Community-Unterstützung für Pseudo-SSO-Systeme

Jan Zibuschka

Fraunhofer IAO
Nobelstraße 12
70569 Stuttgart
jan.zibuschka@iao.fraunhofer.de

Abstract: Für Benutzer ist es im Allgemeinen nicht möglich, für jeden Dienst ein unabhängiges, starkes Passwort zu wählen. Diese Situation legt es nahe, den Nutzern ein Single Sign-on (SSO) anzubieten, das es erlaubt, sich mit nur noch einem Passwort an mehreren Diensten anzumelden. Pseudo-SSO-Systeme müssen nicht vom Dienstanbieter unterstütz werden, und sind daher für den Nutzer sofort nützlich, bei ihrer Realisierung gibt es aber sowohl im Bereich Sicherheit, als auch bei der Benutzerführung noch Hindernisse. In diesem Beitrag werden Ansätze zur Community-Unterstützung von Pseudo-SSO-Systemen vorgestellt.

1 Motivation

Das sogenannte *strong password dilemma* [1] ist in der Sicherheit seit Jahren geläufig. Es ist für Nutzer im Allgemeinen nicht möglich, für jeden genutzten Dienst im Web ein unabhängiges, hinreichend langes und zufälliges Passwort zu wählen, ohne ein erhöhtes Risiko einzugehen, dieses Passwort später nicht aus dem Gedächtnis abrufen zu können [2]. Die klassische Kognitionspsychologie legt nahe, dass Nutzer nicht fähig sind, deutlich mehr als sieben voneinander unabhängige Passwörter im Gedächtnis zu behalten [3]. Aktuelle Studien des Nutzerverhaltens im Web bestätigen diese Vermutung [4], zeigen aber gleichzeitig, dass Nutzer diese Passwörter nutzen, um sich an durchschnittlich 25 Diensten zu authentisieren.

Diese Situation legt es nahe, den Nutzern ein *Reduced Sign-on* [5] bzw. *Single Sign-on* (SSO) anzubieten, das es erlaubt, sich mit nur noch einem oder zumindest wenigen Passwörtern an mehreren Diensten anzumelden. Hier hat es verschiedene Ansätze zur Realisierung gegeben, in den letzten Jahren etwa Browser-Plugins wie LastPass [6] oder föderierte Identitätsmanagementsysteme (FIM) wie OpenID [7] oder CardSpace [8].

erschienen im Tagungsband der INFORMATIK 2011
Lecture Notes in Informatics, Band P192
ISBN 978-3-88579-286-4
weitere Artikel online:
http://informatik2011.de/519.html

Diese Systeme haben sich jedoch oftmals nicht so weit verbreitet wie erwartet. Da mit einem globalen SSO-Protokoll ein erheblicher Standardisierungsaufwand verbunden ist, haben einige Forscher sogar Bedenken angemeldet, ob SSO überhaupt realistisch ist [5]. In den letzten Jahren hat aber insbesondere Facebook Connect einen mehr als beachtlichen Adoptionserfolg erzielt [9]. Aus der Perspektive des Datenschutzes ist es jedoch beunruhigend zu beobachten, dass sich in den letzten Jahren insbesondere vernetzte FIM Systeme umso besser durchsetzen, je datenschutzunfreundlicher sie sind [10]. Als eine Lösung, die dieses Standardisierungsproblem bis zu einem gewissen Grad umgeht, da sie die vorhandenen Authentisierungsschnittstellen unterstützen [11] wurden *Pseudo-SSO* Systeme [12] vorgeschlagen, die aufgrund ihrer Architektur Adapter für existierende Mechanismen bereit stellen können. Im Web sind die existierenden Schnittstellen üblicherweise passwortbasiert.

Pseudo-SSO bringt aber das Problem mit sich, wie Passwörter auf allen Systemen synchron verfügbar gehalten werden können, besonders da dies u.U. auch auf noch nie genutzten Systeme anwendbar sein muss. Auch für dieses Problem wurden mehrere Lösungsansätze vorgestellt, insbesondere *password synching*, die Synchronisation von Passwortspeichern auf mehreren Systemen [6], und *password hashing*, die spontane Erzeugung dienstespezifischer Passwörter aus einem *master password* (MPW) [13], vorgeschlagen. Beide Ansätze haben Vor- und Nachteile.

Das password hashing hat Probleme, dienstespezifischen Passwortrichtlinien zu adressieren. Die von Web-Diensten verwendeten Passwortrichtlinien sind alles andere als einheitlich, und die Einschränkungen erscheinen oftmals willkürlich [14]. Daher können die aus hash-Werten generierten dienstspezifischen Passwörter oftmals die Passwortrichtlinien nicht adressieren [15], was einen manuellen Eingriff des Nutzers, oder aber eine Unterstützung durch einen zentralen Server, auf dem Passwortrichtlinien der Web-Dienste hinterlegt sind (z.B. für jeden Nutzer wie etwa im Fall von [16]), notwendig macht. Dieses Problem soll in diesem Beitrag, ähnlich wie in [17] vorgeschlagen, mittels Crowdsourcing adressiert werden.

Beim password syncing gibt es Bedenken, weil der Dienstanbieter in naiven Implementierungen Kenntnis aller Nutzerpasswörter gewinnen kann. Um dies zu adressieren, werden solche Systeme üblicherweise in Verbindung mit *host-proof hosting* [18] verwendet: Die Passwörter werden vor der Synchronisierung auf den Server des Anbieters lokal durch die JavaScript-Engine des Browsers oder ein Browser-Plugin verschlüsselt. Jedoch muss der Benutzer hierfür üblicherweise zwei Passwörter wählen: eines zur Authentisierung am Server des Anbieters, eines zur Verschlüsselung seiner Passwörter vor dem Hochladen. Hierdurch können Angriffe auf die Authentisierungsmechanismen des Anbieters ein erhebliches Risiko für Nutzer Darstellen, die die Passwörter nicht hinreichend unabhängig gewählt haben, etwa weil sie den unterliegenden Mechanismus und die Notwendigkeit von zwei Passwörtern nicht nachvollziehen konnten. Diese Schwäche hat erst kürzlich nach einem Angriff auf Lastpass zu erheblichen Sicherheitsbedenken trotz host-proof hosting geführt [19]. Um dieses Problem zu Adressieren, wird eine neuartige, verteilte Speicherung der Passwörter vorgeschlagen, die nur noch ein Passwort benötigt, und kein einzelnes Angriffsziel mehr bietet.

ISBN 978-3-88579-286-4

2 Ansatz

In diesem Beitrag werden Ansätze vorgestellt, ein klassisches Pseudo-SSO-System für das Web, das Passwörter verwaltet (wie etwa [13] [16] [20]) mit Community-Funktionen zu erweitern, um so bekannte Schwächen solcher Systeme zu adressieren. Eine Übersicht des Referenzsystems ist in Abbildung 1 dargestellt.

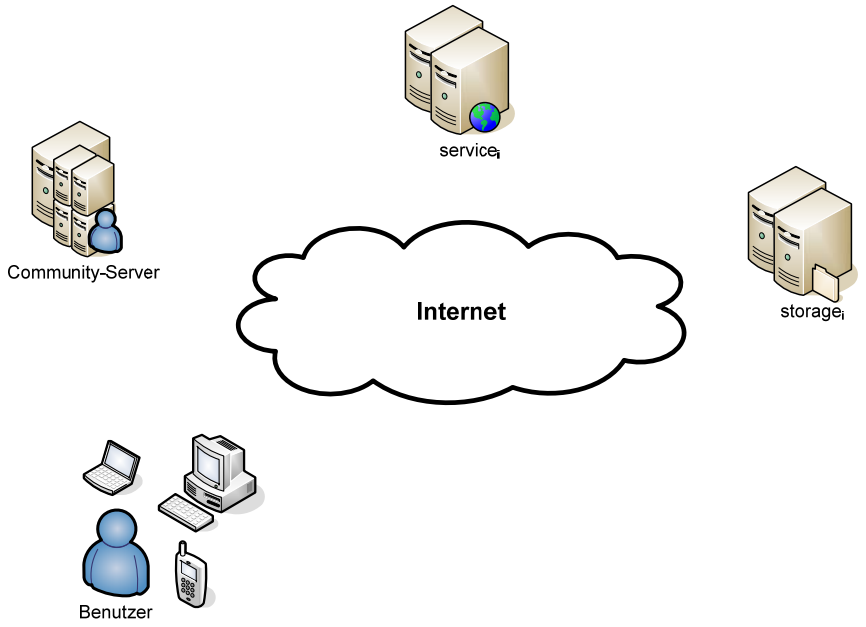

Abbildung 1: Übersicht Pseudo-SSO-System

Das System enthält zwei neuartige Komponenten, den Community-Server und die Dienste $storage_i$, die im Rahmen der weiteren Ausführungen genutzt werden, um das Pseudo-SSO-System mit Communityfunktionen zu erweitern: Die $storage_i$ sind Communities oder andere Web 2.0-Dienste [21], in denen nutzergenerierte Inhalte hinterlegt werden können, was zur sicheren verteilten Hinterlegung von Identitätsinformationen genutzt wird.

Der Community-Server erlaub eine Auslagerung der Konfigurationskosten, die in Systemen wie [13] [16] von jedem Nutzer einzeln zu erbringen sind, an die Community aller Nutzer des Systems.

ISBN 978-3-88579-286-4

3 Identitätsinformationen in Nutzergenerierten Inhalten

Hier wird ein hybrider Ansatz zur gegen Diensteanbieter resistente Hinterlegung von Identitätsinformationen in nutzergenerierten Inhalten vorgestellt, der die Vorzüge von password syncing und password hashing verbindet, indem er die beiden Ansätze kombiniert. Das password hashing wird in Kombination mit *direct login* und *lazy registration* [18] genutzt um die automatische Anmeldung an einem Subset von Web-Diensten möglich zu machen, die nutzergenerierte Inhalte akzeptieren. Dort werden die eigentlichen Nutzdaten des Identitätsmanagementsystems mittels host-proof hosting hinterlegt. Darüber hinaus wird der Einsatz zusätzlicher kryptografischer Mechanismen angeregt, die die Handhabung von Passwörtern ohne Kooperation der Serviceanbieter einfacher und sicherer gestalten können.

Das System soll dem Nutzer Zugriff auf beliebige Dienste im Web ($service_i$) gestatten, die mit Passwortmechanismen geschützt sind. Der hier vorgestellte Ansatz verwendet grundsätzlich password syncing, also eine Synchronisierung der Passwörter zwischen den genutzten Endgeräten, etwa auf Basis eines im Browser integrierten Passwort-Speichers. Zusätzlich können auch weitere Identitätsinformationen, etwa Name und Emailadressen zum automatischen Befüllen von Browserformularen hinterlegt werden. Die Passwörter werden jedoch nicht bei einem einzelnen Anbieter hinterlegt, sondern verteilt bei verschiedenen Diensten abgelegt.

Hierfür muss das System zunächst – möglichst weitgehend automatisiert – Konten bei den für die Hinterlegung genutzten Dienste ($storage_i$), etwa Communities, die nutzergenerierte Inhalte akzeptieren, anlegen. Interaktive Teile des Registrierungsvorgangs, beispielsweise CAPTCHAs, die zur Registrierung bei den Diensten notwendig sind, können an den Nutzer durchgereicht werden, und müssen nur einmal zur Einrichtung des Systems auf dem ersten genutzten Endgerät durchgeführt werden. Diese Anmeldung an den zum password syncing verwendeten Diensten wird über eine password hashing-Komponente durchgeführt, die die Passwortrichtlinien der $storage_i$ hinterlegt hat, und daher fähig ist, eine automatische Anmeldung durchzuführen und so eine nichtinteraktive, verteilte Synchronisierung ohne einheitliche Schnittstellen zu ermöglichen.

ISBN 978-3-88579-286-4

Hiernach kann – über Adapter zur Hinterlegung, die das System für jeden Hinterlegungs-Dienst $storage_i$ bereitstellen muss - eine über die $storage_i$ verteilte Speicherung von Passwort- und Identitätsdaten. Zur sicheren und gleichzeitig gegen einzelne Ausfälle resistente Methode zur verteilten Datensicherung kann Schwellwertkryptografie [22] zum Einsatz kommen. Sollte dies zu langsam sein, kann eine redundante, vollständige Speicherung der Daten unter einer starken Verschlüsselung mittels eines zufällig generierten kryptografischen Schlüssels erfolgen, der dann selbst mittels Schwellwertkryptografie auf die $storage_i$ verteilt wird. Sollten die nutzergenerierten Inhalte in einem konkreten Format vorliegen, kann das Pseudo-SSO-System entweder auf einen internen Vorrat solcher Medien zurückgreifen, oder aus dem Web frei verfügbare/bearbeitbare Vorlagen beschaffen, die als Vorlagen benutzt und für die Speicherung der Nutzerdaten modifiziert werden. Bestimmte der in Communities oft verwendeten Formate, etwa Bilder, eignen beispielsweise für eine durch Steganografie weiter verschleierte Speicherung der Nutzerdaten. Sollte auf einen Standardsatz von Medien zurückgegriffen werden, ist es sinnvoll, zur Täuschung mehrere Accounts pro $storage_i$ anzulegen, von denen nur einer die tatsächlichen Nutzdaten hält.

4 Crowdsourcing der Konfigurationskosten der Adapter

Wie eingangs ausgeführt, sind bei einem Pseudo-SSO-System erhebliche Aufwände zur Anpassung der Adapter an die sehr heterogenen Passwortrichtlinien der individuellen $service_i$ zu erbringen. Wenn jeder Benutzer für jeden genutzten $service_i$ die Passwortrichtlinie selbst konfigurieren muss, zieht das einen erheblichen Verlust der Benutzbarkeitsvorteile durch SSO nach sich. Darüber hinaus kann ein Benutzer bei manueller Anpassung der Passwörter die Zufallsverteilung stören und so je nach Passwortrichtlinie leicht erratbare Passwörter wählen [15].

Es bietet sich also an, wie von Kolter et al. [17] vorgeschlagen diese Konfigurationskosten innerhalb eines Identitätsmanagementsystems an die Community der Nutzer auszulagern (*Crowdsourcing* [23]). Allerdings wird im hier präsentierten Kontext nicht die Konfiguration einer deklarativen Schnittstelle am Server übernommen, sondern die Anpassung von Passwortrichtlinien zur spezifisch zugeschnittenen Generierung von dienstspezifischen Passwörtern auf dem Endgerät. Technisch ist der Unterschied jedoch zu vernachlässigen, da man sich auch Szenarien vorstellen könnte, in denen die Server ihre Passwortrichtlinien maschinenlesbar deklarieren.

Bei der Erzeugung eines Passworts – egal, ob diese im Rahmen von password hashing pseudozufällig erfolgt, oder bei password syncing basierten Systemen etwa im Rahmen einer lazy registration auf Basis von Zufallszahlen – kann direkt die vorkonfigurierte Passwortrichtlinie verwendet werden, um das erzeugte (Pseudo-)Zufallsmaterial in ein unter dieser Richtlinie akzeptable Passwort zu formen, wie dies etwa der PasswordSitter durchführt [16].

ISBN 978-3-88579-286-4

Somit kann der Interaktionsaufwand weiter minimiert werden. Die Anpassung der Passwörter an spezifische Diensterichtlinien ist natürlich auch für die Automatisierung des password hashing-Schritt des verteilten Synchronisierungsansatzes essentiell, dort muss aber nur eine kleine Menge vorkunfigurierterer Richtlinien (für die $storage_i$) mit ausgeliefert werden.

5 Diskussion

Es wurden neue Ansätze zur sicheren Verwaltung von Nutzerpasswörtern und -identitäten vorgestellt mit Hilfe von Community-Funktionen vorgestellt.

Der Ansatz zur Hinterlegung in nutzergenerierten Inhalten weist grundsätzlich gute Sicherheitseigenschaften auf, da er Offline-Angriffe gegen das Nutzerpasswort stark erschwert, dem Benutzer ein dynamisches Reduced Sign-On bietet, es ihm also transparent erlaubt, eine beliebige Anzahl von Passwörtern für die unterschiedlichen Dienste zu wählen, bis hin zum Single Sign-On.

Ein Crowdsourcing der Konfigurationskosten kann Interaktionsaufwände minimieren und gleichzeitig eine breitere Anwendbarkeit – und damit einen höheren Nutzen - des Systems gewährleisten.

Der Benutzer wird nicht mit neuen Interaktionsparadigmen konfrontiert, was bei der Interaktion mit Identitätsmanagementsystemen oftmals ein Problem darstellt. Stattdessen werden AJAX-Patterns wie direct login genutzt, um die vertraute Anmeldung an Webseiten zu automatisieren. Da es sich um ein Pseudo-SSO-System handelt, welches Passwortschnittstellen unterstützt, entfaltet das System sofort einen Nutzen, und benötigt keine Anpassung auf der Diensteseite.

Die vorliegenden Ergebnisse sind jedoch in vielerlei Hinsicht vorläufig. Eine Implementierung liegt noch nicht vor, ebenso wenig eine Konkretisierung der vorgeschlagenen kryptografischen Verfahren. Daher konnte auch die Performanz noch nicht evaluiert werden. Es ist jedoch davon auszugehen, dass für die relativ geringen anfallenden Datenmengen hinreichend performante Lösungen möglich sein sollten. Die unterliegende Pseudo-SSO Komponente, die auf password hashing basiert,

Es stehen also verschiedene zukünftige Arbeiten zur weiteren Untersuchung der hier vorgestellten Ansätze in der Praxis an. Im Rahmen einer Implementierung müsste zunächst die Auswahl einer konkreten, sicheren Realisierung der kryptographischen Komponenten erfolgen, inklusive einer Evaluierung der Performanz verschiedener Realisierungsansätze. Darüber hinaus stehen eine Optimierung der Benutzerschnittstelle, und eine Evaluierung der wahrgenommenen Nützlichkeit des Systems noch aus.

erschienen im Tagungsband der INFORMATIK 2011
Lecture Notes in Informatics, Band P192
ISBN 978-3-88579-286-4
weitere Artikel online:
http://informatik2011.de/519.html

Danksagung

Der Autor bedankt sich bei allen Kollegen aus Organisationen und Projekten, die mit ihren wertvollen Eingaben die Entwicklung der hier präsentierten Ideen wesentlich mit geprägt haben, insbesondere bei Heiko Roßnagel und Thomas Kriegelstein.

Literaturverzeichnis

1. Smith, R.E.: The Strong Password Dilemma. Computer. 18, (2002).
2. Brown, A.S., Bracken, E., Zoccoli, S., Douglas, K.: Generating and remembering passwords. Applied Cognitive Psychology. 18, 641-651 (2004).
3. Miller, G.A.: The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information. Psychological Review. 63, 81-97 (1956).
4. Florencio, D., Herley, C.: A large-scale study of web password habits. Proceedings of the 16th international conference on World Wide Web. S. 657-666. , New York, NY, USA (2007).
5. Chinitz, J.: Single Sign-On: Is It Really Possible? Information Security Journal: A Global Perspective. 9, 1 (2000).
6. LastPass: LastPass - Password Manager, Formular-Ausfüller, Password Management, http://lastpass.com/.
7. Recordon, D., Reed, D.: OpenID 2.0: a platform for user-centric identity management. Proceedings of the second ACM workshop on Digital identity management. S. 11-16. ACM, Alexandria, Virginia, USA (2006).
8. Cameron, K., Jones, M.B.: Design Rationale behind the Identity Metasystem Architecture. ISSE/SECURE 2007 Securing Electronic Business Processes. S. 117-129 (2007).
9. Facebook's OpenID Goes Live, http://www.allfacebook.com/2009/05/facebooks-openid-live/.
10. Hühnlein, D., Roßnagel, H., Zibuschka, J.: Diffusion of Federated Identity Management. SICHERHEIT 2010. GI, Berlin (2010).
11. Zibuschka, J., Roßnagel, H.: Implementing Strong Authentication Interoperability with Legacy Systems. Policies and Research in Identity Management (IDMAN 07). S. 149-160. Springer (2008).
12. de Clerq, J.: Single Sign-on Architectures. Proceedings of Infrastructure Security, International Conference. S. 40–58. , Bristol, UK (2002).
13. Halderman, J.A., Waters, B., Felten, E.W.: A convenient method for securely managing passwords. Proceedings of the 14th international conference on World Wide Web. S. 471-479. ACM, Chiba, Japan (2005).
14. Summers, W.C., Bosworth, E.: Password policy: the good, the bad, and the ugly. Proceedings of the Winter International Symposium on Information and Communication Technologies. S. 1-6. , Cancun, Mexico (2004).
15. Karp, A.H.: Site-Specific Passwords, http://www.hpl.hp.com/techreports/2002/HPL-2002-39R1.html, (2003).
16. Password Sitter: Home, http://www.passwordsitter.de/.
17. Kolter, J., Kernchen, T., Pernul, G.: Collaborative Privacy – A Community-Based Privacy Infrastructure. In: Gritzalis, D. und Lopez, J. (hrsg.) Emerging Challenges for Security, Privacy and Trust. S. 226-236. Springer Berlin Heidelberg, Berlin, Heidelberg (2009).
18. Mahemoff, M.: Ajax Design Patterns. O'Reilly Media, Inc. (2006).
19. LastPass: LastPass Security Notification, http://blog.lastpass.com/2011/05/lastpass-security-notification.html.

20. Gabber, E., Gibbons, P.B., Matias, Y., Mayer, A.J.: How to Make Personalized Web Browsing Simple, Secure, and Anonymous. Proceedings of the First International Conference on Financial Cryptography. S. 17-32. Springer-Verlag (1997).
21. O'Reilly, T.: What Is Web 2.0: Design Patterns and Business Models for the Next Generation of Software. Communications & Strategies, International Journal of Digital Economics. 65, 17-37 (2007).
22. Gemmell, P.: An introduction to threshold cryptography. RSA CryptoBytes. 2, 7–12 (1997).
23. Howe, J.: The Rise of Crowdsourcing, (2006).

ISBN 978-3-88579-286-4