Implementing SSL with Load Balancing for Oracle MWA

Intellinum

Intellinum

ORACLE
MOBILE WEB APPLICATIONS

Implementing SSL with Load Balancing for Oracle MWA

Over the years, there are many Oracle E-Business WMS and MSCA customers who are looking to enhance the security of communication between the mobile MWA users and Oracle MWA server. As we know, the MWA server used in MSCA and WMS natively runs in Telnet protocol where all data including username and password are sent in clear text. One of the ways to secure the communication is to encrypt the data between client and server and this can be done using SSL (Secure Sockets Layer).

In recent update, Oracle added SSL support in MWA server. However SSL is not supported by MWA Dispatcher, confirmed in “Does MWA Dispatcher (MWADIS) Accept SSL or TLS Connections? (Doc ID 2175344.1)”. This is unfortunate as many customers are using MWA dispatcher to load balance their MWA users. In such case, to make SSL work customer has to either connect the device directly to MWA port or use other types of load balancer which supports SSL passthrough.

This document is written to help customers in two regards. First, it describes steps to enable SSL in Oracle MWA server. For reference, please read information in note "Oracle Mobile Supply Chain Applications for Oracle E-Business Suite Release Notes (Doc ID 2108155.1)" to enable SSL in standard Oracle MWA server. Second, it introduces a high performing popular software load balancer called haproxy and show how to use it to load balance multiple MWA ports running in SSL mode.

The high level steps are as follows:

1. Generate an SSL certificate, we can either get this from a trusted provider (paid) or we can generate a self-signed certificate (free)
2. Configure mwa.cfg to enable the SSL
3. Restart MWA service
4. Install and configure haproxy

To generate a self-signed SSL certificate using the keytool command on Oracle EBS application node:

1. Open a command prompt or terminal
2. Run this command:
   keytool -genkey -keyalg RSA -alias tomcat -keystore mwassl.jks -validity <days> -keysize 2048
   Where <days> indicate the number of days for which the certificate will be valid.
3. Enter a password for the keystore. Note this password as you require this for configuring the server
4. When prompted for a first name and the last name, press Enter to keep it blank.
5. Enter the other details, such as Organizational Unit, Organization, City, State, and Country
6. Confirm that the information entered is correct
7. When prompted with "Enter key" password for <mwassl>, press Enter to use the same password as the keystore password

9. Run this command to verify the contents of the keystore

   keytool -list -v -keystore mwassl.jks

11. When prompted, enter the keystore password note in Step 3. The basic information about the generated certificate is displayed. Verify that the "Owner" and "Issuer" are the same. Also, you should see the information you provided in Step 5 and 6.

The certificate is now complete and ready to use. Move this file to $INST_TOP/admin/install. If you run in 12.2.X, please move to both RUN and PATCH.

Screenshot of how we generate a self-signed certificate.

```
[applmgr@mtr1224dev tmp]$ keytool -genkey -keyalg RSA -alias mwassl -keystore mwassl.jks -validity 365 -keysize 2048
Enter keystore password:
Re-enter new password:
What is your first and last name?
  [Unknown]:
What is the name of your organizational unit?
  [Unknown]:  Intellinum
What is the name of your organization?
  [Unknown]:  Intellinum
What is the name of your City or Locality?
  [Unknown]:  Dallas
What is the name of your State or Province?
  [Unknown]:  TX
What is the two-letter country code for this unit?
  [Unknown]:  US
Is CN=Unknown, OU=Intellinum, O=Intellinum, L=Dallas, ST=TX, C=US correct?
  [no]:  yes

Enter key password for <mwassl>
        (RETURN if same as keystore password):
[applmgr@mtr1224dev tmp]$ ls -alrt
```

Screenshot of how we verify the self-signed certificate.

```
[applmgr@mtr1224dev tmp]$ keytool -list -v -keystore mwassl.jks
Enter keystore password:

Keystore type: JKS
Keystore provider: SUN

Your keystore contains 1 entry

Alias name: mwassl
Creation date: Aug 26, 2019
Entry type: PrivateKeyEntry
Certificate chain length: 1
Certificate[1]:
Owner: CN=Unknown, OU=Intellinum, O=Intellinum, L=Dallas, ST=TX, C=US
Issuer: CN=Unknown, OU=Intellinum, O=Intellinum, L=Dallas, ST=TX, C=US
Serial number: 42cd536b
Valid from: Mon Aug 26 16:19:46 CDT 2019 until: Tue Aug 25 16:19:46 CDT 2020
Certificate fingerprints:
         MD5:  EB:AE:FC:BF:65:11:36:76:6F:12:2D:39:73:64:E0:9F
         SHA1: 6E:B5:66:14:A5:7B:DB:2D:B4:57:76:35:23:C4:4F:F3:CC:78:7C:5D
         SHA256: 25:AD:7A:01:8F:C2:6F:17:96:39:99:17:2D:90:9F:B4:B9:88:59:64:22:11:04:18:28:1C:58:62:E9:28:98:54
         Signature algorithm name: SHA256withRSA
         Version: 3

Extensions:

#1: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: B0 08 21 D9 F2 A0 7E 75   66 F5 82 92 D3 7D C9 9D  ..!....uf.......
0010: A5 D1 C2 25                                        ...%
]
]



*******************************************
*******************************************


[applmgr@mtr1224dev tmp]$
```

Next, we edit $INST_TOP/admin/install/mwa.cfg file to enable the SSL. For 12.2.X, please update for both RUN and PATCH. In the file, there are four properties related to SSL.

mwa.Security.SSLSocket=TRUE
mwa.Security.KeyStore=mwassl.jks
mwa.Security.KeyStorePassword=mwasslpassword
mwa.Security.Protocols=TLSv1
mwa.Security.CipherSuites=SSL_RSA_WITH_RC4_128_SHA,TLS_RSA_WITH_AES_128_CBC_SHA,TLS_DHE_RSA_WITH_AES_128_CBC_SHA,TLS_DHE_DSS_WITH_AES_128_CBC_SHA,SSL_RSA_WITH_3DES_EDE_CBC_SHA,SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA,SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA

The mwa.Security.KeyStore should be filename of the SSL certificate we generated earlier, and mwa.Security.KeyStorePassword should be the password we use when generating the certificate.

Once done, restart the MWA services and now we have MWA server running in secure SSL!

Please note that once MWA server runs in SSL mode, we can no longer use non-secure MWA client, like telnet client, to connect. We have to use MWA client that supports SSL mode to connect to the server. One such client is Intellinum Express Client, a native GUI client that supports SSL, available for Android, Windows CE/Mobile and Windows PC.

As mentioned earlier, once we enable SSL, we can't use MWA dispatcher anymore as it does not decrypt/encrype SSL data. One good option that we have tested to work with Oracle MWA server or our Intellinum Express Server is haproxy load balancer (www.haproxy.org). This is a high performing and reliable TCP/HTTP software load balancer used by many large organizations in the world.

There are many good resources in the Internet on how to install and configure haproxy. Once installed and started, we can verify haproxy is running by using command:
sudo service haproxy status

```
intellinum@flexitest-ubuntu:~$ sudo service haproxy status
haproxy is running.
intellinum@flexitest-ubuntu:~$
```

Similar to MWA which has configuration file mwa.cfg, haproxy has a configuration file haproxy.cfg. The default location is /etc/haproxy/haproxy.cfg. For reference, we have shown below a sample haproxy.cfg which runs the load balancer on port 10860 (think of this like MWA dispatcher port) and load balances to 3 MWA ports running on 192.168.1.232 port 10200, 10202, 10204 respectively.

```
global
        log /dev/log    local0
        log /dev/log    local1 notice
        chroot /var/lib/haproxy
        stats socket /run/haproxy/admin.sock mode 660 level admin
        stats timeout 30s
        user haproxy
        group haproxy
        daemon

defaults
        log     global
        mode    tcp
        timeout connect 30s
        timeout client  1h
        timeout server  1h
        option tcplog

frontend mwaproxy
  mode tcp
  bind *:10860
  default_backend mymwaservers

backend mymwaservers
  mode tcp
  server mwa1 192.168.1.232:10200 check
  server mwa2 192.168.1.232:10202 check
  server mwa3 192.168.1.232:10204 check
  stick-table type ip size 1m expire 30m
  stick on src
  option redispatch
```

The important relevant sections in haproxy.cfg are the frontend and backend. The frontend has the port it should listen for incoming connection and the backend lists all the mwa ports we want to load balance. The stick-table and stick are used for sticky session so in case a client gets disconnected from server (due to WiFi interruption for example), the client can be forwarded to the same server it was connected before the disconnection.

Notice that we do not do anything special to implement TLS passthrough. It's enough to pass the traffic to the backend servers and let them handle the encryption and decryption. That's all there is to TLS passthrough.

Don’t forget to restart haproxy after any changes in haproxy.cfg with the following command:
sudo service haproxy restart

Now we can connect MWA client to host/port where haproxy is running and it should forward the connection to MWA server. We have verified that all features in Intellinum Express Server such as transaction personalization, voice interface, e-signature and quick reconnect are working in SSL stand alone or SSL with haproxy.

Hope this article has been useful for you. If there is any questions on this article, or you would like to try our products Express Client or Express Server, feel free to reach out to inquiry@intellinum.com.