Ohne Fenster zur Welt: Single-Sign-On im App-Zeitalter

Steffo Weber
Oracle
Hamburg

Schlüsselwörter
Single-Sign-On, Access Management, iOS, iPad, iPhone, Mobile, Cloud, REST, SAML, OAuth

Zielpublikum
Der Artikel richtet sich an SSO-Architekten und Implementierer mit Grundkenntnissen in Identity Management. Alle hier nicht explizit erläuterten Begriffe können auf Wikipedia oder anderen Quellen im Netz nachgelesen werden.

Einleitung
Klassisches, web-basiertes Single-Sign-On (SSO) basiert letzendlich immer auf dem Cookiemecha–nismus des Browsers. SSO-Cookies sind die Kerberos-Tickets der Web-Welt. Um an unterschiedlichen Websites (z.B. otn.oracle.com und www.amazon.com) angemeldet sein zu können, speichert der Browser die von den Sites ausgegeben Cookies. Man hat also *ein* Programm, welches auf *viele* Webapplikationen zugreifen kann[1].
In der App-Welt der Smartphones haben wir eine völlig andere Situation. Hier gibt es für Amazon eine eigene App, für Facebook eine eigene App etc. Es gibt jedoch keinen Speicher oder Cookie-Cache, der von beiden Applikationen gleichzeitig benutzt werden kann. Für prominente Dienste wie Facebook und Amazon ist das durchaus akzeptabel. Gibt aber ein Unternehmen an Kunden oder Mitarbeiter verschiedene Apps heraus (z.B. Firmentelefonbuch, Reisekosten etc) möchte der Mitarbeiter nicht bei jedem Start einer Firmen-App erneut ein Passwort eingeben. Ähnliches gilt auch für Kunden-Apps: Verlage, die unterschiedliche Zeitschriften in App-Versionen (z.B. Geo, Art - beide Gruner & Jahr oder Monopol, Sprüth Magers - beide Juno) herausgeben. Jeder App-Aufruf ist mit einer erneuten Anmeldung verbunden, auch wenn der Nutzer bereits durch eine andere App desselben Verlags authentisiert wurde.
Oracle hat die neue Version 11gR2 ihrer Identity & Access Management Suite um Komponenten erweitert, die das Problem des Single-Sign-On für Smartphone Applikationen lösen.

Klassisches Access Management
Die klassische Oracle Access Management Lösung bestand aus dem Access Manager und Identity Federation. Beide liessen und lassen sich integrieren. Mit der Übernahme von Sun Microsystems wurden die Komponenten Secure Token Service (STS) und Oracle OpenSSO Fedlet mit ins Portfolio übernommen.
Das klassische Access Management basiert auf einer Server/Agenten Architektur. Die Agenten prüfen, ob im HTTP Request das erforderliche SSO-Token (SSO Cookie) vorhanden ist und validieren dieses. Ist das Token ungültig schicken Sie den Nutzer mittels eines Redirects (HTTP 302) zunächst zu einer (zentralen) Login-Seite, die nach erfolgreicher Anmeldung des Nutzer wieder zur ursprünglich gewünschten Seite umleiten.
Die Agenten gibt es für WebServer (Oracle HTTP Server, Apache etc) wo sie als Modul per LoadModule Direktive eingebunden werden können oder als Servlet Filter. Oracle liefert darüber

---

[1] Applikations- oder Site-übergreifendes SSO funktioniert dann über Domaincookies oder Federation (SAML, OpenID, proprietäre Lösungen wie Cross-Domain-SSO etc). Federation (und damit Federated Logon) ist ein Mechanismus, bei dem der Nutzer sich beispielsweise zunächst bei amazon.com anmeldet (und für diese Domain ein SSO-Cookie erhält) und anschliessend auf otn.oracle.com ohne erneute Anmeldung zugreifen kann, obwohl er zunächst kein SSO-Cookie für diese Domain hat. Der Cookie-Cache des Browsers enthält nach dem Federated Logon Cookies für amazon.com und otn.oracle.com.

hinaus ein AccessSDK (Java) mit dessen Hilfe leicht ein eigener Agent gebaut werden kann, sollte für das gewünschte System keine vorkompiliertes Binary verfügbar sein.

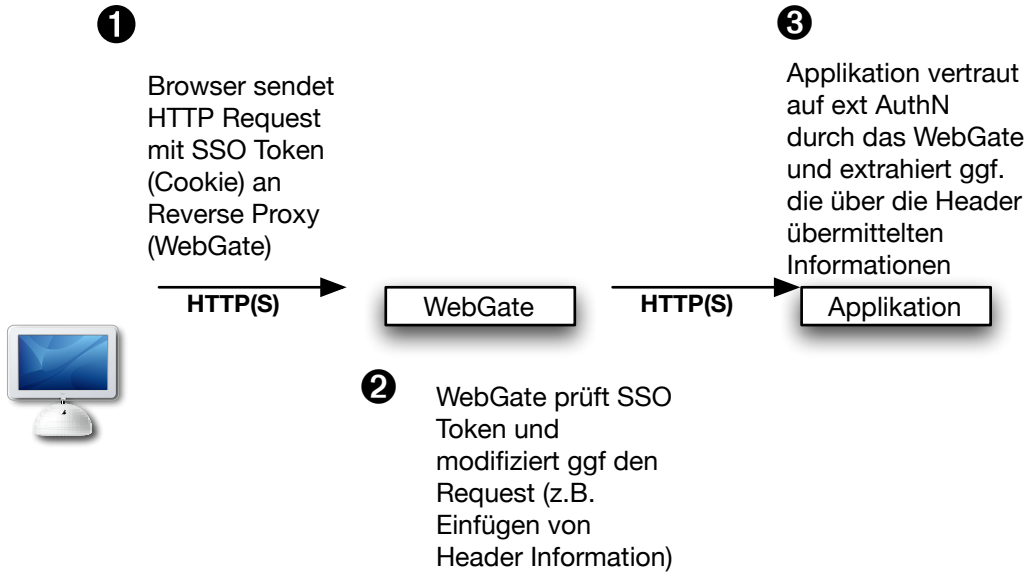

Abb. 1. Überprüfung eines HTTP Zugriffs durch ein WebGate (Agent).

Die obige Abbildung illustriert nochmal, die Kontrolle einer Website durch ein WebGate. Interessant ist hier, wie der Agent (WebGate oder Access SDK) überprüft, ob ein Nutzer am Access Server angemeldet ist. Beide (WebGate und SDK) verwenden hierzu ein proprietäres Protokoll (NUP) Wir werden sehen, dass im neuen 11gR2 Release Features dieses Protokolls per REST zur Verfügung gestellt werden.

Cloud und REST WebServices
Wie oben beschrieben, kann das Access SDK dazu verwendet, Dienste vom Access Server anzufordern. Dienste sind

- Authentisierung
- Autorisierung
- Nutzeranlage und Profildienste

Die Verwendung eines SDKs ist für die (alte) Java Welt durchaus üblich. In einer service-orientierten Welt, wie wir sie mit Clouddiensten vorfinden ist dies jedoch nicht akzeptabel. Cloud ist ein Ökonomiemodell und SaaS Provider haben nur begrenzte Möglichkeiten, die Software anzupassen. Meistens ist eine Installation kundenindividueller JARs nicht möglich. Infolgedessen benötigen Clouddienste eine andere, (Web)service-orientierte Form der Kommunikation.
Im Cloudumfeld sind REST-basierte Dienste durchaus eine alternative zu den SOAP-basierten Diensten, da sie einfach über (serverseitiges) JavaScript aufgerufen werden können. Hierzu ist lediglich ein HTTP Call (ggf mit einfachen Headern) notwendig. Die Komplexität des SOAP Layers entfällt hierbei[2].

---

[2] Es gibt einen Preis, der hierfür zu zahlen ist: Sicherheit (bei SOAP mittels WS-Security spezifiziert), Container Layer (SOAP Request können vom Container behandelt werden) etc sind nur einige Aspekte. Dennoch ist REST durchaus attraktiv im Cloud-Umfeld und es wäre nachlässig, es zu ignorieren.

Die Version 11gR2 des Oracle Access Managers verfügt über REST Dienste[3] zur

- Authentisieung
- Autorisierung
- Nutzer- und Profildienste

Ein einfaches Beispiel zum Anlegen eines Nutzers im OAM Nutzerverzeichnis:

```
curl -H "Content-Type: application/json" --request POST http://
identity.oracle.com:14100/oic_rest/rest/userprofile/people/ -d
'{"uid":"Fred","description":"Another Test User","lastname":"Fredsen",
"commonname":"Fred Fredsen","firstname":"Fred", "password":"test123"}'
```

Der entsprechende Nutzer wird in dem LDAP Verzeichnis angelegt, welches der OAM nutzt (man muss ein wenig aufpassen: pro OAM ist ein Identity Store aktiv. Die Stores für die mobilen REST Dienste können aber unabhängig hiervon konfiguriert werden). Das obige Beispiel beinhaltet keine Sicherheitseinstellungen. In der Praxis wird man hier JWT (Json Web Token) verwenden. Das Token wird dann über entsprechende HTTP Header Variablen mitübertragen.

Ist der Nutzer angelegt, kann er sich per REST Call am OAM (Mobile & Social Server) anmelden.

```
curl -i -H "Content-Type: application/json" --request POST http://
10.140.223.155:14100/oic_rest/rest/jwtauthentication/authenticate -d '{
"X-Idaas-Rest-Subject-Type":"USERCREDENTIAL",
"X-Idaas-Rest-Subject-Username":"steffo",
"X-Idaas-Rest-Subject-Password":"test123",
"X-Idaas-Rest-New-Token-Type-To-Create":"USERTOKEN"}'
```

und OAM antwortet mit

```
HTTP/1.1 200 OKDate: Thu, 13 Sep 2012 15:28:27 GMTTransfer-Encoding:
chunkedContent-Type: application/jsonX-IDAAS-REST-VERSION: v1
Set-Cookie:
JSESSIONID=8ckhQR7b8DRV6qZL8GPyG8zrLj0HlHkYbmJ66qk4qWL1h281QG2p!746662696;
path=/; HttpOnly
X-ORACLE-DMS-ECID:
4e3b51c86012a0d5:-65dc3b22:139a0a61325:-8000-00000000000018f9
X-Powered-By: Servlet/2.5 JSP/2.1

{
"X-Idaas-Rest-Token-Value":"ey....Pr6U",
"X-Idaas-Rest-User-Principal":"steffo",
"X-Idaas-Rest-Provider-Type":"JWT",
"X-Idaas-Rest-Token-Type":"USERTOKEN"
}
```

Dieses (hier gekürzte) Token kann man auch mit Hilfe eines REST Dienstes validieren lassen

```
curl --request GET "http://10.140.223.155:14100/oic_rest/rest/
jwtauthentication/validate?X-Idaas-Rest-Subject-Value=...nI&X-Idaas-Rest-
Subject-Type=TOKEN"
```

und OAM antwortet

---

[3] Eine genaue Dokumentation findet sich hier: http://docs.oracle.com/cd/E27559_01/dev.1112/e27134/appendixcurl.htm#BABIFIJB.

```
{
"X-Idaas-Rest-Token-Value":"eyJh...bnI",
"X-Idaas-Rest-User-Principal":"steffo",
"X-Idaas-Rest-Provider-Type":"JWT",
"X-Idaas-Rest-Token-Type":"USERTOKEN"}
```


Beim Validieren kann der Dienstenutzer also erkennen, welcher Nutzer hinter einem Token steckt. OAM kennt unterschiedliche Arten von Token (hier ist die Analogie zu JWT und OAuth unverkennbar) und entsprechend gibt es drei Parteien: Nutzer, OAM und Resourcen.

User Token – entspricht im wesentlichen einem klassischen SSO Token
Access Token – wird benötigt, um auf eine Resource zuzugreifen. Der Nutzer möchte der Resource nicht sein Usertoken übersenden, welches an seine Identität gebunden ist und fordert daher zunächst vom OAM ein Access Token an, anhand dessen die Resource den Nutzer identifizieren kann aber ohne seine Identität zu übernnehmen.
Client Token – wird im mobilen Umfeld verwendet und erlaubt den Zugriff auf eine Server oder eine Web-Anwendung.

Wie oben skizziert, lassen sich Identitätsdienste leicht (und sicher) über REST Aufrufe nutzen. Neben den Cloud-Diensten sind es im wesentlichen mobile Endgeräte, die hier von der – gegenüber SOAP – schlankeren Lösung – profitieren. Dies möchte ich jetzt näher erläutern.

Mobile Endgeräte

Anwendungen auf mobilen Endgeräten (iOS, Android) kommunizieren in den meisten Fällen mit serverseitigen Diensten: die Wetter App kontaktiert Yahoo, um Wetterdaten zu erfragen, die Bundesliga-App benötigt die aktuellen Spielstände und selbst die Notizzettel-App synchronisiert meine Notizen mit einem Cloud-Dienst (die einzige App, die völlig ohne Netz und Dienste leben kann, scheint die App zum Binden von Krawattenknoten zu sein).
Der Aufruf von WebServices erfolgt – sicherlich auch aufgrund der auf Smartphones geringeren Rechleistung sowie Latenzgründen – mittels REST und nicht mittels SOAP. Die Frage ist nun: wie kommt man mittels der REST Dienste zu einem App-übergreifenden Single-Sign-On. Wir benötigen etwas, das den Cookie Cache des Browsers simuliert. Diese Komponente nennt Oracle “SSO-Agent”[4] und ist Bestandteil von Oracle Mobile and Social (welches wiederum ein Bestandteil von OAM ist). Der SSO-Agent ist eine Art Proxy: iOS Apps, die eine Authentisierung des Nutzers erfodern, wenden sich an den SSO-Agent und dieser wiederum wendet sich an den OAM (Mobile & Social). Der SSO-Agent kann eine eigentständige App sein oder in eine andere App integriert sein; Oracle liefert eine iOS-Library (libmobileSDK). Im OAM wird festgelegt, von welchen Apps (SSO-Clients) der SSO-Agent Anfragen entgegen nimmt.

Anforderungen an mobile Zugriffe

Lassen wir für einen Moment noch mal die oben erwähnten Konzepte von SSO-Agent und SSO-Client ausser acht und legen zunächst fest, wie wir uns die Steuerung mobiler Zugriffe vorstellen. Der Zugriff von einem mobilen Endgerät auf einen WebService soll anhand folgender Parameter kontrollierbar sein

Nutzer. Der Dienst soll nur für authentisierte Nutzer zur Verfügung stehen. Hier unterscheidet sich die mobile Welt nicht von der klassischen Desktop-Welt.

---

[4] Wie die meisten IT Firmen ist auch Oracle unschlagbar in der Überlagerung technischer Begriffe: der SSO-Agent hat nichts mit OpenSSO-Agenten oder WebGates zu tun.

Gerät. Nutzer verfügen oftmals über mehrere mobile Geräte. Falls ein Nutzer jedoch von einem mobilen Gerät, welches bislang nicht beim Diensteanbieter mit dem Nutzer assoziiert war, einen Dienst nutzt, velangt der Anbieter einmalig eine weitere Authentisierung (Step-Up Authentication).

Jail Break. Nutzern, die ein Gerät mit Jailbreak verwendet soll der Zugriff verwehrt werden.

Dies bedeutet, dass eine Applikation, die auf eine vom OAM geschützte Infrastruktur zugreifen möchte, die Geräteregistrierung mit der Nutzeranmeldung an den OAM Server schicken muss. Danach erhält die App das oben bereits erwähnte Access Token[5] mit dem sie den Dienst (z.B. Yahoo Wetter) nutzen kann.

Mobile SDK für iOS

Der Authentisierungsablauf ist wie folgt

1. Der Nutzer startet eine App auf dem mobilen Gerät (z.B. iPhone). Wir gehen im folgenden davon aus, dass der Nutzer zunächst die Wetter-App startet (hierfür muss er in Schritt 3 seine Zugangsdaten eingeben). Wenn später die Zeitschrift-App für die Zeitschrift "Monopol" aufgerufen wird, erledigt der SSO-Agent die notwendigen Schritte. Wichtig ist, dass der SSO-Agent diese Daten nicht einfach speichert[6], sondern die entsprechenden Tokens (User Token, Client Token etc) verwendet. Diese Tokens haben eine beschränkte Gültigkeit (ähnlich wie die bekannten OAM Cookies).

2. Die App möchte auf eine vom Access Manager geschützte Resource zugreifen (z.B. Server des Yahoo Wetterdiensts oder Content Server einer Zeitschrift) und benötigt hierzu ein Access Token. Hierzu wendet die App sich an den SSO-Agent. Der SSO-Agent kann eine eigenständige App sein oder per libmobileSDK in die Wetter- oder Zeitschrift-App integriert sein.

3. Mobile SSO-Agent verlangt vom Nutzer die Eingabe von Nutzername/Passwort oder eine andere Art der Authentisierung (z.B. Facebook, Twitter).

4. Der SSO-Agent sendet die Login Daten an den OAM (Mob&Soc) Server weiter. Darüber hinaus sendet er Informationen über das Endgerät an den Server.

5. Der OAM (Mob&Soc) prüft, ob er das Endgerät kennt. Handelt es sich um ein unbekanntes Gerät, so kann der OAM eine weitere Authentisierung (Step Up Authentication) fordern[7].

6. Der SSO-Agent erhält ein User Token (für den angemeldeten Benutzer) und fordert mit dessen Hilfe ein Access Token an.

7. Die App greift mit Hilfe des Access Tokens auf den WebService (z.B. Wetterdienst) zu.

Damit ist der Nutzer am OAM angemeldet und hat eine gültige Session. Falls der Nutzer nun eine weitere (registrierte) App (z.B. "Monopol") startet, prüft der SSO-Agent, ob das User Token noch gültig ist und verlangt ein weiteres Access Token.

---

[5] Das Konzept mit User Token und Access Token ist neu in der OAM Welt (und auch bei OpenSSO gab es so etwas nicht). Die Begriffe sind in der OAuth Spezifikation hinreichend erläutert.

[6] Es gibt einen – bei Bedarf aktivierbaren – Offline Mode, bei dem die Authentisierungsdaten vorgahlten werden könnne. Dieser soll jedoch hier nicht weiter interessieren.

[7] Dieser Schritt wird nicht vom OAM selbst, sondern vom Oracle Adaptive Access Manager (OAAM) erledigt, der in 11gR2 vollständig in OAM integriert ist.

Zusammenfassung

Durch Oracle Access Manager Mobile & Social kann Single-Sign-On auch für nicht Browser basierte Apps umgesetzt werden: hat man statt eines Desktop Browsers Smartphone Apps, so kann mittels des Oracle iOS SDKs eine Single-Sign-On für (registrierte) iOS Anwendungen erreicht werden. Das hat für den App-Programmierer den Vorteil, dass er relativ einfach SSO-fähige Anwendungen erstellen kann. Zudem bietet der OAM (Mob&Soc) in der Version 11gR2 eine Reihe von REST-bezogenen Funktionen.

- Authentisierung, Autorisierung, Nutzerverwaltung per REST API
- Schutz von REST-Webservices durch OAM Komponenten (wie z.B. WebGates)

Das oben erwähnte SDK (für iOS) gibt es auch in einer Java Version für Desktop Rechner.

Kontaktadresse:
Dr Steffo Weber
Oracle
Hamburg

Telefon:	+49 (0) 40-8909 1614
E-Mail	**steffo.weber@oracle.com**
Internet:	oracle.com