SSL certificates in the Oracle Database without surprises

Nelson Calero

HrOUG - Croatia Oracle UG
October 18th, 2019

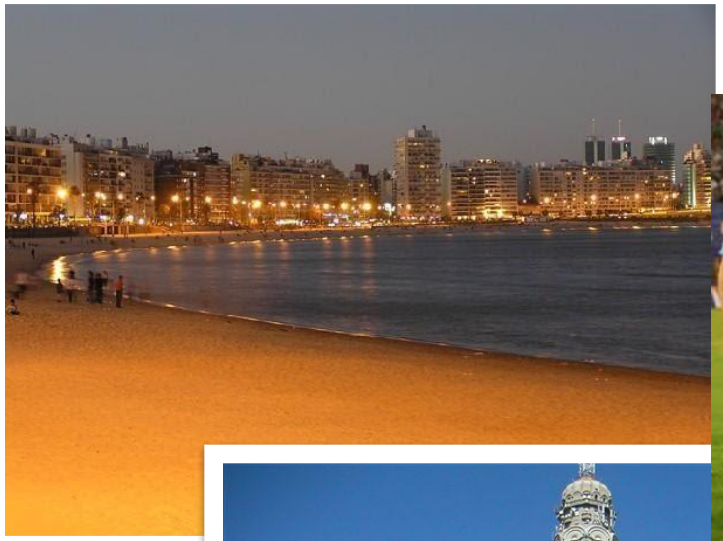
Pythian

24. hroug godišnja konferencija

Today's topics

- Security basis
- Use cases in the Oracle Database
  1. SSL for database connections
  2. Email from PL/SQL
  3. Oracle Enterprise Manager
- Management
- Troubleshooting

Intended audience: DBAs and Developers



About me

- Principal Consultant at Pythian – several roles since 2014
- Working with Oracle tools and Linux environments since 1996
- DBA Oracle (2001) & MySQL (2005)
- Co-founder and President of the Oracle user Group of Uruguay (2009)
- LAOUC Director of events (2013)

- Computer Engineer (1998)
- Oracle ACE (2014), Oracle ACE Director (2017)
- Oracle Certified Professional 10g/11g/12c, OCE, Cloud DB & Infra
- Amazon Solutions Architect – Associate (2016)
- Google Cloud Architect (2017), Google Cloud Data Engineer (2017)
- Oracle University Instructor (2011)
- Blogger and speaker: Oracle Open World, Collaborate, OTN Tour, Regional conferences

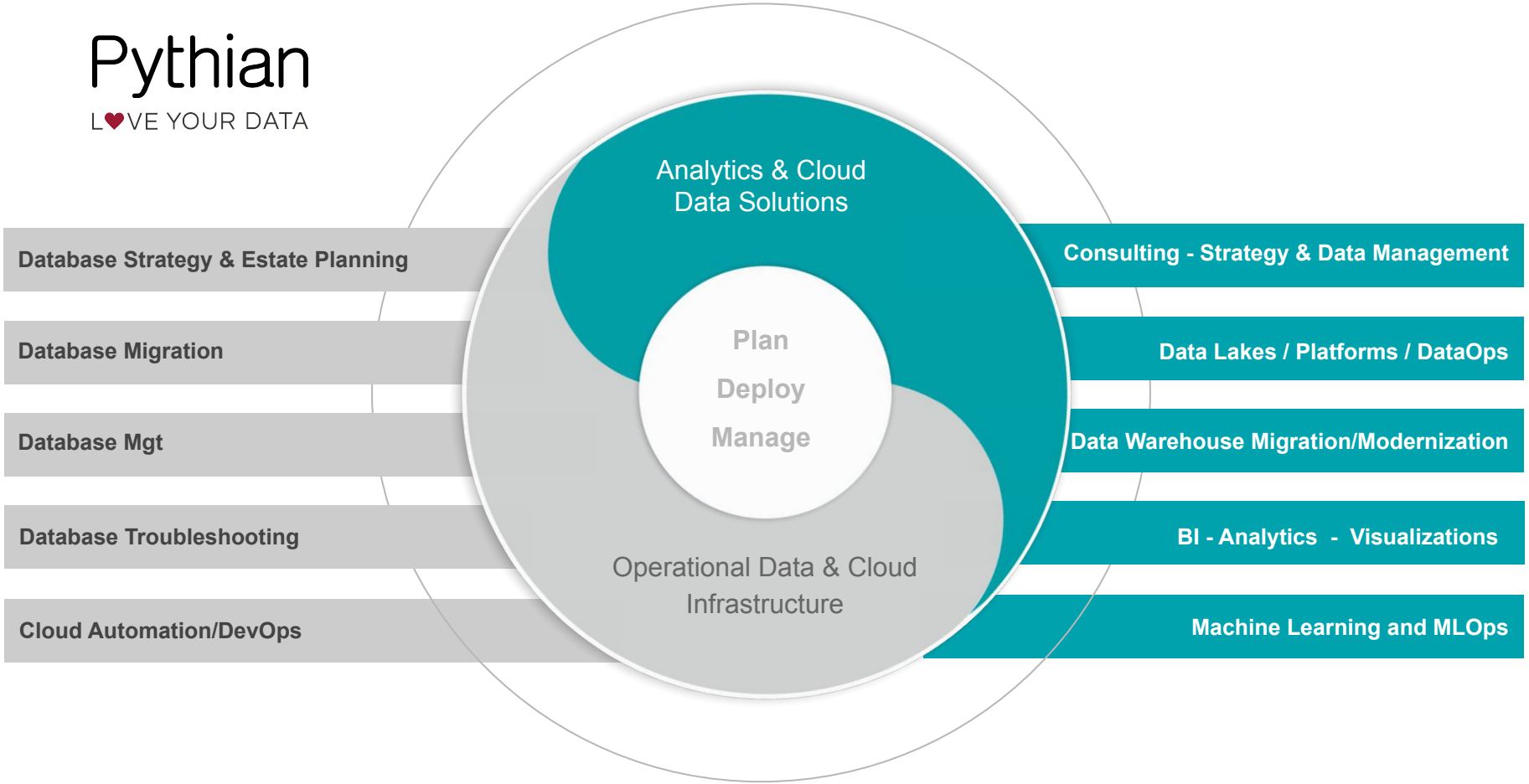

http://www.linkedin.com/in/ncalero


@ncalerouy

Pythian
love your data®

Pythian

LOVE YOUR DATA



500+ Technical Experts Helping Peers Globally

ORACLE ACE Program


ORACLE ACE Director


ORACLE ACE


ORACLE ACE Associate

---

3 Membership Tiers

- Oracle ACE Director
- Oracle ACE
- Oracle ACE Associate

bit.ly/OracleACEProgram

Connect:

oracle-ace_ww@oracle.com

Facebook.com/oracleaces

@oracleace

Oracle Groundbreakers

Nominate yourself or someone you know: **acenomination.oracle.com**

Security Basis

- SSL
- Public/private keys
- PKI
- Certificates
- CA
- Publicly trusted CA
- Signatures
- Certificate chain
- Certificate hierarchy
- Wallets
- Handshake
- Protocols in Oracle versions

Basis – Secure Sockets Layer

- SSL is the standard security technology for establishing an encrypted link between a server and a client
- Provides network-level authentication, data encryption, and data integrity, ensuring that all data passed between them remain private and integral
- Examples:
  - webserver and browser
  - mail server and mail client
- Based on Public Key Cryptography using Public Key Infrastructure (PKI)

SSL with the Oracle database

- Since 10gR2 SSL/TLS are no longer part of the *Advanced Security Option*
- Since 12c SSL/TLS is available in Oracle Standard Edition
- Enabled by default in Oracle Cloud Database services

Basis – SSL History

- SSL - Secure Socket Layers Protocol
  - Developed by Netscape
- TLS - Transport Layer Security Protocol
  - new version of SSL, based on SSL 3.0
  - SSL 2.0 and 3.0 deprecated in 2011 and 2015
- SSL and TLS protocols do not interoperate
- SSL Certificates do not depend on protocol
- Several algorithms to use in signing stages

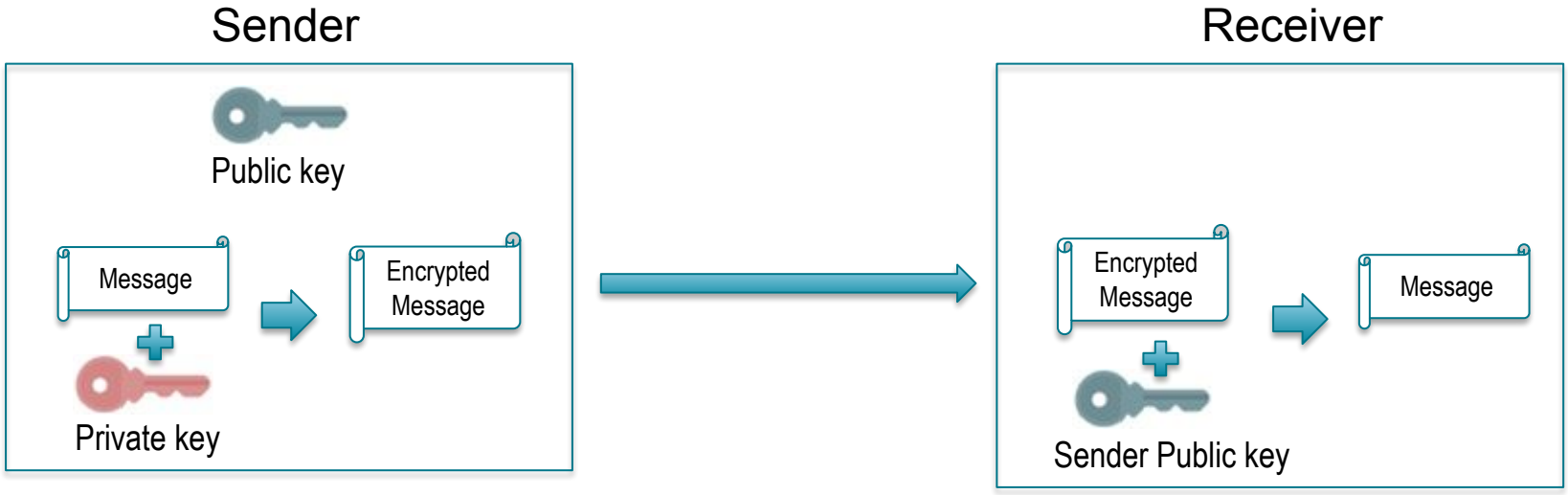
Protocol	Year
SSL 1.0	No
SSL 2.0	1995
SSL 3.0	1996
TLS 1.0	1999
TLS 1.1	2006
TLS 1.2	2008
TLS 1.3	2018

Basis – SSL History

- SSL - Secure Socket Layers Protocol
  - Developed by Netscape
- TLS - Transport Layer Security Protocol
  - new version of SSL, based on SSL 3.0
  - SSL 2.0 and 3.0 deprecated in 2011 and 2015
- SSL and TLS protocols do not interoperate
- SSL Certificates do not depend on protocol
- Several algorithms to use in signing stages

Protocol	Year	Oracle
SSL 1.0	No	
SSL 2.0	1995	
SSL 3.0	1996	
TLS 1.0	1999	11.1 (2007)
TLS 1.1	2006	
TLS 1.2	2008	12.1 (2013)
TLS 1.3	2018	

Basis – Public and private key encryption



- Also known as asymmetric encryption
- Requires more computational power than symmetric encryption
- Vulnerabilities:
  - Brute force: reduced by using larger keys
  - Man in the middle: resolved using signatures and Certificate Authorities (CA)

Basis – Public Key Infrastructure (PKI)

- Framework using public and privates key, certificates and method for key distribution
- SSL/TLS uses a key-exchange algorithm to allow symmetric keys (hybrid cryptosystem), less computationally intensive than using asymmetric keys
- Components:
  - **Certificate Authority (CA):** a trusted third party that certifies the identity of entities, such as users, databases, administrators, clients, and servers.
  - **Certificates**: created when an entity's public key is signed by a trusted certificate authority (CA).
  - **Wallet**: a container that stores authentication and signing credentials, including private keys, certificates, and trusted certificates SSL
  - **Certificates revocation lists**: validity of CA signed certificates

Basis – Certificates

- Used for authentication
- Follows X.509 specification
- Associates public key with an organization's details (AKA DN):
  - A domain name, server name or hostname
  - An organizational identity (i.e. company name) and location
- It also includes
  - CA name, the CA signature, and the certificate effective dates
  - Digital signature (explained later)
- They can be self-signed - for use in test environments

Basis – Certificate Authority (CA)

- A third party trusted by both of the communicating parties (e.g. Verisign)
- Validates, identities and issue/revoke certificates
- The CA uses its private key to encrypt a message
- The CA public key is well known and does not have to be authenticated each time it is accessed (browsers, wallets, etc.)
- Organization can use in-house CA (e.g. MS Certificate services)

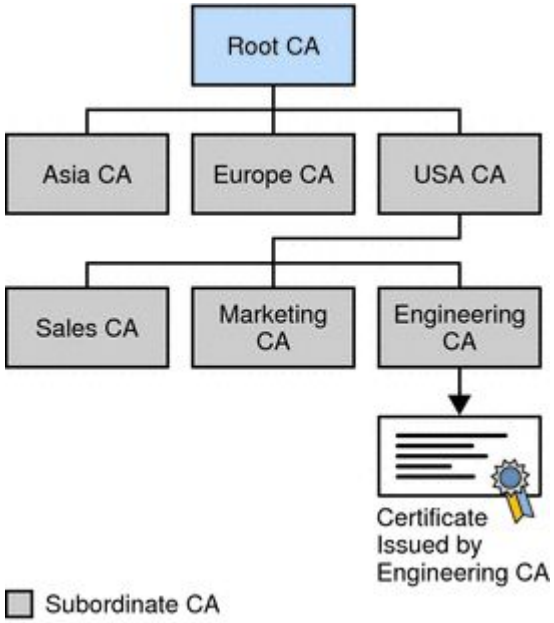
Basis – Signature

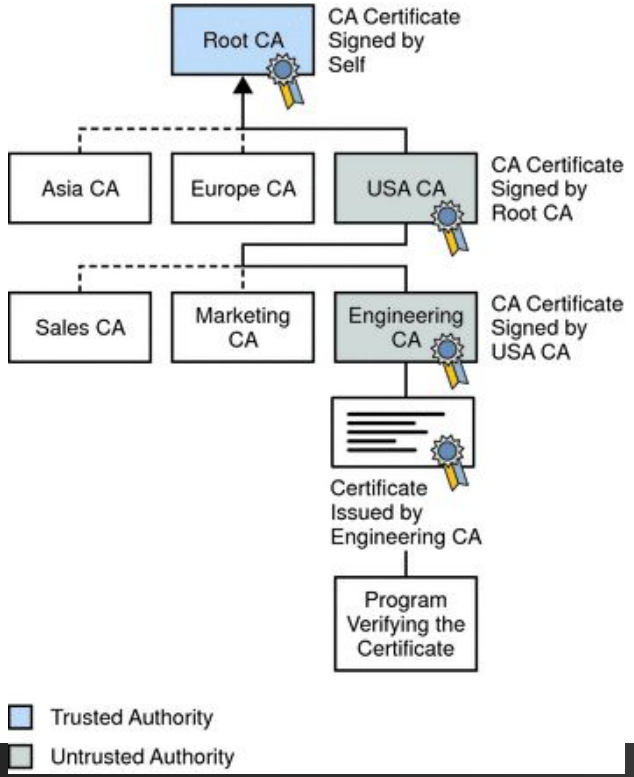
- One-way hash of the data (certificate) encrypted with signer's private key – it cannot be reversed
- Receiver validates the integrity of the data:
  - Receiver gets the data and signature
  - Data is decrypted using sender's public key
  - Signature is decrypted using sender's public key
  - New signature is created using same algorithm
  - Both new and received signature should match if data was not tampered

Basis – Publicly Trusted CAs

- A trusted third party (TTP) used as CA for the certificates
- Commercial
  - Verisign, Digicert, GoDaddy
- Web browsers includes by default public keys of TTPs CA
- De-facto standard for websites, as certificates from non trusted CAs are reported by default as dangerous

Basis – Certificate hierarchy and chain





https://docs.oracle.com/cd/E19424-01/820-4811/gdzen/index.html

Basis – Wallet

- A file storing authentication and signing credentials, including private keys, certificates, and trusted certificates SSL needs.
- Oracle server and client using SSL needs a wallet file
  - configured in sqlnet.ora, listener.ora, optional in tnsnames.ora (instead of sqlnet)
  - Must be auto-login
- Managed with Oracle Wallet Manager and orapki tool
- Creation process for an SSL wallet:
  - Generate a public-private pair
  - Create a certificate request
  - Submit the certificate request to the CA
  - Import the signed server certificate into the wallet
  - Install the wallet into the server
  - Distribute server certificate to clients (exchange w/client)

Basis – SSL handshake in Oracle

1) The client and server establish which cipher suites to use.
   This includes which encryption algorithms are used for data transfers.

2) The server sends its certificate to the client, and the client verifies that the server's certificate was signed by a trusted CA. This step verifies the identity of the server.

3) Similarly, if client authentication is required, the client sends its own certificate to the server, and the server verifies that the client's certificate was signed by a trusted CA.

4) The client and server exchange key information using public key cryptography. Based on this information, each generates a session key. All subsequent communications between the client and the server is encrypted and decrypted by using this session key and the negotiated cipher suite.

Basis – SSL handshake in Oracle

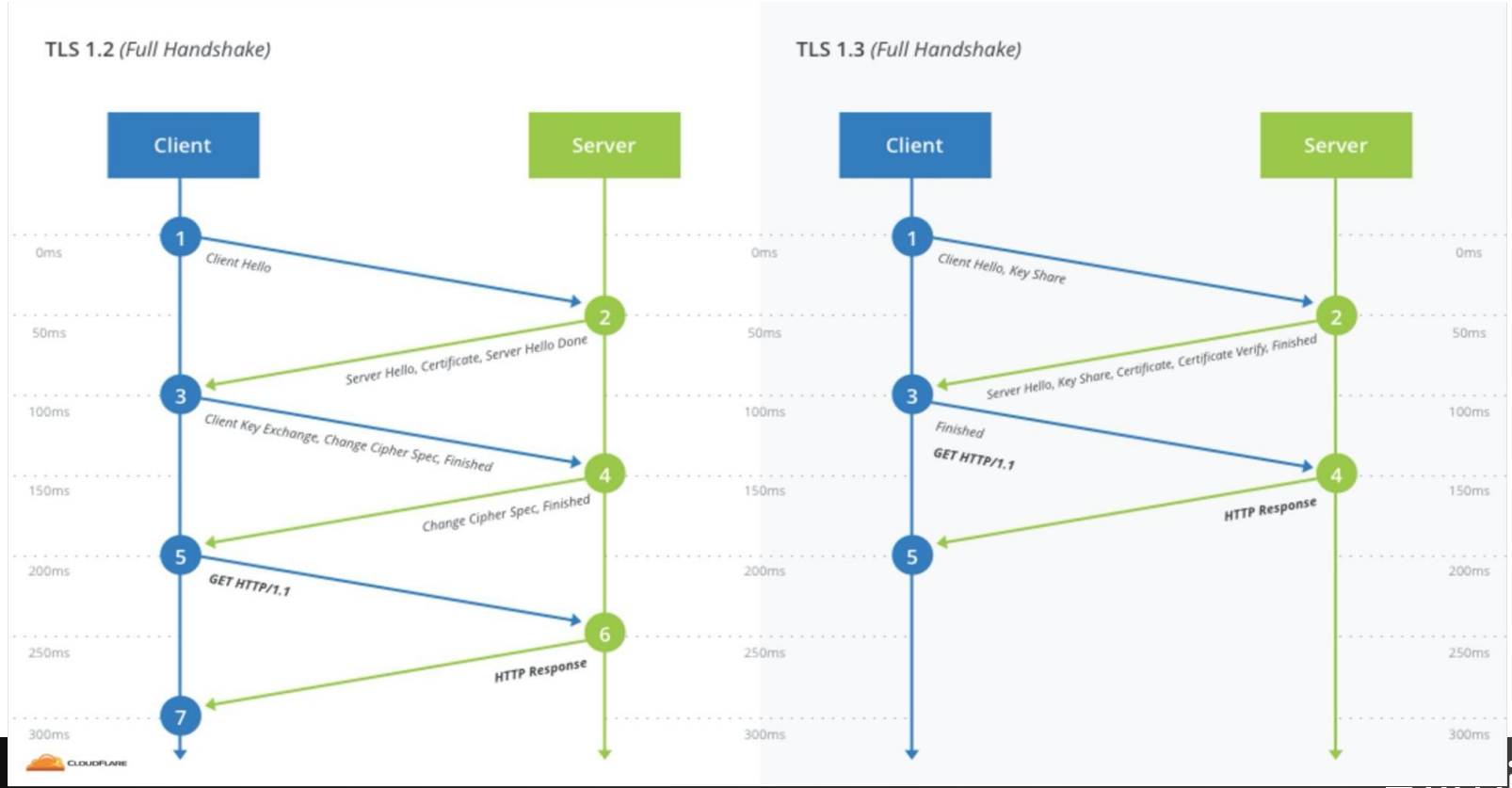
Authentication:

- On a client, the user initiates an Oracle Net connection to the server by using SSL
- SSL performs the handshake between the client and the server
- If the handshake is successful, then the server verifies that the user has the appropriate authorization to access the database

Possible configurations:

- Only the server authenticates itself to the client
- Both client and server authenticate themselves to each other
- Neither the client nor the server authenticates itself to the other, thus using the SSL encryption feature by itself

Basis – handshake changes in TLS v1.3



https://blog.cloudflare.com/introducing-tls-1-3/

Today's topics

- Security basis
- **Use cases in the Oracle Database**
  1. SSL for database connections
  2. Email from PL/SQL
  3. Oracle Enterprise Manager
- Management
- Troubleshooting

Basis - Oracle components

- Wallets
  - Create, import certificates, distribute to clients each time certificates change
- SQL*Net configuration
  - $ORACLE_HOME/network/admin/${sqlnet.ora | listener.ora}
  - Points to wallet file (WALLET_LOCATION keyword)
  - Restricts protocol versions and cipher suites
- Oracle home binaries (client/server)
  - Version and patchlevel defines protocol versions and cipher algorithms supported

Basis – Protocols in Oracle versions

Default TLS version included with Oracle releases:

- Oracle 11.1 - TLS 1.0
- Oracle 12.1 - TLS 1.2

PSUs include update/new algorithms, using MES SDK:

- 11.2.0.4.0 - BSAFE 6.3.2.2 and Certicom SSL plus 3.0 supported SSLv3 (SHA2)
- 12.1.0.2.0 - MESv405 supported TLSv1.2, FIPS 140-2 (MESv405 is deprecated).
- 12.2.0.2.0 - MESv415 supported TLSv1.2, FIPS 140-2 (validated).
- 18c – MES416
- 19c – MES4161

NOTE: no more one-off or overlay patches needed for MES (Doc ID 2274242.1)

11g Standard Edition: TCPS adapter needs to be enabled in oracle binaries before MES bundle patch is applied using Note 1457854.1

Case 1: SSL for database connections

1. Create a wallet – using signed CA in this example
    - Create a certificate request
    - Submit the certificate request to the CA
    - Import the signed server certificate
2. Configure server sqlnet.ora / listener.ora
    - Add wallet directory to both
    - Optionally, restrict cipher algorithms and protocol versions, client authentication
    - Add TCPS listener endpoint
3. Add new TCPS listener to LOCAL_LISTENER parameter
4. Restart listener
5. Distribute wallet to clients
6. Configure wallet directory in client sqlnet.ora / tnsnames.ora

Step by Step Guide To Configure SSL Authentication (Doc ID 736510.1)

Case 1: 1 - Create a wallet

```
orapki wallet create -wallet . -auto_login -pwd $MYPASS
chmod 775 ewallet.p12 cwallet.sso
orapki wallet display -wallet .
orapki wallet remove -trusted_cert_all -wallet . -pwd $MYPASS
```

NOTES:

- Leave only the required SSL certificates
- use a separate wallet for TDE and SSL.
  - Enforced since October 2018 PSU

Case 1: 1 - Create a wallet

```
orapki wallet create -wallet . -auto_login -pwd $MYPASS
chmod 775 ewallet.p12 cwallet.sso
orapki wallet display -wallet .
orapki wallet remove -trusted_cert_all -wallet . -pwd $MYPASS
```

```
[oracle@myserver wallet2018]$ ls -lrt
total 0
[oracle@myserver wallet2018]$ orapki wallet create -wallet . -auto_login -pwd xxx
Oracle PKI Tool : Version 11.2.0.4.0 - Production
Copyright (c) 2004, 2013, Oracle and/or its affiliates. All rights reserved.

[oracle@myserver wallet2018]$ orapki wallet display -wallet .
Oracle PKI Tool : Version 11.2.0.4.0 - Production
Copyright (c) 2004, 2013, Oracle and/or its affiliates. All rights reserved.

Requested Certificates:
User Certificates:
Trusted Certificates:
Subject:        OU=Class 1 Public Primary Certification Authority,O=VeriSign\, Inc.,C=US
Subject:        OU=Class 3 Public Primary Certification Authority,O=VeriSign\, Inc.,C=US
Subject:        OU=Class 2 Public Primary Certification Authority,O=VeriSign\, Inc.,C=US
Subject:        OU=Secure Server Certification Authority,O=RSA Data Security\, Inc.,C=US
Subject:        CN=GTE CyberTrust Global Root,OU=GTE CyberTrust Solutions\, Inc.,O=GTE Corporation,C=US
```

Case 1: 1 - Create a wallet

Note: self signed certificates can be used instead of step a), skipping b)

a. Create a certificate request

```
orapki wallet add -wallet . -dn "CN=myserver.domain,O=Acme,L=New York,ST=New York,C=US" \
       -keysize 2048 -pwd $MYPASS

orapki wallet export -wallet . -dn "CN=myserver.domain,O=Acme,L=New York,ST=New York,C=US" \
       -request ./myserver.req -pwd $MYPASS
```

Known issue: orapki uses MD5 algorithm for requests (cannot be changed).

Above won't work if we want SHA2 signed request (validated by CAs nowadays).

Use openssl instead:

```
openssl pkcs12 -in ewallet.p12 -nodes -out mywallet.pem
openssl req -new -key mywallet.pem -sha256 -out mywallet-sha2.csr
```

Case 1: 1 - Create a wallet

b. Submit the certificate request to the CA

if we don't use external CA (for testing), a Self-Signed Certificate can be created.

```
orapki cert create -wallet $ORACLE_BASE/wallet_CA -request ./myserver.req \
       -cert ./usercert.txt -sign_alg sha256 -validity 3650
```

c. Import the signed server certificate

We can see the cipher algorithm used to sign the certificate received:

```
$ openssl x509 -in mysignedcert.cer -text
Certificate:
Data:
    Version: 3 (0x2)
    Serial Number:
        96:c2:92:71:11:1e:70:c1:5e:5e:62:a3:fe:44:f9:c3
Signature Algorithm: sha256WithRSAEncryption
    Issuer: C=US, ST=MI, L=Ann Arbor, O=Internet2, OU=InCommon, CN=InCommon RSA Server
CA
```

Case 1: 1 - Create a wallet (cont.)

Break up Intermediates/root certificate into the constituent components, based on *-BEGIN CERTIFICATE- / -END CERTIFICATE-* tags, creating one file per each certificate

Then, import them into the wallet:

```
orapki wallet add -wallet . -trusted_cert -cert mycert1.cer -pwd xxx
orapki wallet add -wallet . -trusted_cert -cert mycert2.cer -pwd xxx
orapki wallet add -wallet . -trusted_cert -cert mycert3.cer -pwd xxx
```

We can validate the wallet contains now our certificates:

```
orapki wallet display -wallet .
```

NOTE: if imported into a different server than used to generate the request:

PKI-04006: No matching private key in the wallet

Case 1: 2 - Configure server

- Configure server sqlnet.ora / listener.ora
  - Add wallet directory to both
  - Optionally, restrict cipher algorithms and protocol versions, client authentication
  - Add TCPS listener endpoint

```
WALLET_LOCATION =
 (SOURCE =
   (METHOD = FILE)
   (METHOD_DATA =
     (DIRECTORY = /u01/app/oracle/admin/orcl/wallet)
   )
)
```

NOTE for RAC:

- SSL configuration needs to be on SCAN listeners and local listeners
- Wallet should be available in all nodes

Case 1: 2 – Add TCPS listener endpoints

```
[oracle@myserver ~]$ crsctl stat res -p |grep ENDPOINTS
ENDPOINTS=TCP:1521
ENDPOINTS=TCP:1521
ENDPOINTS=TCP:1521
ENDPOINTS=TCP:1521
ENDPOINTS=TCP:1521

[oracle@myserver ~]$ srvctl modify listener -p "TCP:1521/TCPS:2484"
[oracle@myserver ~]$ srvctl modify scan_listener -p "TCP:1521/TCPS:2484"

[oracle@myserver ~]$ crsctl stat res -p |grep -B20 ENDPOINTS | grep -e ENDPOINTS -e "^NAME="
NAME=ora.LISTENER.lsnr
ENDPOINTS=TCP:1521 TCPS:2484
NAME=ora.LISTENER_SCAN1.lsnr
ENDPOINTS=TCP:1521 TCPS:2484
NAME=ora.LISTENER_SCAN2.lsnr
ENDPOINTS=TCP:1521 TCPS:2484
NAME=ora.LISTENER_SCAN3.lsnr
ENDPOINTS=TCP:1521 TCPS:2484
NAME=ora.MGMTLSNR
ENDPOINTS=TCP:1521
```

Case 1: 3 – Adjust LOCAL_LISTENER

- New TCPS listener should be added to the list of listeners in the LOCAL_LISTENER database parameter
  - Static registration can be used for single instances instead

```
ALTER SYSTEM SET
LOCAL_LISTENER='(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=TCP)...)
(ADDRESS=(PROTOCOL=TCPS)...)))' scope=both;
```

Case 1: Final steps and tests

- 4 - Restart listeners

```
$ srvctl stop scan_listener ; srvctl start scan_listener ; srvctl status scan_listener
$ srvctl stop listener ; srvctl start listener ; srvctl status listener
```

  Do it by node to minimize downtime (add *-n node* parameter)

  Optionally restart database to confirm all changes done to configuration files are good

- 5 - Distribute wallet to clients
  - Clients could use their own certificates to add extra security. Server and Client certificates should be included in each wallet for that (export/import of each certificate)
- 6 - Configure wallet directory in client sqlnet.ora / tnsnames.ora

  tnsnames.ora entry overwrites sqlnet.ora configuration:

```
orclSSL =
   (DESCRIPTION =
      (ADDRESS = (PROTOCOL = TCPS)(HOST = orcl-scan)(PORT = 2484))
      (CONNECT_DATA = (SERVER = DEDICATED)(SERVICE_NAME = orcl) )
      (SECURITY=(MY_WALLET_DIRECTORY=C:\OracleClient\wallet)) )
```

Case 1: Final steps and tests

- 7 - Test connectivity

```
$ sqlplus test/test@ORCLCSSL
...
SQL>  select sys_context('userenv','network_protocol') from dual;

SYS_CONTEXT('USERENV','NETWORK_PROTOCOL')
--------------------------------------------------------------------------------
tcps
```

Case 1: Extra considerations

Wallets for PDBs

- Each PDB use its own wallet with its own certificates for TLS authentication
- Shared sqlnet.ora, place PDB wallet in a subdirectory of the wallet directory where the name of the subdirectory is the GUID of the PDB that uses the wallet
- DBA_PDBS data dictionary view has existing PDBs and their GUIDs
  - example: $ORACLE_HOME/admin/db_unique_name/wallet/PDB_GUID

https://docs.oracle.com/en/database/oracle/oracle-database/18/dbseg/configuring-secure-sockets-layer-authentication.html

Configuration using JDBC

https://blogs.oracle.com/dev2dev/ssl-connection-to-oracle-db-using-jdbc,-tlsv12,-jks-or-oracle-wallets

Case 2: utl_smtp PL/SQL

- Requirements:
  - SSL for utl_smtp available since Oracle 11.2.0.2
  - Credentials to authenticate with mail server
- Steps:
  - Get the SSL certificates used by mail server
  - Import those SSL certificates into an Oracle Wallet
  - Create ACL to allow traffic for the user/port (MOS note 1209644.1)
  - Call utl_smtp using sample code from MOS note 1323140.1
- Example using Amazon Simple Email Service (SES):
  https://blog.pythian.com/oracle-and-amazon-simple-email-service/
- Issues when mail server certificates are changed (it will!)

Case 2: utl_smtp PL/SQL

Get the SSL certificates used by email server

```
$ openssl s_client -showcerts -connect smtp.gmail.com:993
```

```
CONNECTED(00000003)
depth=2 OU = GlobalSign Root CA - R2, O = GlobalSign, CN = GlobalSign
verify return:1
depth=1 C = US, O = Google Trust Services, CN = Google Internet Authority G3
verify return:1
depth=0 C = US, ST = California, L = Mountain View, O = Google Inc, CN = imap.gmail.com
verify return:1
---
Certificate chain
 0 s:/C=US/ST=California/L=Mountain View/O=Google Inc/CN=imap.gmail.com
   i:/C=US/O=Google Trust Services/CN=Google Internet Authority G3
-----BEGIN CERTIFICATE-----
...
-----END CERTIFICATE-----
 1 s:/C=US/O=Google Trust Services/CN=Google Internet Authority G3
   i:/OU=GlobalSign Root CA - R2/O=GlobalSign/CN=GlobalSign
-----BEGIN CERTIFICATE-----
```

Case 3: Oracle Enterprise Manager

It has

- TLS protocol used by its components (OMS, Agent, WLS, OPMN, OHS)
- Ciphers used by SSL certificates on each of those service's ports
- Key strength of those certificates (512, 1024 2048, etc.)

Many changes in each version:

- 12c: use MOS note ***1602983.1*** to enable usage of TLSv1.0 Protocol
- 13.1: EM Oracle Management Service is configured to use TLS v1.0, v1.1 and v1.2 protocols out-of-the-box
  - It can be changed to restrict OMS to use specific protocols like TLS v1.1 or TLS v1.2. Details in MOS note ***2212006.1***
- 13.2: TLS v1.2 is default
- 13.3: OMS has java version 1.7.0_171 out-of-the-box

Case 3: OEM fun – possible changes

OEM 12c: uses 10.3.6 WLS. Demo cert. has 512 bit keystrength signed with MD5withRSA

OEM 13c: uses 12.1.3 WLS. Demo cert. has 2048 bit keystrength signed with SHA256withRSA

We can make changes following steps from below notes:

- How to Check and Increase the Key Strength of Certificates Used in Enterprise Manager Grid Control (Doc ID 1476567.1)
- EM 12c: How to Disable Weak SSLCipherSuites Used by Enterprise Manager 12c Cloud Control (Doc ID 1477287.1)
- New SSL Protocol and Cipher Options for Oracle Fusion Middleware's OPMN/ONS Component (Doc ID 1905314.1)
- Regenerating OEM 12c SSL certificates with Higher Keystrength and Signature Algorithm (Doc ID 1611578.1)
- WebLogic Server - Migrating a 1024-bit key certficate (CSR) to a 2048-bit key (Doc ID 949316.1)
- EM 13c, 12c: How to Configure Enterprise Manager's Weblogic Server (WLS) for Secure Socket Layer Certificates (Doc ID 2220788.1)

Today's topics

- Security basis
- Use cases in the Oracle Database
  1. SSL for database connections
  2. Email from PL/SQL
  3. Oracle Enterprise Manager
- **Management**
- Troubleshooting

Management

- Renew certificates before they expire (periodically)
  - ➢ Regenerate certs, wallets, distribute to clients
- Security policy changes asking to:
  1) use stronger cipher suite (e.g.: SHA2)
     - ➢ Regenerate certs, wallets, distribute to clients, config changes (sqlnet.ora), patch binaries client/server

  2) restrict protocol versions to use stronger ones
     - ➢ config changes (sqlnet.ora), patch binaries client/server
- Security patches (PSU/RU/one-offs)
  - ➢ Patches can include protocol/signature changes, forcing to regenerate certificates to keep up with newer security standards. Ex: Oct 2018 PSU
    1. MD5 signed certificates no longer supported. They need to be replaced
    2. SSL version 3.0 is no longer enabled by default
    3. Cipher suite SSL_RSA_WITH_DES_CBC_SHA is no longer available
    4. Auto-login wallets need to be re-generated if FIPS mode is enabled

Example: Enforcing particular protocols/ciphers

Why?

- To comply with security policies
- If using older Database versions that have weak protocols enabled

Example to enforce TLS 1.2 (disabling 1.0/1.1)

1) sqlnet.ora (GI and DB home)

```
SSL_VERSION=1.2
SSL_CIPHER_SUITES =
((SSL_RSA_WITH_AES_256_CBC_SHA256,SSL_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256,SSL_ECDHE_E
CDSA_WITH_AES_128_CBC_SHA256,SSL_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384,SSL_ECDHE_ECDSA_
WITH_AES_256_GCM_SHA384,SSL_RSA_WITH_AES_128_CBC_SHA256,SSL_RSA_WITH_AES_128_GCM_SHA2
56,SSL_RSA_WITH_AES_256_GCM_SHA384) )
```

Enforcing particular protocols/ciphers (cont.)

2) Restart listeners (SCAN too if using RAC)

```
$ lsnrctl stop listener ; lsnrctl start listener ; lsnrctl status
...
Listening Endpoints Summary...
   (DESCRIPTION=(ADDRESS=(PROTOCOL= tcps)(HOST=172.15.232.29)(PORT=2484)))
Services Summary...
```

3) Test connectivity

Easy way is to configure a client enforcing one of the non-accepted protocols or ciphers, or using a certificate with a non accepted signing algorithm.

```
$ sqlplus test/test@ORCLCSSL
...
SQL>  select sys_context('userenv','network_protocol') from dual;

SYS_CONTEXT('USERENV','NETWORK_PROTOCOL')
--------------------------------------------------------------------------------
tcps
```

Confirm protocols accepted

So far we have enabled SSL in the listener to use all available TLS protocols and signing algorithms for our installed version. To check that:

```
[oracle@myhost ~]$ openssl s_client -host server-vip -port 2484
CONNECTED(00000003)
depth=3 C = SE, O = AddTrust AB, OU = AddTrust External TTP Network, CN = AddTrust External CA Root
…
SSL handshake has read 5942 bytes and written 573 bytes
---
New, TLSv1/SSLv3, Cipher is AES256-SHA
Server public key is 2048 bit
Secure Renegotiation IS supported
Compression: NONE
Expansion: NONE
SSL-Session:
    Protocol  : TLSv1
    Cipher    : AES256-SHA
    Session-ID:
…
---
New, TLSv1/SSLv3, Cipher is RC4-SHA
Server public key is 1024 bit
…
SSL-Session:
    Protocol  : TLSv1
    Cipher    : RC4-SHA
```

Confirm protocols accepted

We can inspect the traffic:

- Enabling SQL*Net trace – no details about protocol used in handshake
- Use tcpdump and wireshark to confirm protocols and ciphers

```
[root@myserver ~]# tcpdump -nnvvXSs0 -i eth1 host myhost.mydomain -w /tmp/tcpdump.out
tcpdump: listening on eth1, link-type EN10MB (Ethernet), capture size 262144 bytes
```

Perform activity from client and stop capture:

```
SQL> select * from dual;

[root@myserver ~]# ^C 926 packets captured
926 packets received by filter
0 packets dropped by kernel
```

Open the generated file using wireshark.

- Mark the port used for SSL (Analyze -> Decode As -> Transport)

Wireshark

668	131.198345	172.22.239.11	172.22.239.22	TLSv1.2	488 Certificate Request, Server Hello Done
662	131.198099	172.22.239.11	172.22.239.22	TLSv1.2	3567 Certificate
661	131.197973	172.22.239.11	172.22.239.22	TLSv1.2	2114 Server Hello
660	131.196986	172.22.239.11	172.22.239.22	TCP	66 2484 → 23250 [ACK] Seq=1 Ack=85 Win=29056 Len=0 TSval=2253885788 TSecr=1146225186
657	131.159598	172.22.239.11	172.22.239.22	TCP	74 2484 → 23250 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=2253885751 TSecr=1146225148 WS=128
644	126.716877	172.22.239.11	172.22.239.22	TCP	66 2484 → 23241 [ACK] Seq=19170 Ack=18215 Win=98560 Len=0 TSval=2253881308 TSecr=1146220706
642	126.716823	172.22.239.11	172.22.239.22	TCP	66 2484 → 23241 [FIN, ACK] Seq=19169 Ack=18214 Win=98560 Len=0 TSval=2253881308 TSecr=1146220705
639	126.715898	172.22.239.11	172.22.239.22	TLSv1.2	151 Application Data
632	124.564876	172.22.239.11	172.22.239.22	TLSv1.2	295 Application Data
631	124.564642	172.22.239.11	172.22.239.22	TCP	66 2484 → 23241 [ACK] Seq=18855 Ack=18076 Win=98560 Len=0 TSval=2253879156 TSecr=1146218553
628	124.563733	172.22.239.11	172.22.239.22	TLSv1.2	503 Application Data
627	124.543199	172.22.239.11	172.22.239.22	TCP	66 2484 → 23241 [ACK] Seq=18418 Ack=17991 Win=98560 Len=0 TSval=2253879135 TSecr=1146218493
608	116.237396	172.22.239.11	172.22.239.22	TLSv1.2	151 Application Data
606	116.236756	172.22.239.11	172.22.239.22	TLSv1.2	151 Application Data
604	116.235944	172.22.239.11	172.22.239.22	TLSv1.2	535 Application Data
602	116.234898	172.22.239.11	172.22.239.22	TLSv1.2	359 Application Data
600	116.233303	172.22.239.11	172.22.239.22	TLSv1.2	471 Application Data
598	116.232034	172.22.239.11	172.22.239.22	TLSv1.2	759 Application Data
596	116.230747	172.22.239.11	172.22.239.22	TLSv1.2	311 Application Data
594	116.229020	172.22.239.11	172.22.239.22	TLSv1.2	295 Application Data
592	116.228096	172.22.239.11	172.22.239.22	TLSv1.2	519 Application Data
590	116.222930	172.22.239.11	172.22.239.22	TLSv1.2	343 Application Data
587	116.221167	172.22.239.11	172.22.239.22	TLSv1.2	1975 Application Data
585	116.191532	172.22.239.11	172.22.239.22	TLSv1.2	519 Application Data
583	116.186177	172.22.239.11	172.22.239.22	TLSv1.2	151 Application Data
581	116.183696	172.22.239.11	172.22.239.22	TLSv1.2	311 Application Data
579	116.179197	172.22.239.11	172.22.239.22	TLSv1.2	263 Application Data
577	116.178592	172.22.239.11	172.22.239.22	TLSv1.2	167 Application Data
575	116.178056	172.22.239.11	172.22.239.22	TLSv1.2	157 Change Cipher Spec, Encrypted Handshake Message
574	116.177860	172.22.239.11	172.22.239.22	TCP	66 2484 → 23241 [ACK] Seq=12103 Ack=12658 Win=63744 Len=0 TSval=2253870769 TSecr=1146210160
569	116.170980	172.22.239.11	172.22.239.22	TCP	66 2484 → 23241 [ACK] Seq=12103 Ack=12031 Win=57984 Len=0 TSval=2253870762 TSecr=1146210160

```
Frame 668: 488 bytes on wire (3904 bits), 488 bytes captured (3904 bits)
Ethernet II, Src: Vmware_af:f0:1d (00:50:56:af:f0:1d), Dst: Vmware_af:1d:37 (00:50:56:af:1d:37)
Internet Protocol Version 4, Src: 172.22.239.11, Dst: 172.22.239.22
Transmission Control Protocol, Src Port: 2484, Dst Port: 23250, Seq: 5550, Ack: 85, Len: 422
Secure Sockets Layer
```

Troubleshooting - Steps

1) Use tnsping to test the connection to the database TNS entry over TCPS
2) Verify the listener.ora and sqlnet.ora files on the database server
3) Verify the permissions of the wallet files
4) Check the SSL_VERSION is supported by Oracle binaries
5) Enable sqlnet tracing for the listener and sqlplus connections
6) Capture traffic with tcpdump and check handshake
7) Is using utl_http service, get a trace for the event 10937
8) Verify patches installed (look for latest MES in PSU since July 2018)

SSL Troubleshooting Guide (Doc ID 166492.1)

- ORA-12560: TNS:protocol adapter error
- ORA-28862: SSL connection failed
  - 28759, 00000, "Failed to open file"
  - 28859, 00000, "SSL negotiation failure"
  - ntzCreateConnection: failed with error 549
- ORA-29024:Certificate Validation Failure
- ORA-29143: Message 29143 not found
- ORA-29106: Can not import PKCS # 12 wallet
- ORA-28860: Fatal SSL error
- ORA-29263: HTTP protocol error
- ORA-28868: certificate chain check failed
- ORA-28750: unknown error
- ORA-28865: SSL connection closed
- ORA-01004: Default username feature not supported; log denied
- ORA-28864: SSL connection closed gracefully
- ORA-01017: invalid username/password; logon denied
- alert.log: "SSL Client: Server DN doesn't contain expected SID name"
- ORA-29113: Cannot access attributes from a PKCS #12 key bag.
- ORA-29002: SSL transport detected invalid or obsolete server certificate
- ORA-29003: SSL transport detected mismatched server certificate
- ORA-28857: Unknown SSL Error

Common errors found

- PKI-04006: No matching private key in the wallet
  - Caused when imported certs into a server that did not generated the request
  - Repeat the same procedure in the server that generated the request worked without errors
- ORA-29024: Certificate validation failure
  - Expired certificate
  - Wallet not in the path (sqlnet.ora, listener.ora), privileges
- ORA-28864: ssl connection closed gracefully
  - Client installation, firewall
- ORA-28865: SSL connection closed
  - No matching protocol or cipher suite on server/client configuration

THANK YOU

Questions?

calero@pythian.com

@ncalerouy

http://www.linkedin.com/in/ncalero

Pythian
love your data®

Oracle Cloud Infrastructure

New Free Tier

oracle.com/gbtour

Always Free

Services you can use for unlimited time

+

30-Day Free Trial

Free credits you can use for more services

References - documentation

- Oracle Database Security Guide 18c – Configuring SSL authentication

  https://docs.oracle.com/en/database/oracle/oracle-database/18/dbseg/configuring-secure-sockets-layer-authentication.html#GUID-6AD89576-526F-4D6B-A539-ADF4B840819F

- Step by Step Guide To Configure SSL Authentication (Doc ID 736510.1)
- TLS 1.2 in Oracle Database and MES415 (Doc ID 2274242.1)
- How To Investigate And Troubleshoot SSL/TLS Issues on the Database And Client SQL*Net Layer (Doc ID 2238096.1)
- Important Usage Notes For Database Using MES415 Crypto Libraries. (Doc ID 2458023.1)
- Unable To Open TDE Encryption Wallet After Applying MES Bundle Patch or After importing SSL certificates into TDE wallet (Doc ID 2192475.1)