Identity Management mit OpenID

Innovative Internet-Technologien und Mobilkommunikation WS2008/2009

Steffen Märkl
Lehrstuhl Netzarchitekturen und Netzdienste
Institut für Informatik
Technische Universität München
Email: maerkl@in.tum.de

***Kurzfassung*— Durch den Wandel des Internets in den vergangen Jahren muss sich ein Benutzer zunehmend mehr Logins und Passwörter für die verschiedensten Dienste merken. Zusätzlich müssen immer wieder die gleichen personenbezogenen Informationen zur Registrierung angeben werden, sei es nun bei Amazon, Ebay oder sonstigen Portalen. Bei dieser Fülle von Anmeldedaten und Logins kann der Anwender schnell den Überblick verlieren. Hier setzt Identity Management an. Der Benutzer soll die Möglichkeit haben, sich mit einem einzigen Account bei einem sog. Identity Provider, dem er seine Daten anvertraut, bei allen gewünschten Websites und -applikationen anzumelden. Dabei werden ausschließlich die Daten übermittelt, die er dem jeweiligen Anbieter zur Verfügung stellen möchte. OpenID ist eines der bekanntesten Identity Management Systeme, welches mittels Open Source Ansatz und einer unkomplizierten und einfach zu implementierenden Technik immer mehr Anbieter und auch Benutzer zählt.**

***Schlüsselworte*— Identity Management, Single Sign-On (SSO), OpenID, Identity Provider**

I. Einleitung

In der heutigen Welt, in der webbasierte Dienste immer mehr an Bedeutung gewinnen, ist jeder Nutzer im Internet täglich mit zahlreichen Authorisierungsmechanismen konfrontiert und muss eine Vielzahl von verschiedenen Identitäten verwalten. Benutzerkennung und Passwort werden für viele Tätigkeiten im Internet benötigt, z.B. um Emails online zu verwalten, Bücher in einem Online-Versandhandel zu kaufen, oder Beiträge in einem Blog oder Forum zu erstellen. Zusätzlich zu diesen regelmäßig genutzen Identitäten müssen sich Internetnutzer oft auch für Dienste anmelden, die sie eher selten in Anspruch nehmen. [1] Hier gilt es den Überblick über sämtliche Logins und Passwörter zu behalten und diese im Sinne einer möglichst hohen Sicherheit regelmäßig zu erneuern, sowie nach bestimmten Richtlinien und Vorgaben zu erstellen. Viele Anwender verwenden aufgrund dessen beim Großteil der von Ihnen genutzen Webdienste ein und dieselben Zugangsdaten. Dadurch haben Angreifer oft ein leichtes Spiel an vertrauliche Informationen zu gelangen oder sich der Identitäten ihrer Opfer zu bemächtigen.
In den letzten Jahren sind sog. Identity Management Systeme immer populärer geworden und erfreuen sich immer größerer Beliebtheit. Diese Systeme ermöglichen es dem Benutzer sich seiten- und dienstübergreifend mit einer virtuell einmaligen Identität Zugang zu verschiedensten Internet-Plattformen und -Diensten zu verschaffen, ohne sich dabei mehrere Zugangsdaten merken zu müssen. Der nachfolgende Artikel befasst sich in erster Linie mit der generellen Frage nach den Vor- und Nachteilen von Identity Management im World Wide Web, dessen Verwendungszweck sowie der genauen Funktionsweise, aber auch mit dem zukünftigen Potenzial. Exemplarisch wird hier einer der momentan bekanntesten Vertreter von Identity Management Systemen, OpenID, betrachtet.

II. Identity Managment

„Als Identity Management (IdM) wird der zielgerichtete und bewusste Umgang mit Identität, Anonymität und Pseudoanonymintät bezeichnet.“ [2] Mit der Einleitung des Internets in das sogenannte Zeitalter des Web 2.0 hat auch die Fragestellung nach der bewussten und unbewussten Preisgabe von privaten Daten und Informationen einen bisher unbekannten Grad an Komplexität erreicht. Sowohl erfahrene, aber vor allem unerfahrene Nutzer verlieren schnell den Überblick über ihre Identitäten, mit denen sie sich im Internet bewegen. Hier soll IdM Abhilfe schaffen. Ziel ist eine konsistente, authentische und dauerhafte Bereitstellung von personenbezogenen Daten und die damit einhergehende Redundanzfreiheit sowie „eine hinreichende Sicherheit über die Identität der Online-Kommunikationspartner zu bekommen“ [2], ohne gleichzeitig unnötig viele personenbezogene Details austauschen zu müssen. Die manuelle Anmeldung durch den Benutzer soll dabei so selten wie möglich durchgeführt werden.
Mittlerweile gibt es eine Vielzahl an IdM Systemen, welche alle unterschiedliche Eigenschaften und Funktionsumfänge besitzen. Einige von ihnen sind sehr umfangreich und decken zusätzliche Attribute, wie z.B. Zugriffsrechte mit ab, andere wiederum sind etwas schlanker, um das System möglichst einfach und kompatibel zu halten.

III. Single Sign-On

Befasst man sich mit IdM, so ist oft von dem Begriff des Single Sign-On (SSO) die Rede. Das Ziel des SSO ist es, „dass ein Benutzer nach einer einmaligen Authentifizierung“ [3], durch Kennworteingabe o.ä., „auf alle Rechner und Dienste, für die er berechtigt ist, zugreifen kann, ohne sich jedes Mal neu anmelden zu müssen.“ [3] Nach der initialen Identifikation durch den Benutzer übernimmt das SSO-System von nun an die Aufgabe, den Anwender an diversen Diensten und Systemen zu identifizieren. Hierbei darf das SSO System

dem ursprünglichen Authentifizierungsverfahren, in puncto Sicherheit in nichts nachstehen. Bei IdM Systemen handelt es sich folglich um, zum Teil erweiterte, SSO Systeme, in unterschiedlichem Umfang.

A. Vorteile

Die Vorteile von SSO Systemen:

- Die einmalige Authentifizierung zu Beginn spart viel Zeit die der Anwender normalerweise für das Suchen bzw. Eingeben der Anmeldeinformationen verschwendet. Dieser Zeitfaktor fällt mit steigender Zahl der benutzen Websites und Dienste zusehends mehr ins Gewicht. Oft ist es für die Anwender, sollten sie unterwegs sein, gar nicht möglich stets alle benötigten Login-Daten für alle, evtl. auch spontan genutzen, Dienste mit sich zu führen, geschweige denn sich zu merken.
- Durch die Tatsache, dass sich der Benutzer nur noch eines anstatt einer Vielzahl von Passwörtern merken muss, kann dieses dafür, durch Erhöhung der Komplexität, umso sicherer gewählt werden.
- Das Passwort muss nur noch einmal zum SSO-Server übertragen werden und bietet somit weit weniger Angriffmöglichkeiten als die multiple Authentifizierung auf mehreren Websites. Durch diesen einzigen Angriffspunkt und das einmalige Senden von UserID und Passwort werden vor allem *Phishing-Attacken* [3] erschwert und die Administration von Sicherungsmechanismen, wie z.B. SSL, erleichtert.
- Es gibt SSO- und IdM-Systeme, die mit *SmartCards* oder bzw. und anderen kryptographischen *Token* gekoppelt werden, wodurch ein sicheres Authentifizierungsverfahren ermöglicht wird. Voraussetzung ist natürlich, dass man die SmartCard oder das Token nicht verliert.
- Die Benutzerdaten sind nur noch in einem Benutzerkonto bzw. den damit verbundenen Identitäten hinterlegt. Somit ergibt sich durch die zentrale Datenhaltung eine Verbesserung der Konsistenz und eine erleichterte Administration.
- Der Benutzer an sich wird sensibilisiert nicht mehr überall seine Benutzerdaten zu hinterlassen. Durch den Gebrauch des SSO- bzw. IdM-Systems wissen deren Anwender genau auf welcher Website sie ihr, in anderen Fällen womöglich mehrfach genutztes, Passwort eingeben dürfen.

B. Nachteile

Ebeneso ist es auch wichtig sich der Nachteile, die SSO Systeme mit sich bringen, bewusst zu werden:

- Der bereits bei den Vorteilen (III. A.) genannte Punkt der zentralen Datenhaltung und der Einfachheit eine UserID und ein Passwort für alle Websites und Dienste zu verwenden, entpuppt sich bei genauerer Betrachtung als ein „Single Point of Attack“. Hat ein Angreifer erst einmal die Identität eines Benutzers angenommen, so hat er von diesem Moment Zugriff auf all dessen Dienste. Dieser Punkt ist selbstverständlich nur insofern ein Nachteil, als dass der Benutzer jeden seiner Dienste ansonsten mit einem separaten, sicheren Passwort absichern würde und die Authentifizierung am SSO-System nicht mit der erwähnten SmartCard- oder Token-Unterstützung stattfindet.
- Die Entscheidung für eine Authentifizierung der Benutzer durch einen bestimmten SSO- bzw. IdM-Service birgt auch Risiken für die Betreiber der Dienste. Sollten potenzielle Benutzer dem verwendeten System und dessen Sicherheit kein Vertrauen schenken, so könnten diese als Kunden verloren gehen.
- Der bereits mehrfach erwähnte, zentrale Authentifizierungs- und Authorisierungsdienst ist stets abhängig von der „Verfügbarkeit des Single Sign-On Systems“ [3]. Hieraus ergibt sich ein sog. „Single Point of Failure“. Sollte der SSO-Dienst einmal nicht funktionsbereit sein, sei es durch einen Hardwaredefekt, Wartungsarbeiten o.ä., so würde auch der Zugriff auf sämtliche Systeme in dieser Zeit verwehrt bleiben.

IV. OpenID

A. Einführung

OpenID und das zugrundeliegende Protokoll wurden „ursprünglich im Jahr 2005 von Brad Fitzpatrick, dem Chief Architect der Firma Six Apart Ltd.“ [4] und „Gründer von LiveJournal“ [5], entwickelt. Bei dem IdM-System OpenID handelt es sich um ein offenes, dezentral angelegtes [6] Open Source System für digitale Benutzeridentitäten. Es bringt die bereits im vorherigen Abschnitt über SSO-Systeme erläuterten Vor- und Nachteile mit sich und ersetzt klassischen Anmeldeprozess, durch Eingabe von Benutzernamen und Passwort pro verwendetem Dienst.
Der Unterschied zu anderen großen IdM-Systemen wie Shibboleth [7] oder dem Liberty Alliance Project [8] ist, dass es sich bei OpenID um ein wesentlich schlankeres, weniger komplexes, doch dadurch auch funktionsärmeres System handelt, welches zudem einen anderen Ansatz verfolgt.
Unterstützt eine Seite OpenID, so muss hier zur Anmeldung lediglich eine OpenID-Identität in Form einer URL, z.B. benutzer.provider.tld, angegeben werden, welche man bei der Registrierung bei einem OpenID Provider bekommt. Der Authentisierungsprozess wird beim Provider, und nicht mehr beim Betreiber der besuchten Website, durchgeführt. Abbildung 1. zeigt die Weboberfläche mit Anmeldemaske eines der bekanntesten Vertreter, *myOpenID.com*.

Der Benutzer hat die Möglichkeit sich den OpenID Provider auszusuchen, der seinen Ansprüchen, vor allem in puncto Sicherheit und Vertrauen, am nächsten kommt. Besonderheit ist hierbei, dass im Prinzip jeder, bedingt durch die dezentrale und komplett freie Gestaltung, einen OpenID Server betreiben kann und somit zum OpenID Provider für wiederum andere Benutzer werden kann Eine Registrierung oder Zustimmung irgendeiner Organisation ist dafür nicht notwendig. Der Ansatz von OpenID wird auch als *user-centric* [6] bezeichnet. Vertreter dieses Ansatzes stellen den Zugang bereit und lassen Benutzern die Kontrolle darüber wie ihre Identität online verwaltet und verwendet wird. Die Identity Provider, auf denen

Abbildung 1. Startseite des OpenID Providers myOpenID.com

die Benutzerdaten hinterlegt sind, sind dezentralisiert und, im Gegensatz zu den sog. *institution-centric* IdM Systemen, wie Shibboleth, gehören sie keiner Organisation an. Der Unterschied ist also, dass bei Shibboleth eine Institution versichert, dass der Benutzer der ist der er vorgibt zu sein, während dies bei OpenID dem Benutzer überlassen wird. [9] Hierdurch kann sich jeder Benutzer im Prinzip vor kommerziellem *Data Mining* durch Organisationen schützen, indem er einfach seinen eigenen Provider betreibt. In diesem Fall verlassen seine privaten Daten die eigenen vier Wände nicht ohne seine ausdrückliche Zustimmung. [10]

Nach der bisherigen Betrachtung stellt sich die Frage inwiefern sich OpenID von einem normalen SSO-System unterscheidet. Die großen Unterschiede sind zum einen die bereits erwähnte dezentrale Gestaltung mit einer großen Menge an Identity Providern und der damit verbundenen Entscheidungsfreiheit wo man seine Daten hinterlegt, zum anderen aber auch die Möglichkeit beim Provider mehrere Angaben, so genannte Attribute, zur Identität zu hinterlegen. Diese Option wird auch als *Attribute Exchange (AX)* bezeichnet. So kann man z.B. seinen vollständigen Namen, seine Email-Adresse, seine bevorzugte Sprache oder seine Geburtsdaten speichern. Diese Daten können wiederum als Identitäts-Profile gespeichert werden. Von nun an wählt der Anwender bei jeder Anfrage des Webservers (bzw. der Webapplikation), auf dem er sich mittels OpenID einloggen möchte, aus, welches Profil der Provider an diesen übermitteln darf. [11] Der Anwender hat somit stets die volle Kontrolle über seine personenbezogenen Daten und weiß zu jeder Zeit welchem Anbieter er welche Informationen mitteilen möchte. Laut OpenID.net befindet sich das Projekt immer noch in der Einführungsphase, erfreut sich jedoch, bedingt durch die Akzeptanz und Verbreitung durch große Organisationen wie Yahoo, „AOL, Microsoft, Sun, Novell, etc.“ [6], immer größerer Popularität. Es wird geschätzt, dass es mittlerweile über 368 Millionen (Stand: Januar 2008) OpenID-URIs gibt, mit knappen Zehntausend Seiten die den Login mittels OpenID unterstützen. [5]

B. Entitäten

Besucht ein Benutzer eine Website und loggt sich mit einer OpenID dort ein, so findet eine Kommunikation zwischen dem Webbrowser und dem Webserver statt. Der Webserver wiederum kommuniziert mit dem Identity Provider, der Benutzer mit dem Identity Provider und später wieder der Benutzer mit dem Webserver. Doch das ist nur ein grober Überblick. Der genaue Protokollablauf wird im Unterpunkt „Funktionsweise“ geschildert. Zuvor gilt es noch einige Begrifflichkeiten zu klären. Abbildung 2. soll hierbei einen Überblick über die Entitäten, die an einem OpenID Anmeldevorgang beteiligt sind, geben und deren Verbindungen untereinander veranschaulichen.

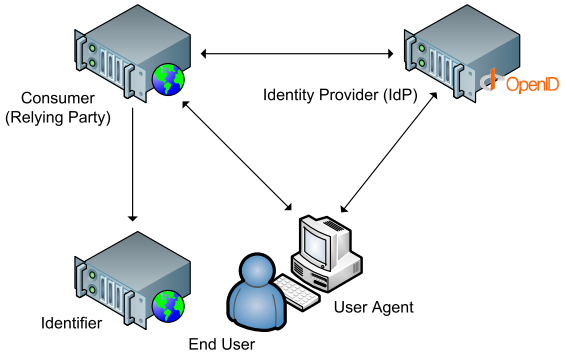

Abbildung 2. Komponenten (Entitäten) die bei einem OpenID-Anmeldevorgang beteiligt sind.

- End User:
  Hierbei handelt es sich um den Benutzer, der sich mit seiner OpenID bei einer Website anmelden möchte.
- Consumer:
  Der Consumer ist die Website bzw. der Webserver an dem sich der Benutzer, mittels OpenID, anmelden will. Damit ist er der eigentliche Konsument der vom Identity Provider gelieferten Daten bzw. des gesamten Dienstes. Folglich sind alle Websites, die eine Anmeldung mittels OpenID gestatten, Consumer. In der OpenID Spezifikation 2.0 [6] wird er auch als Relying Party bezeichnet, da er den vom Identity Provider gelieferten Informationen vertraut.
- Identifier:
  Der Identifier ist eine Website die der End User als OpenID angeben kann. Handelt es sich bei dem Server nicht gleichzeitig um einen Identity Provider, so befindet sich hier ein Verweis auf diesen. Der Consumer erfährt also von dem Identifier an welchen Identitiy Provider er sich bezüglich der Authentifizierung wenden kann.
- Identity Provider (IdP):
  Beim IdP handelt es sich um den Server wo die eigentliche Authentifizierung abläuft. Außerdem sind auf ihm die Benutzerdaten hinterlegt. Die OpenID URI verweist entweder direkt auf diesen IdP oder der Identifier. Während des Authentifizierungsvorgangs tauscht der Consumer Informationen mit dem auch OpenID Provider (OP) ge-

nannten Server aus um eine ID auf ihre Richtigkeit hin zu prüfen.
- User Agent:
  Der User Agent ist der Internet Browser mit dem der End User, durch Tastatur- und Mauseingaben, direkt kommuniziert.

C. Freiheit bei der Wahl der OpenID

Da sich OpenID, anders als diverse andere IdM Systeme, auf Webanwendungen konzentriert, liegt es auf der Hand, dass die Benutzer sich nicht durch klassische Benutzernamen, sondern durch URIs (Uniform Ressource Identificators) bzw. URLs (Uniform Resource Locators) ausweisen. Zum Einsatz kann hier prinzipiell jede beliebige Webadresse kommen, solange ein Browser sie aufrufen darf. [11] Jeder Nutzer einer OpenID muss sich selbstverständlich zuerst bei einem Identity Provider, z.B. myopenid.com, pip.verisignlabs.com oder myid.net (vollständige Liste unter wiki.openid.net/OpenIDServers), anmelden um eine initiale ID zu bekommen. Danach kann er sich jedoch jede beliebige URL als OpenID nehmen die er besitzt. „In dieser Seite muss der Webentwickler lediglich einen *Tag* als Verweis auf den eigentlichen Dienstleister, in den Quellcode der Seite, einbauen." [11] Genaueres dazu im folgenden Abschnitt „Identifier und Identity Provider" (IV. D.). Eine Vielzahl von Benutzern besitzen sogar schon eine OpenID, ohne es zu wissen. Wer bereits einen Benutzeraccount bei einem der in Abbildung 3. aufgelisteten „Webanwendung wie Blogger oder einem Portal wie AOL oder Yahoo hat, darf die dort vorhandenen Authentifizierungsmechanismen nämlich auch über das OpenID-Protokoll nutzen." [11]

Abbildung 3. Identitätsprovider einiger Webanwendungen [6]

D. Identifier und Identity Provider

In der Regel wird sich ein Benutzer bei einem IdP, durch Anmeldung, eine OpenID holen und diese fortan zum Login auf diversen Websites verwenden. Dies ist allerdings nicht verpflichtend. Theoretisch kann jede beliebige URL als OpenID verwendet werden. Ausschlaggebend hierfür sind die Informationen die auf der als OpenID angegebenen Seite, der sog. *OpenID Identity URL Page* [12], vorhanden sind. Hier muss ein Tag im Quellcode der Website einen Verweis auf den eigentlichen IdP enthalten. (Siehe Abbildung 4.)
Durch den ersten Eintrag wird der IdP (hier www.myopenid.com) bekanntgegeben der benutzt werden soll um den Besitzer der OpenID (hier beispielname.myopenid.com) in der zweiten Zeile, zu verifizieren. Duch Einbauen dieser zwei Zeilen in die eigene Homepage oder den eigenen Blog, sind die Anwender also in der Lage ihre persönliche Adresse als OpenID zu verwenden und diese je nach Lust und Laune zu ändern. [11]

```
<link rel=\openid.server\
href=\www.myopenid.com\/>

<link rel=\openid.delegate\
href=\beispielname.myopdenid.com\/>
```

Abbildung 4. HTML Code für die OpenID Identity URL Page

E. Funktionsweise

In dem folgenden Abschnitt wird der genaue Ablauf der einzelnen, am Authentifizierungsvorgang beteiligten, Komponenten veranschaulichen, erklärt und vor allem auf die zwei alternativen Kommunikationsmodi zwischen Website oder Webserver und deren Unterschiede eingegangen. Die nachfolgenden Sequenzdiagramme und Erklärungen sind in Anlehnung an das „OpenID Book" [12] und die OpenID Specification 2.0 [13] entstanden.
Man unterscheidet zwischen zwei grundlegenden Modi, die je nach „Intelligenz" des Consumers bzw. dessen Konfiguration zum Einsatz kommen: dem „Smart mode" und dem „Dumb mode". Ein intelligenter Consumer (Smart Mode) hat anfangs etwas mehr Arbeit und dafür gegen Ende weniger, benötigt hierfür allerdings *lokales Caching* von Statusinformationen. Ein Consumer im Dumb mode hingegen ist komplett zustandslos und benötigt daher einen zusätzlichen *HTTP request*. [14]

1) Smart mode: In diesem Modus vereinbaren der Consumer und der IdP ein sog. *shared secret*. Dabei handelt es sich um einen zeitlich begrenzt gültigen, gemeinsamen Schlüssel, welcher gecached und verwendet wird um die künftigen Kommunikationsnachrichten zu authentifizieren. Dieses shared secret kann entweder im Klartext übertragen oder per Diffie-Hellman-Verfahren zum Austausch von Schlüsseln erzeugt werden. [14] Dadurch wird der HTTP Verkehr zwischen Consumer und IdP am Ende des Authentifikationsprozesses reduziert. Dieser Schritt ist zwar optional, aber nach OpenID Spezifikation 1.1 [14] empfohlen und auch üblich.
Der genaue Ablauf ist im Folgenden und zuerst für den ersten Login (Abbildung 5.) und danach für die darauffolgenden Logins (Abbildung 6.) beschrieben.

a) Erster Login:

1) Der Benutzer besucht die Website des Consumers auf der er sich mittels OpenID einloggen möchte. (z.B. livejournal.com)
2) Er bekommt die Seite vom Consumer geliefert.
3) Nun wählt er bei der Anmeldemaske die Option zur Authentifizierung mittels OpenID auf der Website aus, trägt seine OpenID URL (OpenID Identity) ein (z.B. beispielid.myopenid.com) und startet mit der Übermittlung an den Webserver den Anmeldevorgang.
   Die Hauptaufgabe des Webservers ist nun herauszufinden ob der Benutzer, der sich soeben mit einer ID anmelden will, wirklich die Identität besitzt die er im vorherigen Schritt vorgibt zu besitzen. Der Consumer extrahiert nun aus der OpenID URL die Second-Level-Domain (oder „OpenID Provider Endpoint URL“) des Identifiers und nimmt Kontakt zu diesem auf. Dabei muss es sich nicht unbedingt um den IdP handeln. (Schritt 3a) (Hierfür kann seit der Spezifikation 2.0 [13] auch Yadis [15] zum Einsatz kommen.)
4) Nachdem sich der Consumer die Site des Identifiers geholt hat, wird diese geparst um die Adresse des IdP zu ermitteln. Die Information hierfür ist im Code der HTML Webpage enthalten. Dieser gesamte Prozess wird auch als *Discovery* bezeichnet. Ist der Identifier nicht gleichzeitig der IdP so leitet der Consumer den User Agent, mittels *Browser redirect*, an den richtigen IdP weiter, um die Richtigkeit der vom Benutzer angegebenen Identität einzuholen. Dies geschieht mit der *HTTP GET* Methode. Optional kann der Consumer an diesem Punkt auch eine Verbindung mit dem IdP aufbauen und mit diesem ein shared secret austauschen, welches durch den Austausch von Public Keys, mit Hilfe des Diffie-Hellman-Verfahrens, auf beiden Seiten erstellt werden kann. Der IdP benutzt dieses shared secret um die nachfolgenden Nachrichten zu signieren und der Consumer um diese zu verifizieren. Hierdurch entfällt die Notwendigkeit von direkten Anfragen zur Verifizierung der Signierung nach jedem *Authentifizierungs-request* oder *-response*. [13] (Schritt 4a)
5) Vorausgesetzt der End User ist noch nicht beim IdP eingeloggt, so bekommt er nun dessen Anmeldemaske geschickt. Hier muss er nun das zugehörige Passwort zu seiner OpenID eingeben. Der Authentifizierungsprozess des Benutzers am IdP ist nicht Teil der Spezifikation und bleibt somit dem Provider selbst überlassen. Sollte der Benutzer bereits eingeloggt sein, so entfällt dieser Schritt.
6) Der IdP gibt nun eine Nachricht mit seiner Signatur, wiederum mittels Browser redirect, an den Consumer zurück. Diese gibt Auskunft über die erfolgreiche oder fehlgeschlagene Anmeldung. Zum Einsatz kommt wieder die HTTP GET Methode. Wichtig ist, dass es sich hierbei um eine indirekte Kommunikation zwischen IdP und Consumer handelt.
7) Mit dem shared secret, welches sich im Cache befindet, kann der Consumer nun die signierte Nachricht verifizieren und somit ermitteln, ob der Identity Provider die Nachricht wirklich signiert hat. Handelt es sich bei der Nachricht um eine Mitteilung über eine positive Authentifizierung und konnte der IdP als Absender verifiziert werden, so wird der End User auf der Website des Consumers eingeloggt.

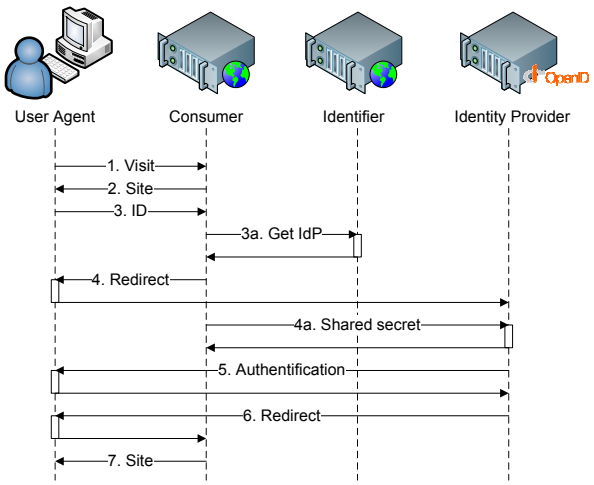

Abbildung 5. Kommunikationsablauf im Smart mode beim ersten Login.

b) Folgende Logins: Hat sich ein Benutzer zuvor schon bei seinem IdP eingeloggt und der Consumer hat das shared secret mit dem IdP bereits in seinem Cache, so wird die Kommunikation der Entitäten untereinander sehr simpel. Der End User kann sich beim Consumer einloggen ohne zuvor interaktiv mit dem IdP kommunizieren, d.h. ohne sich beim IdP nochmals anmelden, zu müssen. Er bekommt dann direkt, nach der Eingabe seiner OpenID URL beim Consumer, sofortigen Zugang zur Website. Hierzu können vom IdP verschiedene Techniken, wie aktive *Browser Sessions* oder *Cookies*, verwendet werden.

1) Der Benutzer besucht die Website des Consumers.
2) Er bekommt die Website mit Anmeldemaske zurück.
3) Der Benutzer gibt seine OpenID URL ein und schickt diese zurück an den Consumer. Dieser leitet daraus wieder die URL des Identifiers ab und nimmt Kontakt zu diesem auf. (Schritt 3a)
4) Nachdem er sich die Seite des Identifiers geholt hat und diese geparst hat, erfolgt die Weiterleitung des User Agents, mittels Browser redirect, an den IdP zur Überprüfung der angegebenen Identität.
5) Der IdP gibt die Information über die erfolgreiche oder fehlgeschlagene Authentifizierung der OpenID mit seiner Signatur, via Browser redirect, an den Consumer zurück.
6) Im Anschluss daran verifiziert der Consumer die Information mittels des im Zwischenspeicher liegenden shared secret.

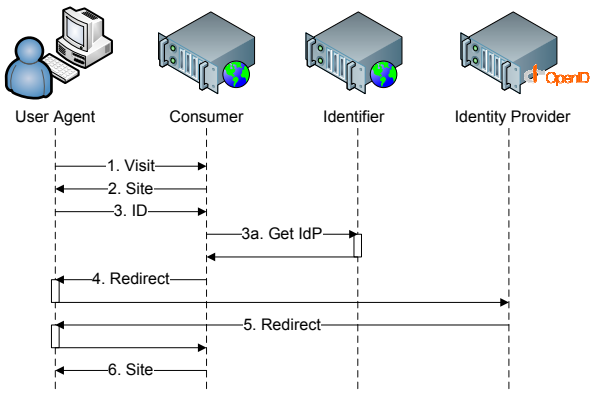

Abbildung 6. Kommunikationsablauf im Smart mode mit bereits bestehendem shared secret.

2) Dumb mode: Dieser Modus kann verwendet werden, wenn der Consumer außerstande ist shared secrets zu erstellen oder diese in einem Cache zu speichern. Daher wird er auch als *Stateless mode* bezeichnet. [13]
Der Consumer muss, anders als beim Smart mode, einen extra Schritt durchführen um den Benutzer zu authentifizieren. Es gibt demnach einen HTTP request und folglich eine response message mehr zwischen Consumer und IdP. Der genaue Ablauf ist im Folgenden beschrieben.

1) Der End User besucht die Website des Consumers.
2) Der End User bekommt die Website mit der Anmeldemaske zurück.
3) Hier hat der User die Möglichkeit seine OpenID URL einzugeben und diese per Submit zu übermitteln. Der Consumer extrahiert die Identifier URL und kontaktiert diesen. (Schritt 3a)
4) Nachdem sich der Consumer die Seite des Identifiers geholt hat, wird diese geparst und der Standort bzw. die IP oder URL des IdP ermittelt. Nach diesem Vorgang leitet der Consumer den User Agent, mittels Browser redirect, zum IdP weiter um die benötigten Informationen über die Gültigkeit der angegebenen OpenID zu bekommen.
5) Nun bekommt der End User die Anmeldemaske des IdPs und muss sich bei diesem eingeloggen.
6) Der Identity Provider schickt dem Consumer nun, wieder via Browser redirect, eine Nachricht über eine erfolgreiche oder fehlgeschlagene Authentifizierung.
7) Sollte die Nachricht erfolgreich sein, so stellt der Consumer nochmals eine direkte Verbindung mit dem IdP, vorzugsweise über SSL, her. Er fragt dann die Authentifizierungsinformationen direkt beim IdP an und vergleicht diese, inklusive der Rückgabe URL, mit der umgeleiteten Information, die er über den User Agent bekommen hat. Mit diesem Verfahren der doppelten Überprüfung soll sichergestellt werden, dass der User Agent (bzw. der End User oder ein potenzieller Angreifer) keine Möglichkeit zum Betrug hat.
8) Stimmen die beiden Informationen aus den vorhergehenden Schritten überein, so wird der End User erfolgreich auf der Website des Consumers eingeloggt.

Der *Dumb mode* sollte ausschließlich verwendet werden, wenn dies der Consumer verlangt bzw. ihm die nötigen Fähigkeiten für den Smart mode fehlen. Je nach Implementierung des OpenID Providers kann dieser Modus ohne Caching verwundbar gegenüber sog. *Replay-Attacken* sein und sollte deshalb, wenn möglich, vermieden werden. [16]

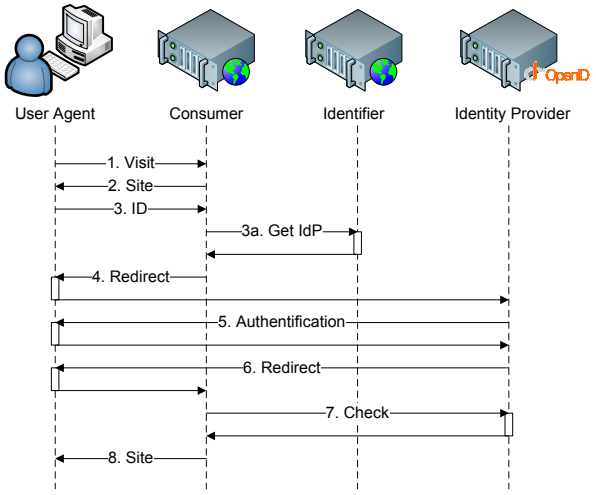

Abbildung 7. Kommunikationsablauf im Dumb mode.

War der Authentifizierungsvorgang erfolgreich, so ist der Benutzer mit seiner OpenID Identität auf der Website angemeldet und kann die gewünschten Dienste nutzen.

F. Implementierung

Wie bereits anfangs erwähnt, steht es jedem interessierten und technisch versierten Anwender frei, seinen eigenen Identity Provider aufzusetzen. Entsprechende Open Source Pakete liegen sowohl für die Server- als auch die Consumer-Variante in nahezu allen gängigen Programmiersprachen vor (z.B. C#, C++, Java, Perl, Python, PHP). Damit steht einer Umsetzung auf einem Großteil der Server nichts mehr im Wege, da diese meist PHP, Perl oder Python als cgi-Scripts unterstützen und fast jeder Client heutzutage Java ausführen kann. [10] Zur Implementierung gibt es z.B. in PHP das komplett fertige Paket „phpMyID“. Die darin enthaltenen PHP-Dateien müssen im Verzeichnis der Homepage abgelegt werden. Wurden diese angepasst, muss der Besitzer der Website die URL zu der PHP-Datei nur noch in die beiden bereits im obigen Abschnitt „Identifier und Identity Provider“ (IV. D.) gezeigten Zeilen im Quellcode der Startseite einzutragen. Danach ist der eigene IdP einsatzfähig. Die Dateien inklusive Installationsanleitung und die genauen Voraussetzungen befinden sich direkt auf der Homepage von phpMyID [17].

G. Sicherheit

Geht es um Authentifizierungsmechnismen so stellt sich auch immer die Frage nach deren Sicherheit. Wenn jeder als

Provider agieren darf, wie sicher kann das Verfahren dann sein? Prinzipiell kann gesagt werden, dass das Verfahren nur so sicher wie die hostende Website bzw. deren Quellcode selbst ist. Damit liegt es in den Händen des Betreibers. Lässt sich die Seite aufgrund unsicherer Skripte o.ä. manipulieren oder fremde Codesegmente importieren, so kann auch hier keine Garantie über die wirkliche Authentizität des sich anmeldenden Benutzers gegeben werden. Denn ist es möglich die Website zu manipulieren, so kann man auch die gesamte Kommunikation zu einem anderen IdP umleiten oder einen solchen betreiben. [11] Die Technologie von OpenID ist demnach gegenüber *Phishing-Attacken* anfällig. Der Grund hierfür ist die Tatsache, dass eine Weiterleitung auf die Seite des OpenID Providers nötig ist. „Als Betreiber einer Seite, welche OpenID zur Anmeldung benutzt, kann man auf einfache Weise eine Weiterleitung auf eine Seite erstellen, welche der Providerseite gleicht, jedoch als Proxy dient und Benutzername und Passwort an den Betreiber weiterleitet. Die OpenID Provider können dies umgehen, indem sie z.B. *Cookies* setzen, die ein individuelles Bild zeigen oder indem sie ein clientseitiges *TLS-Zertifikat* zur Authentifizierung nutzen. Insbesondere letzteres wird von vielen Providern unterstützt.“ [5] Im Internet finden sich Seiten auf denen sich ein solches *Phishing* in Form einer Demo mitverfolgen lässt.
Kann ein Angreifer eine Identität fälschen oder sich gegebenenfalls sogar einer fremden bemächigen, sprich lässt sich die Kommunikation zwischen dem Consumer und dem IdP abhören und die dabei erhaltenen Informationen speichern, so könnte ein Angreifer, versuchen eine der vom Identity Provider an den Consumer gesendeten Bestätigungen über eine erfolgreiche Authentifizierung zu imitieren. Auch das Abfangen einer Nachricht, um diese später als Bestätigung zur Anmeldung mit einem falschen Namen zu verwenden, oder evtl. kritische Benutzerdaten aus der Kommunikation zwischen Consumer und Identity Provider zu gewinnen, wären mögliche Szenarien. Hiergegen ist OpenID jedoch gut gerüstet. Wie bereits zuvor erwähnt „nutzt OpenID TLS, um die Verbindung abzusichern und fügt jeder Abfrage zusätzlich eine Challenge bei“ [11], wodurch jede Antwort jeweils nur einmal Gültigkeit besitzt und nicht wieder verwendet werden kann. [11] Letztendlich sind diese Sicherheitsbedenken zwar zu beachten, jedoch ist nicht OpenID das Problem, sondern eher wie es eingesetzt wird. Ein Customer wird standardmäßig allen IdPs vertrauen mit denen er Kontakt aufnimmt. Dies ist für die meisten Anwendungen die OpenID als Authentifizierungsmethode benutzen durchaus akzeptabel, für Anwendungen mit höheren Sicherheitsanforderungen jedoch nicht. Hierfür lässt sich der Consumer so konfigurieren, dass er nur einigen wenigen, großen und gut abgesicherten IdPs, wie z.B. Verisign, vertraut.

H. Bewertung

Grundsätzlich bleibt zu sagen, dass OpenID mit seinem Ansatz und seiner Funktionsweise in die richtige Richtung geht, da es, bedingt durch seine Funktionen, dem Anwender das Leben erheblich erleichtern kann. Im Endeffekt sind IdM Systeme eine gute Sache, die den Anwendern auch viel ihrer Privatsphäre und vor allem die Kontrolle über die eigenen Daten zurück geben und ihnen einen gewissen Komfort im Umgang mit ihrer Identität im Internet zur Verfügung stellen. Dabei hat der Consumer zwei Möglichkeiten. Entweder er verzichtet auf ein gewisses Maß an Sicherheit, vertraut jedem IdP und gewährt dem Nutzer die größtmögliche Freiheit bei der Wahl seines IdP, oder er legt erhöhten Wert auf Sicherheit, vertraut nur ausgewählten IdP und schränkt den Anwender dadurch wiederum in seiner Freiheit an. Denkbar wäre in diesem Zusammenhang, um eine maximale Sicherheit bei den IdP zu erlangen, dass die Erstellung eines Accounts bei den IdPs, ähnlich einer Eröffnung eines Kontos bei einer Online-Bank, ausschließlich durch erfolgreiche Verifizierung mittels PostIdent möglich ist. In diesem Fall wäre es möglich, dass die IdPs für die Validität der persönlichen Daten Ihrer Kunden bürgen. Es hängt also immer von der Art der Anwendung ab für die OpenID als Authentifizierungsmechanismus dienen soll, inwiefern der eigentliche Grundgedanke von Fitzpatrick umgesetzt werden kann.
Fest steht jedoch leider, dass die Vorteile von OpenID erst richtig zum tragen kommen, sobald eine flächendeckende Lösung gefunden ist. Die Zusammenarbeit vieler unterschiedlicher Unternehmen und besonders technologischen Wegbereitern wie Sun, IBM, Intel uvm. an der Weiterentwicklung der Interoperabilität und die Bemühungen andere Techniken in eigene Lösungen zu integrieren, ist definitiv ein Anfang. Solange dem Benutzer jedoch keine einheitliche Lösung präsentiert werden kann und er sich lediglich zusätzlich zu seinen bisher zu merkenden Logins und Passwörtern auch noch seine OpenID und deren Passwort merken muss, steht OpenID bzw. IdM noch ein weiter Weg bevor. Denn eines steht fest, „am Ende muss der Anwender selbst den Überblick behalten - auch ein Identity Management System kann ihn dabei nur unterstützen.“ [11]

V. Zusammenfassung und Ausblick

Identity Management bringt, wie beschrieben, viele Vor- aber auch einige Nachteile mit sich. So braucht sich der Anwender, bedingt durch die SSO Funktionalität, nicht wie zuvor für jede Website und Webapplikation einen Benutzernamen mit zugehörigem Passwort merken, sondern hat durch die einmalige Authentifizierung beim IdP ein deutlich erhöhtes Maß an Komfort. Zusätzlich hat er durch die Verwendung von Attributen, die er selbst verwalten kann, die Möglichkeit eine genaue Kontrolle darüber zu haben, welchem Anbieter er welche personenbezogenen Daten zur Verfügung stellen möchte und kann dies zentral verwalten.
Betrachtet man die in puncto Sicherheit bereits erwähnten potenziellen Schwachstellen, so sind diese nicht auf die Arbeitsweise oder Technik des OpenID-Protokolls zurückzuführen. Grund sind mangelnde Absicherung oder die falsche Konfiguration durch die Betreiber der Consumer-Websites, aber auch die Unaufmerksamkeit der Benutzer. [10] „Für Anwendungen und Dienste die nur eine vergleichweise wenig komplexe Identitfizierung erfordern, wie beispielsweise Blogs u.ä., erfreut

sich OpenID zwar steigender Beliebtheit. In elektronische Geschäftsprozesse jedoch hat OpenID bisher keinen Einzug gefunden.“ [10]

Das größte Handicap von OpenID, ist die zwar wachsende aber dennoch recht geringe Verbreitung unter den Anbietern und die damit einhergehende noch mäßige Popularität unter den Benutzern. Hinzu kommt ein Problem, welches auf IdM Systeme im Allgemeinen zu beziehen ist, nämlich die mangelnde Interoperabilität der verschiedenen Lösungen untereinander, was auch gerade im Hinblick auf die teils sehr großen Unterscheide im Leistungsspektrum der verschiedenen Lösungen zurückzuführen ist. Wenngleich OpenID immer mehr Unterstützung findet und die Zahl der Anwendungen, die OpenID nutzen, steigt, müssen dennoch „einige richtig große Anwendungen erst noch zeigen, ob sie allen operativen und konzeptionellen Anforderungen an Vertraulichkeit und auch Verfügbarkeit gewachsen sind.“ [11] So gibt es zeitweise immer noch Ausfälle bei OpenID Providern, während derer die Endnutzer ihre virtuellen Identitäten nicht nutzen können.

Die Zukunft wird zeigen ob die unterschiedlichen Ansätze, die von den verschiedenen Vertretern aus der Industrie unterstützt werden, sich dauerhaft etablieren können. Ob OpenID sich nach und nach bei allen Betreibern von Websites mit Authentifizierungsmechanismen durchsetzt und vielleicht auch Einzug in die Unternehmenswelt halten wird, bleibt abzuwarten.

LITERATUR

[1] A. Myllyniemi, “Identity management systems - a comparison of current solutions,” Helsinki University of Technology, Tech. Rep., 2006. [Online]. Available: http://www.tml.tkk.fi/Publications/C/22/papers/Myllyniemi_final.pdf

[2] Wikipedia, “Identitätsmanagement,” 2008. [Online]. Available: http://de.wikipedia.org/wiki/Identitätsmanagement

[3] ——, “Single sign-on.” [Online]. Available: http://de.wikipedia.org/wiki/Single_Sign-On

[4] D. Recordon and D. Reed, “Openid 2.0: A platform for user-centric identity management,” ACM - Association for Computing Machinery, Tech. Rep., 2006. [Online]. Available: http://portal.acm.org/citation.cfm?id=1179532

[5] Wikipedia, “Openid.” [Online]. Available: http://de.wikipedia.org/wiki/OpenID

[6] OpenID. [Online]. Available: http://openid.net

[7] Shibboleth. [Online]. Available: http://shibboleth.internet2.edu/

[8] L. Alliance. [Online]. Available: http://www.projectliberty.org/

[9] V. Smith, “Openid vs. shibboleth: power to the people.” [Online]. Available: http://vsmith.info/OpenID

[10] C. J. Ruwe, “Openid - management/authentifikation digitaler identitäten im web,” Helmut Schmidt Universität - Universität der Bundeswehr Hamburg, Tech. Rep., 2007. [Online]. Available: http://academics.cruwe.de/files/openid.pdf

[11] N. Magnus, “Identitätsverwaltung im web mit open id - offene identität,” *Linux-Magazin*, vol. 07, pp. 50–53, 2008.

[12] R. U. Rehmann, *The OpenID Book - A comprehensive guide to OpenID protocol and running OpenID enabled websites*. Conformatix Technologies Inc., 2008. [Online]. Available: http://openidbook.com/

[13] OpenID.net, “Openid authentication 2.0 - final,” 2007. [Online]. Available: http://openid.net/specs/openid-authentication-2_0.html

[14] D. Recordon and B. Fitzpatrick, “Openid authentication 1.1,” 2006. [Online]. Available: http://openid.net/specs/openid-authentication-1_1.html

[15] Wikipedia, “Yadis.” [Online]. Available: http://de.wikipedia.org/wiki/Yadis

[16] O. Wiki. [Online]. Available: http://wiki.openid.net

[17] phpMyID. [Online]. Available: http://siege.org/projects/phpMyID/