Objectives:

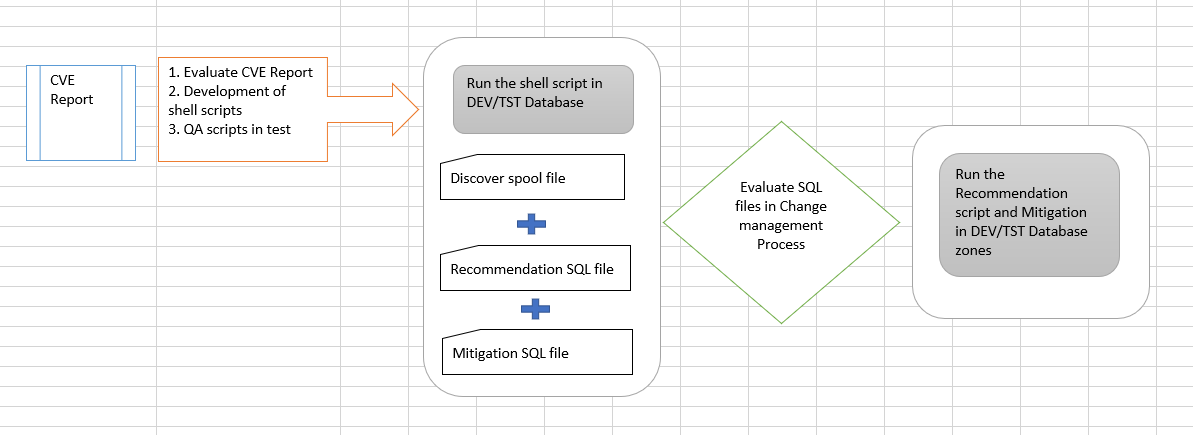
1. Executable scripts that perform database checks in form of SQL or shell commands and generate the suggestions and migitations.
2. Use the environment variables locally in shell script or in config file, ORACLE\_HOME, PATH, SQLPLUS\_HOME, ORACLE\_SID
3. Pass the sys user password in script, or remain connect with “/”.
4. Change the shell scripts with u+x permission, Run the shell script using Oracle installed user

Modules in the shell scripts

1. SQL to check the Oracle DB version
2. Command to execute a SQL as discovery function,
3. Print recommendation and mitigation on screen and in SQL file after matching return of the function.

Design placement consideration:

1. **Phase 1:** Develop scripts to run as a single shell script
2. **Phase 2:** Merge the code in consolidated master shell script.
3. During publishing of every CVE report (JUL-21, Oct-21), After development, there would be
   1. Single consolidated file like CVE-2021-Jul.sh
   2. Multiple shell scripts as deliverables CVE-2021-2328.sh
4. Repeat the same process for future CVE Report
5. Artifacts to be develop/ shared- Yet to decide.



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CVE Month** | **Vulnerability** | **Version Impacted** | **Code Development** | **Included Discovery and Mitigations** |
| Jul-21 | CVE-2021-2351 | 12.1.0.2, 12.2.0.1, 19c | Yes | Yes |
| CVE-2021-2328 | 12.1.0.2, 12.2.0.1, 19c | Yes | Yes |
| CVE-2021-2329 | 12.1.0.2, 12.2.0.1, 19c | Yes | Yes |
| CVE-2021-2337 | Yes | Yes |
| CVE-2020-27193 | Prior to 21.1.0.00.01 | Yes | Yes |
| CVE-2020-26870 | Yes | Yes |
| CVE-2021-2460 | Yes | Yes |
| CVE-2021-2333 | 12.1.0.2, 12.2.0.1, 19c | Yes | Yes |
| CVE-2019-17545 | 12.2.0.1, 19c | Yes | Yes |
| CVE-2021-2330 | 19c | Yes | Yes |
| CVE-2020-7760 | 19c | Working |  |
| CVE-2021-2438 | 12.1.0.2, 12.2.0.1, 19c | Yes | Yes |
| Oct-21 | CVE-2021-35599 | 21c | No |  |
| CVE-2021-35619 | 12.1.0.2, 12.2.0.1, 19c, 21c | No |  |
| CVE-2021-2332 | 12.1.0.2, 12.2.0.1, 19c | No |  |
| CVE-2021-35551 | 12.2.0.1, 19c, 21c | No |  |
| CVE-2021-35557 | 12.1.0.2, 12.2.0.1, 19c, 21c | No |  |
| CVE-2021-35558 | No |
| CVE-2021-26272 | 12.1.0.2, 12.2.0.1, 19c | No |  |

https://github.com/deepakdba/cve\_checklist

|  |  |  |  |
| --- | --- | --- | --- |
| **Vulnerability** | **Version Impacted** | **Steps to check** | **Steps to Mitigate** |
| CVE-2021-2351  Code-added | 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 |
| CVE-2021-35599 | 21c | PROTECT/HARDEN:  Vulnerability is specific to "local logon", i.e. connection on the same host computer where Oracle database is installed, thus harden operating system access (ssh, telnet) and SQL Command Line (or other Oracle command-line utility) which allows local connection.  Specifically for Oracle Zero Downtime Migration (ZDM) software.  Protect OS zdm group and zdmuser, and ZDM Oracle home binaries.  Ensure relevant firewall security on port 1521 which must be open for ZDM node connections to source/target databases.  Use Private SSH keys between ZDM host, and source/target database host.  When no longer in use, shut down the vulnerable service(s).  Zero Downtime Migration Software Shutdown / Uninstallation  1. Connect as user with which you want to uninstall the ZDM software on the Linux host  2. Stop the Zero Downtime Migration service  $ <ZDM\_HOME>/bin/zdmservice stop  3. Run the following command to uninstall the software.  $ <ZDM\_HOME>/bin/zdmservice stop deinstall |  |
| CVE-2021-25122 | 12.2.0.1, 19c, 21c | DISCOVER:  Do you have RAC database/Grid Infrastructure installed?  Do you have Oracle Quality of Service (QoS) Management enabled and configured?  Which HTTP port is configured for QoS Tomcat server?  Check #1  Check QoS Management Server is a CRS Resource  With Grid Infrastructure Home environment set  CMD #1 {  crsctl status res -t  }  Results from Check #1 Should Show qosmserver as resource if installed  --------------------------------------------------------------------------------  Cluster Resources  --------------------------------------------------------------------------------  ora.qosmserver  1 ONLINE ONLINE dbn02 STABLE  Check #2 QoS Management Server enabled check.  With Grid Infrastructure Home environment set  CMD #2 {  srvctl status qosmserver  }  Results from Check #2 Should Detail if Enabled and on What node  QoS Management Server is enabled.  QoS Management Server is running on node <node>  Check #3 As vulnerability is specific to Apache Tomcat HTTP protocol, what port is http  On server where qosmserver is running.  CMD #3 {  ps -ef | grep tomcat  }  Results of Check #3 should show an os process, with "-Doracle.http.port=<port details". Default of port 8888 or 8885 is expected.  Check #4 Even where qosmserver enabled in CRS, further Enabling configuration is required for QoS.  To check if QoS is enabled in the GUI see following documentation:  https://docs.oracle.com/en/database/oracle/oracle-database/12.2/apqos/determining-if-qos-is-enabled.html#GUID-DB443BE3-D21C-4909-A168-3285D145C052  To determine if enabled from command line, check status of APPQOSSYS user, which is an internal database user/schema for exclusive use  by the Quality of Service Management functionality.  SQL #1 {  select username, account\_status  from dba\_users  where username = 'APPQOSSYS';  }  Result of Check #4 SQL by default would be APPQOSSYS account "EXPIRED & LOCKED" and QoS not configured. If account open then account has been activated for QoS use. | PROTECT/HARDEN:  The Tomcat distribution provided with GI for QoS is customized profiled version of Tomcat, with limited JAR files deployed for the Tomcat Web Container, thus is distributed as a secure application.  It is recommended to firewall the ports that QoS uses, namely 8888(RMI) and 8895(HTTP) defaults.  View configuration for QoS.  With Grid Infrastructure Home environment set  CMD #4 {  srvctl config qosmserver  }  Where required modify the HTTP port.  CMD #5 {  srvctl modify qosmserver -httpport port  srvctl stop qosmserver  srvctl start qosmserver  }  See:  srvctl qosmserver commands.  https://docs.oracle.com/en/database/oracle/oracle-database/12.2/cwadd/server-control-command-reference.html#GUID-D8A9FE81-F5AA-443E-B489-FFF557442E14  For Attack Surface Reduction, if QoS Is Not Being Used (confirmed by Check #4 in DISCOVER section) then disable qosmserver.  CMD #6 {  srvctl disable qosmserver -node <node>  } |
| CVE-2021-35619  To develop | 12.1.0.2, 12.2.0.1, 19c, 21c | DISCOVER:  Perform SQL Checks To Verify Threat Landscape.  SQL #1 Check the Database Embedded JVM (OJVM) status in the database.  SQL #1 {  SELECT SUBSTR(comp\_name, 1, 30) comp\_name, SUBSTR(version, 1, 20) version, status  FROM dba\_registry  ORDER BY comp\_name;  }  SQL #1 Oracle JVM component is installed if "JServer JAVA Virtual Machine" and "Oracle Database Java Packages" are detailed in DBA\_REGISTRY, and usable is status is "VALID".  SQL #2 Check Java usage. For Oracle version 11.2 or later query the DBA\_FEATURE\_USAGE\_STATISTICS view to confirm if the Java features are being used.  SQL #2 {  select currently\_used, name  from dba\_feature\_usage\_statistics  where name like '%Java%';  }  If SQL #2 Returns Rows With Currently In Used = "TRUE" - Feature is in use.  SQL 3# Validate users/roles who have the privilege required to exploit the vulnerability and whether they have Admin option allowing the grantee to grant this privilege  to another database account.  SQL #3 {  SELECT GRANTEE, PRIVILEGE, ADMIN\_OPTION  FROM DBA\_SYS\_PRIVS  WHERE PRIVILEGE = 'CREATE PROCEDURE';  }  Where Rows returned for SQL 3# with ADMIN\_OPTION = 'YES' user has both exploitable privilege and ability to assign this privilege to other users.  SQL #4 Validate users/roles who have the privilege required to exploit the vulnerability and Java permissions to load and run 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  where grantee IN (SELECT GRANTEE FROM DBA\_SYS\_PRIVS WHERE PRIVILEGE = 'CREATE PROCEDURE');  }  SQL #4 Will Return Java Permissions For Users With Exploit Privilege For Assessment of PROTECT/HARDEN STIGS | PROTECT/HARDEN:  Ensure that any users with the exploit privilege 'CREATE PROCEDURE' cannot delegate it to other users and those assigned are authorized to have this privilege assignment.  SQL #5 For users returned in SQL #3 that are not administrators, or should not have  SQL #5 {  'CREATE PROCEDURE' privilege revoke privilege.  revoke CREATE PROCEDURE FROM <grantee>;  }  SELECT 'REVOKE CREATE PROCEDURE FROM '|| GRANTEE ||';' FROM DBA\_SYS\_PRIVS WHERE PRIVILEGE = 'CREATE PROCEDURE' AND GRANTEE NOT IN ('DBA','SYS','SYSTEM','DBSNMP','DV\_REALM\_RESOURCE','RECOVERY\_CATALOG\_OWNER','DVF','RESOURCE','MDSYS');  Where privilege is required, revoke and recreate 'CREATE PROCEDURE' without Admin option.  SQL #6 Validate Java permissions for users returned in SQL 4#.  Each user or schema must be assigned the proper permissions to access operating system resources, such as sockets, files, and system properties to load and run Java in the database. DBA\_JAVA\_POLICY describes Java security permissions for all users in the database.  SQL #6 {  select trim(TYPE\_NAME)||' -- '||trim(NAME)||' -- '||trim(action) "Perm. granted to User/Role"  from dba\_java\_policy  where grantee = '<User/Role>';  }  SQL #6 Will Return Java Permissions (such as java.net.SocketPermission) Assigned to Users With Host/Port Details. These should be audited.  SQL #7 Remove permissions to unauthorized users. In order to remove a Java permission from the database, it must first be disabled using the dbms\_java.revoke\_permission procedure.  Then the permissions can be removed via the DELETE\_PERMISSION procedure of the DBMS\_JAVA package to remove or DROP a particular permission from the database. Example below -  SQL #7 {  select type\_name, name, action, enabled, seq  from dba\_java\_policy  where grantee = '<user>';  }  Where rows returned for SQL #7 for unauthorized permissions, which are 'ENABLED' revoke and delete permissions via:  SQL #7B {  execute dbms\_java.revoke\_permission ('<user>','<name>','<action>','<enabled>');  execute dbms\_java.delete\_permission(<seq>);  }  SQL #8 Run the following Security Benchmarks in Sqlplus to ensure Java security in the database meets security framework standards and escalation of privileges cannot be performed outside of administrator control.  Ensure 'EXECUTE' is revoked from 'PUBLIC' on "Java" Packages.  SQL #8 {  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 #8 Returns Rows, then PUBLIC (everyone in database) has access to execute Java packages. Test revoking public access.  revoke execute on DBMS\_JAVA from PUBLIC;  SQL #9 Ensure membership of the JAVASYSPRIV and DEBUG roles is revoked from unauthorized 'GRANTEE'. Below SQL could be used to identify accounts -  SQL #9 {  SELECT GRANTEE, GRANTED\_ROLE  FROM DBA\_ROLE\_PRIVS  WHERE GRANTED\_ROLE IN ('JAVASYSPRIV','JAVADEBUGPRIV','JAVAUSERPRIV','JAVA\_ADMIN','JAVA\_DEPLOY')  AND GRANTEE != 'SYS';  }  If SQL #9 Returns Rows for Grantees other than OWBSYS, DBA, XDB or JAVA\_ADMIN, verify role and revoke privileges.  SQL #9B {  revoke <role> from <grantee>;  }  Role List up to R12 R1. JAVA\_DEPLOY, JAVAUSERPRIV, JAVAIDPRIV, JAVASYSPRIV, JAVADEBUGPRIV, JAVA\_ADMIN  Role List from R12.2 JAVA\_DEPLOY, JAVAUSERPRIV, JAVAIDPRIV, JAVASYSPRIV, JAVADEBUGPRIV, DBJAVASCRIPT, JAVA\_ADMIN    SQL #10 JAVA on OS should be owned by a user other than root or any similar account with high privileges  Below SQL could be used to identify accounts -  SQL #10 {  SELECT OS\_USERNAME  FROM SYS.JAVA$RUNTIME$EXEC$USER$  WHERE OWNER#=0;  }  If SQL #10 Returns Rows, Ensure OS User is low privileged OS user and not Oracle/Application owner, and ensure secure Use of Runtime.exec Functionality in Oracle Database.  https://docs.oracle.com/database/121/JJDEV/chten.htm#JJDEV13638  SQL #10B {  EXEC DBMS\_JAVA.SET\_RUNTIME\_EXEC\_CREDENTIALS('<username>','<password>');  }  SQL #11 Ensure java.io.FilePermission.execute is revoked from unauthorized 'GRANTEE'.  SQL #11 {  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 #11 Returns Rows, revoke Execute All Files Permissions from Unauthorized users.  SQL #11B {  EXEC DBMS\_JAVA.REVOKE\_PERMISSION ('<username>', 'java.io.FilePermission', '<<ALL FILES>>','execute');  }  SQL #12 Ensure java.security.All Permission is revoked from unauthorized 'GRANTEE'.  SQL #12 {  SELECT GRANTEE  FROM DBA\_JAVA\_POLICY  WHERE TYPE\_NAME = 'java.security.AllPermission'  AND KIND = 'GRANT' AND ENABLED='ENABLED'  AND GRANTEE != 'SYS';  }  If SQL #12 Returns Rows, revoke AllPermission from Unauthorized users.  SQL #12B {  EXEC DBMS\_JAVA.REVOKE\_PERMISSION ('<username>','java.security.AllPermission');  }  SQL #13 Ensure wildcard java.io.FilePermission.write is revoked from unauthorized 'GRANTEE'.  SQL #13 {  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 #13 Returns Rows, revoke Write Permissions for All Files from Unauthorized users.  SQL #13B {  EXEC DBMS\_JAVA.REVOKE\_PERMISSION ('<username>','java.io.FilePermission','<<ALL FILES>>','write');  }  SQL #14 Ensure java.lang.RuntimePermission.loadLibrary is revoked from unauthorized 'GRANTEE'.  SQL #14 {  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 #14 Returns Rows, revoke Java Load Library permission by SEQ number  SQL #14B {  EXEC DBMS\_JAVA.DELETE\_PERMISSION(<seq>);  }  SQL #15 Ensure 'ALL' Is Revoked from Unauthorized 'GRANTEE' on database sensitive objects.  SQL #15 {  SELECT GRANTEE, PRIVILEGE  FROM DBA\_TAB\_PRIVS  WHERE TABLE\_NAME like 'JAVA$POLICY$';  If SQL #16 Returns Rows, revoke authorized access from SYS.JAVA$POLICY$  REVOKE <privilege> ON JAVA$POLICY$ FROM <username>;  } |
| CVE-2021-2332 | 12.1.0.2, 12.2.0.1, 19c | Are you using LogMiner?  SQL #1 Run as DBA privileged used.  SQL #1 {  select count(\*) from x$logmnr\_session;  }  Repeat for views. Note, query GV$ views for RAC cluster databases.  View Displays information about  =================== ==================================================  V$LOGMNR\_DICTIONARY The dictionary file in use.  V$LOGMNR\_PARAMETERS Current parameter settings for LogMiner.  V$LOGMNR\_LOGS Which redo log files are being analyzed.  V$LOGMNR\_CONTENTS The contents of the redo log files being analyzed.  If results from SQL #1 queries return No Rows – Oracle Log Miner is not being used in this database  Verify if LogMiner is being used via assessing the alert log for following message type:  LOGMINER: Begin mining logfile for session 1 thread 2 sequence 1234, <PATH> Wed Jan 15 07:02:0 | PROTECT/HARDEN:  Vulnerability requires high level DBA privileges with LogMiner packages owned by the SYS schema. Audit access to the Log Miner tables and packages.  SQL #2 Review Who Has Been Granted DBA Role Privilege Required To Exploit Vulnerability  SQL #2 {  SELECT 'GRANT' AS PATH, GRANTEE, GRANTED\_ROLE  FROM DBA\_ROLE\_PRIVS  WHERE GRANTED\_ROLE = 'DBA' AND GRANTEE NOT IN ('SYS', 'SYSTEM')  UNION  SELECT 'PROXY', PROXY || '-' || CLIENT, 'DBA'  FROM DBA\_PROXIES  WHERE CLIENT IN (SELECT GRANTEE FROM DBA\_ROLE\_PRIVS WHERE GRANTED\_ROLE = 'DBA');  }  Audit SQL #2 – Only authorized administrators should have DBA role access required to exploit vulnerability.  SQL #3 Review Granted Privileges on Sensitive Log Miner Tables/Views  SQL #3 {  SELECT GRANTEE, PRIVILEGE, TABLE\_NAME  FROM DBA\_TAB\_PRIVS  WHERE TABLE\_NAME like '%V%LOGMNR%';  }  If SQL #3 returns privileges on these tables other than DBA or SELECT\_CATALOG\_ROLE then revoke privileges.  SQL #4 Check Execute Privileges on Log Miner Packages  SQL #4 {  SELECT GRANTEE, PRIVILEGE, TABLE\_NAME  FROM DBA\_TAB\_PRIVS  WHERE PRIVILEGE = 'EXECUTE' AND TABLE\_NAME like '%DBMS\_LOGMNR%';  }  LogMiner packages are owned by the SYS schema. If SQL #4 returns privileges other than SELECT\_CATALOG\_ROLE then revoke privileges.  Specifically procedure DBMS\_LOGMNR which can be used to start/stop Log Miner. (i.e. DBMS.LOGMNR.START\_LOGMNR).  For Log Miner Dictionary File Created In DBA Directory, do not use UTL\_FILE\_DIR.  SQL #4B {  select filename from v$logmnr\_dictionary;  }  Verify if the file location is set in UTL\_FILE\_DIR.  SQL #4C {  show parameters utl\_file\_dir  }  If Log miner is not being used, ensure Log Miner is stopped in the database.  Stop Log Miner  SQL #4D {  BEGIN  DBMS\_LOGMNR.end\_logmnr;  END;  /  }  SQL #5 Verify if any Log Miner operations are in progress  SQL #5 {  SELECT OPERATION, STATUS, SQL\_REDO  FROM V$LOGMNR\_CONTENTS;  } |
| CVE-2021-35551 | 12.2.0.1, 19c, 21c | DISCOVER:  As this is RDBMS security issue, all DB's are impacted. Ensure node checking/whitelisting has been configured for Oracle Net Protocol.  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:  The vulnerability requires the attacker to DBA privileges thus perform a privilege analysis to ensure only authorised administrators have DBA role access or SYSDBA system privilege access.  Review SYSDBA system privilege assignment.  SQL #1 As password file (orapwd) enables remote users to connect with administrative privileges through SQL\*Net ensure 'REMOTE\_LOGIN\_PASSWORDFILE' Is Set to  'EXCLUSIVE' or 'NONE'.  SQL #1 {  SELECT UPPER(VALUE) "PARAMETER"  FROM V$PARAMETER  WHERE UPPER(NAME) = 'REMOTE\_LOGIN\_PASSWORDFILE';  }  SQL #1 should return:  EXCLUSIVE - The password file can be used by only one database and the password file can contain names other than SYS.  NONE - Oracle ignores any password file. Therefore, privileged users must be authenticated by the operating system.  SQL #2 Since Oracle database version 12.2, there is no limitation of max entries of password file, thus ensure to routinely check SYS privilege assignment.  SQL #2 {  col username for a10  select \* from v$pwfile\_users;  }  SQL #2 should return SYS username and only trusted administrators granted SYSDBA system privilege. If entries exist that are not trusted "revoke sysdba from <user>;".  Review DBA role assignment.  SQL #3 Ensure 'DBA' Is Revoked from Unauthorized 'GRANTEE'  SQL #3 {  SELECT GRANTEE, GRANTED\_ROLE  FROM DBA\_ROLE\_PRIVS  WHERE GRANTED\_ROLE='DBA' and grantee in (select distinct username from dba\_users) AND GRANTEE NOT IN ('SYS','SYSTEM');  }  SQL #3B {  SELECT ROLE, GRANTED\_ROLE  FROM ROLE\_ROLE\_PRIVS  WHERE GRANTED\_ROLE = 'DBA';  }  SQL #3 Neither query should return rows. First query confirms user assignment of DBA only granted to SYS/SYSTEM, and second query confirms DBA role has not been granted to  another role. If entries exist that are not authorized "revoke dba from <grantee/role>;  SQL #4 Ensure 'DBA' Has Not Been Granted With Admin Option. Admin option allows the grantee to grant privilege to another database account/role thus is a delegation of duties.  SQL #4 {  select 'User '||grantee||' Granted Admin Option On Role '||granted\_role||'' "User Admin Option Grants"  from dba\_role\_privs  where admin\_option = 'YES'  and granted\_role = 'DBA'  and grantee in (select distinct username from dba\_users) order by grantee, granted\_role;  }  SQL #4 {  select 'Role '||role||' Granted Admin Option On Role '||granted\_role||'' "Role Admin Option Grants"  from role\_role\_privs  where admin\_option = 'YES'  and granted\_role = 'DBA'  order by role, granted\_role;  }  SQL #4 First query should return only SYS/SYSTEM users are assigned DBA with Admin option, second query should return no rows confirming DBA role has not been granted to  another role with Admin option.  Note, Revoking DBA or RESOURCE Roles Revokes UNLIMITED TABLESPACE from the User, thus ensure to reassign tablespace quotas after revocation.  Review DBA Privilege Escalation  To prevent the user who is granted DBA role from granting DBA to another user.  With GRANT ANY ROLE,GRANT ANY OBJECT PRIVILEGE, GRANT ANY PRIVILEGE, the user who is granted DBA role can grant DBA role to other user. In order to prevent them from granting DBA role to another user, instead of DBA role,  it is recommended to create a custom nole not including these system privileges.  e.g.  SQL #5 {  CREATE ROLE RESTRICT\_DBA;  }  SQL #6 {  GRANT EXP\_FULL\_DATABASE,DATAPUMP\_EXP\_FULL\_DATABASE,DELETE\_CATALOG\_ROLE,EXECUTE\_CATALOG\_ROLE,GATHER\_SYSTEM\_STATISTICS,JAVA\_ADMIN,JAVA\_DEPLOY,OLAP\_DBA,  OLAP\_XS\_ADMIN,SCHEDULER\_ADMIN,SELECT\_CATALOG\_ROLE,WM\_ADMIN\_ROLE,XDBADMIN,XDB\_SET\_INVOKER TO RESTRICT\_DBA;  }  This should restrict allocation of the custom DBA role.  SQL #7 {  create user <USERNAME1> identified by <PASSWORD>;  User created.  }  SQL #8 {  grant RESTRICT\_DBA to <USERNAME1>;  Grant succeeded.  }  SQL #9 {  create user <USERNAME2> identified by <PASSWORD>;  User created.  }  SQL #10 {  conn <USERNAME1>/<PASSWORD>  Connected.  grant RESTRICT\_DBA to <USERNAME2>;  grant RESTRICT\_DBA to <USERNAME2>  \*  ERROR at line 1: ORA-01932: ADMIN option not granted for role 'RESTRICT\_DBA'  } |
| CVE-2021-35557  CVE-2021-35558 | 12.1.0.2, 12.2.0.1, 19c, 21c | DISCOVER:  As these are core RDBMS issues, all DB's are impacted. Ensure node checking/whitelisting has been configured for Oracle Net Protocol.  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 these vulnerabilities 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' Privilege  Note: As the CREATE TABLE privilege is powerful, ensure that any non authorized administrators with CREATE TABLE privilege have a dedicated tablespace  and or a quota. If the default tablespace is shared with an application, it could cause a denial of service.  SUGGESTIONS:  Use the below SQL query, and cross reference those users that have both CREATE TABLE privilege and unrestricted tablespace quotas.  SQL #2 Cross Reference Users With Create Table Privilege Do Not Have UNLIMITED TABLESPACE.  SQL #2 {  SELECT GRANTEE,PRIVILEGE,ADMIN\_OPTION  FROM DBA\_SYS\_PRIVS  WHERE PRIVILEGE = 'UNLIMITED TABLESPACE' ;  }  If SQL#2 Returns Rows, audit whether user should be allowed to use an unlimited amount of any tablespace in the database, and override all explicit  tablespace quotas for the user. Otherwise introduce quotas.  SQL #3 {  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 #3 Returns Rows ensure User Quota on Tablespace. e.g. alter user HR quota 100M on USERS; |
| CVE-2021-26272 | 12.1.0.2, 12.2.0.1, 19c | DISCOVER:  Which version of Oracle Application Express (APEX) is installed?  SQL #1 {  SELECT SCHEMA "APEX\_SCHEMA", VERSION  FROM dba\_registry  WHERE comp\_id = 'APEX';  }  If SQL #1 Returns Rows – APEX schema installed and version details will be displayed  Where APEX is installed, check which access methods are being used, and which HTTP server is configured?  Noting that:  - APEX requires a web server with PL/SQL Gateway / mod\_plsql support to access XML DB via a browser or desktop tool.  - It can be installed with Oracle REST Data Services (ORDS), Embedded PL/SQL Gateway (EPG) Configuration or Oracle HTTP Server (OHS).  - As Oracle REST Data Services (ORDS), formerly known as the APEX Listener, allows APEX applications to be deployed without the use of Oracle HTTP Server (OHS) and mod\_plsql or the Embedded PL/SQL Gateway then this vulnerability does not apply to this configuration.  - Where Embedded PL/SQL Gateway (EPG) or HTTP server is used for APEX, it utilises the XML DB Protocol Server, to support standard Internet protocols. The XDB protocol server is not required for standard database functionality. As this vulnerability is specific to HTTP protocol, check the http port configuration -  SQL #2 {  SELECT DBMS\_XDB.gethttpport  FROM DUAL;  }  If SQL #2 Returns Port Number, APEX HTTP is configured, whereas Port 0 is disabled.  Note: Please ensure this port has relevant network firewall configurations, controls and restructions in place  Additional checks  A pre-requisite of using XML DB Protocol Server is setting the DISPATCHERS database initialization parameter, to allow XDB protocol registration with the Listener.  The DISPATCHERS parameter is set in the init.ora/spfile.  Check if DISPATCHERS initialization parameter is set in the database. Here is how you could check -  As SYSDBA  SQL #3 {  show parameter dispatchers  }  If SQL #3 Returns a service string then dispatch processes are enabled for XDB  Where Oracle HTTP server is used, it utilises APEX\_PUBLIC\_USER account, which is used for any Database Access Descriptors (DADs), thus check status of account to determine utilisation.  SQL #4 {  select account\_status  from dba\_users  where username = 'APEX\_PUBLIC\_USER';  }  SQL #4 will return account status. If 'OPEN' status this account is active, whereas any other status is inactive | DISCOVER:  Which version of Oracle Application Express (APEX) is installed?  SQL #1 {  SELECT SCHEMA "APEX\_SCHEMA", VERSION  FROM dba\_registry  WHERE comp\_id = 'APEX';  }  If SQL #1 Returns Rows – APEX schema installed and version details will be displayed  Where APEX is installed, check which access methods are being used, and which HTTP server is configured?  Noting that:  - APEX requires a web server with PL/SQL Gateway / mod\_plsql support to access XML DB via a browser or desktop tool.  - It can be installed with Oracle REST Data Services (ORDS), Embedded PL/SQL Gateway (EPG) Configuration or Oracle HTTP Server (OHS).  - As Oracle REST Data Services (ORDS), formerly known as the APEX Listener, allows APEX applications to be deployed without the use of Oracle HTTP Server (OHS) and mod\_plsql or the Embedded PL/SQL Gateway then this vulnerability does not apply to this configuration.  - Where Embedded PL/SQL Gateway (EPG) or HTTP server is used for APEX, it utilises the XML DB Protocol Server, to support standard Internet protocols. The XDB protocol server is not required for standard database functionality. As this vulnerability is specific to HTTP protocol, check the http port configuration -  SQL #2 {  SELECT DBMS\_XDB.gethttpport  FROM DUAL;  }  If SQL #2 Returns Port Number, APEX HTTP is configured, whereas Port 0 is disabled.  Note: Please ensure this port has relevant network firewall configurations, controls and restructions in place  Additional checks  A pre-requisite of using XML DB Protocol Server is setting the DISPATCHERS database initialization parameter, to allow XDB protocol registration with the Listener.  The DISPATCHERS parameter is set in the init.ora/spfile.  Check if DISPATCHERS initialization parameter is set in the database. Here is how you could check -  As SYSDBA  SQL #3 {  show parameter dispatchers  }  If SQL #3 Returns a service string then dispatch processes are enabled for XDB  Where Oracle HTTP server is used, it utilises APEX\_PUBLIC\_USER account, which is used for any Database Access Descriptors (DADs), thus check status of account to determine utilisation.  SQL #4 {  select account\_status  from dba\_users  where username = 'APEX\_PUBLIC\_USER';  }  SQL #4 will return account status. If 'OPEN' status this account is active, whereas any other status is inactive |

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 |
|  |