Logon Security: Last Man Standing

Jan Schreiber
Loopback.ORG GmbH
Hamburg / Frankfurt / München

Schlüsselworte
Datenbank Sicherheit Verzeichnis Anmeldung Kerberos

Einleitung

Früher war es einfach: In der Datenbank werden Benutzerkonten angelegt, für jeden Mitarbeiter eines, und ein Passwort zugewiesen. Der Mitarbeiter musste sich dieses merken, zusammen mit den Account-Namen und Passwörtern der vielen anderen Systeme und Anwendungen im Unternehmen. Also schrieb er es auf. Dann kamen Passwortverwaltungsprogramme auf, die die Zugangsdaten sammelten und ablegten. Allerdings blieb ein Grundproblem ungelöst: Ein Mitarbeiter konnte in verschiedenen Systemen verschiedene Konten mit unterschiedlich aktualisierten Berechtigungen haben, und eine zentrale Änderung der Zugriffsrechte eines Benutzers war nicht unmittelbar und ich Echtzeit möglich. Die Organisation verfügte über keine einheitliche, belastbare Zuordnung ihrer Mitarbeiter zu deren digitalen Identitäten. Hier kamen Identity Management Systeme (IDM) zum Einsatz, die neben einem zentralen Inventar der Rechte und Rollen in der Organisation auch eine zentrale Implementation von Sicherheitskomponenten wie Passwort-Komplexitätsanforderungen, Ablauffristen oder Two-Factor-Authentication ermöglichten.

Die meist verwendete IDM-Lösung ist Microsoft Active Directory, das neben einem LDAP-kompatiblen zentralen Benutzerverzeichnis auch eine Implementation von Kerberos mitbringt. Kerberos ist ein kryptographisch abgesichertes Sicherheitsverfahren, das über Zugriffsrechte über Tickets zuteilt. Die Identifikation des Benutzers über Passwörter, biometrische Merkmale oder den Besitz von Dingen wie Zugangstokens ist nur bei der ersten Anmeldung notwendig, weitere Rechte können über das dann vergebene Ticket gewährt. Andere Lösungen sind UNIX-basierte LDAP-Server, Smart-Card-basierte Verfahren und SSL-Kryptographie basierte Verfahren, die mit Zertifikaten arbeiten, wie sie auch für Webserver und Email verwendet werden.

Neben der Rechteverwaltung für Entwickler, Administratoren und Anwender spielt auch die Konfiguration von maschinellen Zugriffen eine Rolle. Für Application Server oder anderen Datenbanken, die sich über Datenbank-Links anmelden, werden oft statische Passwörter vergeben, die sich dann allerdings nur mit Aufwand regelmäßig ändern lassen. Auch hier existieren alternative Authentifikationsverfahren, beispielsweise über SSL-Zertifikate oder Datenbank Wallets.

Im folgenden Artikel werden die verschiedenen Technologien zur Authentifikation und Authorisation an der Datenbank gegenübergestellt und nach Einsatzaufwand, Einsatzvoraussetzungen, Wartbarkeit und Sicherheit verglichen.

Passwörter und deren Hashes in der Datenbank

Wie Passwörter von direkten Benutzern in der Oracle Datenbank geändert werden, ist jedem DBA bekannt: Über ALTER USER IDENTIFIED BY wird direkt in SQL ein neues Passwort gesetzt.

Oracle erzeugt aus dem Klartext-Passwort eine kryptographische Quersumme, einen sogenannten Hash, der im Allgemeinen möglichst die Eigenschaft haben soll, kollisionsfrei zu sein, also zu gewährleisten, das keine zwei unterschiedlichen Passwörter zu einem gleichen Hash führen.

Außerdem darf es natürlich nicht möglich sein, aus dem Hash auf das verwendete Passwort zurückzuschießen.

Die Datenbank legt die Hashes in der Tabelle SYS.USER ab. Ein Datenbankbenutzer mit Lese-Rechten auf diese Spalte kann sie auslesen und entweder direkt für die Authentifikation verwenden oder versuchen, das Original-Passwort herauszufinden.

Führt das gleiche Passwort stets zum gleichen Hash-Wert, ist es mit Hilfe von Abbildungslisten, sogenannten Rainbow Tables, möglich, aus einem bekannten Hash rückwärts auf das Original-Passwort zu schließen. Um dies zu verhindern, kann ein sogenanntes Salt verwendet werden, ein möglichst zufälliger Wert, der dem Passwort vor dem Ermitteln des Hash-Wertes zugeführt wird.

Im Laufe der Versionsgeschichte wurden verschiedene Hash-Algorithmen verwendet:

DES

Der Algorithmus DES wurde in den Versionen 6 bis 10gR2 verwendet. Er stammt von Bob Baldwin, der auch die Windows NT und VMS Algorithmen entworfen hat, und wurde 1993 veröffentlicht. Oracle fügt in diesen Versionen den Benutzernamen als Salt hinzu, bevor der Hash erzeugt wird. Das Salt ist also nicht zufällig. Der DES Hash ist ganz klassisch in der Spalte PASSWORD der Tabelle SYS.USER$ hinterlegt.

SHA1

wurde in 11gR1 bis 11.2.0.4 verwendet, ist aber in der Datenbank 12.2.0.2 weiterhin vorhanden. Dieses Verfahren verwendet eine in den Passwörtern vorhandene Groß / Kleinschreibung. Als Salt dient hier ein durch die Datenbank erzeugter Zufallswert, der aber allerdings zusammen mit dem Hash im Abschnitt S: des Feldes SYS.USER$.SPARE4 abgelegt wird. Außerdem wird der Salt über das Netzwerk an den Client übermittelt (siehe unten). SHA1 ist schnell zu berechnen, was einen erheblichen Nachteil für die Passwort-Sicherheit bedeutet. Außerdem ist SHA1 kryptographisch gebrochen.

SHA2

wird seit 12.1.0.2 verwendet. Der Hash wird im Wert "T" der Spalte SPARE4 abgelegt. Es wird bei der Oracle-Anmeldung eine Kombination aus PBKDF2 (auf dem Client), SHA2 (auf dem Server) und SHA512 verwendet. SHA2 ist wesentlich schwerer zu berechnen als SHA1.

HTTP Digest

wird ab 12.1.0.1 für alle Benutzerkonten mitgerechnet. Der Algorithmus ist schwächer als SHA2 und basiert auf MD5. Ein Salt fehlt. Aus dem Hash kann mit einfachen Mitteln auf das Passwort geschlossen werden.

Problematisch ist, das standardmäßig alle Hashes gerechnet werden und im Data Dictionary vorhanden sind. Um das Passwort zu ermitteln, muss nur das schwächste Glied der Kette gebrochen werden. Wird ein Hash-Algorithmus einmal verwendet, ist es sehr schwierig, ihn wieder zu entfernen, da Database Links und andere Anmeldungen die mit ihm erzeigten Hashes verwenden und ohne sie nicht mehr funktionieren.

Oracle Datenbank-Passwörter mit mehr als 11 Zeichen lassen sich zur Zeit nicht mit verfügbarer Hardware zurückberechnen.

Während der Datenbank-Anmeldung wird ein Challenge-Response-Verfahren verwendet, um den Benutzer zu authentifizieren. Dazu werden zwischen der Datenbank und dem Datenbank-Client sogenannte Session Keys gebildet. Bei der Anmeldung sendet der Client das Benutzer-Passwort mit dem Session Key verschlüsselt an die Datenbank. Die Session Keys werden mit dem Passwort-Hash verschlüsselt ausgetauscht. Es gibt einen Server- und einen Client-erzeugten Session Key, die zusammen verwendet werden, um das Benutzerpasswort zu verschlüsseln, welches dann über das Netzwerk versendet wird.

Mit einem Mitschnitt des Netzwerk-Verkehrs ist es möglich, einen Brute Force Angriff auf das Passwort durchzuführen, auch ohne den Hash in der Datenbank zu kennen. Darüber hinaus kann über einen Man-In-The-Middle-Angriff die vereinbarte Version des Anmeldeprotokoll "künstlich" herabgesetzt werden ("Downgrading"), um die Verwendung einfacherer Passwort-Hashes zu erzwingen. In einem neueren Angriff ist es möglich, die Anmeldung anzugreifen, auch ohne den Netzwerkverkehr mitzuschreiben: Über die OCIPasswordChange API ist es möglich, das Passwort während der Anmeldung zu ändern.

Bei einem Angriff auf das O5LOGON-Anmeldeprotokoll (Oracle 11g stealth password cracking), der nicht einmal Spuren in den Server-Protokollen hinterlässt, weil die Anmeldung nicht korrekt zu Stande kommt, kann der Session Key mit dem Salt ausgelesen werden. So kann das Passwort angegriffen werden, ohne die Hashes zu kennen.

Weitere Angriffsvektoren auf Passwörter und Hashes sind:

- Verbindungsaufbau mit erratenen Passwörtern (Connect Brute Force
- Ausprobieren von Standard-Passwörtern (Default Passworts)
- Ausprobieren von Wörterbüchern (Dictionary Attacks)
- Ausprobieren von beliebigen Zeichenfolgen (Brute Force)
- Permutationen
- Listen und Wörterbücher in verschiedenen Sprachen

Übliche Gegenmaßnahmen

- Verbergen von Hashes durch Zugriffsbeschränkungen auf USER$, USER_HISTORY$, EXU..$ - Tabellen
- Export- und DataPump-Dateien können ebenfalls Hashes enthalten
- Zugriff aus Data-, Reno-, und Archive Logfiles einschränken
- CREATE ANY Privilegien einschränken
- Verwendung komplexer Passwörter
- Abschalten älterer Logon Protokolle: SQLNET.ALLOWED_LOGON_VERSION=12 (erfordert Upgrade auf 11.2.0.3)
- Abschalten des 11g-Logon-protokolls und Zurückstellen auf 10g (SEC_CASE_SENSITIVE_LOGON=FALSE)
- Verwendung einer Password Verify Function. Als Ausgangspunkt kann die 12c Funktion utlpwdmg.sql dienen. Sie sollte allerdings angepasst und durch Wrappen unkenntlich gemacht werden. ALTER USER IDENTIFIED BY VALUES kann die Verify-Funktion allerdings umgehen.

Passwörter in Oracle Wallets

Oracle bietet mit den Wallets einen PKCS#12-basierten Container an, in dem neben Passwörtern auch SSL-Zertifikate hinterlegt werden können. Hat ein Anwender oder ein Skript Lese-Zugriff auf die

Wallet-Datei und gegebenenfalls das Wallet-Passwort, kann er das Zertifikat auslesen und sich damit an der Datenbank anlegen. Eine weitere Passwort-Abfrage ist nicht mehr nötig.

Um auch automatisiert mit Wallets arbeiten zu können, gibt es die Auto-Login-Funktion. Wird diese aktiviert, wird eine zusätzliche Datei im Wallet erzeugt, die sogenannte Single Sign On Datei (.sso). Diese ist ebenfalls verschlüsselt, aber nicht mit einem benutzerdefinierten Passwort, sondern mit einem Standard-Passwort. Auto-Login-Wallets werden in der Regel verwendet, um eine automatisierte Anmeldung an der Datenbank durchführen zu können, ohne dass ein Passwort im Klartext in Skripten, Konfigurationsdateien oder Umgebungsvariablen abgelegt werden muss.

Damit es nicht möglich ist, ein Wallet einfach zu kopieren und von einem anderen Rechner aus zu verwenden, gibt es die Auto-Login-Local-Funktion. Wird diese für ein Wallet eingestellt, so kann das Auto Login Wallet nur auf dem Rechner verwendet werden, auf dem es erzeugt wurde.

Sicherheit und Einführungsvorraussetzungen

Leider hat die Auto Login Funktion den Nachteil, das das Passwort, mit dem die SSO-Datei verschlüsselt ist, ja irgendwo herkommen muss. Entweder ist es immer gleich, oder…. es ist irgendwo in den Wallet-Dateien versteckt.

Tatsächlich ist es einfach im Header der cwallet.sso-Datei selbst abgelegt:

```
0x00 - 0x4C     Header:
    0x00 - 0x02     First 3 bytes are always A1 F8 4E (wallet recognition?)
    0x03            Type = SSO: 36; LSSO: 38
    0x04 - 0x06     00 00 00
    0x07            Version (10g: 05; 11g:  06)
    0x08 - 0x0A     00 00 00
    0x0B - 0x0C     11g: always the same (41 35)
    0x0D - 0x1C     DES key
    0x1D - 0x4C     DES secret (DES -> CBC -> PKCS7 padding) which contains
the PKCS#12 password
    0x4D - EOF      PKCS#12 data (ASN.1 block)
```

Das Auslesen des internen Schlüssels eines Oracle AutoLogin Wallets ist mit einfachen Mitteln möglich, beispielsweise mit dem Tool ssoDecrypt. Danach kann ein neues, vollständiges Wallet (.p12-Datei) erzeugt und mit dem erhaltenen Schlüssel geöffnet werden. Auch der Kopierschutz ist nun nicht mehr wirksam.

```
$ ./ssoDecrypt.sh ../PX-Linux11/cwallet.sso
sso key: c29XXXXXXXXX96
sso secret: 71c61e1XXXXXXXXXX99c77d747fa0f53e79ccd170409964b
p12 password (hex): 1e482XXXXXXXXXX1f1f0b296f6178021c
```

Auto-Login-Local-Wallets funktionieren über einen String, der mit HMAC SHA1 gehashed wird, um das PKCS#12-Passwort zu erhalten. Der String besteht aus dem Benutzernamen und dem Servernamen des Rechners, auf dem das Wallet erstellt wurde.

Die Wallets müssen auf den Datenbanken und allen teilnehmenden Clients verteilt werden, was einen beträchtlichen administrativen Aufwand erfordert (siehe unten).

LDAP

Grundvoraussetzung für den Einsatz von Verzeichnisservern, sowohl für LDAP, Kerberos der SSL, ist die Einführung von individuellen Datenbank-Anmeldungen für jeden Anwender, Administrator und (Applikations-)Server, soweit noch nicht geschehen. Der Vorteil ist, dass Zugriffe innerhalb der Datenbank klar identifiziert und Rechte feingranular auf Benutzerebene erteilt werden können. Werden Sammel-Konten verwendet, beispielsweise, indem sich alle Administratoren als SYSTEM anmelden, kann nur über Umwege oder gar nicht ermittelt werden kann, welche Person sich hinter der jeweiligen Anmeldung verbirgt. Die so erteilten Berechtigungen lassen sich später nicht mehr auf individueller Basis entziehen, denn jeder Anwender kann sich die Passwörter gemerkt haben. Das Passwort eines solchen geteilten Kontos muss also nach jeder Änderung im Personalstamm geändert werden, was wiederum zu erheblichem Aufwand führt, ohne das Grundproblem zu beheben.

Eine bessere Methode ist das Einführen individueller Konten für jeden zugreifenden Applikationsserver und jeden Benutzer in der Datenbank. Diese können entweder als lokaler Datenbank-Benutzer oder über Enterprise User Security (EUS) in einem Verzeichnis angelegt werden. Die mit dem bisherigen Sammelkonto verbundenen Rechte für den Zugriff auf Datenbankobjekte (Beispielsweise Tabellen, Views oder Packages) werden nun durch entsprechende Berechtigungen (grants) innerhalb der Datenbank erteilt. Zur Vereinfachung werden hier Rollen verwendet. Dies geschieht nach dem Grundsatz, dass die Tabellen selbst möglichst immer einem Schema-Owner gehören, der selbst möglichst kein Anmelderecht (CONNECT) besitzt. Alle Zugriffe erfolgen durch entsprechende Rechte (SELECT, UPDATE, INSERT...) an die entsprechende Rolle (etwa Entwickler, Administrator, Anwender) und darüber an den Benutzer selbst. Die Rechte-Einschränkung funktioniert nur, wenn der Schema-Owner kein Anmelderecht mehr besitzt, da die Rechte auf Objekte, die einem Konto gehören, in der Oracle Datenbank generell nicht eingeschränkt werden können. Die Software-Installation, also das massenweise Verändern der Datenbankobjekte des Schema-Owners, kann über ein Mitglied der DBA-Rolle erfolgen. Dieses kann, falls es erforderlich ist, das Schema-Owner-Konto auch temporär für die direkte Anmeldung freischalten (den Account entsperren). Im Rahmen dieser Änderung werden zu verwendende Objekte zukünftig vollständig angegeben ("select * from **scott.**emp"). Sollte eine bestehende Anwendung die Objekte zwingend im gleichen Schema erwarten, kann dies über öffentliche Synonyme behandelt werden. Diese ermöglichen es jedem Konto, die Objekte, für die Synonyme eingerichtet sind, weiterhin mit relativen Namen anzusprechen ("select * from emp").

Transportverschlüsselung über SSL

Um den Sicherheitsgewinn der Auslagerung der Authentifikation aus der Datenbank ausschöpfen zu können, sollte die Verbindung zwischen Client und Datenbank über das Netzwerk verschlüsselt werden. Hierfür wird SSL verwendet. Für die Datenbank und die Oracle-Clients werden Zertifikate erstellt, entweder selbstsigniert oder mit der unternehmensweiten PKI.

Die Transportverschlüsselung schützt gegen unerlaubtes Mitschneiden des Datenflusses zwischen Anwender-PC bzw. Applikationsserver und der Datenbank und wäre als Grundschutz-Maßnahme auch ohne die Umsetzung der hier beschriebenen Konzepte auch alleinstehend sinnvoll. Die Verwaltung der Zertifikate erfolgt mit Hilfe von Oracle-Wallets, die auf dem Client bereits vorhanden sind. War früher die Advanced Security Option (ASO) als Oracle-Lizenz erforderlich, ist dies mittlerweile nicht mehr erforderlich.

Einrichtung

Die Anbindung der Datenbank an AD als LDAP-Server wirft das Problem auf, wo die Passwort Hashes für die Oracle Datenbank gespeichert werden sollen. Ein AD-Server erzeugt ersteinmal nur seine eigenen (NTLM) Passwort-Hashes. Die Oracle-Hashes können im AD abgelegt werden, dazu

muss das AD-LDAP-Schema allerdings erweitert und auf dem AD-Server eine zusätzliche Software-Komponente installiert werden. Dies ist in der Regel im Unternehmen schwierig durchzusetzen. Alternativ kann ein weiterer Verzeichnisserver neben den AD-Server gestellt werden, der die Schema-Erweiterungen für Oracle, den sogennten OracleContext, verwaltet. Hierfür wird in der Regel das Produkt Oracle Universal Directory (OUD) verwendet. Auch die Enterprise User Security Informationen können hier abgelegt werden.

Für Details zu dieser Installation in WebLogic und 12c siehe: Configure EUS with OUD, AD and DB12c.

Die wesentlichen Schritte für die Registrierung der Oracle Datenbank an einem LDAP-Server lassen sich im Datenbank-Konfigurationsassistent DBCA vornehmen. Zusätzlich muss die TNS- und SQLNet-Konfiguration der Datenbank geändert werden. Die Kommunikation zwischen Datenbank und LDAP-Server sollte wie die zwischen Datenbank und Clients SSL-verschlüsselt werden.

Anwendung

Danach kann sich ein Benutzer anmelden, dessen Anmeldeverfahren als GLOBAL oder EXTERNAL im LDAP-Namensraum eingerichtet worden ist.

Eine noch zentralere Struktur wird durch Enterprise User Security (EUS) erreicht. Hier werden in der Datenbank globale Rollen definiert, die mit einem Gruppen-Element im Verzeichnis korrespondieren. Ein im LDAP angelegter Benutzer kann sich in einer EUS-Umgebung direkt mit seinen LDAP-Credentials an der Datenbank anmelden und ist dort als ein bestimmer, der Verzeichnisgruppe zugewiesener globaler Benutzer eingeloggt. Dieses Verfahren kommt mit einer minimalen Konfiguration in der Datenbank selbst aus, lediglich die globalen Rollen müssen einmalig angelegt werden. Über den einzelnen Benutzer selbst muss in der Datenbank keine Information konfiguriert werden.

Einrichtungsaufwand in der Praxis

Ist ein Oracle-kompatibler LDAP-Server bereits vorhanden und ein OracleContext bereits vorhanden, ist das Einbinden einer weiteren Datenbank eine Sache von unter einer Stunde. Hinzu kommt das Einrichten der globalen oder externen Benutzer, bzw. das Ändern von lokalen Benutzern auf eine externe Authentifikation. In größeren Umgebungen mit vielen ähnlichen Datenbanken lassen sich diese Aufgaben gegebenenfalls skripten.

Das Einbinden einer weiteren Datenbank in eine bereits existierende Umgebung mit LDAP-Infrastuktur (OUD) und Enterprise User Security (EUS) sowie entsprechenden globalen Rollen kann mit einem einzelnen Skriptaufruf in Minuten passieren.

Muss die gesamte LDAP- und EUS-Struktur erst aufgebaut werden, erreicht das Projekt je nach Redunganzanforderungen an die Directory-Server leicht eine Laufzeit von Wochen und Monaten - insbesondere, wenn die Verzeichnisserver hoch verfügbar und mit einer gegenseitigen Replikation aufgebaut werden sollen. Wird (ausschließlich) EUS mit globalen Rollen verwendet, kann sich kein Anwender mehr anmelden, wenn die Verzeichnisserver nicht erreichbar sind.

Angriffsvektoren

Wird die SSL-Transportverschlüsselung verwendet, sind keine erfolgreichen Angriffsmethoden über das Netzwerk mehr bekannt. Ein Angriff, der das Auslesen und Knacken der Passwort Hashes zum Ziel hat, ist ebenfalls sinnlos, da sich keine Passwort Hashes mehr in der Datenbank befinden.

Ohne Transportverschlüsselung ist ein Angriff auf die Session Keys möglich, wenn im LDAP-Verzeichnis SHA1-Passwörter abgelegt sind.

Kerberos

Funktionsweise

Die Anmeldung für die einzelnen Benutzer ist mit im Active Directory vorhandenen Konten möglich. Hierzu werden die Datenbankserver in die Active-Directory-Domäne aufgenommen. Die Authentifikation der einzelnen Benutzer erfolgt über vom AD-Server gestellte Kerberos-Tickets. Die Datenbank-Konten werden "*externally*" authentifiziert angelegt.

Vorbedingung für die Lösung ist, das sämtliche Anwender, aber auch sämtliche Datenbanken, Datenbank-Server, Applikations-Server in das Active Directory Verzeichnis eingetragen werden. Auch für alle Skripte oder Programme, die sich an der Datenbank anmelden sollen, müssen entsprechende technische Benutzerprofile im Verzeichnis angelegt werden.

Einrichtung

Zunächst muss im Active Directory ein SPN/Benutzer für die Datenbank eingerichtet werden. Dann wird eine Service Key Table für den Datenbank-Server mit Hilfe des KTPass-Kommandos angelegt:

```
PS C:\Users\Administrator> ktpass.exe -princ
oracle/ioaotow01.tested.lcl@TESTED.LCL -mapuser ioaotow01 -crypto RC4-HMAC-
NT -pass XXX -out c:\ioaotow-hmac2.keytab -ptype KRB5_NT_PRINCIPAL
```

Für die Nutzung zur Datenbank-Anmeldung müssen einige Parameter in der TNS-Konfiguration der Datenbank selbst angepasst die Kerberos-Konfigurationsdatei *krb5.conf* angelegt werden. Dann ist ist es möglich, einen Datenbank-Benutzer anzulegen, der über Kerberos authentifiziert wird:

```
SQL > create user "TEST" identified externally as 'TEST@LOOPBACK.ORG';
SQL > grant CONNECT TO TEST;
```

Anschließend kann die Anmeldung getestet werden:

```
[oracle~]$ oklist Kerberos Utilities for Linux: Version 11.2.0.4.0 –
Ticket cache: /tmp/krb5cc_501
Default principal: test@LOOPBACK.ORG
Valid Starting           Expires              Principal
22-Aug-2015 11:24:27     22-Aug-2015 19:17:27 krbtgt/LOOPBACK.ORG@LOOPBACK.ORG

[oracle~]$ sqlplus /
SQL >show user
USER is TEST
```

Damit ist die Datenbank für AD-Benutzer an der Datenbank ermöglicht. Für Details zur Konfiguration von Kerberos in der Datenbank 12c siehe: *Neuer Kerberos Stack: AD Authentifikation für Datenbank 12c*

Anwendung

Die Datenbank-Anmeldung über Kerberos ist eine für die Datenbank externe Authentifizierung über das Betriebssystem. Meldet sich der ein Benutzer an seinem Arbeitsplatz (-PC) an, erhält er ein

Kerberos-Ticket, das eine passwortlose Anmeldung an entsprechend eingerichteten Datenbanken ermöglicht. Für im Hintergrund laufende Prozesse, etwa auf Applikationsservern, ist es notwendig, das das Kerberos-Ticket vom Betriebssystem zur Verfügung gestellt (also vom AD-Server geholt) und aktuell gehalten wird. Außerdem müssen, wie oben beschrieben, entsprechende technische Konten im AD angelegt werden, die den Betriebssystem-Konten (Unix-Useraccounts) entsprechen, unter denen die Applikationen laufen.

Einrichtungsaufwand

Die Einbindung einer Datenbank an eine bestehenden Kerberos-Infrastruktur mit Windows Active Directory Servern selbst ist zwar etwas "tricky", wenn das Verfahren aber einmal entwickelt worden ist, in wenigen Stunden erledigt.

Der Aufbau der AD-Infrastruktur und die Einbindung der Windows Arbeitsplätze in die Domäne ist sicherlich nicht für ein Datenbank-Sicherheits-Projekt planbar. die Kerberos-Anbindung lohnt sich nur, wenn diese bereits vollständig vorhanden ist.

Soll EUS verwendet werden, gilt das im Abschnitt LDAP geschriebene: Je nach Redundanz-Anforderungen ist sicherlich eine mehrwöchige Aufbauphase einzuplanen. Statt EUS können auch Skripte zum Einsatz kommen, die periodisch für jeden im Verzeichnis eingetragenen Benutzer extern authentifizierte Datenbank-Benutzer anlegen. Diese können mit Linux-Bordmitteln (Puppet) ausgerollt werden.

Der Aufbau eines Proof Of Concept in einem Linux-Labor mit Red Hat FreeIPA als Kerberos-Server, Windows-PCs und Linux-Workstations als Arbeitsplätze und Linux DB12c Datenbank hat ohne Vorbereitung eine Woche gedauert.

Angriffsvektoren

Angriffe auf Datenbank Hashes und Sesion Keys sind gegenstandslos, da sich keine Hashes in der Datenbank befinden. Die oben erwähnten Angriffe auf Session Keys funktionieren ebenfalls nicht, da keine Session Keys verwendet werden, sondern Kerberos-Tokens, die im Client-Betriebssystem vorhanden sind und an die Datenbank weitergereicht werden.

Für einen erfolgreichen Angriff muss das Kerberos-Ticketsystem angegriffen werden. Auf Kerberos 5 im Allgemeinen und die Windows-Implementation im Besonderen sind eine Reihe von Angriffen bekannt:

- Mimikatz Golden Ticket Attack
- Pass the Hash / Pass the Ticket
- CVE-2014-6324 (Hash Vulnerability)
- Time Stamp Replay Attack

Die Sicherheit der Datenbank-Anmeldungen liegt hier aber im Betriebssystem bzw. wird durch die Domänen-Sicherheit verantwortet und muss nicht separat vom Sicherheits-Team im Datenbank-Team betrachtet werden - alles dies stellt in der Praxis vieler Unternehmen einen beträchtlichen organisatorischen Vorteil dar.

SSL-PKI

Die Authentifikation über Active Directory hat den Nachteil, dass jeder Anwender zwar nur noch ein Passwort hat, dieses auch an einer zentralen Stelle verwaltet wird, aber eben immer noch eingegeben werden muss. Eine ganz andere, auch ergänzend einsetzbare Möglichkeit, Authentifizierung aus der Datenbank selbst auszulagern, besteht in der Nutzung einer Public Key Infrastruktur (PKI). Hierfür müssen Benutzerzertifikate erstellt und verteilt werden, die von einer zentralen Autorität als gültig erklärt und signiert worden sind.

Architektur

Eine Public Key Infrastructure (PKI) regelt die Verteilung von Vertrauen in einer Organisation. Eine zentrale Stelle, die sogenannte Certification Authority (CA), beglaubigt die Identitäten von Mitarbeitern, Maschinen oder auch die Authentizität von Dokumenten, indem sie diese digital unterschreibt (signiert). Die CA besteht aus einem Handling-Mechanismus beziehungsweise damit betrauten Mitarbeitern sowie einem öffentlichen Wurzel-Zertifikat und einem geheimen Schlüssel. Mit diesem Schlüssel signiert sie die Zertifikate der zu beglaubigenden Dokumente. Mit Hilfe des öffentlichen Wurzelzertifikates (Root-CA Zertifikat) kann die Echtheit der Unterzertifikate von allen beteiligten Parteien geprüft und bestätigt werden. Ihre Autorität kann die CA auch delegieren, um zum Beispiel das Ausstellen von Zertifikaten für einen Unterbereich der Organisation von einer untergeordneten CA durchführen zu lassen. Hierzu erhält diese Sub-CA ein Zwischenzertifikat der Root-CA. Mitarbeiter können ein eigenes Zertifikat, das von der CA unterschrieben wird, erhalten. Es existieren eine Reihe von Software-Produkten für die Implementation einer PKI. Ebenso ist es möglich, den Prozess mit Hilfe von OpenSSL komplett ohne kommerzielle Software abzubilden. Außerdem existieren mehrere Open Source PKI Lösungen. Oft ist die CA bereits durch die AD Domänen Administration vorhanden und in Betrieb.

SSL-PKI Anmeldung an der Datenbank mit Wallets

Oracle bietet mit den *Wallets* einen PKCS#12-basierten Container an, in dem neben Passwörtern auch SSL-Zertifikate hinterlegt werden können. Hat ein Anwender oder ein Skript Lese-Zugriff auf die Wallet-Datei und gegebenenfalls das Wallet-Passwort, kann er das Zertifikat auslesen und sich damit an der Datenbank anlegen. Eine weitere Passwort-Abfrage ist nicht mehr nötig.

Für eine zertifikatsbasierte Anmeldung muss zunächst auf dem Datenbank-Server ein Wallet angelegt, dann ein Request erstellt und exportiert werden. Nach dem Signieren des Requests durch die CA wird das gelieferte User Certificate zusammen mit der Root-CA, dem sogenannten Trusted Certificate, importiert. Als nächstes werden auf dem Datenbank-Client (PC oder Applikationsserver) ebenfalls ein Wallet und ein Request erzeugt und exportiert. Nachdem die CA den Request signiert hat, können User und Trusted Certificate auch hier importiert werden.

Nun muss das Wallet zur Benutzung noch in der TNS-Konfiguration von Datenbank und Client eingetragen werden. Außerdem sind einige weitere Einstellungen in sqlnet.ora, tnsnames.ora und listener.ora erforderlich. Für Details zu dieser Konfiguration siehe: Enterprise Security mit LDAP und PKI

Benutzer müssen als EXTERNAL oder GLOBAL angelegt werden, um sich anmelden zu können:

```
SQL> create user JANS identified externally as 'CN=jans';
SQL> grant create session to JANS;
```

Nun kann sich der User, der Zugriff auf das oben angelegte Client Wallet besitzt, direkt anmelden:

```
$ sqlplus /@DB12C
Connected.
SQL>
```

Die Verbindung zur Datenbank kann mit sqlplus, OCI oder JDBC erfolgen. Die Konfiguration des SQL Developer mit SSL-Wallets ist in MOS Note: 401251 beschrieben.

Wallet- und Zertifikats-Verteilung

Die SSL Zertifikate für Transportverschlüsselung und SSL-Authentifikation (PKI) werden in Wallets abgelegt. Diese Wallets müssen ausgerollt und gewartet werden. Für SSL-Transportverschlüsselung und Server-Authentifikation muss mit der Installation einmalig ein Wallet eingerichtet werden, dass den Server authentifiziert. Für die Benutzer-Authentifikation muss ein individuelles Zertifikat vorhanden sein, wenn der Anwender, beispielsweise der Entwickler, sich an der Datenbank anmelden möchte. Die pro Rechner notwendigen Oracle Wallets werden mit der Installation der Oracle-Software zusammen angelegt. Sie sind statisch und bleiben über die Lebensdauer der Oracle-Software-Installation gleich. Die Zertifikate für die Oracle-Installation einhalten den Request (und damit den Schlüssel) für die Identität des Rechners, auf dem sie installiert sind, und das von der CA ausgestellte Zertifikat. Der Prozess der Zertifikatsanforderung wird nach der Installation der Oracle Software wie oben beschrieben durchgeführt. Die notwendigen *ORAPKI*-Aufrufe können als ein Skript bereitgestellt werden. Individuelle Zertifikate für Entwickler und Administratoren werden von der CA bereitgestellt. Es ist unter Umständen möglich, sie in den Oracle Wallets zu referenzieren. Hierzu wird im Wallet ein Pseudo-Eintrag hinterlegt, der auf das im Betriebssystem vorhandene Zertifikat verweist. Im Fall eines PKCS#11-Zertifikates funktioniert dies beispielsweise so:

```
orapki wallet p11_add -wallet . -p11_lib "c:\Windows\System32\opensc-
pkcs11.dll" -pwd Oracle1234
```

Domänen-Windows-Clients bekommen bei der Anmeldung an der AD-Domäne automatisch Zertifikate zugewiesen. Diese sind im Windows-Zertifikatsstore zu finden. Für die Benutzer werden zur Zeit noch keine Zertifikate verteilt. Dies ist aber technisch möglich und kann in der AD-Konfiguration eingeschaltet werden. Zum Erlangen eines Zertifikates muss es im Windows Certificate Manager des Clients entsprechend angefordert werden. Der Prozess kann so eigerichtet werden, das er ohne Verwaltungstätigkeit des AD-Teams funktioniert. In der AD CA kann auch konfiguriert werden, das ein Benutzerzertifikat automatisch erstellt wird (Autoenrollment). Für die Linux-Clients in der Entwickler-Cloud funktioniert das AutoEnrollment an der PKI zur Zeit nicht. Hier muss eine separate Lösung gefunden werden. Eventuell eignet sich ein Ansatz über das Linux Identity Management System mit Hilfe der DogTag CA als Subsidiary der PKI CA.

Bei der Notwendigkeit einer Neuverteilung von Wallets oder Zertifikaten können vonhandene Werkzeuge zum Rollout verwendet werden. Hier bieten sich Puppet und Cloud Control (Enterprise Manager) an.

Aufwand für die Installation

Die Einrichtung der SSL-Infrastruktur, der CA in allen Arbeitsplätzen ist erheblich. An jedem Arbeitsplatz muss ein Oracle-Wallet installiert werden, in dem die Zertifikate der CA importiert werden, sowie ein Zertifikat erstellt werden, das den Benutzer ausweist. Dieses muss von der CA signiert werden. Wir haben im vorgestellten Projekt keine Softwareverteilungs-Lösung gefunden, die das in Windows ausrollen kann. Der Weg, die CA von Active Directory selbst zu verwenden, war am praxistauglichsten. Hier müssen die entsprechenden Regeln zum Erstellen von Benutzer-Zertifikaten eingestellt werden.

Die Einrichtung der Wallet-Infrastruktur in den Datenbanken lässt sich in wenigen Schritten durchführen oder auch automatisiert skripten. Die Zertifikate im Datenbank-Wallet müssen ja nach Ablauffrist erneuert werden. Auch dies lässt sich beispielsweise mit Puppet automatisieren.

Angriffsvektoren

Angriffe auf Hashes und Session Keys sind hier ebenfalls gegenstandslos.

Möglich ist ein Angriff auf das Database Wallet. Ist dieses nicht gegen Veränderung auf Dateisystem-Ebene gesichert, kann die komplette CA ausgetauscht werden, indem ein neues Wallet implantiert wird. Danach kann sich der Angreifer mit selbst ausgestellten Credentials (Schlüsseln) der gefälschten CA anmelden. Um dies zu verhindern, können Hardware-Sicherungsmodule (HSMs) eingesetzt werden, die den Zugriff auf das Wallet physikalisch erschweren. Dies geschieht in der Regel über Smart-Cards.

SmartCards

Smartcards können verschiedene Informationen auf einem Medium speichern, das möglichst nicht direkt ausgelesen werden kann. Damit lassen sich die oben beschriebenen SSL-Zertifikate besonders schützen. Interessant ist eine Smart-Card-basierte Oracle-Anmeldung vor allem, wenn ein SmartCard-basierter Mitarbeiterausweis im Unternehmen bereits eingeführt ist. Ist dies nicht der Fall, wird alleine die Anmeldung an den Datenbanken in den wenigsten Fällen die Einführung von Smart-Cards für deren Anwender und/oder Administratoren rechtfertigen können.

Aufwand

Leider sind die Treiber für Smart-Cards, die sogenannte *Middleware*, durch jeden Hersteller unterschiedlich implementiert und daher ist die Installation im PC-Betriebssystem nicht ganz einfach. Wird über eine Smart-Card-Einführung nachgedacht, ist es auf jeden Fall zielführend, zu untersuchen, ob nicht die Mehrzahl der in der Organisation vorhandenen Anwendungen auf einer Karte vereint werden können - beispielsweise Zugangskontrolle für das Gebäude, Zugriff zum (Virtual-) Desktop, Windows/Linux-Anmeldung und (SSL-)Schlüsselbund.

Der Installationsaufwand für das Einrichten der Oracle-Authentifikation in einer bereits bestehenden Smart-Card-Umgebung mit bereits auf den Karten vorhandenen Zertifikaten zur Anmeldung in Windows entspricht der der oben beschriebenen SSL-Einrichtung plus der Anbindung der Windows-Treiber.

Sicherheit & Angriffsvektoren

Smartcard-basiertes SSL verbindet die Vorteile von passwortlosen Authentifizierungsverfahren (keine Hashes und Session Keys) mit einer physikalischen Sicherungsschicht auf die CA und bietet die höchste Sicherheit der hier vorgestellten Verfahren.

Fazit

Die vorgegebenen Einstellungen für das Erzeugen von Passwort-Hashes in der Oracle Datenbank sind nicht immer glücklich gewählt. Gelingt es einem Angreifer, die Hashes der Passwörter, die nicht besonders geschützt in der Datenbank liegen, auszulesen, ist es in der Regel möglich, die Original-

Passwörter zu errechnen. Noch verheerender wirken sich Angriffe auf den Verbindungsaufbau zur Datenbank aus.

Auch in Oracle Passwort Wallets oder anderen Passwort-Managern sind Passwörter nicht sicher aufgehoben.

Eine zusätzliche Gefahr ergibt sich aus der mehrfachen Nutzung gleicher Passwörter für verschiedene Zugänge, die unter Umständen für unterschiedliche Sicherheitsklassen funktionieren. Das Management von verschiedenen Zugangskennungen pro Person ist hier komplex und hochproblematisch.

Eine Alternative bieten zentrale Identitätsverwaltungs-Systeme. In der hier vorgenommenen Analyse wurden ActiveDirectory/LDAP, Kerberos und SSL/Smart-Cards in der Anwendbarkeit für Entwickler-Zugriffe und die Authentifikation von maschinellen Zugriffen verglichen. Für den Entwicklerzugriff hat die Identifikation über Kerberos/Active Directory die besten Ergebnisse erzielt, für die Absicherung der maschinellen Zugriffe haben sich SSL-Zertifikate als am interessantesten herausgestellt. Ihr gesamtes Potenzial entfalten diese Systeme allerdings erst zusammen mit der zentralen Verwaltung von Rechten und Rollen mit Oracle Enterprise User Security.

Für die Sicherheit von Anmeldungen ist eine SSL-PKI auf SmartCards erste Wahl. Aber auch mit Kerberos in Active Directory lassen sich sehr gute Lösungen aufbauen - mit deutlich weniger Aufwand, aber etwas größerer Angriffsoberfläche im Betriebssystem. Für maschinelle Anwendungen sind SSL-Zertifikate in Datenbank-Wallets gut geeignet, da kein Passwort eingegeben werden muss.

Soll kein Microsoft Active Directory Kerberos verwendet werden, ist FreeIPA in Linux eine gute Alternative.

Weiterführende Literatur

- Trustwave: Changes in Oracle Database 12c password hashes
- Oracle: Oracle Advanced Security Technical White Paper
- Esteban Martínez Fayó: Cryptographic flaws in Oracle Database authentication protocol
- Pete Finnigan: Oracle Database Password Security
- SkullSecurity: Padding oracle attacks: in depth
- Marcel Lambrechts: Cryptographic flaws in Oracle Database authentication protocol
- Oracle Database 11g stealth password cracking vulnerability in logon protocol (CVE-2012-3137)
- Oracle: Oracle DB 11.2 Database Security Guide
- Jan Schreiber: Enterprise Security mit LDAP und PKI – Varianten der zentralen Benutzerverwaltung für Oracle Datenbanken, DOAG Konferenz 2015
- Oracle Database Security: Wie viel darf es denn sein?, Stefan Oehrli 2012
- Oracle: Technology Global Price List
- Oracle: Unified Directory Overview
- Oracle Enterprise User Security mit Active Directory, Jürgen Kühn 2009
- Oracle DB Security: Enterprise User Security für DBAs (EUS4DBAS), Carsten Mützlitz 2014
- RFC#5280, X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, 2008
- Defending Against Pass-the-Hash Attacks, Microsoft Security Intelligence Report
- Why Crack When You Can Pass the Hash,
- Abusing Microsoft Kerberos - Sorry you guys don't get it
- Protection from Kerberos Golden Ticket
- Red vs. Blue: Modern Active Directory Attacks, Detection, and Protection Whitepaper
- Wikipedia: PKCS
- PKCS #11: Cryptographic Token Interface Standard, RSA 2009

- SSL With Oracle JDBC Thin Driver, An Oracle Technical White Paper, April 2010
- USING THE ORACLE ODBC DRIVER WITH SSL (Doc ID 1064404.1)
- Oracle Wallets hacken, Loopback Blog, Jan Schreiber 2015
- Automatic Renewal of Kerberos Tickets (LinuxQuestions)
- Kerberos für die Datenbank - Dr. Günter Unbescheid
- A-Team: Configuring your Oracle Database for Kerberos authentication
- Kerberos - Single Sign On ganz einfach, Trivadis, Jürgen Kühn 2011
- Loopback Wiki: Oracle Database 12c Kerberos Configuration
- Loopback Blog: Neuer Kerberos Stack: AD Authentifikation für Datenbank 12c
- Loopback Wiki: Configure EUS with OUD, AD and DB12c

Kontaktadresse:
Jan Schreiber
Loopback.ORG GmbH
An der Alster 83
D-20099 Hamburg

Telefon:	+49 (0) 40-2263236 0
Fax:	+49 (0) 40-2263236 99
E-Mail	info@loopback.org
Internet:	www.loopback.org