A New MultiPathTCP Flooding Attacks Mitigation Technique

Adwan Yasin
Department of Computer Science
Arab American University
Jenin, Palestine

Hamzah Hijawi
Department of Computer Science
Arab American University
Jenin, Palestine
.

***Abstract*—MPTCP is a new protocol proposed by IETF working group as an extension for standard TCP, it adds the capability to split the TCP connection across multiple paths. It provides higher availability and improves the throughput between two multi-address endpoints. Many Linux distributions have been developed to support MPTCP, most of them are open source which can be modified and compiled to support different experimental scenarios. Splitting the single path TCP connection across multiple paths adds new challenges in paths management and raises new security threats. Some of these threats include flooding and hijacking attacks performed by on-path and off-path attackers. In this article, we propose a new algorithm to mitigate the flooding and hijacking attacks in MPTCP, the proposed method allows a stateful processing of the initial SYN message and it's following SYN_JOIN messages.**

Keywords*—*TCP, MPTCP, flooding, hijack, on-path, off-path, flooding, DoS

I. Introduction

TCP is the most transport protocol used on the internet today, it has been used by most applications as a reliable transfer protocol to transfer data between endpoints. TCP first design was in the 1970s, it has been evolved and enhanced to the current design we have today. TCP was implemented as a layer four protocol in the OSI model stack and as a design decision, the separation from the network layer is intended to be hidden. Five tuples are used to distinguish different streams from each other and to demultiplex packets to their appropriate destination. Source and destination IP addresses are used to forward the pack from source point to the destination, source and destination port numbers are used to identify the running processes on the source and destination while protocol identifier is used to indicate that the connection is using TCP. Therefore, any TCP connection is bounded to a unique socket through a single path between two endpoints [1]. However, if one of the five tuples is changed after the connection is established then the connection will fail.

The design of networks is changed, servers are becoming multi-homed, data centers have many redundant links and mobile devices have multiple wireless interfaces [2]. In order to make a use of these redundant connections, a new TCP design was evolved, it is called multipath TCP [3]. MPTCP allows multiple paths if they exist between the two communicating hosts to be effectively and concurrently used by a single TCP connection. MPTCP has obvious benefits for availability, reliability and load balancing [3]. It is more robust and can achieve better performance compared with a single-path TCP. One of the primary MPTCP design goals is maintaining the compatibility with existing applications and network infrastructure. This is achieved by presenting the MPTCP as a sublayer under TCP layer and let the TCP handles the upper layers applications [4, 5].

MPTCP connection consists of one or more TCP connection. Thus, the risk of vulnerabilities exist in MPTCP would be at least of the same risk in TCP, and particularly the attacks which performed by an on-path attacker who may impersonate one of the communicating parties and eavesdropping, forging, dropping or hijacking the session [6]. One of the design goals of MPTCP is that it is should at least perform as the standard TCP. So, the set of new vulnerabilities exist from the capability of adding new paths to an ongoing connection must be explored. Mainly, flooding and hijacking attacks which are performed by off-path and on-path attackers, and can result in redirection the traffic to unintended target [6, 7].

This paper addresses the flooding and hijacking attacks on MPTCP and proposes a new solution to mitigate these types of attacks. The rest of the paper is organized as follow. Related work is provided in section 2, section 3 gives an overview about MPTCP. Connection establishment in MPTCP is explained in section 4. In section 5, multiple flooding attacks scenarios are explained. The hijacking attack is described in section 6. The proposed solution to mitigate these attacks is provided in section 7 and conclusions are discussed in section 8.

II. Related Work

MPTCP is a new approach towards efficient load balancing between endpoints participating in the TCP connection, it was implemented in many Linux-based distributions [8]. As a design decision, MPTCP is totally backward compatible with existing applications and network devices. A comprehensive study on the impacts that the protocol may have on TCP applications was summarized in [9] and the compatibility issues between MPTCP and standard TCP have been discussed. A performance analysis of MPTCP

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

have been made in [10], in which, throughput comparisons were made between standard TCP and MPTCP with different scenarios, the experiments show how MPTCP outperforms standard TCP in terms of throughput and handover capability when the connection lost. MPTCP offers benefits for availability and connectivity, but there is also a security risk which must be addressed. One of the potential risks comes from the traffic fragmentation between the different paths between the two endpoints. However, modern network security technologies like IPS and IDS are not ready for MPTCP, they are not currently able to re-assemble a full MPTCP session from the different paths and properly inspect and represent a potential security risk [11].

A threat analysis for MPTCP is provided in RFC6181 [12], the analysis identified and characterized the new vulnerabilities which may appear after supporting multiple paths in a single TCP connection. As one of design goals of MPTCP, it is assumed that any MPTCP connection should at least perform as a single path TCP, this means that any potential risk in TCP must be addressed in MPTCP. The studies in [6, 12] provided analyses about the most common potential threats which may exploit the MPTCP connection, this includes flooding and hijacking attacks. A basic solution to mitigate these attacks were provided in [6], in which the sender asks the receiver for each new sub-flow if it can accept data from this new connection If yes then they exchange a random token for authentication purpose. The architecture design for MPTCP provided in [13] suggested three key security requirements, MPTCP should be able to provide a mechanism to confirm that the endpoints participating in a sub-flow handshaking are the same endpoints in the original connection establishment. MPTCP should also provide a mechanism to verify that a host can receive traffic at a given address before opening the sub-flow, it should also provide replay protection in order to verify that a request to add or remove a sub-flow is fresh.

III. MPTCP Overview

Today's networks are becoming multipath, most of the servers, data centers, and mobile devices have redundant network interfaces and more than one IP address at the same time. MPTCP was designed to utilize all available paths between the two communicating points [3]. Figure 1 shows MPTCP connection for a mobile device which has two network interfaces. WiFi is the main connection and 3G is the backup one.

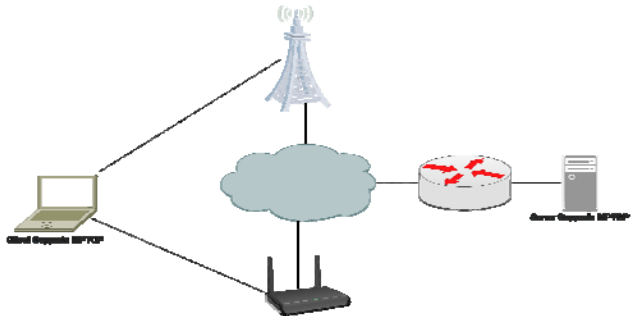

Figure 1. MPTCP topology

MPTCP was designed to achieve a set of requirements which are summarized in three main design goals. The first one is improving the throughput compared with a single-path TCP. The second one, do not harm; MPTC should not take capacity more than a standard TCP would take if both share the same path. The last goal is balancing the congestion; MPTCP should move the traffic to the least congested paths [4, 6]. Two design decision were taken in consideration in MPTCP implementation; application and network compatibility. Application compatibility means that MPTCP should work with existing applications running with TCP without any modification and the network compatibility means that MPTCP should operate with existing networks [3]. As a result of these two design decisions, MPTCP is implemented as a sub-layer in the transport layer, and this implementation achieves the transparency of existing multiple paths to the upper layers as shown in figure 2.

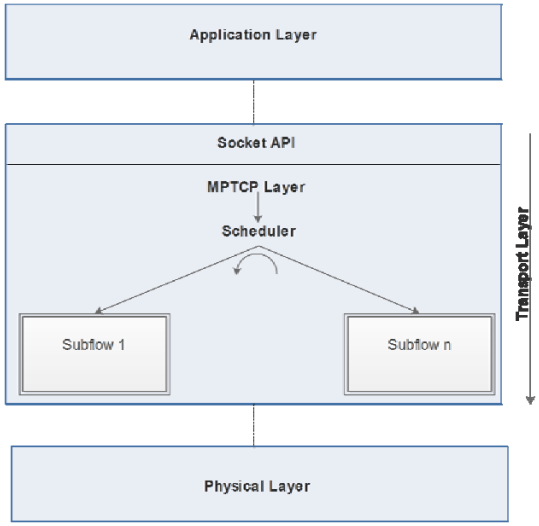

Figure 2. MPTCP in TCP/IP stack

For each path between the source and destination, a new sub-flow is created, each sub-flow can be considered as a normal TCP connection and can be distinguished with the five tuples. Scheduler is a part of MPTCP implementation, it is used to schedule the traffic between all related sub-flows [15]. With the possibility of using multiple paths between the source and destination, concerns have arisen about congestion control over these paths. However, congestion control in MPTCP is different from standard TCP. One of the requirements for MPTCP congestion control algorithm is to be fair to the standard TCP if both share the same link. Another requirement is to transfer more traffic to the least congested path [10], this requirement is needed to utilize the paths between the source and the destination as much as possible. However, if one of the paths is congested then MPTCP decreases the window size on this path and increases the window on the least congested paths [16].

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

IV. MPTCP Connection Establishments

Standard TCP connection can be divided into three stages; connection establishment, data transfer, and connection release. Connection establishment starts with a three-way handshake. However, in order to open a connection, the client sends a synchronize request SYN to a port in which the server is listing. All connection relevant information are sent in SYN request, this includes the source port and the initial sequence number. The server then acknowledges the SYN request with SYNACK reply message. After that, the client acknowledges the SYNACK and then the connection is established. The connection is now established and both hosts can start sending data packets. After the data transfer is over, the connection must be closed, this happens by using FIN packets, the connection is terminated after the FIN packet is acknowledged by both hosts [3].

Multipath TCP connection is established in the same way as TCP connection is established, it uses a three-way handshakes and the options field in the TCP header. The mp_capable option is set in the SYN packet to indicate that the source can perform MPTCP. The destination then replies with SYNACK packet, if it also supports MPTCP then MP_CAPABLE is set, the source then replies with ACK packet which has the MP_CAPABLE option to ensure that this is a MPTCP connection. After the connection is established, participating hosts can add new sub-flows to the connection by using the same negotiation procedure applied in the connection establishment. MP_JOIN option is used instead of MP_CAPAPLE with the connection identifier to inform the destination which connection it would like to join [17]. Once the connection has multiple sub-flows, scheduler decides how to distribute the traffic between them. Each sub-flow can be considered as a standalone TCP connection which has its own congestion control algorithm and sequence numbers space. A new sub-flow connection can be established and added if a new path is available and can be removed if the path is vanished [18]. MPTCP connection establishment is summarized in figure 3.

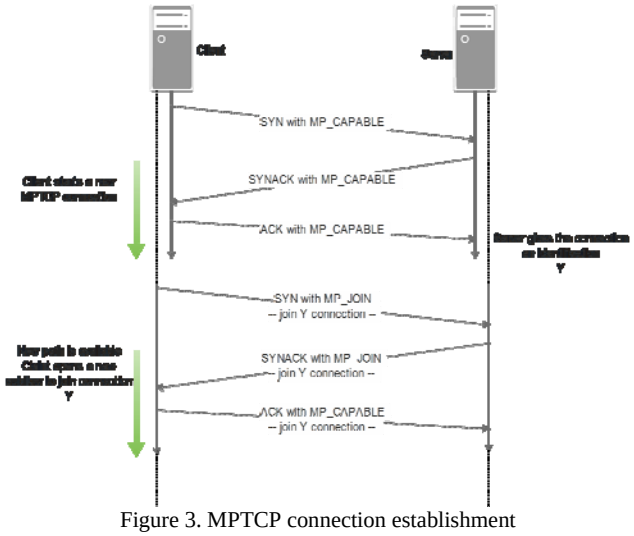

Figure 3. MPTCP connection establishment

V. MPTCP Flooding Attacks

MPTCP flooding attack is one of the attacks introduced by address agility, the goal from this attack is to exhaust the victim by a heavy traffic causing a denial of service. Figure 4 illustrates the redirection attack in which the attacker uses a streaming server to redirect a huge amount of traffic to the victim host.

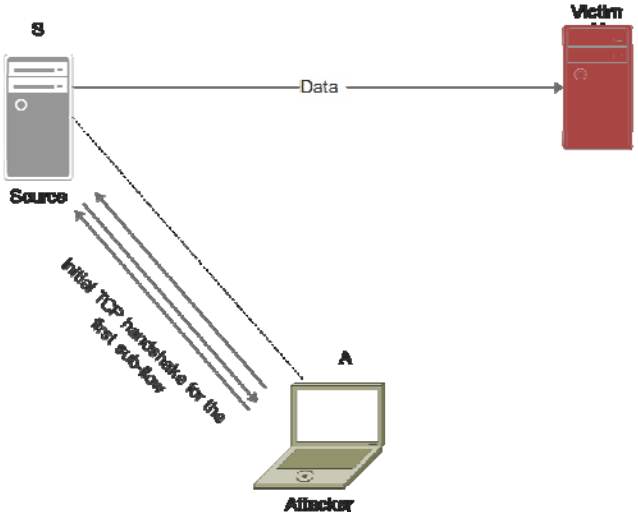

Figure 4. Flooding attack using a stream server

First, the attacker opens a MPTCP connection with the traffic source S and starts downloading a heavy traffic, this connection involves the IP addresses of the attacker and the server S. While the heavy traffic is coming to the attacker from S, the attacker adds the victim IP as one of the available addresses for the connection. After this step, the connection has two IP addresses for the attacker A and a single IP address for the traffic source S. The attacker goal at this point is to send the heavy traffic load from source S to the victim node V. To achieve that, the attacker pretends that the path between him and the source is congested while the path between the traffic source S and the victim V is not. As a result, most of the traffic will be shifted to the least congested path between S and V. In order to successfully complete this step, the attacker acknowledges the traffic that flows between S and V and does not acknowledge the traffic that flows between S and A. ACKs must be sent using packets contain the IP address of the victim as a source address. Sequence numbers of the data being transmitted between S and V should also be known by the attacker. Once the attacker manages to send ACKs in path between S and V, the traffic will start hitting the victim machine while source S thinks it is sending the traffic to A. In order to increase the amount of the traffic hitting the victim, the attacker needs to increase the windows size for the path between S and V, in addition to simulate the congestion in the path between the source and the attacker nodes.

The effect of this type of flooding attacks can be significantly increased if the attacker uses more than one streaming server at the same time causing a distributed attack. However, the attacker can repeat the previous scenario with

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

many servers causing the traffic to be redirected from multiple servers at the same time, as shown in figure 5.

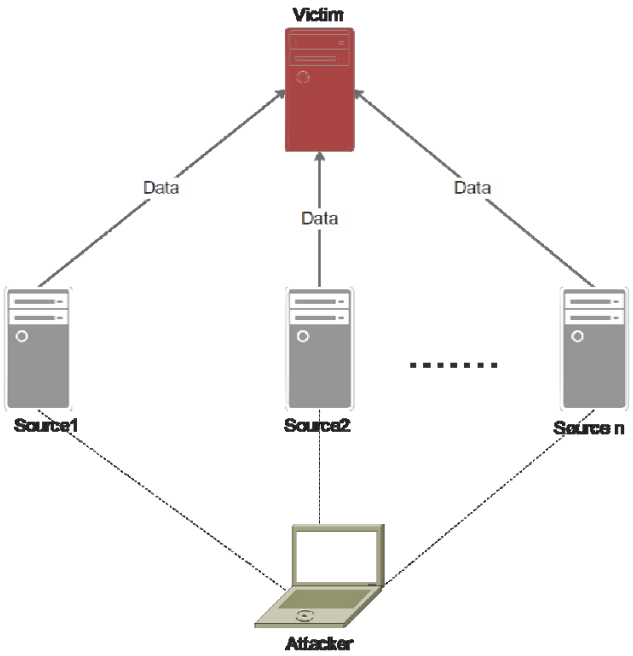

Figure 5. Flooding attack using multiple stream servers

Another type of flooding attacks is MPTCP SYN flooding attack. This attack uses the SYN messages in order to exhaust the victim resources and prevents new sub-flows connections [19]. The attacker starts with a normal MPTCP session by sending regular SYN packet and then sends many MP_JOIN requests as supported by the server, each join message is sent with different source IP and source port combinations. This is an amplification attack, in which the cost on the server side is the cost needed for the initial SYN request in addition to the cost needed for all following SYN MP_JOIN requests. Figure 6 illustrates this attack, the attacker uses a list of N IP addresses to open one MPTCP connection with N-1 sub-flows.

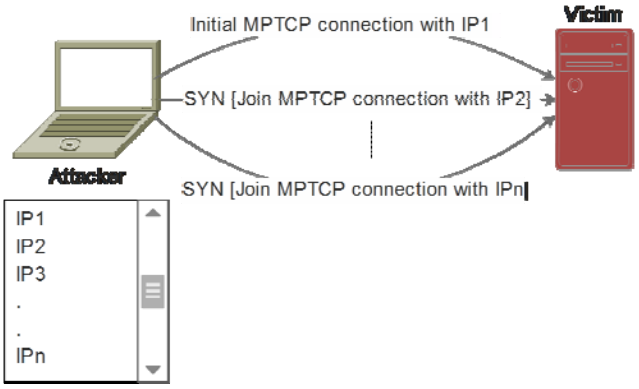

Figure 6. Single connection MPTCP SYN flooding attack

In order to increase the effect of SYN flooding attack, the attacker can use each IP in the list to open a new MPTCP connection instead of joining an existing one, the rest of IP addresses can be used with different ports combinations to join the connection. In this case, the attacker can open N MPTCP connections with N-1 sub-flows as shown in figure 7.

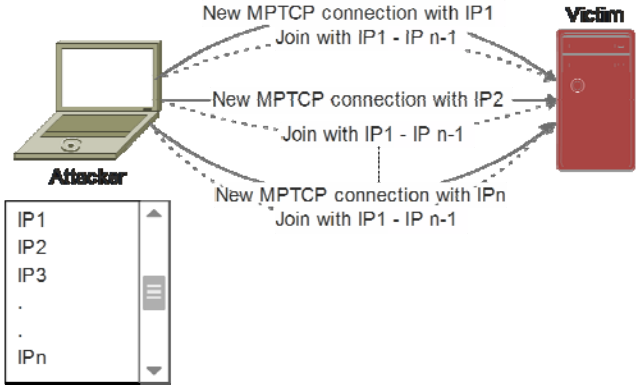

Figure 7. Multiple connections MPTCP SYN flooding attack

VI. MPTCP Hijacking Attacks

In this type of attacks, the attacker attempts to hijack the MPTCP connection in order to personate one of the legitimated peers. It happens after the initial MPTCP connection is established and the two peers are exchanging data. The target from this attack is either eavesdropping or altering the data being transferred between the two peers. Figure 8 shows the general overview of MPTCP hijacking attack.

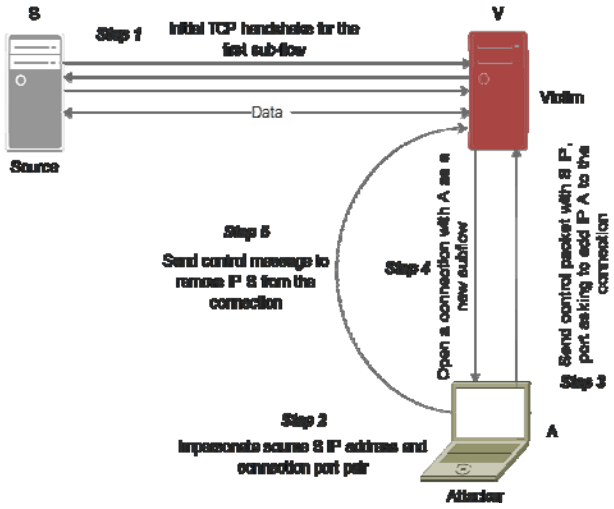

Figure 8. MPTCP hijacking attack

After the connection is established in step 1, the attacker needs to figure out the fourth tuples used to distinguish this connection. This information is needed in order to send a fake control packet which asks the victim machine to open a new sub-flow with the attacker. This request is sent by using S IP address and port number and it includes IP A in ADD_ADDR field in the MPTCP header. Since the request is sent by using source S information, the victim thinks it is a legitimated

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

request so it opens a new sub-flow with the attacker. Now the connection has two sub-flows, the first one between the source and the victim and the second one between the victim and the attacker. At this point, the traffic started to be split between these two connections. In order to complete the attack, the attacker sends another control message to remove IP S from the list of addresses related to the connection. After this step, the hijacking attack is successfully completed and the traffic starts flowing between the victim and the attacker. The attacker may modify or just eavesdropping on the data and then forward it to the source machine. In order to keep the connection alive, the attacker repeats the same procedure with the source machine which makes both peers think they are talking to each other while they are talking to the attacker.

VII. Proposed Algorithm

The idea behind the proposed algorithm is mainly based on three actions. First, we enforce the client by the implementation to send all relevant information about all paths that may come up while the connection is active, we called this information a Metadata. Information includes the physical interface MAC address and possible IP address for the path. This information if it is available will give the server an indication about future sub-flows that may come up. Second, limiting the maximum number of sub-flows for each MPTCP connection, this is already implemented in most Linux distributions [4, 20], we suppose that the maximum limit is five. The third one is using a hash key value to authenticate each sub-flow before allocating the resources on the server side. Suppose there are two hosts A and B, A wants to start MPTCP connection with B. Following steps must be followed in order to mitigate the flooding and hijacking attacks in the connection between A and B.

1. In order to start MPTCP connection, A sends SYN packet with MPTCP_Capable.
2. A includes in the SYN packet the Metadata information about all its candidate sub-flows (interfaces MAC address and possible IP addresses).
3. When B receives the first SYN packet, it stores temporarily the information related to this connection with all candidate sub-flows.
4. B replies with SYNACK packet with a crypto hash key generated related to the connection; the hash key is generated from the Metadata related to this sub- flow in addition to a random value chosen by the server.
5. B also includes a random key in the message. This key is necessary to eliminate the hijacking attacks; each future request to add a new sub-flow will have this key.
6. A stores the random key related to this connection.
7. B stores the hash key and the random key in a table related to this connection as shown in table1.

Table 1. Sub-flows hash table

Connection: Y, Random key: randKey	
Hash value	Is connection active
Hash1	Yes
Hash2	No
Hash3	No
Hash4	No
Hash5	No

8. When A receives SYNACK packet, it responses with ACK packet.
9. A should include the same hash key in the ACK packet.
10. When the ACK packet is received by B, it checks for the hash value.
11. If the hash value is the same as the one which was sent by SYNACK and the value of “is connection active” is no, then this connection is validated and “is connection active” value is set to yes.
12. Now, suppose there is a new sub-flow exists.
13. A sends SYN_JOIN packet to B, it includes the random key obtained in step 6 in the request.
14. B checks the table related to this connection and validate the random key. If it is validated, B continues with next steps.
15. B repeats the same authentication steps mentioned previously in steps 4, 8-11.
16. B checks if the new sub-flow information exists in the Metadata for this connection.
17. If yes, and the new sub-flow is authenticated, then B will add the new sub-flow to the connection.

For the case of the flooding attack described in figures 4 and 5, when the attacker starts a MPTCP connection with the streaming server, Metadata information is sent with SYN packet. If the attacker informs the server that it has a new IP address and it wants to start a new sub-flow, server checks if this IP exists in the Metadata. If not, then the server will immediately ignore the request. If it exists, then the server will send the crypto hash key to authenticate the new sub-flow. This process is repeated for each new sub-flow request. The algorithm grantee that only traffic requested by the host can reach it.

For the scenarios described in figure 6 and figure 7, the attacker wants to perform SYN flooding attack on the victim machine, the maximum number of allowed sub-flows for any MPTCP connection is assumed to be five. The attacker starts with SYN packet which contains MP_CAPABLE option, it forced by the implementation to send Metadata information about all possible sub-flows. When the victim receives the SYN packet, it generates a crypto hash key for the first five sub-flows and stores the values in a table as shown in table1. When the attacker attempts to perform a SYN flooding attack

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

by sending multiple SYN JOIN packets to the victim, for each request the victim calculates the crypto hash key, if the value exists in the table then this request could be legal. The victim continues with authentication process by sending SYNACK packet to the attacker with the hash related to the sub-flow merged with a random hash key. If attacker replies with ACK which contains the same hash key, then the sub-flow request is authenticated and added to the connection paths. The resources related to the sub-flow are only allocated after the sub-flow is validated.

For the case of hijacking attack described in figure 8, when the attacker sends a control packet to add its IP address to an existing MPTCP connection, it should include the random key obtained in the first MPTCP connection initialization as described in step 5 in the proposed algorithm. Since the attacker didn't start the connection it will not have the random key and will not be able to send a valid request to open a new sub-flow with its IP address.

VIII. Conclusion

Supporting multipath over TCP is the most significant change happens to TCP since the first design in the 1970s. It allows the traffic related to a single connection to be split over multiple paths which in term improves the reliability and increases throughput. Due to the address agility provided by MPTCP, new security threats appears, this includes flooding and hijacking attacks. In this article, we analyzed multiple flooding and hijacking attacks scenarios which may occur in any MPTCP connection, we also provided a proposed solution to mitigate these types of attacks.

References

[1] Postel, Jon. "RFC793: Transmission Control Protocol. USC." Information Sciences Institute 27 (1981): 123-150.

[2] Wischik, Damon, Costin Raiciu, Adam Greenhalgh, and Mark Handley. "Design, Implementation and Evaluation of Congestion Control for Multipath TCP." In NSDI, vol. 11, pp. 8-8. 2011.

[3] Bonaventure, Olivier, Mark Handley, and Costin Raiciu. "An overview of Multipath TCP."; login: 37, no. 5 (2012): 17-23.

[4] Ford, C. Raiciu, and M. Handley. "TCP Extensions for Multipath Operation with Multiple Addresses" draft-ietf-mptcpmultiaddress (work in progress)." (2010).

[5] Scharf, Michael, and Alan Ford. Multipath TCP (MPTCP) application interface considerations. No. RFC 6897. 2013.

[6] Díez, Javier, Marcelo Bagnulo, Francisco Valera, and Iván Vidal. "Security for multipath TCP: a constructive approach." International Journal of Internet Protocol Technology 6, no. 3 (2011): 146-155.

[7] Eddy, Wesley M. "SYN Flood Attack." In Encyclopedia of Cryptography and Security, pp. 1273-1274. Springer US, 2011.

[8] "MPTCP Linux Kernel Implementation." MultiPath TCP. Accessed May 01, 2016. http://mptcp.info.ucl.ac.be/.

[9] Scharf, Michael, and Alan Ford. Multipath TCP (MPTCP) application interface considerations. RFC 6897, March, 2013.

[10] Hijawi, H. M., & Hamarsheh, M. M. Performance Analysis of Multi-path TCP Network. International Journal of Computer Networks & Communications (IJCNC), 8(2), 145-157.

[11] "The Dangers and Promise of Multipath TCP." The Dangers and Promise of Multipath TCP. Accessed May 01, 2016. http://www.enterprisenetworkingplanet.com/netsp/the-dangers-and-promise-of-multipath-tcp.html.

[12] Bagnulo, Marcelo. "RFC 6181: Threat Analysis for TCP Extensions for Multipath Operation with Multiple Addresses." Internet Engineering Task Force (2011).

[13] Ford, Alan, Costin Raiciu, Mark Handley, Sebastien Barre, and Janardhan Iyengar. "Architectural guidelines for multipath TCP development." IETF, Informational RFC 6182 (2011): 2070-1721.

[14] Fall, Kevin R., and W. Richard Stevens. TCP/IP illustrated, volume 1: The protocols. addison-Wesley, 2011.

[15] Singh, Ashutosh, Carmelita Goerg, Andreas Timm-Giel, Michael Scharf, and T-R. Banniza. "Performance comparison of scheduling algorithms for multipath transfer." In Global Communications Conference (GLOBECOM), 2012 IEEE, pp. 2653-2658. IEEE, 2012.

[16] Raiciu, Costin, Christoph Paasch, Sebastien Barre, Alan Ford, Michio Honda, Fabien Duchene, Olivier Bonaventure, and Mark Handley. "How hard can it be? designing and implementing a deployable multipath TCP." InProceedings of the 9th USENIX conference on Networked Systems Design and Implementation, pp. 29-29. USENIX Association, 2012.

[17] Eardley, Philip. "Survey of MPTCP implementations." (2013).

[18] Blanchet, Marc, and Pierrick Seite. "Multiple interfaces and provisioning domains problem statement." (2011).

[19] Bagnulo, Marcelo, Olivier Bonaventure, Fernando Gont, Christoph Paasch, and Costin Raiciu. "Analysis of MPTCP residual threats and possible fixes."Analysis (2013).

[20] Barré, Sébastien, Christoph Paasch, and Olivier Bonaventure. "Multipath TCP: from theory to practice." In NETWORKING 2011, pp. 444-457. Springer Berlin Heidelberg, 2011.

AUTHORS

Adwan Yasin is an associate Professor, Former dean of Faculty of Engineering and Information Technology of the Arab American University of Jenin, Palestine. Previously he worked at Philadelphia and Zarka Private University, Jordan. He received his PhD degree from the National Technical University of Ukraine in 1996. His research interests include Computer Networks, Computer Architecture, Cryptography and Networks Security.

Hamzah M. A. Hijawi received his BS in computer system engineering from Birzeit University, Ramallah, Palestine, in 2011. He is currently working with Exalt technologies as a software engineer since 2010, and currently pursuing his Master of Computer Science from Arab American University, Jenin, Palestine. His researches interest include Computer Networks, Information Security, Wireless Sensor Networks and Data Mining.