Implementing Encryption in Oracle GoldenGate

Practice Overview

In this practice you will implement the following encryption GoldenGate options:

- Encrypting the database login credentials using the credential store.
- Encrypting the data in the remote trail files using the Wallet method.

Implementing Encryption in Oracle GoldenGate

A. Encrypting DB login credential using the credential store

In this section of the practice, you will encrypt the database login username and password using the credential store solution.

1. Stop the Extract and the Replicat, if they are running.

2. On the source database, create the credential store wallet and save the database credential in it using the following `ggsci` commands.

   The created credential store can be used in the parameter files as well as in the `Dblogin` commands.

   If you make a mistake entering a user, the command to remove a user is `Alter CredentialStore Delete User <username>`

```
Create Wallet
Add CredentialStore
Alter CredentialStore Add User ogg@db1 Password oracle Alias oggdb1
Alter CredentialStore Add User ogg@db2 Password oracle Alias oggdb2
```

3. View the information of the created credential store.

```
Info CredentialStore
```

4. In the `ggsci` command prompt of the source system, test the created credentials.

```
DBLogin UserIDAlias oggdb1
```

5. If the previous DBLogin was successful, copy the created source wallet files to the target system. “sso” extension stands for single sign-on.

```
scp -p ./dircrd/cwallet.sso oracle@ggsrv2:/u01/app/oracle/product/ogg/dircrd
scp -p ./dirwlt/cwallet.sso oracle@ggsrv2:/u01/app/oracle/product/ogg/dirwlt
```

6. In the source system, open the parameter file of the Extract and remove the USERID parameter from it. Replace it with the UserIDAlias parameter as follows:

```
…
~~USERID ogg, PASSWORD oracle~~
UserIDAlias oggdb1
…
```

7. In the target system, open the parameter file of the Replicat and remove the USERID parameter from it. Replace it with the UserIDAlias parameter as follows:

```
…
~~USERID ogg, PASSWORD oracle~~
UserIDAlias oggdb2
…
```

8. Start up the processes and make sure their status is RUNNING.

B. Encrypting the trails using the Wallet

In this section of the practice, you will encrypt the data in the trail files using the Wallet.

9. Stop the Extract, the Data Pump and the Replicat.

10. On the source database, open the Wallet file.

    Because you already created a wallet in the previous section of the practice, you should not create the Wallet again. You just need to open it. If there is no existing Wallet, you should create it.

```
Open Wallet
```

11. Add a master key to the Wallet.

```
Add MasterKey
```

12. Display the version of the master key.

```
Info MasterKey
```

13. Copy the source wallet files to the target system.

```
scp -p ./dircrd/cwallet.sso oracle@ggsrv2:/u01/app/oracle/product/ogg/dircrd
scp -p ./dirwlt/cwallet.sso oracle@ggsrv2:/u01/app/oracle/product/ogg/dirwlt
```

14. Verify that the master key made it to the target. The output of the `Info` command should match the output of the same command in the source system.

```
Open Wallet
Info MasterKey
```

15. In the source system, open the parameter file of the Data Pump process and add the `EncryptTrail` parameter straight after the Extract parameter, as shown below.

    Replicat will automatically decrypt the data from the trail files. No need to configure that in its parameter file.

```
edit params psrv1
```

```
Extract psrv1
-- Add the following:
EncryptTrail AES256
...
```

16. Start up all the processes and make sure their status is RUNNING.

17. View the report of the Data Pump process and verify the trail file encryption. It appears in the end of the report as follows:

```
INFO    OGG-05519  Output trail file encryption: AES256.
```

18. Run the following query in both databases to make sure that the change synchronization is working properly.

```
col rtimestamp format a30
SQL> select * from SAMPLE order by RTIMESTAMP desc;
```

19. Use the `logdump` utility to verify that the data in the remote trail files is encrypted.

a. In the target system, specify the most recent remote trail files.

```
ls -alh ./dirdat/rt*
```

b. Start the Logdump utility and open the file.

```
Logdump> open ./dirdat/rt000000***
```

c. Move the pointer to the end of the file.

```
Logdump> position eof
```

d. Scan for the header in reverse direction to check out the last operation registered in the file. Verify the data is encrypted.

```
sfh prev
```

Summary

Wallet can be used to encrypt the database login credential in the parameter files as well as in the `DBLogin` command.

Wallet can also be used to encrypt the messages sent from the Data Pump to the remote target systems and also the data in the trail files.

By the end of this practice, following are the code in each process parameter file:

```
Extract esrv1
INCLUDE /u01/app/oracle/product/ogg/dirprm/header.mac
UserIDAlias oggdb1
ALLOWDUPTARGETMAP
ExtTrail ./dirdat/es
Table HR.JOB_HISTORY;
Table HR.EMPLOYEES,
TOKENS ( TK-OSUSER = @GETENV ('GGENVIRONMENT' , 'OSUSERNAME'),
         TK-HOST = @GETENV('GGENVIRONMENT' , 'HOSTNAME'));
Table HR.JOBS;
Table HR.DEPARTMENTS;
Table HR.LOCATIONS;
Table HR.REGIONS;
Table HR.SAMPLE;
TABLE HR.EVENTS, FILTER (@STREQ (EVENT, 'STOP EXTRACT' )), EVENTACTIONS (IGNORE
TRANS,STOP);
TABLE HR.EVENTS;
```

```
Extract psrv1
EncryptTrail AES256
RmtHost ggsrv2, MgrPort 7810
RmtTrail ./dirdat/rt
Passthru
Table HR.*;
```

```
Replicat rsrv2
INCLUDE /u01/app/oracle/product/ogg/dirprm/header.mac
DiscardFile ./dirrpt/rsrv2.dsc, Purge
UserIDAlias oggdb2
Map HR.EMPLOYEES, Target HRTRG.EMPLOYEES,
 SQLEXEC (id GET_TITLE, QUERY ' SELECT JOB_TITLE FROM HRTRG.JOBS
 WHERE JOB_ID = :V_JOB_ID ',
 PARAMS (V_JOB_ID = JOB_ID)),
 COLMAP (USEDEFAULTS, IT_JOB_FLAG = @IF (@STREQ (JOB_ID, 'IT_PROG'), 'Y', 'N'),
  WORKING_DAYS = @DATEDIFF ('DD', HIRE_DATE, @DATENOW ()),
  TITLE = @GETVAL(GET_TITLE.JOB_TITLE))
);
Map HR.EMPLOYEES, Target HRTRG.EMP_HISTORY, INSERTALLRECORDS,
ColMap (USEDEFAULTS,
 OP_TYPE = @GetEnv('GGHEADER', 'OPTYPE'),
 TRAN_TIME = @GetEnv('GGHEADER', 'COMMITTIMESTAMP'),
 BEFORE_AFTER = @GETENV ('GGHEADER', 'BEFOREAFTERINDICATOR'),
 OSUSERNAME = @TOKEN('TK-OSUSER'),
 HOSTNAME = @TOKEN('TK-HOST')
```

```
);
MAP HR.EVENTS, TARGET HRTRG.EVENTS, FILTER (@STREQ (EVENT, 'STOP REPLICAT' )),
EVENTACTIONS (IGNORE TRANS,STOP);
Map HR.*, Target HRTRG.*;
```