Einsatz von Portal Delegation und SAML Assertions bei der Authentifizierung und Autorisierung

Projekt „Nutzung von kurzlebigen Zertifikaten in Portalbasierten Grids“

– Förderkennzeichen 01IG09003 –

„Service Grids für Forschung und Entwicklung“
des Bundesministeriums für Bildung und Forschung (BMBF)

Arbeitspakete:	Tasks 1, 3 und 4
Autoren:	Stefan Pinkernell, Bernadette Fritzsch (AWI)
Version:	1.0
Publikationsdatum:	24.02.2011
Kontakt:	Stefan Pinkernell
Email:	Stefan.Pinkernell@awi.de

GEFÖRDERT VOM

Bundesministerium
für Bildung
und Forschung

1 Einführung

Der Zugang zu D-Grid-Ressourcen beschränkt sich derzeit auf eine zertifikatbasierte Authentifizierungs- und Autorisierungs-Infrastruktur, in der nur X.509-Zertifikate einer EUGridPMA akkreditierten Zulassungsstelle akzeptiert werden (vgl. [3], [6]). Dies bietet für weite Nutzerkreise eine hohe Einstiegsschwelle in das Grid, da dieses Verfahren aufgrund des umständlichen Handlings der Zertifikate, des teilweise mehrfach wiederholten persönlichen Ausweisens bei der Beantragung und Verlängerung, der z.T. fehlenden regionalen Authorities, den Problemen bei der Zertifikatkonvertierung, etc. zu kompliziert ist. Zudem ist dieses Verfahren problematisch bei der Verwendung von Web-basierten Zugangsportalen.

Für portalbasierte Grids kann die Authentifizierung für den Nutzer wesentlich vereinfacht werden. Bei der Portal Delegation [16] werden die Prozesse im Auftrage des Nutzers durch das Portal erledigt und somit für den Nutzer transparent gemacht. Es werden kurzlebige Zertifikate (Short Lived Certificate, SLC) verwendet, die jeweils erst bei Bedarf von einer Online-CA ausgestellt werden [7]. Dazu ist von Nutzerseite nur ein Login mit Nutzername und Passwort notwendig, das sie von ihren Heimateinrichtungen her kennen. Voraussetzung dafür ist, dass die Heimateinrichtung Mitglied in der DFN-AAI ist und einen IdentityProvider betreibt. Hauptvorteil für den Nutzer ist der entfallende Verwaltungsaufwand für das Zertifikat.

Neben der Authentifizierung kann auch die Autorisierung durch das Portal vereinfacht werden. Mittels Attributen kann der Zugriff auf Grid-Ressourcen feingranular geregelt werden. Bei den meisten Grid-Middleware-Systeme (Globus, gLite, UNICORE) kann dies mittels SAML (**S**ecurity **A**ssertion **M**arkup **L**anguage) Assertions (vgl. Kap. 2.2) realisiert werden. Das Projekt Gridshib for Globus Toolkit hat hier schon erste gute Vorarbeiten geleistet (vgl. [12],[13]). Darauf aufbauend werden hier in einem neuen Anwendungsfall sowohl Campus Attribute als auch Rollen-Informationen aus einer Virtuellen Organisation herangezogen, die im Portal im Namen des Nutzers zu einer neuen SAML Assertion zusammengefasst und in das vom SLC abgeleitete Proxy-Zertifikat eingebettet werden.

Dieses Dokument fasst die durchgeführten Arbeiten zusammen und soll die Nachnutzung der entwickelten Komponenten erleichtern. Im ersten Teil werden die zugrundeliegenden Dienste vorgestellt und beschrieben. Danach folgt die Beschreibung der implementierten Software, um an einem Portal das SLC sowie die Campus- und VO-Attribute zu beziehen, die neue, signierte SAML Assertion auszustellen und in ein Proxy Zertifikat einzubetten. Dies schließt auch die Modifikation einer von Gridshib for Globus Toolkit verwendeten Bibliothek zur Auswertung der neu erstellten, portalsignierten SAML Assertion an der Grid Ressource ein. Die hier vorgestellte Lösung ermöglicht mit wenigen Mausklicks, per Shibboleth geschützt die notwendigen Zertifikate und Attribute über eine Weboberfläche zu beziehen und alle für die Autorisierung an einer Grid-Ressource notwendigen Attribute als SAML Assertion in ein Proxy-Zertifikat einbetten zu lassen. Weitere Informationen finden sich zudem in der Dokumentation zu den Projekt-Tasks 1 und 3 [1][2].

2 Grundlagen zu den Diensten

2.1 Der DFN-SLCS

Der DFN bietet mit dem DFN-SLCS [7] seit 2009 (als DFN-Test-SLCS bereits seit 2007) einen Dienst zur Ausgabe von kurzlebigen X.509 Zertifikaten an, der auf der Authentifizierungs- und Autorisierungs-Infrastruktur DFN-AAI (siehe [5]) basiert. Die Gültigkeitsdauer dieser Zertifikate ist auf maximal 1.000.000 Sekunden begrenzt, was etwa 11,5 Tagen entspricht. Beantragt werden diese Zertifikate online. Dazu enthält der SLCS eine Online Zertifizierungsstelle, die die kurzlebigen Zertifikate automatisch ausstellen kann, sowie Online-Schnittstellen um Zertifikat-Requests (CSR) einreichen und kurzlebige Zertifikate direkt ausliefern zu können.

Derzeit existieren beim DFN zwei von der technischen Implementierung her identische Varianten dieses Dienstes:

1. Der EUGridPMA-akkreditierte SLCS der DFN-PKI: Die Nutzung dieses SLCS ist an einige Voraussetzungen gebunden. So muss die Person einen Eintrag in einem in die DFN-AAI integrierten Identity Provider (IdP) haben, wo für sie der URN "urn:geant:dfn.de:dfn-pki:slcs" im Attribut "eduPersonEntitlement" eingetragen ist. Weiterhin sind an die Einrichtung und den Betrieb der SLCS-RA strenge Anforderungen gestellt (vgl. [6]).
2. Daneben gibt es noch einen nicht akkreditierten SLCS, der als Test-System in der Test-AAI verfügbar ist.

Da ein kurzlebiges Zertifikat aus dem akkreditierten SLCS eine vergleichbare Vertrauensbasis wie ein persönliches Zertifikat darstellt und daher von den Ressourcenprovidern als gleichwertig behandelt wird, sind bei seiner Handhabung ähnlich strenge Anforderungen zu erfüllen. So muss der private Schlüssel zu den kurzlebigen Zertifikaten ausschließlich der direkten Kontrolle des jeweiligen Nutzers unterliegen und darf niemals ungeschützt und dauerhaft in einem Portal abgelegt werden.

Bezug des SLC mit dem Credential Retriever

Auch wenn der Fokus dieses Dokuments auf der Portal Delegation liegt, soll hier zunächst der konventionelle Bezug von SLCs beschrieben werden, da daran die Komplexität der ablaufenden Prozesse relativ einfach dargestellt werden kann. Abbildung 1 zeigt den schematischen Ablauf des webbasierten SLC-Bezugs.

Zunächst ruft der Nutzer die Website des DFN-SLCS[1] auf (1). Von dort aus wird er zunächst zum Where-Are-You-From-Service (WAYF) weitergeleitet (2) und selektiert dort aus einer Liste seine Heimateinrichtung, zu dessen Identity-Provider-Seite er daraufhin weitergeleitet wird (3). Dort authentifiziert er sich mit seinem Nutzernamen und Passwort. Nach erfolgreicher Authentifizierung wird er auf die Website des DFN-SLCS zurückgeleitet (4), wo nun ein Zugang möglich ist. Von dieser Seite aus kann dann eine Java-Webstart-Applikation (Gridshib CA Credential Retriever) gestartet werden, die auf dem Rechner des Nutzers ein Schlüsselpaar und einen Zertifikat-Request erzeugt (5). Dieser Zertifikatrequest wird an die Online CA gesendet (6). Das kurzlebige Zertifikat wird dann zusammen mit dem privaten Schlüssel im Dateisystem des Rechners des Nutzers gespeichert und kann innerhalb seiner Gültigkeit vom Nutzer in gleicher Weise wie ein langlebiges Zertifikat genutzt werden

[1] Test-SLCS: https://test-slcs.pca.dfn.de/cgi-bin/login.cgi
SLCS: https://slcs.pca.dfn.de/cgi-bin/login.cgi

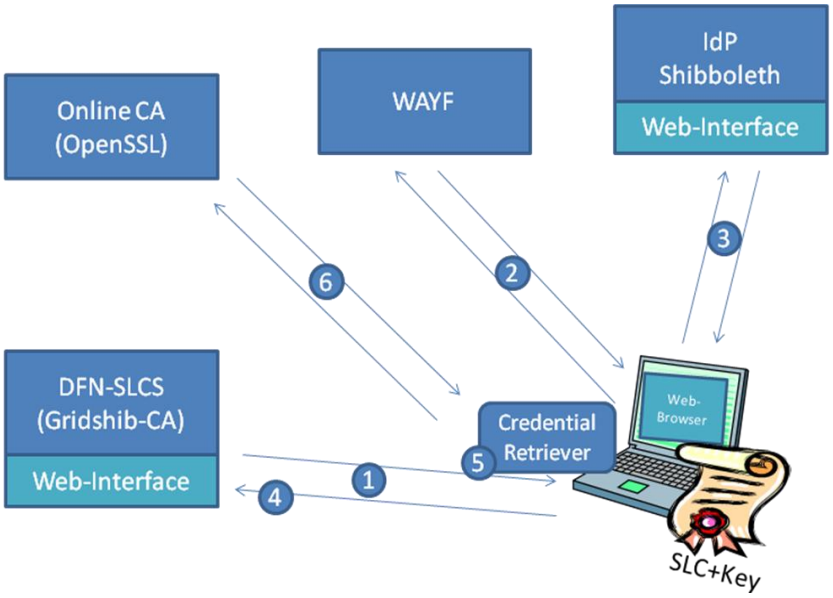

Abbildung 1: Bezug eines SLC über das Web-Interface

2.2 SAML

SAML ist ein ab 2001 von OASIS entwickeltes XML Framework zur Übertragung von Informationen zur Authentifizierung und Autorisierung [18]. Hauptanwendungsgebiet neben Single-Sign-On Systemen sind Autorisierungsdienste.

In der SAML Spezifikation sind SAML-Assertions, SAML-Protokoll, SAML-Bindings und SAML-Profile definiert. Für diesen Task wird nur ein Teil der Spezifikation genutzt: SAML Assertions, in denen die Attribute enthalten sind.

Es existieren zwei Haupt-Versionen: SAML 1 [19], [20] und SAML 2 [21], [22]. Die Version 2 setzt sich mittlerweile immer mehr durch. Eine wachsende Anzahl von Identity Providern fährt diese Version und stellt damit die Campusattribute aus der Shibboleth-Umgebung zur Verfügung. Jedoch gibt es daneben auch noch einige Installationen von SAML 1 und insbesondere die verwendete Version von Gridshib for Globus Toolkit benötigt noch SAML Assertions Version 1.

OpenSAML ist eine Bibliothek für die Sprachen Java und C++, um SAML zu verarbeiten [26]. Es existieren mehrere Entwicklungszweige dieser Bibliothek: OpenSaml Version 1 ist nur in der Lage, SAML 1 Assertions zu verarbeiten. Das Projekt Gridshib verwendet eine modifizierte Version von OpenSAML 1, die von den Gridshib-Entwicklern speziell angepasst wurde. Erst mit der Version 2 von OpenSAML, bei der es sich um eine komplette Neuentwicklung handelt, ist es möglich, sowohl SAML 1 als auch SAML 2 Assertions zu verarbeiten.

Um einerseits die Vorarbeiten von Gridshib nachnutzen und andererseits die immer weiter verbreiteten SAML2 Installationen berücksichtigen zu können, wurde hier eine hybride Lösung unter Verwendung von OpenSaml 2.3.1 implementiert, die sowohl die Informationen aus SAML 2 Assertions extrahieren als auch eine neue SAML 1 Assertion erstellen kann.

2.3 Shibboleth & Campus Attribute

Shibboleth bietet ein Verfahren zur verteilten Authentifizierung und Autorisierung auf Basis von SAML im Umfeld der Webservices und Webanwendungen [11]. Bei diesem Single-Sign-On-Verfahren muss sich

der Nutzer zu Beginn der Session bei seiner Heimateinrichtung anmelden und kann dann ohne erneute Anmeldung auf teilnehmende Dienste, auch anderer Anbieter, zugreifen.

Ein Shibboleth-System besteht dabei aus drei unabhängigen Teilen: Die Heimateinrichtungen stellen zumindest einen Identity Provider bereit, an dem sich die Nutzer anmelden können. Anbieter von Inhalten nutzen den Service Provider, um eine Autorisierungsentscheidung für ihre Dienste treffen zu können. Zudem existiert ein Discovery Service (WAYF – Where are you from), der den noch nicht authentifizierten Nutzer zur Anmeldung an seine Heimateinrichtung weiterleitet.

Die Campus Attribute von der Heimateinrichtung können den Service Providern zur Verfügung gestellt werden. Dabei kann auf Seiten des Identity Providers detailliert festgelegt werden, welcher Service Provider welche Attribute zu sehen bekommt. Die Attribute können dann zum einen aus den Headern der jeweiligen Session geladen oder gesammelt in Form einer SAML Assertion bereit gestellt werden.

Verwendet wurde der Shibboleth Service Providers 2.2.1 sowie ein Shibboleth Identity Provider Version 2.1.5.

2.4 VOMS

Mit dem Virtual Organisation Membershib Service (VOMS) kann eine Gruppe von Personen, die auch aus unterschiedlichen Organisationen stammen können, zu einer virtuellen Arbeitsgruppe verbunden werden. Den Mitgliedern dieser Arbeitsgruppe können dann über ein Rollen-Konzept bestimmte Attribute zugeordnet werden, die dann z.B. an einer Grid-Ressource zu Autorisierungs- und Authentifizierungszwecken genutzt werden können. Diese Attribute werden ebenfalls in Form einer SAML Assertion bereitgestellt und können mit einer Clientsoftware über einen Webservice abgerufen werden [4].

Verwendet wurde VomsAdmin 2.0.18 aus gLite 3.1.

2.5 Globus Toolkit und Gridshib for Globus Toolkit

In der benutzten Testumgebung wird auf den Grid Ressourcen die Software Globus Toolkit in der Version 4.0.8 als Grid Middleware eingesetzt. Zur Authentifizierung werden x.509 Zertifikate genutzt, wobei alle Nutzer in ein *gridmap-File* eingetragen werden müssen.

Globus Toolkit selbst bietet keine Möglichkeit zur SAML basierten Autorisierung. Dazu wird die Software Gridshib for Globus Toolkit eingesetzt, bei der es sich um ein Plug-In für die Middleware Globus Toolkit handelt [12]. Dieses Plug-In ermöglicht die Nutzung von SAML mit X.509 Zertifikaten und erweitert die Grid Security Infrastructure (GSI, [17]) damit um die Nutzung von Attributen zur attributbasierten Autorisierung [13][14].

Die Gridshib SAML Tools [15] bilden einen weiteren Teil dieser Software. Als Stand-Alone Software eingesetzt werden diverse Tools bereitstellt. Es gibt u. a. ein Tool für die Ausstellung der SAML Assertions und ein Binding Tool zur Verknüpfung der SAML Assertion mit dem Proxy. Teile der Gridshib SAML Tools werden auch als Java-Bibliothek von Gridshib for Globus Toolkit verwendet. Sie mussten im Rahmen dieser Arbeit erweitert werden (vgl. Kap 4).

Globus Toolkit wurde in der Version 4.0.8 eingesetzt, Gridshib for Globus Toolkit 0.6.1 und die Gridshib SAML Tools in der Version 0.5.

3 Portal Delegation

3.1 Konzept

Im Grid-Portal-Szenario übernimmt das Portal das Management der Zertifikate. Der Nutzer meldet sich an einem per Shibboleth geschützten Portal an und kann nach erfolgreichem Login das SLC direkt per Portal Delegation von einer Online CA beziehen. Dieses steht damit samt privatem Schlüssel (nur) am Portal zur Verfügung. In einem zweiten Schritt wird von diesem SLC ein Proxy-Zertifikat abgeleitet, das im weiteren Verlauf vom Portal genutzt werden kann [16]. Optional werden noch weitere benutzerspezifische Attribute aus unterschiedlichen Quellen gesammelt und in das Proxy-Zertifikat eingebettet, wobei die Zusammenfassung der Attribute durch das Portal signiert wird. Die Attribute werden in Form von SAML Assertions aus der Shibboleth Umgebung sowie von einem VOMS-Server bezogen.

Der Ablauf der Portal Delegation gestaltet sich dabei wie folgt (siehe Abbildung 2): Zunächst werden am Portal ein Schlüsselpaar und ein Zertifikat-Request generiert. Durch einen Klick auf die entsprechende Schaltfläche wird der Nutzer auf die Website der Gridshib-CA weitergeleitet und, falls noch nicht geschehen, spätestens an dieser Stelle per Shibboleth über die DFN-AAI Föderation authentifiziert. Nach erfolgreicher Authentifizierung wird ein kurzlebiges Zertifikat ausgestellt und der Nutzer samt Zertifikat an das Portal zurückgeleitet.

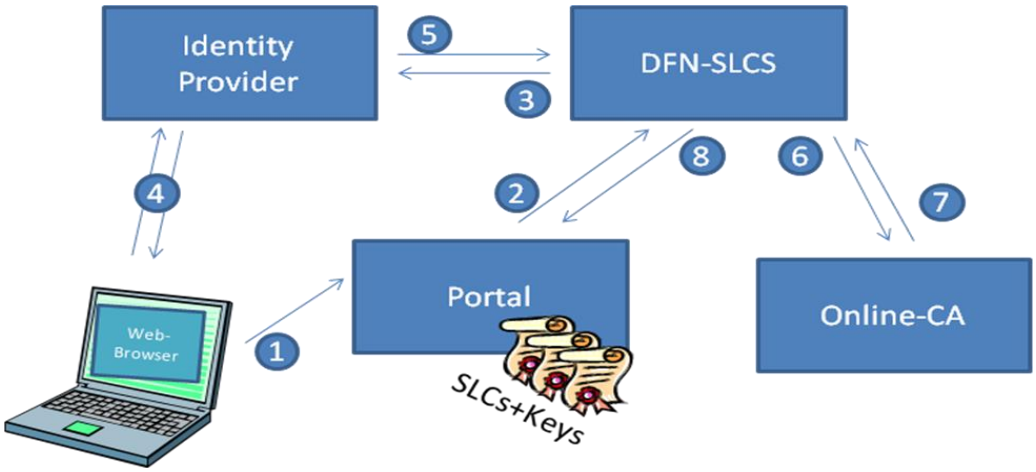

Abbildung 2: Konzept Portal Delegation

Wesentlich an diesem Szenario ist, dass das Portal im Auftrag des Nutzers das SLC holt und das Zertifikat sowie der persönliche Schlüssel auf dem Portalserver liegen. Ein dauerhaftes ungeschütztes Abspeichern des privaten Schlüssels würde dabei jedoch eine Verletzung der Nutzungsbedingungen für den akkreditierten DFN-SLCS darstellen. Daher wurde bei der Implementierung darauf geachtet, das Schlüsselmaterial nur während der Dauer der Session im Arbeitsspeicher vorzuhalten und auf gar keinen Fall ungeschützt persistent zu speichern.

Die hier vorgestellten Arbeiten wurden ausschließlich mit dem DFN-Test-SLCS aus der DFN-Test-AAI durchgeführt. Wegen der Baugleichheit der beiden Dienste sind die Erfahrungen sowie die entwickelte Software aber sofort auch mit dem eigentlichen DFN-SLCS und der akkreditierten DFN-AAI nutzbar.

Um Portal Delegation nutzen zu können, muss das Portal allerdings einige Bedingungen erfüllen:

- Der Zugriff muss per https geschützt sein.
- Das Portal muss ein Schlüsselpaar sowie einen Zertifikat-Antrag erstellen können.

- Der Benutzer muss (per Shibboleth) authentifiziert werden
- Die Session muss von der Erzeugung des Schlüsselpaars bis zu dem Zeitpunkt, an dem das Zertifikat zurückgegeben wird, aufrechterhalten bzw. gespeichert werden können.

Für die Autorisierung an der Grid Ressource sollen sowohl Informationen der Heimateinrichtung als auch aus der Virtuellen Organisation verwendet werden, die jeweils als SAML Attribute bereitgestellt werden. Da die Software Gridshib for Globus Toolkit jedoch nur eine einzelne in ein Proxy-Zertifikat eingebettete SAML Assertion auswerten kann, wird das Trust-Proxy-Verfahren eingesetzt: Im Namen des Nutzers werden durch das Portal sowohl das kurzlebige Zertifikat als auch die Campus- und VO-Attribute gesammelt. Die Attribute stammen damit aus zwei unterschiedlichen Quellen und werden zu einer portalsignierten SAML Assertion zusammengefasst, die in das vom SLC abgeleitete Proxy-Zertifikat eingebettet wird. Details zur Verarbeitung dieser SAML Assertions sind in Kapitel **Fehler! Verweisquelle konnte icht gefunden werden.** beschrieben.

Wesentlich für das Portal Delegation Szenario ist die Veränderung der Vertrauensbeziehungen. Eine Gridressource muss bei der Authentifizierung des Nutzers zum einen der Authority vertrauen, auf die das Zertifikat zurückzuführen ist. Zum anderen muss die Ressource auch dem Portal vertrauen, da die Zusammenstellung der Attribute zur Autorisierung des Nutzers für die jeweiligen Dienste durch das Portal erfolgt, das somit als Attribute-Authority auftritt.

3.2 Bezug des kurzlebigen Zertifikats

3.2.1 Perl-Implementierung

Das Gridshib Projekt bietet eine Implementierung für ein Portal Delegation Szenario in Form von Perl-CGI Scripts an [16]. Diese Software besteht aus zwei Komponenten: Auf Seiten der Online CA wird eine Komponente eingesetzt, die die Zertifikat-Requests entgegennimmt und daraufhin die kurzlebigen Zertifikate ausliefert. Die auf Seiten des DFN-SLCS installierte Software basiert auf dieser Komponente.

Das Portal Delegation Script, das auf Seiten des Portals Verwendung findet, bildet die zweite Komponente. Dieses Script dient dazu, den Zertifikat Request zu erstellen und abzuschicken, sowie das fertige kurzlebige Zertifikat in Empfang zu nehmen.

Da es in der Perl Implementierung kein richtiges Session Management gibt, wird das Script stattdessen über den Parameter *status* gesteuert. Beim ersten Aufruf des Scripts hat dieser Parameter noch keinen Wert. In diesem Fall wird zunächst ein Schlüsselpaar erzeugt und ein Zertifikat-Request erstellt, welcher dann zusammen mit einem String zur Identifikation (*portData*) an den DFN-SLCS gesendet wird. Nach Interaktion mit dem Nutzer (Login beim heimatlichen IdP) sendet der DFN-SLCS das kurzlebige Zertifikat zurück. In diesem Fall hat der Parameter *status* den Wert *success*. Da es sich bei dem Perl-Script um eine Server Anwendung handelt, die natürlich mehrere Nutzer gleichzeitig bedienen kann, müssen die Zertifikate eindeutig den richtigen Schlüsseln bzw. Nutzern zugeordnet werden. Dazu wird der Parameter *portData* verwendet. Darin enthalten ist ein eindeutiger String, der zusammen mit dem Zertifikat-Request an den Server gesendet wurde und auch zusammen mit dem fertigen Zertifikat wieder zurück gesendet wird. Dieser String ist auch Teil des Dateinamens, unter dem der private Schlüssel sowie das Zertifikat abgespeichert werden, und dient zur eindeutigen Zuordnung. Zusätzlich wird getestet, ob das zurückgesendete Zertifikat auch wirklich zum zuvor erstellten Schlüsselpaar passt, indem der Modulus des gespeicherten privaten Schlüssels mit dem des erhaltenen Zertifikats verglichen wird und übereinstimmen muss. Falls kein Zertifikat ausgestellt werden konnte, hat der Parameter *status* den Wert *rejected*, was zur Ausgabe einer Fehlermeldung führt. Jeder Wert des Parameters, außer den erlaubten Werten „success“, „rejected“ oder einem leeren Wert, führt zur Ausgabe einer Fehlermeldung.

3.2.2 Java-Implementierung

Um die eben beschriebene Funktionalität als Webanwendung nutzen zu können, wurde ein Java-Servlet implementiert. Die fertige Software kann relativ einfach als Portlet in einer Portal-Software wie Liferay oder GridSphere genutzt werden, was nur wenige Veränderungen erfordert (vgl. Kap. **Fehler! Verweisquelle konnte nicht gefunden werden.**).

Ein Servlet wird in der Regel über die Methode *doPost()* oder *doGet()* aufgerufen. In allen Ausbaustufen der Software sollen beide Aufrufe unterstützt werden, weswegen beide Methoden den Aufruf an die Methode *processRequest()* weiterleiten, die die Steuerung übernimmt.

Für die Portlet Portierung bietet sich der Standard JSR-168 bzw. der neuere JSR-286 an. Eine Portierung und Anpassung an das eigene Portal ist nicht sehr aufwändig. Statt der *doGet()* bzw. *doPost()*-Methoden kann z.B. die Methode *doView()* aufgerufen werden, um das Portlet zu starten. Zudem wird die Methode *processAction()* genutzt, wenn das kurzlebige Zertifikat vom DFN-SLCS zurückkommt.

In der Methode *processAction()* (bzw. *processRequest()* im Falle eines Servlets) wird, analog zu Perl-Implementierung, zunächst der Request-Parameter *status* geladen und ausgewertet: Falls dieser Parameter noch keinen Wert hat, so handelt es sich um einen Erst-Aufruf und die Methode *makeRequest()* wird aufgerufen. Hat der Parameter *status* den Wert *success*, so bedeutet dies, dass ein Zertifikat zurück gesendet worden ist, woraufhin die Methode *success()* ausgeführt wird. Hat *status* den Wert *rejected* oder einen anderen Wert, so wird eine Fehlermeldung ausgegeben. Dieses Verhalten wurde aus der Perl Implementierung übernommen.

Alle Implementierungen besitzen die beiden Methoden *makeRequest()* und *success()*. In der Methode *makeRequest()* wird sowohl das Schlüsselpaar als auch der PKCS10-Request erstellt [27]. Da diese Funktionalität von allen Servlet und Portlet Varianten verwendet wird, ist der Code in die Klasse *ServletHelper* ausgelagert worden. Zudem erstellt diese Methode auf der zugehörigen Web-Oberfläche einen Button, mit dem der Request an den Server am DFN gesendet werden kann.

Die Methode *success()* wird aufgerufen, wenn das fertige Zertifikat zurückgeschickt worden ist. Zunächst wird anhand der Moduli des gespeicherten privaten Schlüssel und des erhaltenen Zertifikats überprüft, ob das Zertifikat und der zuvor generierte private Schlüssel zusammengehören. Ist dies der Fall, wird in der einfachsten Version lediglich ein Proxy-Zertifikat vom erhaltenen kurzlebigen Zertifikat abgeleitet. In den weiteren Ausbaustufen wird zusätzlich eine SAML-Assertion ausgestellt und in das Proxy-Zertifikat mit eingebunden, die (je nach Version) sowohl Campus-Attribute aus der Shibboleth-Umgebung als auch Attribute und Rollen aus der virtuellen Organisation enthalten kann. Der Codeaufruf dafür findet sich in den Methoden *createProxy()* bzw. *createSamlProxy()*.

3.3 Verarbeitung der Nutzer-Attribute

Zu Autorisierungszwecken sollen Attribute Verwendung finden. SAML Assertions werden zum einen als Übertragungsformat für Campus-Attribute genutzt, die vom Portal Delegation Portlet aus der Shibboleth Umgebung bezogen werden, sowie für VO-Attribute vom VOMS-Server. In beiden Fällen handelt es sich um SAML Assertions der Version 2. Die Attribute aus diesen Assertions werden in einer eigenen Datenstruktur gesammelt und anschließend zu einer SAML Assertion der Version 1 zusammengefasst, die von der Software Gridshib for Globus Toolkit ausgewertet werden kann (vgl. Kap. 4). Dieser Vorgang findet am Portal statt. Die resultierende SAML Assertion wird daher auch durch das Portal-Zertifikat signiert. In diesem Kapitel finden sich einige Hinweise zu Implementierung und Funktion. Zur Verarbeitung der SAML Assertions wird dabei durchgehend die Bibliothek OpenSAML verwendet.

3.3.1 Datenstruktur für gesammelte Attribute

Um die zu sammelnden Attribute zur Ausstellung einer neuen SAML Assertion zwischenspeichern zu können, wurde eine entsprechende Datenstruktur geschaffen.

Die Klasse *de.awi.gapslc.slc.tools.saml2.SAMLAttributes* bietet Methoden, einzelne Werte oder komplette Attribute hinzuzufügen. Ein einzelnes Attribut wird dabei von der Inneren Klasse *Attribute* repräsentiert. Die öffentlichen Methoden sind im Klassendiagramm in Abbildung 3 dargestellt.

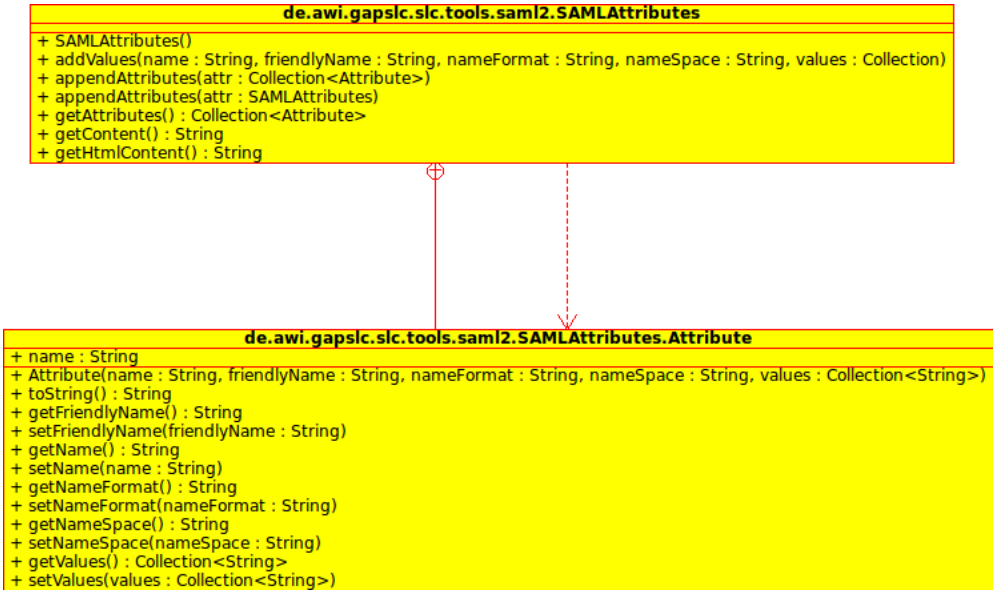

Abbildung 3: Datenstruktur für gesammelte SAML Attribute

3.3.2 Bezug der Campus Attribute

Der Bezug der Campus Attribute aus der Shibboleth Umgebung ist relativ einfach. Zunächst werden mit Hilfe des Attributes *Shib-Assertion-Count* die Anzahl der Assertions bestimmt und dann nacheinander alle Assertions geladen. Implementiert ist dies in der Klasse *de.awi.gapslc.slc.tools.saml2.AssertionHandler*.

Das Attribut *Shib-Assertion-xx* enthält die URL der Assertion, wobei das *xx* für die fortlaufende Nummer der Attribut-Namen steht. Die URL kann direkt an einen Parser übergeben werden, der die XML Datei lädt und ein Java Objekt vom Typ org.*w3c.dom.Document* zurück gibt.

Nun muss diese XML Struktur in ein SAML Objekt überführt werden. Dieser Prozess, Unmarshalling genannt, wird für den Root-Knoten dieser XML Struktur ausgeführt. Das Ergebnis ist ein Objekt des Typs *org.opensaml.saml2.core.Assertion*, für das entsprechende Methoden zur weiteren Verarbeitung von der OpenSAML Bibliothek bereitgestellt werden.

Wenn alle Assertions auf diese Weise geladen wurden, werden die Attribute der Reihe nach ausgelesen und in der in Kapitel 3.3.1 beschriebenen Datenstruktur abgelegt. Eine vereinfachte Übersicht der beteiligten Klassen ist in Abbildung 4 dargestellt.

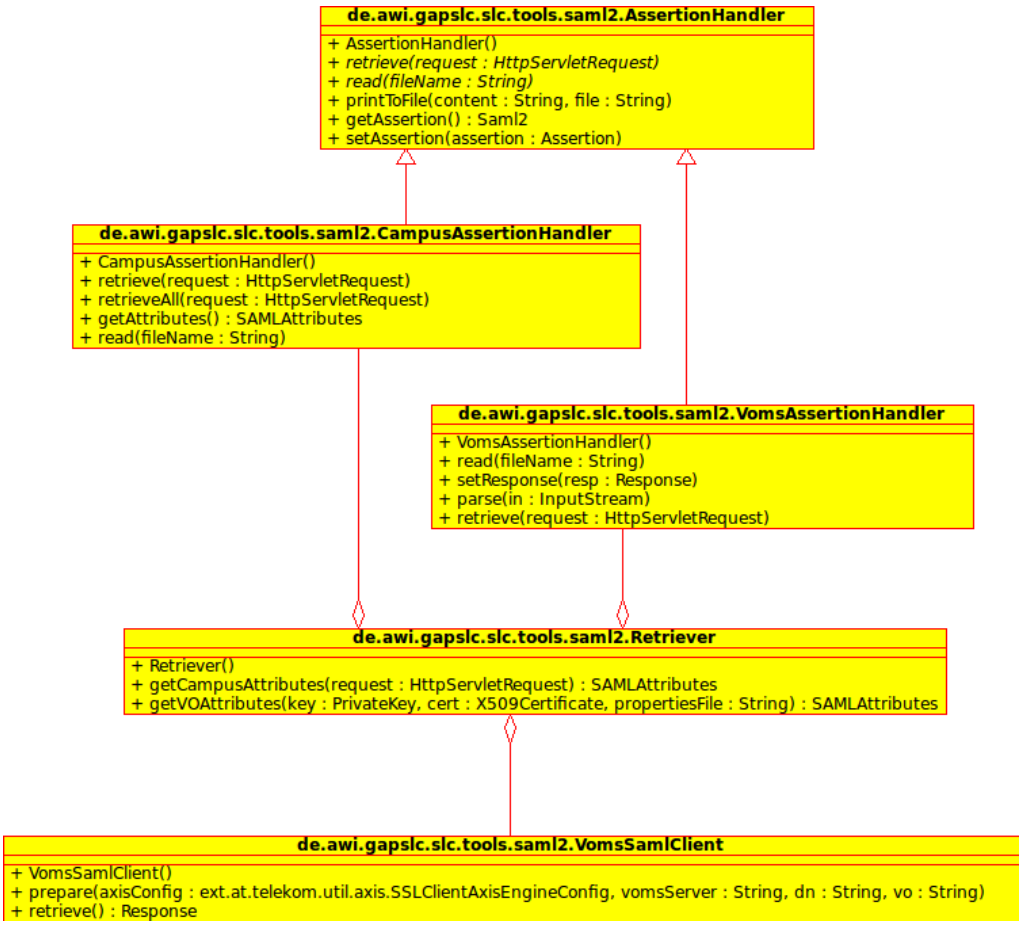

Abbildung 4: Retriever

3.3.3 Bezug der VO-Attribute

Für die Interaktion mit dem VOMS Server muss der Nutzer bereits mit einem gültigen Zertfikat bei der VO registriert sein. Danach wird bei jeder weiteren Anfrage das ankommende Zertifikat auf seine Gültigkeitsdauer und die Vertrauenswürdigkeit des Stammzertifikats geprüft, sowie ob Subject und Issuer mit den Angaben des für die Registrierung verwendeten Zertifikats übereinstimmen.

Um die SAML Assertion mit den VO-Attributen vom VOMS-Server abzuholen, wird ein Client genutzt, der auf der Implementierung eines bestehenden Testclients[2] beruht, der im Rahmen von SAML VOMS entwickelt wurde. Dieser Client kontaktiert einen Web-Service, an dem zur Authentifizierung das Zertifikat des Nutzers verwendet werden muss, mit dem auch die Registrierung an der jeweiligen VO durchgeführt worden ist.

In dieser Software werden kurzlebigen Zertifikate verwendet, die vom Portal bezogen werden und die einmalig bei der VO registriert werden müssen. Damit kann dann das Portal im Namen des Nutzers ohne jegliche weitere Interaktion auch die VO-Attribute des Nutzers beziehen und nutzen (Details unter [4]).

Der Client ist in der Klasse *de.awi.gapslc.slc.tools.saml2.VomsSamlClient* implementiert (vgl. Abbildung 4). Dieser Client verbindet sich mit dem Web-Service-Modul des VOMS-Servers und authentifiziert sich dort im Namen des Nutzers mit dessen kurzlebigem Zertifikat. Die Echtheit des VOMS Servers wiederum wird über den Abgleich mit dem gespeicherten öffentlichen Zertifikat des Nutzers geprüft. Die Attribute

[2] Download per CVS: cvs –d :pserver:anonymous@glite.cvs.cern.ch:/cvs/glite/ co –r glite-security-voms-admin-server_R_2_0_18_1 org.glite.security.voms-admin-server

werden dann mit Hilfe der Klasse *de.awi.gapslc.slc.tools.saml2.VomsAssertionHandler* extrahiert und zu den anderen bereits gesammelten Attributen hinzugefügt.

3.3.4 Ausstellung einer neuen Assertion

Nachdem die Attribute aus den unterschiedlichen Quellen zusammengestellt worden sind, kann eine neue SAML-Assertion (Version 1) ausgestellt werden. Dazu wird die neu implementierte Klasse *de.awi.gapslc.slc.tools.saml2.Saml1* genutzt. Sie repräsentiert eine Assertion und bietet Methoden zum Erstellen und Auswerten. Die Attribute können direkt in der in Kapitel 3.3.1 bereits beschriebenen Datenstruktur übergeben werden. Aufgerufen wird der Code direkt aus dem Portlet von der Methode *createSamlProxy()*. Wenn die SAML Assertion erstellt ist, wird sie bei der anschließenden Erstellung des Proxy-Zertifikats gleich mit darin eingebunden.

3.4 Anpassungen der Software

3.4.1 Axis-Mod

Wie schon erwähnt, muss sich der Client mit seinem Zertifikat am VOMS Server authentifizieren. Nur ein authentifizierter Client bekommt seine Attribute in Form einer SAML Assertion zugeschickt. Für den Test-Client sowie den Server wurde Axis 1.4 verwendet, wobei es sich beim Client um ein „Stand-Alone"-Programm handelt, das direkt von der Kommandozeile aufgerufen werden kann.

Mit einer einfachen Einbettung des Axis-Clients in das Servet kann man zwar eine SAML Assertion beziehen. Da intern jedoch das Zertifikat nicht ausgetauscht wird, funktioniert dies nur für jeweils einen Nutzer. Erst nach Neustart des Programms – in diesem Fall bedeutet dies den Neustart des Tomcat-Containers – kann ein anderer Nutzer bedient werden.

Es handelt sich dabei um ein generelles Problem, das im Apache-Wiki[3] bereits dokumentiert wurde und wofür ein Workaround existiert, der hier übernommen werden konnte. Konkret handelt es sich um eine Anpassung der Klasse *SocketFactoryFactory* aus der Axis-Bibliothek sowie die Ergänzung um einige weitere Klassen.

3.4.2 Änderung der HTTP-Methode

Aufgrund eines bekannten Fehlers am DFN-SLCS während des Shibboleth-Login-Dance muss vorerst für die Übertragung des Zertifikat-Request an den DFN-SLCS „http-post" statt „http-get" verwendet werden.

Die Implementierung erfolgte in der Methode makeRequest() in der jeweiligen Klasse. Dort findet sich jeweils am Ende der Methode HTML Code für einen Button. Im Portlet dient dieser Button dazu, den Zertifikate-Request an den DFN-SLCS zu senden. An dieser Stelle muss „http-post" eingetragen werden.

3.5 Installation der Software

Soll die Software in einem Portal genutzt werden, sollte die Portlet Variante gewählt werden. Die Installation von Portlets in einer Portalsoftware wie Liferay ist in der Regel sehr einfach über die Portalsoftware durchzuführen. Es existieren zwei Portierungen[4]: Eine Basis-Version, in der nur das SLC bezogen wird und keine SAML Attribute integriert werden, sowie eine Variante mit Campus- und VO-Attributen.

Die Servlet Implementierung kann als eigenständige Webanwendung genutzt werden und eignet sich daher bevorzugt zu Test- und Demozwecken. Benötigt wird nur ein geeigneter Container, z.B. Tomcat, und ein Apache Webserver.

[3] http://wiki.apache.org/ws/FrontPage/Axis/DynamicSSLConfig
[4] http://aforge.awi.de/gf/project/gapslc/frs/?action=FrsReleaseBrowse&frs_package_id=34

In beiden Fällen ist zu beachten, dass für den Bezug des kurzlebigen Zertifikats eine Shibboleth Session vorliegen muss.

Beispielkonfiguration für Servlets

Im Unterverzeichnis *sites-enabled* der Apache Konfiguration kann die Shibbolisierung der Site eingetragen werden:

```
<Location "/slc">
   AuthType shibboleth
   ShibRequireSession On
   ShibRequireAll On
   ShibRedirectToSSL 443
   require valid-user
   ShibExportAssertion On
</Location>
```

Um Tomcat zusammen mit dem Apache Webserver verwenden zu können, wurde mod_jk benutzt. Im Unterverzeichnis *conf.d* muss daher in der Datei *mod-jk.conf* folgender Eintrag hinzugefügt werden:

```
JkMount /slc* ajp13_worker
JkMount /slc ajp13_worker
```

Wird die war-Datei, in der die Software u.a. vorliegt, in den webapps-Ordner von Tomcat kopiert, wird automatisch ein Autodeploy-Prozess gestartet. Konfiguriert wird das Servlet über eine Properties-Datei, die unter */usr/local/etc/slc.properties* vorliegen muss. In dieser Datei muss der Speicherort für die Proxy-Zertifikate, die URL des DFN SLCS, die eigene Portal-URL, der Pfad zum Portal-Zertifikat und Schlüssel sowie der Name der Campus-Assertion vom IdP konfiguriert werden. Im Folgenden ist eine Beispieldatei mit allen relevanten Einträgen aufgeführt.

```
# This properties file is used to configure the portal delegation servlet.
# Additionally, settigs to collect the campus- and vo-assertions can be made.

#Path on the local file system to store the slc, key and proxy.
#Please make sure this folder really exists!!!
cred_path = /tmp/

#URL of the short lived credential service
target_url = https://test-slcs.pca.dfn.de/cgi-bin/portalLogin.cgi

#URL of your own portal
my_url = https://shib2-sp.awi.de/slc/

#Location of cert and key used to sign the saml assertion
container_cert = /etc/grid-security/containercert.pem
container_key = /etc/grid-security/containerkey.pem

#VO-Settings
trustStore = /home/globus/trustStore.jks
trustStorePasswd = 123456
server = voms2.awi.de
vo = testslc1

#Store proxy-certificate (only for *T-Versions)
storeProxy = no
```

Beispielkonfiguration für Portlets

Um die Software als Portlet deployen zu können ist ein installiertes Portal erforderlich, dass per Shibboleth geschützt sein muss. Der Shibboleth Schutz ist analog zur Servlet-Variante zu konfigurieren.

Die Konfiguration der Portlet Variante unterscheidet sich nur geringfügig von der Servlet-Konfiguration. Hauptunterschied ist, dass nun statt der Properties-Datei eine XML Datei verwendet wird. Das folgende Code-Fragment zeigt, welche Einstellungen der Datei portlet.xml hinzugefügt werden müssen.

```
<init-param>
        <name>cred_path</name>
        <value>/tmp/</value>
      </init-param>
      <init-param>
        <name>target_url</name>
        <value>https://test-slcs.pca.dfn.de/cgi-bin/portalLogin.cgi</value>
      </init-param>
      <init-param>
        <name>my_url</name>
        <value>https://shib2-sp.awi.de/web/guest/simpleSLC-portlet/</value>
      </init-param>
```

Für die erweiterte Portlet Variante mit eingebetteter SAML Assertion sind folgende Konfigurationseinstellungen hinzuzufügen:

```
<init-param>
        <name>credPath</name>
        <value>/tmp/</value>
      </init-param>
      <init-param>
        <name>targetUrl</name>
        <value>https://test-slcs.pca.dfn.de/cgi-bin/portalLogin.cgi</value>
      </init-param>
      <init-param>
        <name>trustStore</name>
        <value>/home/globus/trustStore.jks</value>
      </init-param>
      <init-param>
        <name>trustStorePassword</name>
        <value>123456</value>
      </init-param>
      <init-param>
        <name>vomsServer</name>
        <value>voms2.awi.de</value>
      </init-param>
      <init-param>
        <name>signingCert</name>
        <value>/etc/grid-security/containercert.pem</value>
      </init-param>
      <init-param>
        <name>signingKey</name>
        <value>/etc/grid-security/containerkey.pem</value>
      </init-param>
```

In der Portlet-Variante wird der Name der VO für jeden Nutzer individuell über „Custom Attributes“ konfiguriert. Dazu muss der Portal Administrator im Liferay Portal im „Control Panel“ unter „Custom

Fields“ in der Kategorie „User“ ein neues Feld hinzufügen. Das Feld muss den Typ „text“ haben und mit dem Schlüssel „voName“ angelegt werden. Desweiteren sind unter „Permissions“ die Update- und Lese-Rechte für die Portalnutzer für die Rollen *Owner* und *User* zu setzen.

4 Auswertung der SAML Assertions an der Grid-Ressource

Die verschiedenen SAML Assertions, deren Zusammenfassung am Portal im vorigen Kapitel beschrieben wurde, können bei den Ressourcenprovidern ausgewertet werden. Dazu können mehrere Policy Information Points (PIP) sowie Policy Decision Points Points (PDP) konfiguriert werden, an denen Informationen gesammelt bzw. die Autorisierungsentscheidungen getroffen werden. In Abbildung 5 ist die Kette der PIP und PDP von Gridshib dargestellt. Interessant für diesen Task ist speziell der SAML PDP, an dem Informationen aus einer an das verwendete Proxy-Zertifikat gebundene SAML Assertion ausgewertet werden [14].

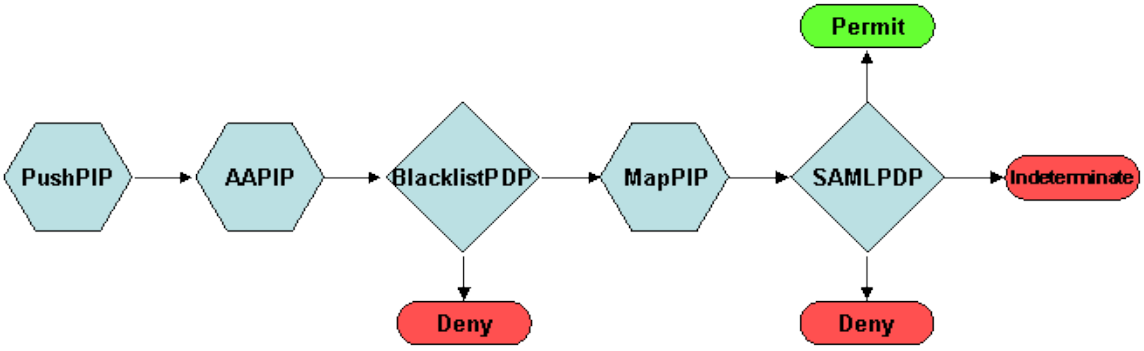

Abbildung 5: Gridshib Policy Decision Point [14]

Für die Testumgebung wurden folgende Programmversionen verwendet:

- Betriebssystem: Ubuntu 10.04 LTS
- Java 6
- Globus Toolkit 4.0.8
- Gridshib for Globus Toolkit 0.6.1
- Gridshib SAML Tools 0.5.0

4.1 Auswertung signierter SAML Assertions mit dem Globus Toolkit und Gridshib for Globus Toolkit

Gridshib unterscheidet zwar konzeptionell zwischen dem Aussteller des Zertifikats und dem Aussteller der SAML Assertion, jedoch haben die Entwickler bisher nur solche Anwendungsfälle vorgesehen, bei denen beide identisch sind und die Signatur des Zertifikats auch gleichzeitig die Assertions einschließt. Dritte Akteure kommen in ihren Use Cases nicht vor [25].

Dies deckt sich leider nicht mit unserem Anwendungsfall, wo das Zertifikat vom DFN-Verein ausgestellt und signiert wird und erst durch das Portal eine eigene Assertion eingefügt wird. Um trotzdem eine Vertrauensbasis für die zugesicherten Attribute zu gewährleisten, muss also auch diese eingefügte Assertion nochmals signiert werden, wobei die Signatur allerdings auf eine andere Zertifikatkette zurückgeht als die des umgebenden X.509 Zertifikats (vgl. Abbildung 6). Daher wird eine solche Assertion von der Originalversion der Gridshib Software nicht für die Auswertung verwendet, sondern einfach ignoriert.

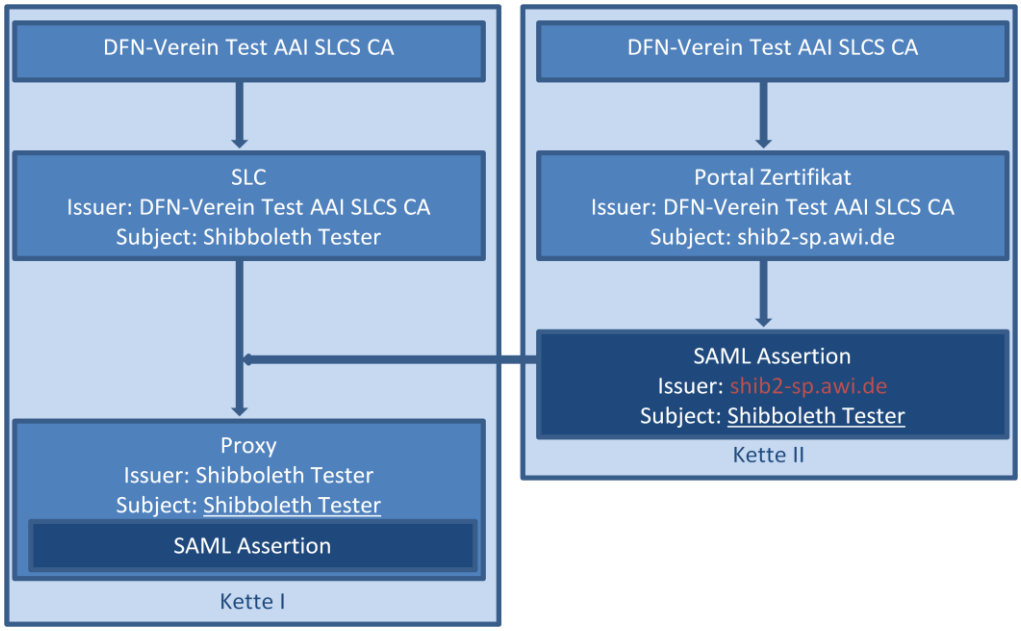

Abbildung 6: Beteiligte Zertifikatketten

Die hier dokumentierte Software ist insofern modifiziert und kann entscheiden, ob eine Assertion zur Auswertung genutzt werden darf. Dazu werden zunächst einige Prüfungen durchgeführt, bevor die Assertion akzeptiert wird:

- Syntaxcheck der SAML-Assertion
- Überprüfung der Signatur der SAML Assertion
- Extension Check, Validity Check, Restricted Policy Check, CRL Check, etc.
- Test, ob Issuer der Assertion in der trustedAuthoritiesList eingetragen ist
- Test, ob SAML Assertion Subject und Proxy Subject identisch sind

Diese zusätzlichen Überprüfungen mussten teilweise neu implementiert werden, was im Folgenden beschrieben wird.

4.2 Implementierung

Für die Validierung der in das Proxy-Zertifikat eingebetteten SAML Assertion wird vom Attribute Acceptance Policy Information Point (AAPIP, vgl. Abbildung 5) die Klasse *SAMLUtil* aus den Gridshib-SAML-Tools genutzt. Um aber den gegenüber dem ursprünglichen GridShib Anwendungsfall geänderten use case (Einbettung der SAML Assertion in Proxy-Zertifikat und Nutzung der Zertifikat-Kette II gemäß Abbildung 6) bearbeiten zu können, mussten Modifikationen vorgenommen werden: Sie betreffen die zusätzliche Prüfung, ob die Assertion auch wirklich für das Proxy-Zertifikat ausgestellt wurde, in das sie eingebettet ist. Dazu wird ein einfacher String-Vergleich des Inhalts des Subject-Feldes der Assertion mit dem Inhalt des Subject-Feldes des umgebenden Proxy-Zertifikates durchgeführt.

Für die Überprüfung der Signatur der SAML-Assertion wird die Klasse *ProxyPathValidator* aus dem Globus Toolkit verwendet. Diese Klasse dient ursprünglich dazu, die Unterschriftenkette eines Proxy-Zertifikates zu überprüfen. Dazu wird auf die Standard-Ablageorte für Root-Zertifikate zugegriffen und versucht, die Zertifikatkette des Proxy-Zertifikats bis zu einem Zertifikat aus diesem Ordner zurückzuverfolgen. Sie kann ohne weitere Modifikation auch direkt von der Klasse *SAMLUtil* aus genutzt werden, um das Zertifikat des Ausstellers der SAML-Assertion (in Abbildung 6 ist dies das Portal-Zertifikat) zu überprüfen.

Als nächster Schritt wird in der Klasse *AAPIPImpl* überprüft, ob der Aussteller der SAML-Assertion überhaupt vertrauenswürdig ist. Dazu existiert im Verzeichnis */etc/grid-security/metadata* die Datei *trustedAutoritiesList*, in der alle zulässigen Aussteller von SAML-Assertions aufgelistet sind. All dies setzt natürlich voraus, dass der entsprechende DN des umgebenden Proxy-Zertifikats auch im *grid-mapfile* eingetragen ist.

In den Methoden *consumeSAMLAssertion()* und *isAssertionValid()* werden die oben genannten Tests durchgeführt.. Sie liefern je nach Ergebnis *true*, um die Assertion zu akzeptieren, oder *false*, um die Assertion abzuweisen.

Insgesamt konnten die Änderungen auf eine einzelne Klasse (*SAMLUtil*) beschränkt werden, die in der Java-Bibliothek gridshib-common-0_5_0.jar ersetzt wird.

4.3 Installation der modifizierten SAML-Tools-Bibliothek

Dieser Abschnitt beschreibt die Integration der modifizierten Bibliothek in ein **installiertes** Globus Toolkit 4.0.8 und Gridshib for Globus Toolkit 0.6.

Bei der Installation von Gridshib for Globus Toolkit werden bereits diverse Java Bibliotheken mitgeliefert, darunter auch die Originalversion von gridshib-common-0_5_0.jar. Im *lib*-Verzeichnis des Globus-Containers wird nun die alte durch die modifizierte Version dieser Bibliothek ersetzt. Um die Änderungen wirksam werden zu lassen, sollte der Globus-Container neu gestartet werden.

Diese modifizierte Java Bibliothek kann entweder direkt heruntergeladen[5] oder aus den Quellen selbst erzeugt werden. Die Bibliothek kommt ursprünglich aus den Gridshib-SAML-Tools, wo der Quellcode der Klasse SAMLUtil durch die neue Version ersetzt wird. Mit Hilfe des enthaltenen Ant-Scipts werden eine Vielzahl von Java Bibliotheken erzeugt, von denen aber nur die Datei gridshib-common-0_5_0.jar benötigt wird.

4.4 Anpassung der Konfiguration des Globus Containers

Die durch Gridshib for Globus Toolkit bereitgestellten Features (die Möglichkeit Blacklists zu erstellen, auf Basis der enthaltenen Attribute eine Autorisierung sowie das Mapping auf lokale Nutzerkonten vorzunehmen, etc.) stehen auch in der modifizierten Version voll zur Verfügung. Details dazu und weitere Hinweise zur Konfiguration der PIPs und PDPs finden sich in der Dokumentation zu Gridshib for Globus Toolkit [12]. Im Folgenden werden die Spezifika zur Konfiguration der neuen Version beschrieben.

In der Datei *global-authz-policy.xml* wird ein Attribut zur Autorisierung konfiguriert. Es können sowohl mehrere Werte für ein Attribut als auch mehrere Attribute angegeben werden.

Beispielkonfiguration der Attribute für Autorisierung

in Datei: */etc/grid-security/policy/global-authz-policy.xml*

[5] http://aforge.awi.de/gf/project/gapslc/frs/

```
<AttributePolicy
  xmlns="http://gridshib.globus.org/namespaces/2005/08/policy"
  xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
  xmlns:saml1="urn:oasis:names:tc:SAML:1.0:assertion">
 <entry>
  <listOfAttributes>
   <saml:Attribute AttributeName="urn:mace:dir:attribute-def:eduPersonEntitlement"
     AttributeNamespace="urn:mace:shibboleth:1.0:attributeNamespace:uri">
        <saml:AttributeValue>urn:geant:dfn.de:dfn-pki:slcs</saml:AttributeValue>
   </saml:Attribute>
  </listOfAttributes>
 </entry>
</AttributePolicy>
```

Achtung: Die Attribute müssen genau so eingetragen werden, wie sie auch in der SAML Assertion vorliegen. Dabei ist auch der korrekte Namespace zu beachten!

Beispielkonfiguration der verwendeten Gridshib PDP und PIP

Ausschnitt aus Datei: */opt/globus-4.0.8/etc/globus_wsrf_core/ global_security_descriptor.xml*

```
<authz value="global:org.globus.gridshib.SAMLAssertionPushPIP
          global:org.globus.gridshib.AttributeAcceptancePIP
          global:org.globus.gridshib.SAMLBlacklistPDP
          global:org.globus.gridshib.SAMLMapPIP
          global:org.globus.gridshib.SAMLAttributePDP"/>
```

Beispielkonfiguration der Attribute für Autorisierung

In der Datei *server-config.wsdd* wird eingetragen, dass die Datei *global-authz-policy.xml* für die Autorisierung verwendet werden soll. Dieses Beispiel zeigt nur die globale Konfiguration des Servers. Einzelne Teile, wie z.B. gram, rft, etc. können natürlich auch separat konfiguriert werden.

Werden in einer Datei mehrere Attribute bzw. mehrere Werte für ein Attribut angegeben, so wird *PERMIT* zurückgegeben, sobald ein Attribut vorhanden ist. Dies entspricht somit einer logischen ODER Entscheidung. Wenn auf zwei oder mehr Attribute getestet werden soll, die alle vorhanden sein sollen (logisches UND), so müssen diese auf mehrere Dateien verteilt werden. Dazu wird im Security Descriptor ein weiterer *SAMLAttributePDP* eingetragen und in der Datei *server.config* ein zweiter *attributeAuthzPolicyFile*-Eintrag definiert.

Ausschnitt aus Datei: */opt/globus-4.0.8/etc/globus_wsrf_core/ server-config.wsdd*

```
<!-- GRIDSHIB CONFIGURATION, SCOPE: GLOBAL -->
<parameter name="global-metadataPath" value="/etc/grid-security/metadata"/>
<parameter name="global-attributeAuthzPolicyFile" value="/etc/grid-security/policy/global-authz-
policy.xml"/>
<parameter name="global-attributeMappingPolicyFile" value="/etc/grid-security/policy/global-mapping-
policy.xml"/>
<parameter name="global-blacklistPrincipalNamesFile" value="/etc/grid-security/blacklists/blacklist-principal-
names.txt"/>
<parameter name="global-blacklistIPAddressesFile" value="/etc/grid-security/blacklists/blacklist-ip-
addresses.txt"/>
<parameter name="global-consultDefaultGridmap" value="false"/>
<parameter name="global-enableGridmap" value="false"/>
```

4.5 Beispielkonfiguration und Test der Software

Um das Zusammenspiel aller beteiligten Komponenten überprüfen zu können, kann eine einfache Testumgebung genutzt werden. Die hier verwendete Umgebung enthält einen Rechner, auf dem das Portal sowie ein Globus Toolkit 4.0.8 läuft, mit dem die Grid-Jobs abgeschickt werden. Dieser Rechner ist als Shibboleth Servive Provider konfiguriert. Auf weiteren Servern, die als Test-Grid-Ressourcen genutzt werden, ist als Middleware ebenfalls Globus 4.0.8 mit installiert. Daneben existieren ein Shibboleth IdP und der eigens aufgesetzte VOMS Server. Zudem werden der WAYF und der DFN-Test-SLCS des DFN-Vereins genutzt.

In der Testumgebung wird das notwendige Credential zum Starten von Grid-Jobs mit dem bereits vorgestellten Portlet (vlg. Kapitel 3) erzeugt. Dabei werden vom Portlet das kurzlebige Zertifikat und die SAML-Assertions mit den Campus und VO-Attributen bezogen. Eine neue, durch das Portal signierte SAML-Assertion mit den gesammelten Attributen wird ausgestellt und in ein neu erstelltes Proxy-Zertifikat eingebettet. Dieses Proxy-Zertifikat wird für die Tests in diesem Kapitel vom Portlet direkt im Dateisystem abgespeichert und verwendet, um mit dem Programm globusrun-ws einen Grid-Job auf einer Test-Grid-Resource zu starten.

Auf dem Portal-Rechner, von dem aus auch die Jobs abgeschickt werden, müssen die Variablen *X509_USER_CERT, X509_USER_KEY* und *X509_USER_PROXY* alle auf das Proxy-Zertifikat zeigen , das zudem die Unix-Rechte 400 haben sollte. Der Grid-Job wird per Kommandozeile mit dem Programm *globusrun-ws* abgesetzt. Die Test-Ressourcen mit Globus 4.0.8 und Gridshib for Globus Toolkit 0.6 installiert ist.

4.5.1 Verwendung Originalvariante der Bibliothek

Zu Testzwecken wird zunächst die originale Version von SAMLUtil verwendet. Die eingebettete SAML Assertion kann nicht ausgewertet werden, so dass die enthaltenen Attribute nicht zur Autorisierung herangezogen werden. Der abgeschickte Grid-Job wird daher nicht ausgeführt.

```
$/opt/globus/globus-4.0.8/bin/globusrun-ws -submit -s -F gs4gt5.awi.de:8443 -c /bin/date
Delegating user credentials...Done.
Submitting job...Failed.
Cleaning up any delegated credentials...Done.
globusrun-ws: Error submitting job
globus_soap_message_module: SOAP Fault
Fault code: soapenv:Server.userException
Fault string: org.globus.wsrf.impl.security.authorization.exceptions.AuthorizationException: "/C=DE/O=DFN-
Verein/OU=DFN-PKI/OU=SLCS/OU=Alfred-Wegener-Institut/CN=Stefan Pinkernell - Stefan.Pinkernell@awi.de"
is not authorized to use operation:{http://www.globus.org/namespaces/2004/10/gram/job}createManagedJob
on this service
```

4.5.2 Mit der modifizierten Bibliothek

Bei Verwendung der modifizierten Bibliothek werden auch die Attribute der durch das Portal signierten SAML Assertions in die Autorisierungsentscheidung mit aufgenommen. Befinden sich die entsprechenden Attribute in der SAML Assertion und ist das Proxy-Zertifikat gültig und vertrauenswürdig, so wird der per globusrun-ws gestartete Job ausgeführt.

```
$ /opt/globus/globus-4.0.8/bin/globusrun-ws -submit -s -F gs4gt5.awi.de:8443 -c /bin/date
Delegating user credentials...Done.
Submitting job...Done.
Job ID: uuid:0623816e-7879-11df-a92a-000c292b9aa5
Termination time: 06/16/2010 12:25 GMT
Current job state: Active
Current job state: CleanUp-Hold
Tue Jun 15 14:25:06 CEST 2010
Current job state: CleanUp
Current job state: Done
Destroying job...Done.
Cleaning up any delegated credentials...Done.
```

4.5.3 Fehlende Attribute zur Autorisierung

In diesem Test wird überprüft, dass fehlende Attribute in der Assertion zum Abweisen des Jobs führen. Dazu wird ein weiterer SAMLAttributePDP mit einem neuen Attribut, das nicht in der Assertion vorhanden ist, am Container konfiguriert. Die Ausführung des Grid-Jobs wird somit nicht zugelassen.

```
$ /opt/globus/globus-4.0.8/bin/globusrun-ws -submit -s -F gs4gt5.awi.de:8443 -c /bin/date
Delegating user credentials...Done.
Submitting job...Failed.
Cleaning up any delegated credentials...Done.
globusrun-ws: Error submitting job
globus_soap_message_module: SOAP Fault
Fault code: soapenv:Server.userException
Fault string: org.globus.wsrf.impl.security.authorization.exceptions.AuthorizationException: "/C=DE/O=DFN-
Verein/OU=DFN-PKI/OU=SLCS/OU=Alfred-Wegener-Institut/CN=Stefan Pinkernell - Stefan.Pinkernell@awi.de"
is not authorized to use operation:
{http://www.globus.org/namespaces/2004/10/gram/job}createManagedJob on this service
```

4.5.4 Ungültige SAML Assertion

Wird eine SAML Assertion mit einer Unterschrift eines nicht verifizierbaren Portal-Zertifikats benutzt, so ist auf der Client-Seite folgende Fehlermeldung zu sehen:

```
globusrun-ws: Error submitting job
globus_soap_message_module: SOAP Fault
Fault code: soapenv:Server.userException
Fault string: java.rmi.RemoteException: Job creation failed.; nested exception is:
      java.lang.Exception: Unable to retrieve a username mapping for authenticated user /C=DE/O=DFN-
Verein/OU=DFN-PKI/OU=SLCS/OU=Alfred-Wegener-Institut/CN=Stefan Pinkernell – Stefan.Pinkernell@awi.de.
```

In der Log-Datei *var/container.log* des Globus-Containers findet sich folgende Fehlermeldung, die auf eine fehlerhafte Signatur schließen lässt:

```
16.11.2009 16:11:20 org.globus.gridshib.security.util.SAMLUtil isAssertionValid
SCHWERWIEGEND: null
. Caused by COM.claymoresystems.cert.CertificateVerifyException: The signature of
'C=DE,O=GridGermany,OU=Alfred-Wegener-Institut,CN=shib2-sp.awi.de' certificate does not match its issuer
```

5 Fazit

Im es Projekt GapSLC wurde das Portal Delegation Verfahren implementiert und getestet. Im Namen des Nutzers kann mit dieser Software ein kurzlebiges Zertifikat aus einer Portalumgebung heraus bezogen werden. Zusätzlich dazu werden noch Attribute aus unterschiedlichen Quellen gesammelt und in einer SAML Assertion zusammengefasst. Diese Assertion wird in das vom kurzlebigen Zertifikat abgeleitete Proxy Zertifikat eingebunden und liefert damit Attribute, die an einer Grid-Ressource zur Autorisierung verwendet werden können.

Bereitgestellt wird die Software in Form eines Portlets, wodurch eine einfache Integration in eine Portalsoftware wie Liferay ermöglicht wird. Zudem existieren einige Servlet-Implementierungen mit unterschiedlichem Funktionsumfang, die nicht in ein Portal eingebettet werden und sich eher zu Demonstrations- und Testzwecken eignen.

Um eingebettete, portalsignierte SAML Assertion in einem Globus 4.0.x Containern auswerten zu können, musste auf Seiten der Grid-Ressource eine Bibliothek angepasst werden. Damit können nun Ressourcenprovider zusätzlich zur Authentifizierung über das kurzlebige Zertifikat eine sehr feingranulare Autorisierung anhand der enthaltenen Attribute konfigurieren.

Für den sinnvollen Einsatz des Portal Delegation Portlets sollte ein shibbolisiertes Portal verwendet werden. Der Einsatz dieser Software in einem nicht shibbolisierten Portal ist zwar auch so schon möglich da der DFN-SLCS per Shibboleth geschützt ist. In diesem Falle müsste sich der Nutzer jedoch zweimal authentifizieren: Zunächst am Portal und dann ein zweites Mail am Shibboleth Identity-Provider.

6 Literatur

[1] GapSLC-D1: *Vereinfachung der Handhabung des kurzlebigen Zertifikats (SLC) für Nutzer*, http://gap-slc.awi.de/documents/GapSLC-D1-V1.0.pdf

[2] GapSLC-D3: *Nutzung von kurzlebigen Zertifikaten in portalbasierten Grids*, http://gap-slc.awi.de/documents/GapSLC_D3_V1.0.pdf

[3] DGI-2, Fachgebiet 3.1, Koordination und Sicherheitsmanagement, *Verwendung von Zertifikaten im D-Grid*, http://dgi2.d-grid.de/fileadmin/user_upload/documents/DGI2-FG3/FG3-1/DGI-2_FG-3.1_Zertifikate_im_D-Grid_v10.pdf

[4] Benjamin Henne, DGI2 FG3.2, *Test des VOMS-SAML-Services des DGrid VOMS*, Juni 2009, http://dgi.d-grid.de/fileadmin/user_upload/documents/DGI2-FG3/FG3-2/DGI-2_FG-3.2_Test_VOMS-SAML-Service.pdf

[5] http://www.dfn.de/dienstleistungen/dfnaai/

[6] *Certificate Policy and Certification, Practice Statement of the Public Key Infrastructure in the Deutsche Forschungsnetz*, http://www.pki.dfn.de/fileadmin/PKI/DFN-PKI_SLCS-CPCPS_v11.pdf

[7] DFN-SLCS, https://www.pki.dfn.de/index.php?id=slcs

[8] EuGridPMA, *Protection of private key data for end-users in local and remote systems*, http://www.eugridpma.org/guidelines/pkp/pk-protection-1.0-20091016.pdf

[9] SWITCHHaai, http://www.switch.ch/aai/

[10] UK Access Management Federation for Education and Research, http://www.ukfederation.org.uk/

[11] Shibboleth, http://shibboleth.internet2.edu

[12] Gridshib, http://gridshib.globus.org/about.html

[13] Gridshib for Globus Toolkit, http://gridshib.globus.org/docs/gridshib-gt-0.6.1/readme.html

[14] Gridshib Quick Start, http://gridshib.globus.org/docs/gridshib/quick-start.html

[15] Gridshib SAML Tools, http://gridshib.globus.org/docs/gridshib-saml-tools-0.5.0/readme.html

[16] Portal Delegation, http://gridshib.globus.org/docs/gridshib-ca-0.5.1/portal-delegation.html

[17] GSI, http://www.globus.org/security/overview.html

[18] OASIS SAML, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security

[19] J. Hughes et al., OASIS Committee Draft, Technical *Overview of the OASIS Security Assertion Markup Language (SAML) V1.1.* , May 2004. http://www.oasis-open.org/committees/download.php/6837/sstc-saml-tech-overview-1.1-cd.pdf

[20] E. Maler et al., *Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V1.1. OASIS Standard*, September 2003. http://www.oasis-open.org/committees/download.php/3406/oasis-sstc-saml-core-1.1.pdf

[21] N. Ragouzis et al., OASIS Committee Draft, *Security Assertion Markup Language (SAML) V2.0 Technical Overview*, March 2008.

[22] http://www.oasis-open.org/committees/download.php/27819/sstc-saml-tech-overview-2.0-cd-02.pdf

[23] S. Cantor et al., *Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0. OASIS Standard*, March 2005.

[24] http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf

[25] SAML in X.509 validation, http://dev.globus.org/wiki/SAML_in_X.509_Validation

[26] OpenSaml, https://spaces.internet2.edu/display/OpenSAML/Home/

[27] RSA Laboratories, *PKCS#10 v1.7:Certification Request Syntax Standard*, Mai 2000, ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-10/pkcs-10v1_7.pdf