Single Sign On mit Forms direkt gegen ein Active Directory

Wolf G. Beckmann
TEAM GmbH
Paderborn

Schlüsselworte

Forms, SSO, Kerberos

Einleitung

Sicherheit ist zurzeit ein hochaktuelles Thema und gerät deshalb aktuell stärker in den Fokus der IT-Abteilungen. Über Single Sign On (SSO) wird nicht nur die Sicherheit erhöht sondern auch die Benutzerfreundlichkeit. Dennoch wird es eher selten eingesetzt. Das liegt auch daran, dass unterschiedliche Standards hier den Aufwand und die Nutzung zusätzlich zu lizenzierender Produkte erforderlich machen. In Windows ist SSO mit Kerberos in der Windows-Domäne direkt verankert. Forms beherrscht Single Sign On, benötigt aber das Oracle Internet Directory (OID), um im Rahmen der Windows-Domäne eingesetzt werden zu können.

Die Aufgabenstellung

Die Aufgabe hört sich einfach an: Wir möchten, dass unsere Formsanwendung beim Starten keine Login-Maske zeigt, aber dennoch sich mit dem Windows-Benutzer, der sich in der Windows-Domäne bereits erfolgreich angemeldet hat, auch in der Applikation anmeldet.

Somit brauchen wir in der Applikation Benutzer, die zu den Windows-Benutzern „passen“. In diesem Fall haben wir Datenbank-Benutzer, die genauso heißen, wie die Windows-Benutzer.

Was bietet Oracle dazu an?

Oracle bietet für Forms dazu das Produkt „Oracle Single Sign-On“ bzw. „Oracle Access Manager“ in Kombination mit dem „Oracle Internet Directory“ (OID) an. Dabei ist das OID quasi eine Kopie des Active Directory (AD) vom Windows Domänen Controller und muss somit kontinuierlich synchronisiert werden. Weiterhin müssen diese Produkte auch lizenziert werden. Das ist für die Aufgabenstellung ggf. viel Aufwand.

Also kommt die Frage nach Alternativen auf

Idee: In der DB wird ein nichtöffentliches Standardpasswort gespeichert und der Benutzer von Windows wird ausgelesen und übermittelt.
Hier gibt es leider mehrere Mängel: Wird das Passwort bekannt, muss die Applikation angepasst werden ist eher der Schönheitsfehler, das eigentliche Problem ist, dass ich nicht wirklich authentifiziert bin. Wenn ich an einem Rechner im Netz einen lokalen Benutzer anlege, der den gleichen Namen wie mein Chef hat, kann ich mich als mein Chef anmelden.

Also brauchen wir eine Authentifizierung:
Im Internet sieht man sehr häufig OAuth, beim kurzen Betrachten fehlt aber die Integration in Windows.

In Windows ist Kerberos das Standard-Protokoll für die Authentifizierung und Kerberos ist netterweise auch in Unix/Linux Umgebungen anzutreffen. Somit beschäftigen wir uns damit, wie Kerberos funktioniert:

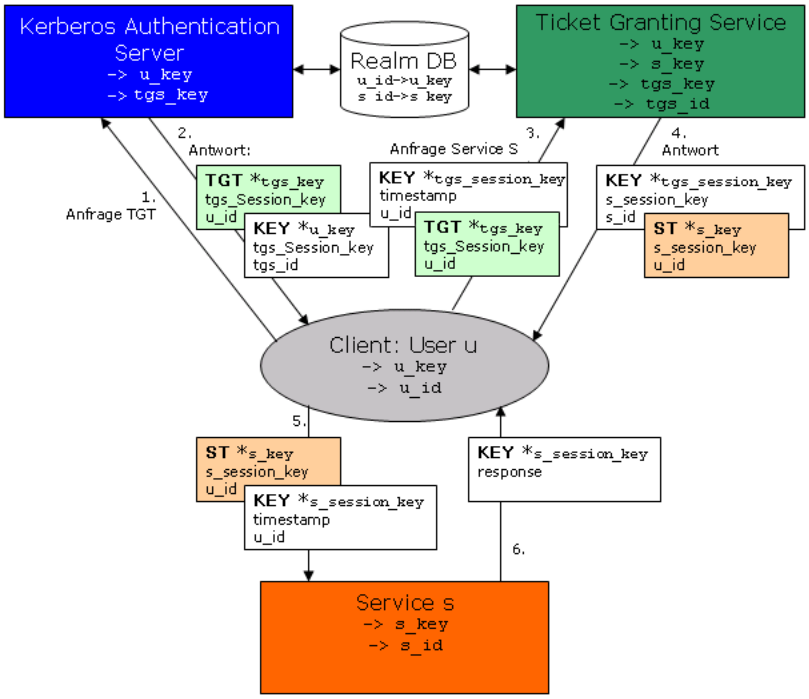

Abb. 1: Kerberos Authentifizierung

Also grob gesprochen: Nach der Authentifizierung am Kerberos Authentication Server bekommt man ein Ticket Granting Ticket (TGT). Damit kann der Nutzer vom Ticket Granting Service ein Session Ticket (ST) Anfordern, um sich damit am Service anzumelden.

Wie hilft uns Kerberos bei unserer Aufgabenstellung?

Erst einmal nicht so viel. Die Kerberos-Authentifizierung kann direkt für SQL-Plus Sessions vom Client zur Datenbank verwendet werden, aber hier soll sich ja Forms bzw. der Forms-Nutzer vom Application-Server aus authentifizieren.

Das Konzept: Der Benutzer hat kein Passwort

Also brauchen wir ein Konzept, in dem der Benutzer kein Passwort hat bzw. kennt:

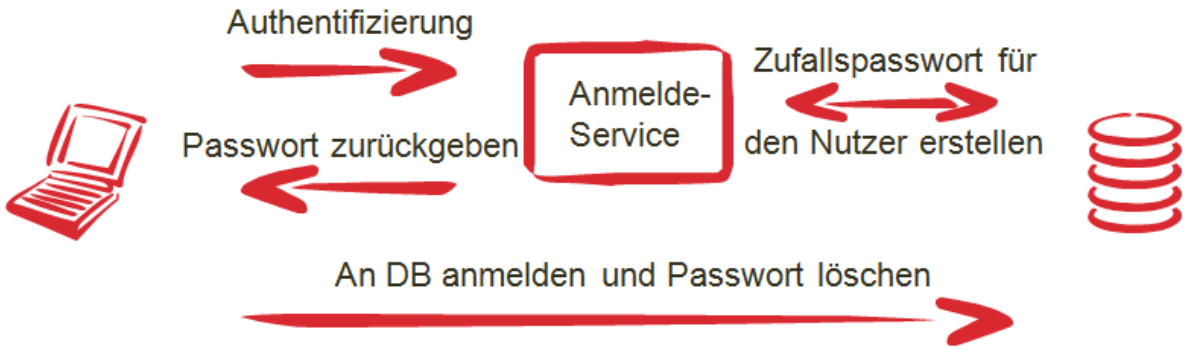

Abb. 2:Konzept kein DB-Passwort

Also, wir entwickeln einen kleinen Dienst, an den die Forms-Applikation den Windows-Benutzer übermittelt. Der Dienst erstellt dann ein Passwort, mit dem Forms sich an der Datenbank anmeldet und danach das Passwort wieder löscht.
Durch dieses Konzept brauchen die Benutzer kein Passwort mehr und es wird auch Client-Seitig kein Passwort gespeichert.

Nun kommt die NSA und der Mann in der Mitte ins Spiel

Das Konzept sieht gut aus, hat aber das Problem, dass eine IT-Abteilung, die auf Sicherheit bedacht ist, es sofort verwirft. Da sowohl über einen Netzwerk-Sniffer das Passwort leicht zu ermitteln ist und zum anderen das Prinzip für eine „Man-in-the-Middle“ Attacke anfällig ist. Da es hier um Sicherheit geht, müssen wir das Konzept noch verfeinern.

Damit kommen wir zur Frage:

Wie funktioniert eine sichere Kommunikation?

Hier gibt es 3 Punkte, die zu berücksichtigen sind:

> **Vertraulichkeit**
> Also die Notwendigkeit, dafür zu sorgen, dass die Daten nicht von anderen gelesen werden können.
> **Integrität**
> Bedeutet, dass dafür gesorgt werden muss, dass die Daten integer sind - also sicherstellen, dass die Daten nicht verändert wurden bzw. werden können.
> **Authentizität**
> Also sicherstellen, dass der jeweilige Kommunikationspartner auch derjenige ist, mit dem man kommunizieren möchte.

Wie bekommen wir die 3 Punkte unter einen Hut:

Die Vertraulichkeit können wir über Verschlüsslung für den Hin- und Rückweg erreichen. Die Integrität bekommt man am einfachsten über eine Checksumme der Daten.

Kommen wir zur Authentizität: Dass wir uns mit Kerberos beschäftigt haben, war also gar keine verschwendete Zeit: Jetzt können wir das Wissen benutzen, um die Authentizität des Absenders zu garantieren. Dass die Antwort auch tatsächlich von dem Empfänger, also unserem Service kommt,

erreichen wir auf folgendem Wege:
Das Paket, welches wir dem Service senden wird asymmetrisch verschlüsselt, also der Client hat den öffentlichen Schlüssel des Services und verschlüsselt damit die Daten. So kann nur der Service die Daten lesen. Wir übermitteln jetzt in den verschlüsselten Daten dem Service noch zusätzlich einen per Zufall generierten Schlüssel, mit dem der Service die Daten, die zum Client zurückgesendet werden, symmetrisch verschlüsselt.

Selbst wenn wir nun die Vertraulichkeit und die Authentizität sichergestellt haben, können wir die Integrität der Daten nur indirekt darüber feststellen, dass das Passwort ggf. nicht stimmt, weil die verschlüsselten Daten verändert wurden. Um sicher zu stellen, dass die Daten „Integer“ sind, wird sowohl auf dem Hinweg, als auch auf dem Rückweg ein Hash über die Nutzdaten erstellt und mit verschlüsselt.

Somit ergibt sich folgendes Bild:

Abb.3:Sichere Übertragung der Daten

Hinweis: Durch das Kerberos Service Ticket kann der Anmelde-Service den Benutzernahmen ermitteln

Das Gesamtkonzept steht, nun folgt die Umsetzung…

Zuerst müssen wir unsere Komponenten aufteilen. Um plattformunabhängig zu sein, bietet es sich an, den Dienst in Java und ihn abhängig von der Lizenzierung auf dem Weblogic-Server, einem eigenen Tomcat-Server oder Stand-Alone umzusetzen.

Aus Forms heraus kommen wir nicht ans System, um das Kerberos Service Ticket zu erstellen. Also bietet es sich an, eine JavaBean zu erstellen. Die kann auch gleich die Kommunikation mit dem Anmelde-Service übernehmen.

Und in der Datenbank erstellen wir die Funktionen für die Passwörter und zur Sicherheit einen Job, der per Timeout die Passwörter wieder „löscht“, wenn dies nicht von Forms aus geschieht.

Windows will es uns nicht einfach machen…

Um das Konzept Client-seitig umzusetzen, müssen wir in den Anmeldeprozess von Forms eingreifen. Damit wir auch bei Dialogwechseln die Kontrolle haben und bei neuen Anmeldungen, die im

Hintergrund geschehen, auch den Anmelde-Service benutzen, passen wir den Forms-Trigger „On-LOGON“ an.

Dort rufen wir unsere JavaBean auf. Um an unser Service-Ticket zu kommen, wird unter Java der „Java Authentication and Authorization Service“ (JAAS) eingesetzt. Nur wenn man sich „brav“ an alle Tutorials hält, stellt man fest, dass es unter Windows nicht funktioniert. Ein Grund dafür liegt an dem Sicherheits-Feature allowtgtsessionkey. Windows verbietet im Standard, dass ein Session-Key zurückgegeben wird.

Das Problem kann man durch eine Änderung in der Registry beheben. Doch das möchte kaum ein Sicherheitsbeauftragter in einem Unternehmen.

Eine Lösung ist, nativ auf die Windows-Schnittstelle zuzugreifen. Das hat schon vor uns jemand erkannt und hat die Lösung als das quelloffene Waffle (http://dblock.github.io/waffle/) jedermann zur Verfügung gestellt.

Somit haben wir alle Herausforderungen gemeistert…

… und die Benutzer unserer Forms-Applikation brauchen kein Passwort mehr auf einem Zettel unter der Schreibtischunterlage aufbewahren.

Kontaktadresse:
Wolf G. Beckmann
TEAM GmbH
Hermann-Löns-Straße 88, 33104 Paderborn
D-33104 Paderborn

Telefon:	+49 (0) 5254-8008 39
Fax:	+49 (0) 5254-8008 19
E-Mail	wb@team-pb.de
Internet:	www.team-pb.de