Internet Engineering Task Force (IETF)
Request for Comments: 7430
Category: Informational
ISSN: 2070-1721

M. Bagnulo
UC3M
C. Paasch
UCLouvain
F. Gont
SI6 Networks / UTN-FRH
O. Bonaventure
UCLouvain
C. Raiciu
UPB

July 2015

Analysis of Residual Threats and Possible Fixes for Multipath TCP (MPTCP)

Abstract

This document analyzes the residual threats for Multipath TCP (MPTCP) and explores possible solutions to address them.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This document provides a complement to the threat analysis for Multipath TCP (MPTCP) [RFC6824] documented in RFC 6181 [RFC6181]. RFC 6181 provided a threat analysis for the general solution space of extending TCP to operate with multiple IP addresses per connection. Its main goal was to leverage previous experience acquired during the design of other multi-address protocols, notably Shim6 [RFC5533], the Stream Control Transmission Protocol (SCTP) [RFC4960], and Mobile IPv6 (MIP6) [RFC6275] for designing MPTCP. Thus, RFC 6181 was produced before the actual MPTCP specification (RFC 6824) was completed and documented a set of recommendations that were considered during the production of that specification.

This document complements RFC 6181 with a vulnerability analysis of the mechanisms specified in RFC 6824. The motivation for this analysis is to identify possible security issues with MPTCP as currently specified and propose security enhancements to address these identified security issues.

The goal of the security mechanisms defined in RFC 6824 was to make MPTCP no worse than currently available single-path TCP. We believe that this goal is still valid, so we will perform our analysis on the same grounds. This document describes all the threats identified that are specific to MPTCP (as defined in RFC 6824) that are not possible with single-path TCP. This means that threats that are common to TCP and MPTCP are not covered in this document.

For each attack considered in this document, we identify the type of attacker. We can classify the attackers based on their location as follows:

- Off-path attacker. This is an attacker that does not need to be located in any of the paths of the MPTCP session at any point in time during the lifetime of the MPTCP session. This means that the off-path attacker cannot eavesdrop any of the packets of the MPTCP session.

- Partial-time on-path attacker. This is an attacker that needs to be in at least one of the paths during part of the lifetime of the MPTCP session (but not the entire lifetime). The attacker can be in the forward and/or backward directions for the initial subflow and/or other subflows. The specific needs of the attacker will be made explicit in the attack description.

- On-path attacker. This attacker needs to be on at least one of the paths during the whole duration of the MPTCP session. The attacker can be in the forward and/or backward directions for the initial subflow and/or other subflows. The specific needs of the attacker will be made explicit in the attack description.

We can also classify the attackers based on their actions as follows:

- Eavesdropper. The attacker is able to capture some of the packets of the MPTCP session to perform the attack, but it is not capable of changing, discarding, or delaying any packet of the MPTCP session. The attacker can be in the forward and/or backward directions for the initial subflow and/or other subflows. The specific needs of the attacker will be made explicit in the attack description.

- Active attacker. The attacker is able to change, discard, or delay some of the packets of the MPTCP session. The attacker can be in the forward and/or backward directions for the initial subflow and/or other subflows. The specific needs of the attacker will be made explicit in the attack description.

In this document, we consider the following possible combinations of attackers:

- an on-path eavesdropper

- an on-path active attacker

- an off-path active attacker

- a partial-time on-path eavesdropper

- a partial-time on-path active attacker

In the rest of the document, we describe different attacks that are possible against the MPTCP protocol specified in RFC 6824 and propose possible security enhancements to address them.

2. ADD_ADDR Attack

Summary of the attack:

Type of attack: MPTCP session hijack enabling a man-in-the-middle (MitM) attack

Type of attacker: off-path active attacker

Description:

In this attack, the attacker uses the ADD_ADDR option defined in RFC 6824 to hijack an ongoing MPTCP session and enable himself to perform a man-in-the-middle attack on the MPTCP session.

Consider the following scenario. Host A with address IPA has one MPTCP session with Host B with address IPB. The MPTCP subflow between IPA and IPB is using port PA on Host A and port PB on Host B. The tokens for the MPTCP session are TA and TB for Host A and Host B, respectively. Host C is the attacker. It owns address IPC. The attack is executed as follows:

1. Host C sends a forged packet with source address IPA, destination address IPB, source port PA, and destination port PB. The packet has the ACK flag set. The TCP sequence number for the segment is i, and the ACK sequence number is j. We will assume all these are valid; later, we discuss what the attacker needs to figure them out. The packet contains the ADD_ADDR option. The ADD_ADDR option announces IPC as an alternative address for the connection. It also contains an 8-bit address identifier that does not provide any strong security benefit.

2. Host B receives the ADD_ADDR message and replies by sending a TCP SYN packet.

   > Note: The MPTCP specification [RFC6824] states that the host receiving the ADD_ADDR option may initiate a new subflow. If the host is configured so that it does not initiate a new subflow, the attack will not succeed. For example, on the current Linux implementation, the server does not create subflows. Only the client does so.

   The source address for the packet is IPB; the destination address for the packet is IPC; the source port is PB’; and the destination port is PA’ (it is not required that PA=PA’ nor that PB=PB’). The sequence number for this packet is the new initial sequence number for this subflow. The ACK sequence number is not relevant as the ACK flag is not set. The packet carries an MP_JOIN option and the token TA. It also carries a random nonce generated by Host B called RB.

3. Host C receives the SYN+MP_JOIN packet from Host B and alters it in the following way. It changes the source address to IPC and the destination address to IPA. It sends the modified packet to Host A, impersonating Host B.

4. Host A receives the SYN+MP_JOIN message and replies with a SYN/ACK+MP_JOIN message. The packet has source address IPA and destination address IPC, as well as all the other needed parameters. In particular, Host A computes a valid Hashed Message Authentication Code (HMAC) and places it in the MP_JOIN option.

5. Host C receives the SYN/ACK+MP_JOIN message and changes the source address to IPC and the destination address to IPB. It sends the modified packet to IPB, impersonating Host A.

6. Host B receives the SYN/ACK+MP_JOIN message. Host B verifies the HMAC of the MP_JOIN option and confirms its validity. It replies with an ACK+MP_JOIN packet. The packet has source address IPB and destination address IPC, as well as all the other needed parameters. The returned MP_JOIN option contains a valid HMAC computed by Host B.

7. Host C receives the ACK+MP_JOIN message from B and alters it in the following way. It changes the source address to IPC and the destination address to IPA. It sends the modified packet to Host A, impersonating Host B.

8. Host A receives the ACK+MP_JOIN message and creates the new subflow. At this point, the attacker has managed to place itself as a MitM for one subflow for the existing MPTCP session. It should be noted that the subflow between addresses IPA and IPB that does not flow through the attacker still exists, so the attacker has not completely intercepted all the packets in the communication (yet). If the attacker wishes to completely intercept the MPTCP session, it can do the following additional step.

9. Host C sends two TCP RST messages. One TCP RST packet is sent to Host B, with source address IPA, destination address IPB, and source and destination ports PA and PB, respectively. The other TCP RST message is sent to Host A, with source address IPB, destination address IPA, and source and destination ports PB and PA, respectively. Both RST messages must contain a valid sequence number. Note that figuring the sequence numbers to be used here for subflow A is the same difficulty as being able to send the initial ADD_ADDR option with valid sequence number and ACK value. If there are more subflows, then the attacker needs to find the sequence number and ACK for each subflow. At this point, the attacker has managed to fully hijack the MPTCP session.

Information required by the attacker to perform the described attack:

In order to perform this attack the attacker needs to guess or know the following pieces of information. The attacker needs this information for one of the subflows belonging to the MPTCP session.

- the four-tuple {Client-side IP Address, Client-side Port, Server-side Address, Server-side Port} that identifies the target TCP connection

- a valid sequence number for the subflow

- a valid ACK sequence number for the subflow

- a valid address identifier for IPC

TCP connections are uniquely identified by the four-tuple {Source Address, Source Port, Destination Address, Destination Port}. Thus, in order to attack a TCP connection, an attacker needs to know or be able to guess each of the values in that four-tuple. Assuming the two peers of the target TCP connection are known, the Source Address and the Destination Address can be assumed to be known.

> Note: In order to be able to successfully perform this attack, the attacker needs to be able to send packets with a forged source address. This means that the attacker cannot be located in a network where techniques like ingress filtering [RFC2827] or source address validation [RFC7039] are deployed. However, ingress filtering is not as widely implemented as one would expect and hence cannot be relied upon as a mitigation for this kind of attack.

Assuming the attacker knows the application protocol for which the TCP connection is being employed, the server-side port can also be assumed to be known. Finally, the client-side port will generally not be known and will need to be guessed by the attacker. The chances of an attacker guessing the client-side port will depend on the ephemeral port range employed by the client and whether or not the client implements port randomization [RFC6056].

Assuming TCP sequence number randomization is in place (see e.g., [RFC6528]), an attacker would have to blindly guess a valid TCP sequence number. That is,

```
   RCV.NXT =< SEG.SEQ < RCV.NXT+RCV.WND or RCV.NXT =<
   SEG.SEQ+SEG.LEN-1 < RCV.NXT+RCV.WND
```

As a result, the chances of an attacker succeeding will depend on the TCP receive window size at the target TCP peer.

> Note: Automatic TCP buffer tuning mechanisms have become common for popular TCP implementations; hence, very large TCP window sizes of values up to 2 MB could end up being employed by such TCP implementations.

According to [RFC793], the acknowledgement number is considered valid as long as it does not acknowledge the receipt of data that has not yet been sent. That is, the following expression must be true:

SEG.ACK <= SND.NXT

However, for implementations that support [RFC5961], the following (stricter) validation check is enforced:

SND.UNA - MAX.SND.WND <= SEG.ACK <= SND.NXT

Finally, in order for the address identifier to be valid, the only requirement is that it needs to be different from the ones already being used by Host A in that MPTCP session, so a random identifier is likely to work.

Given that a large number of factors affect the chances of an attacker successfully performing the aforementioned off-path attacks, we provide two general expressions for the expected number of packets the attacker needs to send to succeed in the attack: one for MPTCP implementations that support [RFC5961] and another for MPTCP implementations that do not.

Implementations that do not support RFC 5961:

Packets = (2^32/(RCV_WND)) * 2 * EPH_PORT_SIZE/2 * 1/MSS

Where the new parameters are:

Packets:
   Maximum number of packets required to successfully perform an off-path (blind) attack.

RCV_WND:
   TCP receive window size (RCV.WND) at the target node.

EPH_PORT_SIZE:
Number of ports comprising the ephemeral port range at the "client" system.

MSS:
Maximum Segment Size, assuming the attacker will send full segments to maximize the chances of getting a hit.

Notes:
The value "2^32" represents the size of the TCP sequence number space.

The value "2" accounts for two different ACK numbers (separated by 2^31) that should be employed to make sure the ACK number is valid.

The following table contains some sample results for the number of required packets, based on different values of RCV_WND and EPH_PORT_SIZE for an MSS of 1500 bytes.

Ports \ Win	16 KB	128 KB	256 KB	2048 KB
4000	699050	87381	43690	5461
10000	1747626	218453	109226	13653
50000	8738133	1092266	546133	68266

Table 1: Maximum Number of Packets for Successful Attack

Implementations that do support RFC 5961:

$$\text{Packets} = (2^{32}/(\text{RCV\_WND})) * (2^{32}/(2 * \text{SND\_MAX\_WND})) * \text{EPH\_PORT\_SIZE}/2 * 1/\text{MSS}$$

Where:

Packets:
Maximum number of packets required to successfully perform an off-path (blind) attack.

RCV_WND:
TCP receive window size (RCV.WND) at the target MPTCP endpoint.

SND_MAX_WND:
Maximum TCP send window size ever employed by the target MPTCP endpoint (MAX.SND.WND).

EPH_PORT_SIZE:
Number of ports comprising the ephemeral port range at the "client" system.

Notes:
The value "2^32" represents the size of the TCP sequence number space.

The parameter "MAX.SND.WND" is specified in [RFC5961].

The value "2 * SND_MAX_WND" results from the expression "SND.NXT - SND.UNA - MAX.SND.WND", assuming that, for connections that perform bulk data transfers, "SND.NXT - SND.UNA == MAX.SND.WND". If an attacker targets a TCP endpoint that is not actively transferring data, "2 * SND_MAX_WND" would become "SND_MAX_WND" (and hence a successful attack would typically require more packets).

The following table contains some sample results for the number of required packets, based on different values of RCV_WND, SND_MAX_WND, and EPH_PORT_SIZE. For these implementations, only a limited number of sample results are provided (as an indication of how [RFC5961] increases the difficulty of performing these attacks).

Ports \ Win	16 KB	128 KB	256 KB	2048 KB
4000	45812984490	715827882	178956970	2796202

Table 2: Maximum Number of Packets for Successful Attack

Note:
In the aforementioned table, all values are computed with RCV_WND equal to SND_MAX_WND.

2.1. Possible Security Enhancements to Prevent This Attack

1. To include the token of the connection in the ADD_ADDR option. This would make it harder for the attacker to launch the attack, since the attacker needs to either eavesdrop the token (so this can no longer be a blind attack) or to guess it, but a random 32-bit number is not easy to guess. However, this would imply that any eavesdropper that is able to see the token would be able

to launch this attack. This solution then increases the vulnerability window against eavesdroppers from the initial 3-way handshake for the MPTCP session to any exchange of the ADD_ADDR messages.

2. To include the HMAC of the address contained in the ADD_ADDR option. The key used for the HMAC is the concatenation of the key of the receiver and the key of the sender (in the same way they are used for generating the HMAC of the MP_JOIN message). This makes it much more secure, since it requires the attacker to have both keys (either by eavesdropping it in the first exchange or by guessing it). Because this solution relies on the key used in the MPTCP session, the protection of this solution would increase if new key generation methods are defined for MPTCP (e.g., using Secure Socket Layer (SSL) keys as has been proposed).

3. To include the destination address of the SYN packet in the HMAC of the MP_JOIN message. As the attacker requires changing the destination address to perform the described attack, protecting it would prevent the attack. It wouldn’t allow hosts behind NATs to be reached by an address in the ADD_ADDR option, even with static NAT bindings (like a web server at home).

Of the options described above, option 2 is recommended as it achieves a higher security level while preserving the required functionality (i.e., NAT compatibility).

3. DoS Attack on MP_JOIN

Summary of the attack:

Type of attack: MPTCP denial-of-service attack, preventing the hosts from creating new subflows

Type of attacker: off-path active attacker

Description:

As currently specified, the initial SYN+MP_JOIN message of the 3-way handshake for additional subflows creates state in the host receiving the message. This is because the SYN+MP_JOIN contains the 32-bit token that allows the receiver to identify the MPTCP session and the 32-bit random nonce used in the HMAC calculation. As this information is not re-sent in the third ACK of the 3-way handshake, a host must create state upon reception of a SYN+MP_JOIN.

Assume that an MPTCP session exists between Host A and Host B, with tokens TA and TB. An attacker, sending a SYN+MP_JOIN to Host B, with the valid token TB, will trigger the creation of state on Host B. The number of these half-open connections a host can store per MPTCP session is limited by a certain number and is implementation-dependent. The attacker can simply exhaust this limit by sending multiple SYN+MP_JOINs with different 5-tuples. The (possibly forged) source address of the attack packets will typically correspond to an address that is not in use, or else, the SYN/ACK sent by Host B would elicit a RST from the impersonated node, thus removing the corresponding state at Host B. Further discussion of traditional SYN flooding attacks and common mitigations can be found in [RFC4987].

This effectively prevents Host A from sending any more SYN+MP_JOINs to Host B, as the number of acceptable half-open connections per MPTCP session on Host B has been exhausted.

The attacker needs to know the token TB in order to perform the described attack. This can be achieved if it is a partial-time on-path eavesdropper observing the 3-way handshake of the establishment of an additional subflow between Host A and Host B. If the attacker is never on-path, it has to guess the 32-bit token.

3.1. Possible Security Enhancements to Prevent This Attack

The third packet of the 3-way handshake could be extended to also contain the 32-bit token and the random nonce that has been sent in the SYN+MP_JOIN. Further, Host B will have to generate its own random nonce in a reproducible fashion (e.g., a hash of the 5-tuple + initial sequence number + local secret). This will allow Host B to reply to a SYN+MP_JOIN without having to create state. Upon the reception of the third ACK, Host B can then verify the correctness of the HMAC and create the state.

4. SYN Flooding Amplification

Summary of the attack:

Type of attack: The attacker uses SYN+MP_JOIN messages to amplify the SYN flooding attack.

Type of attacker: off-path active attacker

Description:

SYN flooding attacks [RFC4987] use SYN messages to exhaust the server’s resources and prevent new TCP connections. A common mitigation is the use of SYN cookies [RFC4987] that allow stateless processing of the initial SYN message.

With MPTCP, the initial SYN can be processed in a stateless fashion using the aforementioned SYN cookies. However, as described in the previous section, as currently specified, SYN+MP_JOIN messages are not processed in a stateless manner. This opens a new attack vector. The attacker can now open an MPTCP session by sending a regular SYN and creating the associated state but then sending as many SYN+MP_JOIN messages as supported by the server with different combinations of source address and source port, consuming the server’s resources without having to create state in the attacker. This is an amplification attack, where the cost on the attacker side is only the cost of the state associated with the initial SYN while the cost on the server side is the state for the initial SYN plus all the state associated with all the following SYN+MP_JOINs.

4.1. Possible Security Enhancements to Prevent This Attack

1. The solution described for the previous DoS attack on MP_JOIN would also prevent this attack.

2. Limiting the number of half-open subflows to a low number (e.g., three subflows) would also limit the impact of this attack.

5. Eavesdropper in the Initial Handshake

Summary of the attack:

Type of attack: An eavesdropper present in the initial handshake where the keys are exchanged can hijack the MPTCP session at any time in the future.

Type of attacker: partial-time on-path eavesdropper

Description:

In this case, the attacker is present along the path when the initial 3-way handshake takes place and therefore is able to learn the keys used in the MPTCP session. This allows the attacker to move away from the MPTCP session path and still be able to hijack the MPTCP session in the future. This vulnerability was readily identified when designing the MPTCP security solution [RFC6181], and the threat was considered acceptable.

5.1. Possible Security Enhancements to Prevent This Attack

There are many techniques that can be used to prevent this attack, and each of them represents different trade-offs. At this point, we limit ourselves to enumerate them and provide useful pointers.

1. Use of hash chains. The use of hash chains for MPTCP has been explored in [HASH-CHAINS].

2. Use of SSL keys for MPTCP security as described in [MPTCP-SSL].

3. Use of Cryptographically Generated Addresses (CGAs) for MPTCP security. CGAs [RFC3972] have been used in the past to secure multi-addressed protocols like Shim6 [RFC5533].

4. Use of tcpcrypt [TCPCRYPT].

5. Use of DNSSEC. DNSSEC has been proposed to secure the Mobile IP protocol [DNSSEC].

6. SYN/JOIN Attack

Summary of the attack:

Type of attack: An attacker that can intercept the SYN/JOIN message can alter the source address being added.

Type of attacker: partial-time on-path eavesdropper

Description:

The attacker is present along the path when the SYN/JOIN exchange takes place. This allows the attacker to add any new address it wants to by simply substituting the source address of the SYN/JOIN packet for one it chooses. This vulnerability was readily identified when designing the MPTCP security solution [RFC6181], and the threat was considered acceptable.

6.1. Possible Security Enhancements to Prevent This Attack

It should be noted that this vulnerability is fundamental due to the NAT support requirement. In other words, MPTCP must work through NATs in order to be deployable in the current Internet. NAT behavior is unfortunately indistinguishable from this attack. It is impossible to secure the source address, since doing so would prevent MPTCP from working through NATs. This basically implies that the solution cannot rely on securing the address. A more promising approach would be to look into securing the payload exchanged and thus limiting the impact that the attack would have in the communication (e.g., tcpcrypt [TCPCRYPT] or similar).

7. Recommendations

The current MPTCP specification [RFC6824] is Experimental. There is an ongoing effort to move it to Standards Track. We believe that the work on MPTCP security should follow two threads:

- The work on improving MPTCP security so that the MPTCP specification [RFC6824] can become a Standards Track document.

- The work on analyzing possible additional security enhancements to provide a more secure version of MPTCP.

We expand on these in the following subsections.

7.1. MPTCP Security Improvements for a Standards Track Specification

We believe that in order for MPTCP to progress to Standards Track, the ADD_ADDR attack must be addressed. We believe that the solution that should be adopted in order to deal with this attack is to include an HMAC to the ADD_ADDR message (with the address being added used as input to the HMAC as well as the key). This would make the ADD_ADDR message as secure as the JOIN message. In addition, this implies that if we implement a more secure way to create the key used in the MPTCP connection, then the security of both the MP_JOIN and the ADD_ADDR messages is automatically improved (since both use the same key in the HMAC).

We believe that this is enough for MPTCP to progress as a Standards Track document because the security level is similar to single-path TCP per our previous analysis. Moreover, the security level achieved with these changes is exactly the same as other Standards Track documents. In particular, this would be the same security level as SCTP with dynamic addresses as defined in [RFC5061]. The Security Considerations section of RFC 5061 (which is a Standards Track document) reads:

> The addition and or deletion of an IP address to an existing association does provide an additional mechanism by which existing associations can be hijacked. Therefore, this document requires the use of the authentication mechanism defined in [RFC4895] to limit the ability of an attacker to hijack an association.
>
> Hijacking an association by using the addition and deletion of an IP address is only possible for an attacker who is able to intercept the initial two packets of the association setup when

the SCTP-AUTH extension is used without pre-shared keys. If such a threat is considered a possibility, then the [RFC4895] extension MUST be used with a preconfigured shared endpoint pair key to mitigate this threat.

This is the same security level that would be achieved by MPTCP with the addition of the ADD_ADDR security measure recommended in this document.

7.2. Security Enhancements for MPTCP

We also believe that is worthwhile to explore alternatives to secure MPTCP. As we identified earlier, the problem of securing JOIN messages is fundamentally incompatible with NAT support, so it is likely that a solution to this problem involves the protection of the data itself. Exploring the integration of MPTCP and approaches like tcpcrypt [TCPCRYPT] and exploring integration with SSL seem promising.

8. Security Considerations

This whole document is about security considerations for MPTCP.

9. References

9.1. Normative References

[RFC793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, DOI 10.17487/RFC0793, September 1981, <http://www.rfc-editor.org/info/rfc793>.

[RFC3972] Aura, T., "Cryptographically Generated Addresses (CGA)", RFC 3972, DOI 10.17487/RFC3972, March 2005, <http://www.rfc-editor.org/info/rfc3972>.

[RFC4895] Tuexen, M., Stewart, R., Lei, P., and E. Rescorla, "Authenticated Chunks for the Stream Control Transmission Protocol (SCTP)", RFC 4895, DOI 10.17487/RFC4895, August 2007, <http://www.rfc-editor.org/info/rfc4895>.

[RFC5061] Stewart, R., Xie, Q., Tuexen, M., Maruyama, S., and M. Kozuka, "Stream Control Transmission Protocol (SCTP) Dynamic Address Reconfiguration", RFC 5061, DOI 10.17487/RFC5061, September 2007, <http://www.rfc-editor.org/info/rfc5061>.

[RFC5961] Ramaiah, A., Stewart, R., and M. Dalal, "Improving TCP's Robustness to Blind In-Window Attacks", RFC 5961, DOI 10.17487/RFC5961, August 2010, <http://www.rfc-editor.org/info/rfc5961>.

[RFC6056] Larsen, M. and F. Gont, "Recommendations for Transport-Protocol Port Randomization", BCP 156, RFC 6056, DOI 10.17487/RFC6056, January 2011, <http://www.rfc-editor.org/info/rfc6056>.

[RFC6528] Gont, F. and S. Bellovin, "Defending against Sequence Number Attacks", RFC 6528, DOI 10.17487/RFC6528, February 2012, <http://www.rfc-editor.org/info/rfc6528>.

[RFC6824] Ford, A., Raiciu, C., Handley, M., and O. Bonaventure, "TCP Extensions for Multipath Operation with Multiple Addresses", RFC 6824, DOI 10.17487/RFC6824, January 2013, <http://www.rfc-editor.org/info/rfc6824>.

9.2. Informative References

[DNSSEC] Kukec, A., Bagnulo, M., Ayaz, S., Bauer, C., and W. Eddy, "ROAM-DNSSEC: Route Optimization for Aeronautical Mobility using DNSSEC", 4th ACM International Workshop on Mobility in the Evolving Internet Architecture (MobiArch), 2009.

[HASH-CHAINS] Diez, J., Bagnulo, M., Valera, F., and I. Vidal, "Security for multipath TCP: a constructive approach", International Journal of Internet Protocol Technology, Vol. 6, No. 3, 2011.

[MPTCP-SSL] Paasch, C. and O. Bonaventure, "Securing the MultiPath TCP handshake with external keys", Work in Progress, draft-paasch-mptcp-ssl-00, October 2012.

[RFC2827] Ferguson, P. and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing", BCP 38, RFC 2827, DOI 10.17487/RFC2827, May 2000, <http://www.rfc-editor.org/info/rfc2827>.

[RFC4960] Stewart, R., Ed., "Stream Control Transmission Protocol", RFC 4960, DOI 10.17487/RFC4960, September 2007, <http://www.rfc-editor.org/info/rfc4960>.

[RFC4987] Eddy, W., "TCP SYN Flooding Attacks and Common Mitigations", RFC 4987, DOI 10.17487/RFC4987, August 2007, <http://www.rfc-editor.org/info/rfc4987>.

[RFC5533] Nordmark, E. and M. Bagnulo, "Shim6: Level 3 Multihoming Shim Protocol for IPv6", RFC 5533, DOI 10.17487/RFC5533, June 2009, <http://www.rfc-editor.org/info/rfc5533>.

[RFC6181] Bagnulo, M., "Threat Analysis for TCP Extensions for Multipath Operation with Multiple Addresses", RFC 6181, DOI 10.17487/RFC6181, March 2011, <http://www.rfc-editor.org/info/rfc6181>.

[RFC6275] Perkins, C., Ed., Johnson, D., and J. Arkko, "Mobility Support in IPv6", RFC 6275, DOI 10.17487/RFC6275, July 2011, <http://www.rfc-editor.org/info/rfc6275>.

[RFC7039] Wu, J., Bi, J., Bagnulo, M., Baker, F., and C. Vogt, Ed., "Source Address Validation Improvement (SAVI) Framework", RFC 7039, DOI 10.17487/RFC7039, October 2013, <http://www.rfc-editor.org/info/rfc7039>.

[TCPCRYPT] Bittau, A., Boneh, D., Hamburg, M., Handley, M., Mazieres, D., and Q. Slack, "Cryptographic protection of TCP Streams (tcpcrypt)", Work in Progress, draft-bittau-tcp-crypt-04, February 2014.

Acknowledgements

We would like to thank Mark Handley for his comments on the attacks and countermeasures discussed in this document. We would also like to thank to Alissa Cooper, Phil Eardley, Yoshifumi Nishida, Barry Leiba, Stephen Farrell, and Stefan Winter for their comments and reviews.

Marcelo Bagnulo, Christoph Paasch, Oliver Bonaventure, and Costin Raiciu are partially funded by the EU Trilogy 2 project.

Authors' Addresses

Marcelo Bagnulo
Universidad Carlos III de Madrid
Av. Universidad 30
Leganes, Madrid 28911
Spain

Phone: 34 91 6249500
Email: marcelo@it.uc3m.es
URI: http://www.it.uc3m.es

Christoph Paasch
UCLouvain

Email: christoph.paasch@gmail.com

Fernando Gont
SI6 Networks / UTN-FRH
Evaristo Carriego 2644
Haedo, Provincia de Buenos Aires 1706
Argentina

Phone: +54 11 4650 8472
Email: fgont@si6networks.com
URI: http://www.si6networks.com

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

Email: olivier.bonaventure@uclouvain.be

Costin Raiciu
Universitatea Politehnica Bucuresti
Splaiul Independentei 313a
Bucuresti
Romania

Email: costin.raiciu@cs.pub.ro