TLS Working Group
Internet-Draft
Intended status: Experimental
Expires: May 14, 2015

A. Langley
Google
N. Modadugu
Independent
B. Moeller
Google

November 10, 2014

Transport Layer Security (TLS) False Start
draft-bmoeller-tls-falsestart-01

Abstract

This document specifies an optional behavior of TLS implementations, dubbed False Start. It affects only protocol timing, not on-the-wire protocol data, and can be implemented unilaterally. The TLS False Start feature leads to a latency reduction of one round trip for certain handshakes.

Status of This Memo

Copyright Notice

Table of Contents

1. Requirements Notation 2
2. Introduction 2
3. False Start Compatibility 5
4. Client-side False Start 5
5. Server-side False Start 6
6. Security Considerations 7
6.1. Symmetric Cipher 7
6.2. Protocol Version 8
6.3. Key Exchange and Client Certificate Type 8
7. Acknowledgments 9
8. IANA Considerations 9
9. References 9
9.1. Normative References 9
9.2. Informative References 10
Appendix A. Implementation Notes 10
Authors' Addresses 10

1. Requirements Notation

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 RFC 2119 [RFC2119].

2. Introduction

A full TLS handshake as specified in [RFC5246] requires two full protocol rounds (four flights) before the handshake is complete and the protocol parties may begin to send application data. Thus, using TLS can add a latency penalty of two network round-trip times for application protocols in which the client sends data first, such as HTTP [RFC2616]. An abbreviated handshake (resuming an earlier TLS session) is complete after three flights, thus adding just one round-trip time if the client sends application data first.

```
      Client                                               Server

      ClientHello                  -------->
                                                      ServerHello
                                                     Certificate*
                                               ServerKeyExchange*
                                              CertificateRequest*
                                   <--------      ServerHelloDone
      Certificate*
      ClientKeyExchange
      CertificateVerify*
      [ChangeCipherSpec]
      Finished                     -------->
                                               [ChangeCipherSpec]
                                   <--------             Finished
      Application Data             <------->     Application Data
```

Figure 1 [RFC5246]. Message flow for a full handshake

```
   Client                                               Server

   ClientHello                   -------->
                                                   ServerHello
                                            [ChangeCipherSpec]
                                 <--------            Finished
   [ChangeCipherSpec]
   Finished                      -------->
   Application Data              <------->    Application Data
```

Figure 2 [RFC5246]. Message flow for an abbreviated handshake

This document describes a technique that alleviates the latency burden imposed by TLS: the TLS False Start. If certain conditions are met, application data can be sent when the handshake is only partially complete -- i.e., when the sender has sent its own "ChangeCipherSpec" and "Finished" messages (thus having updated its TLS Record Protocol write state as negotiated in the handshake), but has yet to receive the other side's "ChangeCipherSpec" and "Finished" messages. (By section 7.4.9 of [RFC5246], each party would have to delay sending application data until it has received and validated the other side's "Finished" message.) This achieves an improvement of one round-trip time

o for full handshakes if the client sends application data first,

o for abbreviated handshakes if the server sends application data first.

Accordingly, the latency penalty for using TLS with HTTP can be kept at one round-trip time regardless of whether a full handshake or an abbreviated handshake takes place.

In a False Start, when a party sends application data before it has received and verified the other party's "Finished" message, there are two possible outcomes:

o The handshake completes successfully: Once both "Finished" messages have been received and verified, this retroactively validates the handshake. In this case, the transcript of protocol data carried over the transport underlying TLS will look as usual, apart from the different timing.

o The handshake fails: If a party does not receive the other side's "Finished" message, or if the "Finished" message's contents are not correct, the handshake never gets validated. This means that an attacker may have removed, changed, or injected handshake

messages. In this case, data has been sent over the underlying transport that would not have been sent without the False Start.

The latter scenario makes it necessary to restrict when a False Start is allowed, as described in this document. Section 3 considers basic requirements for using False Start. Section 4 and Section 5 specify the behavior for clients and servers, respectively, referring to important security considerations in Section 6.

3. False Start Compatibility

TLS False Start as described in detail in the subsequent sections, if implemented, is an optional feature.

A TLS implementation (not necessarily offering the False Start option itself) is defined to be "False Start compatible" if it tolerates receiving TLS records on the transport connection early, before the protocol has reached the state to process these. To successfully use False Start in a TLS connection, the other side has to be False Start compatible. Out-of-band knowledge that the peer is False Start compatible may be available, e.g. if this is mandated by specific application profile standards. As discussed in Appendix A, the requirement for False Start compatibility does not pose a hindrance in practice.

4. Client-side False Start

This section specifies a change to the behavior of TLS client implementations in full TLS handshakes.

When the client has sent its "ChangeCipherSpec" and "Finished" messages, its default behavior following [RFC5246] is to not send application data until it has received the server's "ChangeCipherSpec" and "Finished" messages, which completes the handshake. With the False Start protocol modification, the client MAY send application data earlier (under the new Cipher Spec) if each of the following conditions is satisfied:

o The application layer has requested the TLS False Start option.

o The symmetric cipher defined by the cipher suite negotiated in this handshake has been whitelisted for use with False Start according to the Security Considerations in Section 6.1.

o The protocol version chosen by ServerHello.server_version has been whitelisted for use with False Start according to the Security Considerations in Section 6.2.

- o The key exchange method defined by the cipher suite negotiated in this handshake has been whitelisted for use with False Start according to the Security Considerations in Section 6.3.

- o In the case of a handshake with client authentication, the client certificate type has been whitelisted for use with False Start according to the Security Considerations in Section 6.3.

The rules for receiving application data from the server remain unchanged.

Note that the TLS client cannot infer the presence of an authenticated server until all handshake messages have been received. With False Start, unlike with the default handshake behavior, applications are able to send data before this point has been reached: from an application point of view, being able to send data does not imply that an authenticated peer is present. Accordingly, it is recommended that TLS implementations allow the application layer to query whether the handshake has completed.

5. Server-side False Start

This section specifies a change to the behavior of TLS server implementations in abbreviated TLS handshakes.

When the server has sent its "ChangeCipherSpec" and "Finished" messages, its default behavior following [RFC5246] is not to send application data until it has received the client's "ChangeCipherSpec" and "Finished" messages, which completes the handshake. With the False Start protocol modification, the server MAY send application data earlier (under the new Cipher Spec) if each of the following conditions is satisfied:

- o The application layer has requested the TLS False Start option.

- o The symmetric cipher defined by the cipher suite of the session being resumed has been whitelisted for use with False Start according to the Security Considerations in Section 6.1.

The rules for receiving application data from the client remain unchanged.

Note that the TLS server cannot infer the presence of an authenticated client until all handshake messages have been received. With False Start, unlike with the default handshake behavior, applications are able to send data before this point has been reached: from an application point of view, being able to send data does not imply that an authenticated peer is present. Accordingly,

it is recommended that TLS implementations allow the application layer to query whether the handshake has completed.

6. Security Considerations

In a TLS handshake, the "Finished" messages serve to validate the entire handshake. These messages are based on a hash of the handshake so far processed by a PRF keyed with the new master secret (serving as a MAC), and are also sent under the new Cipher Spec with its keyed MAC, where the MAC key again is derived from the master secret. The protocol design relies on the assumption that any server and/or client authentication done during the handshake carries over to this. While an attacker could, for example, have changed the cipher suite list sent by the client to the server and thus influenced cipher suite selection (presumably towards a less secure choice) or could have made other modifications to handshake messages in transmission, the attacker would not be able to round off the modified handshake with a valid "Finished" message: every TLS cipher suite is presumed to key the PRF appropriately to ensure unforgeability. Once the handshake has been validated by verifying the "Finished" messages, this confirms that the handshake has not been tampered with, thus bootstrapping secure encryption (using algorithms as negotiated) from secure authentication.

Using False Start interferes with this approach of bootstrapping secure encryption from secure authentication, as application data may have already been sent before "Finished" validation confirms that the handshake has not been tampered with -- so there is generally no hope to be sure that communication with the expected peer is indeed taking place during the False Start. Instead, the security goal is to ensure that if anyone at all can decrypt the application data sent in a False Start, this must be the legitimate peer: while an attacker could be influencing the handshake (restricting cipher suite selection, modifying key exchange messages, etc.), the attacker should not be able to benefit from this. The TLS protocol already relies on such a security property for authentication -- with False Start, the same is needed for encryption. This motivates the following rules.

6.1. Symmetric Cipher

Clients and servers MUST NOT use the False Start protocol modification in a handshake unless the cipher suite uses a symmetric cipher that is considered cryptographically strong.

Implementations may have their own classification of ciphers (and may additionally allow the application layer to provide a classification), but generally only symmetric ciphers with an

effective key length of 128 bits or more can be considered strong. Also, various ciphers specified for use with TLS are known to have cryptographic weaknesses regardless of key length (none of the ciphers specified in [RFC4492] and [RFC5246] can be recommended for use with False Start). The AES_128_GCM_SHA256 or AES_256_GCM_SHA384 ciphers specified in [RFC5288] and [RFC5289] can be considered sufficiently strong for most uses. Implementations that support additional cipher suites have to be careful to whitelist only suitable symmetric ciphers; if in doubt, False Start should not be used with a given symmetric cipher.

While an attacker can change handshake messages to force a downgrade to a less secure symmetric cipher than otherwise would have been chosen, this rule ensures that in such a downgrade attack no application data will be sent under an insecure symmetric cipher. With respect to server-side False Start, if a client has negotiated a TLS session using weak symmetric cryptography, this rule prevents attackers from seeing the server encrypt more data under this session than normally (if an attacker makes up a "ClientHello" message asking to resume such a session, no False Start will happen).

6.2. Protocol Version

Clients MUST NOT use the False Start protocol modification in a handshake unless the protocol version chosen by ServerHello.server_version has been whitelisted for this use.

Generally, implementations should whitelist only the protocol version(s) for which they would not send TLS_FALLBACK_SCSV [downgrade-scsv].

The details of nominally identical cipher suites can differ between protocol versions, so this reinforces Section 6.1.

6.3. Key Exchange and Client Certificate Type

Clients MUST NOT use the False Start protocol modification in a handshake unless the cipher suite uses a key exchange method that has been whitelisted for this use. Furthermore, when using client authentication, clients MUST NOT use the False Start protocol modification unless the client certificate type has been whitelisted for this use.

Implementations may have their own whitelists of key exchange methods and client certificate types (and may additionally allow the application layer to specify whitelists). Generally, out of the options from [RFC5246] and [RFC4492], the following whitelists are recommended:

- Key exchange methods: DHE_RSA, ECDHE_RSA, DHE_DSS, ECDHE_ECDSA

- Client certificate types: rsa_sign, dss_sign, ecdsa_sign (or no client authentication)

However, if an implementation that supports only key exchange methods from [RFC5246] and [RFC4492] does not support any of the above key exchange methods, all of its supported key exchange methods can be whitelisted for False Start use. Care is required with any additional key exchange methods or client certificate types, as these may not have similar properties.

The recommended whitelists are such that if cryptographic algorithms suitable for forward secrecy would possibly be negotiated, no False Start will take place if the current handshake fails to provide forward secrecy. (Forward secrecy can be achieved using ephemeral Diffie-Hellman or ephemeral Elliptic-Curve Diffie-Hellman; there is no forward secrecy when a using key exchange method of RSA, RSA_PSK, DH_DSS, DH_RSA, ECDH_ECDSA, or ECDH_RSA, or a client certificate type of rsa_fixed_dh, dss_fixed_dh, rsa_fixed_ecdh, or ecdsa_fixed_ecdh.) As usual, the benefits of forward secrecy may need to be balanced against efficiency, and accordingly even implementations that support the above key exchange methods might whitelist further key exchange methods and client certificate types from [RFC5246] and [RFC4492].

7. Acknowledgments

The authors wish to thank Wan-Teh Chang, Ben Laurie, Eric Rescorla, and Brian Smith for their input.

8. IANA Considerations

None.

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC4492] Blake-Wilson, S., Bolyard, N., Gupta, V., Hawk, C., and B. Moeller, "Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS)", RFC 4492, May 2006.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[RFC5288] Salowey, J., Choudhury, A., and D. McGrew, "AES Galois Counter Mode (GCM) Cipher Suites for TLS", RFC 5288, August 2008.

[RFC5289] Rescorla, E., "TLS Elliptic Curve Cipher Suites with SHA-256/384 and AES Galois Counter Mode (GCM)", RFC 5289, August 2008.

9.2. Informative References

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[downgrade-scsv]
Moeller, B. and A. Langley, "TLS Fallback Signaling Cipher Suite Value (SCSV) for Preventing Protocol Downgrade Attacks", Work in Progress, draft-ietf-tls-downgrade-scsv-01, November 2014.

Appendix A. Implementation Notes

TLS False Start is a modification to the TLS protocol, and some implementations that conform to [RFC5246] may have problems interacting with implementations that use the False Start modification. If the peer uses a False Start, application data records may be received directly following the peer's "Finished" message, before the TLS implementation has sent its own "Finished" message. False Start compatibility as defined in Section 3 ensures that these records with application data will simply remain buffered for later processing.

A False Start compatible TLS implementation does not have to be aware of the False Start concept, and is certainly not expected to detect whether a False Start handshake is currently taking place: thanks to transport layer buffering, typical implementations will be False Start compatible without having been designed for it.

Authors' Addresses

Adam Langley
Google Inc.
345 Spear St
San Francisco, CA 94105
USA

Email: agl@google.com

Nagendra Modadugu
Independent

Email: nagendra@cs.stanford.edu

Bodo Moeller
Google Switzerland GmbH
Brandschenkestrasse 110
Zurich 8002
Switzerland

Email: bmoeller@acm.org