TLS Working Group
Internet-Draft
Intended status: Informational
Expires: November 14, 2015

V. Krasnov
CloudFlare Inc.

May 13, 2015

Transport Layer Security (TLS) Jump Start
draft-vkrasnov-tls-jumpstart-00

Abstract

This document specifies an optional behavior of TLS implementation called Jump Start. It alters the way the initial Client and Server handshake messages reach their destination, but not the protocol data, and can be implemented unilaterally. The TLS Jump Start feature leads to a latency reduction of one round trip for all handshakes (on average).

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on November 14, 2015.

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

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 [RFC2119].

2. Introduction

A full TLS handshake over TCP [RFC0793] as specified in [RFC5246] first requires establishing a TCP connection. The TCP connection establishment requires a full round (two flights) before the TCP handshake is complete and the client can send the first TLS handshake message. The TLS handshake itself 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) [RFC5246] is complete after three flights, thus adding just one round-trip time if the client sends application data first.

Figure 1: Message flow for a TCP handshake followed by a full TLS 1.2 handshake

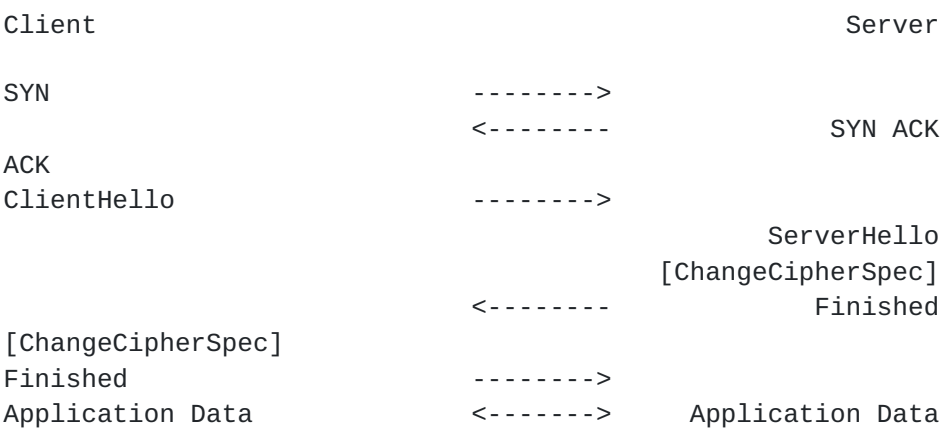


Figure 2: Message flow for a TCP handshake followed by an abbreviated TLS 1.2 handshake

This document describes a technique that alleviates the latency burden imposed by TLS: the TLS Jump Start. The TLS Jump Start technique, hides the latency of the TCP protocol handshake, by sending the initial client/server messages over the UDP protocol [RFC0768] prior to the establishment of a TCP connection.

Under optimal conditions the technique saves a full round worth of latency for any existing and future TLS protocols, regardless of which side sends the data first. The method does not change the data transferred over the wire, but rather the timing and delivery protocol. For an abbreviated handshake it is possible to have a TLS connection in place with almost zero latency over the simple TCP handshake.

In TLS Jump Start, the first round takes place over the UDP protocol in full, while subsequent rounds, if required, continue over the TCP connection.

- Under TLS 1.2 the client would send the initial ClientHello message over UDP.
- Under TLS 1.2 the server would respond with ServerHello, optional Certificate, ServerKeyExchange, CertificateRequest and finally ServerHelloDone over UDP
- For an abbreviated handshake the server would send the ServerHello and Finished messages over UDP

3. Jump Start Compatibility

TLS Jump Start, described below, is an optional feature that is backwards compatible with existing implementations.

If the ClientHello message sent over UDP is lost, does not reach the server in time, or is ignored by the server, then the client will not get a response over UDP from the server and will assume the server does not support this extension. In the case where server is known or suspected to not support this option, a new ClientHello will be sent over TCP.

4. Client-side Jump Start

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

When the client attempts to open a Jump Start connection, it MUST first send the ClientHello message over the UDP protocol to the server.

After, or simultaneously, with the ClientHello, and independently of the server response the client MUST initiate a TCP connection with the server. The ports used by the TCP and UDP SHOULD have the same number.

After or during the establishment of the TCP, the client MUST check if the server responded over the UDP protocol. If the server responded, and the response is complete (i.e. no packets of the response were lost), the client MAY use the handshake data to continue the handshake over the open TCP connection, starting with the ClientKeyExchange message or the optional client Certificate message. If the response is partial the client MUST initiate a new TLS handshake by sending a new ClientHello message, with a new client random_bytes value. If there was no response the client MAY wait for a possible response via UDP for a short period of time. After that the client MUST assume the server does not support Jump Start and initiate a new TLS handshake by sending a new ClientHello message over the TCP channel.

5. Server-side Jump Start

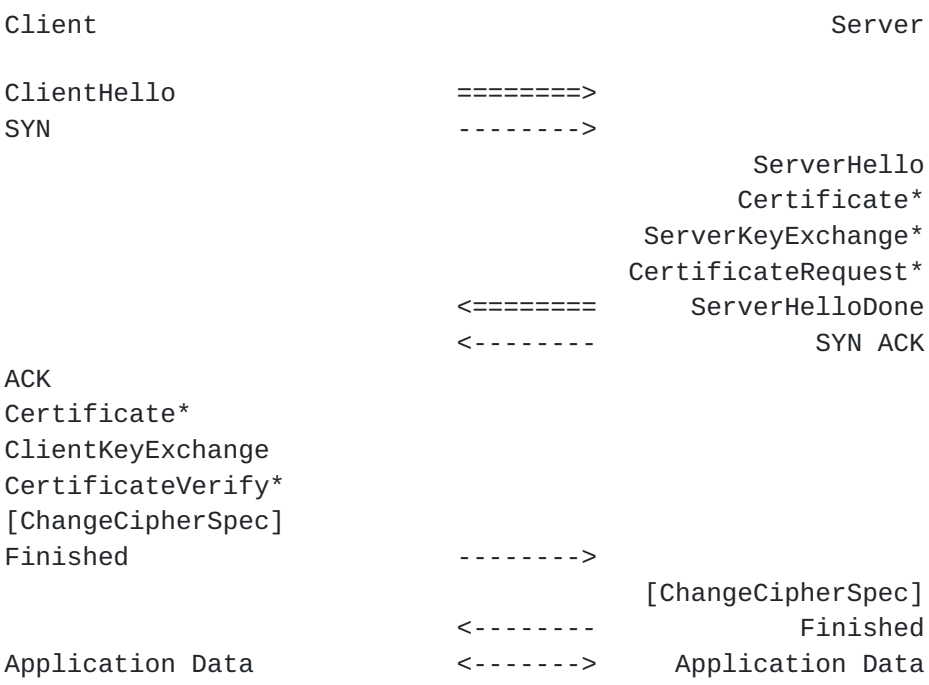
This section specifies a change to the behavior of TLS server implementations.

A server supporting the Jump Start technique, MUST listen for incoming ClientHello messages over the same UDP port number it uses for TCP TLS connections.

When a ClientHello reaches the server over the UDP protocol, the server MUST determine the source of the message by looking at the IP:PORT pair. If a TCP connection for the given source is open, the server MUST ignore the message. The server MUST keep a history of incoming ClientHello messages with the corresponding response messages. It MAY limit the history to a certain timeout period. If an entry exists in the history for a ClientHello for a given IP address (on any port), the server MUST ignore the new ClientHello.

If the ClientHello belongs to a new connection, the server MUST respond with the appropriate response - ServerHello with the optional Certificate, ServerKeyExchange, CertificateRequest and the ServerHelloDone messages for the full handshake, or with ServerHello and Finished messages for the abbreviated handshake. The server MUST then put the handshake state in the history.

When a new TCP connection is opened on the appropriate port, the server will wait for the first message from the client. If the first message is ClientHello, the server will perform a legacy handshake. The server MAY check the history for a previous ClientHello over UDP for a given source, and remove the relevant from history if it exists. If the first message is ClientKeyExchange or Certificate or Finished, the server MUST retrieve the appropriate handshake state from the history. If the retrieval failed, it indicates an error. Otherwise the server MUST continue the handshake process using the stored state, first removing the entry from history.



The exact order in which the TCP and UDP information arrives is not guaranteed.

Figure 3: Message flow for a full Jump Start handshake

The exact order in which the TCP and UDP information arrives is not guaranteed.

Figure 4: Message flow for an abbreviated Jump Start handshake

<-------> Indicates data sent over TCP

<========> Indicates data sent over UDP

6. Security considerations

The integrity of the handshake is not affected by the delivery method. The same security considerations and guarantees applicable to TLS apply here as well.

The use of a UDP protocol makes the technique vulnerable to reflection-amplification attacks similar to other UDP protocols. However due to the fact a single server will only reply once for a given IP, the amplification is very limited. Also coordinating an attack between many servers is difficult, when the network latency between the servers and the attack destination is not uniform.

It is possible to further reduce the risk of such attack, by requiring the UDP ClientHello message to include a large amount of redundant data. The server will check the size of the ClientHello, and respond with a response proportionate to the original message by a certain factor.

7. IANA considerations

None.

8. Implementation considerations

Given the nature of the UDP protocol, it might be desirable to add some kind of an out-of-order packet processing on the client side, in case the server response is split into several UDP packets. The protocol is not designed to deal with UDP packet loss. In case of an unreliable network, it is assumed that the TCP will be a better alternative.

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.

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.

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

[RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980.

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.

Author's Address

Vlad Krasnov
CloudFlare Inc.
Birchin Court, 20 Birchin Lane
London EC3V 9DU
UK

Email: vlad@cloudflare.com