Name: Thomas Finn

Email: [thomasjf@usc.edu](mailto:thomasjf@usc.edu)

User: usc430af

QUESTIONS

1. Explanation how the TCP SYN flood attack works.

A TCP SYN flood attack is a form of DoS attack that consumes a large amount of resources for a target system thus making the system unresponsive to legitimate traffic. When a TCP connection is made to a server, a client attempts a handshake which involves three steps. First the client sends a SYN or synchronize message to the server to request the connection. Then the server sends back a SYN-ACK or an acknowledgment message to agree to the connection. Lastly, the client responds with an ACK to acknowledge the servers reply thus opening a viable connection. The SYN flood takes advantage of the necessary steps of this handshake and works by not completing the last step which is responding to the server with the expected ACK message. The attacker can do this by either simply not sending the ACK, or spoofing the source IP in the SYN so that the server will send the SYN-ACK to a different falsified IP thus never leading the client to respond in the first place. The server will wait for a timeout period before dropping the request thus using up resources. By sending many half-open connection, the attacker forces the server to exceed the resources available and drop any other legitimate requests made after the fact.

1. Explanation how SYN cookies work to prevent denial-of-service effect from SYN flood attack

SYN cookies are a common technique used to resist flood attacks. One of the values in the initial SYN packets sent by the client is a sequence number. SYN cookies are initial sequence numbers that are carefully constructed according to the following:

* Let t be a slowly increasing timestamp with a resolution of 64 seconds
* Let m be the maximum segment size (MSS) value that the server would have stored in the SYN queue entry
* Let s be the 24 bit result of a cryptographic hash function of the server and clients IP address and port number and the value t
* The SYN cookie is computed as follows:
  + Top 5 bits: t mod 32
  + Middle 3 bits: An encoded value representing m
  + Bottom 24 bits: s

SYN cookies deflect DoS by allowing the server to avoid dropping connections when the SYN queue fills up. Instead, if the queue is nearing completion, the server behaves as if the SYN queue has been enlarged. The server sends back the appropriate SYN-ACK response to the client but discards the SYN queue entry. If the server receives a subsequent ACK response from the client that uses an n+1 acknowledgement number where n is the initial sequence number, then the server is able to reconstruct the SYN queue entry using the information encoded in the TCP sequence number. The server will check the value t to see if the connection has expired, recompute s to determine if it is a valid SYN cookie, and decode m which can reconstruct the entry.

1. Your legitimate client script

|  |
| --- |
| #!/bin/bash  while true;  do  wget server &  sleep 1  rm index\*  done |

1. Your attack command (for flooder)

|  |
| --- |
| Attack command: sudo flooder --dst server --highrate 10000 --proto 6 --dportmin 80 --dportmax 80 --src 1.1.2.0 --srcmask 255.255.255.0 |
| Server tcpdump command: sudo tcpdump ip -i eth1 -nn -tt To write to file: sudo tcpdump ip -i eth1 -w dump.pcap -nn -tt To read from file: sudo tcpdump ip -i eth1 -r dump.pcap -nn -tt | less |

1. The connection duration graphs you drew in task 5 (one with SYN cookies, one without SYN cookies). Indicate on the graphs using vertical lines or arrows the start and the end of the attack.

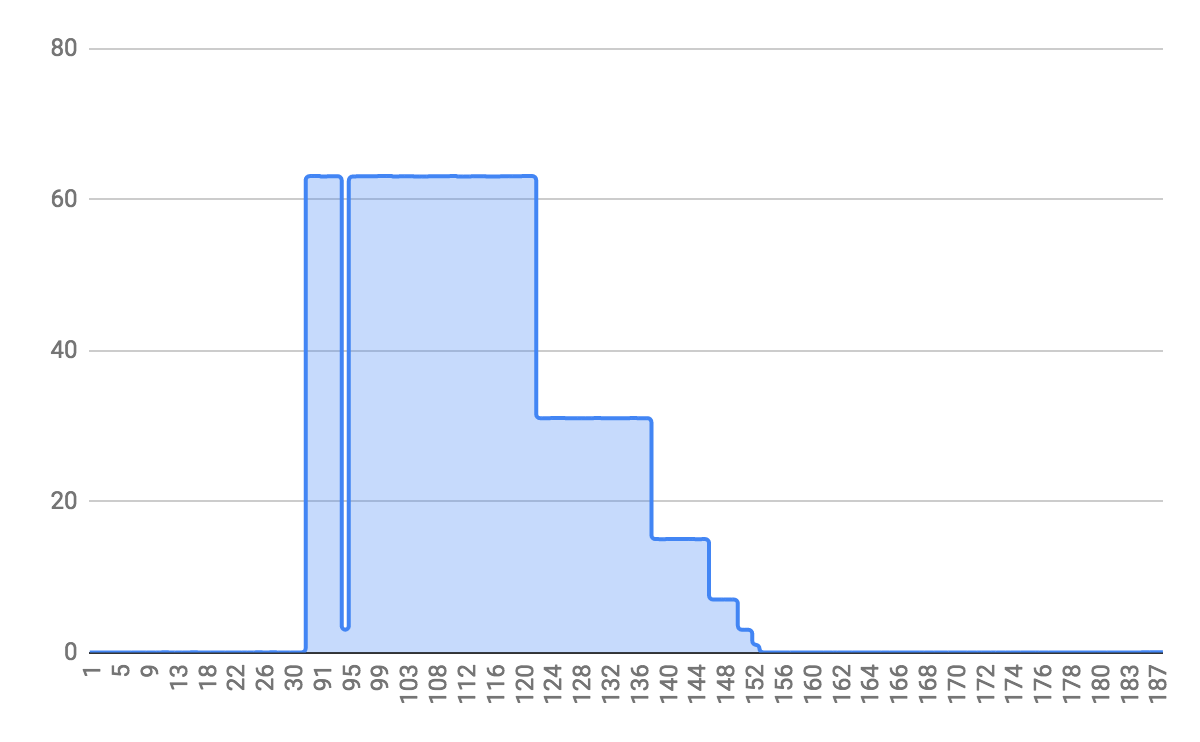
  
 **…………………...[S]======ATTACK=====[F]……………...………….....................**

Image 1: Without SYN Cookies

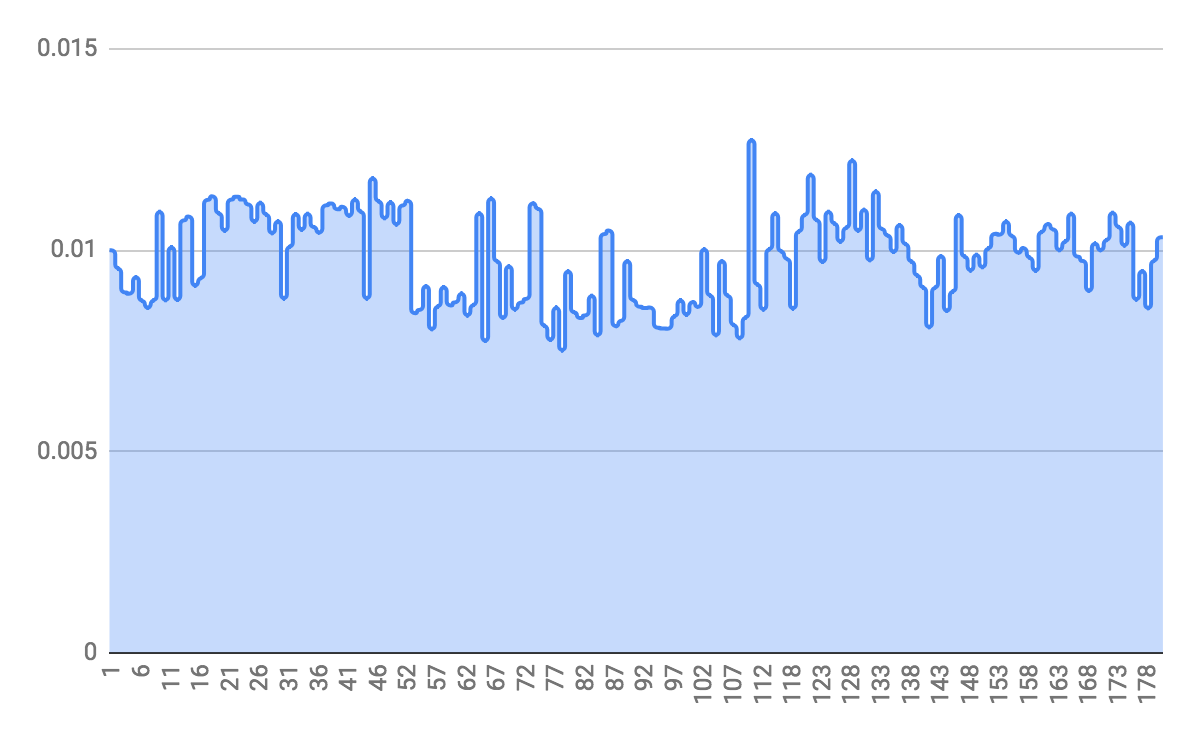
  
**…………………...[S]=============ATTACK===========[F]........................**

Image 2: With SYN Cookies

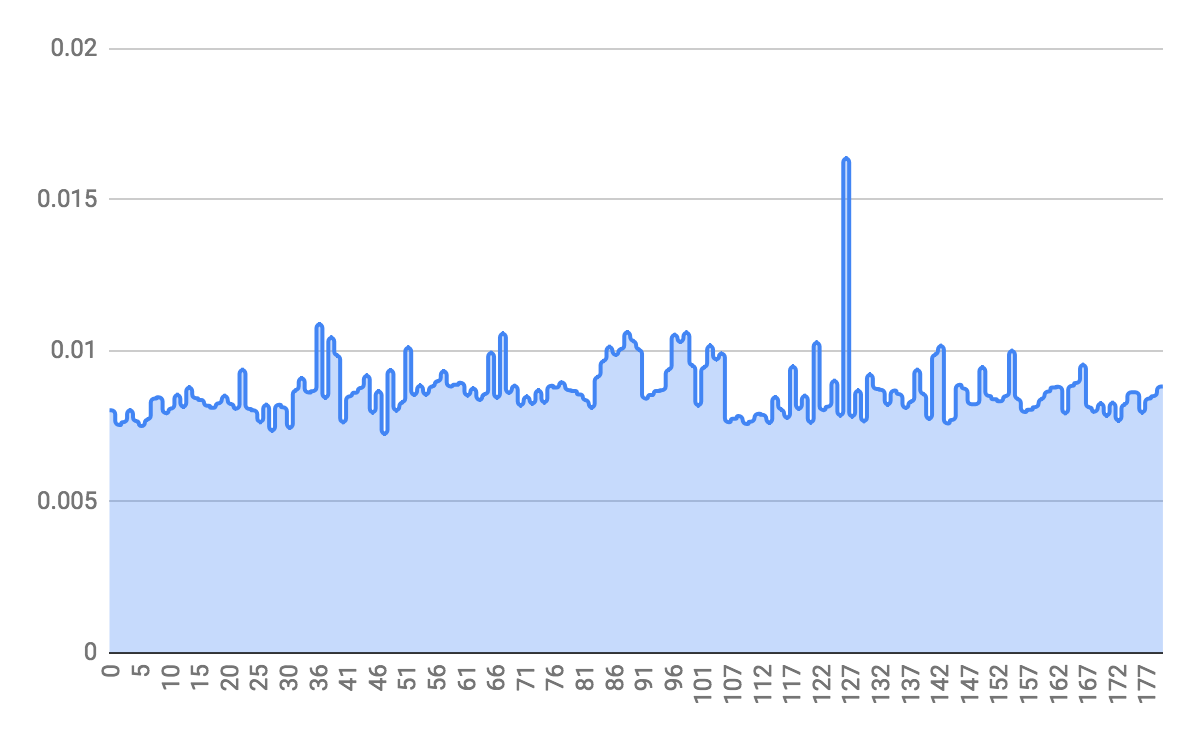
1. Explanation what happens in each case. Is the attack effective? How can you tell this from the graphs?

For the first case where SYN cookies are absent from the trial, the attack is very effective. For both trials the attack started at the 30 second mark and went to the 150 second mark. During the time of the attack, congestion became very apparent and the html requests made in the beginning had to wait the longest to get a response. It has a general downward slope because when the attack ends, the server starts to quickly respond to the queued requests meaning the later requests will have to wait a shorter time. In the case where SYN cookies are present in the trail, the attack is basically unnoticeable. The slope of the estimated trendline is almost 0 which means the attack does little to deter the server from replying to requests. If anything, in the middle of the attack, the response time seems to dip lower than when the attack was not present.

1. Answers to extra credit questions if any.
   * Refer to extra credit section

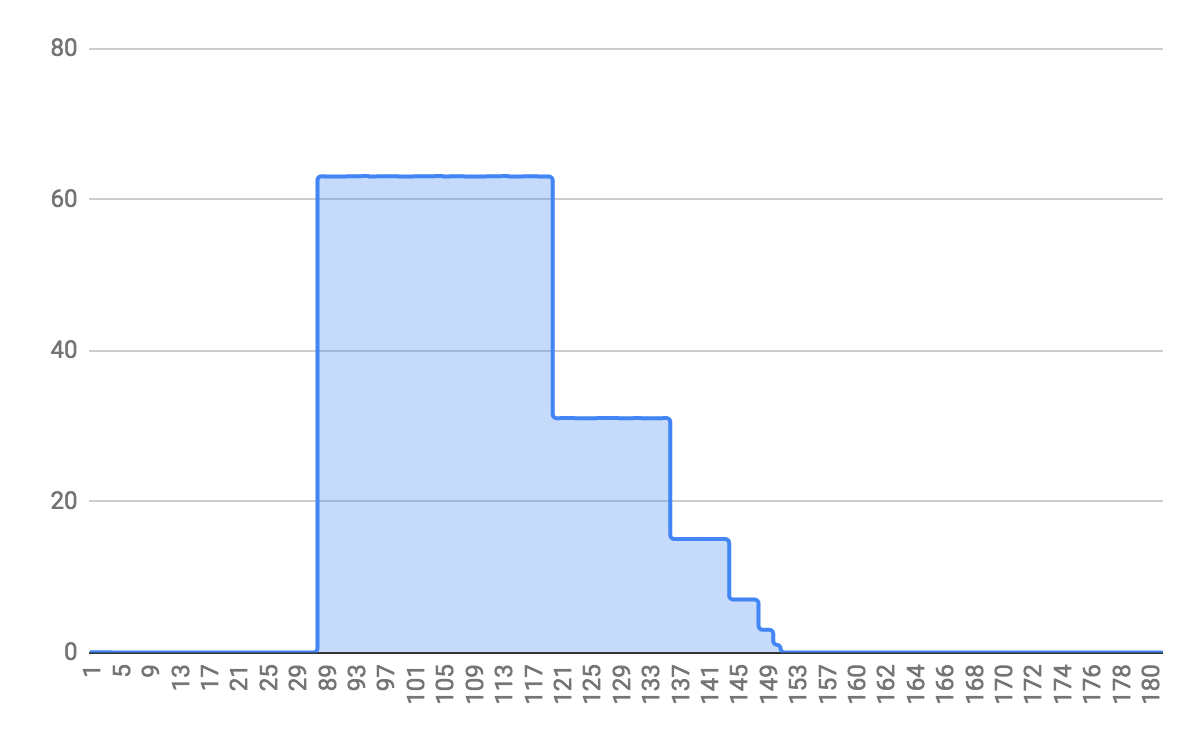
EXTRA CREDIT

1. Remove spoofing from the attack. Repeat the exercise without SYN cookies and observe and explain the effect. What happens? Can you explain why this happens? For hints run a tcpdump on the **server** node and look for traffic patterns. Can you modify the attack so that it is effective without spoofing and how would you do this?



Looking at the resulting duration graph, almost all the durations are below 0.01 seconds. This goes to show that the attack failed to deter the server from serving the requests to the client. Looking at a pcap taken on the server during the attack, it’s pretty easy to see why the attack fails. It’s entirely made up of SYN and RST packets where connections last about a thousandth of a second. Since there's only being one html request per second by the client, these extremely fast transactions are almost non existent at using the server's resources. The reason this is happening is because the attacker is responding to the SYN-ACKS that the server sends back which allows them to move the TCB out of SYN-RECEIVED. To make this attack work, you would have to either filter outgoing packets to the target or filter incoming packets so that any SYN-ACKS are discarded before reaching the local TCP processing code. Basically you want to make it so that the process is stalled after the initial SYN and avoid getting a RST message as an attacker.

1. Modify the NS file to introduce point-to-point routes, using the *Modify Experiment* option. Hint, you need to remove the server's route to lan1 and to add routes from the server to the attacker, and from the server to the client. Then click on *Submit*. It will take several minutes for the experiment to be restarted and you will receive an email notification once this is done. Now repeat the exercise without SYN cookies and observe and explain the effect. What happens? Can you explain why this happens? For hints run a tcpdump on the **server** node and look for traffic patterns.



In this experiment, there was a successful attack. It follows the first experiments path except for removing the lan1 from the topology. In the server tcpdump, it follows a standard attack expectation and when the attack starts is filled completely with just SYN messages from spoofed IPs. The only difference that I see is that during the attack, the client is absolutely helpless and none of its requests ever get to the server while in the original attack with the lan, some requests made it to the server, but it was still rare. I’m expecting this happens because the attacker has a direct link to the router rather than going through a lan. This means less opportunities for its attack to be filtered and dropped so more of its attack is succeeding in reaching the server and detering its services.

OTHER FILE SOURCES

Python file for duration calculation: <https://drive.google.com/open?id=1b8SZeg348eBNjZjIqEkHmM5therYxgkF>

Graphs with data: <https://drive.google.com/open?id=1wSap05CGrjMMTr6PL-FWIKaiuvQDYgHM3605f0OpkLs>

NS file for extra credit part 2: <https://drive.google.com/open?id=1wGD-h6ZNDwS6gGIQD8jzwdhVmk0oOVb9>