OPENSSL BASICS

Basic openssl testing command

Abstract

This document will cover the basic SSL command to test the Protocols whether it is enabled or disabled along with certification creation

Sahu, Haramohan

Hara.sahu@gmail.com

Contents

What is OpenSSL 3
What does OpenSSL S_client do? 3
What is an SSL Certificate? How Does SSL Work? 3
What is an SSL Certificate? How Does SSL Work? 3
What's the Difference Between TLS and SSL? 3
Where Can I Get an SSL Certificate? 3
OpenSSL Version Command 4
OpenSSL and CSR Creation 5
Deciding on Key Generation Options 5
Generating Your Private Key 6
How to see the key 6
Extracting Your Public Key 6
Creating Your CSR 6
Using the -subj Switch 7
Creating Your CSR with One Command 8
Viewing Certificate Information 8
Verifying Your Keys Match 8
A brief about TLS 8
Differences between SSL and TLS 9
How to verify if the Target is using a SSL/TLS protocol 10
How to identify if a SSL/TLS protocol is enabled/disabled. 10
How to Disable/Enable SSL/TLS protocols in Apache/Linux Server in dfm? 11
Example on target without SSLv3 enabled. 14
Enabled SSL/TLS Protocol output 15
-curves curvelist 18
How to debug ssl server 20
How to do dfm ssl server setup: 21
DFM commands for plink testing: 23
DFM LDAP SET UP: 24
DFM PORT 8488: 24
Where does SSL binary resides in installed directory(DFM) 25

What is OpenSSL

OpenSSL is an open-source command line tool that is commonly used to generate private keys, create CSRs, install your SSL/TLS certificate, and identify certificate information.

What does OpenSSL S_client do?

The **s_client** command implements a generic **SSL**/TLS client which connects to a remote host using **SSL**/TLS. It is a very useful diagnostic tool for **SSL** servers.

What is an SSL Certificate? How Does SSL Work?

A Secure Socket Layer (SSL) certificate is a security protocol which secures data between two computers by using encryption.

Note: Simply put, an SSL certificate is a data file that digitally ties a Cryptographic Key to a server or domain and an organization's name and location.

What is an SSL Certificate? How Does SSL Work?

Typically, SSL certificates are used on web pages that transmit and receive end-user sensitive data,
such as Social Security Number, credit card details, home address or password.
Online payment forms are a good example and usually encrypt the aforementioned delicate information
using 128 or 256-bit SSL technology.

SSL certificates ensure the identity of a remote computer, most commonly a server,
but also confirms your computer's identity to the remote computer to establish a safe connection.
Keeping the internet safe has always been a two-way street and thanks to SSL encryption,
the server "shakes hands" with your personal computer and both sides know with whom they are communicating.

What's the Difference Between TLS and SSL?

There is none. Transport Layer Security (TLS) is an updated version of Secure Socket Layer (SSL).
Even though most secure connections are via TLS protocols, people keep calling it SSL.

Google is sure to persuade you. Namely, starting from July 2018 Google flags each website without SSL as unsafe.

Where Can I Get an SSL Certificate?

SSL certificates are verified and issued by a Certificate Authority (CA). You apply by generating a CSR with a key pair on your server that would, ideally, hold the SSL certificate. The CSR contains crucial organization details which the CA verifies.

Generate a CSR and key pair locally on your server. The key pair consists of a public and private key.
Send the CSR and public key to a CA who will verify your legal identity and whether you own and
control the domain submitted in the application. The Certificate Authority runs a check on your organization and validates if the organization is registered at the location provided in the CSR and whether the domain exists. When verified, the organization receives a copy of their SSL certificate including business details as well as the public key. The organization can now install the certificate on their server. When a CA issues the certificate, it binds to a certificate authority's "trusted root" certificate. Root certificates are embedded into each browser and connected to individually issued certificates to establish an HTTPS connection.

OpenSSL Version Command

$ openssl version

Openssl Flgas:

openssl version –help

There are eight (8) valid options that allow you to narrow your search. The option that provides the most comprehensive set of information is: This command compiles all the information contained under the individual flags into a single output.

```
~$ openssl version -help
Usage: version [options]
Valid options are:
 -help  Display this summary
 -a     Show all data
 -b     Show build date
 -d     Show configuration directory
 -e     Show engines directory
 -f     Show compiler flags used
 -o     Show some internal datatype options
 -p     Show target build platform
 -v     Show library version
```

./openssl version -a
WARNING: can't open config file: /opt/NTAPdfm/openssl/openssl.cnf
OpenSSL 1.0.2u-fips 20 Dec 2019
built on: reproducible build, date unspecified
platform: linux-x86_64
options: bn(64,64) rc4(ptr,int) des(idx,cisc,16,int) blowfish(idx)
compiler: gcc -I. -I.. -I../include -D_GNU_SOURCE -fPIC -DOPENSSL_PIC -DOPENSSL_THREADS -D_REENTRANT -DDSO_DLFCN -DHAVE_DLFCN_H -m64 -DL_ENDIAN -O3 -Wall -I/u/pramodks/openssl_build_2u/apache/include
OPENSSLDIR: "/opt/NTAPdfm/openssl"

```
~$ openssl version -a
OpenSSL 1.1.0g  2 Nov 2017
built on: reproducible build, date unspecified
platform: debian-amd64
compiler: gcc -DDSO_DLFCN -DHAVE_DLFCN_H -DNDEBUG -DOPENSSL_THREADS -DOPENSSL_N
O_STATIC_ENGINE -DOPENSSL_PIC -DOPENSSL_IA32_SSE2 -DOPENSSL_BN_ASM_MONT -DOPENS
SL_BN_ASM_MONT5 -DOPENSSL_BN_ASM_GF2m -DSHA1_ASM -DSHA256_ASM -DSHA512_ASM -DRC
4_ASM -DMD5_ASM -DAES_ASM -DVPAES_ASM -DBSAES_ASM -DGHASH_ASM -DECP_NISTZ256_AS
M -DPADLOCK_ASM -DPOLY1305_ASM -DOPENSSLDIR="\"/usr/lib/ssl\"" -DENGINESDIR="\"
/usr/lib/x86_64-linux-gnu/engines-1.1\""
OPENSSLDIR: "/usr/lib/ssl"
ENGINESDIR: "/usr/lib/x86_64-linux-gnu/engines-1.1"
```

./openssl genrsa -out yourdomain.key 2048
WARNING: can't open config file: /opt/NTAPdfm/openssl/openssl.cnf
Generating RSA private key, 2048 bit long modulus
.....+++++
.....+++++
e is 65537 (0x10001)

./openssl version -d
WARNING: can't open config file: /opt/NTAPdfm/openssl/openssl.cnf
OPENSSLDIR: "/opt/NTAPdfm/openssl"

```
~$ openssl version -d
OPENSSLDIR: "/usr/lib/ssl"
```

OpenSSL and CSR Creation

Deciding on Key Generation Options

When generating a key, you have to decide three things: the key algorithm, the key size, and whether to use a passphrase.

Key Algorithm

For the key algorithm, you need to take into account its compatibility. For this reason, we recommend you use RSA. However, if you have a specific need to use another algorithm (such as ECDSA), you can use that too, but be aware of the compatibility issues you might run into.

Note: This guide only covers generating keys using the RSA algorithm.
Key Size

For the key size, you need to select a bit length of at least 2048 when using RSA and 256 when using ECDSA; these are the smallest key sizes allowed for SSL certificates. Unless you need to use a larger key size, we recommend sticking with 2048 with RSA and 256 with ECDSA.

Note: In older versions of OpenSSL, if no key size is specified, the default key size of 512 is used. Any key size lower than 2048 is considered unsecure and should never be used.
Passphrase

For the passphrase, you need to decide whether you want to use one. If used, the private key will be encrypted using the specified encryption method, and it will be impossible to use without the passphrase. Because there are pros and cons with both options, it's important you understand the implications of using or not using a passphrase. In this guide, we will not be using a passphrase in our examples.

Generating Your Private Key

Use the following command to generate your private key using the RSA algorithm:

```
openssl genrsa -out yourdomain.key 2048
```

This command generates a private key in your current directory named yourdomain.key (-out yourdomain.key) using the RSA algorithm (genrsa) with a key length of 2048 bits (2048). The generated key is created using the OpenSSL format called PEM.

How to see the key

```
cat yourdomain.key
```

Use the following command to decode the private key and view its contents:

```
openssl rsa -text -in yourdomain.key -noout
```

The -noout switch omits the output of the encoded version of the private key.

Extracting Your Public Key

```
openssl rsa -in yourdomain.key -pubout -out yourdomain_public.key
```

Creating Your CSR

```
openssl req -new -key yourdomain.key -out yourdomain.csr
```

After entering the command, you will be asked series of questions. Your answers to these questions will be embedded in the CSR. Answer the questions as described below:

Country Name (2 letter code)	The two-letter country code where your company is legally located.
State or Province Name (full name)	The state/province where your company is legally located.

Locality Name (e.g., city)	The city where your company is legally located.
Organization Name (e.g., company)	Your company's legally registered name (e.g., YourCompany, Inc.).
Organizational Unit Name (e.g., section)	The name of your department within the organization. (You can leave this option blank; simply press **Enter**.)
Common Name (e.g., server FQDN)	The fully-qualified domain name (FQDN) (e.g., www.example.com).
Email Address	Your email address. (You can leave this option blank; simply press **Enter**.)
A challenge password	Leave this option blank (simply press **Enter**).
An optional company name	Leave this option blank (simply press **Enter**).

Some of the above CSR questions have default values that will be used if you leave the answer blank and press **Enter**. These default values are pulled from the OpenSSL configuration file located in the **OPENSSLDIR** (see Checking Your OpenSSL Version). If you want to leave a question blank without using the default value, type a "**.**" (period) and press **Enter**.

Using the -subj Switch

Another option when creating a CSR is to provide all the necessary information within the command itself by using the **-subj** switch.

Use the following command to disable question prompts when generating a CSR:

```
openssl req -new -key yourdomain.key -out yourdomain.csr \
-subj "/C=US/ST=Utah/L=Lehi/O=Your Company, Inc./OU=IT/CN=yourdomain.com"
```

This command uses your private key file (**-key yourdomain.key**) to create a new CSR (**-out yourdomain.csr**) and disables question prompts by providing the CSR information (**-subj**).

Creating Your CSR with One Command

Instead of generating a private key and then creating a CSR in two separate steps, you can actually perform both tasks at once.

Use the following command to create both the private key and CSR:

```
openssl req -new \
-newkey rsa:2048 -nodes -keyout yourdomain.key \
-out yourdomain.csr \
-subj "/C=US/ST=Utah/L=Lehi/O=Your Company, Inc./OU=IT/CN=yourdomain.com"
```

This command generates a new private key (-newkey) using the RSA algorithm with a 2048-bit key length (rsa:2048) without using a passphrase (-nodes) and then creates the key file with a name of yourdomain.key (-keyout yourdomain.key).

The command then generates the CSR with a filename of yourdomain.csr (**-out yourdomain.csr**) and the information for the CSR is supplied (**-subj**).

Note: While it is possible to add a subject alternative name (SAN) to a CSR using OpenSSL, the process is a bit complicated and involved. If you do need to add a SAN to your certificate, this can easily be done by adding them to the order form when purchasing your DigiCert certificate.

Viewing Certificate Information

Use the following command to view the contents of your certificate:

```
openssl x509 -text -in yourdomain.crt -noout
```

Verifying Your Keys Match

1. openssl rsa -modulus -in yourdomain.key -noout | openssl sha256
2. openssl req -modulus -in yourdomain.csr -noout | openssl sha256
3. openssl x509 -modulus -in yourdomain.crt -noout | openssl sha256

A brief about TLS

TLS means Transport Layer Security, which is a cryptographic protocol successor of SSL 3.0, which was released in 1999.

1. TLS 1.0 TLS 1.0 which was upgrade of SSL v.3.0 released in January 1999 but it allows connection downgrade to SSL v.3.0.

2. TLS 1.1 After that, TLS v1.1 was released in April 2006, which was an update of TLS 1.0 version. It added protection against CBC (Cipher Block Chaining) attacks. In March 2020, Google, Apple, Mozilla and Microsoft has announced for deprecation of TLS 1.0 and 1.1 versions.

3. TLS 1.2 TLS v1.2 was released in 2008 that allows to specification of hash and algorithm used by the client and server.It allows authenticated encryption, which was added more support with extra data modes. TLS 1.2 was able to verify length of data based on cipher suite.

4. TLS 1.3 TLS v1.3 was released in August 2018 and had major features that differentiate it with its earlier version TLS v1.2 like removal of MD5 and SHA-224 support, require digital signature when earlier configuration used, compulsory use of Perfect forward secrecy in case of public-key based key exchange, handshake messages will now be encrypted after “Server Hello”.

Differences between SSL and TLS

However, the differences between SSL and TLS are very minor. In fact, only a technical person will be able to spot the differences. The notable differences include:

1. Cipher suites

SSL protocol offers support for Fortezza cipher suite. TLS does not offer support. TLS follows a better standardization process that makes defining of new cipher suites easier like RC4, Triple DES, AES, IDEA, etc.

2. Alert messages

SSL has the “No certificate” alert message. TLS protocol removes the alert message and replaces it with several other alert messages.

3. Record Protocol

SSL uses Message Authentication Code (MAC) after encrypting each message while TLS on the other hand uses HMAC — a hash-based message authentication code after each message encryption.

4. Handshake process

In SSL, the hash calculation also comprises the master secret and pad while in TLS, the hashes are calculated over handshake message.

5. Message Authentication

SSL message authentication adjoins the key details and application data in ad-hoc way while TLS version relies on HMAC Hash-based Message Authentication Code.

In nutshell, SSL is obsolete and TLS is new name of older SSL protocol as modern encryption standard using by everybody. Technically, TLS is more accurate, but everyone knows SSL.

How to verify if the Target is using a SSL/TLS protocol

Below command is to test if a particular set of SSL/TLS protocol is enabled on the system.

How to identify if a SSL/TLS protocol is enabled/disabled.

In order to test if everything is okay and only TLS 1.2 is available you can use either nmap or openssl

Example namp command:

- nmap --script ssl-enum-ciphers -p 443 10.193.113.50
- nmap --script ssl-enum-ciphers -p 8443 10.193.113.50 | grep -E "TLSv|SSLv"

./nmap --script ssl-enum-ciphers -p 443 10.193.113.50

```
Starting Nmap 6.40 ( http://nmap.org ) at 2020-08-07 09:14 EDT
Nmap scan report for scspa1934315001.rtp.openenglab.netapp.com (10.193.113.50)
Host is up (0.000082s latency).
PORT    STATE SERVICE
443/tcp open  https
| ssl-enum-ciphers:
|   TLSv1.1:
|     ciphers:
|       TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA - strong
|       TLS_RSA_WITH_AES_128_CBC_SHA - strong
|       TLS_RSA_WITH_AES_256_CBC_SHA - strong
|     compressors:
|       NULL
|   TLSv1.2:
|     ciphers:
|       TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA - strong
|       TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 - strong
|       TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 - strong
|       TLS_RSA_WITH_AES_128_CBC_SHA - strong
|       TLS_RSA_WITH_AES_128_GCM_SHA256 - strong
|       TLS_RSA_WITH_AES_256_CBC_SHA - strong
|       TLS_RSA_WITH_AES_256_CBC_SHA256 - strong
|     compressors:
|       NULL
|_  least strength: strong
```

Nmap done: 1 IP address (1 host up) scanned in 0.44 seconds

[root@netapp ~]# nmap --script ssl-enum-ciphers -p 8443 10.193.113.50 | grep -E "TLSv|SSLv"
| SSLv3: No supported ciphers found
| TLSv1.0:
| TLSv1.1:
| TLSv1.2:
[root@netapp ~]#

How to Disable/Enable SSL/TLS protocols in Apache/Linux Server in dfm?

#dfm service stop httpd

Open httpd.conf file from dfm installed directory.
Add below line

#We disabled TLS 1.0/1.1 and SSL 2.0/3.0 here

SSLProtocol all -SSLv2 -SSLv3 -TLSv1 -TLSv1.1

```
<VirtualHost *:443>
   ServerName www.yourdomain.com
   DocumentRoot /var/www/html
   SSLEngine on
#We disabled TLS 1.0/1.1 and SSL 2.0/3.0 here
SSLProtocol all -SSLv2 -SSLv3 -TLSv1 -TLSv1.1

   SSLCertificateFile /etc/apache2/certificates/certificate.crt
   SSLCertificateKeyFile
/etc/apache2/certificates/certificate.key
   SSLCertificateChainFile
/etc/apache2/certificates/intermediate.crt
</VirtualHost>
```

Then restart httpd server.
#dfm service start httpd

Then test with openssl s_client -connect ipaddress:port -[protocol]

Perform the following command and use the following syntax to test on different protocol
openssl s_client -connect ipaddress:port -[protocol]

Like other protocols:

To Enable Only TLS 1.0:

```
SSLProtocol +TLSv1
```

- This will only allow TLS 1.0 protocol.

Note: This will disable all other protocols, so you no need to mention like

```
SSLProtocol -SSLv2 -SSLv3 -TLSv1.1 -TLSv1.2 +TLSv1
```

To Enable Only TLS 1.1:

```
SSLProtocol +TLSv1.1
```

- This will only allow TLS 1.1 protocol.

Note: This will disable all other protocols, so you no need to mention like

```
SSLProtocol -SSLv2 -SSLv3 -TLSv1 -TLSv1.2 +TLSv1.1
```

To Enable Only TLS 1.2:

```
SSLProtocol +TLSv1.2
```

- This will only allow TLS 1.2 protocol.

Note: This will disable all other protocols, so you no need to mention like

```
SSLProtocol -SSLv2 -SSLv3 -TLSv1 -TLSv1.1 +TLSv1.2
```

To Enable TLS 1.1 and TLS 1.2:

```
SSLProtocol +TLSv1.1 +TLSv1.2
```

- This will only allow TLS 1.1 and TLS 1.2 protocols.

Note: This will disable all other protocols, so you no need to mention like

```
SSLProtocol -SSLv2 -SSLv3 -TLSv1 +TLSv1.1 +TLSv1.2
```

To Enable TLS 1.0, TLS 1.1 and TLS 1.2:

```
SSLProtocol +TLSv1 +TLSv1.1 +TLSv1.2
```

- This will only allow TLS 1.0, TLS 1.1 and TLS 1.2 protocols.

Note: This will disable all other protocols, so you no need to mention like

```
SSLProtocol -SSLv2 -SSLv3 +TLSv1 +TLSv1.1 +TLSv1.2
```

Example Virtual Host File which only allows TLS 1.1 and TLS 1.2

```
<VirtualHost *:443>
   ServerName www.yourdomain.com
   DocumentRoot /var/www/html
   SSLEngine on
   SSLProtocol +TLSv1.1 +TLSv1.2
   SSLCertificateFile /etc/apache2/certificates/certificate.crt
   SSLCertificateKeyFile
/etc/apache2/certificates/certificate.key
   SSLCertificateChainFile
/etc/apache2/certificates/intermediate.crt
</VirtualHost>
```

Once changed this VirtualHost configuration file, you need to restart the apache. To restart the apache service use below command in Putty Command Window.

-ssl3	- just use SSLv3
-tls1_2	- just use TLSv1.2
-tls1_1	- just use TLSv1.1
-tls1	- just use TLSv1
-dtls1	- just use DTLSv1

Disabled SSL/TLS Protocol output

Example on target without SSLv3 enabled.
Command used:
openssl s_client -connect ipaddress:port -ssl3

Output: If the protocol is not enabled for the particular target, you will see a SSL error and the output will indicate that it is not able to obtain any certificate information.

[root@netapp sbin]**#./openssl s_client -connect 10.193.113.50:8488 -ssl3**
WARNING: can't open config file: /opt/NTAPdfm/openssl/openssl.cnf
CONNECTED(00000003)
write:errno=104
---
no peer certificate available

No client certificate CA names sent

SSL handshake has read 0 bytes and written 0 bytes

New, (NONE), Cipher is (NONE)
Secure Renegotiation IS NOT supported
Compression: NONE
Expansion: NONE
No ALPN negotiated
SSL-Session:
    Protocol : SSLv3
    Cipher   : 0000
    Session-ID:
    Session-ID-ctx:
    Master-Key:
    Key-Arg   : None
    PSK identity: None
    PSK identity hint: None
    Start Time: 1596803756
    Timeout   : 7200 (sec)
    Verify return code: 0 (ok)

Enabled SSL/TLS Protocol output

Example on target without TLS 1.2 enabled.

Command used:

openssl s_client - 10.193.113.50:8488 -tls1_2

Output: If a protocol is enabled for the particular target, you will see the complete certificate of the server and other information such as the SSL/TLS protocol negotiated, the server public key size, cipher suite negotiated and among other information.

./openssl s_client -connect 10.193.113.50:8488 -tls1_2

WARNING: can't open config file: /opt/NTAPdfm/openssl/openssl.cnf
CONNECTED(00000003)
depth=0 C = US, ST = California, L = San Jose, O = NetApp, OU = Storage Management, CN = netapp, emailAddress = support@NetApp.com
verify error:num=18:self signed certificate
verify return:1
depth=0 C = US, ST = California, L = San Jose, O = NetApp, OU = Storage Management, CN = netapp, emailAddress = support@NetApp.com
verify return:1

Certificate chain
 0 s:/C=US/ST=California/L=San Jose/O=NetApp/OU=Storage Management/CN=netapp/emailAddress=support@NetApp.com
   i:/C=US/ST=California/L=San Jose/O=NetApp/OU=Storage Management/CN=netapp/emailAddress=support@NetApp.com
---
Server certificate
-----BEGIN CERTIFICATE-----
MIIE2jCCA8KgAwIBAgIJALx07xsuIhapMA0GCSqGSIb3DQEBCwUAMIGXMQswCQYD
VQQGEwJVUzETMBEGA1UECAwKQ2FsaWZvcm5pYTERMA8GA1UEBwwIU2FuIEpvc2Ux
DzANBgNVBAoMBk5ldEFwcDEbMBkGA1UECwwSU3RvcmFnZSBNYW5hZ2VtZW50MQ8w
DQYDVQQDDAZuZXRhcHAxITAfBgkqhkiG9w0BCQEWEnN1cHBvcnRATmV0QXBwLmNv
bTAeFw0yMDA4MDYwNzI5MDRaFw0yMjA4MDYwNzI5MDRaMIGXMQswCQYDVQQGEwJV
UzETMBEGA1UECAwKQ2FsaWZvcm5pYTERMA8GA1UEBwwIU2FuIEpvc2UxDzANBgNV

BAoMBk5ldEFwcDEbMBkGA1UECwwSU3RvcmFnZSBNYW5hZ2VtZW50MQ8wDQYDVQQD
DAZuZXRhcHAxITAfBgkqhkiG9w0BCQEWEnN1cHBvcnRATmV0QXBwLmNvbTCCASIw
DQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBALzyNjmnRzLP+9dZ3XztQfqO8s97
ahiv8Nj4VugrhuXC88rLjJ4oqI3jTOSkDcLGy0Of7jvWbU4Ps7t+gOvKLdj1FCyG
zfIDBRBbmlQ04TFvmnZX1QRM1IJuNg+q4xXpGf3nmrN2Vgb2TBAY/78tTJ15Fjk2
Q4ltBTvTVYS/21juPZ+8BL6tKk8AZgkXe5wxqQUVxxpZTuXC0+9amC+H2zxm4+Zp
vrQSu8oyYi+4kUE152etbjdqgvvX6ARQ/m3Stri9QHBKz6cDnXmcPpF19wADDAJi
6kQC6V6sL5BW/AFPcX0rOjGiKNviDTfB97tIwITp+YFLaYQhHZTLP2u7m7UCAwEA
AaOCASUwggEhMB0GA1UdDgQWBBQBI8hzhXnL9Q6HQZRvX7GMjw/18TCBzAYDVR0j
BIHEMIHBgBQBI8hzhXnL9Q6HQZRvX7GMjw/18aGBnaSBmjCBlzELMAkGA1UEBhMC
VVMxEzARBgNVBAgMCkNhbGlmb3JuaWExETAPBgNVBAcMCFNhbiBKb3NlMQ8wDQYD
VQQKDAZOZXRBcHAxGzAZBgNVBAsMElN0b3JhZ2UgTWFuYWdlbWVudDEPMA0GA1UE
AwwGbmV0YXBwMSEwHwYJKoZIhvcNAQkBFhJzdXBwb3J0QE5ldEFwcC5jb22CCQC8
dO8bLilWqTAxBgNVHREEKjAopCYwJDEiMCAGA1UEAwwZRGF0YUZhYnJpYyBNYW5h
Z2VyIHNlcnZlcjANBgkqhkiG9w0BAQsFAAOCAQEAn7QbCmBxt8khR0Ulf7y356M4
PpSHivnK+TfGJHzBg23fjzleRx88qsQ6+LW+VbZjO6rFuqUzXD69x+vN6t4yNkHv
18xTy+PBDyk0o8PqH1I9zCggqIPNRieQdXdLl4WVB+ra3XFLoZkm6zGSGQ0sSfv0
9yi+eXKGoNIOu1lBXGgq9K0+Ke6jR9VSpWIe+rr5VtMRD93gXPiDmZh0lVM/cSfQ
Z8KQohL7+okPN3VpS7WJwjIQDBO/XA6/89W04yduea6xN+Quq+xT+b2wIKLPbAnU
VVIu1QL7uSzJ83KLWbuGLODNBv3uK6iLD1sM7DsAGYos23B8DQMPJTNZJzvf3g==
-----END CERTIFICATE-----
subject=/C=US/ST=California/L=San Jose/O=NetApp/OU=Storage Management/CN=netapp/emailAddress=support@NetApp.com
issuer=/C=US/ST=California/L=San Jose/O=NetApp/OU=Storage Management/CN=netapp/emailAddress=support@NetApp.com
---
No client certificate CA names sent
Client Certificate Types: RSA sign, DSA sign, ECDSA sign
Requested Signature Algorithms: RSA+SHA512:DSA+SHA512:ECDSA+SHA512:RSA+SHA384:DSA+SHA384:ECDSA+SHA384:RSA+SHA256:DSA+SHA256:ECDSA+SHA256:RSA+SHA224:DSA+SHA224:ECDSA+SHA224:RSA+SHA1:DSA+SHA1:ECDSA+SHA1
Shared Requested Signature Algorithms: RSA+SHA512:DSA+SHA512:ECDSA+SHA512:RSA+SHA384:DSA+SHA384:ECDSA+SHA384:RSA+SHA256:DSA+SHA256:ECDSA+SHA256:RSA+SHA224:DSA+SHA224:ECDSA+SHA224:RSA+SHA1:DSA+SHA1:ECDSA+SHA1

SSL handshake has read 1601 bytes and written 635 bytes

New, TLSv1/SSLv3, Cipher is AES256-GCM-SHA384
Server public key is 2048 bit
Secure Renegotiation IS supported
Compression: NONE
Expansion: NONE
No ALPN negotiated
SSL-Session:
    Protocol : TLSv1.2
    Cipher   : AES256-GCM-SHA384
    Session-ID:
228F77EA478F5D4F964AE732A184575C4DBDBA43FC442200EF6927AD6ADD9001
    Session-ID-ctx:
    Master-Key:
940111EBDD6B091BD71B927E8764FDD9BBDE670D0355C0F7F449DCBBC36C2ED327D12
A4DBE502D2C19AC84C0964A2BDF
    Key-Arg   : None
    PSK identity: None
    PSK identity hint: None
    TLS session ticket lifetime hint: 300 (seconds)
    TLS session ticket:
    0000 - 86 bb 19 80 43 9f 98 1d-15 58 bc 22 a4 95 06 2d   ....C....X."...-
    0010 - 4a 60 ca bd 9f 22 46 84-65 33 98 48 20 d0 fa a3   J`..."F.e3.H ...
    0020 - aa cc 71 77 b5 c7 4b ce-d5 03 b6 71 62 07 69 be   ..qw..K....qb.i.
    0030 - 00 42 cb e8 e7 d8 88 35-7d a5 bb 2e 32 d5 da c0   .B.....5}...2...
    0040 - 2d f2 33 52 60 5e 37 83-5e ae 74 77 6e 71 59 35   -.3R`^7.^.twnqY5
    0050 - a3 90 0f f5 38 37 4a c4-7b 0f bf b8 f5 4b 97 a3   ....87J.{....K..
    0060 - 0e 3e 7b 93 23 20 01 e5-a9 26 d4 ed 84 3c 1a d8   .>{.# ...&...<..
    0070 - 9e d4 78 e0 30 f5 66 1f-a4 ec ed 2b 00 f9 b9 1b   ..x.0.f....+....
    0080 - 0f 07 36 91 c7 1f d5 8c-25 c1 3a 40 7d f1 bb b5   ..6.....%.:@}...
    0090 - 0d 2b 7c da 0e cd 56 cc-fc a4 a1 67 a5 86 70 f5   .+|...V....g..p.

    Start Time: 1596803973
    Timeout   : 7200 (sec)
    Verify return code: 18 (self signed certificate)
---
closed

-curves curvelist

Specifies the list of supported curves to be sent by the client. The curve is is ultimately selected by the server. For a list of all curves, use:

./openssl ecparam -list_curves

```
WARNING: can't open config file: /opt/NTAPdfm/openssl/openssl.cnf
  secp112r1 : SECG/WTLS curve over a 112 bit prime field
  secp112r2 : SECG curve over a 112 bit prime field
  secp128r1 : SECG curve over a 128 bit prime field
  secp128r2 : SECG curve over a 128 bit prime field
  secp160k1 : SECG curve over a 160 bit prime field
  secp160r1 : SECG curve over a 160 bit prime field
  secp160r2 : SECG/WTLS curve over a 160 bit prime field
  secp192k1 : SECG curve over a 192 bit prime field
  secp224k1 : SECG curve over a 224 bit prime field
  secp224r1 : NIST/SECG curve over a 224 bit prime field
  secp256k1 : SECG curve over a 256 bit prime field
  secp384r1 : NIST/SECG curve over a 384 bit prime field
  secp521r1 : NIST/SECG curve over a 521 bit prime field
  prime192v1: NIST/X9.62/SECG curve over a 192 bit prime field
  prime192v2: X9.62 curve over a 192 bit prime field
  prime192v3: X9.62 curve over a 192 bit prime field
  prime239v1: X9.62 curve over a 239 bit prime field
  prime239v2: X9.62 curve over a 239 bit prime field
  prime239v3: X9.62 curve over a 239 bit prime field
  prime256v1: X9.62/SECG curve over a 256 bit prime field
  sect113r1 : SECG curve over a 113 bit binary field
  sect113r2 : SECG curve over a 113 bit binary field
  sect131r1 : SECG/WTLS curve over a 131 bit binary field
  sect131r2 : SECG curve over a 131 bit binary field
  sect163k1 : NIST/SECG/WTLS curve over a 163 bit binary field
  sect163r1 : SECG curve over a 163 bit binary field
  sect163r2 : NIST/SECG curve over a 163 bit binary field
  sect193r1 : SECG curve over a 193 bit binary field
  sect193r2 : SECG curve over a 193 bit binary field
  sect233k1 : NIST/SECG/WTLS curve over a 233 bit binary field
  sect233r1 : NIST/SECG/WTLS curve over a 233 bit binary field
  sect239k1 : SECG curve over a 239 bit binary field
  sect283k1 : NIST/SECG curve over a 283 bit binary field
  sect283r1 : NIST/SECG curve over a 283 bit binary field
  sect409k1 : NIST/SECG curve over a 409 bit binary field
  sect409r1 : NIST/SECG curve over a 409 bit binary field
  sect571k1 : NIST/SECG curve over a 571 bit binary field
  sect571r1 : NIST/SECG curve over a 571 bit binary field
  c2pnb163v1: X9.62 curve over a 163 bit binary field
  c2pnb163v2: X9.62 curve over a 163 bit binary field
  c2pnb163v3: X9.62 curve over a 163 bit binary field
```

```
  c2pnb176v1: X9.62 curve over a 176 bit binary field
  c2tnb191v1: X9.62 curve over a 191 bit binary field
  c2tnb191v2: X9.62 curve over a 191 bit binary field
  c2tnb191v3: X9.62 curve over a 191 bit binary field
  c2pnb208w1: X9.62 curve over a 208 bit binary field
  c2tnb239v1: X9.62 curve over a 239 bit binary field
  c2tnb239v2: X9.62 curve over a 239 bit binary field
  c2tnb239v3: X9.62 curve over a 239 bit binary field
  c2pnb272w1: X9.62 curve over a 272 bit binary field
  c2pnb304w1: X9.62 curve over a 304 bit binary field
  c2tnb359v1: X9.62 curve over a 359 bit binary field
  c2pnb368w1: X9.62 curve over a 368 bit binary field
  c2tnb431r1: X9.62 curve over a 431 bit binary field
  wap-wsg-idm-ecid-wtls1: WTLS curve over a 113 bit binary field
  wap-wsg-idm-ecid-wtls3: NIST/SECG/WTLS curve over a 163 bit binary field
  wap-wsg-idm-ecid-wtls4: SECG curve over a 113 bit binary field
  wap-wsg-idm-ecid-wtls5: X9.62 curve over a 163 bit binary field
  wap-wsg-idm-ecid-wtls6: SECG/WTLS curve over a 112 bit prime field
  wap-wsg-idm-ecid-wtls7: SECG/WTLS curve over a 160 bit prime field
  wap-wsg-idm-ecid-wtls8: WTLS curve over a 112 bit prime field
  wap-wsg-idm-ecid-wtls9: WTLS curve over a 160 bit prime field
  wap-wsg-idm-ecid-wtls10: NIST/SECG/WTLS curve over a 233 bit binary field
  wap-wsg-idm-ecid-wtls11: NIST/SECG/WTLS curve over a 233 bit binary field
  wap-wsg-idm-ecid-wtls12: WTLS curvs over a 224 bit prime field
  Oakley-EC2N-3:
      IPSec/IKE/Oakley curve #3 over a 155 bit binary field.
      Not suitable for ECDSA.
      Questionable extension field!
  Oakley-EC2N-4:
      IPSec/IKE/Oakley curve #4 over a 185 bit binary field.
      Not suitable for ECDSA.
      Questionable extension field!
  brainpoolP160r1: RFC 5639 curve over a 160 bit prime field
  brainpoolP160t1: RFC 5639 curve over a 160 bit prime field
  brainpoolP192r1: RFC 5639 curve over a 192 bit prime field
  brainpoolP192t1: RFC 5639 curve over a 192 bit prime field
  brainpoolP224r1: RFC 5639 curve over a 224 bit prime field
  brainpoolP224t1: RFC 5639 curve over a 224 bit prime field
  brainpoolP256r1: RFC 5639 curve over a 256 bit prime field
  brainpoolP256t1: RFC 5639 curve over a 256 bit prime field
  brainpoolP320r1: RFC 5639 curve over a 320 bit prime field
  brainpoolP320t1: RFC 5639 curve over a 320 bit prime field
  brainpoolP384r1: RFC 5639 curve over a 384 bit prime field
  brainpoolP384t1: RFC 5639 curve over a 384 bit prime field
  brainpoolP512r1: RFC 5639 curve over a 512 bit prime field
  brainpoolP512t1: RFC 5639 curve over a 512 bit prime field
[root@netapp sbin]#
```

How to debug ssl server

s_client can be used to debug SSL servers. To connect to an SSL HTTP server the command:

openssl s_client -connect 10.193.113.50:8488

would typically be used (https uses port 443). If the connection succeeds then an HTTP command can be given such as "GET /" to retrieve a web page.

…..

..

GET /

HTTP/1.1 505 HTTP Version Not Supported

Server: libzapid-httpd

Content-Type: text/html

Content-Length: 120

Date: Fri, 07 Aug 2020 12:51:18 GMT

<HTML><HEAD><TITLE>505 HTTP Version Not Supported</TITLE></HEAD><BODY><H1>HTTP Version Not Supported</H1></BODY></HTML>

closed

./openssl s_client -connect 10.193.113.50:8443

WARNING: can't open config file: /opt/NTAPdfm/openssl/openssl.cnf

CONNECTED(00000003)

depth=0 C = aa, ST = aaa, L = aaa, O = ss, OU = ss, CN = d, emailAddress = hara@gmail.com

verify error:num=18:self signed certificate

verify return:1

depth=0 C = aa, ST = aaa, L = aaa, O = ss, OU = ss, CN = d, emailAddress = hara@gmail.com

verify return:1

140439592908464:error:14094438:SSL routines:ssl3_read_bytes:tlsv1 alert internal error:s3_pkt.c:1498:SSL alert number 80

140439592908464:error:140790E5:SSL routines:ssl23_write:ssl handshake failure:s23_lib.c:177:

Certificate chain

0 s:/C=aa/ST=aaa/L=aaa/O=ss/OU=ss/CN=d/emailAddress=hara@gmail.com

i:/C=aa/ST=aaa/L=aaa/O=ss/OU=ss/CN=d/emailAddress=hara@gmail.com

---

Server certificate

-----BEGIN CERTIFICATE-----

MIICmzCCAkWgAwIBAgIJAJ/Y7dtwm71tMA0GCSqGSIb3DQEBCwUAMG4xCzAJBgNV

BAYTAmFhMQwwCgYDVQQIDANhYWExDDAKBgNVBAcMA2FhYTELMAkGA1UECgwCc3Mx

CzAJBgNVBAsMAnNzMQowCAYDVQQDDAFkMR0wGwYJKoZIhvcNAQkBFg5oYXJhQGdt

YWlsLmNvbTAeFw0yMDA4MDcxMjUzNDBaFw0yMTA4MDcxMjUzNDBaMG4xCzAJBgNV
BAYTAmFhMQwwCgYDVQQIDANhYWExDDAKBgNVBAcMA2FhYTELMAkGA1UECgwCc3Mx
CzAJBgNVBAsMAnNzMQowCAYDVQQDDAFkMR0wGwYJKoZIhvcNAQkBFg5oYXJhQGdt
YWlsLmNvbTBcMA0GCSqGSIb3DQEBAQUAA0sAMEgCQQC2TurAS0XPlqWfnhGXGq11
5QWIAzG0HG447PiK3vSd4JPvLSyimNoD+SK9nPAqjNDrsYWafuua0s0skF/Td9KR
AgMBAAGjgcUwgcIwHQYDVR0OBBYEFA9cgPWxXzWeBMFPWf6d/4Kjv402MIGgBgNV
HSMEgZgwgZWAFA9cgPWxXzWeBMFPWf6d/4Kjv402oXKkcDBuMQswCQYDVQQGEwJh
YTEMMAoGA1UECAwDYWFhMQwwCgYDVQQHDANhYWExCzAJBgNVBAoMAnNzMQswCQYD
VQQLDAJzczEKMAgGA1UEAwwBZDEdMBsGCSqGSIb3DQEJARYOaGFyYUBnbWFpbC5j
b22CCQCf2O3bcJu9bTANBgkqhkiG9w0BAQsFAANBAIX+Aob9pRURpdEEgglnLG/D
9T2/TFDH+epgW2RCax1kX/Sz7vZg8XC/0Pl1KKHIdDV3fQV44mlvEWkUfqq0Lz4=
-----END CERTIFICATE-----
subject=/C=aa/ST=aaa/L=aaa/O=ss/OU=ss/CN=d/emailAddress=hara@gmail.com
issuer=/C=aa/ST=aaa/L=aaa/O=ss/OU=ss/CN=d/emailAddress=hara@gmail.com
---
No client certificate CA names sent

SSL handshake has read 764 bytes and written 0 bytes

New, (NONE), Cipher is (NONE)
Server public key is 512 bit
Secure Renegotiation IS supported
Compression: NONE
Expansion: NONE
No ALPN negotiated
SSL-Session:
    Protocol : TLSv1.2
    Cipher   : 0000
    Session-ID:
    Session-ID-ctx:
    Master-Key:
    Key-Arg   : None
    PSK identity: None
    PSK identity hint: None
    Start Time: 1596804933
    Timeout   : 300 (sec)
    Verify return code: 18 (self signed certificate)

How to do dfm ssl server setup:
dfm ssl server setup

Supply the following information:

Key Size (minimum = 512..1024..2048..) [default=512]: 1024

Certificate Duration (days) [default=365]:
Country Name (e.g., 2 letter code): in
State or Province Name (full name): ka
Locality Name (city): bangalore
Organization Name (e.g., company): Netapp
Organizational Unit Name (e.g., section): dfm
Common Name (fully-qualified hostname): netappvm
Email Address: mkomarth@netapp.com
WARNING: can't open config file: /usr/local/ssl/openssl.cnf
Generating a 1024 bit RSA private key
.....++++++
.....++++++
writing new private key to 'C:\Program Files\NetApp\DataFabric Manager\DFM\conf\
server.key'

dfm option set httpsEnabled=Yes"

The httpsEnabled option is currently disabled.
To enable HTTPS, issue "dfm option set httpsEnabled=Yes".
The DataFabric Manager server SSL server has been initialized with new certificates.
You must now restart the http service:

    dfm service stop http
    dfm service start http

C:\Users\Administrator>dfm service stop http
Service: http stopped.

C:\Users\Administrator>dfm service start http
Service: http started.

C:\Users\Administrator>dfm option list httpsenabled
Option Value
--- ---
httpsEnabled No

C:\Users\Administrator>
C:\Users\Administrator>
C:\Users\Administrator>
C:\Users\Administrator>dfm option set httpsEnabled=Yes
Changed HTTPS enabled to Yes.
You must now restart the webui service:

dfm service stop webui
dfm service start webui

You must now restart the http service:

dfm service stop http
dfm service start http

C:\Users\Administratordf>dfm service stop webui
Service: webui stopped.
Also stopped the following dependent services:
http

C:\Users\Administrator>dfm service start webui
Service: webui started.

C:\Users\Administrator>dfm service stop http
Service: http stopped.
Not attempting to stop http because service is stopped already.

C:\Users\Administrator>dfm service start http
Service: http started.

C:\Users\Administrator>
C:\Users\Administrator>dfm option list httpsenabled
Option Value
--- ---
httpsEnabled Yes

C:\Users\Administrator>dfm option set httpsEnabled=Yes
Left option settings unchanged.

DFM commands for plink testing:

dfm host add -N 10.225.39.75
dfm host add -N 10.225.39.76
dfm host list
dfm host diag 131
dfm host diag 132

dfm host set 131 hostlogin=root hostpassword=netapp1!
dfm host set 132 hostlogin=root hostpassword=netapp1!

dfm host set 131 hostndmplogin=root hostndmppassword=netapp1!
dfm host set 132 hostndmplogin=root hostndmppassword=netapp1!

dfm host discover 131
dfm host discover 132

dfm host diag 131
dfm host diag 132

dfm run cmd 131 aggr status
dfm run cmd 131 version
dfm run cmd 195 aggr status
dfm run cmd 131 vol status

Go to Task manager => add " new command line " column under details menu.

Dfm cmd window prompt and task manager window both keep as side by side for viewing the password changes .

Then run the command ' dfm host diag 131 ' or 'dfm run cmd 131 aggr status' and observe the task manager window plink file password showing messages.

DFM LDAP SET UP:

dfm ldap add 172.19.233.90
dfm ldap template netscape
dfm options set ldapEnabled=Yes
dfm option set ldapBaseDN="dc=netapp,dc=local"
dfm option set ldapBindDN="CN=test user (contractor),OU=test,DC=netapp,DC=local"
dfm options set ldapBindPass=mohan@143
dfm options set ldapGID=memberOf
dfm options set ldapMember=member
dfm options set ldapUGID=CN
dfm options set ldapUID=sAMAccountName
dfm options set ldapVersion=3

dfm option set ldapBindDN="CN=test user \28contractor\29,OU=test,DC=netapp,DC=local"

dfm option set ldapBaseDN="dc=netapp,dc=local"

dfm option set ldapBindDN="CN=test user (contractor),OU=test,DC=netapp,DC=local"
dfm options set ldapBindPass=mohan@143
dfm ldap test test mohan@143

DFM PORT 8488:

Port 8488 that provides TLS1.0 connectivity is used for communication with the Netapp Management Console(NMC) –

https://library.netapp.com/ecmdocs/ECMP1217273/html/GUID-ED658F6C-8218-4FE3-A418-82119FF0F90D.html . You can install the NMC locally on the host running OnCommand Unified Manager and close off port 8488 from the firewall. Downside of this is that you will need to access the OCUM host to use the NMC, but this will prevent vulnerable connections to the host. Web UI connections would not be affected as the web service is listening on another port and it is running on the apache server which allows TLS protocol re-configuration.

Where does SSL binary resides in installed directory(DFM)

Linux path: /opt/NTAPdfm/sbin

Windows path: C:\Program Files\NetApp\DataFabric Manager\DFM\bin

Library path linux: /opt/NTAPdfm/lib

Windows: C:\Program Files\NetApp\DataFabric Manager\DFM\bin