OpenID unter Sicherheitsgesichtspunkten

Tobias Dussa, KIT-CERT

tobias.dussa@kit.edu

Abstract: Der Authentifizierungsstandard OpenID ist im Web-2.0-Umfeld weit verbreitet. Ziel von OpenID ist es, dem Benutzer eine für alle OpenID unterstützenden Webseiten die gleiche Anmeldeprozedur zur Verfügung zu stellen, die zur Authentifizierung auf einen zentralen Dienst zugreift, so dass der Benutzer lediglich ein einziges Passwort verwalten muss. Dieser Beitrag beleuchtet den OpenID-Standard unter Sicherheitsgesichtspunkten und diskutiert insbesondere die Eignung als Authentifizierungsverfahren für andere Anwendungen als die klassischen Web-2.0-Plattformen.

1 Einleitung und Motivation

In den letzten Jahren sind im World Wide Web zahlreiche Dienste entstanden, die als »Web 2.0« zusammengefasst werden. Beispiele hierfür sind Blogs, Wikis, Webmaildienste und Plattformen für soziale Netzwerke. Ihnen gemeinsam ist, dass sie nicht nur statische, für alle Benutzer gleiche Inhalte anbieten, sondern auch individuelle, an den einzelnen Benutzer angepasste Mehrwerte erbringen. Hierfür ist es Voraussetzung, einerseits die verschiedenen Benutzer voneinander unterscheiden, andererseits deren Daten voneinander trennen und den Zugriff darauf kontrollieren zu können. Es ist also nötig, den einzelnen Benutzer zu identifizieren und zu authentifizieren.

Das für Webdienste klassische Verfahren, Benutzer zu authentifizieren, ist die traditionelle Abfrage von Benutzername und zugehörigem Passwort. Benutzer können typischerweise ohne weiteres Zutun des Betreibers eigene Konten anlegen und verwalten.

Während dieser Ansatz zur Authentifizierung für den Betreiber eines Webdienstes vorteilhaft ist, weil er sich leicht implementieren lässt und dem Betreiber die Kontrolle über die Authentifizierungsdaten der Benutzerkonten belässt, bringt er den Benutzer rasch an seine Grenzen. Da ein Benutzerkonto in der Regel nur für einen Webdienst gültig ist, muss der Anwender für jeden Webdienst ein neues Konto einzurichten. Im Idealfall kann dabei derselbe Benutzername verwendet werden, dies ist aber wegen potentiell verschiedener Vorschriften für gültige Benutzernamen einerseits, Kollisionen von Benutzernamen andererseits nicht selbstverständlich. Hinzu kommt, dass für jedes Konto üblicherweise auch ein Passwort gewählt werden muss. Der Benutzer ist damit ohne technische Hilfsmittel vor die Wahl gestellt, entweder ein Passwort für mehrere Konten zu verwenden oder zu verfolgen, welches Passwort für welchen Webdienst gilt. Beides ist wenig wünschenswert, so dass sich in der Praxis verschiedene Hilfsmittel etabliert haben, die dem Benutzer die Verwaltung seiner Zugangsdaten erleichtern. Ein bekanntes Beispiel ist die Passwortablage

des Mozilla Firefox, die für gegebene URLs den gültigen Benutzernamen und das dazugehörige Passwort speichert.[Cot] Solche Hilfsmittel bergen auch Nachteile; geht etwa die Passwortablage verloren, so sind damit auch sämtliche damit verwalteten Zugangsdaten nicht mehr verfügbar.

OpenID versucht, diesen Problemen zu begegnen, indem eine global gültige und eindeutige Benutzerkennung eingeführt wird. Es ist für den Benutzer weiterhin nötig, bei jedem Webdienst ein separates Konto anzulegen; hingegen wird die Authentifizierung auf einen einzigen Webdienst konzentriert. Auf diese Weise können einige oben beschriebene Nachteile umgangen werden:

- Die Vielzahl verschiedener Benutzerkennungen wird im Regelfall auf eine einzige OpenID-Benutzerkennung reduziert.
- Durch die Bündelung der Authentifizierung reduziert sich die Anzahl der Passwörter; der Benutzer muss sich nur ein einziges Passwort merken.
- Die Dienstanbieter haben keinen Zugriff mehr auf Authentifizierungsdaten der Benutzer, wodurch der mögliche Schaden bei Fehlverhalten schrumpft.

Während OpenID aus Benutzersicht also einige Vorteile gegenüber den klassischen Anmeldemethoden bietet, birgt das Verfahren bei genauerer Betrachtung sowie aus Sicht eines Dienstanbieters auch Schwachstellen. Diese Perspektive wird etwa dann relevant, wenn der Zugang zu wertvollen Ressourcen, beispielsweise klassischen Rechenressourcen, geschützt werden soll. Entsprechende Ansätze sind etwa in Cloud-Computing-Projekten zu finden, wenn der eigentliche Zugriff auf die Ressourcen zwar mittels SSH durchgeführt, der Zugriff auf die zentrale Verwaltung von Benutzerkonten aber mit Hilfe von OpenID gesichert wird; als konkretes Beispiel sei das OpenCirrus-Projekt genannt, in dem das beschriebene Vorgehen vorgeschlagen und diskutiert wurde.[DKM09]

Nachfolgend werden zunächst verwandte Arbeiten aufgezeigt, um danach die Funktionsweise des OpenID-Protokolls näher zu erläutern. Im nächsten Abschnitt werden daraus resultierende Schwachstellen von OpenID beschrieben. Im fünften Abschnitt werden die vorgestellten Schwachstellen bewertet; der sechste Abschnitt beinhaltet schließlich eine Zusammenfassung und Möglichkeiten des weiteren Vorgehens.

2 Weitere Arbeiten

Die Sicherheitsaspekte von OpenID werden teilweise in Version 2.0 der Spezifikation des OpenID-Standards zur Authentifizierung diskutiert.[FRHH07] Schwerpunkt dieser Diskussion sind Sicherheitsprobleme aus Sicht des Anwenders. Einige Gesichtspunkte werden weiterhin in der Spezifikation der OpenID-Erweiterung “OpenID Provider Authentication Policy Extension 1.0” betrachtet.[RJS08] Zusätzlich stellen die Entwickler von OpenID eine Wikiseite mit Empfehlungen für den sicheren Einsatz bereit.[ART09]

Schließlich stellen Eugene und Vlad Tsyrklevich in ihrem Vortrag “Single Sign-On for the Internet: A Security Story” sowie ihrem Whitepaper weitere Probleme vor.[TT07a, TT07b]

3 Funktionsweise des OpenID-Protokolls

In diesem Abschnitt wird der Ablauf einer Authentifizierung nach OpenID-Standard Version 2.0 zunächst kurz umrissen, dann etwas detaillierter dargestellt.[FRHH07]

Beim OpenID-Authentifizierungsprotokoll sind mehrere Parteien beteiligt:

- Der Benutzer (Alice), der sich bei einem Webdienst anmelden möchte.
- Der Webdienst (Bob), bei dem sich der Benutzer anmelden möchte. Im OpenID-Jargon wird dieser Webdienst auch als "Relying Party" oder "RP" bezeichnet.
- Den OpenID Provider (Trent), dem gegenüber sich der Benutzer authentifiziert. In OpenID-Terminologie wird der OpenID Provider auch als "OP" bezeichnet.

Eine Authentifizierung nach OpenID läuft im wesentlichen in den folgenden Schritten ab:

1. Alice gibt gegenüber Bob einen "User-Supplied Identifier" an, vergleichbar mit einem Accountnamen. Dieser Identifier ist im wesentlichen ein URL oder ein XRI (Extensible Resource Identifier, siehe [RM05]).
2. Bob führt den sogenannten "Discovery Process" mit dem im übergebenen Identifier durch. Dadurch wird der URL des OpenID-Providers, also Trent, ermittelt.
3. Optional führt Bob mit Trent eine sogenannte Association mit Schlüsselaustausch durch, um die weitere Kommunikation zwischen Bob und Trent abzusichern.
4. Bob leitet Alice mittels eines HTTP-Redirects oder eines HTTP-Posts zu Trent um; dieser Schritt wird als "Authentication Request" bezeichnet.
5. Alice authentifiziert sich gegenüber Trent.
6. Trent leitet Alice mit einem weiteren HTTP-Redirect oder -Post zurück zu Bob und übergibt darin Informationen über Alices Authenfizierungszustand an Bob.
7. Bob prüft die übergebenen Informationen zum Authentifizierungszustand von Alice.

Bei einigen Protokollschritten sind nähere Details für die spätere Diskussion wichtig. Aus diesem Grund werden im Folgenden die einzelnen Schritte näher beschrieben.

3.1 Die Identifizierung des Benutzers

Die Authentifizierung wird vom Benutzer, also Alice, initiiert. Alice identifiziert sich gegenüber Bob, gibt also eine OpenID-Identität an, die sie zu besitzen behauptet. Dies kann etwa wie bei klassischen Mechanismen durch ein einfaches Webformular geschehen.

3.2 Der Discovery Process

Beim Discovery Process ermittelt Bob mit Hilfe des User-Supplied Identifiers von Alice den für sie zustandigen OpenIP-Provider, also Trent. Identifier können als URL oder als XRI angegeben werden; ihre Verarbeitung hängt vom verwendeten Format ab.

Handelt es sich beim User-Supplied Identifier um einen XRI, so wird dieser nach den Umwandlungsregeln in [WRC+06] in eine Extensible Resource Descriptor Sequence (XRDS, ebenfalls in [WRC+06]) umgewandelt, die direkt die benötigten Informationen enthält.

Handelt es sich dagegen um einen URL, so soll erst mittels Yadis-Protokoll (siehe [Mil06]) versucht werden, den URL in eine XRDS umzuwandeln. Gelingt dies nicht, so wird der URL heruntergeladen; danach soll ein HTML-basierter Discovery-Prozess durchgeführt werden. Gemäß OpenID-Spezifikation muss der HTML-basierte Discovery-Prozess von der Relying Party, also Bob, unterstützt werden.

Gelingt die Discovery, so weiß Bob, welcher OP, also Trent, für Alice zuständig ist.

3.3 Die Association

Zwischen Bob und Trent kann eine sogenannte Association durchgeführt werden. Während der Assoziierung wird mit einem Diffie-Hellman-Schlüsselaustauschs ein gemeinsames Geheimnis etabliert, mit dem die weitere Kommunikation zwischen diesen Partnern abgesichert werden kann.[Res99]

Gelingt eine Assoziierung nicht, beispielsweise weil Trent dies nicht unterstützt, so kann Bob im Protokoll fortfahren oder die Authentifizierung abbrechen.

3.4 Der Authentication Request

In diesem Protokollschritt beantragt Bob bei Trent die Authentifizierung von Alice. Dieser Antrag wird in Form indirekter Kommunikation gestellt: Bob schickt einen HTTP-Redirect oder einen HTTP-Post an Alice, der Alices Anwendung – etwa ein Webbrowser – an den ermittelten OP, also Trent, weiterleitet. Alice' Client initiiert daraufhin die Authentifizierung gegenüber Trent. In der Weiterleitung werden von Bob einige Parameter der beantragten Authentifizierung an Trent übergeben.

3.5 Die Authentifizierung

Alice authentifiziert sich gegenüber Trent. Der OpenID-Standard spezifiziert diesen Schritt nicht näher, da die Authentifizierung in Trents Verantwortungsbereich fällt.

Falls in der Authentifizierungsanfrage eine Assoziierung zwischen Bob und Trent referen-

ziert wurde, so soll Trent speichern, dass für diese Assoziierung eine Authentifizierung durchgeführt wurde, um ein Wiederverwenden der Assoziierung zu verhindern. Wurde keine Assoziierung übergeben, so muss Trent eine »einseitige Assoziierung« generieren und zum Zwecke der späteren Verifikation speichern.

3.6 Die Authentication Response

Nach erfolgter Authentifizierung leitet Trent Alice' Browser wieder mittels eines HTTP-Redirects oder -Posts zurück zu Bob. Analog zum Authentifizierungsantrag werden auch hier wieder einige Details zur Authentifizierung übertragen; insbesondere müssen eine eindeutige Transaktionsnummer ("Nonce"), die verwendete Assoziierung sowie eine digitale Signatur angegeben sein.

3.7 Die Prüfung der Authentifizierung

Nachdem Alice wieder zurück zu Bob verwiesen wurde, verifiziert Bob die von Trent indirekt übergebene Nachricht. Einige Angaben müssen gemäß OpenID-Standard geprüft werden, unter anderem der verwendete URL der Weiterleitung von Trent zu Bob, die enthaltenen Angaben zur Identität von Alice, Trents Transaktionsnummer sowie die Signatur.

Die Angaben zur Identität von Alice können nur insofern geprüft werden, als dass sie zu Alice' Angaben im ersten Protokollschritt passen. Trents Transaktionsnummer wird dahingehend geprüft, dass sie nicht für eine vorherige Authentifizierung verwendet wurde.

Zur Prüfung der Signatur ist eine Fallunterscheidung nötig. Liegt eine Assoziierung zwischen Bob und Trent vor, so besteht ein gemeinsames Geheimnis, mit dem Bob direkt die Signatur verifizieren kann. Wurde keine Assoziierung durchgeführt, kann Bob die Signatur nicht überprüfen, da ihm der verwendete Signaturschlüssel nicht bekannt ist. In diesem Fall muss Bob bei Trent prüfen, ob die angegebene Signatur gültig ist. In der Anfrage schickt Bob alle signierten Daten sowie die Signatur selber an Trent, der dann entscheidet, ob es sich um eine legitime Signatur handelt; aus diesem Grund muss Trent im Authentifizierungsschritt die einseitig generierte Assoziierung speichern (siehe Abschnitt 3.5).

4 Beschreibung erkannter Schwachstellen

Das OpenID-Authentifizierungsprotokoll weist einige Angriffspunkte auf, die im Folgenden beschrieben werden.

4.1 Klassische Netzwerkangriffe

Dieser Abschnitt geht auf einige klassische Angriffe auf Kommunikationsprotokolle: Mitlesen und Verändern von Inhalten, Man-in-the-Middle- und Replay-Attacken.

Die OpenID-Authentifizierung ist zunächst nicht besonders gegen Mitlesen gesichert. Damit sind zwar keine Authentifizierungsdaten von Benutzern gefährdet – die eigentliche Authentifizierung erfolgt außerhalb der OpenID-Spezifikation gegenüber Trent –, es kann aber durchaus festgestellt werden, welcher Benutzer sich wann gegenüber welchen Webdiensten authentifiziert.

Replay-Attacken, in denen der Angreifer durch erneutes Versenden mitgeschnittener Pakete Vorteile erlangt, sind zwar durch die Verwendung von Nonces in den Nachrichten von Trent an Bob ausgeschlossen, da damit sichergestellt wird, dass die Meldung einer erfolgreichen Authentifizierung nur ein Mal verwendet werden kann. Dies setzt allerdings voraus, dass Alice als erste Bob gegenüber Trents Bestätigung ihrer Authentifizierung präsentiert. Ist ein Angreifer in der Lage, Trents Antwort mitzulesen und schneller zu Bob weiterzuleiten als Alice, so kann er Alices Authentifizierung übernehmen.

Gegen das Verändern von Inhalten bietet OpenID nur teilweise Schutz; insbesondere bei der Rückmeldung von Trent an Bob sind die wesentlichen Teile der Nachricht digital signiert und können nicht ohne weiteres unbemerkt verändert werden. Es ist bis zu diesem Protokollschritt aber möglich, Inhalte unbemerkt zu modifizieren; damit kann sich ein Angreifer als Trent ausgeben, wenn er geschickt die relevanten URLs verändert. Auf diese Weise können etwa Alices Authentifizierungsdaten ausgespäht werden.

Auch Man-in-the-Middle-Attacken sind möglich. Ein Angreifer, der sich in Protokollschritt 3, der Assoziierung, zwischen Bob und Trent setzen kann, ist in der Lage, den Diffie-Hellman-Schlüsselaustausch zu unterlaufen. Er kann danach gegenüber Bob als Alice aufzutreten, indem er korrekt signierte Antwortnachrichten herstellt, und so als Alice auftreten, ohne ihre Authentifizierungsdaten kennen zu müssen.

4.2 Besondere Risiken für Benutzer

Zusätzlich zu den Risiken, die von klassischen Angriffen ausgehen, ist OpenID noch gegenüber anderen Gefahren verwundbar.

Ziel des OpenID-Standards ist es, dem Benutzer die Verwaltung einer Vielzahl verschiedener Benutzerkonten zu ersparen, indem ein einziger OpenID-Identifier bei allen teilnehmenden Webdiensten zur Identifizierung und Authentifizierung verwendet wird. Die Authentifizierung erfolgt gegenüber einem einzigen OpenID-Provider, was dem Benutzer das Leben erleichtert, aber auch Risiken birgt.

Sämtliche Anmeldevorgänge eines Benutzers gegenüber Webdiensten werden an einen zentralen OP weitergeleitet. Der OP ist damit in der Lage, das Benutzerverhalten zu protokollieren und zu profilieren. Da Anmeldungen je nach Implementierung seitens des OP auch ohne Benutzerinteraktion erfolgen können – etwa mit Hilfe von zeitlich begrenzt

gültigen Cookies –, kann unbemerkt ein detailliertes Bild der aufgerufenen Webdienste erstellt werden.

Ist eine Single-Sign-On-Funktionalität wie eben umrissen implementiert, so eröffnen sich einem Angreifer noch schwerwiegendere Angriffsmöglichkeiten. Hat ein Benutzer eine erfolgreiche Authentifizierung durchlaufen, so kann ein Angreifer vom Benutzer nicht beabsichtigte Aktionen bei Webdiensten auslösen, indem er den Webbrowser des Benutzer dazu bringt, einen entsprechenden URL zu laden.

Zudem spricht der Benutzer durch die Verwendung von OpenID dem OP implizit besonderes Vertrauen aus, denn der OP ist jederzeit in der Lage, sich gegenüber einem OpenID-Dienst als Benutzer auszugeben. Dies impliziert eine besondere Verantwortung des OP im Sinne eines sicheren Betriebs, aber auch die Erwartung eines besonders korrekten eigenen Handelns.

4.3 Besondere Risiken für Dienstanbieter

Im vorigen Abschnitt wurden die Gefahren aus Nutzersicht betrachtet. Aus Sicht eines Dienstbetreibers ergeben sich weitere Schwierigkeiten.

Für den einzelnen Benutzer stellt der für ihn zuständige OP eine Authentifizierungsbündelung dar. Der OP ist die einzige Stelle, der gegenüber der Benutzer sich authentifiziert. Im Extremfall kann der Benutzer sogar seinen eigenen OP betreiben so dass er die vollständige Kontrolle über seine Anmeldedaten behält.

Für einen Dienstanbieter – Bob – ist dieses Konzept keine Zentralisierung, sondern eine Dezentralisierung. Bob ist zur Authentifizierung auf den vom Benutzer angegebenen OP angewiesen. Insbesondere weiß Bob nichts darüber, welche Güte die Authentifizierung eines gegebenen OPs hat. Folglich muss Bob im Zweifel davon ausgehen, dass eine OpenID-Authentifizierung praktisch wertlos ist.

Zusätzlich birgt das OpenID-Protokoll noch grundsätzlichere Probleme für den Dienstanbieter. Gibt Alice einen Identifier in URL-Form an, so ist Bob gezwungen, diesen URL herunterzuladen, solange er mit `http://` oder `https://` beginnt – in Version 1.1 des OpenID-Standards gab es gar keine Einschränkungen, so dass etwa URLs der Form `file://` gültig[RF06] waren. Damit sind Denial-of-Service-Angriffe trivial durchführbar, indem ein Angreifer einen URL angibt, der eine große Datenmenge referenziert. Auch Portscans anderer Maschinen sind denkbar; diese Scans gehen dann von Bob aus und führen nicht unmittelbar zum eigentlichen Angreifer.

Schließlich ist zu bemerken, dass der Alice sich gegenüber Bob authentifiziert, indem sie auf einen OP verweist, der Bob bestätigt, dass Alice tatsächlich Alice ist. Dieser OP steht in keinem definierten vorherigen Vertrauensverhältnis zu Bob und kann beliebige Informationen zurückliefern.

5 Bewertung der Schwachstellen

Die Tragweite der im obigen Abschnitt diskutierten Schwachstellen ist abhängig von den Umständen, unter denen OpenID eingesetzt werden soll.

Die in Abschnitt 4.1 beschriebenen Angriffe sind nicht spezifisch für OpenID. Ihnen kann in der Regel leicht und wirkungsvoll etwa durch den Einsatz von SSL begegnet werden. Diesem Umstand wird auch im OpenID-Standard Rechnung getragen, indem empfohlen wird, SSL einzusetzen. Dies ist insofern bemerkenswert, als dass damit der Diffie-Hellman-Schlüsselaustausch obsolet wird, da er dann nichts zur Sicherheit beiträgt.

Wird auf die Absicherung mittels SSL verzichtet, so ist praktisch der gesamte Authentifizierungsvorgang nicht nur gegenüber Mitlesen, sondern auch gegenüber einem breiten Spektrum an netzwerkbasierten Angriffen verwundbar. Ein Angreifer kann beispielsweise die Identität eines Opfers annehmen, indem er eine erfolgreiche Authentifizierung abfängt und selber verwendet, oder die Authentifizierungsdaten eines Opfers ausspähen, indem er die Weiterleitungen zum OpenID-Provider abfängt, das Opfer zu sich selber weiterleitet und vorgibt, der fragliche OP zu sein. Es ist daher praktisch unerlässlich, durchgängig SSL zur Sicherung der Übertragung einzusetzen.

Die in Abschnitt 4.2 diskutierten Risiken für den Anwender sind teilweise erheblich, hängen aber davon ab, in welchem Umfang OpenID eingesetzt wird. Je mehr ein Benutzer von OpenID Gebrauch macht, desto genauer wird das Profil, das der OP erstellen kann.

Aus Betreibersicht stellt sich der Einsatz von OpenID gerade im Hochschulumfeld je nach genauem Einsatzzweck als problematisch heraus. Unkritisch ist die Verwendung nur dann, wenn die damit geschützten Inhalte keinen besonderen Wert für den Betreiber einerseits, aber auch keine besonderen persönlichen Daten der Benutzer andererseits umfassen. Im allgemeinen besteht kein Anlass für den Betreiber, einem beliebigen OpenID-Provider besonderes Vertrauen entgegenzubringen. Eine Authentifizierung ist daher ohne weitere Rahmenbedingungen nicht aussagekräftig, zumal im Zweifel ein Benutzer seinen eigenen OP betreiben kann. Der Zugriff auf wertvolle Ressourcen, etwa einen Höchstleistungsrechner, sollte daher nicht ohne weiteres auf der Authentifizierung mittels OpenID beruhen. Allerdings kann hier Abhilfe geschaffen werden, indem nicht beliebige OPs, sondern nur tatsächlich vertrauenswürdige OPs – etwa alle von Hochschulen betriebenen OPs – akzeptiert werden. Allerdings hebelt diese Einschränkung das Ziel von OpenID, dass jeder Anwender nur eine einzige Identität benötigt, teilweise aus. Hier ist die 2008 spezifizierte “OpenID Provider Authentication Policy Extension 1.0” bemerkenswert.[RJS08] Diese Spezifikation soll es ermöglichen, die Güte einer Authentifizierung zu messen, indem der OP dem Dienstanbieter Informationen über seine Authentifizierungsverfahren zur Verfügung stellt. Dies ändert aber offensichtlich nichts am grundsätzlichen Problem, dass der Dienstanbieter den Aussagen des OP blind glauben muss. Hier handelt es sich um eine konzeptionelle Schwäche von OpenID, die nicht ohne weiteres behoben werden kann.

Noch eindeutiger wird die Lage, wenn sensible Bereiche persönlicher Informationen berührt werden. Dies ist beispielsweise dann der Fall, wenn der Zugang zu Studienportalen mittels OpenID geregelt würde. In derartigen Portalen können Studierende typischerweise Einblick in ihre Studienleistungen nehmen, sich zu Prüfungen an- und abmelden oder sich

sogar zurückmelden oder exmatrikulieren. Wird hier ohne weitere Einschränkung OpenID zur Authentifizierung verwendet, würde jedes der oben beschriebenen Sicherheitsprobleme dazu führen, dass unberechtigt Zugriff auf diese Daten genommen werden könnte. Die Hochschule hat hier schon aus rechtlichen Gründen für einen hinreichenden Schutz derartig sensibler Daten zu sorgen, der mit OpenID sicher nicht ohne weiteres gegeben ist.

Für Anwendungen, die nicht derartig hohe Bedürfnisse an die Sicherheit haben, ist OpenID dagegen gut zur Authentifizierung geeignet. Dieser Umstand spiegelt das Umfeld wider, für das OpenID ursprünglich konzipiert war: OpenID soll einen Ersatz für die klassische Benutzername-Passwort-Authentifizierung für Web-2.0-Dienste bieten. Diese Dienste haben in der Regel sehr geringe Sicherheitsanforderungen. Ein Benutzerkonto kann häufig durch den Benutzer selber angelegt und aktiviert werden; die Angaben des Benutzers werden in der Regel nicht oder nur oberflächlich geprüft. Gemessen an diesen Sicherheitsstandards ist OpenID in der Tat eine attraktive Alternative, weil sie dem Benutzer den Umgang mit einer Vielzahl von Web-2.0-Diensten erleichtert, ohne wesentliche Einbußen in der Sicherheit mit sich zu bringen.

Für Dienstbetreiber mit höheren Sicherheitsanforderungen ist OpenID nur dann eine Option, wenn es einerseits zwingend mit SSL abgesichert wird, um den klassischen Netzwerkangriffen wirkungsvoll entgegenzutreten, und andererseits nur als vertrauenswürdig bekannte OpenID-Provider zugelassen werden, da ansonsten die Authentifizierung praktisch wertlos ist. Da aber gerade die Freiheit der Wahl des OpenID-Providers ein zentrales Merkmal von OpenID ist, scheint es damit fraglich, ob nicht die Verwendung anderer Verfahren, etwa Shibboleth, einen ähnlichen Komfortgewinn für den Benutzer bringen würde, ohne dieselben Schwachstellen wie OpenID aufzuweisen.

6 Zusammenfassung und Ausblick

In diesem Artikel wurde das OpenID-Authentifizierungsverfahren vorgestellt und dessen Sicherheitsschwachstellen diskutiert. Neben klassischen Angriffen, denen vergleichsweise einfach durch den Einsatz von SSL abgeholfen werden kann, birgt OpenID aufgrund seiner Architektur weitere Schwachstellen. Die Bündelung aller Authentifizierungsvorgänge eines Benutzers bei einem OpenID-Providers gewährt diesem Zugriff auf eine Vielzahl sensibler Daten. Ist zudem eine Single-Sign-On-Funktionalität implementiert, so kann durch einen Angriff auf den Browser eines Anwenders Zugriff auf sämtliche OpenID unterstützenden Webdienste erlangt werden.

OpenID bietet für den Anwender die Chance, die Authentifizierung zu vereinheitlichen und in der Handhabung zu vereinfachen, aber auch sicherer zu gestalten. Aus Betreibersicht müssen zusätzliche Maßnahmen zur Absicherung in Betracht gezogen werden, so dass der Einsatz von OpenID möglicherweise nicht sinnvoll ist. Zudem muss berücksichtigt werden, dass einige Schwachstellen protokollbedingt nicht behebbar sind.

Für die weitere Entwicklung erscheinen zwei Bereiche besonders interessant. Derzeit sind gerade aus Sicht von Dienstanbietern mit höheren Sicherheitsanforderungen noch einige Probleme offen; hier ist mit der “OpenID Provider Authentication Policy Extension 1.0”

bereits ein erster Schritt getan, der weiter ausgebaut werden könnte.

Zweitens bietet OpenID Angriffspunkte, die es einem Angreifer erlauben, einen OpenID unterstützenden Dienstanbieter in begrenztem Maße als Proxy für Attacken auf weitere Systeme zu missbrauchen oder sehr leicht eine Denial-of-Service-Attacke gegen den Webdienst selbst auszuführen. Hier liegt weiteres Entwicklungspotential.

Literatur

[ART09] Andrew Arnott, David Recordon und Allen Tom. OpenID Security Best Practices. Seite im OpenID-Wiki, Juni 2009. http://wiki.openid.net/OpenID-Security-Best-Practices, aufgerufen am 18. November 2009, letze Änderung am 8. Juni 2009.

[Cot] Sean Cotter. PSM 2.1 & Privacy Help: Status and Work in Progress. Webseite. http://www.mozilla.org/projects/security/pki/psm/help_21.

[DKM09] Tobias Dussa, Michael Kozouch und Dejan Milojicic. Open Cirrus Global Sign-On. Technical report, OpenCirrus Project, Mai 2009.

[FRHH07] Brad Fitzpatrick, David Recordon, Dick Hardt und Josh Hoyt. OpenID Authentication 2.0 — Final. Spezifikation, Dezember 2007. http://openid.net/specs/openid-authentication-2_0.txt.

[Mil06] Joaquin Miller. Yadis Specification Version 1.0. Spezifikation, März 2006. http://yadis.org/papers/yadis-v1.0.pdf.

[Res99] Eric Rescorla. Diffie-Hellman Key Agreement Method. Request for Comments, Juni 1999. http://www.faqs.org/rfcs/rfc2631.txt.

[RF06] David Recordon und Brad Fitzpatrick. OpenID Authentication 1.1. Spezifikation, Mai 2006. http://openid.net/specs/openid-authentication-1_1.txt.

[RJS08] David Recordon, Michael B. Jones und Nat Sakimura. OpenID Provider Authentication Policy Extension 1.0. Spezifikation, Dezember 2008. http://openid.net/specs/openid-provider-authentication-policy-extension-1_0.txt.

[RM05] Drummond Reed und Dave McAlpin. Extensible Resource Identifier (XRI) Syntax V2.0. Spezifikation, November 2005. http://www.oasis-open.org/committees/download.php/15376.

[TT07a] Eugene Tsyrklevich und Vlad Tsyrklevich. Single Sign-On for the Internet: A Security Story. Vortrag auf der Konferenz “Black Hat Las Vegas 2007”, August 2007. http://www.blackhat.com/presentations/bh-usa-07/Tsyrklevich/Presentation/bh-usa-07-tsyrklevich.pdf.

[TT07b] Eugene Tsyrklevich und Vlad Tsyrklevich. Single Sign-On for the Internet: A Security Story. Whitepaper zum Vortrag auf der Konferenz “Black Hat Las Vegas 2007”, August 2007. https://www.blackhat.com/presentations/bh-usa-07/Tsyrklevich/Whitepaper/bh-usa-07-tsyrklevich-WP.pdf.

[WRC+06] Gabe Wachob, Drummond Reed, Les Chasen, William Tan und Steve Churchill. Extensible Resource Identifier (XRI) Resolution V2.0. Arbeitsentwurf 10, März 2006. http://www.oasis-open.org/committees/download.php/17293.