TCP Established Flooding

Christos Tziortzios - christos.tziortzios@os3.nl
Dimitar Pavlov - dimitar.pavlov@os3.nl
University of Amsterdam
System and Network Engineering (MSc)
(Dated: May 29, 2012)

Denial of Service attacks are a known threat to web servers. In this project we examine a variant of the TCP SYN flood attack – the Established flood attack. We test the efficiency of this attack by executing it against different Linux-based Operating Systems running a web server. We compare the Established flood to the SYN flood by considering the amount of resources exhausted – bandwidth, memory usage and CPU power – on the server side by each of the two attacks. Our results show that the Established flood attack is a more lightweight, precise and possibly more dangerous attack than the SYN flood, since it exhausts more resources, impacts the overall system performance, and, in certain cases, it is possible that a single attacking host render a web server unavailable.

I. INTRODUCTION

The vast majority of traffic on the Internet uses either the Transmission Control Protocol (TCP) or the User Datagram Protocol (UDP) as a transport protocol. TCP is a connection oriented protocol, which has limited security features. Because of that, there are several known attacks against it [1]. One of the more widely used TCP attacks is known as "SYN flooding" [2]. This is a Denial of Service (DoS) attack, in which the attacker sends an excessive number of connection-initializing `TCP SYN` packets to the victim. This causes the victim to try to continue the establishment of the requested connection by replying with `TCP SYN-ACK` packets, to which the attacker does not reply. This causes the victim to store data about a fake connection – at least for a small amount of time. However, due to the large number of TCP SYN packets sent by the attacker, the victim eventually becomes unavailable to new TCP connections and cannot serve its legitimate users.

In this project we examine a variant of the SYN flood attack, the TCP Established flood attack, in which the attacker performs a valid TCP connection establishment procedure (the TCP three-way handshake). In this case, the fake connection would transition from the `SYN-RCVD` to the `ESTABLISHED` state.

The TCP standard does not explicitly define a procedure for timing-out `ESTABLISHED` connections [3]. However, implementations typically include inactivity timers, which allow either party to attempt to close the connection after a period of inactivity. However, these inactivity timers typically allow connections in the `ESTABLISHED` state to be kept longer than connections in the `SYN-RCVD` state. Also, `ESTABLISHED` connections require more state data to be stored than `SYN-RCVD` connections. Finally, the SYN flood attack is well-known and there exist countermeasures, which may mitigate the attack. As the currently proposed attack resembles normal TCP activity, the same countermeasures are not directly applicable to this attack. These observations suggest that fully establishing a TCP connection during a flood attack will be more effective in overloading the server than just flooding with initializing `SYN` messages.

A. Research Question

Based on the above observations, we formulated the following research question:

> *Is the TCP Established Flood attack a feasible Denial of Service attack?*

Related to the formulated research question, we outlined the following subquestions:

- Given its increased complexity, how does this attack compare to the SYN flood attack with regards to target impact?
- What does the efficiency of the attack depend on?
- How can this attack be mitigated or prevented?

We believe that we can answer the research question by researching and answering each of the identified subquestions.

B. Related Research

TCP SYN flood attacks have been known since at least 1996 [4], and mitigation techniques have been developed [2]. With regards to TCP, other attacks have also been identified and investigated [1].

With regards to TCP Established flooding, only sporadic mentions can be found in the literature, with most of them referencing a specific piece of malware – the Trinity DDoS Linux-only malware kit [5]. Apart from those

mentions, there is no academic research on the topic of feasibility or efficiency of the Established flood attack. It appears that either the attack is not widely known, or not actively used on the Internet.

II. THEORY

In this section we provide the required theoretical background for understanding this report. We give theoretical definitions to the most significant terms and concepts used within the report.

A. TCP

TCP is a connection-oriented protocol, providing a reliable byte-stream service [6]; i.e., if two hosts need to send data to each other, they must first establish a connection, which will guarantee that data is transferred in a reliable and ordered way. The process of establishing a connection is called a "Three-way Handshake". Furthermore, TCP makes use of specially designed algorithms that take care of transmission speed adjustment and retransmission of lost packets. These algorithms rely on specific timers and timeouts to operate correctly.

B. TCP Three-way Handshake

Let A be a client and B a server. The Three-way Handshake is the exchange of the packets seen in Figure 1:

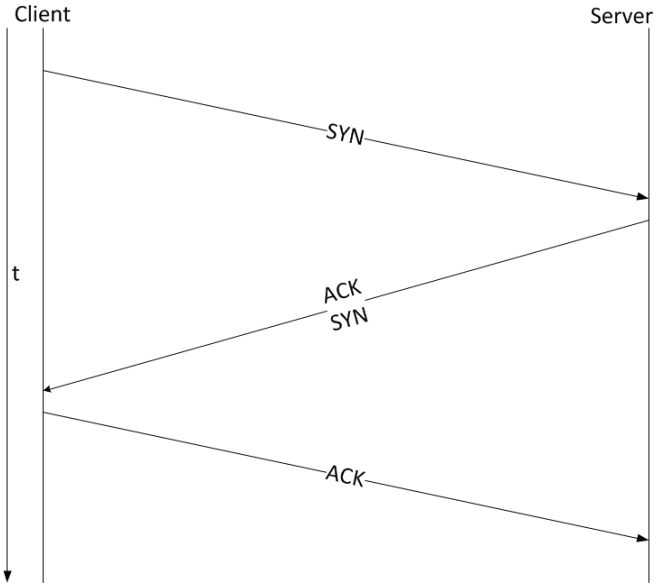

FIG. 1: Three-way Handshake

In the first packet the client requests that a connection is opened by sending a `SYN` packet, which includes an initial *sequence number*. This sequence number will always be incremented by one in subsequent packages, which allows the other party to identify missing packages. As a response, the server acknowledges the reception of the first packet and informs the client about *its* first sequence number (messages 2 and 3 are merged into one packet – the `SYN-ACK`). In the last step of the Three-way Handshake the initiating client acknowledges the reception of the `SYN-ACK`. At this point the two hosts may exchange data [7].

C. TCP SYN Flood Attack

From a security perspective, the Three-way Handshake has an intrinsic flaw. Whenever there is an incoming TCP packet with the SYN flag set, the burden is on the server to keep track of the partially open connection. Thus, the server opens a new connection in the `SYN-RCVD` state and stores information about it in-memory. This means that if the client does not manage to fully establish the connection for some reason, the server has wasted a part of its capacity by storing data about the partially open connection.

The TCP SYN Flood Attack is a *Denial of Service* (DoS) attack, which exploits this weakness. Since the resources of the server are limited, it is the attacker's goal to use up these resources, so that legitimate users can no longer connect to the server. The attacker transmits an excessive number of TCP SYN packets to the server, exhausting the server's resources.

This attack affects the following server resources:

- operational memory;
- number of open files;
- network bandwidth;
- CPU time.

Operational memory is exhausted by keeping track of all the half-opened connections until they time out. The *number of open files* limit can become a blocking factor on the server as well, since for each half-open connection the server needs to open a socket file. Also, since the attacker sends an enormous quantity of packets, the server's *network bandwidth* may be exhausted. Lastly, the server may exhaust all of its *CPU cycles*, processing the incoming requests.

Any of these restrictions can prevent the server from serving its legitimate clients.

Typically, the attacker does not use her own IP address – instead, she would use somebody else's IP address, which

makes her much harder to trace. Normally, using another's IP address would prevent one from establishing a connection, but for the attacker this is not a problem, as her goal is not to transfer data to or from the server, but to make it unavailable to legitimate clients.

D. TCP Established Flooding

The TCP Established Flooding attack is another DoS attack, which is similar to the SYN Flood attack. The main difference between the two attacks is that in the case of the TCP Established Flooding attack, instead of leaving the connection half-open, the attacker performs the complete three-way handshake. This attack exploits the fact that TCP `ESTABLISHED` connections require more resources than connections left in the `SYN-RCVD` state. As a result, the attacker can be more efficient – instead of sending a bulk of `SYN` packets, the attacker sends fewer packets, but exhausts the server resources more quickly.

From the attacker's point of view, a side effect of performing the whole TCP Three-way Handshake is that the source IP address cannot be masked easily. The attacking host needs to provide its real IP address in order to be able to receive the `SYN-ACK` reply packet from the server and establish the connection.

Another side effect is the fact that establishing a connection could also use up resources on the attacking host. Thus, the three-way handshake must be performed in such a way, as to make sure that the attacking host does not run out of resources during the attack.

E. TCP stack variables

The performance of the TCP stack can be altered by changing the values of certain variables. Some of the relevant variables include the following:

a. tcp_max_syn_backlog The `tcp_max_syn_backlog` variable indicates how many unprocessed SYN requests a machine is allowed to keep in memory – requests for which the third packet in a 3-way handshake has not been received yet.

b. tcp_synack_retries The `tcp_synack_retries` setting tells the kernel how many times to retransmit the SYN, ACK reply to an SYN request. In other words, this tells the system how many times to try to establish a passive TCP connection that was started by another host.

c. tcp_rmem This variable determines the amount of memory that each TCP socket can use to buffer incoming data before it is processed.

d. tcp_wmem This variable determines the amount of buffer space each socket has for transmission.

e. tcp_timestamps This variable determines whether to use TCP timestamps or not.

III. EXPERIMENTAL METHODS

In order to test the efficiency of the TCP Established flooding attack, we created a tool that enables and executes the attack (available at `https://github.com/zupper/tcpcflood`).

Apart from establishing connections, we needed to ensure that the attacking host would not exhaust its own resources. Normally, when a TCP connection is being established, the OS TCP stack in the kernel keeps track of the establishment process on both sides – both the client and the server keep state. This means that an attacker would be exhausting her own resources at the same pace as the target. However, the TCP `SYN-ACK` packet contains all the needed information for the attacker to send a valid final `ACK` packet. This means that by simply replying to received `SYN-ACK` packets with valid `ACK` packets, the attacker could statelessly make the target keep track of bogus `ESTABLISHED` connections.

We also needed to test the efficiency of TCP SYN flood attack in order to compare the two kinds of attacks. To do so, we used *hping3* [8], which is a popular tool for testing the security of networks and hosts.

A. Test Setup

The hardware resources available on the target machine were as follows:

- Intel(R) Pentium(R) D CPU 3.00GHz (2 cores)
- 2 GiB RAM

Each of the attacking hosts had the following hardware resources available:

- Intel(R) Xeon(R) CPU L3426 1.87GHz (4 cores)
- 8 GiB RAM

We tested the attacks against three different Linux Operating Systems (OS) with different variations of the same main Linux kernel version. We used the "stable" release versions of these distributions, as these versions are typically used with active servers. The Linux distributions we used are:

1. CentOS 2.6.32-220.el6.x86_64
2. Debian Squeeze 2.6.32-5-amd64

3. Ubuntu 2.6.32-38-server

DoS attacks are most commonly implemented against HTTP servers. Thus, we used the default version of the Apache web server from the software repositories of each system. We first tested the attacks against the default configuration settings of Apache. Afterwards, we performed optimization tuning of the Apache configuration and repeated the attacks.

The TCP variable values that were changed are the following:

Variable Name	Default Value	Set Value
tcp_max_syn_backlog	1024	2048
tcp_synack_retries	5	2

The Apache variable values that were changed are the following:

Variable Name	Default Value	Set Value
StartServers	8	16
MinSpareServers	5	10
MaxSpareServers	20	40
ServerLimit	256	768
MaxClients	256	768

Also, we needed to examine the apparent effect of attacking the web servers from the point of view of a regular user. To do that, we measured the response time of five subsequent requests to the web server using the script given in Appendix B. Judging from our own experience as web users, we decided to limit the waiting time for each request to 30 seconds, as most users would determine that the web page is not available at that point and either retry to access it, or give up altogether.

B. Restrictions & Assumptions

The link between the attacking hosts and the target had a bandwidth of 1Gbps. We assume that traffic that was generated by other means did not have a significant effect on our results.

No virtualization was used in our test setup, as that could potentially introduce influencing factors that the time frame of this project did not allow us to examine in detail.

C. Test Plan

The test plan was created after performing some preliminary tests with our Established flood tool and with hping3. The results of those tests were as follows:

- TCP Established: The attack could easily establish all available connections on port 80
- hping: One attacking host was not able to successfully perform a DoS attack, while two attacking hosts were sufficient

As a result of these observations, we used two different set-ups for our tests. For each OS tested, we would test against its default configuration but also against a tuned configuration (see above), which improved the performance of both the TCP stack and of the Apache web server, allowing for more connections to be established simultaneously. Increasing the maximum number of concurrent established connections too much lead to the server exhausting its own resources; this made the OS use the swap disk space which practically killed the server. Thus, we tuned Apache to accept a maximum number of 768 simultaneous connections for all the different set-ups. Since we found out that one attacking host could not effectively perform a DoS attack using hping3, we decided that we would test with two attacking hosts, which was sufficient.

IV. RESULTS

The results from the performed tests can be divided in two categories – SYN Flood attack measurements and Established Flood attack measurements. The attacks were performed with the target computer rebooted and under no particular load. For the SYN Flood, we needed to perform the attack from two attacking hosts to observe the target become unavailable. For the Established Flood, we found that a single attacking host made the target unavailable.

The results from the SYN Flood attack are shown in Tables I and III. For this attack we recorded the maximum bandwidth used in MiB/s. We observed large discrepancies in the resource utilization between the different Linux distributions that we used.

The results from the Established Flood attack are shown in Tables II and IV. For this attack we recorded the maximum bandwidth used in KiB/s. This attack did not show such large discrepancies in the resource utilization between the different Linux distributions that we used.

We also made measurements regarding the time needed for the index page of the web server to be returned. We measured the response time for such requests multiple times and then calculated the mean of the values we obtained and the the standard deviation for each result set. Unless the page was returned within 30 seconds, we considered that the server could not fulfil that request. As this meant that the attack was successful, we set the time field to "infinity" since most users would either refresh or close the web page in that time frame. However, in the calculations of the mean and the standard deviation, we treat these results as 30 seconds.

V. ANALYSIS

To analyse the results from testing the TCP Established flood, we compared it with a SYN Flood attack. We made use of several metrics, which allowed us to make a direct comparison between the two attacks on a metric-by-metric basis.

A. Metric-based comparison

By comparing SYN flooding with Established flooding on a metric-by-metric basis we estimated the impact both attacks have on the attacked server with regards to resource utilization – CPU cycles, RAM, network traffic. We used this estimation to draw conclusions about the severity of the Established flood attack from the point of the administrator of the web server.

All of the measurements for the metric-based comparison can be examined in Tables I, II.

1. Number of Attacking Hosts

In our test plan III C we had decided to perform both attacks from two separate attack hosts. In the case of SYN flooding, this was a requirement, as the attack did not produce any noticeable effect when performed from a single host. Also, we noticed a significant increase in the resource utilization on the *attacking* hosts while performing that attack, which was most likely caused by the generation of a large number of packets.

However, in the case of Established flooding, we determined that a single host was sufficient to produce a noticeable effect on the attacked server – increased response time, increased RAM and CPU load, overall decrease in responsiveness of the server. Also, we did not notice any increased resource utilization on the attacking host with this attack. This has implication regarding the amount of attacking nodes an attacker needs to render a target unavailable.

2. Peak Bandwidth

In the case of SYN flooding, the attack consumed as much bandwidth as was available at that moment. As can be seen from the test results, this meant that the peak bandwidth for the SYN flood attack was significantly higher than the peak bandwidth for the Established flood attack. In the case of SYN flooding, the amount of needed bandwidth is much higher than in the case of Established flooding. This has implications regarding the amount of bandwidth needed for an attacker to make a host unavailable.

3. Traffic/Packets In and Out

The SYN flood attack generates many times the traffic that is generated when performing Established flooding. This is due to the fact that SYN flooding has a more blunt, brute-force-like approach to making the target unavailable. When performing Established flooding, each `ESTABLISHED` connection requires just two packets sent from the attacking host.

With regards to traffic sent *to* the target, when performing SYN flooding the amount of traffic is noticeably more than the amount of traffic needed for an Established flood attack. This means that for a SYN flood to be effective, significantly more traffic needs to reach the target.

With regards to traffic sent by the target, the situation is more or less the same – the target needs to transmit more traffic in the case of a SYN flood attack, as there are many more SYN packets transmitted to which the target replies.

4. CPU Load Increase

We did not not any significant increase in CPU load when performing a SYN flood – for all tested Linux distributions, the increase in load was minimal.

However, when performing the Established flood attack, in almost all cases the increase of the system load was noticeable. We also noticed that with time, the load, along with the RAM usage increased gradually. We believe this is related to the number of open sockets (see below).

5. RAM Usage Increase

Similarly to the CPU load, the SYN flood did not produce a noticeable increase in RAM usage. In one case there was an exception, but we believe that the overall memory requirements for handling a SYN flood attack are not that high in this test setup.

Again, similarly to the CPU load, the Established flood attack *did* require more RAM than the SYN flood. We also noticed that the amount of used memory was growing with time (along with the system load). We believe this is related to the number of open sockets (see below).

6. Open Sockets

In the case of SYN-flooding, the open TCP sockets were all in the `SYN-RCVD` state, which is normal for this type of attack. However, with this attack the sockets get discarded of after the `SYN-ACK` retransmits take place and they do not pass through other states (`FIN-WAIT1`, `FIN-WAIT2`, etc).

We noted that the SYN flood attack was being restricted in the amount of open `SYN-RCVD` sockets on the server by a hard limit of 256 on all tested systems. We believe this is related to the popularity of the SYN flood attack and to the built-in protection mechanisms on a system level.

When performing Established flooding, however, the attack results in open TCP sockets in different states – `ESTABLISHED`, `SYN-RCVD` and `FIN-WAIT1`. The amount of sockets in the `ESTABLISHED` state was limited by the web server settings and the number of clients/requests it can handle (according to its configuration). We can see that by tuning the attacked systems we get a higher number of `ESTABLISHED` connections, which, in a normal usage situation, would result in more requests being handled simultaneously.

We also observed that since the Established flood attack produced sockets in the `ESTABLISHED` state, those were transitioning to the `FIN-WAIT1` state after a certain amount of time. Since the `FIN-WAIT1` sockets have a much higher timeout than `SYN-RCVD` sockets [7], the number of `FIN-WAIT1` sockets was growing in a stable manner. The growth was also related to a stable increase in the system load and in the amount of used RAM. However, the time frame for this project did not allow us to further investigate this trend.

B. Response Time

The results of the response time tests show that the SYN Flood attack was more effective – the mean response time is higher in most cases. The effects of both attacks seem to be smaller when the web server runs on Ubuntu; however the sample size is not big enough to lead to strong evidence. Furthermore, the Established flood attack is more effective against the default configuration; tuning the configuration mitigated the effects of the attack to a certain extent, but we could easily intensify the attack by increasing the desired number of established connections per second.

Another remark is related to standard deviation. We noticed that there are cases, in which we got a response in less than 10 seconds and then could not get a response at all while performing the same attack. In the case of the Established flood attack, we were noticed that the amount of connections in the `ESTABLISHED` state would fluctuate while performing the attack, although the amount of packets sent per second was constant. There were certain moments, in which the attacking host was able to establish as many connections as available, but the number of connections would drop significantly afterwards, only to grow larger again after a short period of time. We believe that this instability is caused by the fact that our attack tool did not synchronize with the target machine's connection expiration timers. Therefore, it was possible for the target machine to discard several `ESTABLISHED` connections before our attack tool would get a chance at re-establishing them.

C. Recovery From Attack

Recovery from the SYN flood attack happened almost instantly – both in the default and in the optimized configurations on all variants of the OS. The web page was available as soon as the attack was stopped. Moreover, the resource usage was back to normal in a few seconds. On the other hand, recovery from the Established flood attack took approximately 10 seconds, but the resource usage, especially memory and system load, took minutes before the system could get back to a "normal" non-loaded state.

VI. CONCLUSION

We proved that TCP Established Flood attack is a feasible Denial of Service attack. Moreover, since we imitate legitimate user requests to establish TCP connections, it is hard for the attack to be mitigated with the methods used to mitigate the SYN Flood attack. In the analysis of our results we see that the attack has a significant effect on the victim's memory usage in certain cases. The attack not only makes the the web server unavailable to its legitimate users but it also makes the whole system unresponsive overall.

We also showed that the attack has different effects on different OS with different variants of the Linux kernel and variations in the TCP stack implementations. Moreover, tuning certain TCP and Apache variables made the Established flood attack less effective. Although there was an increased delay, a legitimate user could still be able to get the web page after waiting for a few seconds.

We also see that the attack is lightweight, expecially when comparing it to a SYN flood – a minimal amount of network bandwidth, CPU cycles and memory is needed to execute the attack successfully. This make it possible for the attack to be performed even with a smartphone.

Another important finding is that a single host is capable of making a web server unavailable. This can compensate for the fact that the attacker cannot use a fake IP address;

a single compromised host can DoS a web server running with a default configuration.

With regards to mitigating the attack, we could not come up with a full-proof method for mitigation. While it is possible to simply block traffic originating from one or more attacking hosts, this protection mechanism becomes ineffective in the case of a distributed attack against a single target.

VII. SUGGESTIONS FOR FURTHER RESEARCH

There is a lack of previous academic research on this type of DoS attack. Although we have shown it to be a feasible DoS attack, there is still need for further research.

Firstly, the created attack tool can be greatly improved. More functionality can be added and a better throtling mechanism can be implemented that could make the attack even more severe. Also, we believe that the severity of the attack can be furher increased by sending an HTTP-GET request to the web server immediately, after establishing the TCP connection. This would utilize even more resources on the server side, such as bandwidth and CPU cycles. Also, it would imitate legitimate traffic even better.

Due to the small time frame of this project, we did not investigate the efficiency of the attack against Windows-based systems.

Furthermore, more and more web servers are running on virtualized hosts. The different virtualization platforms could perform in different ways when under such an attack. Moreover, although Virtual Machines (VM) running on the same physical hosts are thought to be independent of one another, it is possible that the attack influences the physical machine and, through this – all of the VMs running on it. For instance, improper Apache configuration can lead to using swap disk space, which in turn may influence the physical machine's performance.

Lastly, the feasibility of the attack should also be tested against web servers running on in commercial cloud environments or behind load balancers. Typically, customers of these services have to pay more when more resources, such as CPU cycles or bandwidth, are being consumed. This means that apart from the damage done because of making their services unavailable to legitimate users, the attack could lead to an increased spending for hosting services.

VIII. ACKNOWLEDGEMENTS

We would like to thank Boyan Krosnov for coming up with this attack and for giving us many insightful ideas about its implementation. We would also like to thank Vasil Kolev for providing us with with his feedback about the attack tool. Their help was invaluable.

Finally, we would like to than Jeroen van Beek for supervising us during this project and for his support and ideas.

[1] Guang Yang. Introduction to TCP/IP Network Attacks. 2002.

[2] W. Eddy. RFC4987: TCP SYN Flooding Attacks and Common Mitigations. `http://www.ietf.org/rfc/rfc4987.txt`.

[3] RFC1122: Requirements for Internet Hosts – Communication Layers, Section 4.2.3.6. `http://tools.ietf.org/html/rfc1122#page-101`.

[4] Fred Cohen. Internet holes Part 13: The SYN flood. *Network Security*, 1996(10):7 – 9, 1996.

[5] David Sheridan. Trinity v3 DDOS: Tomorrows headline? 2000. `https://en.wikipedia.org/wiki/Transmission_Control_Protocol#Connection_termination`.

[6] W.R. Stevens and G.R. Wright. *TCP/IP Illustrated: The Protocols*. Number v. 1 in Addison-Wesley Professional Computing Series. Addison-Wesley Publishing Company, 1994.

[7] RFC793: Transmission Control Protocol. `http://www.ietf.org/rfc/rfc793.txt`.

[8] hping website. `http://www.hping.org/`.

Appendix A: Results & Analysis

Distribution	**CentOS**		**Debian**		**Ubuntu**	
Settings	**Default**	**Optimized**	**Default**	**Optimized**	**Default**	**Optimized**
Traffic In	31.4 MiB	13.4 MiB	2400 MiB	2000 MiB	10.3 MiB	19.4 MiB
Traffic Out	26.9 MiB	8.7 MiB	11.7 MiB	10.5 MiB	7.4 MiB	14.6 MiB
Packets In	488 K	202 K	40000 K	34000 K	160 K	310.8 K
Packets Out	366.3 K	78 K	190.3 K	169.7 K	89 K	209.5 K
Peak B/W	4 MiB/s	0.3 MiB/s	12.2 MiB/s	11.8 MiB/s	2.5 MiB/s	7.37 MiB/s
RAM Increase	14 MB	84 MB	1400 MB	50 MB	1 MB	1 MB
Load Increase	0	0.03	0.05	0	0.15	0.06
SYN-RCVD	0	245	256	256	256	256

TABLE I: Test results from SYN Flood attack (bandwith is in MiB/s)

Distribution	**CentOS**		**Debian**		**Ubuntu**	
Settings	**Default**	**Optimized**	**Default**	**Optimized**	**Default**	**Optimized**
Traffic In	1.5 MiB	1.6 MiB	1.7 MiB	2.2 MiB	1.9 MiB	2.3 MiB
Traffic Out	0.75 MiB	1.4 MiB	1.3 MiB	3 MiB	1.8 MiB	4.9 MiB
Packets In	23.6 K	25.4 K	26.7 K	35 K	30.5 K	37 K
Packets Out	9.9 K	18.6 K	15.2 K	48.4 K	26 K	76.6 K
Peak B/W	15 KiB/s	23.1 KiB/s	18.3 KiB/s	18.6 KiB/s	15.6 KiB/s	17.2 KiB/s
RAM Increase	1200 MB	1800 MB	98 MB	491 MB	70 MB	394 MB
Load Increase	0.5	28.93	0	0.5	0.36	0.6
ESTABLISHED	386	898	280	833	277	773
SYN-RCVD	240	800	256	256	256	256
FIN-WAIT	512	254	1206	1900	900	1152

TABLE II: Test results from Established Flood attack (bandwith is in KiB/s)

Distribution	**CentOS**		**Debian**		**Ubuntu**	
Settings	**Default**	**Optimized**	**Default**	**Optimized**	**Default**	**Optimized**
1st request	4.54	8.76	inf	inf	6.18	7.38
2nd request	inf	inf	inf	inf	5.57	13.55
3rd request	11.96	inf	inf	inf	4.74	5.01
4th request	2.22	inf	inf	inf	14.86	6.14
5th request	2.48	inf	inf	inf	inf	inf
Mean	10.25	25.77	30	30	12.27	12.42
Std. Dev	11.75	9.51	0	0	10.73	10.39

TABLE III: Test results from SYN Flood attack (Response time in seconds)

Distribution	**CentOS**		**Debian**		**Ubuntu**	
Settings	**Default**	**Optimized**	**Default**	**Optimized**	**Default**	**Optimized**
1st request	27.32	13.09	inf	4.59	10.71	4.93
2nd request	inf	1.88	7.51	1.59	11	2.03
3rd request	inf	0.95	19.97	1.79	inf	2.02
4th request	4.61	0.33	17.97	5.24	8.02	2.02
5th request	inf	0.29	6.08	1.19	10.98	1.3
Mean	24.4	3.31	16.31	2.88	14.15	2.46
Std. Dev	11.12	5.5	9.83	1.88	8.97	1.42

TABLE IV: Test results from Established Flood attack (Response time in seconds)

Appendix B: Testing Response Time

```
#!/bin/bash

COUNTER=0
LIMIT=5

while [ $COUNTER -lt $LIMIT ]
do
        COUNTER=$((COUNTER + 1))
        echo Try $COUNTER
        /usr/bin/time -p -a -o $1 curl --silent --max-time 30 \
                http://1.1.1.1/ >/dev/null 2>&1
done
```