Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 4, 2018

M. Green
Cryptography Engineering LLC
R. Droms
Interisle Consulting
R. Housley
Vigil Security, LLC
P. Turner
Venafi
S. Fenter

July 3, 2017

Data Center use of Static Diffie-Hellman in TLS 1.3
draft-green-tls-static-dh-in-tls13-01

Abstract

Unlike earlier versions of TLS, current drafts of TLS 1.3 have instead adopted ephemeral-mode Diffie-Hellman and elliptic-curve Diffie-Hellman as the primary cryptographic key exchange mechanism used in TLS. This document describes an optional configuration for TLS servers that allows for the use of a static Diffie-Hellman private key for all TLS connections made to the server. Passive monitoring of TLS connections can be enabled by installing a corresponding copy of this key in each monitoring device.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Unlike earlier versions of TLS, current drafts of TLS 1.3 [I-D.ietf-tls-tls13] do not provide support for the RSA handshake -- and have instead adopted ephemeral-mode Diffie-Hellman (DHE) and elliptic-curve Diffie-Hellman (ECDHE) as the primary cryptographic key exchange mechanism used in TLS.

While ephemeral (EC) Diffie-Hellman is in nearly all ways an improvement over the TLS RSA handshake, the use of these mechanisms complicates certain enterprise settings. Specifically, the use of ephemeral ciphersuites is not compatible with current enterprise network monitoring tools such as Intrusion Detection Systems (IDS) and application monitoring systems, which leverage the current TLS RSA handshake passively monitor intranet TLS connections made between

endpoints under the enterprise's control. This traffic includes TLS connections made from enterprise network security devices (firewalls) and load balancers at the edge of the enterprise network to internal enterprise TLS servers. It does not include TLS connections traveling over the external Internet.

Such monitoring of the enterprise network is ubiquitous and indispensable in some industries. This monitoring is required for effective and safe operation of enterprise networks. Loss of this capability may slow adoption of TLS 1.3.

This document describes an optional configuration for TLS servers that is compatible with the TLS 1.3 ephemeral ciphersuites without precluding enterprise network monitoring. This configuration allows for the use of a static (EC) Diffie-Hellman private key for all TLS connections made to the server. Passive monitoring of TLS connections can be enabled by installing a corresponding copy of this key in each authorized monitoring device.

An advantage of this proposal is that it can be implemented using software modifications to the TLS server and enterprise network monitoring tools, without the need to make changes to TLS client implementations.

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

This document introduces the term "static (elliptic curve) Diffie-Hellman ephemeral", generally written as "static (EC)DHE", to refer to long-lived finite field or elliptic curve Diffie-Hellman keys or key pairs that will be used with the TLS 1.3 ephemeral ciphersuites to negotiate traffic keys for multiple TLS sessions.

For clarity, this document also introduces the term "ephemeral (elliptic curve) Diffie-Hellman ephemeral", generally written as "ephemeral (EC)DHE", to denote finite field or elliptic curve Diffie-Hellman keys or key pairs that will be used with the TLS 1.3 ephemeral ciphersuites to negotiate traffic keys for a single TLS sessions.

1.2. ASN.1

The Cryptographic Message Syntax (CMS) [RFC5652] and asymmetric key packages [RFC5958] are generated using ASN.1 [X680], which uses the

Basic Encoding Rules (BER) and the Distinguished Encoding Rules (DER) [X690].

2. Enterprise Out-of-band TLS Decryption Architecture

This document describes the use of a static (elliptic-curve) Diffie-Hellman (static (EC)DHE) private key by servers for use in TLS 1.3 sessions internal to an enterprise network where network monitoring is required. In Figure 1, the Web Servers use a static (EC)DHE key pair with the standard TLS 1.3 handshake for connections from the Load Balancer, and the Back-End Services use static (EC)DHE for connections from the Web Servers. The Load Balancer uses ephemeral (EC)DHE key pairs with the standard TLS 1.3 handshake for connections from external Browsers over the Internet, to provide Forward Secrecy on those connections that are exposed to third-party monitoring. Internally, the static (EC)DHE keys are provided to authorized TLS Decrypter devices, such as intrusion detection systems, application monitoring systems or network packet capture devices.

```
                                   ********************************
                                  *                                *
                                 *            +--------+            *
                                *   TLS       |  Web   |             *
                               * Termination  + Server +              *
                              *      |       /|        |\              *
    +---------+  +----------+ * +----|-----+/ +--------+ \+----------+ *
    |         |  |          | * |   Load   +              + Back-end | *
    | Browser +--+ Internet |-*-+ Balancer |              |  Server  | *
    |         |  |          | * |    |     +              +          | *
    +---------+  +----------+ * +----------+\ +--------+ /+----------+ *
                              *      |      .\|  Web   |/.             *
                              *            .  + Server +  .            *
                              *            .  |        |  .            *
                              *            .  +--------+  .            *
                              *            .              .            *
                              *            .   --------   .            *
                              *             . /  TLS   \ .             *
                              *              | Decrypter|              *
                               *              \        /              *
                                *              --------              *
                                 *                                  *
                                  *** Enterprise Network Boundary **

                                     |
    <------ Ephemeral (EC)DHE ------>|<-------- Static (EC)DHE -------->
                                     |
```

Figure 1: Enterprise TLS Decryption Architecture

3. Enterprise Requirements for Passive (out-of-band) TLS Decryption

Enterprise networks based on this architecture have operational requirements for traffic monitoring and ex post facto analysis for purposes of:

- Application troubleshooting and performance analysis
- Fraud monitoring
- Security, including intrusion detection, malware detection, confidential data exfiltration and layer 7 DDoS protection
- Audit compliance
- Customer Experience Monitoring

Specific requirements to meet the listed operational requirements include:

- o TLS decryption for network security monitoring tools must be done in real time with no gaps in decryption.

- o The solution must be able to decrypt passively captured pcap traces.

- o The solution must scale to handle thousands of TLS sessions/sec.

- o Key material must be preserved for back-in-time analysis. The period for key retention depends upon local policy, reflecting operational, security and compliance requirements.

- o Key material must be encrypted during network transit

- o The solution must not negatively impact the enterprise infrastructure (servers, network, etc.)

- o The solution must be able to decrypt the session when a TLS session is reused. This may involve the use of a TLS decryption appliance.

- o The solution must be able to decrypt in a physical data center, in a virtual environment, and in a cloud.

4. Summary of the Existing Diffie-Hellman Handshake

In TLS 1.3, servers exchange keys using two primary modes, DHE and ECDHE. In a simplified view of the full handshake, the following steps occur:

1. The client generates an ephemeral public and private key, and transmits the public key within a "key_share" message, along with a random nonce (ClientHello.random).
2. The server generates an ephemeral public and private key, and transmits the public key within a "key_share" message, along with a random nonce (ServerHello.random).
3. The two parties now calculate a shared (EC)DHE secret by combining the other party's ephemeral public key with their own ephemeral private key.
4. A series of traffic and handshake keys is derived by combining this shared secret with various inputs from the handshake, including the ClientHello.random and ServerHello.random.
5. Data encryption is performed using the shared secret.

5. Using static (EC)DHE on the server

The proposal embodied in this draft modifies the standard TLS handshake summarized above in the following ways:

For each elliptic curve (and FF-DH parameter length) supported by the server, the server is provisioned with a static (EC)DHE private/public key pair. This key pair may be either:

* generated at server installation, and rotated at periodic intervals appropriate for any long-term server key,

* generated at a central key management server and distributed (in a secure encrypted form) to the appropriate endpoint servers.

All steps of the original handshake proceed as above, with the following modification to server behavior. Step (2) proceeds as follows:

2. The server transmits the static public key within a "key_share" message, along with a random nonce (ServerHello.random).

6. Key Representation

The Asymmetric Key Package [RFC5958] MUST be used to transfer the centrally managed Diffie-Hellman key pair. The key package contains at least one Diffie-Hellman key pair. Each Diffie-Hellman key pair is associated with a set of attributes, including the key validity period for that Diffie-Hellman key pair.

OneAsymmetricKey is defined in Section 2 of [RFC5958]. The fields are used as follows:

o version MUST be set to v2, which has an integer value of 1.

o privateKeyAlgorithm MUST be set to the algorithm identifier of the Diffie-Hellman key pair. For convenience, some popular algorithm identifiers are listed in Figure 2.

o privateKey MUST be set to the Diffie-Hellman private key encoded as an OCTET STRING.

o attributes MUST be included even though the field is optional. The set of attributes MUST include the key validity period attribute defined in Section 15 of [RFC7906]. Other attributes MAY be included as well.

- publicKey MUST be included even though the field is optional. It MUST be set to the Diffie-Hellman public key, encoded as a BIT STRING. This is the same BIT STRING that would be included in an X.509 certificate [RFC5280] for this public key.

```
+------------------------------------------------------------------+
|                                                                  |
| Finite Field Diffie-Hellman                                      |
|  object identifier: { 1 2 840 10046 2 1 }                        |
|  parameter encoding: DomainParameters, Section 2.3.3 of [RFC3279] |
|  private key encoding: INTEGER                                   |
|  public key encoding: INTEGER                                    |
|                                                                  |
| Elliptic Curve Diffie-Hellman                                    |
|  object identifier: { 1 3 132 1 12 }                             |
|  parameter encoding: ECParameters, Section 2.1.2 of [RFC5480]    |
|                        (MUST use the namedCurve CHOICE)          |
|  private key encoding: ECPrivateKey, Section 3 of [RFC5915]      |
|  public key encoding:  ECPoint, Section 2.2 of [RFC5480]         |
|                                                                  |
+------------------------------------------------------------------+
```

Figure 2: Popular Diffie-Hellman Algorithm Identifiers

The CMS protecting content types [RFC5652][RFC5083] can be used to provide authentication and confidentiality protection for the Asymmetric Key Package:

- SignedData can be used to apply a digital signature to the Asymmetric Key Package.

- EncryptedData can be used to encrypt the Asymmetric Key Package with previously distributed symmetric encryption key.

- EnvelopedData can be used to encrypt the Asymmetric Key Package, where the sender and the receiver establish a symmetric encryption key using Diffie-Hellman key agreement.

- AuthEnvelopedData can be used to protect the Asymmetric Key Package where the sender and the receiver establish a symmetric authenticated encryption key using Diffie-Hellman key agreement.

7. TLS Static DH Key (TSK) Protocol

The TLS Static DH Key (TSK) Protocol is used in cases where the Diffie-Hellman keys are centrally managed. The two main roles in the TSK protocol are "key manager" and "key consumer". Key consumers can

be TLS servers or TLS decrypters. The key manager generates, distributes, and tracks static (EC)DHE keys used by key consumers. TSK messaging is based on HTTPS [RFC2818]. Keys are transferred as Asymmetric Key Packages [RFC5958], using the profile in Section 6 of this document.

```
                    --------------       -----------------
                    | TLS server |-------|  key manager  |
                    --------------       -----------------
                           |                     |
                           |                     |
                           |                     |
                           |             -----------------
                           |------------>| TLS decrypter |
                           |             -----------------
                           |
                           |
                    --------------
                    | TLS client |
                    --------------
```

Figure 3: TSK protocol components

The key manager can push keys to key consumers:

```
        TLS server               key manager             TLS decrypter
            |                         |                        |
            |                         |--                      |
            |                         |  \ Generate            |
            |                         |  / key pair            |
            |                         |<-                      |
            |                         |                        |
            |                         |----------------------->|
            |                         |      Push key pair     |
            |<------------------------|                        |
            |       Push key pair     |                        |
```

Figure 4: TSK protocol push model

Alternatively, key consumers can request (or pull) keys from the key manager.

```
        TLS server               key manager             TLS decrypter
            |                         |                        |
            |                         |--                      |
            |                         |  \ Generate            |
            |                         |  / key pair            |
            |                         |<-                      |
            |                         |                        |
            |                         |<-----------------------|
            |                         |    Request key pair    |
            |------------------------>|                        |
            |     Request key pair    |                        |
```

Figure 5: TSK protocol pull model

7.1. Key Push

An HTTPS-based TSK push is composed of the appropriate HTTP headers, followed by the binary value of the BER (Basic Encoding Rules) encoding of the Asymmetric Key Package.

The Content-Type header MUST be application/cms [RFC7193] if the Asymmetric Key Package is encrypted with CMS [RFC6032]. The Content-Type header MUST be application/pkcs8 if the Asymmetric Key Package is transferred in plain text (within the encrypted HTTPS stream).

7.2. Key Request

A key consumer may request a key by providing a fingerprint [RFC6234] of the public key. The key manager is responsible for determining if the key consumer is authorized to receive a copy of the key being requested.

Example with plain text Asymmetric Key Package:

```
GET /tsk/key/PublicKeyFingerprint
Accept: application/pkcs8
```

Example with CMS encrypted and/or signed Asymmetric Key Package:

```
GET /tsk/key/PublicKeyFingerprint
Accept: application/cms
```

The response to the TSK push is composed of the appropriate HTTP headers, followed by the binary value of the BER (Basic Encoding Rules) encoding of the Asymmetric Key Package.

The Content-Type header MUST be application/cms [RFC7193] if the Asymmetric Key Package is encrypted with CMS [RFC6032]. The Content-Type header MUST be application/pkcs8 if the Asymmetric Key Package is transferred in plain text (within the encrypted HTTPS stream).

8. Alternative Solutions for Enterprise Monitoring and Troubleshooting

- Export of ephemeral keys

- Export of decrypted traffic from TLS proxy devices at the edge of the enterprise network

- Placement of TLS proxies in the enterprise network

- Reliance on TCP/IP headers not encrypted by TLS

- Reliance on application/server logs

- Doing troubleshooting and malware analysis at the endpoint.

- Adding a TCP or UDP extension to provide the information needed to do packet analysis.

9. Weaknesses of Alternative Solutions

Export of ephemeral keys: Scale - In a large enterprise there will be billions of ephemeral keys to export and manage. There will also be difficulty in transporting these keys to real time tools that need decrypted packets. The complexity of the solution is a problem that adds risk.

Export of decrypted traffic from TLS proxy devices: Decrypted traffic at only the edge of the network is not adequate for the enterprise requirements listed above (troubleshooting, network security monitoring, etc...)

TLS proxies in the network: Inline TLS proxies will not scale to the number of decryption points needed within an enterprise. Each inline proxy adds cost, latency, and production risk.

Reliance on TCP/IP headers: IP and/or TCP headers are not adequate for the enterprise requirements listed above. Troubleshooters must be able to find transactions in a pcap trace, identified by markers like userids, session ids, URLs, and time stamps. Threat Detection teams must be able to look for Indicators of Compromise in the payload of packets, etc.

Reliance on Application/server logs: Logging is not adequate for the enterprise requirements listed above. Code developers cannot anticipate every possible problem and put a log message in just the right place. There are billions of lines of code in a data center, and it's not scalable to try and improve logging.

Troubleshooting and malware analysis at the endpoint: Endpoints don't have the robustness to do their own workload and handle the burden of the various enterprise requirements listed above. These requirements would include always-on full packet capture at the endpoint with no packet drops.

Adding TCP/UDP extensions: An important part of troubleshooting, network security monitoring, etc. is analysis of the application-specific payload of the packet. It is not possible to anticipate ahead of time, among thousands of unique applications, which fields in the application payload will be important.

10. Security considerations

We now consider the security implications of the change described above:

1. The shift from fully-ephemeral (EC)HDE to static (EC)DHE affects the security properties offered by the TLS 1.3 handshake by eliminating the Forward Secrecy property provided by the server. If a server is compromised and the private key is stolen, then an attacker who observes any TLS handshake (even one that occurred prior to the compromise) performed with this static (EC)DHE key pair will be able to recover session traffic encryption keys and will be able to decrypt traffic.

2. As long as the server static secret key is not compromised, the resulting protocol will provide strong cryptographic security, as long as the Diffie-Hellman parameters (e.g., finite-field group or elliptic curve) are correctly generated and provide security at a sufficient cryptographic security level.

3. A flaw in the generation of finite-field Diffie-Hellman parameters or the use of an insecure implementation could leak some bits of the static secret key over time. This risk is not present in ephemeral DH implementations. Implementers should use care to avoid such pitfalls.

Thus the modification described in Section 10 represents a deliberate weakening of some security properties. Implementers who choose to include this capability should carefully consider the risks to their

infrastructure of using a handshake without Forward Secrecy. Static (EC)DHE key pairs should be rotated regularly.

11. IANA Considerations

This document contains no actions for IANA.

12. Acknowledgements

This modification to TLS was initially suggested by Hugo Krawczyk.

13. Normative References

[I-D.ietf-tls-tls13]
Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", draft-ietf-tls-tls13-20 (work in progress), April 2017.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, DOI 10.17487/RFC2818, May 2000, <http://www.rfc-editor.org/info/rfc2818>.

[RFC3279] Bassham, L., Polk, W., and R. Housley, "Algorithms and Identifiers for the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 3279, DOI 10.17487/RFC3279, April 2002, <http://www.rfc-editor.org/info/rfc3279>.

[RFC5083] Housley, R., "Cryptographic Message Syntax (CMS) Authenticated-Enveloped-Data Content Type", RFC 5083, DOI 10.17487/RFC5083, November 2007, <http://www.rfc-editor.org/info/rfc5083>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <http://www.rfc-editor.org/info/rfc5280>.

[RFC5480] Turner, S., Brown, D., Yiu, K., Housley, R., and T. Polk, "Elliptic Curve Cryptography Subject Public Key Information", RFC 5480, DOI 10.17487/RFC5480, March 2009, <http://www.rfc-editor.org/info/rfc5480>.

[RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, DOI 10.17487/RFC5652, September 2009, <http://www.rfc-editor.org/info/rfc5652>.

[RFC5915] Turner, S. and D. Brown, "Elliptic Curve Private Key Structure", RFC 5915, DOI 10.17487/RFC5915, June 2010, <http://www.rfc-editor.org/info/rfc5915>.

[RFC5958] Turner, S., "Asymmetric Key Packages", RFC 5958, DOI 10.17487/RFC5958, August 2010, <http://www.rfc-editor.org/info/rfc5958>.

[RFC6032] Turner, S. and R. Housley, "Cryptographic Message Syntax (CMS) Encrypted Key Package Content Type", RFC 6032, DOI 10.17487/RFC6032, December 2010, <http://www.rfc-editor.org/info/rfc6032>.

[RFC6234] Eastlake 3rd, D. and T. Hansen, "US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)", RFC 6234, DOI 10.17487/RFC6234, May 2011, <http://www.rfc-editor.org/info/rfc6234>.

[RFC7193] Turner, S., Housley, R., and J. Schaad, "The application/cms Media Type", RFC 7193, DOI 10.17487/RFC7193, April 2014, <http://www.rfc-editor.org/info/rfc7193>.

[RFC7906] Timmel, P., Housley, R., and S. Turner, "NSA's Cryptographic Message Syntax (CMS) Key Management Attributes", RFC 7906, DOI 10.17487/RFC7906, June 2016, <http://www.rfc-editor.org/info/rfc7906>.

[X680] ITU-T, "Information technology -- Abstract Syntax Notation One (ASN.1): Specification of basic notation", ITU-T Recommendation X.680, 2015.

[X690] ITU-T, "Information technology -- ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)", ITU-T Recommendation X.690, 2015.

Authors' Addresses

Matthew Green
Cryptography Engineering LLC
4506 Roland Ave
Baltimore, MD 21210
USA

Email: mgreen@cryptographyengineering.com

Ralph Droms
Interisle Consulting

Email: rdroms.ietf@gmail.com

Russ Housley
Vigil Security, LLC
918 Spring Knoll Drive
Herndon, VA 20170
USA

Email: housley@vigilsec.com

Paul Turner
Venafi
175 East 400 South, Suite 300
Salt Lake City, UT 84111
USA

Email: paul.turner@venafi.com

Steve Fenter

Email: steven.fenter58@gmail.com