Oracle Database Security
Top Things You Could & Should Be Doing Differently

Simon Pane – November 17, 2016

Pythian
About ME

Pythian Solution Architect

• Working with Oracle DB since version 6
• Oracle Certified Professional: Oracle 8, 8i, 9i, 10g, 11g, 12c
• Oracle Certified Expert
• Oracle Certified Partner Specialist
• Oracle ACE Associate
• MOS Communities Simon_DBA
ABOUT PYTHIAN

Pythian’s 400+ IT professionals help companies adopt and manage disruptive technologies to better compete.
Systems currently managed by Pythian

11,800

Pythian experts in 35 countries

400

Millennia of experience gathered and shared over 19 years

2
TECHNICAL EXPERTISE

CLOUD
Using the disruptive nature of cloud for accelerated, cost-effective growth

ADVANCED ANALYTICS
Mining data for insights & business transformation using data science

DATABASES
Ensuring databases are reliable, secure, available and continuously optimized

BIG DATA
Harnessing the transformative power of data on a massive scale

DEVOPS
Providing critical velocity in software deployment by adopting DevOps practices

INFRASTRUCTURE MANAGEMENT
Transforming and managing the IT infrastructure that supports the business
This session is **NOT** about

All great products / tools:
- Oracle Database Vault
- Oracle Audit Vault and Database Firewall
- Privilege Analysis
- Oracle Key Vault
- Oracle Identity Management
- Transparent Data Encryption
- Oracle Data Redaction
- VPD / FGAC / Real Application Security (RAS)
- Oracle Data Masking and Subsetting
- Oracle Total Recall
- Advanced Security Option

…. Instead it is about

- What can we do with what we already have?
- What we need to watch out for?
- How should we change our operational practices?
AGENDA

“Top Recommendations”

• Applies to Standard Edition & Enterprise Edition
• Various releases: 10g, 11g, 12c

The take away: *tips, suggestions, cautions!*
Avoid Clear Text Passwords in Oracle Net
Avoid Clear Text Password in Oracle Net

• **Authentication** is automatically encrypted - other commands are not!

• Never use “... identified by ...” through Oracle Net
  • What about our tools?

• Use the “**password**” command instead
  • This command does encrypt the password in flight
Avoid Clear Text Password in Oracle Net

• Easy for a DBA to trace (just to prove):

```bash
$ cat $ORACLE_HOME/network/admin/sqlnet.ora
trace_level_client=support
trace_directory_client=/tmp/trace
diag_adr_enabled=OFF
```

• Easy for anyone else on the network to trace
  • Network sniffer (i.e. WireShark)
Avoid Clear Text Password in Oracle Net

• Resulting trace file:

```
[21-AUG-2015 16:27:20:058] nsbasic_bsd: 00 00 00 00 00 27 61 6C |......'al|
[21-AUG-2015 16:27:20:058] nsbasic_bsd: 64 65 6E 74 79 70 70 72 |sWord...|
```

• Using `$ORACLE_HOME/bin/bin/trcasst`: (11g)
Avoid Clear Text Password in Oracle Net

• What about if you’re going through a public network?
  • Or between services in a public cloud?

• From Amazon RDS documentation:

  2. After Amazon RDS has installed Statspack on your DB instance, you must log in to the DB instance using your master user name and master password. You must then reset the PERFSTAT password from the randomly generated value Amazon RDS created when Statspack was installed. After you have reset the PERFSTAT password, you can log in using the PERFSTAT user account and run the Statspack scripts.

  Use the following command to reset the password:

  ALTER USER perfstat IDENTIFIED BY <new_password> ACCOUNT UNLOCK;
Avoid Clear Text Password in Oracle Net

- Wireshark to AWS with VPC security group:
Avoid Clear Text Password in Oracle Net

• Better method (example: AWS RDS using SQL Developer):
Oracle Net Encryption

• Network encryption no longer part of ASO

  (TDE) master encryption keys. Strong authentication services (Kerberos, PKI, and RADIUS) and network encryption (native network encryption and SSL/TLS) are no longer part of Oracle Advanced Security and are available in all licensed editions of all supported release of the Oracle database.

  ▪ Source: http://docs.oracle.com/database/121/DBLIC/editions.htm

• Backported to 11gR2:

  Network encryption (native network encryption and SSL/TLS) and strong authentication services (Kerberos, PKI, and RADIUS) are no longer part of Oracle Advanced Security and are available in all licensed editions of all supported releases of the Oracle database.

  ▪ Source: http://docs.oracle.com/cd/E11882_01/license.112/e47877/options.htm#DBLIC143
Oracle Net Encryption

• Network encryption can be as simple as two parameters in the `sqlnet.ora` file:

```plaintext
SQLNET.ENCRYPTION_SERVER = required
SQLNET.ENCRYPTION_TYPES_SERVER = (RC4_256, AES256)
```

• Many other options also available including TLS
Protect Password
Hash Values & Salts
Protect Password Hash VALUES & SALTS

• Easy to crack offline (particularly old 10g DES hashes)
• Protect network traffic:
  • “Oracle Network Encryption” or other tunneling software
  • Exposures pre-11.2.0.4 – see CVE-2012-3137 – Fixed OCT2012 PSU
• Protect and audit in the catalog:
  • SYS.USER_HISTOR Y$ and SYS.USER$.SPARE4 (includes the salt)
• Consider Data Pump export files
• Force complexity: Use a password-verification function and profiles
Protect Password Hash VALUES & SALTS

• Never post hashes into documentation or ticket system:

**Implementation Steps:**

```sql
alter user SAP identified by **********;
```

**Back Out Steps:**

```sql
alter user SAP identified by values 'S:50B3BDC919C94145916327B9349BE1799523A66C1AFE8BB89BFF11F071D87; 7B4659B2953D7DF1';
```

© 2016 Pythian. Confidential
Avoid Passwords in Script Variables
Avoid Passwords in Script Variables

• Don’t put passwords into script variables:
  • Variables can be exposed in diagnostic dumps
  • Also shouldn’t ever need to “export” variables!!

• Scripting mistake:

```bash
#!/bin/bash
export DB_USER=backup_user
export DB_PASS=`./secure_pwd_extractor`
#$ export DB_PASS=backltup
$ORACLE_HOME/bin/rman << EOF
  connect target $DB_USER/$DB_PASS
  shutdown immediate
  startup mount
  backup database;
  alter database open;
EOF
```
Avoid Passwords in Script Variables

• Listener dump:

```
LSNRCTL for Linux: Version 11.2.0.4.0 - Production on 21-AUG-2015 16:45:21
Copyright (c) 1991, 2013, Oracle. All rights reserved.
Welcome to LSNRCTL, type "help" for information.
LSNRCTL> set displaymode verbose
Service display mode is VERbose
LSNRCTL>
LSNRCTL> services
Connecting to (ADDRESS=(PROTOCOL=tcp)(HOST=)(PORT=1521))
Services Summary...
Service "ORA11G" has 1 instance(s).
  Instance "ORA11G", status READY, has 1 handler(s) for this service...
    Handler(s):
      "DEDICATED" established:10 refused:0 state:ready
      LOCAL SERVER
        (ADDRESS=(PROTOCOL=TCP) (PROGRAM=/u01/app/oracle/product/11.2.0/dbhome_1/bin/oracle) (PROTOCOL=TCP) (PROGRAM=/u01/app/oracle/product/11.2.0/dbhome_1/bin/oracle)
        RALINUX, SELINUX_ROLE_REQUESTED=, DB_PASS=backup, SHELL=/bin/bash, TERM=xterm, HISTSIZE=1000, SS
```
Avoid Passwords in Script Variables

• Variables are in: /proc/\$\{pmon\_pid\}/environ

```bash
strings /proc/\`ps -ef | awk '/[p]mon/\{print \$2\}\`/environ | grep -i DB_  
DB\_PASS=back\_up  
DB\_USER=backup\_user
```

• And RMAN script when running:

```bash
ps -ef | awk '/[r]man/\{print \$2\}'  
21355

strings /proc/21355/environ | grep -i pass  
DB\_PASS=back\_up
```
Avoid Passwords in Script Variables

• Better solution: “Secure External Password Store” (SEPS)
• Doesn’t negate “/ AS SYSDBA” connections
• Compatible with SQLNET.AUTHENTICATION_SERVICES=NONE

```bash
$ mkstore -wrl "/u01/app/oracle/wallet" -listCredential
Oracle Secret Store Tool : Version 11.2.0.3.0 - Production
Copyright (c) 2004, 2011, Oracle and/or its affiliates. All rights reserved.

Enter wallet password:

List credential (index: connect_string username)
1: ORCL monitoring_user
$
Avoid Passwords in Script Variables

• “Secure External Password Store” usage:

```
$ whoami
oracle
$ sqlplus /@ORCL

SQL*Plus: Release 11.2.0.3.0 Production on Tue Oct 20 17:35:07 2015
Copyright (c) 1982, 2011, Oracle. All rights reserved.

Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.3.0 - Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options

SQL> show user
USER is "MONITORING_USER"

RMAN> connect target "/@ORCL"
connected to target database: ORCL (DBID=1333945358)
RMAN>
```
Use the Scheduler
Use the Scheduler

• CRON scripts often run .SQL files
• Monitoring jobs
• Application jobs
• Backup jobs

• How are the DB credentials for those secured?
• Why schedule from the CRON and then have to worry about DB credentials?
Use the Scheduler

- Run DB work directly in the DB if possible → PL/SQL

- But if necessary can also use the Scheduler to run OS scripts
  - (.sh or .cmd)

- If shell access is required to run OS scripts:
  - Use a Scheduler Credential (11g and 12c) ← Preferred
  - Use externaljob.ora (10g, 11g, and 12c) ← Not recommended
RISK: Using the Scheduler Without a Credential

• OS Security context for *non*-SYS jobs to use when running external jobs is defined in:

  $ORACLE_HOME/rdbms/admin/externaljob.ora

• Be careful this doesn’t become an OS-injection risk
  ▪ The program being run could be “sh”
  ▪ Actual OS commands could be the arguments
RISK: Using the Scheduler Without a Credential

• Default UNIX settings in externaljob.ora:
  run_user = nobody
  run_group = nobody

• Be cautious if changing to a more powerful account
  run_user = oracle
  run_group = dba

© 2016 Pythian. Confidential
RISK: Using the Scheduler Without a Credential

- Windows adds the service: `OracleJobScheduler<ORACLE_SID>`
- Service ownership defaults to `LocalSystem` (11g)
- Change to a low level user or disable (default)
- This service runs non-SYS scheduler jobs
Use the Scheduler

• Scheduler best practices:

1. Don’t shell out to the OS if possible (use PL/SQL instead)
2. If running OS scripts use Scheduler Credentials (11g & 12c) instead of externaljob.ora
3. Use a dedicated least-privileged DB account (never “SYS”)
4. Use a dedicated least-privileged OS account (not “oracle”)
5. Use the new 12c job type: SQL_SCRIPT
Set Security Initialization Parameters
Set Security Initialization parameters

• Often overlooked initialization parameters:

SEC_MAX_FAILED_LOGIN_ATTEMPTS
  • Not password attempts – client to server process

SEC_PROTOCOL_ERROR_FURTHER_ACTION
  • What the server process does when receiving bad packets

SEC_PROTOCOL_ERROR_TRACE_ACTION
  • Trace, log, alert, or nothing with bad packets

SEC_RETURN_SERVER_RELEASE_BANNER
  • Is the DB version returned to the client

Changes in 12.1.0.2
Set Security Initialization parameters

• In `sqlnet.ora` pre-12c:
  `SQLNET.ALLOWED_LOGON_VERSION`

• In `sqlnet.ora` as of 12c:
  `SQLNET.ALLOWED_LOGON_VERSION_SERVER`
  `SQLNET.ALLOWED_LOGON_VERSION_CLIENT`

• Controls “authentication protocol”:
  • i.e. O3L, O4L, O5L, O5L_NP, O7L_MR
  • `DBA_USERS.PASSWORD_VERSIONS`

© 2016 Pythian. Confidential
Consider Secure Application Design
Consider Secure Application Design

- Secure application design would include separate schemas for:
  - Data - Code - Access (users)

- Implement data & code separation using:
  - Fully qualified object references or private synonyms
  - Promote a “least privilege” model

- Data schemas (with full DML) can be **LOCKED**!
  - If not, there is always a risk
Consider Secure Application Design

• Even more secure if all data interaction is through PL/SQL objects (APIs)?
• Controls use of object privileges
• No dynamic SQL
• Use bind variables and/or `DBMS_ASSERT` to prevent SQL Injection:

```
SELECT * FROM users WHERE name = '' OR '1'='1';
```

```
SELECT * FROM users WHERE name = '' OR '1'='1' -- ';
```

Source: https://en.wikipedia.org/wiki/SQL_injection
Consider Secure Application Design

- **Data Holding Schema**
  - Locked
  - Can’t perform DDL on base tables
  - Sanitize inputs with `DBMS_ASSERT`

- **Code Schema**
  - Locked

- **EXECUTE**

- **Users**
  - Open
  - Can’t perform DDL on tables or code
  - Can’t perform ad-hoc queries
Use Role Security
Use Role Security

• Can be a solution to try to prevent “application bypass”
• TOAD / SQL Developer / SQL*Plus connections

• Typically protect with a password or a package:

```sql
CREATE ROLE <role> IDENTIFIED BY <password>;
CREATE ROLE <role> USING <schema.package>;
```
Use Role Security

• Activate password protected roles:

  SET ROLE <role> IDENTIFIED BY <password>;

• Limitations of password protected roles:
  • Passwords sent as clear-text through Oracle Net
  • Pre-11.2.0.4: by-pass through nesting:

  GRANT secure_role TO non_secure_role;
  GRANT non_secure_role TO SCOTT;
Use Role Security

• Application roles:
  • Authentication logic resides in PL/SQL package
    ▪ Must use “invoker’s rights”
    ▪ Must include one or more security checks (i.e. using SYS_CONTEXT)
    ▪ Must issue SET ROLE or DBMS_SESSION.SET_ROLE

• After granting, manually change to not be a “default role” – otherwise logic is bypassed!
Listener Protection
Listener Protection

• Valid Node Checking
• Provides basic firewall functionality

• Only allow connections from authorized application servers and DBA desktops
  ▪ Prevent “application by-passing”

• Prevent connections from un-authorized servers
  ▪ From TEST/DEV/QA to PROD
  ▪ Through DB links after refreshes

© 2016 Pythian. Confidential
DB Links and passwords should be adjusted after a RMAN duplicate but often are missed.
Listener Protection

• Valid Node Checking implementation:
  • Three parameters in server’s sqlnet.ora:
    
    \[
    \begin{align*}
    \text{tcp.validnode\_checking} &= \{\text{YES, NO}\} \\
    \text{tcp.invited\_nodes} &= ( \{\text{hostnames/IPs}\} ) \\
    \text{tcp.excluded\_nodes} &= ( \{\text{hostnames/IPs}\} )
    \end{align*}
    \]

• Can’t use hostname or IP ranges (no wildcards)
• “invited’ and “excluded” are mutually exclusive
Listener Protection

• Protect the database from a connection storm through listener parameters:
  RATE_LIMIT
  CONNECTION_RATE_<listener>

• But there’s still risks:
  • TNS connections can by-passing the listener and connect directly to dispatchers (if port is discovered)!
    ▪ Check the default shared server related initialization parameters (“DISPATCHERS”) ↳ XDB!
Protect at the OS Level
Protect at the OS Level

• If someone can copy the .DMP and RMAN backup files they have everything!
• What if backed up to remote device / NFS share?

Make sure owner and group is correct & file permissions are limited
Protect at the OS Level

- Tools can extract data right out of DBF files:

```bash
$ ls -lh /u01/app/oracle/oradata/CDB121/
total 2.7G
drwxr-x--- 3 oracle dba 21 Sep 3 15:19 1CA811B627CB516BE0537B01A8C0DF56
drwxr-x--- 3 oracle dba 21 Sep 3 15:38 1CA815247F9F5221E0537B01A8C08138
drwxr-x--- 3 oracle dba 21 Sep 3 16:03 1EDF4E3155837E5E0537B01A8C09B50
-rw-r----- 1 oracle dba 18M Sep 4 21:12 control01.ctl
-rw-r----- 1 oracle dba 18M Sep 4 21:12 control02.ctl
drwxr-x-- 2 oracle dba 41 Sep 3 15:10 datafile
drwxr-x-- 2 oracle dba 61 Aug 6 08:22 pdbseed
drwxr-xr-x 2 oracle dba 85 Aug 6 10:33 PLUG1
drwxr-xr-x 2 oracle dba 85 Aug 6 10:34 PLUG2
-rw-r----- 1 oracle dba 51M Sep 4 14:47 redo01.log
-rw-r----- 1 oracle dba 51M Sep 4 16:00 redo02.log
-rw-r----- 1 oracle dba 51M Sep 4 21:12 redo03.log
-rw-r----- 1 oracle dba 661M Sep 4 21:12 sysaux01.dbf
-rw-r----- 1 oracle dba 791M Sep 4 21:12 system01.dbf
-rw-r----- 1 oracle dba 67M Sep 4 21:01 temp01.dbf
-rw-r----- 1 oracle dba 1.1G Sep 4 21:12 undotbs01.dbf
-rw-r----- 1 oracle dba 5.1M Sep 4 21:12 users01.dbf
```

Make sure group is correct **and** membership is limited; Ensure (o)thers have no permissions
Use Proxy Authenticated Connections
Use “Proxy Authenticated Connections”

• DBA & Developers don’t need to know any schema passwords (i.e. application schema passwords)

• Use “Proxy Authenticated Connections”
  • “Connect as specified schema using your own credentials”
Use “Proxy Authenticated Connections”

**Example:**

```sql
SQL> alter user PSDBOWNER grant connect through SIMON_DBA;
User altered.

SQL> connect SIMON_DBA[PSDBOWNER]/passw0rd
Connected.
SQL> alter session set current_schema = SCOTT;
Session altered.

SQL> select sys_context('USERENV','SESSION_USER') as session_user,
       2   sys_context('USERENV','SESSION_SCHEMA') as session_schema,
       3   sys_context('USERENV','PROXY_USER') as proxy_id,
       4   user
       5 from dual;

<table>
<thead>
<tr>
<th>SESSION_USER</th>
<th>SESSION_SCHEMA</th>
<th>PROXY_ID</th>
<th>USER</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSDBOWNER</td>
<td>SCOTT</td>
<td>SIMON_DBA</td>
<td>PSDBOWNER</td>
</tr>
</tbody>
</table>

SQL>
```
Use “Proxy Authenticated Connections”

• Viewing configuration:

```
SQL> select proxy, client, flags from proxy_users;

PROXY       CLIENT             FLAGS
------------ ------------------ ------------
SIMON_DBA    PSDBOWNER         PROXY MAY ACTIVATE ALL CLIENT ROLES
```

• Viewing active connections:

```
SQL> select distinct s.sid, s.serial#, s.username, c.authentication_type
  2  from v$session s, v$session_connect_info c
  3  where username is not null
  4  and s.sid = c.sid
  5  and s.sid != sys_context('USERENV','SID');

  SID  SERIAL#  USERNAME    AUTHENTICATION_TYPE
       ------  ----------  ------------------------
     237      17  PSDBOWNER  PROXY
```
Use “Proxy Authenticated Connections”

**DBA_AUDIT_SESSION:**

```sql
SQL> select username, sessionid, proxy_sessionid 
    2   from dba_audit_session;

USERNAME     SESSIONID  PROXY_SESSIONID
-----------------  ---------  ------------
PSDBOWNER      40142     40141
```

**DBA_AUDIT_TRAIL:**

```sql
SQL> select username, sessionid, proxy_sessionid, comment_text 
    2   from dba_audit_trail;

USERNAME     SESSIONID  PROXY_SESSIONID
-----------------  ---------  ------------
PSDBOWNER      40142     40141

Authenticated by: PROXY;EXTERNAL NAME: oracle
```
Use “Proxy Authenticated Connections”

• Advantages:
  • Proxy authenticated connections provide more functionality than “ALTER SESSION SET CURRENT_SCHEMA ... ”
  • Works with most tools and utilities (including wallets)
  • Works with JDBC thin connections
  • All access individualized for audit
  • Easier off-boarding of DBA / Dev staff:
    ▪ Who no longer works here yet still knows the password because it hasn’t been changed!
Wrap Up!
Summary

• Think about DBA and Developer processes:
  • Don’t just focus on profiles and object privileges

• Think outside of the database
  • Protecting backups can be just as important
  • Consider the network and data in-flight
Summary

• Understand the capabilities of the products you already have:
  • Consider new features and changes between versions / releases
  • Explore features and functions that you may not be very familiar with

• Continuous Improvement:
  • **Modernize & evolve** both the technology and process
  • As technology changes
  • As business needs change
  • As technical requirements change
Thanks and Q&A

For Follow-ups:

pane@pythian.com