DOAG
Deutsche ORACLE-Anwendergruppe e.V.

SIG Security Köln
Dienstag, 15. Oktober 2012

Der DBA und seine Tools – DAS ideale Opfer?

Ist das wirklich mein Code, der da gerade ausgeführt wird?

WOHIN NUR MIT DEN PASSWÖRTERN? WIE SCHÜTZE ICH MEINE SCRIPTS?

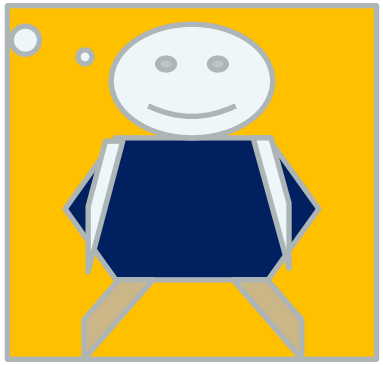
Agenda

- Passwörter finden und schützen
  - Wo werden Passwörter oft „vergessen“?
  - Lassen sich Passwörter schützen?
- Skripte schützen und sichern
  - Wie verhindere ich das mein Code verändert wird?
  - Wie kann ich veränderten Code von der DB Seite erkennen?

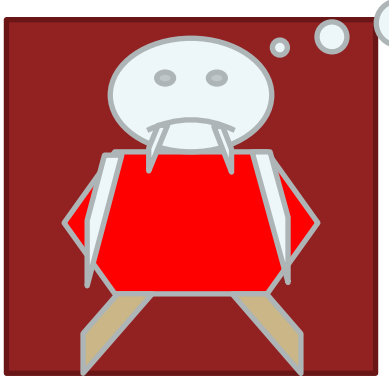
Unser Angriff Szenario – die DB Umgebung

Oh je, so viel Arbeit ..

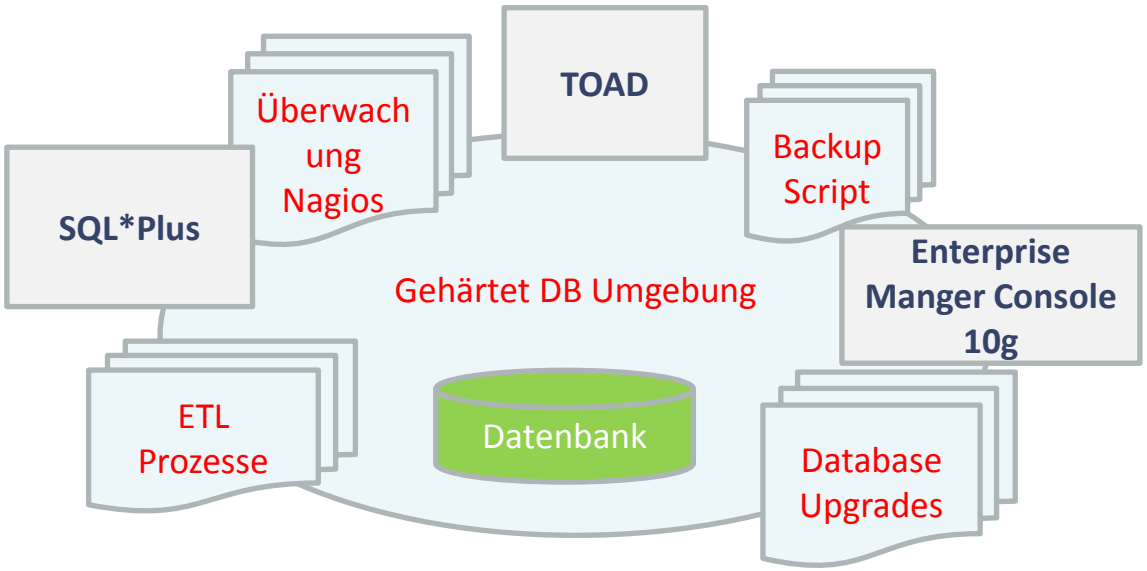
Das Opfer =>



Der Feind =>

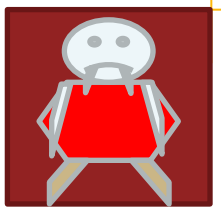


Hurra, so viele Chancen!



- An wie vielen Stellen hat der DBA wohl die Passwörter versteckt ?
- Wo und wie kann ich dem DBA etwas in seinen Skripten unterschieben?

Der einfache Angriff – Arbeitsplatz (1)

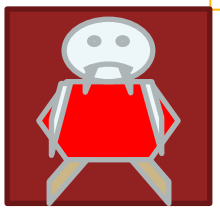


- Hinterlegte Passwörter in Tools
  - Wie (ältere) TOAD / der Oracle 10g EM Java Manager
    - Einfach verschlüsselte Passwort Liste „stehlen“
    - => Fertig! Rest macht ja dann das Tool
    - Wie => Angriff auf den Rechner des DBA mit „klassischen Mitteln“, wie untreue Kollegen, Putzfrau / Service Mann nimmt den Rechner im Urlaub mit ….
    - Logfile / Temp Files

Key Logger, Trojaner, Video Überwachung von außen, und und ….

Stichwort: Password Roaming

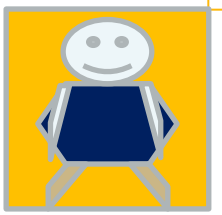
Der einfache Angriff – Arbeitsplatz (2)



- Die Oracle / TOAD Login SQL
  - Die Datei glogin.sql, login.sql wird ausgeführt wenn SQL*Plus gestartet wird
  - gleicher Mechanismus auch in TOAD möglich (toad.ini)
  - SQLPATH Umgebungsvariable manipulieren
    - Mit der Umgebungsvariable SQLPATH die Umgebung umbiegen .-)

Einfacher Schutz

- Keine Passwörter auf dem Arbeitsrechner speichern oder anderweitig hinterlegen



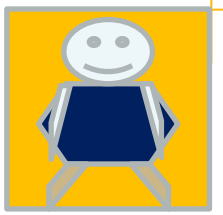
Chef => Gehört in den Arbeitsvertrag – Muss beim Audit regelmäßig geprüft werden !
DBA => Der hat ja einen Vog… - Wie soll ich dann das schaffen!

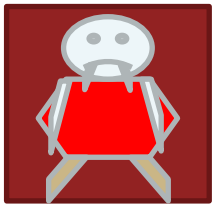
=> Sicherheit bedeutet auch Zeit und Sorgsamkeit - hier hilft nichts anders!

- Festplattenverschlüsselung für die Arbeitsplätze der DBA‘s
  - Mit Kennwort Abfrage beim Start des Rechners!
- USB Security bzw. keine USB Ports am Arbeitsplatz
- Schutz der login.sql Dateien vor Veränderungen
  - Rechte aggressiv setzen
  - Teure Tools oder Scripting Lösung wie z.B. PowerShell beim Login mit hinterlegter Quersumme prüfen lassen

Versuchen wir es doch auf dem DB Server

- Kann uns nicht passieren, da kommt keiner drauf!



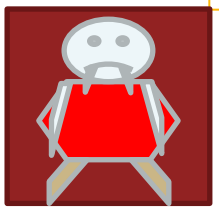


  - Aber wie war das ist mit dem lieben Kollegen mit seinem Nagios, den netten Externen, dem Remote Support von Firma x aus dem fernen Osten, der sicher immer Nachts einwählt, diese seltsamen Software die mein ERP Hersteller installiert haben will, den sicheren Update aus dem Netz, unsern zuverlässigen Virenscanner und war da nicht vor kurzen der HP Techniker um die Platte zu tauschen …. und wo ist eigentlich die Acronis Sicherung vom letzten Monat geblieben?

Fangen wir mit dem Suchen an (1)

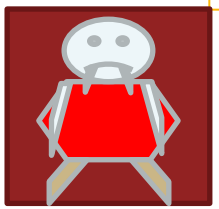
- Anmelden mit „/ as sysdba“ auf dem lokalen DB Server?
  - Geht nicht?
  - Meist fehlt nur der NTS Eintrag in der sqlnet.ora

```
SQLNET.AUTHENTICATION_SERVICES= (NTS)
```



Achtung: RAC unter Windows -> Für die ASM Instance werden diese Rechte benötigt!

Fangen wir mit dem Suchen an (2)



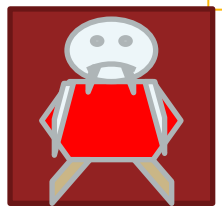
- Angreifer darf die Datei nicht bearbeiten?
  - sqlnet.ora / tnsnames.ora nach /tmp kopieren
  - editieren
  - TNS_ADMIN variable auf /tmp setzen
  - Und nun nehmen wir unsere eigene sqlnet.ora!
- Geht immer noch nicht? => Unter Windows schnell mal in die DB Gruppe eintragen!

```
PowerShell Beispiel:

# auf die Gruppe zugreifen
$group = [ADSI]"WinNT://$env:computername/ora_dba,group"

# User Gast der Gruppe hinzufügen
$User="Gast"
$group.add("WinNT://$env:computername/$user")
```

Ein erster Schutz

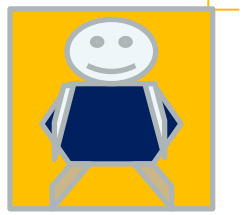
- Accounts trennen
  - Unterschiedliche User für unterschiedliche Wartungsarbeiten
  - Oracle Software unter eigene Account installiert
  - Sql*Net Dateien schreibschützen, so das der Wartungszugang diese nicht editieren kann
  - Rechte so vergeben, dass der Wartungszugang sich selbst nicht der ORA_DBA Gruppe hinzufügen kann

Sehen wir uns die Skripte an



- Wichtige/aktive Skripte finden:
  - Job Steuerung auslesen
    - Linux crontab –l
    - Windows: Powerschell/GUI
    - Oft steht das Passwort auch als Parameter im der Job Steuerung
- Skripte der Jobs durchsuchen

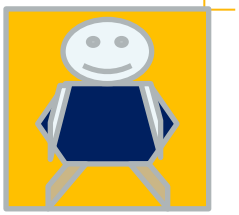
Ein erster einfacher Schutz der Skripte



- Zugriffberechtigung so minimal wie möglich
- Keine Passwörter
  in den Skripten in Klarschrift hinterlegen!
- Spool / log Files der Skripte prüfen und jeden Hinweise auf User oder gar Passwörtern entfernen
  - Beispiel SQL*Plus:

```
Spool backup.log
Prompt Start the backup
Spool off
connect backupuser/@myzertconnect
Spool backup.log append
```

Windows: Passwort mit SecureString (1)

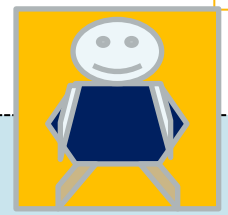


- Windows: PowerShell Beispiel
  - Unter Windows können Passwörter hart verschlüsselt hinterlegt werden (zum Beispiel in Konfigurationsdateien)
  - Beispiel:

```
------ Encrypt ------------------------------------------------
# the clear passwort
$clear_password="mypassword"

# wirte password to secure string
$secure_container = New-Object System.Security.SecureString
$clear_password.GetEnumerator() | foreach {$secure_container.AppendChar($_)}
$secure_container.MakeReadOnly()

# read the secure String
$secure_password=$secure_container|convertFrom-SecureString

# show the encrypted password
$secure_password
```

Windows Passwort mit SecureString (2)

– Wieder auslesen



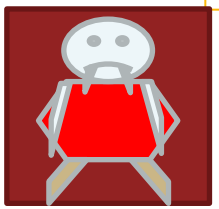
```
---- Decrypt -----------------------------------------------------
$secure_pwd_string=ConvertTo-SecureString -String $secure_password

$secure_container =
[System.Runtime.InteropServices.Marshal]::SecureStringToCoTaskMemUnicode($secure_pwd_string)

$result = [System.Runtime.InteropServices.Marshal]::PtrToStringUni($secure_container)

# Clean Cache
[System.Runtime.InteropServices.Marshal]::ZeroFreeCoTaskMemUnicode($secure_container)

# print the result
$result
```

DEMO: PowerShell Backup Script Demo

Beispiel Skripte für das Sichern der Datenbank mit der PowerShell unter http://orapowershell.codeplex.com

SecureString auf der selben Maschine auslesen

- Da der Schlüssel für die Verschlüsselung die lokale Maschine ist, kann hier auch jederzeit der auf der Maschine der String gelesen werden



PowerShell:
Zusätzlich SecureString mit eigenen Key Verschlüssen

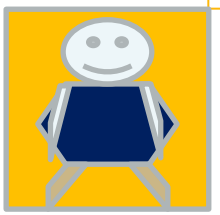
Stichwort:
System.Security.Cryptography.RSACryptoServiceProvider

Siehe http://msdn.microsoft.com/en-us/library/9eat8fht.aspx

Das alte Sorgenkind => Wohin mit dem Schlüssel der Verschlüsselung?!?

Linux – Strings verschlüsseln

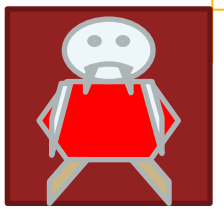
- Beispiel mit openssl:



```
# get SYSTEMIDENTIFIER
SYSTEMIDENTIFIER=`ls -l /dev/disk/by-uuid/ | awk '{ print $9 }' | tail -1`
export SYSTEMIDENTIFIER

################################################################################
# Password file handling
encryptPWDFile () {

  if [ -f "/usr/bin/openssl" ]; then
    # Encrypting the password file
    printLine "Trying to encrypt the password file :"
    openssl des3 -salt -in ${PWDFILE} -out ${PWDFILE}.des3 -pass pass:"${SYSTEMIDENTIFIER}" > /dev/null
    printf "%s encrypted file :: %s to %s.des3 \n" "--" "${PWDFILE}" "${PWDFILE}"
    rm ${PWDFILE}
  else
    printError "OpenSSL could not be found - the password file is not encrypted."
  fi

}
decryptPWDFile() {

  if [ -f "/usr/bin/openssl" ]; then
    # Decrypting the password file
    printLine "Trying to decrypt the password file :"
    openssl des3 -d -salt -in ${PWDFILE}.des3 -out ${PWDFILE} -pass pass:"${SYSTEMIDENTIFIER}" > /dev/null
    printf "%s decrypted file :: %s.des3 to %s \n" "--" "${PWDFILE}" "${PWDFILE}"
  else
    printError "OpenSSL could not be found - the password file is not decrypted."
  fi

}
```

Prozessliste auswerten - Linux

- ps -x Problem
  Aufruf Parameter in der Process Liste anzeigen
  - Alle in der Kommando Zeile übergebenen Passwörter in der Process Liste in Klarschrift sichtbar ….



– ABER 11g =>

```
sqlplus system/oracle_@test.sql
ps -x | grep sql
 6143 tty1     S      0:00 sqlplus                @test.sql
```

Wird ausgeblendet .-)

– Mit der Prozess ID die Umgebung eines Aufrufs aus dem Proc File System lesen

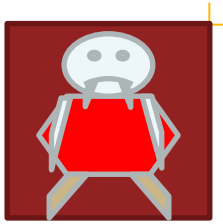
```
cd /proc/6143
cat enviorn
```

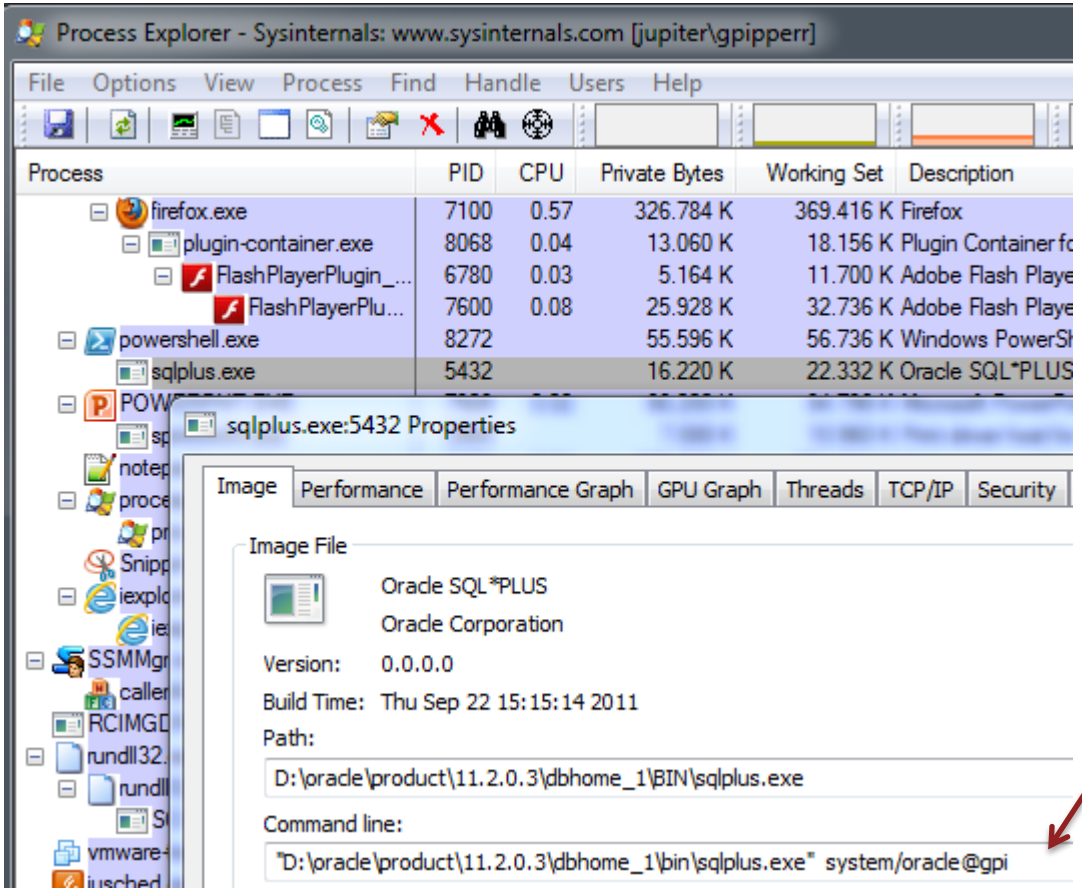
```
ADR_BASE=/u01/app/oracleHOSTNAME=VM1SETERM=linuxSHELL=/bin/bashHISTSIZE=1000ORACLE_UNQNAME=MCNGDBNLS_LANG=american_america.al32utf8CLUSTER_NAME=falseRAC_ENV=falseVIPNODES=VM1SEUSER=oracleLS_COLORS=no=00:fi=00:di=01;34:ln=01;36:pi=40;33:so=01;35:bd=40;33;01:cd=40;33;01:or=01;05;37;41:mi=01;05;37;41:ex=01;32:*.cmd=01;32:*.exe=01;32:*.com=01;32:*.btm=01;32:*.bat=01;32:*.sh=01;32:*.csh=01;32:*.tar=01;31:*.tgz=01;31:*.arj=01;31:*.taz=01;31:*.lzh=01;31:*.zip=01;31:*.z=01;31:*.Z=01;31:*.gz=01;31:*.bz2=01;31:*.bz=01;31:*.tz=01;31:*.rpm=01;31:*.cpio=01;31:*.jpg=01;35:*.gif=01;35:*.bmp=01;35:*.xbm=01;35:*.xpm=01;35:*.png=01;35:*.tif=01;35:ORACLE_SID=MCNGDBORACLE_HOSTNAME=VM1SEORACLE_BASE=/u01/app/oracleTNS_ADMIN=/u01/app/oracle/product/11.2.0/dbhome_1/network/adminMAIL=/var/spool/mail/oraclePATH=/usr/kerberos/bin:/usr/local/bin:/bin:/usr/bin:/u01/app/oracle/product/11.2.0/dbhome_1/binINPUTRC=/etc/inputrcPWD=/home/oracleNODE_ID=LANG=en_US.UTF-8ORACLE_INVENTORY_HOME=/u01/app/oraInventorySQLPATH=/home/oracle/sqlSSH_ASKPASS=/usr/libexec/openssh/gnome-ssh-askpassSHLVL=1HOME=/home/oracleLOGNAME=oracleCVS_RSH=sshLESSOPEN=|/usr/bin/lesspipe.sh %sORACLE_HOME=/u01/app/oracle/product/11.2.0/dbhome_1ASM_ENV=falseDEFAULT_ORACLE_HOME=/u01/app/oracle/product/11.2.0/dbhome_1G_BROKEN_FI
```

history nicht vergessen .-)

Prozessliste auswerten - Windows

- Sysinternals `sqlplus system/oracle@gpi`
  - Aufruf SQL*Plus in der Kommando Zeile

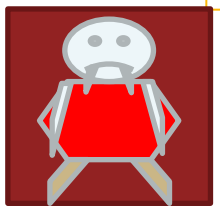




Prozesse auswerten - Windows

- Speicher mit Sysinternals auslesen

http://technet.microsoft.com/en-US/sysinternals



Anmelden mit /nolog und Connect im Script/über die Console:

```
D:\OraPowerShellCodePlex\sql>sqlplus /nolog

SQL*Plus: Release 11.2.0.3.0 Production on Do Okt 11 17:20:45 2012

Copyright (c) 1982, 2011, Oracle.  All rights reserved.

@-?>connect gpi/x1x2x3x4x5x6x7x8
Connect durchgef³hrt.
GPI@GPI-WORKGROUP\JUPITER>
```

Dump mit Process Explorer ziehen:

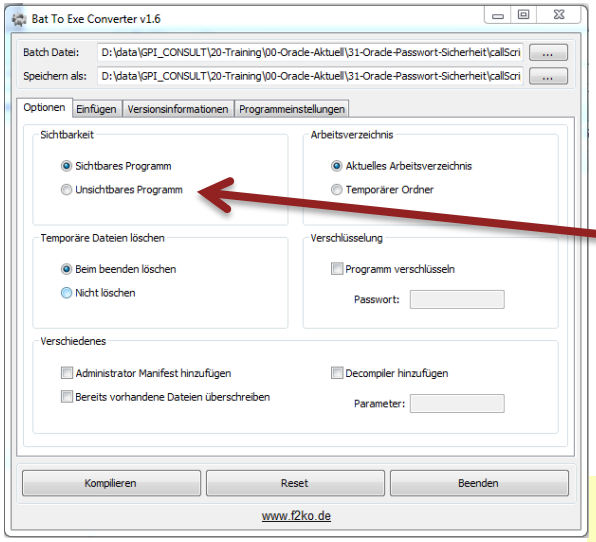
sqlplus.exe	8908	12.948 K	21.708 K Oracle SQL*PLUS	Oracle Corporation
cmd.exe	2964	2.688 K	3.264 K Windows-Befehlsprozessor	Microsoft Corporation
sqlplus.exe	9288	12.772 K	109.500 K Oracle SQL*PLUS	Oracle Corporation

Dump Auswerten , nach connect suchen:

```
00024a50  00 00 00 00 63 6f 6e 6e 65 63 74 20 67 70 69 2f  ....connect gpi/
00024a60  78 31 78 32 78 33 78 34 78 35 78 36 78 37 78 38  x1x2x3x4x5x6x7x8
00024a70  0d 0a 32 00 78 00 33 00 78 00 34 00 78 00 35 00  ..2.x.3.x.4.x.5.
00024a80  78 00 36 00 78 00 37 00 78 00 38 00 0d 00 0a 00  x.6.x.7.x.8.....
```

Nach den typischen Verstecken suchen

- Windows
  - Dos Console öffnen
    - => getarnt als Umgebungsvariable
  - Gepackte Batch Dateien als Exe?
    - => Kein Problem, einfach ausführen und in der Console das Passwort auslesen

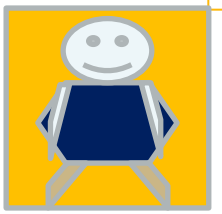


Ausgabe in der Console unterdrücken!

http://www.f2ko.de/programs.php?lang=de&pid=b2e

Passwörter in der Prozessliste vermeiden (1)

- Unix „echo §PWD | sqlplus userName“
  - Beispiel für eine Passwort Datei



```
umask 0077
PWDFILE=$(mktemp)

echo oracle > $PWDFILE

cat $PWDFILE | sqlplus -s system @job.sql

rm $PWDFILE
umask 0022
```

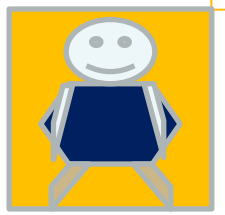
Nachteil: Password steht wieder in einem Script
Alternativ: Passwort verschlüsseln und erst bei Aufruf entschlüsselt mit diesen Mechanismus übergeben

Alternative können auch Skripte dieser Art sein:
http://rhadmin.org/oracle/cloak.ksh.txt

Passwörter in der Prozessliste vermeiden (2)

- User ohne Passwörter verwenden
  - Möglichkeiten
    - OPS$ Account – das Betriebssystem ist nun verantwortlich
      - **Con:**
        OPS$ User muss sich am OS angemeldet haben, damit das wirkt, oft ungeschickt für die Ausführung als Dienst/Jobs
    - Das Passwort in einer Oracle Wallet hinterlegen
      - **Pro**: einfaches Handling, echtes Passwort muss niemand kennen d.h. es können sehr sichere Passwörter verwendet werden und trotzdem kann auf der lokalen Maschine sich jeder anmelden
      - **Con**: Kennt man den passenden tnsnames String, braucht man ja auch kein Passwort mehr …..

Oracle Wallet - Secure External Passwort Store

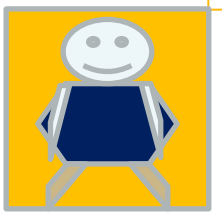


- Ab 10g R2
  - Eine Wallet anlegen
  - Zugriff auf Wallet in sqlnet.ora konfigurieren
  - TNSAlias in der tnsnames.ora setzen
  - Testen
- Ab 11g R1 kann die Wallet auf Benutzer/Server eingeschränkt werden
  - **“orapki** wallet create -wallet "." **-auto_login_local”**

Siehe auch:
http://www.pipperr.de/dokuwiki/doku.php?id=dba:passwort_schuetzen&s[]=wallet

Oracle Wallet - Secure External Passwort Store

- DEMO



1) - Wallet anlegen

```
mkdir d:\wallet
cd d:\wallet
orapki wallet create -wallet "." -auto_login_local

#Kennung hinterlegen
mkstore  -wrl .  -createCredential oragpi system oracle
```

2) - Edit sqlnet.ora

```
#Wallet Konfiguration
WALLET_LOCATION =
  (SOURCE =
    (METHOD = FILE)
    (METHOD_DATA =
      (DIRECTORY = D:\wallet)
   )
 )
SQLNET.WALLET_OVERRIDE = TRUE
SSL_CLIENT_AUTHENTICATION=FALSE
```

3) - Edit tnsnames.ora

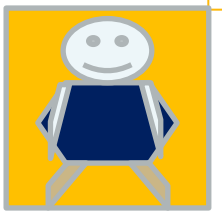
```
# eigener TNS Alias
ORAGPI =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = jupiter)(PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = GPI)
    )
  )
```

4) – test

```
 sqlplus /@oragpi
```

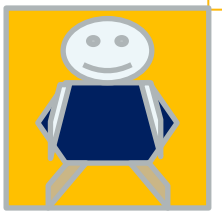
Überprüfen mit den Wallet Manager:

Proxy Connection verwenden (1)



- Die Oracle Wallet ist ideal in Kombination mit dem „Proxy Connection Feature“
- Regel: Niemand darf sich als Applikation Schema Eigentümer anmelden
- Lösung:
  - User A meldet sich an der DB an (A Password in Wallet)
  - User A „wird“ durch das Proxy Connect Feature zu User B in der DB
  - Zusätzlicher Schutz, Rollen aktiv per Password einschalten

Proxy Connection verwenden (2)



- Beispiel
  - Applikation ist unter „OPERA“ installiert
  - Connect erfolgt über „gpipperr“
  - Anlegen gpipperr

```
Sql> create user gpipperr identified by „x2x3!!x4x5“;
Sql> Grant create session to gpipperr;
```

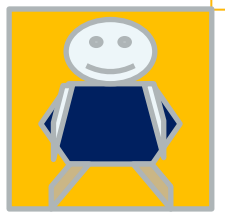
  - Proxy connect über gpipperr erlauben

```
Sql> alter user opera grant connect through  gpipperr;
```

  - Anmelden als gpipperr

```
Sql> connect gpipperr[opera]/”x2x3!!x4x5“
Sql> Select user form dual;
User
--------
opera
```

Konzept- Doppelter Schutz mit Rollen



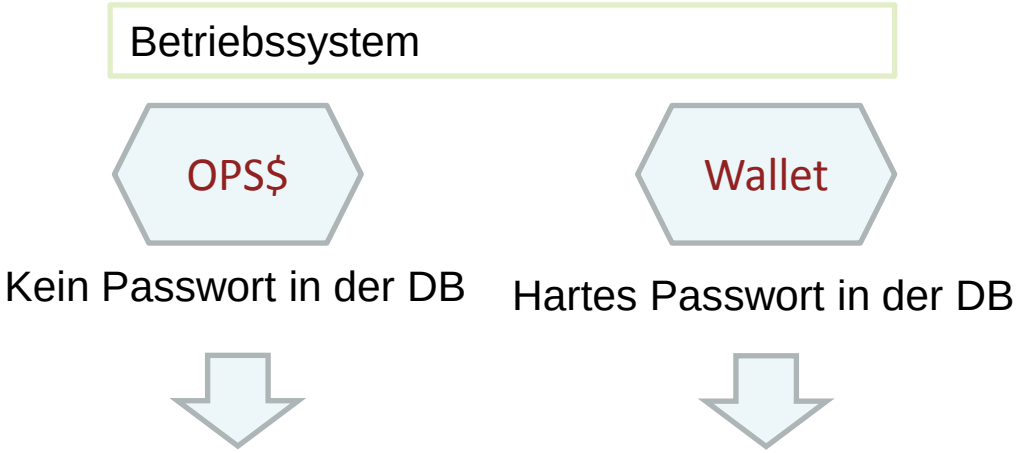
- User meldet sich an
- Keine Rolle ist aktiv
- Rollen werden aktiv geschaltet wenn Bedingung erfüllt ist (Einsatz von Logon Trigger)
  - Wie richtige Tageszeit, richtiger Host
    - Rollen aktiv schalten mit:

```
set role testrole identified by "x1x3!!#!!x4x5";
```

- Passwort Schutz möglich, nur wo hin mit dem PWD?
  - Konzept
    - User meldet sich ohne Passwort an (Wallet),
    - notwenige Rolle wird dann per Passwort und Regelwerk freigeschaltet

Passwort Strategien Zusammenfassung

Wo hin mit dem Passwort im Script ? Regel: Kein Passwort verwenden!

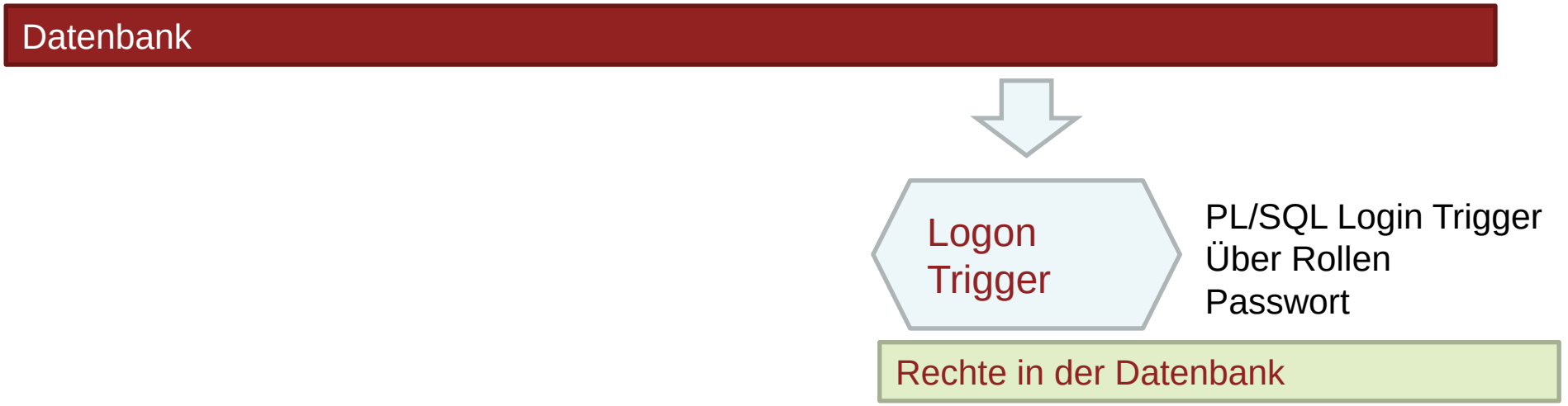


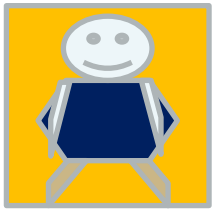
Rollen Passwort verwenden!

Konzept:

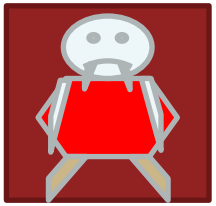
Einmal Passwort zur zusätzlichen Absicherung!

- Skript holt während der Ausführung das Erlaubnis/Password extern ab



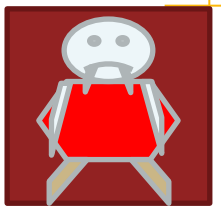


Geschafft – an das Passwort kommt mir keiner ran



Mir egal, will ich eigentlich auch gar nicht wissen, Hauptsache DU führst deine Script mit meinen Code aus .-)

Per Installation oder ein Update angreifen



- Ich will mein Rootkit in die DB bekommen …
  - Da war doch das ungeschützte Installationsverzeichnis in der Firma mit all den vielen Oracle Zip Files
  - Einfügen von eigenen Schadcode in die Oracle Installationsroutine und neu einpacken!

Einfacher Schutz durch MD5

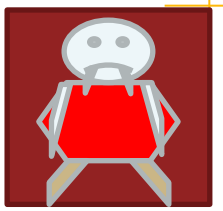
Schützt auch vor einigen „seltsamen“ Installationsprobleme mit defekten Downloads!

- Vertrauen ist gut - Kontrolle ist besser
  - Prüfen Sie IMMER den MD5 Hash der Oracle Installation/Upgrade/Patch Dateien
  - Tipp für Windows 2008:
    - Windows 2008 hat kein MD5Sum Check Programm
    - **`Fciv.exe ->`** Nachinstallieren (run as administrator!) mit der Datei windows-kb841290-x86-neu (Download auf http://support.microsoft.com/kb/841290)
    - Datei am einfachsten unter c:\windows installieren

```
fciv -add p10404530_112030_MSWIN-x86-64_1of7-database-part1.zip -md5

//
// File Checksum Integrity Verifier version 2.05.
//
0624981eca74b85df410e324682a1934 p10404530_112030_mswin-x86-64_1of7-
database-part1.zip
```

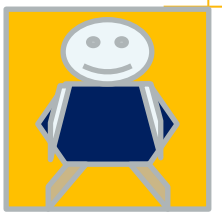
Script verändern

- Das Backup/Export/ETL Script schreibe ich dem DBA einfach um….
  - Potentielle Angriffsziele:
    - Alle Jobs wie Backup, ETL Ladevorgänge
    - Login.sql Datei auf dem Server und auf dem Client des Administrators



Skript vor Veränderung schützen - Windows

- PowerShell - nur noch signierten Code verwenden
  - Skript signieren



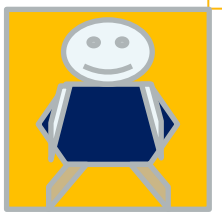
```
Set-AuthenticodeSignature $profile @(Get-ChildItem cert:\CurrentUser\My -codesigning)[0]
```

  - Ausführung nur von signierten Skripten erlauben

```
Set-ExecutionPolicy -scope CurrentUser AllSigned
```

Siehe auch: http://www.pipperr.de/dokuwiki/doku.php?id=windows:powershell_script_aufrufen

Skript vor Veränderung schützen - Linux

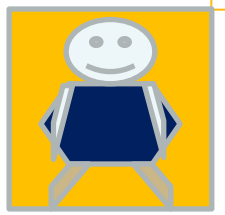


- Nur noch geschützten Code verwenden?
  - Mit externen Tools oder
    Script vor dem Ausführen jeweils MD5 Hash prüfen
  - Shell Script Compiler einsetzen
    - siehe z.B. http://linux.die.net/man/1/shc
  - SELinux Features mitverwenden

Anwenden:
http://www.thegeekstuff.com/2012/05/encrypt-bash-shell-script/

Reverse :
http://www.linuxjournal.com/article/8256

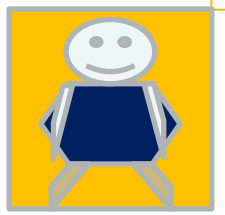
SQL Code beim Aufruf valideren



- SQL Skript MD5 Hash extern hinterlegen
- SQL Skript
  wird z.B. über PowerShell Mapper aufgerufen
  - Logon Trigger erlaubt nur PowerShell Connection
  - Hashwert von Skript in der Software vergleichen (aus Tag in SQL Skript)
  - Hashwert in Logtabelle in DB eintragen, Trigger prüft auf korrektes Skript
  - Skript wird ausgeführt

Demo: PowerShell Script

SQL-Skripte aktiv von der DB prüfen lassen



- Login Trigger und Passwort Rollen
  - Konzept: Password der Rolle basiert auf dem MD5 Hash der Script Datei
  - SQL*Plus Skript startet
    - Login mit wallet
    - Prüft mit Host Exit den MD5 Hash des gerade aufgerufenen Scripts und übergibt diesen an eine PL/SQL Funktion für die Rollen Freischaltung

Zu klärende Fragen:

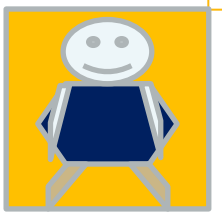
MD5 Hash als Aufruf Parameter für das Script sichern?

Error Verhalten (Exit on Error usw.) für Test verwenden?

MD5 Hashs zuvor in der DB ablegen?

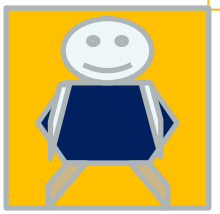
MD5 Hash mit der PowerShell erzeugen

- Beispiel:



```
# Vorlagen aus http://blog.brianhartsock.com/2008/12/13/using-powershell-for-md5-checksums/
#
function local-getMD5Hash {
        param(
                $file
        )

    if (get-ChildItem $file -ErrorAction silentlycontinue ) {

       $algorithmus = [System.Security.Cryptography.HashAlgorithm]::Create("MD5")
       $stream = New-Object System.IO.FileStream($file, [System.IO.FileMode]::Open)

       $md5StringBuilder = New-Object System.Text.StringBuilder
       $algorithmus.ComputeHash($stream) | % { [void] $md5StringBuilder.Append($_.ToString("x2")) }

       $stream.close()
       $stream.Dispose()
       return $md5StringBuilder.ToString()
    }
    else {
       Write-host "Error - $file not found"
    }
}
```

Md5 Hash nach SQL*Plus lesen

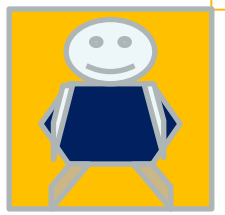


- Wie kann ich in SQL*Plus einen MD5 Hash „zurückbekommen“?
  - Linux

```
SQL> host echo def md5_hash=$(md5sum /home/oracle/sql/asm.sql) >
/tmp/hash.sql

SQL> @/tmp/hash.sql

SQL> host rm /tmp/hash.sql

SQL> def md5_hash

SQL> select '&md5_hash.' from dual;
```

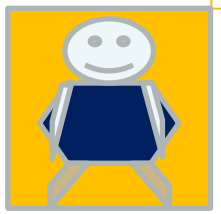
  - Windows
    - PowerShell Script für MD5 bzw. fcvi output mit Host Befehl wieder einlesen

Schutz RMAN-Skripte



- User aktiv ein/ausschalten
  - RMAN User wird nur aktiviert
    wenn zuvor ein SQL*Plus Script
    erfolgreich mit der Prüfroutine gelaufen ist
  - Nach Abschluss des Backups wird der RMAN User wieder „abgeschaltet“
- RMAN Script wird stets neu generiert
  - Aufrufendes Skript erzeugt jedes Mal neu den Code
  - Schutz des aufrufenden Skripts über Signaturen etc.

Die ultimative Lösung - keine Skripte verwenden



- Skripte in die Datenbank packen
- Umsetzung in PL/SQL
  - Wie zum Beispiel: DataPump Export können komplett in die DB verlegt werden
  - PL/SQL der DB mit externen Tools auf Veränderungen prüfen (wie McAfee DSS von Herrn Kornbrust)

Beispiel für DataPump

```
CREATE OR REPLACE PROCEDURE db_export_mc
IS
  v_dp_handle   NUMBER;
  PRAGMA AUTONOMOUS_TRANSACTION;
BEGIN
  -- Create Data Pump Handle - "TABLE EXPORT" in this case
  v_dp_handle :=
  DBMS_DATAPUMP.open (operation => 'EXPORT', job_mode => 'SCHEMA', job_name => 'GPI_EXPORT_' || TO_CHAR (SYSDATE, 'DD_HH24'));

  DBMS_DATAPUMP.set_parallel (handle => v_dp_handle, degree => 4);

  -- Export the complete schema
  DBMS_DATAPUMP.metadata_filter (handle => v_dp_handle, name => 'SCHEMA_EXPR', VALUE => 'IN (''MC'')');

    -- Specify target file - make it unique with a timestamp
  DBMS_DATAPUMP.add_file (handle      => v_dp_handle
                         ,filename    => 'MC_' || TO_CHAR (SYSDATE, 'YYYYMMDD-HH24MISS') || '%U.dmp'
                         ,directory   => 'GPI_EXPORT'
                         ,reusefile   => 1                                          -- overwrite existing files
                         ,filesize    => '50000M');

    -- Logfile
    DBMS_DATAPUMP.add_file (handle      => v_dp_handle
                         ,filename    => 'MC_' || TO_CHAR (SYSDATE, 'YYYYMMDD-HH24MISS') || '.log'
                                     ,filetype    => DBMS_DATAPUMP.KU$_FILE_TYPE_LOG_FILE
                         ,directory   => 'GPI_EXPORT'
                         ,reusefile   => 1                                          -- overwrite existing files
                         ,filesize    => '10000M');

  --    MERGE => that each partitioned table is re-created in the target database as an unpartitioned table
  -- DBMS_DATAPUMP.set_parameter (handle => v_dp_handle, name => 'PARTITION_OPTIONS', VALUE => 'MERGE');

  -- Do it!
  DBMS_DATAPUMP.start_job (handle => v_dp_handle);

  COMMIT;

--    DBMS_DATAPUMP.detach (handle => v_dp_handle);
END;
/
```

Skript Strategie – Wie schütze ich meine Code

Zugriffschutz

Betriebssystem

Signieren

Windows PowerShell

Aktiv überprüfen

Vor und/oder beim Aufruf aktiv überprüfen

Eigene Lösung über DB Trigger und Rollen

Password der Rolle ist der MD5 Hash der SQL Datei

Betriebssystem

Datenbank

Zusammenfassung

- Die Wartungs-Skripts für die Datenbank sind ein Einfalltor für jeden Angreifer
- Daher =>
  - Anzahl von Scripts minimieren
    - Möglichst viel in die DB verlagern
  - Passwörter durch die Oracle Wallet ersetzen
  - Script vor Veränderungen schützen
  - Code signieren, wenn immer möglich
  - Zusätzliche Schutz Mechanismus über Rollen und DB Trigger aktivieren

Password Handling
Schutz von Skripten

Referent

Technology Consultant

Gunther Pippèrr
Dipl. Ing. Technische Informatik (FH)

>12 Jahre IT Beratungserfahrung
Freiberuflich tätig

Schwerpunkte der Beratungstätigkeit

- IT System Architekt und technische Projektleitung
- Sicherheit in Datenbank Umgebungen
- Entwurf und Umsetzung von IT Infrastrukturen zum Datenmanagement mit Oracle Basis Technologien
- Training rund um die Oracle Datenbank

Beruflicher Hintergrund

- Dipl. Ing. Technische Informatik (FH) Weingarten
- Mehr als10 Jahre Erfahrung in komplexen IT Projekten zum Thema Datenhaltung/Datenmanagement
- Freiberufliche Projektarbeit
- Consultant bei großen Datenbank Hersteller

Projekterfahrung

- Architekt und technische Umsetzung für eine Data Warehouse Anwendung für die Analyse von Verbindungsdaten in der Telekomunikation

- Architekt und technische Projektverantwortung für ein Smart Metering Portal für das Erfassen von Energiezählerdaten und Asset Management

- Architekt und technische Projektverantwortung für IT Infrastrukturprojekte, z.B.:
  - Umsetzung von PCI Regularien für DB Systeme
  - Zentrale Datenhaltung für Münchner Hotelgruppe mit über 25 Hotels weltweit,
  - Redundante Cluster Datenbank Infrastrukturen für diverse größere Web Anwendungen wie Fondplattform und Versicherungsportale

Mehr über Datenbank Backup
mit der PowerShell unter
http://orapowershell.codeplex.com

Kontaktdaten: E-Mail: gunther@pipperr.de – Mobil:+49- (0)17180656113