

Best Practices for Oracle Databases

Hardening Oracle 10.2.0.3 / 10.2.0.4

Alexander Kornbrust

- Passwords
- (Security) Patches
- Database Settings
- PUBLIC Privileges
- Database Trigger
- Compiling Views
- Next Steps & Summary

Weak and default passwords is still problem No.1 in most Oracle databases.

Even if Oracle default accounts like SYS, SYSTEM, DBSNMP, ... are getting better, user accounts and technical accounts are often using weak passwords (password=username).

- It is useless to spend time for Oracle Security if the database is using weak/default passwords
- Check (Oracle) passwords on a regular basis against a custom dictionary file

Check Passwords Regularly

Do not use weak passwords and check all passwords on a regular basis, e.g. with checkpwd or repscan.

```
C:\>checkpwd system/strongpw66@123.34.54.123:1521/ORCL password_list.txt
```

```
Checkpwd 1.23 [Win] - (c) 2007 by Red-Database-Security GmbH  
Oracle Security Consulting, Security Audits & Security Training  
http://www.red-database-security.com
```

```
MDSYS has weak password MDSYS [EXPIRED & LOCKED]  
ORDSYS has weak password ORDSYS [EXPIRED & LOCKED]  
DUMMY123 has weak password DUMMY123 [OPEN]  
DBSNMP OK [OPEN]  
SCOTT has weak password TIGER [OPEN]  
CTXSYS has weak password CHANGE_ON_INSTALL [EXPIRED & LOCKED]  
SH has weak password CHANGE_ON_INSTALL [EXPIRED & LOCKED]  
OUTLN has weak password OUTLN [EXPIRED & LOCKED]  
DIP has weak password DIP [EXPIRED & LOCKED]  
DUMMY321 has weak password 123YMMUD [OPEN]  
[...]  
SYS OK [OPEN]  
SYSTEM OK [OPEN]
```

```
Done. Summary:  
Passwords checked : 13900828  
Weak passwords found : 23  
Elapsed time (min:sec) : 0:54  
Passwords / second : 265486
```

(Security) Patches

If the passwords are good it is time to apply (security) patches.

You should always try to upgrade at least to a supported version (e.g. 10.2.0.3 / 10.2.0.4).

After that you should apply the latest security patch from Oracle (January 2009 CPU).

For many reasons (newer version not supported, too many instances, ...) this is not always possible. In this case you should try to use a solution like Virtual Patching.

Exploits for problems fixed with the January 2009 CPU are already published on the internet:

```
exec EXFSYS.DBMS_EXPFIL_DR.GET_EXPRSET_STATS  
( 'EXFSYS' , 'EXF$VERSION' , 'EXFVERSION' ,  
'YYYYYYYY" and 1=EVILPROC()-- ' )
```

Database Settings

The next step is to change the default audit settings from Oracle.

audit_sys_operations

By default the database is not auditing SQL commands executed by the user SYS. To change this behaviour it is necessary to change this value to TRUE. A reboot of the database is necessary after changing this value.

Command:

```
SQL> alter system set audit_sys_operations=true  
scope=spfile;
```


audit_trail

By default the database is not auditing SQL commands. To enable auditing it is necessary to change this parameter to DB. In this case Oracle is writing all audit information from the database (but not the database vault audit information) into the table SYS.AUD\$. Other options could be OS, DB, XML,EXTENDED . A reboot of the database is necessary after changing this value.

Extended is a new feature since Oracle 10g Rel.2

Command:

```
SQL> alter system set audit_trail=DB,EXTENDED  
scope=spfile;
```

Now it's time to remove dangerous privileges. The only question is

“What is a dangerous package?”

Now it's time to remove dangerous privileges. The only question is

“What is a dangerous package?”

If we look at the Oracle Security Checklist (Jul 2008) from Oracle, Oracle recommends to remove the privileges from

- UTL_TCP
- UTL_SMTP
- UTL_MAIL
- UTL_HTTP
- UTL_INADDR
- UTL_FILE

What are the most dangerous packages in an Oracle database?

- `dbms_sql`
- `utl_file`
- `utl_mail`
- `utl_inaddr`
- `utl_tcp`
- `dbms_lob`
- `dbms_xmlgen`
- `dbms_aw_xml`
- `ctxsys.drithsx`
- `ordsys.ord_dicom`
- `kupp$proc`

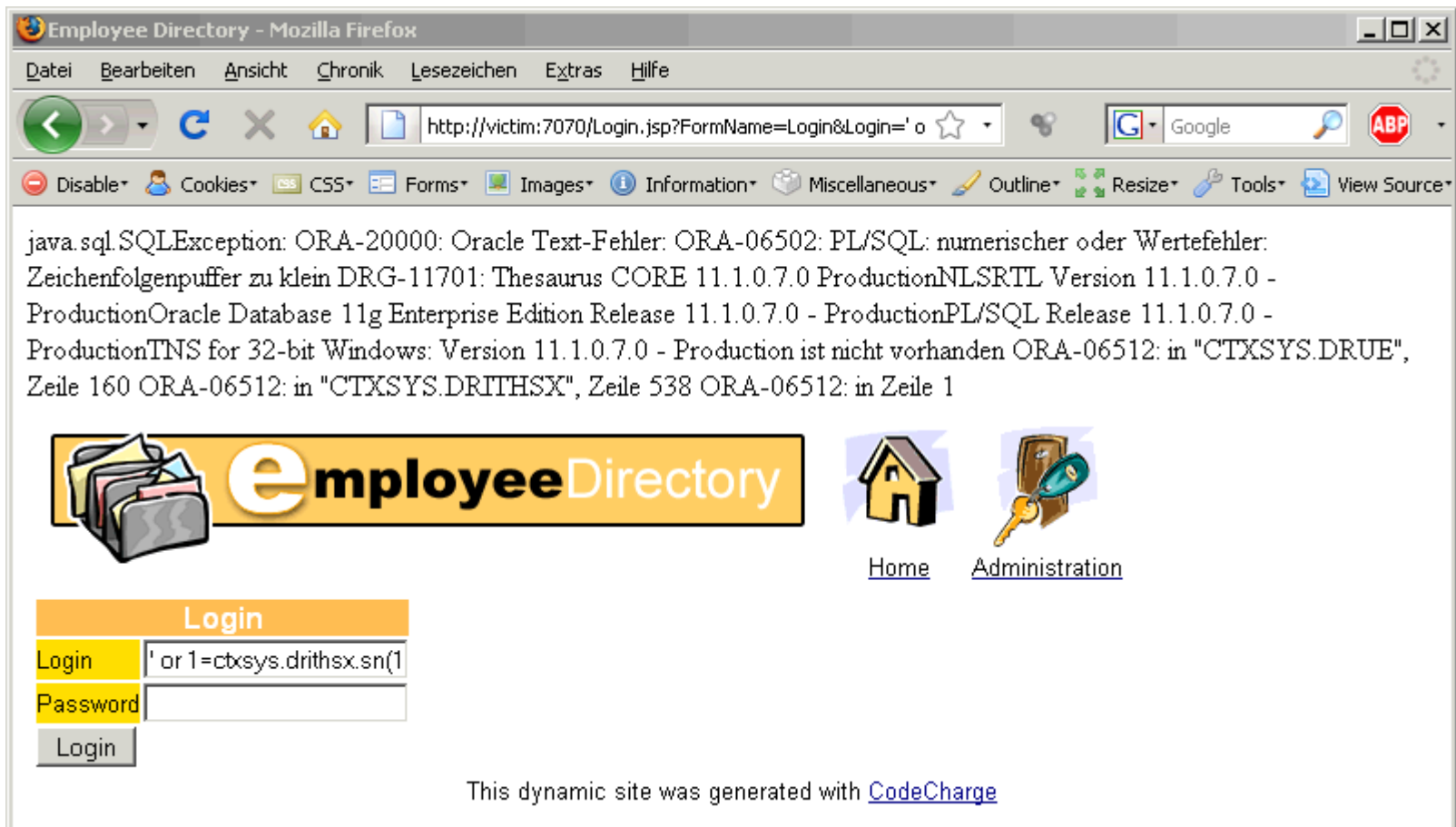
What is the most dangerous package in an Oracle database?

- **dbms_sql** (No. 1, allows privilege escalation)
- `utl_file`
- `utl_mail`
- `utl_inaddr`
- **utl_tcp** (No. 3, overtake the DB via TNS Listener)
- `dbms_lob`
- **dbms_xmlgen** (No. 2, steal the entire DB with a single SQL Injection)
- `dbms_aw_xml`
- `ctxsys.drithsx`
- `ordsys.ord_dicom`
- `kupp$proc`

PL/SQL Packages - Sample

Via a vulnerable web application it is possible to retrieve information via error messages

' or 1=ctxsys.drithsx.sn(1,(select sys.stragg(distinct banner)||' ' from v\$version))--



The screenshot shows a Mozilla Firefox browser window titled "Employee Directory - Mozilla Firefox". The address bar contains the URL: `http://victim:7070/Login.jsp?FormName=Login&Login=' or 1=ctxsys.drithsx.sn(1,(select sys.stragg(distinct banner)||' ' from v$version))--`. The browser's developer tools or error console displays the following Oracle database error message:

```
java.sql.SQLException: ORA-20000: Oracle Text-Fehler: ORA-06502: PL/SQL: numerischer oder Wertefehler:  
Zeichenfolgenpuffer zu klein DRG-11701: Thesaurus CORE 11.1.0.7.0 ProductionNLSRTL Version 11.1.0.7.0 -  
ProductionOracle Database 11g Enterprise Edition Release 11.1.0.7.0 - ProductionPL/SQL Release 11.1.0.7.0 -  
ProductionTNS for 32-bit Windows: Version 11.1.0.7.0 - Production ist nicht vorhanden ORA-06512: in "CTXSYS.DRUE",  
Zeile 160 ORA-06512: in "CTXSYS.DRITHSX", Zeile 538 ORA-06512: in Zeile 1
```

Below the error message, the web application interface for "Employee Directory" is visible. It includes a navigation bar with "Home" and "Administration" links, and a "Login" form with the following fields:

Login	
Login	' or 1=ctxsys.drithsx.sn(1
Password	
<input type="button" value="Login"/>	

At the bottom of the page, it states: "This dynamic site was generated with [CodeCharge](#)".

Revoke Public Privileges I

utl_ and dbms_**

These packages are powerful and allow network access (e.g. `utl_tcp`, `utl_http`,...), file access (`dbms_advisor`, `utl_file`, ...), unsecure (`dbms_random`) or other powerful operations (e.g. `dbms_obfuscation_toolkit`). Execution privileges on these package should not be granted to public.

Command (as user SYS):

```
SQL> revoke execute on utl_http from public force;
SQL> revoke execute on utl_tcp from public force;
SQL> revoke execute on utl_file from public force;
SQL> revoke execute on utl_inaddr from public force;
SQL> revoke execute on utl_smtp from public force;
SQL> revoke execute on utl_dbws from public force;
SQL> revoke execute on dbms_lob from public force;
SQL> revoke execute on dbms_random from public force;
SQL> revoke execute on dbms_obfuscation_toolkit from
public force;
```

Revoke Public Privileges II

```
SQL> revoke execute on dbms_crypto_toolkit from public force;
SQL> revoke execute on dbms_advisor from public force;
SQL> revoke execute on dbms_ldap from public force;
SQL> revoke execute on dbms_ldap_utl from public force;
SQL> revoke execute on dbms_job from public force;
SQL> revoke execute on dbms_scheduler from public force;
SQL> revoke execute on dbms_ddl from public force;
SQL> revoke execute on dbms_epg from public force;
SQL> revoke execute on dbms_xmlgen from public force;
SQL> revoke execute on dbms_aw_xml from public force;
SQL> revoke execute on ctxsys.drithsx from public force;
SQL> revoke execute on ordsys.ord_dicom from public force;
```


Revoke dbms_sql from public

dbms_sql

dbms_sql allows privilege escalation via the cursor technique. This problem is fixed in Oracle 11g but still possible in all previous Oracle versions.

Command (as user SYS):

```
SQL> create role ROLE_DBMSSQL;
SQL> grant execute on dbms_sql to ROLE_DBMSSQL;
SQL> spool grantdbmssql.sql
SQL> select distinct 'grant ROLE_DBMSSQL to
''||owner||'';' from all_dependencies where
referenced_name = 'DBMS_SQL' and owner not in
('PUBLIC');
SQL> spool off
SQL> @grantdbmssql
SQL> revoke execute on dbms_sql from PUBLIC;
```

Revoke public privileges from Object Types

To harden the database it is necessary to revoke some privileges from mighty object types.

HTTPUriType

This object type allows every user to do HTTP-request. This can be used in SQL Injection attacks to transfer data out of the database.

Command (as user SYS):

```
SQL> revoke execute on HTTPUriType from public force;
```

Database Trigger

Using Database trigger (LOGON, LOGOFF, DDL, GRANT, ERROR, SHUTDOWN, STARTUP) is a easy and powerful way to control the database. Especially DLL trigger and Error trigger can help to achieve a better control over the database.

DDL_TRIGGER

This trigger is monitoring all DDL modifications (grant, alter, create, drop) on the production database. It's necessary to change the IP address inside the trigger.

Command (as user SYS):

```
SQL> create or replace trigger DDLTrigger
AFTER DDL ON DATABASE
DECLARE
    rc VARCHAR(4096);
BEGIN
    begin
rc:=utl_http.request('http://192.168.2.201/user='||ora_login_user||';
DDL_TYPE='||ora_sysevent||';DDL_OWNER='||ora_dict_obj_owner||';DDL_NAME='||ora_dict_obj_name||';sysdate='||to_char(sysdate, 'YYYY-MM-DD
hh24:mi:ss');
        exception
            when utl_http.REQUEST_FAILED then null; end;
END;
/
```

Logon Trigger

All logon requests should be monitored with a tamperproof audit log. This could be implemented by using the a database logon trigger. This trigger is sending all logon activities to a webserver. It's necessary to change the IP Address.

Command (as user SYS):

```
SQL> create or replace trigger sec_logon after logon on database
DECLARE
    rc VARCHAR(4096);
begin
    begin
rc:=utl_http.request('http://192.168.2.201/logon_user='||user||';sessionid
='||sys_context('USERENV','SESSIONID')||';host='||sys_context('USERENV','H
OST')||';ip='||ora_client_ip_address||';sysdate='||to_char(sysdate, 'YYYY-
MM-DD hh24:mi:ss')));
    exception
        when utl_http.REQUEST_FAILED then null; end;
End sec_logon;/
```

Error trigger (optional)

This trigger is storing all Oracle error messages occurred on the server. This is really useful to detect attacks, e.g. from SQL Injection

Command (as user SYS):

```
SQL> CREATE OR REPLACE TRIGGER after_error
AFTER SERVERERROR ON DATABASE
DECLARE pragma autonomous_transaction; id NUMBER;
sql_text ORA_NAME_LIST_T; v_stmt CLOB; n NUMBER;
BEGIN
n := ora_sql_txt(sql_text);
IF n >= 1 THEN
FOR i IN 1..n LOOP
v_stmt := v_stmt || sql_text(i);
END LOOP;
END IF;
FOR n IN 1..ora_server_error_depth LOOP
IF ora_server_error(n) in (
'900','906','907','911','917','920','923','933','970','1031','1476','1719'
,'1722','1742','1756','1789','1790','24247','29257','29540') THEN
INSERT INTO system.oraerror VALUES (SYS_GUID(), sysdate, ora_login_user,
ora_client_ip_address, ora_server_error(n), ora_server_error_msg(n),
v_stmt);
END IF; END LOOP;
END after_error; /
```

Oracle Auditing – Problems and Issues

Oracle Auditing is a 95% solution. If you can live with a 95% solution Oracle Auditing will be sufficient for you.

Oracle Auditing problems:

- can be bypassed using various ways
- interesting statement/object can not be audited
- sometimes the wrong statement is logged

Oracle Auditing – Bypassing Auditing

The following problem was fixed with the January CPU 2009. Running a job with any PL/SQL statement via dbms_ijob does not leave any traces...

```
Declare
```

```
jj      integer := 666666;      -- job number
begin sys.dbms_ijob.submit(JOB =>          jj,
LUSER =>      'SYS', PUSER =>      'SYS', CUSER =>      'SYS',
NEXT_DATE => sysdate, INTERVAL =>      null,
BROKEN =>      false, WHAT =>
    ' declare jj      integer := '||jj||';
begin execute immediate 'alter system archive log
current';
sys.dbms_ijob.remove(jj);
delete from sys.aud$ where obj$name = 'DBMS_IJOB';
commit;
end;', sys.dbms_ijob.run(jj);
end;
/
```


Oracle Auditing – Important objects not auditable

Important objects can not be audited. It is not possible to audit important tables like sys.user\$. This table contains all user / role and password information from the Oracle database.

A password change could be performed by updating the table directly.

```
SQL> update sys.user$ set password = 'D4DF7931AB130E37'  
where name='SYSTEM';
```

This can not be audited.

```
SQL> audit all on sys.user$;  
audit all on sys.user$  
ERROR at line 1:  
ORA-00701: object necessary for warmstarting database  
cannot be altered
```

Oracle Auditing – Important objects not auditable II

Another way to bypass Oracle Auditing is to modify the data dictionary object directly. A user is normally created with the command "CREATE USER myuser identified by mypassword".

Instead of using "CREATE USER" we can get the same result using "CREATE ROLE" plus an "UPDATE SYS.USER\$"

```
SQL> create role myuser identified by mypassword;
```

```
-- convert a role into a user
```

```
SQL> update sys.user$ set type#=1 where name='MYUSER';
```

```
-- alternative update, creates an invisible database user
```

```
SQL> update sys.user$ set type#=2 where name='MYUSER';
```

Oracle Auditing – Wrong statements logged

Since Oracle 10g it is possible to log the statement which caused the audit entry.

This sounds like a good feature but the database is sometimes (e.g. if VPD, QueryRewrite, ... is used)modifying the SQL statement which was submitted. In this case Oracle is auditing the previous statement and not the statement which was executed.

This technique can be used to steal information from audited tables without leaving traces...

Enable Auditing

Audit interesting activities.

Command (as user SYS):

```
AUDIT CREATE USER BY ACCESS;
```

```
AUDIT ALTER USER BY ACCESS;
```

```
AUDIT DROP USER BY ACCESS;
```

```
AUDIT CREATE ROLE BY ACCESS;
```

```
AUDIT SELECT ON DBA_USERS BY ACCESS;
```

```
AUDIT CREATE EXTERNAL JOB BY ACCESS; -- 10g Rel.2
```

```
AUDIT CREATE JOB BY ACCESS;      -- 10g Rel.1
```

```
AUDIT CREATE ANY JOB BY ACCESS;
```

```
AUDIT CREATE ANY LIBRARY BY ACCESS;
```

```
AUDIT ALTER DATABASE BY ACCESS;
```

```
AUDIT ALTER SYSTEM BY ACCESS;
```

```
AUDIT AUDIT SYSTEM BY ACCESS;
```

```
AUDIT EXEMPT ACCESS POLICY BY ACCESS;
```

```
AUDIT GRANT ANY PRIVILEGE BY ACCESS;
```

Command (as user SYS):

```
AUDIT GRANT ANY ROLE BY ACCESS;  
AUDIT ALTER PROFILE BY ACCESS;  
AUDIT CREATE ANY PROCEDURE BY ACCESS;  
AUDIT ALTER ANY PROCEDURE BY ACCESS;  
AUDIT DROP ANY PROCEDURE BY ACCESS;  
AUDIT CREATE PUBLIC DATABASE LINK BY ACCESS;  
AUDIT CREATE PUBLIC SYNONYM BY ACCESS;  
AUDIT EXECUTE ON DBMS_FGA BY ACCESS;  
AUDIT EXECUTE ON DBMS_RLS BY ACCESS;  
AUDIT EXECUTE ON DBMS_FILE_TRANSFER BY ACCESS;  
AUDIT EXECUTE ON DBMS_SCHEDULER BY ACCESS;  
AUDIT EXECUTE ON DBMS_JOB BY ACCESS;  
AUDIT SELECT ON SYS.V_$SQL BY ACCESS;  
AUDIT SELECT ON SYS.GV_$SQL BY ACCESS;  
AUDIT EXECUTE ON SYS.KUPP$PROC BY ACCESS;  
AUDIT EXECUTE ON DBMS_XMLGEN BY ACCESS;  
AUDIT EXECUTE ON DBMS_NETWORK_ACL_ADMIN BY ACCESS; -- 11g
```

Recompile all view

To get rid of the "create view" problem it is necessary to recompile all views. This can be done with the script

"\$ORACLE_HOME/CPU/cpuapr2008/view_recompile/view_recompile_apr2008cpu.sql". This can take up to 4 hours depending of the size of your database.

Command (as user SYS):

```
cd $ORACLE_HOME/CPU/cpuapr2008/view_recompile
SQL> @view_recompile_apr2008cpu.sql
```

Next steps & Summary

This was just the baseline security for Oracle databases. If you need more this baseline

- ***Check your own application code***
- ***Train the DBAs, Developers and Security People***
- ***Perform regular security audit***
- ***Run database scanners regularly***
- ***Use 3rd-party products to increase the security***

Oracle Password Checker:

<http://www.red-database-security.com/software/checkpwd.html>

<http://www.red-database-security.com/software/repscan.html>

Exploit Code for January 2009 CPU:

<http://blog.red-database-security.com/2009/01/21/exploit-for-january-cpu-2009-published/>

http://blog.red-database-security.com/2009/01/16/proof-of-concept-how-to-bypass-oracle-auditing-using-dbms_ijob/

Oracle Security Checklist:

http://www.oracle.com/technology/ deploy/security/database-security/pdf/twp_security_checklist_database.pdf

Oracle SQL Injection Tutorial:

<http://blog.red-database-security.com/2009/01/17/tutorial-oracle-sql-injection-in-webapps-part-i/>

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: [info @ red-database-security.com](mailto:info@red-database-security.com)