Code Objectives:

1. Pass the ORACLE\_HOME, PATH, SQLPLUS\_HOME, ORACLE\_SID in script
2. Pass the sys user password in script
3. Create SQL file, Command Output files in PWD

Modules within shell scripts

1. Check the Oracle DB version
2. Command to execute a SQL, print output
3. Print output spool file from function

Design consideration:

1. Single shell script or multiple shell script
2. Possibility to plug more tests in future
3. Tasks mentioned under each CVE require code to put and test

|  |  |  |  |
| --- | --- | --- | --- |
| **Vulnerability** | **Version Impacted** | **Steps to check** | **Steps to Mitigate** |
| CVE-2021-2351 | 12.1.0.2, 12.2.0.1, 19c | **Task1** : Shell script to run the command and  compare the result with defined values  create the text file as recommendations  **Run "$ORACLE\_HOME/bin/adapters"**  **Task 2**: Shell script to run the **sql command** and  compare the result with possible values  create the text file as recommendations for  *server side setting*  *client side settings*  *jdbc connection pool setting*  *allowed\_login\_version configuration*  **SQL>select network\_service\_banner**  **from v$session\_connect\_info**  **where sid in (select distinct sid from v$mystat);** | Recommended Server Side Setting:  SQLNET.ENCRYPTION\_SERVER=REQUIRED  SQLNET.ENCRYPTION\_TYPES\_SERVER=(AES256)  SQLNET.CRYPTO\_CHECKSUM\_SERVER=REQUIRED  SQLNET.CRYPTO\_CHECKSUM\_TYPES\_SERVER=(SHA512)  Recommended Client Side Settings:  SQLNET.ENCRYPTION\_CLIENT=REQUIRED  SQLNET.ENCRYPTION\_TYPES\_CLIENT=(AES256)  SQLNET.CRYPTO\_CHECKSUM\_CLIENT=REQUIRED  SQLNET.CRYPTO\_CHECKSUM\_TYPES\_CLIENT=(SHA512)  Verification  To confirm the encryption level of the network traffic, use either client or server side sqlnet tracing.  e.g. From the client edit the sqlnet.ora and add a line.  trace\_level\_client=16 - Ensure to switch off during normal operations.  Note: Verify the same for JDBC connections:  e.g. in JDBC URL string:  prop.setProperty(OracleConnection.CONNECTION\_PROPERTY\_THIN\_NET\_ENCRYPTION\_TYPES, "(AES256)");  prop.setProperty(OracleConnection.CONNECTION\_PROPERTY\_THIN\_NET\_CHECKSUM\_TYPES, "(SHA512)");  For earlier Databases see Oracle Net services parameter  - SQLNET.ALLOWED\_LOGON\_VERSION  For Oracle 12c onwards see  - SQLNET.ALLOWED\_LOGON\_VERSION\_SERVER  - SQLNET.ALLOWED\_LOGON\_VERSION\_CLIENT |
| CVE-2021-2328 Code-added | 12.1.0.2, 12.2.0.1, 19c | **Task1**: Shell script to run the command of **SQL #1 and SQL#2** with values when Oracle Text is installed  If matches,   * **run SQL #3** and spool the output * run **SQL #4 (with added privilege “CREATE ANY” “ALTER ANY” )** spool the output * Generate output to suggest CTXSYS user account lock and expire/ generate command/ execute **SQL #5**     **SQL #1**  SELECT idx\_id,idx\_owner,idx\_name,idx\_type,idx\_sync\_type  FROM ctxsys.ctx\_indexes  WHERE idx\_owner <> 'CTXSYS'  ORDER BY 1,2,3;  If SQL #1 Returns No Rows – Oracle Text is not being used in this database  **SQL #2**  select name, max(version), detected\_usages  from DBA\_FEATURE\_USAGE\_STATISTICS  where name like '%Text%'  group by name, detected\_usages;  If SQL #2 Returns No Rows – Oracle Text is not in use  **SQL #3**  SELECT GRANTEE,PRIVILEGE,ADMIN\_OPTION  FROM DBA\_SYS\_PRIVS  WHERE PRIVILEGE IN ('CREATE ANY PROCEDURE','ALTER ANY TABLE');  If SQL #3 Returns No Rows – Nobody has "Create Any Procedure" and "Alter Any Table" Privileges  Note: If SQL #3 returns any users. Access and privileges would need to be reviewed and revoked (wherever possible)  **SQL #4**  select \* from dba\_role\_privs  where grantee = 'PUBLIC';  *If SQL #4 Returns Data - Please review the data returned and ensure that none of the "CREATE ANY" or "ALTER ANY" privileges are granted to PUBLIC role* | **SQL #5**  alter user CTXSYS account lock password expire; |
| CVE-2021-2329  Code-added  CVE-2021-2337  Code-added | 12.1.0.2, 12.2.0.1, 19c | **MVP 1**  **Task 1**: Shell script to run the command of **SQL #1**  Check the version, if 10g through 11g, generate output with  “From 10g through 11g, use catnoqm.sql to remove XML DB.”  **MVP 2**  **Task 1:** run **SQL #1** check if XDB is present  Check if following products are installed  Check the version, if 10g through 11g,  “Run command **catnoqm.sql** to remove XML DB.” 🡨 check more information like path and variable, automate this  **SQL #1**  select comp\_name, version, status  from dba\_registry  where comp\_id = 'XDB';  If SQL #1 Returns Rows – Stutus Column should be ‘VALID’ if the XML Component is in use. | Single shell for 2329 and 2337 |
| CVE-2020-27193  CVE-2020-26870  CVE-2021-2460 Code-added | Prior to 21.1.0.00.01 | **MVP 1**  **Task 1**: Shell script to run the command and check the output   * compare the result of **SQL #1,** if matches, print the recommendations * If matches run **SQL #2**, print the recommendations to secure the network port * If matches run **SQL #3**, check the value, compare and spool the output * If matches to value 8081, print mitigation to disable the XDB * Run **SQL #4** check and compare the value and spool the output with name and status * Run **SQL #7**, compare the output, spool output if result does not match with ADMIN   As Mitigation   * run **SQL #6** and print output * run **SQL #8** and print output,   + print the recommendations for using TCPS in dispatcher if SSL is configured   + recommend running **SQL #9** * check version, if print version with suggestion to remove APEX/ generate commands/ execute commands * configure the xdbconfig.xml file **🡨 TODO more investigation**   **SQL #1**  SELECT SCHEMA APEX\_VER  FROM dba\_registry  WHERE comp\_id = 'APEX';  **SQL #2**  SELECT DBMS\_XDB.gethttpport  FROM DUAL;  **SQL #3**  select value from v$parameter where name='dispatchers';  **SQL #4**  select account\_status  from dba\_users  where username = 'APEX\_PUBLIC\_USER';  **SQL #7**  select w.short\_name workspace\_name,  u.user\_name,  u.first\_name,  u.last\_name,  u.account\_expiry  from apex\_040000.wwv\_flow\_fnd\_user u,  apex\_040000.wwv\_flow\_companies w  where u.security\_group\_id = w.provisioning\_company\_id  order by 1,2; | conn / as sysdba  **SQL #6**  exec dbms\_xdb.sethttpport(0);  exec dbms\_xdb.setftpport(0);  alter system register;  **SQL #8**  SELECT username, account\_status  FROM dba\_users where username = 'ANONYMOUS';  If SQL #8 Returns No Rows  **SQL #9**  @apxremov.sql  If SQL #8 Returns Rows |
| CVE-2021-2333    Code-added | 12.1.0.2, 12.2.0.1, 19c | **MVP 1**  **Task1** : Shell script to run the command and check the output  compare the result of **SQL #1,**  **SQL #1**  select comp\_name, version, status  from dba\_registry  where comp\_id = 'XDB'; | compare the value of status from **SQL #1** and run **SQL #2**  MVP 1: Print result with suggested commands to   * revoke the privilege * audit the privilege   MVP 2: Print result and execute suggested commands  **SQL #2**  SELECT GRANTEE,PRIVILEGE,ADMIN\_OPTION  FROM DBA\_SYS\_PRIVS  WHERE PRIVILEGE = 'ALTER USER';  If SQL #2 Returns Rows - List users with ALTER USER Privilege |
| CVE-2019-17545  Code-added | 12.2.0.1, 19c | **MVP 1**  **Task1** : Shell script to run **SQL #1, SQL #2, #3 and #4**  compare the result   * print the commendations * generate commands to **drop user or lock the user** * execute commands to **drop user or lock the user**   check the oracle version and prepare commands to lock, expire, drop accordingly  Generate commands to drop user and lock and expired the users  Run **SQL #6** and spool the output and provide recommendations  **SQL #1**  select value from v$option where parameter = 'Spatial';  If SQL #1 Returns Rows - 'True' returned if Spatial is in use  Check 2  Also query registry.  **SQL #2**  SELECT substr(comp\_id,1,12) Comp\_ID,  Status, substr(Version,1,10 Version,  substr(Comp\_Name,1,40) Comp\_Name  FROM DBA\_Registry  WHERE comp\_id = 'SDO';  *select version from dba\_registry where comp\_id='SDO';*  If SQL #2 Returns Rows - If SDO isn't list no action required  What Spatial Features are being used?  MDSYS.SDO\_FEATURE\_USAGE table contains one row for each Spatial and Graph feature whose usage is automatically tracked.  **SQL #3**  select \* from MDSYS.SDO\_FEATURE\_USAGE  where USED = 'Y';  If SQL #3 Returns Rows  Check presence of Spatial object data types in database.  **SQL #4**  select owner, object\_name, object\_type  from dba\_objects  where object\_name like 'SDO\_%';  If SQL #4 Returns Rows - Note: Spatial and Graph utilises XML DB. | If Spatial is not required:  **SQL #5**  drop user MDSYS cascade;  drop user mddata cascade;  -- Only created as of release 11g:  drop user spatial\_csw\_admin\_usr cascade;  drop user spatial\_wfs\_admin\_usr cascade;  If SQL #5 Returns Rows  Note: If you do not wish to drop the user, alternatively, the MDSYS schema could be locked and expired.  It is best practices to list the privileges on all spatial objects, applying the principle of least privilege:  **SQL #6**  col grantor for a10  col grantee for a10  col privilege for a10  select grantor,  grantee,  table\_name,  privilege,  grantable  from all\_tab\_privs  where table\_schema = 'MDSYS'  order by table\_name, privilege;  If SQL #6 Returns Rows - Apply Principle of Least Privelege |
| CVE-2021-2330  Code-added | 19c | **MVP 1**  **Task 1**: Shell script to check the content of sqlnet.ora  Compare values and print values and recommendations  **Task 2**:  Run the **SQL #1** and print the output  Run the SQL #2 , validate result for any users of SQL #1 does not have any default tablespace **🡨 TODO more investigation**  This could be verified by reviewing the sqlnet.ora file for parameters mentioned in the example below -  tcp.validnode\_checking = yes  tcp.invited\_nodes = (hostname1, hostname2)  PROTECT/HARDEN:  To ensure this vulnerability are not exploited, ensure the CREATE TABLE privilege is assigned as per the principle of least privilege.  SQL #1  SELECT GRANTEE,  PRIVILEGE,  ADMIN\_OPTION  FROM DBA\_SYS\_PRIVS  WHERE PRIVILEGE = 'CREATE TABLE';  If SQL #1 Returns Rows - Users with 'CREATE TABLE' Privilige | SQL #2  col ownr format a20 justify c heading 'Owner'  col name format a20 justify c heading 'Tablespace' trunc  col qota format a12 justify c heading 'Quota (KB)'  col used format 999,999,990 justify c heading 'Used (KB)'    break on ownr skip 1    select  username ownr,  tablespace\_name name,  decode(greatest(max\_bytes, -1),  -1, 'Unrestricted',  to\_char(max\_bytes/1024, '999,999,990')  ) qota,  bytes/1024 used  from  dba\_ts\_quotas  where  max\_bytes!=0  or  bytes!=0  order by  1,2  If SQL #2 Returns Rows |
| CVE-2020-7760 | 19c | **MVP 1**  **Task 1**: Shell script to run the command and check the output  compare the result of **SQL #1,**  **Task 2:**  Shell script that runs command  **lsnrctl status** and **lsnrctl services**  Compare the output of service with output **of SQL #1**  Check the multitenancy if matches run **SQL #4**  Print **SQL #5** as text output to reset  **SQL #1**  select value from v$parameter where name='dispatchers';  If SQL #1 Returns Rows - Should Return: dispatchers="(PROTOCOL=TCP)(SERVICE=<sid>XDB)"  Verify that the XDB dispatchers are running by reviewing LSNRCTL STATUS and LSNRCTL SERVICES output.  lsnrctl status. Where configured the XDB protocols and ports should be displayed.  Example below -  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=<name>.com)(PORT=8080))(Presentation=HTTP)(Session=RAW))  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=<name>.com)(PORT=2100))(Presentation=FTP)(Session=RAW)) | **SQL #2**  SELECT 'https://'||SYS\_CONTEXT('USERENV','SERVER\_HOST')||'.'||  SYS\_CONTEXT('USERENV','DB\_DOMAIN')||':'||  dbms\_xdb\_config.gethttpsport()||'/em/'  from dual;  If SQL #2 Returns Rows  **SQL #3**  SELECT 'https://'||SYS\_CONTEXT('USERENV','SERVER\_HOST')||':'||dbms\_xdb\_config.gethttpsport()||'/em/'  from dual;  If SQL #3 Returns Rows  For non Multitenant Container Database (CDB) run  **SQL #4**  SELECT dbms\_xdb\_config.gethttpsport()  FROM dual;  If SQL #4 Returns Rows  **SQL #5**  alter system reset dispatchers scope=spfile; |
| CVE-2021-2438 Code-added | 12.1.0.2, 12.2.0.1, 19c | **MVP 1**  **Task 1**: Shell script to run the command and check the output  compare the result of **SQL #1,**  if result matches run **SQL #2**, print output with recommendations  if result matches run **SQL #3**, **SQL #4** print output,   * run **SQL #5 , #6** for matched role or user and print output * Prepare **SQL #7, #8 🡨 TODO more investigation** * Run **SQL #9** and print output * Check Oracle version, Customize and run **SQL #10** * Run **SQL #11, SQL #12, SQL #13, SQL #14, SQL #15, SQL #16** and print output   **SQL #1**  SELECT SUBSTR(comp\_name, 1, 30) comp\_name,  SUBSTR(version, 1, 20) version, status  FROM dba\_registry  ORDER BY comp\_name;  **SQL #2**  select currently\_used, name  from dba\_feature\_usage\_statistics  where name like '%Java%';  If SQL #2 Returns Rows - Feature is in use.  **SQL #3**  SELECT GRANTEE,PRIVILEGE,ADMIN\_OPTION  FROM DBA\_SYS\_PRIVS  WHERE PRIVILEGE = 'CREATE PROCEDURE';  If SQL #3 Returns Rows - Users with "CREATE PROCEDURE" privilege.  Check 4  Check which users have been added to the Java Policy to be able to load/run java in the database and cross reference it against users with the Create Procedure system privilege (i.e. who can create procedures and load java in the database)  **SQL #4**  col "kind" for a8  col "grantee" for a10  col "type" for a25  col "name" for a10  col "action" for a10  select kind "kind", grantee "grantee", type\_name "type", name "name", action "action"  from dba\_java\_policy;  If SQL #4 Returns Rows | **SQL #5**  select trim(TYPE\_NAME)||' -- '||trim(NAME)||' -- '||trim(action) "Perm. granted to User/Role"  from dba\_java\_policy  where grantee = '<User/Role>';  If SQL #5 Returns Rows  **SQL #6**  select type\_name, name, action, enabled, seq  from dba\_java\_policy  where grantee = '<user>';  If SQL #6 Returns Rows - Revoke permissions,  **SQL #7**  execute dbms\_java.revoke\_permission ('<user>','<name>','<action>','<enabled>');  If SQL #7 Returns Rows, Then delete permission  **SQL #8**  execute dbms\_java.delete\_permission(<seq>);  If SQL #8 Returns Rows  **SQL #9**  SELECT TABLE\_NAME, PRIVILEGE, GRANTEE  FROM DBA\_TAB\_PRIVS  WHERE GRANTEE='PUBLIC' AND PRIVILEGE='EXECUTE'  AND TABLE\_NAME IN ('DBMS\_JAVA','DBMS\_JAVA\_TEST');  If SQL #9 Returns Rows  **SQL #10**  SELECT GRANTEE  FROM DBA\_ROLE\_PRIVS  WHERE GRANTED\_ROLE IN ('JAVASYSPRIV','JAVADEBUGPRIV','JAVAUSERPRIV','JAVA\_ADMIN','JAVA\_DEPLOY')  AND GRANTEE != 'SYS';  If SQL #10 Returns Rows  Note, add 'JAVAIDPRIV' to granted\_role list for 11g/12.1, and 'DBJAVASCRIPT' for 12.2  **SQL #11**  SELECT OS\_USERNAME  FROM SYS.JAVA$RUNTIME$EXEC$USER$  WHERE OWNER#=0;  If SQL #11 Returns Rows  **SQL #12**  SELECT GRANTEE  FROM DBA\_JAVA\_POLICY WHERE TYPE\_NAME = 'java.io.FilePermission'  AND ACTION LIKE '%execute%'  AND (NAME='<<ALL FILES>>' OR NAME LIKE '%\*%')  AND KIND = 'GRANT'  AND ENABLED='ENABLED'  AND GRANTEE != 'JAVASYSPRIV';  If SQL #12 Returns Rows  **SQL #13**  SELECT GRANTEE  FROM DBA\_JAVA\_POLICY  WHERE TYPE\_NAME = 'java.security.AllPermission'  AND KIND = 'GRANT' AND ENABLED='ENABLED'  AND GRANTEE != 'SYS';  If SQL #13 Returns Rows  **SQL #14**  SELECT GRANTEE  FROM DBA\_JAVA\_POLICY  WHERE TYPE\_NAME = 'java.io.FilePermission'  AND ACTION LIKE '%write%'  AND NAME='<<ALL FILES>>'  AND KIND = 'GRANT'  AND ENABLED='ENABLED'  AND GRANTEE != 'JAVASYSPRIV';  If SQL #14 Returns Rows  **SQL #15**  SELECT GRANTEE,  NAME, SEQ  FROM DBA\_JAVA\_POLICY  WHERE TYPE\_NAME = 'java.lang.RuntimePermission'  AND NAME LIKE '%loadLibrary%'  AND KIND = 'GRANT'  AND ENABLED='ENABLED'  AND GRANTEE NOT IN ('SYS', 'ORDSYS');  If SQL #15 Returns Rows  **SQL #16**  SELECT GRANTEE,  PRIVILEGE  FROM DBA\_TAB\_PRIVS  WHERE TABLE\_NAME = 'JAVA$POLICY$';  If SQL #16 Returns Rows |
|  |  |  |  |

Possible Template of Shell script

|  |
| --- |
| Discovery\_SQL\_1 {  Echo "Discovery\_SQL\_1"  }    Discovery\_SQL\_2 {  Echo "Discovery\_SQL\_2"  }    Discovery\_SH\_1 {  Echo "Discovery\_SH\_1"  }    Mitigation\_SQL\_1 {  Echo "Mitigation\_SQL\_1"  }    Mitigation\_SQL\_2 {  Echo "Mitigation\_SQL\_2"  }    Mitigation\_SQL\_3 {  Echo "Mitigation\_SQL\_3"  }    Mitigation\_SH\_1 {  Echo "Mitigation\_SH\_1"  }    SYSPASS="MyP@ssw0rd"  echo $ORACLE\_HOME  Check\_oracle\_version {  Return version  }    if Discovery\_SQL\_1 = "YES" and Discovery\_SQL\_2 = "Yes"    then      Mitigation\_SQL\_1      Mitigation\_SQL\_1    else      Mitigation\_SH\_1  fi |
|  |