INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Fortgeschrittenenpraktikum

Konzeption und Implementierung einer organisationsübergreifenden Web Single Sign-On Testumgebung für das Münchner Wissenschaftsnetz

Matthias Ebert
Sebastian Zunhammer

Aufgabensteller: Prof. Dr. Heinz-Gerd Hegering

Betreuer: Wolfgang Hommel

Helmut Reiser

Abgabetermin: 08. Juni 2005

INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Fortgeschrittenenpraktikum

Konzeption und Implementierung einer organisationsübergreifenden Web Single Sign-On Testumgebung für das Münchner Wissenschaftsnetz

Matthias Ebert
Sebastian Zunhammer

Aufgabensteller: Prof. Dr. Heinz-Gerd Hegering

Betreuer: Wolfgang Hommel

Helmut Reiser

Abgabetermin: 08. Juni 2005

Zusammenfassung

Zahlreiche IT-Basisdienstleistungen können über Web-Schnittstellen mit einem Browser genutzt werden. Der Zugang zu jedem dieser Dienste ist traditionell mit einem Passwort geschützt. Vielen Benutzern ist es jedoch zu mühsam, sich viele verschiedene Passwörter zu merken, so dass sie überall dasselbe einsetzen, was ein nicht zu unterschätzendes Sicherheitsrisiko darstellt. Diesem Problem versucht man durch so genannte Web Single Sign-On Lösungen zu begegnen. Das Ziel dabei ist, dass sich der Benutzer einmal am System anmeldet und danach ohne weitere Passworteingaben sämtliche Dienste nutzen kann.
Ziel des Fortgeschrittenenpraktikums ist die Konzeption und Implementierung einer Testumgebung für das auf OpenSAML basierende Softwarepaket Shibboleth. In dem zugrunde liegenden Szenario werden prototypisch die LMU, das LRZ und die TU wiedergespiegelt, welche sich in einer sogenannten Föderation befinden. Auf angebotene Dienste, in diesem Fall geschützte Webseiten, soll organisationsübergreifend zugegriffen werden können.
Zu Beginn der Ausarbeitung werden Begriffe und Grundkonzepte von Shibboleth erklärt. Anschließend folgt ab Kapitel 2 eine mögliche Vorgehensweise zur Installation und Konfiguration entsprechender Softwarekomponenten und das Web Single Sign-On-System Shibboleth. Dabei auftretende Probleme werden erklärt und Lösungen zu deren Behebung vorgeschlagen. Ab Abschnitt 6 erfolgt eine genaue Anleitung zur Konfiguration der Shibbolethkomponenten auf entsprechenden Linux-Servern. Zum Schluss wird ein Ausblick auf mögliche Einsatzgebiete gewährt, sowie auf noch unbehandelte Problemstellungen hingewiesen.

Gliederung:

1 Einleitung.....6
1.1 Was ist Shibboleth?.....6
1.2 Aufgabenstellung.....6
1.3 Begriffsklärung.....8
1.4 Vorgehensweise und Probleme.....10
2 Benötigte Grundinstallationen für Shibboleth (Origin).....11
3 Shibboleth Origin Installation.....24
3.1 Installation.....24
3.2 Grundkonfiguration.....25
4 Benötigte Grundinstallationen für Shibboleth (Target).....27
5 Shibboleth Target Installation.....28
5.1 Installation.....28
5.2 Grundkonfiguration.....29
6 Konfiguration von Shibboleth.....31
6.1 Aufsetzen eines eigenständigen WAYFs.....31
6.2 SAML-Konfiguration.....32
6.2.1 Target.....32
6.2.2 Origin.....34
6.3 Start-Scripts.....35
6.3.1 Target.....35
6.3.2 Origin.....35
6.4 PubCookie und neue Zertifikate.....37
6.5 trust.xml.....37
7 Zusammenfassung und Ausblick.....38
8 Literaturverzeichnis.....39

1 Einleitung

1.1 Was ist Shibboleth?

Durch den Einsatz des Zugriffsystems Shibboleth wird es Benutzern verschiedenster Web-Dienstleistungen ermöglicht, diese organisationsübergreifend und durch einmaliges Anmelden nutzen zu können. Man spricht hier auch von einem sogenannten Web Single Sign-On System.

Shibboleth bietet eine leistungsfähige, erweiterbare und benutzerfreundliche Lösung um geschützten digitalen Inhalt zugänglich zu machen. Es kann auf bereits vorhandene Infrastruktur für das Identitäts- und Zugangsmanagement aufgesetzt werden, um Einzelpersonen zu authentisieren, entsprechende Informationen an Dienstleister weiterzuleiten, mit deren Hilfe autorisierungsrelevante Entscheidungen getroffen werden. Shibboleth stellt im Moment ein sicheres, auf offiziellen Standards basierendes und providerübergreifendes Zugriffskontroll-System für Forschung und Entwicklung zur Verfügung.

Über Shibboleth werden nur unkritische Informationen (Attribute über eine Person, die Zugriff verlangt) ausgetauscht. Das erlaubt Organisationen mit unterschiedlichen Architekturen und Sicherheitssystemen auf einfachste Weise zusammenzuarbeiten, ganz ohne zusätzliche Proxies oder dem Management tausender externer oder temporärer Accounts. Auf diese Weise müssen sich Benutzer nicht für jede Seite auf die sie Zugriff haben Kennwörter (und oft auch IDs) merken.

Wenn sich Organisationen entschließen, gemeinsam Shibboleth zu benutzen, um damit ihren Benutzern Dienste organisationsübergreifend zur Verfügung zu stellen, nennt man das eine Föderation. Shibboleth unterstützt Föderationen durch leicht erweiterbare Methoden des Managements und der Verteilung von Konfigurations- und Sicherheitsinformationen über viele Organisationen. Es wird auch ein breites Spektrum an möglichen Benutzer-Attributen bereitgestellt.

Implementiert wird Shibboleth unter Java und ist somit auf fast allen Betriebssystemen benutzbar.

1.2Aufgabenstellung

Ziel des Fortgeschrittenenpraktikums war die Implementierung des oben beschriebenen Zugriffsystems Shibboleth mit folgenden Vorgaben:

- Die Föderation besteht aus LMU, TU, LRZ.
- An der LMU und TU gibt es also Informatiker und BWLer. Am LRZ gibt es Informatiker.
- Die jeweiligen Fakultätsseiten sind alle durch Shibboleth geschützt.
- Auf die jeweilige Startseite (von der es zu den geschützten Fakultäten geht) kommt jeder, z.B: https://lxshib07.lrz-muenchen.de/lrz.html .
- Alle Informatiker der Föderation können sich über Shibboleth in alle Informatik-Seiten einloggen, kommen aber nicht auf die BWL-Seiten.
- Die BWLer können ebenfalls über Shibboleth auf vorhandene BWL-Seiten zugreifen, kommen aber nicht auf die Informatik-Seiten.
- Hat sich ein Benutzer erfolgreich bei PubCookie angemeldet, so muss er dies 8 Stunden lang nicht mehr tun, es sei denn, er schließt den Browser.

Das dazu erstellte Test-Szenario besteht aus 7 VM-Servern mit folgender Einteilung:

(Noch unbekannte Begriffe und Abkürzungen werden in den folgenden Abschnitten ausführlich beschrieben und erklärt.)

lxshib01.lrz-muenchen.de: LMU Origin
lxshib02.lrz-muenchen.de: LMU Target
lxshib03.lrz-muenchen.de: WAYF
lxshib04.lrz-muenchen.de: LRZ Origin
lxshib05.lrz-muenchen.de: LDAP des LRZ
lxshib06.lrz-muenchen.de: TU Origin und Target
lxshib07.lrz-muenchen.de: LRZ Target

Wie man dieser Auflistung entnehmen kann, besteht die Föderation des Testszenarios aus 3 Relying Parties, nämlich der LMU, TU und dem LRZ. Siehe auch folgendes Schaubild:

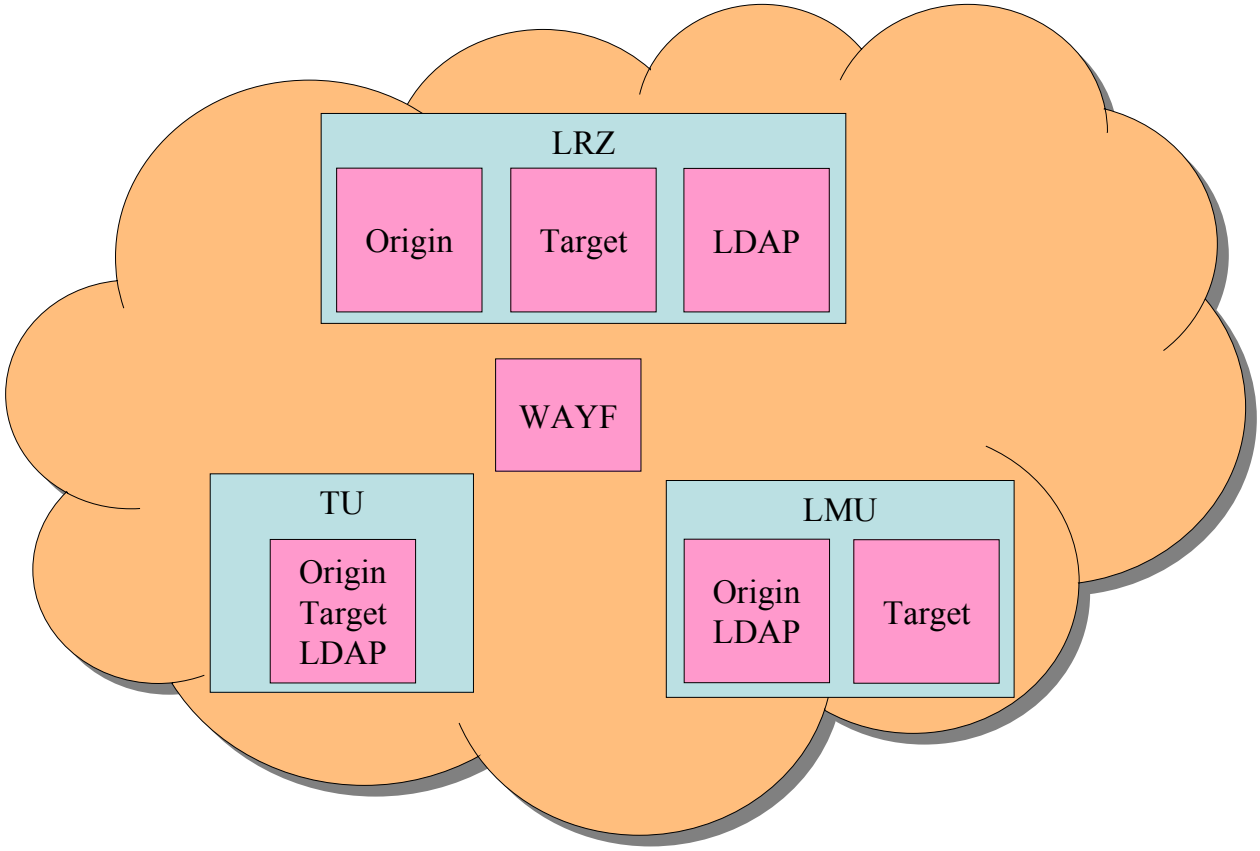

Abbildung 1: Shibboleth Testszenario

Dazu werden folgende Benutzer angelegt.

Benutzer der LMU:

UID: ebertm	PASSWD: filou	AFFILIATION: informatik
UID: zunhamme	PASSWD: sunny	AFFILIATION: informatik
UID: bwler	PASSWD: bwler	AFFILIATION: bwl

Benutzer des LRZ:

UID: hommelw	PASSWD: wolfi	AFFILIATION: informatik

Benutzer der TU:

UID: meyersm	PASSWD: horror	AFFILIATION: informatik
UID: kruegerf	PASSWD: horror	AFFILIATION: bwl

1.3 Begriffsklärung

Die Architektur von Shibboleth besteht aus mehreren Komponenten, die sowohl bei den Anbietern (Target), als auch bei den Identitätsverwaltern (Origin) angesiedelt sind (siehe Abbildung 2). Auf Seiten eines Origins werden dabei die folgenden beiden Komponenten betrieben:

- Handle Service (HS)

Der Handle Service ist für die lokale Authentifizierung des Nutzers zuständig. Dazu wird er an ein vorhandenes Authentifizierungssystem (im konkreten Fall PubCookie) angebunden. An dieser Stelle kann auch Single Sign-On realisiert werden. Nach erfolgreicher Authentifizierung liefert er an das Target ein Handle (User Reference) zurück.

- Attribute Authority (AA)

Die Attribute Authority nimmt Anfragen von Anbietern nach Attributen von Nutzern entgegen und beantwortet sie. Dabei werden nur solche Attribute weitergegeben, die aufgrund der für diesen Anbieter geltenden, ggf. nutzerspezifischen, Attribute Release Policy (ARP) übermittelt werden dürfen. In den ARPs werden benutzerspezifische Attributfreigaben eingetragen. Zur Ermittlung der Attribute bedient sich die AA dabei eines lokalen Verzeichnisdienstes (openLDAP).

Auf Seiten des Targets kommen folgende drei Komponenten zum Einsatz:

- Shibboleth Indexical Reference Establisher (SHIRE)

Aufgabe des Shibboleth Indexical Reference Establisher ist die Ermittlung eines Handle für den Nutzer mit Hilfe des zuständigen Handle Service, ggf. unter Mitwirkung des WAYF-Dienstes (siehe unten).

- Shibboleth Attribute Requester (SHAR)

Der Shibboleth Attribute Requester ermittelt mit Hilfe der zuständigen Attribute Authority autorisierungsrelevante Benutzerinformationen, überprüft und filtert diese ggf. anhand seiner Attribute Acceptance Policy (AAP) und leitet sie an den Resource Manager weiter.

- Resource Manager (RM)

Der Resource Manager entscheidet anhand der ihm bekannten Informationen und Attribute des Nutzers über die Gewährung des Zugriffs auf angebotene Dienste und sonstige Ressourcen.

Daneben gibt es noch den Where Are You From (WAYF) Dienst, der die Ermittlung des für einen Nutzer zuständigen Identitätsverwalters bzw. konkret des zuständigen Handle Service übernimmt.

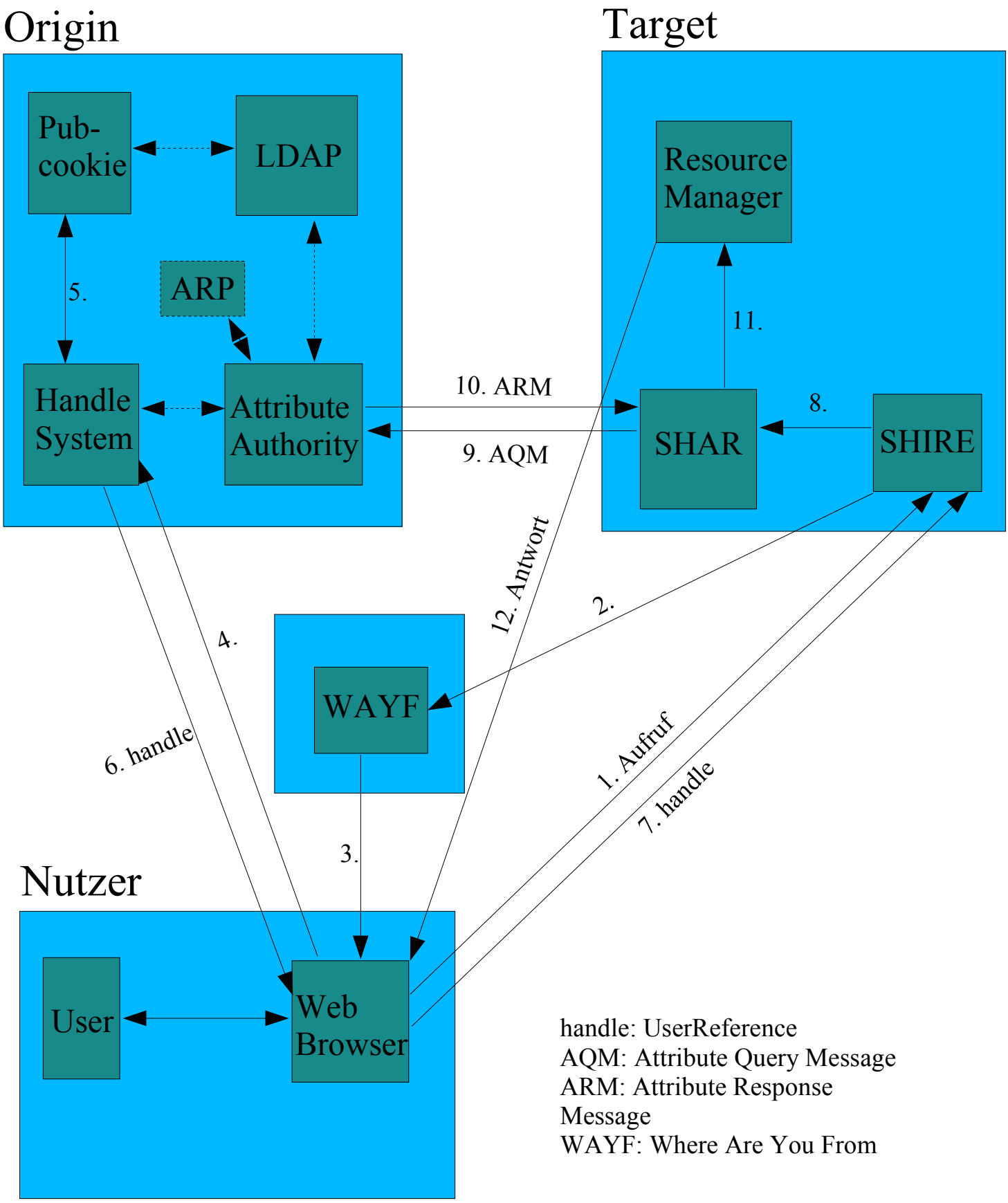

Abbildung 2:Übersicht über die Funktionsweise von Shibboleth

Im folgenden wird der Kommunikationsablauf aus Abbildung 2 genauer erläutert:

1:
Bei Aufruf eines vom User gewünschten Webdienstes wird die Anfrage zuerst an den ShibbolethIndexicalReferenceEstablisher (SHIRE) des Targets weitergeleitet.
2:
Diese Anfrage wird an das (zentrale) WhereAreYouFrom (WAYF) weitergegeben, welches das HandleSystem (HS) des Origins ausfindig machen muss. Dies geschieft durch eine manuelle Benutzereingabe.
3 / 4:
Ist dies geschehen wird das entsprechende HS kontaktiert um authentifizierungsrelevante Attribute zu erfragen..
5:
Hat sich der User noch nicht am Origin authentifiziert, geschieht dies im nächsten Schritt mit Hilfe des Authentifizierungssystems Pubcookie und dem Verzeichnisdienst openLDAP.
6 / 7:
Nach erfolgreicher Anmeldung wird ein entsprechendes Handle (übertragen mit openSAML) zurück an das SHIRE geschickt. In dieser Nachricht sind Informationen, etwa wo die AttributeAuthority (AA) des Users liegt.
8:
Dieses Handle wird an das ShibbolethAttributeRequester (SHAR) übergeben, welches mit Hilfe der somit gewonnenen Informationen eine AttributeQueryMessage (AQM) an die AA des Origins sendet. Ziel dabei ist die Erlangung von autorisierungsrelevanten Attributen des Users.
10:
Nachdem die AA unter Berücksichtigung der AttributeResolvePolicies (ARP) die geforderten Attribute in eine AttributeResponseMessage (ARM) gepackt hat, wird diese zurück an das SHAR gesendet.
11:
Dort angekommen werden alle erhaltenen Informationen an den ResourceManager (RM) weitergegeben, welcher nun entscheidet, ob dem User Zugriff gewährt wird - oder nicht.

1.4 Vorgehensweise und Probleme

Zur Einarbeitung in die Materie war die Gewinnung eines groben Überblicks über das Thema Web Single Sign-On, SAML, OpenSAML notwendig. Tieferes Verständnis erforderten die Begriffe und Konzepte von Shibboleth. Nach der Einarbeitung musste sämtliche für Shibboleth benötigte Software wie OpenSSL, OpenSAML, Apache, Java, Jakarta Tomcat, diverse Connectoren, PubCookie, OpenLDAP und NTP installiert werden. Das Zusammenspiel all dieser Komponenten erwies sich als sehr schwierig herzustellen, so dass dies bereits ein Großteil der Bearbeitungszeit verschlang.
Danach folgte eine erste Shibboleth-Origin-Installation, die mit Hilfe von InQueue online getestet wurde. InQueue ist ein von den Shibboleth-Entwicklern bereitgestelltes Portal zum Testen der selbst aufgesetzten Origins und Targets. Nach der Installations- und Testphase des ersten Origins begann die Installation und Konfiguration des ersten Shibboleth-Targets, ebenfalls mit Hilfe von InQueue. Um diese beiden Server von InQueue abzukoppeln war jetzt ein eigener WhereAreYouFrom notwendig. Deshalb musste im nächsten Schritt ein eigenes WAYF aufgesetzt werden. Nun konnten alle drei Komponenten aufeinander abgestimmt und einer intensiven Testphase unterzogen werden.

Nach Abschluss dieser Testphase wurden die Origin- und Targetinstallationen dupliziert um die bisherige Installation auf das verlangte Testszenario zu erweitern. Dieses besteht aus einer Föderation mit drei Relying Parties, welche vertrauenswürdige Informationen im Prinzip des Web Single Sign-On austauschen können.

Konkrete Probleme wurden bei
- der Anbindung von Tomcat an Apache mit SSL
- der Verbindung zwischen PubCookie und LDAP
- den Zertifikaten für den PubCookie-Server und Client
- der XML-Konfiguration für Shibboleth
- der NTP-Synchronisation des Testszenarios

festgestellt. Genaue Problembeschreibungen und deren Lösungen folgen im Verlauf dieser Ausarbeitung.

2 Benötigte Grundinstallationen für Shibboleth (Origin)

Im Folgenden erklären wir Schritt für Schritt die Installation und Konfiguration der für Shibboleth 1.2.1a vorausgesetzen Software.

Pfadangaben der Form **XXX_PATH** (z.B. APACHE_PATH) sind undefinierte Umgebungsvariablen, die geeignet zu definieren und dann konsequent einzuhalten sind.

Kopieren folgender Programme in das Verzeichnis /root/requirements:

Befehl: `scp -r FILE root@server.lrz-muenchen.de:/root/requirements`

- Apache 1.3.33 (Webserver) Siehe Literaturverzeichnis [1].
- Java SDK 1.4.2_04 (Programmier-Entwicklungsumgebung) Siehe Literaturverzeichnis [4].
- Tomcat 4.1.27 (Servlet-Container) Siehe Literaturverzeichnis [2].
- mod_ssl 8.22 (Connector zwischen Apache und openSSL) Siehe Literaturverzeichnis [3].
- mod_jk 1.2.6 (Connector zwischen Apache und Tomcat) Siehe Literaturverzeichnis [3].
- openssl 0.9.7e (Verschlüsselungssoftware)
- PubCookie 3.1.1 (Authentisierungssoftware) Siehe Literaturverzeichnis [5].

Entpacken der jeweiligen Programme in das Verzeichnis /root/requirements:

Befehl: `tar -zxvf FILE`

Installieren der Programme:

- **Java SDK 1.4.2_04**
  Siehe Literaturverzeichnis [4].
  Befehle: Das RPM-Package mit YaST installieren, danach folgende Umgebungsvariablen setzen:

```
PATH=/usr/java/j2sdk1.4.2_04/bin:$PATH
```

```
JAVA_HOME=/usr/java/j2sdk1.4.2_04
```

- **Tomcat 4.1.27**
  Siehe Literaturverzeichnis [2].
  Kopieren des entpackten Programms aus unserem Requirements-Verzeichnis in das gewünschte Verzeichnis, in dem später Tomcat laufen soll: (bei uns war es /usr/jakarta-tomcat-4.1.31)

  Befehle:

```
cp /root/requirements/jakarta-tomcat-4.1.31 TOMCAT_PATH

cd TOMCAT_PATH/conf

mkdir auto
```

- **openssl 0.9.7e**

  In das entpackte Verzeichnis wechseln und dann:

```
./config

make

make install

PATH=/OPEN_SSL_PATH:$PATH
export PATH
```

  Zur Erstellung eines neuen SSL-Schlüssels und des zugehörigen Zertifikates sind folgende Befehle notwentig:

```
openssl genrsa -des3 1024 > /KEY_PATH/DNS_NAME.key

openssl req -new -key /KEY_PATH/DNS_NAME.key > /KEY_PATH/
DNS_NAME.csr
```

  Hierbei ist zu beachten, dass in der Abfrage als Common Name der DNS_NAME eingegeben wird.

```
openssl req -x509 -key /KEY_PATH/DNS_NAME.key -in /KEY_PATH/
DNS_NAME.csr > /KEY_PATH/DNS_NAME.crt

./configure --with-apache=/APACHE_SOURCE_PATH --with-
crt=/KEY_PATH/DNS_NAME.crt --with- key=/KEY_PATH/DNS_NAME.key
```

  (`openssl rsa -in SERVER.key -out UNENCRYPTED.key` um Pass-Phrase zu entfernen)

- **Apache 1.3.33**
  Siehe Literaturverzeichnis [1].
  Anbindung von openSSL an Apache und Apache-Installation:

Befehle:

```
cd /root/requirements/apache_1.3.33

export SSL_BASE=/root/requirements/openssl-0.9.7e

./configure --enable-module=ssl --prefix=/APACHE_PATH –
enable-shared=ssl

make
```

Wenn bei der Ausführung von make folgender Fehler auftritt:

```
...
flex -Pssl_expr_yy -s -B ssl_expr_scan_l
make[4]: flex: Command not found
...
```

kann dieser durch Download (ftp://ftp.gnu.org/non-gnu/flex/) und Installation von flex-2.5.4a (Parser-Programm) behoben werden.

```
make install
```

Anbindung von Tomcat an Apache:

Siehe Literaturverzeichnis [3].

Befehle:

```
cd /root/requirements/jakarta-tomcat-connectors-jk-1.2.6-
src/jk/native

./buildconf.sh

./configure --with-apxs=/APACHE_PATH/bin/apxs --enable-EAPI

make

make install

cd TOMCAT_PATH/conf

cp -p server.xml server.xml.ORIG
```

In der Datei **server.xml**:

Alle Vorkommen von `localhost` werden mit DNS_NAME ersetzt.

Unter der Zeile

```
<Server port="8005" shutdown="SHUTDOWN" debug="0">
```

wird folgende Zeile eingefügt:

```
<Listener
className="org.apache.ajp.tomcat.config.ApacheConfig"
modJk="/APACHE_PATH/libexec/mod_jk.so" />
```

Nach der Zeile

```
<Host name="DNS_NAME" debug="0" appBase="webapps"
unpackWARs="true" autoDeploy="true">
```

wird folgende Zeile eingefügt:

```
<Listener className="
org.apache.ajp.tomcat.config.ApacheConfig"
modJk="/APACHE_PATH/libexec/mod_jk.so" />
```

In der Datei **httpd.conf** (/APACHE_PATH/conf)

ganz unten Folgendes hinzufügen (für http):

```
Include /TOMCAT_PATH/conf/auto/mod_jk.conf
```

Im Pfad /TOMCAT_PATH/conf

muss ein "worker" angelegt werden, der Anfragen von Apache kommend übernimmt:

```
mkdir jk

cd jk
```

Hier die Datei **workers.properties** mit folgendem Inhalt anlegen:

Diese Zeilen müssen eingefügt werden:

```
worker.list=ajp13
worker.ajp13.port=8009
worker.ajp13.host=DNS_NAME
worker.ajp13.type=ajp13
```

Apache, SSL und Tomcat:

Damit Apache+**SSL** und **Tomcat** funktioniert muss Folgendes getan werden:

Bei den anderen `IfModule`-Tags nach der `AddModule`-Liste (in httpd.conf) Folgendes hinzufügen:

```
<IfModule !mod_jk.c>
     LoadModule jk_module "APACHE_PATH/libexec/mod_jk"
</IfModule>
JkWorkersFile "TOMCAT_PATH/conf/jk/workers.properties"
JkLogFile "TOMCAT_PATH/logs/mod_jk.log"
JkLogLevel emerg
```

Nun im `VirtualHost`-Tag für SSL als Test Folgendes hinzufügen:

Copy and paste aus der Datei /TOMCAT_PATH/conf/auto/mod_jk.conf alles was in `##### localhost:/examples #####` steht.

Konfiguration:

Apache:

In der Datei **httpd.conf**

in der Zeile `ServerName` die "Example-Adresse" auf DNS_NAME des Rechners abändern.

Erstellen einer ersten kleinen Web-Site:

In der httpd.conf die index.html zu TEST_SEITE.html geändert.

Firewall-Einstellungen:

SuSEfirewall2:

hier haben wir http mit YAST2 freigeschalten.

Zusätzliche Kommandozeilen-Befehle:

- `rcSuSEfirewall2 [start,stop,restart]`

Test:

Erst Tomcat starten: `./startup.sh`
Dann Apache mit SSL starten: `./apachectl startssl`

Zum Testen nun den Konqueror öffnen und dort eingeben:

http://lxshib01.lrz-muenchen.de : wenn es geht läuft der Apache
http://lxshib01.lrz-muenchen.de/examples :wenn es geht läuft Apache mit Tomcat
https://lxshib01.lrz-muenchen.de : wenn das geht läuft Apache mit SSL
https://lxshib01.lrz-muenchen.de/examples : wenn das nun geht läuft Apache mit SSL und Tomcat und wir sind soweit zufrieden.

NTP:

Unter Anderem setzt PubCookie einen laufenden NTP-Daemon voraus. Unter Linux läuft dieser normalerweise schon, man muss ihn aber noch konfigurieren.
NTP steht für Network Time Protocol und synchronisiert die Systemzeit mit Hilfe eines entfernten Zeitservers.
In unserem Fall mussten wir in der Yast-Config von NTP die lokale Systemuhr aus der Liste von Servern entfernen. NTP-Server sind bei uns ntp1.lrz-muenchen.de und ntp2.lrz-muenchen.de. Damit bekommt man auf allen benötigten Systemen eine einheitliche Zeiteinstellung.

NTP führt eine "drift"-Datei, in der angepasst wird, wie schnell der Rechner zu schnell oder zu langsam läuft, entsprechend wird die Zeit dann reguliert. Leider ist "unsere" lxshib01-Uhr dennoch nicht korrekt.

Um dies einigermaßen zu regulieren helfen folgende Befehle:

`ntpq` (for monitoring)

`minpoll` und `maxpoll` zwischen 4 and 17 einstellen(Zeitintervall in dem die Zeitserver angepollt werden, gemessen in Quadratsekunden)

Befehl zum Prüfen der Zeit in der Kommandoziele: `clock`

Befehl zur Zeiteinstellung (der Hardware-Uhr): `hwclock -set -date hh:mm:ss`

NTP-Daemon:
/usr/sbin/ntpd (started den ntpd-Daemon. MUSS UNBEDINGT AUSGEFÜHRT WERDEN, wenn man ohne cronjobs arbeitet. cron siehe unten)

/etc/ntp.conf (Konfigurations-Datei)

/var/log/ntp (logfile)

/var/lib/ntp/etc/ntp.conf (logfile)

Die durch VMware realisierten Server hatten große Abweichungen in der Zeitberechnung. Manche gingen in einer realen Stunde bis zu 2 Stunden vor bzw. nach.
Folgende Maßnahmen schafften es, dieses Problem zu umgehen:

NTP-Prozess beenden (auch restart im YAST ausschalten)

Stattdessen mit Hilfe von **cron** im periodischen Abstand den Befehl `ntpdate` ausführen:

`crontab -l` zeigt die momentanen Jobs an
`crontab -e` öffnet die Datei, in die man folgende Zeile einträgt:

```
  * * * * *   /usr/sbin/ntpdate ntp1.lrz-muenchen.de
```

`crontab -l` zum überprüfen, ob Job akzeptiert wurde.

Somit wird nun jede Minute die Zeit durch ntpdate über den Server ntp1.lrz-muenchen.de aktualisiert.

Statt den fünf Sternen kann man auch Zahlen eintragen, die je für Folgendes stehen:

`[Minuten] [Stunden] [Tag im Monat] [Monat] [Tag der Woche]`

Protokolliert wird in /var/mail/root .

LDAP:

Siehe Literaturverzeichnis [6], [7].
LDAP steht für Lightweight Directory Access Protocol und stellt eine Menge von Protokollen für den Zugriff auf hierarchische Verzeichnisse.

Installation:
Mit Hilfe von YAST: openldap2, openldap2-client, openldap2-devel

In der Datei **slapd.conf** (/etc/openldap) ergeben sich folgende Einstellungen:

```
database  bdb
suffix    "dc=lxshib01,dc=lrz-muenchen,dc=de"
rootdn    "cn=Manager,dc=lxshib01,dc=lrz-muenchen,dc=de"
rootpw    secret
directory /var/lib/ldap (nach Belieben)
```

Für LDAP V2 muss folgende Zeile eingefügt werden:

```
allow     bind_v2
```

In ldap.conf diese "Base" angeben:

```
BASE dc=lxshib01,dc=lrz-muenchen,dc=de
```

Starten des Daemons slapd: (das angegebene Directory muss zu Beginn vorhanden und leer sein)

In /usr/lib/openldap

```
./slapd ("-d 255" hinzufügen für debugging)
```

Test, ob der Daemon läuft:

```
ldapsearch -x -b '' -s base ‘(objectclass=*)' namingContexts
```

LDAP-Einträge werden in sogenannten LDIF-Dateien gespeichert. LDIF bedeutet "LDAP Data Interchange Format“.
Erstellen des ersten LDIF-Eintrags:

z.B. in /var/lib/ldap die Datei **first_entry.ldif** mit folgenden Zeilen anlegen:

```
dn:           dc=lxshib01,dc=lrz-muenchen,dc=de
objectclass:  dcObject
objectclass:  organization
o:            LMU
dc:           lxshib01

dn:           cn=mustermann,dc=lxshib01,dc=lrz-
              muenchen,dc=de
objectclass:  inetOrgPerson
cn:           mustermann
```

```
sn:             mann
uid:            Test_User_01 (Wichtig: diese Form der uid muss vorerst
                                eingehalten werden)
userPassword:   irgendwas
mail:           muster@web.de
departmentNumber:16
o:              LMU
ou:             IFI
```

Durch die Leerzeile werden die einzelnen Einträgen voneinander getrennt.
Dn bedeutet distinguished name und bezeichnet die Wurzel jedes Eintrags.
Objectclass ist das Schema für den jeweiligen Eintrag. Es legt fest, welche Attribute verwendet werden können.

Nun muss das ldif hinzugefügt werden:

```
ldapadd -x -D "cn=Manager,dc=lxshib01,dc=lrz-muenchen,dc=de"
-W -f /var/lib/ldap/first_entry.ldif
```

Test, ob es zur DB hinzugefügt wurde:

```
ldapsearch -x -b ‘dc=lxshib01,dc=lrz-muenchen,dc=de'
‘(objectclass=*)'
```

Einträge löschen:

Eine loeschen.ldif Datei anlegen, in der der komplette dn des zu löschenden Eintrags steht, z.B.:

```
dn:             cn=mustermann,dc=lxshib01,dc=lrz-
                muenchen,dc=de
```

Dann:

```
ldapdelete -f /PATH_TO_LDIF/loeschen.ldif -x -D
"cn=Manager,dc=lxshib0x,dc=lrz-muenchen,dc=de" -W
```

Für einen benutzerfreundlicheren Zugriff wurde ein LDAP BROWSER/EDITOR installiert:

Man folge den Anleitungen auf http://www-unix.mcs.anl.gov/~gawor/ldap

Nach der Installation starten mit `./lbe`
Dann bei Quickconnect als Host lxshib01.lrz-muenchen.de
und als Base-Dn dc=lxshib01,dc=lrz-muenchen,dc=de Port 389 eingeben.

Ordnungsgemäßes Stoppen des slapd-Daemons:

```
kill -INT 'cat /var/run/slapd/slapd.pid'
```

oder:

`ps -awx` (siehe dort ps-Nummer)
`kill <ps-Nummer>`

Will man die ObjektKlasse ***eduPerson*** benutzen, so muss man sich das entsprechende Schema aus dem Internet laden und zu den anderen Schema-Dateien kopieren.

Download unter: http://lab.ac.uab.edu/vnet/documents/ldif/eduperson.schema

Kopieren nach /etc/openldap/schema/eduperson.schema

Danach muss man in der slapd.conf eine entsprechende include-Zeile hinzufügen, die auf die Schema-Datei verweist.

Schema:
`objectclass: eduPerson`

Attribute:
```
eduPersonAffiliation
eduPersonNickname
eduPersonOrgDN
eduPersonOrgUnitDN
eduPersonPrimaryAffiliation
eduPersonPrincipalName
eduPersonEntitlement
eduPersonPrimaryOrgUnitDN
```

Achtung:

In der späteren Testumgebung wird mit der Objektklasse **eduPerson** gearbeitet. Verwendet werden dabei die Attribute **eduPersonAffiliation** und **eduPersonPrimaryAffiliation**, um durch Shibboleth Autorisierung zu erlangen.

PubCookie 3.1.1:

Siehe Literaturverzeichnis [5].
Der PubCookie Login Server nutzt die index.cgi, welche die zentrale Komponente von PubCookie darstellt und zur Browserinteraktion mit dem Endnutzer dient. Zur Erstellung von symmetrischen Schlüsseln verwendet der Login Server den sogenannten Keyserver.

PubCookie (Server)
Im entpackten Source-Verzeichnis von PubCookie:

```
./configure --enable-ldap --enable-login --disable-apache --
prefix=PubCookie_PATH

make
make install
```

Schlüssel:
Für den KeyClient und den KeyServer können wir unsere bereits vorhandenen Apache-keys benutzen (siehe Schlüsselerzeugung auf Seite 15). Davor entfernen wir mit folgendem Befehl die Passphrase, da PubCookie die Aufforderung zur Passphrase nicht unterstützt:

```
openssl rsa -in SERVER.key -out UNENCRYPTED.key

mv UNENCRYPTED.key PubCookie_PATH/keys
```

(Die Verzeichnisse in denen die Schlüssel liegen sollten mit entsprechenden Zugriffsrechten versehen werden.)

Für das Siginieren und Verifizieren von Cookies benötigt PubCookie ein weiteres Keypair:

In PUBKOOKIE_PATH/keys:

```
openssl req -new -x509 -out PubCookie_GRANTING.cert -newkey
rsa:1024 -nodes -keyout PUBKOOKIE_GRANTING.key
```

Konfiguration:

In der Datei PUBCOOKIE_PATH/**config:**

```
basic_verifier: ldap

ldap_uri: ldap://lxshib01.lrz-muenchen.de/dc=lxshib01%
      2cdc=lrz-muenchen%2cdc=de???(uid=%s)?x-
      BindDN=cn=Manager%2cdc=lxshib01%2cdc=lrz-muenchen%
      2cdc=de,x-Password=secret
```

(Die URI ist folgendermaßen aufgebaut:
ldap://<DNS_NAME>/
<ROOT_OF_DATABASE>???
(uid=<hier wird automatisch die Benutzer-Uid hinzugefügt>)?
x-BindDN=<ROOT-DN>
x-Password=<PASSWORD>
Man beachte, dass Kommas mit %2c und Leerzeichen mit %20 dargestellt werden.)

```
ssl_key_file: /PUBKOOKIE_PATH/unencrypted.key
ssl_cert_file: /APACHE_PATH/conf/ssl.key/SERVER.crt

granting_key_file: /PubCookie_PATH/keys /
PubCookie_GRANTING.key
granting_cert_file: /PubCookie_PATH/keys/
PubCookie_GRANTING.crt
```

Weiterhin müssen in dieser Datei die Einträge "server config" und die "keyserver config" angepasst werden (für das CA_Bundle_file nehmen wir unser Apache-Zertifikat her).

Es muss unbedingt darauf geachtet werden, dass Apache und PubCookie nicht die identischen SSL-Session-Files benutzen. Um nicht durch Dateisperren ungewollte Fehler zu bekommen, müssen die Schlüssel und Zertifikate zumindest umbenannt werden.

Danach muss unter /etc/xinetd.d ein neuer Service für den keyserver angelegt werden: xinetd ist ein erweiterter Internet Service Daemon, welcher Internetservices, wie z.B. unseren keyserver verwaltet.

Hier die Datei **keyserver** erstellen und Folgendes einfügen:

```
service keyserver
{
     type           = UNLISTED
     protocol       = tcp
     port           = 2222
     disable        = no
     socket_type    = stream
     wait           = no
     user           = root
     group          = tty
     server         /PubCookie_PATH/keyserver
}
```

Danach xinetd neu starten: `rcxinetd restart`

```
cp /PubCookie_PATH/starter.key  /PubCookie_PATH/keys/
ABGEFRAGTER_NAME_DEN_WIR_BEIM_ERSTELLEN_DES_KEYS_FESTGELEGT_H
ABEN
```

Es folgen Änderungen in der **httpd.conf**:

- Zu DirectoryIndex `index.cgi` vor `index.html` hinzufügen.

- Folgende Zeile einkommentieren:
`AddHandler cgi-script .cgi`
- bei jedem Pfad (in httpd.conf <Directory>), in dem CGI ausgeführt werden soll, muss bei `options ExecCGI` hinzugefügt werden. Man kann das auch gleich standardmäßig im `dafault-directory <directory />` festlegen.

Sollte es nun nicht funktionieren, müssen folgende Änderungen in der **hosts.allow** vorgenommen werden:

(hosts.allow ist eine Konfigurationsdatei für tcp-wrappers)

in /etc/hosts.allow muss der Keyserver freigeschaltet werden. Dazu am Ende einfügen:

```
keyserver:ALL:ALLOW
keyclient:ALL:ALLOW
slapd:ALL:ALLOW
```

Natürlich muss hier nicht alles (ALL) erlaubt werden, doch für das zu erreichende Testszenario sollte diese grobe Einstellung genügen.

PubCookie (Module)

Das PubCookie Modul dient der Verbindung zwischen Apache und dem PubCookie Server.

Im entpackten Source-Verzeichnis von PubCookie:

```
./configure --enable-apache --prefix=MOD_PubCookie_PATH --
with-apxs=/APACHE_PATH/bin --with-ssl=/OPENSSL_PATH

make

make install
```

Konfiguration des Keyclients:

Der Keyclient ist ein Programm, welches den Keyserver kontaktiert um einen symmetrischen Schlüssel zu erhalten.

Im Pfad /MOD_PubCookie_PATH die Datei **config** so ändern:

Einfügen folgender Zeilen:

```
#ssl-config
ssl_key_file: /PubCookie_PATH/keys/DNS_NAME.key
ssl_cert_file: /PubCookie_PATH/keys/DNS_NAME.crt
```

Hier dürfen auf keinen Fall die selben Files wie in der httpd.conf benutzt werden.
Um Dateisperren zu vermeiden, wurden die Apache-Keys und -Certs in das PubCookie-Verzeichnis kopiert. In der offiziellen Anleitung steht aber sowieso, dass man ein zweites Keypair erstellen soll.

```
#keyclient-specific config
keymgt_uri: https://DNS_NAME_DES_KEYSERVERS:2222
ssl_ca_file: /APACHE_PATH/conf/ssl.key/DNS_NAME.key
```

Hier kann man ohne Probleme das Apache-Keyfile benutzen.

Jetzt `./keyclient` ausführen

Download des PubCookie "Granting" Zertifikats:

```
./keyclient -G /MOD_PubCookie_PATH/keys/DNS_NAME.crt
```

In **httpd.conf**:

Im Tag `<Directory "/DIRECTORY_ROOT">`:

```
AllowOverride AuthConfig
```

Hinzufügen folgender Zeile unter "LoadModule ssl_module libexec/libssl.so":
"LoadModule PubCookie_module libexec/mod_PubCookie.so"

Hinzufügen folgender Zeile unter "AddModule mod_ssl.c":
"AddModule mod_PubCookie.c"

Hinzufügen folgender Zeilen am Ende des Files:

```
<IfDefine SSL>
<IfModule mod_PubCookie.c>
#
# PubCookie configuration section
#
PubCookieGrantingCertFile /PubCookie_PATH/
keys/PubCookie_granting.cert
PubCookieSessionKeyFile /PATH_OF_APACHE_KEYFILE/DNS_NAME.key
PubCookieSessionCertFile /PATH_OF_APACHE_CERTFILE /
DNS_NAME.crt
PubCookieLogin https://DNSNAME/
PubCookieDomain .DOMAIN_OF_DNSNAME
PubCookieKeyDir /PubCookie_PATH/keys/
PubCookieAuthTypeNames uid

# Disable inactivity timeout by default
<Directory "/DOCUMENT_ROOT">
PubCookieInactiveExpire -1
</Directory>

</IfModule>
</IfDefine>
```

Testen der Einstellungen:

```
./apachectl startssl
```

In /APACHE_DIRECTORY_ROOT/:
```
mkdir test

cd test
```
Hier eine beliebige Index-Seite (index.html) erstellen.

Dann eine Datei .htaccess erstellen.

Einfügen folgender Zeilen:
```
AuthType uid
require valid-user
```

Browser öffnen und URL testen:

https://DNS_NAME/test

Bei erfolgreicher Einrichtung von PubCookie muss nun ein entsprechendes Interface erscheinen, in dem Benutzername und Passwort angegeben werden müssen.

3 Shibboleth Origin Installation

Siehe Literaturverzeichnis [8].
Nachdem alle für das Shibboleth Origin benötigte Komponenten eingerichtet sind, kann nun mit dessen Installation begonnen werden.

3.1 Installation:

Entpacken des .tarballs:

```
cd /opt
tar -zxvf /TARBALL_PATH/shibboleth-origin-1.2.1.tar.gz
```

Kopieren der Java-Dateien nach Tomcat:

```
cp /opt/shibboleth-origin-1.2.1/dist/shibboleth.war /
TOMCAT_PATH/webapps/
```

Um Probleme mit der JVM zu vermeiden:

```
cp /opt/shibboleth-origin-1.2.1/endorsed/*.jar /
TOMCAT_PATH/common/endorsed
```

Hierbei muss man auf jeden Fall sichergehen, dass die Xerces-Dateien in den 'endorsed'-Ordner kopiert werden. Sollten in diesem Ordner schon Dateien vorhanden sein, so müssen diese gelöscht werden.

Restart Tomcat:

```
cd /TOMCAT_PATH/bin
./shutdown.sh
./startup.sh
```

Damit Apache die Url's für das Shibboleth HS und AA nach Tomcat mappt:

in der httpd.conf, direkt unter die anderen JkMounts:

```
JkMount /shibboleth/*   ajp13
```

Änderungen in der server.xml (/TOMCAT_PATH/conf)

im `<Ajp13Connector>`-Tag Folgendes hinzufügen:
`address="DNS_NAME_DES_SERVERS"` (hier muss DNS eingegeben werden nicht 127.0.0.1 !)
`tomcatAuthentication="false"`

Da mbeans nicht benötigt wird, können alle mbeans-Tags auskommentiert werden.

Um das AA mit SSL zu schützen, Folgendes in die httpd.conf eintragen: (direkt über die vorher eingefügte JkMount-Zeile)

```
<Location /shibboleth/AA>
#SSLVerifyClient optional
#SSLOptions +StdEnvVars +ExportCertData
</Location>
```

#SSL steht so in der Anleitung drin, haben wir aber auskommentiert, da es bei uns eh im <IfDefine SSL> -Tag steht.

```
<Location /shibboleth/HS>
     AuthType uid
     require valid-user
</Location>
```

3.2 Grundkonfiguration:

Nach der erfolgreichen Installation von Shibboleth/Origin muss man zum Testen des Origins folgendes Formular ausfüllen, um bei InQueue Mitglied zu werden:

(InQueue stellt diverse Targets, WAYFs, Origins, etc. zum Testen bereit)

http://inqueue.internet2.edu/join.html

hier folgendes Eintragen:

- lrz-muenchen.de (hier darauf achten, dass wirklich die Domain angegeben wird und nicht der Hostname. Das haben wir zuerst gemacht, und das hat uns lange Sorgen bereitet)
- https://lxshib01.lrz-muenchen.de/shibboleth/HS
- https://lxshib01.lrz-muenchen.de/shibboleth/AA
- lxshib01.lrz-muenchen.de
- Ludwig Maximilians Universitaet Muenchen
- diverse Email-Adressen
- https://lxshib01.lrz-muenchen.de/lmu.html
- eine Beschreibung zum Verwendungszweck

Somit wird das Origin an das bestehende InQueue angehängt. Danach erhält man folgende XML-Datei:

```
<OriginSite ErrorURL="https://lxshib01.lrz-muenchen.de/lmu.html"
               Name="urn:mace:inqueue:lxshib01.lrz-muenchen.de">

<Alias>Ludwig Maximilians Universitaet Muenchen</Alias>

<Contact Email="sebastian.zunhammer@stusta.mhn.de" Name="Zunhammer"
Type="technical"/>

<Contact Email="ebertm@cip.ifi.lmu.de" Name="Ebert"
Type="administrative"/>
```

```
<HandleService Location="https://lxshib01.lrz-
muenchen.de/shibboleth/HS" Name="lxshib01.lrz-muenchen.de"/>

<AttributeAuthority Location="https://lxshib01.lrz-
muenchen.de/shibboleth/AA" Name="lxshib01.lrz-muenchen.de"/>

<Domain regexp="false">lxshib01.lrz-muenchen.de</Domain>
```

in **origin.xml:**
(/TOMCAT_PATH/webapps/shibboleth/WEB-INF/classes/conf)

```
AAUrl="https://lxshib01.lrz-muenchen.de/shibboleth/AA"

defaultRelyingParty="urn:mace:inqueue"

providerId="urn:mace:inqueue:lxshib01.lrz-muenchen.de"
```

Dann die zweite RelyingParty einkommentieren und dort gegebenenfalls die `ProviderId` anpassen.

Danach das FileResolver-Element wie folgt abgeändert:

```
<Credentials xmlns="urn:mace:shibboleth:credentials:1.0">
    <FileResolver Id="inqueue_cred">
          <Key format="PEM">
              <Path>file:///usr/apache_1.3.33/conf/ssl.key/lxshib01
                    .lrz-muenchen.de.key</Path>
          </Key>
          <Certificate format="PEM">
              <Path>file:///usr/apache_1.3.33/conf/ssl.crt/lxshib01
                    .lrz-muenchen.de.crt</Path>
          </Certificate>
   </FileResolver>
</Credentials>
```

Zum Schluss muss noch der für InQueue zuständige FederationProvider einkommentiert werden.

Jetzt muss Apache und Tomcat neu gestartet werden.

Attribute Repository benutzen:
(hier wird festgelegt, welche Attribute an wen herausgegeben werden)

Ändern der **origin.xml**:
das Attribut `resolverConfig` muss geändert werden, so dass es auf /conf/resolver.ldap.xml zeigt.

Ändern der **resolver.ldap.xml**:

- Die Elemente eduPersonScopedAffiliation und eduPersonPrincipalName einkommentieren und bei den jeweiligen smartScope Attributen den DNSNAME eintragen.
- Einkommentieren des `JNDIDirectoryDataConnector` Elements
- Das Search Element wie folgt ändern: `filter uid=%PRINCIPAL%`
- Für das annonyme Binden des LDAP-directories folgende Elemente löschen: `java.naming.security.principal` und `java.naming.security.credentials` Property
- Ändern des Wertes des Elements `java.naming.provider.url` Property nach `ldap://DNS_NAME/dc=...,dc=...` .

Metadatatool nutzen:

Metadatatool ist ein Werkzeug, mit dem man sich die aktuellen trust.xml Dateien von InQueue herunterladen kann.

```
export SHIB_HOME=/opt/shibboleth-origin-1.2.1/

cd /opt/shibboleth-origin-1.2.1/bin/

./metadatatool -i https://wayf.internet2.edu/InQueue/sites.xml -k /
opt/shibboleth-origin-1.2.1/conf/internet2.jks -p shib123 -a
sitesigner -o /usr/jakarta-tomcat-4.1.31/webapps/shibboleth/WEB-
INF/classes/conf/IQ-sites.xml
```

Nun mit folgendem Link testen: (nach Neustart von Tomcat und Apache)

https://wayf.internet2.edu/InQueue/sample.jsp

4 Benötigte Grundinstallationen für Shibboleth (Target)

Siehe Literaturverzeichnis [9].
Nach dem erfolgreichen Testen des Origins kann nun mit dem Target und den damit verbundenen Softwarevoraussetzungen weitergemacht werden.

Für den folgenden Abschnitt benötigte Pakete findet man unter http://wayf.internet2.edu/shibboleth/.

- Für das Target werden Apache 1.3.33 und openSSL benötigt. Entsprechende Installation und Konfiguration:
  siehe *Benötigte Grundinstallationen für Shibboleth (Origin).*

- Installation von log4cpp (c++Bibliothek für flexibles Logging):
  Siehe Literaturverzeichnis [14].
  `tar -zxvf FILE_NAME`

  `./configure --with-pthreads`

```
make

make install
```

- Installation von Xerces (Xerces ist ein XML-Parser):

  mit Yast xerces-2.5.0 und xerces-2.5.0-devel installieren

- Installation von libxml2-devel:

  mit Yast installieren

- Installation von xsec (Bibliothek von Sicherheitsstandards für XML):
  Paket ist unter http://xml.apache.org/security/dist/c-library/ verfügbar.

```
cd /SOURCE_PATH

./configure

make

make install
```

- Installation von libcurl:
  Curl ist ein Kommandozeilentool zum URL-basierten Datenaustausch.
  curl-devel mit YAST installieren

- Installation von openSAML:
  Siehe Literaturverzeichnis [12].

```
./configure

make

make install
```

5 Shibboleth Target Installation

Nun kann mit der Installation des Shibboleth-Targets begonnen werden.

5.1 Installation:

Für eine Checkliste, die man sehr gut abarbeiten kann: Siehe Literaturverzeichnis [10].

Entpacken des .tarballs:

```
cd /opt
tar -zxvf /TARBALL_PATH/shibboleth-1.2.1.tar.gz
```

Nun in das entpackte Verzeichnis wechseln und:

```
./configure --prefix=/opt/shibboleth-target-1.2.1 --enable-apache-
13 --with-apxs=/APACHE_PATH/bin/apxs

make

make install
```

5.2 Grundkonfiguration:

In **httpd.conf** am Ende einfügen:
`Include /opt/shibboleth-target-1.2.1/etc/shibboleth/apache.config`

Die Apache-Config-Datei für Shibboleth befindet sich unter:
`/opt/shibboleth-target-1.2.1/etc/shibboleth/`**`apache.config`**
In dieser Datei wird der zu schützende Ordner auf shibboleth abgeändert (nicht secure).

Shire Log:
`/opt/shibboleth-target-1.2.1/var/log/shibboleth/`**`shire.log`**

Das SHAR muss **vor** Apache gestartet werden, also fügen wir es einfach in das Apache-Start-Script ein (apachectl):
`/opt/shibboleth-target-1.2.1/bin/`**`shar`** `-f &`

im stop-Tag entsprechend den Befehl `killall shar` eingeben, damit beim Beenden von Apache auch das SHAR beendet wird.

Konfiguration von **shibboleth.xml**:

(/opt/shibboleth-target-1.2.1/etc/shibboleth/)

Im Tag Applications haben wir folgende providerId festelegt:

```
providerId="https://lxshib02.lrz-muenchen.de/shibboleth/"
```

Im Tag Applications gibt es einen Tag FederationProvider, der folgendermaßen geändert werden muss:

```
<FederationProvider type="edu.internet2.middleware.shibboleth.common.
                                    provider.XMLMetadata">
  <SiteGroup Name="https://lxshib02.lrz-muenchen.de/shibboleth"xmlns=
                                    "urn:mace:shibboleth:1.0">
    <OriginSite Name="https://lxshib01.lrz-muenchen.de">
      <Alias>LMU ORIGIN</Alias>
      <Contact Type="technical" Name="WASTL+MATZE" Email="root@lxshib01.
                                    lrz-muenchen.de"/>
      <HandleService Location="https://lxshib01.lrz-muenchen.de/
                                    shibboleth/HS" Name="lxshib01"/>
      <AttributeAuthority Location="https://lxshib01.lrz-muenchen.de/
                                    shibboleth/AA" Name="lxshib01"/>
      <Domain>lrz-muenchen.de</Domain>
    </OriginSite>
```

```
  </SiteGroup>
</FederationProvider>
```

Weiterhin muss man in Tag `Credentials` die Einträge auf die Apache-Keys abändern.

Anmeldung des Service Providers bei InQueue:
Siehe Literaturverzeichnis [11].
http://inqueue.internet2.edu/join.html

Folgendes in die Felder eintragen:

- `LMU Service Provider`
- `https://lxshib02.lrz-muenchen.de/shibboleth`
- `lxshib02.lrz-muenchen.de (dies ist der Name des Keys. Wir nehmen hier einfach die schon eingerichteten Apache-Keys her)`
- `Namen und e-mails`
- `https://lxshib02.lrz-muenchen.de/Shibboleth.shire`
- `Verwendungszweck`

Erhaltene Metadata:

```
<DestinationSite Name"https://lxshib02.lrz-muenchen.de/shibboleth">
    <alias>LMU Service Provider</Alias>
    <Contact Email="sebastian.zunhammer@stusta.mhn.de"
     Name="Sebastian Zunhammer" Type="technical"/>
    <Contact Email="ebertm@cip.ifi.lmu.de" Name="Matthias Ebert"
     Type="administrative"/>
    <assertionConsumerServiceURL Location="https://lxshib02.lrz-
     muenchen.de/Shibboleth.shire"/>
    <attributeRequester Name="lxshib02.lrz-muenchen.de"/>
 </DestinationSite>
```

Nachdem man von InQueue akzeptiert wurde:

siterefresh:
Tool zum herunterladen der neuesten trust.xml-Daten von InQueue.
Muss unbedingt ausgeführt werden, damit das Target das Origin (bzw. dessen Keys) auch akzeptiert.

```
cd /opt/shibboleth-target-1.2.1/etc/shibboleth

../../bin/./siterefresh --url http://wayf.internet2.edu/InQueue/IQ-
sites.xml --out IQ-sites.xml --cert inqueue.pem

../../bin/./siterefresh --url http://wayf.internet2.edu/InQueue/IQ-
trust.xml --out IQ-trust.xml --cert inqueue.pem
```

Um beim Testen keinen Fehler (Seite nicht gefunden) zu erhalten, muss man in dem Pfad /APACHE_DOCUMENT_ROOT/shibboleth/ eine kleine Test-html-Datei anlegen.

TEST:

Nun kann man das Target testen, in dem man mit dem Browser auf https://lxshib02.lrz-muenchen.de/shibboleth geht. Man wird über das WAYF (momentan noch von InQueue gestellt) an unser Origin auf lxshib01 weitergeleitet, an dem man sich nun anmelden kann. Danach wird der Zugriff auf lxshib02 gewährt. Momentan wird noch Jedem Zugriff gewährt, der sich erfolgreich AUTHENTIFIZIEREN kann. AUTORISIERUNG und andere Konfigurationen folgen.

6 Konfiguration von Shibboleth

6.1 Aufsetzen eines eigenständigen WAYFs

Installation:

auf lxshib01

```
cd /opt/shibboleth-origin-1.2.1
./ant package-wayf
cp dist/shibboleth-wayf.war /PATH_TO_TOMCAT/webapps
```

Konfiguratiuon:

```
cd /PATH_TO_TOMCAT/webapps/shibboleth-wayf/WEB_INF/classes/conf
cp IQ_sites.xml sites.xml
```

dort werden folgende Tags eingefügt:

```
<OriginSite Name="urn:mace:inqueue:lrz-muenchen.de">
       <Alias>LMU Origin</Alias>
         <Contact Type="technical" Name="Wolfgang Hommel"
                   Email="Wolfgang.Hommel@lrz-muenchen.de"/>
         <Contact Email="Helmut.Reiser@ifi.lmu.de" Name="Helmut Reiser"
                   Type="administrative"/>
         <HandleService Location="https://lxshib01.lrz-muenchen.de/
                   shibboleth/HS" Name="lxshib01.lrz-muenchen.de"/>
         <AttributeAuthority Location="https://lxshib01.lrz-muenchen.de/
                   shibboleth/AA" Name="lxshib01.lrz-muenchen.de"/>
         <Domain>lrz-muenchen.de</Domain>
</OriginSite>

<DestinationSite Name="https://lxshib02.lrz-muenchen.de/shibboleth">
     <Alias>LMU Service Provider</Alias>
     <Contact Email="sebastian.zunhammer@stusta.mhn.de" Name="Sebastian
          Zunhammer" Type="technical"/>
     <Contact Email="ebertm@cip.ifi.lmu.de" Name="Matthias Ebert"
          Type="administrative"/>
     <AssertionConsumerServiceURL Location="https://lxshib02.lrz-
          muenchen.de/Shibboleth.shire"/>
     <AttributeRequester Name="lxshib02.lrz-muenchen.de"/>
 </DestinationSite>
```

Folgende Änderungen in der httpd.conf:

zu den anderen Locations
`<Location /shibboleth-wayf></Location>`
hinzufügen und danach
`JkMount /shibboleth-wayf/* ajp13`
einfügen.

Zum Schluss noch im Target (shibboleth.xml) die alte wayfURL durch die neue (https://lxshib01.lrz-muenchen.de/shibboleth-wayf/WAYF) ersetzen.

6.2 SAML-Konfiguration

6.2.1 Target

- in shibboleth.xml ergeben sich folgende Einstellungen:

```
<Applications xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
   id="default" providerId="https://lxshib02.lrz-muenchen.de/
   shibboleth">
```

Der Tag `Applications` beinhaltet die einzelnen `Application-Tags`, eine eindeutige `Shire-URL`, `WAYF`

```
<Sessions lifetime="7200" timeout="3600" checkAddress="true"
        shireURL="/Shibboleth.shire" shireSSL="false"
     wayfURL="https://lxshib01.lrz-muenchen.de/shibboleth-
        wayf/WAYF"/>
```

Die Attribute folgender Einträge werden angefordert, z.B:

```
<saml:AttributeDesignator AttributeName="urn:mace:dir:attribute-
def:eduPersonAffiliation "AttributeNamespace="urn:mace:shibboleth:1.0:
attributeNamespace:uri"/>
```

Verweis auf die AAP.xml, in der Attribute und deren Werte gefiltert werden:

```
<AAPProvider type="edu.internet2.middleware.shibboleth.target.
  provider.XMLAAP" uri="/opt/shibboleth-target-
  1.2.1/etc/shibboleth/AAP.xml"/>
```

Hier ist der Verweis auf die sites.xml, in der die relying parties zu finden sind:

```
<FederationProvider type="edu.internet2.middleware.shibboleth.common.
  provider.XMLMetadata "uri="/opt/shibboleth-target-
  1.2.1/etc/shibboleth/sites.xml"/>
```

Hier stehen die Zertifikate:

```
    <TrustProvider type="edu.internet2.middleware.shibboleth.common.
      provider.XMLTrust"uri="/opt/shibboleth-target-1.2.1/etc/
      shibboleth/trust.xml"/>
```

Hier steht, welche Keys und Zertifikate benutzt werden sollen:

```
  <CredentialsProvider type="edu.internet2.middleware.shibboleth.
     common.Credentials">
        <Credentials xmlns="urn:mace:shibboleth:credentials:1.0">
            <FileResolver Id="defcreds">
                <Key format="PEM">
                    <Path>/usr/apache_1.3.33/conf/ssl.key/lxshib02.lrz-
                        muenchen.de.key</Path>
                </Key>
                <Certificate format="PEM">
                    <Path>/usr/apache_1.3.33/conf/ssl.crt/lxshib02.lrz-
                        muenchen.de.crt</Path>
                </Certificate>
            </FileResolver>
```

- in AAP.xml ergeben sich fogende Einstellungen:

In der AAP wird festgelegt, welche Seiten, mit welchen Attributen, mit welchen Attributwerten, durchgelassen werden. Beispiel für eine Attribut-Regel:

In diesem Beispiel werden die Attribute 'Informatik' und 'BWL' zugelassen.

```
<AttributeRule Name="urn:mace:dir:attribute-def:eduPersonAffiliation"
Header="Shib-EP-UnscopedAffiliation" Alias="unscoped-affiliation">
  <AnySite>
    <Value Type="regexp">^[I|i][N|n][F|f][O|o][R|r][M|m][A|a][T|t][I|i]
[K|k]$</Value>
    <Value Type="regexp">^[B|b][W|w][L|l]$</Value>
  </AnySite>
</AttributeRule>
```

- in der sites.xml ergeben sich folgende Änderungen:

Hier befinden sich alle Origins und Destinations der Federation, z.B:

```
<OriginSite ErrorURL="https://lxshib01.lrz-muenchen.de/lmu.html"
     Name="urn:mace:inqueue:lrz-muenchen.de">
        <Alias>Ludwig Maximilians Universitaet Muenchen</Alias>
        <Contact Email="sebastian.zunhammer@stusta.mhn.de"
     Name="Zunhammer" Type="technical"/>
        <Contact Email="ebertm@cip.ifi.lmu.de" Name="Ebert"
     Type="administrative"/>
        <HandleService Location="https://lxshib01.lrz-
     muenchen.de/shibboleth/HS" Name="lxshib01.lrz-muenchen.de"/>
```

```
        <AttributeAuthority Location="https://lxshib01.lrz-
     muenchen.de/shibboleth/AA" Name="lxshib01.lrz-muenchen.de"/>
        <Domain regexp="false">lrz-muenchen.de</Domain>
</OriginSite>

<DestinationSite Name="https://lxshib02.lrz-muenchen.de/shibboleth">
         <Alias>LMU Service Provider</Alias>
         <Contact Email="sebastian.zunhammer@stusta.mhn.de"
          Name="Sebastian Zunhammer" Type="technical"/>
         <Contact Email="ebertm@cip.ifi.lmu.de" Name="Matthias Ebert"
          Type="administrative"/>
         <AssertionConsumerServiceURL Location="https://lxshib02.lrz-
          muenchen.de/Shibboleth.shire"/>
         <AttributeRequester Name="lxshib02.lrz-muenchen.de"/>
</DestinationSite>
```

- apache.config

In der apache.config werden die Ordner angegeben, die von Shibboleth bewacht werden sollen. Zusätzlich werden die Attribute und Attributwerte angegeben, die zur Autorisierung benötigt werden, z.B:

```
     <Location /shibboleth>
          AuthType shibboleth
          ShibRequireSession On
           require unscoped-affiliation informatik
     </Location>
```

require:
um die Attributwerte in Form eines regulären Ausdrucks angeben zu können, gilt folgende Syntax:
`~ ^regulärerAusdruck$`
Will man also zwei verschiedene Attributwerte zulassen:
`require unscoped-affiliation ~^informatik|bwl$`

6.2.2 Origin

- arp.site.xml

Hier befinden sich die Regeln, welche Attribute zu welchen Targets freigegeben werden, z.B:

```
<Rule>
     <Target>
         <AnyTarget/>
     </Target>
     <Attribute name="urn:mace:dir:attribute def:eduPersonAffiliation">
         <AnyValue release="permit"/>
     </Attribute>
</Rule>
```

6.3Start-Scripts

Siehe Literaturverzeichnis [13].

6.3.1 Target

Um die Arbeit auf der Konsole zu erleichtern, muss das apachectl-Skript folgendermaßen erweitert werden:

In der Konfiguration-Section (unter oder über "HTTPD=...":
`SHAR=/opt/shibboleth-target-1.2.1/bin/shar`

dann weiter zu den `startssl`- und `stop`-Skripten:

`startssl` benennen wir in `start` um

`start` sieht nach unseren Änderungen so aus:

```
    start*)
        export PATH=/usr/java/j2sdk1.4.2_04/bin:$PATH
        export JAVA_HOME=/usr/java/j2sdk1.4.2_04
        export SSL_BASE=/usr/local/ssl
        export SHIB_HOME=/opt/shibboleth-origin-1.2.1
        if [ $RUNNING -eq 1 ]; then
            echo "$0 $ARG: httpd (pid $PID) already running"
            continue
        fi
        if $SHAR -f & then
            echo "$0 $ARG: shar started"
        else
            echo "$0 $ARG: shar could not be started"
            ERROR=3
        fi
        if $HTTPD -DSSL; then
            echo "$0 $ARG: httpd started"
        else
            echo "$0 $ARG: httpd could not be started"
            ERROR=3
        fi
        ;;
```

6.3.2 Origin

Auch hier wird die apachectl modifiziert:

In der Konfigurations-Section:
```
TOMCAT_START=/usr/jakarta-tomcat-4.1.31/bin/startup.sh
TOMCAT_STOP=/usr/jakarta-tomcat-4.1.31/bin/shutdown.sh
```

Weiter zu den startssl- und stop-Skripten:

`startssl` wird in `start` umbenannt.

`start` sieht nach den Änderungen so aus:

```
case $ARG in
    start*)
        export PATH=/usr/java/j2sdk1.4.2_04/bin:$PATH
        export JAVA_HOME=/usr/java/j2sdk1.4.2_04
        export SSL_BASE=/usr/local/ssl
        export SHIB_HOME=/opt/shibboleth-origin-1.2.1
        if [ $RUNNING -eq 1 ]; then
            echo "$0 $ARG: httpd (pid $PID) already running"
            continue
        fi
        if $TOMCAT_START; then
            echo "$0 $ARG: tomcat started"
            sleep 5
        else
            echo "$0 $ARG: tomcat could not be started"
            ERROR=3
    fi
        if $HTTPD -DSSL; then
            echo "$0 $ARG: httpd started"
        else
            echo "$0 $ARG: httpd could not be started"
            ERROR=3
        fi
        ;;
```

`stop` sieht nach der Modifikation so aus:

```
if [ $RUNNING -eq 0 ]; then
            echo "$0 $ARG: $STATUS"
            continue
        fi
        $TOMCAT_STOP
        if kill $PID ; then
            echo "$0 $ARG: httpd stopped"
        else
            echo "$0 $ARG: httpd could not be stopped"
            ERROR=4
        fi
        ;;
```

6.4 PubCookie und neue Zertifikate

Im Laufe der Installationsarbeiten traten immer wieder Probleme mit PubCookie und neuen Zertifikaten auf. Deshalb gibt es hier nochmal eine Zusammenfassung aller wichtigen Schritte, um neue Zertifikate zu installieren:

1) Löschen aller vorhandenen Zertifikate!!!!
   z.B. im PubCookie_PATH (bei uns "/usr"), MOD_PubCookie_PATH(bei uns "/usr/PubCookie-ldap"), APACHE_PATH/conf/ssl.XXX
2) Erstellen eines neuen Keys mit folgendem Befehl:
   `openssl genrsa -des3 1024 > /KEY_PATH/DNS_NAME.key`
3) Entfernen der Passphrase aus dem key mit folgendem Befehl:
   `openssl rsa -in /KEY_PATH/DNS_NAME.key -out /KEY_PATH/DNS_NAME.key`
4) Erstellen eines neuen csr-Files ('Certificate Signing Request') mit folgendem Befehl:
   `openssl req -new -key /KEY_PATH/DNS_NAME.key > /KEY_PATH/DNS_NAME.csr`
5) Erstellen eines neuen Zertifikates mit folgendem Befehl:
   `openssl req -x509 -key /KEY_PATH/DNS_NAME.key -in KEY_PATH/DNS_NAME.csr > /KEY_PATH/DNS_NAME.crt`
6) Kopieren des neuen Keys und Zertifikates an die wichtigen Stellen, z.B:
   PubCookie_KEY_PATH, APACHE_PATH/conf/ssl.XXX
7) Falls für SSL und PubCookie die selben Zertifikate benutzt werden, diese nochmal extra unter einem anderen Dateinamen abspeichern, z.B. als:
   /PubCookie_PATH/PubCookie_granting.key, /PubCookie_PATH/PubCookie_granting.crt
8) Falls die neuen Zertifikate andere Namen haben als die Alten, müssen die config-files jetzt noch entsprechend geändert werden.
9) Kopieren der starter keys in den KEY_PATH:
   ```
   cd PubCookie_PATH
   cp starter.key keys/DNS_NAME
   ```
10) Apache, Tomcat und xinetd neu starten
11) Im PubCookie_PATH den keyclient ausführen:
   ```
   cd /PubCookie_PATH
   ./keyclient
   ```
12) Im MOD_PubCookie_PATH den keyclient ausführen:
   ```
   cd /MOD_PubCookie_PATH
   ./keyclient
   ```
13) Im MOD_PubCookie_PATH das Granting-Zertifikat abholen:
   ```
   cd /MOD_PubCookie_PATH
./keyclient -G /MOD_PubCookie_PATH/keys/DNS_NAME.crt
   ```

6.5 trust.xml

Damit ein Target, im Bezug auf Zertifikate, einem (neu hinzugekommenen) Origin vertraut, kann man entweder das "Apache-CRT" des Origins in die trust.xml des Targets hinzufügen, oder man holt sich vom Link https://bossie.doit.wisc.edu:3443/cert/i2server/csr mit Hilfe eines selbst erzeugtem csr ein crt, welches man dann als "Apache-CRT" hernimmt. Diese werden von der trust.xml allgemein akzeptiert.

7 Zusammenfassung und Ausblick

Durch das in dieser Arbeit beschriebene Vorgehen konnte das vorgesehene Testszenario erfolgreich umgesetzt werden. Trotz Anfangsschwierigkeiten, z.B. wegen Apache und SSL, gestaltete sich der weitere Verlauf des Fortgeschrittenenpraktikums sehr lehrreich. Ausführliche Tests mit SHIBBOLETH verliefen sehr positiv, das System läuft stabil und sicher. Mit Shibboleth ist es möglich feingranulare Einstellungen, z.B. im Bezug auf die ARPs, vorzunehmen, um so gegebenen Anforderungen gerecht zu werden. Trotz des immensen Installationsaufwands ist das jetzt aufgebaute System leicht erweiterbar. So können weitere Relying Parties ohne größere Komplikationen in die bereits bestehende Föderation eingebunden werden.

Allerdings könnte sich die Realisierung einer derart großen Föderation wie LMU – TU – LRZ – FH als relativ schwierig herausstellen, da die einzelnen Organisationen ihre eigenen Vorstellungen haben, wie eine solche Zusammenarbeit ablaufen könnte. Der hier notwendige Abstimmungsaufwand muss durch hohe Kompromissbereitschaft aller Beteiligten minimiert werden.

Ausblick:
Die Virtuelle Hochschule Bayern erwägt den Einsatz von Shibboleth.
Ausserdem schaffen bereits laufende Projekte an der TUM und am LRZ eine gute Grundlage für Shibboleth. Shibboleth soll also bald ‚wirklich' verwendet werden.
Offen bleiben hier natürlich neben den technischen noch viele organisatorische Fragen, z.B.

- wer hostet die notwendigen Server
- wer verwaltet die Attributfreigaben
- welche Organisationen werden in die Föderation aufgenommen
- wer trägt die Gesamtverantwortung
- was passiert bei Serverausfällen?

8 Literaturverzeichnis

[1] The Apache Software Foundation: Apache HTTP Server Version 1.3.
URL: http://httpd.apache.org/docs/

[2] The Apache Software Foundation: Jakarta Tomcat
URL: http://nagoya.apache.org/wiki/apachewiki.cgi?Tomcat/Links

[3] O'Reilly Media Inc. : Configuring Tomcat and Apache With JK 1.2
URL: http://www.onjava.com/pub/a/onjava/2002/11/20/tomcat.html

[4] SUN Microsystems, Inc.: Java
URL: http://java.sun.com/

[5] University of Washington: PubCookie
URL: http://www.PubCookie.org/docs

[6] Luiz Ernesto Pinheiro Malere: LDAP
URL:http://www.tldp.org/HOWTO/LDAP-HOWTO/

[7] EDUCAUSE: LDAP
URL:http://www.educause.edu/content.asp?PAGE_ID=949&bhcp=1

[8] Internet2: Shibboleth Origin
URL: http://shibboleth.internet2.edu/guides/deploy-guide-origin1.2.1.html

[9] Internet2: Shibboleth Target
URL: http://shibboleth.internet2.edu/guides/deploy-guide-target1.2.1.html

[10] Internet2: Shibboleth Checklist
URL: http://shibboleth.internet2.edu/guides/identity-provider-checklist.html

[11] Internet2: InQueue
URL: http://inqueue.internet2.edu/

[12] Internet2: SAML
URL: http://www.opensaml.org/

[13] Vivek G. Gite: Linux Shell Scripting Tutorial
URL: http://www.freeos.com/guides/lsst/

[14] SourceForge: Log library for C++
URL: http://sourceforge.net/projects/log4cpp/