu sammeln und auszuwerten, den Tatbestand festzustellen und den Angreifer zu identifizieren. [Ges11]

In größeren Unternehmen befassen sich spezielle IT-Sicherheitsgruppen – sog. Computer Security Incident Response Teams (CSIRT) – mit der Bearbeitung von Sicherheitsvorfällen. Diese Teams verfahren nach einem Security Incident Response (SIR) Prozess, der alle sicherheitsrelevanten Vorfälle über ihre ganzen Lebenszyklen betrachtet. Dabei stehen die Erfassung, Bewertung und Beseitigung des Problems im Vordergrund. Eine detaillierte Dokumentation der Incidentbearbeitung ermöglicht eine größtmögliche Nachvollziehbarkeit der Vorgehensweise, welche bei Post Incident Reviews, der Auswertung von Schlüsselkennzahlen und im Rahmen der Beweisführung von Bedeutung sind. Gerade die Dokumentation der Problemlösung kann im Sinne einer Wissensdatenbank für die künftige Bearbeitung ähnlicher Sicherheitsvorfälle hilfreich sein.

1.2. Zielsetzung

Am Leibniz-Rechenzentrum (LRZ) werden momentan Microsoft Word- und Excel-Dokumente als Hilfsmittel zur Dokumentation von Information Security Incidents eingesetzt, welche zwar eine strukturierte Bearbeitung der Vorfälle ermöglichen, aber Defizite insbesondere im Bereich der Auswertung aufweisen. So sind beispielsweise die automatisierte Erstellung von Statistiken sowie die Identifikation von Security-Trends nur sehr schwer möglich. Auch die effiziente Durchsuchbarkeit der abgeschlossenen Information Security Incidents nach vergleichbaren Vorfällen ist praktisch unmöglich.

Zielsetzung dieser Arbeit ist die Entwicklung eines webbasierten Tools zur Optimierung des Information Security Incident Managements. Das LRZ folgt dabei dem „tools follow processes“-Ansatz, wobei keine Standardlösung, die Prozessanpassungen erfordern würde, sondern ein individuell konzipiertes, den bestehenden Managementprozess abbildendes Softwaretool zum Einsatz kommen soll.

1.3. Aufgabenstellung

Im Rahmen dieser Arbeit soll eine webbasierte Information Security Incident Managementsoftware für das LRZ konzipiert und implementiert werden. Die Webanwendung soll der Dokumentation, Verwaltung und Auswertung von Sicherheitsvorfällen am LRZ dienen und das CSIRT bei der effizienten Umsetzung des Information Security Incident Managementprozesses unterstützen.

Die Webapplikation soll sich inhaltlich an den vorhandenen Dokumentationsvorlagen orientieren und die Vorgaben der Prozessbeschreibung [MHR11] abbilden. Daher liegt der Fokus dieser Arbeit weniger auf inhaltlichen Fragen. Vielmehr sind hier die Gestaltung und Umsetzung eines webbasierten Managementsystems von zentraler Bedeutung. Den Funktionsumfang betreffend werden insbesondere folgende Merkmale gefordert:

- Die Erfassung von Sicherheitsvorfällen mittels Webformular muss derart gestaltet sein, dass eine spätere Ergänzung und Korrektur der Informationen möglich ist. Die Computer Security Incident Response Team (CSIRT)-Mitglieder müssen den aktuellen Informationsstand eines Security Incidents einsehen und per E-Mail versenden können.
- Die Unterscheidung zwischen regulären Security Incidents und Standard Security Incidents soll den Umfang der von Hand einzugebenden Informationen reduzieren.
- Die Ergebnisse eines Post Incident Reviews, sollen in der Webapplikation unter Zuordnung zum jeweiligen Security Incident erfasst werden können.
- Eine Such-/Filterfunktion soll die parameterbasierte Auswahl von Security Incidents ermöglichen. Ferner wird eine einfache statistische Auswertungsmöglichkeit gefordert.
- Die in der Webapplikation verwalteten Daten sollen mittels einer Exportfunktion in mehreren Dateiformaten lokal gespeichert werden können. In diesem Zusammenhang soll eine spezielle Exportvorlage erstellt werden, welche die Meldung von Security Incidents an die Allianz für Cyber-Sicherheit vereinfacht.

Hinsichtlich der Softwarekonzeption und Datenmodellierung wird auf eine produkt- und plattformunabhängige Arbeitsweise Wert gelegt. Zudem wird die Lauffähigkeit der Webapplikation auf einem Standard-Webserver des LRZ verlangt.

1.4. Struktur dieser Arbeit

In Kapitel 2 werden Security Incident relevante Grundlagen zur Informationssicherheit behandelt. Anschließend wird im 3. Kapitel der konkrete Information Security Incident Managementprozess des LRZ betrachtet, woraus Optimierungsmöglichkeiten abgeleitet werden. Die Ergebnisse fließen in die in Kapitel 4 durchgeführte Analyse der funktionalen und nichtfunktionalen Anforderungen an ein softwarebasiertes Managementsystem ein. Die folgenden Kapitel befassen sich mit dem Entwicklungsprozess einer webbasierten Security Incident Managementsoftware: Die aus der Anforderungsanalyse abgeleitete plattformneutrale Konzeption der Software sowie die Datenmodellierung werden in Kapitel 5 beschrieben. In Kapitel 6 wird die Implementierung der Webanwendung dokumentiert. Schließlich kommt es in Kapitel 7 zur prototypischen Anwendung der Webanwendung anhand eines fiktiven Security Incidents. Das 8. Kapitel fasst die Ergebnisse dieser Arbeit zusammen und gibt einen Ausblick auf Erweiterungsmöglichkeiten der Webanwendung.

2. Grundlagen

Das Information Security Management (ISM) zielt als an Bedeutung zunehmender Teilbereich des IT Service Managements (ITSM) darauf ab, „die Informationssicherheit sowie die darin enthaltene IT-Sicherheit auf die Unternehmensbedürfnisse auszurichten und diese entsprechend zu überwachen.“ [Kle13] Der ISM-Prozess umfasst im Wesentlichen:

- die Erstellung und Lenkung der Leitlinien im Bereich der Informationssicherheit,
- die Analyse von sicherheitstechnischen Bedrohungen, Schwachstellen und Risiken,
- die Planung, Einführung und den Betrieb von präventiven, detektiven und reaktiven Schutzmaßnahmen,
- die Behandlung von Information Security Incidents sowie
- die Überwachung, Bewertung und Optimierung des Security Managements.

Das im Fokus dieser Arbeit stehende *Security Incident Management (SIM)* stellt eine reaktive Schutzmaßnahme organisatorischer Art dar. Die nachfolgende Abbildung veranschaulicht dessen Einordnung in den ITSM Kontext.

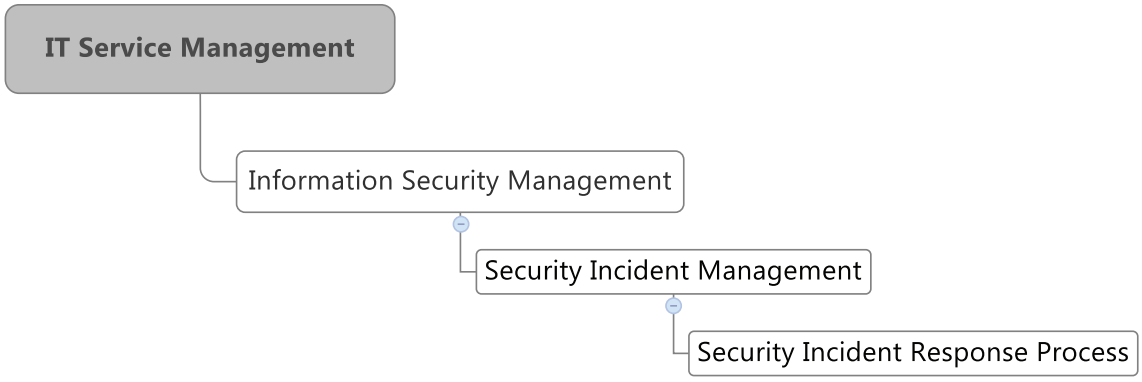

Abbildung 2.1.: Einordnung des Security Incident Managements in den ITSM Kontext

Hinsichtlich der Begrifflichkeiten sei darauf hingewiesen, dass anstelle der präzisen Begriffe „Information Security Incident“ bzw. „Computer Security Incident“ der besseren Lesbarkeit dieser Arbeit wegen meist die Kurzform „Security Incident“ verwendet wird, womit jedoch stets ein informationstechnischer Sicherheitsvorfall gemeint ist. Die Definition der Begrifflichkeit „Information Security Incident“ erfolgt in Abschnitt 2.2.

Die Literatur zeigt, dass für die Behandlung von Sicherheitsvorfällen sowohl nationale als auch internationale Standards und Empfehlungen existieren. Dieses Kapitel gibt einen Überblick über Maßnahmen und Ziele des Security Incident Managements unter Einbeziehung der zunächst erläuterten Standards.

2.1. Nationale und internationale Standards

Im Folgenden werden die für das Management von Sicherheitsvorfällen relevanten Standards zusammenfassend dargestellt. Während auf nationaler Ebene die Empfehlungen des Bundesamts für Informationssicherheit (BSI) betrachtet werden, findet im internationalen Umfeld neben der IT Infrastructure Library (ITIL) und den Ausführungen des amerikanischen National Institute of Standards and Technology (NIST) insbesondere die Normenreihe ISO/IEC 27000 Berücksichtigung. Auf die Leitlinien der Europäischen Agentur für Netz- und Informationssicherheit (ENISA) wird nicht näher eingegangen, da diese im Wesentlichen mit den zuvor genannten Standards übereinstimmen.

2.1.1. ITIL 2011

Die ITIL ist eine prozessfokussierte Sammlung von Best Practices für die ITSM-Umsetzung in Unternehmen und findet internationale Anwendung. Das „Information Security Management“ ist ein Bestandteil der ITIL 2011 Publikation „Service Design“ und soll den Schutz aller Daten und IT-Services einer Organisation hinsichtlich ihrer Vertraulichkeit, Integrität und Verfügbarkeit sicherstellen. ITIL ist als Framework zu sehen, das einen Überblick über die wichtigsten Aspekte des Information Security Managements gibt, diese jedoch nicht im Detail beschreibt, da hierfür spezielle Standards (z.B. ISO/IEC 27001, 27002) existieren.

2.1.2. ISO/IEC 27000 Normenreihe

Die internationale Normenreihe ISO/IEC 27000 versucht, die Grundprinzipien des Qualitätsmanagements auf das Management der Informationssicherheit zu übertragen und besteht aus zahlreichen Einzelstandards.

Dabei werden durch die Norm ISO/IEC 27001 [ISO13b] die Mindestanforderungen an Herstellung, Einführung, Betrieb, Überwachung, Wartung und Verbesserung eines dokumentierten Information Security Management Systems (ISMS) spezifiziert. Hierbei handelt es sich um ein Managementframework, mit dem ein Unternehmen seine IT-Risiken identifizieren und analysieren kann. Neben der risikogetriebenen Vorgehensweise steht die kontinuierliche Verbesserung des Information Security Managements im Mittelpunkt. Dabei wird sichergestellt, dass die Sicherheitsvorkehrungen flexibel justierbar sind, um Veränderungen der Sicherheitsbedrohungen, Schwachstellen und Unternehmensprozesse effizient abbilden zu können.

Das Information Security Incident Management (SIM) betreffend ist der Maßnahmenkatalog A.16 in ISO/IEC 27002 [ISO13c] von besonderer Bedeutung. Er umfasst sieben Einzelmaßnahmen (*Security Controls*) und stellt eine logische, effektive Herangehensweise an das Management von Sicherheitsvorfällen sicher. Darüber hinaus widmet sich künftig eine eigene, dreiteilige Norm speziell dem SIM: Auch wenn sich die Norm ISO/IEC 27035 [ISO13d] erst im Entwurfsstadium befindet, so ist dennoch davon auszugehen, dass an deren Gliederung keine größeren Änderungen mehr vorgenommen werden. Im ersten Teil werden die grundlegenden Prinzipien des Incident Managements und die prozessuale Vorgehensweise in fünf Phasen beschrieben. Teil 2 zeigt Richtlinien für die Planung und Vorbereitung einer effizienten und effektiven Vorfallbearbeitung auf. Anhand typischer Incidents wird im dritten Teil auf die tatsächliche Verwaltung von Sicherheitsvorfällen eingegangen. Dort wird auch die Etablierung eines Security Incident Response Teams angesprochen.

2.1.3. IT-Grundschutz (BSI)

Der vom Bundesamt für Sicherheit in der Informationstechnik (BSI) herausgegebene Katalog zum IT-Grundschutz enthält unter anderem auch Empfehlungen zum Information Security Incident Management. So wird in Baustein B1.8 (Behandlung von Sicherheitsvorfällen) zunächst ein Überblick über typische Vorfälle und mögliche Auslöser gegeben. Anschließend wird eine systematische Vorgehensweise zur Erstellung eines Security Incident Management-Konzepts sowie dessen Umsetzung und Einbettung in ein Unternehmen beschrieben. Alle in diesem Baustein enthaltenen Vorschläge stammen aus dem Maßnahmenkatalog M6 „Notfallvorsorge“.

Der IT-Grundschutzbaustein umfasst insgesamt 24 Maßnahmen mit konkreten Handlungsempfehlungen, praxisnahen Beispielen und Prüffragen. Er unterstützt IT-Sicherheitsbeauftragte dabei, einen ISO/IEC 27001 konformen Security Incident Response (SIR)-Prozess in ihrem Unternehmen zu etablieren. [Bun13] [Bun12a] [Bun12b]

2.1.4. NIST SP 800-61

Auch das amerikanische National Institute of Standards and Technology (NIST) gibt in der zweiten Auflage seiner Sonderveröffentlichung 800-61 [CMGS12] Empfehlungen zum Umgang mit Sicherheitsvorfällen. Dabei stehen drei Aspekte im Vordergrund. Zum einen werden methodische und organisatorische Aspekte hinsichtlich der Gründung und des Betriebs eines Computer Security Incident Response Teams (CSIRT) diskutiert. Weiterhin wird die strukturierte Incidentbearbeitung im Rahmen eines SIR-Prozesses modellhaft beschrieben, wobei konkrete Vorschläge für alle Prozessphasen dargelegt werden. Zuletzt werden Möglichkeiten der interorganisationalen Zusammenarbeit aufgezeigt. Dabei wird neben den Vorteilen einer organisationsübergreifenden Arbeitsweise auch auf die Herausforderungen hinsichtlich der Prozess- und Teamkoordination sowie des Informationsflusses eingegangen.

2.2. Definition „Information Security Incident“

Das im Mittelpunkt dieser Arbeit stehende Information Security Incident Management beinhaltet Prozesse, die den Umgang mit Informationssicherheitsvorfällen regeln. Dabei stellt ein *Information Security Incident* einen versuchten oder erfolgreichen unerlaubten Zugriff auf Informationen sowie deren Verwendung, Offenlegung, Änderung oder Vernichtung dar. In der Literatur verwendete Begriffsklärungen beziehen sich zumeist auf den internationalen Standard ISO/IEC 27000. Dort wird ein Informationssicherheitsvorfall wie folgt definiert:

> *„An **information security incident** is made up of one or more unwanted or unexpected information security events that could possibly compromise the security of information and weaken or impair business operations.“* — ISO/IEC 27000:2014 [ISO13a]

Als Information Security Incident wird folglich ein einzelnes Ereignis bzw. eine Reihe von unerwünschten oder unerwarteten Ereignissen bezeichnet, die möglicherweise zu einer Beeinträchtigung der Sicherheit von Informationen und Geschäftsprozessen führen könnten. Das LRZ erweitert diese Definition um eine potenzielle Gefährdung von IT-Diensten, IT-Systemen oder IT-Anwendungen und konkretisiert den Sicherheitsbegriff auf die Aspekte Vertraulichkeit, Integrität, Authentizität und Verfügbarkeit der genannten Ressourcen. [Met13]

Die in der Incidentdefinition erwähnten Information Security Events werden in der Norm ISO/IEC 27000 wie folgt beschrieben:

> *„An **information security event** is a system, service, or network state, condition, or occurrence that indicates that information security may have been breached or compromised or that a security policy may have been violated or a control may have failed.“* — ISO/IEC 27000:2014 [ISO13a]

Ein Informationssicherheitsereignis beschreibt folglich das Auftreten eines – die Informationssicherheit gefährdenden oder kompromittierenden – System-, Dienst- oder Netzwerkzustands, dem eine mögliche Verletzung einer Sicherheitsrichtlinie oder das Versagen von Sicherheitsmaßnahmen zu Grunde liegen könnte.

Bei der gemeinsamen Betrachtung der beiden Definitionen ist festzustellen, dass Sicherheitsereignisse als Indizien für das Vorliegen eines Security Incidents dienen können. Führt die Bewertung eines einzelnen Sicherheitsereignisses oder die Korrelation mehrerer Security Events zu dem Ergebnis, dass die Events tatsächliche oder potenziell negative Auswirkungen auf die Informationssicherheit haben, so liegt ein Information Security Incident vor. Da jeder Incident automatisch ein Event ist, aber nicht jedes Event ein Incident, lässt sich der Zusammenhang auch mengentheoretisch beschreiben, wobei die Incidents eine Teilmenge der Obermenge Events darstellen.

Auch das BSI definiert einen Sicherheitsvorfall als „ein unerwünschtes Ereignis (...), das Auswirkungen auf die Informationssicherheit hat und in der Folge große Schäden nach sich ziehen kann“ [Bun09], beispielsweise durch das Abgreifen, Manipulieren oder Zerstören von vertraulichen Informationen. Zu den typischen Sicherheitsvorfälle zählen folgende Szenarios:

- Fehlkonfigurationen, die zur Offenlegung vertraulicher Daten, zum Verlust der Integrität schutzbedürftiger Daten oder zum Verlust der Daten selbst führen
- Ausnutzung von Sicherheitslücken in Hard- oder Software
- Schadsoftware befallene Systeme
- Portscans durchführende Systeme
- Externe SSH-Scans
- Identitätsdiebstahl durch Ausspähung von Zugangsdaten
- Kompromittierung eines Webservers oder eines virtualisierten Servers
- Denial of Service Attacken, die zum temporären Ausfall eines IT-Services führen

Um derartige Security Incidents sowohl effektiv als auch effizient bearbeiten zu können, empfehlen die vorgestellten Standards eine strukturierte Vorgehensweise anhand eines dokumentierten Security Incident Response-Prozesses.

2.3. Der Information Security Incident Response Prozess

Der Information Security Incident Response (SIR) Prozess ist ein erprobtes und für Stresssituationen ausgelegtes Verfahren zur raschen und effizienten Bearbeitung von Sicherheitsvorfällen. Er gewährleistet ein konstantes Qualitätsniveau, sodass auf einen Angriff stets schnellst- und bestmöglich reagiert werden kann. Der SIR-Prozess betrachtet den gesamten Lebenszyklus eines sicherheitsrelevanten Vorfalls. Er regelt die Aktivitäten und Abläufe für die Incidentbearbeitung, definiert die am Prozess beteiligten Rollen und weist Verantwortlichkeiten zu.

Ein Sicherheitsvorfall durchläuft typischerweise die folgenden Prozessschritte: Identifikation eines Vorfalls, Meldung, Kategorisierung, Priorisierung, Untersuchung und Diagnose, Lösung und Wiederherstellung des Regelbetriebs sowie Abschluss des Vorfalls.

Für die Behandlung von Sicherheitsvorfällen ist das *Computer Security Incident Response Team* (CSIRT) zuständig. Hierbei handelt es sich um ein Team von erfahrenen und vertrauenswürdigen Spezialisten, das einen Vorfall über den gesamten Lebenszyklus des SIR-Prozesses hinweg kompetent begleitet (vgl. BSI-Grundschutz-Maßnahme M6.123). ISO/IEC 27002 fordert darüber hinaus in Maßnahme A.16.1.5, dass für jeden Vorfall mindestens ein Ansprechpartner festgelegt wird. Verschiedene Strukturmodelle für Incident Response Teams werden im Abschnitt 2.4 der NIST Sonderveröffentlichung 800-61 vorgestellt.

Da eine verlustfreie Kommunikation maßgeblich zum Prozesserfolg beiträgt, ist seitens CSIRT auf eine adäquate Wahl der Kommunikationskanäle zu achten (vgl. NIST SP 800-61, 3.2.7). Dies gilt sowohl für den teaminternen Austausch als auch für die Kommunikation mit dem Vorfallmelder und mit externen Parteien. Gemäß ISO/IEC 27002 (Maßnahme A.16.1.5) und BSI-Grundschutz-Maßnahme M6.65 sind die betroffenen Personen oder Organisationen über den Vorfall und eventuell relevante Details zu informieren. Bei allen Kommunikationsvorgängen ist stets zu dokumentieren, wer wann welche Informationen in welchem Detaillierungsgrad erhalten hat.

Der Aspekt einer *nachvollziehbaren Dokumentation* nimmt innerhalb des SIR-Prozesses einen besonders hohen Stellenwert ein. So wird nicht nur in ISO/IEC 27002 (Maßnahme A.16.1.5) gefordert, dass alle Aktivitäten während des gesamten Prozesses für eine spätere Nachvollziehbarkeit adäquat zu dokumentieren sind. Auch in der BSI-Grundschutz-Maßnahme M6.134 wird eine möglichst detaillierte und standardisierte Aufzeichnung aller Prozessschritte empfohlen. Eine derartige Dokumentation sollte idealerweise neben den durchgeführten Aktionen auch die Zeitpunkte und die Namen der handelnden Personen sowie die Protokolldateien der betroffenen Systeme enthalten.

Anforderungen an die *Planung und Konzeption eines SIR-Prozesses* sind in ISO/IEC 27002 (Maßnahme A.16.1.1) enthalten. Eine ähnliche Vorgehensweise empfiehlt auch das BSI im Rahmen der folgenden Grundschutz-Maßnahmen: Etablierung einer Vorgehensweise zur Behandlung von Sicherheitsvorfällen (M6.58) sowie die Festlegung von Verantwortlichkeiten (M6.59), Meldewegen (M6.60), Eskalationsstrategien (M6.61) und Prioritäten (M6.52).

Im Folgenden werden die Phasen des SIR-Prozessablaufs unter Einbeziehung der vorgestellten Standards in der Theorie beschrieben. Auf den konkreten SIR-Prozess des LRZ wird in Kapitel 3 eingegangen.

2.3.1. Incidentmeldung

Sicherheitsvorfälle und sicherheitsrelevante Schwachstellen sollen laut ISO/IEC 27002 (Maßnahmen A.16.1.2/3) mittels geeigneter Kommunikationskanäle schnellstmöglich gemeldet werden. Zu diesem Zweck sollen die Mitarbeiter eines Unternehmens eine „Incident Awareness“ [Ges11] entwickeln – also ein Bewusstsein für ihre Verantwortung, entdeckte Störungen oder das anormale Verhalten eines Dienstes unverzüglich anzuzeigen. Darüber hinaus sollen die Mitarbeiter über die Vorgehensweise der Vorfallmeldung informiert sein.

Bei der Wahl der geeigneten Meldewege ist gemäß BSI-Grundschutz-Maßnahme M6.130 zu berücksichtigen, dass Sicherheitsvorfälle auf unterschiedlichen Wegen (vgl. NIST SP 800-61, 3.2.3) bekannt werden können:

- Feststellung durch einen Benutzer
- Erkennung durch ein Monitoring System oder ein Intrusion Detection System
- Feststellung durch einen Mitarbeiter einer IT-Abteilung
- Feststellung durch einen externen Partner
- Information durch Strafverfolgungsbehörden oder Presse

Dementsprechend wird der Security Incident Response Prozess gemäß ITIL (Service Operation / Incident Management) durch die Meldung von einem Benutzer, einem Trigger-Ereignis oder durch andere Quellen initiiert. Die hierfür geeigneten Kommunikationskanäle sind vielfältig und umfassen Telefon, Fax und E-Mail ebenso wie persönliche Gespräche, SNMP-Traps und das Meldeformular eines Störungserfassungssystems.

2.3.2. Incidentpriorisierung

Gemäß BSI-Grundschutz-Maßnahme M6.131 ist jeder Sicherheitsvorfall zunächst zu qualifizieren und zu bewerten, wobei u.a. die von der Störung betroffenen Anwendungen, IT-Systeme und Dienste betrachtet werden.

Da Sicherheitsvorfälle nicht nach dem Prinzip „first-come, first-served“ abgearbeitet werden sollen, ist jeder Vorfall einem unternehmensspezifischen Priorisierungsverfahren (M6.62) zu unterziehen. Von der Prioritätensetzung hängt ab, in welcher Reihenfolge die Vorfälle bearbeitet werden sollen, sodass höher eingestufte Incidents vorrangig behandelt werden.

Bei der Festlegung der Incidentpriorität werden gemäß ITIL (Service Operation / Incident Management) die Auswirkung und Dringlichkeit des Sicherheitsvorfalls bewertet. Während die Dringlichkeit (*Urgency*) ausdrückt, wie schnell der Vorfall zu lösen ist, berücksichtigt die Auswirkung (*Impact*) den Umfang und das Schadenpotenzial des Incidents. Das NIST verfolgt hinsichtlich der Priorisierung einen anderen Ansatz auf Basis der Faktoren *Functional Impact*, *Information Impact* und *Recoverability Effort* (vgl. NIST SP 800-61, 3.2.6).

2.3.3. Incidentanalyse

Möglichst zeitnah nach dem Auftreten bzw. Bekanntwerden eines Vorfalls sind laut ISO/IEC 27002 (Maßnahme A.16.1.5) Beweise zu sammeln und ggf. einer forensischen Analyse zu unterziehen. Im Rahmen der Incidentanalyse soll u.a. der Angriffsweg ermittelt werden. Weitet sich der Incident aus, so ist eine Eskalation nötig.

Zur Identifizierung, Sammlung, Erfassung und Erhaltung von Beweismitteln (vgl. ISO/IEC 27002, A.16.1.7 sowie NIST SP 800-61, 3.3.2) sollen IT-forensische Prozesse angewendet werden. Es ist eine lückenlose Beweiskette anzufertigen. Dabei ist neben der Dokumentation und der Einhaltung der gesetzlichen Vorgaben insbesondere auf die Sicherheit der Beweismittel zu achten.

Die BSI-Grundschutz-Maßnahme M6.133 sieht vor, die betroffenen IT-Systeme vom Netz zu nehmen und alle relevanten Dateien – insbesondere Protokolldateien – zu sichern. Diese können aufschlussreiche Informationen über Art und Ursache des Sicherheitsproblems geben. Darüber hinaus sind u.a. Konfigurationsdateien auf Manipulationen zu untersuchen.

2.3.4. Incidentlösung

Sobald die Ursache eines Sicherheitsvorfalls identifiziert worden ist, sind die für dessen Behebung erforderlichen Maßnahmen zu definieren und auszuführen (vgl. NIST SP 800-61, 3.3.4). Zur effizienten Lösung des aktuellen Vorfalls sollen laut ISO/IEC 27002 (Maßnahme A.16.1.6) und BSI-Grundschutz-Maßnahme M6.64 die Erfahrungen und das durch vorherige Security Incidents erlangte Wissen genutzt werden. Daher ist zu prüfen, ob bereits eine ähnliche Störung und eine dafür geeignete Lösung erfasst wurde.

Nach der Wiederherstellung des Regelbetriebs sollen gemäß BSI-Grundschutz-Maßnahme M6.133 die betroffenen IT-Systeme und Netzübergänge mit geeigneten Überwachungswerkzeugen beobachtet werden, um die Wirksamkeit der Incidentlösung verifizieren und im Falle einer erneut erfolgreichen Attacke zeitnah reagieren zu können.

2.3.5. Post Incident Aktivitäten

Nach der erfolgreichen Lösung des Vorfalls ist der Incident laut ISO/IEC 27002 (Maßnahme A.16.1.5) formell abzuschließen und die Dokumentation fertigzustellen.

Darüber hinaus soll gemäß BSI-Grundschutz-Maßnahme M6.66 ein Post Incident Review durchgeführt werden, da sich hierbei oftmals Verbesserungsmöglichkeiten im Umgang mit Sicherheitsvorfällen ergeben und Rückschlüsse auf die Wirksamkeit der getroffenen Sicherheitsmaßnahmen ziehen lassen. Dabei können folgende Aspekte berücksichtigt werden:

- Reaktionszeit
- Wirksamkeit der Eskalationsstrategie
- Effektivität der Untersuchung
- Effizienz der internen Kommunikation
- Kommunikation mit dem Vorfallmelder
- Motivation des Täters
- Entwicklung einer Handlungsanweisung für den Umgang mit ähnlichen Vorfällen

Die NIST Sonderveröffentlichung 800-61 enthält in Abschnitt 3.4.1 eine ähnliche Auflistung von Fragestellungen, die im Rahmen eines Lessons Learned Meetings geklärt werden sollen. Beispielsweise werden auch Verbesserungsmöglichkeiten für den Informationsaustausch mit anderen Organisationen gesucht.

Zusätzlich den den Post Incident Reviews ist in ITIL (Service Operation / Incident Management) eine kontinuierliche Prozessoptimierung vorgesehen. Die korrespondierende BSI-Grundschutz-Maßnahme M6.68 empfiehlt ebenfalls, in regelmäßigen Abständen eine Effizienzprüfung des SIR-Prozesses durchzuführen. Dabei soll neben der Auswertung von Messgrößen wie der Reaktionszeit und der Bearbeitungszeit anhand simulierter Vorfälle getestet werden, ob der definierte Prozessablauf eingehalten wird. Gegebenenfalls sind entsprechende Prozessanpassungen vorzunehmen.

3. Information Security Incident Management am LRZ

Das LRZ ist das gemeinsame Rechenzentrum der Münchner Hochschulen und der Bayerischen Akademie der Wissenschaften und zählt zu den bedeutendsten technisch-wissenschaftlichen Rechenzentren in Deutschland. Seine Hauptaufgaben umfassen im Wesentlichen die Bereitstellung von IT-Diensten und der IT-Infrastruktur im Münchner Wissenschaftsnetz (MWN). Darüber hinaus betreibt das LRZ Hochleistungsrechensysteme für alle bayerischen Hochschulen und den Bundeshöchstleistungsrechner SuperMUC. Mehr als 100.000 Studenten, Professoren und Mitarbeiter der Münchner Hochschulen nutzen täglich die Dienste des LRZ. Die zunehmende Zahl von Angriffen auf IT-Systeme im Hochschulumfeld erfordert eine kontinuierliche Anpassung und Optimierung des Informationssicherheitsmanagements.

Insbesondere stellt die Behandlung von IT-Sicherheitsvorfällen einen wichtigen und zugleich sensiblen Arbeitsbereich dar. Am LRZ befasst sich ein zehnköpfiges CSIRT mit der Bearbeitung von Information Security Incidents. Da das LRZ kein ständiges CSIRT unterhält, wird dieses Team abteilungsübergreifend aus Mitarbeitern mit einschlägiger Sicherheitskompetenz ad hoc zusammengestellt. Durch die alleinige Bearbeitung eingetretener Sicherheitsvorfälle weist das LRZ-CSIRT aktuell eine rein reaktive Arbeitsweise auf. Dessen Zuständigkeitsbereich ist auf LRZ-interne und vom LRZ betreute Systeme begrenzt. [MHR11]

3.1. Aktuelle Situation am LRZ

Die Bearbeitung von Information Security Incidents erfolgt am LRZ nach einem im Jahr 2010 formal eingeführten Prozess. Dieser mit dem Standard ISO/IEC 27001 konforme SIR-Prozess ist in einer etwa 20-seitigen Prozessbeschreibung [Met13] dokumentiert und umfasst im Wesentlichen die aus Abbildung 3.1 ersichtlichen Phasen Incident Aufnahme, Klassifikation, Eskalation, Analyse und Diagnose, Lösung und Abschluss. Neben dem Prozessablauf werden in diesem Dokument auch prozessspezifische Rollen und am Prozess beteiligte Funktionen definiert.

Der SIR-Prozess wird durch die Meldung eines sicherheitsrelevanten Vorfalls instanziiert, wobei die Meldung telefonisch über eine Sammelrufnummer, per E-Mail an ein Gruppenpostfach oder persönlich an ein Mitglied des CSIRT erfolgen kann. Hierbei werden sowohl LRZ-interne Quellen (LRZ-Mitarbeiter, Monitoring- bzw. Sensorsysteme) als auch externe Quellen (Netzverantwortliche aus dem MWN, Meldungen durch externe CSIRTs) berücksichtigt. Telefonisch oder persönlich eingegangene Meldungen werden durch das aufnehmende Teammitglied per E-Mail an das gesamte CSIRT multipliziert.

Nach Erhalt der Meldung wird die personelle Zusammensetzung des Security Incident Teams festgelegt, das aus dem CSIRT-Hotliner, der die Meldung entgegen genommen hat, dem Security Incident Coordinator (SIC), dem Vorfallmelder und den zuständigen System- und

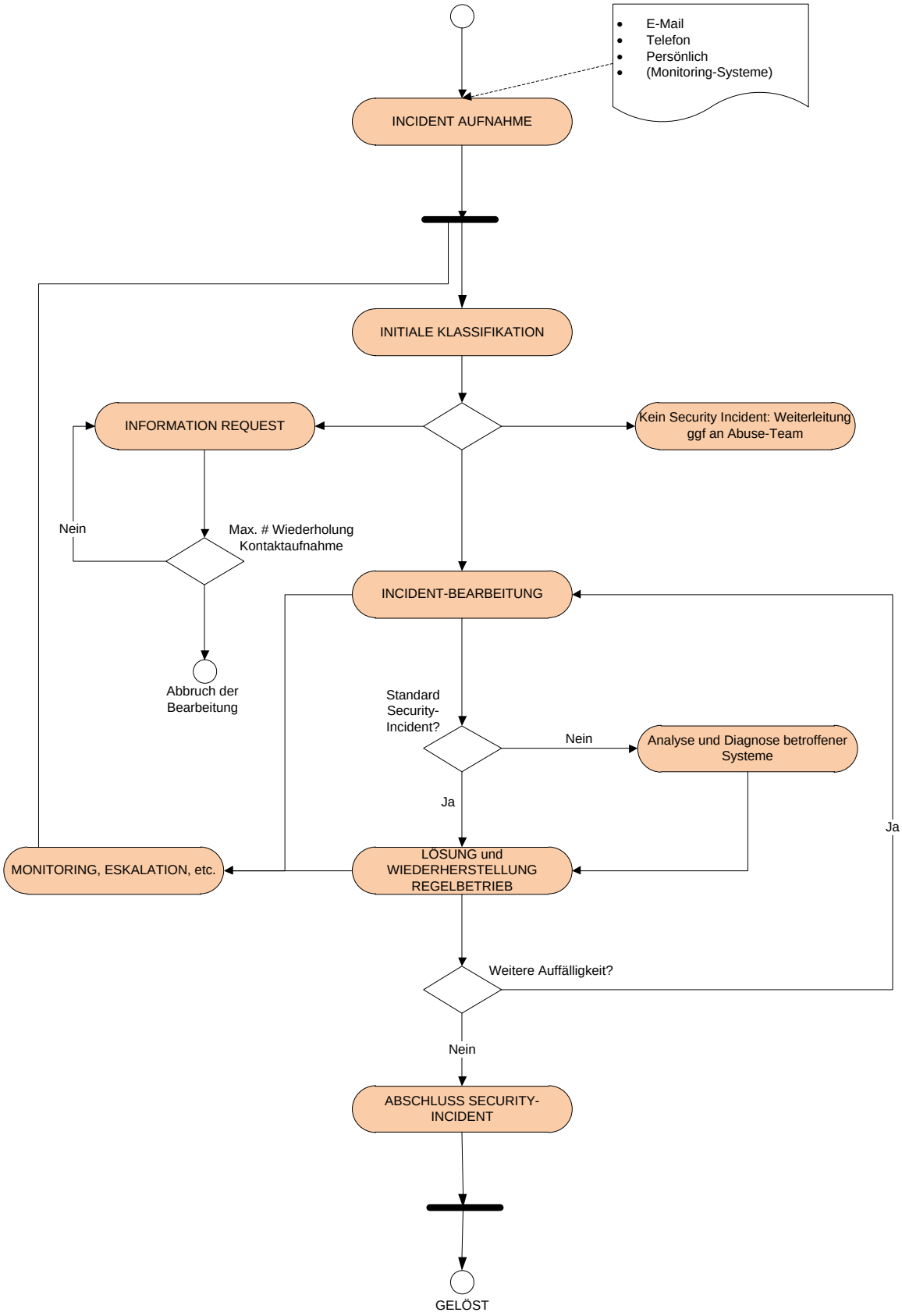

Abbildung 3.1.: Ablauf des Security Incident Response Prozesses am LRZ, vgl. [Met13]

Netzadministratoren besteht. Nach der Prüfung auf Vollständigkeit werden die gemeldeten Informationen – die Kontaktdaten des Vorfallmelders und eine Kurzbeschreibung des entdeckten Sicherheitsvorfalls – in einen Microsoft Word basierten Bearbeitungslaufzettel eingetragen und auf einem Netzlaufwerk gespeichert.

Im Rahmen der initialen Klassifikation wird geklärt, ob es sich überhaupt um einen Security Incident handelt und falls ja, welche Dienste und Informationen betroffen sind, welche Dienstabhängigkeiten bestehen, wie viele Systeme betroffen sind und ob es sich dabei um LRZ-interne, MWN zugehörige oder externe Zielsysteme handelt. Sofern es sich um ein externes Zielsystem handelt, ist der Vorfall für das LRZ-CSIRT nur dann relevant, wenn der Angriff von einem internen oder einem vom LRZ betreuten Quellsystem ausging. Über die Standortbestimmung des Angreifers hinaus werden die Art des Angriffs und etwaige Zusammenhänge zu früheren Sicherheitsvorfällen ermittelt. Auf Basis dieser Kriterien erfolgt anhand einer Gewichtungsformel die Festlegung der Priorität (niedrig, mittel, hoch und sehr hoch), wobei ein Vorfall der höchsten Stufe zum Major Incident ausgeweitet wird und eine umgehende Benachrichtigung der LRZ-Leitung zur Folge hat. Unabhängig davon sind Sicherheitsvorfälle grundsätzlich als dringend einzustufen. Anhand der Klassifizierungskriterien werden neben der Priorisierung auch die Auswirkung eines Security Incidents und die im Rahmen von Service Level Agreements (SLA) zugesicherten Reaktionszeiten festgelegt. Ergeben sich während der Incidentbearbeitung neue Erkenntnisse, so können diese die Klassifikation beeinflussen und zu einer Neubewertung des Vorfalls führen.

Sollte kein Information Security Incident, sondern ein *Information Request* vorliegen, so wird der Vorfall mit der Beantwortung der Frage bzw. Weiterleitung an die zuständige Bearbeitergruppe abgeschlossen.

Im Folgenden wird davon ausgegangen, dass ein individuell zu bearbeitender *Information Security Incident* vorliegt. Nach der initialen Klassifikation und Priorisierung wird der Bearbeitungslaufzettel mit allen zu diesem Zeitpunkt vorliegenden Informationen an das CSIRT verteilt. Die weitere Vorgehensweise wird durch den Security Incident Coordinator (SIC) festgelegt, der im Rahmen der Vorfallbearbeitung über ausgewählte Entscheidungsbefugnisse verfügt und einen regelmäßigen Informationsaustausch mit dem Vorfallmelder sowie eine lückenlose Dokumentation aller Vorgänge sicherstellt.

Die Analyse der betroffenen Systeme stellt den wichtigsten Teil des SIR-Prozesses dar und ist meist mehrfach zu durchlaufen. Sie soll Hinweise auf verdächtige Aktivitäten liefern, anhand derer die für den Einbruch oder die Bedrohung angewendete Methode oder Schwachstelle identifiziert werden kann. Ferner haben die Analysearbeiten die Ermittlung des Angreifers und des entstandenen oder potenziellen Schadens und letztendlich die schnellstmögliche Rückkehr zum Regelbetrieb zum Ziel. Die Ergebnisse der vom Security Incident Team durchgeführten Analyse werden vom CSIRT-Hotliner gesammelt und im Bearbeitungslaufzettel dokumentiert. Daraufhin folgt die Auswertung der Analyseresultate durch das CSIRT, wobei auch die Auswertungsergebnisse zu dokumentieren sind.

Der SIC entscheidet über die einzuleitenden Maßnahmen, wie beispielsweise das Trennen der Netzverbindung betroffener Systeme, die Beweissicherung durch Backups bzw. Snapshots, erste forensische Analysen durch Auswertung von Monitoringdaten, die Sperrung kompromittierter Nutzerkennungen, die Neuinstallation betroffener Systeme, und die fachliche oder hierarchische Eskalation. Nach der Identifikation des Einbruchswegs bzw. der Schwachstelle und der Sicherung von Beweisen ist eine geeignete Lösung zur Beseitigung der Ursache zu finden, wobei eine zeitnahe Wiederherstellung des Regelbetriebs anzustreben ist.

Wiederholt auftretende Sicherheitsvorfälle werden als Standard Security Incidents (SSI) bezeichnet. Da für diese die Vorgehensweise in der Analyse-, Diagnose- und Lösungsphase des SIR-Prozesses immer gleich sind, können bestimmte Vereinfachungen im Prozessablauf vorgenommen werden. So werden beispielsweise Brute-Force-Angriffe zum Erraten von SSH-Kennwörtern als Standard Security Incident (SSI) behandelt, wenn die Angriffe nicht von einem System im MWN ausgehen und nicht gezielt LRZ-Kennungen angreifen. Somit kann die Vielzahl an Angriffen mit geringen Erfolgsaussichten, die ein vernachlässigbares Risiko darstellen, effizient gehandhabt werden. Zugleich bleibt sichergestellt, dass die Vorfälle erfasst und in der für die Planung weiterer Sicherheitsmaßnahmen relevanten statistischen Auswertung Berücksichtigung finden.

Im Rahmen der Incident Nachsorge werden nach der Aufklärung von Vorfällen die betroffenen IT-Komponenten für einen Zeitraum von mehreren Tagen unter intensive Beobachtung gestellt. Durch diese Monitoringmaßnahme sollen eventuelle weitere Auffälligkeiten schnellstmöglich erkannt werden, um eine zeitnahe Reaktion sicherzustellen. Sofern im Beobachtungszeitraum keine weiteren Auffälligkeiten festzustellen sind, ist davon auszugehen, dass der Vorfall erfolgreich gelöst bzw. die Sicherheitslücke wirksam geschlossen wurde, sodass der Security Incident abgeschlossen werden kann. Hierbei sind alle im Rahmen dieses Vorfalls eingerichteten Filtermechanismen zu deaktivieren. Ferner ist die gesamte Dokumentation auf dem Bearbeitungslaufzettel auf Vollständigkeit, Korrektheit und Nachvollziehbarkeit zu prüfen.

In regelmäßigen Abständen werden Post Incident Reviews (PIR) zur retrospektiven Bewertung des Bearbeitungsablaufs und der erfolgten Maßnahmen durchgeführt. Dabei wird diskutiert, ob und wie der SIR-Prozess und seine Teilabläufe optimiert werden können.

3.2. Möglichkeiten der Prozessoptimierung

Der im vorigen Abschnitt 3.1 erläuterte SIR-Prozess stellt eine standardkonforme, praxiserprobte Vorgehensweise zur Behandlung von Security Incidents am LRZ dar. Die intensive Auseinandersetzung sowohl mit dem Prozess an sich als auch mit seinen einzelnen Phasen hat ergeben, dass eine *werkzeugunterstützte Prozessoptimierung durch Nutzung eines softwarebasierten Managementsystems* möglich ist und folgende Vorteile bietet:

- Reduzierung des Verwaltungsoverheads
- Verbesserung des Gesamtsicherheitsniveaus
- Erhöhung von Transparenz und Nachvollziehbarkeit
- Steigerung der Effizienz
- Verbesserung der Qualität

3.2.1. Reduzierung des Verwaltungsoverheads

Der Einsatz einer Softwarelösung kann die organisatorischen Tätigkeiten des CSIRT im Rahmen des SIR-Prozesses stark vereinfachen. Beispielsweise kann die manuelle Berechnung der initialen Incident Klassifikation wegfallen, wenn das System nach Auswahl weniger Parameter automatisch die Auswirkung des Incidents, seiner Priorität und der Reaktionszeit berechnet. Auch ist eine manuelle Erfassung von Security Incidents durch das CSIRT nur noch in Ausnahmefällen nötig, da alle LRZ-Mitarbeiter die Meldung eines Security Incidents selbstständig direkt im System vornehmen können. Die Beispiele zeigen, dass sich durch die Nutzung einer Managementsoftware der verwaltungstechnische Aufwand deutlich reduzieren lässt. Daher können sich die Mitglieder des CSIRT auf produktive Tätigkeiten im Rahmen der Analyse und Lösung des Incidents konzentrieren und somit zu einer schnelleren Wiederherstellung des Regelbetriebs beitragen.

3.2.2. Verbesserung des Gesamtsicherheitsniveaus

Bislang war der manuelle Aufwand zur Anfertigung statistischer Auswertungen für die Analyse zurückliegender Sicherheitsvorfälle unverhältnismäßig hoch. Hingegen kann durch eine softwarebasierte, strukturierte Datenhaltung jederzeit auf die sicherheitsrelevanten Key-Performance-Indikatoren (KPIs) zugegriffen werden. Beispielsweise ermöglichen die Kennzahlen eine faktenorientierte Anpassung des Risikomanagements. Ferner können dadurch regelmäßige Trendanalysen durchgeführt werden, wobei auf Basis automatisch generierter Reports ermittelt wird, welche Konsequenzen aus den vergangenen Security Incidents zu ziehen sind. Die Analyseergebnisse können in die Optimierung der präventiven Sicherheitsmechanismen und proaktiven -maßnahmen einfließen und somit wesentlich zur Verbesserung des Gesamtsicherheitsniveaus beitragen.

3.2.3. Erhöhung von Transparenz und Nachvollziehbarkeit

Im beschriebenen SIR-Prozess wird auf die Dokumentation der Fakten und die Nachvollziehbarkeit der Vorgehensweise besonderer Wert gelegt. Dennoch sind Umfang und Genauigkeit der Aufzeichnungen wegen der manuellen Ausführung beschränkt. Der Einsatz einer Softwarelösung bietet hingegen eine automatisierte Dokumentation der eingegeben Daten sowie sämtlicher Vorgänge. Sowohl während der Incidentbearbeitung als auch im Nachhinein bei einem PIR kann die gesamte Vorgehensweise exakt nachvollzogen werden. Insbesondere kann bei strittigen Fragen eindeutig aufgeklärt werden, wer wann welche Information eingegeben oder geändert hat, beispielsweise welcher Mitarbeiter zu welchem Zeitpunkt die Reduzierung einer Incidentpriorität veranlasst hat.

Das System erhöht zudem die Transparenz der Kommunikation während der Incidentbearbeitung, indem die über die Software verschickten E-Mails automatisch dokumentiert werden. Zudem können eingehende E-Mails sowie Telefonate und persönliche Absprachen im System erfasst werden. Da Security Incidents in der Regel durch mehrere Personen gemeinsam abgewickelt werden, ist es von Vorteil, wenn jedes einzelne Teammitglied die vorfallbezogenen Kommunikationsaktivitäten des gesamten CSIRT einsehen kann. Dadurch kann ein einheitliches Informationsniveau unter allen Teammitgliedern hergestellt werden, sodass überflüssige Nachfragen und Fehlkommunikation vermieden werden können.

3.2.4. Steigerung der Effizienz

Durch die bereits beschriebene Reduzierung des Verwaltungsoverheads sowie die softwarebasierte Teilautomatisierung des SIR-Prozesses können Sicherheitsvorfälle effizienter bearbeitet werden. Dies bedeutet eine Reduzierung der durchschnittlichen Bearbeitungsdauer und daraus resultierend eine niedrigere Ausfalldauer bzw. kürzere Beeinträchtigung der betroffenen Dienste. Darüber hinaus steht den CSIRT-Mitgliedern durch eine effizientere Vorfallbearbeitung mehr Zeit für andere Aufgaben – beispielsweise im präventiven Bereich – zur Verfügung.

3.2.5. Verbesserung der Qualität

Neue, noch unerfahrene CSIRT-Mitglieder, die sich noch nicht intensiv in die Thematik eingearbeitet haben, müssen im Notfall ihre Aktivitäten an der ausführlichen SIR-Prozessbeschreibung und dem Bearbeitungslaufzettel ausrichten. Gerade in Stresssituationen kann die Unsicherheit des Mitarbeiters zu einer intuitiven, nicht oder nur teilweise prozesskonformen Vorgehensweise führen. Die Nutzung eines softwarebasierten Managementwerkzeugs kann erheblich zur Verbesserung der SIR-Prozessqualität beitragen, indem es eine konstant hohe Prozesskonformität bei der Bearbeitung von Sicherheitsvorfällen begünstigt und Qualitätskennzahlen wie z.B. die Reaktionszeit und die durchschnittliche Bearbeitungsdauer automatisch gemessen werden. Dabei bietet das Managementsystem den CSIRT-Mitgliedern eine Prozessunterstützung, indem es sie schrittweise durch alle Phasen einer Security Incident Behandlung begleitet. Diese Orientierungshilfe ermöglicht trotz einer lediglich kurzen Einarbeitung eine Verbesserung der Qualität in Form einer vergleichsweise hohen Prozesskonformität bei unerfahrenen CSIRT-Mitgliedern und in Vertretungsfällen.

3.2.6. Fazit

Die dargestellten Aspekte stellen im Ergebnis wesentliche Vorteile der Nutzung einer Softwarelösung gegenüber der Arbeit mit Microsoft Word basierten Bearbeitungslaufzetteln dar. Damit das LRZ künftig von den Vorteilen profitieren kann, wird eine werkzeuggestützte Arbeitsweise und eine dementsprechende Anpassung des SIR-Prozesses empfohlen. Zunächst ist jedoch ein Anforderungsprofil zu erstellen, das die Grundlage für die Konzeption einer Individualsoftware bildet.

4. Anforderungsanalyse

Zur Ermittlung der Anforderungen an das softwarebasierte Information Security Incident Managementwerkzeug wird der Security Incident Response (SIR) Prozess [Met13] des LRZ analysiert. Zudem werden die in der Aufgabenstellung (Kapitel 1.3) genannten Forderungen und die Möglichkeiten der Prozessoptimierung (Kapitel 3.2) berücksichtigt. Zur Vervollständigung des Anforderungsprofils wurden im Rahmen eines Brainstormings weitere Aspekte zusammengetragen. Die insgesamt 40 Anforderungen wurden fortlaufend nummeriert und einer dreistufigen Gewichtung hinsichtlich ihrer Umsetzungsrelevanz unterzogen:

- **Muss-Anforderungen** beschreiben die Mindestanforderungen an die Software und umfassen deren wesentliche Merkmale sowie Kernfunktionen, die zwingend zu erfüllen sind. Wegen ihrer hohen Relevanz werden die Muss-Anforderungen dreifach gewichtet.
- **Soll-Anforderungen** beschreiben Merkmale, deren Vorhandensein vorteilhaft, aber nicht zwingend notwendig ist. Sie erweitern den Basisumfang der Software um Funktionalitäten, die einen relevanten Mehrwert bieten, und werden daher zweifach gewichtet.
- **Kann-Anforderungen** beschreiben optionale Anforderungen an die Software, deren Erfüllung zwar wünschenswert, aber von untergeordneter Bedeutung ist. Aufgrund der niedrigen Relevanz werden die Kann-Anforderungen einfach gewichtet.

Dieses Kapitel spezifiziert zunächst thematisch gruppiert die funktionalen Anforderungen an die Software, beschreibt im zweiten Teil die nicht-funktionalen Anforderungen und fasst schließlich alle Anforderungen in Form eines tabellarischen Überblicks zusammen. Die Begriffe „System“, „Software“, „Applikation“ und „Anwendung“ werden im Folgenden synonym verwendet. Die Mitglieder des LRZ-CSIRT werden als „Anwender“ bezeichnet.

4.1. Funktionale Anforderungen

Die nachfolgend formulierten funktionalen Anforderungen spezifizieren die Funktionalitäten und Dienste, die das System bereitstellen soll.

4.1.1. Incidenterfassung

[R01] Incidenterfassung mittels Webformular

Die Erfassung von Information Security Incidents soll an zentraler Stelle mittels Webformular erfolgen, wobei die Eingaben automatisiert weiterzuverarbeiten sind. Auf diese Meldemöglichkeit sollen alle LRZ-Mitarbeiter im Intranet zugreifen können. Die herkömmlichen Meldewege (per E-Mail, telefonisch, persönlich) sollen durch eine manuelle Erfassung seitens CSIRT weiterhin zur Verfügung stehen. Der neue Meldeweg ist jedoch zu bevorzugen, da hier die Meldung und Erfassung von Information Security Incidents in einem einzigen Arbeitsschritt vereint werden. – *Gewichtung:* 3 (Muss-Anforderung).

[R02] Automatisierte Incidenterfassung durch SIEM-Lösungen

Die Webanwendung soll eine Schnittstelle zur Anbindung von SIEM-Lösungen zur Verfügung stellen. Dadurch soll am LRZ eingesetzte Drittanbietersoftware in die Lage versetzt werden, automatisiert Incidents innerhalb der Webanwendung anzulegen, wenn beispielsweise ein schwerwiegendes Ereignis im LRZ-internen Netz detektiert wird. Dabei entscheidet die Software durch Korrelation vieler Einzelereignisse, wann ein Security Incident ausgelöst wird. Die Meldeschnittstelle ermöglicht somit eine effiziente und zuverlässige Einbindung der Resultate von Monitoringsystemen in den Prozessablauf. – *Gewichtung:* 1 (Kann-Anforderung).

4.1.2. Incidentbearbeitung

[R03] Initiale Incident Klassifikation

Das System hat das CSIRT über jeden neuen Incident zu informieren. Es muss dem CSIRT-Hotliner die Möglichkeit geben, eingegangene Meldungen dahingehend zu klassifizieren, ob ein Information Security Incident oder ein Information Request vorliegt. Für die weitere Bearbeitung hat die Anwendung in Abhängigkeit von der Klassifikation unterschiedliche Funktionen bereitzustellen, welche in den nachfolgenden Anforderungen spezifiziert sind. – *Gewichtung:* 3 (Muss-Anforderung).

[R04] Umgang mit Incident Duplikaten

Tritt ein Security Incident ein, der eine größere Nutzerzahl betrifft, so ist es möglich, dass der Security Incident von verschiedenen Nutzern parallel gemeldet wird. Daher soll das System mit Incident Duplikaten sinnvoll umgehen können und mehrfach gemeldete Vorfälle zu einem einzigen zusammenfassen. – *Gewichtung:* 2 (Soll-Anforderung).

[R05] Bearbeitung von Information Requests

Falls die initiale Klassifikation ergibt, dass es sich um einen Information Request handelt, so hat das System die Beantwortung der Frage an den Vorfallmelder bzw. die Weiterleitung der Anfrage an die zuständige Bearbeitergruppe zu ermöglichen und die Kommunikation in geeigneter Weise zu dokumentieren. Wegen der Abwicklung von Information Requests innerhalb des Systems können diese zahlenmäßig bei der Auswertung berücksichtigt werden. Liegt eine verhältnismäßig hohe Anzahl an Anfragen vor, können daraus informationsstrategische Optimierungsmöglichkeiten abgeleitet werden. – *Gewichtung:* 1 (Kann-Anforderung).

[R06] Initiale Bearbeitung von Security Incidents

Sofern während der initialen Klassifikation das Vorliegen eines Information Security Incidents festgestellt wird, sind durch den CSIRT-Hotliner und SIC in der Applikation weitere Schritte durchzuführen und zu dokumentieren. Diese betreffen die Zusammensetzung des Teams, die Ermittlung der relevanten Administratoren, die Prüfung der Kontaktinformationen des Vorfallmelders, die Klärung des Vorfallzeitpunkts sowie die Entscheidung, ob ein Major Incident zu erwarten ist. – *Gewichtung:* 3 (Muss-Anforderung).

[R07] Unterstützung von Standard Security Incidents

Die Software soll zwischen regulären Security Incidents und Standard Security Incidents (SSIs) unterscheiden können, wobei im Falle eines SSI in Abhängigkeit vom Typ des Vorfalls gewisse Felder mit Textbausteinen vorbelegt werden sollen. Diese Funktion soll den Umfang der von Hand einzugebenden Informationen reduzieren und somit eine Effizienzsteigerung bewirken. – *Gewichtung:* 3 (Muss-Anforderung).

[R08] Priorisierung mittels Auswirkungsmatrix

Insbesondere soll das System den Anwender hinsichtlich der Festlegung von Auswirkung, Priorität und Reaktionszeit durch Implementierung der in der SIR-Prozessbeschreibung definierten Auswirkungsmatrix unterstützen. Hierbei sollen vom Anwender Angaben zum Standort des Zielsystems, zu den betroffenen Diensten und Informationen, zur Anzahl der betroffenen Systeme und zum Standort des Quellsystems gemacht werden. Auf Grundlage dieser Informationen soll das System anhand einer Gewichtungsformel automatisch die gemäß SIR-Prozess vorgesehene Auswirkung, Priorität und Reaktionszeit berechnen. Die Automatisierung dieses bislang manuellen Arbeitsschritts stellt eine Arbeitserleichterung für das CSIRT dar und schließt eine Fehlerquelle aus. – *Gewichtung:* 2 (Soll-Anforderung).

[R09] Information Request an den Vorfallmelder

Sollte die Incidentmeldung unvollständig sein, so bietet die Software dem Anwender die Möglichkeit, die fehlenden Informationen per E-Mail beim Vorfallmelder anzufordern. Auch dieser Arbeitsschritt soll zum Zweck einer lückenlosen Dokumentation innerhalb des Systems durchgeführt werden können. – *Gewichtung:* 1 (Kann-Anforderung).

[R10] Weitere Incidentbearbeitung

Das System soll dem Anwender die Möglichkeiten der Festlegung von Erstmaßnahmen bzw. der weiteren Vorgehensweise, der Eingabe der Analyseergebnisse, der Dokumentation der Rückkehr zum Regelbetrieb und der Incidentlösung, der Festlegung der Monitoringmaßnahmen und der Dokumentation weiterer Auffälligkeiten bieten. Dabei kann der Anwender jederzeit Ergänzungen und Korrekturen an den bereits eingegebenen Daten vornehmen. – *Gewichtung:* 3 (Muss-Anforderung).

[R11] Erweitertes Monitoring

Um eine hohe Qualität bei der Security Incident Nachsorge zu gewährleisten, können in Ergänzung zu `R10` die konkreten Monitoringaufgaben, die verantwortlichen Systemadministratoren sowie Monitoringdauer und -intervall im System festgelegt werden. Das System erinnert die verantwortlichen Personen per E-Mail im vorgegebenen Rhythmus an die Durchführung der jeweiligen Monitoringaktivitäten. Die Erledigung kann durch Klick auf einen Link in der E-Mail bestätigt werden. Etwaige Auffälligkeiten können direkt an das CSIRT gemeldet werden. – *Gewichtung:* 1 (Kann-Anforderung).

[R12] Wiedereröffnung von abgeschlossenen Incidents

Falls im Zusammenhang eines bereits abgeschlossenen Security Incidents weitere Auffälligkeiten entdeckt werden, so soll der vorhandene Incident wieder eröffnet werden können, um die weitere Bearbeitung in diesem Kontext durchführen zu können. – *Gewichtung:* 2 (Soll-Anforderung).

[R13] Erfassung der Post Incident Review Ergebnisse

Die Ergebnisse der zum Zwecke der Qualitätssicherung und kontinuierlichen Verbesserung durchgeführten Post Incident Reviews (PIR) sollen im System unter Verknüpfung mit dem zugehörigen Security Incident erfasst werden können. – *Gewichtung:* 3 (Muss-Anforderung).

[R14] Archivierung abgeschlossener Security Incidents

Bereits abgeschlossene Security Incidents sollen weiterhin im System verbleiben und bei der Bearbeitung ähnlicher Vorfälle als Wissensdatenbank dienen, um diese effizienter bearbeiten zu können. Ferner sollen die Daten aus statistischen Gründen und zu Auswertungszwecken archiviert werden. Archivierte Incidents sollen mit der später beschrieben Such- und Filterfunktion auffindbar sein. – *Gewichtung:* 3 (Muss-Anforderung).

4.1.3. Systemfunktionen

[R15] Prozess-Unterstützung für Anwender

Um dem CSIRT die prozesskonforme Bearbeitung von Information Security Incidents zu erleichtern, bildet die Anwendung den Prozess in digitaler Form ab und geleitet den Anwender durch den gesamten Lebenszyklus eines Incidents hindurch. Dabei bietet es genau diejenigen Funktionen an, die im jeweiligen Prozessstadium sinnvoll sind. Beispielsweise soll das System die Festlegung von Monitoringmaßnahmen erst dann ermöglichen, wenn die Rückkehr zum Regelbetrieb erfolgt ist. Die Software sorgt durch ihre Prozesskonformität für einen hohen Qualitätsstandard. Zugleich fühlen sich die Anwender organisatorisch sicher, da sie nur in Ausnahmefällen in der umfangreichen Prozessbeschreibung nachlesen müssen, und können sich auf fachlicher Ebene vollkommen auf den Vorfall konzentrieren. – *Gewichtung:* 2 (Soll-Anforderung).

[R16] Dashboard

Die Startseite der Anwendung soll in Form eines Dashboards aufgebaut sein, sodass sich die Anwender unmittelbar nach dem Login einen Überblick über die aktuelle Sachlage verschaffen können. Die Widgets enthalten die Darstellung von einfachen Auswertungen als Diagramme oder Kennzahlen, wie beispielsweise die Anzahl der neuen und in Bearbeitung befindlichen Security Incidents, die Anzahl der im aktuellen Monat und im laufenden Quartal abgeschlossenen Fälle. Zudem wird eine Auflistung der neuen und in in Bearbeitung befindlichen Security Incidents unter Angabe von Titel und Statusinformationen erwartet. Das Dashboard kann als Ausgangspunkt für die Arbeit mit der Anwendung dienen. – *Gewichtung:* 1 (Kann-Anforderung).

[R17] Incidentstatusseite

Sämtliche zu einem Security Incident gespeicherten Informationen sollen auf einer Statusseite dargestellt werden. Auf diese Weise kann sich der Anwender einen detaillierten Überblick über den aktuellen Zustand eines Security Incidents verschaffen. – *Gewichtung:* 3 (Muss-Anforderung).

[R18] Dateiverwaltung

Der Vorfallmelder soll die Möglichkeit haben, im Rahmen der Incidentmeldung Dateien zu übermitteln, die für das CSIRT im Rahmen der Incidentbearbeitung hilfreich sein können. Darüber hinaus sollen auch die CSIRT-Mitglieder incidentbezogen Dateien in der Anwendung speichern können. Zu diesem Zweck soll die Anwendung eine Dateiverwaltung bereitstellen. Durch die Integration einer Dateiverwaltung in die Applikation können beispielsweise Analyseergebnisse und Logfiles im Rahmen der Beweissicherung an zentraler Stelle aus der Webanwendung heraus geöffnet werden und sind daher auch zu einem späteren Zeitpunkt schnell auffindbar. – *Gewichtung:* 2 (Soll-Anforderung).

[R19] Automatisierte Handlungsempfehlungen

Sobald während der Incidentbearbeitung genug Daten vorliegen, soll das System – basierend auf bereits abgeschlossenen Incidents, welche eine signifikante Ähnlichkeit zum aktuellen Vorfall aufweisen – automatisiert Informationen anzeigen, die in der Vergangenheit nützlich waren. Anhand der ermittelten Parallelfälle soll das System dem Anwender die Lessons Learned aufzeigen und somit Handlungsempfehlungen für die Festlegung der weiteren Vorgehensweise bieten. Dadurch, dass der Anwender keine manuelle Suche ausführen muss, wird durch die Empfehlungsautomatik vorhandenes Wissen optimal in den Prozessablauf eingebunden. – *Gewichtung:* 1 (Kann-Anforderung).

[R20] Dokumentation aller Aktionen

Alle im System durchgeführten Aktionen sind zu dokumentieren, sodass zu jedem Zeitpunkt nachvollziehbar ist, welcher Anwender wann welche Aktion durchgeführt hat. Die Dokumentation sorgt für Transparenz und ermöglicht beispielsweise im Rahmen eines Post Incident Reviews (PIR) eine exakte Analyse der Vorgehensweise, die zur Prozessoptimierung beitragen kann. Zudem bietet die Dokumentation eine Art Beweissicherheit für das CSIRT, wenn es im Zweifelsfall die prozesskonforme Vorgehensweise nachzuweisen hat. Die Dokumentation muss daher manipulationssicher sein. – *Gewichtung:* 2 (Soll-Anforderung).

4.1.4. Kommunikation

[R21] E-Mail-Benachrichtigungen

Eine zielorientierte Kommunikation der Prozessbeteiligten trägt maßgeblich zur effektiven und effizienten Incidentbearbeitung bei. Das System soll die Anwender bei der Kommunikation unterstützen. Dies betrifft insbesondere die in der Prozessbeschreibung verankerten, an Bedingungen geknüpften Benachrichtigungen des Vorfallmelders, der Mitglieder des CSIRT, der Systemadministratoren, der LRZ-Leitung, der Abteilungs- und Gruppenleitung, des Major Incident Coordinator (MIC) und ggf. externer CSIRTs (DEISA, DFN, EGI) und

der Strafverfolgungsbehörden. Darunter fällt auch der Mailversand des aktuellen Stands an das CSIRT. Da nur die unbedingt notwendigen Informationen herausgegeben werden sollen, muss der Umfang der per E-Mail zu versendenden Daten editierbar sein. – *Gewichtung:* 3 (Muss-Anforderung).

[R22] Dokumentation der Kommunikation

Die systemgestützte Dokumentation der Kommunikation soll neben den über die Anwendung versendeten E-Mails auch die Erfassung von eingehenden E-Mails sowie geführten Telefonaten und persönlichen Absprachen umfassen. Dadurch sollen sämtliche, den jeweiligen Security Incident betreffenden Kommunikationsvorgänge und die damit verbundenen Informationen auf transparente Art und Weise allen Mitgliedern des CSIRT zur Verfügung stehen. Hiermit können interne Rückfragen auf ein Minimum reduziert werden. Zudem ist die Kommunikation auch zu einem späteren Zeitpunkt noch nachvollziehbar, sodass diese beispielsweise im Rahmen eines Post Incident Reviews (PIR) analysiert und daraus Optimierungsmöglichkeiten abgeleitet werden können. – *Gewichtung:* 2 (Soll-Anforderung).

[R23] Meldungen an die Allianz für Cyber-Sicherheit

Die Software stellt eine Exportvorlage zur Verfügung, welche die Meldung von Security Incidents an die Allianz für Cyber-Sicherheit in einem geeigneten Format ermöglicht. Dadurch ist keine manuelle Datenübernahme mehr nötig. Dies soll zur Vereinfachung des Meldevorgangs beitragen, die Effizienz steigern und Fehlerquellen reduzieren. *Gewichtung:* 3 (Muss-Anforderung).

4.1.5. Suchen, Filtern, Exportieren

[R24] Such- und Filterfunktion

Das System stellt eine parameterbasierte Such- und Filterfunktion bereit. Anhand der gewählten Parameter können die zutreffenden Security Incidents aus dem Datenbestand selektiert werden. – *Gewichtung:* 3 (Muss-Anforderung).

[R25] Tagging von Security Incidents

Eine Tagging-Funktion soll das Anreichern von Security Incidents mit Metainformationen ermöglichen. Durch die Verwendung gleicher Tags können ähnliche Vorfälle flexibel gruppiert werden. Dies erleichtert das spätere Auffinden aller Sicherheitsvorfälle zum relevanten Tag und ist zudem bei der Durchführung von Ähnlichkeitsvergleichen hilfreich. – *Gewichtung:* 1 (Kann-Anforderung).

[R26] Einzelexport eines Security Incidents

Die Einzelexport-Funktion soll es dem Anwender ermöglichen, den aktuellen Datenbestand eines Security Incidents in mehreren Datenformaten zu speichern. Dies ist im Rahmen der Incidentbearbeitung vor allem für die Zusammenarbeit mit externen Kollegen von Bedeutung, die keinen Zugriff auf das System haben. – *Gewichtung:* 3 (Muss-Anforderung).

[R27] Massenexport von Security Incidents

Mehrere mittels Such- und Filterfunktion selektierte Security Incidents sollen in einem einzigen Vorgang exportiert werden können. Die Exportfunktion soll mehrere Datenformate unterstützen. Dies ermöglicht eine flexible, systemunabhängige Weiterverarbeitung des Datenmaterials. – *Gewichtung:* 1 (Kann-Anforderung).

4.1.6. Statistische Auswertungen

[R28] Einfache statistische Auswertungen

Basierend auf dem vom System verwalteten Datenbestand können einfache statistische Auswertungen durchgeführt werden. Das System berechnet hierbei ausgewählte Kennzahlen für einen durch den Anwender gewählten Kunden im definierten Zeitraum. Diese tragen als Datengrundlage für interne und kundenseitige Trendanalysen zur Optimierung präventiver und proaktiver Sicherheitsmaßnahmen bei. – *Gewichtung:* 3 (Muss-Anforderung).

[R29] Export von Auswertungen

Die im System erstellten Auswertungen können exportiert werden. Dies ermöglicht eine flexible, systemunabhängige Weiterverarbeitung des Datenmaterials, beispielsweise als Jahresreport für einen Kunden. – *Gewichtung:* 2 (Soll-Anforderung).

[R30] Automatisierter E-Mail-Versand von Auswertungen

Das System soll automatisch in regelmäßigen Abständen Auswertungen an die Mitglieder des CSIRT per E-Mail senden. Durch die automatische Zustellung eines wöchentlichen oder monatlichen Reports sind die CSIRT-Mitglieder stets über die sicherheitsrelevanten Fakten informiert. – *Gewichtung:* 1 (Kann-Anforderung).

[R31] LRZ-Branding der PDF-Dokumente

Neben den statistischen Auswertungen können im System auch andere Informationen als PDF-Reports exportiert werden. Auf allen PDF-Dokumenten soll das LRZ-Logo als Corporate Design Element abgebildet werden und so die Reports optisch aufwerten. – *Gewichtung:* 1 (Kann-Anforderung).

4.1.7. Authentifizierung und Rechtevergabe

[R32] Benutzerauthentifizierung

Um das System vor unbefugten Zugriffen zu schützen, muss es über einen geeigneten Mechanismus zur Benutzerauthentifizierung verfügen. Eine gemeinsame Zugangskennung für das gesamte CSIRT würde hingehen einer Gruppenauthentifizierung entsprechen und somit im Widerspruch zur Anforderung stehen, dass alle Aktionen zu dokumentieren sind, womit eine benutzerbezogene Nachvollziehbarkeit der Aktivitäten innerhalb der Webanwendung gemeint ist. Darüber hinaus soll das System gewisse Anwenderdaten als Benutzerprofil verwalten können, sodass keine wiederholte Eingabe von Kontaktinformationen notwendig ist. – *Gewichtung:* 3 (Muss-Anforderung).

[R33] Rechtevergabe anhand statischer Benutzergruppen

Zusätzlich zur Benutzerauthentifizerung sollen auf Gruppenebene Berechtigungen vergeben werden können, die die Nutzung verschiedener Funktionen der Softwareanwendung aufgrund Zugehörigkeit zu einer statischen Benutzergruppe ermöglichen. Während dem CSIRT der Zugriff auf alle Funktionen gestattet ist, sollen LRZ-Mitarbeiter lediglich die Meldefunktion verwenden können. – *Gewichtung:* 2 (Soll-Anforderung).

[R34] Rechtevergabe anhand dynamischer Benutzergruppen

Das System soll das Security Incident Team zur Laufzeit als dynamische Benutzergruppe abbilden und auf dieser Ebene Berechtigungen vergeben können. Beispielsweise sollen alle Prozessbeteiligten eines konkreten Security Incidents unabhängig von ihrer statischen Gruppenmitgliedschaft Lese- und Schreibrechte für diesen einen Security Incident erhalten. Dadurch können nicht nur die CSIRT-Mitglieder, sondern auch der Vorfallmelder sowie beteiligte Systemadministratoren und Abteilungsleiter den aktuellen Stand einsehen und Analyseergebnisse eintragen. Somit können vermeidbare Nachfragen und das manuelle Übertragen von Informationen minimiert werden. – *Gewichtung:* 1 (Kann-Anforderung).

4.2. Nichtfunktionale Anforderungen

Die nachfolgend formulierten nichtfunktionalen Anforderungen beschreiben Einschränkungen und Qualitätsmerkmale an den Betrieb und die Entwicklung des Systems.

4.2.1. [R35] Webbasierte Technik

Security Incidents werden selten nur durch eine einzelne Person, sondern in der Regel durch ein Security Incident Team bearbeitet, das aus dem CSIRT-Hotliner, weiteren Mitgliedern des LRZ-CSIRT, den Netz- und Systemadministratoren sowie dem Vorfallmelder und involvierten IT-Forensikern dynamisch zusammengestellt wird. Das Managementsystem soll die Zusammenarbeit der Teammitglieder unterstützen und muss daher netzwerkfähig sein. Das Security Incident Management System soll als Webanwendung realisiert werden, da auf diese Weise keine Client-Installation notwendig ist und – gerade in heterogenen Netzwerken – plattformunabhängig damit gearbeitet werden kann. Die Verwendung der Webapplikation ist zudem – Internet- und VPN-Zugang vorausgesetzt – ortsunabhängig möglich. Die Realisierung des Systems als Webanwendung entspricht der heute üblichen Praxis und bietet hinsichtlich der genannten Aspekte eine größtmögliche Flexibilität. – *Gewichtung:* 3 (Muss-Anforderung).

4.2.2. [R36] Zentrale Datenhaltung in einer Datenbank

Zur Realisierung der meisten funktionalen Anforderungen ist eine zentrale Datenhaltung in einer Datenbank notwendig. Eine datenbankbasierte Arbeitsweise ist insbesondere für eine effiziente Implementierung der Such- und Filterfunktion unerlässlich. – *Gewichtung:* 3 (Muss-Anforderung).

4.2.3. [R37] Lauffähigkeit auf einem Standard-Webserver

Als Rahmenbedingung wird die Kompatibilität der Anwendung mit den Linux basierten Standard-Webservern des LRZ gefordert, da die Anwendung im eigenen Haus gehostet werden soll. Durch den hausinternen Betrieb des Systems ergeben sich Vorteile in puncto Sicherheit, Performance und Kosten. – *Gewichtung:* 3 (Muss-Anforderung).

4.2.4. [R38] Gebrauchstauglichkeit und Benutzerfreundlichkeit

Das Managementsystem soll grundsätzlich die Anwender dabei unterstützen, Security Incidents effektiv, effizient und zufriedenstellend zu bearbeiten. Über die Gebrauchstauglichkeit hinaus soll die Software eine hohe Usability aufweisen, also eine leicht verständliche und intuitiv bedienbare Benutzeroberfläche bieten, sodass die Bedienung des Systems ohne größeren Einarbeitungsaufwand erlernbar ist und in Form einer positiven User Experience von den Anwendern als angenehm empfunden wird. – *Gewichtung:* 2 (Soll-Anforderung).

4.2.5. [R39] IT-Sicherheit

Da das Managementsystem die Abwicklung von Security Incidents zum Ziel hat, soll insbesondere die Software selbst ein hohes Sicherheitslevel erreichen. Einerseits wird die Gewährleistung einer hohen Verfügbarkeit verlangt, sodass das System auch im Notfall betriebsbereit ist. Andererseits sind Integrität und Authentizität des Systems sicherzustellen, wobei eine etwaige Verfälschung von Daten und Manipulation von Systemfunktionen zu verhindern sind. Darüber hinaus sind im Rahmen der Vertraulichkeit sowohl die Anwendung als auch die Daten vor unbefugtem Zugriff zu schützen. – *Gewichtung:* 3 (Muss-Anforderung).

4.2.6. [R40] Wartbarkeit und Erweiterbarkeit

Die Webanwendung soll beispielsweise bei Änderungen des SIR-Prozesses mit vertretbarem Aufwand gewartet und erweitert werden können. Insofern ist der Quellcode in geeigneter Weise zu dokumentieren. – *Gewichtung:* 2 (Soll-Anforderung).

4.3. Übersicht der Anforderungen

In der folgenden tabellarischen Auflistung werden die in den vorherigen Kapiteln (4.1, 4.2) beschriebenen funktionalen (f) und nichtfunktionalen (nf) Anforderungen – gruppiert nach ihrer dreistufigen Gewichtung – dargestellt. Darüber hinaus wird auf die jeweiligen Kapitel hinsichtlich Konzept und Implementierung verwiesen.

ID	Anforderung	Art	Konzept	Impl.
Muss-Anforderungen (Gewichtung: 3)				
R01	Incidenterfassung mittels Webformular	f	5.2	6.3
R03	Initiale Incident Klassifikation	f	5.3.1	6.4.1
R06	Initiale Bearbeitung von Security Incidents	f	5.3.2	6.4.3
R07	Unterstützung von SSIs	f	5.3.1	6.4.1
R10	Weitere Incidentbearbeitung	f	5.3.3/4/5	6.4.3/5
R13	Erfassung der PIR-Ergebnisse	f	5.3.6	6.4.7
R14	Archivierung abgeschlossener Security Incidents	f	5.3.5	6.4.6

R17	Incidentstatusseite	f	5.3.5	6.2.5
R21	E-Mail-Benachrichtigungen	f	5.1.6	6.2.6
R23	Meldungen an die Allianz für Cyber-Sicherheit	f	5.1.6	6.2.6
R24	Such- und Filterfunktion	f	5.4.1	6.5.1
R26	Einzelexport eines Security Incidents	f	5.4.3	6.5.3
R28	Einfache statistische Auswertungen	f	5.1.5, 5.4.2	6.5.2
R32	Benutzerauthentifizierung	f	5.1.2	6.2.2
R35	Webbasierte Technik	nf	5.1.1	6.2.1
R36	Zentrale Datenhaltung in einer Datenbank	nf	5.1.1	6.2.1
R37	Lauffähigkeit auf einem Standard-Webserver	nf	5.1.1	6.2.1
R39	IT-Sicherheit	nf	5.1.3	6.2.3
Soll-Anforderungen (Gewichtung: 2)				
R04	Umgang mit Incident Duplikaten	f	5.3.1	6.4.2
R08	Priorisierung mittels Auswirkungsmatrix	f	5.3.2	6.4.3
R12	Wiedereröffnung von abgeschlossenen Incidents	f	5.3.5	6.4.6
R15	Prozess-Unterstützung für Anwender	f	5.1.5	6.2.5
R18	Dateiverwaltung	f	5.2	—
R20	Dokumentation aller Aktionen	f	5.1.5	6.2.5
R22	Dokumentation der Kommunikation	f	5.1.6	6.2.6
R29	Export von Auswertungen	f	5.4.2/3	6.5.3
R33	Rechtevergabe anhand statischer Benutzergruppen	f	5.1.2	6.2.2
R38	Gebrauchstauglichkeit und Benutzerfreundlichkeit	nf	5.1.4	6.2.4
R40	Wartbarkeit und Erweiterbarkeit	nf	5.1.1	6.1
Kann-Anforderungen (Gewichtung: 1)				
R02	Automatisierte Incidenterfassung durch SIEM-Lösungen	f	5.2	—
R05	Bearbeitung von Information Requests	f	5.3.1	6.4.2
R09	Information Request an den Vorfallmelder	f	5.3.2	6.2.6
R11	Erweitertes Monitoring	f	5.3.5	—
R16	Dashboard	f	5.1.4	6.2.4
R19	Automatisierte Handlungsempfehlungen	f	5.3.3	—
R25	Tagging von Security Incidents	f	5.3.5, 5.4.1	—
R27	Massenexport von Security Incidents	f	5.4.3	6.5.3
R30	Automatisierter E-Mail-Versand von Auswertungen	f	5.4.2	—
R31	LRZ-Branding der PDF-Dokumente	f	5.4.3	6.5.3
R34	Rechtevergabe anhand dynamischer Benutzergruppen	f	5.1.2	6.2.2

Tabelle 4.1.: Übersicht der Anforderungen

5. Konzept und Datenmodell

Bei zusammenfassender Betrachtung des im vorherigen Kapitel 4 ermittelten Anforderungsprofils wird deutlich, dass sich derart spezielle Anforderungen nicht durch den Einsatz einer Standardsoftware erfüllen lassen, sondern eine individuelle Softwarelösung erfordern. Das vorliegende Konzept stellt den Entwurf eines webbasierten Security Incident Managementtools dar, welches den SIR-Prozess des LRZ digital abbildet und dabei die im vorherigen Kapitel 4 ermittelten Anforderungen berücksichtigt. Abschließend wird ein Datenmodell erarbeitet, auf dem die speziell für das LRZ konzipierte Anwendung basiert.

Zunächst ist jedoch zu klären, welche Zielsetzung das LRZ mit der Einführung einer derartigen Software verfolgt. Zum einen ist hier die *technische Messbarkeit der Prozessperformance* anzuführen, die eine Auswertung von KPIs zur Identifikation von Trends und zur Prüfung, ob die in den SLAs zugesicherten Reaktionszeiten eingehalten werden, ermöglicht. Darüber hinaus lässt sich eine *Optimierung der Prozessperformance* durch effizienzsteigernde Teilautomatisierung und Vereinfachung von organisatorischen Tätigkeiten herbeiführen. Als weitere Ziele sind die Verbesserung der Qualität durch *Steigerung der Prozesskonformität* und die lückenlose Dokumentation zur *Erhöhung der Nachvollziehbarkeit* zu erwähnen. Letztendlich soll die Nutzung eines Managementtools zu einem professionelleren und zugleich effizienteren Umgang von Sicherheitsvorfällen und somit zur *Verbesserung des Gesamtsicherheitsniveaus* am LRZ beitragen.

Aus den im vorigen Kapitel ermittelten Anforderungen ergeben sich im Wesentlichen *drei Funktionsklassen*, die von einem webbasierten Security Incident Managementwerkzeug bereitzustellen sind:

1. Erfassung von Sicherheitsvorfällen
2. Prozesskonforme Bearbeitung von Sicherheitsvorfällen
3. Vielfältige Nutzung des Datenbestands

Während die erste Funktionsklasse die strukturierte Inputgenerierung in Form von Incidentdatensätzen umfasst, stellt die Datenverarbeitung im Prozesskontext als zweite Funktionsklasse die Kernaufgabe des Tools dar. Betrachtet man die abgeschlossenen Sicherheitsvorfälle als Prozessoutput, so dient die dritte Funktionsklasse der Datennutzung zu Recherche-, Auswertungs- und Exportzwecken.

Nach Analyse des SIR-Prozesses des LRZ ist festzustellen, dass dieser einem workfloworientierten Prozessmodell folgt und im Gegensatz zu anderen ITSM-Prozessen wie dem Change Management oder dem Configuration Management einen besonders hohen Reifegrad aufweist. Ausschlaggebend hierfür ist die fein gegliederte Strukturierung des Prozesses sowie eine kontinuierliche Prozessoptimierung durch regelmäßige Reviews. Da der Prozesskontext für die Konzeption der Webapplikation von sehr zentraler Bedeutung ist, werden die prozessspezifischen Grundlagen bereits an dieser Stelle definiert:

Zum einen stellt die Software unterschiedliche Subprozesse für die Bearbeitung von Security Incidents, Duplikaten und Information Requests bereit. Zum anderen stellt sie die Einhaltung der korrekten Reihenfolge der Prozessschritte sicher. Um eine derartige Prozesskontrolle technisch realisieren zu können, ist eine Unterteilung des Incident Lebenszyklus in mehrere Einzelphasen notwendig: `Neu`, `Klassifikation`, `Analyse`, `Lösung`, `Monitoring` und `Geschlossen`. Dabei wird jede Prozessphase systemintern durch eine Status-ID (vgl. Abbildung 5.1) repräsentiert, sodass die Software jeden Vorfall exakt in den Prozessablauf einordnen kann.

Abbildung 5.1.: Prozessphasen eines Sicherheitsvorfalls

Der Strukturierung des SIR-Prozesses wegen erfordert die gesamte zweite Funktionsklasse der Webanwendung eine workfloworientierte Ausrichtung. Das Flussdiagramm (Abbildung 5.2) gibt einen Überblick über die Abläufe des SIR-Prozesses und stellt die Prozessphasen mit ihren Aktivitäten sowie Prozessverzweigungen als Entscheidungsbäume dar.

Im Folgenden werden die Architekturentscheidungen erläutert, welche nach Prüfung verschiedener Möglichkeiten für die Realisierung der ermittelten Anforderungen getroffen werden. Bevor auf die Funktionsklassen eins bis drei eingegangen und schließlich das der Software zu Grunde liegende Datenmodell vorgestellt wird, richtet sich der Fokus zunächst auf grundlegende Designentscheidungen, welche die gesamte Webanwendung betreffen.

5.1. Grundlegende Designentscheidungen

Zu Beginn des Konzepts werden Entscheidungen getroffen, die eine hohe globale Relevanz für alle Funktionsklassen der Software aufweisen. Dazu zählen ebenso technische Aspekte hinsichtlich Programmiersprache und Datenbank, wie die Gestaltung der Benutzeroberfläche. Darüber hinaus sind Fragen der Prozessunterstützung, Benutzerauthentifizierung, IT-Sicherheit und Kommunikation zu klären.

5.1.1. Technik

Das Security Incident Managementwerkzeug wird der heute üblichen Praxis entsprechend als Webanwendung (`R35`) realisiert. Die webbasierte Technik unterstützt eine plattformunabhängige Nutzung mittels Webbrowser und ermöglicht mehreren Anwendern einen gleichzeitigen und ortsunabhängigen Zugriff.

Bei der Auswahl der Programmiersprache kommen daher grundsätzlich nur internettaugliche Sprachen in Frage: ASP.NET, JavaServer Pages, Perl, PHP, Python und Ruby on Rails. Nach Abwägung der Charakteristika der einzelnen Sprachen fällt die Wahl letztendlich auf die serverseitig interpretierte Skriptsprache PHP. Diese weist einen sehr hohen Verbreitungsgrad auf und ist für die Erstellung von Webanwendungen besonders geeignet, da sie eine saubere Implementierung des Internetprotokolls bietet und eine flexible Datenbankanbindung unterstützt. Darüber hinaus sind zahlreiche Funktionsbibliotheken frei verfügbar, die eine schnellere und komfortablere Projektrealisierung ermöglichen.

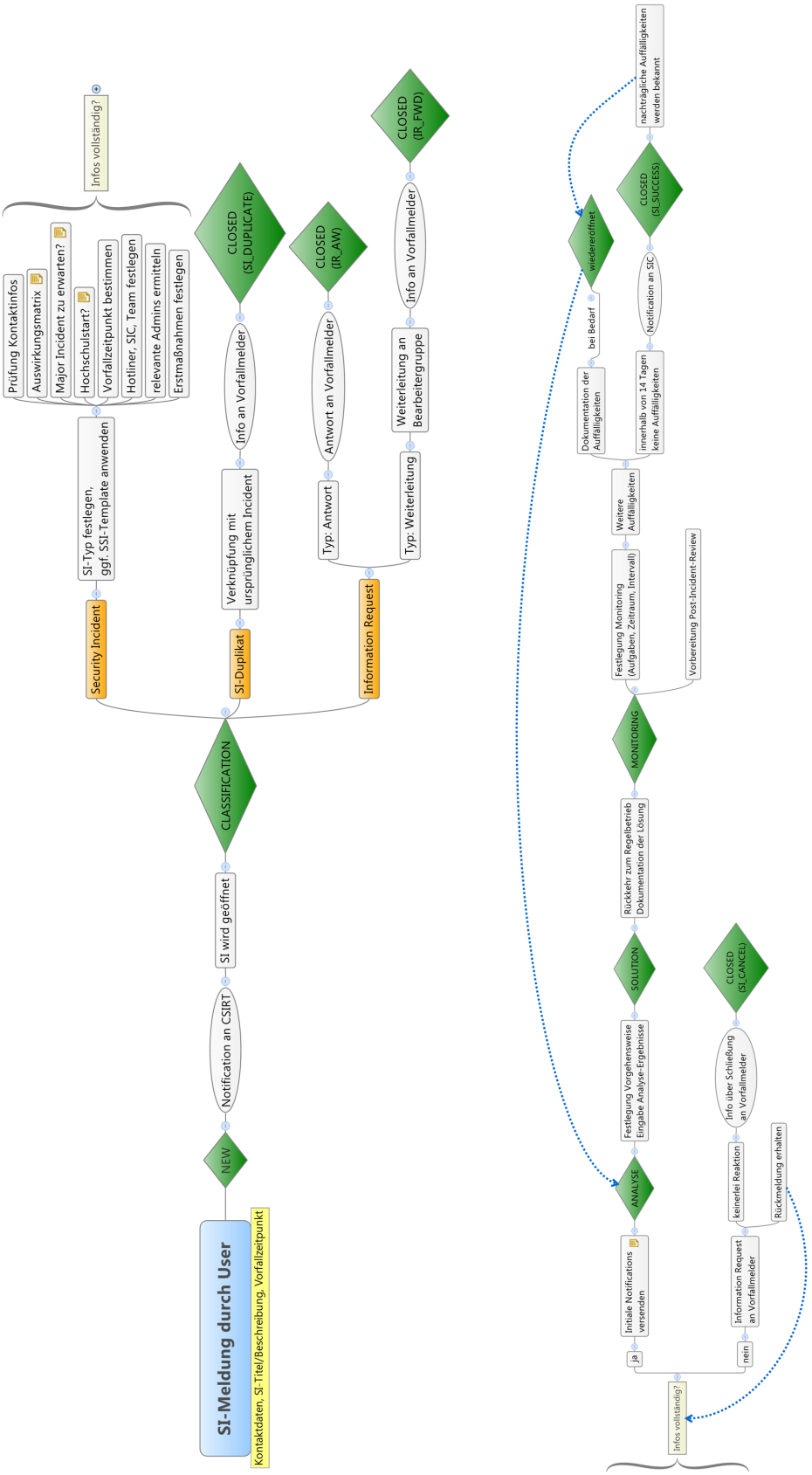

Abbildung 5.2.: Darstellung des SIR-Prozesses als Flussdiagramm

Für die Umsetzung einer zentralen Datenhaltung (R36)werden die relationalen Datenbanken MySQL, PostgreSQL und SQLite in Erwägung gezogen. Durch die Verwendung von PHP Data Objects, die eine Abstraktionsebene für den Datenbankzugriff darstellen, kann hinter der Webanwendung eine beliebige SQL basierte Datenbank stehen. Aufgrund der weiten Verbreitung kommt in diesem Projekt eine MySQL-Datenbank zum Einsatz. Sollte später dennoch ein Umstieg auf eine andere Datenbank erforderlich sein, so ist die Portierung aufgrund des abstrahierten Datenbankzugriffs mit minimalem Aufwand realisierbar.

Da der als Entwicklungsumgebung vom LRZ bereitgestellte Linux-Webserver sowohl über einen PHP-Interpreter in der Version 5.3.14 als auch über eine MySQL-Datenbank der Version 5.1.73 verfügt, werden die technischen Voraussetzungen für die Lauffähigkeit der PHP-MySQL-Anwendung auf einem Standard-Webserver gemäß R37 als erfüllt angesehen. Zugleich ist eine Einschränkung auf MyISAM-Tabellen zu berücksichtigen, da der MySQL-Server das Speichersubsystem InnoDB nicht unterstützt. Die beim Speichersubsystem MyISAM fehlende referenzielle Integrität von Fremdschlüsseln ist durch die Webanwendung sicherzustellen.

Um den Aufwand für eine spätere Anpassung und Erweiterung der Webanwendung gemäß R40 möglichst niedrig zu halten, ist auf eine flexible Implementierung und geeignete Kapselung der Softwarebestandteile zu achten. Eine hohe Flexibilität ist besonders bei Aufzählungsdatentypen wie Incident- und Lösungstypen hilfreich. Damit diese unkompliziert und ohne Arbeiten am Quellcode geändert und erweitert werden können, erscheint die zentrale Speicherung der Werte in einer Typentabelle anstelle einer festen Codierung der Ausprägungen im Quelltext vorteilhaft.

Komplexe Bereiche des Quellcodes sind aus Gründen der besseren Nachvollziehbarkeit ausführlich zu dokumentieren. Insbesondere ist eine einheitliche Dokumentation aller Funktionen sicherzustellen, welche jeweils eine Beschreibung und den Rückgabewert der Funktion beinhaltet sowie die Input-Parameter spezifiziert.

Zur komfortablen Wartung der Webanwendung können alle Systemeinstellungen an zentraler Stelle editiert werden. Hierfür erscheint die Verwendung einer Konfigurationsdatei und einer Datenbanktabelle für die Speicherung von Systemeinstellungen sinnvoll.

5.1.2. Benutzerverwaltung, Authentifizierung und Rechtevergabe

Damit der Zugriff auf die Webanwendung und die von ihr verwalteten sicherheitsrelevanten Informationen nur befugten Anwendern möglich ist, muss sie gemäß Anforderung R32 über einen geeigneten Mechanismus zur Benutzerauthentifizierung verfügen. Darüber hinaus soll das System die Kontaktdaten jedes Anwenders als Benutzerprofil verwalten können, sodass keine wiederholte Eingabe von Kontaktinformationen notwendig ist.

Als primitive Möglichkeit der Authentifizierung auf Benutzerebene ist der .htaccess-Verzeichnisschutz zu betrachten. Die beiden Konfigurationsdateien, welche u.a. die Benutzernamen und Passwörter der Anwender enthalten, befinden sich im Root-Verzeichnis der Webanwendung und schränken den Zugriff auf dieses Verzeichnis inklusive seiner Unterverzeichnisse ein. Diese Authentifizierungsmöglichkeit wäre zwar einfach zu implementieren, ist aber für eine derartige Webanwendung aus mehrerlei Hinsicht nicht geeignet. Als wesentliche Nachteile seien die starke Verzeichnisorientierung sowie die eingeschränkten Funktionen hinsichtlich Sitzungsverwaltung und feingranularer, gruppenbasierter Rechtevergabe erwähnt.

Die Kombination des Session-Konzepts mit der Verwaltung von Benutzerdaten und Kennwörtern in einer Datenbank stellt einen bei Websystemen häufig angewendeten Ansatz dar. Dabei wird nach erfolgreicher Authentifizierung für jeden Benutzer eine Sitzung eröffnet, welche durch die Webanwendung verwaltet wird und eine benutzerbezogene Nachvollziehbarkeit der Aktivitäten ermöglicht. Je nach Implementierung kann sowohl eine Rechtevergabe anhand statischer (R33) als auch dynamischer (R34) Benutzergruppen realisiert werden. Dieser Ansatz bedeutet jedoch in Umgebungen mit vielen Anwendern einen hohen administrativen Aufwand. Beispielsweise müsste bei der Einstellung eines neuen Mitarbeiters ein neues Benutzerprofil in der Webanwendung angelegt und beim Ausscheiden wieder gelöscht werden. Darüber hinaus würde jeder Mitarbeiter separate Zugangsdaten für dieses System benötigen. Zuletzt müsste die Webanwendung die Konformität mit der LRZ-Passwortrichtlinie sicherstellen und Service-Funktionen zum Zurücksetzen des Passworts bereitstellen.

Zur Vermeidung der genannten Nachteile wird die Anwendung des Session-Konzepts mit einer LDAP-Anbindung für sinnvoll erachtet. Da das LRZ bereits einen LDAP-Verzeichnisdienst betreibt, bietet sich dessen Integration zur Nutzerauthentifizierung und zum Auslesen von Gruppenzugehörigkeit und Kontaktdaten bestens an. Dadurch kann sich jeder LRZ-Mitarbeiter mit seiner zentralen LRZ-Kennung an der Security Incident Management Software anmelden, ohne ein separates Benutzerprofil anlegen und pflegen zu müssen. Ein minimaler Administrationsaufwand, ein höchstmöglicher Komfort für die Anwender durch einen Single-Sign-On-ähnlichen Login-Mechanismus und der Zugriff auf stets aktuelle Kontaktdaten zeichnen diese Variante aus. Damit auch bei Ausfall des LDAP-Dienstes noch ein Zugriff auf die Webanwendung möglich ist, existiert ein Root-Nutzer mit lokalem Authentifizierungsverfahren.

Um einen performanten Zugriff auf die Kontaktdaten aller Benutzer zu gewährleisten, werden diese durch die Webanwendung automatisch in einer Datenbanktabelle gespeichert und täglich durch einen LDAP-Abgleich auf dem aktuellen Stand gehalten.

Die mit dem Login initiierte Benutzersitzung gilt zunächst für 30 Minuten, wobei die Sitzungsdauer bei jedem Seitenwechsel von Neuem beginnt. Der Anwender wird rechtzeitig vor dem Ablaufen der Sitzung daran erinnert, seine Eingaben zu speichern. Erfolgt dies nicht, ist davon auszugehen, dass der Nutzer nicht mehr aktiv ist, weswegen die Session aus Sicherheitsgründen automatisch beendet wird.

Über die Benutzerauthentifizierung hinaus ist eine Rechtevergabe anhand statischer Benutzergruppen (R33) vorgesehen. Es werden drei Benutzergruppen bzw. Berechtigungslevel unterschieden:

- `USER`: LRZ-Mitarbeiter können nur Incidents melden.
- `CSIRT`: CSIRT-Mitglieder können Vorfälle melden, bearbeiten, filtern und durchsuchen sowie auswerten.
- `ROOT`: Root-Nutzer haben alle CSIRT-Rechte zuzüglich des Änderungsprivilegs für System-Einstellungen.

Die Rechtevergabe anhand dynamischer Benutzergruppen ist in R34 als wünschenswerte Anforderung beschrieben. Dennoch soll die Rechtevergabe grundsätzlich auf statischen Benutzergruppen basieren. Diese wird in speziellen Fällen durch eine dynamische Rechtevergabe ergänzt. Beispielsweise sollen alle Benutzer, die an einem Incident beteiligt sind, unabhängig

von ihrer Zugehörigkeit zu einer Benutzergruppe die Statusinformationen eines Vorfalls abrufen können. Dadurch können sich auch die Prozessbeteiligten ohne CSIRT-Rechte (beispielsweise der Vorfallmelder, Abteilungsleiter oder Netzadministrator) jederzeit über den aktuellen Zustand informieren, was wiederum eine Reduzierung vermeidbarer Nachfragen begünstigt.

5.1.3. IT-Sicherheit

Bedingt durch die Hauptaufgabe des Systems, das Management von Sicherheitsvorfällen effizienzsteigernd zu unterstützen, ist die IT-Sicherheit dieser Managementsoftware speziell zu berücksichtigen. Neben den in R39 beschriebenen Anforderungen hinsichtlich Verfügbarkeit, Integrität, Vertraulichkeit und Authentizität des Systems sind auch die in den „Goldene(n) Regeln für (...) Webanwendungen“ [Bun12a] enthaltenen Empfehlungen des BSI zu beachten.

Um externe Angriffe auf das System und die von ihm verwalteten Informationen auszuschließen, erscheint es sinnvoll, die Erreichbarkeit der Webanwendung auf das LRZ-Netz zu beschränken. Da das Vorhandensein von Innentätern insbesondere in einem Rechenzentrum nicht auszuschließen ist, sind darüber hinaus weitere Sicherheitsmaßnahmen zu treffen.

Da die Webanwendung vertrauliche Informationen verwaltet und über das Netz überträgt, sollte die Datenübermittlung verschlüsselt erfolgen. Es ist daher bei der Implementierung sicherzustellen, dass die Webanwendung das HTTPS-Protokoll unterstützt und die schützenswerten Daten „nicht in der URL übertragen werden“ [Bun12a].

Das Risiko eines Shoulder Surfing Angriffs im öffentlichen Raum – beispielsweise in der S-Bahn – wird als wesentlich höher eingeschätzt als in den LRZ-Büros. Daher wird im Rahmen einer präventiven Sicherheitsmaßnahme die mobile Nutzung der Applikation auf Smartphones und Tablets explizit unterbunden.

Darüber hinaus ist eine „unbefugte Nutzung geschützter Funktionen sowie die Einsicht und Manipulation von geschützten Informationen“ [Bun12a] durch eine serverseitig zu implementierende Zugriffskontrolle zu verhindern. Die Themen „Authentifikation“ und „Sitzungsverwaltung“ wurden bereits im vorigen Abschnitt ausführlich diskutiert. Ergänzend sei erwähnt, dass aufgrund der LDAP-Authentifizierung keine Notwendigkeit besteht, die Passwörter der Anwender in der Datenbank zu speichern. Dadurch können selbst im Falle eines böswilligen Datenbankzugriffs keine Kennwörter ausgelesen werden. Darüber hinaus werden fehlgeschlagene Anmeldeversuche vom LDAP-Verzeichnisdienst registriert, wobei beim Erreichen eines Grenzwerts die Kennung automatisch gesperrt wird. Zusätzlich stellt die Webanwendung eine Logging-Funktion bereit, die beispielsweise fehlerhafte Login-Versuche protokolliert, sodass derartige „sicherheitskritische Vorfälle nachvollzogen werden können“ [Bun12a].

Alle Benutzereingaben sind grundsätzlich als nicht-vertrauenswürdig einzustufen. Zur Vermeidung von Schwachstellen wie Cross-Site-Scripting (XSS) und (SQL)-Injections sind alle Eingaben vor deren Weiterverarbeitung serverseitig zu validieren und durch Maskierung unschädlich zu machen. Dabei kann die Validierung durch einen clientseitig vorgeschalteten Prüfungsmechanismus unterstützt werden.

Zum Schutz der Anwendungslogik „gegen automatisierte und logische Angriffe“ [Bun12a] wird jede Aktion mit einem Token autorisiert, der für die Dauer einer Session konstant bleibt. Würde sich das Token bei jedem Seitenaufruf ändern, könnte man zwar ein noch

höheres Maß an Sicherheit realisieren, schränkt jedoch gleichzeitig die parallele Nutzung der Webanwendung in mehreren Browserfenstern ein. Da an modernen Arbeitsplätzen üblicherweise zwei Monitore zur Verfügung stehen, könnte am linken Monitor ein zu bearbeitender Sicherheitsvorfall geöffnet werden, dem der Token A zugewiesen wird. Öffnet man zu Recherchezwecken auf dem rechten Bildschirm mittels Filterfunktion einen ähnlichen Incident, so wird diesem ein neuer Token B zugeordnet. Führt man nun am zu bearbeitenden Vorfall im linken Browserfenster eine Aktion durch, so wird diese vom Server abgewiesen, da der alte Token A übermittelt wurde, das System aber den aktuellen Token B erwartet hat. Da die Nutzbarkeit der Anwendung im Multi-Monitor-Betrieb als notwendig erachtet wird, kommt ein Session-Token zum Einsatz.

Um das Risiko der Verfälschung von Daten und der Manipulation von Systemfunktionen zu reduzieren, sind sowohl für die FTP-Kennung als auch für den MySQL-Benutzer sichere Passwörter zu verwenden, die in regelmäßigen Abständen geändert werden. Zur Optimierung der Integrität und Authentizität des Systems kann der Benutzerzugriff auf Ordner, in denen Systemdateien abgelegt sind, durch den `.htaccess`-Verzeichnisschutz unterbunden werden. Dadurch wird das Aufrufen von Systemdateien mit manipulierten Parametern außerhalb des zulässigen Kontexts und das Einbinden dieser in fremde Skripte verhindert.

Zur Gewährleistung einer hohen Verfügbarkeit der Webanwendung könnte diese auf einem separaten virtuellen Web- und Datenbankserver gehostet werden, der auf einer redundant ausgelegten Hardware betrieben wird. Das Hosting auf einem dedizierten Webserver stellt einen gegenüber anderen Anwendungen, die die Stabilität und Performance des Webservers beeinflussen könnten, isolierten Betrieb sicher. Daraus ergibt sich eine Entkapselung der Webanwendung von den Sicherheitsrisiken anderer Systeme und deren potenziellen Auswirkungen. Bei den Überlegungen zur Verfügbarkeit sind auch die Abhängigkeiten der Webanwendung von anderen Diensten zu betrachten. Da die Benutzerauthentifizierung von der Verfügbarkeit des LDAP-Verzeichnisdienstes abhängt, wäre die Webanwendung bei einem LDAP-Ausfall zwar verfügbar, aber nicht verwendbar. Obwohl dieser Dienst als hochverfügbar gilt, ist für den Notfall eine unabhängige Login-Möglichkeit für den Root-User vorgesehen. Eine weitere Abhängigkeit besteht hinsichtlich des Mailservers, der für den Versand von Benachrichtigungen zuständig ist. Da der E-Mail-Dienst für die Webanwendung nicht von essenzieller Bedeutung und ohnehin hochverfügbar ist, wird für den Mailversand keine Fallbacklösung bereitgestellt.

5.1.4. Benutzeroberfläche

Eine leicht verständliche und intuitiv bedienbare Benutzeroberfläche trägt wesentlich dazu bei, dass die Bedienung der Webanwendung von den Anwendern als angenehm empfunden wird und ohne größeren Einarbeitungsaufwand erlernt werden kann. Um den Fokus auf die wesentlichen Aspekte der Webanwendung richten zu können, zugleich aber ein professionelles Userinterface gemäß `R38` realisieren zu können, kommt ein modernes HTML5/CSS3-Designtemplate zum Einsatz.

Die Benutzeroberfläche weist eine einheitliche visuelle Struktur auf und ermöglicht dadurch eine schnelle Orientierung. Die farblich abgesetzte horizontale Kopfzeile beinhaltet neben der Softwarebezeichnung auch Funktionen für den Schnellzugriff. Am linken Fensterrand befindet sich eine vertikale, einklappbare Menüleiste, sodass im minimierten Zustand mehr Fläche für den Inhaltsbereich zur Verfügung steht. Dies ermöglicht beispielsweise eine komfortable

Bedienung an einem Notebook mit kleinem Display und niedriger Auflösung. Der großflächige Inhaltsbereich erstreckt sich über das gesamte restliche Browserfenster. Die flexible Anpassung des Designrahmens an die Fensterbreite erlaubt eine übersichtliche Anordnung der Informations- und Bedienelemente. Eine durchdachte Farbgestaltung, die nicht ablenkt, sondern zum Ziel führt, unterstützt eine effiziente Arbeitsweise.

Der Anwender kann zwischen allen Hauptbereichen der Webanwendung unkompliziert hin- und herwechseln. Daher werden alle wesentlichen Funktionen auf der ersten Ebene der Navigationsleiste in sinnvoller Reihenfolge angeordnet:

- Dashboard
- Meldung
- Bearbeitung
- Suchen & Filtern
- Auswertungen
- Einstellungen

Auf eine Untergliederung der Funktionen in Untermenüs wird zu Gunsten einer optimalen Bedienbarkeit explizit verzichtet. Einzig für den Menüpunkt „Einstellungen“ ist für die Abbildung der verschiedenen Konfigurationsbereiche eine Erweiterung auf eine zweite Navigationsebene vorgesehen.

Weniger relevante Funktionen (Benutzerprofil, About, Logout) werden in ein schlankes Menü innerhalb der Kopfleiste ausgelagert, welches nur bedarfsweise eingeblendet wird. Darüber hinaus kann an dieser Stelle ein Überblick über neue und in Bearbeitung befindliche Vorfälle eingeblendet werden. Zudem ist durch Eingabe der ID eines Vorfalls ein effizienter Schnellzugriff auf dessen Statusseite möglich, ohne die Suchfunktion aufrufen zu müssen.

Das Dashboard fungiert nach dem Login als Einstiegsseite der Webanwendung und setzt sich in Abhängigkeit von der Gruppenzugehörigkeit des angemeldeten Benutzers aus verschiedenen Widgets zusammen (`R16`). CSIRT-Mitglieder und Root-User können sich durch Betrachtung des Dashboards einen Überblick über die aktuelle Sachlage verschaffen. Zu diesem Zweck werden in einem Widget alle neuen und noch laufenden Security Incidents unter Angabe von ID, Bezeichnung, Priorität, Status, Incidenttyp, IP-Adresse, Kunde und Meldedatum tabellarisch aufgelistet. Ein anderes Widget stellt aktuelle Ereignisse dar, aus denen ersichtlich ist, welcher Vorfall zuletzt bearbeitet wurde und welche Daten dabei eingegeben bzw. geändert wurden. Bei allen übrigen LRZ-Mitarbeitern, die der Gruppe `USER` angehören, setzt sich das Dashboard aus zwei anderen Widgets zusammen. Während das erste einen Schnellzugriff auf das Formular zur Meldung eines neuen Incidents ermöglicht, stellt das zweite Widget unter Berücksichtigung der zuvor genannten Parameter eine Auflistung aller Vorfälle dar, an denen der Mitarbeiter beteiligt ist oder war. An dieser Stelle hat der Mitarbeiter Zugriff auf eine angepasste Version der zugehörigen Incidentstatusseiten, um sich aktiv über den Status der für ihn relevanten Vorfälle informieren zu können.

5.1.5. Prozessunterstützung

Eine der zentralen Aufgaben der Webanwendung besteht darin, den SIR-Prozess des LRZ digital abzubilden. Wie bereits eingangs erwähnt, wird zu diesem Zweck der Lebenszyklus eines

Sicherheitsvorfalls in durch das System kontrollierbare Phasen unterteilt. Dabei stellt die Anwendung einerseits die richtige Reihenfolge der Prozessschritte sicher, wobei die Zulässigkeit von Phasenübergängen durch Prüfung von Vorbedingungen überwacht wird. Andererseits soll die Software den Anwender gemäß `R15` bei der Bearbeitung des Prozesses unterstützen, indem sie ihn auf intuitive Weise durch die Prozessphasen begleitet und zu jedem Zeitpunkt genau diejenigen Funktionen bereithält, die im jeweiligen Prozessstadium sinnvoll sind. Zur Konkretisierung dieser Anforderung sind mehrere Maßnahmen denkbar:

Eine geschickte Ergänzung der grafischen Benutzeroberfläche soll es den CSIRT-Mitgliedern erleichtern, den Überblick zu behalten. Zu diesem Zweck ist für den Bereich der Incidentbearbeitung eine zusätzliche Menüleiste vorgesehen, die dem Anwender eine Navigation durch die Prozessphasen des aufgerufenen Vorfalls ermöglicht. Dabei sind die Phasen als Menüpunkte wie auf einem Zeitstrahl horizontal angeordnet:

- **Übersicht:** Auf der Incidentstatusseite werden gemäß `R17` alle zu einem Vorfall vorhandenen Informationen übersichtlich dargestellt.
- **Vorfall:** Diese Seite dient der Bearbeitung der Vorfalldaten sowie der Klassifikation und Priorisierung.
- **Team:** Im Team-Bereich wird das Security Incident Team verwaltet. Dabei werden der Hotliner und der Security Incident Coordinator festgelegt. Auch Systemadministratoren können dem Team hinzugefügt werden.
- **Analyse:** Dokumentation der Maßnahmen und Analyseergebnisse während der Analysephase.
- **Lösung:** Dokumentation der Wiederherstellung des Regelbetriebs und der Incidentlösung während der Lösungsphase.
- **Monitoring:** Festlegung der Monitoringparameter und Dokumentation von Auffälligkeiten während der Monitoringphase.
- **Review:** Dokumentation der Ergebnisse des Post Incident Reviews.
- **Kommunikation:** Das Kommunikationsmodul ermöglicht neben dem Versand von E-Mails und der Erfassung von Gesprächsnotizen einen Überblick über alle Kommunikationsvorgänge.
- **Verlauf:** Alle Aktivitäten werden automatisch im Verlaufsprotokoll dokumentiert.

Damit sich der Anwender optisch einfacher orientieren kann, wird der aktuell aufgerufene Menüpunkt farbig hervorgehoben. Zudem werden nicht relevante Punkte als inaktiv dargestellt, sodass bei einem Vorfall, der sich beispielsweise gerade in der Analysephase befindet, die noch nicht relevanten Punkte Lösung, Monitoring und Review grau dargestellt werden und nicht anwählbar sind.

Im gesamten Bereich der Incidentbearbeitung befinden sich am rechten Fensterrand drei Widgets in Form einer Sidebar. Der erste Block stellt die wesentlichen Vorfalldaten zusammenfassend dar, sodass beispielsweise stets der Incidenttyp, die Priorität, der Kunde und die aktuelle Prozessphase auf einen Blick ersichtlich sind, ohne zur Statusseite wechseln zu müssen. Das zweite Widget enthält Empfehlungen für den Anwender. Diese können anhand der Incident Eigenschaften ermittelt werden. Beispielsweise wird der Anwender daran erinnert, die LRZ-Leitung zu informieren, falls die Priorität eines Vorfalls mindestens hoch ist

oder angegeben wurde, dass ein Major Incident zu erwarten ist. Im letzten Fall wird außerdem empfohlen, den Major Incident Coordinator zu kontaktieren. Sofern der Hochschulstartdienst betroffen ist, weist das Widget darauf hin, dass spezielle Rücksprachen erforderlich sind. Der Empfehlungsmechanismus berücksichtigt einige Detailhinweise aus der Prozessbeschreibung und kann dadurch zur Steigerung der Prozesskonformität beitragen. Das dritte Widget stellt in Abhängigkeit von der aktuellen Prozessphase jeweils sinnvolle Aktionen bereit. Während der Schnellzugriff auf die Kommunikationsfunktionen (E-Mail und Gesprächsnotiz) und die verschiedenen Exportmöglichkeiten immer zur Verfügung steht, ist das Abbrechen eines Incidents nur möglich, solange dieser noch nicht abgeschlossen wurde. Eine Meldung an die Allianz für Cyber-Sicherheit ist hingegen erst möglich, sobald der Incident gelöst wurde und die Wiedereröffnung eines Incidents setzt voraus, dass dieser gerade geschlossen ist. Die kontextabhängige Zusammenstellung der Funktionen an einem zentralen Ort unterstützt die intuitive Bedienbarkeit der Software.

Um im Rahmen von Auswertungen gemäß `R28` Kennzahlen ableiten zu können, ist eine Messung der Performance auf Prozessphasenebene nötig. Hierfür wird der Performancefaktor „Zeit“ zu Rate gezogen, indem die Dauer jeder Prozessphase berechnet wird, wobei zwischen der tatsächlichen Dauer und der Dauer während der in den SLAs zugesicherten Servicezeitfenstern zu unterscheiden ist. Während sich die `Reaktionszeit` über die ersten beiden Phasen erstreckt, bezieht sich die `Analysezeit` auf die dritte Phase, die `Lösungszeit` auf die vierte und die `Nachlaufzeit` auf die Monitoringphase.

Der Nachweisbarkeit einer prozesskonformen Arbeitsweise wegen sind während allen Prozessphasen sämtliche Änderungen automatisch in einem incidentbezogenen Verlaufsprotokoll zu dokumentieren (`R20`). Dadurch ist jederzeit nachvollziehbar, welcher Nutzer zu welchem Zeitpunkt welche Informationen eingegeben bzw. geändert hat. Um später eine exakte Analyse der Vorgehensweise durchführen zu können, wird eine feingranulare Dokumentation als notwendig erachtet, wobei je geändertem Datenbankfeld ein eigener Eintrag in der Historisierungstabelle erfolgt. Jeder Datensatz umfasst folgende Attribute:

- Zeitstempel der Aktivität
- Betroffenes Datenbankfeld
- Wert des Feldes vor der Änderung
- Wert des Feldes nach der Änderung
- Agierender Benutzer

Das Verlaufsprotokoll ist manipulationssicher zu implementieren, sodass Einträge weder gelöscht noch editiert oder in anderer Weise verfälscht werden können. Ferner muss sichergestellt sein, dass keine Datenänderungen möglich sind, die den Dokumentationsautomatismus umgehen.

5.1.6. Kommunikation

Da eine reibungslose Kommunikation die effiziente Incidentbearbeitung unterstützt, dient die Webanwendung dem CSIRT auch als vollintegrierte Kommunikationszentrale und ist auf Incident Ebene angesiedelt. Im Gegensatz zum E-Mail-Versand mit einem E-Mail-Client wie Microsoft Outlook bietet ein zentrales Kommunikationsmodul mehrere Vorteile: Da die E-Mail-Adressen der Prozessbeteiligten im System hinterlegt sind, entfällt das manuelle

Übertragen der Empfängeradresse(n) in das externe E-Mail-Programm. Auch incidentbezogene Daten, die der Webanwendung ohnehin zur Verfügung stehen, können ohne manuellen Transferaufwand verschickt werden. Schließlich ist die versendete Nachricht nicht dezentral im Ordner „Gesendete Elemente“ des persönlichen Mailpostfachs dokumentiert, sondern im Kommunikationsprotokoll des Vorfalls für alle CSIRT-Mitglieder abrufbar. Um alle relevanten Kommunikationskanäle zu berücksichtigen, können auch Gesprächsnotizen von Telefonaten und persönlichen Besprechungen sowie im E-Mail-Client eingegangene E-Mails zentral dokumentiert werden. Daraus ergibt sich ein lückenloses Protokoll aller Kommunikationsvorgänge. Da dieses für alle CSIRT-Mitglieder einsehbar ist, werden interne Rückfragen in vielen Fällen überflüssig.

Darüber hinaus könnte das Kommunikationsmodul dahingehend erweitert werden, dass sich die Anwender innerhalb des Systems gegenseitig Nachrichten schicken und Dateianhänge beifügen können. Der E-Mail-Versand wäre dann nur noch für externe Parteien relevant, die über keinen Systemzugriff verfügen. Da dieser Aspekt im Anforderungsprofil nicht enthalten ist, wird diese optionale Funktionalität im Rahmen dieser Arbeit nicht weiter betrachtet, könnte aber in einem späteren Softwarerelease ergänzt werden.

Zunächst soll der in `R21` geforderte E-Mail-Versand über das Kommunikationsmodul näher betrachtet werden: Für die Auswahl des Empfängers steht dem Anwender ein Drop-Down-Adressbuch zur Verfügung, das alle am Prozess beteiligten Personen enthält, deren E-Mail-Adressen dem System bekannt sind. Alternativ kann auch eine Empfängergruppe (Security Incident Team, CSIRT oder LRZ-Leitung) ausgewählt werden, wobei die gleiche E-Mail an alle Mitglieder der Gruppe gesendet wird. Zuletzt ist auch die manuelle Eingabe eines Empfängers möglich. Anschließend sind sowohl der Betreff als auch die Nachricht einzutragen. Eine Besonderheit stellt die Möglichkeit dar, Incident Daten anzufügen. Da nur die unbedingt benötigten Informationen herausgegeben werden sollen, kann der Anwender den Informationsumfang mittels Checkboxen bestimmen. Ihm stehen hierfür folgende Datengruppen zur Verfügung: Vorfalldaten, Kontaktdaten, Maßnahmen, Lösung, Monitoring und Review. Die selektierten Informationen werden an den E-Mail-Text angehängt und als Vorschau angezeigt. Dadurch hat der Absender die Möglichkeit, die zu versendenden Daten weiter einzuschränken und zu editieren.

Nach dem Versand der E-Mail wird diese automatisch im System gespeichert. Darüber hinaus können gemäß `R22` auch alle anderen Kommunikationsvorgänge wie Telefonate und persönliche Gespräche im System dokumentiert werden. Bei deren Erfassung ist zunächst das Medium und anschließend die Kommunikationsrichtung (eingehend, ausgehend) mittels Dropdown-Felder auszuwählen. Auch der Kommunikationszeitpunkt ist zu erfassen, wobei der aktuelle Zeitstempel voreingestellt ist. Auch wenn die Kernaufgabe der Funktion in der Erstellung von Gesprächsnotizen besteht, so können auch eingegangene E-Mails mit ihren korrekten Zeitstempeln dokumentiert werden. Der Gesprächspartner bzw. E-Mail-Absender kann aus dem Adressbuch gewählt oder manuell eingegeben werden. Während der Betreff der Gesprächsnotiz das Thema kurz zusammenfasst, beinhaltet das Nachrichtenfeld die ausgetauschten Informationen und die konkreten Gesprächsergebnisse.

Das so entstehende Kommunikationsprotokoll listet alle Einträge chronologisch sortiert auf und beinhaltet für jeden Kommunikationsvorgang folgende Informationen: Zeitpunkt, Medium, Richtung, Empfänger, Betreff und das agierende Teammitglied. Für jeden Kommunikationsvorgang existiert eine Detailseite mit den vollständigen Informationen inklusive der eigentlichen Nachricht.

Das Kommunikationsmodul wird gemäß R21 auch durch die Webanwendung selbst für den automatisierten Versand von ereignisbedingten E-Mails verwendet:

- Information an das CSIRT bei Eingang eines neuen Incidents
- Eingangsbestätigung des neuen Incidents an den Vorfallmelder
- Information an den Vorfallmelder beim erfolgreichen Abschluss eines Incidents
- Information an den Vorfallmelder bei Wiedereröffnung eines Incidents
- Information an den Vorfallmelder bei Abbruch eines Incidents
- Information an den Vorfallmelder, wenn ein Incident als Duplikat klassifiziert und abgeschlossen wurde
- Information an den Vorfallmelder, wenn ein Incident als Information Request klassifiziert und dieser an eine Bearbeitergruppe weitergeleitet wurde

Darüber hinaus werden einmal täglich E-Mails an die Security Incident Koordinatoren aller derjenigen Vorfälle geschickt, die sich seit mindestens fünf Tagen in der gleichen Prozessphase befinden. Das System vermutet aufgrund der hohen Verweildauer in der gleichen Prozessphase, dass ein Vorfall nicht optimal bearbeitet wird und bittet den SIC um Prüfung des Vorfalls.

Eine Ausnahme bilden Sicherheitsvorfälle, die sich in der Monitoringphase befinden. Wegen der längeren Dauer dieser Phase werden zugehörige Security Incidents nicht bereits nach fünf, sondern erst nach 14 Tagen angemahnt. Dabei empfiehlt das System zu prüfen, ob weitere Auffälligkeiten festgestellt wurden und ob der Vorfall geschlossen werden kann.

Gemäß R23 soll die Webanwendung eine Möglichkeit zur vereinfachten Meldung von Sicherheitsvorfällen an die Allianz für Cyber-Sicherheit (ACS) zur Verfügung stellen. Diese vom Bundesamt für Sicherheit in der Informationstechnik (BSI) gegründete Initiative verfolgt das Ziel, das „Know-how zum Thema Cyber-Sicherheit in Deutschland zu bündeln und die Allianz als zentrale Anlaufstelle für Informationen zu Cyber-Sicherheitsfragen zu etablieren“ [Bun14]. Unter anderem generiert die ACS ein einheitliches Lagebild. Damit die am LRZ aufgetretenen Sicherheitsvorfälle in dieses Lagebild einfließen können, werden diese bislang über das Online-Meldeformular an die Allianz für Cyber-Sicherheit übertragen. Die Webanwendung soll die Benachrichtigung der ACS effizienter gestalten. Dabei wird eine Übertragung der Daten mittels XML-Schnittstelle, vorausgefülltem Webformular oder speziell formatierter E-Mail in Betracht gezogen. Eine E-Mail-Anfrage an die Meldestelle des BSI soll klären,

1. ob für die Meldung eine (XML-)Schnittstelle verfügbar ist,
2. wie die Daten im Falle einer E-Mail-Meldung aufbereitet sein müssten, um die Informationen seitens BSI automatisiert weiterverarbeiten zu können,
3. ob der Aufbau des Online-Meldeformulars und dessen Feldbezeichnungen für längere Zeit unverändert bleiben, um die Formularfelder mittels URL-Parameter vorbelegen zu können.

Ein Mitarbeiter des BSI teilt daraufhin telefonisch mit, dass die eingehenden Meldungen zum aktuellen Zeitpunkt (07.10.2013) nicht automatisiert weiterverarbeitet werden und daher weder ein XML-Schema noch eine vergleichbare Webschnittstelle existiert. Hinsichtlich der

zweiten und dritten Variante konnte der Mitarbeiter auch nach interner Rücksprache keine Festlegung treffen, welcher Meldeweg letztendlich durch das BSI präferiert wird.

Da das BSI nicht garantieren kann, dass sich das Meldeformular nicht in näherer Zeit im Aufbau ändern und gelegentlich an anderer Stelle zu erreichen sein wird, kommt die Variante der Formularvorbelegung mit `GET`-Parametern nicht in Frage, zumal auch Seiteneffekte wie maximal mögliche Parameterlängen im Browser zu berücksichtigen wären. Im Vergleich dazu erscheint die E-Mail-Lösung wesentlich robuster, da keine Abhängigkeit hinsichtlich des Aufbaus des Meldeformulars besteht und die Meldestelle langfristig unter der gleichen E-Mail-Adresse erreichbar sein wird.

Folglich bildet die Webanwendung das ACS-Meldeformular nach, wobei die dem System bekannten Felder bereits vorausgefüllt werden. Die Formulareingaben werden schließlich per E-Mail an `Meldestelle@bsi.bund.de` gesendet und im Kommunikationsprotokoll der Webanwendung dokumentiert. Es wird darauf hingewiesen, dass alltägliche Sicherheitsvorfälle nicht zu melden sind, da die ACS in erster Linie an Incidents von besonderer Relevanz interessiert ist. Daher ist die Meldung an die Allianz für Cyber-Sicherheit nach Abschluss eines Sicherheitsvorfalls als optional zu betrachten.

5.2. Erfassung von Sicherheitsvorfällen

Unter Berücksichtigung der grundlegenden Designentscheidungen richtet sich nun der Fokus auf die Erfassung von Sicherheitsvorfällen (erste Funktionsklasse).

Nach dem Bekanntwerden eines Sicherheitsvorfalls soll dieser unverzüglich an das CSIRT gemeldet werden. Abbildung 5.3 stellt die möglichen Meldewege bis hin zur Incidenterfassung in der Webanwendung dar und berücksichtigt dabei auch Drittanbietersoftware als Meldequelle.

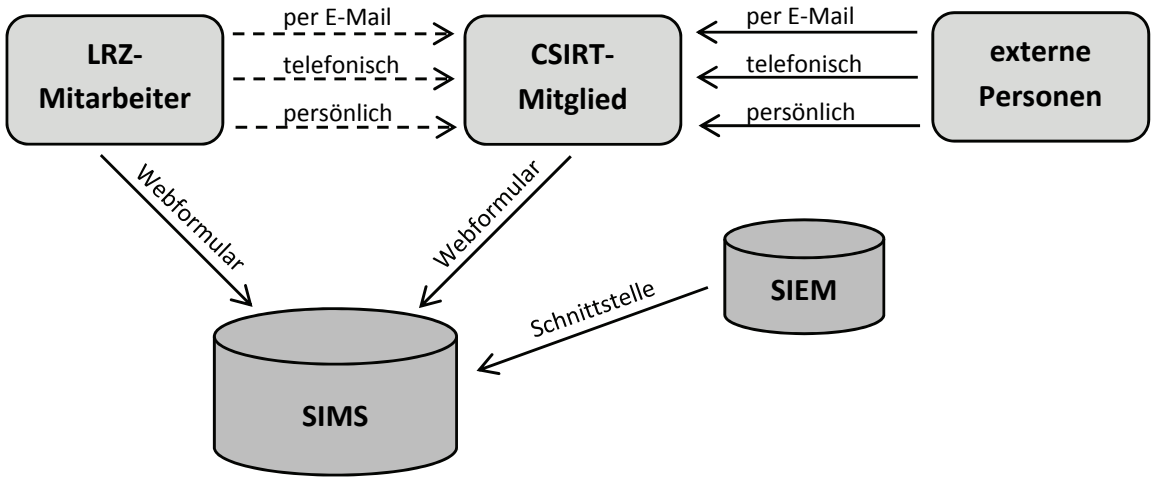

Abbildung 5.3.: Meldewege und Incidenterfassung in der Webanwendung

Um eine rasche Initiierung des SIR-Prozesses zu bewirken, werden Security Incidents – unabhängig vom Meldeweg – mittels Webformular zentral erfasst. Zur Vermeidung von Medienbrüchen muss gemäß `R01` jeder LRZ-Mitarbeiter Security Incidents direkt im System erfassen können. Eine Incidentmeldung soll die folgenden Angaben umfassen, wobei nur die ersten beiden Angaben zwingend erforderlich sind:

- Kurzbezeichnung des Incidents
- Ausführliche Beschreibung des Incidents
- Zeitpunkt bzw. Zeitraum des Incidents
- Betroffener Kunde
- Betriebssystem des betroffenen Systems
- Hostname des betroffenen Systems
- IP-Adresse des betroffenen Systems
- Zeitpunkt der Meldung

Auch wenn die Eingabe sämtlicher Incident Details in ein einziges Textfeld grundsätzlich denkbar wäre, so ergeben sich aus der atomaren Eingabemöglichkeit in die dafür vorgesehenen Felder essenzielle Vorteile. Einerseits unterstützt die explizite Abfrage der speziellen Daten den Vorfallmelder dabei, die ihm bekannten Informationen so umfassend und detailliert wie möglich zu übermitteln. Andererseits stellt ein strukturierter Meldesatz eine Arbeitserleichterung für den CSIRT-Hotliner dar, der sonst die Informationen aus der Fließtextmeldung extrahieren und in das Datenschema überführen müsste.

Sowohl der Kunde als auch das Betriebssystem werden jeweils aus einer Liste mit vorgegebenen Werten ausgewählt. Im Gegensatz zur Eingabe dieser Informationen in Freitextfelder, garantieren Auswahllisten eine einheitliche Codierung von Informationen, sodass eine zuverlässige Filterung anhand dieser Merkmale unkompliziert möglich ist. Folgende Kunden und Betriebssysteme sind momentan für das LRZ relevant:

`Kunden:` LRZ, LMU, TUM, HM, BSB, Hochschulstart, Sonstige
`Betriebssysteme:` Windows, Linux, Mac OS, Mobiles System

Die im Rahmen der Incidenterfassung abgefragten Kontaktdaten umfassen den Vornamen und Nachnamen des Vorfallmelders, dessen E-Mail-Adresse, LRZ-Kennung, Telefon- und Handynummer sowie ggf. Anmerkungen zur Erreichbarkeit oder Vertretung. Da der Großteil dieser Informationen automatisch aus dem LDAP-Verzeichnis übernommen werden kann, sind durch den LRZ-Mitarbeiter nur noch fehlende Angaben wie die Handynummer zu ergänzen. Die LDAP-Integration bedeutet in diesem Zusammenhang einerseits eine Arbeitserleichterung für den Vorfallmelder, stellt andererseits die Aktualität der Daten sicher und reduziert letztlich durch eine manuelle Eingabe bedingte Fehler.

Der Zeitpunkt der Vorfallmeldung wird zum Zwecke späterer Berechnungen dokumentiert. Während ein LRZ-Mitarbeiter den Meldezeitpunkt nicht manipulieren kann, haben CSIRT-Mitglieder eine diesbezügliche Bearbeitungsmöglichkeit, um telefonisch, persönlich oder per E-Mail eingehende Meldungen korrekt datiert (beispielsweise mit dem Zeitstempel der E-Mail) erfassen zu können. Ebenfalls können CSIRT-Mitglieder ihre persönlichen Daten mit den Kontaktdaten des tatsächlichen Vorfallmelders überschreiben.

Als Ergänzung der klassischen Meldewege wird eine in R02 beschriebene, automatisierte Incidenterfassung durch eine SIEM-Lösung grundsätzlich als wünschenswert erachtet. Die Konzeption und Implementierung einer entsprechenden Schnittstelle zur Anbindung von Drittsystemen ist jedoch im Rahmen dieser Bachelorarbeit nicht realisierbar.

Im Rahmen der Incidentmeldung kann der Vorfallmelder beispielsweise Auszüge aus Logfiles oder andere Dateien an das CSIRT übermitteln. Zur technischen Umsetzung dieser in R18 geforderten Funktionalität sind Überlegungen hinsichtlich der Dateiverwaltung zu treffen. Hierbei würde sich als zentraler Speicherort ein vorhandenes Netzlaufwerk eignen, welches im Gegensatz zur Datenhaltung auf dem Webserver nicht nur einen applikationsinternen Dateizugriff, sondern auch eine Dateiverfügbarkeit außerhalb der Software – beispielsweise über den Windows-Explorer – ermöglicht. Dadurch, dass die Dateien nur als Verknüpfungen in der Anwendung repräsentiert sind, erfolgt die Steuerung des Dateizugriffs durch den Fileserver, sodass die Webanwendung keine Dateizugriffsrechte verwalten muss. Zur strukturierten Dateiablage soll die Software bei der Erfassung eines neuen Incidents automatisch einen Unterordner – benannt nach der ID des Security Incidents – auf dem Netzlaufwerk erstellen. Dabei könnte der schreibende Zugriff auf das Netzlaufwerk in Form des Speicherns oder Löschens von Dateien über das Server Message Block (SMB) Protokoll oder das File Transfer Protocol (FTP) erfolgen. Da die Dateiverwaltungsfunktionalität nicht zu den Kernaufgaben der Webanwendung zählt, wird diese Anforderung unter Berücksichtigung ihrer mittleren Gewichtung im Rahmen dieser Arbeit nicht implementiert.

5.3. Prozesskonforme Bearbeitung von Sicherheitsvorfällen

Den wichtigsten Teil der Webanwendung bildet die zweite Funktionsklasse mit der prozesskonformen Bearbeitung von Sicherheitsvorfällen. In den folgenden Abschnitten werden die einzelnen Prozessphasen konzipiert, die ein Information Security Incident durchläuft. Abschließend werden Möglichkeiten zur Integration von Post Incident Reviews betrachtet.

5.3.1. Prozessphase „Initiale Klassifikation“

Unmittelbar nach der Erfassung eines neuen Security Incidents ist das CSIRT gemäß R03 über den Vorfall zu informieren. Eine ausschließlich passive Benachrichtigung des CSIRTs durch Anzeige des neuen Vorfalls auf dem Dashboard der Webanwendung würde erfordern, dass der CSIRT-Hotliner regelmäßig die Webanwendung auf das Vorhandensein eines neuen Security Incidents überprüft. Um die Reaktionszeit für Vorfälle mit sehr hoher Priorität einhalten zu können, müsste die manuelle Prüfung mindestens viertelstündlich erfolgen, was jedoch als nicht praktikabel erscheint. Daher ist zusätzlich zur Dashboard-Darstellung eine aktive Benachrichtigung des CSIRTs via E-Mail, SMS oder RSS-Feed zwingend erforderlich. Während die Benachrichtigung per RSS aus Gründen der Informationssicherheit als bedenklich angesehen wird, können die Informationen per E-Mail und SMS gezielt an definierte Empfänger zugestellt werden. Hinsichtlich des übertragbaren Informationsumfangs bietet die E-Mail entscheidende Vorteile gegenüber der SMS. So können in der E-Mail Details zum Vorfall mitgeteilt werden, die bereits vorab eine Einschätzung der Priorität zulassen. Eine SMS hingegen könnte nur über das Vorhandensein eines neuen Securiy Incidents informieren. Da die E-Mail-Erreichbarkeit des CSIRTs während der Servicezeiten sichergestellt ist, ist eine zusätzliche SMS-Benachrichtigung nicht erforderlich.

Die Webanwendung versendet daher automatisch nach Erfassung eines neuen Vorfalls eine E-Mail mit den bisher bekannten Daten an das CSIRT und fordert den Hotliner auf, die Webanwendung zu öffnen und mit der Incidentbearbeitung zu beginnen. Wie in R03 gefordert, erfolgt im Rahmen der initialen Klassifikation eine Prüfung dahingehend, ob es sich tatsächlich um einen neuen Security Incident, um einen bereits gemeldeten Vorfall oder um einen Information Request handelt. Die Klassifikation erfolgt mittels Auswahlfeldern, wird im Incidentdatensatz gespeichert und ist irreversibel, da sie die Weichen für den weiteren Systemablauf stellt.

Sollte ein Incident Duplikat laut R04 vorliegen, fasst die Webanwendung mehrfach gemeldete Vorfälle zu einem logischen zusammen. Hinsichtlich der Vorgehensweise könnte die Software ein übergeordnetes Elternelement anlegen und die Kindelemente (Original und Duplikate) mit einer Parent-ID versehen. Ein einfacherer Ansatz, der keine separate Verwaltung von Elternelementen erfordert, realisiert die Zusammenführung von Original und Duplikaten auf gleicher Ebene, wobei für jeden Incident eine Duplicate-ID gespeichert wird. Ist diese 0, so handelt es sich um ein Original, ansonsten ist dort die ID des Original-Incidents als Verweis gespeichert. Dadurch können auf unkomplizierte Weise beliebig viele Duplikate mit einem Original verknüpft werden. Die Webanwendung orientiert sich an diesem Modell und schließt den Vorfall nach der Zuordnung des Duplikats zum Haupt-Incident ab. Zudem wird der Vorfallmelder automatisch über das Vorliegen eines Duplikats und den Abschluss des Incidents per E-Mail informiert.

Stellt sich während der initialen Klassifikation heraus, dass es sich um einen Information Request gemäß R05 handelt, entscheidet der Hotliner, ob er die Anfrage selbst beantwortet oder an eine Bearbeitergruppe weiterleitet und den Vorfallmelder über die Weiterleitung informiert. Die Beantwortung oder Weiterleitung eines Information Requests könnte per Telefon, Fax, E-Mail oder Briefpost erfolgen. Es erscheint jedoch sinnvoll, auch den Prozesszweig der Requestbearbeitung in der Software abzubilden und dabei das Kommunikationsmedium E-Mail zu wählen, um eine systematische Dokumentation der Information Requests zu erzielen. Im Falle einer Weiterleitung stellt das System dem Hotliner eine editierbare E-Mail-Vorlage bereit, welche alle gemeldeten Daten enthält. Als Standardempfänger wird das LRZ-Abuse-Team eingetragen, da dieses Team das häufigste Weiterleitungsziel darstellt. Nach der Beantwortung bzw. Weiterleitung der Anfrage wird der Incident geschlossen. Zu Auswertungszwecken verbleiben die Information Requests dauerhaft in der Datenbank, sodass später aus der Anzahl und Art der Requests ggf. informationspolitische Optimierungsmöglichkeiten abgeleitet werden können. Diese können zu einer Reduzierung des Requestaufkommens beitragen, wenn den Anwendern die zuständigen Bearbeitergruppen direkt bekannt sind und häufig gestellte Fragen auf einer FAQ-Seite zentral beantwortet werden.

Im Regelfall wird jedoch das Vorliegen eines neuen Security Incidents festgestellt, woraufhin der CSIRT-Hotliner zunächst den Incidenttyp auswählt. Die in R07 geforderte Unterstützung von Standard Security Incidents beginnt bereits bei der Festlegung des Incidenttyps. Dabei stellt jeder der folgenden *Standard Security Incidents* einen eigenen Incidenttyp dar:

- Externer SSH Scan
- Kompromittierter Webserver
- Kompromittierung eines virtualisierten Servers
- Viren-infiziertes LRZ-System

Trifft keiner der genannten Typen zu, so liegt kein SSI vor und der Vorfall wird als Security Incident vom Typ „Kein Standard SI“ eingestuft. Mit der Auswahl des Incidenttyps ändert sich dessen Status von *Neu* auf *Klassifikation*.

Für Standard Security Incidents ist eine vereinfachte Bearbeitung vorgesehen. Zwar muss auch bei SSIs der formale Bearbeitungsprozess durchlaufen werden, jedoch sind die meisten Formularfelder mit Standardwerten vorbelegt. Da nur noch einzelne Anpassungen der Textvorgaben notwendig sind, wird der Umfang der von Hand einzugebenden Informationen erheblich reduziert, was eine Effizienzsteigerung bewirkt. Anstelle die Standardwerte fest im Programmcode zu hinterlegen, wird es für sinnvoller erachtet, jeden SSI als Template-Datensatz in einer separaten Datenbanktabelle zu speichern. Daraus ergibt sich eine komfortable Pflegemöglichkeit für die SSI-Templates an zentraler Stelle und eine große Flexibilität hinsichtlich des Hinzufügens weiterer SSIs.

Auf Grundlage der zugehörigen Prozessbeschreibung [Fre13] wurde für den SSI „Umgang mit kompromittierten Webservern“ im Rahmen dieser Arbeit ein datenbanktaugliches Template erstellt, welches im Anhang A einzusehen ist.

5.3.2. Prozessphase „Weitere Klassifikation und Priorisierung“

Nach Durchführung der initialen Klassifikation hat der CSIRT-Hotliner die Möglichkeit, die gemeldeten Informationen zu ergänzen. Er legt schließlich fest, ob ein Major Incident zu erwarten ist und ob der Hochschulstartdienst betroffen ist. Bei der Priorisierung des Incidents werden folgende Aspekte betrachtet:

- Standort des Zielsystems
- Standort des Quellsystems
- Betroffene Dienste
- Betroffene Informationen
- Anzahl der betroffenen Systeme

Auf Grundlage dieser Angaben kann das System anhand einer Gewichtungsformel gemäß `R08` automatisch die vorgesehene Auswirkung, Priorität und Reaktionszeit berechnen. Der Anwender kann die Automatik deaktivieren, um diese Werte manuell festzulegen.

Stellt der CSIRT-Hotliner fest, dass die gemeldeten Informationen für die Priorisierung oder die Festlegung von Erstmaßnahmen nicht ausreichend sind, kann er einen Information Request an den Vorfallmelder (`R09`) auslösen. Dieser kann telefonisch oder per E-Mail erfolgen und wird über das Kommunikationsmodul der Webanwendung abgewickelt, welche die Dokumentation dieses Vorgangs sicherstellt.

Im Anschluss an die Festlegung der Erstmaßnahmen stellt der CSIRT-Hotliner gemeinsam mit dem SIC das Security Incident Team zusammen (`R06`). Dieses besteht mindestens aus dem Vorfallmelder, dem Hotliner und dem Security Incident Coordinator. Weitere Teammitglieder (CSIRT-Mitglieder, Administratoren, Abteilungsleiter, Gruppenleiter, Sonstige) werden jeweils unter Zuordnung zu ihrer Funktion innerhalb des Security Incidents hinzugefügt. Statt der manuellen Erfassung der Namen und Kontaktdaten wird auf das LDAP-Verzeichnis als aktuelle Datenbasis zugegriffen. Während bei der Festlegung von Hotliner

und SIC nur die CSIRT-Mitglieder zur Auswahl stehen, kann bei den übrigen Funktionen aus allen LRZ-Mitarbeitern gewählt werden. Darüber hinaus wäre es wünschenswert, auch externe Teammitglieder und deren Kontaktdaten erfassen zu können, wenn beispielsweise ein externer IT-Forensiker hinzugezogen wird. Die Kontaktdaten sämtlicher Teammitglieder sind incidentbezogen editierbar. Zudem können mit Ausnahme des Vorfallmelders alle Teammitglieder gelöscht und neu hinzugefügt werden. Somit kann die Teamzusammenstellung im Laufe der Incidentbearbeitung flexibel angepasst werden.

Nach erfolgter Klassifikation, Priorisierung und Teamzusammenstellung startet der CSIRT-Hotliner die nächste Prozessphase. Unter Angabe des aktuellen – für den Fall einer manuellen Nachtragung editierbaren – Zeitstempels ändert sich der Incidentstatus von *Klassifikation* auf *Analyse* und die Messung der Reaktionszeit wird beendet.

5.3.3. Prozessphase „Analyse“

Während der Analysephase werden zunächst die durchzuführenden Maßnahmen festgelegt und anschließend deren Resultate dokumentiert. Solange der Sicherheitsvorfall nicht vollständig nachvollzogen werden konnte, sind weitere Aktionen zu definieren und nach deren Erledigung die Analyseergebnisse festzuhalten. Da meist mehrere Analyseiterationen erforderlich sind, ist es in dieser Prozessphase laut `R10` besonders wichtig, Ergänzungen und Korrekturen an den bereits eingegebenen Daten vornehmen zu können.

Auch wenn es zunächst sinnvoll erscheint, jede Analysemaßnahme und das zugehörige Ergebnis als separates Wertepaar zu erfassen, so wird von diesem Ansatz Abstand genommen, da sich sowohl die Eingabe als auch die Änderung und die Verwaltung der Daten als unverhältnismäßig komplex darstellen. Als benutzerfreundliche und unkompliziert zu realisierende Alternative kommen zwei Textarea-Felder „Maßnahmen“ und „Analyseergebnisse“ in Betracht, wobei hinzukommende Informationen an beliebiger Stelle in den bestehenden Feldinhalt eingefügt werden können. Da sich diese Dokumentationsart kaum vom bisherigen Bearbeitungslaufzettel unterscheidet, so ist mit einer hohen Akzeptanz durch das CSIRT zu rechnen. Darüber hinaus gestaltet sich bei der Textarea-Variante die Datenhaltung vorteilhaft, sodass jeder Vorfall in einem einzigen Datensatz abgebildet werden kann, was wiederum die weitere Datennutzung – beispielsweise als Incident Export – vereinfacht.

Nach Abschluss der Analysearbeiten startet der CSIRT-Hotliner die nächste Prozessphase. Unter Angabe des aktuellen – für den Fall einer manuellen Nachtragung editierbaren – Zeitstempels ändert sich der Incidentstatus von *Analyse* auf *Lösung* und die Messung der Analysezeit wird beendet.

Basierend auf bereits abgeschlossenen Sicherheitsvorfällen, welche eine signifikante Ähnlichkeit zum aktuellen Incident aufweisen, könnte das System den SIC bei der Festlegung der Analysemaßnahmen unterstützen, indem es automatisiert Informationen anzeigt, die in der Vergangenheit nützlich waren. Diese als Kann-Anforderung `R19` beschriebene Funktionalität automatisierter Handlungsempfehlungen wird als sinnvolle Ergänzung angesehen, die allerdings im Rahmen der ersten Ausbaustufe des Systems nicht realisierbar ist. Dennoch werden an dieser Stelle grundlegende Überlegungen getroffen, die bei einer späteren Umsetzung Berücksichtigung finden könnten:

Eine derartige Wissensmanagementfunktionalität würde voraussetzen, dass das System zunächst alle zum Basisvorfall ähnlichen Security Incidents ermitteln kann. Anschließend müsste es den Erfolg der durchgeführten Analysemaßnahmen aller als ähnlich geltenden Vorfälle

unterscheiden und qualitativ vergleichen. Die Feststellung der Ähnlichkeit kann mittels einer geeigneten Heuristik erfolgen, welche durch den Abgleich der aktuellen Daten mit denen vergangener Vorfälle signifikante Ähnlichkeiten ermittelt. Beispielsweise kann bei Übereinstimmung der Angriffsart in Kombination mit dem Betriebssystem des Zielrechners bereits von einer groben Ähnlichkeit ausgegangen werden. Um schließlich Handlungsempfehlungen für die weitere Vorgehensweise ableiten zu können, muss das System über den Erfolgsgrad der bei ähnlichen Vorfällen bereits durchgeführten Einzelmaßnahmen verfügen. Hierzu könnte entsprechend der ursprünglichen Überlegung jede Maßnahme mit dem zugehörigen Ergebnis ein Wertepaar bilden, dessen Erfolg durch das CSIRT bewertet wird. Durch die Anwendung einer fünfstufigen Skala (sehr hilfreich, etwas hilfreich, neutral, weniger hilfreich, gar nicht hilfreich) liegen dem System differenzierte Informationen zum Erfolgsgrad vor.

Zur weiteren Optimierung könnte neben der Eingabe der Maßnahmenparameter in ein Freitextfeld auch der Typ aus einem Maßnahmenkatalog mittels Dropdown-Feld ausgewählt werden. Dadurch kann das System relative Häufigkeiten der angewendeten Maßnahmentypen ermitteln. Diese ermöglichen zusammen mit dem Erfolgsgrad eine differenzierte Sortierung der Empfehlungen, wobei die Maßnahmen absteigend nach Erfolgsgrad und innerhalb des gleichen Grades absteigend nach relativer Häufigkeit sortiert werden. Dadurch sind die besonders relevanten Empfehlungen für das CSIRT sofort erkennbar.

Abschließend ist anzumerken, dass die Durchführung einer Kosten-Nutzen-Analyse für diese Funktionalität für empfehlenswert erachtet wird, da – abgesehen vom Implementierungsaufwand und der laufenden Pflege des Maßnahmenkatalogs – jede Maßnahme durch das CSIRT einzeln zu typisieren und zu bewerten ist. Wird jedoch bei der Umsetzung auf eine hohe Benutzerfreundlichkeit im Sinne einer nahtlosen Integration in den gewohnten Arbeitsablauf geachtet, so könnte sich der Zusatzaufwand für das CSIRT relativieren.

5.3.4. Prozessphase „Lösung“

Während der Lösungsphase erfolgt zunächst die Wiederherstellung des Regelbetriebs. Für die Dokumentation der durchgeführten Aktivitäten im Rahmen der Wiederherstellung des Regelbetriebs und für die Beschreibung der Lösung sind – in Anlehnung an den bisherigen Bearbeitungslaufzettel – zwei Textarea-Felder vorgesehen. Diese ermöglichen eine flexible Eingabe und Änderung von Informationen und entsprechen somit der Anforderung `R10`.

Es erscheint jedoch sinnvoll, die Incidentlösung nicht nur als Freitext zu erfassen, sondern auch den Lösungstyp zu erfragen. Hierfür bietet sich zunächst ein Dropdown-Auswahlfeld an, welches eine einheitliche Kodierung des Lösungstyps und somit die Filterbarkeit ermöglicht. Alternativ wäre eine Realisierung als Mehrfachauswahlfeld denkbar, um sich bei der Kategorisierung der Lösung nicht auf den zutreffendsten Typ beschränken zu müssen. Erste Überlegungen hinsichtlich geeigneter Lösungstypen haben folgenden Katalog ergeben:

- Neuinstallation des Systems
- Bereinigung des Systems
- Anpassung der Systemkonfiguration
- Einspielung eines Backups
- Installation eines Patches/Softwareupdates

- Sperrung einer Kennung
- Zurücksetzen einer Kennung
- Einrichtung/Anpassung einer Firewallregel

Zur flexiblen Erweiterbarkeit dieser Auflistung werden die Lösungstypen nicht statisch kodiert, sondern in der Typentabelle der Datenbank hinterlegt.

Sobald der Sicherheitsvorfall gelöst und die Lösung dokumentiert wurde, startet der CSIRT-Hotliner die nächste Prozessphase. Unter Angabe des aktuellen – für den Fall einer manuellen Nachtragung editierbaren – Zeitstempels ändert sich der Incidentstatus von *Lösung* auf *Monitoring* und die Messung der Lösungszeit wird beendet.

5.3.5. Prozessphasen „Monitoring“ und „Abschluss“

Zu Beginn der Security Incident Nachsorge wird festgelegt, wie lange und wie häufig die Überwachung des Zielsystems erfolgen soll. Die Angabe der Dauer in Tagen soll ebenso ganzzahlig erfolgen wie die des Monitoringintervalls in Stunden. Darüber hinaus werden die bei jedem Überwachungsvorgang durchzuführenden Aufgaben definiert und die verantwortlichen Personen benannt. Während des Monitoringzeitraums werden alle das überwachte System betreffenden Auffälligkeiten dokumentiert.

Aus den bereits erläuterten Gründen (`R10`) werden – wie in den vorangegangenen Prozessphasen – auch hier zur Erfassung und Bearbeitung von Monitoringaufgaben und erkannten Auffälligkeiten zwei Textarea-Felder verwendet.

Alternativ könnten die Monitoringaktivitäten aufgabenweise erfasst und gespeichert werden, wobei jeder Aufgabe neben der Dauer und dem Intervall ein verantwortlicher Administrator aus dem bereits in der Applikation bestehenden Security Incident Team zugeordnet werden. Das System könnte die zuständigen Personen im vorgegebenen Rhythmus per E-Mail an die Erledigung ihrer Monitoringaufgabe erinnern. Die Durchführungsbestätigung könnte durch Klick auf den Link in der E-Mail erfolgen und würde die automatische Erstellung eines Monitoringprotokolls ermöglichen. Für den Fall neuer Auffälligkeiten könnten diese direkt durch denjenigen Administrator im System erfasst werden, der diese festgestellt hat. Allerdings hat in diesem Fall eine automatisierte Benachrichtigung des LRZ-Hotliners zu erfolgen, um die Art der Auffälligkeit zeitnah prüfen und ggf. die Priorität des Vorfalls anpassen zu können. Bei der in `R11` beschriebenen Erweiterung der Monitoringfunktionen handelt es sich um eine optionale Anforderung, die im Rahmen eines späteren Softwarereleases integriert werden kann.

Ergeben sich während der Monitoringphase keine weiteren Auffälligkeiten, so schließt der CSIRT-Hotliner den Security Incident ab. Unter Angabe des aktuellen – für den Fall einer manuellen Nachtragung editierbaren – Zeitstempels ändert sich der Incidentstatus von *Monitoring* auf *Abgeschlossen* und die Messung der Nachlaufzeit wird beendet. Um einer späteren Datenmanipulation vorzubeugen, sind nach Abschluss des Vorfalls keine Änderungen an Incident Daten mehr möglich.

Nun stellt sich die Frage, wie mit weiteren Auffälligkeiten bezüglich eines bereits abgeschlossenen Sicherheitsvorfalls umzugehen ist. Beispielsweise könnte zur Dokumentation der Auffälligkeiten ein neuer Incident eröffnet werden, der mittels Parent-ID dem ursprünglichen

Vorfall zugeordnet wird. Dieser Ansatz erscheint zwar zunächst technisch einfach umsetzbar, da durch die Instanziierung eines neuen Prozesses keine Weiterführung der ursprünglichen Prozessinstanz realisiert werden muss. Bei näherer Betrachtung ergeben sich jedoch gravierende Nachteile: Da die Informationen zu einem einzigen Vorfall auf mehrere Datensätze verteilt wären, würde es dem CSIRT schwer fallen, den Überblick zu behalten. Darüber hinaus widerspricht dieser Ansatz der Systemlogik, jeden Incident in genau einem Datensatz abzubilden und würde sowohl den Datenexport als auch die statistischen Auswertungen unnötig verkomplizieren. Beispielsweise würde sich die für die Auswertungen benötigte Berechnung der Prozessphasenlaufzeiten unverhältnismäßig komplex gestalten, da die Zeitwerte des ursprünglichen Datensatzes mit den Wiedereröffnungsinstanzen aggregiert werden müssten.

Bei der Betrachtung alternativer Möglichkeiten zum Umgang mit weiteren Auffälligkeiten bei abgeschlossenen Security Incidents erscheint es naheliegend, die bestehende Prozessinstanz durch Ergänzung des vorhandenen Datensatzes fortzuführen. Durch die Wiedereröffnung gemäß `R12` wird der Security Incident in die Analysephase zurückgesetzt. Er durchläuft daraufhin alle folgenden Prozessschritte erneut bis hin zur Schließung des Vorfalls, wobei durch dieses Konstrukt eine mehrfache Wiedereröffnung eines Vorfalls möglich ist. Es ist darauf zu achten, dass die Bearbeitungszeiten aller Prozessiterationen korrekt erfasst und zu den bisherigen Zeitwerten hinzuaddiert werden. Zum Zweck einer späteren Nachvollziehbarkeit wird die Wiedereröffnung als Metainformation durch Setzen des `reopen`-Flags gespeichert. Diese Lösung erlaubt die Beibehaltung des ursprünglichen Statuskonzepts und bietet dem CSIRT im Gegensatz zur ersten Variante ein hohes Maß an Flexibilität und Komfort. Darüber hinaus ergeben sich keinerlei Nachteile hinsichtlich des Datenexports und der statistischen Auswertungen, da wiedereröffnete Vorfälle durch das System nicht gesondert behandelt werden müssen.

Abgeschlossene Sicherheitsvorfälle sollen laut `R14` archiviert werden und als Wissensdatenbank für eine effizientere Bearbeitung ähnlicher Vorfälle dienen. Beim Abschluss eines Sicherheitsvorfalls könnte dessen Datensatz von der Incident Tabelle in eine Archivtabelle verschoben werden. Beim Wiedereröffnen eines archivierten Vorfalls müsste dann der Datensatz wieder in die Incident Tabelle überführt werden. Zudem würden die Filterung von Sicherheitsvorfällen und die Durchführung von statistischen Auswertungen an Komplexität zunehmen, da die hierfür notwendigen Datenbankabfragen dann auf zwei Tabellen durchzuführen und anschließend zu aggregieren wären. Bei Betrachtung der aufgeführten Nachteile erscheint es sinnvoller, die Sicherheitsvorfälle in der Incident Tabelle zu belassen. Die Speicherung von laufenden und archivierten Vorfällen in der gleichen Tabelle hat zwar zur Folge, dass Datenbankabfragen länger dauern. Bei weniger als 500 Security Incidents pro Jahr dürfte jedoch kein Performanceunterschied zu bemerken sein. Da bei dieser Größenordnung kein Bedarf für eine separate Archivierung gegeben ist, werden die Sicherheitsvorfälle direkt in der Incident Tabelle archiviert.

Nach Abschluss des SIR-Prozesses liegen zu jedem Sicherheitsvorfall zahlreiche Metainformationen vor, allerdings nur implizit in Textform. Da es aber wünschenswert wäre, die Metadaten zum Zwecke von Filterung und Auswertung explizit zu nutzen, so könnte dies eine Tagging-Funktion (`R25`) ermöglichen. Dabei können jedem Security Incident beliebig viele Tags zugewiesen werden. Die Verwendung gleicher Tags bewirkt eine flexible Gruppierung ähnlicher Vorfälle. Dies erleichtert zwar das themenspezifische Auffinden aller zugeordneten Sicherheitsvorfälle, wird aber nicht als Kernaufgabe der Software betrachtet und daher in der ersten Ausbaustufe nicht weiter berücksichtigt. Da sich das Tagging von Incidents

auch positiv auf die Durchführung von Ähnlichkeitsvergleichen auswirken würde, wäre eine gemeinsame Umsetzung zusammen mit den zuvor beschriebenen Funktionen im Bereich Wissensmanagement gut vorstellbar.

5.3.6. Post Incident Review

Zur kontinuierlichen Verbesserung des SIR-Prozesses ist die Durchführung eines Post Incident Reviews vorgesehen. Nach Abschluss eines Sicherheitsvorfalls werden in einer retrospektiven Betrachtung sowohl positive als auch negative Erfahrungswerte gesammelt. Daraus können konkrete Optimierungsvorschläge für einzelne Prozessschritte abgeleitet werden. Das System ermöglicht die incidentbezogene Erfassung der Reviewergebnisse gemäß `R13`. Hierfür werden drei Textarea-Felder bereitgestellt, die eine flexible Dateneingabe ermöglichen. Darüber hinaus wäre eine automatische Zusammenstellung aller Optimierungsvorschläge für einen definierbaren Zeitraum wünschenswert. Diese Auflistung könnte als Input für den jährlichen Review des SIR-Prozesses dienen.

5.4. Vielfältige Nutzung des Datenbestands

Das bei der Erfassung und Bearbeitung von Information Security Incidents entstandene Datenmaterial bietet vielfältige Nutzungsmöglichkeiten (dritte Funktionsklasse). Diese stellen im Vergleich zur bisherigen Situation einen erheblichen Mehrwert dar, da die Microsoft Word basierte Datenhaltung keine flexible Nutzung des Datenbestands zulässt. Im Folgenden werden die Durchsuchbarkeit und Filterung des Datenbestands, die Erstellung von statistischen Auswertungen und verschiedene Möglichkeiten des Datenexports betrachtet.

5.4.1. Filterung und Suche

Ein entscheidender Vorteil der zentralen Speicherung der Sicherheitsvorfälle in einer Datenbank ist die einfache und flexible Selektion aller Sicherheitsvorfälle, die mit den eingegebenen Parametern übereinstimmen. Durch Anwendung der Filteroperationen können einerseits ähnliche Security Incidents ermittelt werden, die die Bearbeitung des aktuellen Vorfalls erleichtern. Andererseits können Informationen abgeleitet werden, die im Rahmen einer Trendanalyse oder bei der Optimierung des SIR-Prozesses von Bedeutung sind. Filtert man beispielsweise alle Sicherheitsvorfälle heraus, die keinem SSI entsprechen, so können durch eine geschickte Verfeinerung des Filters neue Incident Muster identifiziert werden. Auf Basis dieser gemeinsamen Incident Charakteristika können neue Standard Security Incidents definiert werden, die wegen der Template-Unterstützung künftig effizienter bearbeitet werden können.

Für eine grundlegende Filterung gemäß `R24` können Zeitraum, Kunde und Klassifikation (Security Incident, Duplikat, Information Request) festgelegt werden, um beispielsweise alle Sicherheitsvorfälle eines bestimmen Kunden innerhalb des definierten Zeitraums abfragen zu können. Zur Verfeinerung der Filterung werden folgende Parameter als sinnvoll erachtet:

- Incidenttyp
- Priorität des Incidents
- Betriebssystem des betroffenen Zielsystems

- Art der Incidentlösung
- Aktueller Status des Vorfalls
- Status bei Abschluss des Vorfalls

Eine noch speziellere Incident Selektion ist auf Basis der für die Ermittlung von Auswirkung, Priorität und Reaktionszeit zu Rate gezogenen Parameter möglich. Hierbei werden der Standort von Ziel- und Quellsystem, die Kritikalität der Dienste, die Vertraulichkeit der Informationen sowie die Anzahl der Zielsysteme berücksichtigt.

In der Standardeinstellung werden alle Vorfälle des aktuellen Jahres angezeigt. Für die weiteren Filterparameter gilt jeweils eine UND-Verknüpfung, sodass beispielsweise alle Sicherheitsvorfälle mit hoher Priorität ermittelt werden können, die durch Einspielung eines Backups gelöst wurden. Auch können alle Vorfälle eines Kunden abgefragt werden, die – obwohl streng vertrauliche Informationen betroffen waren – abgebrochen wurden. Die Filtereinstellungen werden für die Dauer der Benutzersitzung gespeichert, damit der Anwender auch nach Verlassen der Filterungsfunktion wieder auf Basis der letzten Konfiguration weiterarbeiten kann. Selbstverständlich ist es auch möglich, den Filter zurückzusetzen.

Ein Filterkonfigurator würde das Maximum an Flexibilität bieten, indem er feingranulare Filterregeln und deren Gruppierung zulässt und schließlich eine UND-/ODER-Verknüpfung der Filtergruppen ermöglicht. Da die Bedienung eines Filterkonfigurators im Gegensatz zum oben beschriebenen Filtermodell erheblich komplexer ist, einen höheren Zeitaufwand für das CSIRT bedeutet und derart spezielle Abfragen nur selten erforderlich sind, beschränkt sich die erste Version der Software auf den Standardfilter. Sollte sich bei der alltäglichen Nutzung der Software jedoch herausstellen, dass der implementierte Filter zu oft an seine Grenzen stößt, könnte der Filterkonfigurator als zusätzliche Variante für Spezialfälle nachgerüstet werden. Darüber hinaus könnte die bereits beschriebene Tagging-Funktion gemäß `R25` eine noch flexiblere Filterung von Security Incidents ermöglichen.

Die Ergebnisse der Filterung werden aus Gründen der Übersichtlichkeit tabellarisch dargestellt und umfassen für jeden Vorfall dessen ID, Bezeichnung, Incidenttyp, Priorität, Status, Kunde, Host, IP-Adresse, Betriebssystem und Meldedatum. Für eine optimale Bedienbarkeit kann die Ergebnistabelle spaltenweise auf- und absteigend sortiert werden. Beim Klick auf einen Eintrag wird die Statusseite des Vorfalls angezeigt. Um die selektierten Daten komfortabel außerhalb des Systems weiterverarbeiten zu können, können diese in mehreren Dateiformaten exportiert werden.

Neben der Filterung von Sicherheitsvorfällen wird in `R24` auch eine stichwortbasierte Suchfunktion gefordert. Diese könnte sich als Volltextsuche über alle Felder der Incident Tabelle erstrecken. Die Darstellung der Suchergebnisse sollte auf übersichtliche Weise erfolgen und die gefundenen Textstellen farblich hervorheben. Zur Festlegung der Ergebnisreihenfolge ist die Relevanz aller Ergebnisse anhand eines geeigneten Algorithmus zu ermitteln.

Alternativ könnten Such- und Filterfunktion kombiniert werden, sodass der Anwender im ersten Schritt die Grundmenge der zu durchsuchenden Sicherheitsvorfälle durch geschickte Wahl der Filterparameter einschränkt und im zweiten Schritt die Suchoperation auf der Filtermenge durchführt. Durch die vorherige Eingrenzung der Datensätze steigt die Relevanz der Suchergebnisse, sodass im Vergleich zur Volltextsuche auf der Gesamtmenge von einer höheren Qualität der Treffer auszugehen ist. Angesichts dieses Vorteils setzt die Applikation auf eine derartige Kombination von Filter- und Suchfunktion.

5.4.2. Auswertungen

Wie in R28 gefordert, stellt das System einfache statistische Auswertungen zur Verfügung. Diese dienen einerseits der Überprüfung, ob die in den SLAs zugesicherten Reaktionszeiten auch tatsächlich eingehalten werden. Andererseits lassen sich durch die Interpretation der Key-Performance-Indikatoren (KPI) Rückschlüsse auf den Leistungsgrad des SIR-Prozesses ziehen.

Auch wenn eine grafische Informationsvisualisierung in Form von Balken- und Kreisdiagrammen grundsätzlich wünschenswert wäre, um Zahlenverhältnisse auf einen Blick optisch wahrnehmen zu können, beschränkt sich die erste Softwareversion auf eine übersichtliche Darstellung in Tabellenform.

Der Auswertungszeitraum kann tagesgenau festgelegt werden, wobei das aktuelle Jahr als Standard voreingestellt ist. Darüber hinaus kann der Kunde mittels Dropdown-Feld ausgewählt werden. Standardmäßig ist kein Kunde selektiert, sodass die Statistik sämtliche im System erfassten Vorfälle berücksichtigt.

Die Auswertung setzt sich aus vier Teilen zusammen. Die erste Tabelle gibt einen Überblick über die Anzahl der erfolgreichen und abgebrochenen Sicherheitsvorfälle, der Duplikate sowie der beantworteten und weitergeleiteten Information Requests für jeden einzelnen Kunden. Daher wird diese Auswertung nur dann eingeblendet, wenn kein spezieller Kunde ausgewählt wurde. Im zweiten Teil werden Vorfälle nach Klassifikation und Incidenttyp aufgeschlüsselt. Darüber hinaus werden die durchschnittlichen Laufzeiten aller Prozessphasen dargestellt, wobei die Reaktionszeit detailliert betrachtet wird:

- Absolute Anzahl der Vorfälle, bei denen die vorgegebene Reaktionszeit überschritten wurde und prozentualer Anteil dieser an der Gesamtmenge
- Dauer der durchschnittlichen Reaktionszeitüberschreitung
- Dauer der kleinsten Reaktionszeitüberschreitung
- Dauer der größten Reaktionszeitüberschreitung

Der dritte Teil zeigt für jede Incidentpriorität die Anzahl der Vorfälle, deren durchschnittliche Reaktionszeit sowie die absolute und prozentuale Abweichung von der für die jeweilige Priorität vorgegebenen Reaktionszeit auf. In der Aufstellung sind außerdem die durchschnittliche Analyse-, Lösungs- und Nachlaufzeit enthalten. Anhand dieser Tabelle kann geprüft werden, ob die in den SLAs vereinbarten Reaktionszeiten tatsächlich realisierbar sind. Darüber hinaus ist feststellbar, ob die unterschiedlichen Prioritäten bei der Bearbeitung berücksichtigt werden. Würde die Prioritätsstufe „mittel“ eine kürzere durchschnittliche Reaktionszeit aufweisen als die Priorität „hoch“, so wäre zu folgern, dass die Priorisierung keinen Einfluss auf die Bearbeitungsreihenfolge hat und somit ihren Zweck verfehlt.

Im vierten Teil richtet sich der Fokus auf die Incidenttypen. Hierbei wird dargestellt, wie viele Vorfälle auf die einzelnen Incidenttypen entfallen und wie lange die einzelnen Prozessphasen durchschnittlich dauern. Die Bearbeitungszeiten der Standard Security Incidents sollten wegen der Verwendung von Templates im Vergleich zu regulären Vorfällen deutlich niedriger ausfallen. Falls kein erheblicher Unterschied feststellbar ist, so empfiehlt sich eine Optimierung des jeweiligen SSI-Templates.

Da anhand der relativen Veränderungen von Kennzahlen im Zeitverlauf Trends leichter zu identifizieren sind, wäre es wünschenswert, die Werte des gewählten Zeitraums in Relation zu einem Referenzzeitraum zu setzen, sodass prozentuale Veränderungen sichtbar werden.

Im System erstellte Auswertungen lassen sich im PDF-Format gemäß R29 exportieren. Die auf diese Weise generierten Reports können beispielsweise zum Nachweis der in den SLAs zugesicherten Leistungsparameter gegenüber den Kunden verwendet werden und der professionellen Berichterstattung gegenüber der LRZ-Leitung dienen. Eine Automatisierung der Reportgenerierung mit regelmäßiger E-Mail-Zustellung an das CSIRT gemäß R30 würde das Security Incident Reporting perfektionieren und kann zu einem späteren Zeitpunkt unkompliziert nachgerüstet werden.

5.4.3. Exportfunktionen

Die Tatsache, dass der Systemzugriff auf LRZ-Mitarbeiter und das LRZ-CSIRT beschränkt ist, erschwert die Zusammenarbeit mit Kunden sowie externen Kollegen. Daher ist es an einigen Stellen im System notwendig, Informationen in verschiedenen Dateiformaten exportieren zu können, um die flexible und systemunabhängige Weiterverarbeitung des Datenmaterials zu erleichtern.

Als Dateiformate kommen diejenigen in Frage, die einen hohen Verbreitungsgrad und eine hohe Interoperabilität mit anderen Systemen aufweisen sowie plattform- und implementierungsunabhängig sind. Während sich das PDF-Format am besten für druckfähig formatierte Reports eignet, sind kommagetrennte CSV-Dateien für den Austausch einfach strukturierter Daten weit verbreitet und bieten sich besonders zur flexiblen Weiterverarbeitung der Informationen an. Beispielsweise könnten Security Incidents mittels CSV-Datei in andere Softwaresysteme importiert werden. Darüber hinaus wäre es denkbar, die CSV-Datei als Datenquelle für die Erstellung von Microsoft Word Seriendrucken oder Microsoft Excel Diagrammen zu verwenden. Das insbesondere im Internet weit verbreitete XML-Dateiformat ermöglicht eine Darstellung von hierarchisch strukturierten Daten und lässt daher einen komplexeren Dateiaufbau zu, als dies bei CSV-Dateien der Fall ist. Beispielsweise ließen sich sämtliche zu einem Incident erfassten Daten inklusive der Kontaktdaten aller Teammitglieder und der Historyeinträge in einer einzigen Datei abbilden. Da jedoch kein einheitliches XML-Schema für Sicherheitsvorfälle existiert, wird das Datenaustauschformat XML zum jetzigen Zeitpunkt als nicht relevant betrachtet.

Der vermutlich häufigste Anwendungsfall besteht im Export der im System zu einem konkreten Sicherheitsvorfall gespeicherten Daten. Dabei erscheinen für den Einzelexport eines Security Incidents gemäß R26 folgende Varianten als sinnvoll:

- **Export des Incidentdatensatzes inklusive der Teamdaten als übersichtlich formatierte PDF-Datei.** Der Aufbau der PDF-Datei orientiert sich an der Gliederung der Incidentstatusseite.
- **Export des Incidentdatensatzes im CSV-Format.** Die kommagetrennte Textdatei beinhaltet in der ersten Zeile die Namen der Datenfelder und in der zweiten Zeile den Incidentdatensatz.
- **Export der Teamdaten im CSV-Format.** Die exportierte Datei enthält nach der Kopfzeile die Kontaktdaten aller an einem Security Incident beteiligten Personen, wobei je Zeile ein Datensatz abgebildet wird.
- **Export der Incident History im CSV-Format.** Um den exakten Verlauf eines Sicherheitsvorfalls außerhalb des Systems nachvollziehen oder besonders aufbereiten zu können, enthält diese Exportdatei nach der Kopfzeile alle Einträge des Incident Verlaufsprotokolls.

- **Flexibler Export als Plaintext für den E-Mail-Versand.** Die incidentbezogenen Daten können auch als Plaintext per E-Mail exportiert werden. Dabei wird zunächst der zu versendende Informationsumfang selektiert, woraufhin im zweiten Schritt das Exportresultat vor dem Versand editiert werden kann, um beispielsweise vertrauliche Informationen, die nicht für den Empfänger dieser E-Mail bestimmt sind, zu entfernen. Da der Versand per E-Mail-Versand optional ist, kann die Funktion auch zum alleinigen Plaintext-Export verwendet werden.

Über den soeben beschriebenen Export eines einzelnen Sicherheitsvorfalls hinaus ist im Bereich „Suchen und Filtern“ gemäß `R27` der Massenexport von Security Incidents gefragt. Auch wenn der Massenexport grundsätzlich durch eine Vielzahl manueller Einzelexporte und eine anschließende Zusammenführung möglich wäre, so trägt die technische Automatisierung des Vorgangs zu einer effizienten Arbeitsweise des CSIRTs bei. Die gefilterten Ergebnisse können in einem Vorgang exportiert werden, wobei sich auch hier die Dateiformate PDF und CSV anbieten. Die inhaltliche Ausgestaltung der Exportdateien erfolgt analog zum Einzelexport.

Schließlich können auch die im System erstellten Auswertungen gemäß `R29` exportiert werden. Hierfür bietet sich das PDF-Format an, sodass auf unkomplizierte Weise ein versandfertiger Jahresreport für einen Kunden generiert werden kann. Der inhaltliche Aufbau der PDF-Datei orientiert sich an der Bildschirmdarstellung der Auswertung. Auf allen im System erstellten PDF-Reports ist gemäß `R31` das LRZ-Logo als Corporate Design Element in der Kopfzeile abgebildet, wodurch ein professioneller Eindruck vermittelt wird. Zudem ist auf den ersten Blick erkennbar, dass es sich um ein Dokument des LRZ handelt.

5.5. Datenmodell

Nach Untersuchung der zu verwaltenden Informationen wird unter Berücksichtigung von `R36` eine strukturierte Datenhaltung mittels relationaler Datenbank als notwendig erachtet. Das erarbeitete Datenmodell umfasst insgesamt neun Relationen, wobei zwischen incidentbezogenen Tabellen und Systemtabellen unterschieden wird:

`Incidentbezogene Tabellen:` incidents, contacts, communication, history
`Systemtabellen:` ssiTpl, users, types, settings, syslog

Die Speicherung der Nutzdaten erfolgt in vier Tabellen. Dabei wird jeder Security Incident als einzelner Datensatz in der Haupttabelle `incidents` abgebildet. Die Ergebnisse der nach Abschluss von Security Incidents durchgeführten Reviews werden ebenfalls in der `incidents`-Tabelle protokolliert, da jeder Review eindeutig einem Sicherheitsvorfall zuzuordnen ist. Die folgenden Tabellen referenzieren diese zentralen Security Incidentdatensätze durch Fremdschlüsselbeziehungen. Die Kontakt- und Rolleninformationen aller Prozessbeteiligten werden in einer separaten Personentabelle `contacts` geführt, da diese eine effiziente und flexible Datenhaltung ermöglicht, sodass beispielsweise nicht nur ein einziger Systemadministrator, sondern beliebig viele Administratoren einem Incident zugeordnet werden können. Diverse Kommunikationsvorgänge im Rahmen der Bearbeitung eines Security Incidents werden in der `communication` Tabelle abgelegt. Zur Nachvollziehbarkeit der Anwenderaktivitäten werden sämtliche incidentbezogenen Vorgänge und Datenänderungen einzeln in der `history` Tabelle dokumentiert.

Über die soeben aufgezählten incidentbezogenen Tabellen hinaus werden fünf Systemtabellen mit Hilfsdaten vorgesehen. Zur schnelleren Erledigung von Standard Security Incidents

stellt die Tabelle `ssiTpl` SSI-Templates in Form von Datensätzen mit Textbausteinen für die verschiedenen Arten von SSIs zur Verfügung, die der automatischen Vorbelegung von Formularfeldern dienen. Die Verwaltung der Benutzerdaten erfolgt incidentunabhängig in der Tabelle `users`. Während die Tabelle `types` die Typausprägungen verschiedener Kategorien wie beispielsweise Incidenttypen oder Betriebssysteme bereitstellt, beinhaltet die `settings` Tabelle variable Systemeinstellungen in Form von Schlüssel-Wert-Paaren. Benutzer- und Systemereignisse werden zu Debuggingzwecken in der `syslog` Tabelle aufgezeichnet.

Das in Abbildung 5.4 visualisierte relationale Datenbankschema wird im Folgenden detailliert dargestellt. Die Auflistung enthält neben dem Feldnamen auch den Datentyp und eine kurze Beschreibung des Feldinhalts. Primärschlüssel sind als PK (Primary Key), Fremdschlüssel als FK (Foreign Key) gekennzeichnet.

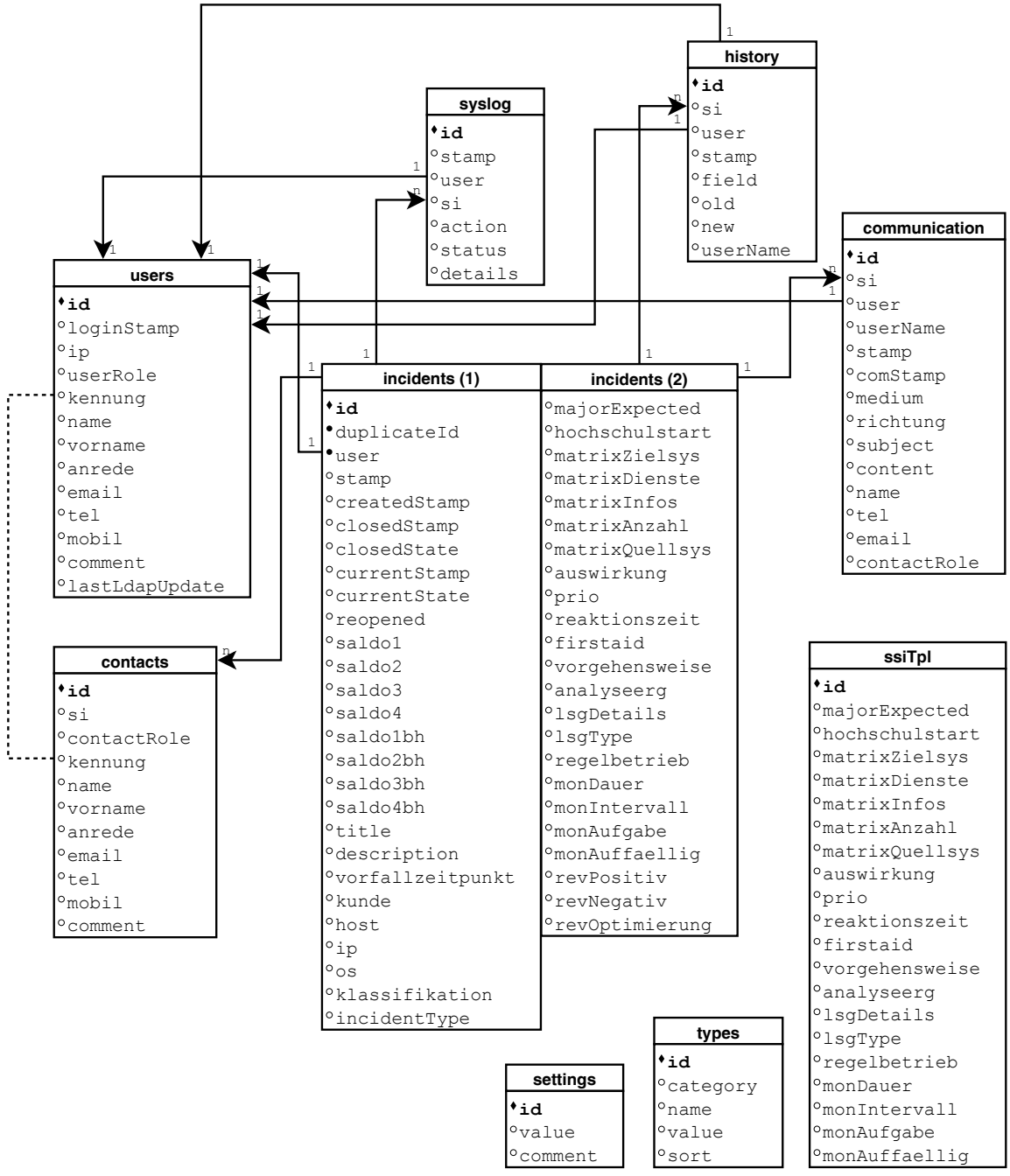

Abbildung 5.4.: Datenmodell der Webanwendung als relationales Datenbankschema

5.5.1. incidents

Bei der `incidents` Relation handelt es sich um die Mastertabelle, welche stets die aktuellen Daten aller Vorfälle inklusive diverser Metainformationen sowie der Ergebnisse der Post Incident Reviews beinhaltet. Dabei wird jeder gemeldete Vorfall unabhängig von seiner Klassifikation als Datensatz mit 50 Attributen abgebildet.

Bezeichner	**Datentyp**	**Kommentar**
id	MEDIUMINT (PK)	ID des Incidents
duplicateId	MEDIUMINT (FK)	falls Duplikat: ID des Haupt-Incidents
user	INT (FK)	ID des Benutzers, der den Incident angelegt hat
stamp	TIMESTAMP	Zeitstempel der Datensatzerfassung
createdStamp	TIMESTAMP	Zeitstempel der Incidentmeldung
closedStamp	TIMESTAMP	Zeitstempel beim Schließen des Incidents
closedState	ENUM	Abschluss-Status des Security Incidents
currentStamp	TIMESTAMP	Zeitstempel des Eintritts in den neuen Status
currentState	ENUM	Aktueller Status des Security Incidents
reopened	TINYINT (Bool)	Flag gesetzt, wenn der Incident wiedereröffnet wurde
saldo1	INT	Tatsächliche Reaktionszeit in Minuten
saldo2	INT	Tatsächliche Anlaufzeit in Minuten
saldo3	INT	Tatsächliche Bearbeitungszeit in Minuten
saldo4	INT	Tatsächliche Nachlaufzeit in Minuten
saldo1bh	INT	Reaktionszeit in Minuten während der Business Hours
saldo2bh	INT	Anlaufzeit in Minuten während der Business Hours
saldo3bh	INT	Bearbeitungszeit in Minuten während der Business Hours
saldo4bh	INT	Nachlaufzeit in Minuten während der Business Hours
title	VARCHAR(255)	Kurzbezeichnung des Incidents
description	TEXT	gemeldete Details
vorfallzeitpunkt	VARCHAR(255)	Zeitpunkt/Zeitraum des Auftretens
kunde	FLEX-ENUM	Betroffener Kunde
host	VARCHAR(160)	Hostname des betroffenen Systems
ip	VARCHAR(39)	IP-Adresse des betroffenen Systems
os	FLEX-ENUM	Betriebssystem des betroffenen Systems
klassifikation	ENUM	INCIDENT, DUPLICATE, REQUEST
incidentType	FLEX-ENUM	Kategorisierung des Incidents
majorExpected	TINYINT (Bool)	Wird ein Major Incident erwartet?
hochschulstart	TINYINT (Bool)	Ist ein Hochschulstart-Dienst betroffen?
matrixZielsys	TINYINT [0-3]	Kriterien zur Bestimmung der Auswirkung
matrixDienste	TINYINT [0-3]	s.o.

matrixInfos	TINYINT [0-3]	s.o.
matrixAnzahl	TINYINT [0-3]	s.o.
matrixQuellsys	TINYINT [0-3]	s.o.
auswirkung	TINYINT [0-4]	NONE, Gering, Mittel, Groß, Sehr groß
prio	TINYINT [0-4]	NONE, Niedrig, Mittel, Hoch, Sehr hoch
reaktionszeit	TINYINT [0-4]	NONE, 10 Std., 4 Std., 2 Std., 15 Min.
firstaid	TEXT	Beschreibung der Erstmaßnahmen
vorgehensweise	TEXT	Beschreibung der Vorgehensweise
analyseerg	TEXT	Beschreibung der Analyseergebnisse
lsgDetails	TEXT	Beschreibung der Incidentlösung
lsgType	FLEX-ENUM	Kategorisierung der Lösung
regelbetrieb	TEXT	Beschreibung der Wiederherstellung des Regelbetriebs
monDauer	TINYINT	Dauer des Monitorings (Tage)
monIntervall	TINYINT	Frequenz der Monitoringaktivitäten (Stunden)
monAufgabe	TEXT	Beschreibung des Monitoringumfangs und der Verantwortlichkeiten
monAuffaellig	TEXT	Dokumentation von Auffälligkeiten während des Monitorings
revPositiv	TEXT	Review: Was hat gut funktioniert?
revNegativ	TEXT	Review: Was hat nicht (gut) funktioniert?
revOptimierung	TEXT	Review: Welche Optimierungsvorschläge können abgeleitet werden?

Tabelle 5.1.: Datenbanktabelle `incidents`

5.5.2. contacts

Die Kontaktdaten aller Prozessbeteiligten werden unter Zuordnung zu einem Security Incident und Zuweisung der Benutzerrolle innerhalb des Security Incidents in der `contacts` Relation gespeichert. Dadurch kann eine Person zahlreichen Vorfällen zugeordnet werden und darüber hinaus mehrere Rollen innerhalb eines Security Incidents wahrnehmen.

Bezeichner	**Datentyp**	**Kommentar**
id	INT (PK)	Item-ID
si	MEDIUMINT (FK)	ID des Security Incidents
contactRole	FLEX-ENUM	Rolle des Benutzers innerhalb des Security Incidents (melder, hotl, sic, csirt, admin, abtl, grul, sonst)
kennung	VARCHAR(40)	LDAP-Kennung des Benutzers
name	VARCHAR(160)	Name des Benutzers
vorname	VARCHAR(160)	Vorname des Benutzers
anrede	ENUM	Anrede des Benutzers (Frau, Herr)

email	VARCHAR(160)	Mailadresse des Benutzers
tel	VARCHAR(160)	Telefonnummer des Benutzers
mobil	VARCHAR(160)	Handynummer des Benutzers
comment	VARCHAR(255)	Anmerkungen zum Benutzer

Tabelle 5.2.: Datenbanktabelle `contacts`

5.5.3. communication

Während des Prozessverlaufs findet ein kommunikativer Austausch von Informationen zwischen den Prozessbeteiligten statt. Zu Dokumentationszwecken werden in der `communication` Relation sowohl versendete als auch empfangene E-Mails, Notizen zu ein- und ausgehenden Telefonaten und persönliche Absprachen gespeichert.

Bezeichner	**Datentyp**	**Kommentar**
id	INT (PK)	Item-ID
si	MEDIUMINT (FK)	Incident-ID
user	INT (FK)	ID des agierenden Benutzers
userName	VARCHAR(160)	Name des agierenden Benutzers
stamp	TIMESTAMP	Zeitstempel der Datensatzerfassung
comStamp	TIMESTAMP	Zeitstempel des Kommunikationsereignisses
medium	ENUM	TEL, MAIL, PERS
richtung	ENUM	IN, OUT, NONE
subject	VARCHAR(255)	Betreff
content	TEXT	Kommunikationsinhalt
name	VARCHAR(80)	Name des Ansprechpartners
tel	VARCHAR(30)	Telefonnummer des Ansprechpartners
email	VARCHAR(160)	Mailadresse des Ansprechpartners
contactRole	FLEX-ENUM	Rolle/Funktion des Ansprechpartners

Tabelle 5.3.: Datenbanktabelle `communication`

5.5.4. history

Die `history` Relation umfasst sämtliche Änderungen an Incidentdatensätzen sowie incidentbezogene Ereignisse wie das Hinzufügen einer Person zum Security Incident Team. Dabei wird jede Änderung auf Datenfeldebene atomar als einzelner Datensatz mit dem alten und dem neuen Wert sowie dem agierenden Nutzer gespeichert.

Bezeichner	**Datentyp**	**Kommentar**
id	INT (PK)	Item-ID
si	MEDIUMINT (FK)	ID des Security Incidents
user	INT (FK)	ID des agierenden Benutzers
stamp	TIMESTAMP	Zeitstempel der Änderung

field	VARCHAR(40)	Geändertes Feld
old	TEXT	Alter Wert
new	TEXT	Neuer Wert
userName	TEXT	Name des agierenden Benutzers

Tabelle 5.4.: Datenbanktabelle `history`

5.5.5. ssiTpl

Um eine effiziente Bearbeitung von Standard Security Incidents zu ermöglichen, wird in der `ssiTpl` Relation für jeden SSI ein Template hinterlegt. Dieses beinhaltet Standardwerte zur Vorbelegung von Formularfeldern.

Bezeichner	**Datentyp**	**Kommentar**
id	VARCHAR(12) (PK)	ID des Standard Security Incidents
majorExpected	TINYINT (Bool)	Wird ein Major Incident erwartet?
hochschulstart	TINYINT (Bool)	Ist ein Hochschulstart-Dienst betroffen?
matrixZielsys	TINYINT [0-3]	Kriterien zur Bestimmung der Auswirkung
matrixDienste	TINYINT [0-3]	s.o.
matrixInfos	TINYINT [0-3]	s.o.
matrixAnzahl	TINYINT [0-3]	s.o.
matrixQuellsys	TINYINT [0-3]	s.o.
auswirkung	TINYINT [0-4]	NONE, Gering, Mittel, Groß, Sehr groß
prio	TINYINT [0-4]	NONE, Niedrig, Mittel, Hoch, Sehr hoch
reaktionszeit	TINYINT [0-4]	NONE, 10 Std., 4 Std., 2 Std., 15 Min.
firstaid	TEXT	Beschreibung der Erstmaßnahmen
vorgehensweise	TEXT	Beschreibung der Vorgehensweise
analyseerg	TEXT	Beschreibung der Analyseergebnisse
lsgDetails	TEXT	Beschreibung der Incidentlösung
lsgType	ENUM	Kategorisierung der Lösung
regelbetrieb	TEXT	Beschreibung der Wiederherstellung des Regelbetriebs
monDauer	TINYINT	Dauer des Monitorings (Tage)
monIntervall	TINYINT	Frequenz der Monitoringaktivitäten (Stunden)
monAufgabe	TEXT	Beschreibung des Monitoringumfangs und der Verantwortlichkeiten
monAuffaellig	TEXT	Dokumentation von Auffälligkeiten während des Monitorings

Tabelle 5.5.: Datenbanktabelle `ssiTpl`

5.5.6. users

Alle Nutzer der Webanwendung werden in der `users` Relation als Datensatz gespeichert. Dieser enthält neben den Kontaktdaten auch Informationen zum letzten Login sowie die gruppenabhängige und für die Rechtevergabe relevante Benutzerrolle. Beim Hinzufügen von Personen zu einem Security Incident werden in der Relation `contacts` Instanzen der Benutzer aus der Relation `users` erzeugt. Eine bidirektionale Zuordnung eines `contacts` zu einem `user` und umgekehrt ist durch die LDAP-Kennung als gemeinsames Identifikationsmerkmal möglich.

Bezeichner	**Datentyp**	**Kommentar**
id	INT (PK)	ID des Benutzers
loginStamp	TIMESTAMP	Zeitpunkt des letzten Logins
ip	VARCHAR(40)	IP-Adresse des Benutzers beim letzten Login
userRole	ENUM	Rolle des Benutzers innerhalb der Webanwendung (ROOT, CSIRT, USER)
kennung	VARCHAR(40)	LDAP-Kennung des Benutzers
name	VARCHAR(160)	Name des Benutzers
vorname	VARCHAR(160)	Vorname des Benutzers
anrede	ENUM	Anrede des Benutzers (Frau, Herr)
email	VARCHAR(160)	Mailadresse des Benutzers
tel	VARCHAR(160)	Telefonnummer des Benutzers
mobil	VARCHAR(160)	Handynummer des Benutzers
comment	VARCHAR(255)	Anmerkungen zum Benutzer
lastLdapUpdate	DATE	Datum des letzten LDAP-Abgleichs dieses Datensatzes

Tabelle 5.6.: Datenbanktabelle `users`

5.5.7. types

In der `types` Relation werden die Ausprägungen für verschiedene Kategorien wie beispielsweise Incidenttypen, Lösungstypen, Kunden und Betriebssysteme gespeichert. In anderen Relationen kommen Attribute vom Typ `FLEX-ENUM` vor. Dabei handelt es sich um flexible `ENUM`-Felder vom eigentlichen Datentyp `VARCHAR(12)`, deren Ausprägungen in dieser Tabelle gespeichert sind.

Bezeichner	**Datentyp**	**Kommentar**
id	MEDIUMINT (PK)	Item-ID
category	VARCHAR(40)	Kategorie
value	VARCHAR(255)	Wert/Ausprägung
order	TINYINT	Reihenfolge der Werte einer Kategorie

Tabelle 5.7.: Datenbanktabelle `types`

5.5.8. settings

Die `settings` Relation beinhaltet systemrelevante Einstellungen als Schlüssel-Wert-Paare.

Bezeichner	Datentyp	Kommentar
id	VARCHAR(30) (PK)	Setting-ID
value	VARCHAR(255)	Setting-Wert
comment	VARCHAR(255)	interne Setting-Notizen

Tabelle 5.8.: Datenbanktabelle `settings`

5.5.9. syslog

Zu Dokumentations- und Debuggingzwecken werden in der `syslog` Relation Benutzeraktivitäten – beispielsweise Loginversuche – und Systemereignisse gespeichert.

Bezeichner	Datentyp	Kommentar
id	INT (PK)	Item-ID
stamp	TIMESTAMP	Aktueller Zeitstempel
user	INT (FK)	ID des agierenden Benutzers
si	MEDIUMINT (FK)	ID des betroffenen Incidents
action	VARCHAR(40)	Durchgeführte Aktion
status	ENUM	OK, ERROR, INFO
details	VARCHAR(255)	ggf. Parameter der durchgeführten Aktion

Tabelle 5.9.: Datenbanktabelle `syslog`

6. Implementierung

Dieses Kapitel beschreibt die Implementierung der zuvor konzipierten Webanwendung anhand ausgewählter Codebeispiele. Zunächst wird im Rahmen der Softwarearchitektur auf die Komponenten der Webanwendung (Module, Aktionen, Funktionen, Cronjobs) und deren Zusammenhänge eingegangen. Darüber hinaus werden technische Grundlagen hinsichtlich Authentifizierung, IT-Sicherheit, Userinterface und Kommunikation betrachtet. Dabei richtet sich der Fokus auf die Implementierung der prozessunterstützenden Funktionen. Anschließend wird auf die Erfassung von Sicherheitsvorfällen und deren in Prozessphasen gegliederte Bearbeitung eingegangen. In diesem Abschnitt wird unter anderem erläutert, wie die Klassifikation und Priorisierung von Vorfällen sowie die Zusammenstellung eines Security Incident Teams realisiert wird. Darüber hinaus wird aufgezeigt, wie Incident Duplikate und Information Requests gehandhabt werden. Abschließend wird die Umsetzung der Filterungs-, Auswertungs- und Exportfunktionen beschrieben.

6.1. Softwarearchitektur

Zu Beginn der Implementierung stellt sich die Frage, ob eines der zahlreichen PHP-Frameworks wie das Zend Framework, Symfony oder CodeIgniter eingesetzt werden soll. Derartige Webframeworks geben die Architektur einer Webanwendung vor und stellen eine Vielzahl an fertig implementierten Modulen bereit, die den Programmieraufwand reduzieren sollen. Aufgrund ihrer universellen Ausrichtung sind die meisten Frameworks funktionell überladen und an einigen Stellen wenig flexibel. Wegen der sehr speziellen Anforderungen an das zu implementierende Security Incident Management System wird auf die Verwendung eines Frameworks verzichtet. Stattdessen wird für diese Webanwendung eine individuelle, leichtgewichtige Systemarchitektur entwickelt, die nur die tatsächlich benötigten Funktionsbibliotheken enthält und eine flexible Implementierung zulässt.

Als wesentliches Strukturprinzip ist die Trennung von Anwendungslogik und Benutzeroberfläche zu betrachten. Die im Browser aufrufbaren Bestandteile der Webanwendung werden als PHP-Dateien im Root-Verzeichnis gespeichert. Diese beinhalten einen Teil der Anwendungslogik und führen auf deren Grundlage Datenbankabfragen durch, deren Ergebnisse an eine Template-Engine übergeben werden. Diese bereitet die Daten unter Zuhilfenahme von HTML-Vorlagen für die Darstellung im Browser auf. Der wesentliche Teil der Anwendungslogik befindet sich in Moduldateien, die wiederum Aktionen beinhalten. Diese verarbeiten alle Interaktionen des Nutzers mit dem System und erfordern eine Parameterübergabe mittels `POST`- oder `GET`-Requests. Die soeben beschriebene Ablauflogik der Softwarekomponenten wird in Abbildung 6.2 veranschaulicht.

Mehrfach benötigte Codeabschnitte werden als Funktionen implementiert und zentral bereitgestellt. Während die Datei `/inc/functions.si.php` insgesamt 18 speziell für diese Anwendung programmierte Basisfunktionen enthält, beinhaltet die Datei `/inc/functions.si.php` weitere 14 Funktionen, die für die prozesskonforme Bearbeitung der Security Incidents erforderlich sind.

Zur Realisierung von komplexen Standardaufgaben wird auf fünf frei verfügbare Funktionsbibliotheken zurückgegriffen:

- Die Templateklasse `RainTPL` fungiert als Schnittstelle zwischen dem Userinterface und den darunterliegenden Systemkomponenten und ermöglicht somit die Trennung von Benutzeroberfläche und Anwendungslogik.
- Zur Benutzerauthentifizierung und zur Synchronisation von Personendaten wird die Klasse `adLDAP` verwendet. Diese bildet die Schnittstelle zum Active Directory basierten LDAP-Verzeichnisdienst.
- Für den Versand von E-Mails wird die `PHPMailer`-Klasse eingesetzt.
- Die Generierung von PDF-Dokumenten übernimmt die Klasse `TCPDF`.
- Die Klasse `mobileDetect` dient der serverseitigen Erkennung von Anwendern, die mittels mobilem Endgerät auf die Anwendung zugreifen.

Die eingesetzten PHP-Klassen sind unter der GNU Lesser General Public License bzw. MIT-Lizenz veröffentlicht und somit frei verwendbar.

Der Einsatz einer flachen Verzeichnishierarchie ermöglicht eine übersichtliche Anordnung der Softwarekomponenten und ein unkompliziertes Zusammenfügen der einzelnen Dateien zu einer Applikation. Die Ordnerstruktur ist gemäß Abbildung 6.1 aufgebaut. Während sich auf der obersten Ebene alle durch den Nutzer im Browser aufrufbaren Seiten der Webanwendung befinden, beinhaltet der `actions`-Ordner die Anwendungslogik (Aktionen bzw. Module) für die Benutzerinteraktion. Das Verzeichnis `assets` stellt die für die Benutzeroberfläche benötigten Grafiken, Stylesheets und JavaScripts bereit. Die an verschiedenen Stellen der Software zu inkludierenden Dateien wie beispielsweise die Funktionsbibliotheken und die Konfigurationsdatei werden im Ordner `inc` gespeichert. Das Verzeichnis `mobile` beinhaltet nur eine Datei, die im Falle eines mobilen Zugriffs im Browser angezeigt wird. Während im `tmp`-Verzeichnis temporäre, von der Template-Engine erzeugte Cachingdateien liegen, sind die zugehörigen HTML-Vorlagen der Benutzeroberfläche im `tpl`-Ordner vorzufinden.

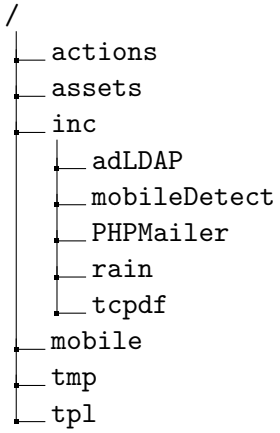

Abbildung 6.1.: Ordnerstruktur

Im Folgenden wird die einheitliche Codestruktur der Webanwendung anhand der Datei `si-monitoring.php` (vgl. Listing 6.1) exemplarisch dargestellt. In Zeile 1 wird definiert, ob die Datei die Template-Engine benötigt. Da alle Dateien auf der Root-Ebene im Browser darstellbar sind, ist die Boole'sche Variable `$useTemplate` dort mit `TRUE` belegt. Im Gegensatz dazu weist diese Variable bei den Moduldateien im `actions`-Ordner den Wert `FALSE` auf, da für die Verarbeitung der Nutzereingaben die Template-Klasse nicht benötigt wird. Daraufhin wird in Zeile 2 die `$feature`-Variable mit dem Dateinamen ohne Extension belegt, sodass der weitere Skriptverlauf in Abhängigkeit von der `$feature`-Bezeichnung gesteuert werden kann. In den Zeilen 4 bis 7 werden die grundlegenden Codebestandteile geladen, die für die weitere Skriptverarbeitung benötigt werden:

- In der `config`-Datei werden diverse Zugangsdaten (Root-User, Datenbank, LDAP, SMTP) definiert. Einstellungen, die beispielsweise die Template-Engine und die PDF-Klasse betreffen, werden in Abhängigkeit von den in den Zeilen 1 und 2 festgelegten Variablen geladen.
- Die `pdo`-Datei baut unter Verwendung der in der Konfigurationsdatei gespeicherten Zugangsdaten eine Verbindung zur Datenbank auf und erzeugt hierfür ein PDO-Objekt.
- Verschiedene zentrale Aufgaben erledigt die `basics`-Datei. Sie erzwingt beispielsweise eine HTTPS-Verbindung, filtert eventuelle Benutzereingaben und erneuert die Session-Variable `lastUrl`. Diese Variable beinhaltet die URL der zuletzt aufgerufenen Seite. Treten im Laufe der Skriptverarbeitung Fehler auf, so kann der Nutzer auf die zuletzt besuchte Seite umgeleitet werden. Ggf. wird schließlich die Template-Engine in Abhängigkeit von der `$useTemplate`-Variable initialisiert. Allgemeine Systemdaten, Informationen zum eingeloggten Benutzer, das geparste Navigationsmenü und eventuelle Systemmeldungen sollen in allen Bereichen der Webanwendung zur Verfügung stehen und werden daher an zentraler Stelle an das Template-Objekt übergeben.
- Mit der `functions`-Datei werden die allgemeinen Systemfunktionen geladen. Ob auch die incidentbezogenen Funktionen geladen werden, wird zur Laufzeit entschieden. Sofern die `$feature`-Bezeichnung die Zeichenkette „si-“ enthält, wird auch die Datei `/inc/functions.si.php` geladen. Zudem werden wesentliche Daten des aufgerufenen Vorfalls aus der Datenbank abgefragt und als `$siMeta`-Array an das Template-Objekt übergeben.

Listing 6.1: Codesturktur am Beispiel `si-monitoring.php`

```
1  $useTemplate = TRUE;
2  $feature = "si-monitoring";
3
4  include_once("inc/config.php");
5  include_once("inc/pdo.php");
6  include_once("inc/basics.php");
7  include_once("inc/functions.php");
8
9  checkUser("CSIRT,ROOT");
10 check("Status",$_GET['si'],$siMeta["currentState"],"MON,CLOSED");
11 check("Klassifikation",$_GET['si'],$siMeta["klassifikation"],"INCIDENT");
12 checkTeamComplete($db,$_GET['si']);
13
14 $query = "SELECT id,monDauer,monIntervall,monAufgabe,monAuffaellig FROM ".
       DBPREFIX."incidents WHERE id=".$_GET['si'];
15 $stmt = $db->query($query);
16 $si = $stmt->fetch(PDO::FETCH_ASSOC);
17
18 $tpl->assign( 'si', $si );
19
20 $db = null;
21 $tpl->draw($feature);
```

Nach den `include`-Operationen werden in den Zeilen 9 bis 11 anhand der zuvor geladenen `check` Funktionen die Voraussetzungen zum Ausführen dieser Datei geprüft. Im Falle der Monitoringseite haben nur eingeloggte Benutzer der Gruppen `CSIRT` und `ROOT` Zugriff. Zudem kann die Monitoringphase nur dann aufgerufen werden, wenn sich der Vorfall im Status

MON oder CLOSED befindet. Da die Monitoringphase nicht bei Information Requests oder Incident Duplikaten, sondern nur bei originalen Security Incidents vorgesehen ist, ist auch eine Prüfung der Klassifikation notwendig. Schließlich wird in Zeile 12 überprüft, ob die Mindestanforderungen an das Secuirty Incident Team erfüllt sind, ob also neben dem Vorfallmelder auch Hotliner und SIC festgelegt sind. Schlägt eine der Prüfungen fehl, wird die Ausführung dieser Datei abgebrochen und der Nutzer wird unter Angabe einer Fehlermeldung zur vorigen Seite zurückgeleitet. Eine genauere Beschreibung der Funktion `checkTeamComplete` erfolgt im Abschnitt 6.2.5.

Sind alle Voraussetzungen erfüllt, so werden die für die jeweilige Seite erforderlichen Datenbankabfragen durchgeführt (Zeilen 14-16). Daraufhin werden die selektierten Datensätze in Zeile 18 an das Template-Objekt übergeben. Nach dem Schließen des Datenbankhandlers in Zeile 20 wird schließlich das passende Template gerendert und an den Browser ausgegeben (Zeile 21).

Abbildung 6.2 verdeutlicht die Zusammenhänge der einzelnen Softwarekomponenten anhand des soeben beschriebenen Monitoringbeispiels. Den Ausgangspunkt stellt die im Browser aufgerufene Datei `/si-monitoring.php` dar, die in der Abbildung gelblich unterlegt und rot umrandet ist. Da die `include`-Dateien direkter Bestandteil der aufrufenden Datei werden, sind diese ebenfalls gelblich eingefärbt. Während die Datenbankabfrage als roter Pfeil dargestellt ist, wird der Weg des Datensatzes von der Datenbank zum Template mit grünen Pfeilen visualisiert. Auch das blau hinterlegte HTML-Template inkludiert weitere Template-Bestandteile. Die im HTML-Quellcode enthaltene Template-Syntax ist fett gesetzt. Durch Absenden des Formulars wird ein `POST`-Request ausgelöst (blauer Pfeil), der die `updateIncidentValues` Aktion des grün eingefärbten `incidents`-Moduls aufruft. Die `updateIncidentValues` Aktion ändert die Daten in der Datenbank (roter Pfeil) und speichert eine Erfolgs- bzw. Fehlermeldung in einer Session-Variablen (violetter Pfeil). Schließlich leitet sie den Nutzer zur Ursprungsseite zurück (blauer Pfeil). Dabei wird die im Session-Kontext gespeicherte Systemmeldung geladen und angezeigt, um dem Anwender zu signalisieren, ob der Vorgang erfolgreich verlaufen ist.

Der Entwurf der Systemarchitektur ermöglicht durch Kapselung der Softwarebestandteile (Templates, Aktionen, Module, Funktionen) eine unkomplizierte Weiterentwicklung der Webanwendung. Darüber hinaus werden SSI-Templates und Aufzählungsdatentypen nicht fest im Quellcode hinterlegt. Stattdessen werden beispielsweise Kunden, Incident- und Lösungstypen sowie Prioritäten und Reaktionszeiten in einer Typentabelle in der Datenbank gespeichert, sodass eine effiziente Anpassung und Erweiterung dieser Typen möglich ist, falls beispielsweise ein neuer Kunde hinzukommt oder ein weiterer Standard Security Incident definiert wird. Zur komfortablen Handhabung dieser speziellen Datentypen werden die später beschriebenen Funktionen `getTypeName`, `getTypesArray` und `parseDropdown` angewendet. Dadurch können Dropdown-Felder bedarfsweise generiert und stets mit aktuellen Einträgen aus der Datenbank befüllt werden.

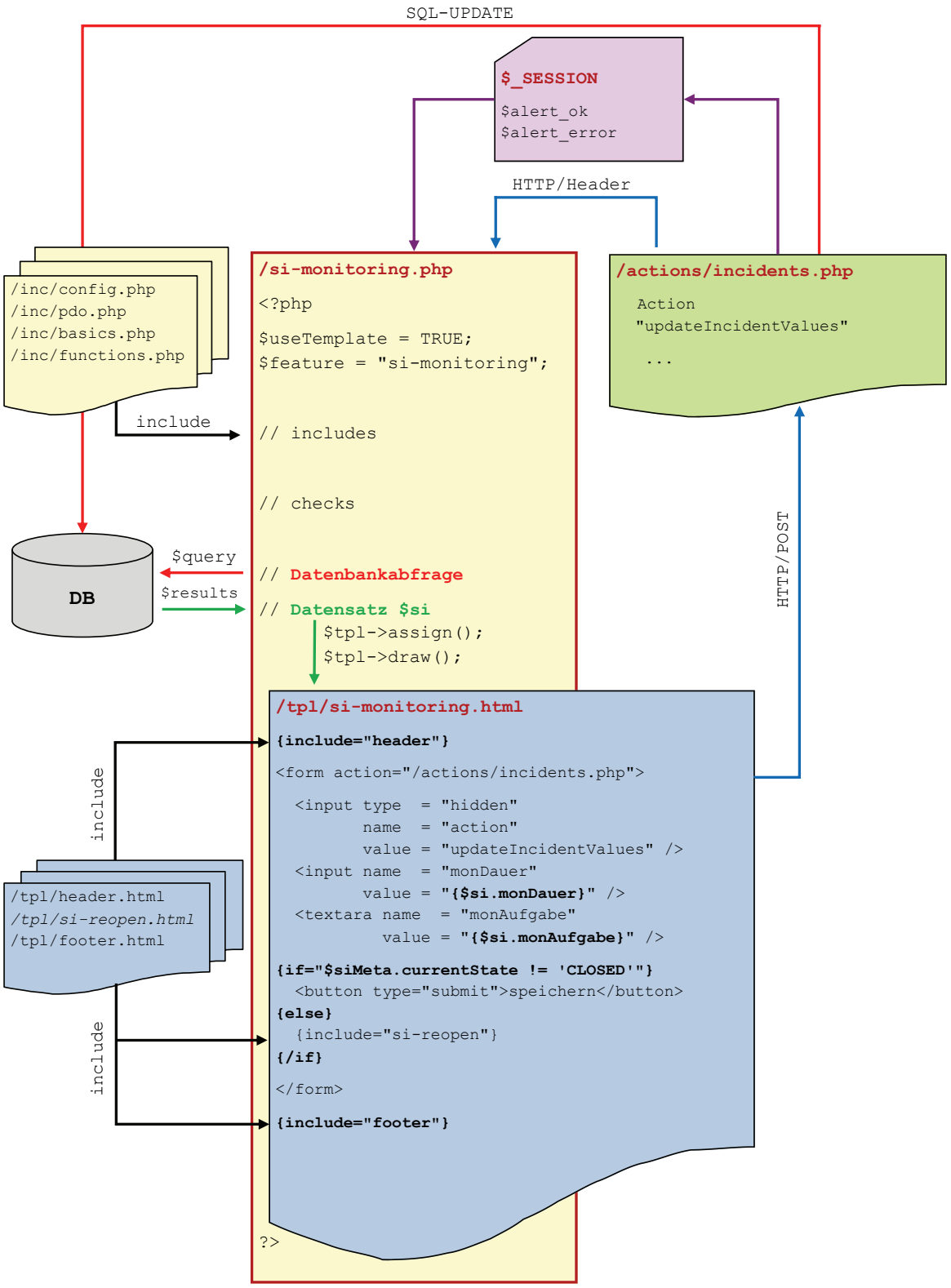

Abbildung 6.2.: Strukturdiagramm und Ablauflogik der Softwarekomponenten

6.1.1. Module und Aktionen

Die Webanwendung stellt insgesamt 31 Aktionen für die Benutzerinteraktion bereit. Soll beispielsweise ein Sicherheitsvorfall wiedereröffnet werden, ruft der Nutzer die Aktion `reopen Incident` auf, welche die Anwendungslogik für diesen Vorgang beinhaltet. Aufgrund der Vielzahl an Aktionen werden diese anhand ihres logischen Zusammenhangs gruppiert und in folgenden elf Modulen gebündelt:

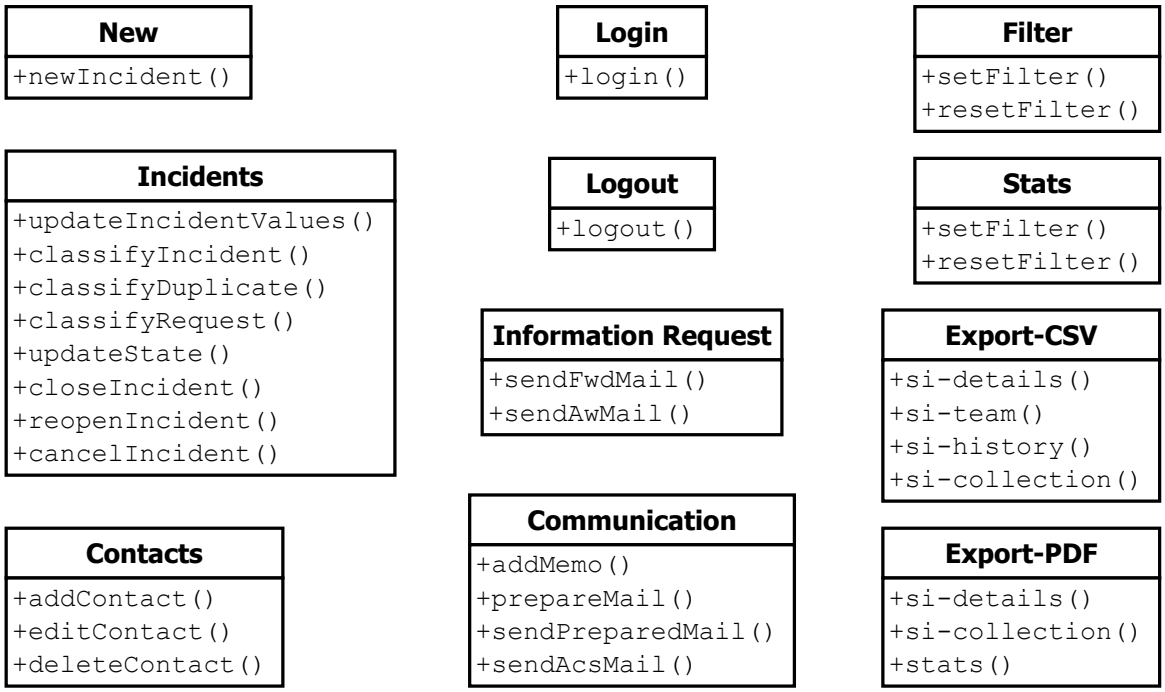

Abbildung 6.3.: Module und Aktionen

New – Dieses Modul umfasst die `newIncident` Aktion zum Anlegen eines neuen Vorfalls inklusive der damit verbundenen Vorgänge wie dem Versand von E-Mail-Benachrichtigungen.

Incidents – Das Incidents-Modul beinhaltet acht Aktionen für die Bearbeitung von Sicherheitsvorfällen wie beispielsweise die Klassifizierung eines Incidents sowie das Schließen, Wiedereröffnen und Abbrechen eines Vorfalls. Die meistgenutzte Aktion `updateIncidentValues` ermöglicht das automatisch dokumentierte Speichern der eingegeben Daten während aller Prozessphasen. Im Prozesskontext stellt `updateState` die wichtigste Aktion dar. Diese ermöglicht den Wechsel in die nächste Prozessphase und übernimmt die automatische Berechnung der Laufzeit der abgeschlossenen Prozessphase.

Contacts – Für die flexible Verwaltung des Security Incident Teams eines jeden Vorfalls werden die beteiligten Personen als Kontakte betrachtet. Dieses Modul stellt Aktionen zum Hinzufügen und Löschen von Teammitgliedern sowie zum Bearbeiten ihrer Kontaktdaten bereit.

Login – Die Benutzerauthentifizierung im Rahmen des Anmeldevorgangs und die Initialisierung einer Session wird durch das Login-Modul abgewickelt.

Logout – Das Logout-Modul beendet die Benutzersitzung und dokumentiert die Abmeldung im Systemlog.

Information Request – Wird ein Vorfall als Information Request eingestuft, so stellt dieses Modell die für die Bearbeitung erforderlichen Aktionen bereit. Falls der Request an die zuständige Bearbeitergruppe weiterzuleiten ist, wird die `sendFwdMail` Aktion ausgeführt. Im Falle einer Beantwortung des Requests gegenüber dem Vorfallmelder kommt die Aktion `sendAwMail` zum Einsatz.

Communication – Das Kommunikationsmodul ermöglicht eine dokumentierte Abwicklung der Prozesskommunikation und beinhaltet Aktionen zur Erfassung von Gesprächsnotizen, zur Vorbereitung und zum Versand von E-Mails sowie zum Versand einer Meldung an die Allianz für Cyber-Sicherheit.

Filter – Für die Filterung von Incidents stehen zwei Aktionen zum Setzen und Zurücksetzen des Filters zur Verfügung.

Stats – Zur Anpassung der Auswertungszeitraums und zur Auswahl eines Kunden stellt das Statistik-Modul die Aktionen `setFilter` und `resetFilter` bereit.

Export-CSV – Dieses Modul ermöglicht den Export von incidentbezogenen Daten im CSV-Format. Es existieren Aktionen sowohl für den Export der Incident Daten als auch der Kontaktdaten der Teammitglieder und der Einträge des Verlaufsprotokolls. Die Aktion `si-collection` ermöglicht einen Massenexport von selektierten Incident Daten.

Export-PDF – Dieses Modul ermöglicht den Export von incidentbezogenen Daten im PDF-Format. Sowohl für den Einzelexport einer Incidentstatusseite als auch für den Massenexport stehen Aktionen zur Verfügung. Schließlich lassen sich auch die Auswertungen als PDF-Datei exportieren.

Abbildung 6.3 stellt die Module und Aktionen überblicksartig dar. Auch bei der Implementierung der Aktionen wird zu Optimierungszwecken häufig auf Funktionen zurückgegriffen. Diese werden in den beiden folgenden Abschnitten beschrieben.

6.1.2. Allgemeine Basisfunktionen

Zur Optimierung des Programmieraufwands werden mehrfach vorkommende Aufgaben zentral als Funktionen implementiert, die in der gesamten Webanwendung wiederverwendet werden können. Die folgende Auflistung gibt einen alphabetisch sortierten Überblick über die 18 in der Datei `/inc/functions.php` bereitgestellten Basisfunktionen:

calcAbweichung Berechnet die absolute und prozentuale Abweichung eines tatsächlichen Zeitwerts in Minuten von einem vorgegebenen Zeitwert. Die Ausgabe erfolgt unter Anwendung der Funktion `secondsToString` als formatierter String. Bei Überschreitung der Vorgabe ist der String rot gefärbt, bei Unterschreitung grün.

check Universelle Prüfungsfunktion, ob ein übergebener Wert (`$current`) in einer zugelassenen Wertemenge (`$allowed`) enthalten ist. Da beide Parameter übergeben werden, ist kein Datenbankzugriff erforderlich. Ist die Prüfung erfolgreich, wird `TRUE` zurückgegeben. Im Fehlerfall wird nach Umleitung zur zuletzt besuchten Seite eine entsprechende Meldung angezeigt.

checkToken Prüft, ob der übergebene Token mit dem Session-Token übereinstimmt. Im Erfolgsfall ist der Return-Wert `TRUE`. Im Fehlerfall wird nach Umleitung zur zuletzt besuchten Seite eine entsprechende Meldung angezeigt.

checkUser Prüft anhand von Session-Variablen, ob ein User angemeldet ist und seine letzte Aktion nicht länger als 30 Minuten zurückliegt (Session-Timeout). Im Fehlerfall wird der Anwender unter Angabe einer Fehlermeldung zur Login-Seite umgeleitet. Darüber hinaus wird geprüft, ob der eingeloggte User eine für den Kontext zugelassene Benutzerrolle aufweist. Dabei wird die als Session-Variable gespeicherte Benutzerrolle mit den als Inputparameter übergebenen zugelassenen Werten abgeglichen. Im Fehlerfall erfolgt eine Umleitung zur zuletzt besuchten Seite und es wird auf die fehlenden Rechte hingewiesen. Im Erfolgsfall wird die Session-Laufzeit verlängert und `TRUE` zurückgegeben.

diffBusinessHours Berechnet den Unterschied zwischen zwei Zeitstempeln in Sekunden, wobei nur Zeiten während der Business Hours berücksichtigt werden. Als Servicezeiten gelten die Zeiträume von Montag bis Donnerstag jeweils von 8 bis 18 Uhr und freitags von 8 bis 16 Uhr.

getContactRole Liefert die Rolle einer Person, die sie innerhalb eines bestimmten Incidents einnimmt (z.B. `MELDER`, `HOTLINER`, `SIC`). Die Abfrage der `contacts`-Tabelle erfolgt anhand der Kennung des Benutzers.

getContactRoleById Liefert die Rolle einer Person, die sie innerhalb eines bestimmten Incidents einnimmt (z.B. `MELDER`, `HOTLINER`, `SIC`). Die Abfrage der `contacts`-Tabelle erfolgt anhand der ID des Benutzers.

getSetting Liefert den Wert eines bestimmten Einstellungsdatensatzes aus der `settings`-Tabelle.

getTypeName Liefert den Namen eines bestimmten Werts einer Kategorie aus der Typentabelle zur Übersetzung eines codierten Wertes in einen verständlichen String. Beispielsweise wird für eine eingegebene Priorität „2“ der String „mittel“ zurückgegeben.

getTypesArray Liefert alle Namen-Wert-Paare einer bestimmten Kategorie aus der Typentabelle als Array. Dient beispielsweise als Input für die Funktion `parseDropdown`, die ein Dropdown-Feld mit allen Lösungstypen als Auswahlmöglichkeiten dynamisch generiert.

getUsersIncidents Liefert die IDs aller Vorfälle als Array, an denen eine anhand ihrer Kennung identifizierte Person beteiligt ist.

parseCsirtDropdown Liefert den HTML-Code eines Dropdown-Felds mit allen CSIRT-Nutzern aus der `users`-Tabelle. Nutzer mit „Funktionskennung“ als Name werden ausgeschlossen. Alte User, die nicht mehr via LDAP aktualisiert werden (vgl. Abschnitt 6.2.2), werden anhand des Feldes `lastLdapUpdate` identifiziert und ebenfalls ausgeschlossen.

parseDropdown Liefert den HTML-Code eines Dropdown-Feldes zurück, dessen Einträge flexibel in einem Array übergeben werden. Die Funktion ist universell für alle Aufzählungstypen geeignet. Meist wird das Input-Array durch die Funktion `getTypesArray` generiert.

parseUserDropdown Liefert den HTML-Code eines Dropdown-Felds mit allen in der `users`-Tabelle vorhandenen Nutzern zurück, die den Benutzerrollen `USER` oder `CSIRT` angehören. User mit „Funktionskennung“ als Name werden ausgeschlossen. Alte User, die nicht mehr via LDAP aktualisiert werden (vgl. Abschnitt 6.2.2), werden anhand des Feldes `lastLdapUpdate` identifiziert und ebenfalls ausgeschlossen.

secondsToString Umwandlung einer Zeitdauer von Sekunden in einen formatierten String (Tage, Std., Min.) unter Anwendung der Modulo-Funktion.

setAlert Speichert eine übergebene Meldung in einer Session-Variable. Diese dient der Benachrichtigung des Nutzers über den Erfolg oder Fehler einer von ihm durchgeführten Aktion. Die Session-Variable wird beim nächsten Seitenaufruf ausgelesen, woraufhin die Meldung angezeigt wird.

setSetting Aktualisiert den Wert eines bestimmten Einstellungsdatensatzes in der `settings`-Tabelle.

writeSyslog Anlegen eines Eintrags in der `syslog`-Tabelle. Dabei werden die ID des agierenden Users, der Typ der durchgeführten Aktion (z.B. `login`, `updateState`), der Status (`OK`, `ERROR`, `INFO`) und ggf. Details der Aktion in die Datenbank geschrieben. Darüber hinaus wird die ID des Vorfalls dokumentiert, falls es sich um eine incidentbezogene Aktion handelt.

6.1.3. Incidentbezogene Funktionen

Besonders im Bereich der Incidentbearbeitung sind wiederholt identische Aufgaben zu erledigen. Daher werden in incidentbezogenen PHP-Dateien zusätzlich zu den Basisfunktionen automatisch 14 weitere Funktionen hinzugeladen, die die Implementierung des Incident Bereichs erleichtern. Die folgende Auflistung gibt einen alphabetisch sortierten Überblick über die in der Datei `/inc/functions.si.php` bereitgestellten Sonderfunktionen:

addComItem Anlegen eines Eintrags in der `communication`-Tabelle zur Dokumentation von E-Mails und Gesprächsnotizen. Die Funktion liefert im Erfolgsfall `TRUE`, im Fehlerfall `FALSE` zurück.

applySsiTemplate Wendet ein Template auf einen Standard Security Incident an. Die Funktion liest den passenden Templatedatensatz aus der Tabelle `ssiTpl` aus und aktualisiert die übereinstimmenden SI-Felder mit den Templatedaten. Liefert im Erfolgsfall `TRUE`, im Fehlerfall `FALSE` zurück.

checkKlassifikation Prüft, ob die tatsächliche Klassifikation eines Vorfalls, die von der Funktion `getIncidentValue` ermittelt wird, in der Menge der erlaubten Klassifikationen

enthalten ist, welche als Input-Parameter übergeben wird. Im Fehlerfall erfolgt eine Umleitung zur zuletzt aufgerufenen Seite mit einer entsprechenden Fehlermeldung. Diese Funktion dient der klassifikationsabhängigen Zugriffskontrolle und wird häufig in den Aktionen bzw. Modulen aufgerufen.

checkState Prüft, ob der tatsächliche Status eines Vorfalls, der von der `getIncidentValue` Funktion ermittelt wird, in der Menge der erlaubten Zustände enthalten ist, welche als Input-Parameter übergeben wird. Im Fehlerfall erfolgt eine Umleitung zur zuletzt aufgerufenen Seite mit einer entsprechenden Fehlermeldung. Diese Funktion dient der statusabhängigen Zugriffskontrolle und wird häufig in den Aktionen bzw. Modulen aufgerufen.

checkTeamComplete Prüft, ob für einen konkreten Vorfall das Team mindestens minimal besetzt ist, also ein Hotliner und ein SIC definiert wurden, sofern dies abhängig vom Status und der Klassifikation erforderlich ist. Falls ein Duplikat oder ein Information Request vorliegt, ist diese Prüfung obsolet und wird mit dem Return-Wert `TRUE` übersprungen. Auch im Status `NEW` und `CLASS` können der Hotliner und der SIC noch nicht festgelegt worden sein, sodass die Prüfung mit dem Return-Wert `TRUE` übersprungen wird. Im Fehlerfall erfolgt eine Umleitung zur Teamseite mit der Aufforderung zur Teamvervollständigung.

existsIncident Prüft, ob ein Vorfall mit der übermittelten ID in der Datenbank existiert. Im Fehlerfall erfolgt eine Umleitung zum Dashboard mit Fehlerhinweis.

getIncidentValue Flexible Funktion zum Abfragen eines SI-Attributs. Liefert den Wert eines Datenbankfeldes für einen SI. Wird beispielsweise in den `check` Funktionen angewendet, um die Klassifikation bzw. den aktuellen Status eines Vorfalls abzufragen.

insertHistory Anlegen eines Eintrags in der `history`-Tabelle. Wird u.a. bei der Änderung von Incident Daten aufgerufen. Die Funktion liefert im Erfolgsfall `TRUE`, im Fehlerfall `FALSE` zurück.

mailToContact Versendet eine E-Mail an einen Prozessbeteiligten und dokumentiert diese anhand der `addComItem` Funktion in der `communication`-Tabelle.

mailToGroup Versendet eine E-Mail an eine Gruppe von Personen und dokumentiert diese anhand der `addComItem` Funktion in der `communication`-Tabelle.

mailToSomebody Versendet eine E-Mail an eine beliebige Person und dokumentiert diese anhand der `addComItem` Funktion in der `communication`-Tabelle.

parseIncidentsDropdown Liefert den HTML-Code eines Drop-Down-Feldes zurück, welches alle nicht archivierten, als Security Incident klassifizierten Vorfälle enthält, wobei die IDs als Wert und die Titel als Bezeichnung fungieren. Dabei wird der Eintrag, dessen ID mit dem Parameter `$active` übereinstimmt, als `selected` markiert.

parseSiContactsDropdown Liefert den HTML-Code eines Dropdown-Feldes zurück, welches alle beteiligten Personen eines konkreten Security Incidents enthält, wobei die Contact-IDs als Wert und die Namen der Personen als Bezeichnung fungieren. Dabei wird der Eintrag, dessen ID mit dem Parameter `$active` übereinstimmt, als `selected` markiert. Dient als Adressbuch der Prozessbeteiligten eines Security Incidents.

updateState Aktualisiert den Status eines Incidents nach Plausibilitätsprüfung des übermittelten Zeitstempels und berechnet die Dauer der aktuellen Prozessphase als Gesamtzeit und während der Servicezeiten unter Anwendung der Funktion `diffBusinessHours`. Die Funktion liefert im Erfolgsfall `TRUE` zurück. Im Fehlerfall erfolgt eine Umleitung zur zuletzt besuchten Seite unter Angabe einer entsprechenden Fehlermeldung.

6.1.4. Cronjobs

Zusätzlich zu den benutzergesteuerten Programmabläufen sind folgende Aktionen regelmäßig und automatisiert auszuführen:

- Synchronisation der `users`-Tabelle mit dem LDAP-Verzeichnisdienst
- Benachrichtigung des SICs, wenn sich ein Incident seit mindestens fünf Tagen in der gleichen Prozessphase befindet
- Benachrichtigung des SICs, wenn sich ein Incident seit mindestens 14 Tagen in der Monitoringphase befindet

Die Aktionen sollen einmal täglich durchgeführt werden. Da der exakte Zeitpunkt der Skriptausführung nicht relevant ist, wird auf die Nutzung der Linux Crontab verzichtet. Stattdessen wird ein einfacher Cron-Dienst innerhalb der Webanwendung implementiert.

Da davon auszugehen ist, dass die Login-Seite der Webanwendung täglich aufgerufen wird, wird das Cron-Skript am Anfang der Datei `index.php` (vgl. Listing 6.2) positioniert. Selbst wenn die Anwendung nicht täglich gestartet werden sollte, so erfolgt die LDAP-Synchronisation spätestens unmittelbar vor dem nächsten Login. Dadurch ist sichergestellt, dass die Personeninformationen stets auf dem aktuellen Stand gehalten werden.

Listing 6.2: Cron-Skript auf der Login-Seite (`/index.php`)

```
1 if (strtotime(getSetting($db,"CRON_LAST_RUN")) < strtotime(date("Y-m-d",time()
    ))){
2   include($appConfig['path']."/inc/cron.php");
3   setSetting($db,"CRON_LAST_RUN",date("Y-m-d",time()));
4 }
```

Beim Aufruf der Datei `index.php` wird in Zeile 1 überprüft, ob das letzte Ausführungsdatum des Cronjobs kleiner als das aktuelle Datum ist. Liefert diese if-Abfrage `TRUE` zurück, so wurde der Cronjob an diesem Tag noch nicht durchgeführt. Folglich wird in Zeile 2 die Cronjob-Datei `/inc/cron.php` geladen und ausgeführt. Diese beinhaltet die Befehle zur Erledigung der oben genannten Aufgaben. Auf die Implementierung der LDAP-Synchronisation wird im nächsten Abschnitt (vgl. Listing 6.5) eingegangen. Abschließend wird in Zeile 3 das in der `settings`-Tabelle gespeicherte Ausführungsdatum aktualisiert. Ergibt die if-Abfrage hingegen `FALSE`, so wurde der Cronjob an diesem Tag bereits ausgeführt und deshalb nun übersprungen.

6.2. Technische Grundlagen

Im Folgenden werden die technischen Grundlagen der Webanwendung hinsichtlich Authentifizierung, IT-Sicherheit, Userinterface, Prozessunterstützung und Kommunikation anhand von Codebeispielen erläutert. Zunächst werden jedoch die Systemvoraussetzungen definiert und die Installation der Webanwendung beschrieben.

6.2.1. Systemvoraussetzungen und Setup

Um den Betrieb der Anwendung auf einem LRZ-Standardwebserver gewährleisten zu können, werden die für den Betrieb der webbasierten Security Incident Management Software erforderlichen technischen Voraussetzungen dementsprechend spezifiziert:

- Linux-Webserver mit PHP Version 5.3.14
- Datenbankserver MySQL Version 5.1.73
- LDAP-Server zur Nutzerauthentifizierung und Kontaktdatenabfrage
- Mailserver zum Mailversand via SMTP

Da die Inbetriebnahme der Webanwendung in wenigen Schritten manuell durch einen Administrator durchgeführt werden kann und keine technischen Spezialkenntnisse erfordert, wird auf die Implementierung eines webbasierten Installationsassistenten verzichtet. Während der Installation der Webanwendung sind zunächst die Datenbanktabellen gemäß Anhang B anzulegen. Anschließend ist die Typentabelle gemäß Anhang C mit den Standarddatensätzen zu initialisieren. Das Datenbank-Setup erfolgt durch den Import von zwei SQL-Dateien, die sich im Ordner `/inc` befinden. Hierfür wird die Verwendung des Administrationstools `phpMyAdmin` empfohlen. Nach der Vorbereitung der Datenbank sind nur noch die Konfigurationsparameter in der Datei `/inc/config.php` anzupassen. Dort sind beispielsweise die Zugangsdaten für Datenbank, LDAP und SMTP einzutragen. Abschließend ist die Webanwendung via FTP auf dem Webserver bereitzustellen.

6.2.2. Benutzerverwaltung, Authentifizierung und Rechtevergabe

Da der LDAP-Verzeichnisdienst des LRZ nicht nur die Kontaktdaten aller LRZ-Mitarbeiter bereitstellt, sondern auch zur Nutzerauthentifizierung verwendet werden kann, wird bei der Implementierung mehrfach auf die Klasse `adLDAP` zurückgegriffen.

Um sich gegenüber der Webanwendung zu authentifizieren, trägt der Benutzer auf der Startseite (`/index.php`) seine LRZ-Kennung und das zugehörige Passwort ein. Mit dem Absenden des Formulars werden die Zugangsdaten via `POST`-Request an das Login-Modul (`/actions/login.php`) übergeben. Dieses prüft zunächst, ob die übermittelten Daten mit den in der Konfigurationsdatei gespeicherten Zugangsdaten des Root-Users übereinstimmen. Trifft dies nicht zu, so wird eine LDAP-Authentifizierung in die Wege geleitet. Dieser Abschnitt des Login-Moduls (vgl. Listing 6.3) wird im Folgenden näher betrachtet:

Listing 6.3: Nutzerauthentifizierung via LDAP (`/inc/login.php`)

```
1  elseif ($adldap->user()->authenticate($_POST["kennung"].$ldapSuffix, $_POST["
       pwd"])){
2
3    $query = "SELECT * FROM ".DBPREFIX."users WHERE kennung = '".$_POST["kennung
       "]."'";
4    $stmt = $db->query($query);
5    if ($stmt->rowCount() == 1) {
6      $userData = $stmt->fetch(PDO::FETCH_ASSOC);
7      $userid = $userData["id"];
8      writeSyslog($db,$userid,NULL,'login','OK',$_POST["kennung"].' login
       successful');
9    }
10   else {
11     writeSyslog($db,0,NULL,'login','ERROR',$_POST["kennung"].' LDAP Login
       fehlgeschlagen. User nicht in DB.');
12     header("Location: ".$appConfig['url']."/index.php?msg=5");
13     exit;
14   }
15
16   if ($adldap->user()->inGroup($userData["kennung"],$ldapGroupNameCsirt)) {
       $userRole = "CSIRT";}
17   elseif ($adldap->user()->inGroup($userData["kennung"],$ldapGroupNameUser)) {
       $userRole = "USER";}
18   else {
19     writeSyslog($db,0,NULL,'login','ERROR',$_POST["kennung"].' LDAP Login
       fehlgeschlagen. Ungültige Gruppenzugehörigkeit.');
20     header("Location: ".$appConfig['url']."/index.php?msg=5");
21     exit;
22   }
23
24   $sql = "UPDATE ".DBPREFIX."users SET loginStamp=?, ip=?, userRole=? WHERE id
       =?";
25   $query = $db->prepare($sql);
26   $query->execute(array($stamp,$ip,$userRole,$userid));
27
28   $auth = TRUE;
29 }
```

In Zeile 1 werden sowohl die Kennung als auch das Passwort des Nutzers an die `authenticate` Methode des `adLDAP`-Objekts übergeben. Diese liefert bei erfolgreicher Authentifizierung `TRUE` zurück, sodass die Anweisungen in den Zeilen 3 bis 28 ausgeführt werden. In den Zeilen 3 und 4 wird anhand der der LRZ-Kennung der passende Datensatz aus der `users`-Tabelle abgerufen. Enthält die SQL-Abfrage genau einen Datensatz, so werden die Nutzerdaten für die weitere Verwendung lokal gespeichert. Der erfolgreiche Login wird durch die `writeSyslog` Funktion im Systemlog dokumentiert. Der Fall, für einen authentifizierten Benutzer in der `users`-Tabelle kein Eintrag existiert (Zeile 10), dürfte aufgrund der nachfolgend beschriebenen Synchronisation nicht auftreten. Dennoch wird dieser Fehler in den Zeilen 11 bis 13 behandelt.

Da die Rechtevergabe innerhalb der Webanwendung statisch auf Basis der Benutzerrolle erfolgt, hätte eine fehlerhafte Zuweisung erhebliche Auswirkungen auf die Informationssicherheit. Daher verlässt sich die Webanwendung nicht auf die in Zeile 3 bereits aus der `users`-Tabelle abgefragte `userRole`. Obwohl eine Manipulation der Benutzerrollen in der

Datenbank sehr unwahrscheinlich ist, aber dennoch nicht vollständig ausgeschlossen werden kann, wird die Benutzerrolle in Echtzeit über den LDAP-Verzeichnisdienst ermittelt. Unter Anwendung der `inGroup`-Methode des `adLDAP`-Objekts wird im Rahmen eines if-Konstrukts zunächst auf Mitgliedschaft des Benutzers in der CSIRT-LDAP-Gruppe (Zeile 16) und anschließend auf Mitgliedschaft in der LDAP-Gruppe der LRZ-Mitarbeiter (Zeile 17) geprüft. Ist der authentifizierte Benutzer weder ein CSIRT-Mitglied noch ein LRZ-Mitarbeiter, so wird ihm der Zugriff zur Webanwendung verwehrt (Zeilen 19 bis 21).

Abschließend wird in den Zeilen 24 bis 26 die IP-Adresse des Anwenders, seine `userRole` und der Login-Zeitstempel in der `users`-Tabelle aktualisiert. Der erfolgreiche Abschluss der Authentifizierung wird systemintern durch Setzen der `$auth`-Variable repräsentiert (Zeile 28).

Falls die die Authentifizierung fehlgeschlagen ist, so wird der Anwender unter Angabe einer entsprechenden Fehlermeldung zur Login-Seite zurückgeleitet. Zudem wird die Benutzerkennung des gescheiterten Loginversuchs im Systemlog dokumentiert. Ist hingegen das `$auth`-Flag gesetzt, so werden abschließend noch allgemeine Login-Operationen durchgeführt. Dazu zählt die Belegung von Session-Variablen mit den Benutzerdaten, die Generierung des Session-Tokens sowie die Umleitung zum Dashboard.

Beim Aufruf des Dashboards wird durch den Funktionsaufruf `checkUser(USER,CSIRT,ROOT)` sichergestellt, dass entweder ein LRZ-Mitarbeiter, ein CSIRT-Mitglied oder der Root-User eingeloggt ist und seine Sitzung noch nicht abgelaufen ist. Die `checkUser` Funktion schützt sowohl die Benutzeroberfläche als auch die in den Modulen enthaltene Anwendungslogik vor unberechtigten Zugriffen. Jedoch stößt die gruppenbasierte Rechtevergabe bei der Incidentstatusseite an ihre Grenzen, da diese neben den CSIRT-Mitgliedern auch von allen an einem Incident beteiligten Personen unabhängig von ihrer Benutzerrolle aufrufbar sein soll. Für diesen Fall wird eine dynamische Rechtevergabe implementiert:

Listing 6.4: Dynamische Rechtevergabe (`/inc/si-overview.php`)

```
1 if (!(getContactRole($db,$_SESSION['userKennung'],$_GET['si']) != FALSE) OR
    checkUser("CSIRT,ROOT"))){
2   setAlert('error','Sie besitzen nicht die erforderlichen Rechte, um den
    Vorgang auszuführen.');
3   header("Location: dashboard.php");
4   exit;
5 }
```

Die if-Bedingung in Listing 6.4 besteht im Wesentlichen aus der Negation einer OR-Verknüpfung von Boole'schen Werten. Im ersten Teil wird anhand der `getContactRole` Funktion geprüft, ob der Anwender am Incident beteiligt ist. Handelt es sich beim Anwender beispielsweise um den Vorfallmelder, wird der Vergleich `'MELDER' != FALSE` zu `TRUE` ausgewertet, sodass der zweite Teil der OR-Verknüpfung übersprungen wird. Damit alle CSIRT-Mitglieder und der Root-User auch unabhängig von ihrer konkreten Prozessbeteiligung die Statusseite einsehen können, liefert die `checkUser` Funktion für diese beiden Benutzergruppen `TRUE` zurück. Da der if-Clause den Negativfall – also den Zugriff eines Anwenders, der weder am Vorfall beteiligt noch CSIRT- oder Root-User ist – behandelt, wird der gesamte Ausdruck negiert.

Möchte sich der Anwender vom System abmelden, so ruft er das Logout-Modul (`/actions/logout.php`) auf. Dieses dokumentiert zunächst den Vorgang im Systemlog (`writeSyslog`) und löscht daraufhin mit der Funktion `session_unset` alle Session-Variablen, die gegenwärtig

registriert sind. Nach der Beendigung der Session mit der Funktion `session_destroy` wird der Anwender zur Loginseite umgeleitet. Der Aufruf von geschützten Seiten ist nun mangels Session nicht mehr möglich und würde einen erneuten Login erfordern.

Was die Benutzerverwaltung anbelangt, so wird aus Performancegründen darauf verzichtet, für jeden Zugriff auf Benutzerdaten eine LDAP-Abfrage in Echtzeit durchzuführen. Stattdessen werden die Personen- und Kontaktinformationen von allen LRZ-Mitarbeitern und CSIRT-Mitgliedern nur einmal täglich und in einem Vorgang vom Verzeichnisdienst abgerufen und in der `users`-Tabelle gespeichert. Dadurch wird einerseits die Belastung des LDAP-Dienstes durch die Webanwendung auf ein Minimum reduziert und andererseits stehen dem Root-User bei einem LDAP-Ausfall dennoch die Kontaktdaten aller Kollegen zur Verfügung, die er insbesondere bei der Zusammenstellung eines Security Incident Teams benötigen würde. Die LDAP-Synchronisation erfolgt im Rahmen des auf der Login-Seite implementierten Cronjobs (vgl. Listing 6.5) automatisch maximal einmal täglich. Nach dem Laden der Datei `/inc/ldap.php`, in der das `$adldap`-Objekt erstellt wird, baut die lesende LDAP-Kennung in Zeile 3 eine Verbindung zum Verzeichnisdienst auf. Kann die Verbindung nicht aufgebaut werden, so erfolgt in den Zeilen 6 bis 12 die Fehlerbehandlung. Bei der folgenden Synchronisierung wird insofern auf die Reihenfolge geachtet, als dass zunächst die LRZ-Mitarbeiter und erst dann die CSIRT-Mitglieder aktualisiert werden, weil die für die Berechtigungsvergabe relevante Benutzerrolle anhand der Gruppenzugehörigkeit festgelegt wird. Da jedes CSIRT-Mitglied auch der Gruppe der LRZ-Mitarbeiter angehört, wird in den Zeilen 44 bis 48 zuerst die allgemeinere Gruppe der LRZ-Mitarbeiter und anschließend die speziellere CSIRT-Gruppe aktualisiert.

Zur Synchronisation wird die in den Zeilen 14 bis 42 implementierte Funktion `ldapSync` aufgerufen. Als Eingabeparameter werden neben dem Datenbankhandler und dem `adLDAP`-Objekt die mit der `adLDAP`-Klasse abgefragten Mitglieder der jeweiligen Gruppe sowie die zugehörige Benutzerrolle übergeben. Bei dieser Funktion handelt es sich im Wesentlichen um eine große foreach-Schleife (Zeilen 20 bis 40), die über alle übergebenen Gruppenmitglieder iteriert. Dabei werden in Zeile 21 zunächst die Kontaktdaten der jeweils aktuellen Person aus dem LDAP-Verzeichnis abgerufen. Anschließend wird in den Zeilen 23 bis 25 der korrespondierende Eintrag anhand der LRZ-Kennung aus der `users`-Tabelle der Datenbank selektiert. Liefert die Datenbankabfrage kein Ergebnis zurück (Zeile 27), so handelt es sich um einen neuen Mitarbeiter, für den in der `users`-Tabelle ein neuer Datensatz angelegt wird (Zeilen 28 bis 30). Ergibt die Datenbankabfrage ein Ergebnis (Zeile 33), so wird dieser Datensatz mit den LDAP-Werten überschrieben (Zeilen 34 bis 37). Sowohl bei der `INSERT`- als auch bei der `UPDATE`-Operation auf der Datenbank wird die `userRole` gemäß Inputparameter gesetzt. Darüber hinaus wird das Feld `lastLdapUpdate` aktualisiert, sodass die Webanwendung unterscheiden kann, ob es sich um einen aktuellen Datensatz oder um einen ausgeschiedenen Mitarbeiter handelt. Die in der Funktion verwendeten Counter geben am Ende Auskunft über die Gesamtzahl der verarbeiteten Datensätze sowie die Anzahl der neu angelegten und aktualisierten Benutzer. Der Rückgabewert der Funktion `ldapSync` enthält die Counter-Informationen als String und wird im Systemlog gespeichert.

Betrachtet man abschließend ein CSIRT-Mitglied, so wird dieses beim ersten Aufruf der Funktion `ldapSync` als `USER` in der Datenbank hinterlegt, da alle CSIRT-Mitglieder auch der Gruppe der LRZ-Mitarbeiter angehören. Beim zweiten Aufruf der `ldapSync` Funktion wird ihm die speziellere Benutzerrolle `CSIRT` zugewiesen. Dementsprechend ist bei einer späteren Implementierung weiterer Benutzergruppen insbesondere auf die Reihenfolge während der LDAP-Synchronisation zu achten.

Listing 6.5: LDAP-Synchronisation (/inc/cron.php)

```
require_once($appConfig['path']."/inc/ldap.php");

if($adldap->user()->authenticate($ldapUser.$ldapSuffix,$ldapPwd)) {
  writeSyslog($db,NULL,NULL,'ldapSync','OK','Login for Sync ok.');
}
else {
  writeSyslog($db,NULL,NULL,'ldapSync','ERROR','Login for Sync failed.');
  $adldap->close();
  $db = null;
  header("Location: ".$appConfig['url']."/index.php?msg=5");
  exit;
}

function ldapSync ($db,$adldap,$group,$userRole){
  $updateCounter = 0;
  $insertCounter = 0;
  $generalCounter = 0;
  $stamp = date("Y-m-d",time());

  foreach ($group as $item){
    $user = $adldap->user()->infoCollection($item, array('cn','sn','givenname'
    ,'telephonenumber','mail'));

    $query = "SELECT id FROM ".DBPREFIX."users WHERE kennung = '$user->cn'";
    $stmt = $db->query($query);
    $result = $stmt->fetch(PDO::FETCH_ASSOC);

    if (!$result) { // Neuer User -> anlegen
      $sql = "INSERT INTO ".DBPREFIX."users (userRole,kennung,name,vorname,
    email,tel,lastLdapUpdate) VALUES (?,?,?,?,?,?,?)";
      $query = $db->prepare($sql);
      $query->execute(array($userRole,$user->cn,$user->sn,$user->givenname,
    $user->mail,$user->telephonenumber,$stamp));
      $insertCounter++;
    }
    else {  // User existiert -> Update
      $sql = "UPDATE ".DBPREFIX."users SET userRole=?, name=?, vorname=?,
    email=?, tel=?, lastLdapUpdate=? WHERE kennung=?";
      $query = $db->prepare($sql);
      $query->execute(array($userRole,$user->sn,$user->givenname,$user->mail,
    $user->telephonenumber,$stamp,$user->cn));
      $updateCounter++;
    }
    $generalCounter++;
  }
  return "$userRole / $generalCounter Items / updated: $updateCounter / new:
    $insertCounter";
}

$syncUsers = ldapSync ($db,$adldap,$adldap->group()->members(
    $ldapGroupNameUser),"USER");
writeSyslog($db,NULL,NULL,'ldapSync','OK',$syncUsers);

$syncCsirt = ldapSync ($db,$adldap,$adldap->group()->members(
    $ldapGroupNameCsirt),"CSIRT");
writeSyslog($db,NULL,NULL,'ldapSync','OK',$syncCsirt);

$adldap->close();
```

6.2.3. IT-Sicherheit

Die im Konzept beschriebenen Aspekte der IT-Sicherheit werden – soweit dies im Rahmen der Entwicklungsumgebung möglich ist – bei der Implementierung berücksichtigt. Es wird beispielsweise grundsätzlich darauf geachtet, dass keine schützenswerten Daten als `GET`-Parameter in der URL übertragen werden. Da die Implementierung von Benutzerauthentifikation und Sitzungsverwaltung bereits im vorigen Abschnitt ausführlich beschrieben wurde, wird im Folgenden die Umsetzung weiterer Sicherheitsaspekte betrachtet.

Obwohl die Entwicklungsumgebung nicht über ein SSL-Zertifikat verfügt, wird dennoch das HTTPS-Protokoll verwendet. Die Datei `/inc/basics.php` erzwingt eine verschlüsselte Verbindung, sodass HTTP-Aufrufe automatisch in HTTPS-Requests umgewandelt werden. Wie aus Listing 6.6 ersichtlich ist, wird dabei in der ersten Zeile anhand der PHP-Servervariable geprüft, ob *keine* HTTPS-Verbindung besteht. Ergibt die Auswertung der if-Abfrage `FALSE`, besteht bereits eine verschlüsselte Verbindung, sodass diesbezüglich keine Aktionen erforderlich sind. Liefert die if-Abfrage hingegen ein `TRUE` zurück, so ist eine Umleitung der HTTP-Adresse auf die korrespondierende HTTPS-Adresse erforderlich. Hierfür wird in Zeile 3 die HTTPS-URL zusammengesetzt. In den Zeilen 5 bis 7 werden eventuell vorhandene GET-Parameter an die neue URL angehängt. Schließlich wird in Zeile 10 der Aufbau einer verschlüsselten Verbindung durch Umleitung auf die zuvor generierte HTTPS-URL in die Wege geleitet, sofern bislang noch keine anderen Header gesendet wurden.

Listing 6.6: Erzwingung einer HTTPS-Verbindung (`/inc/basics.php`)

```
1  if (!(isset($_SERVER['HTTPS']) && ($_SERVER['HTTPS'] == '1' || strtolower(
       $_SERVER['HTTPS'])=='on'))){
2
3    $url = 'https://'.$_SERVER['HTTP_HOST'].$_SERVER['PHP_SELF'];
4
5    if(!empty($_SERVER['QUERY_STRING'])) {
6      $url .= '?'.$_SERVER['QUERY_STRING'];
7    }
8
9    if(!headers_sent()) {
10     header('Location: '.$url);
11   }
12 }
```

Um das Aufrufen von Systemdateien außerhalb des zulässigen Kontexts und das Einbinden dieser in fremde Skripte zu verhindern, werden die Verzeichnisse `/inc`, `/tmp` und `/tpl` mit einem `.htaccess`-Verzeichnisschutz versehen. Dadurch werden externe Direktzugriffe u.a. auf die Konfigurationsdatei, die Funktionsbibliotheken und die HTML-Templates sowie deren temporäre Cachingdateien unterbunden. Listing 6.7 zeigt, wie sich die Verzeichnisinhalte durch den Einsatz von `.htaccess`-Dateien vor externen Zugriffen schützen lassen. Da sich die `DENY FORM ALL` Anweisung nur auf externe Zugriffe bezieht, können interne Skripte der Webanwendung weiterhin auf die geschützten Dateien zugreifen.

Listing 6.7: `.htaccess`-Verzeichnisschutz

```
1 order deny,allow
2 deny  from all
```

Zur Reduzierung der Sicherheitsrisiken von Cross-Site-Scripting (XSS) und (SQL)-Injections werden alle Benutzereingaben einer serverseitigen Filterung unterzogen. Die Implementierung dieser Sicherheitsmaßnahme (vgl. Listing 6.8) erfolgt vor der Weiterverarbeitung der unbehandelten Daten an zentraler Stelle in der Datei `/inc/basics.php`. Da die Benutzereingaben seitens PHP im `POST`-Array und `GET`-Array gespeichert werden, werden beide Arrays mit der Standard-PHP-Funktion `filter_input_array` rekursiv durchlaufen. Der dabei angewandte `FILTER_SANITIZE_STRING` Filter macht die Benutzereingaben unschädlich, indem er potenziell gefährliche Zeichenfolgen entfernt und Sonderzeichen wie Hochkommata maskiert. Die gefilterten Daten werden in die ursprünglichen Arrays zurückgespeichert und können nun komfortabel und gefahrlos weiterverarbeitet werden.

Listing 6.8: Serverseitige Eingabefilterung

```
1 $_GET  = filter_input_array(INPUT_GET, FILTER_SANITIZE_STRING);
2 $_POST = filter_input_array(INPUT_POST, FILTER_SANITIZE_STRING);
```

Darüber hinaus wird zum Schutz der Anwendungslogik jede Aktion mit einem Session-Token autorisiert. Listing 6.9 fasst die relevanten Codeabschnitte zusammen. Da sich der Token von der dem Browser bekannten Session-ID unterscheiden soll, wird er während des Loginvorgangs entsprechend der ersten Zeile als 32-stelliger MD5-Hash einer eindeutigen Zufalls-ID generiert. Er wird in einer Session-Variable gespeichert und bleibt für die Dauer der Session unverändert. Um bei der Übermittlung eines Formulars die Korrektheit des Tokens prüfen zu können, muss dieses mit den Formulardaten übergeben werden. Zu diesem Zweck enthält jedes Formular ein verstecktes Token-Feld entsprechend der Darstellung in Zeile 3. Die Moduldatei, welche die abgesendeten Formulardaten als `POST`-Array empfängt, ruft in Zeile 5 die Funktion `checkToken` auf und übergibt den empfangenen Token als einzigen Parameter an die Funktion. Diese vergleicht in Zeile 8 den übergebenen Token mit dem in einer Session-Variable gespeicherten Token. Stimmen die beiden Token überein, gibt die Funktion `TRUE` zurück. Falls nicht, wird in Zeile 10 mit der Funktion `setAlert` eine Fehlermeldung gesetzt. In Zeile 11 wird der Anwender zur zuletzt aufgerufenen Seite umgeleitet, deren Adresse der Session-Variable `lastUrl` entnommen wird, woraufhin die Ausführung des Skripts beendet wird. Der Token bleibt so lange erhalten, bis die Session abläuft oder die Sitzung während des Logoutvorgangs – wie in den Zeilen 17 und 18 dargestellt – geschlossen wird.

Listing 6.9: Implementierung des Session-Tokens

```
1  $_SESSION['token'] = md5(uniqid(rand(), true));
2
3  <input name="token" value="{$token}" type="hidden" />
4
5  checkToken($_REQUEST['token']);
6
7  function checkToken($token){
8    if ($token == $_SESSION['token']) {return TRUE;}
9    else {
10     setAlert('error','Der Token ist ungültig.');
11     header("Location: ".$_SESSION['lastUrl']);
12     exit;
13     return FALSE;
14   }
15 }
16
17 session_unset();
18 session_destroy();
```

Da die Webanwendung aus Gründen der Informationssicherheit nicht von mobilen Endgeräten aus bedient werden soll, wird am Anfang der Login-Seite `index.php` (vgl. Listing 6.10) ein Mechanismus zur Erkennung von Smartphones und Tablets implementiert. Hierfür wird zunächst die Klasse `mobileDetect`[1] geladen und ein neues Objekt dieser Klasse erstellt. Die if-Abfrage in Zeile 3 prüft, ob der Seitenaufruf durch ein mobiles Gerät erfolgt ist. Liefert diese `TRUE` zurück, so erfolgt in Zeile 4 eine Umleitung ins Verzeichnis „mobile“. Zeile 5 beendet die Ausführung der PHP-Datei. Die am Endgerät dargestellte Datei `/mobile/index.php` informiert den Nutzer über den Grund der Zugriffssperre. Liefert die if-Abfrage `FALSE` zurück, wurde kein mobiles Endgerät erkannt, sodass mit der Ausführung der nächsten Skriptzeilen fortgefahren wird.

Listing 6.10: Unterbindung der Nutzung auf mobilen Endgeräten (`/index.php`)

```
1 require_once("inc/mobileDetect/Mobile_Detect.php");
2 $detect = new Mobile_Detect;
3 if ( $detect->isMobile() ) {
4   header("Location: /mobile");
5   exit;
6 }
```

Eine Prüfung des Endgeräts genügt auf der Login-Seite, da diese Seite vom Anwender in jedem Fall besucht werden muss, um eine Nutzersitzung initiieren zu können. Im Falle eines Direktzugriffs beispielsweise auf die Datei `dashboard.php`, stellt die Anwendung fest, dass der Zugriff mangels Session nicht autorisiert ist und leitet den Anwender zur die Login-Seite um. Dort wird dann die Endgeräteprüfung vorgenommen, woraufhin der Nutzer im Falle eines mobilen Zugriffs nicht die Login-Maske, sondern den in Abbildung 6.4 dargestellten Hinweis bekommt.

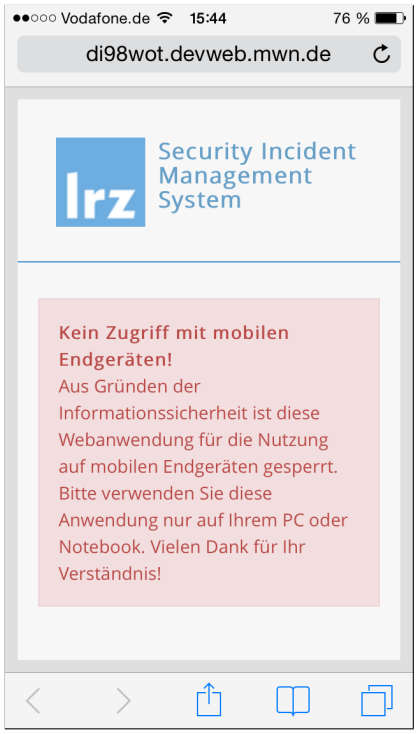

Abbildung 6.4.: Screenshot: Sperrung von mobilen Endgeräten

[1] `http://mobiledetect.net/`

6.2.4. Benutzeroberfläche

Obwohl der Schwerpunkt der Arbeit nicht auf der grafischen Benutzerschnittstelle liegt, so wird dennoch eine professionelle, intuitiv bedienbare Benutzeroberfläche erwartet. Die Implementierung der im Konzept beschriebenen Aspekte erfolgt daher auf Basis einer modernen Designvorlage, für deren Nutzung eine Lizenz erworben wurde. `Ace`[2] ist ein leichtgewichtiges, auf Bootstrap 3 basierendes GUI-Framework für Webanwendungen, das trotz einer hohen Funktionsvielfalt eine komfortable Nutzung erlaubt. Dieses Rahmenwerk stellt zahlreiche Elemente für den Aufbau und die Gestaltung des Userinterfaces bereit. Dazu zählen neben Tabellen, Widgets und Formularen auch Icons, typografische Elemente und die Navigationsleiste. Die Benutzeroberfläche der Webanwendung besteht insgesamt aus 36 HTML-Dateien. Die Masken enthalten Platzhalter, die beim Aufruf durch die Template-Klasse `rainTPL`[3] dynamisch mit Nutzdaten befüllt werden. Abbildung 6.5 zeigt die gerenderte Darstellung des Dashboard Templates (`/tpl/dashboard.html`).

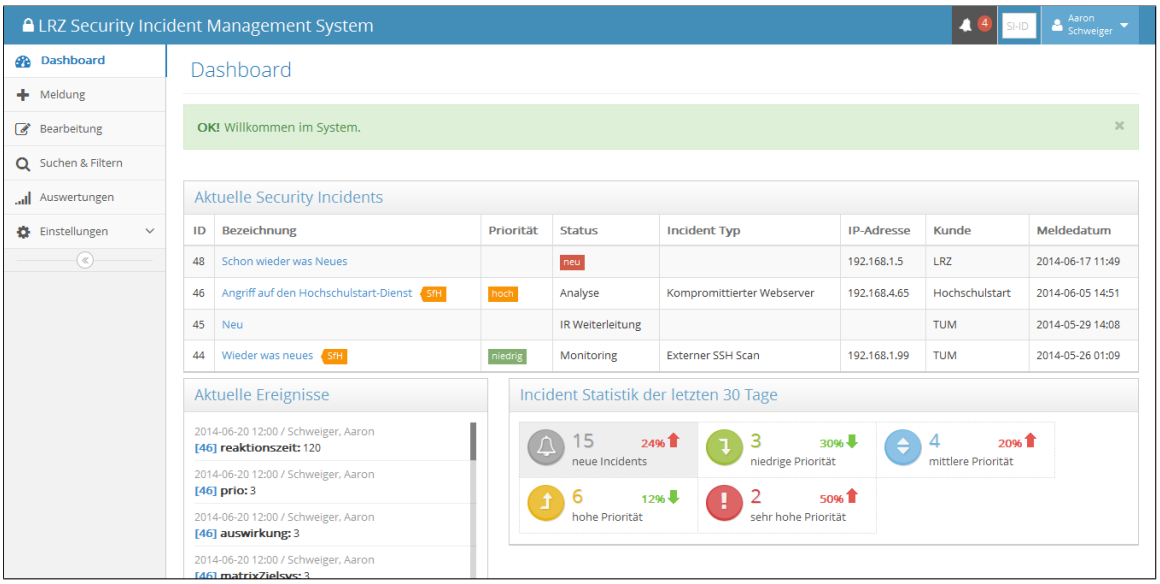

Abbildung 6.5.: Screenshot: Dashboard (`/dashboard.php`)

Wie im Konzept vorgesehen, beinhaltet die mit `Ace`-Elementen kreierte Benutzeroberfläche eine Kopfzeile, eine einklappbare Menüleiste und einen großflächigen variablen Bereich für die individuellen Bestandteile der jeweiligen Maske. Die Navigationsleiste wird in Abhängigkeit von der `userRole` des eingeloggten Anwenders und der aufgerufenen Seite in einer for-Schleife (`/inc/tpl.nav.php`) dynamisch erzeugt und als HTML-Code an das Template übergeben. Auch bei der Zusammenstellung der Kopfzeile wird die Benutzerrolle berücksichtigt, da die beiden Schnellzugriff-Elemente nur für die CSIRT-Mitglieder und den Root-User sichtbar sein sollen. Durch Klick auf das Glockensymbol lässt sich eine Übersicht der neuen und gerade in Bearbeitung befindlichen Vorfälle sowie deren Bearbeitungsfortschritt einblenden. Wählt man dort einen Vorfall aus, so wird man vom Incident Router (`/si-router.php`) automatisch auf die zur Prozessphase passenden Seite geleitet. Für die Routingentscheidung wird

[2] `https://wrapbootstrap.com/theme/ace-responsive-admin-template-WB0B30DGR`
[3] `http://www.raintpl.com/Documentation/`

der aktuelle Status des Vorfalls aus der Datenbank abgefragt. Mit einer switch-Anweisung wird die zum Status gehörende Zielseite ermittelt. Das in der Kopfzeile angrenzende Schnellzugriffsfeld ermöglicht das gezielte Aufrufen eines Vorfalls durch Eingabe seiner ID. Da diese Funktion zum effizienten Zugriff auf bereits abgeschlossene Vorfälle konzipiert ist, wird statt des Routers direkt die Statusseite aufgerufen.

Um den Anwender über den Erfolg oder Misserfolg der von ihm durchgeführten Aktion zu informieren, ermöglicht die Benutzeroberfläche die Ausgabe von Systemmeldungen. Die in Screenshot 6.5 grün hinterlegte Meldung signalisiert den erfolgreichen Abschluss der Login-Aktion. Derartige Systemmeldungen können durch die in Listing 6.11 abgebildete Funktion `setAlert` erstellt werden. Falls es sich um eine Erfolgsmeldung handelt, so wird in Zeile 2 die Session-Variable `$_SESSION['alert_ok']` mit dem Meldungstext initialisiert. Im Falle einer Fehlermeldung wird mit der Session-Variable `$_SESSION['alert_error']` analog verfahren.

Listing 6.11: Funktion `setAlert` (`/inc/functions.php`)

```
1 function setAlert($type,$text){
2   if($type == "ok") {$_SESSION['alert_ok'] = $text; return TRUE;}
3   elseif($type == "error") {$_SESSION['alert_error'] = $text; return TRUE;}
4 }
```

Beim Aufruf der nächsten Seite wird gemäß Listing 6.12 zunächst geprüft, ob die Session-Variable `$_SESSION['alert_ok']` gesetzt ist (Zeile 1). Trifft dies zu, so wird in Zeile 2 der HTML-Code mit dem Meldungstext in der lokalen Variable `$alert` gespeichert. Eine eventuelle Fehlermeldung wird in den Zeilen 5 und 6 analog behandelt. Anschließend wird der Inhalt der `$alert` Variable an das Template übergeben (Zeile 9). Abschließend werden in den Zeilen 10 und 11 durch das Rücksetzen der beiden Session-Variablen eventuell vorhandene Meldungen gelöscht, damit eine bereits angezeigte Meldung nach einem Seitenwechsel nicht erneut ausgegeben wird. Die Integration der Systemmeldungen in die Benutzeroberfläche erfolgt durch den Aufruf {`$parsedAlert`} in der Datei `/tpl/header.html`.

Listing 6.12: Ausgabe von Systemmeldungen (`/inc/tpl.alert.php`)

```
1 if (isset($_SESSION["alert_ok"])){
2   $alert .= "<div class='alert alert-block alert-success'><button class='close
      ' data-dismiss='alert' type='button'><i class='icon-remove'></i></button><
    strong>OK! </strong>".$_SESSION["alert_ok"]."</div>";
3 }
4
5 if (isset($_SESSION["alert_error"])){
6   $alert .= "<div class='alert alert-block alert-danger'><button class='close'
      data-dismiss='alert' type='button'><i class='icon-remove'></i></button><
    strong>Fehler! </strong>".$_SESSION["alert_error"]."</div>";
7 }
8
9 $tpl->assign('parsedAlert',$alert);
10 unset($_SESSION["alert_ok"]);
11 unset($_SESSION["alert_error"]);
```

6.2.5. Prozessunterstützung

Im Folgenden wird auf die Umsetzung prozessunterstützender Maßnahmen eingegangen. Diese umfassen die Sicherstellung des ordnungsgemäßen Prozessablaufs, die Zeitmessung während allen Phasen, eine manipulationssichere Dokumentation aller Aktivitäten sowie die Erweiterung der Benutzeroberfläche um eine Incident Menüleiste und eine Sidebar.

Der Wechsel in die nächste Prozessphase wird durch den Anwender ausgelöst. Hierzu sendet der Anwender das in Abbildung 6.6 dargestellte Formular ab. Standardmäßig ist der aktuelle Zeitstempel hinterlegt. Dieser kann jedoch im Falle einer Nachtragung geändert werden.

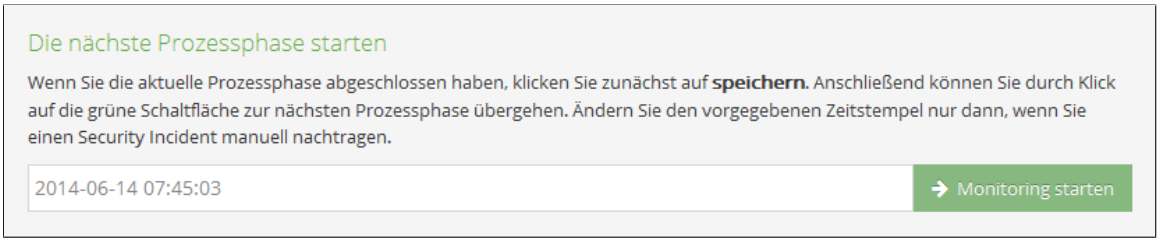

Abbildung 6.6.: Screenshot: Statuswechsel

Durch Klick auf den grünen Button wird der Zeitstempel an die Aktion `updateState` (vgl. Listing 6.13) übergeben. In Zeile 1 wird durch Aufruf der `checkState` Funktion sichergestellt, dass diese Aktion nur ausgeführt werden kann, wenn sich der Vorfall entweder den Status Klassifikation, Analyse, Lösung oder Monitoring aufweist. Anschließend wird in Zeile 2 die `updateState` Funktion aufgerufen, die den Wechsel der Prozessphase durch Änderung des Incidentstatus vornimmt.

Listing 6.13: Aktion `updateState` (`/actions/incidents.php`)

```
1 checkState($db,$_POST['si'],"CLASS,ANAL,SOL,MON");
2 updateState($db,$_POST['si'],$_POST['stamp']);
```

Durch die Auslagerung des Statuswechsels in eine separate Funktion können auch andere Aktionen – wie beispielsweise `classifyIncident`, `closeIncident` und `reopenIncident` – den Übergang in die nächste Prozessphase herbeiführen. Diese für die Anwendungslogik sehr zentrale Funktion `updateState` (vgl. Listing 6.14) erwartet als Inputparameter lediglich den Datenbankhandler, die ID des Vorfalls sowie den Zeitstempel. Um eine potenzielle Manipulation der Anwendungslogik zu verhindern, ist weder die Übergabe des aktuellen noch des Folgestatus möglich. Stattdessen ermittelt die Funktion in den Zeilen 3 und 4 den aktuellen Status und den Beginn der aktuellen Prozessphase durch Anwendung der `getIncidentValue` Funktion aus der Datenbank. In den Zeilen 7 bis 21 findet eine Fehlerbehandlung hinsichtlich des übergebenen Zeitstempels statt. Ist der Zeitstempel formal nicht korrekt oder kleiner als der Zeitstempel zu Beginn der Prozessphase, so wird die Ausführung der Funktion beendet, wobei der Anwender unter Angabe einer Fehlermeldung zur vorigen Seite zurückgeleitet wird. Der Switch-Block (Zeilen 23 bis 36) definiert in Abhängigkeit vom aktuellen Incidentstatus den nächsten Status `$newState` gemäß Prozessmodell, die zugehörige Zielseite `$nextFeature` sowie den Zeitzähler `$saldoId`, dem die aktuelle Prozessphase zuzurechnen ist. Dabei entspricht `saldo1` der Reaktionszeit, `saldo2` der Analysezeit, `saldo3` der Lösungszeit und `saldo4` der Nachlaufzeit.

Listing 6.14: Funktion `updateState` (`/inc/functions.si.php`)

```
function updateState($db,$si,$stamp){
  global $appConfig;
  $currentState = getIncidentValue($db,$si,"currentState");
  $currentStamp = getIncidentValue($db,$si,"currentStamp");
  if ($stamp == FALSE){$stamp = date("Y-m-d H:i:s",time());}

  if(strtotime($stamp) === FALSE){
    $db = null;
    setAlert('error',"Der Status wurde nicht aktualisiert, da der Zeitstempel ungültig ist.");
    header("Location: ".$_SESSION['lastUrl']);
    exit;
    return FALSE;
  }

  if (strtotime($stamp) <= strtotime($currentStamp)) {
    $db = null;
    setAlert('error',"Der Status wurde nicht aktualisiert, da der Zeitstempel größer als zu Beginn der letzten Prozessphase sein muss.");
    header("Location: ".$_SESSION['lastUrl']);
    exit;
    return FALSE;
  }

  switch ($currentState){
    case "NEW": $newState = "CLASS"; $saldoId = "saldo1";
      $nextFeature = "si-data"; break;
    case "CLASS": $newState = "ANAL"; $saldoId = "saldo1";
      $nextFeature = "si-tasks"; break;
    case "ANAL": $newState = "SOL"; $saldoId = "saldo2";
      $nextFeature = "si-solution"; break;
    case "SOL": $newState = "MON"; $saldoId = "saldo3";
      $nextFeature = "si-monitoring"; break;
    case "MON": $newState = "CLOSED"; $saldoId = "saldo4";
      $nextFeature = "si-overview"; break;
    case "CLOSED": $newState = "ANAL"; $saldoId = "saldo4";
      $nextFeature = "si-overview"; break;
  }

  $query = "SELECT currentStamp,$saldoId,".$saldoId."bh FROM ".DBPREFIX."incidents WHERE id = '$si'";
  $stmt = $db->query($query);
  $old = $stmt->fetch(PDO::FETCH_ASSOC);

  if($currentState == "CLOSED") {
    $diff = 0;
    $diffMin = 0;
    $newSaldo = $old[$saldoId];
    $diffBh = 0;
    $diffMinBh = 0;
    $newSaldoBh = $old[$saldoId.'bh'];
  }
  else {
    $diff = strtotime($stamp) - strtotime($old["currentStamp"]);
    $diffMin = round($diff/60); // in Minuten
    $newSaldo = $old[$saldoId] + $diffMin; // in Minuten
```

```
54
55      $diffBh = diffBusinessHours($old["currentStamp"],$stamp);
56      $diffMinBh = round($diffBh/60); // in Minuten
57      $newSaldoBh = $old[$saldoId.'bh'] + $diffMinBh; // in Minuten
58    }
59
60    $sql = "UPDATE ".DBPREFIX."incidents SET currentState=?, currentStamp=?,
        $saldoId=?, ".$saldoId."bh=? WHERE id=?";
61    $query = $db->prepare($sql);
62    $query->execute(array($newState,$stamp,$newSaldo,$newSaldoBh,$si));
63    writeSyslog($db,$_SESSION["userId"],$si,"updateState","OK","$currentState ->
        $newState / $saldoId / $old[$saldoId] -> $newSaldo ($diff)");
64
65    insertHistory($db,$si,$_SESSION["userId"],"currentState",$currentState,
        $newState,$_SESSION["userName"].", ".$_SESSION["userVorname"]);
```

Da das System sowohl die tatsächliche Dauer jeder Prozessphase als auch die Dauer während der Servicezeiten berechnen soll, werden in den Zeilen 38 bis 40 die bisherigen Zeitwerte aus der Datenbank geladen. In den Zeilen 42 bis 58 findet die Berechnung der Zeitdifferenz zwischen dem übergebenen Zeitstempel und dem Zeitpunkt zu Beginn der Prozessphase statt. Falls der Incident aktuell geschlossen ist (Zeile 42), bewirkt die `updateState` Funktion das Wiedereröffnen des Vorfalls und das Zurücksetzen in die Analysephase. In diesem Fall ist jedoch keine Zeitberechnung erforderlich, da der Zeitraum, in dem der Vorfall geschlossen war, für die Performancemessung irrelevant ist, sodass die Zeitdifferenzen auf Null gesetzt werden und der neue Saldo dem alten entspricht. In allen anderen Fällen (Zeilen 50 bis 58) wird zunächst die gesamte Zeitdifferenz in Sekunden (Zeile 51) und Minuten (Zeile 52) berechnet. Der neue Saldo ergibt sich aus der Addition des alten Saldos mit der errechneten Differenz, sodass die Laufzeiten auch bei einem mehrfachen Durchlaufen der Prozessphasen korrekt erfasst werden. Analog dazu erfolgt die Berechnung der Zeitdifferenz während der Servicezeiten, wobei anstelle der Subtraktion die Funktion `diffBusinessHours` aufgerufen wird. Deren Vorgehensweise wird im Anschluss näher beschrieben. Nach der Berechnung der Phasenlaufzeit werden die neuen Werte in die Datenbank geschrieben (Zeilen 60 bis 62) und im Systemlog vermerkt (Zeile 63). Abschließend wird die Statusänderung durch die Funktion `insertHistory` im incidentbezogenen Verlaufsprotokoll dokumentiert.

Nachfolgend wird die Implementierung der zuvor aufgerufenen Funktion `diffBusinessHours` erläutert, die den Unterschied zwischen zwei Zeitstempeln in Sekunden berechnet, wobei nur Zeiten während der Business Hours berücksichtigt werden. Als Servicezeiten gelten die Zeiträume von Montag bis Donnerstag jeweils von 8 bis 18 Uhr und freitags von 8 bis 16 Uhr. Da es sich um ein komplexes Problem handelt, für dessen Lösung keine frei verfügbare PHP-Klasse auffindbar ist, wird ein eigener Algorithmus entwickelt. Abbildung 6.7 visualisiert die ermittelte Reaktionszeit von zwei Stunden für einen Incident, der freitags um 17 Uhr gemeldet und montags um 10 Uhr klassifiziert wird.

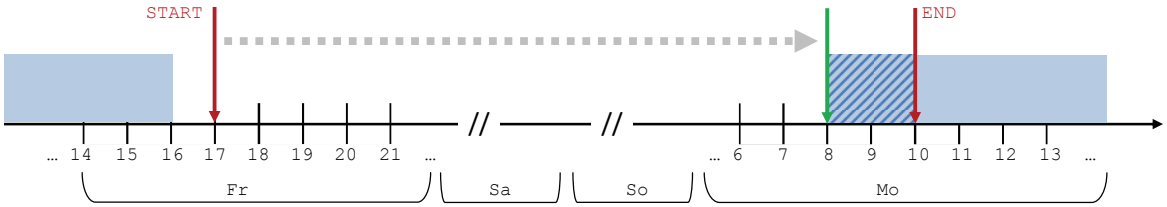

Abbildung 6.7.: Visualisierung `diffBusinessHours`

Die grundlegende Idee hinter dem Algorithmus besteht darin, das gesamte Zeitintervall in drei Abschnitte – Starttag, Zwischentage und Endtag – zu unterteilen und deren Dauer getrennt zu berechnen. Schließlich liefert die Addition der Subintervalle das Gesamtergebnis. Der Berechnungsalgorithmus geht allerdings davon aus, dass sowohl der Start- als auch der Endzeitpunkt innerhalb der Servicezeit liegen. Deshalb verschiebt der Algorithmus zu Beginn `$start` und `$end` an die Grenze der Servicezeit, falls dies erforderlich ist. Während der Implementierung wurde der Algorithmus zahlreichen Tests unterzogen, um zu gewährleisten, dass der Algorithmus auch Sonderfälle korrekt berechnet. Dazu zählen Zeitintervalle, die außerhalb der Servicezeit beginnen und enden, ebenso wie mehrwöchige Intervalle, die mehrere Freitage und Wochenenden beinhalten.

Die Funktion `diffBusinessHours` (vgl. Listing 6.15) setzt sich aus zahlreichen verschachtelten if-Anweisungen und einer for-Schleife zusammen. Die in den Zeilen 1 bis 4 anhand der Eingabeparameter ermittelten Wochentage werden benötigt, um den Endzeitpunkt an das Ende der vorigen Servicezeit zu verschieben (Zeilen 7 bis 21). Liegt das Ende auf einem Samstag oder Sonntag, so wird es in Zeile 7 auf den vorigen Freitag um 16 Uhr gesetzt. Nachdem in Zeile 9 die Uhrzeit des Endzeitpunkts in einen sechsstelligen String ohne Trennzeichen umgewandelt wurde, wird in den Zeilen 11 bis 14 der Fall betrachtet, dass das Zeitintervall morgens vor 8 Uhr endet. Handelt es sich um einen Montag, so wird das Ende auf 16 Uhr vor drei Tagen (also Freitag) justiert. An allen anderen Tagen wird das Ende auf 18 Uhr des Vortags korrigiert. Anschließend wird in den Zeilen 16 bis 21 der Fall betrachtet, dass der Zeitpunkt abends nach dem Ende der Servicezeit liegt. Da freitags die Servicezeit bereits um 16 Uhr endet, greift eine wochentagsabhängige Unterscheidung. Liegt ein Zeitpunkt an einem Freitag nach 16 Uhr, so wird der Zeitstempel auf 16 Uhr geändert. Handelt es sich um einen anderen Wochentag nach 18 Uhr, so erfolgt eine Rücksetzung auf 18 Uhr. Die Verschiebung des Startzeitpunkts an den Beginn der nächsten Servicezeit erfolgt analog dazu und ist in den Zeilen 23 bis 34 implementiert. Da sich durch die Justierung der Zeitpunkte die Wochentage geändert haben können, werden diese in den Zeilen 36 und 37 neu bestimmt.

Nach der eventuellen Verschiebung der Zeitpunkte an den Beginn bzw. das Ende der nächstliegenden Servicezeit erfolgt in den Zeilen 39 bis 55 die eigentliche Berechnung der Zeitdifferenz in Sekunden und wird in der Variable `$sec` gespeichert. Zunächst wird in Zeile 39 geprüft, ob das Datum von Start- und Endzeitpunkt identisch ist. Trifft dies zu, so wird die Zeitdifferenz durch eine einfache Subtraktion der Zeitpunkte berechnet. Liegen Start und Ende an verschiedenen Tagen, gestaltet sich die Berechnung aufwändiger: Zunächst wird in Zeile 41 die Anzahl der ganzen Tage zwischen Ende und Anfang berechnet und in der Variablen `$daysBetween` gespeichert. Ist diese positiv, so wird in den Zeilen 43 und 44 die Anzahl der Sekunden des Starttages durch Subtraktion der wochentagsabhängigen Serviceschlusszeit von der individuellen Startzeit ermittelt und in der Variable `$sec` gespeichert. Anschließend wird in den Zeilen 45 bis 51 die Anzahl der Sekunden der zwischen Start und Ende liegenden Tage berechnet und addiert. Die hierfür verwendete for-Schleife iteriert über alle Tage vom zweiten bis zum vorletzten Tag. Die in Zeile 45 mit der zahlenmäßigen Repräsentation des Start-Wochentags belegte Zählvariable `$day` wird bereits zu Beginn der Schleife inkrementiert (Zeile 47), sodass der zuvor berechnete erste Tag nicht mehr betrachtet wird. Verlässt die Zählvariable durch das Inkrementieren den gültigen Wertebereich (0-6), so wird diese in Zeile 48 auf 0 zurückgesetzt, da nach dem sechsten Tag (Samstag) kein siebter, sondern der nullte (Sonntag) folgt. Liegt der aktuell betrachtete Wochentag zwischen Montag (1) und Freitag (5), so wird in den Zeilen 49 und 50 die gesamte Servicezeit des Tages zur Variable `$sec` hinzuaddiert. Die Servicezeit beträgt freitags 8 Stunden (Zeile 50) und an den anderen Werktagen 10 Stunden (Zeile 49).

Listing 6.15: Funktion `diffBusinessHours` (`/inc/functions.php`)

```
function diffBusinessHours($start,$end){
  $start = strtotime($start);
  $startDoW = date('w',$start);
  $end = strtotime($end);
  $endDoW = date('w',$end);

  if($endDoW == 0 || $endDoW == 6){$end = strtotime('Last Friday 4pm',$end);}

  $time = date('His', $end);

  if($time < "080000"){
    if($endDoW == 1){$end = strtotime('-3days 4pm',$end);}
    else {$end = strtotime('-1day 6pm',$end);}
  }

  if($endDoW == 5){
    if($time > "160000"){$end = strtotime('4pm',$end);}
  }
  else {
    if($time > "180000"){$end = strtotime('6pm',$end);}
  }

  if($startDoW == 0 || $startDoW == 6){$start = strtotime('Next Monday 8am',
    $start);}

  $time = date('His', $start);

  if($time < "080000"){$start = strtotime('8am',$start);}

  if($startDoW == 5){
    if($time > "160000"){$start = strtotime('Next Monday 8am',$start);}
  }
  else {
    if($time > "180000"){$start = strtotime('+1day 8am',$start);}
  }

  $endDoW = date('w',$end);
  $startDoW = date('w',$start);

  if (date("Y-m-d",$start) == date("Y-m-d",$end)) {$sec = $end-$start;}
  else {
    $daysBetween = floor(($end-$start)/(24*60*60));
    if ($daysBetween > 0) {
      if($startDoW == 5){$sec = strtotime('4pm', $start) - $start;}
      else {$sec = strtotime('6pm', $start) - $start;}
      $day = $startDoW;
      for($i=1; $i < $daysBetween; $i++) {
        $day++;
        if ($day > 6){$day = 0;}
        elseif ($day > 0 && $day < 5) {$sec += (10*60*60);}
        elseif ($day == 5) {$sec += (8*60*60);}
      }
      $sec += $end - strtotime('8am', $end);
    }
    else {$sec = 0;}
  }
  return $sec;
}
```

Schließlich wird in Zeile 52 noch der Anteil des letzten Tages zur Zwischensumme `$sec` addiert, wobei der individuelle Endzeitpunkt von der täglichen Startzeit (8 Uhr) subtrahiert wird. Die else-Anweisung in Zeile 54 wird nur dann ausgeführt, wenn beide ursprünglichen Zeitpunkte am gleichen Wochenende liegen. Da die automatische Justierung in diesem Fall den Beginn in die Zukunft (Montag 8 Uhr) und das Ende in die Vergangenheit (Freitag 16 Uhr) verschiebt, sind die im if-Clause in Zeile 42 geprüften `$daysBetween` negativ. In diesem Sonderfall ist folglich keine Zeitdifferenz zu berechnen, sodass die Funktion 0 Sekunden zurückliefert.

Eine weitere im Prozesskontext besonders relevante Funktion ist die `checkTeamComplete` Funktion (vgl. Listing 6.16). Bei jedem Aufruf einer incidentbezogenen Seite wird anhand dieser Funktion geprüft, ob für den konkreten Vorfall das Security Incident Team mindestens minimal besetzt ist, also ein Hotliner und ein SIC definiert wurden. Zunächst wird in den Zeilen 2 und 3 die `getIncidentValue` Funktion zur Ermittlung der Klassifikation und des Incidentstatus angewendet. Darüber hinaus wird in Zeile 4 die Anzahl derjenigen Personen aus der Datenbank abgefragt, die als Hotliner oder SIC dem Vorfall zugeordnet sind. Da die Überprüfung der Teambesetzung im Falle eines Duplikats oder eines Information Requests obsolet ist, wird diese nach Prüfung der Klassifikation in Zeile 5 mit dem Return-Wert `TRUE` übersprungen. Da auch in den Prozessphasen `NEW` und `CLASS` die beiden Teampositionen noch nicht zwangsweise festgelegt sind, greift bei einem derartigen Incidentstatus der else-Zweig in Zeile 13, der die Funktion mit dem Rückgabewert `TRUE` verlässt. Nur dann, wenn ein Sicherheitsvorfall vorliegt, der sich mindestens in der Analysephase befindet, wird in Zeile 6 die zuvor aus der Datenbank abgerufene Personenzahl (Hotliner und SIC) betrachtet. Ist diese ungleich 2, so wird in Zeile 7 mit `setAlert` eine Aufforderung zur Teamvervollständigung erzeugt, woraufhin der Anwender zur Teamseite des Incidents umgeleitet wird. Da sich ein bereits festgelegter Hotliner oder SIC jederzeit wieder löschen lässt, reicht eine einmalige Teamprüfung nicht aus, weswegen die `checkTeamComplete` in allen Prozessphasen aufgerufen wird. Dadurch wird eine weitere Bearbeitung des Vorfalls unterbunden, solange nicht beide Teampositionen zugeordnet sind.

Listing 6.16: Funktion `checkTeamComplete` (`/inc/functions.si.php`)

```
1  function checkTeamComplete($db,$si){
2    $klassifikation = getIncidentValue($db,$si,'klassifikation');
3    $currentState = getIncidentValue($db,$si,'currentState');
4    $query = "SELECT COUNT(*) FROM ".DBPREFIX."contacts WHERE si=".$_GET['si']."
       AND contactRole IN ('HOTLINER','SIC')";
5    if ($klassifikation != "INCIDENT") {return TRUE;}
6    elseif ($currentState != "NEW" AND $currentState != "CLASS" AND $db->query(
       $query)->fetchColumn() != 2) {
7      setAlert('error',"Team unvollständig! Bitte legen Sie den Hotliner und den
       SIC fest!");
8      $db = null;
9      header("Location: /si-team.php?si=$si");
10     exit;
11     return FALSE;
12   }
13   else {return TRUE;}
14 }
```

Die in Listing 6.17 abgebildete Funktion `insertHistory` ermöglicht eine incidentbezogene Dokumentation aller Einzelschritte des Prozessverlauf. Sie ist durch den Anwender nicht direkt aufrufbar, sondern wird im Kontext anderer Aktionen und Funktionen automatisch aufgerufen. In den Zeilen 2 und 3 wird ein neuer Datensatz in der `history`-Tabelle angelegt, der neben den IDs des Incidents und des Anwenders die Bezeichnung des betroffenen Datenbankfeldes, den Wert des Feldes vor und nach der Änderung sowie den Namen des Anwenders enthält. Im Normalfall liefert die Funktion `TRUE` zurück (Zeile 9). Tritt beim schreibenden Zugriff auf die `history`-Tabelle hingegen ein Fehler auf, so wird dieser in den Zeilen 5 bis 8 behandelt. Dabei wird zunächst mit `writeSyslog` ein Eintrag im Systemlog erzeugt und anschließend die Funktion mit dem Return-Wert `FALSE` verlassen.

Listing 6.17: Funktion `insertHistory` (`/inc/functions.si.php`)

```
1  function insertHistory ($db,$si,$user,$field,$old,$new,$userName){
2    $sql = "INSERT INTO ".DBPREFIX."history (si,user,field,old,new,userName)
       VALUES ('$si','$user','$field','$old','$new','$userName')";
3    $query = $db->query($sql);
4
5    if(!$query){
6      writeSyslog($db,$user,$si,'insertHistory','ERROR','DB: '.var_dump($db->
       errorInfo()));
7      return FALSE;
8    }
9    else {return TRUE;}
10 }
```

Dadurch, dass der Aufruf der Funktion `insertHistory` fest im Code der incidentbezogenen Aktionen bzw. Funktionen verankert ist und nicht durch das Setzen spezieller Parameter umgangen werden kann, ist sichergestellt, dass jeder Vorgang zuverlässig und unverfälscht anhand der Originaldaten dokumentiert wird. Da die Webanwendung weder Funktionen zum Löschen und Ändern noch zum direkten, ereignisunabhängigen Hinzufügen von Historyeinträgen bereitstellt, ist die Manipulationssicherheit des Verlaufsprotokolls gewährleistet.

Um den Prozessverlauf für die CSIRT-Mitglieder möglichst komfortabel und effizient zu gestalten, wird die Benutzeroberfläche im Bereich der Vorfallbearbeitung um eine zusätzliche Incident Menüleiste sowie eine Sidebar erweitert (vgl. Abbildung 6.8). Die technische Umsetzung richtet sich nach den Vorgaben des Konzepts und kann in den Dateien `/inc/tpl.si.nav.php` bzw. `/tpl/si-sidebar.html` eingesehen werden. Die auf dem Screenshot abgebildete Incidentstatusseite (`/si-overview.php`) fasst alle zu einem Vorfall vorliegenden Informationen auf einer Seite übersichtlich zusammen und beinhaltet auch Metainformationen wie die Laufzeiten der einzelnen Prozessphasen. Die Statusseite ist flexibel implementiert, sodass auch Duplikate und Information Requests dargestellt werden können. Zudem ist bei Sicherheitsvorfällen neben der reduzierten Standardversion eine erweiterte Fassung abrufbar.

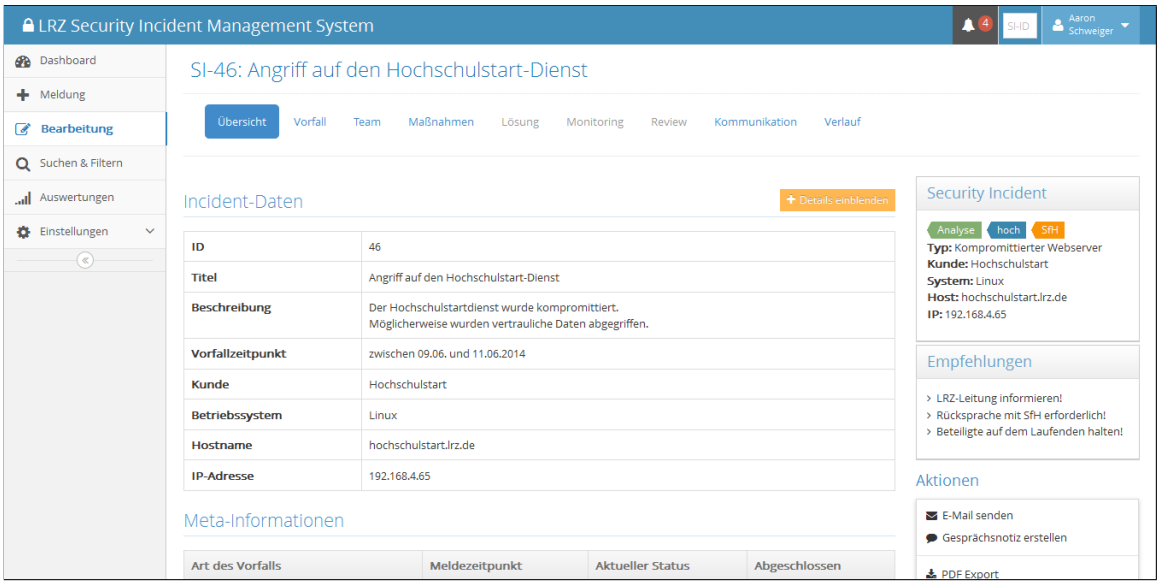

Abbildung 6.8.: Screenshot: Incidentstatusseite (`/si-overview.php`)

6.2.6. Kommunikation

Da eine reibungslose Kommunikation die effiziente Incidentbearbeitung begünstigt, stellt die Webanwendung ein voll integriertes Kommunikationsmodul bereit. Dieses unterstützt einerseits verschiedene Arten des Mailversands (`/si-com-mail.php`), wobei die Kontaktdaten der Teammitglieder als Adressbuch zur Verfügung stehen, und andererseits die Erfassung von Gesprächsnotizen und eingegangenen E-Mails (`/si-com-memo.php`). Dabei werden sämtliche Vorgänge mit der Funktion `addComItem` automatisch in einem incidentbezogenen Kommunikationsprotokoll (`/si-com.php`) dokumentiert.

Für den tatsächlichen Versand von E-Mails baut die Klasse PHPMailer[4] eine SSL/TLS verschlüsselte Verbindung zu einem SMTP-Server auf, sodass die Übertragung der sensiblen Daten über einen gesicherten Kommunikationskanal gewährleistet ist. Die Webanwendung unterscheidet im Bereich des Mailversands drei Anwendungsfälle:

- Ist beispielsweise eine Anfrage weiterer Informationen an den Vorfallmelder zu richten, da die gemeldeten Daten unvollständig sind, so wird für den Mailversand die Funktion `mailTo Contact` verwendet. Diese Funktion eignet sich auch für E-Mails an ein anderes Mitglied des Security Incident Teams, das aus dem adressbuchartigen Dropdown-Feld (`parseSiContactsDropdown`) komfortabel ausgewählt werden kann. Anhand der selektierten `contactId` werden die für den Mailversand benötigten Empfängerdaten aus der `contacts`-Tabelle abgerufen.
- Mit der Funktion `mailToSomebody` können E-Mails an einen beliebigen Empfänger – beispielsweise an einen externen IT-Forensiker – gesendet werden. In diesem Fall werden die Empfängerdaten als Inputparameter an die Funktion übergeben.
- Die dritte Funktion `mailToGroup` ermöglicht das Senden einer E-Mail an mehrere Empfänger, beispielsweise an alle Mitglieder des Security Incident Teams. Hierbei sind die Empfängerdaten in Form eines mehrdimensionalen Arrays, das alle Namen und E-Mail-Adressen enthält, an die Funktion zu übergeben.

[4] `https://github.com/PHPMailer/PHPMailer`

Ereignisbedingte E-Mails wie die Benachrichtigung des Vorfallmelders über den Abschluss seines Incidents werden durch Aufruf einer der genannten Mailfunktionen innerhalb des jeweiligen Aktionskontextes (hier: `closeIncident`) ausgelöst. Der Versand zeitlich gesteuerter E-Mails wird hingegen durch einen Cronjob veranlasst.

Für den Versand individueller E-Mails über die Webanwendung ist die Seite (`/si-com-mail.php`) aufzurufen. Nach der Auswahl des Empfängers (Adressbuch, Gruppe, manuelle Eingabe) sind der Betreff und der E-Mail-Inhalt einzugeben. Außerdem besteht die Möglichkeit, durch Checkbox-Auswahl verschiedene Vorfallinformationen anzufügen: Basisdaten, Kontaktdaten, Analyse, Lösung, Monitoring, Review. Mit dem Absenden des Formulars wird die Aktion `prepareMail` aufgerufen, welche die ausgewählten Informationen aus der Datenbank abfragt und am Ende der Nachricht einfügt. Der Anwender kann die angezeigte E-Mail-Vorschau vor dem endgültigen Mailversand (`sendPreparedMail`) editieren, um beispielsweise einzelne Daten zu entfernen, die für den Empfänger nicht relevant sind. Welche der drei Mailfunktionen anzuwenden ist, entscheidet das System zur Laufzeit in Abhängigkeit vom Empfängertyp.

Darüber hinaus ermöglicht das Kommunikationsmodul auch die Meldung eines Incidents an die Allianz für Cyber-Sicherheit. Die Seite `/si-com-acs.php` enthält eine Nachbildung des offiziellen Meldeformulars, das bereits soweit wie möglich vorausgefüllt ist. Dieses Formular umfasst acht Fragestellungen, die im Wesentlichen durch das Anklicken von Checkboxen zu beantworten sind. Im Vergleich zur Nutzung des externen Meldeformulars reduziert die anwendungsinterne Meldung den Umfang manuell einzugebender Informationen und somit den Zeitaufwand. Darüber hinaus ist eine automatische Dokumentation der Meldung im Kommunikationsprotokoll gewährleistet. Auf technischer Ebene betrachtet, werden die Formulareingaben an die Aktion `sendAcsMail` übermittelt. Diese generiert anhand der Eingaben den Mailinhalt und übergibt diesen an die Funktion `mailToSomebody`, wobei die Empfängerinformationen aus der Konfigurationsdatei stammen.

Die Vorgehensweise der Mailfunktionen wird am Beispiel `mailToSomebody` (vgl. Listing 6.18) vorgestellt. Für den Versand der E-Mail via `PHPMailer` und die Dokumentation des Vorgangs in der `communications`-Tabelle erwartet die Funktion neun Eingabeparameter:

- `$db` – Handler für Zugriff auf die `communications`-Tabelle
- `$si` – ID des Sicherheitsvorfalls zur Zuordnung der versendeten E-Mail
- `$name` – Name des Empfängers
- `$email` – E-Mail-Adresse des Empfängers
- `$contactRole` – Funktion des Empfängers
- `$subject` – Betreff der E-Mail
- `$body` – Inhalt der E-Mail
- `$senderData` – Absender: `SYSTEM` oder `SESSION`. Ist als Absender `SYSTEM` eingetragen, so wird der in der Konfigurationsdatei festgelegte Standardabsender verwendet. Die Option `SESSION` bewirkt, dass die in der Session gespeicherten Daten des agierenden Benutzers als Absenderdaten verwendet werden.
- `$autoGreeting` – Flag: Gibt an, ob eine automatische Grußformel an den Mailtext angehängt werden soll. Dieses Flag ist standardmäßig gesetzt und muss daher nicht zwingend an die Funktion übergeben werden.

Zunächst wird in Zeile 3 ein neues `simsMailer` Objekt der Klasse `PHPMailer` erzeugt. Anschließend werden die Absenderdaten und die Grußformel festgelegt. Falls das System als Absender fungieren soll (Zeile 5), gelten automatisch die in der Konfigurationsdatei hinterlegten Daten, sodass in Zeile 6 nur die Grußzeile definiert wird. Soll hingegen der agierende Nutzer als Absender auftreten (Zeile 8), beinhaltet die Grußformel dessen Vor- und Nachnamen (Zeile 9). In den Zeilen 10 und 11 werden die E-Mailadresse und der Name des Anwenders – sofern vorhanden – als Absenderdaten an das `simsMailer` Objekt übergeben. Ist der Name oder die E-Mail-Adresse des Anwenders nicht verfügbar, so werden automatisch die Standarddaten aus der Konfigurationsdatei verwendet.

In Zeile 14 wird der übergebene Betreff um ein Präfix (ID des Vorfalls) ergänzt. Dadurch wird sichergestellt, dass die E-Mail auf den ersten Blick einem Vorfall zugeordnet werden kann. Falls das `$autoGreeting` Flag gesetzt ist, wird in Zeile 15 der Mailbody zusammengesetzt, wobei eine neutrale Begrüßung vorangestellt und die zuvor definierte Grußformel angehängt wird. Andernfalls bleibt der E-Mail-Inhalt unverändert (Zeile 16). Anschließend werden die E-Mail-Bestandteile Empfänger, Betreff und Inhalt in den Zeilen 17 bis 19 an das `simsMailer` Objekt übergeben. Der E-Mail-Versand erfolgt in Zeile 21. Ist dieser erfolgreich, so wird die ausgehende E-Mail mit dem aktuellen Zeitstempel mittels `addComItem` incidentbezogen in der `communications`-Tabelle gespeichert (Zeile 22) und ist somit für alle CSIRT-Mitglieder im Kommunikationsprotokoll des Vorfalls einsehbar. Im Erfolgsfall liefert die Funktion `TRUE`, im Fehlerfall `FALSE` zurück, wobei die Fehlerbehandlung in der aufrufenden Aktion erfolgt.

Listing 6.18: Funktion `mailToSomebody` (`/inc/functions.si.php`)

```
1  function mailToSomebody($db,$si,$name,$email,$contactRole,$subject,$body,
       $senderData,$autoGreeting=TRUE){
2    global $appConfig;
3    $mail = new simsMailer;
4
5    if($senderData == "SYSTEM") {
6      $greeting = "\n\nBeste Grüße von Ihrem\n\nSecurity Incident Management
       System";
7    }
8    else {
9      $greeting = "\n\nBeste Grüße\n\n".$_SESSION['userVorname']." ".$_SESSION['
       userName']."\nComputer Security Incident Response Team (CSIRT)";
10     if ($_SESSION["userEmail"] != ""){$mail->From = $_SESSION["userEmail"];}
11     if ($_SESSION["userName"] != ""){$mail->FromName = $_SESSION["userVorname"
       ]." ".$_SESSION["userName"]." (".$appConfig['sysname']." )";}
12   }
13
14   $newSubject = "[SI-$si] ".$subject;
15   if($autoGreeting == TRUE) {$newBody = "Hallo,\n\n".$body.$greeting;}
16   else {$newBody = $body;}
17   $mail->addAddress($email,$name);
18   $mail->Subject = $newSubject;
19   $mail->Body = $newBody;
20
21   if($mail->send()){
22     addComItem($db,$si,$senderData,date("Y-m-d H:i:s",time()),"MAIL","OUT",
       $newSubject,$newBody,$name,null,$email,$contactRole);
23     return TRUE;
24   }
25   else {return FALSE;}
26 }
```

6.3. Erfassung von Sicherheitsvorfällen

Für die Erfassung eines neuen Sicherheitsvorfalls wird in der Datei `/new.php` ein zweigeteiltes Formular bereitgestellt. Während im oberen Teil Informationen zum Vorfall einzutragen sind, umfasst der untere Teil die Kontaktdaten des Vorfallmelders. Das Formular besteht aus einzeiligen Textfeldern, mehrzeiligen Textareas sowie Dropdown-Feldern. Letztere dienen der Auswahl des betroffenen Kunden und des Betriebssystems. Die Listen der Kunden und Betriebssysteme werden durch die Funktion `getTypesArray` ermittelt und an die Funktion `parseDropdown` zur dynamischen Generierung der Dropdown-Felder übergeben.

Aufgrund der LDAP-Synchronisation stehen die Kontaktdaten des Vorfallmelders in der `users`-Tabelle bereit. Folglich werden die Kontaktinformationen zur Vorbelegung der Formularfelder aus der Datenbank abgefragt und an das Template übergeben. Dadurch, dass nur noch optionale Angaben zur Handynummer und Erreichbarkeit zu machen sind, reduziert sich der manuelle Eingabeaufwand von Personeninformationen erheblich. Pflichtfelder wie der Titel und die Beschreibung des Vorfalls werden im HTML-Code mit dem Parameter `required` versehen. Dies bewirkt eine browserseitige Prüfung dieser Felder beim Absenden des Formulars. Der Meldezeitpunkt und die Benutzerkennung sind durch die Template-Anweisung {`if = '$userRole == USER'` } `readonly` {`/if`} für LRZ-Mitarbeiter nicht editierbar. CSIRT-Mitglieder hingegen können beispielsweise bei der nachträglichen Erfassung eines telefonisch gemeldeten Vorfalls einen abweichenden Meldezeitpunkt und die Kennung des meldenden Kollegen eintragen.

Durch das Absenden des Formulars werden die eingegebenen Daten mittels `POST`-Request an die Aktion `newIncident` (vgl. Listing 6.19) übergeben. Zunächst wird in Zeile 1 eine serverseitige Prüfung der Pflichtfelder vorgenommen. Enthält eines dieser Felder keine Daten, so werden die übermittelten Daten im Rahmen einer foreach-Schleife (Zeile 2) in der Session-Variable `formerror` zwischengespeichert (Zeile 3). Mit `setAlert` wird in Zeile 6 eine Fehlermeldung generiert, woraufhin der Anwender in Zeile 7 zum Meldeformular zurückgeleitet wird. Der hierbei übergebene `GET`-Parameter `formerror=1` sorgt dafür, dass die Eingaben aus der Session-Variablen abgerufen und zur Vorbelegung der Formularfelder verwendet werden. Durch die Zwischenspeicherung gehen die zuvor gemachten Angaben nicht verloren, sodass eine Vervollständigung auf Basis des bisherigen Datenbestands möglich ist.

Ist die Prüfung der Pflichtfelder erfolgreich, so wird anhand der erfassten Daten in den Zeilen 11 bis 18 ein neuer Datensatz mit dem Status `NEW` in der `incidents`-Tabelle erzeugt. Ein eventueller Datenbankfehler wird in den Zeilen 14 bis 17 behandelt, wobei der Anwender zum Meldeformular zurückgeleitet wird. Im Standardfall werden nun die Incident Daten in die `history`-Tabelle eingetragen. Da die Dokumentation feldweise atomar erfolgen soll, ruft die foreach-Schleife für jedes Feld die Funktion `insertHistory` auf (Zeilen 20 und 21). Anschließend erfolgt in den Zeilen 24 bis 26 die Speicherung der Kontaktdaten des Vorfallmelders in der `contacts`-Tabelle. Zuletzt werden die CSIRT-Mitglieder und der Vorfallmelder über den Eingang eines neuen Incidents per E-Mail benachrichtigt. Für den Versand der E-Mail an das CSIRT in den Zeilen 29 bis 31 wird die Funktion `mailToSomebody` verwendet, wobei die Empfängerdaten aus dem Array `$csirtRecipient` der Konfigurationsdatei stammen. Die Benachrichtigung des Vorfallmelders (Zeilen 33 bis 35) erfolgt hingegen mit der `mailToContact` Funktion, wobei die Empfängerdaten anhand der `$contactId` aus der `contacts`-Tabelle abgefragt werden. Abschließend wird eine Erfolgsmeldung gesetzt (Zeile 38), woraufhin der Anwender in den Zeilen 39 bis 42 in Abhängigkeit von seiner `userRole` entweder zum Dashboard (LRZ-Mitarbeiter) oder zur Incidentstatusseite (CSIRT-Mitglieder) weitergeleitet wird.

Listing 6.19: Aktion `newIncident` (`/actions/new.php`)

```php
if ($_POST['title'] == "" || $_POST['description'] == "" || $_POST['createdStamp'] == "" || $_POST['vorname'] == "" || $_POST['name'] == "" || $_POST['email'] == "" || $_POST['tel'] == "" || $_POST['kennung'] == ""){
  foreach(array("title","description","vorfallzeitpunkt","kunde","host","ip","os","createdStamp","kennung","name","vorname","email","tel","mobil","comment") as $key){
    $_SESSION['formerror'][$key] = $_POST[$key];
  }
  $db = null;
  setAlert('error','Es wurden nicht alle Pflichtfelder ausgefüllt.');
  header("Location: ".$appConfig['url']."/new.php?formerror=1"); exit;
}

$_POST['currentState'] = "NEW";
$sql = "INSERT INTO ".DBPREFIX."incidents (user,createdStamp,currentStamp,currentState,title,description,vorfallzeitpunkt,kunde,host,ip,os) VALUES (?,?,?,?,?,?,?,?,?,?,?)";
$query = $db->prepare($sql);
$query->execute(array($_SESSION['userId'],$_POST['createdStamp'],$_POST['createdStamp'],$_POST['currentState'],$_POST['title'],$_POST['description'],$_POST['vorfallzeitpunkt'],$_POST['kunde'],$_POST['host'],$_POST['ip'],$_POST['os']));
if(!$query){
  setAlert('error','Der Security Incident konnte nicht angelegt werden.');
  header("Location: ".$_SESSION['lastUrl']); exit;
}
$si = $db->lastInsertId();

foreach(array("createdStamp","currentState","title","description","vorfallzeitpunkt","kunde","host","ip","os") as $key){
  insertHistory($db,$si,$_SESSION['userId'],$key,NULL,$_POST[$key],$_SESSION['userName']." ".$_SESSION['userVorname']);
}

$sql = "INSERT INTO ".DBPREFIX."contacts (si,contactRole,kennung,name,vorname,anrede,email,tel,mobil,comment) VALUES (?,?,?,?,?,?,?,?,?,?)";
$query = $db->prepare($sql);
$query->execute(array($si,'MELDER',$_POST['kennung'],$_POST['name'],$_POST['vorname'],$_POST['anrede'],$_POST['email'],$_POST['tel'],$_POST['mobil'],$_POST['comment']));
$contactId = $db->lastInsertId();

$subject = "Neuer Security Incident eingegangen";
$body = "(...)"; // Mailinhalt zur besseren Übersichtlichkeit entfernt
mailToSomebody($db,$si,$csirtRecipient['name'],$csirtRecipient['email'],"CSIRT",$subject,$body,"SYSTEM");

$subject = "Neuer Security Incident eingegangen";
$body = "(...)"; // Mailinhalt zur besseren Übersichtlichkeit entfernt
mailToContact($db,$si,$contactId,$subject,$body,"SYSTEM");

$db = null;
setAlert('ok','Der Security Inicident mit der ID '.$si.' wurde erfasst.');
if ($_SESSION['userRole'] == "USER"){
  header("Location: ".$appConfig['url']."/dashboard.php"); exit;
}
else {header("Location: ".$appConfig['url']."/si.php"); exit;}
```

6.4. Prozesskonforme Bearbeitung von Sicherheitsvorfällen

Auf Basis der in Abschnitt 6.2.5 beschriebenen Prozessunterstützung wird im Folgenden auf die Implementierung des Bearbeitungsprozesses eingegangen. Dieser beginnt mit der initialen Klassifikation und führt – nach einem Exkurs zu Duplikaten und Information Requests – über die Ermittlung der Priorität zur Teamzusammenstellung. In der Analyse-, Lösungs- und Monitoringphase liegt der Fokus auf der Aktion `updateIncidentValues`. Daraufhin werden die Aktionen zum Schließen, Wiedereröffnen und Abbrechen eines Vorfalls beschrieben. Die Durchführung eines Post Incident Reviews rundet das Kapitel ab.

6.4.1. Initiale Klassifikation

Ein neuer Vorfall ist vor der weiteren Bearbeitung durch den CSIRT-Hotliner initial zu klassifizieren. Hierzu werden auf der Seite `/si-init.php` die Incident Informationen und die Kontaktdaten des Vorfallmelders tabellarisch dargestellt. Auf Grundlage dieser Daten entscheidet der Hotliner, ob es sich um einen neuen Sicherheitsvorfall, um ein Duplikat eines bereits gemeldeten Incidents oder um einen Information Request handelt. Zur übersichtlichen Darstellung der Möglichkeiten werden drei parallel angeordnete Widgets eingesetzt, die jeweils ein Dropdown-Feld und einen Submit-Button enthalten (vgl. Screenshot 6.9).

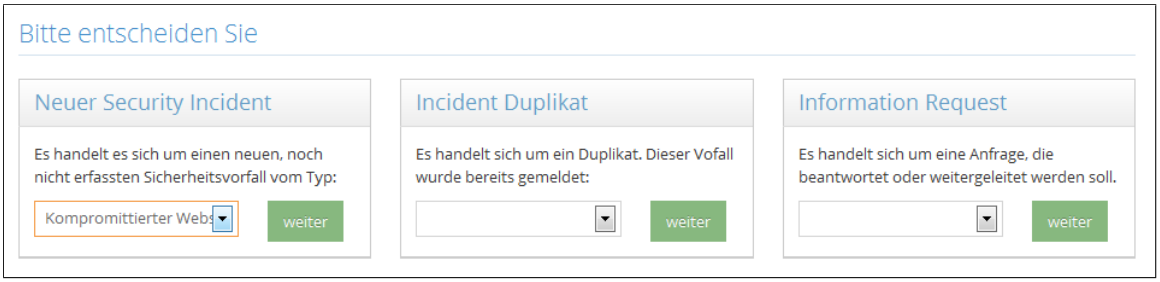

Abbildung 6.9.: Screenshot: Initiale Klassifikation (`/si-init.php`)

Liegt ein neuer Sicherheitsvorfall vor, so wählt der Hotliner im Rahmen der initialen Klassifikation bereits den Incidenttyp aus. Das Dropdown-Feld wird flexibel durch die Funktion `parseDropdown` generiert, wobei neben den vier Standard Security Incidents auch die Auswahl der Restklasse „Kein Standard Security Incident“ möglich ist. Das Absenden des Formulars bewirkt eine Übermittlung des Incidenttyps an die Aktion `classifyIncident` (vgl. Listing 6.20). Da die Klassifikation nur im Status `NEW` zulässig ist, erfolgt in Zeile 1 eine entsprechende Überprüfung mittels `checkState`. Anschließend wird in den Zeilen 3 bis 7 sichergestellt, dass der Vorfall nicht bereits klassifiziert wurde. Ist das Feld `klassifikation` nicht `NULL`, so wird der Hotliner zur Statusseite des Incidents geleitet und über den Fehler informiert. Nachdem die Klassifikation und der Incidenttyp in der Datenbank gespeichert wurden (Zeilen 9 und 10), wird die Klassifikation in Zeile 12 mit einer `insertHistory`-Anweisung dokumentiert. Sofern der übermittelte Incidenttyp nicht der Restklasse entspricht (Zeile 14), wird die anschließend vorgestellte Funktion `applySsiTemplate` aufgerufen, sodass bereits zu Beginn die Daten der Vorlage auf den aktuellen Vorfall übertragen werden. Schließlich wird die reguläre `updateState` Funktion aufgerufen, welche die üblichen Operationen zum Statuswechsel ausführt (vgl. Listing 6.14 in Abschnitt 6.2.5). Als dritter Parameter wird anstelle eines Zeitstempels der Wert `FALSE` übergeben. Dieser sorgt dafür, dass die Funktion den aktuellen Zeitstempel annimmt.

Listing 6.20: Aktion `classifyIncident` (`/actions/incidents.php`)

```
1  checkState($db,$_POST['si'],"NEW");
2
3  if(!is_null(getIncidentValue($db,$_POST['si'],"klassifikation"))){
4    setAlert('error','Der Vorgang SI-'.$_POST['si'].' wurde bereit klassifiziert
       .');
5    header("Location: ".$appConfig['url']."/si-overview.php?si=".$_POST['si']);
6    exit;
7  }
8
9  $sql = "UPDATE ".DBPREFIX."incidents SET klassifikation = 'INCIDENT',
       incidentType='".$_POST['classifyIncident']."' WHERE id=".$_POST['si'];
10 $query = $db->query($sql);
11
12 insertHistory($db,$_POST['si'],$_SESSION['userId'],"klassifikation","","
       INCIDENT",$_SESSION["userName"].", ".$_SESSION["userVorname"]);
13
14 if ($_POST['classifyIncident'] != "SONST") {
15   applySsiTemplate($db,$_POST['si'],$_POST['classifyIncident']);
16 }
17
18 updateState($db,$_POST['si'],FALSE);
```

Die soeben erwähnte Funktion `applySsiTemplate` (vgl. Listing 6.21) erwartet als Eingabeparameter einen Datenbankhandler, die ID des Vorfalls und die ID des Templates, das auf den Vorfall angewendet werden soll. Zunächst wird der Templatedatensatz anhand dessen ID aus der `ssiTpl`-Tabelle abgerufen und in der lokalen Variable `$template` gespeichert. Da die Spaltenbezeichnungen der Vorlagentabelle mit denen der `incidents`-Tabelle übereinstimmen, kann die Datenübernahme effizient abgewickelt werden. Zu deren Vorbereitung wird im Rahmen einer foreach-Schleife (Zeilen 6 bis 10) ein SQL-tauglicher `$updateString` generiert, der mit Ausnahme des `id`-Feldes (Zeile 7) alle Bezeichnungen und Werte des `$template`-Datensatzes umfasst. Am Ende des Strings befindet sich schleifenbedingt ein Komma, das in Zeile 12 entfernt wird, um die Korrektheit der SQL-Syntax zu gewährleisten. Daraufhin wird in den Zeilen 13 und 14 der `UPDATE`-Befehl, der den zuvor generierten und bereinigten `$updateString` enthält, an die Datenbank übermittelt. Schlägt diese Anfrage fehl, so wird in den Zeilen 15 bis 18 eine Fehlermeldung gesetzt und die Funktion wird mit dem Return-Wert `FALSE` verlassen. Im Normalfall (Zeile 19 bis 22) erzeugt die Funktion eine Erfolgsmeldung und gibt `TRUE` zurück.

Listing 6.21: Funktion `applySsiTemplate` (`/inc/functions.si.php`)

```
1  function applySsiTemplate($db,$si,$templateId){
2    $query = "SELECT * FROM ".DBPREFIX."ssiTpl WHERE id='".$templateId."'";
3    $stmt = $db->query($query);
4    $template = $stmt->fetch(PDO::FETCH_ASSOC);
5
6    foreach($template as $key => $value){
7      if ($key != "id"){
8        $updateString .= $key."='".$value."', ";
9      }
10   }
11
12   $updateString = rtrim($updateString, ", ");
13   $sql = "UPDATE ".DBPREFIX."incidents SET $updateString WHERE id='$si'";
14   $query = $db->query($sql);
```

```
15    if(!$query){
16      setAlert("error","Die Felder konnten nicht mit dem SSI Template
        $templateId vorbelegt werden.");
17      return FALSE;
18    }
19    else {
20      setAlert("ok","Die Felder wurden mit dem SSI Template $templateId
        vorbelegt.");
21      return TRUE;
22    }
23  }
```

6.4.2. Umgang mit Duplikaten und Information Requests

Stellt der Hotliner im Rahmen der initialen Klassifikation fest, dass der gemeldete Vorfall bereits von einem anderen Kollegen gemeldet wurde, so handelt es sich um ein Duplikat, das nicht weiter zu bearbeiten ist. In diesem Fall stehen im Dropdown-Feld des mittleren Widgets (vgl. Screenshot 6.9) alle in Bearbeitung befindlichen oder abgeschlossenen Sicherheitsvorfälle zur Auswahl. Auch dieses Feld wird dynamisch generiert, wobei die Funktion `parseIncidentsDropdown` zum Einsatz kommt. Nach der Festlegung des zugehörigen Eltern-Incidents wird dessen ID an die Aktion `classifyDuplicate` (`/actions/incidents.php`) übergeben. Wegen der strukturellen Ähnlichkeit zur `classifyIncident` Aktion (vgl. Listing 6.20), wird hier nur auf die wesentlichen Unterschiede eingegangen:

- `klassifikation: DUPLICATE`
- `currentState: CLOSED`
- `closedState: SI_DUPLICATE`

Auch hier wird die Klassifikation mittels `insertHistory` dokumentiert. Um das Duplikat mit dem Original zu verknüpfen, wird im Feld `duplicateId` des Duplikatdatensatzes die ID des originalen Incidents gespeichert. Da der Vorfall bereits durch das Datensatzupdate geschlossen wurde, ist ein Statuswechsel mit der Funktion `updateState` nicht mehr erforderlich. Schließlich wird der Vorfallmelder per `mailToContact` darüber informiert, dass der Vorfall bereits bekannt ist und als Duplikat abgeschlossen wurde. Darüber hinaus wird ihm die ID des originalen Incidents mitgeteilt, um gezielte Rückfragen zu ermöglichen.

Handelt es sich hingegen um einen Information Request, so wählt der Vorfallmelder im rechten Widget (vgl. Screenshot 6.9) aus, ob er den Vorfall selbst beantworten oder an die zuständige Bearbeitergruppe weiterleiten wird. Dementsprechend speichert die Aktion `classifyRequest` (`/actions/incidents.php`) die Klassifikation in der Datenbank:

- `klassifikation: REQUEST`
- `currentState: IR_AW / IR_FWD`

Nach der Dokumentation der Datensatzänderung mittels `insertHistory` Funktion wird dem Nutzer ein Mailformular zur Beantwortung (`/si-com-request-aw.php`) bzw. Weiterleitung (`/si-com-request-fwd.php`) angezeigt. Während das Antwortformular mit einem Mustertext vorausgefüllt ist, beinhaltet das Weiterleitungsformular alle zum Vorfall bekannten Informationen. Nach der Ergänzung individueller Hinweise wird der Mailinhalt an die Aktion `sendAwMail` bzw. `sendFwdMail` des Requestmoduls (`/actions/request.php`) übergeben.

Da sich die beiden Aktionen nur minimal unterscheiden, wird im Folgenden auf die Aktion `sendAwMail` (vgl. Listing 6.22) eingegangen: Der Mailversand an den Vorfallmelder wird in Zeile 1 mit der Funktion `mailToContact` realisiert, die für die automatische Dokumentation der versendeten E-Mail mittels `addComItem` sorgt. Anschließend wird der Vorfall durch ein Datensatzupdate geschlossen (Zeilen 3 und 4). Diese Änderungen werden in den Zeilen 6 und 7 mittels `insertHistory` im Verlaufsprotokoll gespeichert. Zuletzt wird der Anwender zum Kommunikationsprotokoll dieses Vorfalls geleitet (Zeile 10) und über den erfolgreichen Versand der E-Mail informiert (Zeile 9).

Listing 6.22: Aktion `sendAwMail` (`/actions/request.php`)

```
1  mailToContact($db,$_POST['si'],FALSE,$_POST['subject'],$_POST['content'],"
       SESSION","MELDER");
2
3  $sql = "UPDATE ".DBPREFIX."incidents SET currentState='CLOSED', closedState='
       IR_AW', currentStamp='$stamp', closedStamp='$stamp' WHERE id=".$_POST['si'
       ];
4  $query = $db->query($sql);
5
6  insertHistory($db,$_POST['si'],$_SESSION['userId'],"currentState","","CLOSED",
       $_SESSION["userName"].", ".$_SESSION["userVorname"]);
7  insertHistory($db,$_POST['si'],$_SESSION['userId'],"closedState","","IR_AW",
       $_SESSION["userName"].", ".$_SESSION["userVorname"]);
8
9  setAlert("ok","Die E-Mail an den Vorfallmelder wurde versendet. Der Vorfall
       wurde geschlossen.");
10 header("Location: ../si-com.php?si=".$_POST["si"]);
```

Die Aktion `sendFwdMail` nutzt hingegen die Funktion `mailToSomebody`, um die Anfrage an eine Bearbeitergruppe weiterzuleiten. Zudem wird der Vorfallmelder mittels `mailToContact` automatisch über die Weiterleitung seiner Anfrage und die Kontaktdaten der zuständigen Gruppe informiert, sodass Rückfragen ohne Umweg über das CSIRT direkt erfolgen können.

6.4.3. Weitere Klassifikation und Priorisierung

Im Folgenden wird nun davon ausgegangen, dass ein tatsächlicher Sicherheitsvorfall vorliegt. Als nächster Prozessschritt ist die Ergänzung der Vorfallinformationen (Vorfallzeitpunkt, Kunde, Betriebssystem, Hostname, IP-Adresse) vorgesehen. Zu diesem Zweck stellt die Seite `/si-data.php` im oberen Teil ein Formular zur Bearbeitung der übermittelten Vorfalldaten bereit. Die per `parseDropdown` erzeugten Formularfelder des unteren Bereichs dienen der weiteren Klassifikation und Priorisierung des Incidents. Diese sind bereits mit den Daten eines SSI-Templates vorausgefüllt, sofern im Rahmen der initialen Klassifikation ein Standard Security Incident ausgewählt wurde.

Während der aktuellen Prozessphase, die intern durch die Status-ID `CLASS` repräsentiert wird, ist beispielsweise festzulegen, ob ein Major Incident zu erwarten ist und welche Erstmaßnahmen zu treffen sind. Das in der Prozessbeschreibung des LRZ definierte Verfahren zur Ermittlung der Auswirkung, Priorität und Reaktionszeit des Vorfalls ist ebenfalls Bestandteil dieser Phase. Die fünf hierfür erforderlichen Angaben (Standort des Quell- und Zielsystems, Art der betroffenen Dienste und Informationen sowie die Anzahl der betroffenen Systeme) werden aus dynamisch generierten Dropdown-Feldern ausgewählt, wobei jede Auswahloption zum Zweck der Weiterverarbeitung mit einem Zahlenwert zwischen 1 und 3 codiert ist. Am Beispiel der Dienste entspricht der Wert 1 dem Fall, dass „keine kritischen

Dienste“ betroffen sind. Sollten hingegen „wichtige LRZ-Dienste“ betroffen sein, so wäre der Wert 3 auszuwählen.

Die Webanwendung stellt innerhalb der Aktion `updateIncidentValues` (vgl. Listing 6.23) einen Automatismus bereit, der die Auswirkung, Priorität und Reaktionszeit anhand der in der SIR-Prozessbeschreibung des LRZ [Met13] definierten Vorgehensweise ermittelt. Sofern der Anwender die Automatik aktiviert hat (Zeile 1), wird in Zeile 2 die Summe der fünf Inputparameter gebildet. Da diese bereits als Zahlenwerte vorliegen, ist an dieser Stelle keine Übersetzung mehr nötig. Das sich über die Zeilen 4 bis 23 erstreckende if-Konstrukt setzt die Variablen `$auswirkung`, `$prio` und `$reaktionszeit` in Abhängigkeit vom Ergebnis der vorherigen Summenbildung. Sowohl Auswirkung als auch Priorität werden als Zahlenwerte zwischen 1 (niedrig) und 4 (sehr hoch) codiert. Die Reaktionszeit wird hingegen in Minuten gespeichert. Diese beträgt beispielsweise für einen niedrig priorisierten Vorfall *einen* Arbeitstag (10 Stunden) und weist dementsprechend den Wert `600` auf.

Ist die in Zeile 2 berechnete Summe kleiner fünf oder größer fünfzehn, so liegt das Ergebnis außerhalb des zulässigen Wertebereichs. Dieser Fehlerfall wird in den Zeilen 24 bis 29 behandelt. Dabei erfolgt zunächst ein Eintrag ins Systemlog. Anschließend wird eine Fehlermeldung gesetzt, woraufhin der Anwender zum Formular zurückgeleitet wird und dort seine Eingaben korrigieren kann.

Listing 6.23: Automatische Ermittlung von Auswirkung, Priorität und Reaktionszeit in der Aktion `updateIncidentValues`

```
1  if (($_POST['feature'] == "si-data") AND ($_POST['automatik'] == "1")){
2    $sum = $newValues['matrixZielsys'] + $newValues['matrixDienste'] +
       $newValues['matrixInfos'] + $newValues['matrixAnzahl'] + $newValues['
       matrixQuellsys'];
3
4    if (($sum >= 5) AND ($sum <= 6)) {
5      $newValues['auswirkung'] = "1"; // niedrig
6      $newValues['prio'] = "1"; // niedrig
7      $newValues['reaktionszeit'] = "600"; // 10 Stunden
8    }
9    elseif (($sum >= 7) AND ($sum <= 9)) {
10     $newValues['auswirkung'] = "2"; // mittel
11     $newValues['prio'] = "2"; // mittel
12     $newValues['reaktionszeit'] = "240"; // 4 Stunden
13   }
14   elseif (($sum >= 10) AND ($sum <= 12)) {
15     $newValues['auswirkung'] = "3"; // hoch
16     $newValues['prio'] = "3"; // hoch
17     $newValues['reaktionszeit'] = "120"; // 2 Stunden
18   }
19   elseif (($sum >= 13) AND ($sum <= 15)) {
20     $newValues['auswirkung'] = "4";  // sehr hoch
21     $newValues['prio'] = "4"; // sehr hoch
22     $newValues['reaktionszeit'] = "15"; // 15 Minuten
23   }
24   else {
25     writeSyslog($db,$_SESSION["userId"],$_POST['si'],"automatik","ERROR","
       Ungültiger Punktwert: $sum");
26     setAlert('error','Automatik-Parameter unvollständig!');
27     header("Location: ".$_SESSION['lastUrl']); exit;
28   }
29 }
```

6.4.4. Zusammenstellung des Security Incident Teams

Die Zusammenstellung des Security Incident Teams erfolgt ebenfalls in der Klassifikationsphase, aus Gründen der Übersichtlichkeit jedoch auf einer separaten Seite (`/si-team.php`). Der obere Bereich der Seite gliedert sich in drei nebeneinander angeordnete Widgets und ist für die obligatorischen Teammitglieder reserviert. Während das linke Widget die Kontaktdaten des Vorfallmelders beinhaltet, kann im mittleren der Hotliner und im rechten der SIC aus einem Dropdown-Feld ausgewählt werden. Da für diese Teampositionen nur CSIRT-Mitglieder in Frage kommen, werden die beiden Felder mittels `parseCsirtDropdown` erzeugt. Der untere Bereich beinhaltet eine tabellarische Auflistung aller weiteren Teammitgliedern und ihrer Kontaktdaten.

Um ein Teammitglied hinzuzufügen, wird dessen Kennung und Rolle (`contactRole`) an die Aktion `addContact` (vgl. Listing 6.24) übergeben. Zunächst werden in den Zeilen 1 bis 3 die Kontaktdaten der Person anhand ihrer Kennung aus der `users`-Tabelle abgefragt und anschließend (Zeilen 5 bis 7) unter Zuordnung der Incident-ID und der `contactRole` als neuer Datensatz in der `contacts`-Tabelle gespeichert. Im Falle eines fehlerhaften Schreibzugriffs auf die Datenbank (Zeile 8 bis 12) wird auf der Teamseite eine Fehlermeldung angezeigt. Im Normalfall (Zeilen 13 bis 18) wird jedoch das Hinzufügen des Teammitglieds mittels `insertHistory` im Verlaufsprotokoll dokumentiert. Daraufhin wird der Anwender mit einer Erfolgsmeldung zur Teamseite zurückgeleitet.

Listing 6.24: Aktion `addContact` (`/actions/contacts.php`)

```
1  $query = "SELECT * FROM ".DBPREFIX."users WHERE kennung='".$_POST['kennung']."
       ' LIMIT 1";
2  $stmt = $db->query($query);
3  $user = $stmt->fetch(PDO::FETCH_ASSOC);
4
5  $sql = "INSERT INTO ".DBPREFIX."contacts (si,contactRole,kennung,name,vorname,
       anrede,email,tel,mobil,comment) VALUES (?,?,?,?,?,?,?,?,?,?)";
6  $query = $db->prepare($sql);
7  $query->execute(array($_POST["si"],$_POST["contactRole"],$user["kennung"],
       $user["name"],$user["vorname"],$user["anrede"],$user["email"],$user["tel"],
       $user["mobil"],$user["comment"]));
8  if(!$query){
9    setAlert('error','Das Teammitglied konnte nicht angelegt werden.');
10   header("Location: ".$_SESSION['lastUrl']);
11   exit;
12 }
13 else {
14   insertHistory($db,$_POST['si'],$_SESSION['userId'],"TEAM","","$user[vorname]
       $user[name] wurde als $_POST[contactRole] hinzugefügt.",$_SESSION['
       userName'],", ".$_SESSION['userVorname']);
15   setAlert("ok","$user[vorname] $user[name] wurde zum SI-$_POST[si]-Team als
       $_POST[contactRole] hinzugefügt.");
16   header("Location: ".$_SESSION['lastUrl']);
17   exit;
18 }
```

Die Aktionen zum Ändern von Kontaktdaten (`editContact`) und Löschen von Teammitgliedern (`deleteContact`) sind auf ähnliche Weise implementiert. Beim Löschen wird zusätzlich darauf geachtet, dass es sich beim zu löschenden Kontakt nicht um den Vorfallmelder handelt. Da sich alle anderen Teampositionen im Prozessverlauf ändern können, gilt die Ausnahme nur für die statische Position des Vorfallmelders.

Um eine effiziente Kommunikation mit den Teammitgliedern zu ermöglichen, sind alle Einträge dieses incidentbezogenen Adressverzeichnisses mit einem Telefon- und Mailsysmbol versehen. Möchte der Hotliner eine E-Mail an ein bestimmtes Teammitglied senden oder eine Gesprächsnotiz erfassen, genügt ein Klick auf das entsprechende Symbol, wobei der ausgewählte Mailempfänger bzw. Gesprächspartner via `GET`-Parameter an das Kommunikationsmodul übermittelt wird.

Nach Abschluss der Teamzusammenstellung kann mit der Analyse des Vorfalls begonnen werden. Hierfür ist es erforderlich, in der Webanwendung die nächste Prozessphase zu starten. Dies erfolgt anhand der `updateState` Aktion, deren Funktionsweise bereits im Abschnitt 6.2.5 erläutert wurde.

6.4.5. Analyse, Lösung und Monitoring

Während der Analysephase sind auf der Seite `/si-tasks.php` die durchzuführenden Maßnahmen und die Analyseergebnisse in zwei Textareas einzutragen, wobei beliebig viele Ergänzungen und Änderungen möglich sind. Zur Speicherung der erfassten Daten wird die Aktion `updateIncidentValues` (vgl. Listing 6.25) aufgerufen. Diese schreibt die Änderungen in die Datenbank und dokumentiert sie zugleich im Verlaufsprotokoll des Incidents.

Die Aktion `updateIncidentValues` ist entsprechend flexibel implementiert, sodass sie nicht nur während der Incident Analyse, sondern auch in allen anderen Prozessphasen sowie zur Erfassung der Post Incident Review Ergebnisse eingesetzt werden kann. Aus diesem Grund werden im switch-Block (Zeilen 1 bis 11) für jede Datei (`$feature`) die dort vorhandenen Felder (`$fieldsArray`) definiert. Darüber hinaus werden in der Variablen `$allowedStates` die Prozessphasen festgelegt, in denen die zuvor bestimmten Felder editiert werden dürfen. Durch die `check` Funktion in Zeile 13 wird also sichergestellt, dass nur solche Felder geändert werden, die in der aktuellen Prozessphase zulässig sind. So wird beispielsweise ein Schreibzugriff auf das Feld `monDauer` (Zeile 5) unterbunden, wenn sich der Vorfall noch in der Analysephase befindet. Eine Änderung der Incidentpriorität (Zeile 2) ist hingegen in allen Prozessphasen möglich, solange der Vorfall noch nicht abgeschlossen wurde.

Zur Vorbereitung der nächsten Schritte wird in Zeile 15 das `$fieldsArray` in einen kommagetrennten String konvertiert. Das `implode`-bedingte Komma am Ende des Strings ist unerwünscht und wird daher in Zeile 16 entfernt. Zur Weiterverarbeitung der im `POST`-Array übergebenen Daten werden in einer foreach-Schleife (Zeilen 18 bis 20) nur diejenigen Felder in das `newValues`-Array überführt, die im `$fieldsArray` für die aktuelle Phase als zulässig definiert wurden.

Falls die Aktion von der Seite `/si-data.php` aus aufgerufen und der Anwender den Automatismus aktiviert hat, wird ab Zeile 22 die Ermittlung von Auswirkung, Priorität und Reaktionszeit durchgeführt. Da diese bereits in Listing 6.23 separat vorgestellt wurde, ist das Listing der Aktion `updateIncidentValues` entsprechend gekürzt.

Anhand des in den Zeilen 15 und 16 vorbereiteten Strings werden in den Zeilen 24 bis 26 die alten Werte vor der Änderung aus der Datenbank abgerufen. Die verschachtelte foreach-if-Konstruktion in den Zeilen 28 bis 34 ermittelt alle geänderten Feldinhalte. Dabei vergleicht sie alle neuen Werte (`$newValues`) mit den alten (`$old`). Unterscheidet sich ein Wertepaar, so wurde dieses Feld geändert.

Listing 6.25: Aktion `updateIncidentValues` (`/actions/incidents.php`)

```php
switch ($_POST['feature']){
  case "si-data": $fieldsArray = array("title","description","vorfallzeitpunkt","kunde","os","host","ip","majorExpected","hochschulstart","matrixZielsys","matrixDienste","matrixInfos","matrixAnzahl","matrixQuellsys","auswirkung","prio","reaktionszeit","firstaid"); $allowedStates="CLASS,ANAL,SOL,MON"; break;
  case "si-tasks": $fieldsArray = array("vorgehensweise","analyseerg"); $allowedStates="ANAL,SOL,MON"; break;
  case "si-solution": $fieldsArray = array("regelbetrieb","lsgType","lsgDetails"); $allowedStates="SOL,MON"; break;
  case "si-monitoring": $fieldsArray = array("monDauer","monIntervall","monAufgabe","monAuffaellig"); $allowedStates="MON"; break;
  case "si-review": $fieldsArray = array("revPositiv","revNegativ","revOptimierung"); $allowedStates="CLOSED"; break;
  default:
    setAlert('error','Feature Parameter fehlt!');
    header("Location: ".$appConfig['url']."/dashboard.php");
    exit;
}

check("Status",$_POST['si'],getIncidentValue($db,$_POST['si'],"currentState"),$allowedStates);

$fieldsString = implode(', ', $fieldsArray);
$fieldsString = rtrim($fiedlsString, ', ');

foreach ($fieldsArray as $item){
  $newValues[$item] = $_POST[$item];
}

// Automatische Ermittlung von Auswirkung, Priorität und Reaktionszeit

$query = "SELECT $fieldsString FROM ".DBPREFIX."incidents WHERE id = '".$_POST['si']."'";
$stmt = $db->query($query);
$old = $stmt->fetch(PDO::FETCH_ASSOC);

foreach($newValues as $key => $value){
  if ($value != $old[$key]){
    $doUpdate = TRUE;
    $updateString .= $key."='".$value."', ";
    insertHistory($db,$_POST['si'],$_SESSION['userId'],$key,$old[$key],$value,$_SESSION['userName'].", ".$_SESSION['userVorname']);
  }
}

if ($doUpdate) {
  $updateString = rtrim($updateString, ', ');
  $sql = "UPDATE ".DBPREFIX."incidents SET $updateString WHERE id = '".$_POST['si']."'";
  $query = $db->query($sql);
  if(!$query){setAlert('error','Fehler beim Speichern der Änderungen.');}
  else {setAlert('ok','Die Änderungen wurden gespeichert.');}
}
else {setAlert('error','Es lagen keine Änderungen vor.');}

header("Location: ".$_SESSION['lastUrl']);
```

In diesem Fall wird das Flag `$doUpdate` gesetzt (Zeile 30), das geänderte Feld wird in `$updateString` aufgenommen (Zeile 31) und die Änderung wird mittels `insertHistory` im Verlaufsprotokoll dokumentiert, wobei der alte und der neue Wert des geänderten Feldes gespeichert werden. Ist nach Ende der Schleife das Flag `$doUpdate` *nicht* gesetzt, so wurde die Aktion aufgerufen, obwohl die Daten nicht geändert wurden. In diesem Fall (Zeile 43) wird dem Anwender mittels `setAlert` ein Hinweis gegeben.

Im Normalfall wurde jedoch mindestens eine Datenänderung vorgenommen, sodass die if-Bedingung in Zeile 36 den Wert `TRUE` ergibt. Nach der Bereinigung des `$updateStrings` in Zeile 37 analog zu Zeile 16 erfolgt schließlich die tatsächliche Änderung in der `incidents`-Tabelle (Zeilen 38 und 39). In Abhängigkeit vom Rückgabewert der Datenbankoperation wird in den Zeilen 40 bis 42 eine Fehler- bzw. Erfolgsmeldung gesetzt. Zuletzt wird der Anwender auf die Ursprungsseite zurückgeleitet (Zeile 45).

Nach dem Abschluss der Analyse und deren Dokumentation kann die Messung der Analysezeit beendet und die Lösungsphase gestartet werden. Beides erfolgt anhand der `updateState` Aktion, deren Funktionsweise bereits im Abschnitt 6.2.5 erläutert wurde.

Während der Lösungsphase sind auf der Seite `/si-solution.php` die Maßnahmen zur Wiederherstellung des Regelbetriebs und die im Rahmen der Incidentlösung durchgeführten Schritte in zwei Textareas zu dokumentieren. Darüber hinaus ist der Lösungstyp aus einem mittels `parseDropdown` Funktion generierten Feld auszuwählen. Die Speicherung der Daten erfolgt auch hier durch die universelle Aktion `updateIncidentValues`. Nach der Wiederherstellung des Regelbetriebs wird mit der `updateState` Aktion die Messung der Lösungszeit beendet und die Monitoringphase eingeleitet. Auf der entsprechenden Seite `/si-monitoring.php` sind neben den Monitoringaufgaben und Verantwortlichkeiten auch die Dauer und Frequenz der Überwachungsaktivitäten festzulegen. Darüber hinaus sind eventuelle Auffälligkeiten zu dokumentieren. Auch in dieser Phase werden die Eingaben durch die Aktion `updateIncidentValues` in der Datenbank gespeichert. Wie bereits beschrieben, werden die vorgenommenen Änderungen im Rahmen dieser Aktion automatisch in der `history`-Tabelle dokumentiert. Da nach dem Monitoring keine weitere Prozessphase folgt, wird der Vorfall – wie im folgenden Abschnitt beschrieben – in den Status `CLOSED` überführt.

6.4.6. Abschließen, Wiedereröffnen und Abbrechen von Security Incidents

Wurden während der Monitoringphase keine weiteren Auffälligkeiten erkannt, so kann der Vorfall geschlossen werden. Die Vorfallinformationen werden dabei direkt in der `incidents`-Tabelle durch Setzen der Felder `closedState` und `closedStamp` archiviert und der Vorfallmelder wird per E-Mail über den Abschluss des Sicherheitsvorfalls informiert. Aufgrund der zusätzlichen Schritte ist die universelle Aktion `updateState` für diesen Vorgang nicht ausreichend. Folglich wird eine separate Aktion `closeIncident` implementiert (vgl. Listing 6.26). Während in Zeile 1 durch die Funktion `checkState` sichergestellt wird, dass der Incident nur aus der Monitoringphase heraus geschlossen werden kann, wird in den Zeilen 3 bis 6 u.a. der Abschlussstatus (z.B. `SI_SUCCESS` für den erfolgreichen Incident Abschluss) in der `incidents`-Tabelle gespeichert und mit der Funktion `insertHistory` dokumentiert. Anschließend wird der Vorfallmelder durch Aufruf der `mailToContact` Funktion über den Abschluss des Vorfalls informiert (Zeilen 8 bis 10). Zuletzt wird die reguläre `updateState` Funktion aufgerufen, welche die üblichen Operationen zum Statuswechsel ausführt und die Dauer der Incidentnachsorge berechnet (vgl. Listing 6.14 in Abschnitt 6.2.5).

Listing 6.26: Aktion `closeIncident` (`/actions/incidents.php`)

```
checkState($db,$_POST['si'],"MON");

$sql = "UPDATE ".DBPREFIX."incidents SET closedState='".$_POST['closedState'].
    "', closedStamp='".$_POST['stamp']."' WHERE id=".$_POST['si'];
$query = $db->query($sql);

insertHistory($db,$_POST['si'],$_SESSION['userId'],"closedState","NONE",$_POST
    ['closedState'],$_SESSION["userName"].", ".$_SESSION["userVorname"]);

$subject = "wurde geschlossen";
$body = "wir freuen uns, Ihnen mitteilen zu dürfen, dass der Incident [SI-".
    $_POST['si']."] inzwischen gelöst, überwacht und soeben geschlossen wurde.
    Für Rückfragen stehen wir gerne zur Verfügung.";
mailToContact($db,$_POST['si'],FALSE,$subject,$body,"SESSION","MELDER");

updateState($db,$_POST['si'],$_POST['stamp']);
```

Die für das Wiedereröffnen eines Vorfalls implementierte Aktion `reopenIncident` wird benötigt, wenn nach dem Abschließen eines Sicherheitsvorfalls weitere Auffälligkeiten bekannt werden, sodass der Vorfall erneut zu bearbeiten ist. Sie ist strukturell mit der soeben beschriebenen Aktion `closeIncident` identisch. Unterschiede bestehen hinsichtlich des Mailinhalts und der aktualisierten Datenbankfelder, wobei das `reopened`-Flag gesetzt und der `closedState` zurückgesetzt wird.

Stellt der Hotliner im Laufe der Incidentbearbeitung fest, dass es sich um einen „Fehlalarm“ handelt, kann er den Vorfall abbrechen. Dies geschieht auf ähnliche Weise wie der zuvor beschriebene Abschluss eines Vorfalls. Die Aktion `cancelIncident` kann in allen Prozessphasen der Incidentbearbeitung (Klassifikation, Analyse, Lösung und Monitoring) aufgerufen werden. Sie verzichtet jedoch auf die Anwendung der `updateState` Funktion und erledigt stattdessen alle notwendigen Vorgänge selbst.

6.4.7. Post Incident Review

Nach Abschluss eines Vorfalls kann ein Post Incident Review durchgeführt werden, das die Optimierung des SIR-Prozesses bezweckt. Als Datenbasis können folgende Quellen betrachtet werden:

- Statusseite (`/si-overview.php`)
- Verlaufsprotokoll (`/si-history.php`)
- Kommunikationsprotokoll (`/si-com.php`)

Die Statusseite fasst alle zum Vorfall final vorliegenden Informationen zusammen und gibt Auskunft über die Laufzeiten der einzelnen Phasen. Anhand des Verlaufsprotokolls ist es möglich, die Vorgehensweise chronologisch exakt nachzuvollziehen, um eventuelle Defizite im Ablauf identifizieren zu können. Darüber hinaus kann eine Analyse des Kommunikationsprotokolls Hinweise auf Hindernisse im Informationsfluss geben.

Die Webanwendung stellt für die Dokumentation der Reviewergebnisse eine eigene Seite (`/si-review.php`) bereit, die über die horizontale Incidentmenüleiste aufgerufen werden

kann. Sie beinhaltet drei Textareas zur Erfassung von positiven und negativen Erfahrungen sowie von Optimierungsvorschlägen. Da die Reviewinformationen Teil des Incidentdatensatzes sind, erfolgt die Speicherung der Eingaben durch die universelle `updateIncidentValues` Aktion. Dadurch ist sichergestellt, dass auch die Durchführung des Post Incident Reviews im Verlaufsprotokoll des Vorfalls dokumentiert ist.

6.5. Vielfältige Nutzung des Datenbestands

Die strukturierte Erfassung der Sicherheitsvorfälle innerhalb der Webanwendung bietet vielfältige Möglichkeiten zur Nutzung des Datenbestands. Zu diesem Zweck stellt die Software eine Filterkomponente mit integrierter Suchfunktion bereit. Darüber hinaus können mit wenigen Klicks kundenabhängige Auswertungen generiert werden, die über das Incidentaufkommen und die Prozessperformance während des Auswertungszeitraums informieren. Schließlich ermöglichen die Exportfunktionen (CSV und PDF) eine flexible Weiterverarbeitung der Incidentdaten in anderen Kontexten außerhalb der Webanwendung.

6.5.1. Filterung und Suche

Die im Screenshot 6.10 abgebildete Seite ermöglicht die Filterung des Datenbestandes und eine anschließende Suche in der Ergebnisliste. Im oberen Bereich der Seite können insgesamt 15 Filterparameter festgelegt werden. Zur besseren Übersicht sind die Auswahlfelder auf drei Gruppen verteilt. Während als Basisparameter sowohl der Filterzeitraum, als auch der Kunde und die Klassifikation festgelegt werden können, ermöglicht die Gruppe der erweiterten Parameter beispielsweise eine Filterung nach dem dem Betriebssystem oder dem Incidenttyp, der die Standard Security Incidents abbildet. Die dritte Gruppe enthält die für die Priorisierung des Vorfalls verwendeten Parameter wie die Anzahl der Zielsysteme und den Standort des Quellsystems. Auch hier werden die Dropdown-Felder mittels `parseDropdown` erst zur Laufzeit generiert. Um die weitere Verarbeitung effizient zu gestalten, werden die Bezeichnungen der Filterfelder so gewählt, dass diese mit den korrespondierenden Spaltennamen der `incidents`-Tabelle übereinstimmen.

Mit dem Absenden des Formulars werden die Parameter an die Aktion `setFilter` (vgl. Listing 6.27, Zeilen 6 bis 18) übermittelt. Diese prüft zunächst in den Zeilen 7 bis 11 die formale Korrektheit des zu filternden Zeitraums und verwendet im Fehlerfall das aktuelle Jahr. Anschließend werden die mittels `POST`-Request übergebenen Filterparameter in einer foreach-Schleife in das Session-Array `$filter` gespeichert (Zeile 13). Das hierbei verwendete Array `$filterFields` stammt aus der Datei `/inc/basics.php` und beinhaltet die Bezeichnungen aller Filterfelder. Da diese mehrfach benötigt werden, sind sie an zentraler Stelle definiert (Zeilen 1 bis 4) und stehen somit in allen Dateien der Filterkomponente zur Verfügung. Schließlich wird in den Zeilen 15 bis 18 der in dieser Aktion nicht benötigte Datenbankhandler geschlossen, woraufhin der Anwender mit einem Erfolgshinweis zur Ursprungsseite (`/filter.php`) geleitet wird.

Dort werden die in der Session-Variablen `$filter` zwischengespeicherten Parameter abgerufen und mit einer UND-Verknüpfung in die SQL-Abfrage integriert. Dabei iteriert eine foreach-Schleife (Zeile 21) über alle Filterfelder und ergänzt den jeweiligen Parameter – sofern

er nicht leer ist – im String `$whereFilter` (Zeile 22). Nach dem letzten Schleifendurchlauf enthält dieser String alle Filterbedingungen mit Ausnahme des Zeitraums in valider SQL-Syntax. In Zeile 25 wird die Datenbankabfrage vorbereitet, wobei sich die `WHERE`-Bedingung aus dem Filterzeitraum und den übrigen – im String `$whereFilter` enthaltenen – Parameter zusammensetzt. Schließlich werden in den Zeilen 26 und 27 die selektierten Datensätze aus der `incidents`-Tabelle abgerufen und im Array `$result` zur Weiterverarbeitung gespeichert.

In der unteren Hälfte des Screenshots 6.10 werden die Ergebnisse der Filterung tabellarisch dargestellt, wobei das jQuery Plugin `DataTables`[5] auf die Tabelle angewendet wird. Dies ermöglicht neben der spaltenbasierten Sortierung der Einträge auch die Suche innerhalb der Tabelle.

Die Filterparameter bleiben so lange im Session-Kontext gespeichert, bis die Aktion `resetFilter` aufgerufen oder die Sitzung geschlossen wird. Verlässt der Anwender die Filterseite, um beispielsweise ein Ergebnis detailliert zu betrachten, so kann er anschließend ohne weitere Eingaben direkt zu der vorher gefilterten Ergebnisliste zurückkehren. Die selektierten Vorfälle lassen sich sowohl im PDF als auch im CSV-Format exportieren. Auf die Implementierung der Exportmodule wird im übernächsten Abschnitt 6.5.3 eingegangen.

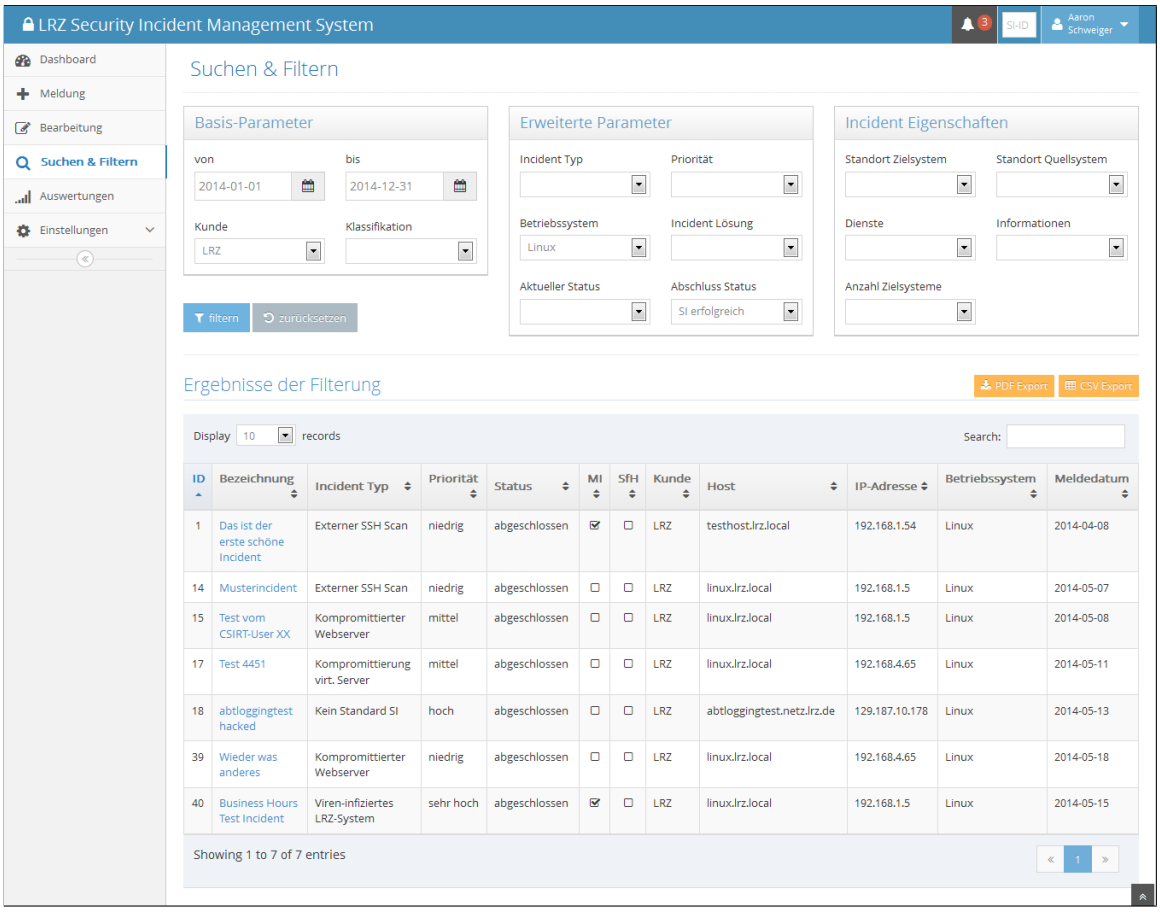

Abbildung 6.10.: Screenshot: Suchen & Filtern (`/filter.php`)

[5] http://www.datatables.net/

Listing 6.27: Implementierung der Incidentfilterung

```php
// Zentrale Definition der Filterfelder (/inc/basics.php)
if ($feature == "filter") {
  $filterFields = array('kunde','klassifikation','incidentType','prio','os','lsgType','currentState','closedState','matrixZielsys','matrixQuellsys','matrixDienste','matrixAnzahl','matrixInfos');
}

// setFilter Aktion (/actions/filter.php)
if (strtotime($_POST['from']) === FALSE){$_SESSION['filter']['from'] = date("Y",time())."-01-01";}
else {$_SESSION['filter']['from'] = date("Y-m-d",strtotime($_POST['from']));}

if (strtotime($_POST['to']) === FALSE){$_SESSION['filter']['to'] = date("Y",time())."-12-31";}
else {$_SESSION['filter']['to'] = date("Y-m-d",strtotime($_POST['to']));}

foreach($filterFields as $item){$_SESSION['filter'][$item] = $_POST[$item];}

$db = null;
setAlert('ok','Die Einträge wurden gemäß der Vorgaben gefiltert.');
header("Location: ".$appConfig['url']."/filter.php");
exit;

// Abfrage der gefilterten Datensätze (/filter.php)
foreach($filterFields as $item){
  if($_SESSION['filter'][$item] != "") {$whereFilter .= "AND $item = '".$_SESSION['filter'][$item]."' ";}
}

$query = "SELECT id,title,description,incidentType,prio,currentState,klassifikation,majorExpected,kunde,host,ip,os, createdStamp FROM ".DBPREFIX."incidents WHERE createdStamp BETWEEN '".$_SESSION['filter']['from']."' AND '".$_SESSION['filter']['to']."' $whereFilter ORDER BY id";
$stmt = $db->query($query);
$result = $stmt->fetchAll(PDO::FETCH_ASSOC);
```

6.5.2. Auswertungen

Die Webanwendung stellt auf der Seite `/stats.php` einfache statistische Auswertungen bereit. Damit bietet sie dem CSIRT einen zahlenmäßigen Überblick über das Incidentaufkommen und den Leistungsgrad des SIR-Prozesses in Form von Key-Performance-Indikatoren. Wie auf dem Screenshot 6.11 zu erkennen ist, orientiert sich die Struktur der Auswertungskomponente exakt an den Vorgaben des Konzepts (vgl. Abschnitt 5.4.2).

Standardmäßig erstreckt sich die Auswertung auf alle über die Webanwendung erfassten Vorfälle das laufenden Kalenderjahres, wobei eine die Festlegung eines abweichenden Auswertungszeitraums und die Auswahl eines Kunden möglich ist. Diese Eingrenzung der Datenbasis wird technisch über die `setFilter` Aktion des `stats`-Moduls realisiert. Die Funktionsweise dieser Aktion entspricht im Wesentlichen ihrem Äquivalent im `filter`-Modul und wird daher nicht genauer beschrieben. Stattdessen wird auf ausgewählte Datenbankabfragen eingegangen, um zu demonstrieren, wie die auf der Auswertungsseite gezeigten Zahlen zu Stande kommen:

Listing 6.28: Überwachung der Reaktionszeiten (`/stats.php`)

```
1  SELECT
2    COUNT(*) AS anzahl, MIN(differenz) AS min,
3    MAX(differenz) AS max, AVG(differenz) AS avg
4  FROM
5    ( SELECT saldo1bh - reaktionszeit AS differenz
6      FROM incidents
7      WHERE
8        createdStamp BETWEEN '".$_SESSION['filter']['from']." ' AND
9        '".$_SESSION['filter']['to']." ' AND closedState='SI_SUCCESS'
10       $whereFilter AND saldo1bh > reaktionszeit
11   ) sub;
```

Das in Listing 6.28 abgedruckte SQL-Statement ermittelt die absolute Anzahl der Vorfälle, bei denen die vorgegebene Reaktionszeit überschritten wurde sowie die Dauer der kleinsten, größten und durchschnittlichen Reaktionszeitüberschreitung. Dabei wird zunächst die Subquery namens `sub` in den Zeilen 5 bis 11 ausgeführt. Diese selektiert für alle erfolgreich abgeschlossenen Vorfälle, die innerhalb des gewählten Zeitraums erfasst wurden (Zeilen 8 und 9) und zudem eine Überschreitung der Reaktionszeit aufweisen (Zeile 10) die Differenz der Spalten `saldo1bh` und `reaktionszeit` (Zeile 5), also den Unterschied zwischen der tatsächlichen Reaktionszeit während der Servicezeiten und der Vorgabe in Minuten. Falls eine kundenspezifische Auswertung erstellt werden soll, enthält der String `$whereFilter` in Zeile 10 die hierfür erforderliche Ergänzung der `WHERE`-Bedingung. Ist kein Kunde gewählt, bleibt die Variable leer, sodass die Vorfälle aller Kunden berücksichtigt werden.

Die Hauptquery wendet in den Zeilen 2 und 3 mehrere SQL-Funktionen auf das Ergebnis der Subquery an und ermittelt dadurch die Anzahl der Einträge (`COUNT`), die kürzeste Dauer (`MIN`), die größte Dauer (`MAX`) und den Durchschnitt aller Zeitwerte (`AVG`). Anschließend werden die Minutenangaben in Sekunden umgerechnet und mit der `secondsToString` Funktion in einen lesbaren String konvertiert. Die Darstellung der Zahlen erfolgt in der Tabelle „Überwachung der Reaktionszeiten“ und ist auf dem Screenshot 6.11 in der rechten Spalte der zweiten Reihe zu sehen.

Listing 6.29: Auswertung nach Prioritäten (`/stats.php`)

```
1  SELECT
2    sub1.prio AS prio, anzahl, vorgabe, saldo1bh, saldo2bh, saldo3bh, saldo4bh
3  FROM
4    ( SELECT value AS prio
5      FROM types
6      WHERE category='prio'
7    ) sub1
8    LEFT JOIN
9    ( SELECT
10       prio, COUNT(*) AS anzahl, ROUND(AVG(reaktionszeit)) AS vorgabe,
11       ROUND(AVG(saldo1bh)) AS saldo1bh, ROUND(AVG(saldo2bh)) AS saldo2bh,
12       ROUND(AVG(saldo3bh)) AS saldo3bh, ROUND(AVG(saldo4bh)) AS saldo4bh
13     FROM incidents
14     WHERE
15       createdStamp BETWEEN '".$_SESSION['filter']['from']." ' AND
16       '".$_SESSION['filter']['to']." ' AND closedState='SI_SUCCESS'
17       $whereFilter GROUP BY prio
18   ) sub2
19   ON sub1.prio = sub2.prio;
```

Die ebenfalls dort in der dritten Reihe abgebildete Tabelle listet für jede Incidentpriorität die Anzahl der Vorfälle, deren durchschnittliche Reaktionszeit sowie die absolute und prozentuale Abweichung von der für die jeweilige Priorität vorgegebenen Reaktionszeit auf. Zusätzlich werden die Durchschnittswerte für Analysezeit, Lösungszeit und Nachlaufzeit für jede Priorität angegeben. Die zugehörige Datenbankabfrage besteht aus einer Hauptquery und zwei mittels `LEFT JOIN` verbundenen Subqueries. Das SQL-Statement ist in Listing 6.29 abgebildet. Während die erste Subquery `sub1` (Zeilen 4 bis 7) eine Liste aller Prioritäten aus der Typentabelle zurückliefert, werden in der zweiten Subquery `sub2` (Zeilen 9 bis 18) alle innerhalb des gewählten Zeitraums erfassten und erfolgreich abgeschlossenen Vorfälle selektiert (Zeilen 15 und 16), wobei mittels `$whereFilter` (Zeile 17) eine kundenspezifische Eingrenzung erfolgen kann. Aus diesen Datensätzen werden durch die `GROUP BY` Anweisung Prioritätsklassen gebildet (Zeile 17). Diese werden durch die Anwendung verschiedener Aggregatsfunktionen zu einem einzigen Datensatz je Prioritätsklasse zusammengefasst. So wird in den Zeilen 10 bis 12 für jede Priorität die Anzahl der Datensätze (`COUNT(*)`), die im Durchschnitt vorgegebene Reaktionszeit (`AVG(reaktionszeit)`) sowie die durchschnittliche Dauer der tatsächlichen Reaktionzeit (`AVG(saldo1bh)`), Analysezeit (`AVG(saldo2bh)`), Lösungszeit (`AVG(saldo3bh)`) und Nachlaufzeit (`AVG(saldo4bh)`) ermittelt.

Der `LEFT JOIN` der beiden Subqueries in den Zeilen 8 und 19 bewirkt eine Zusammenfassung dieser Relationen. Die `JOIN`-Relation enthält dadurch für alle `sub1`-Prioritätswerte die übereinstimmenden `sub2`-Tupel. Folglich gewährleistet der `LEFT JOIN`, dass die Ergebnisrelation auch diejenigen Prioritäten beinhaltet, die noch nicht verwendet wurden und daher in `sub2` nicht vorkommen. Die Hauptquery selektiert schließlich in Zeile 2 die Attribute der `JOIN`-Relation.

Nach der Datenbankabfrage werden die Ergebnisse für die Darstellung im Browser optimiert. Dabei werden die als Zahl codierten Prioritäten (1-4) mit der Funktion `getTypeName` in eine Bezeichnung wie „sehr hoch“ übersetzt. Darüber hinaus werden alle Zeitwerte mittels `secondsToString` in einen verständlichen String (Tage, Stunden, Minuten) konvertiert. Die Abweichung der tatsächlichen Reaktionzeit von der Vorgabe wird durch die Funkti-

on `calcAbweichung` als absoluter und prozentualer Wert berechnet. Eine positive Differenz bedeutet eine Überschreitung der Reaktionszeit, eine negative eine Unterschreitung. Die Funktion gibt einen formatierten HTML-String zurück, wobei zur besseren Übersicht alle Überschreitungen rot sowie Unterschreitungen grün eingefärbt sind.

Zur professionellen Berichterstattung gegenüber der LRZ-Leitung und den Kunden ist es möglich, die Auswertungsdaten gesamt und kundenspezifisch im PDF-Format zu exportieren. Daher wird im nächsten Abschnitt auf die Implementierung der Exportmodule eingegangen.

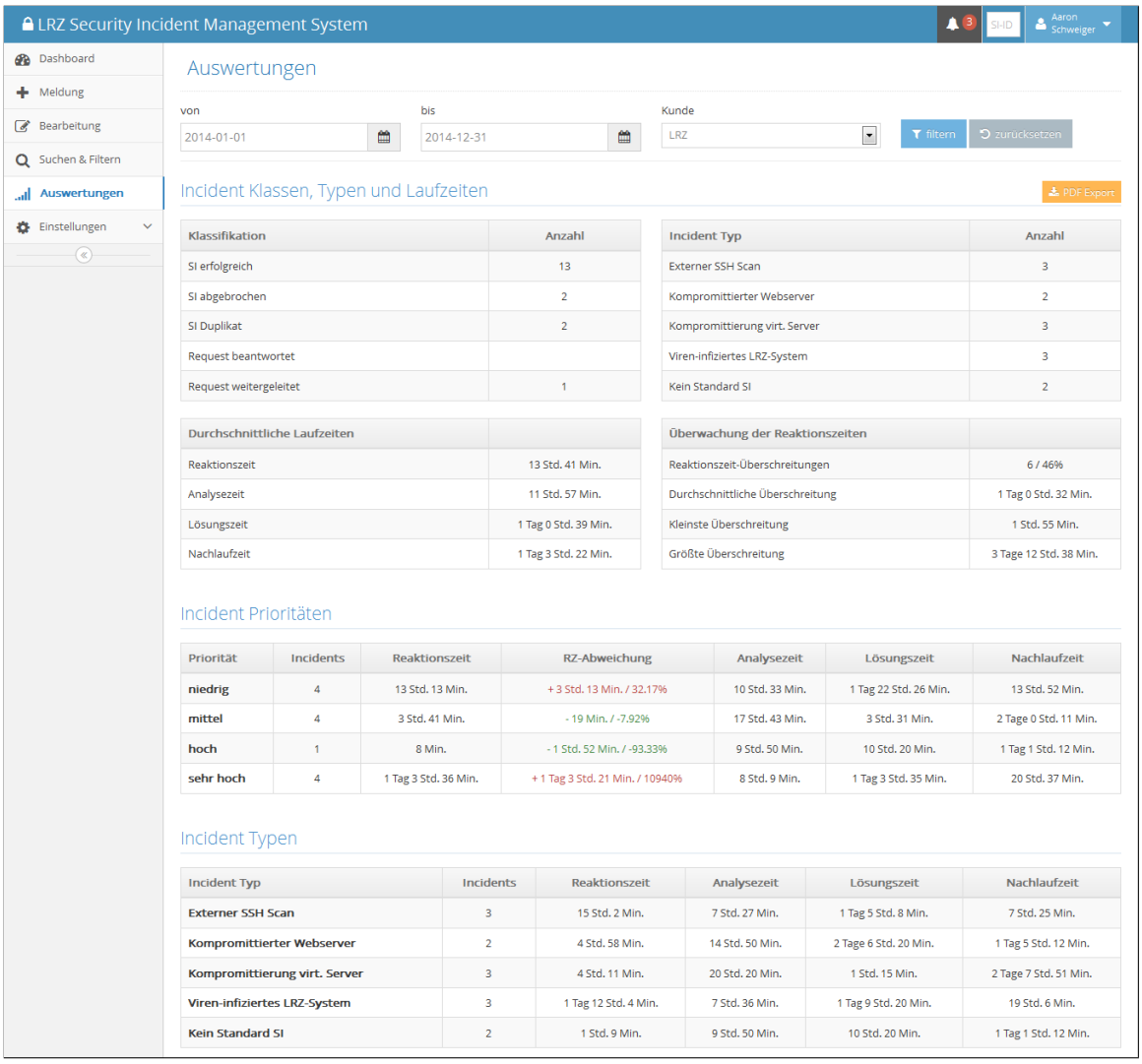

Abbildung 6.11.: Screenshot: Auswertungen (`/stats.php`)

6.5.3. Exportmodule

Zur flexiblen Weiterverarbeitung des Datenmaterials außerhalb der Webanwendung werden zwei Exportmodule für die im Konzept vorgesehenen Dateiformate CSV und PDF implementiert. Während die Erstellung von CSV-Dateien mit der Standard-PHP-Funktion `fputcsv` erfolgt, wird für die Generierung von PDF-Dokumenten auf die frei verfügbare Klasse `TCPDF`[6] zurückgegriffen.

Da beide Module ähnlich aufgebaut sind, richtet sich der Fokus zunächst auf die Aktionen des CSV-Moduls:

- `si-details` – CSV-Export eines Incidentdatensatzes
- `si-team` – CSV-Export der Teamdaten eines Incidents
- `si-history` – CSV-Export der Verlaufsprotokoll eines Incidents
- `si-collection` – CSV-Export mehrerer Incidentdatensätze

Die Implementierung des CSV-Moduls geht aus Listing 6.30 hervor und wird am Beispiel der Aktion `si-collection` beschrieben. Der Export mehrerer Vorfalldatensätze wird im Filtermodul benötigt. Dort werden die IDs der selektierten Datensätze in der Session-Variablen `$collection` gespeichert. Wird nun die Aktion `si-collection` in der Datei `/actions/export-csv.php` aufgerufen, prüft diese zuerst in den Zeilen 1 und 2, ob der Anwender zur Nutzung dieses Moduls berechtigt ist und ob der übergebene Token mit dem Sessiontoken übereinstimmt. Nach den Sicherheitsprüfungen springt die Anwendung anhand der `$action`-Variable innerhalb des Switch-Blocks zum passenden Case (Zeilen 18 bis 23). Anhand der im Session-Kontext zwischengespeicherten Incident-IDs werden die selektierten Datensätze aus der `incidents`-Tabelle abgerufen (Zeilen 19 bis 21) und im Array `$rows` lokal gespeichert. Darüber hinaus wird in Zeile 22 der Dateiname der exportierten Datei definiert. Dieser setzt sich aus der Bezeichnung der Aktion und dem aktuellen Zeitstempel zusammen.

Während der Switch-Block die für die jeweilige Aktion benötigte Datenbankabfrage enthält, dienen die darauffolgenden Anweisungen (Zeilen 26 bis 35) der Erzeugung der CSV-Datei. Dieser Vorgang ist für alle Aktionen des Moduls identisch und wird daher universell implementiert. Zunächst wird in den Zeilen 26 und 27 ein CSV-Header gesendet, der den zuvor festgelegten Dateinamen beinhaltet und durch die Angabe der `Content-Disposition` den Browser dazu anweist, den Dateiinhalt nicht im Browserfenster darzustellen, sondern als Download zu behandeln. Damit Umlaute in der CSV-Datei korrekt dargestellt werden, wird der Zeichensatz `UTF-8` verwendet. Zudem wird in Zeile 28 ein Byte Order Mark zur Definition der Byte-Reihenfolge gesendet. Der PHP-Outputstream wird in Zeile 30 erzeugt. Anschließend werden die Spaltenbezeichnungen in Form einer Kopfzeile mit der Standard-PHP-Funktion `fputcsv` an den Outputstream übergeben (Zeile 31). Da die Spaltennamen jedoch nicht als separater Datensatz vorliegen, wird dieser mit der PHP-Funktion `array_keys` aus den Schlüsselbezeichnungen des ersten Datensatzarrays `$rows[0]` erzeugt.

Abschließend läuft eine foreach-Schleife (Zeile 33 bis 35) über alle Einträge des Datensatzarrays und fügt die Datensätze mit der Funktion `fputcsv` zeilenweise dem Outputstream hinzu (Zeile 34). Diese Funktion erwartet als ersten Parameter einen Dateihandler bzw. den

[6] `http://sourceforge.net/projects/tcpdf/`

Outputstream, an zweiter Position das Datenarray und als dritten Parameter ein Feldtrennzeichen (hier: Semikolon). Da der Datensatz `$row` vom CSV-Format nicht unterstützte Zeilenumbrüche enthalten kann, werden mit der PHP-Funktion `str_replace` alle Tab-, Newline- und Return-Befehle durch Leerzeichen ersetzt, bevor der Datensatz in die CSV-Datei geschrieben wird. Nach dem letzten Schleifendurchlauf wird mit dem Beenden des PHP-Skripts auch der Outputstream geschlossen. Daraufhin kann der Anwender die CSV-Datei auf seinem lokalen System speichern und beispielsweise mit Microsoft Excel verarbeiten.

Listing 6.30: Modul `export-csv` (`/actions/export-csv.php`)

```
checkUser("CSIRT,ROOT");
checkToken($_REQUEST['token']);

switch($_REQUEST['action']){

  case "si-details":
    // gekürzt
  break;

  case "si-team":
    // gekürzt
  break;

  case "si-history":
    // gekürzt
  break;

  case "si-collection":
    $query = "SELECT * FROM ".DBPREFIX."incidents WHERE id IN (".$_SESSION["
    collection"]).")";
    $stmt = $db->query($query);
    $rows = $stmt->fetchAll(PDO::FETCH_ASSOC);
    $csvFileName = 'si-collection-'.date("Ymd-His",time()).'.csv';
  break;
}

header("Content-Type: text/csv; charset=utf-8");
header("Content-Disposition: attachment; filename=$csvFileName");
echo "\xEF\xBB\xBF"; // UTF-8 BOM

$output = fopen('php://output', 'w');
fputcsv($output,array_keys($rows[0]),";");

foreach ($rows as $row) {
  fputcsv($output, str_replace(array("\t","\n","\r"),array(" "," "," "),$row),
    ";");
}
```

Im Gegensatz zum CSV-Format, das u.a. für den Datenimport in ein anderes Softwaresystem geeignet ist, ist das PDF-Format immer dann zu bevorzugen, wenn druckfähig formatierte Dokumente benötigt werden. Die Webanwendung unterstützt sowohl den PDF-Einzelexport eines Vorfalls (`si-details`) auf Basis der Incidentstatusseite, als auch den PDF-Massenexport von selektierten Incidents (`si-collection`) im Filtermodul. Darüber hinaus lassen sich auch Auswertungen (`stats`) im PDF-Format exportieren. Obwohl es sich beim *Portable Document Format* um ein verhältnismäßig komplexes Dateiformat handelt und zudem eine Formatierung der Rohdaten notwendig ist, weist das PDF-Modul eine grundlegende Ähnlichkeit zum CSV-Äquivalent auf.

Durch den Einsatz der Klasse `TCPDF` rückt die Komplexität des Dateiformats jedoch in den Hintergrund. Da diese Klasse HTML-Code in PDF-Syntax konvertieren kann, werden zur Formatierung der Rohdaten zwei HTML/CSS-Vorlagen für Incidents und Auswertungen erstellt. Anhand dieser Templates generiert die bereits eingesetzte Klasse `rainTPL` zur Laufzeit den HTML-Code und übergibt diesen zur Konvertierung an die PDF-Klasse. Bei der Erstellung der Template-Dateien ist zu berücksichtigen, dass der HTML- und CSS-Befehlssatz des `TCPDF`-Parsers stark eingeschränkt ist. Dennoch ist einfach strukturiertes, aber professionell wirkendes Dokumentenlayout realisierbar.

Anhand der zum vorherigen CSV-Beispiel korrespondierenden PDF-Aktion `si-collection` (vgl. Listing 6.31) wird auf Gemeinsamkeiten und Unterschiede der beiden Exportmodule eingegangen: Während die CSV-Aktion (vgl. Listing 6.30, Zeilen 18 bis 23) nur die Datensätze aus der Datenbank ausliest und den Dateinamen festlegt, generiert die PDF-Aktion darüber hinaus fertig formatierte Dokumentseiten aus den Rohdaten und fügt diese dem PDF-Objekt hinzu. Da die PDF-Aktion `si-collection` ein Dokument mit den Incidentstatusseiten der selektierten Vorfälle erzeugen soll, werden zunächst die als String übergebenen Incident IDs in ein Array konvertiert (Zeile 3) und anschließend an die foreach-Schleife (Zeile 5 bis 9) übergeben.

Die Abfrage der Incidentdaten aus der Datenbank und die Erzeugung des HTML-Codes mit der Templateklasse `rainTPL` ist in die lokale Hilfsfunktion `prepareIncidentForPdf` ausgelagert. Diese wird in Zeile 6 – also innerhalb der Schleife – aufgerufen. Anschließend wird dem `$pdf`-Objekt in Zeile 7 eine neue Seite hinzugefügt, woraufhin in Zeile 8 mit der `writeHTML`-Methode der Klasse `TCPDF` der zuvor generierte HTML-Code in PDF-Befehle konvertiert und in das `$pdf`-Objekt geschrieben wird.

Nach dem letzten Schleifendurchlauf wird in Zeile 11 analog zur CSV-Aktion der Dateiname festgelegt. Im aktionsübergreifenden Teil des PDF-Moduls wird schließlich mit dem Befehl `$pdf->Output($pdfFileName,'D');` die PDF-Datei erzeugt und an den Browser übergeben.

Listing 6.31: Aktion `si-collection` im Modul `export-pdf` (`/actions/export-pdf.php`)

```
1  case "si-collection":
2
3    $collection = explode(',',$_SESSION['collection']);
4
5    foreach ($collection as $item){
6      $pdfContent = prepareIncidentForPdf($db,$tpl,$item);
7      $pdf->AddPage();
8      $pdf->writeHTML($pdfContent, true, false, true, false, '');
9    }
10
11   $pdfFileName = 'si-colletion-'.date("Ymd-His",time()).'.pdf';
12
13 break;
```

Die `TCPDF`-Klasse ist derart konfiguriert, dass allen Dokumenten automatisch eine Kopf- und Fußzeile hinzugefügt wird. Während sich die Kopfzeile aus dem Titel des Reports, dem Erstellungsdatum und dem Autor zusammensetzt, beinhaltet die Fußzeile die aktuelle sowie die Gesamtseitenzahl. Darüber hinaus ist im Sinne eines einheitlichen Erscheinungsbildes auf jedem Report das LRZ-Logo abgebildet. Ein Musterdokument ist im Anhang D abgebildet.

7. Prototypische Anwendung der Software

Nach Abschluss der Implementierung wird nun die Funktionsweise der Software durch prototypische Anwendung anhand eines fiktiven Security Incidents demonstriert. In diesem Beispiel hat ein Mitarbeiter des LRZ festgestellt, dass die von ihm betreute WordPress Umgebung gehackt wurde. Daraufhin ruft er die im Intranet verlinkte Security Incident Managementsoftware auf und authentifiziert sich mit seiner Kennung. Nun navigiert er zum Meldeformular (Screenshot 7.1) und trägt dort die ihm bekannten Informationen zum Sicherheitsvorfall ein. Darüber hinaus ergänzt der Incidentmelder die zumeist vorausgefüllten Kontaktfelder um fehlende Angaben wie seine Handynummer. Durch das Absenden des Formulars wird ein neuer Incident angelegt und die Messung der Reaktionszeit beginnt.

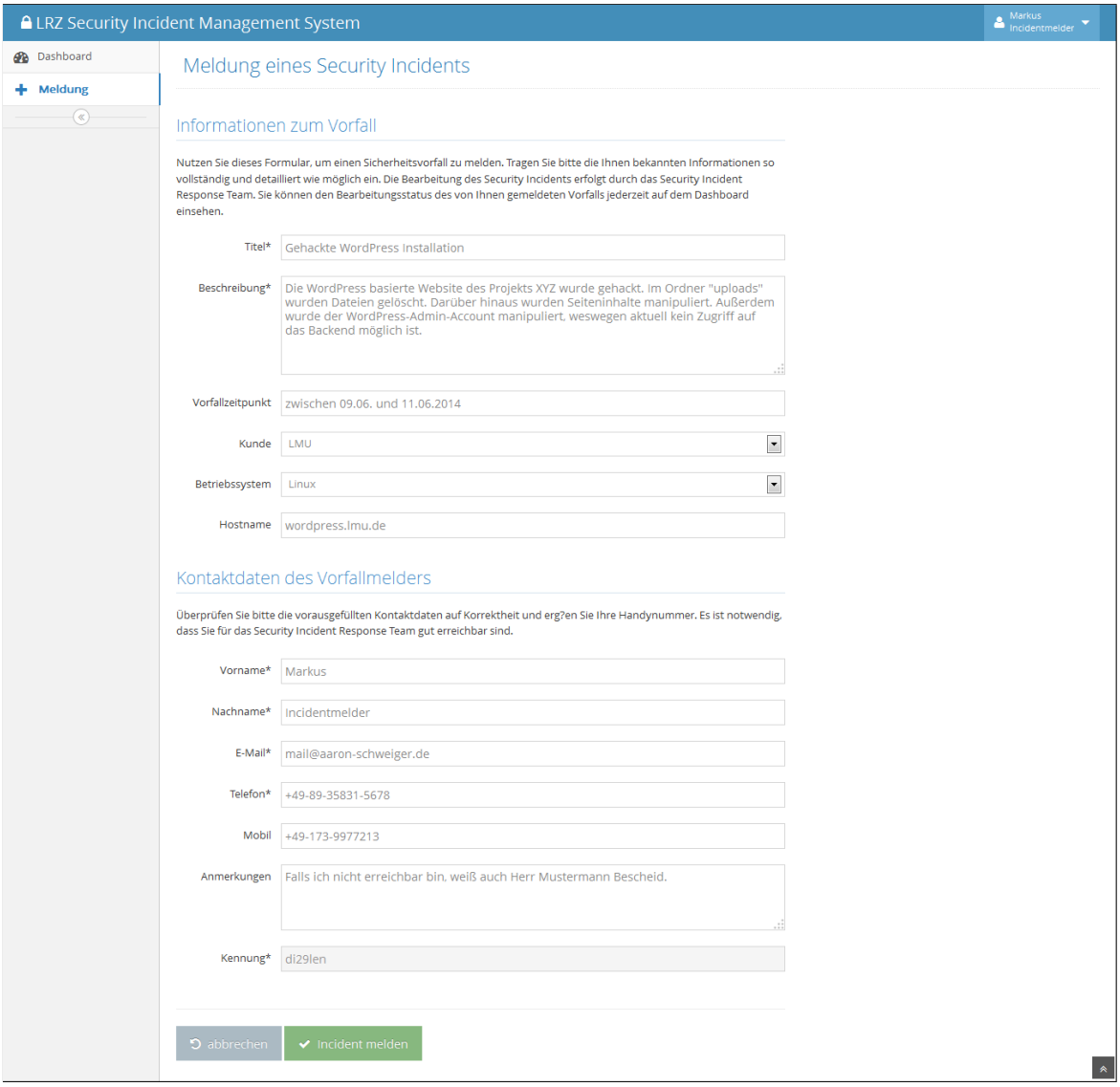

Abbildung 7.1.: Screenshot: Meldung eines Security Incidents

Sobald der Incident in der Datenbank gespeichert wurde, versendet die Webanwendung automatisch eine Bestätigungsmail an den Vorfallmelder (Abbildung 7.2) und benachrichtigt das CSIRT (Abbildung 7.3) über den Eingang eines neuen Incidents.

Von: LRZ Security Incident Management System <sims@mailingapp.de>
Gesendet: Donnerstag, 12. Juni 2014 22:13
An: Aaron Schweiger
Betreff: [SI-47] Neuer Security Incident eingegangen

Hallo Markus Incidentmelder,

wir haben Ihren Security Incident zum Thema 'Gehackte WordPress Installation' unter der ID [SI-47] erfasst und werden schnellstmöglich mit der Bearbeitung beginnen.

Im System können Sie den Status des Incidents mitverfolgen: https://di98wot.devweb.mwn.de

Beste Grüße von Ihrem

Computer Security Incident Response Team (CSIRT)

Abbildung 7.2.: Bestätigungsmail an den Vorfallmelder

Von: LRZ Security Incident Management System <sims@mailingapp.de>
Gesendet: Donnerstag, 12. Juni 2014 22:13
An: Aaron Schweiger
Betreff: [SI-47] Neuer Security Incident eingegangen

Hallo,

ein neuer Security Incident wurde erfasst. Bitte beginnen Sie schnellstmöglich mit der Bearbeitung:
https://di98wot.devweb.mwn.de

[SI-47] Gehackte WordPress Installation

Beschreibung:
Die WordPress basierte Website des Projekts XYZ wurde gehackt. Im Ordner uploads wurden Dateien gelöscht. Darüber hinaus wurden Seiteninhalte manipuliert. Außerdem wurde der WordPress-Admin-Account manipuliert, weswegen aktuell kein Zugriff auf das Backend möglich ist.

Vorfallzeitpunkt: zwischen 09.06. und 11.06.2014
Kunde: LMU
Betriebssystem: LINUX
Hostname: wordpress.lmu.de
IP-Adresse:
Meldezeitpunkt: 2014-06-12 22:00:08

Vorfallmelder

Name: Markus Incidentmelder
Telefon: +49-89-35831-5678
Mobil: +49-173-9977213
E-Mail: mail@aaron-schweiger.de

Abbildung 7.3.: Benachrichtigungsmail an das CSIRT

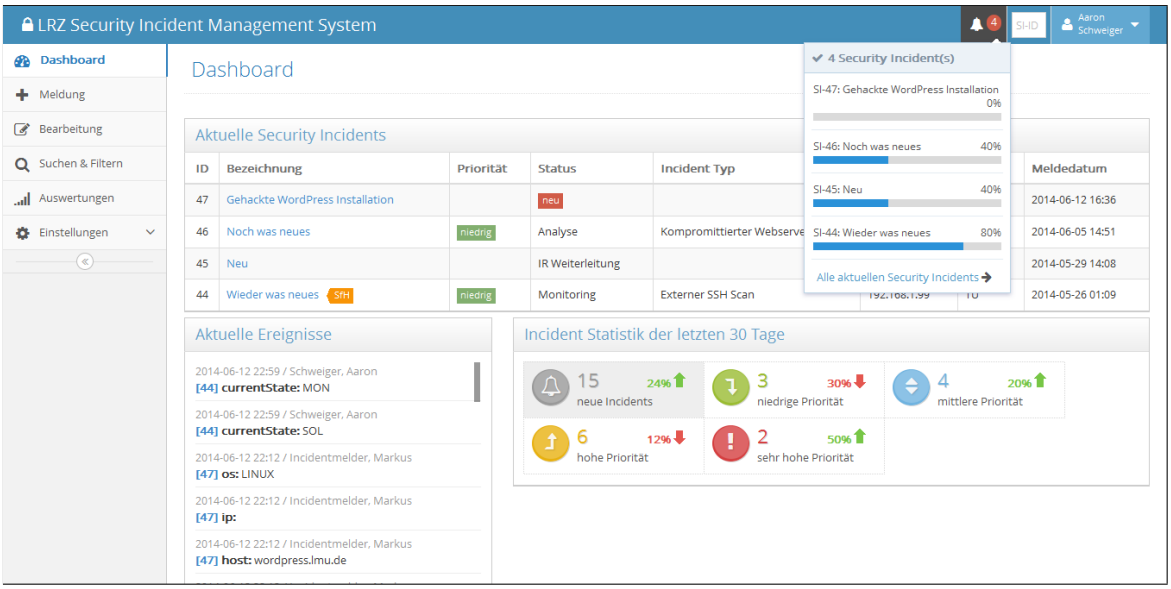

Abbildung 7.4.: Screenshot: Dashboard aus CSIRT-Sicht

Nach dem Erhalt der E-Mail meldet sich der Hotliner in der Webanwendung an und wird vom System als CSIRT-Mitglied erkannt. Daher stehen ihm alle Funktionen für die Incidentbearbeitung und darüber hinaus ein erweitertes Dashboard (Screenshot 7.4) zur Verfügung. Dieses informiert über die neuen und in Bearbeitung befindlichen Vorfälle sowie über aktuelle Ereignisse und gibt einen Überblick über das Incidentaufkommen der letzten 30 Tage. Der Zugriff auf den neu gemeldeten Incident kann sowohl über die Dashboardtabelle, in der der neue Incident rot markiert ist, als auch über den Schnellzugriff in der Kopfleiste erfolgen.

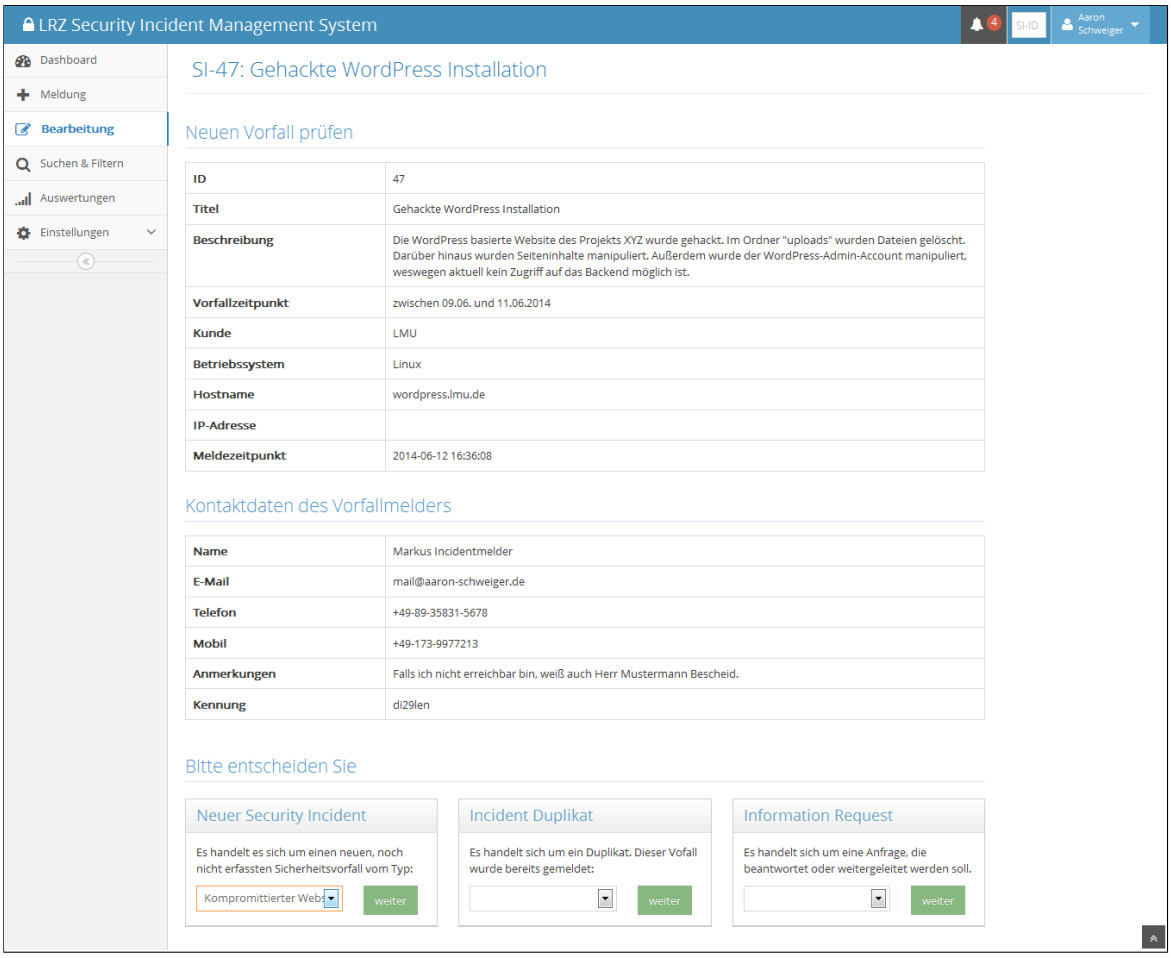

Abbildung 7.5.: Screenshot: Initiale Klassifikation

Der neue Vorfall ist zunächst initial zu klassifizieren (Screenshot 7.5). Auf Grundlage der gemeldeten Informationen entscheidet der Hotliner, ob es sich um einen noch nicht erfassten Sicherheitsvorfall, um einen bereits gemeldeten Incident oder um einen Information Request handelt. Im hier betrachteten Beispiel liegt ein neuer Security Incident vor. Der Hotliner wählt den Incidenttyp „Kompromittierter Webserver“.

Für diese Vorfallart existiert ein Standard Security Incident Template. Dieses wird nun auf den aktuellen Vorfall angewendet, sodass im Folgenden zahlreiche Formularfelder bereits mit den Standardwerten des SSI-Templates vorbelegt sind. Bei normalen, individuell zu behandelnden Sicherheitsvorfällen sind die Felder anfangs leer und im Prozessverlauf manuell auszufüllen.

Im Rahmen der weiteren Klassifikation (Screenshot 7.6) kann der Hotliner fehlende Angaben – zum Beispiel die IP-Adresse – ergänzen. Darüber hinaus wird festgelegt, ob ein Major Incident zu erwarten und ob der Hochschulstart-Dienst betroffen ist. Die für die automatische Ermittlung der Priorität benötigten Inputparameter sind hier aufgrund des SSI-Templates bereits ausgewählt. Wird die Automatik durch den Hotliner deaktiviert, so kann er Auswirkung, Priorität und Reaktionszeit auch manuell bestimmen. Nach der Festlegung der Erstmaßnahmen speichert der Hotliner seine Eingaben, wobei sämtliche Änderungen automatisch im Verlaufsprotokoll dokumentiert werden. Anschließend kann mit der Zusammenstellung des Security Incident Teams begonnen werden.

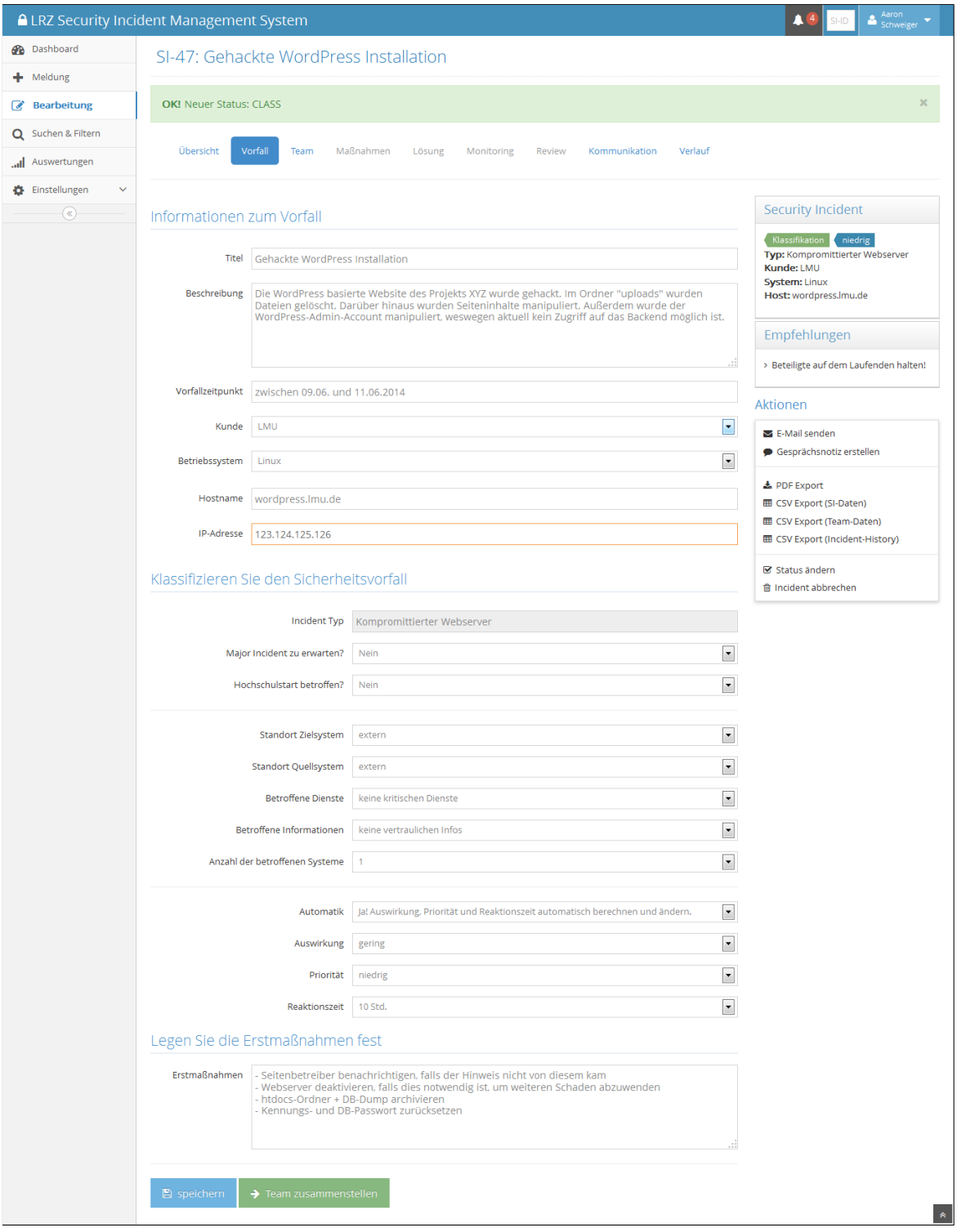

Abbildung 7.6.: Screenshot: Weitere Klassifikation und Priorisierung

Auf der Teamseite (Screenshots 7.7 und 7.8) sind zunächst die Rollen „Hotliner“ und „Security Incident Coordinator“ zu besetzen. Hierfür stehen alle CSIRT-Mitglieder in Dropdown-Feldern zur Auswahl. Darüber hinaus kann das Team gemäß Screenshot 7.9 um zusätzliche LRZ-Mitarbeiter – beispielsweise Administratoren oder Abteilungsleiter – erweitert werden. Für jedes Teammitglied stehen anhand selbsterklärender Symbole Funktionen zum Senden einer E-Mail, zur Erfassung einer Gesprächsnotiz, zum Editieren der Kontaktdaten und zum Löschen des Teammitglieds bereit. Änderungen an der Teamzusammensetzung werden ebenfalls automatisch im Verlaufsprotokoll dokumentiert. Wurden zumindest Hotliner und SIC festgelegt, kann die nächste Prozessphase gestartet werden (Screenshot 7.10).

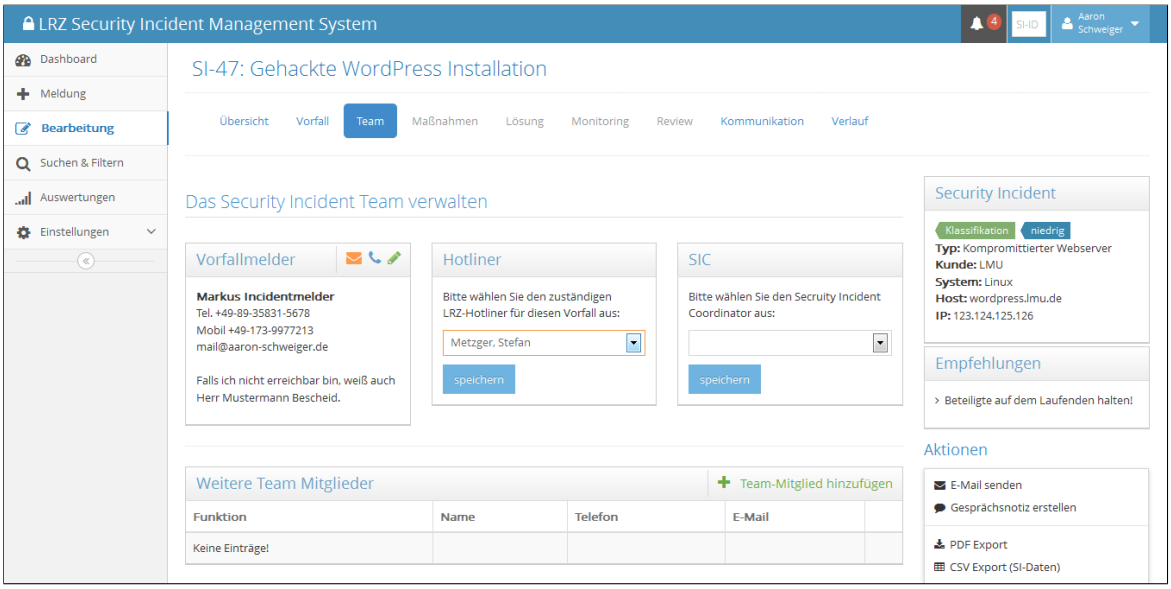

Abbildung 7.7.: Screenshot: Auswahl des Hotliners

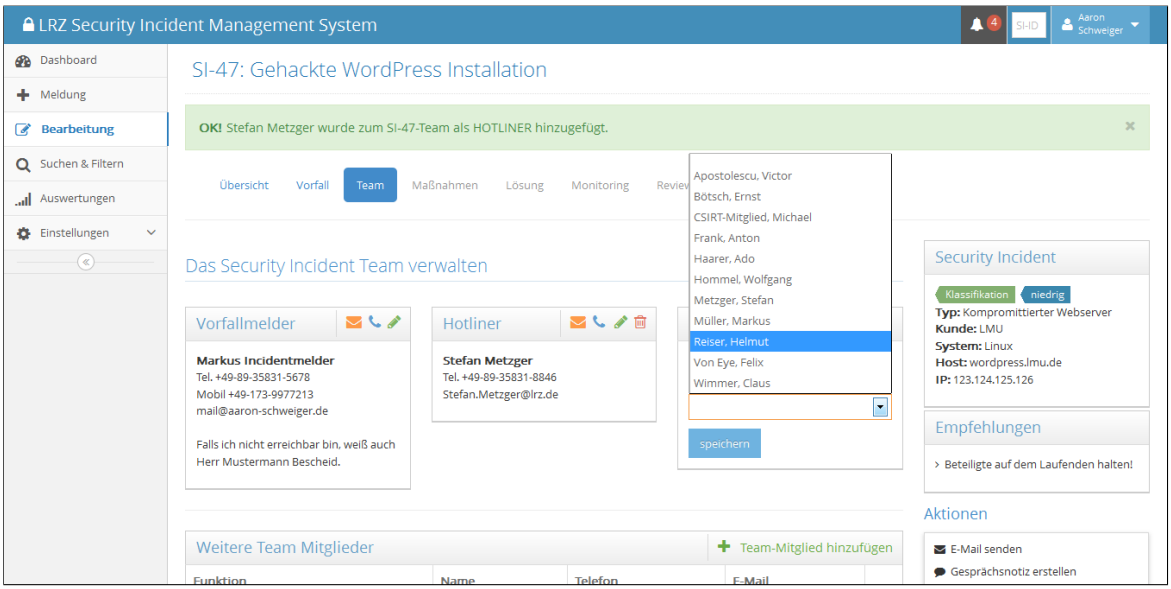

Abbildung 7.8.: Screenshot: Auswahl des Security Incident Coordinators

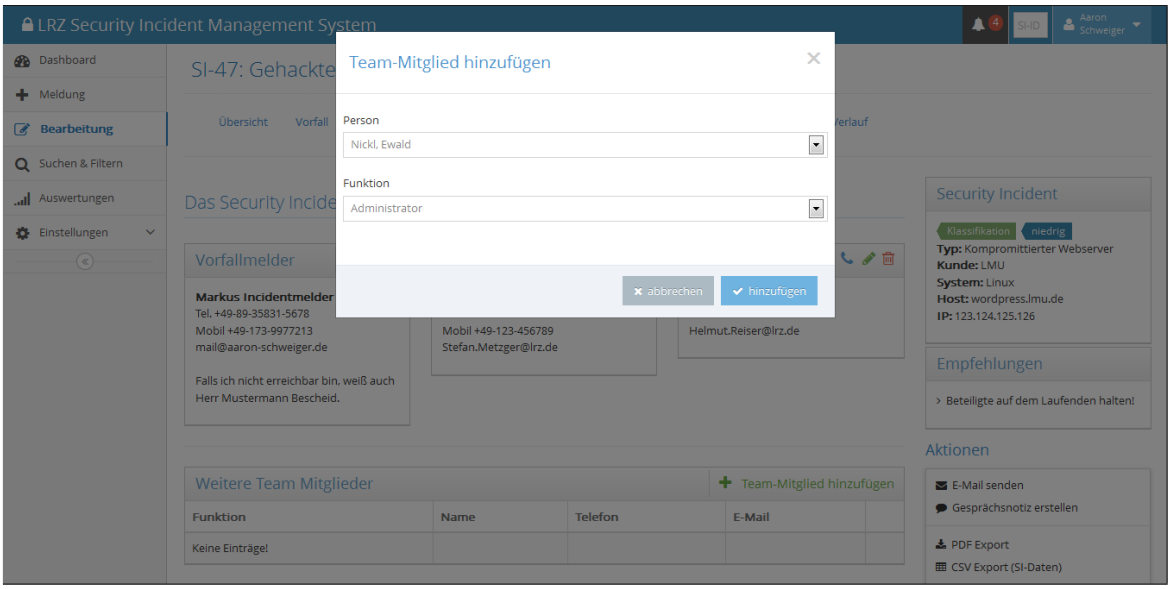

Abbildung 7.9.: Screenshot: Hinzufügen weiterer Teammitglieder

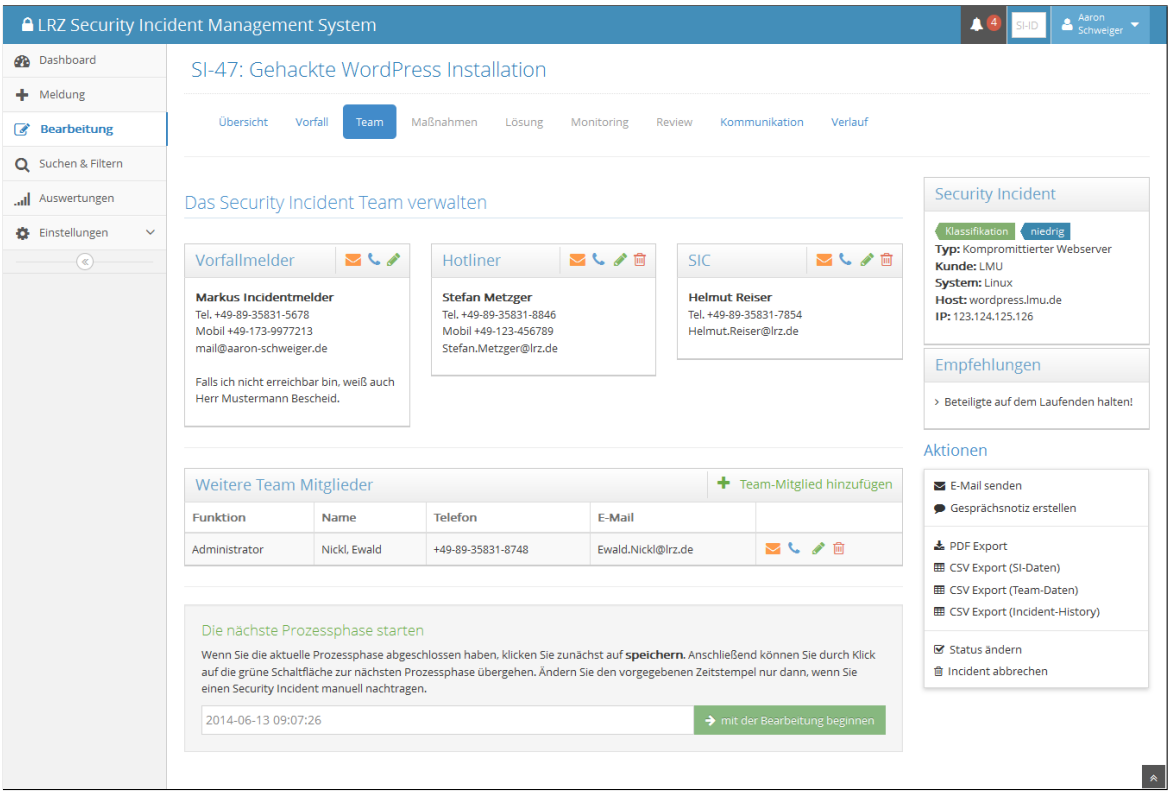

Abbildung 7.10.: Screenshot: Verwaltung des Security Incident Teams

Mit dem Beginn der Analysephase endet die Messung der Reaktionszeit. Zu diesem Zeitpunkt soll der Hotliner sowohl die Vorfallinformationen, als auch die Teamzusammensetzung und die festgelegten Erstmaßnahmen per E-Mail an die CSIRT-Mitglieder senden. Hierfür bietet es sich an, die Mailfunktion (Screenshot 7.11) des Kommunikationsmoduls zu verwenden.

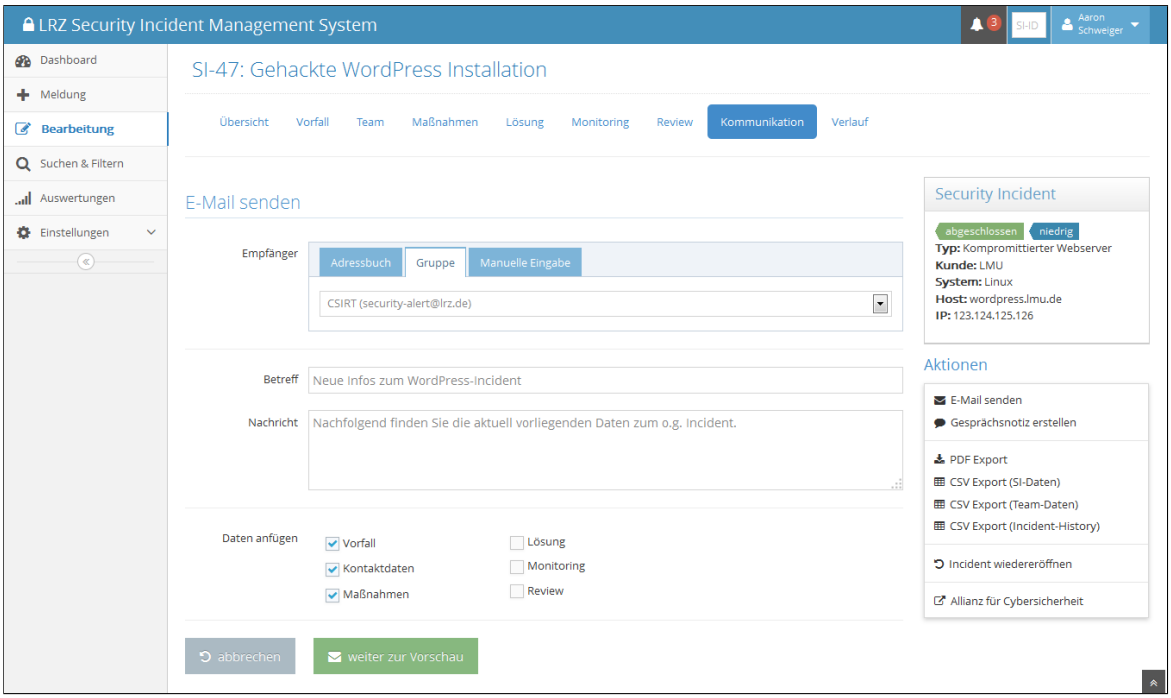

Abbildung 7.11.: Screenshot: Mailfunktion

Während der Analysephase sind zunächst die durchzuführenden Maßnahmen festzulegen und anschließend die Analyseergebnisse zu dokumentieren (Screenshot 7.12). Da es sich bei dem Beispiel um einen Standard Security Incident handelt, sind die für diesen Fall üblichen Analysemaßnahmen bereits voreingestellt. Die Eingaben lassen sich beliebig oft editieren, wobei jede Änderung nachvollziehbar dokumentiert wird.

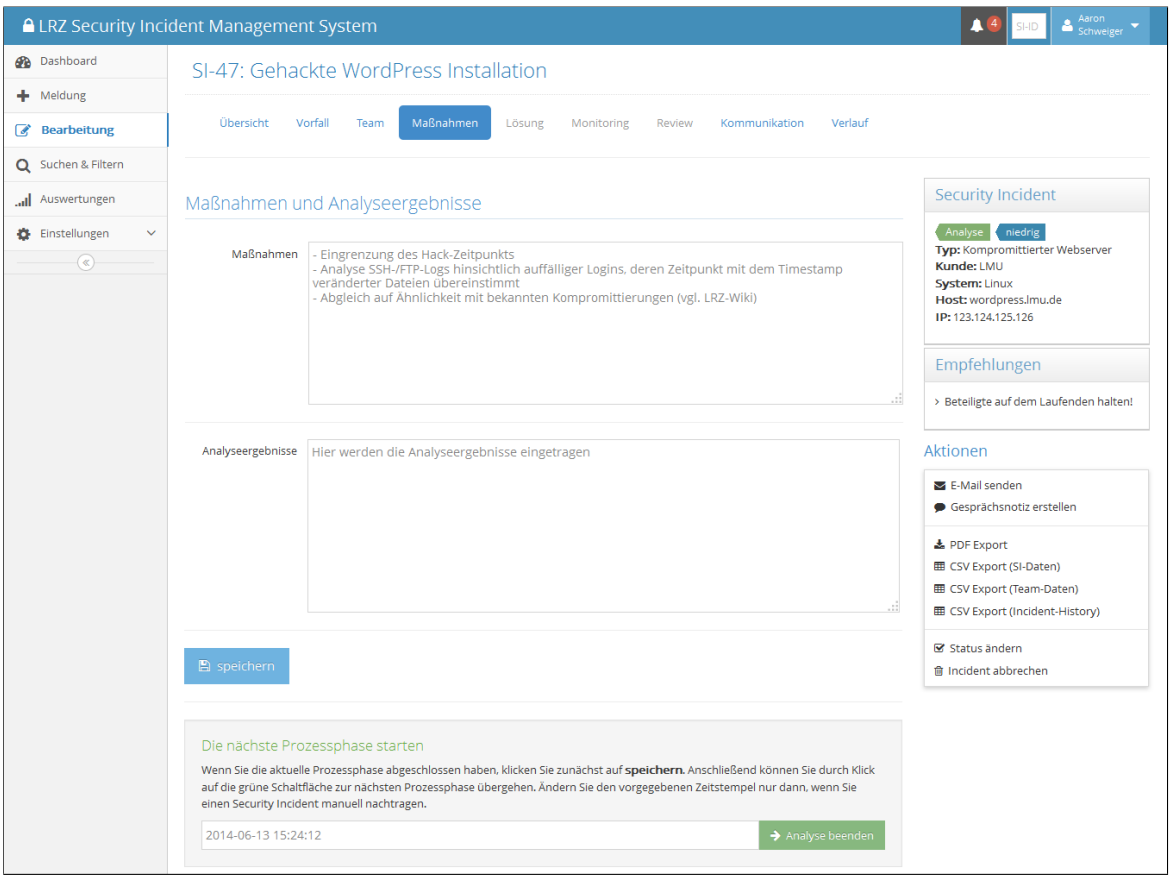

Abbildung 7.12.: Screenshot: Analysephase

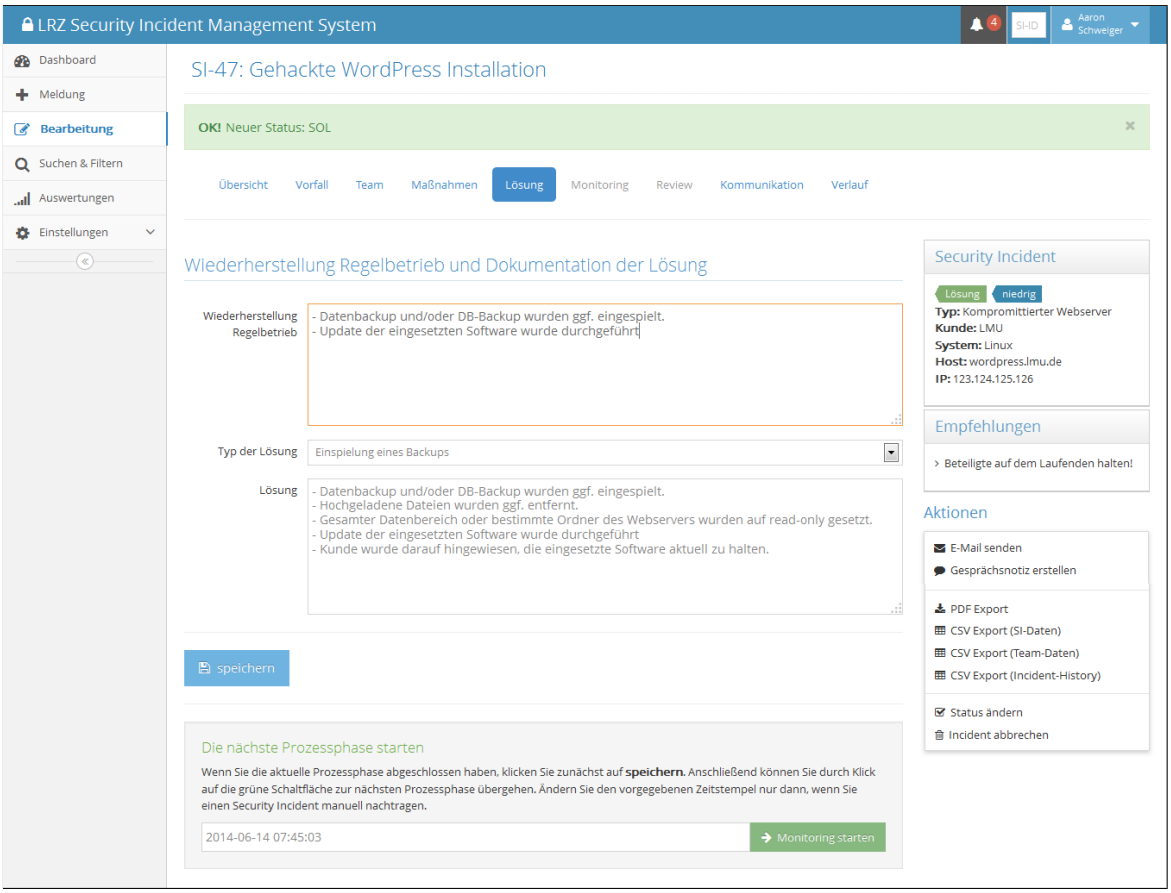

Abbildung 7.13.: Screenshot: Lösungsphase

Nach Abschluss der Analysearbeiten erfolgt der Wechsel in die Lösungsphase, wobei der Zeitstempel des Prozessphasenwechsels angepasst werden kann. Dies ist ausnahmsweise erforderlich, wenn ein Sicherheitsvorfall manuell nachgetragen wird. Mit dem Beginn der Lösungsphase endet die Messung der Analysedauer. Während dieser Phase sind die Wiederherstellung des Regelbetriebs und die Lösung des Vorfalls zu dokumentieren (Screenshot 7.13). Darüber hinaus ist der Lösungstyp aus einem Dropdown-Feld auszuwählen, um eine spätere Filterung aller Vorfälle des gleichen Lösungstyps zu ermöglichen.

In der Sidebar am rechten Rand stehen während allen Prozessphasen verschiedene Kommunikations- und Exportfunktionen bereit. Dort könnte auch der Abbruch eines Incidents bewirkt werden, falls sich ein „Fehlalarm“ herausstellen sollte. Die Sidebar beinhaltet außerdem eine Zusammenfassung der wesentlichen Vorfalldaten sowie ein Empfehlungswidget, das den Hotliner beispielsweise daran erinnert, den Vorfall an die LRZ-Leitung zu melden, falls der Incident eine hohe Priorität aufweist.

Ist die Incidentlösung abschließend dokumentiert, wird die Messung der Lösungsdauer beendet. Gleichzeitig erfolgt ein Wechsel in die Monitoringphase.

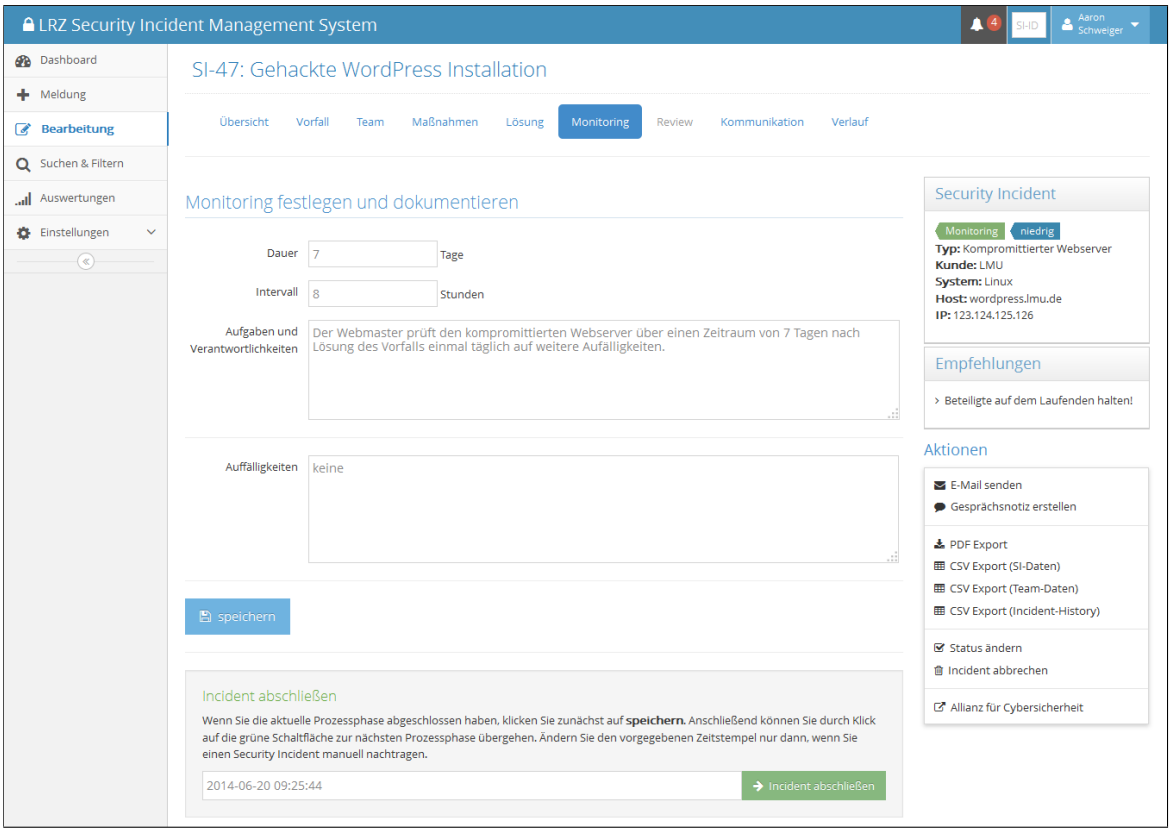

Abbildung 7.14.: Screenshot: Monitoringphase

Während dieser letzten SIR-Prozessphase werden geeignete Monitoringmaßnahmen durchgeführt und dokumentiert (Screenshot 7.14). Das in die Webanwendung integrierte Formular zur Meldung des Vorfalls an die Allianz für Cyber-Sicherheit ist in der Monitoringphase und nach Abschluss des Vorfalls über das Aktionsmenü in der Sidebar erreichbar. Im Beispiel haben sich während des Monitorings keine weiteren Auffälligkeiten ergeben, sodass der Hotliner den Vorfall nun abschließen kann. Der Incidentabschluss löst einerseits eine E-Mail an den Vorfallmelder (Abbildung 7.15) aus und beendet andererseits die Messung der Nachlaufdauer.

Von: Aaron Schweiger (LRZ SIMS) <mail@aaron-schweiger.de>
Gesendet: Donnerstag, 12. Juni 2014 23:16
An: Aaron Schweiger
Betreff: [SI-47] wurde geschlossen

Hallo Markus Incidentmelder,

wir freuen uns, Ihnen mitteilen zu dürfen, dass der Incident [SI-47] inzwischen gelöst, überwacht und soeben geschlossen wurde. Für Rückfragen stehen wir gerne zur Verfügung.

Beste Grüße

Aaron Schweiger
Computer Security Incident Response Team (CSIRT)

Abbildung 7.15.: Benachrichtigung des Vorfallmelders über den Incidentabschluss

Nach dem Abschluss des Vorfalls können keine Änderungen mehr an den Vorfalldaten vorgenommen werden. Falls sich nachträgliche Ergänzungen ergeben und neue Auffälligkeiten bekannt werden, muss der Incident wiedereröffnet werden. Die entsprechende Funktion ist im Aktionsmenü der Sidebar enthalten (Screenshot 7.16).

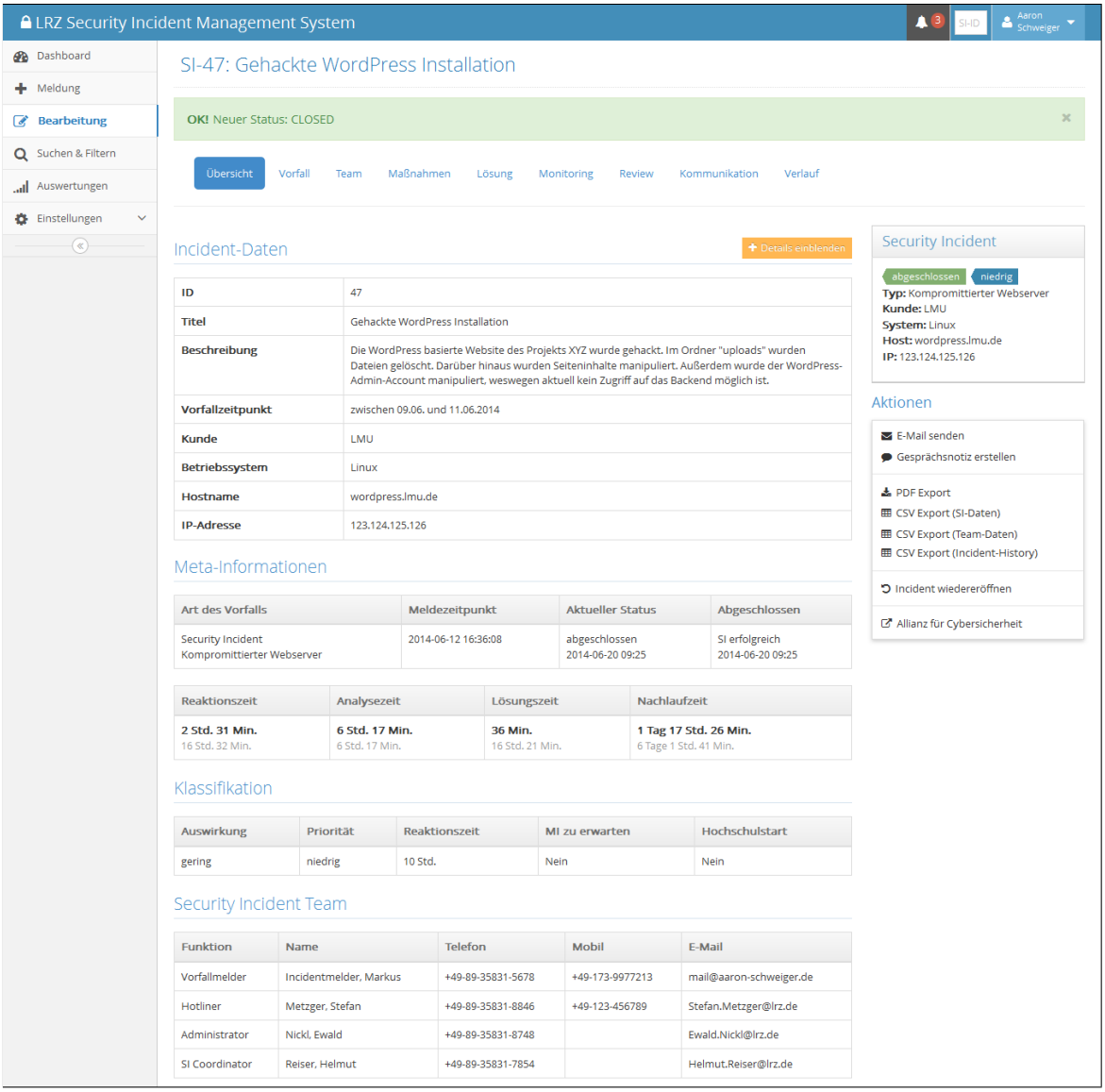

Abbildung 7.16.: Screenshot: Incidentstatusseite

Die Incidentstatusseite (Screenshot 7.16) ist in allen Prozessphasen aufrufbar und fasst die zu diesem Vorfall gespeicherten Informationen zusammen. Sie enthält auch die Laufzeiten der einzelnen Prozessphasen, wobei fett gedruckte Werte die Dauer während der Servicezeiten angeben. Die darunter stehenden Werte beziffern die Gesamtdauer. Standardmäßig wird die überblicksartige Version der Statusseite angezeigt. Jedoch lassen sich mit einem Klick alle Details einblenden.

Auch das Kommunikationsmodul steht in allen Prozessphasen zur Verfügung. Es ermöglicht den E-Mail-Versand von Incidentdaten an einzelne Teammitglieder, Gruppen sowie manuell einzutragende Empfänger. Darüber hinaus ist die Erstellung von Gesprächsnotizen möglich. Das Kommunikationsprotokoll (Screenshot 7.17) listet sämtliche Kommunikationsvorgänge auf. Für jeden Eintrag ist eine Detailseite aufrufbar.

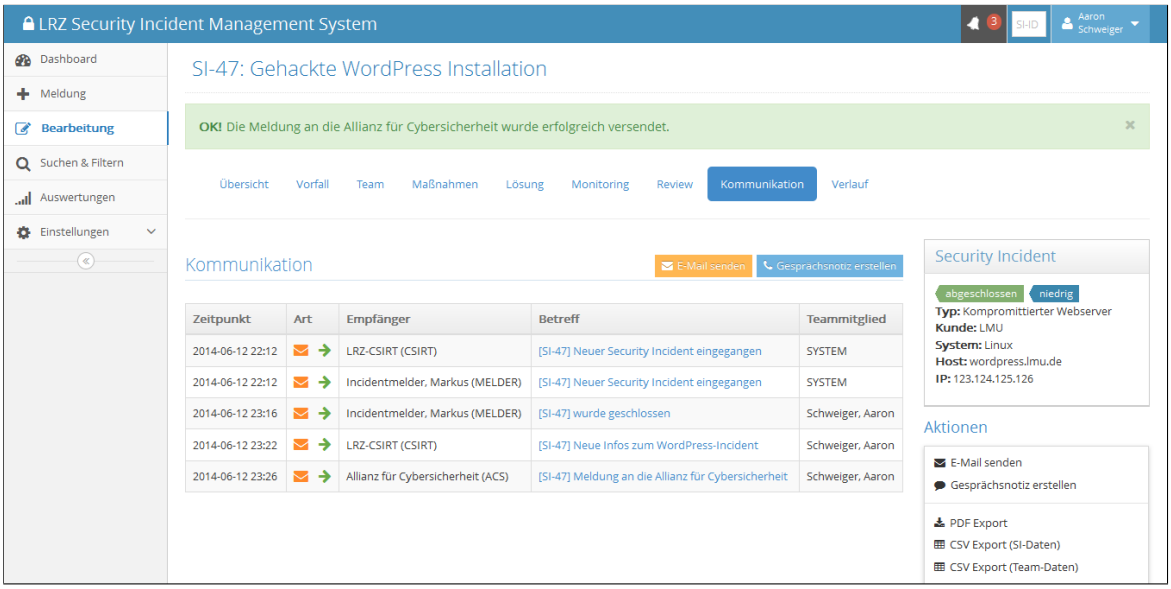

Abbildung 7.17.: Screenshot: Kommunikationsprotokoll

Anhand des incidentbezogenen Verlaufsprotokolls (Screenshot 7.18) können alle Einzelschritte der Vorfallbearbeitung nachvollzogen werden. Das Protokoll ist tabellarisch aufgebaut und kann sowohl sortiert als auch durchsucht werden. Darüber hinaus ist ein Datenexport im CSV-Format möglich.

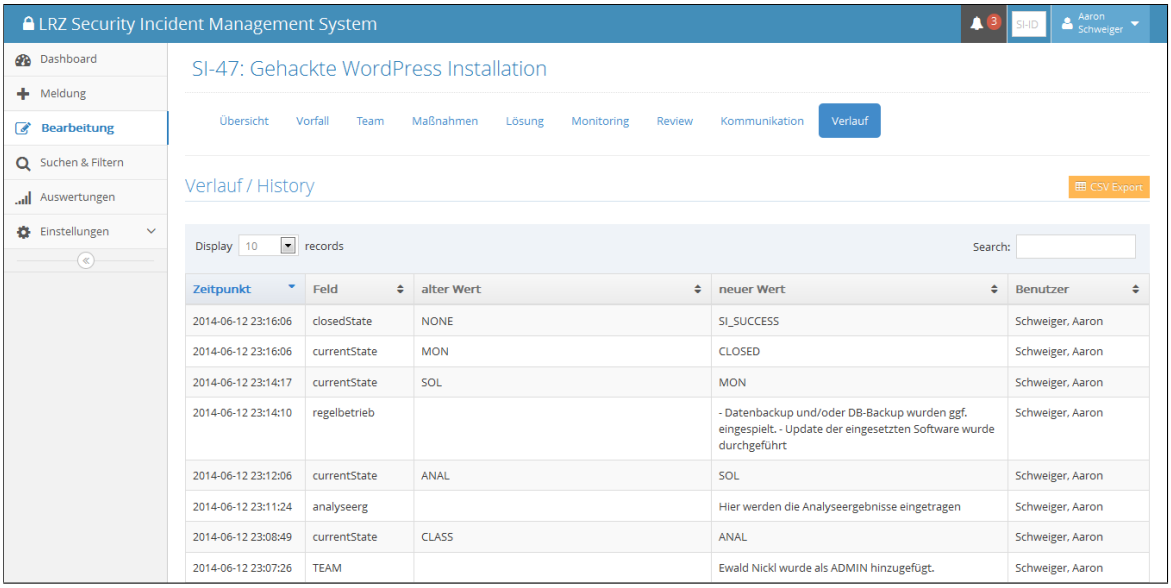

Abbildung 7.18.: Screenshot: Verlaufsprotokoll

Nach dem Abschluss des Sicherheitsvorfalls kann der Hotliner die Review-Seite (Screenshot 7.19) aufrufen, um dort die Ergebnisse des Post Incident Reviews incidentbezogen zu erfassen. Zugleich wird die Durchführung des Reviews im Verlaufsprotokoll automatisch dokumentiert. Die Review-Ergebnisse sind Bestandteil der Incidentstatusseite und des PDF-Reports und können daher auch exportiert werden.

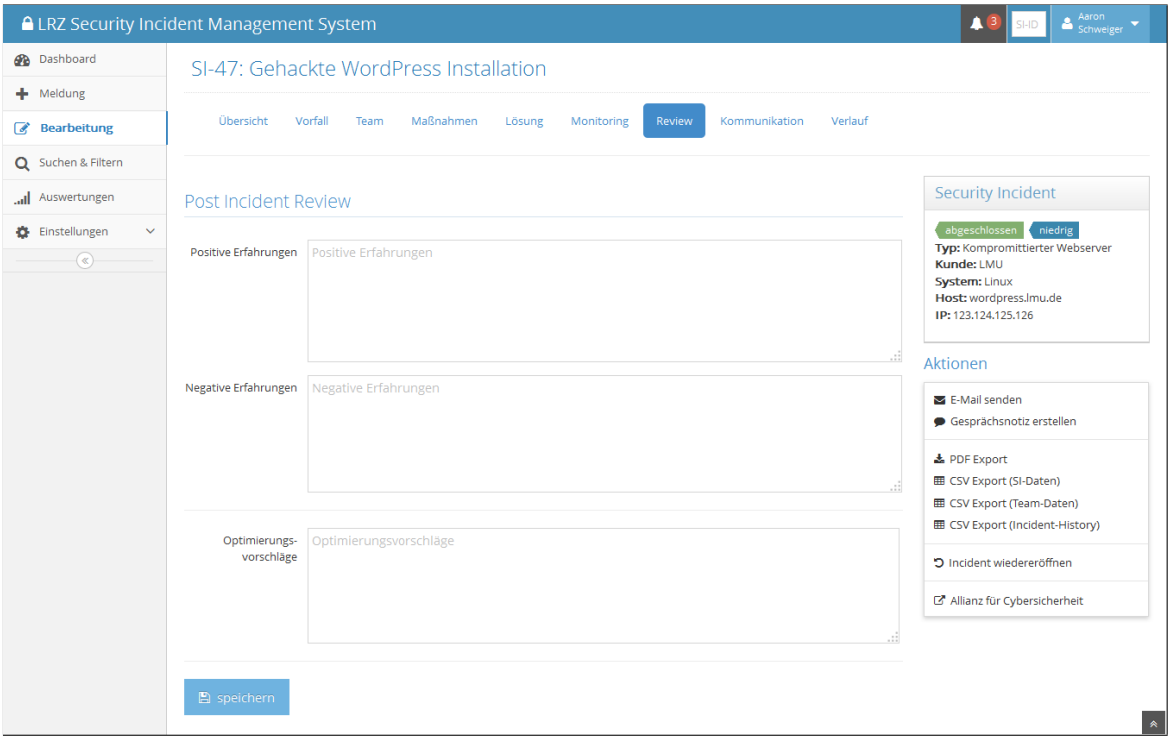

Abbildung 7.19.: Screenshot: Post Incident Review

8. Zusammenfassung und Ausblick

Im Rahmen dieser Arbeit wurde eine webbasierte Information Security Incident Managementsoftware für das Leibniz-Rechenzentrum (LRZ) sowohl konzipiert als auch implementiert. Ausgehend von nationalen und internationalen ITSM-Standards wurde der Security Incident Response Prozess des LRZ betrachtet. Die daraus abgeleiteten Möglichkeiten der Prozessoptimierung sowie die in der Aufgabenstellung verlangten Aspekte dienten der Anforderungsanalyse als Grundlage. Unter Berücksichtigung der insgesamt 40 Anforderungen wurde anschließend ein Konzept für eine individuelle Softwarelösung erarbeitet. Während der Implementierungsphase ist eine benutzerfreundliche Webanwendung entstanden, die den SIR-Prozess des LRZ digital abbildet und zu einer effizienteren Bearbeitung von Sicherheitsvorfällen beiträgt.

Während der erste Abschnitt dieses Kapitel die Ergebnisse der vorliegenden Arbeit im Bezug auf den implementierten Funktionsumfang zusammenfasst, folgt in Abschnitt 8.2 ein Ausblick auf mögliche Erweiterungen der Webanwendung.

8.1. Zusammenfassung

Entsprechend der Zielsetzung dieser Arbeit (vgl. Abschnitt 1.2) wurde ein webbasiertes, den bestehenden SIR-Prozess des LRZ abbildendes Softwaretool zur Optimierung des Information Security Incident Managements entwickelt. Dabei ermöglicht die implementierte Webanwendung im Wesentlichen die standardisierte Erfassung von Sicherheitsvorfällen, deren prozesskonforme Bearbeitung sowie die flexible Nutzung des Datenbestands.

Über die zehn in der Aufgabenstellung (vgl. Abschnitt 1.3) explizit verlangten Funktionen hinaus wurden 23 weitere Anforderungen implementiert. Dazu zählt beispielsweise der Umgang mit Incident Duplikaten ebenso wie die Bearbeitung von Information Requests. Darüber hinaus wurde eine automatisierte Vorfallpriorisierung sowie eine feingranulare Dokumentation sämtlicher Aktionen inklusive der Kommunikationsvorgänge implementiert. Eine LDAP-Integration zum Zweck der Benutzerauthentifizierung und Personendatenabfrage rundet den Funktionsumfang ab. Je nach Blickwinkel richtet sich der Fokus auf verschiedene Aspekte der Individualsoftware:

- Aus *ITSM-Sicht* zeichnet sich die Webanwendung durch eine vollständige SIR-Prozessunterstützung inklusive Zeitmessung auf Prozessphasenebene aus.
- Aus *technischer Sicht* handelt es sich um eine sichere Webanwendung auf PHP-MySQL-Basis, die eine unkomplizierte Wartung und Erweiterung ermöglicht.
- Aus der *Perspektive des Anwenders* steht die übersichtlich strukturierte und komfortabel zu bedienende Benutzeroberfläche im Mittelpunkt.

Die Konzeption und Datenmodellierung erfolgte – wie in der Aufgabenstellung gefordert – auf produkt- und plattformunabhängiger Basis. Da ein Standard-Webserver des LRZ als Entwicklungsumgebung verwendet wurde, ist die Lauffähigkeit der Webanwendung auf einem solchen Server sichergestellt.

Bei einer detaillierten Betrachtung des implementierten Funktionsumfangs ist festzustellen, dass die entstandene Webanwendung alle in der Aufgabenstellung verlangten Funktionen und 91% der ermittelten Anforderungen (gewichtet) erfüllt. Der Implementierungsgrad berechnet sich unter Berücksichtigung der Anforderungsgewichtung als Division der Summe der implementierten Anforderungen durch die Summe aller Anforderungen. Aus Abbildung 8.1 ist zu entnehmen, dass alle 18 Muss-Anforderungen vollständig umgesetzt wurden. Darüber hinaus wurden 10 von 11 der Soll-Anforderungen implementiert. Auch von den weniger relevanten Kann-Anforderungen wurden 45% erfüllt. Tabelle 8.1 zeigt den Implementierungsstatus der einzelnen Anforderungen, wobei die in der Aufgabenstellung explizit verlangten Funktionen grau hinterlegt sind.

Mit der im Rahmen dieser Arbeit entwickelten Webanwendung steht dem LRZ ein Werkzeug zur Verfügung, welches das CSIRT bei der effizienten Umsetzung des Security Incident Response Prozesses unterstützt. Im Gegensatz zur momentanen Microsoft Word und Excel basierten Incidentdokumentation ermöglicht die Webanwendung beispielsweise eine effiziente Filterung und Suche innerhalb der abgeschlossenen Security Incidents nach vergleichbaren Vorfällen. Darüber hinaus ist die technische Messung der Prozessperformance anzuführen, die eine Auswertung von KPIs zur Identifikation von Security Trends bereitstellt. Zudem bewirkt die Webanwendung eine Optimierung der Prozessperformance durch die Teilautomatisierung von organisatorischen Tätigkeiten. Als weitere Vorteile sind die Verbesserung der Qualität durch eine Steigerung der Prozesskonformität und die lückenlose Dokumentation zur Erhöhung der Nachvollziehbarkeit zu erwähnen. Letztendlich kann die Webanwendung zu einem noch professionelleren und zugleich effizienteren Umgang mit Sicherheitsvorfällen und somit zur Verbesserung des Gesamtsicherheitsniveaus am LRZ beitragen.

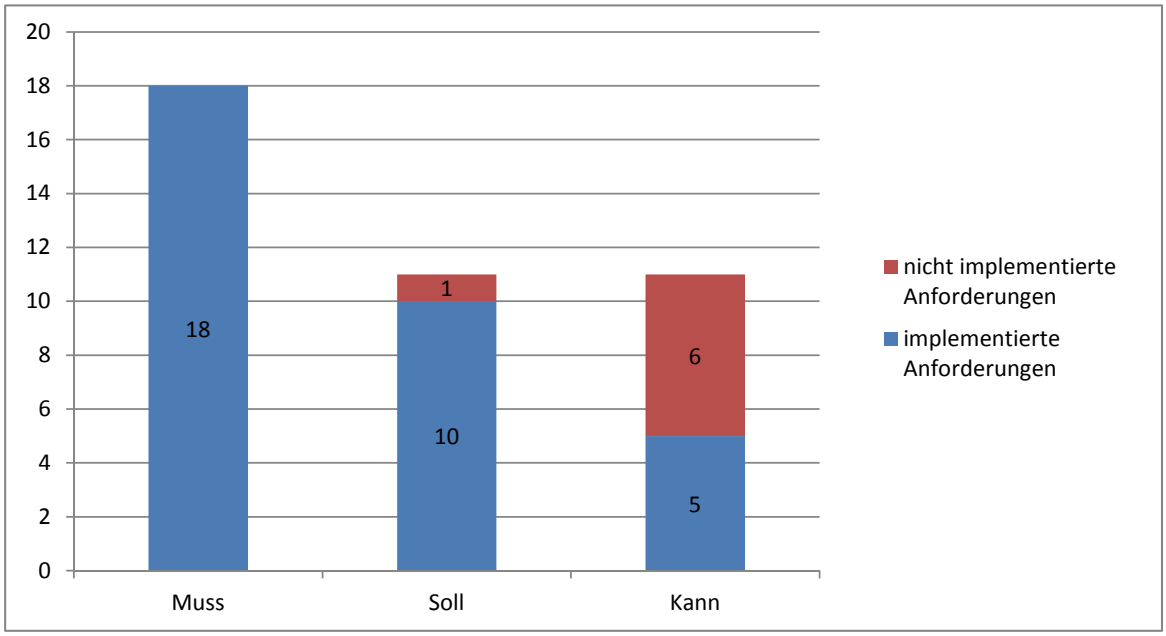

Abbildung 8.1.: Implementierter Funktionsumfang

ID	Anforderung	Impl.	Gewicht
R01	Incidenterfassung mittels Webformular	✓	3
R02	Automatisierte Incidenterfassung durch SIEM-Lösungen	∅	1
R03	Initiale Incident Klassifikation	✓	3
R04	Umgang mit Incident Duplikaten	✓	2
R05	Bearbeitung von Information Requests	✓	1
R06	Initiale Bearbeitung von Security Incidents	✓	3
R07	Unterstützung von SSIs	✓	3
R08	Priorisierung mittels Auswirkungsmatrix	✓	2
R09	Information Request an den Vorfallmelder	✓	1
R10	Weitere Incidentbearbeitung	✓	3
R11	Erweitertes Monitoring	∅	1
R12	Wiedereröffnung von abgeschlossenen Incidents	✓	2
R13	Erfassung der PIR-Ergebnisse	✓	3
R14	Archivierung abgeschlossener Security Incidents	✓	3
R15	Prozess-Unterstützung für Anwender	✓	2
R16	Dashboard	✓	1
R17	Incidentstatusseite	✓	3
R18	Dateiverwaltung	∅	2
R19	Automatisierte Handlungsempfehlungen	∅	1
R20	Dokumentation aller Aktionen	✓	2
R21	E-Mail-Benachrichtigungen	✓	3
R22	Dokumentation der Kommunikation	✓	2
R23	Meldungen an die Allianz für Cyber-Sicherheit	✓	3
R24	Such- und Filterfunktion	✓	3
R25	Tagging von Security Incidents	∅	1
R26	Einzelexport eines Security Incidents	✓	3
R27	Massenexport von Security Incidents	✓	1
R28	Einfache statistische Auswertungen	✓	3
R29	Export von Auswertungen	✓	2
R30	Automatisierter E-Mail-Versand von Auswertungen	∅	1
R31	LRZ-Branding der PDF-Dokumente	✓	1
R32	Benutzerauthentifizierung	✓	3
R33	Rechtevergabe anhand statischer Benutzergruppen	✓	2
R34	Rechtevergabe anhand dynamischer Benutzergruppen	∅	1
R35	Webbasierte Technik	✓	3
R36	Zentrale Datenhaltung in einer Datenbank	✓	3
R37	Lauffähigkeit auf einem Standard-Webserver	✓	3
R38	Gebrauchstauglichkeit und Benutzerfreundlichkeit	✓	2
R39	IT-Sicherheit	✓	3
R40	Wartbarkeit und Erweiterbarkeit	✓	2

Tabelle 8.1.: Übersicht der implementierten und nicht implementierten Anforderungen (explizit verlangte Funktionen grau hinterlegt)

8.2. Ausblick

Im Rahmen einer zukünftigen Arbeit könnte die Webanwendung um die im Folgenden beschriebenen Funktionen erweitert werden.

Als Ergänzung der klassischen Meldewege wäre eine automatisierte Incidenterfassung durch eine SIEM-Lösung wünschenswert. Hierfür wäre eine Schnittstelle zur Anbindung von Drittsystemen zu implementieren. Dadurch wäre es möglich, auch das für reguläre Störungen verwendete Ticketsystem bidirektional zu integrieren, sodass beispielsweise in der iET ITSM-Suite erfasste Störungen mit wenigen Klicks in die Security Incident Managementsoftware übertragen werden können. In der anderen Richtung könnten Information Requests in iET-Tickets konvertiert werden.

Darüber hinaus könnte die Webanwendung mit einem Modul zur Dateiverwaltung ausgestattet werden. Dadurch könnten im Rahmen der Beweissicherung gesammelte Dateien wie beispielsweise Logfiles incidentbezogen zentral gespeichert und aus der Webanwendung heraus geöffnet werden. In diesem Zusammenhang könnte das Meldeformular um eine Uploadmöglichkeit ergänzt werden, sodass der Vorfallmelder beispielsweise Auszüge aus Logfiles oder andere Dateien direkt an das CSIRT übermitteln kann. Verschiedene Ansätze für eine technische Umsetzung werden in Abschnitt 5.2 diskutiert.

Da während der Incidentbearbeitung das Hinzuziehen externer Personen wie zum Beispiel IT-Forensiker erforderlich sein kann, erscheint es sinnvoll, den Bereich der Teamzusammenstellung diesbezüglich zu ergänzen, sodass externe Personen, deren Kontaktdaten nicht via LDAP abrufbar sind, manuell hinzugefügt werden können.

Das Kommunikationsmodul könnte dahingehend erweitert werden, dass sich die Anwender innerhalb des Systems gegenseitig Nachrichten zuschicken und Dateianhänge beifügen können. Der E-Mail-Versand wäre dann nur noch für externe Parteien relevant, die über keinen Systemzugriff verfügen.

Das System könnte anhand von Parallelfällen dem Anwender die Lessons Learned aufzeigen und konkrete Handlungsempfehlungen für die Festlegung der weiteren Vorgehensweise bieten. Durch die Empfehlungsautomatik könnte vorhandenes Wissen optimal in den Prozessablauf eingebunden werden. Weitere Überlegungen zu einer derartigen Wissensmanagementfunktionalität werden in Abschnitt 5.3.3 diskutiert.

Um eine noch höhere Qualität bei der Incident Nachsorge zu erzielen, könnten die vorhandenen Monitoringfunktionen erweitert werden, wobei das System die verantwortlichen Personen per E-Mail pünktlich an die Durchführung der jeweiligen Monitoringaktivitäten erinnert. Die Erledigung könnte durch Klick auf einen Link in der E-Mail bestätigt werden. Außerdem wäre ein Rückkanal für die Mitteilung etwaiger Auffälligkeiten einzurichten.

Weiterhin könnte eine Tagging-Funktion ergänzt werden, die das Anreichern von Security Incidents mit flexiblen Metainformationen ermöglicht. Durch die Verwendung gleicher Tags können ähnliche Vorfälle flexibel gruppiert werden. Diese zusätzlichen Metadaten könnten zum Zweck einer verfeinerten Filterung und Auswertung genutzt werden.

Da anhand der relativen Zu- oder Abnahme von Kennzahlen im Zeitverlauf Trends leichter zu identifizieren sind, könnten bei einer Erweiterung des Auswertungsmoduls die Werte des Auswertungszeitraums in Relation zu einem Referenzzeitraum gesetzt werden, sodass prozentuale Veränderungen sichtbar werden. Zudem wäre ein automatisierter Reportversand an das CSIRT denkbar.

Ebenso wäre es sinnvoll, wenn beteiligte Systemadministratoren und Abteilungsleiter den aktuellen Stand eines Vorfalls einsehen und beispielsweise Analyse- oder Monitoringergebnisse direkt im System erfassen könnten. Hierfür wäre eine Rechtevergabe anhand dynamischer Benutzergruppen notwendig, die bereits für den Zugriff auf die Incidentstatusseite realisiert wurde. Diese Funktionalität könnte zusätzlich im Bereich der Incidentbearbeitung angewendet werden, wobei zunächst ein differenziertes Konzept für die Vergabe von Lese- und Schreibrechten auf Objektebene zu erarbeiten wäre.

Um die Webanwendung im Kontext einer organisationsübergreifenden Vorfallbearbeitung einsetzen zu können, wären vor allem in den Bereichen Benutzerverwaltung, Authentifizierung und Rechtevergabe Anpassungen notwendig.

Da bei der Implementierung auf eine unkomplizierte Erweiterbarkeit der Webanwendung geachtet wurde, sollte eine Weiterentwicklung entsprechend der vorgenannten Aspekte problemlos möglich sein.

Abkürzungsverzeichnis

ACS	Allianz für Cyber-Sicherheit
BSB	Bayerische Staatsbibliothek
BSI	Bundesamt für Sicherheit in der Informationstechnik
BYOD	Bring Your Own Device
CSIRT	Computer Security Incident Response Team
CSS	Cascading Style Sheets
CSV	Comma-separated values
DEISA	Distributed European Infrastructure for Supercomputing Applications
DFN	Deutsches Forschungsnetz
EGI	European Grid Infrastructure
ENISA	European Union Agency for Network and Information Security
FTP	File Transfer Protocol
GUI	Graphical User Interface
HM	Hochschule für angewandte Wissenschaften München
HTML	Hypertext Markup Language
ISMS	Information Security Management System
ISM	Information Security Management
ITIL	IT Infrastructure Library
ITSM	IT Service Management
IT	Informationstechnik *(engl. information technology)*
KPI	Key-Performance-Indikator
LDAP	Lightweight Directory Access Protocol
LGPL	Lesser General Public License
LMU	Ludwig-Maximilians-Universität München
LRZ	Leibniz-Rechenzentrum
MD5	Message-Digest Algorithm 5
MIC	Major Incident Coordinator
MIT	Massachusetts Institute of Technology

MWN Münchner Wissenschaftsnetz
NIST National Institute of Standards and Technology
PDF Portable Document Format
PDO PHP Data Objects
PHP PHP: Hypertext Preprocessor
PIR Post Incident Review
RSS Really Simple Syndication
SIC Security Incident Coordinator
SIEM Security Information and Event Management
SIM Security Incident Management
SIR Security Incident Response
SLA Service Level Agreement
SMB Server Message Block
SMS Short Message Service
SMTP Simple Mail Transfer Protocol
SNMP Simple Network Management Protocol
SQL Structured Query Language
SSH Secure Shell
SSI Standard Security Incident
SSL Secure Sockets Layer *(alte Bezeichnung für TLS)*
TLS Transport Layer Security
TUM Technische Universität München
VPN Virtual Private Network
XML Extensible Markup Language
XSS Cross-Site-Scripting

Abbildungsverzeichnis

Tabellenverzeichnis

Quellcodeverzeichnis

Literaturverzeichnis

[Ark13a] Arkin, Brad: *Illegal Access to Adobe Source Code*. In: *Adobe Secure Software Engineering Team Blog*. Oktober 2013. http://blogs.adobe.com/asset/2013/10/illegal-access-to-adobe-source-code.html.

[Ark13b] Arkin, Brad: *Important Customer Security Announcement*. In: *Adobe Featured Blogs*. Oktober 2013. http://blogs.adobe.com/conversations/2013/10/important-customer-security-announcement.html.

[BIT13] BITKOM Bundesverband Informationswirtschaft, Telekommunikation und neue Medien e.V.: *Leitfaden - Bring Your Own Device*, April 2013. http://www.bitkom.org/files/documents/20130404_LF_BYOD_2013_v2.pdf.

[Bun09] Bundesamt für Sicherheit in der Informationstechnik: *IT-Grundschutz-Baustein B 1.8 Behandlung von Sicherheitsvorfällen*, 2009. https://www.bsi.bund.de/DE/Themen/ITGrundschutz/ITGrundschutzKataloge/Inhalt/_content/baust/b01/b01008.html.

[Bun11] Bundesamt für Sicherheit in der Informationstechnik: *Leitfaden IT-Forensik*, März 2011. https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Internetsicherheit/Leitfaden_IT-Forensik_pdf.pdf.

[Bun12a] Bundesamt für Sicherheit in der Informationstechnik: *Goldene Regeln für den IT-Grundschutz-Baustein B 5.21 Webanwendungen*, November 2012. https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Grundschutz/Download/Vorabversionen/Baustein_Webanwendungen_Goldene_Regeln.pdf.

[Bun12b] Bundesamt für Sicherheit in der Informationstechnik: *IT-Grundschutz-Baustein B 5.21 Webanwendungen*, April 2012. https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Grundschutz/Download/Vorabversionen/Baustein_Webanwendungen.pdf.

[Bun13] Bundesamt für Sicherheit in der Informationstechnik: *Fokus IT-Sicherheit 2013*, Juli 2013. https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/Lageberichte/Fokus_IT-Sicherheit_2013_nbf.pdf.

[Bun14] Bundesamt für Sicherheit in der Informationstechnik: *Allianz für Cyber-Sicherheit*, Juni 2014. https://www.bsi.bund.de/DE/Themen/Cyber-Sicherheit/Strategie/Allianz_fuer_Cybersicherheit/Allianz_node.html.

[CMGS12] Cichonski, Paul, Tom Millar, Tim Grance und Karen Scarfone: *Computer Security Incident Handling Guide. Recommendations of the National Institute of Standards and Technology. NIST Special Publication 800-61 Revision 2*, August 2012. `http://csrc.nist.gov/publications/nistpubs/800-61rev2/SP800-61rev2.pdf`.

[Fra13] Frankfurter Allgemeine Zeitung: *Hacker stehlen Daten von Millionen Adobe-Kunden*, Oktober 2013. `http://www.faz.net/-gqm-7i4k9`.

[Fre13] Freund, Rosa: *Standard-Security-Incident 0002. Prozessbeschreibung. Umgang mit kompromittierten Webservern.* Internes Dokument des Leibniz-Rechenzentrums, Mai 2013.

[Ges11] Geschonneck, Alexander: *Computer-Forensik: Computerstraftaten erkennen, ermitteln, aufklären.* iX-Edition. Dpunkt.Verlag GmbH, September 2011.

[Hei13] Heise Security: *Einbruch bei Adobe: Millionen Kundendaten sowie Sourcecode von ColdFusion und Acrobat geklaut*, Oktober 2013. `http://heise.de/-1972175`.

[ISO13a] ISO/IEC 27000:2014: *Information technology – Security techniques – Information security management systems – Overview and vocabulary*, 2013. ISO/IEC, Geneva, Switzerland.

[ISO13b] ISO/IEC 27001:2013: *Information technology – Security techniques – Information security management systems – Requirements*, 2013. ISO/IEC, Geneva, Switzerland.

[ISO13c] ISO/IEC 27002:2013: *Information technology – Security techniques – Code of practice for information security controls*, 2013. ISO/IEC, Geneva, Switzerland.

[ISO13d] ISO/IEC 27035 (Working Draft): *Information technology – Security techniques – Information security incident management – Part 1-3*, 2013. ISO/IEC, Geneva, Switzerland.

[Kle13] Kleiner, Fritz: *IT Service Management: Aus der Praxis für die Praxis.* Springer Vieweg, Wiesbaden, 2013.

[Klu13] Kluge, Hans-Georg: *Adobe-Datenklau: Passwörter von Adobe-Kunden in Gefahr*, November 2013. `http://www.teltarif.de/adobe-datenklau-passwoerter-gefahr-hack/news/53207.html`.

[Met13] Metzger, Stefan: *Information Security Incident Management. Prozessbeschreibung.* Internes Dokument des Leibniz-Rechenzentrums, Dezember 2013.

[MHR11] Metzger, Stefan, Wolfgang Hommel und Helmut Reiser: *Integriertes Management von Sicherheitsvorfällen.* In: *Sicherheit in vernetzten Systemen, 18. DFN Workshop.* DFN-CERT Services, Hamburg, Februar 2011.

[Pip12] Pipek, Volkmar: *Ubiquitous Computing.* In: *Enzyklopädie der Wirtschaftsinformatik. Online-Lexikon.* Oldenbourg Wissenschaftsverlag, Oktober 2012. `http://www.enzyklopaedie-der-wirtschaftsinformatik.de/wi-enzyklopaedie/lexikon/technologien-methoden/Rechnernetz/Ubiquitous-Computing`.

[Stü13] Stückler, Moritz: *Warum der Adobe-Hack noch viel schlimmer ist als bisher angenommen*, November 2013. `http://t3n.de/news/adobe-hack-noch-viel-schlimmer-506598/`.

[Wei91] Weiser, Mark: *The Computer for the 21st Century.* Scientific American, 265(3):66–75, January 1991.

Anhang

A. SSI-Template: „Umgang mit kompromittierten Webservern“

MI zu erwarten	Nein
Hochschulstart betroffen	Nein
Standort Zielsystem	MWN, Grid-Systeme
Standort Quellsystem	extern
Betroffene Dienste	keine kritischen Dienste
Betroffene Informationen	keine vertraulichen Infos
Anzahl der betroffenen Systeme	1
Auswirkung	gering
Priorität	niedrig
Reaktionszeit	10 Std.
Erstmaßnahmen	- Seitenbetreiber benachrichtigen, falls der Hinweis nicht von diesem kam - Webserver deaktivieren, falls dies notwendig ist, um weiteren Schaden abzuwenden - htdocs-Ordner und DB-Dump archivieren - Kennungs- und DB-Passwort zurücksetzen
Maßnahmen	- Eingrenzung des Hack-Zeitpunkts - Analyse SSH-/FTP-Logs hinsichtlich auffälliger Logins, deren Zeitpunkt mit dem Timestamp veränderter Dateien übereinstimmt - Abgleich auf Ähnlichkeit mit bekannten Kompromittierungen (vgl. LRZ-Wiki)
Analyseergebnisse	*individuell*
Regelbetrieb	*individuell*
Typ der Lösung	Backup eingespielt
Lösung	- Datenbackup und/oder DB-Backup wurden ggf. eingespielt. - Hochgeladene Dateien wurden ggf. entfernt. - Gesamter Datenbereich oder bestimmte Ordner des Webservers wurden auf read-only gesetzt. - Update der eingesetzten Software wurde durchgeführt. - Kunde wurde darauf hingewiesen, die eingesetzte Software aktuell zu halten.
Monitoring-Dauer	7 Tage
Monitoring-Intervall	10 Stunden
Monitoring Aufgaben und Verantwortlichkeiten	Der Webmaster prüft den kompromittierten Webserver über einen Zeitraum von 7 Tagen nach Lösung des Vorfalls einmal täglich auf weitere Auffälligkeiten.
Auffälligkeiten	keine

Tabelle A.1.: SSI-Template: „Umgang mit kompromittierten Webservern“

B. Datenbank-Setup

Listing B.1: Datenbank-Setup: Erstellung der Tabellen

```
-- Tabellenstruktur für Tabelle `sims_communication`
CREATE TABLE IF NOT EXISTS `sims_communication` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `si` mediumint(9) DEFAULT NULL,
  `user` int(11) DEFAULT NULL,
  `userName` varchar(160) COLLATE utf8_unicode_ci DEFAULT NULL,
  `stamp` timestamp NULL DEFAULT CURRENT_TIMESTAMP,
  `comStamp` timestamp NULL DEFAULT NULL,
  `medium` enum('TEL','MAIL','PERS') COLLATE utf8_unicode_ci DEFAULT NULL,
  `richtung` enum('IN','OUT') COLLATE utf8_unicode_ci DEFAULT NULL,
  `subject` varchar(255) COLLATE utf8_unicode_ci DEFAULT NULL,
  `content` text COLLATE utf8_unicode_ci,
  `name` varchar(160) COLLATE utf8_unicode_ci DEFAULT NULL,
  `tel` varchar(160) COLLATE utf8_unicode_ci DEFAULT NULL,
  `email` varchar(160) COLLATE utf8_unicode_ci DEFAULT NULL,
  `contactRole` varchar(12) COLLATE utf8_unicode_ci DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `si` (`si`)
) ENGINE=MyISAM  DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

-- Tabellenstruktur für Tabelle `sims_contacts`
CREATE TABLE IF NOT EXISTS `sims_contacts` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `si` mediumint(9) NOT NULL,
  `contactRole` varchar(12) COLLATE utf8_unicode_ci DEFAULT NULL,
  `kennung` varchar(40) COLLATE utf8_unicode_ci DEFAULT NULL,
  `name` varchar(160) COLLATE utf8_unicode_ci DEFAULT NULL,
  `vorname` varchar(160) COLLATE utf8_unicode_ci DEFAULT NULL,
  `anrede` enum('Frau','Herr') COLLATE utf8_unicode_ci DEFAULT NULL,
  `email` varchar(160) COLLATE utf8_unicode_ci DEFAULT NULL,
  `tel` varchar(160) COLLATE utf8_unicode_ci DEFAULT NULL,
  `mobil` varchar(160) COLLATE utf8_unicode_ci DEFAULT NULL,
  `comment` varchar(255) COLLATE utf8_unicode_ci DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `si` (`si`)
) ENGINE=MyISAM  DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

-- Tabellenstruktur für Tabelle `sims_history`
CREATE TABLE IF NOT EXISTS `sims_history` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `si` mediumint(9) DEFAULT NULL,
  `user` int(11) DEFAULT NULL,
  `stamp` timestamp NULL DEFAULT CURRENT_TIMESTAMP,
  `field` varchar(40) COLLATE utf8_unicode_ci DEFAULT NULL,
  `old` text COLLATE utf8_unicode_ci,
  `new` text COLLATE utf8_unicode_ci,
  `userName` varchar(160) COLLATE utf8_unicode_ci DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `id` (`id`),
  KEY `si` (`si`)
) ENGINE=MyISAM  DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;


```

```
-- Tabellenstruktur für Tabelle `sims_incidents`
CREATE TABLE IF NOT EXISTS `sims_incidents` (
  `id` mediumint(9) NOT NULL AUTO_INCREMENT,
  `duplicateId` mediumint(9) DEFAULT NULL,
  `user` int(11) DEFAULT NULL,
  `stamp` timestamp NULL DEFAULT CURRENT_TIMESTAMP,
  `createdStamp` timestamp NULL DEFAULT NULL,
  `closedStamp` timestamp NULL DEFAULT NULL,
  `closedState` enum('NONE','IR_AW','IR_FWD','SI_DUPLICATE','SI_CANCEL','
    SI_SUCCESS') COLLATE utf8_unicode_ci NOT NULL DEFAULT 'NONE',
  `currentStamp` timestamp NULL DEFAULT NULL,
  `currentState` enum('NEW','CLASS','ANAL','SOL','MON','CLOSED','IR_AW','
    IR_FWD') COLLATE utf8_unicode_ci DEFAULT NULL,
  `reopened` tinyint(1) DEFAULT NULL,
  `saldo1` int(11) NOT NULL DEFAULT '0',
  `saldo2` int(11) NOT NULL DEFAULT '0',
  `saldo3` int(11) NOT NULL DEFAULT '0',
  `saldo4` int(11) NOT NULL DEFAULT '0',
  `saldo1bh` int(11) NOT NULL DEFAULT '0',
  `saldo2bh` int(11) NOT NULL DEFAULT '0',
  `saldo3bh` int(11) NOT NULL DEFAULT '0',
  `saldo4bh` int(11) NOT NULL DEFAULT '0',
  `title` varchar(255) COLLATE utf8_unicode_ci DEFAULT NULL,
  `description` text COLLATE utf8_unicode_ci,
  `vorfallzeitpunkt` varchar(255) COLLATE utf8_unicode_ci DEFAULT NULL,
  `kunde` varchar(12) COLLATE utf8_unicode_ci DEFAULT NULL,
  `host` varchar(160) COLLATE utf8_unicode_ci DEFAULT NULL,
  `ip` varchar(39) COLLATE utf8_unicode_ci DEFAULT NULL,
  `os` varchar(12) COLLATE utf8_unicode_ci DEFAULT NULL,
  `klassifikation` enum('INCIDENT','DUPLICATE','REQUEST') COLLATE
    utf8_unicode_ci DEFAULT NULL,
  `incidentType` varchar(12) COLLATE utf8_unicode_ci DEFAULT NULL,
  `majorExpected` tinyint(1) DEFAULT NULL,
  `hochschulstart` tinyint(1) DEFAULT NULL,
  `matrixZielsys` tinyint(1) DEFAULT NULL,
  `matrixDienste` tinyint(1) DEFAULT NULL,
  `matrixInfos` tinyint(1) DEFAULT NULL,
  `matrixAnzahl` tinyint(1) DEFAULT NULL,
  `matrixQuellsys` tinyint(1) DEFAULT NULL,
  `auswirkung` tinyint(1) DEFAULT NULL,
  `prio` tinyint(1) DEFAULT NULL,
  `reaktionszeit` mediumint(9) DEFAULT NULL,
  `firstaid` text COLLATE utf8_unicode_ci,
  `vorgehensweise` text COLLATE utf8_unicode_ci,
  `analyseerg` text COLLATE utf8_unicode_ci,
  `lsgDetails` text COLLATE utf8_unicode_ci,
  `lsgType` varchar(12) COLLATE utf8_unicode_ci DEFAULT NULL,
  `regelbetrieb` text COLLATE utf8_unicode_ci,
  `monDauer` tinyint(4) DEFAULT NULL,
  `monIntervall` tinyint(4) DEFAULT NULL,
  `monAufgabe` text COLLATE utf8_unicode_ci,
  `monAuffaellig` text COLLATE utf8_unicode_ci,
  `revPositiv` text COLLATE utf8_unicode_ci,
  `revNegativ` text COLLATE utf8_unicode_ci,
  `revOptimierung` text COLLATE utf8_unicode_ci,
  PRIMARY KEY (`id`)
) ENGINE=MyISAM  DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
```

```
108
109 -- Tabellenstruktur für Tabelle `sims_settings`
110 CREATE TABLE IF NOT EXISTS `sims_settings` (
111   `id` varchar(30) COLLATE utf8_unicode_ci NOT NULL,
112   `value` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
113   `comment` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
114   PRIMARY KEY (`id`)
115 ) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
116
117 -- Tabellenstruktur für Tabelle `sims_ssiTpl`
118 CREATE TABLE IF NOT EXISTS `sims_ssiTpl` (
119   `id` varchar(12) COLLATE utf8_unicode_ci NOT NULL,
120   `majorExpected` tinyint(1) DEFAULT NULL,
121   `hochschulstart` tinyint(1) DEFAULT NULL,
122   `matrixZielsys` tinyint(1) DEFAULT NULL,
123   `matrixDienste` tinyint(1) DEFAULT NULL,
124   `matrixInfos` tinyint(1) DEFAULT NULL,
125   `matrixAnzahl` tinyint(1) DEFAULT NULL,
126   `matrixQuellsys` tinyint(1) DEFAULT NULL,
127   `auswirkung` tinyint(1) DEFAULT NULL,
128   `prio` tinyint(1) DEFAULT NULL,
129   `reaktionszeit` mediumint(9) DEFAULT NULL,
130   `firstaid` text COLLATE utf8_unicode_ci,
131   `vorgehensweise` text COLLATE utf8_unicode_ci,
132   `analyseerg` text COLLATE utf8_unicode_ci,
133   `lsgDetails` text COLLATE utf8_unicode_ci,
134   `lsgType` varchar(12) COLLATE utf8_unicode_ci DEFAULT NULL,
135   `regelbetrieb` text COLLATE utf8_unicode_ci,
136   `monDauer` tinyint(4) DEFAULT NULL,
137   `monIntervall` tinyint(4) DEFAULT NULL,
138   `monAufgabe` text COLLATE utf8_unicode_ci,
139   `monAuffaellig` text COLLATE utf8_unicode_ci,
140   `revPositiv` text COLLATE utf8_unicode_ci,
141   `revNegativ` text COLLATE utf8_unicode_ci,
142   `revOptimierung` text COLLATE utf8_unicode_ci,
143   PRIMARY KEY (`id`),
144   UNIQUE KEY `id` (`id`)
145 ) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
146
147 -- Tabellenstruktur für Tabelle `sims_syslog`
148 CREATE TABLE IF NOT EXISTS `sims_syslog` (
149   `id` int(11) NOT NULL AUTO_INCREMENT,
150   `stamp` timestamp NULL DEFAULT CURRENT_TIMESTAMP,
151   `user` int(11) DEFAULT '0',
152   `si` mediumint(9) DEFAULT '0',
153   `action` varchar(40) COLLATE utf8_unicode_ci DEFAULT NULL,
154   `status` enum('OK','ERROR','INFO') COLLATE utf8_unicode_ci DEFAULT NULL,
155   `details` varchar(255) COLLATE utf8_unicode_ci DEFAULT NULL,
156   PRIMARY KEY (`id`)
157 ) ENGINE=MyISAM  DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
158
159 -- Tabellenstruktur für Tabelle `sims_types`
160 CREATE TABLE IF NOT EXISTS `sims_types` (
161   `id` mediumint(9) NOT NULL AUTO_INCREMENT,
162   `category` varchar(40) COLLATE utf8_unicode_ci DEFAULT NULL,
163   `name` varchar(160) COLLATE utf8_unicode_ci DEFAULT NULL,
164   `value` varchar(12) COLLATE utf8_unicode_ci NOT NULL,
```

```
165   `sort` tinyint(4) DEFAULT NULL,
166   PRIMARY KEY (`id`)
167 ) ENGINE=MyISAM  DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci AUTO_INCREMENT
      =93 ;
168
169 -- Tabellenstruktur für Tabelle `sims_users`
170 CREATE TABLE IF NOT EXISTS `sims_users` (
171   `id` int(11) NOT NULL AUTO_INCREMENT,
172   `loginStamp` timestamp NULL DEFAULT NULL,
173   `ip` varchar(40) COLLATE utf8_unicode_ci DEFAULT NULL,
174   `userRole` enum('ROOT','CSIRT','USER') COLLATE utf8_unicode_ci DEFAULT NULL,
175   `kennung` varchar(160) COLLATE utf8_unicode_ci DEFAULT NULL,
176   `name` varchar(160) COLLATE utf8_unicode_ci DEFAULT NULL,
177   `vorname` varchar(160) COLLATE utf8_unicode_ci DEFAULT NULL,
178   `anrede` enum('Frau','Herr') COLLATE utf8_unicode_ci DEFAULT NULL,
179   `email` varchar(160) COLLATE utf8_unicode_ci DEFAULT NULL,
180   `tel` varchar(160) COLLATE utf8_unicode_ci DEFAULT NULL,
181   `mobil` varchar(160) COLLATE utf8_unicode_ci DEFAULT NULL,
182   `comment` varchar(255) COLLATE utf8_unicode_ci DEFAULT NULL,
183   `lastLdapUpdate` date DEFAULT NULL,
184   PRIMARY KEY (`id`),
185   UNIQUE KEY `kennung` (`kennung`)
```

C. Datensätze der types-Tabelle

Listing C.1: Datensätze der types Tabelle

```
INSERT INTO `sims_types` (`id`, `category`, `name`, `value`, `sort`) VALUES
(1, 'kunde', 'LRZ', 'LRZ', 1),
(2, 'kunde', 'LMU', 'LMU', 2),
(3, 'kunde', 'TUM', 'TUM', 3),
(4, 'kunde', 'BSB', 'BSB', 5),
(5, 'kunde', 'Sonstiger', 'SONST', 127),
(6, 'os', 'Windows', 'WIN', 1),
(7, 'os', 'Linux', 'LINUX', 2),
(8, 'os', 'Mac OS', 'MAC', 3),
(9, 'os', 'Mobiles System', 'MOBILE', 4),
(10, 'incidentType', 'Externer SSH Scan', 'SSI-1', 1),
(11, 'incidentType', 'Kompromittierter Webserver', 'SSI-2', 2),
(12, 'incidentType', 'Kompromittierung virt. Server', 'SSI-3', 3),
(13, 'incidentType', 'Viren-infiziertes LRZ-System', 'SSI-4', 4),
(14, 'incidentType', 'Kein Standard SI', 'SONST', 127),
(15, 'lsgType', 'Neuinstallation des Systems', 'NEWINST', 1),
(16, 'lsgType', 'Bereinigung des Systems', 'CLEANING', 2),
(17, 'lsgType', 'Anpassung der Systemkonfiguration', 'CONFIG', 3),
(18, 'lsgType', 'Einspielung eines Backups', 'BACKUP', 4),
(20, 'prio', 'niedrig', '1', 1),
(21, 'prio', 'mittel', '2', 2),
(22, 'prio', 'hoch', '3', 3),
(23, 'prio', 'sehr hoch', '4', 4),
(24, 'matrixZielsys', 'extern', '1', 1),
(25, 'matrixZielsys', 'MWN, Grid-Systeme', '2', 2),
(26, 'matrixZielsys', 'LRZ-intern', '3', 3),
(27, 'matrixQuellsys', 'extern', '1', 1),
(28, 'matrixQuellsys', 'MWN, Grid-Systeme', '2', 2),
(29, 'matrixQuellsys', 'LRZ-intern', '3', 3),
(30, 'matrixDienste', 'keine kritischen Dienste', '1', 1),
(31, 'matrixDienste', 'Dienste für MWN/Grid-Umgebung', '2', 2),
(32, 'matrixDienste', 'Wichtige LRZ-Dienste', '3', 3),
(33, 'matrixInfos', 'keine vertraulichen Infos', '1', 1),
(34, 'matrixInfos', 'vertrauliche Infos bzgl. MWN/Grid', '2', 2),
(35, 'matrixInfos', '(Streng) Vertrauliche Infos', '3', 3),
(36, 'matrixAnzahl', '1', '1', 1),
(37, 'matrixAnzahl', '2-3', '2', 2),
(38, 'matrixAnzahl', '> 3', '3', 3),
(39, 'auswirkung', 'gering', '1', 1),
(40, 'auswirkung', 'mittel', '2', 2),
(41, 'auswirkung', 'groß', '3', 3),
(42, 'auswirkung', 'sehr groß', '4', 4),
(43, 'reaktionszeit', '10 Std.', '600', 1),
(44, 'reaktionszeit', '4 Std.', '240', 2),
(45, 'reaktionszeit', '2 Std.', '120', 3),
(46, 'reaktionszeit', '15 Min.', '15', 4),
(61, 'currentState', 'neu', 'NEW', 1),
(60, 'anrede', 'Herr', 'Herr', 2),
(59, 'anrede', 'Frau', 'Frau', 1),
(47, 'contactrole', 'Vorfallmelder', 'MELDER', 0),
(48, 'contactrole', 'Hotliner', 'HOTLINER', 0),
(49, 'contactrole', 'SI Coordinator', 'SIC', 0),
(50, 'contactrole', 'CSIRT Mitglied', 'CSIRT', 1),
```

```
(51, 'contactrole', 'Administrator', 'ADMIN', 2),
(52, 'contactrole', 'Abteilungsleiter', 'ABTL', 3),
(53, 'contactrole', 'Gruppenleiter', 'GRUL', 4),
(57, 'contactrole', 'SFH-Vertreter', 'SFH', 0),
(55, 'contactrole', 'Major Incident Coordinator', 'MIC', 0),
(56, 'contactrole', 'LRZ-Leitung', 'CEO', 0),
(54, 'contactrole', 'Sonstige', 'SONST', 5),
(62, 'currentState', 'Klassifikation', 'CLASS', 2),
(63, 'currentState', 'Analyse', 'ANAL', 3),
(64, 'currentState', 'Lösung', 'SOL', 4),
(65, 'currentState', 'Monitoring', 'MON', 5),
(66, 'currentState', 'abgeschlossen', 'CLOSED', 6),
(67, 'hochschulstart', 'Nein', '0', 2),
(68, 'hochschulstart', 'Ja', '1', 1),
(69, 'majorExpected', 'Nein', '0', 2),
(70, 'majorExpected', 'Ja', '1', 1),
(71, 'klassifikation', 'Security Incident', 'INCIDENT', 1),
(72, 'klassifikation', 'Duplikat', 'DUPLICATE', 2),
(73, 'klassifikation', 'Information Request', 'REQUEST', 3),
(74, 'closedState', 'Request beantwortet', 'IR_AW', 4),
(75, 'closedState', 'Request weitergeleitet', 'IR_FWD', 5),
(76, 'closedState', 'SI Duplikat', 'SI_DUPLICATE', 3),
(77, 'closedState', 'SI abgebrochen', 'SI_CANCEL', 2),
(78, 'closedState', 'SI erfolgreich', 'SI_SUCCESS', 1),
(79, 'currentState', 'IR Antwort', 'IR_AW', 0),
(80, 'currentState', 'IR Weiterleitung', 'IR_FWD', 0),
(81, 'contactrole', 'Allianz für Cybersicherheit', 'ACS', 0),
(82, 'lsgType', 'Installation eines Patches / Softwareupdates', 'PATCH', 5),
(83, 'lsgType', 'Sperrung einer Kennung', 'LOGINLOCK', 6),
(84, 'lsgType', 'Zurücksetzen einer Kennung', 'LOGINRESET', 7),
(85, 'lsgType', 'Einrichtung/Anpassung einer Firewallregel', 'FIREWALL', 8),
(86, 'lsgType', 'Sonstige', 'SONST', 127),
(87, 'saldo', 'Reaktionszeit', 'saldo1bh', 1),
(88, 'saldo', 'Analysezeit', 'saldo2bh', 2),
(89, 'saldo', 'Lösungszeit', 'saldo3bh', 3),
(90, 'saldo', 'Nachlaufzeit', 'saldo4bh', 4),
(91, 'kunde', 'HM', 'HM', 4),
(92, 'kunde', 'Hochschulstart', 'SFH', 6);
```

D. PDF-Export Musterdokument

Auf den drei folgenden Seiten ist ein beispielhafter Incident Report abgebildet, der durch die Aktion `si-details` des Moduls `export-pdf` dynamisch generiert wurde. Aus pragmatischen Gründen wurden teilweise Platzhaltertexte eingesetzt.

Security Incident Report
LRZ Security Incident Management System
Stand: 2014-06-29 13:58 / Schweiger, Aaron

SI-46: Musterincident für die Erstellung eines PDF-Reports

Incident-Daten

Beschreibung	Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.
Vorfallzeitpunkt	zwischen 09.06. und 11.06.2014
Kunde	Hochschulstart
Betriebssystem	Linux
Hostname	hochschulstart.lrz.de
IP-Adresse	123.124.125.126

Meta-Informationen

Art des Vorfalls	Meldezeitpunkt	Aktueller Status	Abgeschlossen
Security Incident Kompromittierter Webserver	2014-06-05 14:51:04	Monitoring 2014-06-28 20:47	

Reaktionszeit	Analysezeit	Lösungszeit	Nachlaufzeit
11 Std. 9 Min. (3 Tage 1 Std. 4 Min.)	**5 Tage 18 Std. 20 Min.** (18 Tage 18 Std. 26 Min.)	**5 Std. 40 Min.** (1 Tag 10 Std. 28 Min.)	

Klassifikation

Auswirkung	Priorität	Reaktionszeit	MI zu erwarten	Hochschulstart
groß	hoch	2 Std.	Nein	Ja

Standort Zielsystem	LRZ-intern
Standort Quellsystem	extern
Betroffene Dienste	Wichtige LRZ-Dienste
Betroffene Informationen	(Streng) Vertrauliche Infos
Anzahl der betroffenen Systeme	1

Abbildung D.1.: Beispiel für einen PDF-Report (Seite 1)

Security Incident Team

Funktion	Name	Telefon	E-Mail
Hotliner	Metzger, Stefan	+49-89-35831-8846	Stefan.Metzger@lrz.de
Administrator	Pöhn, Daniela	+49-89-35831-8763	Daniela.Poehn@lrz.de
SI Coordinator	Reiser, Helmut	+49-89-35831-7854	Helmut.Reiser@lrz.de
Vorfallmelder	Schweiger, Aaron	+49-8131-7791500	mail@aaron-schweiger.de

Maßnahmen und Analyseergebnisse

Erstmaßnahmen	Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum.
Maßnahmen	Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum.
Analyseergebnisse	Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum.

Wiederherstellung Regelbetrieb und Dokumentation der Lösung

Regelbetrieb	Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum.
Typ der Lösung	Anpassung der Systemkonfiguration
Lösung	Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum.

Abbildung D.2.: Beispiel für einen PDF-Report (Seite 2)

Security Incident Report
LRZ Security Incident Management System
Stand: 2014-06-29 13:58 / Schweiger, Aaron

Monitoring

Dauer	7 Tage
Intervall	8 Stunden
Aufgaben und Verantwortlichkeiten	Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum.
Auffälligkeiten	keine

Post Incident Review

Positive Erfahrungen	
Negative Erfahrungen	
Optimierungs-vorschläge	

Seite 3 / 3

Abbildung D.3.: Beispiel für einen PDF-Report (Seite 3)