Provisioning Oracle Wallets and Accessing SSL/TLS-Based Endpoints on Amazon RDS for Oracle

February 2018

Notices

Contents

Introduction 1

Creating and Uploading Custom Oracle Wallets 2

- Creating and Uploading a Wallet with an Amazon S3 Certificate 3
- Uploading a Customized Wallet Bundle 5

Examples of Using Oracle Wallets to Establish SSL/TLS Outbound Connections 6

- Using UTL_HTTP over an SSL/TLS Endpoint 7
- Establishing Database Links between RDS Oracle DB Instances over an SSL/TLS Endpoint 7
- Sending Emails Using UTL_SMTP and Amazon Simple Email Service (Amazon SES) 7
- Downloading a File from Amazon S3 to an RDS Oracle DB Instance 8
- Uploading a File from RDS Oracle DB Instance to Amazon S3 8

Conclusion 9

Appendix 9

- Sample PL/SQL Procedure to Download Artifacts from Amazon S3 9
- Sample PL/SQL Procedure to Send an Email Through Amazon SES 12

Abstract

This paper explains how to extend outbound network access on your Amazon Relational Database Service (Amazon RDS) for Oracle database instances to connect securely to remote, SSL/TLS-based endpoints. SSL/TLS endpoints require one or more valid Certificate Authority (CA) certificates that can be bundled within an Oracle wallet. By uploading Oracle wallets to your Amazon RDS for Oracle DB instances, certain outbound network calls can be made aware of the uploaded Oracle wallets. This enables outbound network traffic to access any SSL/TLS-based endpoint that can be validated using the CA certificate bundle within the Oracle wallets.

Introduction

Amazon Relational Database Service (Amazon RDS) is a managed relational database service that provides you with six familiar database engines to choose from, including Amazon Aurora, MySQL, MariaDB, Oracle, Microsoft SQL Server, and PostgreSQL.[1] You can use your existing database code, applications, and tools with Amazon RDS, and RDS will handle routine database tasks such as provisioning, patching, backup, recovery, failure detection, and repair.

With Amazon RDS, you can use replication to enhance availability and reliability for production workloads. Using the Multi-AZ deployment option, you can run mission-critical workloads with high availability and built-in, automated failover from your primary database to a synchronously replicated secondary database.

Amazon RDS for Oracle provides scalability, performance, monitoring, and backup and restore support. Multi-AZ deployment for Oracle DB instances simplifies creating a highly available architecture. This is because a Multi-AZ deployment contains built-in support for automated failover from your primary database to a synchronously replicated secondary database in a different Availability Zone. Amazon RDS for Oracle provides the latest version of Oracle Database with the latest Patch Set Updates (PSUs). Amazon RDS manages the database upgrade process on your schedule, eliminating manual database upgrade and patching tasks.

Amazon Virtual Private Cloud (Amazon VPC) is a virtual network dedicated to your AWS account.[2] It is logically isolated from other virtual networks in the AWS Cloud. You can launch AWS resources, such as Amazon RDS DB instances or Amazon Elastic Compute Cloud (Amazon EC2) instances, into your VPC.[3] When you create a VPC, you specify IP address ranges, subnets, routing tables, and network gateways to your own data center and to the internet. You can move RDS DB instances that are not already in a VPC into an existing VPC.[4]

Outbound network access is only supported for Oracle DB instances in a VPC.[5] Using outbound network access, you can use PL/SQL code inside the database to initiate connections to servers elsewhere on the network. This lets you use utilities such as UTL_HTTP, UTL_TCP, and UTL_SMTP to connect your DB instance to remote endpoints. For example, you can use UTL_MAIL or

UTL_SMTP to send emails, or UTL_HTTP to communicate with external web servers. By default, an Amazon DNS server provides name resolutions for outbound traffic from the instances in your VPC. Should you choose to resolve private domain names for outbound traffic, you can configure a custom DNS server.[6]

Always take care when enabling outbound networking, as attackers can use it as a vector to remove data from your systems. In addition to other security best practices, keep the following in mind:

- Carefully configure VPC security groups to only allow ingress from and egress to known networks.
- Use in-database network access control lists (ACLs) to allow only trusted users to initiate connections out of the database.
- Always upgrade to the latest release of Amazon RDS for Oracle to ensure you have the latest Oracle PSU and security fixes.

To protect the integrity and content of your data, you should use Transport Layer Security (TLS, also referred to as Secure Sockets Layer or SSL) to provide encryption and server verification. By default, outbound network access supports only external traffic over and to non-TLS/SSL mediums. For TLS/SSL-based traffic, you can use Oracle wallets to store Certificate Authority (CA) certificates, which enable the verification of remote entities. You can make utilities that use outbound network access traffic (such as UTL_HTTP and UTL_SMTP) aware of these wallets. This enables outbound communication from your DB instance to remote endpoints over SSL.

In this paper, we discuss how to create Oracle wallets and copy them to an Amazon RDS for Oracle DB instance using Amazon S3. We also demonstrate how to use a wallet to protect calls made using UTL_HTTP and UTL_SMTP utilities.

Creating and Uploading Custom Oracle Wallets

To enable SSL/TLS connections from PL/SQL, you can upload custom Oracle wallets to your Amazon RDS for Oracle DB instances. These wallets can contain

public and private certificates to access SSL/TLS-based endpoints from your RDS Oracle DB instances.

First, you create an initial Oracle wallet containing an Amazon S3 certificate as a one-time setup. Then you can securely upload any number of wallets to Amazon RDS for Oracle DB instances through Amazon S3.

Creating and Uploading a Wallet with an Amazon S3 Certificate

1. Download the Baltimore CyberTrust Root certificate.[7]
2. Convert the certificate to the x509 PEM format.

```
openssl x509 -inform der -in BaltimoreCyberTrustRoot.crt -
outform pem -out BaltimoreCyberTrustRoot.pem
```

3. Using the `orapki` utility,[8] create a wallet and add the certificate. This exports the wallet to a file named **cwallet.sso**. Alternatively, if you don't specify an auto-login wallet, you can use **ewallet.p12**. In this case, PL/SQL applications must provide a password when opening the wallet.

```
orapki wallet create -wallet . -auto_login_only
orapki wallet add -wallet . -trusted_cert -cert
BaltimoreCyberTrustRoot.pem -auto_login_only
orapki wallet display -wallet .
```

4. Using high-level `aws s3` commands with the AWS Command Line Interface (CLI),[9] create an S3 bucket (or use an existing bucket) and upload the wallet artifact.

```
aws s3 mb s3://<bucket-name>
aws s3 cp cwallet.sso s3://<bucket-name>/
```

5. Generate a presigned URL for the wallet artifact. By default, presigned URLs are valid for an hour. However, you can set the expiration explicitly.[10]

```
aws s3 presign s3://<bucket-name>/cwallet.sso
```

6. Import the procedure provided in the Appendix into your RDS for Oracle DB instance.
7. Using this procedure, download the wallet from the S3 bucket.
    a. Create a directory for this initial wallet. (Be sure to always store each wallet in its own directory.)

```
exec rdsadmin.rdsadmin_util.create_directory('S3_SSL_WALLET');
```

    b. Whitelist outbound traffic on Oracle's ACL (using the 'user' defined earlier).

```
BEGIN
   DBMS_NETWORK_ACL_ADMIN.CREATE_ACL (
    acl          => 's3.xml',
    description  => 'AWS S3 ACL',
    principal    => UPPER('&user'),
    is_grant     => TRUE,
    privilege    => 'connect');
   COMMIT;
END;
/

BEGIN
   DBMS_NETWORK_ACL_ADMIN.ASSIGN_ACL (
    acl          => 's3.xml',
    host         => '*.amazonaws.com');
   COMMIT;
END;
/
```

    c. Using the procedure above, fetch the wallet artifact uploaded earlier to the S3 bucket. Replace the p_s3_url value with the presigned URL generated in step 5 (after stripping it to be HTTP instead of HTTPS). Although access to this S3 wallet artifact is presigned, it **must** be over HTTP.

```
set define #;

BEGIN
  s3_download_presigned_url(
    p_s3_url => '<URL from step 5>',
    p_local_filename => 'cwallet.sso',
    p_local_directory => 'S3_SSL_WALLET'
  );
END;
/
```

8. Set the **S3_SSL_WALLET** path above for **utl_http** transactions.

```
DECLARE
  l_wallet_path all_directories.directory_path%type;
BEGIN
  select directory_path into l_wallet_path from all_directories
where upper(directory_name)='S3_SSL_WALLET';
  utl_http.set_wallet('file:/'  || l_wallet_path);
END;
/
```

At this point, you can use the wallet to access any artifact (not limited to Oracle wallets) from Amazon S3 over SSL/TLS as long as you're pointing to the wallet directory specified above.

Uploading a Customized Wallet Bundle

With the capability we've described in the previous procedure, you can also download customized Oracle wallets (containing customized selections of public or private CA certificates). For example, you can create a new Oracle wallet containing a wallet bundle of your choice, upload it to an S3 bucket, and use one of the previous procedures to securely download this wallet to an Amazon RDS for Oracle DB instance.

1. Create a new directory (named **MY_WALLET**, for example) for this new wallet bundle.

```
exec rdsadmin.rdsadmin_util.create_directory('MY_WALLET');
```

2. Download the new wallet artifacts from the S3 bucket to the new directory. Notice that we've passed on the **S3_SSL_WALLET** directory from the initial setup above to validate against the S3 bucket certificate. The download is requested over HTTPS.

```
BEGIN
  s3_download_presigned_url(
    '<S3 URL>',
    p_local_filename => 'cwallet.sso',
    p_local_directory => 'MY_WALLET',
    p_wallet_directory => 'S3_SSL_WALLET'
  );
END;
/
```

3. Run this procedure to use this newly uploaded wallet (for example with UTL_HTTP).

```
DECLARE
  l_wallet_path all_directories.directory_path%type;
BEGIN
  select directory_path into l_wallet_path from all_directories
where upper(directory_name)='MY_WALLET';
  utl_http.set_wallet('file:/'  || l_wallet_path);
END;
/
```

Similarly, you can upload and use any generic wallet where it's needed.

Examples of Using Oracle Wallets to Establish SSL/TLS Outbound Connections

Oracle wallets containing CA certificate bundles allow SSL/TLS-based outbound traffic to access any endpoint that can validate itself against one of the CA

certificates in the bundle. Here are a few examples of how you can use wallets to establish SSL/TLS outbound connections.

Using UTL_HTTP over an SSL/TLS Endpoint

Once you create a wallet, accessing an endpoint over SSL/TLS requires setting the wallet path. In this example, robots.txt from status.aws.amazon.com is accessed with an Oracle wallet containing Amazon's CA certificate (obtained from https://www.amazontrust.com/repository).

```
BEGIN
    utl_http.set_wallet('file:/rdsdbdata/userdirs/02');
END;
/

select
utl http.request('https://status.aws.amazon.com/robots.txt') as
ROBOTS_TXT from dual;

ROBOTS TXT
--------------
User-agent: *
Allow: /
```

Establishing Database Links between RDS Oracle DB Instances over an SSL/TLS Endpoint

Database links can be established between RDS Oracle DB instances over an SSL/TLS endpoint as long as the SSL option is configured for each instance.[11] No further setup is required.

Sending Emails Using UTL_SMTP and Amazon Simple Email Service (Amazon SES)

You can use Amazon SES to send emails on UTL_SMTP over SSL/TLS.

1. Obtain the relevant AWS Region endpoint and credentials from Amazon SES.[12]

2. Obtain a Verisign Symantec based CA certificates[13]
3. Create or update an existing wallet containing the relevant certificate. For this example, assume that the wallet has been uploaded to a directory called **SES_SSL_WALLET** created through the RDSADMIN utility.

   Using your Amazon SES SMTP credentials, send an email through UTL_SMTP using this sample code snippet.

Downloading a File from Amazon S3 to an RDS Oracle DB Instance

Using a utility similar to the **s3_download_presigned_url** procedure, you can download files from Amazon S3.

For example:

```
BEGIN
  s3 download presigned url(
    'https://<bucket-name>.s3.amazonaws.com/<sub-
directory>/<file>?AWSAccessKeyId=.....',
    p_local_filename => '<local-filename>',
    p_local_directory => '<target-local-directory>',
    p_wallet_directory => 'S3_SSL_WALLET'
  );
END;
/
```

Uploading a File from RDS Oracle DB Instance to Amazon S3

Uploading an artifact from your database instance to Amazon S3 is possible through HTTP PUT multipart requests using AWS Signature Version 4 signing.[14]

Conclusion

In this paper, we explained how to create Oracle wallets containing CA certificate bundles and copy them to Amazon RDS for Oracle DB instances. We also provided a few examples that showed how you can use wallets to establish SSL/TLS-based outbound connections. You can extend the steps highlighted in this paper to access any secure endpoint from your Amazon RDS Oracle DB instances.

Appendix

Sample PL/SQL Procedure to Download Artifacts from Amazon S3

```
-- Define your user here
define user='admin';

-- Direct-grant required privs
BEGIN
  rdsadmin.rdsadmin_util.grant_sys_object('DBA_DIRECTORIES',
UPPER('&user'));
END;
/

BEGIN
  rdsadmin.rdsadmin_util.grant_sys_object('UTL_HTTP',
UPPER('&user'));
END;
/

BEGIN
  rdsadmin.rdsadmin_util.grant_sys_object('UTL_FILE',
UPPER('&user'));
END;


-- Example download procedure
CREATE OR REPLACE PROCEDURE s3_download_presigned_url (
  p_s3_url IN VARCHAR2,
```

```
  p_local_filename IN VARCHAR2,
  p_local_directory IN VARCHAR2,
  p_wallet_directory IN VARCHAR2 DEFAULT NULL
) AS
  -- Local variables
  l_req utl_http.req;
  l_wallet_path VARCHAR2(4000);
  l_fh utl_file.file_type;
  l_resp utl_http.resp;
  l_data raw(32767);
  l_file_size NUMBER;
  l_file_exists BOOLEAN;
  l_block_size BINARY_INTEGER;
  l_http_status NUMBER;

  -- User-defined exceptions
  e_https_requires_wallet EXCEPTION;
  e_wallet_dir_invalid EXCEPTION;
  e_http_exception EXCEPTION;

BEGIN
  -- Validate input
  IF (regexp_like(p_s3_url, '^https:', 'i') AND
p_wallet_directory IS NULL) THEN
    raise e_https_requires_wallet;
  END IF;

  -- Use wallet, if specified
  IF (p_wallet_directory IS NOT NULL) THEN
    BEGIN
      SELECT directory_path INTO l_wallet_path
      FROM dba_directories
      WHERE upper(directory_name)=upper(p_wallet_directory);
      utl_http.set_wallet('file:' || l_wallet_path);
    EXCEPTION
      WHEN NO_DATA_FOUND
        THEN raise e_wallet_dir_invalid;
    END;
  END IF;

  -- Do HTTP request
  BEGIN
```

```
    l_req := utl_http.begin_request(p_s3_url, 'GET',
'HTTP/1.1');
    l_fh := utl_file.fopen(p_local_directory, p_local_filename,
'wb', 32767);
    l_resp := utl_http.get_response(l_req);

    -- If we get HTTP error code, write that instead
    l_http_status := l_resp.status_code;
    IF (l_http_status != 200) THEN
      dbms_output.put_line('WARNING: HTTP response '
        || l_http_status
        || ' - ' || l resp.reason phrase
        || '. Details in ' || p_local_filename
      );
    END IF;

    -- Loop over response and write to file
    BEGIN
      LOOP
        utl_http.read_raw(l_resp, l_data, 32766);
        utl file.put raw(l fh, l data, true);
      END LOOP;
    EXCEPTION
      WHEN utl_http.end_of_body THEN
        utl_http.end_response(l_resp);
    END;

    -- Get file attributes to see what we did
    utl_file.fgetattr(
      location => p_local_directory,
      filename => p_local_filename,
      fexists => l_file_exists,
      file_length => l_file_size,
      block_size => l_block_size
    );
    utl_file.fclose(l_fh);
    dbms_output.put_line('wrote ' || l_file_size || ' bytes');

  EXCEPTION
    WHEN OTHERS THEN
      utl_http.end_response(l_resp);
      utl_file.fclose(l_fh);
      dbms_output.put_line(dbms_utility.format_error_stack());
```

```
dbms_output.put_line(dbms_utility.format_error_backtrace());
      raise;
  END;

EXCEPTION
  WHEN e_https_requires_wallet THEN
    dbms_output.put_line('ERROR: HTTPS requires a valid wallet
location');
  WHEN e_wallet_dir_invalid THEN
    dbms_output.put_line('ERROR: wallet directory not found');
  WHEN others THEN
      raise;

END s3_download_presigned_url;
/
```

Sample PL/SQL Procedure to Send an Email Through Amazon SES

```
declare
  l_smtp_server varchar2(1024) := 'email-smtp.us-west-
2.amazonaws.com';
  l_smtp_port number := 587;
  l_wallet_dir varchar2(128) := 'SES_SSL_WALLET';
  l_from varchar2(128) := 'user@lorem-ipsum-dolar';
  l_to varchar2(128)  := 'user@lorem-ipsum-dolar';
  l_user varchar2(128) := '<USERNAME>';
  l_password varchar2(128) := '<PASSWORD>';
  l_subject varchar2(128) := 'Test subject';
  l_wallet_path varchar2(4000);
  l_conn utl_smtp.connection;
  l_reply utl_smtp.reply;
  l_replies utl_smtp.replies;
begin
  select 'file:/' || directory_path into l_wallet_path from
dba_directories where directory_name=l_wallet_dir;
```

```
  -- open a connection
  l_reply := utl_smtp.open_connection(
    host => l_smtp_server,
    port => l_smtp_port,
    c => l_conn,
    wallet_path => l_wallet_path,
    secure_connection_before_smtp => false
  );
  dbms_output.put_line('opened connection, received reply ' ||
l_reply.code || '/' || l_reply.text);

  -- get supported configs from server
  l_replies := utl_smtp.ehlo(l_conn, 'localhost');
  for r in 1..l_replies.count loop
    dbms output.put line('ehlo (server config) : ' ||
l_replies(r).code || '/' || l_replies(r).text);
  end loop;

  -- STARTTLS
  l_reply := utl_smtp.starttls(l_conn);
  dbms output.put line('starttls, received reply ' ||
l_reply.code || '/' || l_reply.text);

  --
  l_replies := utl_smtp.ehlo(l_conn, 'localhost');
  for r in 1..l_replies.count loop
    dbms_output.put_line('ehlo (server config) : ' ||
l_replies(r).code || '/' || l_replies(r).text);
  end loop;

  utl_smtp.auth(l_conn, l_user, l_password,
utl_smtp.all_schemes);

  utl_smtp.mail(l_conn, l_from);
  utl_smtp.rcpt(l_conn, l_to);
  utl_smtp.open_data l_conn);
  utl_smtp.write_data(l_conn, 'Date: ' || to_char(SYSDATE, 'DD-
MON-YYYY HH24:MI:SS') || utl_tcp.crlf);
  utl_smtp.write_data(l_conn, 'From: ' || l_from ||
utl_tcp.crlf);
  utl_smtp.write_data(l_conn, 'To: ' || l_to || utl_tcp.crlf);
  utl_smtp.write_data(l_conn, 'Subject: ' || l_subject ||
utl_tcp.crlf);
```

```
  utl_smtp.write_data(l_conn, '' || utl_tcp.crlf);
  utl_smtp.write_data(l_conn, 'Test message.' || utl_tcp.crlf);

  utl_smtp.close_data(l_conn);

  l_reply := utl_smtp.quit(l_conn);
exception
  when others then
    utl_smtp.quit(l_conn);
    raise;
end;
/
```

Notes

[1] https://aws.amazon.com/rds/

[2] https://aws.amazon.com/vpc/

[3] https://aws.amazon.com/ec2/

[4] http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_VPC.WorkingWithRDSInstanceinaVPC.html#USER_VPC.Non-VPC2VPC

[5] http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_Oracle.html#Oracle.Concepts.ONA

[6] http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Appendix.Oracle.CommonDBATasks.System.html#Appendix.Oracle.CommonDBATasks.CustomDNS

[7] https://www.digicert.com/digicert-root-certificates.htm

[8] https://docs.oracle.com/database/121/DBSEG/asoappf.htm#DBSEG610

[9] http://docs.aws.amazon.com/cli/latest/userguide/using-s3-commands.html

[10] http://docs.aws.amazon.com/cli/latest/reference/s3/presign.html

[11] https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Appendix.Oracle.Options.SSL.html

[12] https://docs.aws.amazon.com/ses/latest/DeveloperGuide/send-email-smtp.html

[13] https://www.symantec.com/theme/roots

[14] https://docs.aws.amazon.com/AmazonS3/latest/API/sigv4-authentication-HTTPPOST.html