TLSv1.3: Minor Version, Major Changes

PRESENTED BY:
MegaZone, Sr. Security Engineer – F5 SIRT & Don Laursen, Principal Product Manager
With Special Thanks to Dan Strohl, New Product Intro Engineering Lead
SSL/TLS OVERVIEW
History

SSL developed by Netscape
• SSLv1.0 – Never released
• SSLv2.0 – 1995
• SSLv3.0 – 1996

Moved governance to the IETF and renamed TLS
• TLSv1.0 – 1999
• TLSv1.1 – 2006
• TLSv1.2 – 2008
• TLSv1.3 – 2018

TLS 1.3 Draft 26 Supported in TMOS v14.0.0
TLSv1.3 – More like TLSv4.0

• While the changes between TLSv1.0, TLSv1.1, and TLSv1.2 were relatively minor, TLSv1.3 is deceptive.

• There was a debate over TLSv1.3 vs. TLSv2.0 (new major version) vs. TLSv4.0 (to make it clear it was later than SSLv3), and 1.3 won. It maintains the internal numbering – version 0x0304. (SSLv3.4)

• TLSv1.3 is RFC8446 – published just last Friday, 8/10. https://tools.ietf.org/html/rfc8446 This was formerly Draft 28.

• BIG-IP TLSv1.3 support arrived in 14.0.0 – but only Draft 26. Full Draft 28/RFC8446 support will arrive in a future release.
WHAT’S NEW IN TLSV1.3?
Ch-Ch-Changes

- Cleaned up protocol by removing insecure or obsolete features, including compression, renegotiation, EC point format negotiation, and Change Cipher Spec.
- Removed insecure cipher suites – static RSA & Diffie-Hellman ((EC)DH), DSA, RC4, MD5, SHA-224, EXPORT, weak elliptic curves, and more are all forbidden.
- All TLSv1.2 and earlier cipher suites are unsupported.
- All key exchange must be ephemeral ((EC)DHE) or Pre-Shared Key (PSK)
- All supported cipher suites are AEAD (Authenticated Encryption with Associated Data).
The New Hotness

- TLSv1.3 introduces many improvements and new features.
- 1-RTT (Round Trip Time) handshakes improve performance
- 0-RTT support for session resumption or PSK.
- Elliptic Curves are now in the base RFC and new algorithms such as ed25519 and ed448 have been added.
- Version negotiated via an extension – the Protocol Version field is ‘locked’ at 0x0303 (TLSv1.2), though ClientHello may use 0x0301 (TLSv1.0) for the outer envelope.
- Finite Field Diffie-Hellman (DHE) supports key size negotiation.
TLSv1.3 Cipher Suites

- TLSv1.3 changes the way cipher suites are defined, and they now only specify the AEAD algorithm and the hash algorithm to be used with HKDF (HMAC-based Extract-and-Expand Key Derivation Function) – RFC5869.

- Key exchange and authentication are handled separately and are no longer defined in the cipher suite.

- The TLSv1.3 equivalent to ECDHE-RSA-AES256-GCM-SHA384 would be TLS_AES_256_GCM_SHA384.

- TLSv1.2 and earlier cipher suites do not work in TLSv1.3.

- Note all TLSv1.3 cipher suites begin with “TLS13” on BIG-IP.
TLSv1.2 Cipher Suite Format

ECDH-ECDSA-AES256-GCM-SHA384

Key Exchange - Authentication
Symmetric Encryption Algorithm
MAC

Optionally in the protocol (no UI configuration)

elliptic_curves (extension)
... list of Diffie Helman supported curves
(usually derived from the cipher suite)

signature_algorithms (extension)
... list of supported Certificate Validation algorithms
(PKCS#1 unless specified)
Since only DHE and ECDHE are supported key exchanges, that segment is no longer needed, and is defined by the supported_group.

“TLS13” on BIG-IP

String “TLS”

DH Groups in BIG-IP

supported_groups (extension)

elliptic_curves (extension)

… list of Diffie Helman supported curves

Symmetric Encryption Algorithm

TLS-AES256-GCM-SHA384

MAC

signature_algorithms (extension)

… list of supported Certificate Validation algorithms
TLSv1.3 Cipher Suites – Part Deux

• TLSv1.3 cipher suites are included in ‘DEFAULT’ starting in 14.0.0:

```
[root@localhost:Active:Standalone] config # tmm --clientciphers TLSv1_3
      ID  SUITE                  BITS PROT     CIPHER  MAC     KEYX
0:  4865  TLS13-AES128-GCM-SHA256  128  TLS1.3  AES-GCM   SHA256 ?
1:  4866  TLS13-AES256-GCM-SHA384  256  TLS1.3  AES-GCM   SHA384 ?
```

• How does this look when listing all ciphers?

```
[root@localhost:Active:Standalone] config # tmm --clientciphers DEFAULT | tail -n 4
54:  136  DHE-RSA-CAMELLIA256-SHA  256  TLS1.1  CAMELLIA   SHA   EDH/RSA
55:  136  DHE-RSA-CAMELLIA256-SHA  256  TLS1.2  CAMELLIA   SHA   EDH/RSA
56:  4865  TLS13-AES128-GCM-SHA256  128  TLS1.3  AES-GCM   SHA256 ?
57:  4866  TLS13-AES256-GCM-SHA384  256  TLS1.3  AES-GCM   SHA384 ?
```

• Note that KEYX is ‘?’ for TLSv1.3 – Why is that?
TLSv1.3 Key Exchange

• As I mentioned, Key Exchange is no longer defined in the cipher suite. It is now handled by the “supported_groups” and “key_share” extensions.

• “supported_groups” defines the key exchange algorithms that are supported – currently only ECDHE & DHE.

• “key_share” defines the cryptographic parameters associated with the “supported_groups”.

• TLSv1.3 also supports Pre-Shared Keys. PSKs might be exchanged out-of-band.

Note: BIG-IP does not support PSK.
# Big-IP Cipher Rules with TLS 1.3

```plaintext
# list rule TLS_1_3_ONLY
ltm cipher rule TLS_1_3_ONLY {
  cipher TLS13-AES128-GCM-SHA256-TLS13-AES256-GCM-SHA384
  dh groups DEFAULT
  signature algorithms DEFAULT
}
```

# help ltm.ciphers.rule

cipher rule
  Specifies the OpenSSL compatible cipher string.

dh groups groups
  Specifies the allowed named groups, separated by ":". For example: "P256:X25519"
  The available named groups are: P256, P384, X25519 A special keyword, DEFAULT, represents the recommended set of named groups.

signature algorithms signature algorithms
  Specifies the allowed signatures algorithms, separated by ":". For example: "RSA_PKCS1_SHA256:ECDSA_P256_SHA256"
  The available signature algorithms are: DSA-SHA1, DSA-SHA256, DSA-SHA384, DSA-SHA512, ECDSA-SHA1, ECDSA-SHA256, ECDSA-SHA384, ECDSA-SHA512, RSA-PKCS1-SHA1, RSA-PKCS1-SHA256, RSA-PKCS1-SHA384, RSA-PKCS1-SHA512, RSA-PSS-SHA256, RSA-PSS-SHA384, RSA-PSS-SHA512
  A special keyword, DEFAULT, represents the recommended set of signature algorithms.
supported_groups extension

- Was named “elliptic_curves” and contained elliptic curve groups.
- Now contains both elliptic curves and finite field groups
- Custom elliptic curves are no longer supported.
- Servers may send this to the client, but clients must not use it (other than as future reference)
- Called “DH Groups” in BIG-IP

Finite field groups are not yet supported in BIG-IP – planned for a future release

To see available DH Groups, look at the audit field in the “F5 Default” cipher group or help.
key_share extension

• Clients pre-generate keys for the server for the key exchange, and place the keys in the key_share extension.

• Clients may generate as many or as few keys as they like, but any generated must match the advertised supported_groups.

• If a server selects a key exchange cipher that does not have a key_share item, it will send a ServerRetryRequest to the client.

If wondering why a you are seeing a ServerRetryRequest, see if the option returned in the ServerRetryRequest had a key defined in the ClientHello key_share.
signature_algorithms

• signature_algorithm extension
  Used for signatures in certificates

• signature_algorithm_cert extension
  Used for signatures in CertificateVerify messages
  Optional, if not present, signature_algorithm items are used for Verify

• PSS (Probabilistic signature scheme) support
  Is required in TLS 1.3
  Though a client can choose to not request it.

To see available signature_algorithms, look at the audit field in the “F5 Default” cipher group or help.
TLS 1.3 Ciphers

**DH Group (Key Exchange):**
- **DHE**
  - ffdhe2048\(0x0100\)
  - ffdhe3072\(0x0101\)
  - ffdhe4096\(0x0102\)
  - ffdhe6144\(0x0103\)
  - ffdhe8192\(0x0104\)
- **ECDHE**
  - secp256r1\(0x0017\)
  - secp384r1\(0x0018\)
  - secp521r1\(0x0019\)
  - x25519\(0x001D\)
  - x448\(0x001E\)
- **PSK**
- **PSK w/ (EC)DHE**

**Auth:**
- **RSA**
  - rsa_pkcs1_sha256\(0x0401\)
  - rsa_pkcs1_sha384\(0x0501\)
  - rsa_pkcs1_sha512\(0x0601\)
  - rsa_pss_rsa_sha256\(0x0804\)
  - rsa_pss_rsa_sha384\(0x0805\)
  - rsa_pss_rsa_sha512\(0x0806\)
- **ECDSA**
  - ecdsa_secp256r1_sha256\(0x0403\)
  - ecdsa_secp384r1_sha384\(0x0503\)
  - ecdsa_secp521r1_sha512\(0x0603\)
- **EdDSA**
  - ed25519\(0x0807\)
  - ed448\(0x0808\)

**Legacy algorithms**
- rsa_pkcs1_sha1\(0x0201\)
- ecdsa_sha1\(0x0203\)

**Ciphers:**
- **AES-GCM**
  - TLS_AES_128_GCM_SHA256 \(0x1301\)
  - TLS_AES_256_GCM_SHA384 \(0x1302\)
- **CHACHA20-POLY1305**
  - TLS_CHACHA20_POLY1305_SHA256 \(0x1303\)
- **AES-CCM**
  - TLS_AES_128_CCM_SHA256 \(0x1304\)
  - TLS_AES_128_CCM_8_SHA256 \(0x1305\)
### TLS 1.3 Cipher BIG-IP Names

#### DH Groups

<table>
<thead>
<tr>
<th>RFC Name</th>
<th>BIG-IP Name</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Secp256r1</td>
<td>P256</td>
<td>0x0017</td>
</tr>
<tr>
<td>Secp384r1</td>
<td>P384</td>
<td>0x0018</td>
</tr>
<tr>
<td>X25519</td>
<td>X25519</td>
<td>0x001D</td>
</tr>
</tbody>
</table>

#### Cipher Suites

<table>
<thead>
<tr>
<th>RFC Name</th>
<th>BIG-IP Name</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLS_AES_128_GCM_SHA256</td>
<td>TLS13-AES128-GCM-SHA256</td>
<td>0x1301</td>
</tr>
<tr>
<td>TLS_AES_256_GCM_SHA384</td>
<td>TLS13-AES256-GCM-SHA384</td>
<td>0x1302</td>
</tr>
</tbody>
</table>

#### Signature Algorithms

<table>
<thead>
<tr>
<th>RFC Name</th>
<th>BIG-IP Name</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>rsa_pkcs1_sha256</td>
<td>RSA-PKCS1-SHA256</td>
<td>0x0401</td>
</tr>
<tr>
<td>rsa_pkcs1_sha384</td>
<td>RSA-PKCS1-SHA384</td>
<td>0x0501</td>
</tr>
<tr>
<td>rsa_pkcs1_sha512</td>
<td>RSA-PKCS1-SHA512</td>
<td>0x0601</td>
</tr>
<tr>
<td>rsa_pss_rsaes_sha256</td>
<td>RSA-PSS-SHA256</td>
<td>0x0804</td>
</tr>
<tr>
<td>rsa_pss_rsaes_sha384</td>
<td>RSA-PSS-SHA384</td>
<td>0x0805</td>
</tr>
<tr>
<td>rsa_pss_rsaes_sha512</td>
<td>RSA-PSS-SHA512</td>
<td>0x0806</td>
</tr>
<tr>
<td>ecdsa_secp256r1_sha256</td>
<td>ECDSA-SHA256</td>
<td>0x0403</td>
</tr>
<tr>
<td>ecdsa_secp384r1_sha384</td>
<td>ECDSA-SHA384</td>
<td>0x0503</td>
</tr>
<tr>
<td>ecdsa_secp521r1_sha512</td>
<td>ECDSA-SHA512</td>
<td>0x0603</td>
</tr>
</tbody>
</table>
SECRET HANDSHAKES
TLS 1.2 Handshake

Client Hello

Client Key Exchange
Change Cipher Spec

Client Finished

Encrypted Data

Server Hello
Certificate
Server Key Exchange
Server Hello Done

Change Cipher Spec
Server Finished

Encrypted Data
TLS 1.3 Handshake

Client Hello
key_share (ext)

Session ID is deprecated

Server Hello
key_share (ext)
Certificate

Client Finished

Encrypted Data

Encrypted Data
For backward compatibility reasons with middleboxes the HelloRetryRequest message uses the same structure as the ServerHello, but with Random set to the special value of the SHA-256 of "HelloRetryRequest":

```
CF 21 AD 74 E5 9A 61 11 BE 1D 8C 02 1E 65 B8 91 C2 A2 11 16 7A BB 8C 5E 07 9E 09 E2 C8 A8 33 9C
```
TLS 1.3 Handshake (Cookie)

- **Client Hello**
  - key_share (ext)
- **Client Hello**
  - key_share (ext)
  - Cookie (extension)
- **Client Finished**
  - Encrypted Data
- **Hello Retry Request**
  - Cookie (extension)
- **Server Hello**
  - Certificate
  - key_share (ext)

BIG-IP does not currently send TLS cookies
TLS 1.3 Handshake (+Certificate Messages)

Client Hello
key_share
signature_algorithms
psk_key_exchange_modes
pre_shared_key

ServerHello
key_share
pre_shared_key

Certificate
Certificate Verify
Finished
Application Data

Certificate request
Certificate
CertificateVerify
Finished
Application Data

Application Data
Application Data
TLS 1.3 Handshake (Everything encrypted)

Client Hello
  key_share
  signature_algorithms
  psk_key_exchange_modes
  pre_shared_key

ServerHello
  key_share
  pre_shared_key

Certificate
CertificateVerify
Finished
Application Data

Certificate
Certificate
CertificateVerify
Finished
Application Data

Application Data
Application Data

Certificate request
Certificate
CertificateVerify
Finished
Application Data
Application Data
TLS 1.3 Handshake (Everything encrypted)

Client Hello
  key_share
  signature_algorithms
  psk_key_exchange_modes
  pre_shared_key

ServerHello
  key_share
  pre_shared_key

Certificate
CertificateVerify
Finished
ChangeCipherSpec
Application Data

Certificate request
Certificate
CertificateVerify
Finished
Application Data
Application Data
Application Data

ChangeCipherSpec may be present for compatibility reasons, but is not used.
TLS 1.3 Handshake (Session Resumption)

Client Hello
  key_share
  signature_algorithms
  psk_key_exchange_modes
  pre_shared_key

ServerHello
  key_share
  pre_shared_key

Certificate
CertificateVerify
Finished
Application Data

Certificate request
Certificate
CertificateVerify
Finished
Application Data
Application Data

Initial Handshake
TLS 1.3 Handshake (Session Resumption)

Client Hello
- key_share
- pre_shared_key

ServerHello
- key_share
- pre_shared_key

Application Data

Finished

Application Data

Resumed Handshake

BIG-IP does not support session resumption at this point

Application Data

Application Data

Application Data

Application Data
TLS 1.3 Handshake  (early_data, 0-RTT)

Client Hello
key_share (ext)
Early Encrypted Data

Potentially vulnerable to a replay attack… but only for the first packet.

Client Finished
Encrypted Data

Server Hello
key_share (ext)
Certificate
Encrypted Data

Encrypted Data

Encrypted Data

This must be specifically enabled by the application, not the TLS stack.

0-RTT with early-data requires a shared key – only works with PSK or resumption

BIG-IP does not support early_data at this point.
TLS 1.3 Handshake (0-RTT Cipher Negotiation)

Client Hello
key_share (ext)

Encrypted Data

Client Hello
key_share (ext)

Server Hello
Certificate
key_share (ext)

Encrypted Data

Hello Retry Request

Client is not allowed to try re-sending it with the next ClientHello.

BIG-IP does not support early_data at this point.
Compatibility Mode

• For compatibility with middleboxes (think BIG-IPs!), a compatibility mode is suggested (Appendix D.4 in draft 28)
  • Session ID SHOULD be provided (its not used by TLS 1.3)
  • Dummy change_cipher_spec should be sent immediately after handshake message.

• Negotiation of compatibility mode:
  • If the client sends the session ID, it is assumed they wish compatibility mode:
    • The server MUST return the session ID
    • The server must send the dummy change_cipher_specs.
      • If the client does not send a session ID
    • Server session ID and dummy change_cipher_spec messages are optional.

BIG-IP does not retain Session ID in 1.3 connections at this point.
Handshake Messages Removed

- ServerHelloDone (now finished extension)
- ClientHelloDone (now finished extension)
- ChangeCipherSpec (still seen for compatibility)
- ClientKeyExchange (now key_share extension)
- ServerKeyExchange (now key_share extension)
HIDDEN MEANINGS
HelloRetryRequest – The ServerHello Disguise

• HelloRetryRequest looks exactly like ServerHello – because it is.
• It is identified by a special, fixed value in the ‘Random’ field:
  CF 21 AD 74 E5 9A 61 11 BE 1D 8C 02 1E 65 B8 91 C2 A2 11 16 7A
  BB 8C 5E 07 9E 09 E2 C8 A8 33 9C
• That’s the SHA-256 of ‘HelloRetryRequest’.
• This is done for compatibility with legacy ‘middleboxes’ – aka proxies and the like – which might not handle a new message type.
(Not So) Random Protection

• The ServerHello ‘Random’ field is also used for protection against downgrade attacks.

• If a TLSv1.3 server negotiates TLSv1.2, the last eight bytes of Random MUST be set to "44 4F 57 4E 47 52 44 01".

• If a TLSv1.3 server negotiates TLSv1.1 or below, the last eight bytes of Random MUST be set to "44 4F 57 4E 47 52 44 00".

• A TLSv1.3 client which receives a ServerHello requesting a lower protocol version will check Random and MUST abort the connection if either value is present.

BIG-IP does not currently use the required values – this will be corrected in an upcoming release.
SUPPORTABILITY
Key Points for Troubleshooting

• TLSv1.3 is supported in ClientSSL only for now, not yet in ServerSSL – this will be added in an upcoming release.
• Even in ClientSSL, TLSv1.3 is disabled with ‘no-tlsv1.3’ included by default in the profile ‘Options’. Remove this to enable TLSv1.3 support.
• If you do this in ServerSSL you will receive a warning:
  (tmos)# modify ltm profile server-ssl ssls ssl options {}
There were warnings:
/Common/sssl, TLSv1.3 is not yet supported in ServerSSL profile; it is recommended to keep 'no-tlsv1.3' option enabled.
Enabling TLS 1.3 on BIG-IP

In the ClientSSL Profile…

REMOVE the “No TLSv1.3” option.

By default this option is there (so, TLS 1.3 is disabled)
<table>
<thead>
<tr>
<th>Library</th>
<th>Used By</th>
<th>18</th>
<th>19</th>
<th>20</th>
<th>21</th>
<th>22</th>
<th>23</th>
<th>24</th>
<th>25</th>
<th>26</th>
<th>27</th>
<th>28</th>
</tr>
</thead>
<tbody>
<tr>
<td>F5 TMM</td>
<td>BIG-IP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>NSS</td>
<td>Firefox</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>miTLS</td>
<td>Microsoft</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BoringSSL</td>
<td>Google/Chrome</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wireshark</td>
<td>Wireshark</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>picotls</td>
<td>H2O Server</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Secure Transport</td>
<td>Apple (Mac)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>sChannel</td>
<td>Windows (Edge+)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OpenSSL</td>
<td>Most Servers / Tools</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>wolfSSL</td>
<td>MySQL</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GnuTLS</td>
<td>Synology</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>???</td>
<td>Opera</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tlslite-ng</td>
<td>Python Lib</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>SwiftTLS</td>
<td>Apple</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
</tr>
</tbody>
</table>

* Mostly based on: https://github.com/tlswg/tls13-spec/wiki/Implementations
Further Limitations in 14.0.0

• No Session Resumption yet – upcoming release.
• No Client Cert Auth – Client SSL profiles configured to ‘Request’ or ‘Require’ Client Cert Auth will be automatically downgraded to TLSv1.2 – upcoming release.
• No FFDHE (Finite Field Diffie-Hellman Ephemeral) support – future release.
• No CHACHA20_POLY1305_SHA256 support – future release.
• No 0-RTT Early Data Support – future roadmap.
• 14.0.0 is not suitable for production use of TLSv1.3 is it only supports Draft 26, but can be used to test to prepare for future deployment.
Trouble with Troubleshooting

- No more non-ephemeral key exchange (RSA, DH, ECDH)
- Think about that for a moment…What does this mean for ssldump? What does this mean for Proxy SSL?
- They do not work with TLSv1.3!
- Proxy-ssl-passthrough mode only.
- **K16700**: Decrypting SSL traffic using the SSL::sessionsecret iRules command - does not work with TLSv1.3!
But Wait, There’s More

• All of the handshake messages after ServerHello are now encrypted.
• Furthermore, the new EncryptedExtensions message allows a number of previously visible extensions to be encrypted.
• This is going to further complicate troubleshooting as much less information may be available in a TLSv1.3 pcap unless it is decrypted.
• So, how are we going to be able to troubleshoot this?
Capturing the Secrets – On the BIG-IP

- **iRules to the rescue:**

```bash
ltm rule tls_secret {
    when CLIENTSSL_HANDSHAKE {
        if {[[SSL::cipher version] == "TLSv1.3"]} {
            log local0. "CLIENT_HANDSHAKE_TRAFFIC_SECRET [SSL::clientrandom] [SSL::tls13_secret client hs]"
            log local0. "SERVER_HANDSHAKE_TRAFFIC_SECRET [SSL::clientrandom] [SSL::tls13_secret server hs]"
            log local0. "CLIENT_TRAFFIC_SECRET_0 [SSL::clientrandom] [SSL::tls13_secret client app]"
            log local0. "SERVER_TRAFFIC_SECRET_0 [SSL::clientrandom] [SSL::tls13_secret server app]"
        } else {
            log local0. "RSA Session-ID:[SSL::sessionid] Master-Key:[SSL::sessionsecret]"
        }
    }
}
```

*Caveat: The iRule only logs this information once the handshake is complete. It won't be useful for troubleshooting handshake problems where it isn't successful.*
Capturing the Secrets – On The Client

• Another way to capture the secrets is by setting a file path in the SSLKEYLOGFILE environment variable before opening Chrome or Firefox.
• When this is set Chrome and Firefox will log all SSL/TLS session keys into the specified file.
• This works on Windows, MacOS, and Linux.
• You can use this file in Wireshark just like a PMS file.
• File format: https://developer.mozilla.org/en-US/docs/Mozilla/Projects/NSS/Key_Log_Format
## Decrypting Content

<table>
<thead>
<tr>
<th>Technique</th>
<th>&lt;= TLS 1.2 (no PFS)</th>
<th>&lt;= TLS 1.2 (w/PFS)</th>
<th>TLS 1.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>ssldump</td>
<td>✓</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>tcpdump (keys from SSL::session_secret)</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
</tr>
<tr>
<td>tcpdump (keys from SSL::tls13_secret iRule)</td>
<td>x</td>
<td>x</td>
<td>✓</td>
</tr>
<tr>
<td>tcpdump (keys from client using KEYLOGFILE env var)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
New Cipher Rule Help:

OPTIONS

cipher rule
  Specifies the OpenSSL compatible cipher string.

description
  User defined description.

dh-groups groups
  Specifies the allowed named groups, separated by ":". For example:
  "P256:X25519"

The available named groups are: P256, P384, X25519 A special
keyword, DEFAULT, represents the recommended set of named groups.
More Help!

- New Cipher Rule Help - Continued:

  signature-algorithms

  Specifies the allowed signatures algorithms, separated by ":". For example: "RSA_PKCS1_SHA256:ECDSA_P256_SHA256"

  The available signature algorithms are: DSA-SHA1, DSA-SHA256, DSA-SHA384, DSA-SHA512, ECDSA-SHA1, ECDSA-SHA256, ECDSA-SHA384, ECDSA-SHA512, RSA-PKCS1-SHA1, RSA-PKCS1-SHA256, RSA-PKCS1-SHA384, RCS-PKCS1-SHA512, RSA-PSS-SHA256, RSA-PSS-SHA384, RSA-PSS-SHA512

  A special keyword, DEFAULT, represents the recommended set of signature algorithms.
TLSv1.3 Draft Protocol Versioning

- While the final RFC for TLSv1.3 uses the 0x0304 version number, the drafts use a version number in the format of 0x7f## - where the ‘##’ is the draft version in hex.

Handshake Protocol: Server Hello
Handshake Type: Server Hello (2)
Length: 78
Version: TLS 1.3 (draft 26) (0x7f1a)
Random: 7658c15d9ff90360afb7f1296ff91af8df04d35d52c74a4e...
Cipher Suite: TLS_AES_128_GCM_SHA256 (0x1301)
Extensions Length: 40
Extension: key_share (len=36) *snip*
TLS 1.3 ROADMAP
## The state of F5 SSL/TLS

### SSL strategy and roadmap

<table>
<thead>
<tr>
<th>Today</th>
<th>August 2018</th>
<th>BIG-IP 14.1</th>
<th>BIG-IP 15.0</th>
</tr>
</thead>
</table>
| • TLS 1.3 tolerance  
• F5 cipher suite builder  
• Dynamic CA bundle update  
• SSL visibility  
• C3D – phase one | • TLS 1.3 (Draft 26)  
• Curve25519 | • TLS 1.3 (RFC 8446)  
• ChaCha20-Poly1305 | • TLS 1.3  
• DH 2048  
• 0-RTT  
• TLS 1.3 for SSLO (Client Side and Server Side)  
• C3D – phase two |
## TLS 1.3 support

<table>
<thead>
<tr>
<th>Library</th>
<th>Used By</th>
<th>18</th>
<th>19</th>
<th>20</th>
<th>21</th>
<th>22</th>
<th>23</th>
<th>24</th>
<th>25</th>
<th>26</th>
<th>27</th>
<th>28</th>
</tr>
</thead>
<tbody>
<tr>
<td>F5 TMM</td>
<td>BIG-IP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NSS</td>
<td>Firefox</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>miTLS</td>
<td>Microsoft</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BoringSSL</td>
<td>Google/Chrome</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wireshark</td>
<td>Wireshark</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>picotls</td>
<td>H2O Server</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Secure Transport</td>
<td>Apple (Mac)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sChannel</td>
<td>Windows (Edge+)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>OpenSSL</td>
<td>Most Servers / Tools</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>wolfSSL</td>
<td>MySQL</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GnuTLS</td>
<td>Synology</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>???</td>
<td>Opera</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>tlslite-ng</td>
<td>Python Lib</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>SwiftTLS</td>
<td>Apple</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>
The state of F5 SSL/TLS
Client Certificate Constrained Delegation

Model 1: Local Delegate

Model 2: Remote Delegate
Questions?
F5 Security Incident Response Team

For emergency security response — call F5 Support!

24x7x365 Security Experts
Global availability
Included with your support contract

Report vulnerabilities to f5sirt@f5.com
Thank You