Wie sicher sind Ihre Daten in der Datenbank?

Sven Vetter
Trivadis AG
Glattbrugg, Schweiz

Schlüsselworte:

Oracle Database, Oracle Security, Auditing, Zentrales Auditing, Database Vault, Segregation of duty, Berechtigungen, Rollen

Einleitung

Oracle liefert diverse Optionen und Zusatzprodukte, um die Sicherheit der Daten zu gewährleisten. Schon die Liste der Abkürzungen ist lang (VPD, RLS, ASO, TDE, DBV, AV, ...) ☺

Aber dabei stellen sich einige Fragen:

- Welche Features brauche ich, um meine Risiken abzudecken?
- Können diese wirklich sicherstellen, dass keine Daten gestohlen werden?
- Brauche ich für bestimmte Anwendungsfälle noch weitere Programme - eventuell von Fremdherstellern?

An einem einfachen Beispiel soll demonstriert werden, über welche (vorgesehenen und nicht vorgesehenen) Funktionalitäten auf eine Tabelle zugegriffen werden kann. Außerdem werden jeweils Lösungen und teilweise Alternativen aufgezeigt, um dieses Risiko zu erkennen und abzusichern.
In der DOAG Konferenz Präsentation wird auch jeweils der benötigte Code gezeigt, worauf ich hier größtenteils aus Platzgründen verzichte.

Ziel

Folgende Tabelle im Schema Bank existiert:

```
DESC accounts
 Name              Null?    Type
 ----------------- -------- ----------------
 ACC_NR                     VARCHAR2(4)
 MANDANT                    VARCHAR2(2)
 ACC_NAME                   VARCHAR2(20)
 ACC_BALANCE                VARCHAR2(10)

SELECT * FROM accounts;

ACC_ MA ACC_NAME             ACC_BALANC
---- -- -------------------- ----------
1000 10 Mr. Rich             1000000000
1001 20 Mr. VeryRich         9999999999
```

Hr. Smith soll Mandant 10, Hr. Adams soll Mandant 20 lesen dürfen, ansonsten soll niemand Zugriff auf diese Daten erhalten – sowohl innerhalb als auch außerhalb der Datenbank.

Umsetzung

Der Schemaowner hat immer Zugriff auf seine Tabellen. Meist wird er für den Releaseprozess benötigt und kann deshalb nicht gesperrt werden. In einem securitykritischem Umfeld sollte dann mit dem 4-Augen-Prinzip gearbeitet werden. Eine Organisation (z.B. DBAs) setzt eine Hälfte des Passworts, eine weitere Organisation (z.B. Datenowner) setzt die andere Hälfte. Nur gemeinsam können sie sich dann anmelden und sich gegenseitig überwachen.
Alternativen: Proxyuser, Database Vault, Auditing

Benutzer mit direkten Grants können die Daten sehen. Dies ist in der View dba_tab_privs zu kontrollieren und regelmäßig zu überwachen.
Grants können aber nicht nur direkt an die Benutzer erteilt werden, sondern (bevorzugt) auch an Rollen. Auch dies ist in der View dba_tab_privs zu kontrollieren und regelmäßig zu überwachen. In dba_role_privs ist zu erkennen, wer Mitglied in welcher Rolle ist. Rollen können auch geschachtelt werden - es sind dann für jede Rolle die Benutzer zu kontrollieren.

Ein Sonderfall des Grants ist an "PUBLIC". Damit könnten alle Benutzer der Datenbank die Daten lesen. Auch dies ist in der View dba_tab_privs zu erkennen. Aufpassen - auch eine Rolle könnte an PUBLIC erteilt sein (kontrollieren in dba_role_privs).

Zugriff auf die Daten könnten auch über Views und PL/SQL-Programme erlangt werden. In dba_dependencies sind diese abgängigen Objekte zu erkennen. Für jedes davon muss wiederum überprüft werden, wer darauf Zugriff hat (direkt oder über eine (geschachtelte) Rolle).
Leider sind dort nicht alle abhängigen Objekte zu erkennen. Wird mit dynamischen SQL innerhalb PL/SQL gearbeitet, wird dies nicht aufgelistet.
Beispiel:

```
CREATE OR REPLACE PROCEDURE READ_ACCOUNTS AS
  vNo      VARCHAR2(10);
  vName    VARCHAR2(10);
  vBalance VARCHAR2(10);
BEGIN
  EXECUTE immediate
    'select acc_nr, acc_name, acc_balance
      from accounts where acc_nr=:1'
    INTO vNo, vName, vBalance USING '1000';
  dbms_output.put_line(vNo||' '||vName||' '||vBalance);
END READ_ACCOUNTS;
```

Diese Art von PL/SQL kann nur der Schemaowner erstellen oder jemand, der direkte Grants (mit Grant Option) hat (bzw. jeder, wenn es an PUBLIC erteilt wurde). Alle diese Benutzer müssen (manuell) kontrolliert werden. Manuell deshalb, weil eine Full-Text-Suche auf "ACCOUNTS" in dba_source eventuell nicht ausreicht!
Beispiel:

```
...
vDummy:='UNTS';
EXECUTE immediate
    'select acc_nr, acc_name, acc_balance
      from acco'||vDummy||' where acc_nr=:1'
...
```

Das Wort ACCOUNTS ist hier nicht zu finden, trotzdem wird auf die Tabelle ACCOUNTS zugegriffen.
Diese Art von Code sollte aber eigentlich bei jedem Codereview gefunden werden, da hier kein dynamisches SQL notwendig und sinnvoll ist.

Das Ziel war ja, dass ADAMS und SMITH zwar die Daten lesen dürfen, aber nicht alle Zeilen (nur ihren Mandant). Eine Lösung dafür wäre die "Virtual private database" (VPD) - auch genannt Row Level Security (RLS) oder Security Policies (nur in der Enterprise Edition enthalten). Dabei erzeugt eine PL/SQL-Funktion eine Zeichenkette, welche automatisch als WHERE-Bedingung an jeden Befehl gehangen wird.
Beispiel für eine ganz einfache PL/SQL Funktion:

```
CREATE OR REPLACE  FUNCTION acc_restrict(
      schema IN VARCHAR2,
      tab    IN VARCHAR2)
    RETURN VARCHAR2
AS
BEGIN
  IF sys_context('userenv','session_user')='SMITH' THEN
    RETURN 'mandant=10';
  ELSIF sys_context('userenv','session_user')='ADAMS' THEN
    RETURN 'mandant=20';
  ELSE
    RETURN '1=2';
  END IF;
END acc_restrict;
```

Diese wird per Package dbms_rls mit der Tabelle ACCOUNTS verbunden. Das Ergebnis sieht dann so aus:

```
connect smith
SELECT * FROM bank.accounts;

ACC_ MA ACC_NAME             ACC_BALANC
---- -- -------------------- ----------
1000 10 Mr. Rich             1000000000

connect adams
SELECT * FROM bank.accounts;

ACC_ MA ACC_NAME             ACC_BALANC
---- -- -------------------- ----------
1001 20 Mr. VeryRich         9999999999

connect system
SELECT * FROM bank.accounts;

no rows selected
```

Also auch ein DBA (SYSTEM) sieht die Daten nicht.
Doch halt - leider gibt es ein Systemprivileg (exempt access policy), welches sich der DBA selbst geben kann - und dadurch sieht er wieder alle Zeilen.
Alternativen: Mandantenfähige Views, Zugriff über PL/SQL-API, Für DBAs Database Vault

In den Datenfiles sind die Daten im Klartext, d.h. ein Benutzer mit Berechtigungen auf dem Server kann sie lesen. Dagegen hilft "Transparent Data Encryption". Dieses Feature ist in der Advanced Security Option (ASO) enthalten - und benötigt damit die Enterprise Edition. Außerdem wird für alle schreibenden und lesenden Operationen mehr CPU-Leistung benötigt, es ist vorher abzuklären, ob diese zur Verfügung steht. Verschlüsselt werden kann auf Spaltenebene oder (ab Oracle 11g) auf Tablespace-Ebene.
Alternativen: Verschlüsselung in der Applikation, Verschlüsselung per Appliance im Netzwerk

Sind die Datenfiles nicht verschlüsselt, ist auch das Backup unverschlüsselt - ein Benutzer mit Berechtigungen auf dem Backupsystem kann sie also lesen. Hiergegen hilft die Backupverschlüsselung (in ASO). Für ein direktes verschlüsseltes Backup auf Tape wird Oracle Secure Backup benötigt.
Alternativen: Verschlüsselung in der Applikation, Verschlüsselung der Datenfiles, Verschlüsslung per Backupsystem

Im Netzwerk sind die Daten im Klartext zu sehen, ein Benutzer mit Berechtigungen im Netzwerk kann sie lesen. Hier hilft die Netzwerkverschlüsslung (symmetrisch oder SSL) weiter. Auch dies ist nur in der Advanced Security Option enthalten.
Alternativen: Verschlüsselung in der Applikation

Ein Datenbankadministrator (oder natürlich auch ein OS-Admin) kann sich Zugang zu allen Daten verschaffen. Hiergegen hilft der Einsatz von Oracle Database Vault. Dies ist eine lizenzpflichtige Option, funktioniert also nur mit Enterprise Edition.
Umsetzung:

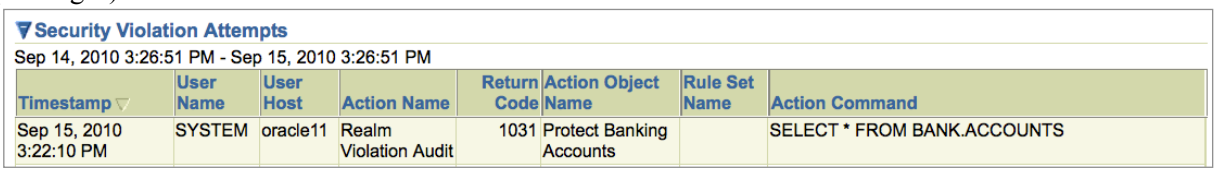
- Neue Rolle für Accountmanagement (damit der DBA keine Passwörter ändern kann)
- Neue Schutzart "Realms", um die ANY-Privilegien des DBAs einzuschränken
- (Möglicher) Schutz vor SYSDBAs, aber mit diversen Nachteilen (z.B. braucht RMAN SYSDBA-Connects)

Beispiel (Realm anlegen und Objekte zuweisen):

```
BEGIN
dbms_macadm.create_realm(
  realm_name    => 'Protect Banking Accounts',
  description   => 'Realm for accounts table',
  enabled       => 'Y',
  audit_options => 1);
END;
/

BEGIN
dbms_macadm.add_object_to_realm(
  realm_name   => 'Protect Banking Accounts',
  object_owner => 'BANK',
  object_name  => 'ACCOUNTS',
  object_type  => 'TABLE');
END;
/
```

Ein Zugriff als DBA auf die Tabelle würde dann einen Fehler erzeugen (ORA-01031: insufficient privileges). Dies ist dann über Views bzw. über die Konsole zu erkennen:



Security Violation Attempts

Sep 14, 2010 3:26:51 PM - Sep 15, 2010 3:26:51 PM

Timestamp	User Name	User Host	Action Name	Return Code	Action Object Name	Rule Set Name	Action Command
Sep 15, 2010 3:22:10 PM	SYSTEM	oracle11	Realm Violation Audit	1031	Protect Banking Accounts		SELECT * FROM BANK.ACCOUNTS

Abb. 1: Fehlermeldung bei unerlaubtem Zugriff in der Database Vault Console

Wichtig ist, dass beim Einsatz von Database Vault ein gutes Rollenkonzept durchgesetzt wird, ansonsten kann eine Person mit genügend Rechten doch wieder Daten lesen.
Ein Beispiel für solch ein Rollenkonzept zeigt diese Tabelle (die rechte Seite müssen Sie selbst ausfüllen, rot sind die Tätigkeiten, die im Vergleich zu einer Datenbank ohne Database Vault anders sind):

Task	Verantwortlich
Betrieb der DB und der Instanz (Erzeugen, Parametrisieren, Instanztuning, Patching, Updates, Tablespace-Management, …)	
Security Management Anlegen von Realms, Definition der zu schützenden Objekte	
Zuteilen der Benutzer zu Realms Anlegen von applikatorischen Rollen Zuordnen von Objektprivilegien zu Rollen/Benutzern	
Account Management + Zuweisen von Rollen	
Anlegen von technischen Rollen, Initiales zuordnen von Systemprivilegien zu Rollen (nicht applikatorische Rollen!)	
Überwachung (Auditing)	

Alternativen:

- Verschlüsselung der Daten vor der Datenbank
  - In der Applikation
  - Per Network Device (z.B. SafeNet Database Encryption)
- Datenbank Firewalls
  - Secerno (neuester Zukauf von Oracle)
  - Imperva Database Firewall

Eine weitere Alternative wäre aber auch der Entscheid, dass man dem DBA den Zugriff nicht verbieten will, diesen aber überwacht, z.b. per Oracle Standard Auditing (mit eingeschaltetem Auditing der SYS Operations), Oracle Audit Vault oder Sentrigo Hedgehog.

Bevor Auditing eingeschaltet wird, sollte folgendes Problem gelöst werden: Die Administratoren arbeiten auf OS-Ebene mit dem User oracle und melden sich an die Datenbank mit "/ as sysdba" an. Im Auditprotokoll würde dann immer oracle und sys stehen, es wäre nicht zu erkennen, wer bestimmte Operationen wirklich durchgeführt hat. Um die Nachweisbarkeit sicherzustellen, ist mit persönlichen Benutzern zu arbeiten und die OS-Anmeldung als oracle ist zu sperren. Befehle mit den Privilegien von oracle (z.B. Softwareinstallationen) sind per SUDO durchzuführen – und aufzuzeichnen. Auch an der Datenbank muss sich per persönlichen Benutzer angemeldet werden (nicht "/ as sysdba" – aber durchaus, wenn notwendig mit "username/password as sysdba").

Bei Oracles Standardauditing sind keine Wildcards und negierenden Bedingungen möglich (z.B. alle Tabellen, die mit ACC% starten, aber nicht, wenn ADAMS und SMITH darauf zugreifen). D.h. es muss pro Benutzer und pro Objekt das Auditing definiert werden, was sehr schwierig werden kann bei vielen Benutzern und vielen Objekten.
Eine Lösung dafür wären Tools von Fremdherstellern, z.B. Sentrigo Hedgehog.

Hedgehog ist sehr leicht zu installieren und zu konfigurieren und durch SQL-ähnliche Bedingungen (like, not in, ...) sehr flexibel. Außerdem bietet es gute Auswertungsmöglichkeiten und kann noch viel mehr als nur Auditing (Stichwort "Virtual patching").

Eine Zugriffsverletzung (SYS greift auf Tabelle ACCOUNTS zu) würde dann so aussehen:

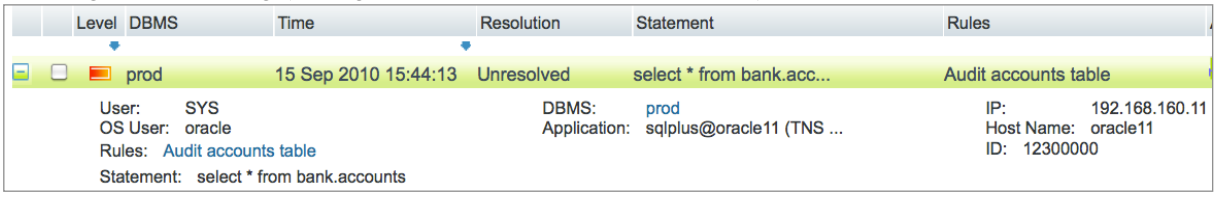


Abb. 2: Zugriffsverletzung in Sentrigo Hedgehog

Fazit

Es gibt viel Produkte/Optionen/Befehle, die den Zugriff auf sensitive Daten limitieren – oder auditieren. Eins davon allein reicht im Normalfall nicht aus, deswegen muss nach einer Risikoanalyse die richtige Kombination gefunden werden.
Das ist nicht immer leicht... - und der Artikel behandelte nur den Datenbankteil. Zusätzlich muss die gesamte Informationskette betrachtet werden (Schnittstellen, Application Server, Applikationen (vielleicht gibt es da ein "Export to Excel"), ...

Aber wir unterstützen Sie gern ☺

Kontaktadresse:

Sven Vetter
Trivadis AG
Europa-Strasse 5
CH-8152 Glattbrugg

Telefon: +41-44-808 70 20
Fax: +41-44-808 70 21
E-Mail Sven.Vetter@trivadis.com
Internet: www.trivadis.com