Projekt: GAP-SLC

Evaluierung der Shibboleth-Integration der Portalsoftware Liferay

erstellt für

BMBF-Projekt
„Nutzung von kurzlebigen Zertifikaten in portalbasierten Grids“ (GapSLC) – Förderkennzeichen 01IG09003 –

von

DAASI International GmbH

Autoren: Martin Haase, Peter Gietz
Version: 0.05 (Evaluierung_Liferay_v0.05b)
Erstellt am: 18.08.2010
Letzte Änderung: 21.10.10

Kontakt:
DAASI International GmbH
Europaplatz 3
D-72072 Tübingen
Tel.: 07071 4071090
Fax: 07071 4071099
E-Mail: info@daasi.de
Internet: http://www.daasi.de

Versionskontrolle:

Version	Datum	Autor	durchgeführte Änderungen
0.01	05.08.10	Martin Haase	Erste vollständige Version Liferay 5
0.02	27.08.10	Peter Gietz	Endredaktion der ersten Version
0.03	15.09.2010	Martin Haase	Ergänzungen Liferay 6, kleinere Korrekturen und weitere Ergänzungen
0.04	05.10.2010	Martin Haase	Ergänzung Casshib
0.05	20.10.2010	Martin Haase, Peter Gietz	Umstrukturierung (Einordnung der Nachträge) und Endredaktion

Finale Abnahme:

Version	Datum	abgenommen durch	Rolle / Position
1.0			

Bemerkung zu dieser Version:
Diese Version ist eine um Liferay 6 erweiterte Version zur Freigabe an D-Grid-Mitglieder

Inhaltsverzeichnis

1 Einleitung

Im Rahmen des Projekts GAP-SLC wurde wie im Antrag spezifiziert, die Portalsoftware Gridsphere und ihre Shibbolethisierung verwendet und evaluiert. Da sich während des Projektverlaufs aber herausstellte, dass im Rahmen von D-Grid die Portalsoftware Liferay eine zunehmende Bedeutung gewann, wurde die Shibbolethisierung von Liferay ebenfalls evaluiert. Dieses Dokument gibt einen Bericht über diese Evaluation.

Es ist zu beachten, dass dieser Text keine allgemeine Evaluierung des Liferay-Portals darstellt, sondern nur die im Kontext einer Shibboleth-Integration bzw. Single-Sign-On relevanten Fragen behandelt. Etwaige darüber hinausgehende Bemerkungen zu Liferay sind nicht systematischer Natur.

Der grundsätzliche Aufbau des Testszenarios ist der folgende: Das Liferay-Portal wird von einem Shibboleth Service Provider (SP) geschützt. Dieser veranlasst beim entsprechenden Portal-Zugriff einen Login-Vorgang am Identity Provider. Benutzerattribute werden nach erfolgreichem Login vom Identity Provider über den Service Provider an das Portal weitergegeben.

Im dem hier beschriebenen Tests wurde nicht der Shibboleth Identity Provider (IdP) verwendet, sondern die Software SimpleSamlPHP in der Funktion als SAML 2.0 Identity Provider. Dies war möglich, da beide Softwarelösungen gleichermaßen SAML2.0-standardkonform und mit dem eingesetzten Shibboleth Service Provider interoperabel sind.

2 Liferay-Testinstallation

2.1 Installation

Liferay (zunächst Version 5.2.3 und im Nachgang Version 6.0.5) wurde im DAASI-Labor auf einer virtuellen Maschine unter OpenSuSE 11.1 installiert. Die Vorgehensweise lehnte sich in Grundzügen an der in [Liferay-Admin], Kapitel 2 dargestellten Weise an. Es wurde die Version verwendet, die im Bündel mit Tomcat geliefert wird. Das Archiv ist auszupacken (das entstandene Wurzelverzeichnis wird im Folgenden mit ~liferay bezeichnet) und der darin befindliche Tomcat in ~liferay/tomcat-5.5.27/bin mit startup.sh gestartet.

Out-of-the-Box ist das Portal nach dem Starten des Tomcat über http://host.domain:8080 ansprechbar. Ein Account mit den Logindaten „test@liferay.com“ / „test“ ist bereits angelegt und gehört zu einem Benutzer „Joe Bloggs“.

Accounts werden per Default an die E-Mail-Adresse eines Benutzers gebunden. Dies wird u.A. dafür benötigt, um nach der Beantragung eines Accounts eine Bestätigungsmail mit dem Passwort zu verschicken. Trotzdem wird zum Benutzer in der Datenbank noch eine interne nummerische ID generiert.

Viele Eigenschaften des Portals sind in einer einzigen großen Initialisierungsdatei abgelegt: ~liferay/portal-ext.properties. Die dort aufgeführten Parameter überschreiben die jeweiligen.

2.2 Funktionsweise

Im Folgenden werden einige kurze Bemerkungen zur getesteten Liferay-Version gemacht:

- Liferay unterstützt OpenID Out-of-the-Box
- Das Portal ist in ca. 20 Sprachen lokalisiert, allerdings mit (zumindest im Deutschen) kleineren Fehlern und Auslassungen
- Das Portal kann auch als Server für Web Services dienen (ServiceBuilder für eine SOA).
  - Dazu müssen die IP-Adressen der erlaubten Web Service Clients in portal-ext.properties aufgeführt werden, z.B.: axis.servlet.hosts.allowed=192.168.123.456
  - Die WSDL-Informationen zu den Services sind dann unter

    http://BenutzerID:Passwort@host.domain:8080/tunnel-web/axis

    zugänglich, wobei BenutzerID nicht die E-Mail-Adresse, sondern die interne nummerische ID ist.
  - Der Web Service Container ist der mittlerweile veraltete Axis1.4
  - Mehr Details sind im Liferay Developer's Guide [Liferay-Devel] zu finden.
- Liferay verwendet JAAS (Java Authentication and Authorization Service) zur Authentifizierung, wie Shibboleth übrigens auch.

3 Shibbolethisierung von Liferay

Im Folgenden werden drei Varianten evaluiert:

1. Liferay Version 5.2.3 mit ShibbolethAutologin aus dem NeISS-Projekt nach der Beschreibung in [Watt]
2. Liferay Version 6.0.5 mit ShibbolethAutologin aus dem NeISS-Projekt nach der Beschreibung in [Watt]
3. Liferay Version 6.0.5 mit Shibbolethisierung über CasShib

3.1 ShibbolethAutologin, Liferay 5.2.3

Es wird nach der Beschreibung in [Watt] vorgegangen, der für das JISC-Projekt NeISS (National e-Infrastructure for Social Simulation)[1] ein Content Configuration Portlet (CCP) beschreibt. Hinter dem CCP verbirgt sich kein Portlet im eigentlichen Sinne, sondern eine Java-Klasse, die zur Shibbolethisierung von Liferay dessen AutoLogin-Interface implementiert.

Da das NeISS-Dokument [Watt] nur auf Anfrage verfügbar ist, werden die dort beschriebenen wesentlichen Schritte im Folgenden aufgeführt.

Die NeISS-Software geht davon aus, dass in der Föderation, an der der Service Provider mit dem Portal teilnimmt, das Attribut eduPersonTargetedId IdP-seitig freigegeben wird und zur

1 Vgl. http://www.geog.leeds.ac.uk/projects/neiss/index.php

Personalisierung verwendet werden kann. Der verwendete IdP wurde entsprechend konfiguriert.

3.1.1 Konfigurationsänderungen

Zunächst muss das Portal von einem Shibboleth Service Provider beschützt werden. Da dieser auf Linux nur als Apache-Modul verfügbar ist, muss für Java-Anwendungen wie Liferay vor den Servlet Container, in dem sie laufen (also Tomcat) der Apache Web Server über das AJP-Protokoll als Proxy vorgeschaltet werden. Dieser kann die von der Shibboleth-Infrastruktur empfangenen Attribute als Header-Variablen an Tomcat weitergeben.

Dazu müssen folgende in [Watt] beschriebene Schritte als Grundvoraussetzung für das Funktionieren des Systems durchgeführt werden:

- mod_proxy und mod_proxy_ajp müssen für den Apache Web Server aktiviert werden
- im Apache-Vhost müssen alle Anfragen an Tomcat weitergeleitet werden:

  ProxyPass / ajp://localhost:8009/ retry=5
- In der server.xml von Tomcat:
  - Aktivierung des AJP 1.3-Tomcat-Connectors auf Port 8009
  - Optional: Deaktivierung des Tomcat-Connectors auf Port 8080
- Erweiterung von ~liferay/portal-ext.properties:

  web.server.http.port=443

  web.server.https.port=443

  web.server.protocol=https

  web.server.host=host.domain
- Nun müssen der Anwendung, also Liferay, Shibboleth-Attribute zur Verfügung gestellt werden. Diese Einstellungen kommen auch in den Vhost:
  - Generelle Verfügbarkeit der Attribute:

    <Location />

    AuthType shibboleth

    require shibboleth

    ShibUseHeaders On

    </Location>
  - Zwingende Shibboleth-Authentifizierung beim Login-Link:

    <Location /c/portal/login>

    AuthType shibboleth

    ShibUseHeaders On

```
require valid-user
</Location>
```

3.1.2 Aktivierung der zusätzlichen Software

Anschließend muss nach [Watt] wie folgt vorgegangen werden, wobei wir eine IDE wie z.B. Eclipse zur Unterstützung der Schritte empfehlen:

- Aus-Checken der Liferay-Quellen (über Subversion von svn://svn.liferay.com/repos/public mit Login „guest“ und leerem Passwort)
- Importieren der Klasse ShibbolethAutoLogin.java (s. Anhang) nach /portal/portal-impl/src/com/liferay/portal/security/auth/, und ggf. anpassen nach Bedarf
- Kompilieren der Klasse mit Java 1.5
- Einfügen der kompilierten ShibbolethAutoLogin.class in ~liferay/tomcat-5.5.27/webapps/ROOT/WEB-INF/lib/portal-impl.jar, dort im Pfad /com/liferay/portal/security/auth/ (dies ist komfortabel mit Tools wie dem Midnight-Commander *mc* möglich)
- Anfügung an ~liferay/portal-ext.properties:

```
auth.pipeline.pre=

auth.pipeline.post=

auth.pipeline.enable.liferay.check=false

auto.login.hooks=com.liferay.portal.security.auth.ShibbolethAutoLogin
```

- Anschließend muss Tomcat neu gestartet werden

Auf der Portal-Eingangsseite kann unter „Welcome“ nun „anmelden“ ausgewählt werden, worauf die Shibboleth-übliche Weiterleitung an den IdP Discovery Service bzw. einen spezifischen IdP zur Eingabe der Benutzerkennung erfolgt.

Hat sich ein bestimmter Benutzer noch nie (weder über Shibboleth als auch über die lokale Benutzerverwaltung) im Portal angemeldet, sieht er zunächst die Nutzungsbedingungen (Terms of Use). Je nach Inhalt der tatsächlich implementierten Klasse ShibbolethAutologin werden nun bestimmte Shibboleth-Attribute vom IdP im Portal übernommen. So erhält in der unmodifizierten Klasse aus dem NeISS-Projekt der neue Benutzer z.B. automatisch die E-Mail-Adresse „change@me.com“.

3.1.3 Evaluierung der Shibboleth-Integration

Im Gegensatz zu anderen Single-Sign-On-Methoden wie CAS, NTLM, SiteMinder, OpenID und OpenSSO, die direkt im Liferay Admin Guide beschrieben und einfach als Modul zuschalt- und konfigurierbar sind, scheint eine solche Plugin-artige Lösung für Shibboleth noch nicht zu existieren. Alle im Netz zugänglichen Beispiele sind Variationen des CCP, d.h. sie erfordern ein Aus-Checken der Liferay-Quellen, ggf. manuelles Anpassen von Java-Code an die Bedürfnisse des Portals bzw. an die Gegebenheiten der vorhandenen Shibboleth-Föderation und Einpassen der kompilierten Klasse in die betreffenden Jar-Dateien.

Weitere Kritikpunkte, die sich aber eher auf die konkrete Integration der NeISS-Autologin-Klasse beziehen, sind:

- Merkwürdig ist, dass nach dem Login als Shibboleth-Benutzer die Portal-URL immer noch auf .../web/guest lautet. Erst beim Auswählen von „Meine Orte“ → „Meine Community„ → „Private Seiten“ kann man Einstellungen an Portlets vornehmen (also z.B. einen Eintrag im Kalender machen)
- Beim Klicken auf „Abmelden“ ist der Benutzer weiterhin angemeldet und hat sogar noch mehr Konfigurationsoptionen, kann Portlets konfigurieren oder verschieben. Die URL ist .../user/change/home
- Ein u.E. schwerwiegender Fehler ist das folgende Verhalten: Wenn ein Benutzer sich neu anmeldet und vergisst, seine Dummy-E-Mail-Adresse von „change@me.com“ in etwas Eindeutiges zu ändern, können sich alle nachfolgenden über Shibboleth authentifizierten Benutzer nicht mehr am Portal anmelden, weil das CCP ihnen wieder diese Dummy-Adresse zuweist. Stattdessen sehen diese Benutzer ein Formular zum Ändern von Mail und Passwort. Dieses Formular akzeptiert aber keine Eingaben, sondern gibt die Fehlermeldung: „You have entered invalid data. Please try again. Please enter a valid login“ zurück.
- Ein weiterer Fehler: Im „Account“-Bereich kann auch ein Shibboleth-Benutzer ein Passwort vergeben. Anschließend kann der Benutzer sich mit diesem Passwort auch direkt einloggen, was einem über Shibboleth authentifizierten Benutzer ja nicht erlaubt sein sollte.
- Unschön: Nach dem Einloggen, wenn sich der Benutzer z.B. beim Ändern der Account-Details befindet, trägt der Link, der zurück auf die Hauptseite führt, die Bezeichnung: „Zurück zu Guest“ anstatt den Namen des eingeloggten Benutzers.

3.2 Shibboleth Autologin, Liferay 6.0.5

Dieser Abschnitt behandelt Version 6.0.5 CE Bunyan / Build 6005, Aug 16 2010, die in gleicher Weise wie in Abschnitt 3.1 getestet wurde. Der Shibboleth-Teil basiert weiterhin auf dem CCP aus dem NeISS-Projekt von John Watt. Das Vorgehen ist entsprechend zu Abschnitt 3.1, im Folgenden werden nur Liferay6-relevante Änderungen erwähnt.

3.2.1 Notwendige Änderungen am Code

Die API der importierten Klassen hat sich zwischen Version 5 und 6 geändert. Dies betrifft die Methoden

- UserLocalServiceUtil.getUserByOpenId(String): hier wurde vor dem einzigen Argument ein zusätzliches Argument vom Typ long eingeführt
- UserLocalServiceUtil.addUser(long,long,boolean,String,String,boolean,String,String,String ,...) es wurde vor dem 9. Argument ebenso ein zusätzliches Argument vom Typ long eingeführt

Um die Klasse also minimal kompilieren zu können, musste dieses long-Argument gesetzt werden. Es wurde in beiden Fällen mit dem Wert 0 belegt.

3.2.2 Evaluierung

Nach Kompilierung der Klasse und entsprechender Integration ins Portal analog zu Kapitel 3 wurden folgendes beobachtet:

1. Das Einloggen von Shibboleth-Benutzern ist nur möglich, wenn kein Benutzer mit einer Email-Adresse "change@me.com" bereits vorhanden ist. Dies ist ähnlich dem in 3.3 gefundenen Verhalten.
2. Bei *jedem* Login-Vorgang wird ein neuer Benutzer angelegt (Der Login-Name ist change.XX, wobei XX eine hochgezählte Nummer ist). Dies gilt insbesondere auch, wenn
   a) sich derselbe Benutzer (neue Browser/IdP-Session) ein zweites Mal einloggt
   b) sich unterdessen ein lokaler User einloggt und wieder abmeldet (somit der aktuelle Shibboleth-User automatisch wieder angemeldet werden sollte, da ja eine SP/IdP-Session besteht)

   Dies bedeutet, dass vorhandene Shibboleth-Benutzer sich nicht mehrmals anmelden können, da das Portal immer ein neues Profil erstellt und benutzerspezifische Einstellungen nicht in dieses übernommen werden.

3.3 Shibbolethisierung über Casshib

Während der Evaluierungsarbeiten zur Shibbolethisierung von Liferay 6 wurde das Projekt Casshib publik[2], vgl. http://code.google.com/p/casshib/. Die Idee ist dabei, das bereits vorhandene CAS-Modul von Liferay nachzunutzen. Casshib agiert dann für die CAS-fähige Anwendung (Liferay) als CAS-Server, ist aber selbst shibbolethisiert, so dass es sich gegenüber dem Service Provider als eine Anwendung verhält, die in einem Servlet Container läuft. Etwaige Benutzerattribute vom IdP, die der SP bekommt, können unter bestimmten Voraussetzungen über Casshib an die Anwendung weitergegeben werden.

3.3.1 Vorgehen

Die Projektseite von Casshib http://code.google.com/p/casshib/ enthält eine recht gute und ausführliche Anleitung, wie eine Anwendung mit Casshib zusammenarbeiten kann. Es werden neben der generellen Funktionsweise von Casshib die spezifischen und hier relevanten Konfigurationsoptionen für Shibboleth (IdP und SP), Apache, Tomcat, Casshib selbst, und der etwaigen Anwendung erläutert.

Im Wesentlichen müssen die folgenden Vorkehrungen getroffen werden:

- Pro Anwendung wird eine bestimmte URL vom SP geschützt, z.B. casshib/shib/demoapp
- Pro Anwendung wird in den Metadaten der Föderation eine bestimmte EntityID registriert
- In Apache werden bestimmte URLs an Tomcat weitergeleitet (/casshib/*), wobei die Shibboleth-Handler-URL (/casshib/shib/*/Shibboleth.sso/*) davon ausgenommen werden muss.
- Benutzerattribute müssen auf SP-Seite (REMOTE_USER in shibboleth2.xml, attribute-map.xml, ggf. attribute-filter.xml) das fest kodierte Präfix "shibattr-" tragen, damit Casshib

2 Vgl. http://groups.google.com/group/shibboleth-users/msg/afac8620da5bb19d

diese erkennen kann.

- Casshib selbst muss konfiguriert werden: Jeder Service bekommt eine ID, einen Namen und einen Passcode. Dabei ist zu beachten:
  - ID: dies ist die Login-Seite der Applikation. Bei Liferay wäre hiermit die URL https://host.org/c/portal/login/ zu setzen, wobei auf den finalen Schrägstrich geachtet werden muss.
  - Name: das Kürzel für die Applikation, damit sie von Casshib referenziert werden kann; es empfiehlt sich, dasselbe Kürzel wie in shibboleth2.xml zu verwenden, also "demoapp", oder "liferay".
  - Passcode: ein Passwort (shared secret), das für jede Anwendung zur Identifizierung ausgegeben wird.
- Nun muss nur noch die Anwendung selbst konfiguriert werden. Für eine Liferay-Installation sollten also die folgenden Werte in die Maske unter Settings → Authentication → CAS eingetragen werden:
  - Login URL: https://host.org/casshib/shib/liferay/login
  - Logout URL: https://host.org/casshib/shib/liferay/logout
  - Server Name: https://host.org
  - Server URL: https://host.org/casshib/shib/12345
  - Service URL: https://host.org/c/portal/login/

Dies ist in groben Zügen der Aufbau des Systems. Auf der Casshib-Seite wird auch eine CAS-fähige Demo-Anwendung bereitgestellt, mit der überprüft werden kann, ob die Konfiguration des Gesamtsystems funktioniert.

Während des Aufsetzens der einzelnen Komponenten wurden einige Beobachtungen gemacht, die für eine Reproduzierung möglicherweise nützlich sind:

- Es hat sich als sinnvoll erwiesen, den Casshib-Server und (ggf. die Demo-Applikation) nicht in dem von Liferay mitgebrachten Tomcat zu installieren: Liferay geht in der Grundeinstellung davon aus, dass der gesamte URL-Raum unterhalb der Server-Root ihm zur Verfügung steht.
- Die Weiterleitung von Apache zu Tomcat ist am einfachsten über mod_jk zu erreichen, wobei mit etwas mehr Aufwand auch das nicht von Casshib dokumentierte mod_proxy_ajp verwendet werden kann.
- Casshib benötigt eine Dependency (cas-server-support-trusted-3.4.2.jar), die in der ausgelieferten Konfiguration nicht enthalten ist. Deshalb muss vor dem Bauen der WAR-Datei mit Maven das folgende XML-Fragment in pom.xml eingefügt werden:

```
<!-- added support for PrincipalBearingCredentialsAuthenticationHandler -->
                <dependency>
                        <groupId>org.jasig.cas</groupId>
                        <artifactId>cas-server-support-trusted</artifactId>
                        <version>${cas.version}</version>
                </dependency>
```

- Die Zertifikate müssen folgende Eigenschaften erfüllen:

- Zum einen benötigt der Tomcat im Keystore die CA-Informationen zum vom Apache eingesetzten Zertifikat
- Zum Anderen benötigt (zumindest im Testlabor) das Zertifikat einen DNS-Eintrag im SubjectAlternativename, der auf den Host lautet

3.3.2 Schwachpunkte

Nachfolgend werden einige Kritikpunkte an der verfolgten Lösung aufgezeigt.

3.3.2.1 Attributverarbeitung Client-seitig

Zunächst muss festgestellt werden, dass die Lösung hauptsächlich für die Authentifizierung über Casshib gedacht ist, der Attributtransport ist jedoch nicht vorgesehen. In der mitgelieferten Demonstrationsanwendung wird beispielsweise für die Extraktion und Anzeige der Benutzerattribute zusätzlich der folgende JSP-Code benötigt:

```
org.jasig.cas.client.validation.Assertion assertion =
    (org.jasig.cas.client.validation.Assertion)
        session.getAttribute("_const_cas_assertion_");
  if(assertion != null)
  {
    java.util.Iterator it =
        assertion.getPrincipal().getAttributes().keySet().iterator();
    while(it.hasNext())
    {
      String name = (String)it.next();
      Object value = assertion.getPrincipal().getAttributes().get(name);
      out.println(name + " = " + value.toString());
    }
  }
```

Wie man sieht, ist zum einen die Fähigkeit zur Attributextraktion vom verwendeten CAS-Client abhängig (der Client muss "attribute-aware" sein, vgl. [CASShib], Abschnitt #Using_a_CAS_client) und läuft auf eine Analyse der SAML-Assertion als Java-Objekt hinaus. Als Ausnahme wird lediglich REMOTE_USER standardgemäß zur Verfügung gestellt über `request.getRemoteUser()`.

3.3.2.2 Prinzipielle Verwendung von Attributen

Liferay geht von einem CAS-Server aus, der seine Account-Information zumindest mittelbar aus einem bestimmten LDAP-Server bezieht. Neue Benutzer können bei einer CAS-Authentifzierung nur angelegt werden, wenn dieser LDAP-Server bekannt und konfiguriert ist. Dies kann in einer Föderation aber nicht vorausgesetzt werden. Die Attribute, die Casshib in dem ihm eigenen Format anbietet, werden nicht hierfür verwendet, sodass der Benutzer nach einem Login am IdP wieder die allgemeine Eingangsseite zu sehen bekommt, https://host.org/web/guest/home.

3.3.2.3 Sicherheitslücke

Auf der Projektseite von Casshib ist eine Warnung abgedruckt, dass durch die Verwendung eines Passcodes, so wie oben beschrieben, das System kompromittiert werden kann, so dass das CAS-Ticket und Benutzerattribute an fremde Anwendungen gelangen können (vergleiche hierzu [CASShib], Abschnitt #Service_registration_and_security).

4 Zusammenfassung

Grundsätzlich kann eine Integration von Liferay und Shibboleth erreicht werden. Für diese Evaluation sind die beiden untersuchten Ansätze zu unterscheiden: die Version der direkten Shibbolethisierung mit ShibbolethAutologin und die Shibbolethisierung über CasShib. Ein Fazit zu beiden Ansätzen wird nachfolgend gegeben.

4.1 Ansätze mit ShibbolethAutologin

Aktuell ist eine einfache und konfigurierbare Shibboleth-Einbindung als Liferay-*Modul* noch nicht verfügbar. Für jede konkrete Liferay-Installation und ggf. abhängig von den Attributen, die der Shibboleth Service Provider in der jeweiligen Föderation bekommt, muss eine spezifische Anpassung am Quellcode einer ShibbolethAutologin-Klasse gemacht werden, diese kompiliert und in die Portal-Installation manuell eingefügt werden. Insbesondere ist bei solchen Anpassungen relevant, ob in der Shibboleth-Föderation nur ein Pseudonym oder noch weitere Attribute zur Personalisierung bereit stehen.

Im Gegensatz zu Liferay 5 kann eine annähernde Shibbolethisierung mit der NeISS-Lösung bei Liferay 6 nicht mehr erreicht werden. Das inakzeptable Verhalten des Anlegens von immer neuen Benutzern muss möglicherweise mit einer tiefgreifenderen Code-Änderung abgestellt werden, als es im Rahmen dieser Evaluation möglich war.

Im Netz sind zwei weitere Implementierungen der ShibbolethAutologin-Klasse zu finden:

- Das „Original“ von James Hong, USC, für Liferay 4.2:

  http://www.liferay.com/community/forums/-/message_boards/message/114134

- Eine Adaption des Originals von James Schappet, U Iowa, für Liferay 5.2.3:

  https://www.icts.uiowa.edu/confluence/display/Sandbox/Shibboleth+Liferay+Authentication

Beide Ansätze unterscheiden sich nur in Detailfragen vom oben beschriebenen CCP. Obwohl dieser Evaluierung der Code aus dem NeISS-Projekt / CCP zugrunde lag, sind der Vollständigkeit halber auch die beiden anderen erwähnten Klassen im Anhang abgedruckt.

4.2 Casshib-Ansatz

Selbst wenn Casshib die Benutzerattribute vom IdP standardgemäß zur Verfügung stellen würde, müsste die Anwendung, also Liferay, überhaupt befähigt werden, diese zu verwenden. Und hier liegt der Hauptkritikpunkt: Bei Liferay findet nur die Authentifizierung über CAS statt. Falls Attribute nötig würden – insbesondere bei der Erstellung von neuen Benutzeraccounts –, können diese nur aus einem konfigurierten LDAP-Server bezogen werden. Die Kritik am vorgelegten Konzept liegt darin, dass in einer Föderationsumgebung dieser LDAP-Server natürlich nicht bekannt und dem Portal verfügbar sein wird. Liferay geht also von einem CAS-Server aus, der seine Account-Information aus einem bestimmten LDAP-Server bezieht. Dies ist bei Casshib nicht der Fall. Das zwischen Shibboleth SP und CAS-fähigem Portal fehlende Puzzlestück CasShib ergibt somit nur bei flüchtiger Betrachtung eine fertige Komplettlösung.

Dessenungeachtet steht es einem Portalbetreiber natürlich frei, eigenen Programmcode für Liferay zu schreiben, der die Attribute in dem von Casshib gelieferten Format ausliest (der mitgelieferte CAS-Client ist prinzipiell dazu in der Lage, also "attribute-aware") und daraufhin neue

Benutzer erstellt[3]. Damit bleibt dem Anwender aber wieder nur der in den vorherigen Kapiteln beschriebene Weg des Code-Schreibens, anstatt fertige Module konfigurieren zu können. Zudem bleibt noch die benannte Sicherheitslücke.

4.3 Ausblick

Eine Aufnahme eines Shibboleth-Moduls – das dann allein durch Optionen und nicht durch Code-Anpassung konfiguriert werden kann – zu den vorhandenen SSO-Lösungen ist auch mit Liferay 6.0.x noch nicht erreicht worden. Bei den Liferay-Entwicklern ist das Problem "Shibboleth" aber mittlerweile erkannt worden. Es bleibt abzuwarten, was hier in Zukunft geschieht.

3 Vgl. die folgende angedeutete Lösung, die hier nicht evaluiert werden konnte und deren Status bezüglich Integration in Liferay unklar ist: http://www.liferay.com/community/forums/-/message_boards/message/1758457#_19_message_1944966

5 Literaturangaben

[CASShib] CASShibExplained, Overview of CASShib: What, why, and how, Updated Feb 04, 2010 by bkoehm, http://code.google.com/p/casshib/wiki/CASShibExplained

[Liferay-Admin] Liferay Portal Administrator's Guide. Richard L. Sezov, Jr., (2009) Online: http://docs.liferay.com/portal/5.2/official/liferay-administration-guide.pdf

[Liferay-Devel] Liferay Development Documentation. Samuel Kong, Editor, (2008). Online: http://docs.liferay.com/portal/5.0/official/liferay-development-documentation-5.0.pdf

[Watt] SPAM-GP: CCP Administrator Guide for the NeISS Project . John Watt. Technischer Report, National e-Science Centre, University of Glasgow. (ohne Jahr)

6 Anhänge

6.1 Java-Code von NeISS (ShibbolethAutoLogin.java)

```
/*
 *   Liferay Shibboleth Auto Login and Role Sync (SPAM-GP CCP V2)
 *   Copyright (C) 2007 John Watt (j.watt@nesc.gla.ac.uk)
 *
 *   This program is free software: you can redistribute it and/or modify
 *   it under the terms of the GNU General Public License as published by
 *   the Free Software Foundation, either version 3 of the License, or
 *   (at your option) any later version.
 *
 *   This program is distributed in the hope that it will be useful,
 *   but WITHOUT ANY WARRANTY; without even the implied warranty of
 *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 *   GNU General Public License for more details.
 *
 *   You should have received a copy of the GNU General Public License
 *   along with this program.  If not, see <http://www.gnu.org/licenses/>
 */

package com.liferay.portal.security.auth;

import com.liferay.portal.kernel.log.Log;
import com.liferay.portal.kernel.log.LogFactoryUtil;
import com.liferay.portal.kernel.util.LocaleUtil;
import com.liferay.portal.kernel.util.StackTraceUtil;
import com.liferay.portal.kernel.util.StringPool;
import com.liferay.portal.kernel.util.Validator;
import com.liferay.portal.model.User;
import com.liferay.portal.service.ServiceContext;
import com.liferay.portal.service.UserLocalServiceUtil;
import com.liferay.portal.util.PortalUtil;
import com.liferay.portal.util.PrefsPropsUtil;
import com.liferay.portal.NoSuchUserException;
import com.liferay.portlet.admin.util.OmniadminUtil;
import com.liferay.util.PwdGenerator;
```

```
import java.util.Calendar;
import java.util.Date;
import java.util.Locale;

import com.liferay.portal.service.RoleLocalServiceUtil;
import com.liferay.portal.model.Role;
import java.util.List;
import java.util.Iterator;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

/**
 * <a href="ShibbolethAutoLogin.java.html"><b><i>View Source</i></b></a>
 *
 * This version works with Liferay 5.2.1.
 *
 * Version 2: This version assumes only the eduPersonScopedAffiliation
 * and eduPersonTargetedId is populated in the HTTP Headers
 */
public class ShibbolethAutoLogin implements AutoLogin {

	public String[] login(
		HttpServletRequest request, HttpServletResponse response) throws
AutoLoginException {

		String[] credentials = null; long companyId = 0; long userId = 0;
		String userName = request.getParameter("userName"); String userScreenName =
null;
		String affiliations = null; String ePTID = null;
		try {
			companyId = PortalUtil.getCompany(request).getCompanyId();
			_log.error("userName extracted: " + userName);
			java.util.Enumeration headers = request.getHeaderNames();
			while (headers!=null && headers.hasMoreElements()) { //Read headers
				String h = (String)headers.nextElement();
//				if (h.startsWith("Shib-")) {
					if (h.toLowerCase().endsWith("targeted-id")) {
						ePTID = request.getHeader(h);
						_log.error("ePTID found: " + ePTID);
					}
					if		(h.toLowerCase().endsWith("affiliation"))
affiliations = request.getHeader(h);
//				}
			}
			if (ePTID == null || ePTID.length() < 1) { // No eduPersonTargetedID
found
				_log.error("No header for 'eduPersonTargetedID' found - check
Apache, AJP and Shibboleth.");
				return credentials;
			}
			if (Validator.isNotNull(ePTID)) {		//Process shib user that
has an ePTargId
				credentials = new String[3];
				User user = UserLocalServiceUtil.getUserByOpenId(ePTID);
//				String[] ePTIDSplit = ePTID.split("@");
//				_log.error("scope resolved as: " + ePTIDSplit[1]);
				if (OmniadminUtil.isOmniadmin(user.getUserId()) ) {
					_log.error(user.getUserId() + " is an Omniadmin!");
					if (userName != null && userName.length() > 0 ) {	//
username unlikely to be populated
						_log.error("impersonating: " + userName);
						userId 
=
UserLocalServiceUtil.getUserByScreenName(companyId,userName).getUserId();
						credentials[0] = String.valueOf(userId);
						userScreenName = userName;
```

```
                                }
                                else {
                                      credentials[0]                                    =
String.valueOf(user.getUserId());
                                }
                          }
                          else {
                                _log.error(user.getUserId() + " is not an Omniadmin!");
                                credentials[0] = String.valueOf(user.getUserId());
                          }
                          credentials[1] = user.getPassword();
                          credentials[2] = Boolean.TRUE.toString();
//                        synchroniseAffiliation(affiliations,    ePTIDSplit[1],    user,
companyId);
                    }
                    return credentials;
              }
              catch (NoSuchUserException e){
              }
              catch (Exception e) {
                    _log.error(StackTraceUtil.getStackTrace(e));
                    throw new AutoLoginException(e);
              }
              try {   // Create New User
                    String password = PwdGenerator.getPassword();
                    String password2 = password;
                    long  creatorUserId  =  0;  boolean  autoUserId  =  false;  boolean
autoPassword = false;
                    boolean autoScreenName = true; Locale locale = Locale.getDefault();
                    String  firstName  =  "changeme";  String  lastName  =  "changeme";  String
middleName = null;
                    int  prefixId  =  0;  int  suffixId  =  0;  boolean  male  =  true;  String
jobTitle = null;
                    int  birthdayMonth  =  Calendar.JANUARY;  int  birthdayDay  =  1;  int
birthdayYear = 1970;
                    long[] groupIds = null; long[] roleIds = null;
                    long[] organisationIds = null; long[] userGroupIds = null;
                    boolean sendEmail = false; String mail = "change@me.com";
                    ServiceContext serviceContext= new ServiceContext();
                    String openId = StringPool.BLANK;
                    User user = UserLocalServiceUtil.addUser(
                          creatorUserId, companyId, autoPassword, password, password2,
                          autoScreenName, userScreenName, mail, ePTID, locale,
                          firstName, middleName, lastName, prefixId, suffixId,
                          male, birthdayMonth, birthdayDay, birthdayYear, jobTitle,
                          groupIds,  organisationIds,  roleIds,  userGroupIds,  sendEmail,
serviceContext);
                    if (user == null) {
                          _log.error("addUser() function returned null");
                    }
                    else {
                          user.setPassword(null);
                    }
                    credentials = new String[3];
                    credentials[0] = String.valueOf(user.getUserId());
                    credentials[1] = password;
                    credentials[2] = Boolean.FALSE.toString();
                    return credentials;
              }
              catch (Exception e) {
                    _log.error(StackTraceUtil.getStackTrace(e));
                    throw new AutoLoginException(e);
              }
        }

/*    public  void  synchroniseAffiliation(String  aff,  String  scope,  User  user,  long
companyId) throws AutoLoginException {
              String[] affRoles = aff.split(";");
```

```
            try {
                  List currentRoles = RoleLocalServiceUtil.getRoles(companyId);
                  for (int i=0; i<affRoles.length; i++) { // Loop to check for new
affiliations
                        int roleExistsFlag = 0;
                        Iterator itx = currentRoles.iterator();
                        while (itx.hasNext() && affRoles[i]!= null) {
                              Role storedRole = (Role)itx.next();
                              if            (affRoles[i].equals(storedRole.getName()))
roleExistsFlag = 1;
                        }
                        if (roleExistsFlag != 1) {


RoleLocalServiceUtil.addRole(user.getUserId(),companyId,affRoles[i],"eduPersonAffiliation
",1);
                              System.out.println("SPAM-GP:   Added   affiliation   "   +
affRoles[i] + " to database");
                        }
                  }
                  List                          storedUserRoles                         =
RoleLocalServiceUtil.getUserRoles(user.getUserId()); // Get roles user holds now
                  Iterator ity = storedUserRoles.iterator();
                  int roleIdIncrementer = 0;
                  long samlStoredRoleIds[] = new long[storedUserRoles.size()];
                  while (ity.hasNext()) {  // Create an array of the stored SAML role
Ids
                        Role userRole = (Role)ity.next();
                        long storedRoleId = userRole.getRoleId();
                        if ((userRole.getName()).endsWith(scope)) {


      }
*/

      private static Log _log = LogFactoryUtil.getLog(ShibbolethAutoLogin.class);
}
```

6.2 Java-Code der University of Southern California (ShibbolethAutologin.java)

Der Folgende Code wurde von James Schappet, U Iowa, für Liferay-Version 5.2.3 erstellt.

```
package edu.uiowa.icts.liferay.shibboleth;

import java.util.Calendar;
import java.util.Locale;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import com.liferay.portal.NoSuchUserException;
import com.liferay.portal.kernel.log.Log;
import com.liferay.portal.kernel.log.LogFactoryUtil;
import com.liferay.portal.kernel.util.StackTraceUtil;
import com.liferay.portal.kernel.util.Validator;
import com.liferay.portal.model.User;
import com.liferay.portal.security.auth.AutoLogin;
import com.liferay.portal.security.auth.AutoLoginException;
import com.liferay.portal.service.ServiceContext;
import com.liferay.portal.service.UserLocalServiceUtil;
import com.liferay.portal.util.PortalUtil;

public class ShibbolethAutoLogin implements AutoLogin {

       private static Log _log = LogFactoryUtil.getLog(ShibbolethAutoLogin.class);
```

```
    public String[] login(HttpServletRequest req, HttpServletResponse res)
            throws AutoLoginException {

                System.out.println("JCS: Starting ShibbolethAutoLogin.login() ");
            String[] credentials = null;
            String userScreenName = null;
            String pvid = null;
            String mail = null;
            String displayName = null;
            long userId = 0;
            String passwd = null;
            String userName = req.getParameter("userName");
            long companyId = 0;

            try {
                    companyId = PortalUtil.getCompany(req).getCompanyId();

                    //userScreenName =
req.getHeader(PrefsPropsUtil.getString(companyId,USCPropsUtil.USC_AUTH_SHIB_UID));
                    userScreenName = req.getHeader("eduPersonTargetedID");
                    userScreenName = "schappetj";
                    //uscpvid =
req.getHeader(PrefsPropsUtil.getString(companyId,USCPropsUtil.USC_AUTH_SHIB_USCOWNERPVID)
);
                    pvid = req.getHeader("eduPersonTargetedID");
                    //pvid = "schappetj";
                    mail = req.getHeader("mail");

                    System.out.println("Got mail: " + mail);
                    //mail = "james-schappet@uiowa.edu";
                    displayName = req.getHeader("displayName");
                    //displayName = "James Schappet";
                    //userScreenName = pvid;
                    if (userScreenName == null || userScreenName.length() < 1) {
                            _log.error("Did not receive uid from Shibboleth");
                            return credentials;
                    }
                    if (Validator.isNotNull(userScreenName)) {

                            credentials = new String[3];
                            //User user =
UserLocalServiceUtil.getUserByScreenName(companyId, userScreenName);
                            User user =
UserLocalServiceUtil.getUserByEmailAddress(companyId, mail);

                            //System.out.println("UserID: " + user.getFullName() );
/*                            if (OmniadminUtil.isOmniadmin(user.getUserId() )) {
                                    _log.error(user.getUserId()+" is an Omniadmin!");
                                  if (userName != null &&  userName.length() > 0) {
                                          _log.error("impersonating: "+userName);
                                          userId =
UserLocalServiceUtil.getUserByScreenName(companyId,userName).getUserId();
                                          credentials[0] = String.valueOf(userId);
                                          userScreenName = userName;
                                  }
                                  else {
                                          credentials[0] =
String.valueOf(user.getUserId());
                                  }
                            }
                            else {
                                     _log.error(user.getUserId()+" is not an
Omniadmin!");
                                    credentials[0] = String.valueOf(user.getUserId());
                            }*/
                            _log.error(user.getUserId()+" is not an Omniadmin!");
```

```
                        credentials[0] = String.valueOf(user.getUserId());
                        credentials[1] = user.getPassword();
                        credentials[2] = Boolean.TRUE.toString();
                }
                return credentials;
        }
        catch (NoSuchUserException e) {
            System.err.println("No Such User: ShibbolethAutoLogin.login()  ");
        }
        catch (Exception e) {
                _log.error(StackTraceUtil.getStackTrace(e));
                throw new AutoLoginException(e);
        }
        try {
                long creatorUserId = 0;
                //boolean autoUserId = false;
                boolean autoPassword = false;
                // we have to set the password to something.  replace this in the
                // future with something smarter
                String password1 = "replaceme";
                String password2 = "replaceme";
                boolean autoScreenName = false;
                String emailAddress = mail;
                Locale locale = Locale.US;
                String firstName = displayName;
                String middleName = null;
                String lastName = ".";
                String openId = "";
                int prefixId = 0;
                int suffixId = 0;
                boolean male = true;
                int birthdayMonth = Calendar.JANUARY;
                int birthdayDay = 1;
                int birthdayYear = 1970;
                String jobTitle = null;

                boolean sendEmail = false;

                // setting password to null returns errors (actually null)
                long[] groupIds = null;
                long[] organizationIds = null;
                long[] roleIds = null;
                long[] userGroupIds = null;
                ServiceContext serviceContext = new ServiceContext();
                User user = null;
                try {
                        user = UserLocalServiceUtil.addUser(creatorUserId,
companyId, autoPassword,
                                        password1,password2, autoScreenName,
userScreenName, emailAddress,
                                        openId, locale,
                                        firstName, middleName, lastName, prefixId,
suffixId,
                                        male, birthdayMonth, birthdayDay, birthdayYear,
jobTitle,
                                        groupIds, organizationIds, roleIds,
userGroupIds,
                                        sendEmail, serviceContext);
                } catch(Exception e){
                     System.out.println(e.getMessage());

                }


                if (user == null) {
                        _log.error("LDAPImportUtil.addOrUpdateUser() returned null");
                } else {
                        // we want a null password but this is not working - why?
```

```
                        user.setPassword(null);
                }
                _log.error("finished calling: LDAPImportUtil.addOrUpdateUser()");
                //ldapLookup(companyId, userScreenName);

                credentials = new String[3];
                credentials[0] = String.valueOf(userId);
                credentials[1] = password1;
                credentials[2] = Boolean.FALSE.toString();
                System.out.println("UserID: " +String.valueOf(userId) + "\n\t");
                return credentials;
            }
            catch (Exception e) {
                _log.error(StackTraceUtil.getStackTrace(e));
                throw new AutoLoginException(e);
            }
    }
}
```

6.3 Java-Code der University of Iowa (ShibbolethAutologin.java)

Der Folgende Code wurde von James Hong, USC, für Liferay-Version 4.2 erstellt.

```
package edu.uiowa.icts.liferay.shibboleth;

import java.util.Calendar;
import java.util.Locale;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import com.liferay.portal.NoSuchUserException;
import com.liferay.portal.kernel.log.Log;
import com.liferay.portal.kernel.log.LogFactoryUtil;
import com.liferay.portal.kernel.util.StackTraceUtil;
import com.liferay.portal.kernel.util.Validator;
import com.liferay.portal.model.User;
import com.liferay.portal.security.auth.AutoLogin;
import com.liferay.portal.security.auth.AutoLoginException;
import com.liferay.portal.service.ServiceContext;
import com.liferay.portal.service.UserLocalServiceUtil;
import com.liferay.portal.util.PortalUtil;

public class ShibbolethAutoLogin implements AutoLogin {

       private static Log _log = LogFactoryUtil.getLog(ShibbolethAutoLogin.class);


    public String[] login(HttpServletRequest req, HttpServletResponse res)
            throws AutoLoginException {

             System.out.println("JCS: Starting ShibbolethAutoLogin.login() ");
            String[] credentials = null;
            String userScreenName = null;
            String pvid = null;
            String mail = null;
            String displayName = null;
            long userId = 0;
            String passwd = null;
            String userName = req.getParameter("userName");
```

```
            long companyId = 0;

            try {
                    companyId = PortalUtil.getCompany(req).getCompanyId();

                                                                        //userScreenName    =
req.getHeader(PrefsPropsUtil.getString(companyId,USCPropsUtil.USC_AUTH_SHIB_UID));
                    userScreenName = req.getHeader("eduPersonTargetedID");
                    userScreenName = "schappetj";
                                                                                //uscpvid    =
req.getHeader(PrefsPropsUtil.getString(companyId,USCPropsUtil.USC_AUTH_SHIB_USCOWNERPVID)
);
                    pvid = req.getHeader("eduPersonTargetedID");
                    //pvid = "schappetj";
                    mail = req.getHeader("mail");

                    System.out.println("Got mail: " + mail);
                    //mail = "james-schappet@uiowa.edu";
                    displayName = req.getHeader("displayName");
                    //displayName = "James Schappet";
                    //userScreenName = pvid;
                    if (userScreenName == null || userScreenName.length() < 1) {
                            _log.error("Did not receive uid from Shibboleth");
                            return credentials;
                    }
                    if (Validator.isNotNull(userScreenName)) {

                            credentials = new String[3];
                                                                        //User   user   =
UserLocalServiceUtil.getUserByScreenName(companyId, userScreenName);
                                                                          User   user   =
UserLocalServiceUtil.getUserByEmailAddress(companyId, mail);

                            //System.out.println("UserID: " + user.getFullName() );
/*                            if (OmniadminUtil.isOmniadmin(user.getUserId() )) {
                                    _log.error(user.getUserId()+" is an Omniadmin!");
                                  if (userName != null &&  userName.length() > 0) {
                                          _log.error("impersonating: "+userName);
                                                                                  userId  =
UserLocalServiceUtil.getUserByScreenName(companyId,userName).getUserId();
                                          credentials[0] = String.valueOf(userId);
                                          userScreenName = userName;
                                  }
                                  else {
                                                                        credentials[0]  =
String.valueOf(user.getUserId());
                                  }
                            }
                            else {
                                                  _log.error(user.getUserId()+" is not an
Omniadmin!");
                                  credentials[0] = String.valueOf(user.getUserId());
                            }*/
                            _log.error(user.getUserId()+" is not an Omniadmin!");
                            credentials[0] = String.valueOf(user.getUserId());
                            credentials[1] = user.getPassword();
                            credentials[2] = Boolean.TRUE.toString();
                    }
                    return credentials;
            }
            catch (NoSuchUserException e) {
             System.err.println("No Such User: ShibbolethAutoLogin.login()  ");
            }
            catch (Exception e) {
                    _log.error(StackTraceUtil.getStackTrace(e));
                    throw new AutoLoginException(e);
            }
            try {
```

```
                    long creatorUserId = 0;
                    //boolean autoUserId = false;
                    boolean autoPassword = false;
                    // we have to set the password to something.  replace this in the
                    // future with something smarter
                    String password1 = "replaceme";
                    String password2 = "replaceme";
                    boolean autoScreenName = false;
                    String emailAddress = mail;
                    Locale locale = Locale.US;
                    String firstName = displayName;
                    String middleName = null;
                    String lastName = ".";
                    String openId = "";
                    int prefixId = 0;
                    int suffixId = 0;
                    boolean male = true;
                    int birthdayMonth = Calendar.JANUARY;
                    int birthdayDay = 1;
                    int birthdayYear = 1970;
                    String jobTitle = null;

                    boolean sendEmail = false;

                    // setting password to null returns errors (actually null)
                    long[] groupIds = null;
                    long[] organizationIds = null;
                    long[] roleIds = null;
                    long[] userGroupIds = null;
                    ServiceContext serviceContext = new ServiceContext();
                    User user = null;
                    try {
                                        user = UserLocalServiceUtil.addUser(creatorUserId,
companyId, autoPassword,
                                                password1,password2,     autoScreenName,
userScreenName, emailAddress,
                                           openId, locale,
                                             firstName,  middleName,  lastName,  prefixId,
suffixId,
                                           male, birthdayMonth, birthdayDay, birthdayYear,
jobTitle,
                                                 groupIds,    organizationIds,    roleIds,
userGroupIds,
                                           sendEmail, serviceContext);
                    } catch(Exception e){
                         System.out.println(e.getMessage());

                    }


                    if (user == null) {
                            _log.error("LDAPImportUtil.addOrUpdateUser() returned null");
                    } else {
                            // we want a null password but this is not working - why?
                            user.setPassword(null);
                    }
                    _log.error("finished calling: LDAPImportUtil.addOrUpdateUser()");
                    //ldapLookup(companyId, userScreenName);

                    credentials = new String[3];
                    credentials[0] = String.valueOf(userId);
                    credentials[1] = password1;
                    credentials[2] = Boolean.FALSE.toString();
                    System.out.println("UserID: " +String.valueOf(userId) + "\n\t");
                    return credentials;
            }
            catch (Exception e) {
                    _log.error(StackTraceUtil.getStackTrace(e));
```

```
                    throw new AutoLoginException(e);
                }
        }
}
```