Reproducing Article: ”Detecing TCP ACK storm attack: a state transition modelling approach”

KAR CHUN TEONG, HONG XU LI, TIAN YI HU, ZHONG YU HUANG

Abstract
In this project, the authors reproduce the experiment of an article named ”Detecing TCP ACK storm attack: a state transition modelling approach”. The main work includes the implementation of state transition model, capture and analysis of TCP packets, simulation of ACK storm on a local machine, and simulation of the whole process of using the state transition model to detect the locally generated ACK storm.

Keywords
TCP, ACK storm, State Transition Model

Contents

Introduction

0.1 ACK storm attacks

ACK storm attacks are injection attacks against an Transmission Control Protocol (TCP) connection. This attack requires sending two packets by the adversary with acknowledgement number greater than the sequence number used in each direction and the two end hosts will attempt to resynchronise the sequence numbers by sending duplicate acknowledgement and enter a loop, thus rendering the connection useless, the process is shown in Figure 1.

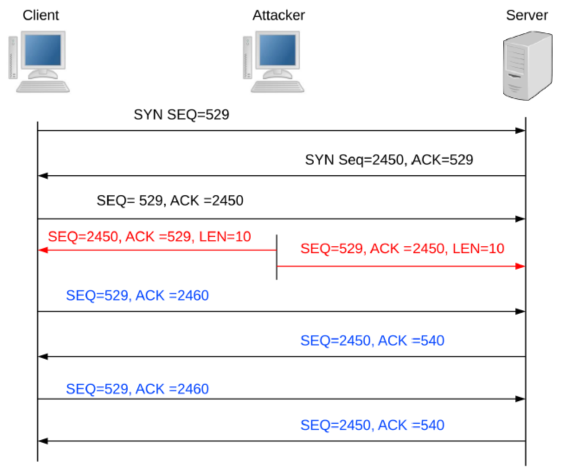


Figure 1. Simple ACK Storm Attack

0.2 Transmission Control Protocol

TCP is the majorly used transport layer protocol in today’s Internet, it is vulnerable to trivial man-in-the-middle (MitM) attacks. This design flaw makes it vulnerable to ACK storm attacks. As many applications use TCP as transport layer protocol, these design flaws of TCP can virtually impact any application used in the Internet today.

0.3 State Transition Model

To detect the ACK storm attack in TCP, the authors proposed a state transition model, the state transition model is defined as shown in Figure 2. The states included:

1. q0: there are no unacknowledged bytes in the TCP connection
2. q1: there are bytes to be acknowledged in the TCP connection
3. q2: abnormal acknowledgement packet received
4. Storm: acknowledgement storm detected

The transition table for each states is shown in Figure 3, the variable *c* shown in both figures indicates the counter of instances of $ACK > SEQ$, as the counter surpasses the preset threshold α, the state transition model deducts that an ACK storm occurs.

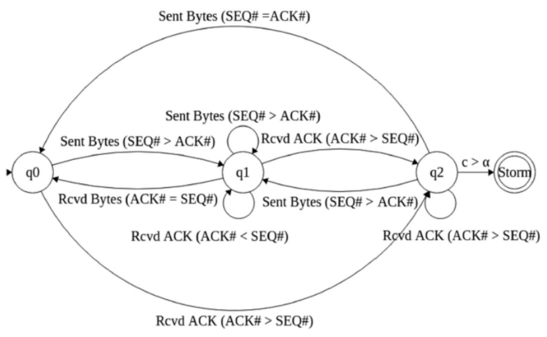


Figure 2. State Transition Model

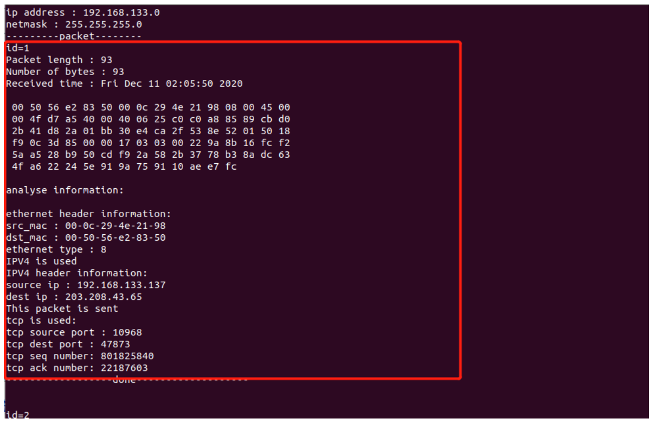
Current state	Next state	Input symbol	Constraint	Counter update
q_0	q_1	Sent Bytes	SEQ# > ACK#	—
q_0	q_2	Rcvd ACK	ACK# > SEQ#	$C \leftarrow 1$
q_1	q_1	Sent Bytes	SEQ# > ACK#	—
q_1	q_0	Rcvd ACK	SEQ# = ACK#	—
q_1	q_1	Rcvd ACK	ACK# < SEQ#	—
q_1	q_2	Rcvd ACK	ACK# > SEQ#	$C \leftarrow 1$
q_2	q_2	Rcvd ACK	ACK# > SEQ#	$C \leftarrow C + 1$
q_2	q_1	Sent Bytes	SEQ# > ACK#	—
q_2	q_0	Sent Bytes	SEQ# = ACK#	—
q_2	*Storm*	ϵ	$C \geq \alpha$	—

Figure 3. Transition Table

1. Implementation

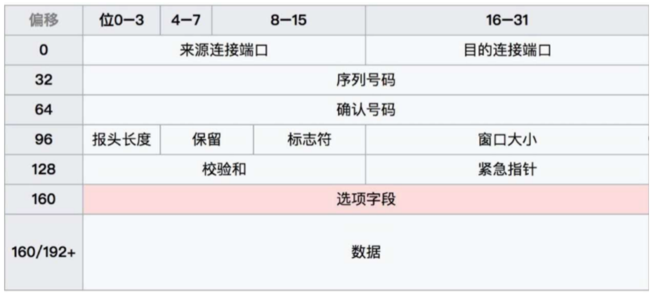
1.1 Capture and Analysis of TCP Packets

We use libpcap library to capture, filter, and analyse the TCP packets, the capture and analysis result of a TCP packet is shown in Figure 4. To achieve accurate analysis, we need to fully grasp the structure of TCP header, which is shown in Figure 5.



```
ip address : 192.168.133.0
netmask : 255.255.255.0
---------packet--------
id=1
Packet length : 93
Number of bytes : 93
Received time : Fri Dec 11 02:05:50 2020

 00 50 56 e2 83 50 00 0c 29 4e 21 98 08 00 45 00
 00 4f d7 a5 40 00 40 06 25 c0 c0 a8 85 89 cb d0
 2b 41 d8 2a 01 bb 30 e4 ca 2f 53 8e 52 01 50 18
 f9 0c 3d 85 00 00 17 03 03 00 22 9a 8b 16 fc f2
 5a a5 28 b9 50 cd f9 2a 58 2b 37 78 b3 8a dc 63
 4f a6 22 24 5e 91 9a 75 91 10 ae e7 fc

analyse information:

ethernet header information:
src_mac : 00-0c-29-4e-21-98
dst_mac : 00-50-56-e2-83-50
ethernet type : 8
IPV4 is used
IPV4 header information:
source ip : 192.168.133.137
dest ip : 203.208.43.65
This packet is sent
tcp is used:
tcp source port : 10968
tcp dest port : 47873
tcp seq number: 801825840
tcp ack number: 22187603
-----------------done--------------------


id=2
```

Figure 4. Analysis of TCP packet



偏移	位0–3	4–7	8–15	16–31
0	来源连接端口			目的连接端口
32	序列号码			
64	确认号码			
96	报头长度	保留	标志符	窗口大小
128	校验和			紧急指针
160	选项字段			
160/192+	数据			

Figure 5. TCP header

In the process of capture and analysis of TCP packets, one of the problems we faced is the connection might be communicating with several different IP addresses at the same time, causing the TCP packets received harder to be organized. The first challenge emerged from this problem is how to identify the different packets' destinations, this can be solved relatively easily, which is by checking the source IP address of the packets.

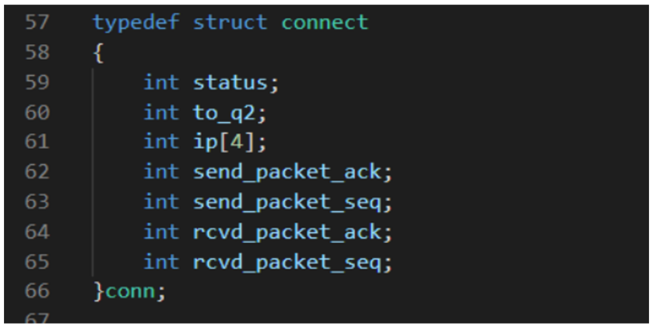
However, as we need to record every received packet corresponding to sent packet of specific IP address to keep track of the state of connection, the other problem emerged is how to specifically locate the corresponding packets within the captured unorganized packets.

1.2 State Transition Model

To solve the previously mentioned problem and to implement the state transition model, we use a data structure named connect to store all necessary information, the structure is shown in Figure 6.

The uses of each data field are listed below:

- status: corresponds to states q0, q1, q2 and storm.
- to_q2: corresponds to the counter of instances of $ACK > SEQ$.
- ip[4]: corresponds to connecting IP addresses.
- send_packet_ack: corresponds to ACK number of sent packet.

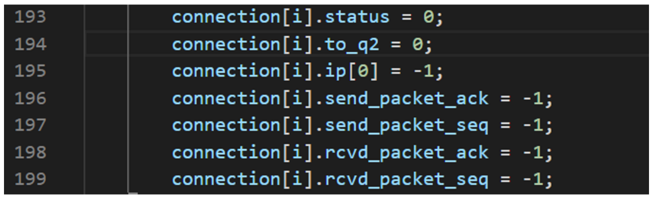


```
typedef struct connect
{
    int status;
    int to_q2;
    int ip[4];
    int send_packet_ack;
    int send_packet_seq;
    int rcvd_packet_ack;
    int rcvd_packet_seq;
}conn;
```

Figure 6. Data Structure

- send_packet_seq: corresponds to SEQ number of sent packet.
- rcvd_packet_ack: corresponds to ACK number of received packet.
- rcvd_packet_seq: corresponds to SEQ number of received packet.

We create an array using this data structure, each element in the array corresponds to one connection with an IP address, and we initialize each element as shown in Figure 7.



```
            connection[i].status = 0;
            connection[i].to_q2 = 0;
            connection[i].ip[0] = -1;
            connection[i].send_packet_ack = -1;
            connection[i].send_packet_seq = -1;
            connection[i].rcvd_packet_ack = -1;
            connection[i].rcvd_packet_seq = -1;
```

Figure 7. Initialization

As we initialized the array, we do the following steps for every packets the connection send and receive:

1. Check IP address
   1.1. Check ip[4] to see if the IP address is stored.
   1.2. If not stored: create a new element to store it.
   1.3. If stored: go to step 2.
2. Renew element's value
   2.1. Determine if the packet is sent or received.
   2.2. Sent packet: renew corresponding variables with the value of ACK and SEQ.
   2.3. Received packet: renew corresponding variables with the value of ACK and SEQ.
3. Renew status
   3.1. Determine if the corresponding value is renewed
   3.2. If value renewed: renew the value of status according to the transition rules.
   3.3. If value remained: skip.

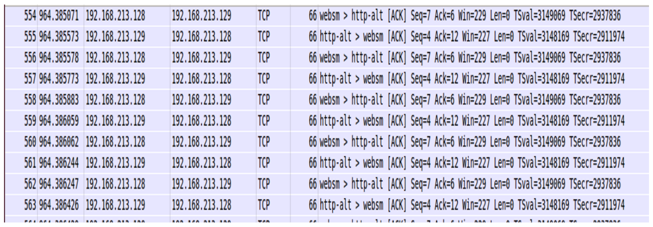
At this point, the state transition model is successfully implemented with the ability to analyse any sent and received TCP packets and record the status of the connection according to the transition rules.

1.3 ACK Storm Attack

After implementing the state transition model, we need to implement the ACK storm attack to test the effectiveness of our model. The implementation steps are shown below:

1. Use nc command to create TCP connection.
   1.1. Server: nc -p 1080.
   1.2. Client: nc -p 9090 192.168.213.129 8080
2. Use wireshark to capture TCP packets, gain the ACK and SEQ number.
3. Third party uses scapy to falsify information.

The result of the ACK storm attack is shown in Figure 8.



```
554 964.385071 192.168.213.128   192.168.213.129   TCP      66 websm > http-alt [ACK] Seq=7 Ack=6 Win=229 Len=0 TSval=3149069 TSecr=2937836
555 964.385573 192.168.213.129   192.168.213.128   TCP      66 http-alt > websm [ACK] Seq=4 Ack=12 Win=227 Len=0 TSval=3148169 TSecr=2911974
556 964.385578 192.168.213.128   192.168.213.129   TCP      66 websm > http-alt [ACK] Seq=7 Ack=6 Win=229 Len=0 TSval=3149069 TSecr=2937836
557 964.385773 192.168.213.129   192.168.213.128   TCP      66 http-alt > websm [ACK] Seq=4 Ack=12 Win=227 Len=0 TSval=3148169 TSecr=2911974
558 964.385883 192.168.213.128   192.168.213.129   TCP      66 websm > http-alt [ACK] Seq=7 Ack=6 Win=229 Len=0 TSval=3149069 TSecr=2937836
559 964.386059 192.168.213.129   192.168.213.128   TCP      66 http-alt > websm [ACK] Seq=4 Ack=12 Win=227 Len=0 TSval=3148169 TSecr=2911974
560 964.386062 192.168.213.128   192.168.213.129   TCP      66 websm > http-alt [ACK] Seq=7 Ack=6 Win=229 Len=0 TSval=3149069 TSecr=2937836
561 964.386244 192.168.213.129   192.168.213.128   TCP      66 http-alt > websm [ACK] Seq=4 Ack=12 Win=227 Len=0 TSval=3148169 TSecr=2911974
562 964.386247 192.168.213.128   192.168.213.129   TCP      66 websm > http-alt [ACK] Seq=7 Ack=6 Win=229 Len=0 TSval=3149069 TSecr=2937836
563 964.386426 192.168.213.129   192.168.213.128   TCP      66 http-alt > websm [ACK] Seq=4 Ack=12 Win=227 Len=0 TSval=3148169 TSecr=2911974
```

Figure 8. ACK storm

1.4 Disconnection when ACK storm attack detected

We were trying to implement in C++ to auto disconnect the TCP connection when an ACK storm attack is detected, however, there's no such library implemented under C++, our alternative solution is to manually input killcx ip: port command in the command prompt to disconnect. This is a weak alternate since it means that the state transition machine isn't fully automated, but regardless we implemented the full process of real time detection of ACK storm attack using a state transition machine.

2. Experiments

2.1 Running the demo

The full code of demo is in the file xx, the code is meant to be run in the command prompt as per instruction.

2.2 Building and monitoring TCP connection

When the state transition machine is monitoring the local machine, we open a connection with baidu.com, we can see that the state transition machine detected that the local machine built a connection with the IP address 182.61.200.7, by using the IP address identification, we know that 182.61.200.7 is the IP address of baidu cloud, which means that the state transition machine correctly monitor the connections built by the local machine.

2.3 Detecting ACK storm attack

todo

3. Contributions

- KAR CHUN TEONG: Researching and selecting article as project topic, designing and creating PPT for mid-term and final presentation, writing final report.
- HONG XU LI: Implementing state transition model.
- TIAN YI HU: Testing and debugging.
- ZHONG YU HUANG: Implementing ACK storm attack.