MPTCP
Internet-Draft
Intended status: Informational
Expires: April 18, 2013

C. Paasch, Ed.
O. Bonaventure
UCLouvain

October 15, 2012

Securing the MultiPath TCP handshake with external keys

draft-paasch-mptcp-ssl-00

Abstract

Multipath TCP currently relies on the exchange of keys in clear during the initial handshake to authenticate the establishment of additional subflows. This document proposes a variant of the Multipath TCP handshake that allows Multipath TCP to reuse keys negotiated by the Application layer protocol above it such as SSL/TLS to authenticate the establishment of additional subflows.

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction

Multipath TCP is an extension to TCP that enables hosts to use multiple paths to exchange data for a single connection. [I-D.ietf-mptcp-multiaddressed] describes the current design of the Multipath TCP protocol. The design of Multipath TCP has been influenced by various factors including the backward compatibility with regular TCP, the fallback to TCP when middleboxes interfere with the Multipath TCP options, ... The design of Multipath TCP has also been affected by security requirements. The security threats against Multipath TCP are documented in [RFC6181]. Multipath TCP aims at being no worse than TCP from a security viewpoint. Other approaches such as [I-D.bittau-tcp-crypt] or [RFC5925] have been proposed to reduce the vulnerability of TCP to attacks. Multipath TCP currently addresses the security threats identified in [RFC6181] by exchanging keys during the handshake for the initial subflow. These keys are then used to generate HMACs to authenticate the establishment of subsequent TCP subflows. Exchanging keys in clear during the initial handshake has obvious shortcomings from a security viewpoint. However, some application-layer protocols like SSL/TLS or ssh already negotiate a shared key between the end-points. In this document we propose a modification to the handshake used by Multipath TCP for the initial and subsequent subflows that enables Multipath TCP to rely on an application-supplied key to authenticate the establishment of the subflows.

2. Connection initiation

The handshake of the initial subflow is a small variation to the handshake of [I-D.ietf-mptcp-multiaddressed] or draft-paasch-mptcp-lowoverhead-00. The header of the MP_CAPABLE option of these two MPTCP-versions has the format as shown in the below figure.

```
                    1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------+---------------+-------+-------+---------------+
|     Kind      |    Length     |Subtype|Version|A|B|C|D|E|F|G|H|
+---------------+---------------+-------+-------+---------------+
```

Header of the MP_CAPABLE option

Figure 1

We propose to use the B bit in this option to indicate whether the host that sent the MP_CAPABLE option will use an application supplied key to authenticate the additional subflows or not. When the B bit is set, it indicates that the authentication key is supplied by the application. If the B bit has not been set in both directions, the authentication mechanism is used as defined by the MPTCP version ([I-D.ietf-mptcp-multiaddressed] or draft-paasch-mptcp-lowoverhead-00).

In MPTCP version 0, even if the B bit is set the end-hosts still have to generate a key that fulfills the requirements as defined in MPTCP version 0. This is necessary to handle the case where the client supports the B bit, but the server not yet. For a more in-depth analysis of this kind of deployment scenario, have a look at Section 5.

By using the same handshake as draft-paasch-mptcp-lowoverhead-00, the proposed handshake can also benefit from the lower overhead for generating the token and thus the faster establishment of the initial subflow.

3. Multipath TCP API

The proposed mechanism requires an interaction between the application and the MPTCP layer. This can be achieved by the means of socket options. Two socket options are necessary:

- MPTCP_ENABLE_APP_KEY : This socket option tells the socket layer that an application supplied key will be used to secure the establishement of additional subflows. This socket option MUST be used before establishing the initial subflow, or before starting to listen on a socket to accept new connections. When this socket option is used, the MP_CAPABLE option is sent with the "B"-bit set to 1.

- MPTCP_KEY : This socket option allows the application to provide a key to the MPTCP layer. Both end-points MUST use this socket option in order to allow the MPTCP-layer to create new subflows. It is up to the application to negotiate the key between the end-points. E.g., in the case of SSL/TLS, the key can be a hash of the shared secret that has been negotiated with the SSL exchange. Separate documents will describe in details how applications such as TLS or SSH can pass a shared secret to Multipath TCP by using this option.

4. Starting a new subflow

The handshake for the establishment of a new subflow is similar to the one specified in [I-D.ietf-mptcp-multiaddressed]. There are two important differences. First, the HMAC is computed by using the keys provided by the application. Second, the token and the client's random number are included inside the third ack to allow stateless operation of the passive opener of an additional subflow.

```
         Host A                                  Host B
        ----------                             ----------
        Address A2                             Address B2
        ----------                             ----------
             |                                       |
             |   SYN + MP_JOIN(Token-B, R-A)         |
             |-------------------------------------->|
             |                                       |
             |   SYN/ACK + MP_JOIN(HMAC-B, R-B)      |
             |<--------------------------------------|
             |                                       |
             |  ACK + MP_JOIN(Token-B, R-A, HMAC-A)  |
             |-------------------------------------->|

        HMAC-A = HMAC(Key, Msg=(R-A+R-B))
        HMAC-B = HMAC(Key, Msg=(R-B+R-A))
```

Handshake of a new subflow.

Figure 2

In order to allow the Token-B and R-A inside the third ack, the HMAC-A must also be a truncated version of the 160-bit HMAC-SHA1. Thus, HMAC-A is the truncated (leftmost 128 bits) of the HMAC as shown in Figure 2.

The message-format of the MP_JOIN-option in the SYN and the SYN/ACK is the same as in [I-D.ietf-mptcp-multiaddressed]. As the third ACK includes the Token and the random nonce, the MP_JOIN message format of the third ack is as show in Figure 3. The length of the MP_JOIN-option in the third ACK is 28 bytes. There remains thus enough space to insert the timestamp option in the third ACK.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------+---------------+-------+-------+---------------+
|     Kind      |    Length     |Subtype|     |B|  Address ID   |
+---------------+---------------+-------+-------+---------------+
|                                                               |
|                Sender's Truncated HMAC (128 bits)             |
|                                                               |
+---------------------------------------------------------------+
|                Sender's Random Number (32 bits)               |
+---------------------------------------------------------------+
|                     Receiver's Token (32 bits)                |
+---------------------------------------------------------------+
```

Format of the MP_JOIN-option

Figure 3

The semantics of the backup-bit "B" and the Address ID are the same as in [I-D.ietf-mptcp-multiaddressed].

5. Deployment

This proposed mechanism assumes that the application uses new socket-options to provide the key to the MPTCP-layer. Thus, the first requirement for deploying this MPTCP handshake is that the TLS/SSL-layer has been modified. There may of course be scenarios, where the client is supporting the proposed solution, but the server not. Thus, the client sends out the MP_CAPABLE with the B bit set, but the server replies without enabling the B bit. Upon reception of the SYN/ACK, it is up to the client's policy how to react. It can either continue with the negotiated version of MPTCP but without using the key from the application or fallback to regular TCP.

The applications will have to pass the shared key to the MPTCP-layer by the means of a socket-option. It may be that the client's application has already done the call to the socket-option but the server's application not yet. The server will receive a SYN with the MP_JOIN-option, without knowing the key. In that case the server should silently drop the SYN. The TCP retransmission mechanism on the client-side will retransmit the SYN after the initial RTO expired (after 1 second). And the server's application potentially will have finally set the key via the socket-option.

6. Security Considerations

It is recommended that the applications do not pass the plain shared key to the MPTCP layer. They should rather pass a hash of their shared secret to the MPTCP layer. These security considerations will be discussed in documents that describe how applications such as TLS/SSL or SSH can interact efficiently with Multipath TCP.

7. Informative References

[I-D.bittau-tcp-crypt]
Bittau, A., Boneh, D., Hamburg, M., Handley, M., Mazieres, D., and Q. Slack, "Cryptographic protection of TCP Streams (tcpcrypt)", draft-bittau-tcp-crypt-03 (work in progress), September 2012.

[I-D.ietf-mptcp-api]
Scharf, M. and A. Ford, "MPTCP Application Interface Considerations", draft-ietf-mptcp-api-05 (work in progress), April 2012.

[I-D.ietf-mptcp-multiaddressed]
Ford, A., Raiciu, C., Handley, M., and O. Bonaventure, "TCP Extensions for Multipath Operation with Multiple Addresses", draft-ietf-mptcp-multiaddressed-10 (work in progress), October 2012.

[RFC5925] Touch, J., Mankin, A., and R. Bonica, "The TCP Authentication Option", RFC 5925, June 2010.

[RFC6181] Bagnulo, M., "Threat Analysis for TCP Extensions for Multipath Operation with Multiple Addresses", RFC 6181, March 2011.

Authors' Addresses

Christoph Paasch (editor)
UCLouvain
Place Sainte Barbe, 2
Louvain-la-Neuve, 1348
BE

Email: christoph.paasch@uclouvain.be

Olivier Bonaventure
UCLouvain
Place Sainte Barbe, 2
Louvain-la-Neuve, 1348
BE

Email: olivier.bonaventure@uclouvain.be