INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Bachelorarbeit

Eine SAML-basierte Testumgebung für föderiertes Identitätsmanagement

Ludwig Zacherl

INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Bachelorarbeit

Eine SAML-basierte Testumgebung für föderiertes Identitätsmanagement

Ludwig Zacherl

Aufgabensteller:	Prof. Dr. Helmut Reiser
Betreuer:	Ebner, Ralf Schmidt, Michael Schmitz, David Ziegler, Jule
Abgabetermin:	11. September 2019

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

München, den 11. September 2019

.....
(Unterschrift des Kandidaten)

Abstract

Aus dem wissenschaftlichen und universitären Umfeld sind Unified Login-Systeme nicht mehr wegzudenken. An einer Universität ist es für Studenten inzwischen Normalität, sich nur mit den von der Universität zugeteilten Benutzerangaben an sämtlichen Diensten wie Bibliotheken anzumelden und auf geschützte Ressourcen - wie Online-Aufsätze in Fachzeitschriften - zuzugreifen. Möglich wird dies durch die Teilnahme der Universität als Organisation an Föderationen. Auch das Leibniz Rechenzentrum (LRZ) nimmt an einer Föderation teil, um entsprechende Dienste für seine Benutzer anbieten zu können. Die vom LRZ verwendete Technologien sind SAML und Shibboleth. Mit diesem Standard und der Software ist eine Implementierung einer Föderation möglich. In dieser Arbeit wird eine Testumgebung konstruiert, durch welche es möglich sein wird, sämtliche Szenarien einer Föderation zu testen. Hierunter fallen neue Authentifizierungsverfahren, andere Betriebssysteme, neue Software und andere technologische Neuerungen. Bei der Planung wurden an das Testbed bestimmte Anforderungen gestellt, damit es als Testbed am LRZ brauchbar ist und zum Produktivsystem des LRZ passt. Bei der praktischen Umsetzung wurde ebenfalls darauf geachtet, dass der Aufbau bestimmter Komponenten der Föderation automatisiert aufgesetzt werden können. Es wird zusammenfassend evaluiert, ob das Testsystem die gestellten Anforderungen erfüllt und welche Erweiterungen sich für Zukünftige Arbeiten daraus ergeben.

Inhaltsverzeichnis

1 Einführung

"Welchen Nutzernamen und Passwort hatte ich für diese Plattform gleich nochmal verwendet?" Diese Frage hat sich im digitalen Zeitalter bestimmt jeder bereits einmal gestellt, da man als Person verschiedenste Dienste im Internet nutzt, wobei man für die meisten Dienste ein Profil benötigt, wodurch man authentifiziert werden kann. Da wäre es für den Benutzer ein hoher Komfortgewinn, wenn er sich nur noch an einer zentralen Stelle authentifizieren müsste und sich so mit einem Benutzernamen und Passwort an jedem einzelnen angebotenen Service authentifizieren könnte. Die Versuche, eine solche Infrastruktur umzusetzen, werden Single-Sign-On (SSO) Systeme genannt.

Im universitären Umfeld ist die Funktion des Single-sign-on nicht mehr wegzudenken. Möchte man auf digital vorliegende Artikel und Literatur aus sämtlichen Bibliotheken Europas zugreifen, ist dies zum Beispiel an der Ludwig-Maximilians-Universität München mit einer einzigen Authentifizierung möglich, nämlich der Eingabe des Nutzernamens und des Passworts, das die Universität einem bei der Immatrikulation zugeteilt hat. Möglich ist dies, da die LMU - wie viele andere europäische Universitäten - an einer sogenannten Föderation teilnehmen, die eben genannte Services ermöglichen. Eine nationale Föderation, wie beispielsweise das Deutsche Forschungsetz, welchem vor allem Organisationen im wissenschaftlichen Umfeld beitreten[1], kann an einem Service namens eduGAIN teilnehmen, um angebotene Dienste von weltweiten Föderationen nutzen zu können. Möglich macht dies das Forschungsprojekt GÉANT. In dieser sogenannten Interföderation können Teilnehmer nun auf alle Dienste mit nur einer digitalen Identität zugreifen.

1.1 Fragestellung

Auch das Leibniz Rechenzentrum (LRZ) bietet Dienste wie Identity Provider und Service Provider für Föderationen im universitären bzw. wissenschaftlichen Umfeld an. Ein Identiy Provider (IDP) stellt die Identität und Authentifizierung eines Benutzers sicher, ein Service Provider (SP) stellt geschützte Ressourcen und Services zur Verfügung, auf die der Benutzer zugreifen möchte. Jedoch existiert für das Produktivsystem bisher noch keine flexible und einfach zu initialisierende Testumgebung, in welcher z.B. neue Protokolle, Authentifizierungsverfahren oder Skalierungs-Fragen getestet werden können. Aber auch für Debugging des produktiven Systems ist ein Testbed von Vorteil, da ein Szenario nachgebaut werden kann und Fehler ohne Störung der laufenden Dienste behoben werden können.

Zum Aufbau eines IDP oder SP geben Anbieter von Föderationen umfangreiche Anleitungen, wie diese für die Teilnahme an einer speziellen Föderation zu installieren und konfigurieren ist. Beispiele hierfür sind die DFN-AAI[2] und die SWITCH-AAI[3]. Diese Dokumentationen sind sehr spezifisch auf die jeweiligen Föderationen zugeschnitten. Eine allgemeine

[1] `https://www.dfn.de/verein/mv/mitglieder/`, aufgerufen am 15.07.2019
[2] `https://doku.tid.dfn.de/de:dfnaai:start`, aufgerufen am 25.08.2019
[3] `https://www.switch.ch/aai/guides/`, aufgerufen am 25.08.2019

Dokumentation für die Implementierung einer Föderation existiert nicht. Möchte man ein solches Projekt z.B. mit der Software Shibboleth[4] umsetzen, ist eine enorme Einarbeitungszeit für die äußerst umfangreiche Dokumentation nötig. Wie aufwendig die Umsetung ist, zeigt sich darin, dass sämtliche Anbieter von Föderationen im Falle einer gewünschten Teilnahme an der Föderation sogar ganze Kurse anbieten, die jedoch nur für die Grundlagen reichen. Auch für das LRZ wird eine allgemeinere Variante benötigt, welche erklärt, wie die einzelnen Komponenten aufgesetzt und konfiguriert werden können, damit sie für das Testbed geeignet sind. Das Ziel dieser Arbeit ist es, ein solches Testbed aufzubauen und geeignet zu dokumentieren, da der Aufwand für Recherche und Anpassung der Konfiguration sehr hoch ist.

1.2 Aufbau der Arbeit

Vor den Erläuterung zur Umsetzung des Testbeds und dessen Herausforderungen, wird ein allgemeiner Blick auf das föderierte Identitätsmanagement geworfen.

In Kapitel 2 werden die Grundlagen des föderierten Identitätsmanagement erläutert. Abschnitt 2.1 beschäftigt sich mit einem historischer Rückblick, indem ein kurzer Umriss geboten wird, woraus das föderierte Identitätsmanagement entstanden ist. Hierbei werden Systeme ohne Identitätsmanagement und deren Problematik betrachtet, alsdann Systeme mit Identity- und Access-Management sowie schließlich in Kapitel 2.2 das föderierte Identitätsmanagement. Im Abschnitt 2.3 wird das Rollenmodell des föderierten Identitätsmanagements erarbeitet, d.h. es werden die teilnehmenden Komponenten und die Architektur des Systems betrachtet. Schließlich wird in Abschnitt 2.4 ein kurzer Blick auf die Workflows des födierierten Identitätsmanagements geworfen.

Kapitel 3 beschäftigt sich mit der Umsetzung des theoretischen Blickwinkels aus Kapitel 2 mit Hilfe der Software Shibboleth. Hierfür wird der benötigte Standard OASIS SAML näher erläutert, sowie ein Überblick über die Software Shibboleth gegeben. In den Abschnitten 3.1 und 3.2 werden die benötigten Standards und Software begutachtet. Hierunter fällt der in dieser Arbeit verwendete Standard OASIS SAML, wobei auf andere mögliche Standards wie OAuth[5] nicht eingegangen wird. Als Software für die Umsetzung des IDP und SP wird Shibboleth 3 verwendet, wobei auch hier auf andere Implementierungen wie Liberty Alliance[6] oder OpenID [7] nicht näher eingegangen wird.

In Kapitel 4 wird erarbeitet, welches Design und welche Anforderungen sich an eine prototypische Implementierung eines Testbeds für föderiertes Identitätsmanagement ergeben, sowohl im Hinblick auf die Architektur als auch die Modellierung, bevor sich Kapitel 5 mit der Dokumentation und den Herausforderungen einer praktischen Implementierung beschäftigt.

Schließlich wird in Kapitel 6 auf die Möglichkeiten und Grenzen der Automatisierung der Installation näher eingegangen.

Zusammenfassend wird evaluiert, ob die Anforderungen aus Kapitel 4 umgesetzt werden konnten und welcher Ausblick sich für das Testbed für weitere Arbeiten ergibt.

[4] `https://www.shibboleth.net/`, aufgerufen am 04.07.2019
[5] `https://oauth.net/`, aufgerufen am 23.06.2019
[6] `http://www.projectliberty.org/`, aufgerufen am 23.06.2019
[7] `https://openid.net/`, aufgerufen am 23.06.2019

2 Föderiertes Identitätsmanagement

In diesem Kapitel wird ein theoretischer Blick auf das föderierte Identitätsmanagement geworfen, insbesondere wie es zur Entwicklung dieser Form des Identitätsmanagement gekommen ist, aber auch wie mögliche Architekturen und Workflows aussehen können.

2.1 Entwicklung

Bevor auf die Funktionsweise des föderierten Identitätsmanagements (FIM) eingegangen wird, soll im Folgenden ein kurzer Überblick gegeben werden, wie sich das FIM entwickelt hat.

2.1.1 Systeme ohne Identity und Access Management

Ein System ohne dezitiertem Identitätsmanagement kann als klassischer Fall angesehen werden. Für jeden Dienst, den ein Benutzer nutzen möchte, muss er sich mit seinen Benutzerdaten - meist bestehend aus Benutzername und Passwort - anmelden. Als Anmeldetoken können aber auch Smartcards oder biometrische Merkmale verwendet werden. Dieser Fall begegnet einem im Alltag am häufigsten, sei es, um sich an einem E-Mail Provider anzumelden oder über Online-Banking eine Überweisung zu tätigen. So muss sich der Benutzer für jeden Dienst einen eigenen Account anlegen, da diese Dienste von unterschiedlichen Anbietern nicht durch ein Identitäts-Managementsystem verbunden sind. Als Folge ergeben sich einige Problematiken:

Verwaltungsaufwand für Benutzer: Der Benutzer muss sich für jeden Dienst einen Benutzernamen und Passwort überlegen und merken. Aus sicherheitstechnischer Sicht sollte nie dasselbe Passwort für unterschiedliche Dienste genutzt werden. Falls nämlich ein Angreifer das Passwort eines Benutzers unrechtmäßig erwirbt, hat er sofort Zugang zu sämtlichen Diensten, worunter auch Online Banking oder andere kritische Dienste fallen können. Dass dies in der Realität trotz Sicherheitshinweise nicht eingehalten wird, zeigt die Studie "Psychologie und Passwörter"[1].

Datenredundanz: Gleiche Informationen (z.B. E-Mail Adresse des Benutzers) werden an verschiedenen Stellen abgelegt. Auch hier besteht ein Sicherheitsrisiko. Hat ein Angreifer ein System kompromittiert, an welchem sich der Benutzer registriert hat, so kann er sich höchstwahrscheinlich mit den Daten an anderen Diensten anmelden.

Inkonsistenzen: Aus der eben beschriebenen Redundanz steigt der Wartungsaufwand für die Daten enorm. Wechselt ein Benutzer z.B. seinen Wohnort, so ist der Aufwand der Konsistenzhaltung der Daten hoch, da er die Angaben für jeden einzelnen Dienst ändern muss. Wird die Datenpflege vom Benutzer nicht gewissentlich betrieben, so steigt das Risiko von inkonsistenten Benutzerdaten [Ebe06, S.1f.].

[1] https://www.itsicherheit-online.com/news/studie-psychologie-der-passwoerter-offenbart-unveraendert-risikoreiches-passwortverhalten, aufgerufen am 12.06.2019

2.1.2 Systeme mit Identity und Access Management

Durch eine zentrale Datenbank eines Identity- und Access-Managements können die soeben beschriebenen Probleme umgangen werden. In dieser Datenbank werden alle Benutzerdaten abgelegt. Alternativ gibt es Lösungen wie Kerberos, da hier kein Passwort über ein Netzwerk geschickt werden muss, da Kerberos den Nutzern ein verschlüsseltes "Ticket" zur Verfügung stellt, womit sich der Nutzer an einem Service authentifizieren kann. Umgesetzt werden diese Varianten meist in größeren Einrichtungen wie Firmen oder Universitäten. Ein Benutzer (z.B. Student oder Arbeitnehmer) kann auf alle Dienste der Einrichtung mit seinen Benutzerdaten zugreifen. Dieses Konzept wird als Unified Login bezeichnet. Hierbei muss aber neben der Authentifizierung auch die Autorisierung geklärt werden. Zu den Benutzerdaten muss die Datenbank auch Informationen darüber bereitstellen, was ein Benutzer darf. In einem IT-Unternehmen sollte z.B. ein Angestellter aus der Buchhaltung nicht unbedingt Zugriff auf die Repositories der Entwickler haben, da es sonst zu unbeabsichtigten Datenverfälschungen oder Löschungen kommen kann. Dieser Ansatz funktioniert innerhalb einer Organisation. Was ist jedoch, wenn eine organisationsübergreifende Authentifizierung bzw. Autorisierung notwendig wird? Der Lösungsansatz hierfür ist die Einführung eines föderierten Identitätsmanagement, welches im nächsten Abschnitt ausführlich erläutert wird [Ebe06, S.2].

2.2 Föderiertes Identitätsmanagement

Im ITU-T X.1250 Standard (International Telecommunication Union Telecommunication Standarization Sector) wird eine Föderation definiert als Verbund von Benutzern, Service Providern und Identity Providern [itu10]. Diese von der ITU-T sehr allgemein gehaltene Definition lässt sich konkretisieren. Eine Föderation bzw. Identitäts-Föderation (Identity Federation) ist abstrakt gesehen eine Vereinigung von unterschiedlichen Institutionen oder Organisationen, welche sich auf ein gemeinsames Regelwerk verständigt haben und sich verpflichten, nach diesem zu agieren. Bei einem solchen Regelwerk verständigen sich die Partner auf gemeinsame Richtlinien, rechtliche Rahmenbedingungen sowie technische Daten und Standards [wik17]. Ziel ist es hierbei für Teilnehmer zugangsbeschränkte Ressourcen zugänglich zu machen, wobei hierfür zusätzlich sicherheitstechnische Vorsehungen getroffen werden, da auch zwischen verschiedenen Identitäts-Domänen sicherheitskritische Daten wie z.B. Namen, Adressen oder Passwörter ausgetauscht werden sollen. Auf diese Weise kann in einer Föderation ein Single-Sign-On (SSO) implementiert werden. Beim SSO muss sich ein Benutzer nur noch an einem System mit seinem Benutzernamen und Passwort anmelden und kann somit ohne erneute Anmeldung auf verschiedene Ressourcen unterschiedlicher Services innerhalb der Föderation zugreifen. Ein viel genutztes SSO-System bietet beispielsweise Google mit den Diensten Google Mail, Google Drive, Youtube etc. an [wik17] [FP14].

Die meisten westlichen Länder betreiben Föderationen im universitären bzw. wissenschaftlichen Umfeld. Jedoch sind diese Föderationen meist durch Landesgrenzen beschränkt, wie "National research and education network" (NREN) schon allein für den europäischen Raum zeigt [nre18]. Angebotene Dienste werden meist in einem von wissenschaftlichen Institutionen aufgebauten Netzwerk angeboten. Die ersten Föderationen dieser Art tauchten um das Jahr 2005 auf. Abbildung 2.1 zeigt einen Überblick, welche Länder im Jahr 2017 eine nationale Föderation betreiben (rot), aber auch welche Länder dabei sind, Pilotprojekte zu starten (violett). Ein Überblick über alle aktuellen Föderationen kann auf der Webseite von

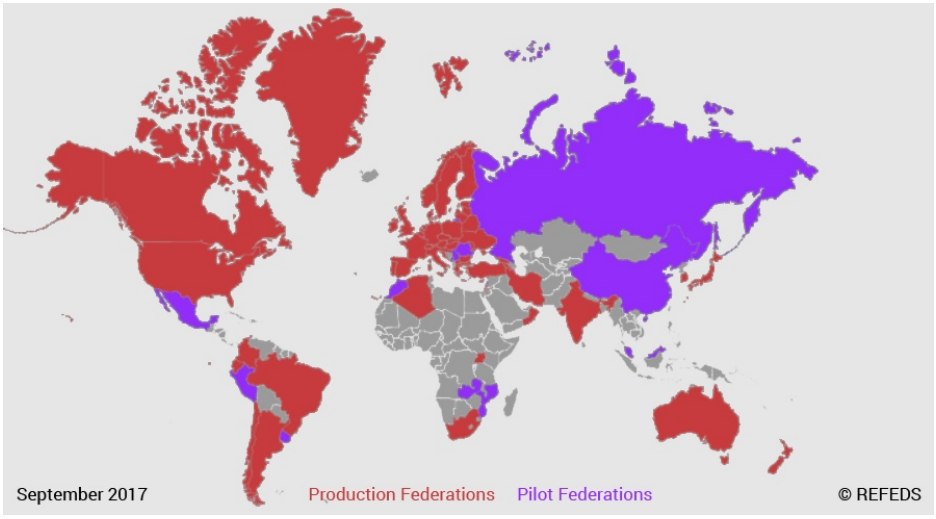

Abbildung 2.1: Überblick über Projekte im föderierten Umfeld weltweit Stand 2017 [wik17]

REFEDS (Research and Education FEDerations)[2] gefunden werden.

Wodurch profitieren teilnehmende Organisationen noch in Föderationen? Besonders die "Empfänger" der identitätsbezogenen Daten können den Aufwand für Speicherung und Pflege der Informationen drastisch reduzieren, wodurch sie ihre Effizienz steigern können. Zudem führt die Zentralisierung der Datenpflege und -überprüfung dazu, dass sich der Aufwand der Überprüfung der personenbezogenen Daten erheblich minimiert, da sich die Teilnehmer auf vertrauenswürdige Identity Provider verlassen können. Aber auch der Benutzer profitiert. So ist es für ihn nicht mehr notwendig, für jeden angebotenen Service persönliche Daten zu hinterlegen, mit dem positiven Nebeneffekt, dass Dateninkonsistenzen vermieden werden. Ändert sich beispielsweise die Telefonnummer eines Benutzers, so muss er diese nicht mehr bei jedem Anbieter einzeln ändern, sondern nur noch an einer zentralen Stelle [Hom07, S.38f.].

Weiterhin erhöht sich die Sicherheit, da durch die Anmeldung an dezidierten Diensten für die Authentifizierung und Autorisierung keine Passwörter an Organisationen herausgegeben werden müssen, da diese nur am IDP gespeichert werden. Mit der Herausgabe von Daten wird also sparsam umgegangen [Mey18, F.10]. Der soeben beschriebene Punkt der Datensparsamkeit erleichtert zudem die Anwendung und Umsetzung der Datenschutz-Grundverordnung (DSGVO), da der Benutzer den zu übertragenen persönlichen Daten zustimmen muss.

Der größte Vorteil einer Föderation ergibt sich jedoch dadurch, dass Nutzer teilnehmender Organisationen auf die Ressourcen aller anderen teilnehmenden Organisationen der Föderation zugreifen können, ohne sich in diesen Föderationen neu mit einem Account registrieren zu müssen. Die Organisationen einer Föderation müssen sich nicht kennen und auch keine extra Verträge zur Nutzung aushandeln. Durch das bereits bestehende Vertrauensverhältnis innerhalb der Föderation reicht es aus, wenn ein Nutzer sich an seiner Heim-Organisation anmeldet. Mit dieser Anmeldung hat er jetzt Zugriff auf sämtliche in der Föderation angebotenen Dienste.

In manchen Föderationen werden die Teilnehmer zwischen Föderationsmitgliedern und

[2] `https://refeds.org/federations`, aufgerufen am 23.05.2019

Föderationspartnern unterschieden. Föderationsmitglieder betreiben in der Regel eigenständige Identity Provider und Service Provider, wie z.B. Universitäten und Forschungseinrichtungen. Sogenannte Föderationspartner hingegen können auch externe Firmen sein, die kommerzielle Inhalte z.B. für Universitäten bereitstellen möchten [wik17].

2.3 Rollenmodell und Architektur des föderierten Identitätsmanagements

Im FIM finden sich Komponenten wie Identity Provider, Service Provider, Attribute Authorities, Discovery Services sowie Schnittstellen zu Föderationsmetadaten, welche im Folgenden näher erläutert werden. Obwohl nicht alle Komponenten im Testbed implementiert werden, sollen sie der Vollständigkeit halber hier kurz dargestellt werden. Hierbei muss allerdings erwähnt werden, dass dieses Rollenmodell für die meisten FIM-Ansätze gültig ist. Es gibt jedoch auch spezielle Lösungen wie OpenID, die ein anderes Architekturmodell umsetzen.

2.3.1 Identity Provider

Ein Identity Provider (IDP) ist eine vertrauenswürdige Datenquelle für alle nötigen Informationen über einen Benutzer. Zudem sorgt der IDP für die Authentifizierung und Autorisierung des Nutzers, der auf bestimmte Dienste in einer Föderation zugreifen möchte. Zusätzlich zu diesen sicherheitskritischen Informationen stellen IDPs auch allgemeine Benutzerinformationen wie z.B. Namen oder Adressen zur Verfügung. Insgesamt sind diese Daten stark personenbezogen. Alle übertragenen Daten sind in der Regel mit sogenannten Metadaten versehen, welche Datenmissbrauch vorbeugen sollen. In Metadaten finden sich beispielsweise kryptographische Signaturen und und Prüfsummen. Hierfür sind Zertifikate bzw. eine Public Key Infrastruktur nötig.

Im föderierten Identitätsmanagement ist jede Identität genau einem IDP zugeordnet. Jedoch kann ein Benutzer verschiedene digitale Identitäten haben, wie beispielsweise eine private und berufliche Identität, wodurch er die Möglichkeit hat, verschiedene IDPs zu verwenden. Da es jeder Identität frei steht, seinen eigenen IDP zu betreiben, kann dies im Extremfall dazu führen, dass jeder Benutzer seinen eigenen IDP betreibt. Hierfür muss er nur die nötige Software auf einer Maschine betreiben (siehe Kapitel 3). Bei dieser Variante ist es natürlich fraglich, ob ein IDP noch als zuverlässige Datenquelle zu werten ist.

Insgesamt gibt es im Föderierten Identitätsmanagement zwei Ansätze bezüglich IDPs: einen zentralisierten und dezentralisierten Ansatz. Bei Ersterem gibt es nur einen IDP für alle Benutzer, bei Letzterem stehen mehrere IDPs zur Verfügung. Wie in Abschnitt 2.3.6 gezeigt wird, hat sich in der Praxis der dezentralisierte Ansatz durchgesetzt [Hom07, S.39-41].

2.3.2 Service Provider

Service Provider (SP) bieten in einer Föderation Dienste wie z.B. Zugänge zu nicht kostenfreien Online-Bibliotheken im wissenschaftlichen Umfeld an, für welche ein Benutzer sich ausweisen muss. Im Föderierten Identitätsmanagement ist der IDP für die Informationserbringung zuständig. Der SP vertraut den vom IDP übermittelten Daten, wenn beide innerhalb einer Föderation agieren. Aufgrund dessen werden SPs auch als “identity data

consumers"[3] bezeichnet. Sind nicht alle benötigten Daten vom IDP übermittelt worden, so können optional Daten vom Nutzer nachgetragen werden. Eine weitere Möglichkeit bieten sogenannte Attribute Authorities, welche im nachfolgenden Abschnitt erläutert werden.

2.3.3 Attribute Authority

Attribute Authorities (AA) können an IDPs allgemeine Attributsauskünfte zu einem Benutzer übermitteln. Sie werden z.B. benötigt, wenn Informationen zu einem Benutzer über mehrere Organisationen hinweg verteilt sind. Dieses Szenario ist im wissenschaftlichen Umfeld durchaus nicht ungewöhnlich, da Wissenschaftler oftmals an mehreren Forschungseinrichtungen weltweit tätig sein können. Wodurch unterscheiden sich AAs nun aber von IDPs? AAs sind nicht für die Authentifizierung von Nutzern zuständig, vielmehr dienen sie unterstützend für IDPs, indem sie entsprechende Attributsdaten bereitstellen. Theoretisch können AAs auch Autorisierungsinformationen bereitstellen, jedoch steigt die Komplexität der Entscheidung von Autorisierung eines Benutzers neben IDPs und SPs enorm, weshalb sich diese Funktion in der Praxis nicht durchgesetzt hat [Hom07, S.41f].

2.3.4 Discovery Service

In größeren Föderationen, in welchen es mehrere IDPs gibt, tritt folgendes nicht-triviales Problem auf: wie soll ein SP den zugehörigen IDP eines Benutzers effizient finden und korrekt zuordnen? Eine Lösung des Problems bietet die Software Shibboleth (siehe Kapitel 3). Hier wird ein Service implementiert, der als "Where Are You From" (WAYF) bezeichnet wird. Shibboleth implementiert also mit WAYF einen Discovery Service (DS) für eine Föderation, wie diese Lösung im Allgemeinen genannt wird. Im SSO wird ein DS verwendet, wenn ein SP einen Benutzer zu seinem IDP zuordnen muss. Wird ein Benutzer, der einen Dienst eines SP nutzen möchte, an den DS weitergeleitet, kann er dort seinen zugehörigen IDP auswählen und der SP erkennt hierdurch, zu welchem IDP der Nutzer gehört. Die Vorteile werden schnell klar: die SPs teilen sich einen zentralen Punkt, an dem die Weiterleitung zu den IDPs stattfindet. Außerdem würde ohne einen zentralen DS dem SP eine zusätzliche Aufgabe zugewiesen werden, da er quasi seinen eigenen DS implementieren müsste [Wid08, S.7f].

2.3.5 Schnittstelle zu Föderationsmetadaten

Damit alle Teilnehmer einer Föderation stets aktuelle Informationen, wie die Anzahl der Teilnehmer oder verwendete Protokolle von allen teilnehmenden Komponenten und Organisationen in der Föderation besitzen, gibt es das Konzept der Föderationsmetadaten. Mit ihrer Hilfe werden folgende Informationen erfasst und validiert:

- Kommunikation: Erfassung sämtlicher verwendeter IP-Adressen bzw. DNS-Namen und Protokolle aller erreichbaren Service Endpoints in Form von URIs.
- Verteilung von Serverzertifikaten: Zur Authentifizierung, Verschlüsselung und Signatur verwenden die Komponenten Zertifikate. In den zentralen Metadaten werden diese gespeichert und so allen Komponenten zugänglich gemacht.

[3] Hommel, Wolfgang: Architektur- und Werkzeugkonzepte für föderiertes Identitäts-Management, Dissertation, Ludwig-Maximilians-Universität München, 2007, S.41

- Ansprechpartner: Zusätzliche Eintragung von technischen und organisatorischen Ansprechpartnern der jeweiligen Komponenten.
- Erforderliche Benutzerdaten: Für SPs kann festgehalten werden, welche Benutzerattribute der SP von den IDPs verpflichtend oder optional erhält.
- Zusicherung von Eigenschaften der IDPs bzw. SPs: Die Einhaltung von Datenschutz- oder Sicherheitsrichtlinien sowie die Mitgliedschaft in Subföderationen kann festgehalten werden.

Die Föderationsmetadaten werden an einer zentralen Stelle bereitgestellt, z.B. über einen Webserver, der für alle Teilnehmer der Föderation erreichbar ist. In regelmäßigen Abständen müssen sich die teilnehmenden Komponenten eine aktuelle Version der Metadaten holen. Für die Pflege und Aktualisierung dieser Daten ist die Föderation selbst verantwortlich. Bei jedem neuen Teilnehmer müssen die Föderationsmetadaten für diesen ergänzt werden, sei es manuell oder automatisiert. Bei den Föderationsmetadaten handelt es sich um eine XML-Datei. Die Komponenten können nur lesend auf sie zugreifen. Wie eine solche Datei aussehen kann, ist in Anhang 1 in der Dokumentation zur Installation des IDP gegeben.

In der Theorie sollten die Metadaten in einem Push-Verfahren ausgeliefert werden, damit jede Komponente zu jeder Zeit die selben gültigen Daten erhält. In der Praxis jedoch werden die Daten in einem Pull-Verfahren von den Komponenten geholt, in der Regel einmal pro Stunde. Der Vorteil dieser Variante besteht in der reduzierten Fehlerbehandlung, da man sich nicht um nicht-verfügbare bzw. nicht-existierende Föderationsteilnehmer kümmern muss, wie es im Push-Verfahren der Fall wäre. Die Datei der Föderationsmetadaten ist mit Hilfe eines Zertifikats elektronisch signiert. Als Protokoll im gebräuchlichen Pull-Verfahren wird HTTPS verwendet [Hom07, S.234-236]. Wie diese Metadaten im Detail aufgebaut sind, wird in Kapitel 5.3 erläutert.

2.3.6 Architekturen

Insgesamt kann man Föderations-Architekturen in drei Kategorien einteilen: Full-Mesh-Föderationen, Hub and Spoke mit verteiltem Login und Hub and Spoke mit zentralem Login. Die am weitesten verbreitete Architektur ist die Full-Mesh-Föderation, welche von ca. 80 Prozent der NREN-Föderationen (National Research and Educational Network) umgesetzt wird. NRENs sind ein Netzwerke, die v.a. Universitäten und wissenschaftlichen Institutionen ein qualitativ hochwertiges Netz anbieten, das sie miteinander zur Nutzung gegenseitiger Services verbindet [nre19].

Die Full-Mesh-Architektur ist in Abbildung 2.2 im Überblick dargestellt. Zur weiten Verbreitung trägt die Tatsache bei, dass in der Full-Mesh-Föderation jede Komponente verteilt vorhanden ist. Es ist also nicht nötig eine zentrale Stelle einzurichten, welche besonders vor Ausfall und Kompromittierung geschützt werden muss. Jede Föderation betreibt eigene IDPs mit eigenem Identity Management sowie einer beliebigen Anzahl an SPs. Die meisten Föderationen betreiben zusätzlich einen zentralen DS/WAYF, wobei in dieser Architektur SPs auch lokal einen DS anbieten können. Eine Herausforderung ist hierbei die Verteilung von konsistenen Föderationsmetadaten, welche in dieser Art von Föderation sehr groß werden können.

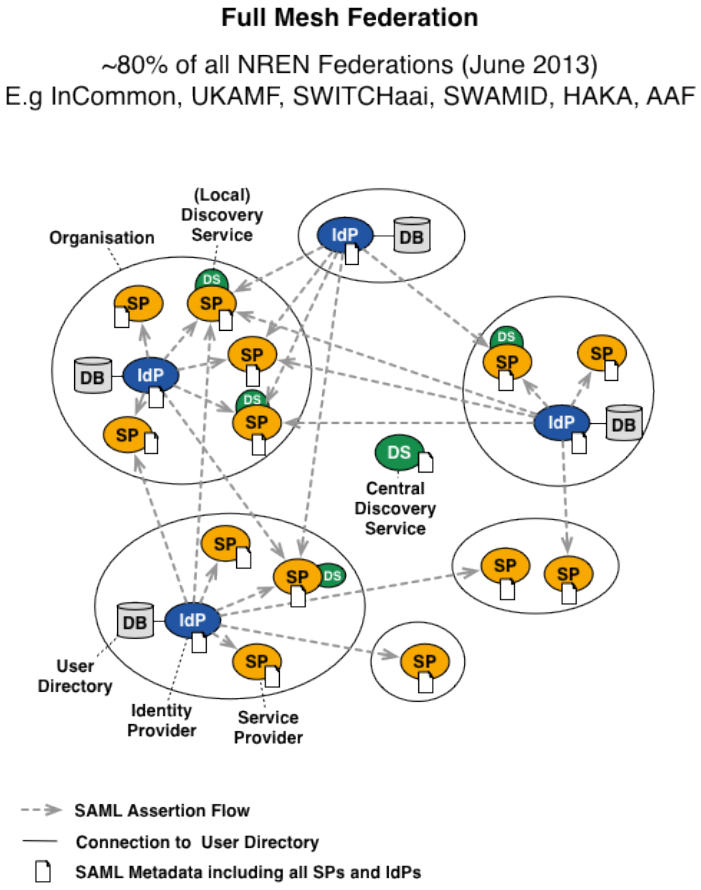

Abbildung 2.2: Full-Mesh Architektur [wik17]

Weitaus weniger verbreitet ist die Hub-and-Spoke Architektur mit verteiltem Login (Abbildung 2.3). Diese implementieren nur ca. 15 Prozent der NREN-Föderationen. Diese Architektur benötigt einen zentralen Hub oder Proxy, über welchen SAML-Assertions in der Föderation ausgetauscht werden können. Der Hub dient den SPs der Föderation als IDP, genauso dient er aber auch den IDPs als SP der Föderation. Metadaten werden also sowohl von den IDPs als auch SPs nur noch vom Hub benötigt. Zusätzlich implementiert der Hub einen zentralen DS, einzelne SPs bieten keinen DS an. Problematisch ist hier, dass der Hub ein Single-Point-of-Failure ist. Fällt er aus, funktioniert die gesamte Föderation nicht mehr. Daher muss der Hub besonders vor Ausfall und Kompromittierung geschützt werden. Von Vorteil hingegen ist, dass nur der Hub die Teilnehmer der Föderation kennt. Zudem kann er kontrollieren, welche Attribute vom IDP zum SP weitergegeben werden. Aus praktischer Sicht wird in der Hub-and-Spoke Architektur mit verteiltem Login der Eintritt in eine Interföderation erschwert. EduGAIN benutzt daher die Full-Mesh-Architektur.

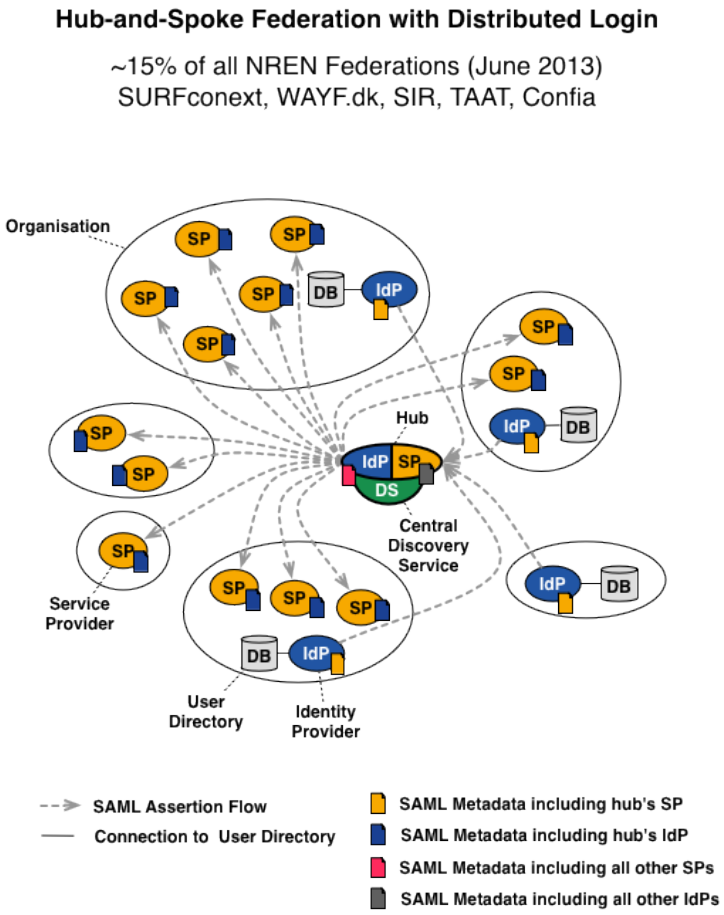

Abbildung 2.3: Hub-and-Spoke Architektur mit verteiltem Login [wik17]

Die letzte Art von Architektur wird nur von ca. 5 Prozent der NREN-Föderationen verwendet: die Hub-and-Spoke Architektur mit zentralisiertem Login (Abbildung 2.4). In dieser Variante gibt es nur einen zentralen IDP in der Föderation. Auch hier ist der zentrale IDP ein Single-Point-of-Failure, welcher besonders geschützt werden muss. Hinzu kommt aber,

dass die Teilnehmer der Organisation, welche den IDP betreibt, ein besonderes Vertrauen entgegen bringen muss. Bei dieser Form der Architektur können auch schnell Probleme bei der Skalierung auftreten. Der große Vorteil hingegen ist, dass durch den zentralen Login problemlos neue Authentifizierungsprotokolle implementiert werden können [wik17].

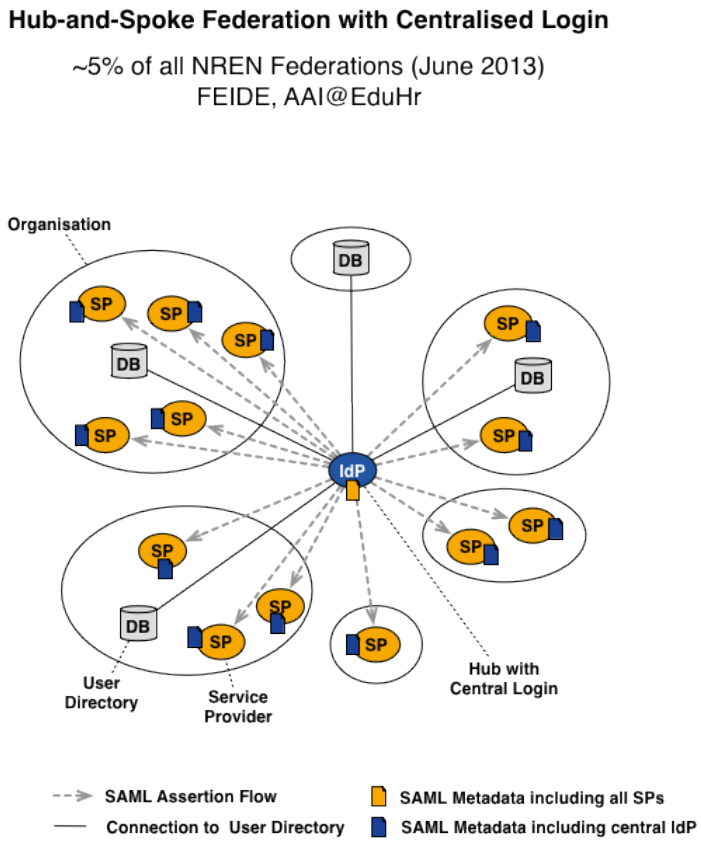

Abbildung 2.4: Hub-and-Spoke Architektur mit zentralem Login [wik17]

2.4 FIM Workflows

In diesem Abschnitt wird aufgezeigt, wie Datenflüsse und Abläufe im FIM generell ablaufen. Auf die technischen Details wird in Kapitel 3 näher eingegangen. Alles beginnt beim Benutzer: soll sich dieser zuerst an einem IdP authentifizieren und dann einen Service eines SPs nutzen? Oder soll der SP den Benutzer bei Bedarf an einen IDP weiterleiten?

Der erste Fall wird als “IDP first Use Case” bezeichnet. Ein Benutzer muss sich in diesem Szenario im allerersten Schritt bei einem IDP authentifizieren, bevor er auf Services von SPs zugreifen kann (siehe Abbildung 2.5). Hat er sich erfolgreich authentifiziert, erhält er Informationen über alle angebotenen Dienste der Föderation. Problematisch ist bei diesem Ansatz, dass die SPs im Vorhinhein bekannt sein müssen und dies am IDP auch vorher konfiguriert werden muss, wodurch die Föderation an Flexibilität einbüßt, da die Skalierbarkeit und Dynamik eingeschränkt werden.

Die gängigere und praxisorientiertere Variante ist der “Service Provider first Use Case”.

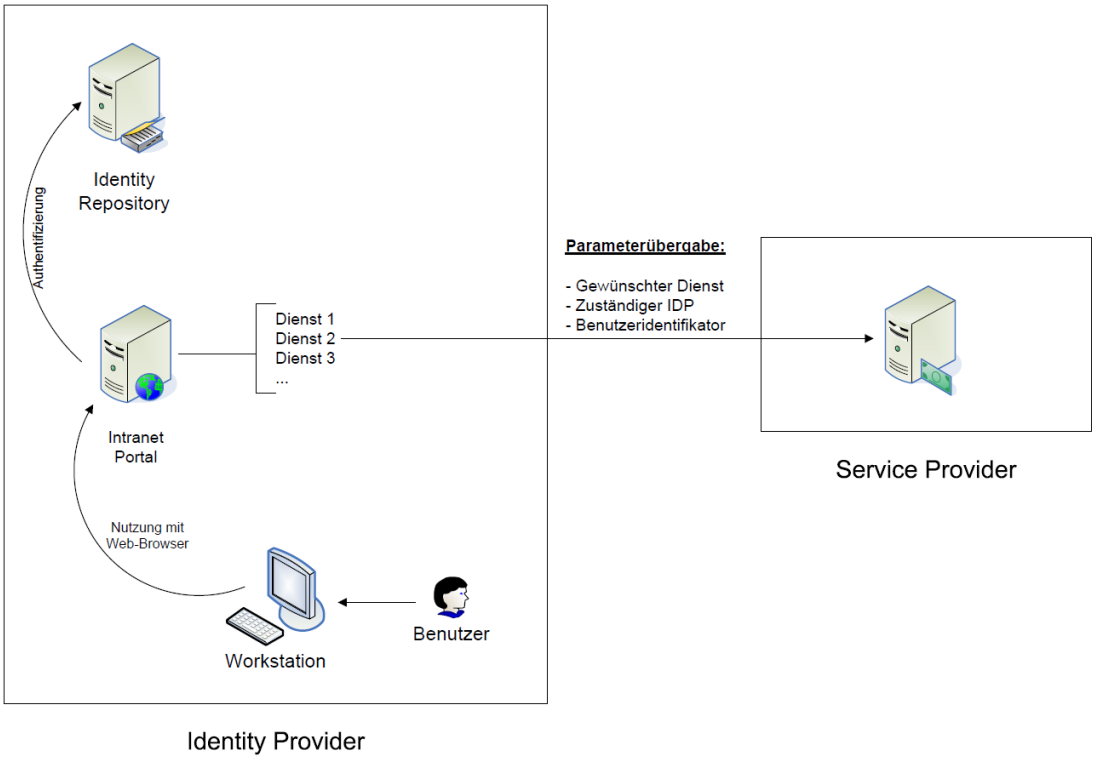

Abbildung 2.5: IDP first Use Case [Hom07, S.51]

Ein Benutzer möchte also an einem SP auf einen Dienst zugreifen, bevor er sich an einem IDP authentifiziert hat. Möchte der Benutzer einen Dienst nutzen, wird er vom SP zum IDP weitergeleitet, dort authentifiziert und schließlich wieder zurück zum SP geleitet. Möglich wird dies durch die Redirect-Funktion des HTTP-Protokolls. Woher weiß der SP aber, zu welchem IDP ein Benutzer gehört? Technisch gesehen ist dieses Problem nicht trivial zu lösen. Es gibt daher verschiedene Varianten, wie die Lösung des Problems aktuell umgesetzt wird [Hom07, S.51-53]:

1. In kleinen Föderationen stellt der SP verschiedene Login-Buttons bereit, die auf die wenigen IDPs verweisen.

2. Für größere Föderationen hat sich der von Shibboleth (vgl. Kapitel 3) implementierte WAYF-Service (Where Are You From) bewährt. Die Abläufe hierbei werden in Kapitel 3.3 erläutert.

3. Eine weitere Möglichkeit ist die Ableitung des zuständigen IDPs aus der IP-Adresse oder des DNS-Namens des Rechners des Benutzers.

Abbildung 2.6 veranschaulicht die Abläufe für die gängige Variante 2: Der Prozess aus Abb. 2.6 lässt sich in sieben Schritten erklären:

1. Der Benutzer möchte auf eine bestimmte Ressource des SP zugreifen.

2. Vom SP wird er zu einem DS via HTTP-Redirect weitergeleitet.

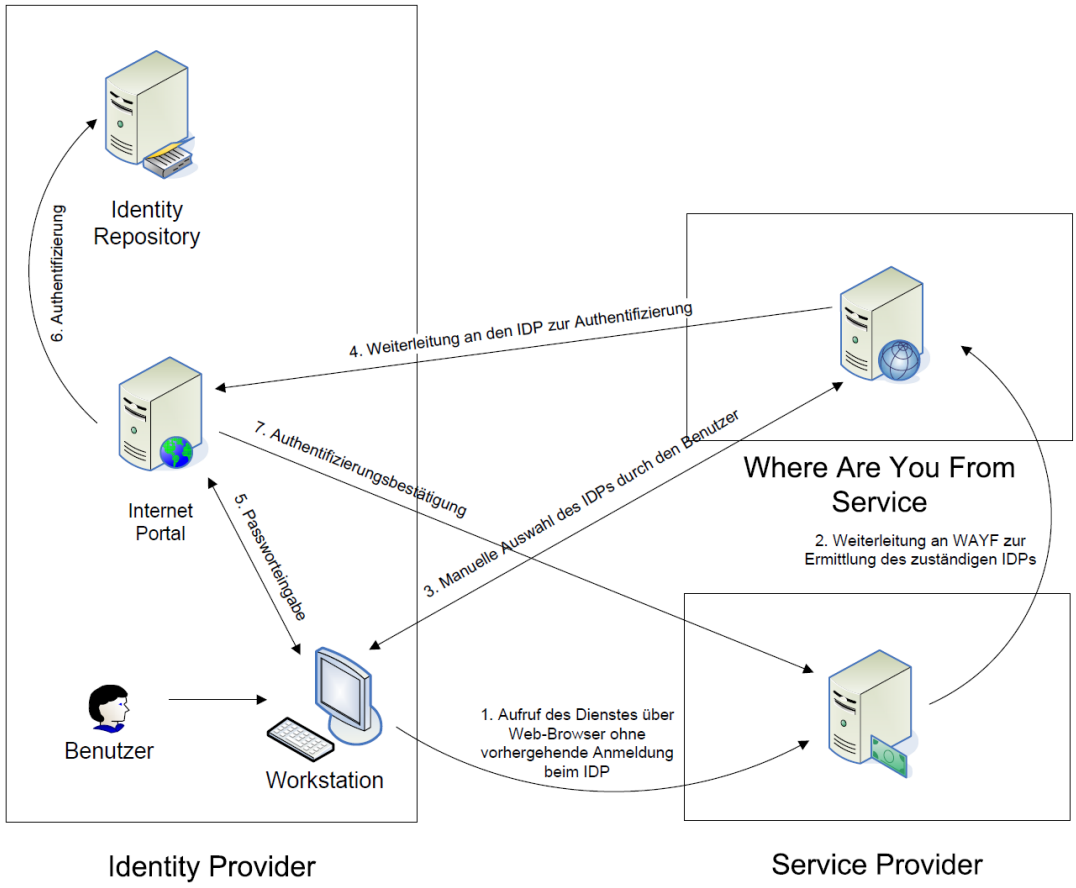

Abbildung 2.6: SP first Use Case [Hom07, S.54]

3. Auf einer Website des DS wird dem Benutzer eine Liste aller verfügbaren IDPs angezeigt. Er muss nun den IDP auswählen, dem er angehört.

4. Der DS leitet den Benutzer erneut mit einem HTTP-Redirect zur Login-Seite des ausgewählten IDPs weiter.

5. Am IDP authentifiziert sich der Benutzer mit seinen Benutzerdaten.

6. Nach erfolgreicher Authentifizierung am IDP wird der Benutzer vom DS zurück zum SP geleitet.

7. Jetzt kann der SP einen Attribute Request (siehe Kapitel 3.1) direkt zum zugehörigen IDP schicken. Sind die Daten des Benutzers gültig, werden die entsprechenden Attribute des Benutzers an den SP geschickt und der Benutzer kann auf die Resource zugreifen.

Der eben beschriebene Ablauf läuft aus der Sicht des Benutzers automatisch ab. Er muss nur bei der Auswahl seines IDPs aktiv eingreifen [LCD18, S.4].

3 Identitätsmanagement mit Shibboleth

Nachden in Kapitel 2 die theoretischen Grundlagen des föderierten Identitätsmanagements ausführlich erläutert wurden, wird in Kapitel 3 der Standard "Security Assertion Markup Language" (SAML) näher betrachtet und auf eine mögliche Implementierung des SAML Standards näher eingegangen: die Software Shibboleth.

3.1 Der Standard OASIS SAML

OASIS (Organization for the Advancement of Structured Information Standards)[1] ist eine Nonprofit-Organisation mit über 5000 Mitgliedern in ca. 600 Organisationen. Das Konsortium beschäftigt sich mit der Entwicklung und Umsetzung von Standards für die weltweite Informationsgesellschaft, wie beispielsweise für das Internet of Things, Cloud-Computing und Notfallmanagement, um nur einige Felder zu nennen [oas19].

Die "Security Assertion Markup Language" wird vom OASIS Security Services Technical Committee (OASIS TC) spezifiziert. In dem Komitee sitzen Vertreter bekannter Hersteller von Identity Management Produkten [Hom07, S.117]. Als erste Version SAML V1.0 wurde der Standard 2001 entwickelt und 2002 als gültig deklariert. Version 1.1 erschien bereits 2003, vorwiegend als Verbesserung und Fehlerkorrektur gegeüber V1.0. Mit der neuen Version wurde SAML immer erfolgreicher, vor allem im Finanzwesen, höheren Bildungswesen und Regierungsangelegenheiten. Schließlich erschien 2005 der immer noch gültige Standard SAML V2.0 [Mad05, S.2].

Durch SAML lassen sich Authentifizierungs- und Autorisierungsbestätigungen sowie allgemeine Attributsauskünfte übermitteln, auch "authentication assertions", "authorization assertions" und "attribute assertions" genannt. In Authentifizierungsbestätigungen wird versichert, dass sich ein Benutzer an einem IDP authentifiziert hat. Voraussetzung ist, dass der Empfänger dem IDP vertraut. Mit Autorisierungsbestätigungen werden positive sowie negative Autorisierungen zur Nutzung von Diensten übermittelt. Allgemeine Attributsauskünfte bzw. allgemeine Benutzerinformationen sind alle übrigen Daten, die nicht zur Authentifizierung bzw. Autorisierung benötigt werden. Hier handelt es sich um Daten wie Adressen und Namen.

Die XML-basierte Sprache SAML definiert die Syntax und Semantik dieser Assertions. Einzig bei den "attribute assertions" gibt es keine Einschränkungen bezüglich deren Inhalte, weshalb sich Föderationsteilnehmer hierbei auf ein Datenmodell einigen müssen [Hom07, S.39f.].

Zur Kommunikation dieser Daten werden Request-Response-Protokolle verwendet. Als Einblick, welche Anforderungen die Protokolle umsetzen sollen, seien hier Beispiele genannt [Can05c, S.35]:

- Anforderung von Assertions mit Authentifizierungsbestätigungen und Attributsauskünften

[1] `https://www.oasis-open.org/`, aufgerufen am 09.06.2019

- Anforderung an einen IDP, einen Teilnehmer zu authentifizieren und die entsprechende Assertion zurückzugeben

- Single-Logout zur nahezu gleichzeitigen Beendigung aller aktiven Sitzungen eines Teilnehmers

Erst durch die sogenannten SAML-Bindings wird definiert, wie ein SAML-Request - z.B. Anforderung von Attributsauskünften - auf ein darunterliegendes Kommunikationsprotokoll abgebildet wird (z.B. HTTP oder SOAP). Ein SAML SOAP Binding definiert also wie eine Nachricht im SAML Protokoll über SOAP kommuniziert werden kann. Ein weiteres Beispiel ist das HTTP Redirect Binding, mit welchen SAML Protokoll-Nachrichten über HTTP-Redirection weitergeleitet werden können [Can05a, S.5] [Mad05, S.4].

Aus den eben beschriebenen Komponenten - Assertions, Protokolle und Bindings - wird das SAML Profil gebildet, womit ein spezieller Use-Case beschrieben werden kann. SAML Profile definieren Bedingungen von Assertions, Protokollen und Bindings, um den Use-Case passend und interoperabel zu beschreiben. In Abbildung 3.1 wird der Zusammenhang der grundlegenden SAML Konzepte veranschaulicht [Rag08, S.11].

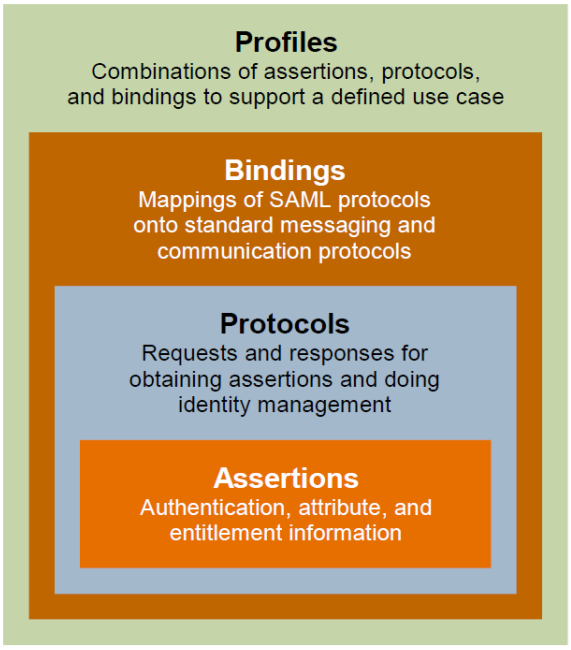

Abbildung 3.1: Überblick über die grundlegenden Zusammenhänge der SAML Konzepte [Rag08, S.11]

Ein Beispiel eines solchen Profils ist das Web Browser SSO Profil, welches im Folgenden näher betrachtet wird. Abbildung 3.2 stellt das Web Browser SSO Profil als Ablaufdiagramm dar.

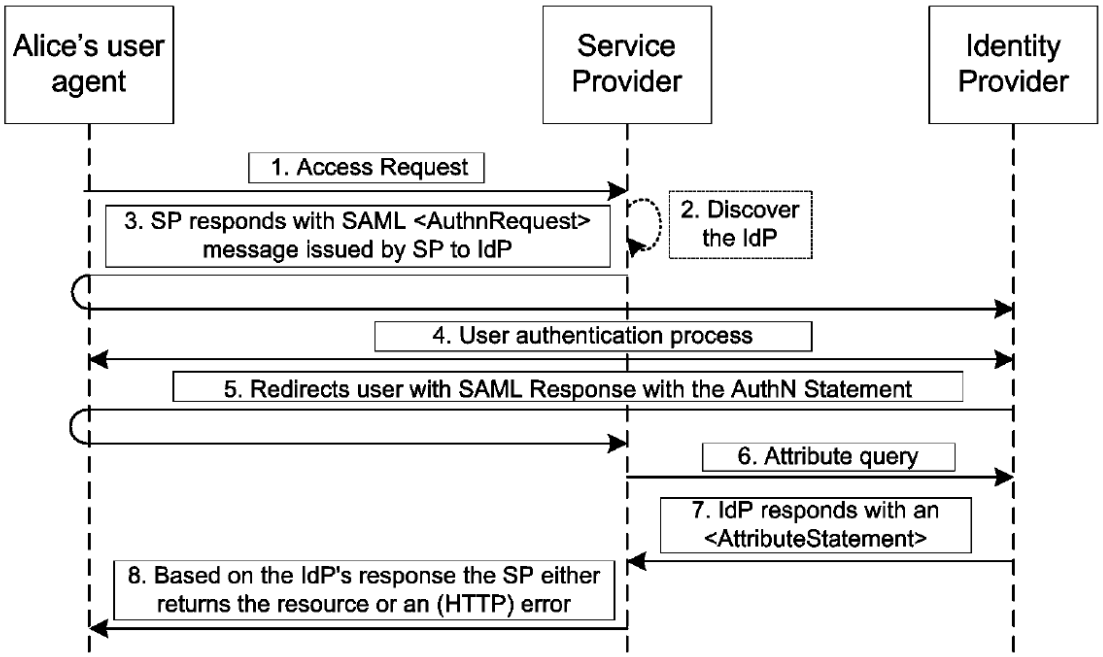

Abbildung 3.2: Ablaufdiagramm zum Web Browser SSO Profil in SAML [TG17, S.22]

Angenommen die Nutzerin Alice möchte über einen User-Agent (z.B. ein Webbrowser) einen Dienst eines SPs nutzen (1) Der SP stellt fest, dass Alice noch nicht authentifiziert wurde und leitet den IDP-Discovery Prozess ein (2). In der Regel geschieht dies mittels Metadaten, die alle Teilnehmer kennen. Kennt der SP den IDP, sendet der eine Authentifizierungsanfrage (SAML AuthnRequest) zum IDP (3) mittels Redirections. In dieser Anfrage sind alle relevanten Informationen wie z.B. der Name des Antragsstellers. Am IDP wird Alice authentifiziert (4) und der IDP antwortet dem SP mit einem AuthnStatement, einer SAML Authentifizierungsbestätigung (5). Der SP benötigt vom IDP zusätzliche Benutzerattribute - wie z.B. die Funktion innerhalb der Organisation oder die Sprache des Nutzers, um eine Autorisierungsentscheidung zu treffen (6). Jetzt überprüft der IDP, welcher SP den Request ausgestellt hat und für welchen Benutzer dieser gilt. Bei erfolgreicher Überprüfung übermittelt der IDP in einem SAML Attribute Statement die angeforderten Attribute an den SP (7). Mit den so erhaltenen Informationen kann der SP entscheiden, ob er Alice Zugang zur geschützten Ressource gewährt oder nicht [TG17, S.22f.].

Ein Beispiel eines solchen Requests eines SP (shib2.srv.lrz.de) an einen IDP (shib1.srv.lrz.de) aus dem Testbed (siehe Kapitel 5) ist in folgendem Listing gegeben. Der Request enthält die wichtigsten Informationen, wie beispielsweise die Information, wo sich der Assertion Consumer Service befindet (Zeile 2) und wer der Aussteller des Requests ist (Zeile 8).

```
1 <samlp:AuthnRequest xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
2                     AssertionConsumerServiceURL="https://shib2.srv.lrz.de/
    Shibboleth.sso/SAML2/POST"
3                     Destination="https://shib1.srv.lrz.de/idp/profile/SAML2/
    Redirect/SSO"
4                     ID="_ea915139c75f0bba6ad0203deef31ed1"
5                     IssueInstant="2019-07-18T09:25:57Z"
```

```
6                         ProtocolBinding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP
    -POST"
7                         Version="2.0">
8   <saml:Issuer xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">https://
        shib2.srv.lrz.de/shibboleth</saml:Issuer>
9   <samlp:NameIDPolicy AllowCreate="1" /></samlp:AuthnRequest>
```

Wie die Antwort des IDP aussieht, wird in nachfolgendem Listing dargestellt. Aus Gründen der Übersichtlichkeit wurden sämtliche Zertifikats- und Signaturwerte, sowie die verschlüsselte Assertion herausgelöscht. Der IDP kontaktiert den Assertion Consumer Service (Zeile 2) und gibt sich selbst als Aussteller der Assertion an (Zeile 8). Die wichtigste Information findet sich in Zeile 13: die Information über die erfolgreiche Authentifizierung. Ab Zeile 15 enthält die Nachricht verschlüsselte Informationen über die Assertion, wie z.B. die Dauer der Gültigkeit der Assertion.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <saml2p:Response Destination="https://shib2.srv.lrz.de/Shibboleth.sso/SAML2/
       POST"
3                   ID="_8b4ea887e51eff79cb8c51f709d1f376"
4                   InResponseTo="_ea915139c75f0bba6ad0203deef31ed1"
5                   IssueInstant="2019-07-18T09:26:13.229Z"
6                   Version="2.0"
7                   xmlns:saml2p="urn:oasis:names:tc:SAML:2.0:protocol">
8    <saml2:Issuer xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion">https://
         shib1.srv.lrz.de/idp/shibboleth</saml2:Issuer>
9    <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
10         <!-- Certificate values -->
11   </ds:Signature>
12   <saml2p:Status>
13     <saml2p:StatusCode Value="urn:oasis:names:tc:SAML:2.0:status:Success" />
14   </saml2p:Status>
15   <saml2:EncryptedAssertion xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion
         ">
16     <!-- EncryptedAssertion containing e.g. validity of assertion etc. -->
17   </saml2:EncryptedAssertion>
18 </saml2p:Response>
```

3.2 Die Software Shibboleth

Shibboleth ist eine Open-Source Implementierung des Konzepts des föderierten Identitätsmanagements. Hinzu kommt, dass Shibboleth ein wichtiger Forschungsansatz im Bereich des SSO und des föderierten Identitätsmanagements ist. Viele Organisationen im wissenschaftlichen Umfeld verwenden die Software, wie zum Beispiel eduGAIN[2] [Hom07, S.128].

Entwickelt wurde die Open Source Software vom Internet2-Konsortium, speziell der Middleware Architecture Committee for Education (MACE). Die erste Version 1.0 wurde 2003 veröffentlicht und schnell von wissenschaftlichen Organisationen wie Universitäten angenommen. Mit SAML V2.0 erschien ein Jahr darauf bereits Shibboleth 2.0. Die aktuelle Version 3.0 wurde 2014 veröffentlicht [Shib]. Shibboleth ist eine webbasierte Middleware, welche Protokolle definiert, die zwischen dem Webbrowser des Benutzers, dessen IDP und der angestrebten Ressource ausgetauscht werden. Im Prinzip funktioniert die Software wie jedes

[2] `https://edugain.org/`, aufgerufen am 04.07.2019

andere SSO System, jedoch beruht Shibboleth stärker auf Standards und kann gegebenenfalls auch Services unterstützen, die außerhalb einer Organisation liegen und dabei trotzdem die Privatsphäre des Benutzers schützen [Shia].

Besonders der von Shibboleth erstmals umgesetzte Discovery Service "Where Are You From?" (WAYF) und das durch Public Key Zertifikate aufgebaute Vertrauensverhältnis der Komponenten zeichnet die Software gegenüber anderen SSO-Anbietern zusätzlich aus. Obwohl der Terminus WAYF laut Shibboleth-Wiki veraltet ist und stattdessen der Begriff IDP Discovery verwendet wird, wird in dieser Arbeit aufgrund der allgemein anhaltenden Verwendung des alten und griffigeren Terminus dieser weiterhin verwendet, obwohl damit der IDP Discovery Service gemeint ist.

3.3 Architektur Shibboleth

In diesem Kapitel werden die von Shibboleth implementierten Komponenten sowie der Workflow des SSO näher betrachtet.

3.3.1 Komponenten

Shibboleth implementiert die in Kapitel 2.3 dargestellten Rollen im föderierten Identitätsmanagement. An dieser Stelle soll auf implementierungsspezifische Details eingegangen werden [Can05b, S.4-7] [Hom07, S.353-355]:

Shibboleth Identity Provider

Der IDP ist eine in Java implementierte Webapplikation, wofür ein Servlet-Container wie Apache Tomcat benötigt wird. Er gibt Assertions an SPs heraus, welche Authentifizierungs- und Attributinformationen enthalten. Die Umsetzung erfolgt nach der SAML Assertions und Protokoll Spezifikation (siehe 3.1). Insgesamt werden folgende Komponenten - nach dem SAML Domain Model - implementiert:

1. Der Single-Sign-On Service (SSO) ist eine HTTP Ressource am IDP, welcher die Authentifizierungsanfragen des SP beantwortet, sobald der Benutzer vom SP bzw. dem WAYF zu seinem IDP umgeleitet wird. Er interagiert mit der Authentication Authority des IDP, um SAML Assertions - gegebenenfalls mit Attributen - in Form einer HTTP Antwort an den Browser des Benutzers herauszugeben. Im letzten Schritt behandelt der SSO den HTTP-Redirect des Benutzers an den SP.

2. Eine Attribute Authority (AA) kann als eine stark reduzierte Form eines IDP angesehen werden. Sie koordiniert den Attribute Resolver (AR) zur Ermittlung von Benutzerattributen aus externen Datenquellen. Die AA erzeugt SAML Assertions. Zudem hat der SP direkten Zugriff auf diesen Service.

Shibboleth Service Provider

Der SP in Shibboleth ist ebenfalls in Java implementiert. Bei ihm handelt es sich um einen ständig laufenden Dämon bzw. Hintergrundprozess, welcher auf Anfragen von Webapplikationen wartet. Aus folgenden Komponenten ist der Shibboleth SP zusammengesetzt:

1. Der Assertion Consumer Service (ACS) leitet den Benutzer an den WAYF-Service bzw. einen IDP weiter.

2. Der Shibboleth Attribute Requester (SHAR) ermittelt vom IDP Benutzerattribute und wertet diese aus.

Shibboleth IDP Discovery Service WAYF

Der WAYF Discovery Service ist - wie der IDP - eine Java-basierte Webapplikation, welche in einem Servlet-Container ausgeführt wird. Er ist für die Auswahl des zuständigen IDPs verantwortlich.

3.3.2 Workflow

Der Authentifizierungs- und Autorisierungsprozess zwischen IDP und SP mit SAML wird im nächsten Abschnitt in zehn Schritten erläutert [AAR16, F.9-11] [Kli19b] [Cha09, S.110-113]:

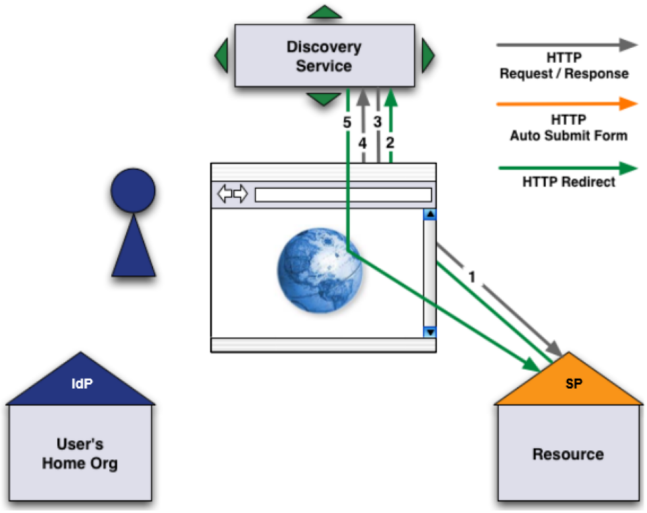

Abbildung 3.3: Schritte 1 - 5 im Authentifizierungs- und Autorisierungsprozess in SAML [AAR16, F.9]

1. Ein Benutzer will auf eine geschützte Ressource zugreifen, wobei der anbietende SP nun den Benutzer authentifizieren und dessen Attribute erhalten muss, damit er auf die Ressource zugreifen kann. Eine zu schützende Ressource kann in der Webserver Konfiguration - wie z.B. bei Apache in der httpd.conf - oder in der Konfigurationsdatei des SP "shibboleth2.xml" definiert werden. Der SP muss wissen, zu welchem IDP der Benutzer gehört. Dies wird durch den IDP Discovery Service (WAYF) implementiert.

2. Über einen HTTP-Redirect wird der Benutzer zum WAYF oder direkt zu einem IDP weitergeleitet. Der WAYF Service kennt alle bzw. eine programmierte Auswahl der IDPs der Shibboleth-Föderation.

3. Der WAYF antwortet mit einer Webpage, auf welcher der Nutzer seinen IDP auswählen kann.

4. Ein IDP wird vom Benutzer augewählt und die Auswahl abgeschickt.

5. Der WAYF sendet an den SP einen Redirect mit dem vom Benutzer ausgewählten IDP.

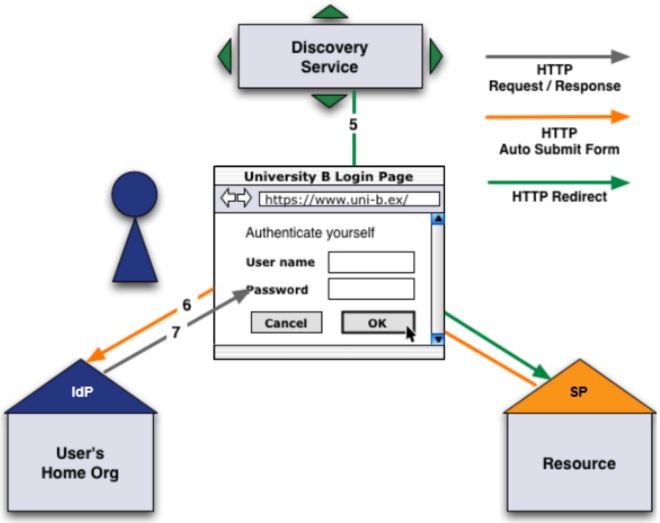

Abbildung 3.4: Schritte 5 - 7 im Authentifizierungs- und Autorisierungsprozess in SAML [AAR16, F.10]

6. Der SP schickt an den IDP einen Authentication Request (AR) zu dessen Authentication Service (Authn Service, AS). Das Format des Requests hängt von den vom SP verwendeten Protokoll und Binding bzw. Profil ab. Wie die Request- und Response-Nachrichten zwischen IDP und SP aussehen, wird den Listings aus Abschnitt 3.1 dargestellt.

7. Der IDP überprüft den AR. Da der Nutzer noch nicht authentifiziert wurde, wird er an die entsprechende Login-Seite des IDP weitergeleitet.

8. Der Nutzer meldet sich mit Benutzername und Passwort am IDP an.

9. Der IDP verifiziert die angegebenen Benutzerdaten am Authn Service und generiert einen zufälligen Identifier, damit Attribute über den Benutzer am SP anonym freigegeben werden können. So erhält der Benutzer sein Token, dass er authentifiziert ist. Mittels eines Attribute Resolvers identifiziert der IDP die Attribute eines Benutzers. Durch einen Attribute Filter wird entschieden, welche Attribute letztendlich freigegeben werden. Der SP erhält nun den Identifier, die vom IDP freigegebenen Attribute sowie die Autorisierung des Benutzers in einer SAML Assertion. Mittels dieser SAML Assertion können diese Informationen zum SP gesendet werden. Die Assertion ist mit dem privaten Schlüssel des IDPs signiert und wird mit dem öffentlichen Schlüssel des SPs verschlüsselt. Der Browser übergibt mit einem POST die Daten dem Assertion Consumer Service (ACS) des betreffenden SP. Der ACS muss die Signatur überprüfen, ob die Daten bzw. die Nachricht gültig ist und entschlüsselt sie.

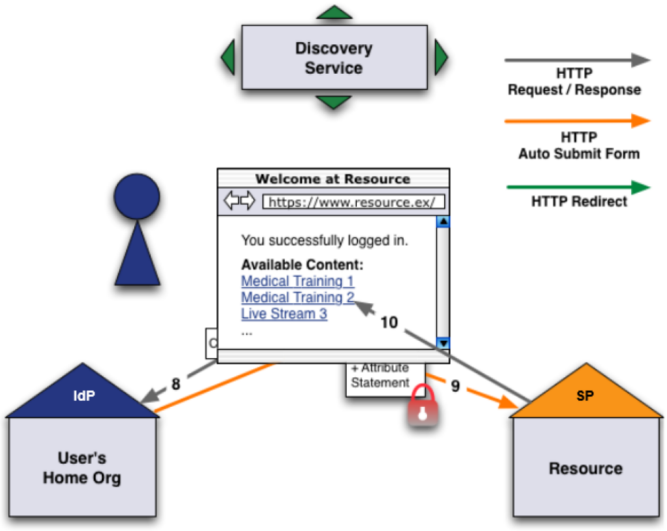

Abbildung 3.5: Schritte 8 - 10 im Authentifizierungs- und Autorisierungsprozess in SAML [AAR16, F.11]

10. Über die SAML Assertion erfährt der SP, ob dem Benutzer der Zugang zur Ressource gewährt oder verweigert wird. Im positiven Fall leitet der SP den Benutzer an die geschützte Ressource weiter, andernfalls wird der Vorgang abgebrochen.

4 Design

Nachdem in Kapitel 3 die theoretischen Aspekte und die Funktionsweise von SAML, Shibboleth und deren Verwendung im föderierten Identitätsmanagement erläutert wurden, wird in diesem Kapitel versucht, mit Hilfe dieser Erkenntnisse ein geeignetes Design für ein Testbed am LRZ zu finden. Zu beantwortende Fragen sind hierbei, welche Komponenten benötigt werden (Abschnitt 3.3.1) und ob der Workflow (Abschnitt 3.3.2) entsprechend umgesetzt werden kann. Es müssen außerdem Anforderungen definiert werden (siehe 4.1), wie ein geeignetes Testsystem für eine Institution konstruiert werden kann, welche bereits produktiv an einer Föderation teilnimmt. Zudem sind Überlegungen für eine Architektur (siehe 4.2) unabdingbar, damit nach Möglichkeit alle definierten Anforderungen erfüllt werden können.

4.1 Anforderungskatalog

Damit das Testbed für das LRZ hinreichende Funktionen zum Test erfüllt, müssen im Hinblick auf das Produktivsystem einige Anforderungen definiert werden.

Konformität mit vorhandener Umgebung Das Testbed muss mit den vom LRZ eingesetzten Systemen des Produktivsystems kompatibel und konform sein. Darunter fällt unter Anderem die Verwendung von im LRZ eingesetzten Betriebssystemen, das verwendete Identity und Access Management sowie Sicherheitsstandards.

Wiederverwendbarkeit Das Testbed soll so installiert und konfiguriert werden, dass es für sämtliche Testszenarien eingesetzt werden kann. Mögliche Testszenarien sind hierbei:

- Erweiterung um sämtliche IDPs und SPs zum Test des Verhaltens in einer Interföderation
- Test neuartiger Protokolle
- Verschiedene Formen des IDP Discovery

Wiederholbarkeit Der Vorgang der Installation und Konfiguration soll ausführlich dokumentiert werden, damit bei Bedarf mit möglichst geringem Aufwand ein weiteres Testbed mit den selben Eigenschaften implementiert werden kann.

Modularität Der Aufbau des Testbeds soll modular gestaltet werden. Es sollte die Möglichkeit bestehen, bei Bedarf beliebig viele Identity Provider und Service Provider einzusetzen, je nach Anfoderungen des zu testenden Szenarios.

Erweiterbarkeit Das Testsystem soll erweiterbar sein. Werden neue Technologien für das Shibboleth Produktivsystem des LRZ in Betracht gezogen, sollen diese ohne größeren Aufwand installiert und getestet werden können. Als Beispiele seien hier einige Punkte genannt:

- neue Authentifizierungsansätze (z.B. 2-Faktor-Authentifizierung)
- SimpleSAMLphp (z.B. Test Shibboleth IDP mit einem SimpleSAMLphp SP)
- Möglichkeiten von IDP Discovery Services
- Test verschiedener Architekturen (siehe Kapitel 2.3.6)
- Anbindung an verschiedene Nutzerverzeichnisse (z.B. Active Directory)

Automatisierbarkeit Der Testaufbau soll mittels eines Frameworks möglichst automatisiert um weitere Komponenten erweitert werden können. Ein geeignetes Framework ist hierfür auszuwählen und auf praktische Einsatzfähigkeit zu prüfen.

Vertraulichkeit und Integrität Die Kommunikation innerhalb des Testbeds soll nach Möglichkeit verschlüsselt stattfinden. Zur Absicherung der Kommunikation werden vertrauenswürdige Zertifikate verwendet, sofern dies praktisch umsetzbar ist. Sind keine von einer vertrauenswürdigen Zertifizierungsstelle unterschriebenen Zertifikate vorhanden, dürfen selbstsignierte Zertifikate verwendet werden.

Ziel ist es, die Authentizität, Integrität und Vertraulichkeit der übermittelten Daten sicherzustellen. Hierfür muss der Kommunikationspartner authentifiziert werden, z.B. durch Server Zertifikate. Die Integrität der Daten soll z.B. über Prüfsummen gesichert werden. Über Verschlüsselung der Daten mit den vertrauenswürdigen Zertifikaten wird die Vertraulichkeit sichergestellt. Mit Vertraulichkeit und Integrität sollen hier nebenbei auch zwei wichtige Ziele der IT-Sicherheit erfüllt werden. Auf das dritte Ziel der Verfügbarkeit kann im Rahmen dieses Testbeds nicht eingegangen werden, da dies die Aufgabe des Betreibers der Systemlandschaft - in diesem Falle das LRZ - ist.

Interföderation Eine Erweiterung des Testbeds zu einer Interföderation soll möglich sein. Im wissenschaftlichen Umfeld ist durch die Vernetzung untereinander die Anforderung zur Teilnahme einer Interföderation kaum verzichtbar.

Architektur Ziel ist es, dass das Testbed nach der Full Mesh Architektur realisiert wird (vgl. Kapitel 2.3.6). Zur Überprüfung der Effizienz von verschiedenen Architekturen soll der Test einer “Hub-and-Spoke with Distributed Login”-Architektur sowie einer “Hub-and-Spoke with Centralised Login”-Architekur möglich sein.

Portabilität Die FIM-Komponenten sollen Plattform- und Betriebssystemunabhängig sein, um die Zusammenarbeit mit verschiedenen Identity und Access Management Systemen testen zu können. Mit den gängigen Betriebssystemen wie Linux und Microsoft sollen die Komponenten kompatibel sein.

Metadaten Metadaten müssen bereitgestellt werden. Dies kann lokal auf den einzelnen Systemen geschehen oder über einen Remote-Server im Netzwerk. Aufgrund der nötigen Konsistenzhaltung der Metadaten ist die Variante der Bereitstellung über einen Remote-Server zu bevorzugen. Von diesem sollen sich die Teilnehmer in vorgegebenen zeitlichen Abständen eine aktualisierte Version der Föderationsmetadaten herunterladen können.

4.2 Architektur des Testbeds

In diesem Abschnitt wird eine mögliche Architektur des Testbeds dargestellt. Hierbei sind stets die Anforderungen des vorherigen Abschnitts zu beachten, da sie für die Architektur unabdingbar sind.

Für das Testbed stehen vom LRZ insgesamt fünf virtuelle Maschinen zur Verfügung, die als Identity bzw. Service Provider aufgesetzt werden können. Als Architekturmodell wird die am weitesten verbreitete "Full Mesh"-Architektur umgesetzt (vgl. Kapitel 2.3.6). Verschiedene virtuelle Maschinen sind hierbei notwendig, da sich die Komponenten in produktiven Systemen in der Regel an verschiedenen Orten befinden und nur durch ein Netzwerk verbunden sind. Als Beispiel könnte eine Bibliothek als SP gelten, welche eine Universität als IDP kontaktieren muss, wenn ein Student auf eine geschützte Ressource, wie einen Onlineartikel, zugreifen möchte.

Von den insgesamt fünf virtuellen Maschinen werden jeweils zwei für IDPs und zwei für SPs eingeplant. Die übrige Maschine kann je nach Bedarf als IDP oder SP hinzugefügt werden.

Die kleinste Form einer Föderation bilden ein IDP und ein SP. Damit sich diese beiden Komponenten vertrauen und damit in einer Föderation agieren können, benötigt der Testaufbau eine Datei namens "Föderationsmetadaten". In dieser Datei sind alle Teilnehmer der Föderation sowie Informationen zu Zertifikaten für die verschlüsselte Kommunikation aufgelistet (vgl. Kapitel 5.3). Zur Erweiterung der Föderation können weitere Komponenten nach dem Muster des IDP 1 und SP 1 aufgesetzt werden (vgl. zur Konfiguration Kapitel 5.2 und 5.4). Als Beispiel dienen IDP 2 und SP 2, wie in der Abbildung 4.1 zu sehen ist. Wie in Abbildung 4.1 zu sehen ist, ist die Minimalanforderung für das einfachste Szenario durch den IDP 1 und den SP 1 gegeben. Die Föderationsmetadaten werden auf dem IDP 1 bereitgestellt. Diese Datei sollte auf einem Webserver liegen, von dem sich alle Komponenten in regelmäßigen Abständen automatisch eine aktuelle Version dieser Datei holen können. Als alternative Variante können die einzelnen Komponenten jeweils eine aktuelle Version der Föderationsmetadaten im lokalen Dateisystem bereitstellen.

Da es keine zwingende Anforderung an das Testbed ist, die Metadaten auf einem Webserver verfügbar zu machen, wurde sich darauf verständigt, dass der IDP 1 die Datei über eine Website zum Download verfügbar macht, da er ohnehin einen Webserver für seine Services betreiben muss (siehe Kapitel 5.1.4). Die einzelnen Komponenten müssen sich manuell eine aktuelle Variante der Datei über diese Webseite herunterladen.

Für die Integrität und Aktualität der Föderationsmetadaten ist der Föderationsbetreiber zuständig. Er muss dafür sorgen, dass stets alle (neuen) Teilnehmer in einer Föderation darin aufgelistet werden. Im Falle des Testbeds sind die Verantwortlichen die Betreiber des Testbeds. Sie sind zur korrekten Funktion der Föderation dazu verpflichtet, bei sämtlichen Skalierungsvorgängen bezüglich der Erweiterung durch Komponenten dies in den Föderationsmetadaten festzuhalten.

Wie in Abbildung 4.1 zu sehen ist, besitzt jeder IDP ein eigenes Identitätsmanagement,

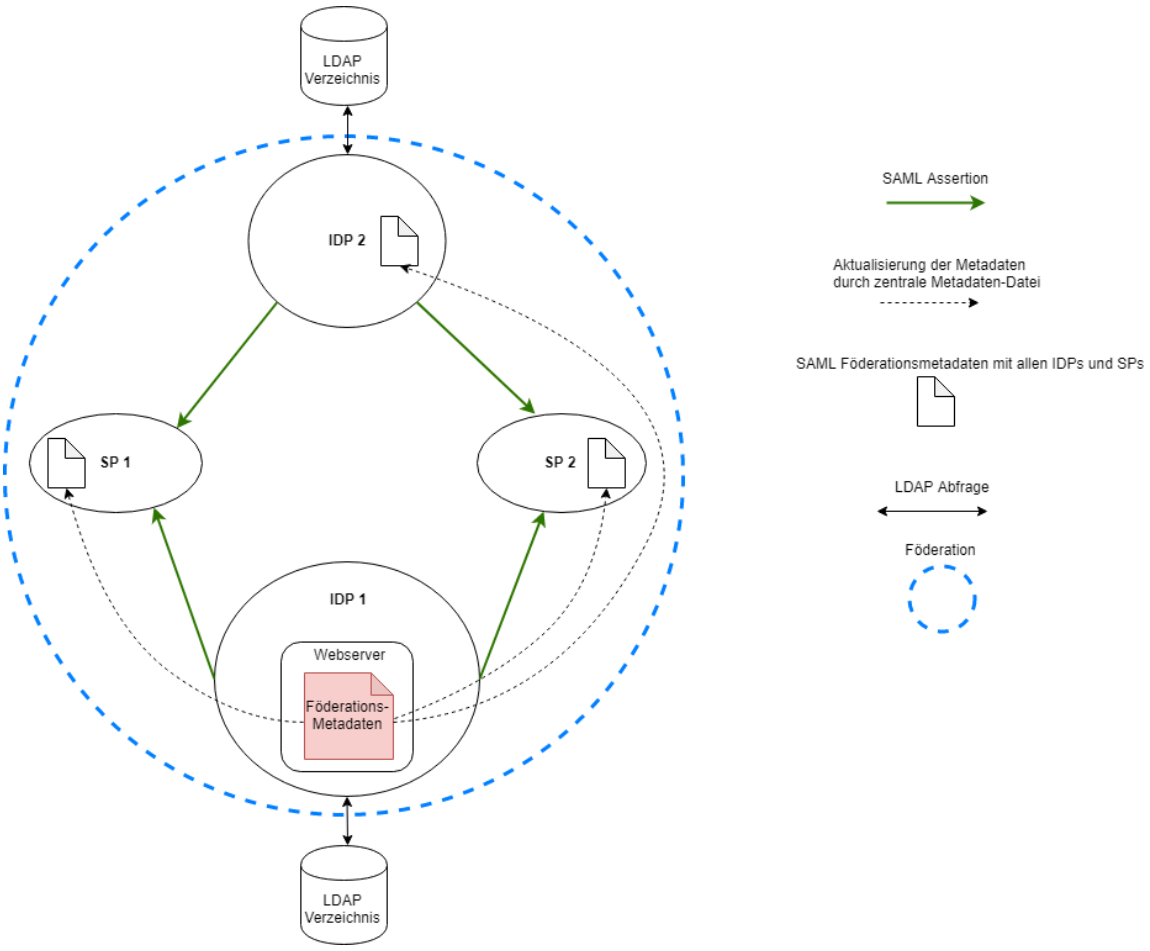

Abbildung 4.1: Architektur des Testbeds

hier ein LDAP-Verzeichnis. Diese Variante - jeder IDP betreibt einen eigenen LDAP-Server - wurde gewählt, da es zu Testzwecken nicht sinnvoll ist, jeden IDP des abgeschlossenen Testsystems an ein echtes Nutzerverzeichnis des Produktivsystems am LRZ anzuschließen, da dieses eine zu große Menge an Daten enthält und nicht beliebig Attribute hinzugefügt und entfernt werden können. Vielmehr ist das Ziel, mit den selbst erstellten Nutzerverzeichnissen beliebig Attribute im LDAP-Verzeichnis generieren zu können und an den IDPs und SPs die Verarbeitung dieser Attribute zu testen. Es wurde die Variante gewählt, dass der LDAP-Server direkt auf einem IDP ausgeführt wird. Wie bereits erwähnt, werden die Nutzerverzeichnisse des produktiven Systems am LRZ auf eigenständigen Systemen ausgeführt. Für das Testbed würde sich demnach die Ausführung des LDAP-Servers auf einer eigenen virtuellen Maschine anbieten, jedoch stehen für die Entwicklung dieses Testbeds lediglich fünf virtuelle Maschinen zur Verfügung. Durch die Verwendung von zwei IDPs ergeben sich zusätzlich zwei LDAP-Server und es stünde nur noch eine Maschine zur Installation eines SP zur Verfügung. Damit nicht unnötig Ressourcen verbraucht werden, werden die LDAP-Nutzerverzeichnisse jeweils auf den zugehörigen IDPs ausgeführt.

Der Fluss der SAML-Assertions von den IDPs zu den SPs, wie in Kapitel 3.3.2 erläutert, wird in Abbildung 4.1 anhand grüner Pfeile dargestellt.

5 Prototypische Implementierung eines Testbeds

Wie im vorangegangenen Kapitel ersichtlich wurde, ist die minimale Voraussetzung für die Simulation einer Test-Föderation das Zusammenspiel von mindestens einem Shibboleth IDP, einem Shibboleth SP und der Existenz von Föderationsmetadaten zur Bildung des Vertrauensverhältnisses. In diesem Kapitel wird für genau diese Komponenten jeweils exemplarisch untersucht, welche Entscheidungen und Ressourcen für die Inbetriebnahme der jeweiligen Komponenten IDP, SP und Föderationsmetadaten erforderlich sind. Ziel ist es hierbei, den Login-Vorgang eines Benutzers beim Zugriff auf eine geschützte Ressource an einem SP in einer Föderation zu simulieren, wie in Unterkapitel 5.1 näher veranschaulicht wird. In den darauffolgenden Unterkapiteln werden die Vorgänge und Entscheidungen, die zur Installation und Konfiguration der einzelnen Komponenten im Testbed getroffen wurden. Hierbei wird sich stets an den theoretischen Betrachtungen aus Kapitel 2 und 3 sowie den Anforderungen aus Kapitel 4 orientiert. Kapitel 5.2 beschäftigt sich mit den Herausforderungen der Installation und Konfiguration eines Shibboleth IDP. Im darauffolgenden Kapitel 5.3 wird erläutert, wie die benötigten Föderationsmetadaten zu konstruieren und erweitern sind. Abschließend wird in Kapitel 5.4 gezeigt, wie ein Shibboleth SP entsprechend der Föderation zu installieren und anzupassen ist.

Soll das Testbed um weitere Komponenten ergänzt werden, kann sich also an den Unterkapiteln 5.2 bis 5.4 orientiert werden, welche Überlegungen hierfür getroffen werden müssen. In Anhang 1 (IDP) und 2 (SP) werden detaillierte Dokumentationen zu Installation und Konfiguration bereitgestellt, womit beliebig viele weitere IDPs und SPs aufgesetzt werden können. Die Kapitelnummerierungen der jeweiligen Unterkapitel korrelieren mit den Kapitelnummern aus Anhang 1 und 2, da so die in diesem Kapitel entstandenen Überlegungen stets mit der praktischen Umsetzung verglichen werden können.

5.1 Login-Vorgang der Testföderation

Nachdem nun definiert wurde, wie eine minimale Föderation aussehen kann, wird in diesem Unterkapitel erläutert, wie ein Login-Vorgang eines Benutzers, der auf eine spezielle geschützte Ressource innerhalb der Föderation zugreifen möchte, aussehen kann. Ziel ist es hierbei in den folgenden Unterkapiteln, den Login-Vorgang funktionsfähig umzusetzen. Zur Veranschaulichung des Login-Vorgangs soll Abbildung 5.1 dienen. Für die Umsetzung des IDP wurde die Maschine `shib1.srv.lrz.de` verwendet (vgl. Kapitel 5.2), für den SP die Maschine `shib2.srv.lrz.de` (vgl. Kapitel 5.4). Als Testbenutzer wird am IDP der Nutzer "shibbolethuser" erzeugt, durch welchen der Login-Vorgang durchgeführt werden soll. Die Föderationsmetadaten liegen jeweils auf IDP und SP, jedoch wurden sie zur Übersichtlichkeit in der Abbildung außerhalb der Maschinen verortet. Zusätzlich zu Veranschaulichung der Tatsache, dass die Föderationsmetadaten in diesem Testszenario die Rolle des IDP Discovery Services übernehmen, werden sie als externe Komponente dargestellt. Für Details sei

jeweils auf die entsprechenden Unterkapitel zu den Komponenten verwiesen.

Wie in Abbildung 5.1 zu sehen ist, gliedert sich der Login-Workflow in 12 Schritte:

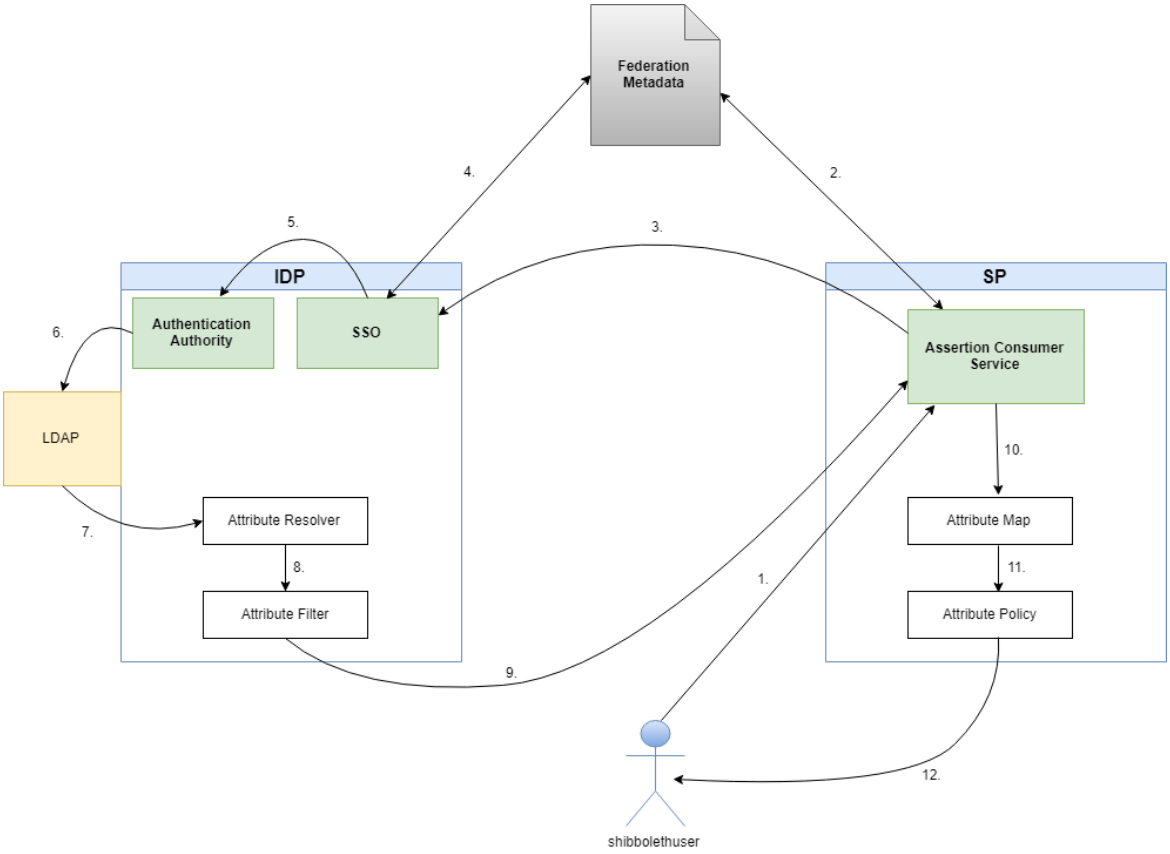

Abbildung 5.1: Darstellung des Login-Vorgangs innerhalb der Testföderation

1. Der User “shibbolethuser” möchte auf eine Ressource zugreifen. Im Testbed wird dies durch den Aufruf der Adresse https://shib2.srv.lrz.de/Shibboleth.sso/Login simuliert. Der SP fängt den User ab und leitet den Request an den Assertion Consumer Service weiter.

2. Der Assertion Consumer Service (ACS) des SP überprüft in den Föderationsmetadaten, ob der angegebene IDP vertrauenswürdig und Teilnehmer der Föderation ist.

3. Wurde ein entsprechender IDP gefunden, schickt der SP einen Authentication Request zum IDP. Der Authentication Request wird an den Browser des Users übergeben und der User wird an einen Endpoint des IDP namens “Single-Sign-On” (SSO) weitergeleitet.

4. Zuerst überprüft der IDP die Entity-ID des SP aus dem Authentication Request über die Föderationsmetadaten. Eine Entity-ID weist einen Teilnehmer einer Föderation eindeutig aus. Meist wird hierfür eine URL mit Angabe des entsprechenden Endpoints verwendet. Hierfür vergleicht er die Entity ID der “AssertionConsumerServiceURL” mit der in den Metadaten angegebenen Entity-ID. Bei erfolgreicher Prüfung wird der

Prozess fortgeführt, andernfalls wird der Login-Vorgang abgebrochen und gilt als gescheitert.

5. Der IDP entscheidet mittels des vom SP erhaltenen Requests, wie der User authentifiziert werden soll. Die Authentication Authority führt jetzt die Authentifizieung durch. Im Testbed wird der User über Username und Passwort authentifiziert, welche er an einer Loginpage des IDP angeben muss.

6. Im Testaufbau soll die Authentifizierung über ein im IDP intern enthaltenes LDAP Verzeichnis stattfinden. Der Authentication Service der Authentication Authority überprüft die vom Benutzer eingegebenen Benutzerdaten im LDAP-Verzeichnis. Vom LDAP Verzeichnis erhält der IDP die Attribute des Nutzers.

7. Die Attribute werden nun an einen Attribute Resolver übergeben. Damit die Daten aus dem LDAP Verzeichnis mit den im IDP definierten Attributen übereinstimmen, werden die Attribute vom Nutzerverzeichnis auf die vom IDP freigegebenen Daten mittels des Attribute Resolvers abgebildet. Mittels des sogenannten Attribute Encondings werden die Attribute in eine Form gebracht, die der SP verstehen kann.

8. Jetzt muss überprüft werden, welche Attribute überhaupt an den SP übergeben werden sollen. Dies geschieht mittels des Attribute Filters.

9. Der IDP übergibt jetzt die Authentifizierungsbestätigung und die freigegebenen Attribute mittels einer SAML Assertion im Browser an den ACS des SP zurück. Die Kommunikation erfolgt verschlüsselt. Der ACS entschlüsselt die Nachricht und führt Sicherheitschecks durch, ob die Integrität der Nachricht besteht.

10. Mittels eines Attribute Extractors werden Attribute, die Authentifizierungsbestätigung und andere nötige Informationen der Nachricht vom ACS extrahiert. Die Attribute werden mit Hilfe der Attribute Map auf interne Variablen des SP abgebildet.

11. Schließlich werden die internen Variablen gefiltert mittels einer Attribute Policy. Durch die Filterung kann festgelegt werden, welche Attribute erlaubt sein sollen.

12. Der SP startet eine Session für den User, falls alle Schritte erfolgreich verlaufen sind. Bei erfolglosem Ausgang wird dem Benutzer mitgeteilt, dass er sich nicht authentifizieren konnte oder keine Autorisierung besitzt, auf die angefragte Ressource zugreifen zu dürfen.

5.2 Shibboleth Identity Provider

Im Folgenden wird exemplarisch beschrieben, wie ein Shibboleth Identity Provider für die Test-Föderation installiert und konfiguriert werden kann. Insgesamt ist dieses Kapitel identisch gegliedert wie die Dokumentation der Installation und Konfiguration aus Anhang 1. Für jedes Unterkapitel kann also das entsprechende Unterkapitel der Dokumentation im Anhang hinzugezogen werden, falls Details der Implementierung benötigt werden.

5.2.1 Voraussetzungen

Bevor mit einer Installation begonnen werden kann, müssen einige Voraussetzungen bestehen:

- Betriebssystem: eine virtuelle Maschine mit einem Linux Betriebssystem, in diesem Fall Debian 9 (Stretch). Generell stehen Shibboleth IDPs und SPs, welche auf Linux-Betriebssystemen ausgeführt werden sollen, die Betriebssysteme Debian und Ubuntu zur Verfügung. Die Auswahl des Betriebssystems Debian ergibt sich aus folgenden Überlegungen: Bei Debian 9 handelt es sich um ein aktuelles stabiles Betriebssystem. In der produktiven Umgebung des LRZ werden die IDPs und SPs ebenfalls auf Linux-basierten Servern ausgeführt (SuSE Linux Enterprise). Die Portierung bzw. Neuinstallation auf Debian soll hierdurch möglich werden.

- Für die Hardware sollten 4 GB RAM und 20 GB Festplattenspeicher eingeplant werden, da die Ausführung des Tomcat Servlet-Containers mindestens 2 GB Hauptspeicher benötigt. Da Shibboleth intern eine Datenbank zur Speicherung von Nutzer-IDs betreiben muss, sollte hier auch ausreichend Festplattenplatz zur Verfügung stehen. Im Testbetrieb wird jedoch nur mit einer begrenzten Anzahl an Benutzern getestet, weshalb hier 20 GB vollkommen ausreichend sind, auch wenn das Testsystem in Zukunft vergrößert wird. In einer produktiven Umgebung mit Anbindung an ein sehr großes Nutzerverzeichnis - wie das Produktivsystem des LRZ - müssen hierfür neue Überlegungen getätigt werden.

- Ein Zugang über SSH mit einem Benutzer mit Root-Rechten muss vorhanden sein, da kein direkter Zugang zu den virtuellen Testservern des LRZ besteht.

- Netzwerk:
  - Es wird ein Hostname (shib1.srv.lrz.de) und eine statische IP-Adresse benötigt (129.187.255.171). Die IP-Adresse wurde vom Netzmanagement des LRZ zugeteilt.
  - Die Firewall sollte für SSL-Verbindungen, über welche externe Komponenten auf den IDP zugreifen, die Ports 443 und 8443 geöffnet haben. Diese Verbindungen sollten direkt möglich sein, ein Proxy ist nicht erlaubt.
  - Falls ein externer NTP-Server verwendet wird, sollte dieser ebenso an der Firewall freigeschaltet werden. Im Netzwerk des LRZ werden die internen NTP-Server verwendet, wie aus der Datei `/etc/ntp.conf` ersichtlich wird:

```
1 # pool.ntp.org maps to about 1000 low-stratum NTP servers.  Your
      server will
2 # pick a different set every time it starts up.  Please consider
      joining the
3 # pool: <http://www.pool.ntp.org/join.html>
4 server ntp1.lrz.de iburst
5 server ntp2.lrz.de iburst
6 server ntp3.lrz.de iburst
```

- Zertifikat: Für den Hostnamen (shib1.srv.lrz.de) sollte ein Zertifikat bereitgestellt werden, dessen Zertifizierungsstelle die gängigen Browser vertrauen. Für den Server wurde

an der LRZ-Zertifizierungsstelle ein Zertifikat beantragt. Zertifikat und Schlüssel werden unter `/etc/ssl/localcerts/` bzw. `/etc/ssl/localkeys/` abgelegt. Das Zertifikat für die Zertifizierungsstelle liegt unter `/etc/ssl/chains/`:

  - Das Zertifizierungsstellen-Zertifikat "DFN-Verein Global Issuing CA" wurde von der übergeordneten vertrauenswürdigen Zertifizierungsstelle "DFN-Verein Certification Authority 2" unterschrieben.
  - Das Serverzertifikat "shib1.srv.lrz.de" ist von der "DFN-Verein Global Issuing CA" ausgestellt und somit vertrauenswürdig.

- LDAP: In dieser Implementierung wird ein eigener LDAP-Server eingerichtet. Wird ein externer Server verwendet, muss dieser an der Firewall freigeschaltet werden. Ein LDAP Verzeichnis bietet sich deshalb an, da auf unixoiden Systemen gearbeitet wird und diese standardmäßig mit LDAP umgehen können. Zudem ist OpenLDAP kostenlos als Open Source unter der OpenLDAP Public License erhältlich. Alternative Verzeichnisdienste wie Microsoft Active Directory können natürlich auch integriert werden, jedoch bietet sich dieser Verzeichnisdienst für die Testumgebung nicht an. Das Microsoft AD ist in der Anschaffung sehr teuer, die Konfiguration ist sehr Microsoft-spezifisch und führt bei der Integration in Linux-Systemen meist zu erhöhtem Aufwand, das Nutzerverzeichnis anzubinden.

5.2.2 Installation und Konfiguration von Basiskomponenten

Bevor mit der Installation des Shibboleth IDP begonnen werden kann, müssen einige Programme installiert werden:

- Apache2: Der Shibboleth IDP soll über einen Webserver erreichbar sein. Hierfür bietet sich die etablierte Lösung des Apache Webservers an, da zur Konfiguration zahlreiche Dokumentationen zu Rate gezogen werden können. Eine leichtgewichtigere Alternative wie nginx kann auch verwendet werden, jedoch empfehlen die meisten Anbieter von Föderationen die Verwendung von Apache. Aus Gründen der Konformität zu anderen Föderationen wurde sich hier auch für die Lösung des Apache Webservers entschieden.

- Apache Tomcat 8: Als Servlet-Container wird Tomcat8 verwendet. Hierbei handelt es sich um eine aktuelle Version der Software. Diese Version wird auch ausdrücklich vom Shibboleth-Wiki empfohlen [shi19], da ältere Versionen vom Projekt nicht mehr unterstützt werden. Tomcat ist ein OpenSource Webserver bzw. Webcontainer, welcher es erlaubt, die in Java geschriebene Anwendung des Shibboleth IDP auf Servlet-Basis auszuführen. Es unterstützt sämtliche Kommunikationsprotokolle wie HTTP oder AJP, welche für die Implementierung des Testbeds zwingend benötigt werden [wik19].

- Java 8: Das Shibboleth-Wiki empfiehlt die Verwendung von "Oracle Java 8 for Linux", jedoch ist die Verwendung des Oracle-Java nicht mehr kostenfrei für eine Produktivumgebung möglich [shi19]. Daher wird auf das "Debian OpenJDK 8" für Debian 9 zurückgegriffen, da es zumindest großteils unterstützt wird.

- Testing und Debugging: Damit bei der Installation und Konfiguration im Falle von Fehlern vernünftig nachvollzogen werden kann, wo die Ursache liegt, werden folgende Programme, die nicht für den Shibboleth IDP erforderlich sind, zusätzlich installiert:

- curl: Zum Abruf von Webseiten innerhalb einer Umgebung ohne grafischer Oberfläche. Curl wird vor allem zum Test der Shibboleth-Statuspage benötigt.
- net-tools: Das Paket net-tools enthält wichtige Programme zum Überprüfen des Netzwerkes, wie z.B. ifconfig, netstat oder route.
- strace: Für das Debugging hat sich "strace" als sehr nützlich erwiesen. Mit diesem Programm kann der Prozessablauf nachvollzogen werden.

5.2.3 Grundlegende Installation des Shibboleth Identity Providers

Die aktuellste Version des Shibboleth IDP kann unter `https://shibboleth.net/downloads/identity-provider/` heruntergeladen werden. Zum Zeitpunkt der Erstellung dieser Arbeit war dies Version 3.4.3.

Bei der Installation des IDP können sämtliche Defaultwerte des Installers übernommen werden, es muss nur der korrekte Hostname `shib1.srv.lrz.de` angegeben werden.

5.2.4 Anpassungen für Tomcat8

Damit Tomcat für den IDP entsprechend funktioniert, sind einige Einstellungen zu ändern. Hierunter fällt z.B. die Angabe einiger Umgebungsvariablen, damit Tomcat seine Ressourcen wie Java findet bzw. genügend Speicherplatz zur Verfügung steht. In `/etc/default/tomcat8` können hierfür die Variablen JAVA_HOME und JAVA_OPTS angegeben bzw. verändert werden. Die Variable JAVA_OPTS kann mit der Option `-Xmx2g` der Java Virtual Machine eine Mindestgröße an Speicherplatz zuweisen, falls dieser standardmäßig unter der Mindestgröße von 2 GB liegt, da die JVM unter Umständen mehr Speicherplatz benötigen kann.

Neben den Umgebungsvariablen muss Tomcat aber noch an weiteren Stellen angepasst werden, um für den IDP korrekt zu arbeiten. Hier seien nur die wichtigsten Punkte genannt, für Details sei auf den entsprechenden Abschnitt in Anhang 1 verwiesen:

1. Tomcat benötig einen Connector auf Port 8009, den sogenannten AJP-Connector (Apache JServ Protocol), womit der Webserver seine Anfragen an den darunterliegenden Tomcat-Applikationsserver weiterleiten kann. Ein Connector ist im Shibboleth IDP ein Endpoint, an dem Anfragen empfangen und Antworten zurückgesendet werden können. Als Default-Connector ist in der Tomcat-Konfiguration der unverschlüsselte HTTP-Connector auf Port 8080 aktiviert. Da die Kommunikation an diesem Port unverschlüsselt erfolgt, muss er deaktiviert werden. Als Beispiel, wie der AJP-Connector in der Datei `/etc/tomcat8/server.xml` aktiviert werden kann, ist folgendes Listing als Codeausschnitt der Datei gegeben:

```
1 <!-- Define an AJP 1.3 Connector on port 8009 -->
2 <Connector port="8009" protocol="AJP/1.3" redirectPort="443" address="
      127.0.0.1" enableLookups="false" tomcatAuthentication="false"/>
```

   Es wird die Protokollversion 1.3 verwendet. Falls ein Request eintrifft, welcher SSL-Verschüsselung benötigt, wird er an Port 443 weitergeleitet. Da der Tomcat-Server nur auf dem Localhost Anfragen vom Webserver an die darunterliegende Schicht weiterleitet, ist die Adresse 127.0.0.1 angegeben. Durch den Wert "enableLookups=false" wird verhindert, dass DNS-Lookups möglich sind. Es soll kein DNS-Name zurückgegeben werden, sondern eine IP-Adresse. Mit dem Wert "tomcatAuthentication=false" wird Tomcat mitgeteilt, dass er keine Authentifizierung durchführen soll.

2. “Session Persistence” soll deaktiviert werden. Gemeint ist hiermit, dass bei Neustart des Tomcat-Servers keine aktive Session überdauern soll. Sie sollen bei Neustarts beendet werden. Um Session Persistence zu verhindern, muss in `/etc/tomcat8/context.xml` der Kommentar in folgender Zeile gelöscht werden. Der Leere Wert der Variable ist hierbei entscheidend:

```
1 <Manager \pathname="" />
```

5.2.5 SSL für Apache mit Tomcat Frontend

Da jegliche Kommunikation zum IDP verschlüsselt erfolgen soll, muss für den Apache-Webserver, welcher sämtliche Anfragen entgegen nimmt, ein SSL-fähiger Virtual Host erstellt werden. Da der IDP sensible Daten über Benutzer hält, sollten zusätzliche Sicherheitsvorkehrungen getroffen werden, wie in den folgenden Punkten näher beschrieben wird.

1. In Apache muss ein SSL-fähiger Virtual Host konfiguriert werden, worüber die Website gehostet werden kann. Dies wird ermöglicht, indem man einen Virtual Host “shib1.srv.lrz.de” unter `/etc/apache2/sites-available/` anlegt. Der Host muss zur sicheren Ausführung folgende Kriterien erfüllen:
   - Der Virtual Host soll nur über den verschlüsselten Port 443 erreichbar sein.
   - Der Servername muss shib1.srv.lrz.de lauten.
   - Es sollen explizite Logfiles für den Virtual Host am Ort `/var/log/apache2/shib1.srv.lrz.de.error.log` und `/var/log/apache2/shib1.srv.lrz.de.access.log` erstellt werden.
   - Die DocumentRoot bleibt als Defaultwert bei `/var/www/html`.
   - Als SSL-Protokolle dürfen SSLv2, SSLv3 und TLSv1 nicht verwendet werden, da sie inzwischen als nicht mehr sicher gelten.
   - Die Verschlüsselungsmethode des Servers soll gegenüber der des Clients bevorzugt werden
   - SSL Compression muss vorsichtshalber deaktiviert werden, da am IDP mit sensiblen Nutzerdaten umgegangen wird. Verwendet ein Benutzer einen älteren verwundbaren Browser, kann hier nämlich eine Attacke namens “BEAST” [Fis12] gestartet werden.
   - HTTP Strict Transport Security (HSTS) muss aktiviert werden. Hierdurch wird versichert, dass die Website nur über HTTPS angesprochen wird, da der Server in die Antwort einen HSTS-Header mit einfügt. Der Browser merkt sich nun, von welchem Server er einen HSTS-Header bekommen hat und weigert sich, diese mit dem HTTP-Protokoll anzusprechen [Hei14].
   - Die AJP-Connectoren müssen aktiviert werden.
   - Bei Aufruf von https://shib1.srv.lrz.de/idp sollen die Anfragen auf den Tomcat-Server-Connector auf Port 8009 weitergeleitet werden.

   Damit die soeben erstellten Anforderungen auch aktiviert werden können, müssen in Apache die Module “ssl_headers” und “proxy_ajp” aktiviert werden. Sind diese Voraussetzungen erfüllt, kann die Website aktiviert werden. Dabei soll die von Apache bereitgestellte Default-Website deaktiviert werden.

2. Damit sich niemand am Webserver am unverschlüsselten Port 80 verbinden kann, wird aus Sicherheitsgründen dieser Port nur für den Localhost geöffnet.

3. Zum Abschluss der Webserverkonfiguration muss sichergestellt werden, dass am Webserver keine Informationen über das am IDP verwendete Betriebssystem sowie die verwendete Version des Webservers ausgelesen werden können. Weiß ein Angreifer über das Betriebssystem des Webservers bzw. über die Version des Webservers selbst Bescheid, kann er dessen bekannte Sicherheitslücken eventuell ausnutzen. Um dies zu vermeiden, können in der Datei `/etc/apache2/conf-enabled/security.conf` folgende zwei Werte verändert werden:
   - `ServerTokens prod`
   - `ServerSignature off`

5.2.6 Nutzerverzeichnis LDAP

Damit das System mit beliebigen Testusern mit verschiedenen Attributen getestet werden kann, wird dem IDP ein eigenes Nutzerverzeichnis in Form eines LDAP-Servers installiert. Hierfür wird prototypisch für den IDP ein eigener einfacher LDAP-Server aufgesetzt und mit einem Testuser "shibbolethuser" versehen. Zusätzlich soll die Kommunikation verschlüsselt erfolgen, weshalb LDAPS verwendet werden soll.

Installation LDAP Server

Bei der Installation des LDAP Servers werden aus den Komponenten des Hostnamens shib1.srv.lrz.de automatisch die nötigen Informationen abgeleitet. Es muss nur ein Passwort für den LDAP-Administrator angegeben werden. Es ergibt sich folgende automatische Konfiguration, die mittels des Kommandos `slapcat` abgerufen werden kann:

- Base-DN: `dc=srv,dc=lrz,dc=de`
- Organization Name: `srv.lrz.de`
- Admin Base-DN: `dn: cn=admin,dc=srv,dc=lrz,dc=de`

Die vom LRZ beantragten virtuellen Maschinen sind sehr restriktiv vorkonfiguriert, weshalb es bei der Verbindung zum LDAP Server zu Problemen kommen kann. Im Falle von Verbindungsproblemen, kann in der Datei `/etc/hosts.allow` eine Erlaubnis für den LDAP-Server gesetzt werden. So erlaubt der LDAP-Server Verbindungen von der Localhost-Adresse und dem eigenen Hostnamen.

Base-DN für Benutzer und Gruppen

Bisher wurde nur der Base-DN für den Administrator angelegt. Für das Management von Usern müssen noch User und Gruppen Base-DNs angelegt werden. Hierfür wird ein "LDAP interchange formate file" mit den nötigen Angaben zu "people" und "groups" erzeugt und der Eintrag wird im LDAP hinterlegt.

Es wird ein Useraccount "shibbolethuser" mittels einer weiteren LDIF Datei `new_user.ldif` erzeugt. Als Vorname und Nachname werden "Hans Meier" gewählt. Mit Hilfe dieses Accounts kann später der Loginvorgang am IDP überprüft werden.

Konfiguration LDAPS

Zur Sicherheit sollte die Kommunikation mit dem LDAP-Server ebenfalls verschlüsselt erfolgen. Als Zertifikate werden hierfür das Serverzertifikat `shib1.srv.lrz.de.pem` und das zugehörige CA-Zertifikat verwendet, da die Beantragung eines LDAP-Zertifikats an der LRZ-PKI für den Testzweck zu aufwendig ist. In einem produktiven System sollten hierfür eigene Zertifikate an einer vertrauenswürdigen Zertifizierungsstelle beantragt werden. Für Details zu den Konfigurationsschritten bei der Aktivierung von LDAPS sei auf das entsprechende Kapitel in Anhang 1 verwiesen.

5.2.7 Shibboleth Identity Provider

An diesem Punkt sollten alle Vorarbeiten für die Konfiguration des Shibboleth Identity Providers beendet sein. Ob der IDP richtig installiert wurde und korrekt läuft, kann mittels eines Shibboleth-internen Status Skripts ermittelt werden. Als Output sollten Informationen über den IDP erscheinen, wie z.B.:

```
1  ### Operating Environment Information
2  operating_system: Linux
3  operating_system_version: 4.9.0-9-amd64
4  operating_system_architecture: amd64
5  jdk_version: 1.8.0_212
6  available_cores: 1
7  used_memory: 172 MB
8  maximum_memory: 1820 MB
9
10 ### Identity Provider Information
11 idp_version: 3.4.3
12 start_time: 2019-07-15T18:30:50+02:00
13 current_time: 2019-07-15T18:30:51+02:00
14 uptime: 825 ms
15
16 service: shibboleth.LoggingService
17 last successful reload attempt: 2019-06-18T05:41:36Z
18 last reload attempt: 2019-06-18T05:41:36Z
19
20 service: shibboleth.ReloadableAccessControlService
21 last successful reload attempt: 2019-06-18T05:42:19Z
22 last reload attempt: 2019-06-18T05:42:19Z
23
24 service: shibboleth.MetadataResolverService
25 last successful reload attempt: 2019-06-18T05:42:16Z
26 last reload attempt: 2019-06-18T05:42:16Z
27
28 [...]
```

Ist die Statusabfrage erfolgreich verlaufen, kann mit der Inbetriebnahme einer Datenbank fortgefahren werden.

In den folgenden Abschnitten “Interne Datenhaltung: MySQL Datenbank”, “Anbindung der Datenbank: Data Source” und “Aktivierung der Persitstent-ID” wird erläutert, wie der Shibboleth IDP seine internen Daten handhabt und speichert bzw. welche Überlegungen für einen effizienten und korrekten Betrieb notwendig sind.

Interne Datenhaltung: MySQL Datenbank

Der Shibboleth IDP betreibt einige Komponenten, die die Speicherung von Daten in einer Datenbank benötigen, da die Default Speicher-Services von Shibboleth Daten zu Sessions, User Consent - insbesondere bezüglich Attributfreigaben - sowie Persistent-IDs (siehe Abschnitt "Aktivierung der Persistent-ID") entweder in Client-Side-Cookies oder im Hauptspeicher des Servers gespeichert werden. Diese Daten sind entweder nur auf Client-Seite gespeichert oder flüchtig im Hauptspeicher, weshalb eine Datenbank zur dauerhaften Speicherung erforderlich ist. Als Notlösung kommt die flüchtige Variante in Frage, jedoch ist dies laut DFN-AAI für Produktivsysteme nicht zu empfehlen, da spezielle SAML-Funktionen wie die Persistent-ID oder die Implementierung eines Single-Logouts ohne eine Speicherung der Daten nicht möglich ist [DFN19]. Zur Auswahl stehen sowohl MySQL als auch PostgreSQL, beide Datenbanken sind kompatibel für den Einstaz im Shibboleth IDP, da beide eine relationale Datenbank implementieren.

Die Entscheidung fiel auf MySQL, da sich PostgreSQL vor allem zur Verwendung in sehr großen Umgebungen anbietet, in denen Lese- und Schreibgeschwindigkeiten äußerst wichtig sind. MySQL ist dagegen die Wahl, wenn eine Datenbank für webbasierte Projekte benötigt wird. Dies ist im Testbed der Fall. In der Testumgebung steht durch den geringen Aufwand an Transaktionen die Effizienz beim Lesen und Schreiben der Datenbank an zweiter Stelle. MySQL ist einfach aufzusetzen und ist ohne großen Konfigurationsaufwand effizient zu nutzen. Hinzu kommt, dass MySQL als eine sehr sichere Datenbank gilt [Bha19].

In diesem Abschnitt wird detailliert ausgeführt, wie eine MySQL Datenbank mittels JPA (Java Persistence API) und Hibernate konfiguriert werden kann. Mit AJP werden die relationalen Daten in Java gehandhabt, mit Hibernate wird die objekt-relationale Abbildung der Daten möglich. Auf Hibernate und JPA wird an dieser Stelle nicht näher eingegangen, es handelt sich hier um Shibboleth-interne Vorgänge. Details sind für den Aufbau des IDP nicht relevant.

Konfiguration MySQL Zuerst muss eine logische Datenbank Namens "shibboleth" erstellt werden. In ihr sollen die Daten gespeichert werden, die der IDP längerfristig speichern muss. Hierfür werden die Tabellen "shibpid" für die Persistent-ID und "StorageRecords", z.B. für Informationen zum User Consent, angelegt. Unter "shibpid" werden sogenannte Persistent-IDs gespeichert. Mit diesen persistenten Identifiern kann zwischen IDP und SP ein eindeutiger Identifier für einen Benutzer ausgetauscht werden, ohne dass private Daten des Users offengelegt werden müssen. Die Persistent-ID wird vom IDP genieriert, sobald ein User zum ersten mal Zugang zu einem spezifischen SP erhalten will. Für die Konfiguration der Datenbank wird ein Skript `shibboleth-dp.sql` angelegt. Es generiert die benötigten Tabellen und definiert einen Benutzer "shibboleth", der auf die Datenbank zugreifen kann (vgl. für Details zum Skript Anhang 1 an entsprechender Stelle).

Normalerweise wird die MySQL-Java-Library von Tomcat beim Start eingelesen. Sie wird in der Tomcat-Konfiguration unter `/etc/tomcat8/catalina.properties` in den "Common Loader" aufgenommen. Damit bei einem Upgrade von MySQL die Datei `/etc/tomcat8/catalina.properties` nicht jedes mal manuell verglichen und neu überarbeitet werden muss, kann das MySQL JAR-File mit dem Runtime-Library-Verzeichnis von Tomcat verlinkt werden. So wird sie dann jedes Mal automatisch eingelesen [DFN19].

Data Source Der Datenbankzugriff wird über den JPAStorageService gekapselt, eine Möglichkeit über JPA und Hibernate ORM Daten abzuspeichern, damit die Suche und die Persistenz der relationalen Datenbank funktionieren. Zur Konfiguration der Verbindung mit der Datenbank müssen der Datei `/opt/shibboleth-idp/conf/global.xml` folgende Beans hinzugefügt werden:

- shibboleth.MySQLDataSource
- shibboleth.JPAStorageService
- shibboleth.JPAStorageService.EntityManagerFactory
- shibboleth.JPAStorageService.JPAVendorAdapter

Die vollständige Konfigurationsdatei ist in Anhang 1 im selben Unterkapitel einzusehen.

Zur Verbindung mit dem IDP müssen dort schließlich noch die in diesem Kapitel für MySQL definierten Eigenschaften für den Datenbank-Zugriff in `/opt/shibboleth-idp/conf/idp.properties` ergänzt werden.

Aktivierung der Persistent-ID Die Persistent-ID ist ein Identifier, welcher die Privatsphäre eines Nutzers schützt, da der Benutzer durch diese zufällig durch den IDP erzeugte ID identifiziert werden kann und keine weiteren Benutzerangaben nötig werden. Die Persistent-ID wird nur zwischen IDP und SP geteilt. Sie wird vom IDP erzeugt, sobald der Benutzer das erste Mal versucht, einen speziellen SP zu kontaktieren. Wie im vorigen Abschnitt zur Datenbank MySQL bereits erwähnt, soll der IDP des Testbeds die Persistent-ID umsetzen. Die nötige Tabelle in der Datenbank wurde hierfür bereits angelegt (shibpid). Die Persistent-ID ist für den selben Benutzer über mehrere Sessions hinweg stets die gleiche, sofern er den selben SP kontaktiert [SWI19].

Die Konfiguration der Generierung eines persistenten Identifiers erfolgt in drei Schritten [DFN19]: Es muss festgelegt werden, wie die ID generiert und abgelegt werden soll. Dies kann in der Konfigurationsdatei `saml-nameid.properties` mit folgenden Parametern festgelegt werden.

- `idp.persistentId.generator = shibboleth.StoredPersistentIdGenerator`: Befehl, dass Shibboleth eine Datenbank zur Speicherung der Persistent-ID verwenden soll.
- `idp.persistentId.salt`: Hier soll ein zufälliger Salt-Wert angegeben werden, damit die Generierung der Persistent-ID möglichst randomisiert abläuft. Für das Testbed wurde hier das Ergebnis des Befehls `openssl rand -base64 36` verwendet. Der Befehl bedeuted, dass der Output im Base64 Format sein soll und der Saltwert die Länge 36 besitzt.
- `idp.persistentId.dataSource = shibboleth.MySQLDataSource`: Verwende die bereits konfigurierte MySQL-Datenbank.
- `idp.persistentId.sourceAttribute = uid`: Hier wird mit “uid” ein Quellattribut aus unserem LDAP-Benutzerverzeichnis angegeben. Die UID wird deshalb verwendet, da hier ein Attribut benötigt wird, das immer für jeden Nutzer eindeutig ist. In der Regel ist das bei dem Attribut UID der Fall. Ist dies nicht der Fall, muss hier ein anderes eindeutiges Attribut aus dem Nutzerverzeichnis gefunden werden.

Um die Konfiguration der Persistent-ID abzuschließen, sind noch zwei weitere kleine Konfigurationsschritte notwendig, wobei an dieser Stelle auf die vollständige Dokumentation in Anhang 1 verwiesen sei.

Jetzt sind die Datenbankverbindung und die Persistent-ID konfiguriert, wodurch Session- sowie User-Consent-Informationen darin abgespeichert werden. Als Nebeneffekt dieser Konfiguration wird die SAML Funktion des Single-Logout ermöglicht. In dieser Arbeit wird auf diese Funktionalität aufgrund der zeitlichen Begrenztheit aber nicht näher eingegangen.

LDAP-Anbindung für Benutzerauthentifizierung und Attribute Resolving

Nachdem in den vorangegangenen Abschnitten die interne Datenhaltung am IDP geklärt wurde, wird in diesem Abschnitt auf die Anbindung an eine externe Datenquelle - in diesem Fall das LDAP-Nutzerverzeichnis - eingegangen.

Ein Shibboleth IDP kann die Authentifizierung mit verschiedenen Datenquellen durchführen, in diesem Testbeispiel wurde sich für LDAP entschieden. Obwohl der IDP den LDAP Server sowohl für die Authentifizierung als auch für das Attribute Resolving benötigt, handelt es sich um zwei unterschiedliche Verbindungen. In Shibboleth reicht es dagegen aus, in `ldap.properties` die LDAP-Anbindung zu definieren. Der Attribute Resolver kopiert einfach die Konfiguration der Authentifizierung. Für die LDAP-Anbindung benötigt man folgende Informationen:

- Die Methode zur Authentifizierung muss auf “bindSearchAuthenticator” gesetzt werden. Dies bedeutet, dass der LDAP Client sich an einen Systemaccount binden muss, bevor er nach Benutzeraccounts suchen kann.

- TLS- und Zertifikatseinstellungen müssen dem LDAP-Server entsprechend gesetzt werden:
  - SSL wird aktiviert.
  - StartTLS wird deaktiviert.
  - Als Vertrauenspunkt wird ein Zertifikat verwendet (`idp.authn.LDAP.sslConfig`). Der IDP kann aber auch `jvmTrust` (ein Trust Store der Java Virtual Machine) oder `keyStoreTrust` (Zertifikate sind in einem speziellen Key Store gespeichert) verwenden.
  - Als Zertifikat wird das vertrauenswürdige Serverzertifikat verwendet.

- LDAP-spezifische Daten für die Anbindung:
  - eine LDAP URL: `ldaps://127.0.0.1:636`.
  - einen Base DN: ou=people,dc=srv,dc=lrz,dc=de.
  - einen User Filter (Defaultwert): (uid=user).
  - einen Bind DN: cn=admin,dc=srv,dc=lrz,dc=de.
  - Bind-DN Credentials: Passwort.

5.2.8 Föderationsmetadaten-Aktualisierung

Der IDP muss sich in regelmäßigen Abständen eine aktuelle Version der Föderationsmetadaten herunterladen, damit stets das Vertrauensverhältnis innerhalb der Föderation erhalten bleibt (vgl. Abbildung 5.1, Schritte 2 und 4). Der IDP muss daher wissen, wie und woher er die Föderationsmetadaten erhalten kann. In `/opt/shibboleth-idp/conf/metadata-providers.xml` kann festgelegt werden, auf welche Weise er die Daten beziehen kann. Insgesamt stehen sieben Varianten zur Auswahl [**?**]:

1. `ChainingMetadataProvider`: ein Container für eine Folge von unterschiedlichen Metadata-Providern.

2. `DynamicHTTPMetadataProvider`: ein dynamischer Provider, welcher bei Bedarf von einem HTTP-Server die Metadaten aktualisiert.

3. `LocalDynamicMetadataProvider`: ein dynamischer Provider, welcher bei Bedarf eine aktuelle Version der Metadaten aus einer lokalen Quelle - wie dem Dateisystem - holt.

4. `FilesystemMetadataProvider`: ein Provider, der die Metadaten aus dem Dateisystem in einem Hintergrund-Thread lädt.

5. `FileBackedHTTPMetadataProvider`: ein Provider, der die Metadaten in einem Hintergrund-Thread von einem HTTP-Server lädt.

6. `ResourceBackedMetadataProvider`: ein Provider, der die Metadaten von einem komplexeren Quelle, wie einem Repository, in einem Hintergrund-Thread laden kann.

7. `InlineMetadataProvider`: dieser Provider erlaubt es, die IDPs und SPs direkt in der Datei anzugeben.

Für das Testsystem wurde aus Gründen der einfachen Handhabung der FileBackedHTTP-MetadataProvider gewählt. Hierbei wird in der Datei `/opt/shibboleth-idp/conf/metadata-providers.xml` der Typ und der Dateipfad angegeben:

```
1 <MetadataProvider id="LocalMetadata" xsi:type="FilesystemMetadataProvider"
      metadataFile="/var/www/html/metadata/federation-metadata.xml"/>
```

Ziel soll es jedoch sein, die Metadaten auf einen Webserver abzulegen, von dem sich der IDP in regelmäßigen Abständen eine aktuelle Version der Datei herunterladen kann. Im bisherigen Testaufbau konnte diese Variante noch nicht umgesetzt werden, da es zu nicht auflösbaren Fehlermeldungen kam.

5.2.9 Attribut-Generierung und -Freigabe

Für den IDP muss konfiguriert werden, welche Attribute freigegeben werden sollen - mittels eines sogenannten “Attribute Filter” - und wie diese vereinheitlicht an einen SP weitergegeben werden sollen - mittels eines “Attribute Resolver” (vgl. Schritte 8 und 9 in Abbildung 5.1).

Attribute Resover

Der Attribute-Resolver des IDP bekommt seine "rohen" Daten vom Data Connector aus dem Identity Management bzw. Nutzerverzeichnis (z.B. LDAP, AD, SQL). Der Data Connector des Testsystems bezieht seine Daten aus einem LDAP Verzeichnis, wie in den Anforderungen definiert wurde (vgl. Abbildung 5.1, Schritt 7). Da die Attribute aus dem LDAP Verzeichnis nicht mit den definierten Attributen übereinstimmen müssen, muss eine Abbildung der Attribute vom Nutzerverzeichnis auf die vom IDP freigegebenen Daten geschehen. Mit dem Attribute Resolver - definiert in der Datei `/opt/shibboleth-idp/conf/attribute-resolver.xml` - werden die Daten auf die definierten Attribute umgeschrieben (Abbildung 5.1, Schritt 8). Der Mechanismus hierfür lautet "Attribute Definition". Es gibt verschiedene Typen, z.B.:

- Simple Attribute Definition
- Mapped Attribute Definition (Viele-zu-viele Mapping)
- Scoped Attribute Definition (z.B. begrenzt auf eine Domain o.Ä.)
- etc.

Nach der Umwandlung werden sie in eine Form gebracht, die der SP versteht. Diesen Vorgang nennt man "Attribute Encoding".

Für das Testbed wurden fünf Attribute ausgewählt, die der IDP freigeben kann:

- eduPersonPrincipalName
- PrincipalName
- mail
- surname
- givenName

Die Attribute eduPersonPrincipalName und PrincipalName wurden gewählt, da sie am LRZ als Standardattribut verwendet werden und daher für das Testsystem sinnvoll zu verwenden sind. Die beiden Attribute sollen aus Benutzereingaben bestehen. Zusätzlich wurden die Attribute "mail", "surname" und "givenName" gewählt, da sie wohl für die häufigsten Benutzer relevant sind. Sie stammen aus dem LDAP-Verzeichnis des Testsystems. Das Format der Attribute in `attribute-resolver.xml` kann aus der Datei `attribute-resolver-full.xml` übernommen werden. Sie dient als umfangreiche Vorlage für Attributsdefinitionen, falls man mehr Attribute auflösen möchte. In der Konfigurationsdatei unter `attribute-resolver.xml` können diese Attribute eingetragen werden. Zudem wird am Ende der Datei der LDAP Connector definiert, wodurch der Attribute Resolver weiß, wie er sich an das LDAP-Nutzerverzeichnis anbinden kann.

Attribute Filter

Der Attribute Filter (`/opt/shibboleth-idp/conf/attribute-filter.xml`) legt fest, welche Attribute an einen SP freigegeben werden dürfen (siehe Abbildung 5.1, Schritt 9). Die passenden Attribute werden mittels der im Attribute Resolver angegebenen ID referenziert.

Ein einfacher Attribute-Filter, passend zum oben definierten Attribute-Resolver, wird im folgenden Listing als Beispiel gegeben. Hier werden alle fünf definierten Attribute als einfacher Testfall an den SP freigegeben. Zu sehen ist dies an dem Wert `permitAny=''true''`.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!--
3      This file is an EXAMPLE policy file.  While the policy presented in this
4      example file is illustrative of some simple cases, it relies on the names
           of
5      non-existent example services and the example attributes demonstrated in
           the
6      default attribute-resolver.xml file.
7
8      This example does contain some usable "general purpose" policies that may
           be
9      useful in conjunction with specific deployment choices, but those policies
            may
10     not be applicable to your specific needs or constraints.
11 -->
12 <AttributeFilterPolicyGroup id="ShibbolethFilterPolicy"
13         xmlns="urn:mace:shibboleth:2.0:afp"
14         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
15         xsi:schemaLocation="urn:mace:shibboleth:2.0:afp http://shibboleth.net/
       schema/idp/shibboleth-afp.xsd">
16
17    <!-- Attribute an unseren Test-IdP freigeben -->
18    <AttributeFilterPolicy id="lrz_test_sp">
19         <PolicyRequirementRule xsi:type="OR">
20             <Rule xsi:type="Requester" value="https://shib2.srv.lrz.de/
                   shibboleth" />
21         </PolicyRequirementRule>
22
23         <AttributeRule attributeID="uid"                    permitAny="true" />
24         <AttributeRule attributeID="eduPersonPrincipalName" permitAny="true" />
25         <AttributeRule attributeID="mail"                   permitAny="true" />
26         <AttributeRule attributeID="surname"                permitAny="true" />
27         <AttributeRule attributeID="givenName"              permitAny="true" />
28   </AttributeFilterPolicy>
29
30 </AttributeFilterPolicyGroup>
```

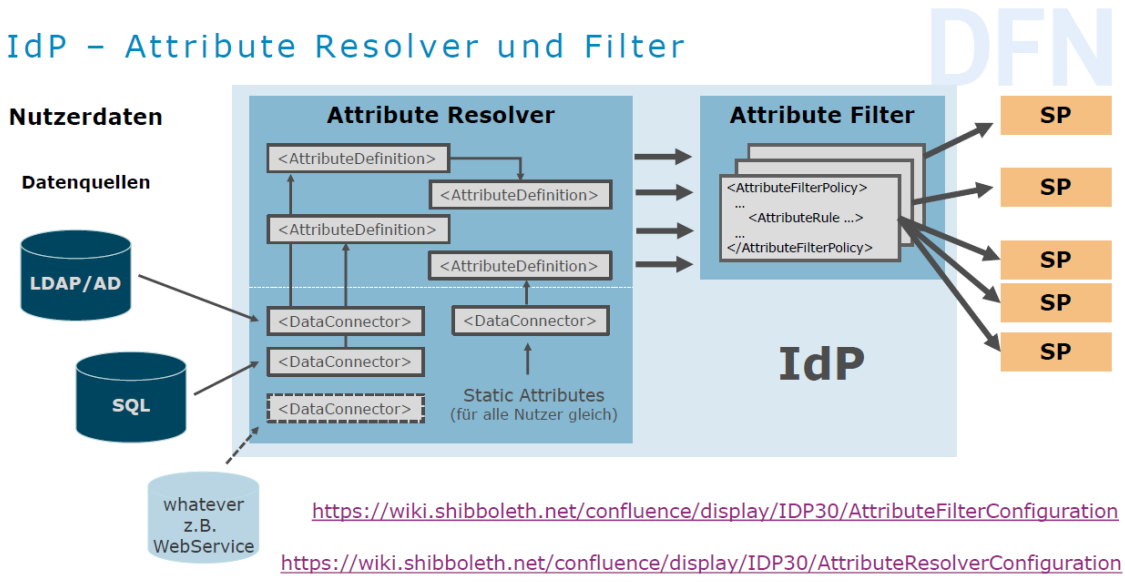

Abbildung 5.2: Zusammenhang zwischen Attribute Resolver und Attribute Filter bei der Attributfreigabe an einen SP [Pem19, F.4]

Einen Überblick, wie Attribute Resolver und Filter funktionieren, gibt Abbildung 5.12. Die Daten-Konnektoren (`DataConnector`) erhalten die Daten aus den angeschlossenen Nutzerverzeichnissen wie LDAP oder SQL. Der Attribute Resolver bildet die Daten aus dem Nutzerverzeichnis auf die definierten Attribut-Definitionen ab, die für die Weitergabe an einen SP definiert wurden. Über den Attribute Filter wird vor der Weitergabe an den entsprechenden SP entschieden, welche Attribute der SP erhält. Die freigegebenen Attribute werden im sogenannten “Attribute Statement” übertragen mit Hilfe einer SAML2 Assertion. Der empfangende SP identifiziert die Attribute anhand eines URI Wertes (meist vom Typ urn:oid).

5.3 Föderationsmetadaten

In einer Föderation ist ein Vertrauensverhältnis unabdingbar. Es muss für jeden Teilnehmer hinterlegt sein, dass er Teilnehmer der Föderation ist. Für diesen Testaufbau wurde entschieden, dass all diese Daten über an der Föderation teilnehmende IDPs und SPs in einer zentralen Datei festgehalten werden. Diese Datei wird als “Föderationsmetadaten” bezeichnet. Bei Kontaktherstellung zwischen IDP und SP oder anders herum kann diese Datei von der anfragenden Komponente konsultiert werden, ob der potentielle Kommunikationspartner in seiner Föderation enthalten ist und damit vertrauenswürdig ist. Ebenso können über die Metadaten spezielle Informationen zur Kontaktaufnahme untereinander festgehalten werden, wie z.B. E-Mail Adressen, aber auch Protokollversionen. Aus dem soeben beschriebenen Szenario wird deutlich, dass diese Datei zu jeder Zeit jeder Komponente frei zugänglich sein muss. In der Regel liegt diese Datei auf einem Webserver, von welchem sich die Komponenten in regelmäßigen Abständen eine aktuelle Kopie der Föderationsmetadaten herunterladen können. Problem ist hierbei allerdings, dass bei diesem Pull-Verfahren nicht

garantiert ist, dass alle Föderationsteilnehmer immer zur gleichen Zeit die aktuelle Version der Datei herunterladen. Aus theoretischer Sicht wäre ein Push-Verfahren konsistenter, da der Webserver in regelmäßigen Abständen an alle Komponenten eine aktuelle Version der Datei ausgeben würde (vergleiche Kapitel 2.3.5). Aus Gründen der Vereinfachung und zeitlichen Begrenztheit der Arbeit wurde entschieden, die Föderationsmetadaten von den Komponenten lokal speichern zu lassen. Hierbei ist beim Betrieb der Föderation darauf zu achten, dass die Komponenten stets die aktuellen Metadaten besitzen.

Für das in dieser Arbeit erstellte Testbed wurde eine solche Metadaten-Datei manuell konstruiert. Im Folgenden wird der Aufbau dieser Datei und die Entscheidungen bei der Erstellung näher beschrieben.

Ganz allgemein sind die Föderationsmetadaten wie in folgendem Listing dargestellt aufgebaut:

```
1  <EntitiesDescriptor Name="https://shib1.srv.lrz.de/metadata/federation-
       metadata.xml"
2
3  <!-- Zertifikatswerte des selbstsignierten Zertifikats der
       Föderationsmetadaten -->
4  <ds:Signature>
5          <!-- ... -->
6  </ds:Signature>
7
8  <!-- In diesen Bereich sind alle IDPs aufgelistet  -->
9
10     <!-- Ein IDP -->
11     <EntityDescriptor entityID="https://shib1.srv.lrz.de/idp/shibboleth">
12
13         <IDPSSODescriptor protocolSupportEnumeration="urn:oasis:names:tc:SAML
               :2.0:protocol">
14                         <!-- ... -->
15         </IDPSSODescriptor>
16
17         <!-- Daten zum IDP -->
18         <Organization>
19                 <!-- Daten zur Organisation -->
20         </Organization>
21         <ContactPerson contactType="technical">
22                 <!-- Daten zur Kontaktperson der Organisation -->
23         </ContactPerson>
24     </EntityDescriptor>
25
26 <!-- In diesem Bereich sind alle SPs aufgelistet -->
27
28     <!-- Ein SP -->
29     <EntityDescriptor entityID="https://shib2.srv.lrz.de/shibboleth">
30
31         <SPSSODescriptor protocolSupportEnumeration="urn:oasis:names:tc:SAML
               :2.0:protocol">
32                         <!-- ... -->
33         </SPSSODescriptor>
34
35         <!-- Daten zum SP -->
36         <Organization>
37                 <!-- Daten zur Organisation -->
```

```
38          </Organization>
39          <ContactPerson contactType="technical">
40                  <!-- Daten zur Kontaktperson der Organisation -->
41          </ContactPerson>
42
43      </EntityDescriptor>
44
45  </EntitiesDescriptor>
```

Die gesamte Datei enthält alle IDPs und SPs der Föderation. Eine Komponente wird in dieser Datei mittels eines `<EntityDescriptor>` beschrieben. Da die Datei sämtliche Entity-Descriptors enthält, sind die Elemente in einem `<EntitiesDescriptor>` eingebettet. Wie in Zeile 1 zu sehen ist, ist dieser Entities-Descriptor unsere Föderationsmetadaten-Datei, da der Name auf den Ort verweist, wo die Datei zu finden ist. Natürlich muss diese Datei für das Vertrauensverhältnis mit einem Zertifikat ausgestattet sein, damit die Komponenten die Integrität der Föderationsmetadaten überprüfen können. Die Werte des Zertifikats werden in das Element `ds:Signature` (Z. 4-6) eingetragen. Hier sollte ebenfalls ein vertrauenswürdiges Zertifikat der LRZ-PKI verwendet werden. Aufgrund der zeitlichen Begrenztheit der Arbeit sowie dem organisatorischen Aufwand, ein solches Zertifikat nur zu Testzwecken zu beantragen, wurde hier entschieden, dass ein selbstsigniertes Zertifikat zu Testzwecken durchaus ausreichend ist.

Es folgt der Abschnitt, in dem sämtliche IDPs der Föderation eingetragen werden. Jeder IDP wird mittels einer `entityID` in einem Entity-Descriptor beschrieben. Als `entityID` wird meist die URL zum Kontakt des IDP-Endpoints verwendet, im Falle des IDPs `https://shib1.srv.lrz.de/idp/shibboleth` (Z.11).

Wichtige technische Informationen zur Erreichbarkeit und notwendigen Daten des SSO des IDPs werden im `IDPSSODescriptor` hinterlegt (Z.13-15). Hierunter fallen Informationen wie

1. Extensions:
   a) Unterstützung von Algorithmen zur Verschlüsselung und Signatur
   b) Anzeige des Namens des IDP im Falle eines Discovery Services (`<mdui:UIInfo>`)
2. Daten zu Verschlüsselungs- und Signaturschlüssel (`<md:KeyDescriptor>`)
3. NameID-Format, z.B. häufig verwendet:
   a) Transient (temporärer anonymer Identifier)
   b) Persistent (Service-spezifischer Identifier, bestehend aus einer pseudo-zufälligen Zahl, die einen Benutzer anonym identifizieren kann, vgl. Abschnitt "Persistent-ID" aus Kapitel 5.2)
   c) Unspecified
4. Bindings, URLs, etc.

Schließlich werden Informationen zur teilnehmenden Organisation (`<Organization>`, Z. 18-20) und der verantwortlichen Kontaktperson (`<ContactPerson>`, Z. 21-23) im Falle von Rückfragen hinterlegt.

Im zweiten Bereich der Datei werden alle SPs der Föderation aufgelistet. Sie werden ebenfalls mittels Entity-Descriptors beschrieben und mit einer `entityID` referenziert (hier shib2.srv.lrz.de, Z. 29).

Auch für den SP werden im Entity-Descriptor mit einem `SPSSODescriptor` sämtliche technische Daten zum Kontakt des SP hinterlegt (Z. 31-33):

1. Extensions zum Discovery Service wie die `DiscoveryResponse`
2. Daten zu Verschlüsselungs- und Signaturschlüssel (`<md:KeyDescriptor>`)
3. NameIDs
4. Angaben zum Binding des Assertion Consumer Service (meist Liste aller möglichen Bindings des betreffenden SP)

Auch hier werden jeweils Daten zur teilnehmenden Organisation (`<Organization>`, Z. 36-38) und der verantwortlichen Kontaktperson (`<ContactPerson>`, Z. 39-41) hinterlegt

Sowohl für IDPs und SPs können natürlich sämtliche zusätzliche weitere Angaben hinterlegt werden, zum Test in einer vereinfachten Testumgebung ist der soeben beschriebene Aufbau jedoch vollkommen ausreichend. Für weitere Informationen sei hier auf das Wiki von Shibboleth verwiesen[1]. Wie die vollständige, für das Testbed konfigurierte Metadaten-Datei aussieht, ist in Anhang 1 zu sehen.

Nun da der allgemeine Aufbau der Föderationsmetadaten erläutert ist, sollen am Beispiel der Föderationsmetadaten des Testbeds die Entity-Descriptors des IDP (shib1.srv.lrz.de) und SPs (shib2.srv.lrz.de) mit Angabe von Beispielen näher betrachtet werden:

1. **IDP (shib1.srv.lrz.de)**
   - `<Extensions>`: Z.B. unter `mdui:UIInfo` kann als Anzeigename für einen Discovery Service ein Name wie “Shibboleth Test IDP” gewählt werden:

```
1 <mdui:UIInfo>
2         <mdui:DisplayName xml:lang="de">Shibboleth Test IDP</mdui:
              DisplayName>
3 </mdui:UIInfo>
```

   - `<KeyDescriptor>`: Zertifikatswert der Serverzertifikats von shib1.srv.lrz.de
   - `<NameIDFormat>`: Z.B. “transient”:

```
1 <NameIDFormat>
2         urn:oasis:names:tc:SAML:2.0:nameid-format:transient
3 </NameIDFormat>
```

   - `<SingleSignOnService>`: Angabe der verschiedenen Bindings, die der IDP akzeptiert (z.B. HTTP-POST oder HTTP-Redirect):

```
1 <SingleSignOnService Binding="urn:oasis:names:tc:SAML:2.0:bindings:
      HTTP-POST"
2         Location="https://shib1.srv.lrz.de/idp/profile/SAML2/POST/SSO
      " />
3
```

[1] https://wiki.shibboleth.net/confluence/display/CONCEPT/Metadata, aufgerufen am 23.07.2019

```
4 <SingleSignOnService Binding="urn:oasis:names:tc:SAML:2.0:bindings:
      HTTP-Redirect"
5           Location="https://shib1.srv.lrz.de/idp/profile/SAML2/Redirect
      /SSO" />
```

- `<Organization>`: Adresse und Name der Organisation (hier `https://shib1.srv.lrz.de/` und "Shibboleth Test IDP"):

```
1 <Organization>
2           <OrganizationName xml:lang="en">shib1.srv.lrz.de</
                OrganizationName>
3           <OrganizationDisplayName xml:lang="de">Shibboleth Test IDP</
                OrganizationDisplayName>
4           <OrganizationURL xml:lang="de">https://shib1.srv.lrz.de/</
                OrganizationURL>
5 </Organization>
```

- `<ContactPerson>`: Name und E-Mail-Adresse einer Kontaktperson (hier "Hans Meier"):

```
1 <ContactPerson contactType="technical">
2            <GivenName>Hans</GivenName>
3            <SurName>Meier</SurName>
4            <EmailAddress>idp-admin@shib1.srv.lrz.de.de</EmailAddress>
5 </ContactPerson>
```

2. **SP (shib2.srv.lrz.de)**

- `<Extensions>`: Z.B. als `DiscoveryResponse` wird das Binding für den Aufruf des DS angegeben:

```
1 idpdisc:DiscoveryResponse xmlns:idpdisc="urn:oasis:names:tc:SAML:
      profiles:SSO:idp-discovery-protocol"
2                 index="1" Binding="urn:oasis:names:tc:SAML:profiles:SSO:
      idp-discovery-protocol"
3                 Location="https://shib2.srv.lrz.de/Shibboleth.sso/DS" />
```

- `<KeyDescriptor>`: Zertifikatswert der Serverzertifikats von shib2.srv.lrz.de
- `<NameIDFormat>`: Z.B. "transient":

```
1 <NameIDFormat>
2           urn:oasis:names:tc:SAML:2.0:nameid-format:transient
3 </NameIDFormat>
```

- `<AssertionConsumerService>`: Angabe der verschiedenen Bindings, die der Assertion Consumer Service des SP akzeptiert (z.B. HTTP-POST). Als Default-Binding ist HTTP-POST gesetzt:

```
1 <AssertionConsumerService index="1" isDefault="true"
2              Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
3              Location="https://shib2.srv.lrz.de/Shibboleth.sso/SAML2/
      POST" />
4 <AssertionConsumerService index="2"
5              Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST-
      SimpleSign"
```

```
6           Location="https://shib2.srv.lrz.de/Shibboleth.sso/SAML2/
        POST-SimpleSign" />
7   <AssertionConsumerService index="3"
8           Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Artifact
        "
9           Location="https://shib2.srv.lrz.de/Shibboleth.sso/SAML2/
        Artifact" />
10  <AssertionConsumerService index="4"
11          Binding="urn:oasis:names:tc:SAML:1.0:profiles:browser-post"
12          Location="https://shib2.srv.lrz.de/Shibboleth.sso/SAML/POST
        " />
13  <AssertionConsumerService index="5"
14          Binding="urn:oasis:names:tc:SAML:1.0:profiles:artifact-01"
15          Location="https://shib2.srv.lrz.de/Shibboleth.sso/SAML/
        Artifact" />
```

- `<Organization>`: Adresse und Name der Organisation (hier `https://shib2.srv.lrz.de/` und "Shibboleth Test SP"):

```
1   <Organization>
2           <OrganizationName xml:lang="en">shib2.srv.lrz.de</
                OrganizationName>
3           <OrganizationDisplayName xml:lang="de">Shibboleth Test SP</
                OrganizationDisplayName>
4           <OrganizationURL xml:lang="de">http://shib2.srv.lrz.de/</
                OrganizationURL>
5   </Organization>
```

- `<ContactPerson>`: Name und E-Mail-Adresse einer Kontaktperson (hier "Berta Müller"):

```
1   <ContactPerson contactType="technical">
2           <GivenName>Berta</GivenName>
3           <SurName>Müller</SurName>
4           <EmailAddress>sp-admin@shib2.srv.lrz.de.de</EmailAddress>
5   </ContactPerson>
```

Die Datei muss nun an einem passenden Ort abgelegt werden. Da die Föderationsmetadaten später auf einem Webserver verfügbar sein sollen, bietet sich der Apache Webserver des IDP als Speicherort an: Den Ordner `/var/www/html/metadata/` anlegen und die Datei `federation-metadata.xml` darin ablegen. Bei Bedarf die `index.html` Datei löschen. Jetzt können sich alle Komponenten über die Adresse `https://shib1.srv.lrz.de/metadata/federation-metadata.xml` eine aktuelle Version der Föderationsmetadaten herunterladen.

5.4 Shibboleth Service Provider

In diesem Abschnitt wird beschrieben, wie ein einfacher Service Provider zu Testzwecken aufgesetzt werden kann. Da aktuell keine konkrete zu schützende Ressource im Testsystem vorhanden ist, soll der SP den Zugriff auf einen speziellen Ordner schützen. In der Praxis bedeutet dies, dass über eine URL (`https://shib2.srv.lrz.de/Shibboleth.sso/Login`) der Vorgang des Web Browser SSO simuliert werden soll (vgl. Kapitel 5.1, Abbildung 5.1).

Im Gegensatz zum Shibboleth IDP läuft auf einem Shibboleth SP ein Dämon namens `shibd` und ein Webserver-Modul `mod_shib`, welches von Apache Webservern und Windows

Internet-Informations-Services (IIS) unterstützt wird. Der Dämon und das Modul laufen im Hintergrund und warten auf Anfragen eines Benutzers.

5.4.1 Voraussetzungen

Bevor mit einer Installation begonnen werden kann, müssen einige Voraussetzungen bestehen:

- Betriebssystem: eine virtuelle Maschine mit einem Linux Betriebssystem, in diesem Fall Debian 9. Für die Hardware sollten 4 GB RAM und 20 GB Festplattenspeicher eingeplant werden (vgl. Überlegungen zu IDP, Kapitel 5.1.1).
- Ein Zugang über SSH mit einem Benutzer mit Root-Rechten muss vorhanden sein (vgl. Kapitel 5.1.1).
- Netzwerk:
  - Es wird ein Hostname (shib2.srv.lrz.de) und eine statische IP-Adresse benötigt (129.187.255.172). Sie werden vom LRZ-Netzmanagement zugewiesen.
  - Die Firewall sollte für SSL-Verbindungen, über welche externe Komponenten auf den SP zugreifen, die Ports 443 und 8443 geöffnet haben.
  - Die DNS Auflösung zum IDP muss möglich sein. Hierfür die IP Adresse des IDP (129.187.255.171) und den FQDN (shib1.srv.lrz.de) für die interne Namensauflösung angeben, da kein externer DNS-Server zur Verfügung steht.
  - Falls ein externer NTP-Server verwendet wird, sollte dieser ebenso an der Firewall freigeschaltet werden (vgl. Kapitel 5.1.1).
- Zertifikat: Für den Hostnamen (shib2.srv.lrz.de) sollte ein Zertifikat bereitgestellt werden, dessen Zertifizierungsstellen die gängigen Browser vertrauen. Für den Server wurde von der LRZ-Zertifizierungsstelle ein Zertifikat beantragt. Schlüssel und Zertifikat werden unter `/etc/ssl/localcerts/` bzw. `/etc/ssl/localkeys/` abgelegt. Das Zertifikat für die Zertifizierungsstelle liegt unter `/etc/ssl/chains/`. Da es sich hier um ein vergleichbares Zertifikat für den Server handelt wie beim IDP, sei zur näheren Erläuterung des Vertrauensverhältnisses auf Kapitel 5.1.1 verwiesen.

5.4.2 Installation und Konfiguration von Basiskomponenten

Bevor mit der Installation des Shibboleth Service Providers begonnen werden kann, müssen auf dem SP einige Programme installiert werden, die für den Betrieb notwendig sind.

- Apache2: Wie für den IDP benötigt der SP zur externen Erreichbarkeit seiner Dienste aus dem Netzwerk einen Webserver. Auch hier wurde der Apache Webserver gewählt, da er das Webserver-Modul `mod_shib` unterstützt. Für weitere Auswahlkriterien sei auch hier auf Kapitel 5.1.1 verwiesen.
- Testing und Debugging (vgl. Erläuterungen Kapitel 5.1.1):
  - curl: Zum Aufruf von Webseiten innerhalb einer Umgebung ohne grafischer Oberfläche.

- net-tools: Das Paktet enthält wichtige Programme zur Überprüfung der Netzwerkaktivitäten.
- strace: Nachvollzug der Prozessabläufe.

Da der Shibboleth Service Provider nur offizielle Binaries für RPM-basierte Linux Distributionen anbietet[2], betreibt die SWITCH-AAI ein eigenes Repository mit Stable-Releases für Debain[3]. Deshalb muss in der `sources.list` die entsprechende Quelle ergänzt werden.

Für den shibd-Dämon müssen schließlich noch Leserechte für den Schlüssel des Zertifikats für shib2.srv.lrz.de vergeben werden.

5.4.3 Installation Shibboleth Service Provider

Aufgrund der nicht-standard Paktetquelle der SWITCH-AAI muss bei der Installation folgendes beachtet werden (für Details sei auf Anhang 1 verwiesen): Das Repository-Package muss zuerst heruntergeladen und installiert werden. Nach der Installation sollte mittels des shibd-Dämons (Aufruf über `shibd -t`) überprüft werden, ob die Installation erfolgreich verlaufen ist. Wichtig ist hierbei die Ausgabe der letzten Zeile:

```
1 overall configuration is loadable, check console for non-fatal problems
```

Sind die bisherigen Schritte erfolgreich verlaufen, kann mit der Konfiguration des Systems fortgefahren werden.

5.4.4 Konfiguration Apache2

Es muss für Apache ein SSL-fähiger Virtual Host angelegt werden, über welchen die Website gehostet werden kann, da auch der SP nur verschlüsselt nach außen kommunizieren soll. Möglich wird dies durch das Anlegen eines Virtual Host "shib2.srv.lrz.de". Da ein Benutzer in der Regel auf eine geschützte Ressource zugreifen möchte, wird zusätzlich unter `/var/www/html/` ein Ordner `secure-all/` angelegt, über welchen die zu schützenden Ressourcen abrufbar sind. An diesem Ort können sämtliche Ressourcen zu Testzwecken abgelegt werden.

Der Virtual Host muss folgende Kriterien erfüllen:

- Der Virtual Host soll nur über den verschlüsselten Port 443 erreichbar sein.
- Der Servername muss shib2.srv.lrz.de lauten.
- Als SSL-Protokolle dürfen SSLv2, SSLv3 und TLSv1 nicht verwendet werden, da sie inzwischen als nicht mehr sicher gelten.
- Die Verschlüsselungsmethode des Servers soll gegenüber der des Clients bevorzugt werden (`SSLHonorCipherOrder`).
- SSL Compression muss vorsichtshalber deaktiviert werden, da am IDP mit sensiblen Nutzerdaten umgegangen wird. Verwendet ein Benutzer einen älteren verwundbaren Browser, kann hier nämlich eine Attacke namens "BEAST"[Fis12] gestartet werden.

[2] https://wiki.shibboleth.net/confluence/display/SP3/LinuxInstall, aufgerufen am 12.06.2019
[3] http://pkg.switch.ch/switchaai/, aufgerufen am 12.06.2019

- HTTP Strict Transport Security (HSTS) muss aktiviert werden. Hierdurch wird versichert, dass die Website nur über HTTPS angesprochen wird, da der Server in die Antwort einen HSTS-Header mit einfügt. Der Browser merkt sich nun, von welchem Server er einen HSTS-Header bekommen hat und weigert sich, diese mit dem HTTP-Protokoll anzusprechen [Hei14].
- Bei Aufruf von `https://shib2.srv.lrz.de/shibboleth` soll auf die Metadaten des SP weitergeleitet werden.
- Eine zu schützende Ressource ist unter `/var/www/html/secure-all/` angelegt. Durch Aufruf der Seite `https://shib2.srv.lrz.de/secure-all/` (Zugriff auf geschützte Ressource) soll der Authentifizierungsprozess des Benutzers initiiert werden.

Damit die soeben angeführte Anforderung nach Verschlüsselung auch in Kraft treten kann, muss für Apache das Modul `ssl_header` aktiviert werden. Zudem ist darauf zu achten, dass die Default-Website deaktiviert wird.

Damit sich niemand am Webserver am unverschlüsselten Port 80 verbinden kann, wird aus Sicherheitsgründen dieser Port nur für den Localhost geöffnet.

5.4.5 Shibboleth Service Provider: Konfiguration

Im Gegensatz zu der recht aufwendigen und durchaus komplexen Konfiguration des Shibboleth IDP kann die Konfiguration für den SP größtenteils in einer einzigen Datei vorgenommen werden, wie im Folgenden dargestellt wird.

Die Konfigurationsdatei shibboleth2.xml

Für den Shibboleth SP kann ein Großteil der Konfiguration in der Datei `shibboleth2.xml` im Element `ApplicationDefaults` erledigt werden. Folgende Informationen werden für den SP benötigt:

1. Die `entityID` des SP muss eingetragen werden:

```
1 <ApplicationDefaults entityID="https://shib2.srv.lrz.de/shibboleth">
```

2. Im `<Sessions>` Element können neben Informationen zu Session-Dauer, Adressüberprüfungen, Cookie-Handling und Protokoll-Handlern weitere Informationen angegeben werden. An dieser Stelle werden nur zu konfigurierende Elemente aufgelistet, für die vollständige Konfiguration mit sämtlichen Default-Werten sei auf die vollständige Konfigurationsdatei in Anhang 1 verwiesen. Beispielsweise wird ein SSO-Endpoint für einen IDP angegeben, damit der SP weiß, wie er sich im Falle einer nötigen Authentifizierung an einen IDP verbinden kann:

```
1 <SSO entityID="https://shib1.srv.lrz.de/idp/shibboleth">
2       SAML2
3 </SSO>
```

   Existieren in der Föderation mehr als ein IDP - was in Produktivsystemen der Fall ist, kann statt der `entityID` eine URL zu einem Discovery Service angegeben werden. Da für das Testbed noch kein DS existiert, wurden hier Platzhalter verwendet:

```
<SSO discoveryProtocol="SAMLDS" discoveryURL="https://ds.example.org/DS/
    WAYF">
      SAML2
</SSO>
```

3. `<MeatdataProvider>`: Wie auch im IDP muss für den SP angegeben werden, wie er die Föderationsmetadaten beziehen kann. Für nähere Informationen zu den unterschiedlichen Möglichkeiten sei hier auf Kapitel 5.1.8 verwiesen. Da sich aus Gründen der schnelleren Testmöglichkeit durch einfacheren Aufbau dafür entschieden wurde, die Föderationsmetadaten vorerst im lokalen Dateisystem zu speichern, muss dem SP dies mitgeteilt werden:

```
<MetadataProvider type="XML" validate="true" path="federation-metadata.
    xml" />
```

   Die Föderationsmetadaten werden relativ zum Shibboleth-Verzeichnis `/etc/shibboleth/` angegeben. Sollen die Metadaten zukünftig über einen Webserver im Pull-Verfahren abgerufen werden, muss an dieser Stelle wie für den IDP ein entsprechendes Bean angegeben werden (vgl. an dieser Stelle vollständige Datei in Anhang 1).

4. Angaben zu den Pfaden der Attribute Map (vgl. nächster Abschnitt) und der Attribute Policy im Dateisystem:

```
<AttributeExtractor type="XML" validate="true" reloadChanges="false" path
    ="attribute-map.xml" />
<AttributeFilter type="XML" validate="true" path="attribute-policy.xml" />
```

5. Angaben zum Pfad des Zertifikats und Schlüssels:

```
<CredentialResolver type="File" key="/etc/ssl/localkeys/shib2.srv.lrz.de-
    nopw.key"
                certificate="/etc/ssl/localcerts/shib2.srv.lrz.de.pem" />
```

Attribute Map

Der SP empfängt vom IDP die SAML-Assertions mit Daten über Attribute und Autorisierung des Benutzers. Über eine eindeutige Kennung der Attribute (URI) identifiziert der SP die Attribute, die ihm vom IDP übermittelt wurden. Mit Hilfe der Datei `/etc/shibboleth/attribute-map.xml` bildet der SP die Attribute auf interne Variablen ab (vgl. Abbildung 5.1, Schritt 10). Mit einer ID werden die Variablennamen definiert und können so abgebildet werden.

Folgende Attribute müssen abgebiltet werden können, da sie am IDP im Attribute Resolver zur Freigabe konfiguriert wurden. Es wurden zu Testzwecken vier der fünf definierten Attribute aus dem IDP ausgewählt:

- eduPersonPrincipalName
- mail
- surname

- givenName

In der Datei `attribute-map.xml` müssen hierfür die soeben identifizierten vier Elemente abgebildet sein:

```
1     <!-- eduPersonPrincipalName -->
2     <Attribute name="urn:oid:1.3.6.1.4.1.5923.1.1.1.6" id="eppn">
3         <AttributeDecoder xsi:type="ScopedAttributeDecoder" caseSensitive="
              false"/>
4     </Attribute>
5     <Attribute name="urn:mace:dir:attribute-def:eduPersonPrincipalName" id="
          eppn">
6         <AttributeDecoder xsi:type="ScopedAttributeDecoder" caseSensitive="
              false"/>
7     </Attribute>
8
9 <!-- CORE ATTRIBUTES -->
10
11    <!-- E-mail -->
12    <Attribute name="urn:oid:0.9.2342.19200300.100.1.3" id="mail"/>
13    <Attribute name="urn:mace:dir:attribute-def:mail" id="mail"/>
14
15    <!-- Given name -->
16    <Attribute name="urn:oid:2.5.4.42" id="givenName"/>
17    <Attribute name="urn:mace:dir:attribute-def:givenName" id="givenName"/>
18
19    <!-- Surname -->
20    <Attribute name="urn:oid:2.5.4.4" id="surname"/>
21    <Attribute name="urn:mace:dir:attribute-def:sn" id="sn"/>
```

Wie am Listing ersichtlich ist, wird z.B. das Attribut “eppn” auf die interne Variable “eduPersonPrincipalName” abgebildet. Mittels der ID können die einzelnen Attribute hierbei referenziert werden. Durch die Datei `attribute-policy.xml` können die Variablen gefiltert werden (vgl. Abbildung 5.1, Schritt 11), d.h. der SP entscheidet, welche Attribute er an den Benutzer letztendlich zurückgeben will (vgl. Abbildung 5.1, Schritt 12). In der Regel reicht die Konfiguration der Default-Datei aus, so auch für diesen Testfall.

6 Automatisierungsmöglichkeiten der Installation eines Shibboleth Identity Providers

Wie aus dem vorangegangenen Kapitel 5 ersichtlich wurde, ist die manuelle Installation und Konfiguration - v.a. des Shibboleth IDP - sehr aufwendig, wodurch sich zwangsläufig die Frage nach Möglichkeiten einer automatisierten Herangehensweise stellt. Ein Ziel dieser Arbeit ist es daher, die Installation des Shibboleth IDP so weit wie möglich zu automatisieren, damit zum Test diverser Szenarien schnell neue IDPs hinzugefügt werden können. Da sich die Installation eines Shibboleth SP im Vergleich zu einem IDP relativ unkompliziert verhält, wurde der Fokus zunächst nur auf die Automatisierung einer IDP Installation gelegt. Warum sich hierfür das Framework "Ansible"[1] anbietet, wird in den nächsten beiden Abschnitten ausführlich diskutiert.

6.1 Funktionsweise Ansible

Ansible ist eine Open-Source Software, mit dem sich die Konfiguration und Administration von Systemen automatisieren lässt, die aber auch andere Möglichkeiten wie Cloud Provisioning bietet. Es gibt zahlreiche Lösungen für die Aufgabe der Automatisierung (vgl. Kapitel 6.2), jedoch zeichnet sich Ansible besonders dadurch aus, dass es mit geringen Voraussetzungen funktionsfähig ist. So müssen die zu verwaltenden Systeme nur OpenSSH und Python installiert haben, welche jedoch für die meisten Linux-Distributionen in der Standardsoftware bei der initialen Installation bereits enthalten sind. Besonders ist hierbei, dass kein User-Agent oder andere zusätzliche Software auf den Zielsystemen installiert werden müssen. Zudem arbeitet Ansible mit der Sprache YAML (Yet Another Markup Language), welche für eine Markup-Sprache sehr schnell zu lernen und einfach zu lesen ist [Aug18].

Ansible verbindet sich mit den zu konfigurierenden Systemen (auch "Nodes" genannt), über SSH-Verbindungen, wodurch es kleine Programme ("Ansible Module") auf den Zielsystemen ausführt. Nach der erfolgreichen Beendigung wird das Modul vom Zielsystem wieder gelöscht. Zur Ausführung von Ansible ist kein extra Server notwendig, es kann von jedem beliebigen Host des Netzwerks aus aufgerufen werden.

Wie ist Ansible nun aber organisiert? Die Maschinen, auf denen gewisse Ansible-Tasks ausgeführt werden sollen, werden in einer einfach zu konfigurierenden INI-Datei festgehalten, auch "Inventar" genannt. In dieser Datei können die Zielsysteme auch gruppiert werden, wie am Listing "example.ini" zu sehen ist:

```
1 [webservers]
2 www1.example.com
3 www2.example.com
4
```

[1] `https://www.ansible.com/`, aufgerufen am 18.07.2019

```
5 [dbservers]
6 db0.example.com
7 db1.example.com
```

Im Beispiel-Listing wurden also zwei Webserver und zwei Datenbankserver in das Inventar aufgenommen [ans19].

Was nun für jeden angegebenen Host automatisiert von Ansible erledigt werden soll, wird in sogenannten "Playbooks" festgehalten. Ein Playbook ist eine YAML-Datei, in der die Schritte der Automatisierung definiert werden. In der Datei werden die auszuführenden Aufgaben als "Roles" bezeichnet. Sie können in Ansible sehr detailliert konfiguriert werden (vgl. Kapitel 6.3). In einer Rolle werden schließlich sämtliche Konfigurationen für das Zielsystem vordefiniert.

Als Beispiel zum besseren Verständis eines Playbooks soll folgendes Listing dienen, es wurde modifiziert von dem Testserver `shib3.srv.lrz.de` übernommen:

```
1 ---
2 - hosts: shib3.srv.lrz.de
3   roles:
4     - apache
5     - jdk
6   serial: 3
```

Ansible wird also auf dem Host `shib3.srv.lrz.de` ausgeführt. Es sollen die Rollen "apache" und "jdk" ausgeführt werden. In der Konfiguration kann z.B. für die Rolle "apache" genau angegeben werden, welche Version von Apache installiert werden soll, Virtual Hosts können vorkonfiguriert werden und vieles mehr. Ebenso kann mit der Rolle "jdk" ein Java Development Kit mit sämtlichen gewünschten Konfigurationseinstellungen installiert werden (vgl. Kapitel 6.3). Mit dem Wert "serial" kann angegeben werden, wie viele Maschinen Ansible parallel bearbeiten soll, in diesem Fall drei Systeme gleichzeitig.

6.2 Auswahlkriterien

In diesem Abschnitt wird erläutert, warum die Wahl des Automatisierungsframeworks auf Ansible fiel. Neben den bereits angedeuteten Vorteilen bietet sich Ansible insbesondere an, da bereits eine Implementierung für einen Shibboleth Identity Provider von Mitarbeitern der IDEM GARR AAI[2] existiert.

Marco Malavolti[3] und Davide Vaghetti[4] haben für die IDEM GARR AAI ein Ansible-Modul geschrieben, dessen Quellcode frei verfügbar ist und daher an die Anforderungen an diese Testföderation angepasst werden kann [MV19].

Neben dieser bedeutenden Grundvoraussetzungen bietet Ansible neben Automatisierungstools wie Puppet[5], SaltStack[6] oder Chef[7] noch weitere Vorteile, die vermutlich auch die Autoren des Ansible-Shibboleth-Moduls zu dessen Verwendung überzeugten. Grundsätzlich eignen sich diese Frameworks natürlich auch für diese Aufgabe, Ansible überzeugt aber zusätzlich aus folgenden Gründen [Bri16]:

[2] `https://www.idem.garr.it/`, aufgerufen am 19.07.2019
[3] `https://github.com/malavolti`, aufgerufen am 23.07.2019
[4] `https://github.com/daserzw`, aufgerufen am 23.07.2019
[5] `https://puppet.com/`, aufgerufen am 30.07.2019
[6] `https://www.saltstack.com/`, aufgerufen am 30.07.2019
[7] `https://www.chef.io/`, aufgerufen am 30.07.2019

1. Ansible ist “Masterless”: Ansible benötigt keinen dezidierten Server, von dem aus es seine Aufgaben erledigt. Es kann von jedem beliebigen Rechner aus gestartet werden, von welchem zu den ausgewählten Hosts SSH-Verbindungen aufgebaut werden können. Für das Testsystem ist dies von Vorteil, da man nicht unnötig Ressourcen des Testsystems (beschränkte Anzahl virtueller Maschinen) verwenden muss, sondern auf bereits vorhandenen Komponenten Ansible ausführen kann.

2. Kein “Agent” erforderlich: Puppet - als Beispiel - benötigt einen Agent, der auf den Zielsystemen im Hintergrund läuft und auf Anforderungen wartet. Ansible benötigt nur OpenSSH und Phython, was in der Regel für sämtliche Linux-Distributionen bereits bei der Grundinstallation des Betriebssystems mit installiert wird, daher ist keine zusätzliche Installation von Software notwendig. Die Nachteile eines Agents sind, dass jedes Zielsystem zuerst mit der Software ausgestattet werden muss, wodurch ein Mehraufwand nötig ist. Zudem muss zusätzliche Software stets gewartet werden und kann abstürzen, was ebenso Mehraufwand verursacht. Aber auch aus sicherheitstechnischer Sicht ergibt sich ein Nachteil, da mit jeder zusätzlich laufenden Software auf einem System auch zusätzliche Sicherheitslücken des Programms von Angreifern ausgenutzt werden kann.

3. Schnelle Einarbeitung: Aufgrund der Verwendung von YAML als leicht lesbare und schnell erlernbarer Konfigurations-Sprache, können Neuanwender sich sehr schnell in Ansible einarbeiten und die Software zeitnah verwenden.

4. Große Community: Ansible wird von einer sehr großen Community weiterentwickelt und gewartet, die Anzahl der Teilnehmer ist stetig steigend [DeK18].

6.3 Installation eines Shibboleth Identity Providers mit Ansible

In diesem Abschnitt wird beschrieben, wie ein Shibboleth IDP für die hier verwendete Testföderation automatisiert mit Ansible aufgesetzt werden kann. Für die praktische Umsetzung aller nötigen Konfigurationseinstellungen sei auf die Dokumentation in Anhang 2 verwiesen. Sie orientiert sich in ihrer Struktur an diesem Kapitel.

6.3.1 Voraussetzungen

Für die Testinstallation eines Shibboleth IDP wurde die virtuelle Maschine `shib3.srv.lrz.de` verwendet. Die Betriebssystemkonfiguration sollte wie für den manuell aufgesetzten IDP aus Kapitel 5.1 gewählt werden, da es sich hier auch um einen Shibboleth IDP handelt und die Vorrausetzungen daher gleich sind.

Aufgrund der restriktiven Konfiguration der SSH-Zugänge zu den virtuellen Maschinen des LRZ konnte Ansible nur auf dem lokalen System getestet werden. Für die Bearbeitung der virtuellen Maschinen über SSH müsste in das Sicherheitskonzept des LRZ für virtuelle Server eingegriffen werden.

In Ansible Shibboleth stehen insgesamt drei Varianten zur Installation und Konfiguration eines IDP auf einem Debian-Betriebssystem zur Auswahl (siehe Template in `/opt/ansible-shibboleth/inventories/test/test.ini-template`):

- IDP mit Identity-Management: Es wird ein Shibboleth IDP mit einem zu konfigurierenden LDAP-Server installiert.

- IDP mit Identity-Management in der GARR-Cloud[8]: Zur Nutzung der spezifischen Services der GARR-AAI Cloud Services.

- IDP ohne Identity-Management: Es wird kein LDAP-Nutzerverzeichnis installiert, da sich der IDP an ein externes Nutzerverzeichnis anbinden soll.

Für das Testsystem ist die erste Variante "IDP mit Identity-Management" auszuwählen, da in Kapitel 4 definiert wurde, dass jeder IDP des Testbeds sein eigenes LDAP-Nutzerverzeichnis besitzen soll.

Ansible Shibboleth verwendet als Servlet-Container für Java Jetty 9 statt Tomcat 8, welches im manuell aufgesetzten IDP verwendet wurde. Das Schreiben einer Tomcat-Rolle konnte aufgrund der beschränkten Zeit nicht durchgeführt werden, hier musste Jetty als Servlet-Container mit Defaulteinstellungen genügen.

6.3.2 Grundlegende Installation und Konfiguration

Für Ansible Shibboleth muss das von Marco Malavolti erstellte Github-Repository auf das System geklont werden. Zudem muss die Software für Shibboleth Version 3.4.3 auf dem System vorhanden sein.

Wie bereits beschrieben, benötigt Ansible ein Inventar. Ansible Shibboleth bietet hier eine Trennung von 3 Inventaren für ein Produktivsystem, ein Entwicklungssystem und ein Testsystem. Für das Testbed bietet sich die Verwendung des Test-Inventars an. Unter `/opt/ansible-shibboleth/inventories/test/` wird eine Datei namens `test.ini` erstellt, in welche die zu konfigurierenden Testserver eingetragen werden:

```
1 # Put here those IdP that should be provided of an Identity Management System
2 [Debian-IdP-with-IdM]
3 shib3.srv.lrz.de
```

Nachdem das Inventar angegeben ist, muss das Ansible-Playbook bearbeitet werden, damit es sämtliche Rollen abarbeiten kann. Ansible Shibboleth bietet drei vorkonfigurierte Playbooks an, je nachdem welche Variante des IDP man wählt (mit IdM, ohne IdM oder Cloud-Services). Für das Testbed soll ein eigenes LDAP-Nutzerverzeichnis installiert werden, also wird das Playbook `shib-idp-idm-servers.yml` gewählt und wie folgt bearbeitet:

```
1  ---
2  # file: shib-idp-idm-servers.yml
3  - hosts: Debian-IdP-with-IdM
4    become: yes
5    become_method: sudo
6    remote_user: debian
7    roles:
8      - common
9      - apache
10     - jdk
11     - jetty
12     - openldap
13     - mysql
14     - idp
15   serial: 3
```

[8] `https://cloud.garr.it/`, aufgerufen am 01.08.2019

Für "hosts" ist die Variable "Debian-IdP-with-IdM" angegeben, es sollen also alle IDPs mit eigenem LDAP-Nutzerverzeichnis mit den in diesem Playbook definierten Rollen versorgt werden. "Debian-IdP-with-IdM" ist hierbei eine Variable, für die die Werte des im vorigen Listing definierten `test.ini` File eingefügt werden. In diesem Fall ist es nur der Testserver `shib3.srv.lrz.de`, jedoch können hier beliebig viele Hosts eingetragen werden. Folgende Rollen sollen für einen IDP ausgeführt werden. Detailliertere Rollenbeschreibungen und Auswahlkriterien erfolgen in Abschnitt 6.3.2:

1. `common`: Hier wird festgelegt, wie das Basissystem konfiguriert werden muss, wie z.B. setzen von Passwörtern, Installation sämtlicher grundlegender Programme und Weiteres. Diese Rolle ist vergleichbar mit dem Abschnitt 5.1.2 der manuellen Installation des IDP.

2. `apache`: Ein Apache Webserver wird installiert und konfiguriert.

3. `jdk`: Installation von Java und Ausführung Java-spezifischer Konfigurationen.

4. `jetty`: Statt Tomcat wird Jetty als Servlet-Container verwendet. Für die manuelle Installation wurde Tomcat als Container gewählt. Da es aufgrund der zeitlichen Begrenztheit nicht möglich war, eine eigene Tomcat-Rolle zu schreiben, wird hier auf die Alternative Jetty zurückgegriffen, da es sich nur um eine Middleware handelt, die für die Konfiguration des IDP nicht im Vordergrund steht.

5. `openldap`: Installation und Konfiguration des LDAP-Nutzerverzeichnisses.

6. `mysql`: Einrichtung einer MySQL Datenbank.

7. `idp`: Mit dieser Rolle wird die Shibboleth IDP Software installiert und die gesamte Konfiguration sämtlicher Komponenten vorgenommen.

6.3.3 Rollen

In diesem Abschnitt wird auf die einzelnen Rollen näher eingegangen, wie sie für das Testbed angepasst werden können, damit für zukünftige Installationen kaum mehr Konfigurationsaufwand nötig ist. Mit den Rollen sollen Einstellungen vorkonfiguriert werden, die für alle zukünftigen Systeme stets gleich sein sollen. Die speziellere Konfiguration für den jeweiligen Host wird im "Inventar" vorgenommen (vgl. Abschnitt 6.3.4).

Jede Rolle (`/opt/ansible-shibboleth/roles/`) ist in Ansible in verschiedene Kategorien aufgeteilt, wodurch Einstellungen sehr detailliert und gekapselt angegeben werden können:

- `defaults`: Hier werden Default-Variablen für diese Rolle abgelegt.

- `handlers`: Handlers funktionieren im Grunde wie `tasks`, jedoch laufen sie nur, wenn sie von einem Notifier aufgeweckt werden. Das besondere ist aber, das ein Handler nur einmal pro Rolle läuft, egal wie oft er von einem Task benachrichtigt wurde.

- `meta`: Hier können Metadaten für die Rolle abgelegt werden, insbesondere wenn Rollen voneinander abhängig sein sollen.

- `tasks`: Enthält alle Aufgaben, die die Ansible-Rolle abarbeiten soll.

- `templates`: Hier werden Dateivorlagen abgelegt. Mit Hilfe von "Jinja2"-Templating können Variablen und dynamische Ausdrücke in die Vorlagen eingefügt werden.
- `vars`: In diesem Ordner werden sämtliche Variablen abgelegt, die nicht unter `default` einzuordnen sind.

Die Grundkonfiguration von Ansible Shibboleth ist für das hier verwendete Testsystem weitestgehend zu übernehmen, welche Änderungen sich jedoch zwangsweise ergeben, wird im Folgenden für die Rollen veranschaulicht, für welche die Default-Einstellungen nicht genügen. Der Großteil der gesamten Konfiguration kann in einer Datei `shib3.srv.lrz.de.yml` zusammengeführt werden, welche im Abschnitt 6.3.4 näher betrachtet wird.

Common Durch diese Rolle kann die Installation und Konfiguration des Basissystems vorgenommen werden (vgl. 5.1.2). Für diese Rolle können die Einstellungen beibehalten werden, jedoch möchte diese Rolle ein neues Root-User-Passwort setzen und den Root-Login über SSH verbieten. Dies ist nicht mit den Voraussetzungen des IDPs verträglich (vgl. Kapitel 5.1.1). Die entsprechenden Elemente müssen in `/opt/ansible-shibboleth/roles/common/tasks/main.yml` auskommentiert werden.

Apache Mit dieser Rolle wird ein Apache Webserver installiert und ein Virtual Host aufgesetzt (vgl. 5.1.5). Auch hier ist die Default-Konfiguration weitesgehend zu übernehmen. Einzig die Verwendung der richtigen Zertifikate für den Apache Virtual Host muss für die Templates in `/opt/ansible-shibboleth/roles/apache/templates/default-ssl.conf.j2` angepasst werden.

JDK In dieser Rolle kann festgelegt werden, ob das Oracle JDK oder das OpenJDK für den IDP verwendet werden soll. Hier sind alle Default-Einstellungen zu übernehmen, da bereits OpenJDK als Default-JDK verwendet wird.

IDP Für die Rolle des Shibboleth IDP müssen einige Daten angegeben werden:

- Der IDP benötigt einen gültigen Pfad zur Java Runtime Environment, andernfalls ist der IDP nicht funktionsfähig.
- Die korrekte Version muss gesetzt werden (Version 3.4.3, vgl. Kapitel 5.1).
- Für die interne MySQL Datenbank muss das Skript zur initialen Generierung und Konfiguration angepasst werden. Der Datenbankbenutzer "shibboleth" soll erzeugt werden und die nötigen Rechte zur Arbeit auf der Datenbank erhalten.
- Föderationsmetadaten werden auf dem lokalen Dateisystem hinterlegt, der Default-Wert ist in dieser Rolle auf Remote-Metadaten gesetzt. Hier müssen die Werte so geändert werden, dass die Metadaten im lokalen Dateisystem hinterlegt werden.

6.3.4 Inventar

Wie weiter oben bereits in Abschnitt 6.3.2 erwähnt, bietet das Ansible Shibboleth-Modul die Auswahl von drei verschiedenen Inventaren: Produktivsystem, Entwicklunssystem und Testsystem. Für das Testbed bietet sich daher die Verwendung des Test-Inventars an.

Ansible Shibboleth bietet einen `files`-Ordner an, in welchen vorkonfigurierte Standarddateien abgelegt werden können, die für alle drei Inventare übertragen werden sollen. Die Zielsysteme benötigen Zertifikate, weshalb an dieser Stelle das Serverzertifikat, der zugehörige Schlüssel sowie das CA-Zertifikat von shib3.srv.lrz.de unter (`/opt/ansible-shibboleth/inventories/files/shib3.srv.lrz.de/common/ssl/` abgelegt werden können, damit sie übertragen werden. Zum Test wurden unter `files/shib3.srv.lrz.de/idp/` ebenso die für den IDP aus Kapitel 5.1 erstellten Dateien `attribute-resolver.xml` und `attribute-filter.xml` abgelegt, damit sie auf das Zielsystem übertragen werden können. Für den Testfall soll der automatisiert aufgesetzte IDP im Hinblick auf Attributfreigabe und -filterung gleich dem manuell aufgesetztem IDP aus Kapitel 5 konfiguriert werden. Für weitere Testläufe können hier beliebig Dateien ergänzt und erweitert werden.

Im Test-Inventar finden sich zwei Ordner und die `test.ini`-Datei zur Angabe der Hosts. Für die beiden Ordner können jeweils Variablen hinterlegt werden, die entweder für mehrere bzw. alle Hosts gelten sollen (`group_vars`) oder sich auf spezifische Hosts beziehen sollen (`host_vars`). Ein Beispiel für `group_vars` ist die Angabe der LRZ-NTP-Server, da sie für alle Systeme stets die gleichen sein werden. Als Beispiel für einen Eintrag in den `host_vars` dient die Angabe zu Zertifikaten oder LDAP-Verbindungen, da sie für jeden Host spezifisch konfiguriert werden müssen.

Unter `host_vars` wird für jeden Zielhost eine YAML-Datei mit dem DNS-Namen des Servers angelegt. In diesem Fall die Datei `shib3.srv.lrz.de.yml`. In dieser Datei können nun sämtliche notwendige Variablen wie Zertifikatsangaben, LDAP-Angaben, MySQL-Konfigurationswerte, Typ des zu verwendenden JDK und vieles mehr definiert werden. Zum vollständigen Überblick über die Möglichkeiten sei auf das entsprechende Kapitel in Anhang 3 verwiesen.

7 Bewertung der Anforderungen an das Testbed

Zusammenfassend wird an dieser Stelle beurteilt, inwiefern die in Kapitel 4 definierten Anforderungen an das in dieser Arbeit konstruierte Testbed erfüllt werden konnten. Hierfür wird jede Anforderung aufgegriffen und darauf untersucht, ob sie erfüllt, teilweise erfüllt oder nicht erfüllt werden konnte.

Konformität mit vorhandener Umgebung Das Testbed soll mit den vom LRZ eingesetzten Systemen des Test- und Produktivsystems im Hinblick auf Betriebssystem, Identity und Access Management sowie Sicherheitsstandard kompatibel und konform sein:

1. Betriebssystem: Auf den Servern des Test- und Produktivsystems des LRZ läuft aktuell das Betriebssystem "SuSE Linux Enterprise Server" (SLES 12.4), ein Linux Betriebssystem. Für das Testbed wurde Debian 9 als Betriebssystem gewählt. So laufen die Komponenten auf beiden Systemen auf Linux-Betriebssystemen, jedoch nicht der gleichen Distribution. Insgesamt sollten daher die Unterschiede aufgrund der gleichen Basis bei der Installation und Konfiguration nicht groß sein. Zudem ermöglicht die Verwendung von Debian-Linux im Testbed den Betreibern der LRZ-Föderationskomponenten, eine Portierung oder Neuinstallation der Komponenten auf Debian zu testen. Diese Teilanforderung ist demnach teilweise erfüllt.

2. Identity und Access Management: Das Identity und Access Management des LRZ basiert auf einem LDAP-Nutzerverzeichnis. Für das Testbed wurde entschieden, ebenso ein LDAP-Nutzerverzeichnis zu verwenden. Diese Teilanforderung ist also erfüllt.

3. Sicherheitsstandards: Die virtuellen Maschinen des Testbeds wurden vom LRZ-Management installiert und nach den Sicherheitsstandards des LRZ konfiguriert. Zudem wurde bei dem Aufbau des Testbeds stets darauf geachtet, sämtliche Kommunikation verschlüsselt und mit aktuellen Protokollversionen durchzuführen. Die Teilanforderung gilt daher als erfüllt.

Insgesamt ist also zu sehen, dass die Anforderung nach Konformität großteils erfüllt wurde.

Wiederverwendbarkeit Das Ziel der Installation des Testbeds war es unter anderem, dass es für sämtliche Testszenarien wiederverwendbar eingesetzt werden kann. Als Beispiele wurden folgende Anwendungsfälle genannt:

- Erweiterung des Testbeds zu einer Interföderation
- Test neuer Protokolle
- Test verschiedener IDP Discovery-Varianten

Bei Bedarf kann das Testsystem um sämtliche IDPs und SPs erweitert werden (vgl. Kapitel 5), IDPs können sogar automatisiert aufgesetzt werden (vgl. Kapitel 6). Zum Aufbau einer Interföderation sind mindestens zwei IDPs und zwei SPs nötig, wobei jeweils ein IDP und SP eine eigene minimale Föderation eingehen. So erhält man zwei minimale Föderationen und kann eine kleine Interföderation aufbauen, um sämtliche Szenarien hierfür testen zu können.

Neuartige Protokolle können ebenso getestet werden. Als Beispiel sei hier das SAML Protokoll genannt. In den Föderationsmetadaten wird das SAML V2 Protokoll verwendet (vgl. Kapitel 5.3), jedoch können nach wie vor ältere Protokollversionen wie SAML V1.1 zu Testzwecken verwendet werden. Sollte ein neuer SAML Protokollstandard veröffentlicht werden, kann die neue Version durch Eintragung der korrekten Versionsnummer an passender Stelle getestet werden.

Insgesamt kamen für das Testbed zwei Varianten des IDP Discovery in Frage: In einer statischen Datei, in der alle IDPs aufgelistet sind oder mittels eines dynamischen IDP Discovery Services (vgl. Kapitel 2.3.4). Für das Testbed wurde vorerst die Variante einer statischen Datei umgesetzt, da sämtliche IDPs in den Föderationsmetadaten enthalten sind. Eine Erweiterung des Testbeds um einen Shibboleth IDP Discovery Service sollte problemlos möglich sein. Ein Discovery Service kann entweder unabhängig von den einzelnen IDPs und SPs zentral aufgesetzt werden oder lokal auf den jeweiligen SPs laufen. Für beide Varianten existieren im Shibboleth Wiki Lösungen, wobei für jede Variante Vor- und Nachteile vorhanden sind, die situationsabhängig bewertet werden sollten. Als Beispiel für die lokale Variante existiert der sogenannte "Embedded Discovery Service" für SPs [Kli19a]. Zusammenfassend ist diese Anforderung als erfüllt zu betrachten.

Wiederholbarkeit Es wurde gefordert, dass der Aufbau des Testbeds zu jeder Zeit wiederholbar ist, d.h. dass sämtliche Komponenten nach gleichem Muster aufgebaut werden können und so jegliches Verhalten rekonstruiert werden kann. Das Mittel hierfür ist eine detaillierte Dokumentation der Installation und Konfiguration der einzelnen Komponenten. Die Anforderung ist erfüllt, für jede Komponenten existiert eine ausführliche Dokumentation im Anhang dieser Arbeit.

Modularität Der Aufbau des Testbeds wurde modular gestaltet, da für jede Komponente (IDP und SP) eine eigene virtuelle Maschine verwendet wurde. Sind genügend Ressourcen zur Verfügung, können also beliebig viele Komponenten für jegliches Testszenario hinzugefügt werden. Für den IDP wurde sogar eine automatisierte Variante untersucht (vgl. Kapitel 6), jedoch noch nicht für einen SP. Die Anforderung gilt daher als erfüllt.

Erweiterbarkeit Das Testbed ist erweiterbar, es können bei Bedarf neue Technologien für das Shibboleth Produktivsystem am LRZ getestet werden. Dies wird an folgenden Beispielen näher diskutiert.

- Authentifizierungsansätze: Für Shibboleth gibt es sämtliche Erweiterungen, mit denen verschiedene Ansätze wie die 2-Faktor-Authentifizierung umgesetzt werden können. Als Beispiel sei hier das "Shibboleth-IdP3-TOTP-Auth"-Modul[1] genannt, welches die 2-Faktor-Authentifizierung mit dem Google Authenticator umsetzt. In Shibboleth können

[1] Siehe auf Website: https://github.com/korteke/Shibboleth-IdP3-TOTP-Auth, aufgerufen am 12.08.2019

aber auch sämtliche Alternativlösungen wie beispielsweise die Verwendung eines RADIUS-Servers hinzugefügt werden[2].

- Test SimpleSAMLphp: Beispielsweise ist die Verwendung eines SimpleSAMLphp Service Provider mit einem Shibboleth Identity Provider ist möglich, da SimpleSAMLphp mit den Shibboleth Komponenten konform ist. Der SimpleSAMLphp SP kann zur Authentifizierung an einen Shibboleth IDP delegieren. Mit SimpleSAMLphp lassen sich aber auch sämtliche andere Komponenten umsetzen [sim].
- Anbindung an verschiedene Nutzerverzeichnisse (z.B. Microsoft Active Directory): Der Shibboleth IDP des Testbeds wurde an ein LDAP-Nutzerverzeichnis angebunden, jedoch ist es laut Shibboleth Wiki möglich, als Nutzerverzeichnis ein Microsoft Active Directory anzubinden [Can18].

Wie an der Erläuterung zur Erweiterbarkeit zu sehen ist, ist diese Anforderung erfüllt.

Automatisierbarkeit Es wurde gefordert, dass das Testbed möglichst automatisierbar aufgesetzt werden kann und ein geeignetes Framework ausgewählt wird. Wie in Kapitel 6 ersichtlich wird, bietet sich für diese Aufgabe das Framework Ansible an. Für den Aufbau eines IDP wurde Ansible im Rahmen dieser Arbeit erfolgreich konfiguriert, jedoch noch nicht für einen SP. Die Anforderung konnte daher nur teilweise erfüllt werden.

Vertraulichkeit und Integrität Die Kommunikation innerhalb des Testbeds sollte nach Möglichkeit nur verschlüsselt erfolgen. Es sollten nur vertrauenswürdige Zertifikate verwendet werden, sofern dies möglich war. Jede virtuelle Maschine des Testbeds hat ein vertrauenswürdiges Zertifikat - unterschrieben von einer vertrauenswürdigen CA - erhalten (vgl. Kapitel 5). Es wurde konsequent zur Absicherung der Kommunikation der Komponenten über HTTPS untereinander verwendet. Zusätzlich wurde für den IDP die verschlüsselte Variante LDAPS umgesetzt, obwohl der LDAP-Server innerhalb des IDP läuft. Da aber in der Produktivumgebung der Anschluss an einen externen LDAP-Server gefordert ist, muss die Kommunikation dort verschlüsselt erfolgen. Jedoch konnte hierfür kein Zertifikat einer vertrauenswürdigen Zertifizierungsstelle erworben werden, es wurde ein selbstsigniertes Zertifikat verwendet. Lediglich die Kommunikation des IDP mit der internen MySQL-Datenbank ist nicht verschlüsselt konfiguriert, was aber nachgeholt werden kann. Da die Kommunikation mit der Datenbank nur intern abläuft, wurde hier auf die Umsetzung einer SSL-Kommunikation verzichtet. Wie gezeigt wurde, ist sämtliche Kommunikation der Komponenten über das Netzwerk durch Zertifikate abgesichert, die Anforderung ist also erfüllt.

Interföderation Wie im Abschnitt “Wiederverwendbarkeit” bereits diskutiert wurde, ist die Erweiterungsmöglichkeit zu einer Interföderation bereits gegeben. Die Anforderung gilt demnach als erfüllt.

Architektur Das Testbed wurde in der Architektur einer “Full Mesh”-Architektur umgesetzt (vgl. Kapitel 4.2). Um andere Architekturen auf Funktion und Effizienz testen zu können (“Hub-and-Spoke with Distributed Login” sowie “Hub-and-Spoke with Centralized

[2]Für Details sei hierzu auf das Shibboleth-Wiki verwiesen: https://wiki.shibboleth.net/confluence/display-/IDP30/Contributions+and+Extensions, aufgerufen am 12.08.2019

Login", vgl. Kapitel 2.3.6), soll das Testbed diese Architekturen umsetzen können. Die Architektur "Hub-and-Spoke with Centralized Login" ist bereits durch die minimale Föderation (vgl. Kapitel 4 und 5) umgesetzt, da es sich hierbei um lediglich einen IDP für alle SPs handelt. Anders ist es für die "Hub-and-Spoke with Distributed Login"-Architektur. Für diese Lösung müsste der "Hub", der als zentraler IDP und SP für alle Teilnehmer der Föderation gilt, eventuell selbst implementiert werden. Zum Zeitpunkt der Erstellung dieser Arbeit ist keine Lösung bekannt. Insgesamt ist diese Anforderung teilweise erfüllt.

Portabilität Die Shibboleth-Komponenten des Testbeds sind Plattform- und Betriebssystemunabhängig. Ein Shibboleth IDP sowie SP kann ebenso auf einem Microsoft Windows Betriebssystem implementiert werden, wie auf gängigen Linux-Distributionen[3]. Da die Systeme des LRZ hauptsächlich auf Windows und Linux-Betriebssystemen basieren, ist diese Anforderung umgesetzt.

Metadaten Metadaten werden im Testbed anhand der Föderationsmetadaten bereitgestellt (vgl. Kapitel 5.3). Aufgrund der unabdingbaren Aktualisierung und Konsistenzhaltung der Metadaten ist die Bereitstellung über einen frei für alle Teilnehmer der Föderation zugänglichen Webserver zu bevorzugen. Im Testbed konnte aufgrund unauflösbarer Fehlerquellen die Bereitstellung über einen zentralen Webserver nicht umgesetzt werden, die Metadaten werden auf den lokalen Systemen gespeichert und müssen manuell aktualisiert werden. Die Anforderung ist daher nur teilweise erfüllt.

Wie an den bisherigen Ausführungen und an der Übersicht in der nachfolgenden Tabelle zu sehen ist, konnte der Großteil der Anforderungen an das Testbed erfüllt werden.

Anforderung	**Erfüllt**	**Teilweise erfüllt**	**Nicht erfüllt**
Konformität mit vorhandener Umgebung	✓		
Wiederverwendbarkeit	✓		
Wiederholbarkeit	✓		
Modularität	✓		
Erweiterbarkeit	✓		
Automatisierbarkeit		✓	
Vertraulichkeit und Integrität	✓		
Interföderation	✓		
Architektur		✓	
Portabilität	✓		
Metadaten		✓	

[3] https://wiki.shibboleth.net/confluence/display/SP3/Installation, aufgerufen am 12.08.2019

8 Zusammenfassung und Ausblick

In dieser Arbeit wurde erfolgreich eine Shibboleth-Testumgebung für föderiertes Identitätsmanagement erstellt. Es können Föderationen gebildet werden und nach Belieben weitere Komponenten wie IDPs und SPs dem Testbed hinzugefügt werden. Durch den eigenständigen Aufbau der Föderation und deren Komponenten steht dem LRZ nun eine neue Testumgebung zur Verfügung, welche ausführlich dokumentiert und für welche teilweise sogar automatisiert neue Komponenten aufgesetzt werden können. Auch wenn nicht alle Funktionen von SAML und Shibboleth umgesetzt wurden, so bietet das Testsystem die Möglichkeit, einen SSO-Vorgang zu simulieren und dementsprechend neue Protokolle, Authentifizierungsverfahren oder andere Technologien zu testen. Noch nicht implementierte Komponenten können problemlos ergänzt werden.

Obwohl der Standard SAML in der Welt der IT als alt gilt (2005), so ist Shibboleth immer noch im wissenschaftlichen und universitären Umfeld ein geeignetes Tool, eine Föderation zu verwalten und geschützte Ressourcen wie z.B. Online-Bibliotheken den Teilnehmern der Föderation zugänglich zu machen, wodurch vor allem die Wissenschaft nach wie vor profitiert. Durch den Aufbau des Testbeds kann hoffentlich ein Beitrag geleistet werden, sodass die föderierten Dienste des LRZ beständig verbessert werden können.

Anhand der gewonnenen Ergebnisse sowie den bei der Arbeit entstandenen Herausforderungen ergeben sich aber auch einige Punkte, an denen in einer zukünftigen Arbeit weiter geforscht werden kann. So wäre es wünschenswert, wenn die Föderation einen eigenständigen Shibboleth Discovery Service implementieren würde (vgl. Kapitel 2.3.4), wodurch das Problem der Zuordnung der Benutzer zu ihren IDPs elegant gelöst werden kann. Ebenso könnte sich ein Mechanismus überlegt werden, wie die Föderationsmetadaten, die jeder Teilnehmer der Föderation in einer stets aktuellen Version benötigt, automatisiert von einer frei zugänglichen Stelle wie einem Webserver angeboten werden können. Der Idealfall für eine Föderation wäre, dass der Anbieter der Föderation in einem Push-Verfahren in regelmäßigen Abständen die aktuelle Version der Metadaten auf die teilnehmenden Komponenten schiebt.

Im Hinblick auf die Automatisierungsmöglichkeiten der Installation und Konfiguration von Komponenten wurde für den Shibboleth IDP ausführlich eingegangen. Für einen Shibboleth SP existiert noch kein Ansible-Modul, womit automatisiert neue SPs aufgesetzt werden können. Hier wäre es nötig, im selben Stil des Ansible Shibboleth IDP-Moduls ein entsprechendes SP-Modul zu implementieren.

Ein weiterer interessanter Aspekt ist der Aufbau einer Interföderation. Bisher wurde nur eine einzige Föderation betrachtet. Was passiert jedoch, sobald mehrere Föderationen zusammengeschlossen werden? Wie interagieren diese? Welche Komponenten und Protokolle sind nötig? Wie funktioniert föderationsübergreifendes SSO? Besonders da die Föderation des LRZ an einer Interföderation - eduGAIN - teilnimmt, wäre eine Erweiterung der Testumgebung durchaus interessant und auch relevant.

Abbildungsverzeichnis

Literaturverzeichnis

[AAR16] AARC - Authentication and Authorisation for Research and Collaboration (Hrsg.): *Shibboleth Service Provider Hands-on Training*. 2016

[ans19] *Overview - How Ansible works*. https://www.ansible.com/overview/how-ansible-works. Version: 2019. – abgerufen am 29. Juli 2019

[Aug18] Augsten, Stephan: *Was ist Ansible?* https://www.dev-insider.de/was-ist-ansible-a-724411/. Version: 2018. – abgerufen am 29. Juli 2019

[Bha19] Bhatia, Sagar: *PostgreSQL vs. MySQL: Everything You Need to Know*. https://hackr.io/blog/postgresql-vs-mysql. Version: 2019. – abgerufen am 24. Juli 2019

[Bri16] Brikman, Yevgeniy: *Why we use Terraform and not Chef, Puppet, Ansible, SaltStack, or CloudFormation*. https://blog.gruntwork.io/why-we-use-terraform-and-not-chef-puppet-ansible-saltstack-or-cloud\protect\discretionary{\char\hyphenchar\font}{}{}-formation-7989dad2865c#d105. Version: 2016. – abgerufen am 30. Juli 2019

[Can05a] Cantor, Scott; Hirsch, Frederick; Kemp, John; Philpott, Rob and Maler, Eve (Hrsg.): *Bindings for the OASIS Security Assertion Markup Language (SAML) V2.0*. OASIS Security Services Technical Committee Standard, März 2005

[Can05b] Cantor, Scott (Hrsg.): *Shibboleth Architecture - Protocols and Profiles*. September 2005

[Can05c] Cantor, Scott; Kemp, John; Philpott, Rob and Maler, Eve (Hrsg.): *Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0*. OASIS Security Services Technical Committee Standard, März 2005

[Can18] Cantor, Scott: *LDAPAuthnConfiguration*. https://wiki.shibboleth.net/confluence/display/IDP30/LDAPAuthnConfiguration. Version: 2018. – abgerufen am 10. August 2019

[Cha09] Chadwick, David W.: Federated Identity Management. In: *Foundations of Security Analysis and Design* 5 (2009), S. 96–120

[DeK18] DeKoenigsberg, Greg: *ANSIBLE COMMUNITY - 2017 YEAR IN REVIEW*. https://www.ansible.com/blog/2017-community-year-in-review. Version: 2018. – abgerufen am 12. Juli 2019

[DFN19] DFN (Hrsg.): *Server-Side-Storage, Sessions, User Consent und Persistent Identifier*. `https://doku.tid.dfn.de/de:shibidp3storage`. Version: 2019. – abgerufen am 24. Juli 2019

[Ebe06] Ebert, Matthias: *Konzeption und Implementierung einer policy-basierten Privacy Management Architektur für föderierte Identitätsmanagementsysteme am Beispiel Shibboleth*. Diplomarbeit an der Ludwig-Maximilians-Universität München, Institut für Informatik, 2006

[Fis12] Fisher, Dennis: *New Attack Uses SSL/TLS Information Leak to Hijack HTTPS Sessions*. `https://threatpost.com/new-attack-uses-ssltls-information-leak-hijack-https-sessions-090512/76973/`. Version: 2012. – abgerufen am 24. Juli 2019

[FP14] Ferdous, Sadek ; Poet, Ron: Managing Dynamic Identity Federations using Security Assertion Markup Language. In: *Journal of Theoretical and Applied Electronic Commerce Research* 10 (2014), Nr. 2, S. 53–67

[Hei14] Heinlein, Peer: *HSTS: Header Strict Transport Security für Webserver einrichten*. `https://www.heinlein-support.de/blog/security/hsts-header-strict-transport-security-fuer-webserver-einrichten/`. Version: 2014. – abgerufen am 24. Juli 2019

[Hom07] Hommel, Wolfgang: *Architektur- und Werkzeugkonzepte für föderiertes Identitäts-Management*. Dissertation, Ludwig-Maximilians-Universität München, 2007

[itu10] *Recommendation ITU-T X.1250 - Baseline capabilities for enhanced global identity management and interoperability*. 2010

[Kli19a] Klingenstein, Nate: *IdPDiscovery*. `https://wiki.shibboleth.net/confluence/display/CONCEPT/IdPDiscovery`. Version: 2019. – abgerufen am 10. August 2019

[Kli19b] Klingenstein, Nate: *FlowsAndConfig*. `https://wiki.shibboleth.net/confluence/display/CONCEPT/FlowsAndConfig`. Version: 2019. – abgerufen am 25. Juni 2019

[LCD18] Li, Mengyi ; Chi, Chi-Hung ; Ding, Cheng et a.: *A Multi-Protocol Authentication Shibboleth Framework and Implementation for Identity Federation*. 2018

[Mad05] Madsen, Paul and Maler, Eve (Hrsg.): *SAML V2.0 Executive Overview*. OASIS Security Services Technical Committee Draft, April 2005

[Mey18] Meyer, Silke: *Einführung in die DFN-AAI*. 2018

[MV19] Malavolti, Marco ; Vaghetti, Davide: *Ansible Code to Install Shibboleth Products*. `https://github.com/malavolti/ansible-shibboleth`. Version: 2019. – abgerufen am 12. Juni 2019

[nre18] *Membership Map*. `https://www.geant.org/About/Membership/Pages/Membership_map.aspx`. Version: 2018. – abgerufen am 23. Juli 2019

[nre19] *NRENs*. https://www.geant.org/About/NRENs. Version: 2019. – abgerufen am 23. Juli 2019

[oas19] *Organization for the Advancement of Structured Information Standards*. https://www.oasis-open.org/org. Version: 2019. – abgerufen am 6. Juni 2019

[Pem19] Pempe, Wolfgang: *Attribute*. 2019

[Rag08] Ragouzis, Nick; Hughes, John; Philpott, Rob; Maler, Eve; Madsen, Paul and Scavo, Tom (Hrsg.): *Security Assertion Markup Language (SAML) V2.0 Technical Overview*. OASIS Security Services Technical Committee Draft, März 2008

[Shia] Shibboleth Consortium (Hrsg.): *How Shibboleth Works: Basic Concepts*. https://www.shibboleth.net/index/basic/. – abgerufen am 20. Juni 2019

[Shib] Shibboleth Consortium (Hrsg.): *What's Shibboleth?* https://www.shibboleth.net/index/. – abgerufen am 20. Juni 2019

[shi19] *SystemRequrements*. https://wiki.shibboleth.net/confluence/display/IDP30/SystemRequirements. Version: 2019. – abgerufen am 23. Juli 2019

[sim] *SimpleSAMLphp as a Service Provider (SP)*. https://simplesamlphp.org/samlsp. – abgerufen am 10. August 2019

[SWI19] SWITCH (Hrsg.): *Using the Persistent ID as a user identifier attribute*. https://www.switch.ch/aai/guides/sp/persistentid/. Version: 2019. – abgerufen am 25. Juli 2019

[TG17] Torroglosa-Garcia, Elena: *Digital Identity Management Through the Interoperability of Heterogeneous Authentication and Authorization Infrastructures*. Dissertation, Universität Murica, 2017

[Wid08] Widdowson, Rod and Cantor, Scott (Hrsg.): *Identity Provider Discovery Service Protocol and Profile*. OASIS Security Services Technical Committee Standard, März 2008

[wik17] *Identity Federations and eduGAIN*. https://wiki.geant.org/display/eduGAIN/Identity+Federations+and+eduGAIN. Version: 2017. – abgerufen am 2. Mai 2019

[wik19] *Documentation*. http://tomcat.apache.org/tomcat-8.5-doc/index.html. Version: 2019. – abgerufen am 23. Juli 2019

Anhang 1: Installation und Konfiguration eines Shibboleth Identity Providers

Folgende Dokumentation erläutert, wie ein Shibboleth IDP für das Testbed am LRZ aufgesetzt werden kann.

Im Folgenden wird exemplarisch beschrieben, wie ein Shibboleth Identity Provider für die Test-Föderation installiert und konfiguriert werden kann. Generell orientiert sich diese Anleitung an folgenden Dokumentationen:

- https://wiki.shibboleth.net/confluence/display/IDP30/Installation
- https://www.switch.ch/aai/guides/idp/installation/

1 Voraussetzungen

Bevor mit einer Installation begonnen werden kann, müssen einige Voraussetzungen bestehen:

- Betriebssystem: eine virtuelle Maschine mit einem Linux Betriebssystem, in diesem Fall Debian 9. Für die Hardware sollten 4 GB RAM und 20 GB Festplattenspeicher eingeplant werden.
- Ein Zugang über SSH mit einem Benutzer mit Root-Rechten muss vorhanden sein
- Netzwerk:
  - Es wird ein Hostname (shib1.srv.lrz.de) und eine statische IP-Adresse benötigt (129.187.255.171).
  - Die Firewall sollte für SSL-Verbindungen, über welche externe Komponenten auf den IDP zugreifen, die Ports 443 und 8443 geöffnet haben. Diese Verbindungen sollten direkt möglich sein, ein Proxy ist nicht möglich.
  - Falls ein externe NTP-Server verwendet wird, sollte dieser ebenso an der Firewall freigeschaltet werden.
- Zertifikat: Für den Hostnamen (shib1.srv.lrz.de) sollte ein Zertifikat bereitgestellt werden, dessen CA die gängigen Browser vertrauen. Für den Server wurde aus der LRZ-internen PKI ein Zertifikat beantragt. Sie werden unter /etc/ssl/localcerts bzw. /etc/ssl/localkeys abgelegt. Das Zertifikat für die CA liegt unter /etc/ssl/chains/.
- LDAP: In dieser Implementierung wird ein eigener LDAP-Server eingerichtet. Wird ein externer Server verwendet, muss dieser an der Firewall freigeschaltet werden.

2 Installation und Konfiguration von Basiskomponenten

Bevor mit der Installation des Shibboleth IDP begonnen werden kann, müssen einige Programme nachinstalliert werden. Der Shibboleth IDP benötigt Tomcat als Servlet-Container. Für Java wird die Version 8 verwendet.

```
apt-get install curl net-tools apache2 tomcat8 default-jdk gnutls-bin strace
```

Vor der Installation sollten außerdem einige Umgebungsvariablen gesetzt werden, vor allem die JAVA_HOME Variable, da sie von Tomcat zwingend benötigt wird. Hierfür die Variablen in `/etc/environment` eintragen: In `/etc/environment` folgende Variablen nachtragen:

```
IDP_HOME=/opt/shibboleth-idp
JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64/jre
IDP_SRC=/usr/local/src/shibboleth-identity-provider-3.4.3
```

Damit die Umgebungsvariablen gültig werden, werden sie jetzt noch gesourced:

```
source /etc/environment
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64/jre
export IDP_SRC=/usr/local/src/shibboleth-identity-provider-3.4.3
```

In `/etc/default/tomcat8` können folgende Variablen angegeben bzw. verändert werden. Falls die JAVA_HOME Variable nicht gesetzt wurde, kann dies auch hier für Tomcat nachgeholt werden. Die Variable JAVA_OPTS kann mit der Option -Xmx2g der JVM Speicherplatz zuweisen, falls dieser standardmäßig unter der Mindestgröße von 2 GB liegt:

```
JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64/jre

JAVA_OPTS="-Djava.awt.headless=true -XX:+DisableExplicitGC -XX:+
    UseParallelOldGC -Xms256m -Xmx2g -Djava.security.egd=file:/dev/./urandom"
```

2.1 Grundlegende Installation des Shibboleth Identity Providers

Die aktuellste Version des Shibboleth IDP kann unter `https://shibboleth.net/downloads/identity-provider/` heruntergeladen werden. Zum Zeitpunkt der Erstellung dieser Arbeit war dies Version 3.4.3. Mit folgenden Kommandos wird Installationsdatei in `/usr/local/src/` heruntergeladen und dort entpackt:

```
cd /usr/local/src
curl -O https://shibboleth.net/downloads/identity-provider/3.4.3/shibboleth-
    identity-provider-3.4.3.tar.gz
tar -zxf shibboleth-identity-provider-3.4.3.tar.gz
```

Jetzt das Install-Skript aktivieren:

```
cd shibboleth-identity-provider-3.4.3
./bin/install.sh
```

Bei der Installation des IDP werden folgende Fragen gestellt:

- Source Directory: aktuellen Ordner bestätigen
- Installation directory: bestätige /opt/shibboleth-idp
- Hostname: shib1.srv.lrz.de
- SAML Entity-ID: https://shib1.srv.lrz.de/idp/shibboleth

- Attribute Scope: localdomain
- Backchannel PKCS12 Password: Passwort
- Cookie encryption password: Passwort

2.2 Konfiguration Tomcat8

Damit Tomcat für den IDP entsprechend funktioniert, sind einige Einstellungen zu ändern.

1. Das IDP WAR File `/etc/tomcat8/Catalina/localhost/idp.xml` generieren:

```
<Context docBase="/opt/shibboleth-idp/war/idp.war"
         privileged="true"
         antiResourceLocking="false"
         unpackWAR=?false?
         swallowOutput="true"/>
```

2. Zur Visualisierung der IdP-Statuspage für Diagnoseinformationen benötigt Tomcat die JavaServer Pages Standard Tag Library (JSTL). Sie wird in den IDP-Home-Ordner `/opt/shibboleth-idp/edit-webapp/WEB-INF/lib` heruntergeladen. Anschließend wird das IDP WAR File neu gebaut:

```
cd /opt/shibboleth-idp/edit-webapp/WEB-INF/lib
sudo curl -O https://repo1.maven.org/maven2/jstl/jstl/1.2/jstl-1.2.jar
# Rebuild idp.war
cd /opt/shibboleth-idp/bin ; ./build.sh -Didp.target.dir=/opt/shibboleth-
    idp
```

3. Der Tomcat-User benötigt Lese-Rechte auf bestimmten Ordnern im IDP, um korrekt arbeiten zu können. Hierfür folgende Kommandos ausführen:

```
chown -R tomcat8 /opt/shibboleth-idp/logs/
chown -R tomcat8 /opt/shibboleth-idp/metadata/
chown -R tomcat8 /opt/shibboleth-idp/credentials/
chown -R tomcat8 /opt/shibboleth-idp/conf/
```

4. In der Datei /etc/tomcat8/server.xml können die Konnektoren für Tomcat festgelegt werden. Es wird ein AJP Konnektor auf Port 8009 benötigt. Der bestehende HTTP Konnektor auf Port 8080 wird nicht benötigt und wird auskommentiert:

```
<!-- A "Connector" represents an endpoint by which requests are received
     and responses are returned. Documentation at :
     Java HTTP Connector: /docs/config/http.html (blocking & non-blocking
         )
     Java AJP  Connector: /docs/config/ajp.html
     APR (HTTP/AJP) Connector: /docs/apr.html
     Define a non-SSL/TLS HTTP/1.1 Connector on port 8080
-->
<!--
        <Connector port="8080" protocol="HTTP/1.1"
            connectionTimeout="20000"
            URIEncoding="UTF-8"
            redirectPort="8443" />
-->
```

```
14
15 [...]
16
17 <!-- Define an AJP 1.3 Connector on port 8009 -->
18 <Connector port="8009" protocol="AJP/1.3" redirectPort="443" address="
       127.0.0.1" enableLookups="false" tomcatAuthentication="false"/>
```

5. Um Session Persistence zu verhindern, muss in `/etc/tomcat8/context.xml` der Kommentar in folgender Zeile gelöscht werden. Der leere Wert der Variable ist hierbei entscheidend:

```
1 <Manager \pathname="" />
```

6. Damit die Änderungen wirksam werden, muss Tomcat neu gestartet werden:

```
1 systemctl restart tomcat8
```

2.3 Konfiguration von SSL für Apache mit Tomcat Frontend

1. In Apache muss ein Virtual Host konfiguriert werden, worüber die Website gehostet werden kann. Hierfür wurde die Datei `/etc/apache2/sites-available/shib1.srv.lrz.de.conf` erstellt und folgendes eingefügt:

```
1  <VirtualHost *:443>
2    ServerName shib1.srv.lrz.de:443
3    ServerAdmin idp-admin@shib1.srv.lrz.de
4    # Explizites Logging an diesen Ort aktivieren
5    CustomLog /var/log/apache2/shib1.srv.lrz.de.access.log combined
6    ErrorLog /var/log/apache2/shib1.srv.lrz.de.error.log
7    DocumentRoot /var/www/html
8
9    SSLEngine On
10   # Kein SSLv2, SSLv3 und TLSv1
11   SSLProtocol all -SSLv2 -SSLv3 -TLSv1
12
13   # Erlaubte und nicht erlaubte Verschlüsselungschiffren
14   SSLCipherSuite "kEDH+AESGCM:ECDHE-RSA-CHACHA20-POLY1305:ECDHE-RSA-
         AES256-SHA384:ECDHE-RSA-AES256-SHA256:ECDHE-RSA-AES256-SHA:ECDHE-RSA
         -AES256-GCMSHA384:ECDHE-RSA-AES256-GCM-SHA256:ECDHE-ECDSA-CHACHA20-
         POLY1305:ECDHE-ECDSAAES256-SHA384:ECDHE-ECDSA-AES256-SHA256:ECDHE-
         ECDSA-AES256-SHA:ECDHE-ECDSAAES256-GCM-SHA384:ECDHE-ECDSA-AES256-GCM
         -SHA256:AES256-GCM-SHA384:!3DES:!DES:!DHE-RSA-AES128-GCM-SHA256:!DHE
         -RSA-AES256-SHA:!EDE3:!EDH-DSS-CBC-SHA:!EDH-DSSDES-CBC3-SHA:!EDH-RSA
         -DES-CBC-SHA:!EDH-RSA-DES-CBC3-SHA:!EXP-EDH-DSS-DES-CBCSHA:!EXP-EDH-
         RSA-DES-CBC-SHA:!EXPORT:!MD5:!PSK:!RC4-SHA:!aNULL:!eNULL"
15
16   # Die Verschlüsselungsmethode vom Server wird bevorzugt
17   SSLHonorCipherOrder on
18
19   # SSL Compression deaktivieren
20   SSLCompression Off
21   # HSTS aktivieren für 2 Jahre
22   Header always set Strict-Transport-Security "max-age=63072000;
         includeSubDomains"
23
```

```
24    # Zertifikate
25    SSLCertificateFile /etc/ssl/localcerts/shib1.srv.lrz.de.pem
26    SSLCertificateKeyFile /etc/ssl/localkeys/shib1.srv.lrz.de-nopw.pem
27    SSLCertificateChainFile /etc/ssl/chains/lrz-cachain-telesec-class2.pem
28
29  # Aktiviere AJP Konnektoren
30  <Proxy ajp://localhost:8009>
31    Require all granted
32  </Proxy>
33
34  ProxyPass /idp ajp://localhost:8009/idp retry=5
35  ProxyPassReverse /idp ajp://localhost:8009/idp retry=5
36  </VirtualHost>
```

In der Konfigurationsdatei für den Virtual Host wurden folgende Optionen gewählt:

- SSLProtocol: SSLv2, SSLv3 und TLSv1 sind für Verschlüsselung nicht erlaubt
- SSLCipherSuit: hier wird definiert, welche Verschlüsselungsalgorithmen akzeptiert werden und welche nicht (gekennzeichnet mit "!" vor Algorithmus)
- SSLHonorCipherOrder: Beim Verbindungsaufbau soll die Verschlüsselungsmethode des Servers bevorzugt verwendet werden, nicht die des Clients
- SSLCompression wird aus Sicherheitsgründen deaktiviert
- Die AJP Konnektoren der Tomcat Konfiguration aus Abschnitt 5.2 werden hier aktiviert

2. Aktivieren der Website: Mit folgenden Kommandos wird die Website aktiviert und die Default-Website deaktiviert. Mit dem "configtest" wird die Konfiguration des Virtual Host überprüft. Schließlich muss Apache neu gestartet werden.

```
1  a2enmod ssl headers
2  a2enmod proxy_ajp
3  a2ensite shib1.srv.lrz.de.conf
4  a2dissite 000-default.conf
5  apachectl configtest
6  systemctl restart apache2
```

3. Aus Sicherheitsgründen sollte Port 80 nur für Localhost geöffnet sein. Dies wird erreicht, indem in `/etc/apache2/ports.conf` hinter die Adresse des Localhost der Port 80 gesetzt wird:

```
1  # If you just change the port or add more ports here, you will likely
       also
2  # have to change the VirtualHost statement in
3  # /etc/apache2/sites-enabled/000-default.conf
4
5  Listen 127.0.0.1:80
6
7  <IfModule ssl_module>
8    Listen 443
9  </IfModule>
10
11 <IfModule mod_gnutls.c>
12   Listen 443
13 </IfModule>
```

4. Damit über den Webserver keine Betriebssysteminformationen gelesen werden können, sollten in `/etc/apache2/conf-enabled/security.conf` folgende Werte geändert werden:

```
[...]
ServerTokens prod
[...]
ServerSignature off
```

Jetzt kann ein kleiner Test gestartet werden. Hierfür die Website via

```
wget --no-check-certificate https://localhost/idp/shibboleth
```

aufrufen. Wurde alles erfolgreich konfiguriert, sollte eine Datei namens "shibboleth" heruntergeladen werden. Sie enthält die Metadaten des IDP.

2.4 LDAP

Damit das System mit beliebigen Testusern mit verschiedenen Attributen getestet werden kann bzw. welche Attribute aus einem Identity und Access Management die IDPs und SPs benötigen, wird dem IDP ein eigenes Identity und Access Management installiert. Hierfür wird prototypisch für den IDP ein eigener einfacher LDAP-Server installiert und mit einem Testuser "shibbolethuser" versehen. Zusätzlich soll die Kommunikation verschlüsselt erfolgen, weshalb LDAPS verwendet wird.

Installation LDAP Server

Der LDAP-Server wird mit folgenden Kommandozeilenbefehlen installiert:

```
apt-get install slapd ldap-utils
```

Der Installationsassistent fragt nach einem Admin-Passwort: Passwort DNS-Name und Organisationsnamen werden bei der Installation aus dem Hostnamen abgeleitet.

Nach Beendigung der Installation kann die aktuelle Konfiguration über das Kommando slapcat abgerufen werden:

```
dn: dc=srv,dc=lrz,dc=de
objectClass: top
objectClass: dcObject
objectClass: organization
o: srv.lrz.de
dc: srv
structuralObjectClass: organization
entryUUID: 33401322-f0ab-1038-85d5-073f8c4aa26b
creatorsName: cn=admin,dc=srv,dc=lrz,dc=de
createTimestamp: 20190411134106Z
entryCSN: 20190411134106.070731Z#000000#000#000000
modifiersName: cn=admin,dc=srv,dc=lrz,dc=de
modifyTimestamp: 20190411134106Z

dn: cn=admin,dc=srv,dc=lrz,dc=de
objectClass: simpleSecurityObject
objectClass: organizationalRole
cn: admin
description: LDAP administrator
userPassword:: e1NTSEF9T0l3eEhUK0IzUlppNGhJSUtaYk4wU3V6cmVNbUpDNkk=
```

```
structuralObjectClass: organizationalRole
entryUUID: 33407164-f0ab-1038-85d6-073f8c4aa26b
creatorsName: cn=admin,dc=srv,dc=lrz,dc=de
createTimestamp: 20190411134106Z
entryCSN: 20190411134106.073189Z#000000#000#000000
modifiersName: cn=admin,dc=srv,dc=lrz,dc=de
modifyTimestamp: 20190411134106Z
```

Dem Output zu Folge ist der LDAP-Server aktuell wie folgt konfiguriert:

- Base-DN: dc=srv,dc=lrz,dc=de
- Organization Name: srv.lrz.de
- Admin Base-DN: dn: cn=admin,dc=srv,dc=lrz,dc=de

Im Falle von Verbindungsproblemen mit dem LDAP-Server, kann in der Datei `/etc/hosts.allow` folgende Zeile eingetragen werden. Wichtig ist, dass der Eintrag vor `ALL:ALL:rfc931:DENY` stattfindet.

```
slapd:127.0.0.1,[::1],localhost,shib1.srv.lrz.de:ALLOW
```

Mit dem Kommando

```
ldapwhoami -H ldap:// -x
```

kann die Konnektivität zum LDAP-Server getestet werden. Ist der Output anonymus, so nimmt der Server Anfragen entgegen. Anonymus deshalb, da sich noch nicht am LDAP-Server eingeloggt wurde.

Base-DN für Benutzer und Gruppen

Bisher wurde nur der Base-DN für den Administrator angelegt. Für das Management von Usern müssen noch User und Gruppen Base-DNs angelegt werden. Hierfür wird ein "LDAP interchange formate file" im `/root/` Ordner mit folgendem Inhalt erzeugt:

```
dn: ou=people,dc=srv,dc=lrz,dc=de
objectClass: organizationalUnit
ou: people

dn: ou=groups,dc=srv,dc=lrz,dc=de
objectClass: organizationalUnit
ou: groups
```

Um den Eintrag im LDAP zu hinterlegen, wird folgendes Kommando mit dem LDAP-Admin Passwort ausgeführt:

```
ldapadd -x -D cn=admin,dc=srv,dc=lrz,dc=de -W -f user_group_base.ldif
```

User Accounts hinzufügen

Es wird ein Useraccount "shibbolethuser" mittels einer weiteren LDIF Datei `new_user.ldif` erzeugt. Als Vorname und Nachname werden Hans Meier gewählt. Zuerst muss ein Passwort mit dem Befehl `slappasswd` generiert werden:

```
slappasswd
New password: <Passwort>
Re-enter new password: <Passwort>
{SSHA}<Hash des Passworts>
```

In die Datei `new_user.ldif` wird folgender Inhalt eingetragen. Unter `userPassword` muss der Hash des Passworts eingetragen werden.

```
dn: uid=shibbolethuser,ou=people,dc=srv,dc=lrz,dc=de
objectClass: inetOrgPerson
objectClass: posixAccount
objectClass: shadowAccount
uid: shibbolethuser
cn: shibbolethuser
givenName: Hans
sn: Meier
userPassword: {SSHA}<Hier Passwort Hash eintragen>
loginShell: /bin/bash
uidNumber: 10000
gidNumber: 10000
homeDirectory: /home/shibbolethuser
shadowMax: 60
shadowMin: 1
shadowWarning: 7
shadowInactive: 7
shadowLastChange: 0

dn: cn=shibbolethuser,ou=groups,dc=srv,dc=lrz,dc=de
objectClass: posixGroup
cn: shibbolethuser
gidNumber: 0
memberUid: shibbolethuser
```

Konfiguration LDAPS

Um LDAP abschließend zu konfigurieren, kann in `/etc/ldap/ldap.conf` festgelegt werden, unter welcher Adresse der Server zu erreichen ist und welche Zertifikate im Falle von TLS verwendet werden sollen. In unserer Testumgebung wird LDAPS auf dem Localhost implementiert, weshalb die URI `ldaps:/127.0.0.1:636` festgesetzt wird. Als CA-Zertifikat für TLS wird die LRZ-CA-Chain aus dem Verzeichnis `/etc/ssl/chains/` angegeben.

```
#
# LDAP Defaults
#

# See ldap.conf(5) for details
# This file should be world readable but not world writable.

BASE     dc=srv,dc=lrz,dc=de
URI      ldaps://127.0.0.1:636

#SIZELIMIT      12
#TIMELIMIT      15
#DEREF          never

# TLS certificates (needed for GnuTLS)
TLS_CACERT      /etc/ssl/chains/lrz-cachain-telesec-class2.pem
```

```
17 TLS_REQCERT allow
```

Abschließend muss in `/etc/ldap/` die Datei `ldaps.ldif` mit folgendem Inhalt erstellt werden. In dieser Datei wird LDAP mitgeteilt, dass es die vom LRZ erhaltenen Zertifikate und Schlüssel verwenden soll:

```
1 dn: cn=config
2 add: olcTLSCACertificateFile
3 olcTLSCACertificateFile: /etc/ssl/chains/lrz-cachain-telesec-class2.pem
4 -
5 add: olcTLSCertificateFile
6 olcTLSCertificateFile: /etc/ssl/localcerts/shib1.srv.lrz.de.pem
7 -
8 add: olcTLSCertificateKeyFile
9 olcTLSCertificateKeyFile: /etc/ssl/localkeys/shib1.srv.lrz.de-nopw.key
```

Die Attribute müssen nun noch zu `cn=config` hinzugefügt werden:

```
1 ldapmodify -Y EXTERNAL -H ldapi:/// -f ./ldaps.ldif
```

3 Konfiguration Shibboleth Identity Provider

An diesem Punkt sollten alle Vorarbeiten für die Konfiguration des Shibboleth Identity Providers beendet sein. Ob der IDP richtig installiert wurde und korrekt läuft, kann mittels eines Shibboleth-internen Status Skripts ermittelt werden. Hierfür im Ordner `/opt/shibboleth-idp/bin/` das Skrip `./status.sh` ausführen. Als Output sollten Informationen über den IDP erscheinen, wie z.B.:

```
1  ### Operating Environment Information
2  operating_system: Linux
3  operating_system_version: 4.9.0-9-amd64
4  operating_system_architecture: amd64
5  jdk_version: 1.8.0_212
6  available_cores: 1
7  used_memory: 172 MB
8  maximum_memory: 1820 MB
9
10 ### Identity Provider Information
11 idp_version: 3.4.3
12 start_time: 2019-07-15T18:30:50+02:00
13 current_time: 2019-07-15T18:30:51+02:00
14 uptime: 825 ms
15
16 service: shibboleth.LoggingService
17 last successful reload attempt: 2019-06-18T05:41:36Z
18 last reload attempt: 2019-06-18T05:41:36Z
19
20 service: shibboleth.ReloadableAccessControlService
21 last successful reload attempt: 2019-06-18T05:42:19Z
22 last reload attempt: 2019-06-18T05:42:19Z
23
24 service: shibboleth.MetadataResolverService
25 last successful reload attempt: 2019-06-18T05:42:16Z
26 last reload attempt: 2019-06-18T05:42:16Z
27
28 [...]
```

Ist die Statusabfrage erfolgreich verlaufen, kann mit der Inbetriebnahme einer Datenbank fortgefahren werden.

MySQL Datenbank

Der Shibboleth IDP betreibt einige Komponenten, die die Speicherung von Daten in einer Datenbank benötigen, da die Default Speicherservices von Shibboleth die Daten entweder in Client-Side Cookies oder im Hauptspeicher des Servers speichern. In diesem Abschnitt wird ausführlich dokumentiert, wie eine MySQL Datenbank mittels JPA (Java Persistence API) und Hibernate konfiguriert werden kann.

MySQL wird mittels

```
1 apt-get install mysql-server mysql-client libmysql-java
```

installiert. Damit bei einem Upgrade von MySQL die Datei `/etc/tomcat8/catalina.properties` nicht jedes mal neu überarbeitet werden muss, kann das MySQL JAR-File mit dem Runtime-Library-Verzeichnis von Tomcat verlinkt werden:

```
1 ln -s /usr/share/java/mysql.jar /var/lib/tomcat8/lib/mysql.jar
```

Jetzt muss Tomcat neu getsartet werden, damit die Änderungen aktiv werden.

```
1 Service tomcat8 restart
```

Konfiguration MySQL Zuerst muss die logische Datenbank `shibboleth` erstellt werden. Hierfür müssen die Tabellen für die "shibpid" und "StorageRecords" angelegt werden. Unter "shibpid" werden sogenannte Persistent-IDs gespeichert. Mit diesen persistenten Identifiern kann zwischen IDP und SP ein eindeutiger Identifier für einen Benutzer ausgetauscht werden, ohne dass private Daten des Users offengelegt werden müssen. Die Persistent ID wird vom IDP genieriert, sobald ein User zum ersten mal Zugang zu einem spezifischen SP erhalten will. Für die Konfiguration der Datenbank wird ein Skript in der Datei `/root/shibboleth-db.sql` angelegt. Es generiert die benötigten Tabellen und definiert einen Benutzer "shibboleth", der auf die Datenbank zugreifen kann:

```
SET NAMES 'utf8';
SET CHARACTER SET utf8;
CHARSET utf8;
CREATE DATABASE IF NOT EXISTS shibboleth CHARACTER SET=utf8;
USE shibboleth;

CREATE TABLE IF NOT EXISTS shibpid (
    localEntity VARCHAR(255) NOT NULL,
    peerEntity VARCHAR(255) NOT NULL,
    persistentId VARCHAR(50) NOT NULL,
    principalName VARCHAR(50) NOT NULL,
    localId VARCHAR(50) NOT NULL,
    peerProvidedId VARCHAR(50) NULL,
    creationDate TIMESTAMP NOT NULL,
    deactivationDate TIMESTAMP NULL,
    PRIMARY KEY (localEntity, peerEntity, persistentId)
```

```
);

CREATE TABLE StorageRecords (
    context VARCHAR(255) NOT NULL,
    id VARCHAR(255) NOT NULL,
    expires BIGINT DEFAULT NULL,
    value TEXT NOT NULL,
    version BIGINT NOT NULL,
    PRIMARY KEY (context, id)
);

CREATE USER 'shibboleth'@'localhost' IDENTIFIED BY 'test-1234';

GRANT ALL PRIVILEGES ON shibboleth.* TO 'shibboleth'@'localhost';

FLUSH PRIVILEGES;
```

Die Konfigurationsdatei wird jetzt in MySQL eingelesen und MySQL wird neu gestartet:

```
1 mysql < ./shibboleth-db.sql
2 systemctl restart mysql
```

Aktivierung der persistenten ID Die Konfiguration der Generierung eines persistenten Identifiers erfolgt in 3 Schritten:

1. Es muss festgelegt werden, wie die ID generiert und abgelegt werden soll. In `/opt/shibboleth-idp/conf/saml-nameid.properties` werden hierfür folgende Parameter gesetzt.

```
1 idp.persistentId.generator = shibboleth.StoredPersistentIdGenerator
2 idp.persistentId.salt = <Ergebnis von 'openssl rand -base64 36'>
3 idp.persistentId.dataSource = shibboleth.MySQLDataSource
4 idp.persistentId.sourceAttribute = uid
5 idp.persistentId.computed = shibboleth.ComputedPersistentIdGenerator
```

   Der Salt-Wert kann über den Befehl

```
1 openssl rand -base64 36
```

   gewonnen werden.

2. Das entsprechende Java Bean muss für die Generierung aktiviert werden. Hierfür in `/opt/shibboleth-idp/conf/saml-nameid.xml` den Kommentar in "PersistentGenerator" entfernen:

```
1 <util:list id="shibboleth.SAML2NameIDGenerators">
2
3     <ref bean="shibboleth.SAML2TransientGenerator" />
4
5     <ref bean="shibboleth.SAML2PersistentGenerator" />
6
7     [...]
```

3. Zur Aktivieren der Verarbeitung der ID in `/opt/shibboleth-idp/conf/c14n/subject-c14n.xml` den Kommentar bei SAML2Persistent Bean entfernen:

```
<!-- Handle a SAML 2 persistent ID, provided a stored strategy is in use. -->
<ref bean="c14n/SAML2Persistent" />
```

Data Source Folgende Beans müssen zu `/opt/shibboleth-idp/conf/global.xml` hinzugefügt werden:

- shibboleth.MySQLDataSource
- shibboleth.JPAStorageService
- shibboleth.JPAStorageService.EntityManagerFactory
- shibboleth.JPAStorageService.JPAVendorAdapter

Hierfür die Beans mit folgendem Inhalt an die Datei `/opt/shibboleth-idp/conf/global.xml` anhängen:

```
<!-- A DataSource bean suitable for use in the idp.persistentId.dataSource property. -->
<bean id="shibboleth.MySQLDataSource" class="org.apache.commons.dbcp.BasicDataSource"
      p:driverClassName="com.mysql.jdbc.Driver"
      p:url="jdbc:mysql://localhost:3306/shibboleth?autoReconnect=true"
      p:username="shibboleth"
      p:password=password
      p:maxActive="10"
      p:maxIdle="5"
      p:maxWait="15000"
      p:testOnBorrow="true"
      p:validationQuery="select 1"
      p:validationQueryTimeout="5" />

<bean id="shibboleth.JPAStorageService" class="org.opensaml.storage.impl.JPAStorageService"
      p:cleanupInterval="\%{idp.storage.cleanupInterval:PT10M}"
      c:factory-ref="shibboleth.JPAStorageService.entityManagerFactory"/>

<bean id="shibboleth.JPAStorageService.entityManagerFactory"
      class="org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean">
      <property name="packagesToScan" value="org.opensaml.storage.impl"/>
      <property name="dataSource" ref="shibboleth.MySQLDataSource"/>
      <property name="jpaVendorAdapter" ref="shibboleth.JPAStorageService.JPAVendorAdapter"/>
      <property name="jpaDialect">
        <bean class="org.springframework.orm.jpa.vendor.HibernateJpaDialect" />
      </property>
</bean>

<bean id="shibboleth.JPAStorageService.JPAVendorAdapter" class="org.springframework.orm.jpa.vendor.HibernateJpaVendorAdapter">
```

```
29          <property name="database" value="MYSQL" />
30  </bean>
```

Schließlich müssen noch die Properties für Datenbank-Zugriff in `/opt/shibboleth-idp/conf/idp.properties` ergänzt werden:

```
1  mysql.class    = org.apache.tomcat.jdbc.pool.DataSource
2  mysql.url      = jdbc:mysql://localhost:3306/shibboleth
3  # - Achtung: ggf. kann es zu einem Timezone-Bug kommen. Dann ist die Angabe
       des Timezone-Parameters erforderlich, z.B. mysql.url = jdbc:mysql://
       localhost:3306/shibboleth/?serverTimezone=UTC
4  mysql.username = shibboleth
5  mysql.password = passwort
```

Jetzt Tomcat neu starten und darauf achten, dass die Datei `global.xml` korrekt eingelesen wird.

```
1  service tomcat8 restart
```

LDAP-Anbindung für Benutzerauthentifizierung und Attribute Resolving

Ein Shibboleth IDP kann die Authentifizierung mit verschiedenen Datenquellen durchführen, in diesem Testbeispiel wurde sich für LDAP entschieden. Obwohl der IDP den LDAP Server sowohl für die Authentifizierung als auch für das Attribute Resolving benötigt, handelt es sich um zwei unterschiedliche Verbindungen. In Shibboleth reicht es dagegen aus, in `/opt/shibboleth-idp/conf/ldap.properties` die LDAP-Anbindung zu definieren. Der Attribute Resolver kopiert einfach die Konfiguration für die Authentifizierung. Für die LDAP-Anbindung benötigt man folgende Informationen:

- Die Methode zur Authentifizierung muss auf “bindSearchAuthenticator” gesetzt werden. Dies bedeutet, dass der LDAP Client sich an einen Systemaccount binden muss, bevor er nach Benutzeraccounts suchen kann.
- TLS- und Zertifikatseinstellungen müssen dem LDAP-Server entsprechend gesetzt werden
  - SSL wird aktiviert
  - StartTLS wird deaktiviert
  - Als Vertrauenspunkt wird ein Zertifikat verwendet (idp.authn.LDAP.sslConfig). Der IDP kann aber auch jvmTrust (ein Trust Store der Java Virtual Machine) oder keyStoreTrust (Zertifikate sind in einem speziellen Key Store gespeichert) verwenden
  - Als Zertifikat wird das vertrauenswürdige Serverzertifikat verwendet
- LDAP-spezifische Daten für die Anbindung
  - eine LDAP URL: `ldaps://127.0.0.1:636`
  - einen Base DN: ou=people,dc=srv,dc=lrz,dc=de
  - einen User Filter (Defaultwert): (uid=user)
  - einen Bind DN: cn=admin,dc=srv,dc=lrz,dc=de
  - Bind-DN Credentials: ¡Passwort¿

In der Datei werden die Werte an folgenden Stellen gesetzt:

```
idp.authn.LDAP.authenticator = bindSearchAuthenticator
idp.authn.LDAP.ldapURL = ldaps://127.0.0.1:636
idp.authn.LDAP.useStartTLS = false
idp.authn.LDAP.useSSL = true
idp.authn.LDAP.sslConfig = certificateTrust
idp.authn.LDAP.trustCertificates = /etc/ssl/localcerts/shib1.srv.lrz.de.pem
idp.authn.LDAP.baseDN = ou=people,dc=srv,dc=lrz,dc=de
idp.authn.LDAP.userFilter = (uid={user})
idp.authn.LDAP.bindDN = cn=admin,dc=srv,dc=lrz,dc=de
idp.authn.LDAP.bindDNCredential = <Passwort>
```

Damit die LDAP-Anbindung wirksam wird, muss Tomcat neu gestartet werden.

4 Föderationsmetadaten

Für diesen Testaufbau wurde entschieden, dass alle Metadaten über an der Föderation teilnehmende IDPs und SPs in einer zentralen Datei festgehalten werden, hier "Föderationsmetadaten" genannt. Die Datei muss für alle Komponenten frei zugänglich sein, z.B. über einen Webserver. Aus Gründen der Vereinfachung und zeitlichen Begrenztheit wurde entschieden, die Föderationsmetadaten von den Komponenten lokal speichern zu lassen. Hierbei ist darauf zu achten, dass die Komponenten stets die aktuellen Metadaten besitzen.

Die Föderationsmetadaten-Datei benötigt ein eigenes Zertifikat. Hierfür wurde aus zeitlichen Gründen ein selbstsigniertes Zertifikat erstellt. In Ordner `/etc/ssl/localcerts/` wechseln und dort ein selbstsigniertes Zertifikat erstellen:

```
openssl req -x509 -newkey rsa:2048 -keyout metadata-key-server.pem -out
    metadata-cert-server.pem -nodes -days 3650
```

Zertifikat und Schlüssel müssen jeweils in `/etc/ssl/localcerts/` und `/etc/ssl/localkeys/` verschoben werden. Für das Zertifikat wurden folgende Angaben gemacht:

- Land: DE
- State: Bayern
- Locality: Munich
- Organization Name: Metadaten Provider Federation
- OU:
- FQDN: metadata.shib1.srv.lrz.de
- E-Mail: idp-admin@shib1.srv.lrz.de

Die vollständige Datei `federation-metadata.xml` sieht folgendermaßen aus. Aus Gründen der Übersichtlichkeit wurden sämtliche Zertifikatswerte entfernt:

```
<EntitiesDescriptor Name="https://shib1.srv.lrz.de/metadata/federation-
    metadata.xml"
    xmlns="urn:oasis:names:tc:SAML:2.0:metadata"
    xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
    xmlns:shibmd="urn:mace:shibboleth:metadata:1.0"
```

```
5       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
6
7           <!-- Selbstsigniertes Zertifikat -->
8           <ds:Signature>
9                   <ds:SignedInfo>
10                          <ds:CanonicalizationMethod Algorithm="http://www.w3.
                                org/2001/10/xml-exc-c14n#" />
11                          <ds:SignatureMethod Algorithm="http://www.w3.org
                                /2001/04/xmldsig-more#rsa-sha256" />
12                          <ds:Reference URI="#SWITCHaai-20190411090101">
13                                  <ds:Transforms>
14                                          <ds:Transform Algorithm="http://www.w3
                                                .org/2000/09/xmldsig#enveloped-
                                                signature" />
15                                          <ds:Transform Algorithm="http://www.w3
                                                .org/2001/10/xml-exc-c14n#" />
16                                  </ds:Transforms>
17                                  <ds:DigestMethod Algorithm="http://www.w3.org
                                        /2001/04/xmlenc#sha256" />
18                                  <ds:DigestValue>
19                                          <!-- Digest Wert -->
20                                  </ds:DigestValue>
21                          </ds:Reference>
22                  </ds:SignedInfo>
23
24                  <ds:SignatureValue>
25                          <!-- Signaturwert -->
26                  </ds:SignatureValue>
27                  <ds:KeyInfo>
28                          <ds:KeyValue>
29                                  <ds:RSAKeyValue>
30                                          <ds:Modulus>
31                                          <!-- Öffentlicher Schlüssel des
                                                Metadaten-Zertifikats -->
32                                          </ds:Modulus>
33                                          <ds:Exponent>AQAB</ds:Exponent>
34                                  </ds:RSAKeyValue>
35                          </ds:KeyValue>
36                          <ds:X509Data>
37                                  <ds:X509Certificate>
38                                  <!-- Zertifikatswert des Metadaten-Zertifikats
                                        -->
39                                  </ds:X509Certificate>
40                          </ds:X509Data>
41                  </ds:KeyInfo>
42          </ds:Signature>
43
44
45  <!-- Auflistung aller IDPs -->
46
47      <!-- Shib1 als IDP -->
48      <EntityDescriptor entityID="https://shib1.srv.lrz.de/idp/shibboleth">
49         <IDPSSODescriptor protocolSupportEnumeration="urn:oasis:names:tc:SAML
               :2.0:protocol">
50            <KeyDescriptor use="signing">
51               <ds:KeyInfo>
52                  <ds:X509Data>
```

```
53                          <ds:X509Certificate>
54                          <!-- Zertifikatswert des Serverzertifikats von Shib1
                                -->
55                          </ds:X509Certificate>
56                      </ds:X509Data>
57                  </ds:KeyInfo>
58              </KeyDescriptor>
59              <NameIDFormat>
60                    urn:oasis:names:tc:SAML:2.0:nameid-format:transient
61              </NameIDFormat>
62
63              <SingleSignOnService Binding="urn:oasis:names:tc:SAML:2.0:bindings:
                    HTTP-POST"
64                           Location="https://shib1.srv.lrz.de/idp/profile/SAML2/POST/
    SSO" />
65
66              <SingleSignOnService Binding="urn:oasis:names:tc:SAML:2.0:bindings:
                    HTTP-Redirect"
67                           Location="https://shib1.srv.lrz.de/idp/profile/SAML2/
    Redirect/SSO" />
68          </IDPSSODescriptor>
69
70          <Organization>
71                  <OrganizationName xml:lang="en">shib1.srv.lrz.de</
                        OrganizationName>
72                  <OrganizationDisplayName xml:lang="en">Shibboleth Test IdP</
                        OrganizationDisplayName>
73                  <OrganizationDisplayName xml:lang="de">Shibboleth Test IdP</
                        OrganizationDisplayName>
74
75                  <OrganizationURL xml:lang="en">http://shib1.srv.lrz.de/</
                        OrganizationURL>
76                  <OrganizationURL xml:lang="de">http://shib1.srv.lrz.de/</
                        OrganizationURL>
77
78          </Organization>
79          <ContactPerson contactType="technical">
80                  <GivenName>Hans</GivenName>
81                  <SurName>Meier</SurName>
82                  <EmailAddress>idp-admin@shib1.srv.lrz.de.de</EmailAddress>
83          </ContactPerson>
84
85      </EntityDescriptor>
86
87  <!-- Auflistung aller SPs -->
88
89      <!-- Shib2 als SP -->
90      <EntityDescriptor entityID="https://shib2.srv.lrz.de/shibboleth">
91          <SPSSODescriptor protocolSupportEnumeration="urn:oasis:names:tc:SAML
                :2.0:protocol urn:oasis:names:tc:SAML:1.1:protocol">
92
93              <Extensions>
94              <!-- Hier kann ein potentieller DS konfiguriert werden -->
95                  <idpdisc:DiscoveryResponse xmlns:idpdisc="urn:oasis:names:tc:
                        SAML:profiles:SSO:idp-discovery-protocol"
96                           index="1" Binding="urn:oasis:names:tc:SAML:profiles:
    SSO:idp-discovery-protocol"
```

```
97                          Location="http://shib2.srv.lrz.de/Shibboleth.sso/DS" />
98                  <idpdisc:DiscoveryResponse xmlns:idpdisc="urn:oasis:names:tc:
                        SAML:profiles:SSO:idp-discovery-protocol"
99                          index="2" Binding="urn:oasis:names:tc:SAML:profiles:
    SSO:idp-discovery-protocol"
100                         Location="https://shib2.srv.lrz.de/Shibboleth.sso/DS"/
    >
101             </Extensions>
102
103         <KeyDescriptor>
104                 <ds:KeyInfo>
105                     <ds:X509Data>
106                         <ds:X509Certificate>
107                         <!-- Zertifikatswert des Serverzertifikats von Shib2
                                -->
108                         </ds:X509Certificate>
109                     </ds:X509Data>
110                 </ds:KeyInfo>
111             </KeyDescriptor>
112
113             <NameIDFormat>urn:oasis:names:tc:SAML:2.0:nameid-format:transient<
                    /NameIDFormat>
114
115             <AssertionConsumerService index="1" isDefault="true"
116                     Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
117                     Location="https://shib2.srv.lrz.de/Shibboleth.sso/SAML2/
    POST" />
118             <AssertionConsumerService index="2"
119                     Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST-
    SimpleSign"
120                     Location="https://shib2.srv.lrz.de/Shibboleth.sso/SAML2/
    POST-SimpleSign" />
121             <AssertionConsumerService index="3"
122                     Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-
    Artifact"
123                     Location="https://shib2.srv.lrz.de/Shibboleth.sso/SAML2/
    Artifact" />
124             <AssertionConsumerService index="4"
125                     Binding="urn:oasis:names:tc:SAML:1.0:profiles:browser-post
    "
126                     Location="https://shib2.srv.lrz.de/Shibboleth.sso/SAML/
    POST" />
127             <AssertionConsumerService index="5"
128                     Binding="urn:oasis:names:tc:SAML:1.0:profiles:artifact-01"
129                     Location="https://shib2.srv.lrz.de/Shibboleth.sso/SAML/
    Artifact" />
130
131         </SPSSODescriptor>
132
133     <Organization>
134         <OrganizationName xml:lang="en">shib2.srv.lrz.de</OrganizationName>
135         <OrganizationDisplayName xml:lang="en">Shibboleth Test SP</
                OrganizationDisplayName>
136         <OrganizationURL xml:lang="en">http://shib2.srv.lrz.de/</
                OrganizationURL>
137     </Organization>
138     <ContactPerson contactType="technical">
```

```
139             <GivenName>Your</GivenName>
140             <SurName>Admin</SurName>
141             <EmailAddress>sp-admin@shib2.srv.lrz.de</EmailAddress>
142         </ContactPerson>
143
144         </EntityDescriptor>
145
146 </EntitiesDescriptor>
```

Die Datei muss nun an einem passenden Ort abgelegt werden. Da die Föderationsmetadaten später auf einem Webserver verfügbar sein sollen, bietet sich der Apache Webserver als Speicherort an: Den Ordner `/var/www/html/metadata/` anlegen und die Datei `federation-metadata.xml` darin ablegen. Bei Bedarf die `index.html` Datei löschen.

5 Konfiguration der Metadaten des IDP

In der Datei `/opt/shibboleth-idp/conf/metadata-providers.xml` kann für den IDP festgelegt werden, wie er Metadaten abrufen kann. Damit der IDP weiß, wo die Föderationsmetadaten abzurufen sind, wird in `/opt/shibboleth-idp/conf/metadata-providers.xml` folgenes eingetragen bzw. entkommentiert. Die Zeile bewirkt, dass der IDP im lokalen Dateisystem nach den Metadaten sucht:

```
1 <MetadataProvider id="LocalMetadata" xsi:type="FilesystemMetadataProvider"
      metadataFile="/var/www/html/metadata/federation-metadata.xml" />
```

Zusätzlich muss der Typ FileBackedHTTPMetadataProvider auskommentiert werden, damit Shibboleth nicht versucht, Remote-Metadaten abzurufen:

```
1  <MetadataProvider id="HTTPMetadata"
2                    xsi:type="FileBackedHTTPMetadataProvider"
3                    backingFile="/opt/shibboleth-idp/metadata/federation-metadata-copy.
       xml"
4                    metadataURL="https://shib1.srv.lrz.de/metadata/federation-metadata.
       xml"
5                    maxRefreshDelay="PT2H">
6       <MetadataFilter xsi:type="SignatureValidation" requireSignedRoot="false"
7                       certificateFile="/etc/ssl/localcerts/metadata-cert-server.pem" />
8       <MetadataFilter xsi:type="EntityRoleWhiteList">
9                      <RetainedRole>md:SPSSODescriptor</RetainedRole>
10      </MetadataFilter>
11 </MetadataProvider>
```

6 Konfiguration der Attribut-Generierung und -Freigabe

Schließlich wird für den IDP konfiguriert, welche Attribute freigegeben werden sollen (Attribute Filter) und wie diese vereinheitlicht an einen SP weitergegeben werden sollen (Attribute Resolver).

6.1 Attribute Resover

Der Attribute-Resolver des IDP bekommt seine "rohen" Daten vom Data Connector aus dem Identity Management bzw. Nutzerverzeichnis (z.B. LDAP, AD, SQL). Unser Data

Connector bezieht seine Daten aus einem LDAP Verzeichnis. Da die Attribute aus dem LDAP Verzeichnis nicht mit den definierten Attributen übereinstimmen müssen, muss eine Abbildung der Attribute vom Nutzerverzeichnis auf die vom IDP freigegebenen Daten geschehen. Mit dem Attribute Resolver - definiert in der Datei `/opt/shibboleth-idp/conf/attribute-resolver.xml` - werden die Daten auf die definierten Attribute umgeschrieben. Der Mechanismus hierfür lautet Attribute Definition. Es gibt verschiedene Typen, z.B.:

- Simple Attribute Definition
- Mapped Attribute Definition (Viele-zu-viele Mapping)
- Scoped Attribute Definition (z.B. begrenzt auf eine Domain o.Ä.)
- etc.

Nach der Umwandlung werden sie in eine Form gebracht, die der SP versteht. Diesen Vorgang nennt man Attribute Encoding.

Für das Testbed wurden fünf Attribute ausgewählt, die der IDP freigeben kann:

- eduPersonPrincipalName
- PrincipalName
- mail
- surname
- givenName

Die Attribute `eudPersonPrincipalName` und `PrincipalName` sollen aus Benutzereingaben bestehen, `mail`, `surname` und `givenName` stammen aus unserem LDAP-Verzeichnis. Das Format der Attribute in `attribute-resolver.xml` kann aus der Datei attribute-resolver-full.xml herauskopiert werden. Sie dient als umfangreiche Vorlage für Attributsdefinitionen. In der Konfigurationsdatei unter `/opt/shibboleth-idp/conf/attribute-resolver.xml` können diese Attribute eingetragen werden. Zudem wird am Ende der Datei der LDAP Connector definiert. Werden mehr Attribute benötigt, können diese aus der sehr umfangreichen Beispieldatei `/opt/shibboleth-idp/conf/attribute-resolver-full.xml` in den hier verwendeten Attribute Resolver kopiert werden.

Die vollständige Attribute Resolver-Datei sieht folgendermaßen aus:

```
1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <AttributeResolver
4          xmlns="urn:mace:shibboleth:2.0:resolver"
5          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
6          xsi:schemaLocation="urn:mace:shibboleth:2.0:resolver http://shibboleth
       .net/schema/idp/shibboleth-attribute-resolver.xsd">
7
8
9      <!-- ========================================== -->
10     <!--      Attribute Definitions                 -->
11     <!-- ========================================== -->
12
13 <!-- Attribute aus Userangaben -->
```

```
14
15      <AttributeDefinition id="eduPersonPrincipalName" xsi:type="Scoped" scope="
            %{idp.scope}">
16          <InputAttributeDefinition ref="uid" />
17          <DisplayName xml:lang="en">Principal name</DisplayName>
18          <DisplayName xml:lang="de">Netz-Id</DisplayName>
19          <DisplayDescription xml:lang="en">A unique identifier for a person,
                mainly for inter-institutional user identification</
                DisplayDescription>
20          <DisplayDescription xml:lang="de">Eindeutige,
                einrichtungsübergreifende Nutzerkennung</DisplayDescription>
21          <AttributeEncoder xsi:type="SAML1ScopedString" name="urn:mace:dir:
                attribute-def:eduPersonPrincipalName" encodeType="false" />
22          <AttributeEncoder xsi:type="SAML2ScopedString" name="urn:oid
                :1.3.6.1.4.1.5923.1.1.1.6" friendlyName="eduPersonPrincipalName"
                encodeType="false" />
23      </AttributeDefinition>
24
25      <AttributeDefinition id="uid" xsi:type="PrincipalName">
26          <DisplayName xml:lang="en">User Name</DisplayName>
27          <DisplayName xml:lang="de">Nutzerkennung</DisplayName>
28          <DisplayDescription xml:lang="en">Local User Id</DisplayDescription>
29          <DisplayDescription xml:lang="de">Nutzerkennung der Heimateinrichtung<
                /DisplayDescription>
30          <AttributeEncoder xsi:type="SAML1String" name="urn:mace:dir:attribute-
                def:uid" encodeType="false" />
31          <AttributeEncoder xsi:type="SAML2String" name="urn:oid
                :0.9.2342.19200300.100.1.1" friendlyName="uid" encodeType="false"
                />
32      </AttributeDefinition>
33
34  <!--- Attribute aus dem IdM (LDAP-Verzeichnis) -->
35
36      <AttributeDefinition id="mail" xsi:type="Simple">
37          <InputDataConnector ref="myLDAP" attributeNames="mail"/>
38          <DisplayName xml:lang="en">E-mail</DisplayName>
39          <DisplayName xml:lang="de">E-Mail</DisplayName>
40          <DisplayDescription xml:lang="en">E-Mail address</DisplayDescription>
41          <DisplayDescription xml:lang="de">E-Mail Adresse</DisplayDescription>
42          <AttributeEncoder xsi:type="SAML1String" name="urn:mace:dir:attribute-
                def:mail" encodeType="false" />
43          <AttributeEncoder xsi:type="SAML2String" name="urn:oid
                :0.9.2342.19200300.100.1.3" friendlyName="mail" encodeType="false"
                />
44      </AttributeDefinition>
45
46      <AttributeDefinition id="surname" xsi:type="Simple">
47          <InputDataConnector ref="myLDAP" attributeNames="sn"/>
48          <DisplayName xml:lang="en">Surname</DisplayName>
49          <DisplayName xml:lang="de">Nachname</DisplayName>
50          <DisplayDescription xml:lang="en">Surname or family name</
                DisplayDescription>
51          <DisplayDescription xml:lang="de">Familienname des Nutzers bzw. der
                Nutzerin</DisplayDescription>
52          <AttributeEncoder xsi:type="SAML1String" name="urn:mace:dir:attribute-
                def:sn" encodeType="false" />
53          <AttributeEncoder xsi:type="SAML2String" name="urn:oid:2.5.4.4"
```

```
            friendlyName="sn" encodeType="false" />
54      </AttributeDefinition>
55
56      <AttributeDefinition id="givenName" xsi:type="Simple">
57          <InputDataConnector ref="myLDAP" attributeNames="givenName"/>
58          <DisplayName xml:lang="en">Given name</DisplayName>
59          <DisplayName xml:lang="de">Vorname</DisplayName>
60          <DisplayDescription xml:lang="en">Given name of a person</
                DisplayDescription>
61          <DisplayDescription xml:lang="de">Vorname des Nutzers bzw. der
                Nutzerin</DisplayDescription>
62          <AttributeEncoder xsi:type="SAML1String" name="urn:mace:dir:attribute-
                def:givenName" encodeType="false" />
63          <AttributeEncoder xsi:type="SAML2String" name="urn:oid:2.5.4.42"
                friendlyName="givenName" encodeType="false" />
64      </AttributeDefinition>
65
66
67      <!-- ========================================================== -->
68      <!--       Data Connectors                                      -->
69      <!-- ========================================================== -->
70
71  <!--
72      <DataConnector id="staticAttributes" xsi:type="Static">
73          <Attribute id="affiliation">
74              <Value>member</Value>
75          </Attribute>
76      </DataConnector>
77  -->
78
79
80     <!-- Mein LDAP Connector -->
81     <DataConnector id="myLDAP" xsi:type="LDAPDirectory"
82          ldapURL="%{idp.attribute.resolver.LDAP.ldapURL}"
83          baseDN="%{idp.attribute.resolver.LDAP.baseDN}"
84          principal="%{idp.attribute.resolver.LDAP.bindDN}"
85          principalCredential="%{idp.attribute.resolver.LDAP.bindDNCredential}"
86          useStartTLS="%{idp.attribute.resolver.LDAP.useStartTLS:true}"
87          connectTimeout="%{idp.attribute.resolver.LDAP.connectTimeout}"
88          trustFile="%{idp.attribute.resolver.LaDAP.trustCertificates}"
89          responseTimeout="%{idp.attribute.resolver.LDAP.responseTimeout}">
90          <FilterTemplate>
91              <![CDATA[
92                  %{idp.attribute.resolver.LDAP.searchFilter}
93              ]]>
94          </FilterTemplate>
95          <ConnectionPool
96              minPoolSize="%{idp.pool.LDAP.minSize:3}"
97              maxPoolSize="%{idp.pool.LDAP.maxSize:10}"
98              blockWaitTime="%{idp.pool.LDAP.blockWaitTime:PT3S}"
99              validatePeriodically="%{idp.pool.LDAP.validatePeriodically:true}"
100             validateTimerPeriod="%{idp.pool.LDAP.validatePeriod:PT5M}"
101             expirationTime="%{idp.pool.LDAP.idleTime:PT10M}"
102             failFastInitialize="%{idp.pool.LDAP.failFastInitialize:false}" />
103     </DataConnector>
104
105
```

```
106 </AttributeResolver>
```

6.2 Attribute Filter

Der Attribute Filter (`/opt/shibboleth-idp/conf/attribute-filter.xml`) legt fest, welche Attribute an einen SP versendet werden dürfen. Die passenden Attribute werden mittels der ID aus dem Attribute Resolver referenziert. Ein einfacher Attribute-Filter, passend zum oben definierten Attribute-Resolver, wird im folgenden Listing als Beispiel gegeben. Hier werden einfach alle fünf definierten Attribute an den SP freigegeben:

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!--
3      This file is an EXAMPLE policy file.  While the policy presented in this
4      example file is illustrative of some simple cases, it relies on the names
           of
5      non-existent example services and the example attributes demonstrated in
           the
6      default attribute-resolver.xml file.
7
8      This example does contain some usable "general purpose" policies that may
           be
9      useful in conjunction with specific deployment choices, but those policies
            may
10     not be applicable to your specific needs or constraints.
11 -->
12 <AttributeFilterPolicyGroup id="ShibbolethFilterPolicy"
13         xmlns="urn:mace:shibboleth:2.0:afp"
14         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
15         xsi:schemaLocation="urn:mace:shibboleth:2.0:afp http://shibboleth.net/
       schema/idp/shibboleth-afp.xsd">
16
17    <!-- Attribute an unseren Test-IdP freigeben -->
18    <AttributeFilterPolicy id="lrz_test_sp">
19         <PolicyRequirementRule xsi:type="OR">
20             <Rule xsi:type="Requester" value="https://shib2.srv.lrz.de/
                   shibboleth" />
21         </PolicyRequirementRule>
22
23         <AttributeRule attributeID="uid"                    permitAny="true" />
24         <AttributeRule attributeID="eduPersonPrincipalName" permitAny="true" />
25         <AttributeRule attributeID="mail"                   permitAny="true" />
26         <AttributeRule attributeID="surname"                permitAny="true" />
27         <AttributeRule attributeID="givenName"              permitAny="true" />
28   </AttributeFilterPolicy>
29
30 </AttributeFilterPolicyGroup>
```

Die freigegebenen Attribute werden im sogenannten “Attribute Statement” übertragen mit Hilfe einer SAML2 Assertion. Der empfangende SP identifiziert die Attribute anhand eines URI Wertes (meist vom Typ urn:oid).

7 Testing

Zum Schluss dieser Dokumentation sollen noch einige Testmöglichkeiten vorgestellt werden.

7.1 Statuspage

Die IDP Statuspage kann über folgenden Befehl abgerufen werden:

```
curl -sk https://localhost/idp/status
```

Ein Output wie Folgender sollte dabei herauskommen:

```
### Operating Environment Information
operating_system: Linux
operating_system_version: 4.9.0-9-amd64
operating_system_architecture: amd64
jdk_version: 1.8.0_212
available_cores: 1
used_memory: 241 MB
maximum_memory: 1820 MB

### Identity Provider Information
idp_version: 3.4.3
start_time: 2019-05-13T14:50:54+02:00
current_time: 2019-05-13T14:54:16+02:00
uptime: 202035 ms
```

Möchte man von einem Rechner außerhalb des Netzwerks auf diese Seite zugreifen, kann in `/shibboleth-idp/conf/access-control.xml` die IP Adresse des entsprechenden Rechners eingetragen werden unter "AccessByIPAddress"'.

7.2 AACLI Skript

Ist in der Testumgebung noch kein SP vorhanden, kann mittels des AACLI Skriptes (`/opt/shibboleth-idp/bin/aacli.sh`) die Attributfreigabe bereits im Vorhinein getestet werden. So kann sichergestellt werden, dass die Attributfreigabe korrekt konfiguriert wurde. Als Beispiel wurde hier die Attributfreigabe mit dem Testuser shibboleth ausgeführt:

```
/opt/shibboleth-idp/bin/aacli.sh -n shibbolethuser -r https://sp.shib2.srv.lrz
    .de/shibboleth
```

Anhang 2: Installation und Konfiguration eines Shibboleth Service Providers

Folgende Dokumentation erläutert, wie ein Shibboleth SP für das Testbed am LRZ aufgesetzt werden kann.

Nach der erfolgreichen Installation und Konfiguration eines Shibboleth IDP widmet sich dieses Kapitel dem Shibboleth Service Provider. Fragestellung ist hier, wie der SP zu konfigurieren ist, damit er die in Kapitel 2.3 und 3.3 theoretisch beschriebenen Funktionen korrekt umsetzt.

1 Voraussetzungen

Bevor mit einer Installation begonnen werden kann, müssen einige Voraussetzungen bestehen:

- Betriebssystem: eine virtuelle Maschine mit einem Linux Betriebssystem, in diesem Fall Debian 9. Für die Hardware sollten 4 GB RAM und 20 GB Festplattenspeicher eingeplant werden.
- Ein Zugang über SSH mit einem Benutzer mit Root-Rechten muss vorhanden sein.
- Netzwerk:
  - Es wird ein Hostname (shib2.srv.lrz.de) und eine statische IP-Adresse benötigt (129.187.255.172).
  - Die Firewall sollte für SSL-Verbindungen, über welche externe Komponenten auf den SP zugreifen, die Ports 443 und 8443 geöffnet haben.
  - Die DNS Auflösung zum IDP muss möglich sein. Hierfür in `/etc/hosts/` die IP Adresse (129.187.255.171) und den FQDN (shib1.srv.lrz.de) angeben.
  - Falls ein externe NTP-Server verwendet wird, sollte dieser ebenso an der Firewall freigeschaltet werden.
- Zertifikat: Für den Hostnamen (shib2.srv.lrz.de) sollte ein Zertifikat bereitgestellt werden, dessen CA die gängigen Browser vertrauen. Für den Server wurde aus der LRZ-internen PKI ein Zertifikat beantragt. Sie werden unter /etc/ssl/localcerts bzw. /etc/ssl/localkeys abgelegt. Das Zertifikat für die CA liegt unter /etc/ssl/chains/.

2 Installation und Konfiguration von Basiskomponenten

Zuerst müssen auf dem SP einige Programme installiert werden, die für den Betrieb notwendig sind. Der Shibboleth SP benötigt einen Apache Webserver, um die Erreichbarkeit eines von einem Benutzer angesteuerten Services zu simulieren.

```
apt-get install apache2 net-tools curl
```

Folgende Umgebungsvariablen wurden in `/etc/environment` eingetragen:

```
APACHE_LOG=/var/log/apache2
SHIB_SP=/opt/shibboleth-sp/etc/shibboleth
```

und gesourced:

```
source /etc/environment
export APACHE_LOG=/var/log/apache2
export SHIB_SP=/opt/shibboleth-sp/etc/shibboleth
```

Da der Shibboleth Service Provider nur offizielle Binaries für RPM-basierte Linux Distributionen anbietet[1], betreibt SWITCH ein eigenes Repository mit Stable-Releases für Debain[2]. Deshalb muss in `/etc/apt/sources.list` folgende Quelle hinzugefügt werden:

```
deb http://deb.debian.org/debian stretch-backports main
```

Die Paketquellen müssen nun via

```

Für den shibd-Daemon müssen schließlich noch Leserechte für den Schlüssel des
    Zertifikats shib2.srv.lrz.de vergeben werden:
\begin{lstlisting}[language=bash]
chmod 644 /etc/ssl/localkeys/shib2.srv.lrz.de-nopw.key
```

3 Installation Shibboleth Service Provider

Zur Installation folgende Schritte ausführen:

1. Repository Package herunterladen in `/root/`:

   ```
   curl --fail --remote-name https://pkg.switch.ch/switchaai/debian/dists/
       stretch/main/binary-all/misc/switchaai-apt-source_1.0.0~bpo9+1_all.deb
   ```

2. Repository Package installieren:

   ```
   sudo apt install ./switchaai-apt-source_1.0.0~bpo9+1_all.deb
   ```

3. Installation mit folgenden Kommandos, damit eventuell auftretende Konflikte in der Namensgebung verhindert werden:

   ```
   sudo apt install -t stretch-backports init-system-helpers libxerces-c3.2
   sudo apt install --install-recommends shibboleth
   sudo apt upgrade
   sudo apt autoremove
   ```

Ob die Installation erfolgreich verlaufen ist, kann über folgenden Aufruf des shibd-Daemons geschehen:

```
shibd -t
```

Wichtig ist hierbei die Ausgabe der letzten Zeile:

```
overall configuration is loadable, check console for non-fatal problems
```

[1] https://wiki.shibboleth.net/confluence/display/SP3/LinuxInstall, aufgerufen am 15.5.2019
[2] http://pkg.switch.ch/switchaai/, aufgerufen am 15.05.2019

4 Konfiguration Apache2

Bevor mit der Konfiguration begonnen wird, muss SSL und Header in Apache aktiviert werden:

```
a2enmod ssl header
```

Da ein Benutzer in der Regel auf eine geschützte Ressource zugreifen möchte, wird unter `/var/www/html/secure-all/` ein Ordner angelegt, in dem die zu schützenden Ressourcen abrufbar sind. In `/etc/apache2/sites-availible/` wird der Virtual Host `shib2.srv.lrz.de.conf` für den Webserver mit folgendem Inhalt erzeugt:

```
<VirtualHost *:443>
  # Servername must match certificate. Shoud be sp.srv...
  ServerName              shib2.srv.lrz.de
  ServerAlias             sp.shib2.srv.lrz.de
  ServerAdmin             sp-admin@shib1.srv.lrz.de

  SSLCertificateFile      /etc/ssl/localcerts/shib2.srv.lrz.de.pem
  SSLCertificateKeyFile   /etc/ssl/localkeys/shib2.srv.lrz.de-nopw.key
  SSLCACertificateFile    /etc/ssl/chains/lrz-cachain-telesec-class2.pem

  SSLEngine on
  SSLCompression off

  #
  # Zur SSL-Konfiguration siehe die Anmerkung unterhalb dieses Beispiels
  #
  SSLProtocol All -SSLv2 -SSLv3
  SSLHonorCipherOrder On
  SSLCompression off
  SSLCipherSuite 'EDH+CAMELLIA:EDH+aRSA:EECDH+aRSA+AESGCM:EECDH+aRSA+SHA384:
      EECDH+aRSA+SHA256:EECDH:+CAMELLIA256:+AES256:+CAMELLIA128:+AES128:+SSLv3
      :!aNULL:!eNULL:!LOW:!3DES:!MD5:!EXP:!PSK:!DSS:!RC4:!SEED:!ECDSA:
      CAMELLIA256-SHA:AES256-SHA:CAMELLIA128-SHA:AES128-SHA'

  Header always set Strict-Transport-Security "max-age=31536000;␣
      includeSubDomains"

  UseCanonicalName On

  AddDefaultCharset UTF-8

  # dieser Alias wird i.d.R. nur benötigt, wenn URL Rewrite o.ä, erfolgt
  # Alias /Shibboleth.sso /Shibboleth.sso
  <Location /Shibboleth.sso>
    SetHandler shib
    Require all granted
  </Location>

  # Metadata unter der entityID-URL:
  Redirect seeother /shibboleth https://shib2.srv.lrz.de/Shibboleth.sso/
      Metadata

  <Location /shibboleth-sp>
    Require all granted
  </Location>
  Alias /shibboleth-sp/main.css /usr/share/shibboleth/main.css
```

```
42    Alias /shibboleth-sp/logo.jpg /usr/share/shibboleth/logo.jpg
43
44    #
45    # Vom SP zu schützende Location
46    #
47    # in diesem Beispiel findet keine Autorisierung statt(!)
48    # zu Autorisierung etc. siehe unter https://www.switch.ch/aai/guides/sp/
          access-rules/
49    <Location /secure-all>
50      AuthType shibboleth
51      ShibRequestSetting requireSession true
52      Require valid-user
53    </Location>
54
55    #
56    # Support für WAYFless-URLs
57    # siehe https://www.ukfederation.org.uk/library/uploads/Documents/
          WAYFlessGuidance.pdf
58    #
59    RedirectMatch /start-session$ /Shibboleth.sso/Login
60  </VirtualHost>
```

Und schließlich die Seite aktivieren und die Default Seite deaktivieren. Mit dem Befehl configtest kann die Konfiguration überprüft werden, bevor Apache neu gestartet wird.

```
1  a2ensite sp.shib2.srv.lrz.de
2  a2dissite 000default...
3  apachectl configtest
4  systemctl restart apache2
```

Port 80 soll nur für den Localhost geöffnet sein. Hierfür in `/etc/apache2/ports.conf` folgendes ändern:

```
1  # If you just change the port or add more ports here, you will likely also
2  # have to change the VirtualHost statement in
3  # /etc/apache2/sites-enabled/000-default.conf
4
5  Listen 127.0.0.1:80
6
7  <IfModule ssl_module>
8    Listen 443
9  </IfModule>
10
11 <IfModule mod_gnutls.c>
12   Listen 443
13 </IfModule>
```

5 Shibboleth Service Provider: Konfiguration

5.1 Die Konfigurationsdatei shibboleth2.xml

Für den Shibboleth SP kann ein Großteil der Konfiguration in der Datei shibboleth2.xml erledigt werden. Zum einen werden die Föderationsmetadaten unter `/etc/shibboleth/federation-metadata.xml` abgelegt. Sie können z.B. über SCP von shib1.srv.lrz.de geholt werden. Dem SP wird in dem Bean MetadataProvider mitgeteilt, dass es lokal im Dateisystem zu finden ist. Das Bean für Remote-Metadaten muss auskommentiert werden. Zum

anderen muss dem SP mitgeteilt werden, wo die Serverzertifikate liegen (unter Credential-Resolver). Alle Änderungen werden in folgendem Listing im Überblick dargestellt:

```
1  [...]
2
3  <ApplicationDefaults entityID="https://shib2.srv.lrz.de/shibboleth"
4       REMOTE_USER="eppn persistent-id targeted-id">
5
6  [...]
7
8        <!-- Example of locally maintained metadata. -->
9
10         <MetadataProvider type="XML" validate="true" path="federation-metadata
               .xml" />
11
12
13         <!-- Our remotely supplied Metadata -->
14 <!--    <MetadataProvider type="XML" uri="https://idp.shib1.srv.lrz.de/metadata
       /federation-metadata.xml"
15                 backingFilePath="federation-metadata.xml" reloadInterval="3600">
16             <MetadataFilter type="Signature" certificate="/etc/ssl/localcerts/
                   federation-cert.pem" />
17         </MetadataProvider>
18 -->
19
20       <!--Pfadangaben zu den Zertifikatsdateien-->
21       <CredentialResolver type="File" key="/etc/ssl/localkeys/shib2.srv.lrz.
             de-nopw.key"
22             certificate="/etc/ssl/localcerts/shib2.srv.lrz.de.pem" />
```

5.2 Attribute Map

Der SP empfängt vom IDP die SAML-Assertions mit Daten über Attribute und Autorisierung des Benutzers. Über eine eindeutige Kennung der Attribute (URI) identifiziert der SP die Attribute. Mit Hilfe der Datei `/etc/shibboleth/attribute-map.xml` bildet der SP die Attribute auf interne Variablen ab. Mit der ID werden die Variablennamen definiert. Durch die Datei `/etc/shibboleth/attribute-policy.xml` können die Variablen gefiltert werden. In der Regel reicht die Konfiguration der Default-Datei aus. Folgende Attribute werden herausgegeben:

- eduPersonPrincipalName
- mail
- surname
- givenName

Folgende Attribute werden nicht herausgegeben:

- UID

Die Datei `attribute-map.xml` muss hierfür folgendermaßen angepasst werden:

```
1  [...]
2
3      <!-- New standard identifier attributes for SAML. -->
4
5      <Attribute name="urn:oasis:names:tc:SAML:attribute:subject-id" id="subject
           -id">
6          <AttributeDecoder xsi:type="ScopedAttributeDecoder" caseSensitive="
               false"/>
7      </Attribute>
8
9      <Attribute name="urn:oasis:names:tc:SAML:attribute:pairwise-id" id="
           pairwise-id">
10         <AttributeDecoder xsi:type="ScopedAttributeDecoder" caseSensitive="
               false"/>
11     </Attribute>
12
13     <!-- The most typical eduPerson attributes. -->
14
15     <!-- eduPersonPrincipalName -->
16     <Attribute name="urn:oid:1.3.6.1.4.1.5923.1.1.1.6" id="eppn">
17         <AttributeDecoder xsi:type="ScopedAttributeDecoder" caseSensitive="
               false"/>
18     </Attribute>
19     <Attribute name="urn:mace:dir:attribute-def:eduPersonPrincipalName" id="
           eppn">
20         <AttributeDecoder xsi:type="ScopedAttributeDecoder" caseSensitive="
               false"/>
21     </Attribute>
22
23 <!--    <Attribute name="urn:oid:1.3.6.1.4.1.5923.1.1.1.9" id="affiliation">
24         <AttributeDecoder xsi:type="ScopedAttributeDecoder" caseSensitive="
               false"/>
25     </Attribute>
26     <Attribute name="urn:mace:dir:attribute-def:eduPersonScopedAffiliation" id
           ="affiliation">
27         <AttributeDecoder xsi:type="ScopedAttributeDecoder" caseSensitive="
               false"/>
28     </Attribute>
29
30     <Attribute name="urn:oid:1.3.6.1.4.1.5923.1.1.1.7" id="entitlement"/>
31     <Attribute name="urn:mace:dir:attribute-def:eduPersonEntitlement" id="
           entitlement"/>
32 -->
33
34     <!--
35     Legacy pairwise identifier attribute / NameID format, intended to be
           replaced by the
36     simpler pairwise-id attribute (see top of file).
37     -->
38
39     <!-- The eduPerson attribute version (note the OID-style name): -->
40     <Attribute name="urn:oid:1.3.6.1.4.1.5923.1.1.1.10" id="persistent-id">
41         <AttributeDecoder xsi:type="NameIDAttributeDecoder" formatter="
               $NameQualifier!$SPNameQualifier!$Name" defaultQualifiers="true"/>
42     </Attribute>
43
44     <!-- The SAML 2.0 NameID Format: -->
```

```
45      <Attribute name="urn:oasis:names:tc:SAML:2.0:nameid-format:persistent" id=
            "persistent-id">
46          <AttributeDecoder xsi:type="NameIDAttributeDecoder" formatter="
                $NameQualifier!$SPNameQualifier!$Name" defaultQualifiers="true"/>
47      </Attribute>
48
49      <!-- Other eduPerson attributes (SAML 2 names followed by SAML 1 names)...
            -->
50
51  <!-- CORE ATTRIBUTES -->
52
53      <!-- E-mail -->
54      <Attribute name="urn:oid:0.9.2342.19200300.100.1.3" id="mail"/>
55      <Attribute name="urn:mace:dir:attribute-def:mail" id="mail"/>
56
57      <!-- Given name -->
58      <Attribute name="urn:oid:2.5.4.42" id="givenName"/>
59      <Attribute name="urn:mace:dir:attribute-def:givenName" id="givenName"/>
60
61      <!-- Surname -->
62      <Attribute name="urn:oid:2.5.4.4" id="surname"/>
63      <Attribute name="urn:mace:dir:attribute-def:sn" id="sn"/>
```

5.3 Testing

An dieser Stelle sollen einige Testmöglichkeiten für den SP vorgestellt werden

Website

Wird die Seite

```
1 wget --no-check-certificate https://localhost/Shibboleth.sso/Session
```

aufgerufen und der Output “A valid session was not found” erscheint, bedeuted dies, dass das Shibboleth Modul geladen wurde und der Webserver mit shibd kommuniziert.

Login Test

Ob die Weiterleitung zum IDP korrekt funktioniert, kann getestet werden, indem in einem Browser folgende Adresse angegeben wird: `https://129.187.255.172/Shibboleth.sso/Login`. Funktioniert der SP korrekt, wird man zur Anmeldeseite des IDP weitergeleitet.

Dump Umgebung Webserver

Mittels eines sehr einfachen PHP Skripts kann die Umgebung des Webservers angezeigt werden. Falls auf dem Server kein PHP installiert ist, muss dies nachgeholt werden. Jetzt muss der Ordner `/var/www/html/secure-all/` angelegt werden, da Apache dort laut Konfiguration die zu schützenden Ressourcen ablegt. In diesem Ordner wird ein Skript namens `phptest.php` abgelegt. Der Inhalt lautet:

```
1 <?php print_r($_SERVER) ?>
```

Nachdem das Skript als ausführbar markiert wurde, kann es mittels

```
1 /usr/bin/php ./phptest.php
```

getestet werden. Damit man in einem Browser den Dump sieht, muss zuerst eine aktive Session gestartet werden (Anmeldung am IDP), dann kann mittels Aufruf der Adresse `https://shib2.srv.lrz.de/secure-all/phptest.php` das Skript aus einem Browser heraus aufgerufen werden.

Anhang 3: Konfiguration von Ansible-Shibboleth

Folgende Dokumentation erläutert, wie Ansible-Shibboleth konfiguriert werden kann, damit automatisiert ein neuer IDP aufgesetzt werden kann. In der folgenden Dokumentation wurde Ansible auf die Maschine `shib3.srv.lrz.de` geladen und auch dort ausgeführt, da die restriktive Konfiguration der LRZ Maschinen eine Verbindung zu einer entfernten Maschine verhindert bzw. eine Lockerung der Sicherheitseinstellungen der entfernten Maschine notwendig gewesen wäre.

1 Vorbereitung des Basissystems

1. Ansible installieren: Hierfür in `/etc/apt/sources.list` folgende Zeile hinzufügen:

```
deb http://ppa.launchpad.net/ansible/ansible/ubuntu trusty main
```

   Jetzt folgende Kommandos für die Installation ausführen:

```
apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 93C4A3FD7BB9C367
apt update
apt install ansible
```

2. PyOpenSSL installieren:

```
apt install python-pip git
```

   Und die für die Ausführung des Ansible-Moduls notwendigen Python-Module nachinstallieren:

```
pip install requests pyopenssl validators
```

3. SSH-Key generieren: Da sich Ansible über SSH auf Port 22 an die lokale Maschine anbindet, wird ein SSH-Key generiert und an die Datei `/root/.ssh/authorized_keys` angehängt:

```
ssh-keygen -b 4096 -t rsa
systemctl restart sshd
```

   In `/root/.ssh/authorized_keys` den Public Key an die Datei anhängen:

```
ssh-rsa <Public Key> root@shib3
```

2 Grundlegende Installation von Ansible-Shibboleth

2.1 Vorbereitung

Zuerst muss das von Marco Malavolti bereitgestellte Github-Repository heruntergeladen werden:

```
1 cd /opt ; git clone https://github.com/GEANT/ansible-shibboleth.git
2 cd /opt/ansible-shibboleth ; git clone https://github.com/GEANT/ansible-
    shibboleth-inventories.git inventories
3 cd /opt/ansible-shibboleth ; git clone https://github.com/GEANT/ans-idpcloud-
    utility.git scripts
```

Jetzt wird das Inventory-File generiert. Hierfür die Datei `/opt/ansible-shibboleth/inventories/test/test.ini` mit Hilfe des Templates `test.ini-template` erstellen:

```
1 # Put here those IdP that should be provided of an Identity Management System
2 [Debian-IdP-with-IdM]
3 shib3.srv.lrz.de
```

Für den verschlüsselten Transport sämtlicher von Ansible übermittelten Dateien, wird ein Passwort generiert und in `.vault_pass.txt` gespeichert:

```
1 cd /opt/ansible-shibboleth/
2 openssl rand -base64 64 > .vault_pass.txt
```

Im nächsten Schritt wird die Shibboleth-IDP Software heruntergeladen, welche Ansible auf dem Zielsystem installieren soll:

```
1 cd /usr/local/src
2 wget https://shibboleth.net/downloads/identity-provider/3.4.3/shibboleth-
    identity-provider-3.4.3.tar.gz
3 tar xzf /usr/local/src/shibboleth-identity-provider-3.4.3.tar.gz
4 rm -f /usr/local/src/shibboleth-identity-provider-3.4.3.tar.gz
```

2.2 Installationsskript

Jetzt kann das Installationsskript gestartet werden. Mit dessen Hilfe werden alle für Ansible-Shibboleth benötigten Dateien erzeugt:

```
1 cd /opt/ansible-shibboleth/scripts/createIdp
2 python createIdp.py shib3.srv.lrz.de --everything
```

Bei der Installation wird bereits vieles abgefragt, das später nicht mehr händisch nachgetragen werden muss:

- Institution Name for ITALIAN language: shib3.srv.lrz.de
- Institution Name for ENGLISH language: shib3.srv.lrz.de
- Insert the Institution domain: srv.lrz.de
- Insert the Institution site for the ITALIAN language: https://shib3.srv.lrz.de/
- Insert the Institution site for the ENGLISH language: https://shib3.srv.lrz.de/
- Insert the URL HTTPS of the Institution Logo (160x120) for the ITALIAN language (press Enter to keep the default value):

- Insert the URL HTTPS of the Institution Logo (160x120) for the ENGLISH language (press Enter to keep the default value):
- Insert the URL HTTPS of the Institution Favicon (32x32) for the ITALIAN language (press Enter to keep the default value):
- Insert the URL HTTPS of the Institution Favicon (32x32) for the ENGLISH language (press Enter to keep the default value):
- Insert the hexadecimal color of the institution (press Enter to generate a random value):
- Insert Institution IdP description for the ITALIAN language (press Enter to keep the default value):
- Insert Institution IdP description for the ENGLISH language (press Enter to keep the default value): Test IdP Setup
- Insert the URL of the Privacy Policy page valid for the Institution in ITALIAN language (press Enter to keep the default value):
- Insert the URL of the Privacy Policy page valid for the Institution in ENGLISH language (press Enter to keep the default value):
- Insert the URL of the Information page valid for the Institution in ITALIAN language (press Enter to keep the default value):
- Insert the URL of the Information page valid for the Institution in ENGLISH language (press Enter to keep the default value):
- Insert the User Support e-mail address for the Institutional IdP (press Enter to keep the default value 'idpcloud-service@example.org'): idp-admin@lrz.de
- Insert your institution address (press Enter to provide it later):
- Insert 'Debian-IdP-with-IdM' or 'Debian-IdP-without-IdM': (press 'Enter' for 'Debian-IdP-with-IdM'): Debian-IdP-with-IdM
- Insert the URL where the CA PEM certificate, used to generate SSL Key and Certificate of the IdP, can be retrieved. Digit the CA URL here:http://cdp1.pca.dfn.de/global-root-g2-ca/pub/cacert/cacert.pem
- Insert the persistent-id salt (press Enter to generate a random value):
- Insert the f-ticks salt (press Enter to generate a random value):
- Insert the username of the user who will have access to the IdP IDM (press Enter to keep the default value 'idm-admin'): admin
- Insert the password of the user who will have access to the IdP IDM (press Enter to generate a random value):
- Insert the openLDAP root password (press Enter to generate a random value):
- Insert the MySQL root password (press Enter to generate a random value):

- Insert the 'shibboleth' user password (press Enter to generate a random value):
- Insert the 'idpuser' user password (press Enter to generate a random value):
- Insert the 'statistics' user password (press Enter to generate a random value): IDM User: admin IDM Password:

Schließlich noch in das Playbook `/opt/ansible-shibboleth/shib-idp-idm-servers.yml` unseren Zielserver `shib3.srv.lrz.de` eintragen. Die Rolle "phpldapadmin" kann entfernt werden:

```
1  ---
2  # file: shib-idp-idm-servers.yml
3  - hosts: shib3.srv.lrz.de
4    become: yes
5    become_method: sudo
6    remote_user: root
7    roles:
8      - common
9      - apache
10     - jdk
11     - jetty
12     - openldap
13     - mysql
14     - idp
15     - sys-update
16   serial: 3
```

3 Konfiguration

In den nächsten Schritten werden die einzelnen im soeben geänderten Playbook definierten Rollen an das Testsystem angepasst. Die Anpassungen der Rollen gelten für jeden zukünftig installierten IDP. Für spezielle Konfigurationen einzelner Hosts ist auf das Inventar im nächsten Abschnitt verwiesen.

3.1 Rollen

Common Die Rolle "Common" versucht das Root-User Passwort zu ändern und den Root-Login via SSH zu verhindern. Dies soll jedoch verhindert werden, weshalb die entsprechenden Tasks in `/opt/ansible-shibboleth/roles/common/tasks/main.yml` auskommentiert werden. Zudem sind die Tasks für eine Cloud überflüssig:

```
1  # Root-User PW soll nicht gesetzt werden
2  #- name: "Add root user's password"
3  #  user:
4  #   name: "root"
5  #   password: "{{ common['root_user_pw'] | password_hash('sha512', 65534 |
       random(seed=inventory_hostname) | string) }}"
6  #  tags: common
7
8  # Root-Login soll möglich sein
9  #- name: "Disable Root SSH Login access"
10 #  lineinfile:
11 #   dest: "/etc/ssh/sshd_config"
```

```
12 #    regexp: "^PermitRootLogin"
13 #    insertafter: "^#PermitRootLogin prohibit-password"
14 #    line: "PermitRootLogin no"
15 #    state: present
16 #   notify:
17 #    - "Restart SSH"
18 #   tags: common
19
20 # Cloud ist nicht benötigt
21 #- name: "Check the existance of cloud-init configuration"
22 #   stat:
23 #     path: /etc/cloud/cloud.cfg
24 #   register: cloud_cfg
25 #   tags: common
26
27 #- name: "Comment out 'manage_etc_host' from /etc/cloud/cloud.cfg"
28 #   lineinfile:
29 #    dest: /etc/cloud/cloud.cfg
30 #    regexp: "manage_etc_hosts"
31 #    line: "#manage_etc_hosts"
32 #    state: present
33 #   when:
34 #    - cloud_cfg.stat.exists
35 #   tags: common
```

Apache Für Apache muss die SSL-Konfiguration des Virtual Host angepasst werden, über welche der IDP später erreichbar sein wird. Es müssen die korrekten Zertifikatswerte des Serverzertifikats von Shib3 eingetragen werden, wobei die Variable fqdn bei der Ausführung von Ansible den Wert `shib3.srv.lrz.de` erhält:

```
1  #   A self-signed (snakeoil) certificate can be created by installing
2  #   the ssl-cert package. See
3  #   /usr/share/doc/apache2/README.Debian.gz for more info.
4  #   If both key and certificate are stored in the same file, only the
5  #   SSLCertificateFile directive is needed.
6  #
7  #   Insert LRZ certificate values
8  SSLCertificateFile    /etc/ssl/localcerts/{{ fqdn }}.pem
9  SSLCertificateKeyFile /etc/ssl/localkeys/{{ fqdn }}-nopw.key
10
11 #   Server Certificate Chain:
12 #   Point SSLCertificateChainFile at a file containing the
13 #   concatenation of PEM encoded CA certificates which form the
14 #   certificate chain for the server certificate. Alternatively
15 #   the referenced file can be the same as SSLCertificateFile
16 #   when the CA certificates are directly appended to the server
17 #   certificate for convinience.
18 #
19 #   LRZ CA chain:
20 SSLCertificateChainFile /etc/ssl/chains/lrz-cachain-telesec-class2.pem
```

OpenLDAP Zur Konfiguration von LDAP in `/opt/ansible-shibboleth/roles/openldap/files/ldap.conf` folgende Defaultwerte eintragen:

```
1 #
```

```
2  # LDAP Defaults
3  #
4
5  # See ldap.conf(5) for details
6  # This file should be world readable but not world writable.
7
8  BASE     dc=srv,dc=lrz,dc=de
9  URI      ldap://127.0.0.1
10
11 #SIZELIMIT      12
12 #TIMELIMIT      15
13 #DEREF          never
14
15 # TLS certificates (needed for GnuTLS)
16 TLS_CACERT      /etc/ssl/chains/lrz-cachain-telesec-class2.pem
17 TLS_REQCERT allow
```

IDP In dieser Rolle werden sämtliche Konfigurationen für den Shibboleth IDP vorgenommen.

Folgender Task wird nicht benötigt und kann daher auskommentiert werden in `/opt/ansible-shibboleth/roles/idp/tasks/main.yml`:

```
1 #- name: "Configure Shibboleth IdP /statistics application"
2 #  import_tasks: idp-statistics.yml
3 #  tags: idp
```

1. Java-Pfad: Da der korrekte JAVA-Pfad für die Ausführung des IDP unabdingbar ist, muss in `/opt/ansible-shibboleth/roles/idp/vars/Debian.yml` der korrekte Pfad gesetzt werden:

```
1 ---
2 # Default Debian Variables
3 java_home_dir: "/usr/lib/jvm/java-8-openjdk-amd64/jre"
```

2. IDP-Version: Damit der automatisiert installierte IDP die selbe Shibboleth-Version verwendet wie der manuell installierte IDP, wird in `/opt/ansible-shibboleth/roles/idp/vars/shib-idp-vars.yml` die Version auf 3.4.3 gesetzt. Ebenfalls muss die Checksum angepasst werden:

```
1 ---
2 ### Variables for Shibboleth Identity Provider (IDP)
3
4 shib_idp_version: "3.4.3"
5 idp_dl_url: "https://shibboleth.net/downloads/identity-provider/{{
      shib_idp_version }}/shibboleth-identity-provider-{{ shib_idp_version
      }}.tar.gz"
6 # SHA256 checksum
7 shib_idp_checksum: "
      eb86bc7b6366ce2a44f97cae1b014d307b84257e3149469b22b2d091007309db"
```

3. Anpassung MySQL-DB Skript: Der Datenbank wird zusätzlich der User “shibboleth” hinzugefügt und die nötigen Rechte zugeteilt:

```
CREATE DATABASE IF NOT EXISTS shibboleth CHARACTER SET=utf8;

# User hinzugefügt
CREATE USER 'shibboleth'@'localhost' IDENTIFIED BY '{{ idp_config['
    shibboleth_db_password'] }}';

GRANT ALL PRIVILEGES ON shibboleth.* TO shibboleth@localhost IDENTIFIED
    BY '{{ idp_config['shibboleth_db_password'] }}';
```

4. Lokale Verwaltung der Metadaten: Da bisher noch keine Bereitstellung von Remote-Metadaten möglich ist, wird das Template `/opt/ansible-shibboleth/roles/idp/templates/conf/metadata-providers.xml.j2` entsprechend angepasst: Der HTTPMetadataProvider wird auskommentiert:

```
    <!--
    Example HTTP metadata provider.  Use this if you want to download the
        metadata
    from a remote source.

    You *MUST* provide the SignatureValidationFilter in order to function
        securely.
    Get the public key from the party publishing the metadata, and
        validate it
    with them via some out of band mechanism.

    The EntityRoleWhiteList saves memory by only loading metadata from
        SAML roles
    that the IdP needs to interoperate with.
    -->

{# commented out for local metadata file
   {% for md in idp_metadata_providers %}
    <MetadataProvider
      id="{{ md['id'] }}"
      xsi:type="FileBackedHTTPMetadataProvider"
      backingFile="%{idp.home}/metadata/{{ md['file'] }}"
      metadataURL="{{ md['url'] }}"
      disregardTLSCertificate="{{ md['disregardTLSCertificate'] }}">

   {% if md['pubKey'] is defined %}
      <!--
          Verify the signature on the root element of the metadata
              aggregate
          using a trusted metadata signing certificate.
      -->
      <MetadataFilter xsi:type="SignatureValidation" requireSignedRoot="
          true">
         <PublicKey>
            {{ md['pubKey'] }}
         </PublicKey>
      </MetadataFilter>

   {% endif %}
 {% if md['maxValidInterval'] is defined %}
      <!--
          Require a validUntil XML attribute on the root element and make
              sure its value is no more than 5 days into the future.
```

```
37          -->
38          <MetadataFilter xsi:type="RequiredValidUntil" maxValidityInterval="
                {{ md['maxValidInterval'] }}" />
39
40      {% endif %}
41          <!-- Consume all SP metadata in the aggregate -->
42          <MetadataFilter xsi:type="EntityRoleWhiteList">
43            <RetainedRole>md:SPSSODescriptor</RetainedRole>
44          </MetadataFilter>
45       </MetadataProvider>
46      {% endfor %}
47  #}
```

Dafür wird das Bean mit den lokalen Metadaten entkommentiert. Die Metadaten sollen unter `/opt/shibboleth-idp/conf/federation-metadata.xml` abgelegt werden:

```
1  <MetadataProvider id="LocalMetadata"  xsi:type="
       FilesystemMetadataProvider" metadataFile="/opt/shibboleth-idp/conf/
       federation-metadata.xml" />
```

5. Zertifikate: Folgende Zeilen müssen in `/opt/ansible-shibboleth/roles/idp/tasks/idp-configure.yml` auskommentiert werden, da Ansible hier versucht, Zertifikate zu kopieren, die nicht existieren. Hier sollten vertrauenswürdige Zertifikate vom LRZ verwendet werden:

```
1  ---
2  # tasks file for IdP Configuration
3  #- name: "Configure Shibboleth IdP credentials"
4  #  copy:
5  #    src: "{{ item.src }}"
6  #    dest: "{{ item.dest }}"
7  #    owner: "jetty"
8  #    group: "root"
9  #    mode: "{{ item.mode }}"
10 #  with_items:
11 #    - { src: "{{ files_dir }}/{{ fqdn }}/idp/credentials/idp-signing.crt
       ", dest: "/opt/shibboleth-idp/credentials/idp-signing.crt", mode:
       "0644" }
12 #    - { src: "{{ files_dir }}/{{ fqdn }}/idp/credentials/idp-signing.key
       ", dest: "/opt/shibboleth-idp/credentials/idp-signing.key", mode:
       "0600" }
13 #    - { src: "{{ files_dir }}/{{ fqdn }}/idp/credentials/idp-encryption.
       crt", dest: "/opt/shibboleth-idp/credentials/idp-encryption.crt", mode
       : "0644" }
14 #    - { src: "{{ files_dir }}/{{ fqdn }}/idp/credentials/idp-encryption.
       key", dest: "/opt/shibboleth-idp/credentials/idp-encryption.key", mode
       : "0600" }
15 #    - { src: "{{ files_dir }}/{{ fqdn }}/idp/credentials/idp-backchannel.
       crt", dest: "/opt/shibboleth-idp/credentials/idp-backchannel.crt",
       mode: "0644" }
16 #    - { src: "{{ files_dir }}/{{ fqdn }}/idp/credentials/idp-backchannel.
       p12", dest: "/opt/shibboleth-idp/credentials/idp-backchannel.p12",
       mode: "0644" }
17 #    - { src: "{{ files_dir }}/{{ fqdn }}/idp/credentials/sealer.jks",
       dest: "/opt/shibboleth-idp/credentials/sealer.jks", mode: "0644" }
18 #    - { src: "{{ files_dir }}/{{ fqdn }}/idp/credentials/sealer.kver",
       dest: "/opt/shibboleth-idp/credentials/sealer.kver", mode: "0644" }
```

Jetzt muss in der selben Datei der Download der Föderationsmetadaten auskommentiert werden, da dies aktuell nicht funktioniert.

```
---
#- name: "Download Federation's Metadata files"
#  get_url:
#   url: "{{ item.url }}"
#   dest: "/opt/shibboleth-idp/metadata/{{ item.file }}"
#   owner: "jetty"
#   group: "root"
#   mode: "0644"
#   validate_certs: "{{ item.disregardTLSCertificate }}"
#  with_items: "{{ idp_metadata_providers }}"
```

4 Inventar

Im folgenden Abschnitt wird auf den Inventar-Ordner eingegangen, in welchem die speziellen Konfigurationsdateien für jeden einzelnen Zielhost angegeben werden können.

4.1 Files

Im Inventar können für jeden einzelnen Host Dateien abgelegt werden. In diesem Fall wurde ein Ordner `shib3.srv.lrz.de` vom Install-Skript erstellt. Hier werden statische Dateien wie Zertifikate, aber auch zum Test z.B. der Attribute Resolver von Shib1 abgelegt.

1. Zertifikate: Das Serverzertifikat und der zugehörige private Schlüssel von Shib3 müssen unter `/opt/ansible-shibboleth/inventories/files/shib3.srv.lrz.de/common/ssl/` kopiert werden, damit sie an die korrekte Stelle verteilt werden können:

```
cd /opt/ansible-shibboleth/inventories/files/shib3.srv.lrz.de/common/ssl/

cp /etc/ssl/localcerts/shib3.srv.lrz.de.pem .
cp /etc/ssl/localkeys/shib3.srv.lrz.de-nopw.key .
cp /etc/ssl/chains/lrz-cachain-telesec-class2.pem .
```

2. Test Attribute Resolver und Attribute Filter: Testweise werden unter `/opt/ansible-shibboleth/inventories/files/shib3.srv.lrz.de/idp/conf/` der `attribute-resolver.xml` und der `attribute-filter.xml` abgelegt. Achtung: Die Dateien werden im Zielsystem umbenannt zu `attribute-resolver-v3-custom.xml` und `attribute-filter-custom.xml`!

4.2 Test-Inventory

Das Absible-Modul bietet drei unterschiedliche Inventare an: Test, Development und Production. Für die Verwendung im Testbed bietet sich der Test Ordner an. Im Test-Ordner unter `/opt/ansible-shibboleth/inventories/test/` finden sich folgende Dateien:

- `test.ini`: Hier werden alle zu konfigurierenden Hosts eingetragen (vgl. vorige Kapitel).
- `group_vars`: Hier können Variablen eingetragen werden, die alle Hosts betreffen sollen. In unserem Fall können hier z.B. in `../group_vars/all.yml` die NTP-Server des LRZ eingetragen werden:

```
# NTP Server
ntp_servers:
 - ntp1.lrz.de
 - ntp2.lrz.de
 - ntp3.lrz.de
```

An dieser Stelle können aber auch Syslog-Server sowie Nameserver angegeben werden.

- `host_vars`: Hier werden sämtliche Angaben zur Konfiguration der einzelnen Hosts gemacht. Aufgrund des Umfangs der Konfiguration wird dies im nächsten Abschnitt dargestellt.

4.3 Die Datei shib3.srv.lrz.de.yml

Unter `host_vars` wird für jeden Host eine eigene YAML-Datei angelegt. Für den Testfall auf Shib3 ist dies also `shib3.srv.lrz.de.yml`. Im Installationsskript wurden bereits sämtliche Daten abgefragt, jedoch wird das entsprechende Konfigurationsfile im Production-Ordner unter `/opt/ansible-shibboleth/inventories/production/host_vars/shib3.srv.lrz.de.yml` abgelegt. Es kann die entsprechende Datei in den Test-Ordner kopiert werden.

Am Ende sollte sie folgendermaßen aussehen (es werden stets nur die zu ändernden Abschnitte angegeben):

```
---
# file: host_vars/idp.example.org.yml

# idp_config['idp_sourceAttribute']:
# MUST BE an attribute, or a list of comma-separated attributes,
# that uniquely identify the subject of the generated persistent-id.
# It MUST BE: Stable, Permanent and Not-reassignable
#
# idp_config['idp_persistentId_salt']:
# generated by 'openssl rand -base64 36'
#
# idp_config['fticks']['salt']:
# generated by 'openssl rand -base64 48'


### Machine Variables
fqdn: "shib3.srv.lrz.de"
files_dir: "/opt/ansible-shibboleth/inventories/files"

### Common Variables
### For self-signed certs use ca: "{{ fqdn }}.crt"
common:
  ssl: "True"
#ca: "cacert.crt"
  ca: "lrz-cachain-telesec-class2.pem"
# ssl_cert: "{{ fqdn }}.crt"
  ssl_cert: "{{ fqdn }}.pem"
  ssl_key: "{{ fqdn }}-nopw.key"
  # Add(present) or Remove(absent) SWAP file.
  # Remove entirely "swap" section if you use a dedicated partition for SWAP or
      if don't need it.
  swap:
    name_swapfile: "swapfile"
```

```
33    size_swapfile: "2048"
34    state: present
35   # Remove entirely "mirror" section to use the default distribution
         repositories
36  # mirror: "https://mi.mirror.garr.it/mirrors/debian/"
37
38  ### NTP Variables
39  my_timezone: Europe/Berlin
40
41  ### Apache Variables
42  apache:
43   admin_email: "{{ idp_contacts['technical']['mail'] }}"
44
45  ### JDK Variables - Default "openjdk 8"
46  jdk_type: "openjdk"
47  jdk_version: "8"
48
49  ### LDAP Variables
50  idp_ldap_restore: "false"
51
52  ldap:
53   basedn: "dc=srv,dc=lrz,dc=de"
54   domain: "srv.lrz.de"
55   org: "LRZ Test Shib"
56   url: "ldap://127.0.0.1:389"
57   root_dn: "cn=admin,dc=srv,dc=lrz,dc=de"
58   root_pw: "test-1234"
59   create_test_user: "yes"
60
61  ### MySQL Variables
62  idp_db_restore: "false"
63
64  mysql:
65   root_password: "test-1234"
66
67  ### RSYSLOG Server
68  rsyslog_srv:
69   ip: "## RSYSLOG SRV IP FROM INTERNAL NETWORK OR {{ rsyslog_server_ip }} FROM
         'group_vars'##"
70   port: "20514"
71
72  ### IDP Variables
73  idp_config:
74   sealer_pw: "sealer-password"
75   keystore_pw: "sealer-password"
76   shibboleth_db_password: "root-db-password"
77   scope: "example.org"
78   idp_sourceAttribute: "uid"
79   idp_persistentId_salt: "## RANDOM LONG STRING 1 ##"
80   ldap:
81    # Testweise eingefügt, da angeblich nicht definiert
82    url_password_reset: "https://pwreset.lrz.de/"
83    authenticator: "bindSearchAuthenticator"
84    url: "ldap://127.0.0.1:389"
85    useStartTLS: "false"
86    useSSL: "true"
87    connectTimeout: "3000"
```

```
88    sslConfig: "certificateTrust"
89    trustCertificates: "/etc/ssl/certs/cacert.pem"
90    baseDN: "ou=people,{{ ldap['basedn'] }}"
91    subtreeSearch: "true"
92    userFilter: "(uid={user})"
93    bindDN: "cn=admin,{{ ldap['basedn'] }}"
94    bindDNCredential: "test-1234"
95    searchFilter: "(uid=$resolutionContext.principal)"
96    returnAttributes: "*"
97    ppolicy:
98     change_password_url: "{{ url_password_reset }}"
99   fticks:
100   federation: "IDEM GARR AAI"
101   alg: "SHA-256"
102   salt: "## RANDOM_LONG_STRING_2 ##"
103   loghost: "localhost"
104   logport: "514"
105  sup_rs: "yes"
106  sup_coco: "yes"
```

4.4 Ausführung Ansible

Um Ansible auf dem lokalen System und dem soeben konfigurierten Test-Ordner auszuführen, folgenden Befehl starten:

```
1 ansible-playbook -vvv site.yml -i inventories/test/test.ini --limit shib3.srv.
    lrz.de --vault-password-file .vault_pass.txt
```

Zur Verschlüsselung wird das definierte Passwort verwendet.

Möchte man Ansible auf sämtlichen in der `test.ini`-Datei definierten Hosts ausführen, wird folgender Befehl verwendet:

```
1 ansible-playbook site.yml -i inventories/test/test.ini --vault-password-file .
    vault_pass.txt
```