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

C. Paasch, Ed.
O. Bonaventure
UCLouvain

October 15, 2012

MultiPath TCP Low Overhead
draft-paasch-mptcp-lowoverhead-00

Abstract

This document describes a low overhead connection establishment mechanism for Multipath TCP. Its goal is to reduce the computational overhead of establishing an MPTCP connection and the associated TCP subflows in controlled environments where security attacks are not a concern.

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 April 18, 2013.

Copyright Notice

Table of Contents

1. Introduction

This document introduces a variant of the MPTCP handshake that is suitable for an environment where security attacks are not an issue. The proposed handshake is a low overhead, low security version of the MPTCP handshake defined in [I-D.ietf-mptcp-multiaddressed].

Its goal is to provide an MPTCP handshake and authentication mechanism, reducing the computational overhead provided by MPTCP version 0.

2. Connection initiation

MultiPath TCP uses the MP_CAPABLE option in the handshake for the initial subflow. This handshake was designed to meet several requirements. When designing another variant of the Multipath TCP handshake, it is important to have these requirements in mind. These requirements are :

1. Detect whether the peer supports MultiPath TCP.

2. Each host generates a locally unique token that unambiguously identifies the Multipath TCP connection

3. Agree on an Initial Data Sequence Number to initialize the MPTCP state on each direction of the Multipath TCP connection

Before discussing the proposed low overhead handshake, it is important to have in mind how [I-D.ietf-mptcp-multiaddressed] meets the three requirements above.

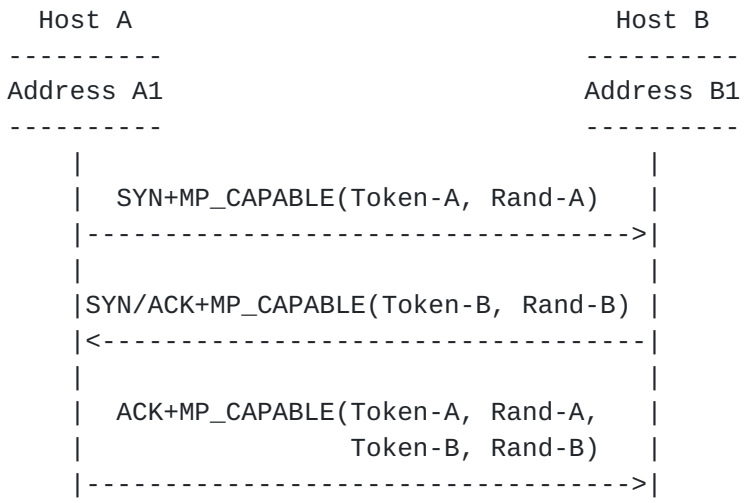
The first requirement is simply met by using a Multipath TCP specific option like all TCP extensions.

To meet the second requirement, a simple solution would have been to encode the token inside the MP_CAPABLE option. However, this would have increased the size of the MP_CAPABLE option. This would have limited the possibility of extending Multipath TCP later by adding new TCP options that require space inside the SYN segments. To minimize the number of option bytes consummed in the SYN segment, [I-D.ietf-mptcp-multiaddressed] uses a hash function to compute the token based on the keys exchanged in clear. However, using hash functions implies that implementations must handle the possible collisions which increases the complexity of the Multipath TCP handshake.

The third requirement is more subtle but is also important to ensure

the reliability of a Multipath TCP connection. Let us assume that Multipath TCP hosts do not agree on an Initial Data Sequence Number. Consider the following scenario. Host A opens the initial TCP subflow of the Multipath TCP connection. Host B opens a second subflow in this Multipath TCP connection. Host B sends one byte with DSN x over the initial subflow, but this data never reaches host A. Host B then sends one byte, starting at DSN x+1 over the second subflow. If host A does not know the Initial Data Sequence Number used by host B, it cannot determine whether the byte received over the second subflow can be acknowledged at the DSN level or not. [I-D.ietf-mptcp-multiaddressed] solves this problem by allowing the two hosts to derive the Initial Data Sequence Number from the keys exchanged in the MP_CAPABLE option. However, this is achieved by computing a hash over the exchanged keys, which increases the computational overhead of generating/processing the MP_CAPABLE option.

The figure below provides a simpler and low overhead handshake that meets the three requirements identified above.



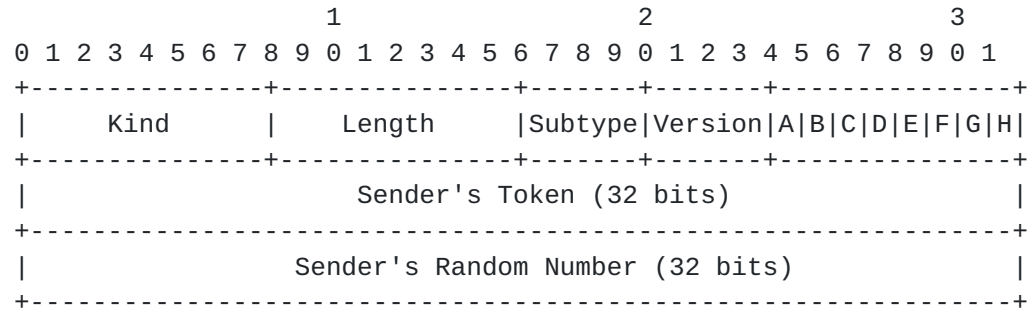
Handshake of the initial subflow.

Figure 1

MPTCP's establishment of the initial subflow follows TCP's regular 3-way handshake, but the SYN, SYN/ACK and ACK packets contain the MP_CAPABLE-option. The proposed MP_CAPABLE option contains one 32 bits token and one 32 bits random number in the SYN and SYN/ACK segments. The third ACK includes an MP_CAPABLE option that contains the two tokens and random numbers. The tokens are used to explictely exchange identifier of the Multipath TCP connection. The random numbers, combined with the tokens produce the Initial Data Sequence Numbers. Echoing all the information back in the third ACK allows

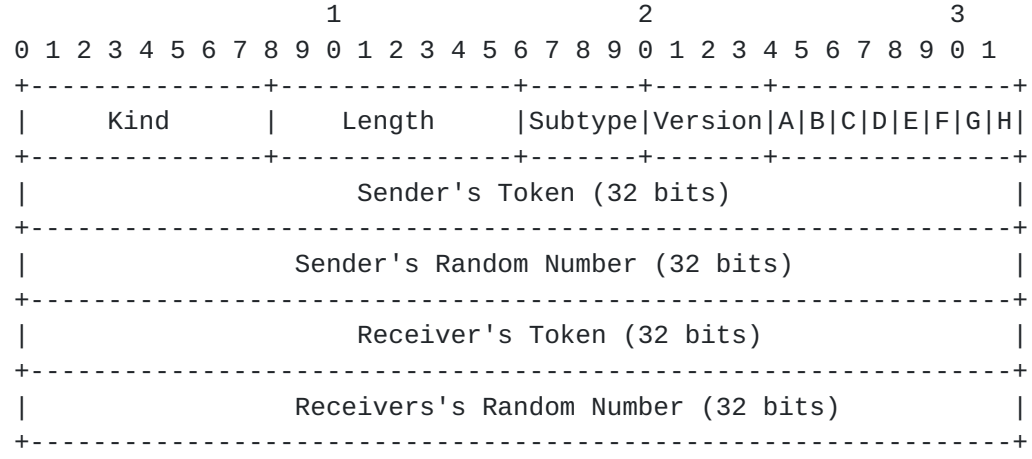
stateless operation of the server.

The format of the proposed MP_CAPABLE option is proposed in the figures below.



Format of the MP_CAPABLE-option in the SYN and SYN/ACK packets

Figure 2



Format of the MP_CAPABLE-option in the third ACK of the handshake

Figure 3

The format of the MP_CAPABLE option is shown in Figure 2. To indicate that this MP_CAPABLE contains tokens/random numbers and not keys (as in [I-D.ietf-mptcp-multiaddressed], the Version-field is set to 1. The message format of the third ACK's MP_CAPABLE option is show in Figure 3.

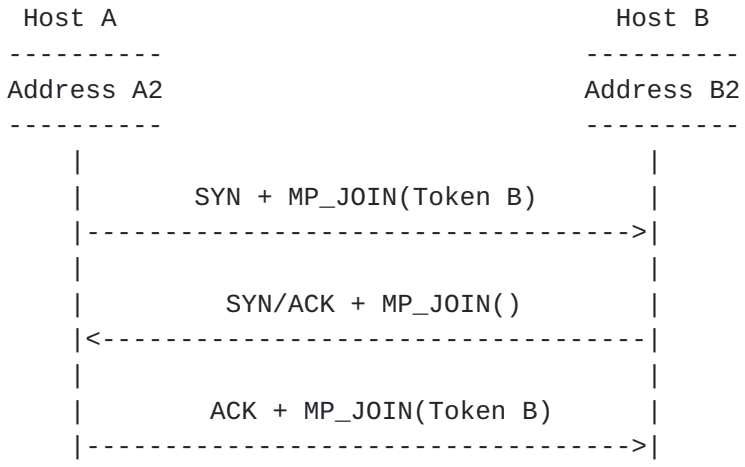
The Initial Data Sequence Number (IDSN) serves to initialize the MPTCP state on the end-hosts in the same way as TCP's sequence numbers do during the 3-way handshake. There is one IDSN for each direction of the data-stream. The IDSN for the data from the client

to the server is the concatenation of Rand-A and Token-A (Rand-A|| Token-A). Rand-A is thus the high-order 32 bits of the IDSN, and Token-A the low-order 32 bits. For the data from server to client, the IDSN is the concatenation of Rand-B and Token-B (Rand-B|| Token-B). Rand-A and Rand-B MUST be random numbers with sufficient randomness so that they are hard to guess. Recommendations for generating random numers for use in keys are given in [RFC4086].

The meaning of the other fields and behavior of the end-hosts during the MP_CAPABLE exchange is the same as specified in [I-D.ietf-mptcp-multiaddressed].

3. Starting a new subflow

Once an MPTCP connection has been established and the tokens exchanged, new subflows can be established. The establishment of the new subflows follows the handshake as show in Figure 4.

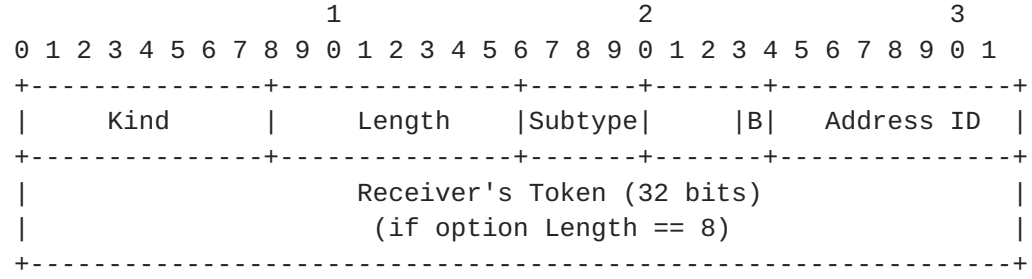


Handshake for a new subflow.

Figure 4

As the low-overhead version of MPTCP does not try to protect against hijacking attacks, the only goal of the MP_JOIN inside the 3-way handshake is to identify the MPTCP connection this subflow is joining. The token inside the MP_JOIN of the SYN-segment allows the server to identify the connection. The SYN/ACK also contains an MP_JOIN option because the server needs to signal to the client that it indeed received the SYN together with the MP_JOIN and that there is no middlebox that removes MPTCP options on this path. Finally, the client replies with the third ack. This third ack contains again token B. This allows the server to handle MP_JOIN's in a stateless manner, as described below. The third ack is sent in a reliable

manner as explained in [I-D.ietf-mptcp-multiaddressed].



Format of the MP_JOIN-option

Figure 5

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

4. Operation

4.1. Generating the token

The token must only be locally unique. The method used to generate the token is implementation specific. One possible way to generate the token is by applying a block-cipher on a counter together with a local secret. This approach has the benefit of a higher probability of uniqueness of the token. We will only have a token collision after the counter has wrapped around. This means, that a connection must have survived 2^32 other connections to cause a collision. Thus, a token collision is less likely to occur than with [I-D.ietf-mptcp-multiaddressed].

4.2. Stateless Servers

To allow stateless SYN+Join handling, the server has to perform the following upon reception of a SYN:

- Check whether there exists an MPTCP-connection corresponding to the token inside the MP_JOIN option.

- Send a SYN/ACK as it is done on today's stateless servers.

When receiving the third ACK (sent reliably as it is done in today's MPTCP), the server verifies that indeed it has generated a SYN/ACK (like regular TCP's SYN-cookie mechanism) and thanks to the token echoed back in the third ACK, the server can find the MPTCP-session this subflow is joining.

Handling the SYN+Join in a stateless manner allows the server to protect itself against attackers that are flooding the server with SYN+Join messages. As the server does not need to create state when sending the SYN/ACK, flooding performed by the attacker will not prevent real clients from establishing new subflows.

5. Security Considerations

The proposed solution removes the HMAC authentication mechanism described in [I-D.ietf-mptcp-multiaddressed]. It is assumed that end-hosts will only use this low-overhead version of MPTCP for non-security critical traffic or in controlled environments like isolated data-centers.

Security-critical traffic is nowadays typically sent over SSL/TLS or similar secure application level protocols. This is done because the transport protocols like TCP do not provide a sufficient security. An application using SSL over MPTCP benefits from the same security provided by SSL. There is one downside of using SSL over MPTCP. If an attacker manages to join an existing connection thanks to a JOIN-exchange, he can inject data into the SSL-session. However, thanks to the MAC-authentication of the SSL messages, the end-hosts will tear down the SSL session.

6. Informative References

[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.

[RFC4086] Eastlake, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, June 2005.

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