Shibboleth: Vollständiges Single Logout durch Kopplung von Anwendungs- und Shibboleth-Session am Apache-Webserver

Frank Schreiterer[1]

Abstract: Single SignOn (SSO) im Rahmen einer Authentication and Authorization Infrastructure (AAI) ist sehr verbreitet. Aktuell werden die Entwicklungen für das Single Logout (SLO) wieder vorangetrieben. Beim SSO-Verfahren Shibboleth besteht das Problem, dass beim Terminieren der Shibboleth-Session am Service-Provider nicht zwangsläufig auch die Anwendungs-Session terminiert wird. Dieser Beitrag zeigt eine Möglichkeit auf, vollständiges Single Logout zu erreichen. Dazu werden Shibboleth- und Anwendungs-Session direkt am Apache-Webserver gekoppelt, ohne in die Anwendung selbst einzugreifen.

Keywords: Shibboleth, Single Logout, SLO, Session-Management

1 Ausgangssituation

Durch Single SignOn, kurz **SSO**, lassen sich, basierend auf der Security Assertion Markup Language**,** kurz **SAML**, (vgl. [OA05], S. 1ff.), zahlreiche Webanwendungen grundsätzlich mittels des Shibboleth-Service-Providers, kurz **SP,** (vgl. [SE15]) authentifizieren und autorisieren. Das Abmelden an allen aufgerufenen Anwendungen, das Single Logout, kurz **SLO**, bereitet jedoch einige Probleme, wodurch die Entwicklung nur schleppend vorankommt. Die größte Herausforderung ist, dass sehr viele Anwendungen ein eigenes Session-Management integriert haben. Sie setzen nicht direkt auf die Shibboleth-Session auf. Somit besteht kein direkter Zusammenhang zwischen Anwendungs-Session und Shibboleth-Session (vgl. [SI15], [SL15]). Daneben existieren Anwendungen, welche nativ, d. h. ohne Shibboleth-Service-Provider, SAML unterstützen (vgl. [Ha14], S. 3). Bei diesen Anwendungen treten diese Probleme nicht auf. Sie werden im Folgenden nicht betrachtet.

Der Shibboleth-Identity-Provider, kurz **IdP**, unterstützt ab der Version 3.2 SLO über die Front-Channel-Implementierung, d. h. über den Webbrowser des Nutzers (vgl. [RN15]). Die vollständige Implementierung des SLO auch über Back-Channel, d. h. im Hintergrund via SOAP (Simple Object Access Protocol), wobei IdP und SP direkt miteinander kommunizieren, ist in Umsetzung.

[1] Otto-Friedrich-Universität Bamberg, Rechenzentrum Abteilung Serversysteme & Nutzerverwaltung, Feldkirchenstraße 21, 96052 Bamberg, frank.schreiterer@uni-bamberg.de

Seit IdP-Version 2.1 bis 2.3 existiert eine Implementierung des NIIF Institute (National Information Infrastructure Development Institute aus Ungarn) für SLO (vgl. [AS16]), die sowohl Front-Channel als auch Back-Channel unterstützt (vgl. [SL15]) und für Version 2.4 als Plugin portiert wurde (vgl. [MH16]). Diesen Implementierungen ist gemeinsam, dass beim Logout nur die Shibboleth-Session (SAML-Session), jedoch nicht die Anwendungs-Session (App-Session) terminiert wird (vgl. Abbildung 1).

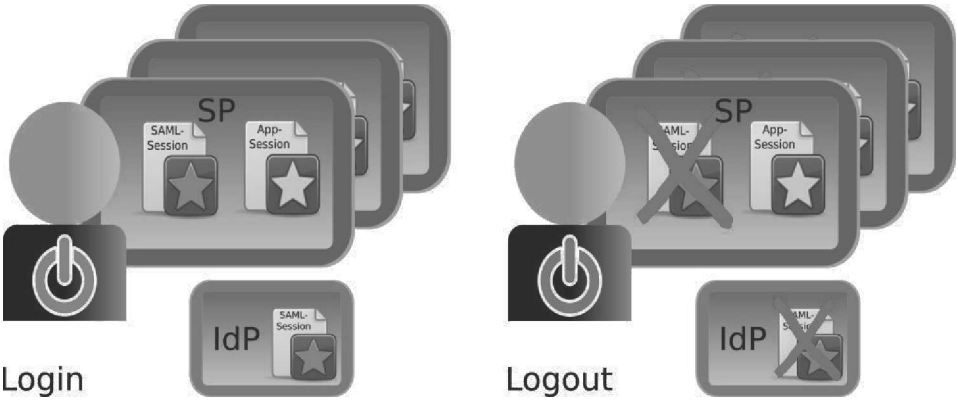

Abbildung 1: Ist-Zustand beim SSO und SLO

Neben diesem Problem resultieren aus der fehlenden Kopplung zwischen Anwendungs- und Shibboleth-Session folgende Probleme:

1. Zugriff ohne Shibboleth-Session
   In diesem Fall besteht die Gefahr eines nicht-autorisierten Zugriffs auf die Anwendung. Dieser erfolgt mit der bestehenden Anwendungs-Session ohne bzw. ungültige Shibboleth-Session, z. B. wenn die Shibboleth-Session am SP beendet wurde und die Existenz einer gültigen Shibboleth-Session nicht Voraussetzung für den Zugriff ist.

2. Manipulation der Anwendungs-Session
   Besitzt ein Angreifer eine gültige Shibboleth-Session, so kann die Anwendungs-Session manipuliert werden und damit ggf. eine nicht mehr bestehende Berechtigung in der Anwendung erreicht werden.

3. Eingriff in bestehende Software unmöglich
   Die Missstände aus 1. und 2. könnten mit Programmieraufwand beseitigt werden. Jedoch setzt dies voraus, dass der Quelltext vorhanden ist und dass die notwendigen Änderungen mit jedem Update wieder geprüft und angepasst werden. Oft ist man nicht in der Lage, diesen erheblichen Wartungsaufwand zu betreiben. Liegt der Quelltext nicht vor, so ist ein Eingriff in die bestehende Software unmöglich.

Ziel ist es daher, die Shibboleth-Session und die Anwendungs-Session direkt am Apache-Webserver ohne Eingriffe in die Anwendung zu koppeln und alle Sessions beim Logout zu terminieren und damit ein vollständiges SLO zu erreichen.

2 Kopplung am Apache-Webserver

Die Abschnitte zeigen Analyse, Entwurf und die Implementierung einer Lösung am Apache-Webserver, mit der eine Kopplung der Anwendungs-Session an die Shibboleth-Session erreicht und damit ein vollständiges SLO ermöglicht wird.

2.1 Analyse

Der Apache-Webserver bietet eine Vielzahl an Kontroll- und Manipulationsmechanismen z. B. bei Umgebungsvariablen und Cookies. Ein sehr mächtiges Werkzeug hierzu ist das Modul **mod_rewrite** (vgl. [AH15], [AM15]). Dieses Modul ermöglicht neben der Manipulation von URIs (Uniform Resource Identifiers) auch das Auslesen von HTTP-Headern und Cookies.

Standardmäßig stellt das Modul des Shibboleth-Service-Providers für den Apache-Webserver neben den Attributwerten die Shibboleth-Session-ID in den Umgebungsvariablen bereit (vgl. [NS15a]). Die Anwendung muss die Session-Information im Cookie oder in der URI speichern, damit diese am Webserver zugänglich ist. Somit ist der Zugriff auf den Identifier der Shibboleth-Session sowie auf den Identifier der Anwendungs-Session mittels mod_rewrite am Apache-Webserver gewährleistet und eine prinzipielle Kopplung von Shibboleth- und Anwendungs-Session möglich.

Die Kopplung muss in einem Datenspeicher abgelegt werden, um Prüfungen durchführen zu können. Prinzipiell kann hier als Datenspeicher memcached, ein einfacher Schlüssel–Wert–Speicher im RAM (vgl. [Mc15]), eingesetzt werden. Allerdings gehen bei einem Reboot oder Neustart des Dienstes die Beziehungen zwischen Shibboleth-Session und der Anwendungs-Session verloren. Gegen die Verwendung einer Datenbank kann der relativ hohe Administrationsaufwand sprechen. Für die Verwendung von memcached spricht, dass dieser zumindest unter Linux sehr einfach über die Paketverwaltung der gängigen Distributionen installiert werden kann.

Anwendungen können die Shibboleth-Session auf drei unterschiedliche Arten anfordern:

1. normal:
   Im Standardfall ist die Anwendung stets durch Shibboleth geschützt (vgl. [NS15b]) und damit ist die Shibboleth-Session immer vorhanden.

2. lazy:
   Die Authentifizierung gegen Shibboleth erfolgt anwendungsgesteuert nach Erfordernis (vgl. [NS15b]). Dadurch ist die Shibboleth-Session nicht stets vorhanden. Shibboleth bleibt aber die einzige Authentifizierungsmöglichkeit.

3. mixedLazy:
   Die Authentifizierung erfolgt anwendungsgesteuert nach Erfordernis. Neben der Authentifizierung gegen Shibboleth kann auch gegen andere Mechanismen authentifiziert werden. Somit kann eine Kopplung zwischen Shibboleth-Session und Anwendungs-Session nur erfolgen, wenn gegen Shibboleth authentifiziert wird.

Nachfolgend wird von den Anwendungsszenarien *normal, lazy und mixedLazy* gesprochen.

2.2 Entwurf

Bevor die Kopplung zwischen Anwendungs-Session und Shibboleth-Session erfolgen kann, muss in den Anwendungsszenarien normal und lazy sichergestellt sein, dass keine Anwendungs-Session vorhanden ist. Ansonsten könnte ungewollt oder vorsätzlich und mutwillig eine nicht mehr zulässige Anwendungs-Session Verwendung finden. Hierfür kann nach erfolgreicher Authentifizierung am IdP und dem Erzeugen der Shibboleth-Session am SP ein **sessionHook** definiert werden, der abgearbeitet wird, bevor das Redirect an die eigentliche Ziel-URI erfolgt (vgl. [NS15c]). Ist eine Anwendungs-Session vorhanden muss die Anfrage abgewiesen werden.

Im Anwendungsszenario mixedLazy kann diese Vorbedingung nicht geprüft werden, da Shibboleth nicht die einzige Authentifizierungsmöglichkeit ist und somit eine vorhandene Anwendungs-Session nicht zum Abweisen der Anfrage führen darf. Erfolgt die Authentifizierung über Shibboleth, so kann jedoch geprüft werden, ob der Identifier der Anwendungs-Session bereits vergeben war. Ist dies der Fall, so muss die Anfrage abgewiesen werden. Der weitere Ablauf erfolgt dann analog zum Anwendungsszenario lazy.

Folgende Parameter müssen somit für Validierung und Kopplung von Shibboleth- und Anwendungs-Session verarbeitet werden:

1. Kontext:
   a) sessionHook, wenn geprüft werden muss, ob eine Anwendungs-Session bereits vorhanden ist; dies ist nur bei den Anwendungsszenarien normal und lazy möglich.
   b) normal, wenn die Anwendung im Szenario normal verwendet wird oder
   c) lazy, wenn die Anwendung in den Szenarien lazy oder mixedLazy verwendet wird.

2. Shibboleth-Session-ID: Der Identifier der Shibboleth-Session, die vom SP-Modul an den Webserver übergeben wird.
3. Name des Cookies der Anwendung-Session, um den Identifier der Session extrahieren zu können.
4. Cookies: Alle Cookies, damit die Kopplung durchführbar ist und Prüfungen vorgenommen werden können. Die Cookies der Shibboleth-Session und der Anwendungs-Session sind in diesen Cookies enthalten.
5. mixedLazy: Ist der Schalter für die Prüfung auf doppelte Identifier der Anwendungs-Session, wobei dieser nur im Anwendungsszenario mixedLazy Verwendung finden darf.

Ausgehend von diesen fünf Parametern findet eine mehrstufige Validierung statt, bis schlussendlich der Identifier der Shibboleth-Session mit dem Identifier der Anwendung-Session assoziiert wird.

2.2.1 Grundsätzliche Validierungen

Cookies aus Parameter 4 (vgl. Abschnitt 2.2) enthalten neben etwaigen anderen Cookies diejenigen mit den Identifiern der Shibboleth-Session sowie der Anwendungs-Session. Der Name des Cookies der Anwendungs-Session bestimmt sich dabei aus Parameter 3. Ebenfalls muss der Identifier der Shibboleth-Session vom SP aus Parameter 2 mit dem Identifier des Shibboleth-Session-Cookies übereinstimmen.

Identifier von Shibboleth- und Anwendungs-Session weisen jeweils ein spezifisches Muster auf. Gegen diese Muster werden beide Sessions geprüft. Weiterhin wird untersucht, dass jeweils nur ein Cookie für die Shibboleth- und Anwendungs-Session übermittelt wurde, d. h. ob nicht versucht wurde, zusätzliche Cookies einzuschleusen.

Im Kontext „sessionHook“ (Parameter 1.a)) wird geprüft, dass keine Anwendungs-Session vorhanden ist. Sind all diese grundsätzlichen Validierungen erfolgreich, so erfolgt anschließend die erweiterte Validierung.

2.2.2 Erweiterte Validierungen

Nach den erfolgreich abgeschlossenen grundsätzlichen Validierungen muss in den Anwendungsszenarien normal und lazy festgestellt werden, ob bereits die Kopplung zwischen Shibboleth-Session und Anwendungs-Session besteht. Wird diese nicht festgestellt, so handelt es sich um eine neue Sitzung. Die Kopplung von Shibboleth-Session und Anwendungs-Session wird im Datenspeicher abgelegt.

Bei einer vorhandenen Kopplung wird zum übergebenen Identifier der Shibboleth-Session der zugehörige Identifier der Anwendungs-Session aus dem Datenspeicher gelesen. Stimmen gelesener und per Cookie übergebener Identifier der Anwendungs-Session

überein, so ist die Anfrage gültig.

Zusätzlich müssen noch zwei Sonderzustände beachtet werden.

1. Im Moment der durchgeführten Authentifizierung mit Shibboleth darf noch keine Anwendungs-Session vorhanden sein, es muss aber eine Shibboleth-Session-ID existieren. Dieser Zustand muss als gültige Anfrage definiert werden.
2. Im Kontext „lazy“ existiert beim ersten Aufruf der Zielressource ein Zustand, bei dem weder die Shibboleth-Session-ID noch die Anwendungs-Session-ID vorhanden sind. Dies muss ebenfalls als gültige Anfrage betrachtet werden.

Das Anwendungsszenario mixedLazy lässt neben Shibboleth die Verwendung von weiteren Authentifizierungsmethoden zu. Damit ist es mit diesem Mechanismus nicht möglich, die Anwendung bei Nicht-Shibboleth-Authentifizierung gegen die Verwendung einer vorhandenen oder manipulierten Anwendungs-Session zu schützen. Hier muss man sich, wie bisher auch, auf die Schutzmechanismen der Anwendung verlassen. Erfolgt die Authentifizierung jedoch über Shibboleth, so ist es zwar nicht möglich, eine Anmeldung mit einer bestehenden Session zu unterbinden. Es kann aber eine Duplikatsprüfung vorgenommen werden.

In der Duplikatsprüfung wird untersucht, ob der Identifier der gelieferten Anwendungs-Session schon einmal an Hand der vorhandenen Daten aus dem Datenspeicher Verwendung fand. Dies bietet bei entsprechender Konfiguration der Verweildauer der Daten im Datenspeicher quasi den gleichen Schutz wie das grundsätzliche Verbot, sich mit einer bestehenden Anwendungs-Session anzumelden. Der Mechanismus zur Vorabprüfung mittels sessionHook auf eine bestehende Anwendungs-Session kann und darf hier nicht durchgeführt werden.

Folgender Abschnitt zeigt die praktische Umsetzung aus Analyse und Entwurf.

2.3 Implementierung

Das Modul mod_rewrite bietet neben den klassischen RewriteRules die Möglichkeit, per *RewriteMap* Skripte zur Bestimmung des Ergebnisses einer RewriteCond (Bedingung) ausführen zu lassen. Auf das Ergebnis der RewriteCond kann dann in einer RewriteRule reagiert werden (vgl. [AM15]). Damit ist es möglich, das zuvor definierte komplexe Regelwerk in einem externen Skript umzusetzen. Versuche, das Regelwerk nur mit RewriteRules umzusetzen, erwiesen sich als nicht zielführend, da Übersicht und Wartungsfreundlichkeit rasch verloren gehen. Als Skriptsprache für die prototypische Implementierung wurde PHP mit memcached als Datenspeicher gewählt[2], da PHP bereits Basis zahlreicher Anwendungen ist. Zur Implementierung der RewriteMap können aber beliebige unterstützte Skriptinterpreter verwendet werden (vgl. [AM15]).

[2] Der Quelltext des Prüfskripts einschließlich Konfiguration und Hilfsskripten für ein vollständiges Logout finden sich unter https://wiki.aai.dfn.de/de:shibslohttpd.

Während der Implementierung trat das Problem auf, dass nicht zu jedem Zeitpunkt der Anfrage die Variablen des Shibboleth-Daemons in den Umgebungsvariablen des Apache-Webservers zur Verfügung stehen. Gelöst konnte dieses nur werden, indem die Shibboleth-Variablen per Direktive *ShibUseHeaders On* auch in den Headern verfügbar gemacht wurden. Dadurch besteht grundsätzlich die Möglichkeit des HeaderSpoofings (vgl. [NS15a]). Jedoch wird vom Shibboleth-Daemon ein HeaderSpoofing mit dem Leeren der Shibboleth-Session quittiert, sodass die Gefahr als eher gering eingestuft werden kann. Es wird jedoch dringend empfohlen, in den Shibboleth nachgelagerten Anwendungen auf Umgebungsvariablen zuzugreifen.

2.3.1 Rückgabewerte Prüfskript

In der RewriteCond wird das Prüfskript mit den Parametern Kontext, Shibboleth-Session-ID, Name des Cookies der Anwendung-Session und den Cookies sowie dem optionalen Parameter 5 „mixedLazy“ aufgerufen (vgl. Abschnitt 2.2). Durch das Prüfskript werden folgende Werte zurückgegeben, auf die mit einer entsprechenden RewriteRule reagiert werden muss:

1. doLogout, d. h. es muss das Logout (vgl. Abschnitt 2.3.3) durchgeführt werden, wenn versucht wurde,
   a) mit einer Anwendungs-Session ohne Shibboleth-Session zuzugreifen,
   b) die Anwendungs-Session während der Sitzung zu verändern,
   c) zusätzlich eine weitere Anwendungs-Session und / oder Shibboleth-Session per Cookie einzuschleusen,
   d) oder die Shibboleth-Session während der Sitzung zu verändern.
2. doAppSession (Sonderzustand 1 aus Abschnitt 2.2.2),
   wenn die Initialisierung der Anwendungs-Session durch einen zusätzlich erzeugten Request sichergestellt werden muss, da ansonsten das Einschleusen einer vorhandenen Anwendungs-Session möglich ist. Der zusätzliche Request lässt sich z. B. durch Hinzufügen eines Refresh-Headers direkt am Webserver erreichen.
3. doLogin (Sonderzustand 2 aus Abschnitt 2.2.2),
   wenn ein Redirect an das Login erfolgen muss, da ansonsten das Einschleusen einer vorhandenen Anwendungs-Session möglich ist. Dieser Rückgabewert kann nur im Kontext lazy erreicht werden.
4. good,
   wenn keine Aktion notwendig ist.

2.3.2 Konfiguration für sessionHook

Wie zuvor ausgeführt, kann durch den sessionHook ein zusätzlicher Request erzeugt werden, der am SP in der shibboleth2.xml mit

```
<ApplicationDefaults
entityID="https://YOURSERVER/shibboleth"
REMOTE_USER="uid"
sessionHook="/checker/checker.php">
```

für die Anwendungsszenarien normal und lazy, jedoch nicht für mixedLazy konfiguriert wird. Das Programm checker.php erzeugt lediglich ein Redirect auf die Anwendung und damit den zusätzlich benötigten Request, um die Prüfung auf die nicht vorhandene Anwendungs-Session durchführen zu können. Am Apache-Webserver wird die Location „checker“ normal mit Shibboleth geschützt und die Gültigkeit der Anfrage per Prüfskript in der RewriteCond validiert, vgl. nachstehender Auszug aus der Konfiguration.

```
<Location /checker>
authType shibboleth
ShibRequestSetting requireSession true
ShibUseHeaders On
RewriteEngine On
#den Wert der Anwendungssession auslesen, um die
#Anwendungssession sauber beim Logout zerstören zu können
RewriteCond %{HTTP:Cookie} APPSESSIONNAME=([^;]+)
RewriteRule .* - [E=appsid:%1]
#Aufruf Prüfskript und Logout bei Rückgabewert doLogout
RewriteCond ${shibchecker:sessionHook,%{HTTP:Shib-
Session-ID},APPSESSIONNAME,%{HTTP:Cookie}} ^doLogout$
RewriteRule .*
https://YOURSERVER/eviluse/SESSIONREMOVER.php?
appsid=%{ENV:appsid
</Location>
```

2.3.3 Entfernen der Sicherungseinträge beim Logout

Im Datenspeicher werden die Sicherungseinträge als Paar der Identifier von Shibboleth-Session und Anwendungs-Session abgelegt. Über diese gespeicherte Kopplung lassen sich diese wieder aus dem Datenspeicher entfernen. Hierzu muss am SP in der Datei shibboleth2.xml das Notify-Element (vgl. [NS15d]) definiert werden:

```
<Notify Channel="back"
Location="https://YOURSERVER/logoutnotify.php"/>
<Notify Channel="front"
Location="https://YOURSERVER/logoutnotify.php" />
```

Das Notify-Element kann jeweils für Front-Channel (Kommunikation über den Browser) und Back-Channel (Kommunikation über SOAP) konfiguriert werden. Dieser Umstand wird genutzt, um per Front-Channel nur die Cookies der Anwendung zu zerstören und um per Back-Channel die Einträge im Datenspeicher zu entfernen und die Anwendungs-Session in der Anwendung selbst zu zerstören.

3 Implikationen für das Single Logout

Mit der vorgestellten Methodik ist es möglich, die Kopplung der Anwendungs-Session an die Shibboleth-Session herzustellen, ohne dass es notwendig ist, Veränderungen an der Anwendung vorzunehmen. Bisherige Implementierungen des NIIF für IdP-Version 2.1 bis 2.3 sowie auch die aktuelle Implementierung im IdP-Version 3.2 lassen das Problem der Kopplung der Anwendungs-Session an die Shibboleth-Session offen. Auch das Shibboleth-Entwicklungsteam zieht sich auf die Aussage zurück, dass die Terminierung der Anwendungs-Session beim Invalidieren der SAML-Session implementiert werden muss (vgl. [SL15], [SI15]).

Mit der vorliegenden Arbeit wird eine Möglichkeit aufgezeigt, die zum breiteren und zuverlässigen Einsatz des vollständigen SLO beitragen kann. Neben dem Problem der fehlenden Session-Kopplung werden von den Shibboleth-Entwicklern weitere technische Probleme diskutiert (vgl. [SI15]), die jedoch als weitestgehend gelöst gelten können. Nicht technische Faktoren wie Nutzerakzeptanz oder auch die organisationsweite Entscheidung, SLO einzusetzen, können hierdurch nicht gelöst, aber entscheidend unterstützt werden.

Damit vollständiges SLO breitere Anwendung findet, wäre allerdings eine Übernahme der Funktionalität direkt in den Shibboleth-Service-Provider notwendig. Mit Unterstützung des DFN-Vereins als Mitglied des Shibboleth-Consortium (vgl. [SC16]) soll auf die Entwickler eingewirkt werden, die Funktionalität direkt in den Shibboleth-Service-Provider aufzunehmen. Damit könnte die Administration einer Anwendung auf die Installation von Zusatzsoftware für den Webserver sowie zusätzliche Speicherstrukturen wie memcached oder Datenbank verzichten und direkt das Session-Management des SPs verwenden. Sollte dies nicht gelingen, so wäre es alternativ denkbar, die Lösung für andere Webserver wie z. B. Internet Information Server oder nginx zu portieren.

Literaturverzeichnis

[AH15] Apache HTTP Server Version 2.4 Documentation, http://httpd.apache.org/docs/2.4/en/, Abruf am 23.11.2015

[AM15] Apache Module mod_rewrite, http://httpd.apache.org/docs/2.4/mod/mod_rewrite.html, Abruf am 23.11.2015

[AS16] AAI Software provided by NIIF Institute, http://software.niif.hu/, Abruf 21.03.2016

[Ha14] Shibboleth-aware Applications, https://www.switch.ch/aai/support/presentations/sp-training-2014/T6-5-Shibboleth-Enabled-Applications.pdf, Abruf am 23.11.2015

[IE16] IdPEnableSLO, https://wiki.shibboleth.net/confluence/display/SHIB2/IdPEnableSLO, Abruf 21.03.2016

[OA05] OASIS (Hrgs.): Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0, https://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf, Abruf 23.11.2015

[Mc15] memcached, https://github.com/memcached/memcached/wiki/Overview, Abruf 20.11.2015

[MH16] Manuel Haim, http://www.staff.uni-marburg.de/~haimm/, Abruf 17.03.2016

[NS15a] NativeSPSpoofChecking, https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPSpoofChecking, Abruf 24.11.2015

[NS15b] NativeSPEnableApplication, https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPEnableApplication, Abruf 25.11.2015

[NS15c] NativeSPApplication, https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPApplication, Abruf 25.11.2015

[NS15d] NativeSPNotify, https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPNotify, Abruf 26.11.2015

[PS14] Shibboleth – Single-Log-Out?, https://www.zki.de/fileadmin/zki/Arbeitskreise/VD/Protokolle/2014-03-20/beitrag_vd-ak_2014-03-20_schreiterer_pempe.pdf, Abruf 21.03.2016

[SC16] Shibboleth Consortium – Members, http://shibboleth.net/consortium/, Abruf 21.03.2016

[SE15] Shibboleth® Enabled Applications and Services, https://wiki.shibboleth.net/confluence/display/SHIB2/ShibEnabled, Abruf 23.11.2015

[SI15] SLOIssues, https://wiki.shibboleth.net/confluence/display/CONCEPT/SLOIssues, Abruf 26.11.2015

[SL15] Single Logout in Shibboleth IdP, https://wiki.aai.niif.hu/index.php/ShibIdpSLO; Abruf 26.11.2015

[RN15] ReleaseNotes, https://wiki.shibboleth.net/confluence/display/IDP30/ReleaseNotes, Abruf 24.11.2015