Oracle 18c mit neuen Datenbank-Security-Features

Norman Sibbing, ORACLE Deutschland B.V. & Co. KG

Oracle stellt mit der Änderung der Release-Strategie, beginnend im Jahr 2018, jedes Jahr neue Funktionalitäten zur Verfügung. Das aktuelle Release 18c bietet auch hinsichtlich Datenbank-Security ein paar Neuerungen. Der Artikel zeigt die interessantesten neuen Security-Features anhand von Beispielen.

Sämtliche Lisitings zu dem Artikel finden Sie als Datei unter dem folgenden Link:

www.doag.org/go/red_stack/201804/sibbing/listings

Schema ohne Passwort

Warum muss eigentlich jedes Schema in einer Oracle-Datenbank eine Authentifizierungsmethode haben? Es gibt doch auch Anwendungsfälle, bei denen es nur darauf ankommt, Datenbank-Objekte abzulegen. Ein schlichter Objekt-Container würde dabei ausreichen, ohne dass die Möglichkeit besteht, sich direkt dagegen verbinden zu können. Bisher musste bei der Erstellung eines Schemas zwingend eine Authentifizierungsmethode angegeben werden. Folgende Methoden stehen zur Verfügung:

- *Authentifizierung durch das Betriebssystem (OS-Authentifizierung, Kerberos, PKI)*
  CREATE USER Benutzer IDENTIFIED EXTERNALLY;

- *Authentifizierung durch die Datenbank (Passwort)*
  CREATE USER Benutzer IDENTIFIED BY 'Geheim';

- *Authentifizierung über ein Identity Management System (Enterprise User Security)*
  CREATE USER Benutzer IDENTIFIED GLOBALLY;

Doch was ist, wenn man keine Authentifizierung für dieses Schema haben möchte oder darf? Dafür gab es bislang nur Workarounds. Einer ist die Verwendung eines sehr langen und kryptischen Passworts „-Jm44!!j2h(7hsdtxl*psub+JTDl-„, das dann auch nach der Vergabe direkt wieder vergessen werden sollte. Eine andere Möglichkeit bestand darin, über die „CREATE USER HR IDENTIFIED BY VALUES"-Klausel einen ungültigen HASH-Wert anzugeben. Diese Variante geht allerdings ab der Datenbank-Version 12c nicht mehr, da hier der HASH-Wert überprüft wird. Wird es dennoch versucht, erhält man die Fehlermeldung „ORA-02153: invalid VALUES password string". Auch das Sperren des Schemas ist keine zufriedenstellende Lösung, da ein gesperrtes Schema kein Proxy-Log-in mehr zulässt.

Zum Beispiel sollte auf ein Applikationsschema nicht direkt zugegriffen werden dürfen – weder durch einen Applikationsserver noch durch einen Entwickler oder Datenbank-Administrator. Für den Zugriff der Applikation auf die Applikationsobjekte ist es empfohlen, einen separaten technischen Applikationsbenutzer in der Datenbank anzulegen, der durch Rollen und gegebenenfalls Synonyme Zugriff auf das Applikationsschema erhält. Nur so ist eine Steuerung der Zugriffsrechte möglich. In diesem Fall ist es unerheblich, ob das Applikationsschema gesperrt oder mit einem unbekannten Passwort geschützt ist oder gar kein Passwort besitzt. Bei der letzteren Variante (ohne Passwort) ist, wie erwähnt, kein Proxy-Log-in möglich.

Für Entwickler und Datenbank-Administratoren bietet sich hingegen der Einsatz eines Proxy-Benutzers an. Der wesentliche Vorteil dabei zeigt sich beim Versuch, Datenbank-Benutzer zu personalisieren, obwohl sie letztendlich Shared-Schemata wie „SYSTEM" oder das Applikations-Schema verwenden. Neben dem vereinfachten Rollen-Management – die Rollen werden im Prinzip vererbt – wird der reale Datenbank-Benutzer im Auditing-Trail eingetragen. Und genau darauf zielt das neue 18c-Datenbank-Feature „Schema Only Accounts" ab. Damit besteht die Möglichkeit, keine Authentifizierungsmethode angeben zu müssen. Mit der Klausel „NO AUTHENTICATION" beim „CREATE USER"- und „ALTER USER"-Statement wird mit „•CREATE|ALTER USER Benutzer NO AUTHENTICATION;" ein Schema Only Account erstellt. Das funktioniert fast mit jedem Datenbank-Benutzer. Ausnahme: Die Klausel „NO AUTHENTICATION" ist bei Datenbank-Benutzern mit Administrations-Privilegien wie „SYSDBA", „SYSOPER", „SYSBACKUP", „SYSKM", „SYSASM", „SYSRAC" und „SYSDG" sowie natürlich bei der Verwendung von Datenbank-Links nicht zulässig.

Das folgende Beispiel zeigt das Feature für die Datenbank-Administration mit dem User „SYSTEM". Aus Sicherheits- und Compliance-Gründen sollte dieser nicht zur Administration verwendet werden. Der Hauptgrund dafür ist, dass entsprechende Aktivitäten keiner realen Person zugeordnet werden können und demzufolge nicht nachvollziehbar sind. Hier bietet sich das neue Feature sehr gut an. Zunächst wird der „SYSTEM"-Benutzer zum Schema Only Account geändert. Er besitzt also kein Passwort mehr und folglich kann sich niemand direkt mit ihm anmelden *(siehe Listing 1)*. Nun wird ein personalisierter Benutzer mit eigenem Passwort angelegt. Der neue Benutzer erhält maximal das „Create Session"-Privileg; zudem ist es ihm erlaubt, den SYSTEM-Benutzer als Proxy-Benutzer zu verwenden *(siehe Listing 2)*. Mehr ist im Prinzip nicht zu tun. Die Anmeldung als personalisierter DBA mit den Rechten des SYSTEM-Benutzers erfolgt dann entsprechend einem Proxy-Log-in *(siehe Listing 3)*.

Schema Only Accounts bieten also eine geniale Möglichkeit, Datenbank-Benutzer auf einfache Weise zu personalisieren. Es werden hiermit gleich mehrere Themen adressiert:

- Benutzer mit „NO AUTHENTICATION" sind nicht mehr direkt nutzbar
- Kein Passwort erforderlich
- Kein Passwort-Profil mehr notwendig
- Jeder auf diese Art personalisierte DBA oder Entwickler verwaltet sein eigenes Passwort
- Im Audit-Trail ist der reale Benutzer enthalten, trotz Verwendung des Shared Account

Das Feature steht in allen Editionen ab Oracle-Datenbank 18c ohne zusätzliche Lizenzen zur Verfügung.

Unified Auditing in SYSLOG und Windows Event Viewer

Die Überwachung spezieller Aktivitäten in IT-Systemen wie Netzwerken, Datenbanken und Betriebssystemen gehört zu den wichtigsten Sicherheitsmaßnahmen in der IT-Sicherheit. Hierbei geht es im Wesentlichen immer um den Nachweis darüber, wer innerhalb der IT-Infrastruktur was wann wo und womit gemacht hat. Die meisten aktiven Komponenten einer IT-Infrastruktur sind in der Lage, bei richtiger Konfiguration Aktivitäten in den von ihnen zur Verfügung gestellten Diensten zu protokollieren. Die Herausforderung ist dabei, die Aktivitätsprotokolle aus diesen unterschiedlichen IT-Komponenten zu konsolidieren, um sie miteinander korrelieren zu können.

Eine dieser IT-Komponenten ist die Oracle-Datenbank; sie war schon immer in der Lage, Aktivitäten zu protokollieren. Dabei werden die durch das Oracle-Auditing generierten Daten in Tabellen und Betriebssystem-Dateien geschrieben. Es besteht auch die Möglichkeit, Datenbank-Protokolldaten in das Unix-SYSLOG beziehungsweise in den Microsoft Event Viewer zu schreiben. Das funktioniert noch bis heute – zumindest für das traditionelle Oracle-Datenbank-Auditing. Mit der Version 12c wurde ein neues Auditing eingeführt, das Oracle Unified Auditing, ein Datenbank-Auditing neben dem traditionellen (alten) Oracle Auditing. Oracle Unified Auditing ist moderner, flexibler und sicherer als das klassische Oracle-Datenbank-Auditing. Hierzu existiert bereits ein Tipp, der das Thema genauer darstellt. Das, was Oracle Unified Auditing bisher nicht konnte, war, die Auditdaten in das Unix-SYSLOG beziehungsweise in den Microsoft Event Viewer zu schreiben.

Diese fehlende Integration führte teilweise dazu, Oracle Unified Auditing nicht zu verwenden, obwohl es wesentlich besser zu kontrollieren ist. Ein Grund dafür ist,

SYSLOG/Event Viewer Name	Spalten im UNIFIED_AUDIT_TRAIL	Beschreibung
TYPE	AUDIT_TYPE	Audit-Typ (Standard, RMAN etc.)
DBID	DBID	Datenbank-ID
SESID	SESSION_ID	Session-ID
CLIENTID	CLIENT_IDENTIFIER	Identifikation des Session-Clients
ENTRYID	ENTRY_ID	Laufende Audit-ID pro Session
STMTID	STATEMENT_ID	Laufende Statement-ID pro Session
DBUSER	DB_USERNAME	Datenbank-Benutzer, dessen Aktionen überwacht wurden
CURUSER	CURRENT_USERNAME	Effektiver Benutzer
ACTION	ACTION	Action-Code (Statement Typ)
RETCODE	RETURN_CODE	ORA-Fehlercode
SCHEMA	OBJECT_SCHEMA	Name des Objekt-Schemas
OBJNAME	OBJECT_NAME	Name des betroffenen Objekts

Tabelle 1: Spalten-Mapping zwischen SYSLOG und Unified Audit Trail

dass viele Kunden Werkzeuge zur Überwachung von Aktivitäten sogenannte „System Information and Event Management"-Tools (SIEM) einsetzen, um die Protokolldaten sämtlicher IT-Komponenten und Services zentral aus dem Unix-SYSLOG beziehungsweise Microsoft Event Viewer zu sammeln, Oracle Unified Auditing aber dort die Daten nicht speichern konnte. Diese sinnvolle Integration ist nun ab der Oracle-Datenbank 18c für Unified Auditing möglich.

Einen kleinen Wermutstropfen gibt es: Nicht alle Informationen, die mit Unified Auditing gesammelt werden, werden auch ins SYSLOG beziehungsweise in den Event Viewer geschrieben. *Tabelle 1* zeigt die Informationen, die zur Verfügung stehen.

Um die Daten des Unified Auditing in das SYSLOG oder in den MS Event Viewer übertragen zu bekommen, sind lediglich zwei Konfigurationen notwendig, eine in der Oracle-Datenbank durchgeführt und eine im Betriebssystem. Zunächst wird die Datenbank vorbereitet. Der ab Oracle DB 18c verfügbare Parameter „UNIFIED_AUDIT_SYSTEMLOG" wird je nach Ziel (SYSLOG oder MS Event Viewer) wie folgt eingestellt:

- *SYSLOG*
  Auswahl der „SYSLOG"-Facility „user: user-level messages" oder „local[0-7]" sowie SYSLOG Severity Level (NOTICE, INFO, DEBUG, WARNING, ERR, CRIT, ALERT, und EMERG)

- *MS Event Viewer*
  TRUE (statt des Defaults FALSE)

Im *Listing 4* (ein UNIX-System) wird der Parameter „UNIFIED_AUDIT_SYSTEMLOG" auf „LOCAL6.NOTICE" gesetzt. Ist das erledigt, muss der System-Administrator („Root") die Konfiguration des SYSLOG durchführen. Im Beispiel wird die Einstellung „local6.notice" mit der LOG-Datei „/var/log/audit_oracle.log" gekoppelt. Dies wird in der Konfigurationsdatei „/etc/rsyslog.conf" beziehungsweise „/etc/syslog.conf" eingetragen. Nach der Änderung nur noch den SYSLOG-Daemon durchstarten und fertig ist die Konfiguration im Betriebssystem *(siehe Listing 5)*. Zum Schluss die Datenbank durchstarten und Einstellungen überprüfen *(siehe Listing 6)*. Nun sollte auch die entsprechende Auditdatei „/var/log/audit_oracle.log" vorhanden sein *(siehe Listing 7)*. Ab sofort werden die oben beschriebenen Audit-Informationen, neben der Speicherung im Unified Audit Trail, auch ins SYSLOG geschrieben *(siehe Listing 8)*. Das Ergebnis lässt sich durch einen privilegierten Betriebssystembenutzer (Root) überprüfen *(siehe Listing 9)*.

In einer Oracle-Multitenant-Umgebung ist bei Bedarf der Parameter „UNIFIED_AUDIT_SYSTEMLOG" in jeder Pluggable Database zu setzten. Hier besteht auch die Möglichkeit, unterschiedliche SYSLOG-Einstellungen zu verwenden, etwa dieselben wie bei der Container-Datenbank *(siehe Listing 10)*. In Beispiel von *Listing 11* schreiben jetzt beide Datenbanken (CDB und PDB1) in dieselbe Auditdatei.

Wie bereits erwähnt, werden nur bestimmte Informationen ins SYSLOG geschrieben. Diese Informationen reichen aber aus, um den gesamten Auditeintrag zu identifizieren. Möchte man zum Beispiel wissen, welche ausgeführte Aktion sich hinter dem Action-Code „227" verbirgt, lässt sich das über die Datenbank-View „audit_actions" abfragen *(siehe Listing 12)*. Detaillierte Informationen über die protokollierte Aktivität stehen im Unified Audit Trail der entsprechenden Datenbank *(siehe Listing 13)*.

Der Action-Code „ACTION:100" zeigt bereits, dass es sich hier um einen Anmeldeversuch (LOGON) handelte. Der Return-Code „RETCODE:1017" gibt an, dass dieser missglückt war. Um nun alle Informationen dieser Aktivität zu erhalten, benötigen wir die Session-ID „SESID:2439550631" und die Statement-ID „STMTID:1", um den Unified Audit Trail der Datenbank abzufragen *(siehe Listing 14)*.

Mit dieser Unified-Auditing-SYSLOG-Integration ist eine Lücke geschlossen, die so manchen davon abgehalten hat, Unified Auditing einzusetzen. Vermutlich sind es dem einen oder anderen noch zu wenig Audit-Informationen, die ins SYSLOG geschrieben werden (wie „SQL_TEXT"). Auch aus diesem Grund ist und bleibt es unumgänglich, die gesamten Informationen aus den Unified Audit Trails der Datenbanken zentral einzusammeln. Eine gute Lösung dafür ist die Verwendung von Oracle Audit Vault und Database Firewall. Im Übrigen besitzen Oracle Audit Vault und Database Firewall eine hervorragende SYSLOG-Integration, die sich über ein Regelwerk gut steuern lässt.

Dieses Feature steht in allen Editionen ab Oracle-Datenbank 18c ohne zusätzliche Lizenzen zur Verfügung.

Verschlüsselung von Anmelde-Informationen im Data Dictionary

Immer häufiger kommen sogenannten „Datenbank-Schwachstellen-Scanner" (Database Vulnerability Scanner) zum Einsatz, um potenzielle Sicherheitsrisiken in Datenbanken aufzuspüren. Die durch diese Werkzeuge entdeckten Schwachstellen und Fehlkonfigurationen lassen sich meist durch entsprechende Sicherheitsmaßnahmen mildern beziehungsweise komplett beheben. Es gibt allerdings auch Dinge, die sich nicht ohne Weiteres mildern oder beheben lassen, da sie systembedingt sind oder waren. Dazu gehören die im Oracle-Dictionary gespeicherten Anmelde-Informationen für Datenbank-Links und -Scheduler. Die hier verwendeten Kennwörter lassen sich über die Data-Dictionary-Tabellen „SYS.LINK$" und „SYS.SCHEDULER$_CREDENTIAL" auslesen. Zur Demonstration dieses Features wird zunächst ein Database Link angelegt *(siehe Listing 15)*. Nun folgt die Abfrage der „LINK$"-Tabelle, um das Kennwort zu erhalten *(siehe Listing 16)*.

Man darf sich von der Darstellung des „PASSWORDX"-Werts nicht täuschen lassen. Er sieht zwar verschlüsselt aus, ist es aber nicht. Zur Speicherung der Anmelde-Informationen von Datenbank-Links und Scheduler-Jobs sind die Kennwörter vielmehr verschleiert (obfuscated) gespeichert. Dabei werden Informationen durch einen rückrechenbaren Algorithmus augenscheinlich unkenntlich gemacht. Es handelt es sich allerdings nicht um einen Verschlüsselungs-Algorithmus im klassischen Sinne, sondern um ein vom entsprechenden Entwickler konzipiertes Verfahren. Ist also das Verfahren einem Dritten bekannt, kann er die über die Data-Dictionary-Tabellen „SYS.LINK$" und „SYS.SCHEDULER$_CREDENTIAL" ausgelesenen Kennwörter lesbar machen. Aus diesem Grund sollte der Zugriff auf diese sensiblen Tabellen weiterhin sehr restriktiv gehandhabt bleiben, obwohl es ab der Version 18c die Möglichkeit gibt, diese sensiblen Informationen mit einem echten Verschlüsselungs-Algorithmus AES256 (Advanced Encryption Standard) zu verschlüsseln. In einer Standard-Installation der Datenbank 18c sind diese Werte weiterhin nur verschleiert.

Verschlüsselung der Anmelde-Informationen von Datenbank-Links und Scheduler-Jobs

Dieses Feature verwendet Funktionalitäten der Transparent Data Encryption (TDE) und steht in allen Editionen ab Oracle 18c zur Verfügung. Eine Advanced-Security-Option-Lizenz ist hier jedoch nicht erforderlich. Der „COMPATIBLE"-Parameter der Datenbank muss dabei auf mindestens 12.2.0.2 gesetzt sein.

Die Verschlüsselung der Anmelde-Informationen in den „SYS.LINK$"- und „SYS.SCHEDULER$_CREDENTIAL"-Tabellen ist ähnlich der Verschlüsselung von Spalten und Tablespaces mittels TDE. Es existiert jeweils ein Schlüsselpaar, bestehend aus einem Master Encryption Key (MEK) und jeweils einem Data Encryption Key (DEK) für die Tabellen „SYS.LINK$" und „SYS.SCHEDULER$_CREDENTIAL". Anders als beim MEK, der extern gespeichert ist, sind die DEKs innerhalb der Datenbank verschlüsselt gespeichert. Der MEK ist zur Ver- und Entschlüsselung der DEKs erforderlich. Mit den so entschlüsselten DEKs können dann die Anmelde-Informationen in den „SYS.LINK$"- und „SYS.SCHEDULER$_CREDENTIAL"-Tabellen gelesen werden. Wie beschrieben, benötigt die Datenbank für diese Funktionalität einen MEK. Das ist derselbe, der beim Einsatz von Transparent Data Encryption erforderlich ist.

Achtung: Wer Transparent Data Encryption bereits einsetzt, muss die nächsten Schritte bis einschließlich der Erstellung des Master Encryption Key überspringen, da für die Datenbank bereits ein Keystore und ein Master Encryption Key existieren. Beim Oracle-Datenbank-Cloud-Service in der Oracle Public Cloud sind der Keystore und der Master Encryption Key ebenfalls bereits vorhanden und es sind keine weiteren Vorbereitungen durchzuführen. Trifft das alles nicht zu, ist zunächst ein Keystore für die Speicherung des Master Encryption Key zu erstellen. Im Standard ist der Keystore eine „PKCS12"-Datei („ewallet.p12"), die in einem Verzeichnis

auf dem Datenbank-Server gespeichert ist. In der Regel liegt der Keystore unter „$ORACLE_BASE/admin/$ORACLE_SID/wallet". Das „wallet"-Verzeichnis ist vorab anzulegen. Es besteht natürlich die Möglichkeit, ein alternatives Verzeichnis (etwa „/u01/oracle/db/orcl/wallet") zu verwenden. Dieses ist in der „SQLNET.ORA" des Datenbank-Servers anzugeben *(siehe Listing 17)*.

Nachdem die „SQLNET.ORA" entsprechend angepasst ist, lässt sich der Keystore erstellen und öffnen. Dafür ist ein „SYSKM"- oder „SYSDBA"-Administrations-Privileg erforderlich *(siehe Listing 18)*. Anschließend kann der Master Encryption Key erstellt werden *(siehe Listing 19)*. Das Ergebnis lässt sich über die View „V$ENCRYPTION_KEYS" überprüfen *(siehe Listing 20)*. Nun sind alle Voraussetzungen zum Verschlüsseln der Anmelde-Information in der „SYS.LINK$"- und „SYS.SCHEDULER$_CREDENTIAL"-Tabelle gegeben.

Damit alle aktuellen und alle zukünftigen Anmelde-Informationen verschlüsselt gespeichert werden, ist einmalig ein neues „ORACLE DDL"-Statement auszuführen. Ab der Oracle Datenbank 18c steht dafür das Statement „ALTER DATABASE DICTIONARY" zur Verfügung. Die Ausführung des Statements ist überschaubar; es existieren lediglich drei Ausführungsvarianten *(siehe Abbildung 1)*.

Das neue Statement lässt sich nur mit „SYSKM"-Administrations-Privilegien ausführen. Dieser Befehl verschlüsselt nun alle Anmelde-Informationen in den „SYS.LINK$"- und „SYS.SCHEDULER$_CREDENTIAL"-Tabellen *(siehe Listing 21)*. Das Ergebnis kann der „SYSDBA" durch Abfrage der „SYS.LINK$"-Tabelle überprüfen *(siehe Listing 22)*. Vergleicht man nun die Werte von „PASSWORDX" mit dem originalen Wert (ohne Verschlüsselung), lässt sich ein Unterschied erkennen *(siehe Listing 23)*.

Besteht ein Grund, die Anmelde-Informationen in den Tabellen „SYS.LINK$" und „SYS.SCHEDULER$_CREDENTIAL" mit einem neuen Data Encryption Key neu zu verschlüsseln, kann dies durch einen einfachen Befehl erfolgen *(siehe Listing 24)*. Durch eine wiederholte Abfrage der „SYS.LINK$"-Tabelle lässt sich die Neu-Verschlüsselung überprüfen *(siehe Listing 25)*. Wenn man die beiden „PASSWORDX"-Werte vergleicht, erkennt man die Unterschiede *(siehe Listing 26)*. Zu beachten ist, dass der Keystore der Datenbank geöffnet sein muss, bevor entsprechende Database-Links beziehungsweise Scheduler-Jobs benutzt oder gestartet werden können *(siehe Listing 27)*. Ist der Keystore geschlossen, erscheint eine Fehlermeldung bei der Verwendung eines Database-Links *(siehe Listing 28)*. Nach Öffnung des Keystore durch den „SYSDBA" oder „SYSKM" funktioniert alles wieder wie gewohnt *(siehe Listing 29)*.

Um alles wieder rückgängig zu machen, reicht ein Befehl *(siehe Listing 30)*. Beim Versuch, zum Beispiel einen Database-Link zu verwenden, erscheint eine Fehlermeldung *(siehe Listing 31)*. Dies lässt sich durch das Neusetzen des Kennworts der Database-Links korrigieren *(siehe Listing 32)*. Dieses Feature schließt eine Sicherheitslücke, die ständig bei Datenbank-Sicherheitsüberprüfungen als Risiko angezeigt wurde und jetzt mit der 18c geschlossen ist. Es steht in allen Editionen ab Oracle-Datenbank 18c ohne zusätzliche Lizenzen zur Verfügung.

Keystore pro Pluggable Database

Keystores werden zur sicheren Speicherung von Krypto-Schlüsseln jeglicher Art verwendet. Auch die Oracle-Datenbank verwendet einen Keystore zur Speicherung des externen Schlüssels, sobald Transparent Data Encryption (TDE) zum Verschlüsseln der Oracle-Datenbank verwendet wird. Neben der Speicherung des TDE Master Key werden weitere Keystores von der Oracle-Datenbank verwendet – zum Beispiel zur Speicherung von Passwörtern, Zertifikaten und anderen „Geheimnissen". Als Standard-Keystore wird das Oracle Wallet verwendet. Bisher war es so, dass jede verschlüsselte Datenbank einen dedizierten Keystore verwenden musste. Dies galt auch für eine Multitenant-Umgebung, in der jede Pluggable Database (PDB) den gemeinsamen Keystore der Container Database (CDB) benutzte. Das widersprach der strikten Trennung zwischen den PDBs untereinander, aber auch zwischen der CDB und den PDBs. Es war also nicht möglich, dass ein PDB-Administrator beziehungsweise der Mandant über seinen eigenen Keystore verfügte. Dabei ist der Zugriff auf Krypto-Schlüssel die „Achillessehne der Datensicherheit".

Derjenige, der Zugriff auf die Schlüssel hat, hat auch Zugriff auf die Daten. Mit der Oracle-Datenbank 18c wird diese Problematik nun gelöst: Jede PDB kann ihren eigenen Keystore besitzen. Das Feature – Keystore for Each Pluggable Database – ermöglicht es in einer Multitenant-Umgebung, dass jede PDB einen eigenen Keystore verwenden kann. Wird diese Eigenschaft genutzt, erhalten die Administratoren der PDB die volle alleinige Kontrolle über ihren Keystore. Hieraus folgt aber auch Verantwortung: Sollte der PDB-Administrator zum Beispiel das Passwort seines Keystore vergessen, gibt es niemanden mehr, der ihm helfen kann.

Um das Feature zu verwenden, wurden zwei neue Parameter und Keystore-Eigenschaften eingeführt, die in den folgenden Abschnitten erklärt werden.

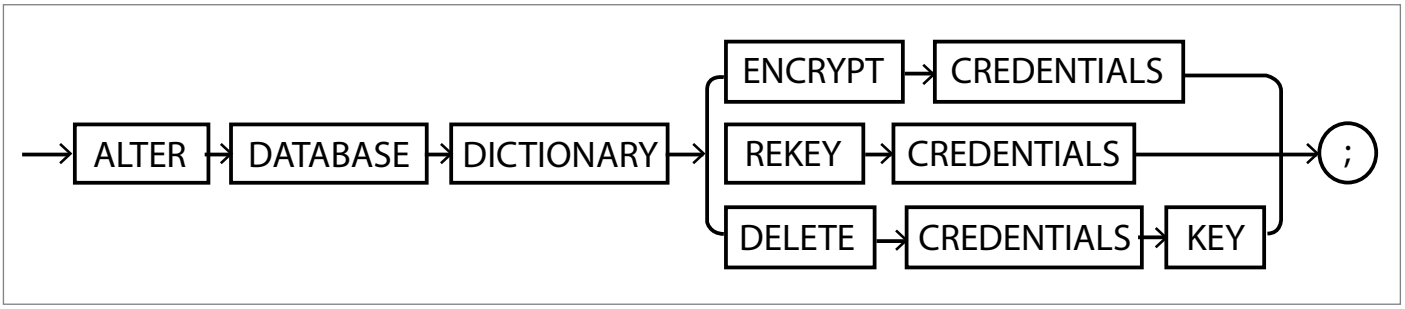

Abbildung 1: Ausführungsvarianten

Der Datenbank-Initialisierungs-Parameter „WALLET_ROOT“

Mit der Version 18c wurde der Datenbank-Initialisierungs-Parameter „WALLET_ROOT“ eingeführt. Dieser ersetzt den „SQLNET.ORA“-Parameter „SQLNET.ENCRYPTION_WALLET_LOCATION“, der ab 18c abgekündigt wurde, aber noch eine gewisse Zeit funktioniert. Ist der Parameter nicht gesetzt, bezieht die Datenbank das Wallet-Verzeichnis weiterhin über den „SQLNET.ENCRYPTION_WALLET_LOCATION“-Parameter aus der „SQLNET.ORA“. „WALLET_ROOT“ wird in der CDB gesetzt. Der Parameter beschreibt das Keystore-Basis-Verzeichnis aller der CDB zugeordneten PDBs. Darauf aufbauend dient die eindeutige „PDB-GUID“ als weiteres Verzeichnis zur Trennung der Keystores. *Listing 33* zeigt die Struktur der „/wallet-root/pdb-guid/tde“.

Der Parameter „WALLET_ROOT“ steht auch in der klassischen Oracle-Datenbank-Betriebsart der sogenannten „NON-CDB“ zur Verfügung. Im Standard ist diese nicht gesetzt und es handelt sich um einen statischen Parameter. Demzufolge ist die Datenbank nach dem Setzen neu zu starten. Das folgende Beispiel verwendet das Verzeichnis „/home/oracle/keystore_root“ als Keystore-Basis-Verzeichnis. Dafür wurde eine Oracle Database Cloud Service Enterprise Edition verwendet *(siehe Listing 34)*.

Der Datenbank-Initialisierungs-Parameter „TDE_CONFIGURATION“

Zudem wurde der Initialisierungs-Parameter „TDE_CONFIGURATION“ eingeführt. Dieser ist nur in einer Oracle Database Enterprise Edition auf Engineered Systems und ab Oracle Database Cloud Service Enterprise Edition aufwärts verwendbar. Bevor dieser Parameter gesetzt werden kann, muss das Keystore-Basis-Verzeichnis mittels „WALLET_ROOT“- Parameter gesetzt sein. „TDE_CONFIGURATION“ sorgt dafür, dass die PDBs wissen, welche Art von Keystore verwendet werden soll. Zur Auswahl stehen folgende Typen von Keystores:

- Das klassische Wallet FILE
- Hardware Security Modul (HSM)
- Oracle Key Vault (OKV)

Gesetzt wird dieser Parameter sowohl in der CDB als auch in jeder PDB. Das Beispiel im *Listing 35* verwendet den Keystore-Typ „FILE“, also ein Wallet als Keystore. Bei einer Multitenant-Umgebung mit zwei Pluggable Databases „PDB1“ und „PDB2“ (ohne „PDB$SEED“) sieht das wie in *Listing 36* aus. Die CDB-Root („CON_ID 1“) ist zu diesem Zeitpunkt immer im Keystore-Modus „NONE“ und alle weiteren PDBs („CON_ID 3“ und „4“), inklusive der „PDB$SEED“ („CON_ID 2“), sind standardmäßig auf „UNITED“ festgelegt. Diese Einstellung bedeutet, dass momentan alle PDBs einen gemeinsamen Keystore verwenden (Standard-Einstellung).

Neu mit 18c ist auch, dass zwei Betriebsvarianten der Keystores möglich sind: ein United- und ein Isolation-Modus. Im United-Modus sind alle TDE-Master-Keys wie gewohnt in einem der CDB zugeordneten Keystore gespeichert. Im Isolation-Modus hingegen erhält jede PDB einen eigenen Keystore. Der entscheidende Vorteil ist hier, dass ausschließlich der Administrator der PDB das Passwort des Keystore kennt. Wird der Keystore im United-Modus für die PDBs verwendet, hat im Gegensatz dazu der CDB-Administrator Zugriff auf alle dort gespeicherten TDE-Master-Keys.

Um jetzt einzelne PDBs in den Keystore-Isolation-Modus zu bringen, ist es notwendig, den bereits beschriebenen Parameter „TDE_CONFIGURATION“ in den entsprechenden PDBs zu setzen *(siehe Listing 37)*. Dies kann bei Bedarf auf andere PDBs angewandt werden. Es besteht die Möglichkeit, für jede PDB individuell einen anderen Keystore-Typ zu verwenden *(siehe Listing 38)*. Wie hier zu sehen ist, besitzt jetzt jede PDB ihren eigenen isolierten, File-basierten Keystore. Im Filesystem sieht das dann wie in *Listing 39* aus.

Dieses Feature ist ein konsequenter und wichtiger Schritt in Richtung „Mandantenfähigkeit“. Es erweitert die Möglichkeiten zur Isolierung von PDBs bezüglich des Schlüsselmanagements und bietet dem Mandanten somit eine bessere Zugriffskontrolle auf seine Daten. Dieser Parameter kann nur in einer Oracle Database Enterprise Edition auf Engineered Systems und ab Oracle Database Cloud Service Enterprise Edition aufwärts verwendet werden.

Centrally Managed Users

Centrally Managed Users (CMU) ermöglicht es, dass Datenbank-Benutzer und Rollen jetzt direkt in einem Microsoft Active Directory verwaltet werden können – und das, ohne einen Oracle-LDAP-Server dazwischen zu betreiben. Damit kann ein Benutzer aus dem Microsoft Active Directory exklusiv einem Datenbank-Benutzer zugeordnet sein oder es teilt sich eine Microsoft-Active-Directory-Gruppe einen Benutzer in der Datenbank. Microsoft-Active-Directory-Gruppen können auch direkt entsprechenden Datenbank-Rollen (Global Roles) zugeordnet sein. Die so zentral verwalteten Benutzer können sich durch Passwörter, Kerberos-Tickets und PKI-Zertifikate authentifizieren.

Für die Passwort-Authentifizierung müssen ein sogenannter „Passwortfilter“ auf dem Active Directory Server installiert und das Active Directory Schema erweitert sein. Dazu liefert Oracle ein Konfigurationswerkzeug, um den Kennwortfilter zu installieren und das Active-Directory-Schema zu erweitern.

Insbesondere bei neuen Identity-Management-Projekten ist dies eine gute Möglichkeit, sowohl Komplexität als auch die Betriebskosten im Hinblick auf Wartung und Entwicklung zu verringern. Dieses Feature steht in allen Enterprise-Editionen ab Oracle-Datenbank 18c ohne zusätzliche Lizenzen zur Verfügung.

Fazit

Wie gezeigt, enthält die Oracle-Datenbank 18c einige nützliche Sicherheits-Features. Jedes einzelne davon macht die IT ein wenig sicherer. Es muss allerdings auch eingesetzt werden.

Norman Sibbing
norman.sibbing@oracle.com