# Troubleshooting TLS Problems With ssldump

**George Watkins, 2010-14-10**

## Introduction

Transport Layer Security (TLS) is used to secure network communications between two hosts. TLS largely replaced SSL (Secure Sockets Layer) starting in 1999, but many browsers still provide backwards compatibility for SSL version 3. TLS is the basis for securing all HTTPS communications on the Internet. BIG-IP provides the benefit of being able to offload the encryption and decryption of TLS traffic onto a purpose specific ASIC. This provides performance benefits for the application servers, but also provides an extra layer for troubleshooting when problems arise.

It can be a daunting task to tackle a TLS issue with tcpdump alone. Luckily, there is a utility called ssldump. Ssldump looks for TLS packets and decodes the transactions, then outputs them to the console or to a file. It will display all the components of the handshake and if a private key is provided it will also display the encrypted application data. The ability to fully examine communications from the application layer down to the network layer in one place makes troubleshooting much easier.

**Note:** The user interface of the BIG-IP refers to everything as SSL with little mention of TLS. The actual protocol being negotiated in these examples is TLS version 1.0, which appears as “Version 3.1” in the handshakes. For more information on the major and minor versions of TLS, see the TLS record protocol section of the Wikipedia article.

## Overview of ssldump

I will spare you the man page, but here are a few of the options we will be using to examine traffic in our examples:

```
ssldump -A -d -k <key file> -n -i <capture VLAN> <traffic expression>

-A      Print all fields
-d      Show application data when private key is provided via -k
-k      Private key file, found in /config/ssl/ssl.key/; the key file can be located under client SSL
-n      Do not try to resolve PTR records for IP addresses
-i      The capture VLAN name is the ingres VLAN for the TLS traffic
```

The traffic expression is nearly identical to the tcpdump expression syntax. In these examples we will be looking for HTTPS traffic between two hosts (the client and the LTM virtual server). In this case, the expression will be "host <client IP> and host <virtual server IP> and port 443”. More information on expression syntax can be found in the ssldump and tcpdump manual pages.

*the manual page can be found by typing 'man ssldump' or online here <http://www.rtfm.com/ssldump/Ssldump.html>

## A healthy TLS session

When we look at a healthy TLS session we can see what things should look like in an ideal situation. First the client establishes a TCP connection to the virtual server. Next, the client initiates the handshake with a ClientHello. Within the ClientHello are a number of parameters: version, available cipher suites, a random number, and compression methods if available. The server then responds with a ServerHello in which it selects the strongest cipher suite, the version, and possibly a compression method. After these parameters have been negotiated, the server will send its certificate completing the the ServerHello. Finally, the client will respond with PreMasterSecret in the ClientKeyExchange and each will send a 1 byte ChangeCipherSpec agreeing on their symmetric key algorithm to finalize the handshake. The client and server can now exchange secure data via their TLS session until the connection is closed.

If all goes well, this is what a “clean” TLS session should look like:

```
New TCP connection #1: 10.0.0.10(57677) <-> 10.0.0.20(443)
1 1  0.0011 (0.0011)  C>S  Handshake
      ClientHello
        Version 3.1
        cipher suites
        TLS_DHE_RSA_WITH_AES_256_CBC_SHA
        [more cipher suites]
        TLS_RSA_EXPORT_WITH_RC4_40_MD5
        Unknown value 0xff
        compression methods
                unknown value
                  NULL
1 2  0.0012 (0.0001)  S>C  Handshake
      ServerHello
        Version 3.1
        session_id[0]=

        cipherSuite         TLS_RSA_WITH_AES_256_CBC_SHA
        compressionMethod                   NULL
1 3  0.0012 (0.0000)  S>C  Handshake
      Certificate
1 4  0.0012 (0.0000)  S>C  Handshake
      ServerHelloDone
1 5  0.0022 (0.0010)  C>S  Handshake
      ClientKeyExchange
1 6  0.0022 (0.0000)  C>S  ChangeCipherSpec
1 7  0.0022 (0.0000)  C>S  Handshake
      Finished
1 8  0.0039 (0.0016)  S>C  ChangeCipherSpec
1 9  0.0039 (0.0000)  S>C  Handshake
      Finished
1 10 0.0050 (0.0010)  C>S  application_data
1    0.0093 (0.0000)  S>C  TCP FIN
1    0.0093 (0.0000)  C>S  TCP FIN
```

**Scenario 1: Virtual server missing a client SSL profile**

The client SSL profile defines what certificate and private key to use, a key passphrase if needed, allowed ciphers, and a number of other options related to TLS communications. Without a client SSL profile, a virtual server has no knowledge of any of the parameters necessary to create a TLS session. After you've configured a few hundred HTTPS virtuals this configuration step becomes automatic, but most of us mortals have missed step at one point or another and left ourselves scratching our heads.

We'll set up a test virtual that has all the necessary configuration options for an HTTPS profile, except for the omission of the client SSL profile. The client will open a connection to the virtual on port 443, a TCP connection will be established, and the client will send a 'ClientHello'. Normally the server would then respond with ServerHello, but in this case there is no response and after some period of time (5 minutes is the default timeout for the browser) the connection is closed. This is what the ssldump would look like for a missing client SSL profile:

```
New TCP connection #1: 10.0.0.10(46226) <-> 10.0.0.20(443)
1 1  0.0011 (0.0011)  C>SV3.1(84)  Handshake
      ClientHello
        Version 3.1
```

```
        random[32]=
          4c b6 3b 84 24 d7 93 7f 4b 09 fa f1 40 4f 04 6e
          af f7 92 e1 3b a7 3a c2 70 1d 34 dc 9d e5 1b c8
        cipher suites
        TLS_DHE_RSA_WITH_AES_256_CBC_SHA
        [a number of other cipher suites]
        TLS_RSA_EXPORT_WITH_RC2_CBC_40_MD5
        TLS_RSA_EXPORT_WITH_RC4_40_MD5
        Unknown value 0xff
        compression methods
                unknown value
                  NULL
1    299.9883 (299.9871)  C>S  TCP FIN
1    299.9883 (0.0000)  S>C  TCP FIN
```

**Scenario 2: Client and server do not share a common cipher suite**

This is a common scenario when really old browsers try to connect to servers with modern cipher suites. We have purposely configured our SSL profile to only accept one cipher suite (TLS_RSA_WITH_AES_256_CBC_SHA in this case). When we try connect to the virtual using a 128-bit key, the connection is immediately closed with no ServerHello from the virtual server. The differentiator here, while small, is the quick closure of the connection and the 'TCP FIN' that arises from the server. This is unlike the behavior of the missing SSL profile, because the server initiates the connection teardown and there is no connection timeout. The differences, while subtle, hint at the details of the problem:

```
New TCP connection #1: 10.0.0.10(49342) <-> 10.0.0.20(443)
1 1  0.0010 (0.0010)  C>SV3.1(48)  Handshake
      ClientHello
        Version 3.1
        random[32]=
          4c b7 41 87 e3 74 88 ac 89 e7 39 2d 8c 27 0d c0
          6e 27 da ea 9f 57 7c ef 24 ed 21 df a6 26 20 83
        cipher suites
        TLS_RSA_WITH_AES_128_CBC_SHA
        Unknown value 0xff
        compression methods
                unknown value
                  NULL
1    0.0011 (0.0000)  S>C  TCP FIN
1    0.0022 (0.0011)  C>S  TCP FIN
```

## Conclusion

Troubleshooting TLS can be daunting at first, but an understanding of the TLS handshake can make troubleshooting much more approachable. We cannot exhibit every potential problem in this tech tip. However, we hope that walking through some of the more common examples will give you the tools necessary to troubleshoot other issues as they arise. Happy troubleshooting!

F5 Networks, Inc. | 401 Elliot Avenue West, Seattle, WA 98119 | 888-882-4447 | f5.com

| | | | |
|---|---|---|---|
| F5 Networks, Inc.<br>Corporate Headquarters<br>info@f5.com | F5 Networks<br>Asia-Pacific<br>apacinfo@f5.com | F5 Networks Ltd.<br>Europe/Middle-East/Africa<br>emeainfo@f5.com | F5 Networks<br>Japan K.K.<br>f5j-info@f5.com |