Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 4, Issue. 6, June 2015, pg.86 – 91

RESEARCH ARTICLE

REMOTE EXTRACTION OF MALICIOUS PAYLOAD FOR SCTP

SUSHMA PATIL[1], B.K.GUDUR[2], V.ANILKUMAR[3]

[1]E&C Department & VTU, India

[2]E&C Department & VTU, India

[3]CSIR-4PI, NAL, India

[1] patilsushmap@gmail.com; [2] bmeshg@gmail.com; [3] anil@cmmacs.ernet.in

Abstract— The Stream Control Transmission Protocol (SCTP) is a message oriented reliable transport layer protocol which uses four way handshake mechanism. It is more robust than TCP, which provides delivery of data between two end points, and message boundaries preservation as in UDP. Additionally it has advantage such as multihoming and multi streaming. These features increase the availability of data chunks and solve problems like head of line blocking, and SYN flooding. In a network, a block of allocated public IP may not have all the IPs to be active or in us; hardly few IPs may be active. Since those IPs are allocated to the particular block, nobody else can use them, and as that block is also not using those IPs, they are completely inactive. It is required to ascertain whether a host is trying to connect to any of these inactive IPs, and if any one tries to connect to them, the intention is not right. They are called as suspicious host. In normal communication, the server receives the request from true client or a malicious client for establishing association. The aim of this work is to demonstrate whether the received SCTP packets are from a trusted or malicious client by developing an active application responder, above SCTP stack; this can be achieved by doing a proper association. Once the connection is established, the packets are sniffed, and analysed to get the information of the fields present in it. An INIT ACK packet is created, according to the sniffed information of the packet, and is sent to the client. If the client responds to INIT Ack, the particular packet is marked as a malicious; as that client wants to connect to the inactive IP.

Keywords— SCTP, TCP, SYN Flooding, Fore way handshake, Scapy

I. INTRODUCTION

The Internet was designed with pertain to the function but not security in mind. For data communication, TCP/IP protocol suite is the most widely used protocol suite. In the Internet infrastructure there is no security built to protect hosts from other hosts and regulating their own behavior. The attackers can forge the source address of IP packet to maintain their security and to redirect the censure for attacks. The spoofing packets consume network bandwidth and often part of some malicious activity, such as a DDoS attack [1]. One of the drawbacks in the present network is that, authentication of the source address. The present network concentrates only on delivery of packets to destination, without considering the authentication aspects of the packet.

The transport layer contains protocols like TCP, UDP and SCTP. Stream Control Transfer Protocol (SCTP) is the new transport protocol designed to solve the some of the problems that occur while using Transmission Control Protocol (TCP). SCTP contains multiple streams. These streams help users to separate unrelated data into different streams and avoid Head of Line Blocking problem. A block of any allocated public IP, may not have all the IPs to be active or in use; hardly 15 or 16 IPs may be active. As these IPs' are allocated to the particular block they are not available to others to use them, and network is also not

using those remaining IPs' they are completely inactive. It is required to ascertain whether a host is trying to connect to any of these inactive IPs, and if any one tries to connect to them, the intention is not right. Such hosts are called as suspicious hosts. To check suspicious host, SCTP protocol is used, which provides secure communication compare to other transport layer protocol. This protocol uses four-way handshake mechanism for association. SCTP protocol does not suffer from SYN flood attack where as the TCP protocol suffers with this attack. Table 1 gives the features of transport layer protocols.

II. RELATED WORK

SYN flood attack [2] [3] exploits vulnerability of the TCP three-way handshake mechanism, namely the server needs to allocate a large data structure for any incoming SYN packet regardless of its reality. During this attack, the attacker sends SYN packets with source IP address that do not exist or are not in use. In three-way handshake mechanism, when the server puts the request information into the memory stack, it will wait for the confirmation from the client that sends the request. The request is waiting to confirmed, it will remain in the memory stack. The source IP address used in SYN flood attacks are nonexistent, the server will not receive confirmation packets for requests created by the SYN flood attack. Every half-open connection will remains on the memory stack until it times out, and it will retransmit the SYN-ACK 5 times, after each retransmission, it doubles the time-out value. Initial time out value is 3 seconds, and it retries at 3, 6, 12, 24 and 48 seconds. After retransmission, more and more request will occupy and fill up memory stack. In the above three-way hay handshake, process the server will remain in half-open connection state before receiving final ACK packet.

In the three-way handshake mechanism client sends SYN packet to server, the server replies with SYN+ACK, after receiving SYN+ACK packet the client send ACK packet. But in SYN flood attack client sends SYN packet continuously, and server replies with SYN+ACK. The client sends again SYN (spoofed) packet, with observing this SYN packet reply, server detects it is SYN flood attack. If the memory stack is filled up, no new request, including legitimate requests, can be processed and the services of the system are disabled.

The space for the memory stack allocated by the operating system is small, and even a small scale SYN flood attack can be descriptive. Fig 1.1 shows the SYN Flood attack, this attack accomplished by compromised machine, by using spoofed IP address/genuine source IP address given these compromised machines are configured to ignore the SYN/ACK packets from the target. The TCP carries 95% of Internet traffic and 80% the total number of flows in the Internet. TCP SYN flooding [4] [5] attack is the most common type of DDoS attack.

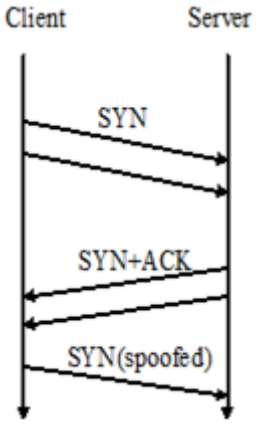

Fig 1.1: SYN Flooding

A. IP Spoofing Today

IP spoofing [6] [7] used to commit criminal activity online and to breach network security. IP spoofing refers to connection hijacking through a fake Internet Protocol (IP) IP address. IP spoofing is the action of masking a computer IP address so that it looks like it is authentic. In IP spoofing, IP headers are masked through a form of Transmission Control Protocol (TCP) in which spoof discover and then manipulate vital information contained in the IP header such as IP address and source information and destination information. Spoofing continues to be severe problem in today Internet word and it is still a prospective tool used by malicious users for attack and misdirection. Using ingress/egress filters it may be concluded that attackers not able to spoof many address. Using following technique TCP SYN flood [8] can be prevent, using the server as the detector of the attack and the local router of the attacker is used to prevent the attack. For establishing the TCP connection with the server, every client should send the SYN signal and have to respond the SYN/ACK signal with the ACK signal. In addition, to identify the attack, the SYN request sent by the client is stored in the server data table until the acknowledgement from the client received by the server for the SYN ACK signal, the client information stored in the table is the IP address and the SYN count. If the count in the table exceeds the threshold limit, the victim closely acquainted details of the attack to its local router

and it will sent to the local router of the attacker to drop all the packets from the respective node. The proposed scheme needs to develop further to accommodate a particular environment.

B. **SYN cookie**

SYN cookie is a technique used to resist SYN flood attacks. This method can be use to prevent the opening of connections to spoofed source addresses. The use of SYN cookies allows a server to avoid dropping connections when the SYN queue fills up. When the server is using SYN cookies [9] it does not allocate resources to a connection until the three-way TCP handshake competes. In this, first the server sends a SYN+ACK packet with a specially encoded initial sequence number, or cookie, that includes a hash of the TCP headers from the client's initial SYN packet, a timestamp, and the client's Maximum Segment Size (MMS). When server receives the client's response, the server can check the sequence number and create the necessary state only if the client's sequence number is the cookie value plus one. Because the cookie uses a hash involving the server's secret key, attackers should not be able to guess the correct cookie values, because of the performance concerns and some incompatibilities with TCP extensions, such as large windows. Operating systems generally do not activate SYN cookie mechanism until the host's SYN queue fills up. When the server is using SYN cookie's it does not allocate resources to a connection until the three-way TCP handshake competes. An attacker send spoofing traffic at a low rate may avoid triggering the SYN cookie mechanism.

III.SCTP PACKET FORAT

The Stream Control Transmission Protocol (SCTP) is a transport layer protocol. It has picked up the best features of UDP and TCP and combined them together. In SCTP a message can be divided in to multiple streams of DATA chunks and can be delivered in multiple packets [10]. However, a DATA chunk cannot contain multiple messages. The control chunks are separate from DATA chunks. An SCTP packet may contain multiple chunks. DATA chunks and control chunks can be bundled in an SCTP packet.

The structure of a common SCTP packet is shown in fig 3.1. It contains a common header and one or more chunks. Each chunk contains a chunk header and chunk value. Chunk value contains different options, control information if any and data if it is a DATA chunk.

Bits	0-7	8-15	16-23	24-31
+0	SCTP Common Header			
96	Chunk 1 Header			
128	Chunk Value			
…	…			
…	Chunk n Header			
…	Chunk Value			

Fig 3.1: SCTP Common header

In SCTP each packet contains a common header of 12 bytes. It contains source port and destination port, each of size 16 bits, verification tag (Vtag) and checksum each of size 32 bits.

In TCP the communication between two end points is called a "connection". In SCTP The communication relation is called "association" this is because it is broader connection than a single connection [11].

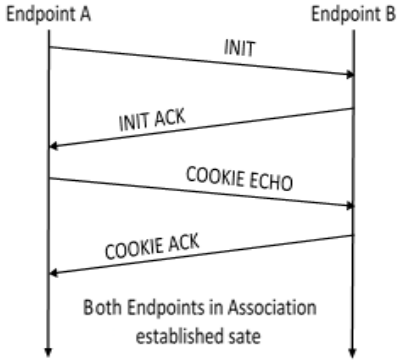

Fig 3.2: SCTP Association

Fig 3.2 shows the SCTP association, in this endpoint A sends an INIT signal to Endpoint B to initiate an association. The Vtag field is set to zero and an Initiate Tag generated randomly. 'A' also selects an Initial TSN it will use for its first DATA to be sent and also defines the maximum inbound and outbound streams. The INIT includes all the IP addresses it will use for this association (For Multi-homing).

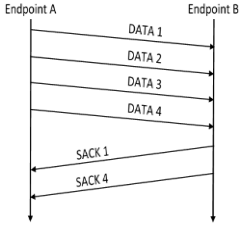

Fig 3.3: SCTP Data transfer

Upon receiving INIT, Endpoint B stores the Initiate Tag as its Vtag and includes it in the common header of all future packets, Initial TSN as peer Initial TSN and all the IP addresses included. Then 'B' sends an INIT ACK to Endpoint A. It contains an Initiate Tag randomly generated by 'B', an Initial TSN is used by 'B' for DATA sending, maximum inbound and outbound streams. Endpoint B also generates a COOKIE using its Private Key and includes it as a parameter to INIT ACK. When INIT ACK reaches Endpoint A, 'A' does the same work with INIT ACK as 'B' does with INIT. It selects the minimum between its own and 'B''s inbound and outbound streams. It takes the cookie from INIT ACK and sends it in a COOKIE ECHO to 'B'. When 'B' receives COOKIE ECHO, it responds with a COOKIE ACK and goes into association-established state.
When 'A' receives COOKIE ACK, it goes in to association-established state. In SCTP resource allocation occurs only when association is established, not before that. Hence, it prevents the SYN Flooding attack of TCP. Figure 3.3 shows the SCTP data transfer.

- An endpoint sends DATA and receives SACK. Number of DATA packet an endpoint can send without receiving a SACK depends on the window size.
- The first DATA packet always contains the Initial TSN shared during association and responded with a SACK. Later, a SACK can acknowledge multiple DATA packets.
- If a DATA has received more than once, an endpoint can say that it has received a duplicate packet in the duplicate field of its SACK.
- If a gap in DATA occurred due to data loss, a SACK is sent with the Cumulative TSN ACK set to the last DATA received in sequence and the gap is informed in the gap field. Hence only the mentioned DATA chunks are retransmitted rather than all the DATA packets from the gap.
- In SCTP a SACK is not retransmitted as the next SACK will acknowledge all the previous DATA and hence optimizes use of the bandwidth.

IV. RESULTS AND DISCUSSIONS

The client sends the INIT packet to initiate association, first four fields such as source port, destination port, tag and checksum belongs to the SCTP common header as shown in figure 4.1. The verification tag is zero as per RFC, the chunk header contains chunk type, chunk flags, chunk length and chunk value. In the INIT chunk the init_tag is a random number and init_tsn is the transmition sequence number which is used in data chunk as transmission sequence number, for every data chunk it is same, as it is coming from the client.


```
###[ SCTP ]###
        sport     = 39362
        dport     = 5001
        tag       = 0x0
        chksum    = 0x522ee74d
###[ SCTPChunkInit ]###
           type      = init
           flags     = 0x0
           len       = 36
           init_tag  = 0x766832f
           a_rwnd    = 106496
           n_out_streams= 10
           n_in_streams= 65535
           init_tsn  = 0x5be51d3a
           \params    \
```

Fig 4.1: INIT Chunk

After receiving the information from INIT packet end B creates INIT ACK. This chunk contains the common header and INIT ACK chunk information. The verification tag is the Initiate Tag of INIT.

The init_tag and init_tsn values are generated randomly. The init_tsn is use for creating the SACK. The number of inbound streams and number of outbound streams should be same as the number of outbound streams of the INIT chunk, it should not be less than outbound streams of INIT chunk. The important field in the INIT-ACK chunk is SCTP Chunk Param State Cookie. The cookie contains timestamp, MAC and TCB. The init_tag and init_tsn is same for the packets header B. the created INIT Ack is send to A. this packet is created to check whether the packet received from A is malicious or true. In this work the cookie value is taken as random number for analysing purpose. This created packet is send to endpoint A, and wait for Cookie echo. If end point A sends Cookie. Echo the that packet is marked as malicious, because it is trying to connect to the inactive IP and if it is not responding with Cookie Echo the connect is terminated.

The created INIT ACK is as shown in figure 4.2. In this case the endpoint A sends Cookie Echo because it is trying to establish the connection. The cookie echo packet contains SCTP header and SCTP Chunk Cookie Echo, the verification tag is the init_tag of the INIT ACK chunk, this value is used to identify the packet which is coming from the end point A.

```
###[ SCTP ]###
        sport     = 5001
        dport     = 39362
        tag       = 0x766832f
        chksum    = 0xab0f5052
###[ SCTPChunkInitAck ]###
           type      = init-ack
           flags     = 0x0
           len       = 36
           init_tag  = 0xf5d5c056
           a_rwnd    = 106496
           n_out_streams= 10
           n_in_streams= 10
           init_tsn  = 0x1363db54
           \params    \
            |###[ SCTPChunkParamStateCookie ]###
            |  type      = state-cookie
            |  len       = 16
            |  cookie    = '4294967295\x00\x00'
.
Sent 1 packets.
```

Fig 4.2: INIT Ack

SCTP Chunk Cookie Echo contains chunk type, flags, length of chunk and cookie. The Cookie Echo packet is as shown in figure 4.3, the cookie value is same as in the INIT ACK cookie value. It is just echo the cookie part of INIT ACK, because it is trying to connect with end B.

```
###[ SCTP ]###
        sport     = 39362
        dport     = 5001
        tag       = 0xf5d5c056
        chksum    = 0xbfabb4be
###[ SCTPChunkCookieEcho ]###
           type      = cookie-echo
           flags     = 0x0
           len       = 16
           cookie    = '4294967295\x00\x00'
.
Sent 1 packets.
```

Fig 4.3: Cookie Echo

With receiving the Cookie Echo end point B marks the received packet as malicious. And end point B is interested to see the payload from malicious packet. End point B sends Cookie Ack. This packet contains SCTP header along with SCTPChunkCookieAck packet. It is used to acknowledge the received Cookie Echo packet. After receiving Cookie ACK end point A starts sending data.

With observing the result it is clear that the endpoint A is sending malicious packets and is trying to connect to the inactive IP. This is the new style, attacker is using for doing attack with this attacker creates the background traffic. With this it clear that SCTP is secure protocol, as SCTP allocates the resources after four way handshake completes, that is after SACK. In this case endpoint B is not sending the SACK, the resource is not allocated still end point B receives data. Using SCTP protocol the data is analyzed and it is secure compare to other protocols.

V. CONCLUSION

This paper demonstrates the features of transport layer protocol SCTP and establishing the association for end points. Whenever a client wants to communicate with the server, first it sends INIT with appropriate fields such as source port, destination port, verification tag, init tag, init TSN etc. After the analysis of the received packet, INIT Ack is created and sends to the client. As INIT Ack contains the cookie, a random cookie is generated for demonstration purpose and sent to the client. The client generates Cookie echo with the same cookie value, with this server marks that packet as malicious. If cookie value is different, the server does not respond to the cookie echo. In this work, the client is checked and if found to be malicious, it is mandatory to ascertain the data it is sending. Whenever the client wants to communicate with the inactive IP, an analysis of the INIT packet, creation of INIT Ack and receipt of cookie echo is sufficient to deduce that the client is malicious. Even though the server knows the client as malicious, it creates cookie ack to determine what data is being sent by the client. Thus, with this discussion it can conclude that SCTP is a secure protocol.

REFERENCES

[1]. N. Meghanathan et al. (Eds.), "***An Effective Approach to Detect DDos Attack",*** Advances in Computing & Inf. Technology, AISC 178, pp. 339–345 springerlink.com © Springer-Verlag Berlin Heidelberg 2013.

[2]. Jin and S. Lin**, "*Analysis and Protection of SYN Flood Attack",*** CSISE 2011, AISC 106, pp. 183–187, 2011.

[3]. Deepak Singh Rana et al ,Int.J, "***A Study and Detection of TCP SYN Flood Attacks with IP spoofing and its Mitigations",*** Computer Technology & Applications,Vol 3 (4), 1476-1480.

[4]. D.Deephi Rani, T.V.Sai Krishna, G.Dayanandam "***TCP Syn Flood Attack Detection And Prevention"*** IJCTT 2013.

[5]. Inwhee Joe, "***SCTP with an Improved Cookie Mechanism for Mobile Ad-Hoc Networks"*** College of Information and Communications Hanyang University.

[6]. D. J. Bernstein, "***SYN Cookies,"*** 1997 [Online]. Available: http://cr.yp.to/syncookies.html

[7]. Jorge Mena Ryan Rusich UC Riverside UC Riverside, ***"SCTP: Stream Control Transmission Protocol An Analysis".***

[8]**. R. Stewart , "*Stream Control Transmission*"** RFC 4960, October 2007.

[9]. Rahul Choudhari, Somanath Tripathy, "***SCTP-Sec: A secure Transmission Control Protocol".***

[10]. Randall Stewart, Chris Metz "**SCTP New Transport Protocol for TCP/IP**" 1089-7801/01 2001 IEEE.

[11]. Wojciech Fraczek, ***"Stream Control Transmission Protocol Steganograph.***

