WHITEPAPER

SPINNAKER SUPPORT

# CVE-2021-2035 and RDBMS Scheduler Security

RDBMS Scheduler Vulnerability
Layered Examination

SPINNAKER SUPPORT

# Table of Contents

# Vulnerability Overview

CVE-2021-2035 is an RDBMS scheduler vulnerability that allows a low privileged user with specific privilege to takeover the RDBMS scheduler.

The following is the full CVE description of the vulnerability.

*Vulnerability in the RDBMS Scheduler component of Oracle Database Server. Supported versions that are affected are 12.1.0.2, 12.2.0.1, 18c and 19c. Easily exploitable vulnerability allows low privileged attacker having Export Full Database privilege with network access via Oracle Net to compromise RDBMS Scheduler. Successful attacks of this vulnerability can result in takeover of RDBMS Scheduler.*

**Component containing the vulnerability.**

The RDBMS scheduler is an oracle database component that can be used to automate not only the simple maintenance tasks, but also the complex business logic. Traditionally, only PL/SQL could be executed in the Scheduler. In later versions, operating system scripts were added to it, and new functionality allows jobs to be run on remote systems and cross platform as well.

### ARE UNSUPPORTED VERSIONS SUSCEPTIBLE TO THIS VULNERABILITY?

Unsupported versions from 10g/11g could also be affected as the RDBMS Scheduler component is part of these versions, and was introduced in 10g R1. Prior to release scheduling was performed via DBMS_JOBS thus this vulnerability is not specific to 8i/9i versions.

### KEY STATEMENTS IN VULNERABILITY DESCRIPTION

The vulnerability details state that:

- The affected supported versions of this RDBMS Scheduler vulnerability are 12c through 19c.
- The vulnerability is not remotely exploitable without authentication. Thus attackers with remote network access to the component cannot exploit the vulnerability without credentials.
- The privilege required to exploit the vulnerability is “Export Full Database”.
- There is no user interaction required to compromise this vulnerability.
- The vulnerability is specific to Oracle software and is not a 3rd Party component vulnerability.
- The vulnerability is specific to Oracle Net protocol.

# Vulnerability Investigation

To assess the impact of this vulnerability to your organisation, the following should be considered, based on the key statements of the vulnerability assessment:

- How and whom can utilize this vulnerability to attack this component of your Oracle database
- Is the attack surface of the vulnerability a threat from both external and internal threat actors
- Who has the privileges required to perform an attack this vulnerability
- How is the attack component being used /monitored in your organization and by whom
- Can the attack component be locked down

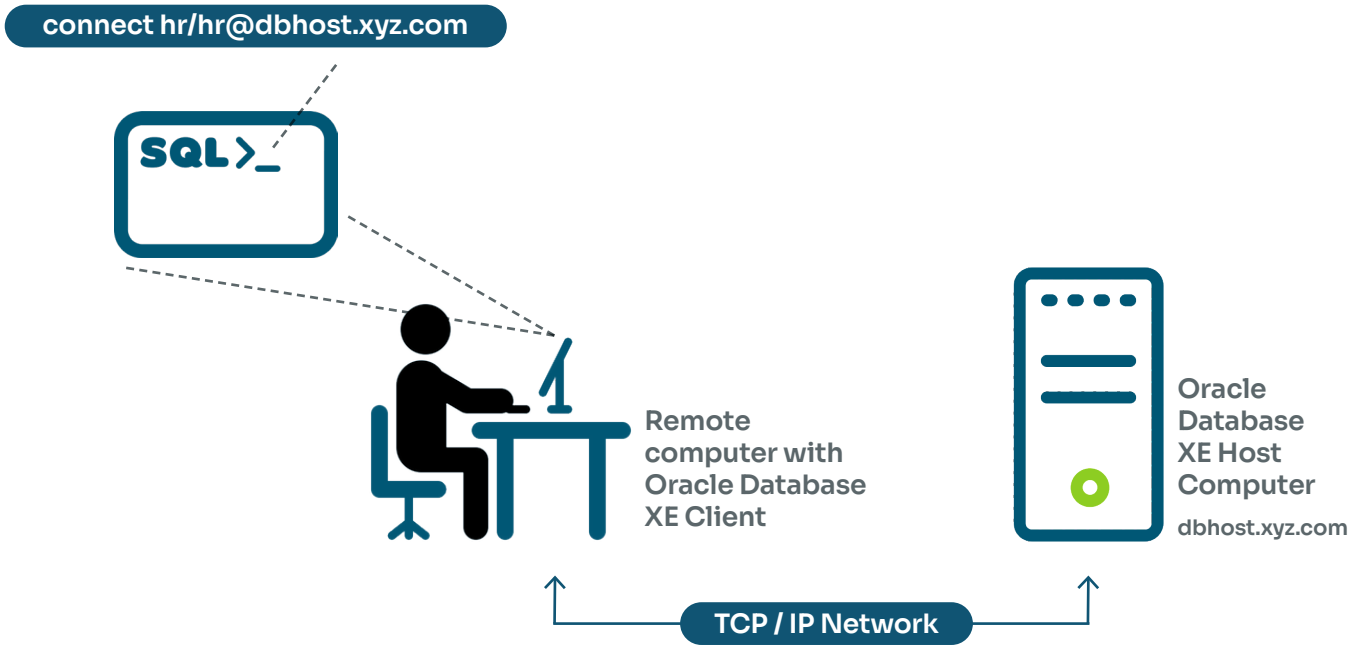
### ATTACK SURFACE ANALYSIS

**How and whom can utilize this vulnerability to attack this component of your Oracle database?**

As per all vulnerabilities, the attacker has to have network access to the component, thus internal systems will have network/perimeter security defense to protect against the attack.

The vulnerability can be attacked using the SQLNet protocol, so an attacker making a database connection remotely over TCP/IP within the network.

Thus best practice security on this protocol such as Valid Node Checking should be considered to reduce the attack surface and who can connect to the database and manipulate the scheduler, e.g. which clients can connect to the database even within a company network.



The attacker requires a specific privilege to be able to takeover the scheduler, thus this threat is limited to database users with that privilege.

**Is the attack surface of the vulnerability a threat from both external and internal threat actors?**

Only those with network access can exploit the vulnerability, and this vulnerability is not exploitable remotely without credentials, e.g. bypass of password/authentication procedures, thus even an attacker who has gained network access should not be able to exploit the vulnerability easily. Thus there is an extremely low risk of external threat actors that originate from sources outside of the organization and its network of partners exploiting this vulnerability.

It is vulnerable to internal threats, i.e. those originating from within the organization, but even those with details of the database connection details will need credentials and of that of an account with the right privileges to make a successful attack. Internal threat actors encompass full-time employees, independent contractors, interns, and other staff. Insiders are trusted and privileged (some more than others).

**Who has the privileges required to perform an attack this vulnerability?**

The Package and/or Privilege Required to exploit this vulnerability is the EXP_FULL_DATABASE privilege, which is an export role with many system privileges.

This role privilege is specific to the original Export utility ( exp ) and not Datapump, which utilizes the “DATAPUMP_EXP_FULL_DATABASE” system privilege.

This privilege should only be granted to authorized users. Normally database administrators should perform export and import operations. Therefore during database privilege analysis assessments, we should find that these grants would only be given to DBA users.

To analyse what System Privileges the EXP_FULL_DATABASE role has the following query can be run:

```
set pagesize 100
col grantee format a20
col privilege format a40

select GRANTEE, PRIVILEGE,ADMIN_OPTION
from sys.dba_sys_privs
where grantee = 'EXP_FULL_DATABASE'
order by grantee;
```

**SAMPLE OUTPUT SHOULD BE AS FOLLOWS:**

| GRANTEE | PRIVILEGE | ADM |
|---|---|---|
| EXP_FULL_DATABASE | READ ANY FILE GROUP | NO |
| EXP_FULL_DATABASE | RESUMABLE | NO |
| EXP_FULL_DATABASE | EXECUTE ANY PROCEDURE | NO |
| EXP_FULL_DATABASE | EXECUTE ANY TYPE | NO |
| EXP_FULL_DATABASE | SELECT ANY TABLE | NO |
| EXP_FULL_DATABASE | ADMINISTER SQL MANAGEMENT OBJECT | NO |
| EXP_FULL_DATABASE | ADMINISTER RESOURCE MANAGER | NO |
| EXP_FULL_DATABASE | BACKUP ANY TABLE | NO |
| EXP_FULL_DATABASE | CREATE SESSION | NO |
| EXP_FULL_DATABASE | SELECT ANY SEQUENCE | NO |
| EXP_FULL_DATABASE | CREATE TABLE | NO |

It is important to ensure the WITH ADMIN OPTION is set to NO for all privileges in this role, as it allows the grantee to grant a role to another database account. Where Admin option has been granted, it is best practice to revoke the privilege and re-grant it without Admin option.

The following STIG benchmark queries can be run as a hierarchical query technique to obtain a list Oracle users ( or roles ) which have EXP_FULL_DATABASE system privilege.

```
set linesize 200
col granted_critic_role format a30

SELECT
DISTINCT A.GRANTEE,
A.GRANTED_ROLE,
'EXP_FULL_DATABASE' GRANTED_CRITIC_ROLE
FROM
(
SELECT
DISTINCT LEVEL LEVEL_DEEP,
GRANTEE,
GRANTED_ROLE
FROM
DBA_ROLE_PRIVS
START WITH GRANTED_ROLE = 'EXP_FULL_DATABASE'
CONNECT BY PRIOR GRANTEE = GRANTED_ROLE
) A,
DBA_USERS B
WHERE
A.GRANTEE = B.USERNAME
AND B.USERNAME NOT IN(
'SYSTEM',
'SYS'
)
AND B.ACCOUNT_STATUS = 'OPEN';
```

**SAMPLE OUTPUT SHOULD BE AS FOLLOWS:**

| GRANTEE | GRANTED_ROLE | GRANTED_CRITIC_ROLE |
|---|---|---|
| TEST_USER | DBA | EXP_FULL_DATABASE |

The only database users outside of SYS and SYSTEM users who should have the EXP_FULL_DATABASE role privilege as part of a hierarchical role to role privilege assignment are DBA role accounts, thus anything other than this is a benchmark finding.

In order to list users with EXP_FULL_DATABASE in the multitenant architecture, we use the below query.

```
SELECT
DISTINCT A.GRANTEE,
A.GRANTED_ROLE,
B.COMMON,
C.NAME,
'EXP_FULL_DATABASE' GRANTED_CRITIC_ROLE
FROM
(
SELECT
DISTINCT LEVEL LEVEL_DEEP,
GRANTEE,
GRANTED_ROLE,
CON_ID
FROM
CDB_ROLE_PRIVS
START WITH GRANTED_ROLE = 'EXP_FULL_DATABASE'
CONNECT BY PRIOR GRANTEE = GRANTED_ROLE
AND PRIOR CON_ID = CON_ID
) A,
CDB_USERS B,
V$CONTAINERS C
WHERE
A.GRANTEE = B.USERNAME
AND B.USERNAME NOT IN(
'SYSTEM',
'SYS'
)
AND B.ACCOUNT_STATUS = 'OPEN'
AND A.CON_ID = C.CON_ID
AND B.CON_ID = C.CON_ID ;
```

**How is the attack component being used/monitored in your organization and by whom?**

Oracle RDBMS scheduler is a server-based scheduler, thus everything is done in the context of the database server.

The Scheduler enables database administrators and application developers to control when and where various tasks take place in the database environment.

It is generally always being used, but to check the usage of the scheduler run the following query.

```
column "Feature" format a30
column "First Time Used" format a20
column "Being Used" format a10

select name "Feature", detected_usages "Usages", first_usage_date "First Time Used", currently_used
"Being Used" from dba_feature_usage_statistics where name = 'Job Scheduler';
```

| FEATURE | USAGES | FIRST TIME USED | BEING USED |
|---|---|---|---|
| JOB SCHEDULER | 12 | 12-MAR-21 | TRUE |

Scheduler tasks take place in the database environment via jobs, which are a combination of a schedule and a program, along with any additional arguments required by the program.

Who and what can be scheduled with access to the RDBMS scheduler?

Job schedules and where they are scheduled is determined by system privileges, thus who can schedule is determined by which users/roles are granted these system privileges.

The following system privileges are associated with the scheduler.

| SYSTEM PRIVILEGE | PURPOSE |
|---|---|
| CREATE JOB | Enables the user to create jobs, schedules and programs in their own schema. The user can always alter and drop jobs, schedules and programs which they own, even when they do not have the CREATE JOB privilege. |
| CREATE ANY JOB | Enables the user to create jobs, schedules, and programs in any schema. This effectively gives the grantee the ability to run code as any user so it must be issued with care. |
| EXECUTE ANY PROGRAM | Enables jobs the ability to use programs from any schema. |
| EXECUTE ANY CLASS | Enables jobs to run under any job class. |
| MANAGE SCHEDULER | Enables the user to create, alter and drop job classes, windows and window groups. It also enables the user to purge scheduler logs and modify scheduler attributes. |

The **SCHEDULER_ADMIN** role has all the above privileges granted to it with the ADMIN option, making it extremely powerful. This role is granted in turn to the DBA with the ADMIN option.

The **SCHEDULER_ADMIN** role gives a user the ability to control every aspect of the scheduler, as well as generating jobs to run as any other user. For this reason you should avoid granting it to anyone other than trusted DBAs.

For the majority of users, the CREATE JOB privilege will be sufficient.

To query what programs have enabled schedules run the following query:

```
SQL>set linesize 300
SQL>select owner, program_name, program_type , enabled from dba_scheduler_programs;
```

**SAMPLE OUTPUT SHOULD BE AS FOLLOWS:**

| OWNER | PROGRAM_NAME | PROGRAM_TYPE | ENABL |
|---|---|---|---|
| SYS | AUTO_SQL_TUNING_PROG | PLSQL_BLOCK | TRUE |
| SYS | AUTO_SPACE_ADVISOR_PROG | STORED_PROCEDURE | TRUE |
| SYS | GATHER_STATS_PROG | STORED_PROCEDURE | TRUE |
| Etc, etc. | | | |

It is important to assess what type of jobs are being scheduled, as this can determine the attack surface.

The different types of job action types, (i.e. program types in DBA_SCHEDULER_PROGRAMS) that can be scheduled are: ('PLSQL_BLOCK', 'STORED_PROCEDURE', 'EXECUTABLE', 'CHAIN', 'EXTERNAL_SCRIPT', 'SQL_SCRIPT', and 'BACKUP_SCRIPT').

"EXTERNAL_SCRIPT" refers to shell scripts.
"EXECUTABLE" refers to OS executable files.

EXTERNAL_SCRIPT, SQL_SCRIPT and BACKUP_SCRIPT are new job types introduced in 12c.

To monitor what jobs have been run in a database, query the DBA_SCHEDULER_JOB_RUN_DETAILS view which displays log run details for all Scheduler jobs in the database providing a history of the job runs, including the status of the run and error messages associated with failed runs.

Can the attack component be locked down?

Generally no, as the scheduler offers key functionality and is preferred in many situations than crontab, as dbms_scheduler (formerly dbms_job) has the ability to detect if the system was down when a job was supposed to fire and re-starts it.

Also, dbms_scheduler allows you to chain many jobs together, branching based on the return codes from previous jobs. However where concerned about an attack, the scheduler can be disabled for investigation.

## IMPLICATION OF A SUCCESSFUL ATTACK

To ensure best strategy risk mitigation, it is good practice to consider the implications of a successful attack (i.e. the risk) for this vulnerability.

As the threat is "*Successful attacks of this vulnerability can result in takeover of RDBMS Scheduler*" then mitigations should be relevant to restrict the scheduler.

The DBMS_SCHEDULER includes the following features which should be risk assessed:

- Logging of job runs (job history)
- Simple but powerful scheduling syntax (similar to but more powerful than cron syntax)
- **Running of jobs outside of the database on the operating system**
- Resource management between different classes of jobs
- Use of job arguments including passing of objects into stored procedures
- Privilege-based security model for jobs
- Naming of jobs and comments in jobs
- Stored, reusable schedules

Features in releases after 10g Release 1 include :

- Dependencies between job units (10gR2 and up)
- Scheduling based on financial calendars and fiscal quarters (10gR2 and up)
- Event based jobs which run when an event is received (10gR2 and up)
- **Running of jobs on remote machines (11gR1 and up)**
- E-mail notifications on job events of interest (10gR2 and up)
- Starting a job based on arrival of a file (10gR2 and up)

*Does the Scheduler contain any other sensitive information that could affect outside of the database?*

Yes, the scheduler can contain operating system user credentials.

### CREDENTIALS

In Oracle 12c the credential related sub-programs of the DBMS_SCHEDULER package have been deprecated and replaced by the new DBMS_CREDENTIAL package. From a usage perspective it feels similar.

e.g. 10g/11g creating credentials.

```
BEGIN
-- Basic credential.
DBMS_SCHEDULER.create_credential(
credential_name => 'SCHEDULER_JOB_CREDENTIAL',
username => 'ora_user',
password => 'password');
/
```

12c onwards.

```
BEGIN
-- Basic credential.
DBMS_CREDENTIAL.create_credential(
credential_name => 'SCHEDULER_JOB_CREDENTIAL',
username => 'ora12c_user',
password => 'password');
/
```

Credentials are database objects that hold a username/password pair for authenticating local and remote external jobs.

They are created using the CREATE_CREDENTIAL procedure in the DBMS_SCHEDULER/ DBMS_CREDENTIAL package depending on Oracle version. The procedure also allows you to specify the Windows domain for remote external jobs executed against a Windows server. Credentials are owned by SYS.

Information about credentials is displayed using the [DBA|ALL|USER]_SCHEDULER_ CREDENTIALS views although from 12c this has been deprecated and replaced by [DBA|ALL|USER]_CREDENTIALS views.

**e.g.**

```
COLUMN credential_name FORMAT A25
COLUMN username FORMAT A20
COLUMN windows_domain FORMAT A20

SELECT credential_name, username, windows_domain
FROM dba_scheduler_credentials
ORDER BY credential_name;
```

Or for higher Oracle versions:

```
COLUMN credential_name FORMAT A25
COLUMN username FORMAT A20
COLUMN windows_domain FORMAT A20

SELECT credential_name, username, windows_domain, enabled
FROM dba_credentials
ORDER BY credential_name;
```

*Can other databases in the network be attacked?*

Yes, but only where an Oracle Scheduler Agent on a machine where remote external jobs are required to be run has been installed and configured.

## REMOTE EXTERNAL JOBS

Oracle 10g introduced the concept of external jobs. Oracle 11g takes this one step further by allowing the database to schedule external jobs which run on a remote server.

The remote server doesn't have to have an Oracle client or database installation, but it must have an Oracle Scheduler Agent installation. This agent is responsible for executing the jobs and communicating with the database server that initiated the job.

For a user to create local or remote external jobs, they must be granted the CREATE JOB and CREATE EXTERNAL JOB privileges, thus privilege analysis should be on performed on users roles who have the “CREATE EXTERNAL JOB” system privilege.

To verify database configured for remote external jobs, query view DBA_SCHEDULER_REMOTE_DATABASES.

DBA_SCHEDULER_REMOTE_DATABASES displays information about the remote databases accessible to the current user that have been registered as sources and destinations for remote database jobs.

Check database links in view for connection information.

SQL> desc DBA_SCHEDULER_REMOTE_DATABASES

| NAME | NULL? | TYPE |
|---|---|---|
| DATABASE_NAME | NOT NULL | VARCHAR2(512) |
| REGISTERED_AS | | VARCHAR2(11) |
| DATABASE_LINK | NOT NULL | VARCHAR2(512) |

# Vulnerability Defense in Depth Recommendations

Although the CVE cannot be exploited remotely, the following mitigations should be considered to ensure that best practice security is configured for the Data and User layers in the defense in depth model. These mitigations not only should focus on the specific CVE (vulnerability) but also the common weaknesses (CWEs) around the threat.

## SQLNET PROTOCOL RECOMMENDATIONS

As the protocol used to communicate with the component that contains the vulnerability is SQLNet, the following mitigations should be considered:

- Define appropriate firewall rules (hosts,ports,services) to prevent remote access.
- Define invited_nodes in sqlnet.ora to deny access from unknown hosts.
- Consider IP based access control lists (ACLs).

### VALID NODE CHECKING

Valid node checking is a security feature available since 9i that protects DBMS instances from malevolent or errant Oracle Net connections over TCP/IP, without the need for a firewall or IP address filtering at the operating system level.

It is used to block clients based on IP address.

Note, TCP Valid node checking should not be confused with Valid Node Checking Registration (VNCR) which is listener based security.

TCP Valid node checking lines are added to the $ORACLE_HOME/network/admin/sqlnet.ora file.

The parameter TCP.VALIDNODE_CHECKING turns this feature on or off. If enabled, then the incoming connections are allowed only if they originate from a node that conforms to the list specified by TCP.INVITED_NODES or TCP.EXCLUDED_NODES parameters.

The parameter TCP.INVITED_NODES list all clients that are allowed access to the database.

The parameter TCP.EXCLUDED_NODES specifies which clients are not allowed to access the database.

**e.g.**

tcp.validnode_checking = yes

tcp.invited_nodes=(localhost,192.168.10.101,192.168.10.102)

tcp.excluded_nodes=( x.x.x.x | name, x.x.x.x | name)

Note* Include either the invited_nodes or excluded_nodes, but do not use both.

**Here are some rules for entering invited/excluded nodes:**

You cannot use wildcards in your specifications.

You must put all invited nodes in one line; likewise for excluded nodes.

TCP.INVITED_NODES list takes precedence over the TCP.EXCLUDED_NODES.

You should always enter localhost as an invited node.

All host names in the TCP.INVITED_NODES or TCP.EXCLUDED_NODES must be able to be resolved or the listener will not start. In case a host name is not resolved, the start of the listener fails with the error message: TNS-00584: Valid node checking configuration error.

Any change of the values in TCP.VALIDNODE_CHECKING, TCP.INVITED_NODES or TCP.EXCLUDED_NODES requires the listener to be stopped and started.

Where TCP Valid node checking is enabled, attempts to make a connect from either an uninvited or excluded node will be rejected with TNS-12547: TNS:lost contact error.

## OBJECT PRIVILEGE SECURITY FRAMEWORK BENCHMARKS

To ensure RDBMS scheduler security there are recommended benchmarks that should be checked. These benchmarks are relevant to all security frameworks but the queries in these examples are specific to CIS/SANS.

e.g. Ensure 'EXECUTE' is revoked from 'PUBLIC' on "Job Scheduler" Packages

SQL> set linesize 250
COLUMN table_name FORMAT A25
COLUMN privilege FORMAT A25
COLUMN grantee FORMAT A20
COLUMN “Grant Option” FORMAT A12
SELECT TABLE_NAME, PRIVILEGE, GRANTEE, GRANTABLE “Grant Option” FROM DBA_TAB_PRIVS WHERE GRANTEE='PUBLIC' AND PRIVILEGE='EXECUTE' AND TABLE_NAME IN ('DBMS_SCHEDULER','DBMS_JOB');

**SAMPLE OUTPUT:**

| TABLE_NAME | PRIVILEGE | GRANTEE | GRANT_OPTION |
|---|---|---|---|
| DBMS_JOB | EXECUTE | PUBLIC | NO |
| DBMS_SCHEDULER | EXECUTE | PUBLIC | NO |

Where execute privileges of these job scheduler packages has been granted publicly this is a finding. Ensure Grant Option (allowing grantee to grant the same level of access to other users or roles) is disabled.

Ensure to revoke and grant execute permissions to specific users or roles.
e.g.
SQL>revoke execute on DBMS_SCHEDULER from PUBLIC;
SQL>grant execute on DBMS_SCHEDULER to <dba_user/role>;

Ensure 'ALL' Is Revoked on 'Sensitive' Scheduler Table

SELECT GRANTEE, PRIVILEGE, TABLE_NAME FROM DBA_TAB_PRIVS WHERE TABLE_NAME = 'SCHEDULER$_CREDENTIAL';

Where any rows are returned outside of normal DBA access/privileges revoke privileges.

SQL>REVOKE ALL ON SYS. SCHEDULER$_CREDENTIAL FROM <grantee>;

Query and remove any unverified DBMS Scheduler Credentials.

SELECT credential_name, username, windows_domain, enabled FROM dba_credentials

ORDER BY credential_name; Unverified Credentials Can Be Removed via:

EXEC DBMS_CREDENTIAL.drop_credential('MY_USER_CREDENTIAL');

## PRIVILEGE ANALYSIS

Although the vulnerability is specific to the EXP_FULL_DATABASE system privilege, a full privilege analysis of system privileges and roles associated with the RDBMS scheduler component is recommended, ensuring the principle of least privilege.

For RDBMS scheduler security the safest option is not to allow any users except the DBAs to create jobs. Not only does this limit any potential security holes, but it also prevents users from scheduling resource intensive and inefficient jobs without understanding the consequences.

Privilege analysis should be performed on the following System Privileges and Roles.

*System Privileges*
CREATE JOB
CREATE ANY_JOB
CREATE EXTERNAL_JOB
EXECUTE ANY PROGRAM
EXECUTE ANY CLASS
MANAGE SCHEDULER

*Roles*
SCHEDULER_ADMIN
EXP_FULL_DATABASE

### PRIVILEGE ANALYSIS ON ASSIGNED PRIVILEGES.

To check which users/roles have been assigned the System privileges that could help an attacker exploit the RDBMS scheduler query the DBA_SYS_PRIVS view.

**e.g.**

```
SELECT GRANTEE, PRIVILEGE, ADMIN_OPTION FROM DBA_SYS_PRIVS
WHERE PRIVILEGE IN ('CREATE JOB','CREATE ANY JOB','CREATE EXTERNAL JOB','EXECUTE
ANY PROGRAM','EXECUTE ANY CLASS','MANAGE SCHEDULER')
AND GRANTEE NOT IN ('DBA','SYS','SYSTEM') order by grantee, privilege;
```

**SAMPLE OUTPUT:**

| GRANTEE | PRIVILEGE | ADM |
|---|---|---|
| APEX_040000 | CREATE JOB | YES |
| CTXSYS | CREATE JOB | NO |
| CTXSYS | MANAGE SCHEDULER | NO |
| OEM_ADVISOR | CREATE JOB | NO |
| OEM_MONITOR | CREATE JOB | NO |
| SCHEDULER_ADMIN | CREATE ANY JOB | YES |
| SCHEDULER_ADMIN | CREATE EXTERNAL JOB | YES |
| SCHEDULER_ADMIN | CREATE JOB | YES |
| SCHEDULER_ADMIN | EXECUTE ANY CLASS | YES |
| SCHEDULER_ADMIN | EXECUTE ANY PROGRAM | YES |
| SCHEDULER_ADMIN | MANAGE SCHEDULER | YES |

i.e. Outside of administrative users and DBA role, only SCHEDULER_ADMIN role should be able to manage the scheduler, thus CTXSYS would be a finding/consideration.

To check which user/roles (privilege inheritance from other roles) have been assigned the roles, that could help an attacker exploit the RDBMS scheduler query the DBA_ROLE_PRIVS view.

```
SELECT GRANTEE, GRANTED_ROLE, ADMIN_OPTION FROM DBA_ROLE_PRIVS
WHERE GRANTED_ROLE IN ('EXP_FULL_DATABASE','SCHEDULER_ADMIN')
AND GRANTEE NOT IN ('DBA','SYS','SYSTEM') order by grantee, granted_role;
```

**SAMPLE OUTPUT:**

| GRANTEE | GRANTED_ROLE | ADM |
|---|---|---|
| DATAPUMP_EXP_FULL_DATABASE | EXP_FULL_DATABASE | NO |
| DATAPUMP_EXP_FULL_DATABASE | EXP_FULL_DATABASE | NO |
| OEM_ADIVSOR | EXP_FULL_DATABASE | YES |

i.e. Outside of administrative users and DBA role, and Datapump roles, OEM_ADVISOR role having this system privilege would be a finding/consideration.

## MAINTAIN CENTRAL CONTROL OF SCHEDULER PRIVILEGES.

In the above queries, the Admin option has been added to the privilege analysis. This is because best security practice recommends restricting the privilege of assigning privileges to authorized personnel. Restricting privilege-granting functions to authorized accounts can help decrease mismanagement of privileges and wrongful assignments to unauthorized accounts.

It is recommended to restrict any privilege that has been granted with admin option. The 'with admin' option serves to relinquish central security control.

The **WITH ADMIN OPTION** allows the grantee to grant a system privilege or role to another database account. Best security practice restricts the privilege of assigning privileges to authorized personnel, such as DBAs. Where privileges/roles are found with Admin option outside of authorized personnel, but the privilege assignment is valid, it is recommended to revoke the privilege and re-grant.

e.g revoke EXP_FULL_DATABASE from OEM_ADVISOR;
grant EXP_FULL_DATABASE to OEM_ADVISOR;

## PRIVILEGE ANALYSIS FEATURE.

Privilege Analysis feature in Oracle Database 12c provides a way to maintain least privilege by reporting on the permissions that a user is actually using and not just the permissions a user has been granted.

Until 19c this functionality was part of the Oracle Database Vault option, however from 19c onward this feature is part of Enterprise Edition, so you no longer need the Database Vault option.

**https://docs.oracle.com/database/121/DVADM/priv_analysis.htm#DVADM591**

Oracle 12c introduced the DBMS_PRIVILEGE_CAPTURE package, which allows you to track the privileges being used, making it much simpler to perform privilege analysis, which in turn allows you to revoke unnecessary privileges and attain a least privilege state.

This allows an extension of audit reports of users which can show privileged access and what roles are granted via querying the DBA_ROLE_PRIVS and DBA_TAB_PRIVS dictionary tables, as these reports show only what permissions are available.

Log into sqlplus /nolog as sysdba and utilise the DBMS_PRIVILEGE_CAPTURE package.

A user with CAPTURE_ADMIN role or SYSDBA privilege can perform privilege analysis.

1. Create a privilege analysis policy. (CREATE_CAPTURE)
2. Enable it. (ENABLE_CAPTURE)
3. Wait for the required analysis period.
4. Disable the privilege analysis policy. (DISABLE_CAPTURE)
5. Analyze the results. (GENERATE_RESULT and query dictionary views)
6. Drop the policy if it, and the recorded data, is no longer needed. (DROP_CAPTURE)

The privilege analysis tool has four different types of analysis it can do. For this vulnerability the G_DATABASE and G_ROLE analyzers would be relevant.

- G_DATABASE: Analyzes all privileges used (except by the SYS user)
- G_ROLE: Analyzes privileges related to specified roles
- G_CONTEXT: Analyzes privileges based on a condition parameter set using the SYS_CONTEXT function
- G_ROLE_AND_CONTEXT: Analyzes privileges for specific roles based on a condition set using the SYS_CONTEXT function

i.e. To perform privilege analysis on EXP_FULL_DATABASE and SCHEDULER_ADMIN roles, create a relevant policy.

```
-- One or more roles (type = G_ROLE).
BEGIN
DBMS_PRIVILEGE_CAPTURE.create_capture(
 name => 'SCHEDULER_ROLE_POL',
 type => DBMS_PRIVILEGE_CAPTURE.g_role,
 roles => role_name_list('SCHEDULER_ADMIN', ' EXP_FULL_DATABASE')
);
END;
/
```

Enable Privilege Analysis Capture example for the policy.

```
BEGIN
DBMS_PRIVILEGE_CAPTURE.enable_capture(
name => 'SCHEDULER_ROLE_POL',
);
END;
/
```

*Generate Privilege Analysis Capture for the policy.*

```
BEGIN
DBMS_PRIVILEGE_CAPTURE.generate_capture(
name => 'SCHEDULER_ROLE_POL',
);
END;
/
```

Check capture results from the following views.

**Views Description**

- DBA_PRIV_CAPTURES Lists information about existing privilege analysis policies
- DBA_[UN]USED_PRIVS Lists the privileges that have [not] been used for reported privilege analysis policies
- DBA_[UN]USED_OBJPRIVS Lists the object privileges that have been [not] used for reported privilege analysis policies. It does not include the object grant paths.
- DBA_[UN]USED_OBJPRIVS_PATH Lists the object privileges that have [not] been used for reported privilege analysis policies. It includes the object privilege grant paths.
- DBA_[UN]USED_SYSPRIVS Lists the system privileges that have [not] been used for reported privilege analysis policies. It does not include the system privilege grant paths.
- DBA_[UN]USED_SYSPRIVS_PATH Lists the system privileges that have [not] been used for reported privilege analysis policies. It includes the system privilege grant paths.
- DBA_USED_PUBPRIVS Lists all the privileges for the PUBLIC role that have been used for reported privilege analysis policies
- DBA_[UN]USED_USERPRIVS Lists the user privileges that have [not] been used for reported privilege analysis policies. It does not include the user privilege grant paths.
- DBA_[UN]USED_USERPRIVS_PATH Lists the user privileges that have [not] been used for reported privilege analysis policies. It includes the user privilege grant paths.

e.g. The following example query could be used after a privilege capture to see how the system privileges for the RDBMS scheduler are being used.

```
COLUMN OS_USER FORMAT A20
COLUMN USERHOST FORMAT A20
COLUMN MODULE FORMAT A20
COLUMN USERNAME FORMAT A20
COLUMN USED_ROLE FORMAT A20
COLUMN SYS_PRIV FORMAT A20
```

```
SELECT OS_USER, USERHOST, MODULE, USERNAME, USED_ROLE, SYS_PRIV, ADMIN_OPTION
FROM dba_used_sysprivs WHERE capture = ' SCHEDULER_ROLE_POL'
AND sys_priv IN IN ('CREATE JOB','CREATE ANY JOB','CREATE EXTERNAL JOB','EXECUTE
ANY PROGRAM','EXECUTE ANY CLASS','MANAGE SCHEDULER')
ORDER BY username, sys_priv;
```

**Create Custom Role and Restrict "SELECT ANY TABLE" Privilege.**

The EXP_FULL_DATABASE role utilized to exploit the RDBMS Scheduler vulnerability is a powerful role with many privileges, including the "SELECT ANY TABLE" privilege.

This privilege allows the ability to perform a SELECT ... FOR UPDATE, thus allowing the grantee the ability lock rows.

**e.g.**

```
GRANT SELECT ON schema1.test_tab1 TO read_only_schema;
```

Issue the following SELECT ... FOR UPDATE query in a session connected to the read-only user. Do not issue commit or rollback after it and keep the session open while you work in a separate session.

```
conn read_only_schema/read_only_schema

SELECT * FROM schema1.test_tab1
FOR UPDATE;
```

While connected as the schema owner, attempt to update one of the rows. You will see it hang, waiting for the readonly session to commit or rollback and thereby release the locks.

```
CONN schema1/schema1

UPDATE test_tab1 SET id = id
WHERE id = 1;
```

Issue a commit in the read-only user session and you will see the update complete in the schema owner session.

The READ object privilege can be utilized to create read-only users that no longer allow the grantee the ability to lock rows in the tables they query, addressing the security flaw with select object privilege.

Introduced in version 12.1.0.2, the "READ" object privilege does not allow the user to lock tables in exclusive mode nor select table for update.

As issue is version 12c and above, perform privilege analysis using DBMS_PRIVILEGE_CAPTURE package.

Note, originally an option as part of Database Vault, but backported to 12c as part of Enterprise Edition license.

Documentation **https://docs.oracle.com/database/121/DVADM/priv_analysis.htm#DVADM1041**

Thus for this vulnerability it is recommended to create a custom version of EXP_FULL_DATABASE role with “SELECT ANY TABLE” system privilege being replaced by “READ ANY TABLE” system privilege.

**CREATE ANY JOB Restriction Due to Privilege exploit.**

The CREATE ANY JOB privilege is a very powerful privilege that enables the user to create jobs, schedules, and programs in any schema and should be highly restricted, as it is vulnerable to a well known privilege escalation exploit.

Privilege escalation with “CREATE ANY JOB” system privilege via job exploit.

```
-- Check the current roles assigned to ANY_USER.

conn sys/password as sysdba

select granted_role from user_role_privs where username = 'ANY_USER';

GRANTED_ROLE
------------------------------

CONNECT
RESOURCE

2 rows selected.

-- Grant excessive privileges to ANY_USER by accident.
grant create any job to any_user;

-- ANY_USER exploits the extra privileges by creating a job as the SYSTEM user to grant the DBA
role to ANY_USER.

conn any_user/any_user

BEGIN
DBMS_SCHEDULER.create_job (
 job_name => 'system.exploit_job',
 job_type => 'PLSQL_BLOCK',
 job_action => 'BEGIN EXECUTE IMMEDIATE ''GRANT DBA TO any_user''; END;',
 start_date => SYSTIMESTAMP,
 enabled => TRUE);
END;
/

-- Check the current roles assigned to ANY_USER. Notice that the DBA role is present.
select granted_role from user_role_privs where username = 'ANY_USER';

GRANTED_ROLE
------------------------------

CONNECT
DBA
RESOURCE
3 rows selected.
```

# REMOTE JOBS SECURITY RECOMMENDATIONS & EXTERNAL CREDENTIAL AUDIT

## REMOTE EXTERNAL JOBS

Where Oracle Scheduler Agents have been installed to allow for remote external job scheduling ensure to firewall the XML DB port used for running remote external jobs.

To determine the port being used, run the following query:

SQL>SELECT DBMS_XDB.gethttpport FROM dual;

e.g. HTTP port that has been set using the GETHTTPPORT function in the DBMS_XDB package.

## GETHTTPPORT

8080

For external databases no longer required for remote job scheduling, drop REMOTE_SCHEDULER_AGENT schema created by script $ORACLE_HOME/rdbms/admin/prvtrsch.plb.

And ensure package DBMS_ISCHED_REMOTE_ACCESS is not executable by PUBLIC.

## AUDIT EXTERNAL CREDENTIALS

Where operating system credentials are being stored in the database for the scheduler it is recommended to do a regular audit of the credentials ensuring password policies are being used, but also ensuring unnecessary credentials are either disabled or removed.

Credentials enabled and disabled using the ENABLE_CREDENTIAL and DISABLE_CREDENTIAL procedures respectively.

```
-- Drop credential 10g/11g.
EXEC DBMS_SCHEDULER.drop_credential('SCHEDULER_JOB_CREDENTIAL');

-- Drop credential 12c.
EXEC DBMS_CREDENTIAL.drop_credential('SCHEDULER_JOB_CREDENTIAL');

-- In 12c credentials can be disabled
EXEC DBMS_CREDENTIAL.disable_credential('SCHEDULER_JOB_CREDENTIAL');
```

## DISABLING RDBMS SCHEDULER OR RESTRICT EXPORT PROCEDURES

It is possible to disable RDBMS Scheduler while investigating attacks, or implementing further security.

When the scheduler came into existence in Oracle 10g, this could be performed via an API call that could be used to temporarily turn the entire scheduler off.

That command was:
SQL> exec dbms_scheduler.set_scheduler_attribute('SCHEDULER_DISABLED','TRUE')
PL/SQL procedure successfully completed.
Where disabled using API call this would be detailed in the scheduler attributes.

Query of Scheduler Attributes.

SQL> select * from **DBA_SCHEDULER_GLOBAL_ATTRIBUTE**;

| ATTRIBUTE_NAME | VALUE |
|---|---|
| MAX_JOB_SLAVE_PROCESSES | |
| LOG_HISTORY | 30 |
| DEFAULT_TIMEZONE | Australia/Perth |
| EMAIL_SERVER | |
| EMAIL_SERVER_ENCRYPTION | NONE |
| EMAIL_SERVER_CREDENTIAL | |
| EMAIL_SENDER | |
| LAST_OBSERVED_EVENT | |
| EVENT_EXPIRY_TIME | |
| FILE_WATCHER_COUNT | 0 |
| CURRENT_OPEN_WINDOW | |
| **SCHEDULER_DISABLED** | **TRUE** |

Since 12c, the supported way to turn off the scheduler is now to set JOB_QUEUE_PROCESSES to zero. You should not use the SCHEDULER_DISABLED attribute.

Note* Starting with Oracle Database release 11.2.0.1, setting JOB_QUEUE_PROCESSES to 0 causes both DBMS_SCHEDULER and DBMS_JOB jobs to not run. Previously, setting JOB_QUEUE_PROCESSES to 0 caused DBMS_JOB jobs to not run, but DBMS_SCHEDULER jobs were unaffected and would still run. As of 11.1, the parameter was changed from basic to non-basic and the default value was changed from 0 to 1000."

To only disable specific job in the scheduler, as a user with MANAGE SCHEDULER privilege run:

SQL>execute dbms_scheduler.disable('<program_name>');

## CUSTOM RESTRICT EXP PROCEDURE

As the attack is specific to the exp utility, locking down the utility via a database access trigger could provide additional security.

Any user that has the CREATE SESSION system privilege, is able to export its own objects. However a procedure to try and prevent users using exp or imp utilities via a database trigger can be used:

Create a staging table of users not permitted to use exp/imp.

```
create table no_exp_imp_users ( block_user varchar2(30) not null );
```

Create an AFTER LOGON DATABASE trigger which will check the staging table and utilize the SYS_CONTEXT function and USERENV provided namespace to get description of the current session.

```
SQL> create or replace trigger prevent_exp_imp
after logon on database
begin
  if ( user in (select block_user from no_exp_imp_users )
  and
  ( upper(sys_context('userenv','module')) in ('exp','imp')
  or
  upper(sys_context('userenv','module')) in ('EXP.EXE','IMP.EXE')
  )
  then
  raise_application_error (-20001, 'You are not allowed to export or import data.');
end if;
end;
/
```

Insert records into staging table no_exp_imp_users with users to prevent running imp/exp.

```
insert into no_exp_imp_user values (‘SCOTT’);
commit;
```

Perform an export test.

```
exp scott/tiger@oracle_db file=xp.dmp log=xp.log

Export: Release 11.2.0.4.0 – Production on Mon Mar 24 16:02:05 2020
Copyright (c) 1982, 2007, Oracle. All rights reserved.

EXP-00056: ORACLE error 604 encountered
ORA-00604: error occurred at recursive SQL level 1
ORA-20001: You are not allowed to export or import data.

ORA-06512: at line 3

Username:
```

## CLEAR UP ORPHAN JOBS WITH NO SESSION ID

Query jobs with no session id and clean up (drop) the jobs.

e.g. Query total running jobs.

SQL> select count(1) from dba_scheduler_running_jobs;

**COUNT (1)**

22

Query running jobs and check for those with no session id details.

SQL> select session_id, slave_process_id, slave_os_process_id, elapsed_time from dba_scheduler_running_jobs;

Check for next schedules.

SQL> select next_run_date, last_start_date from dba_scheduler_jobs where job_name in (select job_name from dba_scheduler_running_jobs) and next_run_date < sysdate;

**1) Stop the scheduler**

SQL> exec dbms_scheduler.set_scheduler_attribute('SCHEDULER_DISABLED','TRUE');
or
SQL>alter system set job_queue_processes=0 scope=both;

**2) For each 'running' job with no session id**

SQL> exec dbms_scheduler.stop_job('SCHEMA.MY_JOB',TRUE);
SQL> exec dbms_scheduler.drop_job('SCHEMA.MY_JOB',TRUE);

**3) Restart the scheduler**

SQL> exec dbms_scheduler.set_scheduler_attribute('SCHEDULER_DISABLED','FALSE');
or
SQL>alter system set job_queue_processes=1 scope=both;

# ENCRYPTING SENSITIVE SCHEDULER CREDENTIAL DATA IN THE DATA DICTIONARY

Oracle Database has two main tables where credential data is stored: SYS.LINK$ and SYS.SCHEDULER$_CREDENTIAL. SYS.LINK$ is used to store sensitive credential data for database links and SYS.SCHEDULER$_CREDENTIAL is used to store sensitive credential data for Oracle Scheduler events.

By default, all the data stored in these tables is obfuscated (unintelligible); however, as a best security practice recommended by Oracle, the data should also be encrypted.

e.g. Although passwords are stored obfuscated, you could extract password with function DBMS_ISCHED.GET_CREDENTIAL_PASSWORD.

```
SQL> SELECT u.name CREDENTIAL_OWNER, O.NAME CREDENTIAL_NAME, C.USERNAME,
DBMS_ISCHED.GET_CREDENTIAL_PASSWORD(O.NAME, u.name) pwd
FROM SYS.SCHEDULER$_CREDENTIAL C, SYS.OBJ$ O, SYS.USER$ U
WHERE U.USER# = O.OWNER#
AND C.OBJ# = O.OBJ#
```

| CREDENTIAL_OWNER | CREDENTIAL_NAME | USERNAME | PWD |
|---|---|---|---|
| TESTUSR | ORACLE_CRED | oracle | os_password |

The functionality to encrypt the data in these tables was introduced in Oracle 18c and is performed using very simple commands.

The sensitive credential data is de-obfuscated and then it is encrypted. The sensitive credential data that is already stored is not encrypted by default, only future password changes made after this feature is enabled are encrypted. The encryption used by this feature is AES256 (Advanced Encryption Standard) algorithm.

Note*This feature makes use of Transparent Data Encryption (TDE), but you do not need to have an Advanced Security Option license to perform this data dictionary encryption.

Checking whether encrypting sensitive credential data is enabled:

The view DICTIONARY_CREDENTIALS_ENCRYPT can say whether the sensitive credential data for SYS.LINK$ and SYS.SCHEDULER$_CREDENTIAL are encrypted:

```
SQL> SELECT ENFORCEMENT FROM DICTIONARY_CREDENTIALS_ENCRYPT;
```

| ENFORCEM |
|---|
| DISABLED |

The follow is an example to implement encryption of SYS.LINK$ and SYS.SCHEDULER$_CREDENTIAL in the data dictionary.

## KEYSTORE LOCATION SETUP & CREATING THE TDE WALLET

A TDE Wallet must exist and be open to use this encryption feature uses. Where it doesn't exist, it must be created beforehand to enable encryption in sensitive credential data; otherwise the following error will be received:

```
SQL> ALTER DATABASE DICTIONARY ENCRYPT CREDENTIALS;

ALTER DATABASE DICTIONARY ENCRYPT CREDENTIALS
*

ERROR at line 1:
ORA-28443: cannot access the TDE wallet
```

Create an operating system directory for the Wallet. The following example is for Unix type operating system.

```
$ mkdir -p /u01/oracle/db/wallet
```

The following configuration for the Wallet needs to be added to SQLNET.ORA:

```
cat $ORACLE_HOME/network/admin/sqlnet.ora

ENCRYPTION_WALLET_LOCATION=
(SOURCE=
(METHOD=FILE)
(METHOD_DATA=
(DIRECTORY= /u01/oracle/db/wallet)))
```

To create and open the TDE Wallet, the SYSKM privilege should be used. SYSKM was introduced in Oracle 12c. SYSKM is short for "system key management", and as the name implies, SYSKM is for key management tasks.

```
sqlplus / as syskm


SQL>ADMINISTER KEY MANAGEMENT CREATE KEYSTORE '/u01/oracle/db/wallet' identified by <password>;

SQL>ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY <password>;

SQL>ADMINISTER KEY MANAGEMENT SET ENCRYPTION KEY IDENTIFIED BY <password> WITH BACKUP;
```

## CHECKING IF THE TDE WALLET IS OPEN:

SQL> select wrl_parameter, status from v$encryption_wallet

| WRL_PARAMETER | STATUS |
|---|---|
| /u01/oracle/db/wallet | OPEN |

Set the TDE Wallet for autologin:

SQL>ADMINISTER KEY MANAGEMENT CREATE AUTO_LOGIN KEYSTORE FROM KEYSTORE '/u01/app/oracle/cdb1/wallet/' identified by <password>;

Enabling Encrypting sensitive credential data in the data dictionary, this command must be executed with SYSKM:

SQL> ALTER DATABASE DICTIONARY ENCRYPT CREDENTIALS;

Now the password columns in the link$ and scheduler$_link passwords should be encrypted. Checking if the feature is enabled:

SQL>SELECT ENFORCEMENT FROM DICTIONARY_CREDENTIALS_ENCRYPT;

| ENFORCEM |
|---|
| ENABLED |

## FURTHER RDBMS SCHEDULER SECURITY RECOMMENDATIONS

In Oracle 19c jobs created using the DBMS_JOB package are converted to DBMS_SCHEDULER jobs.

Due to this conversion it means users require the CREATE JOB privilege to allow them to create jobs using the DBMS_JOB package, where previously they did not.

Attempts to create a DBMS_JOB without this privilege return:

```
ORA-27486: insufficient privileges
ORA-06512: at "SYS.DBMS_ISCHED", line 9387
ORA-06512: at "SYS.DBMS_ISCHED", line 9376
ORA-06512: at "SYS.DBMS_ISCHED", line 175
ORA-06512: at "SYS.DBMS_ISCHED", line 9302
ORA-06512: at "SYS.DBMS_IJOB", line 196
ORA-06512: at "SYS.DBMS_JOB", line 168
ORA-06512: at line 4
27486. 00000 - "insufficient privileges"
```

***Cause:** An attempt was made to perform a scheduler operation without the required privileges.

***Action:** Ask a sufficiently privileged user to perform the requested operation, or grant the required privileges to the proper user(s).

A loophole caused by the refresh group implementation allows users to create a scheduler job with object privilege access only.

**e.g.** If you create a refresh group example, a user is able to create a job without the CREATE JOB privilege.

```
Begin
 dbms_refresh.make(
   name => 'MINUTE_REFRESH',
   list => '',
   next_date => sysdate,
   interval => '/*1:mins*/ sysdate + 1/(60*24)',
   implicit_destroy => false,
   lax => false,
   job => 0,
   rollback_seg => null,
   push_deferred_rpc => true,
   refresh_after_errors => true,
   purge_option => null,
   parallelism => null,
   heap_size => null);
end;
/
```

SELECT job_name, job_action FROM user_scheduler_jobs;

| JOB_NAME | JOB_ACTION |
|---|---|
| MV_RF$J_0_S_199 | dbms_refresh.refresh('"TEST"."MINUTE_REFRESH"'); |

1 row selected.

Whilst this is for a specific purpose, Oracle's security is based on you being able to do whatever you want with objects you already own, so what happens if we try to change the attributes?

```
Begin
 dbms_scheduler.set_attribute (
   name => 'MV_RF$J_0_S_199',
   attribute => 'job_action',
   value => 'begin null; end;'
 );
end;
/
```

SELECT job_name, job_action FROM user_scheduler_jobs;

| JOB_NAME | JOB_ACTION | |
|---|---|---|
| MV_RF$J_0_S_19 | begin null; end; | - PL/SQL block that could include exploits |

1 row selected.

So we can create a job using the DBMS_REFRESH package, then alter it to suit our purpose, giving us the ability to create a job without the need for the CREATE JOB privilege.

To protect against this issue being exploited, revoke EXECUTE on the DBMS_REFRESH package from PUBLIC, as well as DBMS_JOB package.

**e.g.** Query object privileges and revoke any privileges granted to public.

```
set linesize 200
SELECT TABLE_NAME, PRIVILEGE, GRANTEE, GRANTABLE “Grant Option” FROM DBA_TAB_PRIVS
WHERE GRANTEE='PUBLIC' AND PRIVILEGE='EXECUTE' AND TABLE_NAME IN ('DBMS_
REFRESH','DBMS_JOB');
```

**SAMPLE OUTPUT:**

| TABLE_NAME | PRIVILEGE | GRANTEE | Gra |
|---|---|---|---|
| DBMS_JOB | EXECUTE | PUBLIC | NO |
| DBMS_REFRESH | EXECUTE | PUBLIC | NO |

revoke execute on DBMS_REFRESH from PUBLIC;