Keine Option: Datenbanken härten
von Heinz-Wilhelm Fabry, ORACLE Deutschland GmbH

http://www.oracle.com/global/de/community/dbadmin/tipps/haerten/index.html

Jede Komponente der IT Infrastruktur muß optimal gegen Mißbrauch gesichert werden, damit ein akzeptabler Sicherheitsgrad des Gesamtsystems erreicht wird. Datenbanken kommt dabei jedoch eine herausragende Bedeutung zu, denn in ihnen ist das wichtigste Gut eines Unternehmens in konzentrierter Form abgelegt: Informationen.

Natürlich gibt es zahlreiche sogenannte Angriffsvektoren, gegen die DBAs 'ihre' Datenbanken schützen müssen. Dabei können sie unterschiedliche und mitunter auch kostspielige Instrumente nutzen. All diese Instrumente laufen aber ohne einen gewissen Grundschutz völlig ins Leere. Diesen Grundschutz bezeichnet man mit dem Begriff 'Härten' - und sieht man einmal von der einzusetzenden Arbeitszeit ab, steht das Härten völlig kostenlos zur Verfügung. Das ist auch der Grund, warum nicht nur Hochsicherheitsdatenbanken gehärtet werden sollten: Eigentlich sollte jede Datenbank ganz selbstverständlich eine gehärtete Datenbank sein.

Im *Leitfaden IT-Sicherheit* (2007, S. 24) des Bundesamt für Sicherheit in der Informationstechnik (BSI) kann man lesen: "Härten (engl. Hardening) bedeutet in der IT-Sicherheit die Entfernung aller Softwarebestandteile und Funktionen, die zur Erfüllung der vorgesehenen Aufgabe durch das Programm nicht zwingend notwendig sind." Allerdings gibt es im Zusammenhang mit einer Datenbank neben dieser 'Reduktion auf das Notwendige' weitere Praktiken, die den Schutz vor Mißbrauch erhöhen.

Weniger ist mehr

Komponenten, die nicht installiert sind, bieten auch keine Angriffsfläche. Deshalb sollten immer nur die Komponenten installiert werden, die auch tatsächlich genutzt werden. Sollen vorhandene Datenbanken nachträglich gehärtet werden, helfen Views wie V$OPTION, DBA_REGISTRY und DBA_FEATURE_USAGE_STATISTICS bei der Identifizierung der installierten und verwendeten Komponenten. Es gibt unterschiedliche Möglichkeiten, solche Komponenten zu entfernen. Man kann z.B. eine in den Programmcode der Datenbank gelinkte Komponente wie die Option OLAP nur durch erneutes Linken de-installieren. Für das nachträgliche Löschen eines Feature braucht dagegen die Oracle-Software in der Regel nicht neu gelinkt zu werden. Hier enthält der Lieferumfang der Datenbank häufig Skripte zum De-Installieren - wie beispielsweise $ORACLE_HOME/rdbms/admin/catnoexf.sql, mit dem das Feature Expression Filter entfernt wird. Auch für das Entfernen von Demo-Schemas stehen solche Skripte zur Verfügung: Das Skript hr_drop.sql in $ORACLE_HOME/demo/schema/human_resources löscht z.B. den Benutzer HR und seine Objekte. Bei weiteren Fragen zu den von Oracle installierten Benutzern liefert Metalink Note 160861.1 detaillierte Informationen zur Verwendung der Benutzer und ihrer Passwörter.

Benutzer-Management und Passwörter

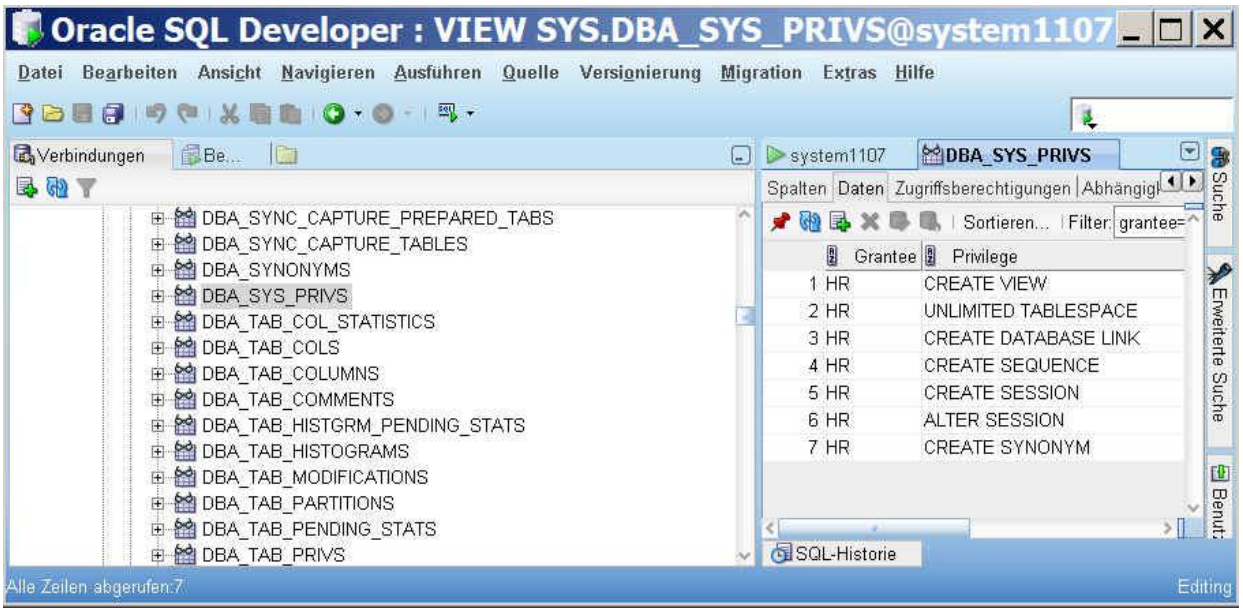
Verlassen Mitarbeiter das Unternehmen, sollten ihre Accounts gelöscht werden. Ist das im Einzelfall nicht ohne weiteres möglich, sollte man den Status mindestens auf EXPIRED & LOCKED setzen. Die Fehlermeldung, die aus dem Versuch des Einloggens in einen derart gesperrten Account resultiert (*ORA-28000: account is locked*), kann allerdings unter Umständen als Sicherheitsrisiko gewertet werden. Denn immerhin gibt die Meldung preis,

dass es den Account gibt. Einem Hacker bietet das vielleicht schon den Anhaltspunkt, wo er mit einem sogenannten *brute force*-Angriff auf das Passwort ansetzen kann. Ein solcher Account könnte daher auch durch ein ungültiges Passwort geschützt werden. Der Befehl dazu könnte etwa so aussehen:

```
ALTER USER exmitarbeiter
IDENTIFIED BY VALUES 'nichtmehr'
```

Beim Versuch, sich in den Account einzuloggen, wird nun die Fehlermeldung *ORA-01017: invalid username/password; logon denied* angezeigt. Sie läßt sowohl die Existenz des Accounts als auch die Möglichkeit eines falschen Passworts offen.

Zum Thema Benutzer-Management sei auch noch angemerkt, dass nur die Privilegien vergeben werden sollten, die auch tatsächlich benötigt werden (*least privilege*). Das gilt ganz besonders für die Vergabe von sogenannten ANY-Privilegien. Aber auch ein so mächtiges Privileg wie CREATE DATABASE LINK sollte nicht jedem offenstehen (das Privileg war vor der Version 10.1 der Datenbank Bestandteil der Rolle CONNECT). Bei der Suche nach den erteilten Privilegien helfen z.B. die entsprechenden Menu-Punkte des Enterprise Manager oder die Berichte des SQL Developer (vgl. die folgende Abbildung zu den System-Privilegien des Benutzers HR).



Auch Passwörter gehören in den Bereich des Account-Managements. Unter UNIX / LINUX ist zu beachten, dass Passwörter über die History-Dateien der UNIX-Shells kompromittiert werden können. Vergleichbare Gefahr geht auch von log- und trace-Dateien der Datenbank und der Werkzeuge aus, die auf die Datenbank zugreifen. Aber nicht nur im Klartext gespeicherte Passwörter stellen eine Gefahr dar, sondern auch Default- und ungenügend komplexe Passwörter. Mit den heutigen Möglichkeiten, z.B. durch den zweckentfremdeten Einsatz von Videokarten, Passwörter zu knacken, sollten die Regeln für gültige Passwörter unbedingt verschärft werden. Mit Oracle11*g* ist das deutlich einfacher geworden. So kann man mit einer *password verify function* (Beispiel in utlpwdmg.sql unter $ORACLE_HOME/rdbms/admin) sicherstellen, dass z.B. Passwörter aus mindestens 12 Zeichen als Mischung aus Groß- und Kleinbuchstaben, Zahlen und Sonderzeichen bestehen. Über Profile lassen sich zusätzlich Account-Charakteristika festlegen wie z.B. der Zeitraum, nach dem ein Passwort verfällt, oder die Anzahl falscher Passwort-Eingaben, nach denen ein

Account gesperrt wird, usw. Das folgende Beispiel zeigt, wie einfach das Arbeiten mit solchen Profilen ist.

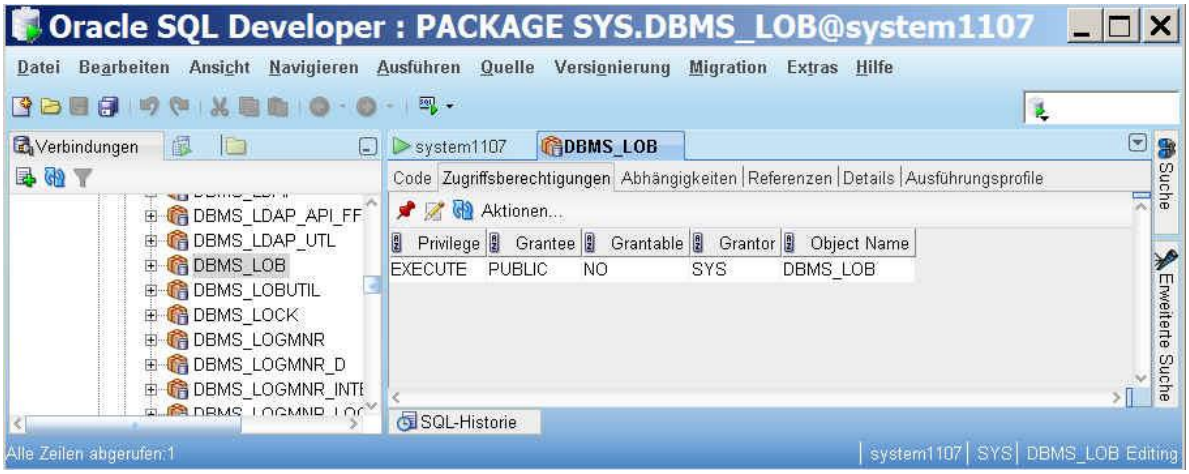
```
CREATE PROFILE pwsichern LIMIT
FAILED_LOGIN_ATTEMPTS 5        -- Anzahl
PASSWORD_LIFE_TIME 30  -- Tage
PASSWORD_REUSE_TIME 7  -- Tage
PASSWORD_REUSE_MAX 3   -- Anzahl
PASSWORD_LOCK_TIME 30  -- Tage
PASSWORD_GRACE_TIME 3  -- Tage
PASSWORD_VERIFY_FUNCTION funktionsname;

ALTER USER mitarbeiter PROFILE pwsichern;
```

Wenn Datenbanken über Skripte aufgesetzt werden, sind eventuell eingerichtete Demo-Accounts nicht EXPIRED & LOCKED (s.o.). Auch die Passwörter dieser Accounts könnten dann die weithin bekannten Default-Passwörter sein. Damit sind sie das ideale Einfallstor für Hacker. Man identifiziert solche Accounts sehr leicht mit dem im Rahmen des Patch 4926128 auf Metalink angebotenen ORACLE DEFAULT PASSWORD SCANNER. Oracle11*g* bietet mit der View DBA_USERS_WITH_DEFPWD einen noch einfacheren Mechanismus, Default-Passwörter zu erkennen. Übrigens sollte diese Überprüfung gelegentlich wiederholt werden, da der ursprüngliche Zustand durch Mitarbeiter, Skripte oder Patches irrtümlich wiederhergestellt worden sein könnte.

Zum Thema Passwörter sei schließlich noch darauf hingewiesen, dass die Änderung eines Passwortes über den Befehl ALTER USER unverschlüsselt (!) über das Netzwerk gesendet wird - sofern der Netzwerkverkehr nicht mit der Oracle Advanced Security Option oder mit anderen Mitteln verschlüsselt wird. Falls die Verwendung von SQL*Plus eine Alternative darstellt, kann der Befehl PASSWORD Abhilfe schaffen: Er schickt das Passwort ebenfalls verschlüsselt an die Datenbank.

PL/SQL-Packages

Im Lieferumfang der Datenbank sind zahlreiche PL/SQL-Packages enthalten, die durch den Benutzer PUBLIC, also durch jeden Datenbankbenutzer, ausgeführt werden können. Damit besteht die Gefahr, dass sie von böswilligen Benutzern mißbraucht werden. Ein Datendieb kann in Oracle10*g* Packages wie UTL_TCP oder UTL_SMTP verwenden, um interne Informationen ganz einfach an externe Adressaten zu schicken. (Dieses Risiko wird in Oracle Database 11*g* durch den Einsatz sogenannter *access control lists* deutlich reduziert.) Ein weiteres Beispiel ist das Package DBMS_LOB: Es erlaubt das Lesen oder Manipulieren von Datenbankdateien - die ja schließlich nichts anderes sind als LOBs. Dem Benutzer PUBLIC sollte deshalb der Zugriff auf diese und andere PL/SQL-Packages durch ein REVOKE EXECUTE genommen werden. Selbstverständlich sollte man sich vorher vergewissern, dass die Packages nicht in Anwendungen oder gar datenbankintern verwendet werden. Die Zugriffsberechtigungen auf die Packages lassen sich sehr bequem mit dem SQL*Developer prüfen (vgl. die folgende Abbildung).



Sicherlich ist es deutlich schwieriger, die Verwendung der Packages in Anwendungen oder durch die Datenbank zu überprüfen.

Oracle-Netzwerk-Konfiguration

Der Zugriff über ein Netzwerk auf die Datenbank ist schon deutlich sicherer zu gestalten, wenn man die Rechner benennen kann, von denen aus ein Zugriff auf die Datenbank ausschließlich möglich oder auch ausdrücklich nicht möglich sein soll. Oracle leistet diese Steuerung über Einträge in der Datei SQLNET.ORA. Nach dem Setzen des Parameters TCP.VALIDNODE_CHECKING auf YES kann sowohl eine Negativ- (Parameter TCP.EXCLUDED_NODES) als auch eine Positiv-Liste (Parameter TCP.INVITED_NODES) erstellt werden.

Der Default-Port 1521 für den LISTENER sollte auf jeden Fall geändert werden. Zwar schützt eine solche Änderung nicht vor Angriffsversuchen, allerdings erhöht die Änderung des Ports den Aufwand, den ein Angreifer betreiben muß, um in das System einzudringen. In Anwendung des Sankt-Florian-Prinzips führt der zusätzliche Aufwand aber vielleicht dazu, dass sich der potentielle Eindringling ein anderes Opfer sucht.

Das Risiko, das der Aufruf manipulierter externer Prozeduren darstellt, sollte nur eingegangen werden, wenn tatsächlich mit solchen Prozeduren gearbeitet wird. Deshalb sollte der standardmäßig für den Listener eingerichtete Service EXTPROC auch nur dann in der Konfigurationsdatei enthalten sein. Das gilt natürlich analog für alle anderen, nicht benötigten Services.

Auch der Listener kann durch ein Passwort geschützt werden. Für ältere Datenbankversionen (vor Oracle10g) empfiehlt sich dieser Schutz unbedingt. Seit Oracle10g ist der Listener allerdings nur lokal zu administrieren. Es ist daher abzuwägen, ob der Schutz durch die ausschließlich lokale Administration nicht dem Risiko vorzuziehen ist, den ein Passwort-geschützter Listener darstellt: Ein Passwort-geschützter Listener ist NICHT nur lokal zu administrieren, sondern auch von anderen Systemen aus. Damit ist er wieder offen für *brute force*-Angriffe auf das Passwort.

Schließlich sei noch darauf hingewiesen, dass in sensiblen Umgebungen die log-Datei des Listeners regelmäßig auf die Anwendungsprogramme hin durchsucht werden sollte, die sich bei der Datenbank angemeldet haben. Tauchen hier Programme auf, die normalerweise nicht im Unternehmen verwendet werden, ist natürlich eine sofortige Überprüfung nötig.

Dateien und Verzeichnisse

Oracle legt im Rahmen von Upgrades Sicherungskopien der alten Programmdateien ab. Sie heißen z.B. oracle0, oracle.bak oder ähnlich. Diese Dateien können gelöscht oder unter UNIX / LINUX mit chmod 000 so verändert werden, dass keinerlei Aktion mit ihnen mehr möglich ist.

Die beiden folgenden Hinweise gelten ausschließlich für UNIX- und LINUX-Systeme. Die Programme der Datenbank sind z.T. über das SUID-bit und ausserdem für jeden (world) ausführbar. Das ist in der Regel nicht sinnvoll, sondern stellt eher ein Sicherheitsrisiko dar. Deshalb sollte der Dateischutz von Dateien wie oracle oder auch sqlplus auf 0700 geändert werden. Das führt dazu, dass auch eigentlich lokale Verbindungen zur Datenbank immer über den Listener unter Angabe z.B. des Service-Namens hergestellt werden müssen. Aber zusätzlich ist auch ein Einloggen mittels AS SYSDBA nur nach Eingabe eines Passwortes möglich. Und das erhöht natürlich die Sicherheit des Systems.

Bei der Installation wird die umask der Oracle-SW-Verzeichnisse auf 022 gesetzt. Das führt dazu, dass jeder z.B. aus den Dump- und Log-Verzeichnissen Dateien kopieren kann. Liegen dann hier auch noch Backup-Sets oder Dump-File-Sets von Data Pump, sind diese quasi öffentlich verfügbar. Deshalb sollte die Einstellung auf jeden Fall geändert werden - z.B. mit umask 177.

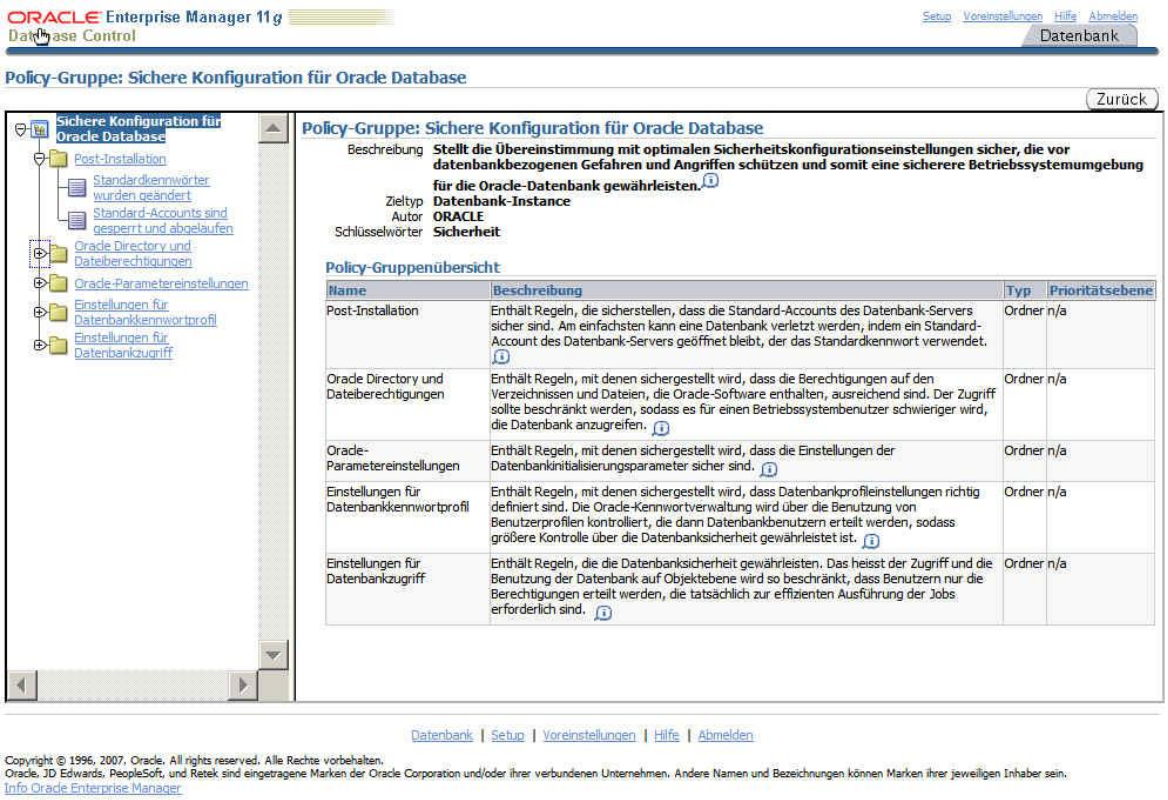
Sicherheitspatches

Es ist klar, dass eine nicht gepatchte Datenbank ein leichtes Opfer für Angreifer ist. Selbst wenn diese nicht selbst über genügend Wissen zum Ausnutzen einer nicht behobenen Sicherheitslücke verfügen, so sind sogenannte Exploits (Anweisungen oder Programme, die eine Sicherheitslücke ausnutzen) für bereits behobene Sicherheitslücken für wenig Geld oder gar umsonst im Internet zu finden.

Probleme mit fehlenden Sicherheitspatches können auch durch alte Datenbankversionen entstehen, für die von Oracle keine Sicherheitspatches mehr zur Verfügung gestellt werden. Damit bieten sie Angreifern einen leichten Zugang, von dem aus kriminelle Aktivitäten auf andere Systeme ausgeweitet werden können. Das gilt übrigens auch für Datenbanken, die eigentlich nur zum Testen aufgesetzt wurden und dann in Vergessenheit gerieten.

Regeln einhalten mit Hilfe von Enterprise Manager

Über die oben genannten Punkte hinaus leistet der Enterprise Manager (vgl. die folgende Abbildung) gute Unterstützung bei der Einhaltung und Überwachung zahlreicher sicherheitsrelevanter Einstellungen. Zu finden ist diese Unterstützung - z.B. auf der Homepage des EM - unter dem Begriff *Policy*. Die Anzahl der Policies wächst mit jedem Release des EM, und natürlich stöt man dort auch auf Policies, die als Thema in diesem Community-Beitrag angesprochen wurden. Mit der neusten Version des Enterprise Manager Grid Control lassen sich übrigens sogar eigene Policies definieren, die dann ebenfalls automatisch und laufend überwacht werden - für einzelne Datenbanken oder sogar für alle Datenbanken eines Unternehmens.



Abschließender Hinweis

Wer sich weiter mit dem Thema beschäftigen möchte, dem sei als Ausgangspunkt Arup Nandas umfangreicher und sehr lesenswerter Artikel Project Lockdown empfohlen.

Zurück zur Community-Seite