Get Your Database Vaccinated: Oracle Transparent Data Encryption with Almost Zero Downtime!

AGENDA

- What is TDE
- Types of TDE
- Methods to implement TDE
- TDE Encryption Algorithms
- Oracle Wallet and Keys
- TDE Table Space Encryption- Fast-Offline Method
- Performance Checklist
- Post TDE Buddy Checks
- Project and Release Management
- Post TDE Issues & Backout Plan
- Wallet Management

What is TDE

Oracle Transparent Data Encryption (TDE) enables the organizations to encrypt sensitive application data on storage media completely transparent to the application.

- **Sensitive data is safe in case the storage media or data file gets stolen.**
- **Implementing TDE helps you address security-related regulatory compliance issues.**
- **Encryption and decryption of data are transparent to the users.**
- **Key management operations are automated. The user or application does not need to manage encryption keys.**

Types of TDE

TDE Column Encryption

- TDE column encryption uses the two-tiered key-based architecture to transparently encrypt and decrypt sensitive table columns.
- This TDE master encryption key encrypts and decrypts the TDE table key, which in turn encrypts and decrypts data in the table column.

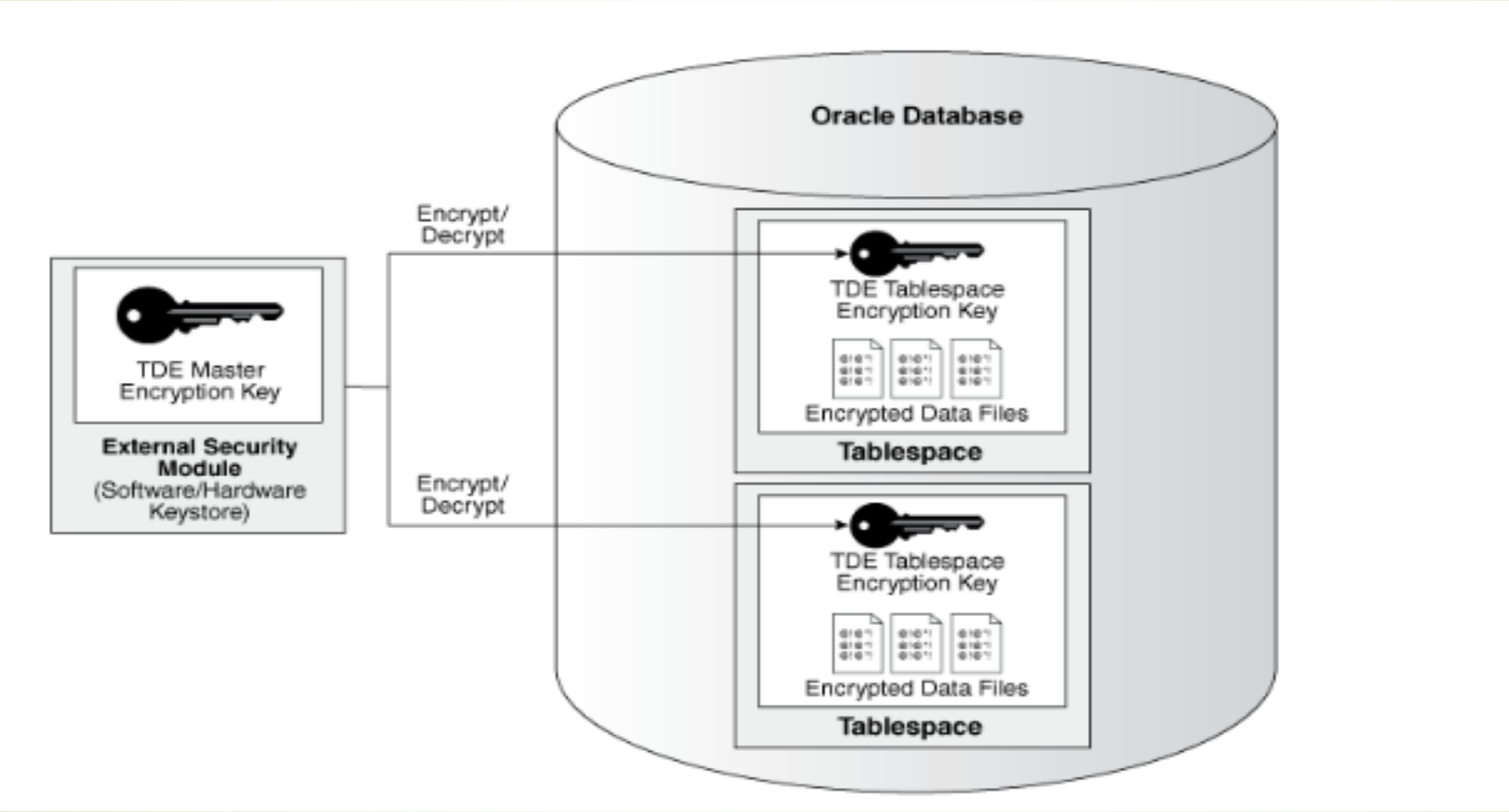
TDE Tablespace Encryption

- TDE tablespace encryption uses the two-tiered, key based architecture to transparently encrypt (and decrypt) tablespaces.
- This TDE master encryption key is used to encrypt the TDE tablespace encryption key, which in turn is used to encrypt and decrypt data in the tablespace.

Note: The table and tablespace keys are encrypted using the master key. The master key is stored in an external security module (ESM) that can be one of the following:

- An Oracle Wallet – A secure container outside of the database. It is encrypted with a password.
- Oracle Key Vault

TDE Tablespace Encryption



Methods to Implement TDE

Option 1: Export / Import (Expdp/Impdp)

Unacceptable application outage times when using Data PUMP for the Export/Import

Option 2: Fast Offline Conversion (with/without DataGuard)

First encrypt PRODUCTION Standby and then switchover to PRODUCTION Standby(already encrypted) during GO-LIVE followed by encrypting new PRODUCTION Standby(Old Production)

Option 3: Move objects using packages (like DBMS_redefinition)

It requires active participation from the Product Development Logical Database team as execution involves moving database objects.

TDE Encryption Algorithms

- Although longer key lengths theoretically provide greater security, there may be trade-off in CPU overhead

  AES256: Enables you to use AES to encrypt a block size of 256 bits.

  AES192: Enables you to use AES to encrypt a block size of 192 bits.

  AES128: Enables you to use AES to encrypt a block size of 128 bits.

- Oracle uses the following default algorithms:

  For tablespace encryption, AES128 is the default.

  For column encryption, AES 192 is default.

Oracle Wallets and Keys

High Level Steps

- Set the encryption wallet location.
- Set the TDE Master key.
- Create auto-login wallet.
- Creating Encrypted Tablespace(s) with recommended algorithm 'AES128' and test the master key configuration.
- Copy the wallet files to all nodes in the configuration (Oracle RAC primary nodes and all standby nodes).

Oracle Wallets and Keys

In this implementation, we are looking at securing the data using Tablespace Encryption using Oracle Wallet.

Set the encryption wallet location

ENCRYPTION_WALLET_LOCATION = (SOURCE = (METHOD = FILE)(METHOD_DATA = (DIRECTORY = /etc/oracle/WALLETS/$ORACLE_SID)))

- The sqlnet.ora should be placed in $TNS_ADMIN location and should point to the location where the wallet files are to be managed.
- Make sure that the wallet location exists. If a non-default wallet location must be used then specify it in the sqlnet.ora file
- The wallet location should be a secure location with read and write permission to oracle rdbms os id. As per best practices, it is not recommended to have it under $ORACLE_HOME or $ORACLE_BASE to prevent accidental backup of this secure file to media.
- Create the corresponding directory on all nodes with the proper ORACLE_SID.

Initiate a new SQL*Plus session. This causes the changes to sqlnet.ora to be picked up by the new session.

Oracle Wallets and Keys

Set the TDE Master key

SQL> ALTER SYSTEM SET ENCRYPTION KEY IDENTIFIED BY "*****";. – 11.2.0.4**

SQL> ALTER SYSTEM SET ENCRYPTION WALLET OPEN identified by "******"; ------- 11.2.0.4**

SQL> ADMINISTER KEY MANAGEMENT CREATE KEYSTORE '/etc/oracle/WALLETS/$ORACLE_SID' IDENTIFIED BY "****!"; ----- 12.1.0.2**

SQL>ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY "*****"; --- 12.1.0.2**

SQL>ADMINISTER KEY MANAGEMENT SET KEY IDENTIFIED BY "**" WITH BACKUP USING 'TDE'; --- 12.1.0.2**

- Ensure that the password string is contained in double quotation marks (" ").
- Reduce permissions on the wallet file from the initial value, determined by 'umask' for the 'Oracle' user, to:

  $ cd /etc/ORACLE/WALLETS/<$ORACLE_SID>

  $ chmod 600 ewallet.p12

Oracle Wallets and Keys

Create auto-login wallet

$ orapki wallet create -wallet /etc/oracle/WALLETS/$ORACLE_SID -auto_login ---------- 11.2.0.4

SQL>ADMINISTER KEY MANAGEMENT CREATE AUTO_LOGIN KEYSTORE FROM KEYSTORE '/etc/oracle/WALLETS/$ORACLE_SID' IDENTIFIED BY "AbCdEfGh!"; ---------- 12.1.0.2

- To protect the Oracle TDE Wallets from being inadvertently deleted, make them 'immutable'.

chattr +i ewallet.p12

chattr +i cwallet.sso

Oracle Wallets and Keys

Create an encrypted tablespace to test table creation.

```
SQL> CREATE TABLESPACE encr_tbs DATAFILE '/u01/app/oracle/oradata/orcl/encr_tbs1.dbf' SIZE 20M AUTOEXTEND ON maxsize 500M ENCRYPTION USING
'AES128’ DEFAULT STORAGE(ENCRYPT);

SQL> SELECT tablespace_name, encrypted FROM dba_tablespaces where tablespace_name='ENCR_TBS';

TABLESPACE_NAME                ENC
------------------------------ ---
ENCR_TBS                       YES

SQL> CREATE TABLE test123 tablespace encr_tbs as select * from v$session;

SQL> select tablespace_name from dba_segments where segment_name='TEST123’;

TABLESPACE_NAME
------------------------------
ENCR_TBS

SQL> select count(*) from test123;

  COUNT(*)
----------
        22
```

Oracle Wallet and Keys

- **Copy the files generated in the keystore directory to all nodes of the primary and standby.**

Copy files to each node:

$ scp /etc/oracle/WALLETS/$ORACLE_SID /* oracle@<host>:/etc/oracle/WALLETS/$ORACLE_SID

- **Ensure the wallet is open on all nodes.**

SQL> select * from gv$encryption_wallet;

INST_ID WRL_TYPE WRL_PARAMETER STATUS

------- -------- ------------------------------------ ------

1 file /etc/oracle/WALLETS/$ORACLE_SID OPEN

TDE Tablespace Encryption – Fast-Offline Method

Pre-requisites

- There is an existing physical standby database for no downtime method.
- A current backup has been taken of the database prior to converting data files.
- The patch 23315889 described in MOS 2148746.1 has been applied to both primary and standby RDBMS installation.

  -Incase there is any one-off patches already applied to Oracle Home, request Oracle for a merge patch
- COMPATIBLE is set to 12.1.0.2 or 11.2.0.4 respectively.
- Require the primary database to have forced logging enabled.
- To ensure there are no unrecoverable blocks the following query at the primary database should return no rows.

See MOS 290161.1 for additional details on nologging operations and handling.

SQL> select NAME from V$DATAFILE where UNRECOVERABLE_CHANGE#>0;

Note: If UNRECOVERABLE_CHANGE# is >0 for any datafile, compare the value on the primary to the value on the standby.

If they are the same the datafile was copied after the unrecoverable change and no action is necessary.

TDE Tablespace Encryption – Fast-Offline Method

Implementation High Level Steps

- Verify that the Data Guard configuration is healthy and contains no gaps.
- Place the standby in a mounted state with recovery stopped.
- On the standby: Encrypt data files in-place and in parallel.

```
SQL> select 'alter database datafile '||chr(39)||df.name||chr(39)||' encrypt;' COMMAND from v$tablespace ts, v$datafile df where ts.ts#=df.ts# and (ts.name not in ('SYSTEM','SYSAUX') and ts.name not in (select value from gv$parameter where name='undo_tablespace'));
```

- On the standby: Restart redo apply and catch up.
- Execute a Data Guard switchover making the encrypted standby the new primary and the unencrypted primary the new standby.
- On the NEW standby: Place the new standby database in a mounted state with recovery stopped.
- On the NEW standby: Encrypt data files in-place and in parallel.
- On the NEW standby: Restart redo apply and catch up.
- Optionally execute a Data Guard switchover to reestablish the original configuration.

Fast-Offline Method – Restrictions

- It only supports AES128 for tablespace keys.
- Master keys are always AES256.
- It requires a certain PSU or a one-off patch or merge patch would need to be requested by Oracle.
- This may require downtime for patching
- Only 11.2.0.4 + is supported
- Encryption requires downtime unless dataguard is used, in that case only a quick downtime for switchover is required.

Performance Checklist

Perform these steps before TDE and after TDE

- Run DBverify and RMAN Validate for all datafiles and ensure no corruptions
- Capture timestamp at server and Database Level
- Compile complete list of invalids
- Create a manual AWR snapshot and capture the time taken.
- Capture AWR and ADDM Reports
- Capture the 'last' RMAN level0 backup timing to disk and tape to compare with post encryption timings
- Validate if AWR retention has been set "high": 30 days? 60-days? 90 days?
- Capture Database Size
- Capture CPU core count
- Database and OS files backups

Post TDE Buddy Checks

- Validate both the value of v$encryption_wallet and sqlnet.ora is having the same entry and the wallet status is OPEN.
- Check the Tablespace status for all tablespaces . It should show encrypted=Yes (except system,sysaux,undo and Temp) .
- The encryption algorithm used should be AES128. Validate: -

```
SELECT t.name , e.encryptionalg , e.encryptedts FROM v$tablespace t , v$encrypted_tablespaces e WHERE t.ts# = e.ts#;
```

- DB Verify logs should not report any error after datafile encryption and it should show used blocks are encrypted.
- Validate and Confirm that the encryption key files present in the wallet locations are immutable.

  The permission of Wallet Directory should be 700 and permission of files should be 600 owned by oracle:oinstall.

  # chattr +i ewallet.p12

  # chattr +i cwallet.sso

- Validate backup of wallet has been scheduled under crontabs.

Project and Release Management

- Detailed Work Breakdown Structure for each of “17” databases
- Separate Change Requests
  - Pre TDE backups
  - Dbverify/RMAN validate
  - TDE encryption
  - Post TDE Backups
- Performance test results document (Comparing Pre-TDE and Post-TDE results)
  - Changes in DB Size
  - Changes in CPU Consumption
  - Changes in Memory Consumption
  - Changes in IO bottleneck
- Executing around the clock using 24/7 DBA/Infrastructure Support Teams

Post TDE Tablespace Encryption Issues

- New datafiles were getting created as non-encrypted
  - In 11.2.0.4 and 12.1.0.2, new datafiles created under encrypted tablespace is always created as encrypted.
  - To create new tablespace, please create with encrypted clause
  - In 12.2, consider setting database initialization parameter “encrypt_new_tablespaces” =ALWAYS

```
SQL> ALTER SYSTEM SET encrypt_new_tablespaces='ALWAYS' SCOPE=BOTH SID='*';
```

- Master Key was showing “AES 128”
  - We are hitting a bug here :- Bug 19636912 : TDE MASTER KEY ENCRYPTION SHOULD USE STRONGEST AVAILABLE ALGORITHM
  - The TDE masterkey is, in fact, always AES256 and cannot currently be changed.
  - The query which we are using (select * from v$database_key_info) relates to tablespace key as per oracle and is due to the BUG .
  - The Master Key : What happens when the master key is used, accessed and reset in TDE ? (Doc ID 1342875.1)
- Known TDE Issue

Bug 25375360 - ACFS File System Encryption Causes TDE Block Corruption and Memory Errors (Doc ID 25375360.8)

Backout Plan

- Command to decrypt the database that was encrypted using fast-offline method

  ALTER DATABASE DATAFILE '<file name>' DECRYPT;

  - Database should be in mount state

- Please do not delete wallet or keys
  - TDE wallet becomes a vital part of the database. It should never be deleted even though there are no encrypted objects in the database.
  - When TDE is implemented master key is also noted in control file, system tablespace and hence cannot be removed. Removing the wallet will cause damage to the system.

Wallet Management

- Strong password should be used.
- Place Oracle Wallet outside of the $ORACLE_BASE directory tree: for example: /etc/ORACLE/WALLETS/<$ORACLE_UNQNAME>
- Change ownership to ‘oracle:oinstall’ and set the permissions to ‘oracle’ only:

```
# cd /etc
# mkdir –pv ORACLE/WALLETS/NASCXTHZNPRD
# chown –R oracle:oinstall ORACLE
# chmod –R 700 ORACLE
```

- The permission of Directory should be 700 and permission of files should be 600 owned by oracle:oinstall

```
$ cd /etc/ORACLE/WALLETS/<$ORACLE_UNQNAME>
$ chmod 600 ewallet.p12
```

- In order to protect the Oracle TDE Wallets from being inadvertently deleted, make them ‘immutable’.
- Always backup the wallet at the same time when backing up your database. However, do not include the wallet on the same media as the database backup.
- Separate wallet for separate databases in same server.

TDE Tablespace Encryption Restrictions

There are few restrictions with TDE tablespace encryption because encrypt/decrypt takes place during read/write as opposed to the SQL layer with column encryption. TDE tablespace encryption restrictions are:

- External Large Objects (BFILEs) cannot be encrypted using TDE tablespace encryption because these files reside outside the database.
- To perform import and export operations on TDE encrypted tablespaces, you must use Oracle Data Pump.
- The offline encryption process will use the AES128 encryption algorithm with the key identifier listed in V$DATABASE_KEY_INFO.
- This process is only for application tablespace data files. SYSTEM, SYSAUX, UNDO, and TEMP tablespaces cannot be encrypted with TDE.

TDE Online Encryption – 18c and above

- In previous releases, the SQLNET.ENCRYPTION_WALLET_LOCATION parameter was used to define the keystore directory location. This parameter has been deprecated. Oracle recommends that you use the WALLET_ROOT static initialization parameter and TDE_CONFIGURATION dynamic initialization parameter instead.
- This method creates a new datafile with encrypted data.
- Example Syntax: SQL> alter tablespace TDE_ORACLEDBWR_TBS encryption online using 'AES192' encrypt file_name_convert=('/u02/app/oracle/oradata/ORADBWR/tde_tbs1.dbf','/u02/app/oracle/oradata/ORADBWR/tde_tbs1_encrypted.dbf');
  Tablespace altered.
- If you're not using Oracle Managed File (OMF), stating from 19c you can omit the FILE_NAME_CONVERT clause, but if you're using an Oracle 12.2 or 18c you still have to specify it, otherwise you'll end up with an error:

  ORA-28425: missing a valid FILE_NAME_CONVERT clause
- If you're using Oracle Managed File (OMF) you can't use the FILE_NAME_CONVERT, or you'll face an error: ORA-28437: unexpected FILE_NAME_CONVERT clause with Oracle Managed Files
- Online method requires twice the space as the tablespace been worked on as work (encryption or decryption) happens out of place.

Heema Satapathy
Manager – Infrastructure & Cloud Solutions
Computer Technology Resources Inc
hsatapathy@ctrworld.com
+1(714)-665-6507 Ext 418