

Group project report

2021 CICT high-quality class
Group Project Report
Cyber security

Project Title	Attacking & Defending DoS Slowloris in Python	
Project Area	DoS attack & defense	
Students	ID	Name
	B1809707	Nguyen Chi Hoang Minh
	B1809720	Nguyen Nhi Thai
	B1809703	Vo Thanh Long
Reporting Date	28/11/2021	

I. Project Outline

1. Title

Attacking & defending DoS Slowloris in Python

2. Group Information

Team Name		TML			
Team Composition		Name	Belong	Department	Position/year
Instructor		Prof. Noh	CIC T	IT Department	
Student	Team Leader	Nguyen Chi Hoang Minh	CIC T	IT Department	4
	Team member 2	Nguyen Nhi Thai	CIC T	IT Department	4
	Team member 3	Vo Thanh Long	CIC T	IT Department	4

Team Photos

II. Project Information

1. Purpose of Project

The purpose of this project is to explore and highlight basic ways to defend DoS Slowloris attacks that should be implemented to strengthen the security status of Web Server. This document is by no means a complete security guide for the Web Server; however, it outlines the basic security hardening of the Web Server, so it may not be a vulnerable target. Many system administrators do not realize that the default Apache Web Server installation is vulnerable to the DoS Slowloris attack. Therefore, this document describes the basics used

to prevent DoS Slowloris attacks in Apache Web Server.

This research project explores and proposes general hardening best practices for common Apache Web Server such as integrating module security `mod_qos`, deploying load balancing model, and using the host-based firewall to increase tolerance and block connect to bad traffic.

The proposed outcome of the project is to identify DoS Slowloris attacks in a production Apache Web Server and the result of such attacks. Many businesses are compromised as a result of such DoS attacks, and this project is expected to explore and suggest best practices to enhance the security posture of Apache Web Servers.

2. Contents and scope

2.1 Contents

Apache Web Server is widely used as a web server around the world. Like other web servers, it has its pros and cons. Security is one of the aspects that are overlooked. Many system administrators assume that Web server itself is secure and they leave many services at the default configurations leaving the server vulnerable and making an easy target for the hackers. Therefore, this project is intended to outline common security solutions like integrating module security, deploying a load balancing model, and using a host-based firewall (IPTABLES) to increase tolerance and block connections to bad traffic. That should be changed and configured properly to harden the security of the server. The project will additionally, outline best practices for preventing the Apache Web Server from DoS Slowloris attacks.

I will follow the KISS principle which says “Keep It as Simple as Possible”. It is my belief that by implementing this type of monitoring with common best practices and changing default configurations to a more secure counterpart can highly reduce the risk of easily being compromised. This project will not address the security and configurations of all the tools and services available for Apache Web Server or network security. This project will only cover Apache Server & attacking DoS Slowloris and the most common defending solutions. This study alone will not make a Linux server completely secure from attacks or vulnerabilities; however it will try to point out common settings and configurations that

will harden the Web server security from DoS Slowloris attacks.

2.2 Scope

- There are various operating systems that may be used in server systems; However, this project will focus on Linux Operating System, Ubuntu 20.04 is selected as the operating system for this project.
- In the scope of this project common services, tools include mod_qos or Module qos which is used to defend an attack according to bandwidth limits., HAProxy for load balancing services, IPTABLES which is used as a host-based firewall.

III. Action Plan

1. Environments & resource

		Details
S/W	OS	Ubuntu 20.04
	IDE	Debian Linux
	Language	Python
	Tool	Slowloris
H/W	Device	Personal PC
	Sensor	None
	Communication	None

2. Role arrangements

Student	Division	Role
1	Plan & design	Using the DoS Slowloris attack tool in Python Choosing environments, parameters, solutions Arranging the work and tracking progress
2	Analysis	Detecting the Slowloris DoS attack Mitigating Slowloris vulnerabilities in the Apache web

		server
3	Implement & test	<p>Installing Apache Web Server on Ubuntu operating system</p> <p>Installing and Configuring Apache module mod_qos, load balancers HAProxy, firewall IPTABLES</p>

3. Project Schedule

Division	Promotion contents	Schedule						
		Mon	Tue	Wed	Thu	Fri	Sat	Sun
Plan	Role sharing and analysis software installation		X		X		X	
Analysiss	Software option analysis		X		X		X	
Test	Analysis using Software function		X		X		X	
Finish	Create result document through analysis		X		X		X	
Online meeting Plan	Information sharing and progress confirmation of each other		X		X		X	

IV. Result of study

1. Meaning of Firewall

A Linux firewall [1] is a device that inspects Network traffic (Inbound /Outbound

connections) and makes a decision to pass or filter out the traffic. Iptables is a CLI tool for managing firewall rules on a Linux machine.

Network Security evolved with different types of Linux firewall. Traditional packet-filtering firewalls deal with Routing and filtering packets (OSI Layers 3 and 4), Where else NGFWs will work with additional functions as with OSI layers (L4-L7 of OSI model).

2. Meaning of Web Server

A web server [2] is software and hardware that uses HTTP (Hypertext Transfer Protocol) and other protocols to respond to client requests made over the World Wide Web. The main job of a web server is to display website content through storing, processing, and delivering webpages to users. Besides HTTP, web servers also support SMTP (Simple Mail Transfer Protocol) and FTP (File Transfer Protocol), used for email, file transfer, and storage.

Web server hardware is connected to the internet and allows data to be exchanged with other connected devices, while web server software controls how a user accesses hosted files. The web server process is an example of the client/server model. All computers that host websites must have web server software.

Web servers are used in web hosting, or the hosting of data for websites and web-based applications -- or web applications.

3. Meaning of DoS attacks

Network Attacks are often referred to as Denial of Service (DoS) [3] attacks. This type of attack takes advantage of the specific capacity limits that apply to any network resources – such as the infrastructure that enables a company’s website. The DoS attack will send multiple requests to the attacked web resource – with the aim of exceeding the website’s capacity to handle multiple requests... and prevent the website from functioning correctly.

4. Meaning of Load Balancing

Load balancers [4] are ideally suited for inclusion within a layered security model. The primary function of a load balancer is to spread workloads across multiple servers to prevent overloading servers, optimize productivity, and maximize uptime. Load balancers also add resiliency by rerouting live traffic from one server to another if a server falls prey to DoS attacks or otherwise becomes unavailable. In this way, load balancers help to eliminate single points of failure, reduce the attack surface, and make it harder to exhaust resources

and saturate links.

5. Meaning of Module security

Host-based firewalls like iptables, UFW, and FirewallD, etc. They work on layer 3 and 4 of the OSI model and take actions based on IP address and port number. Module Security in general, is specialized to focus on HTTP traffic (layer 7 of the OSI model) and takes action based on the content of HTTP request and response

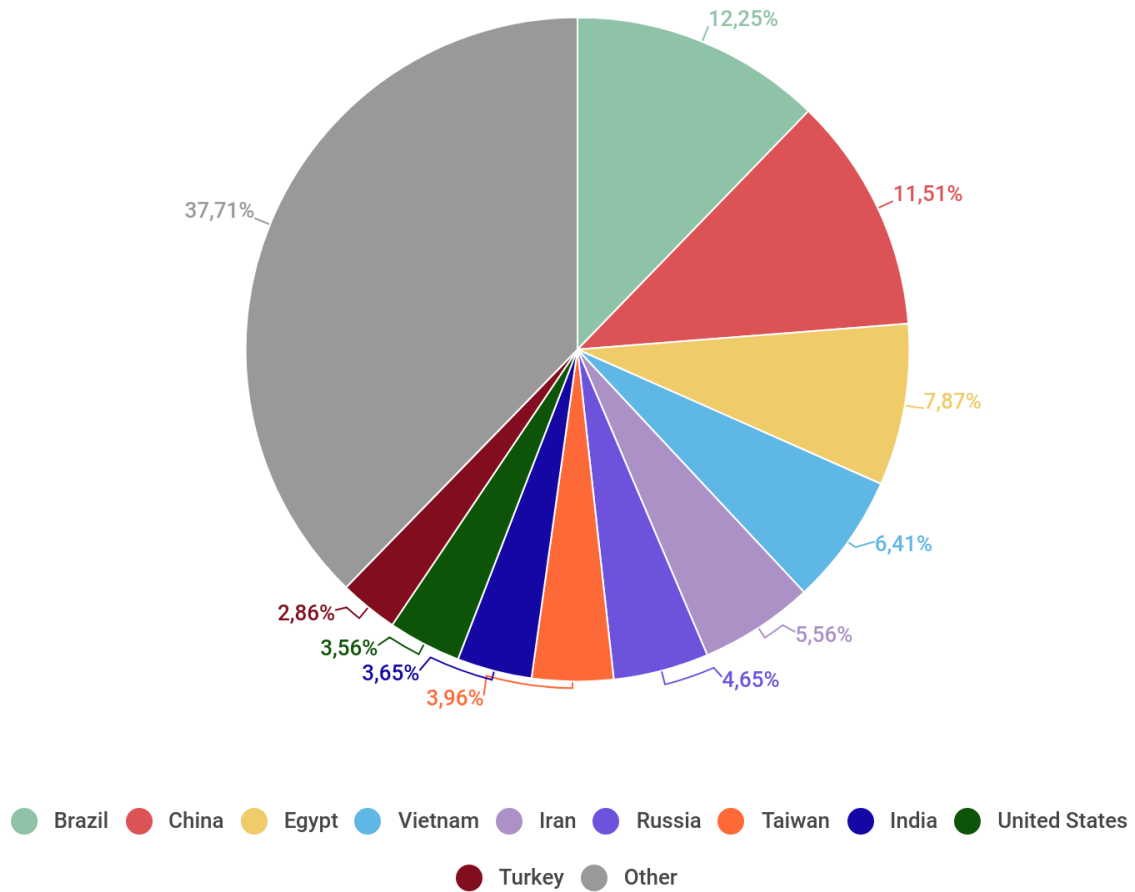
6. Real situation of DoS attack in the World

In addition to 2020 being an all-time high for the number of DoS attacks, a few more records were set as well. The most DDoS attacks recorded in a single month hit a new high at 929,000, while average DoS attacks per month topped 2019 averages by between 100,000 and 150,000.

In all, there was a 20% increase in the number of DoS attacks from 2019 to 2020 [5], and the second half of 2020 was where most were concentrated, with a 22% spike in the last six months of the year.

Much of the rise in DDoS frequency can be attributed to the COVID-19 pandemic. Life has shifted almost entirely to the Web -- people worldwide are now working, studying, shopping, and having fun online like never before.

Here is a breakdown of dos attack by country:



kaspersky

V. Result of practice

[Step 1] Starting the Apache Server

systemctl restart apache2

```
b1809707@b1809707-VirtualBox:~$ sudo systemctl restart apache2
b1809707@b1809707-VirtualBox:~$ sudo systemctl status apache2
● apache2.service - The Apache HTTP Server
   Loaded: loaded (/lib/systemd/system/apache2.service; disabled; vendor preset: enabled)
   Active: active (running) since Thu 2021-11-25 22:09:37 WIB; 1s ago
     Docs: https://httpd.apache.org/docs/2.4/
   Process: 2523 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
  Main PID: 2534 (apache2)
    Tasks: 55 (limit: 2299)
   Memory: 7.5M
    CGroup: /system.slice/apache2.service
            └─2534 /usr/sbin/apache2 -k start
               └─2535 /usr/sbin/apache2 -k start
                  └─2536 /usr/sbin/apache2 -k start
```

[Step 2] Running the DoS attack tool

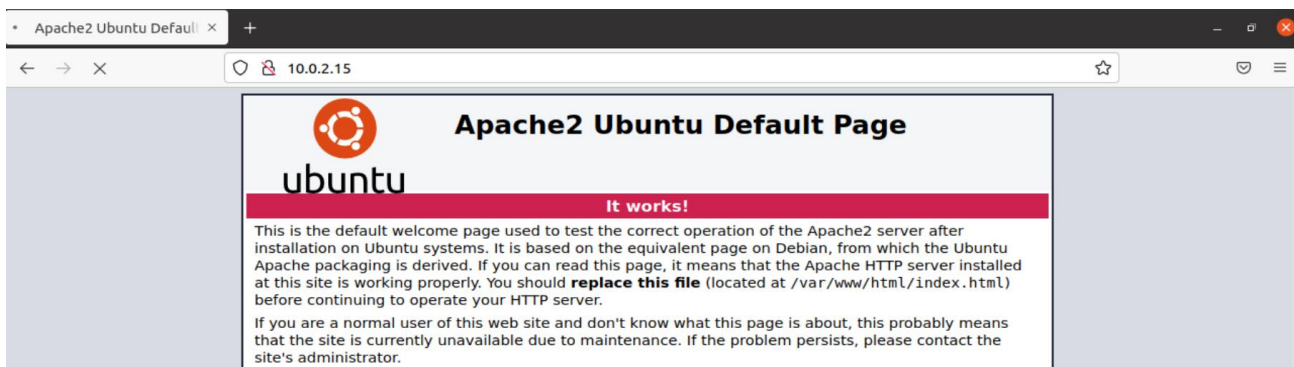
python3 slowloris.py (IP address) -s (Number of sockets)

The result will connect to the target on port 80 and attempt to make 500 connections to Apache and keep them open.

```
b1809707@b1809707-VirtualBox:~/slowloris/slowloris$ python3 slowloris.py 10.0.2.15
-s 500
[25-11-2021 22:13:15] Attacking 10.0.2.15 with 500 sockets.
[25-11-2021 22:13:15] Creating sockets...
[25-11-2021 22:13:15] Sending keep-alive headers... Socket count: 500
[25-11-2021 22:13:30] Sending keep-alive headers... Socket count: 500
[25-11-2021 22:13:45] Sending keep-alive headers... Socket count: 500
[25-11-2021 22:14:00] Sending keep-alive headers... Socket count: 500
[25-11-2021 22:14:15] Sending keep-alive headers... Socket count: 500
[25-11-2021 22:14:31] Sending keep-alive headers... Socket count: 500
[25-11-2021 22:14:46] Sending keep-alive headers... Socket count: 500
[25-11-2021 22:15:01] Sending keep-alive headers... Socket count: 500
[25-11-2021 22:15:16] Sending keep-alive headers... Socket count: 500
[25-11-2021 22:15:31] Sending keep-alive headers... Socket count: 500
[25-11-2021 22:15:46] Sending keep-alive headers... Socket count: 500
[25-11-2021 22:16:01] Sending keep-alive headers... Socket count: 500
[25-11-2021 22:16:16] Sending keep-alive headers... Socket count: 500
[25-11-2021 22:16:32] Sending keep-alive headers... Socket count: 500
```

[Step 3] Checking the website

The site is only loading and loading.



[Step 4]: Detecting Slow HTTP DoS Attack

```
netstat -ntu -4 -6 | awk '/^tcp/{ print $5 }' | sed -r 's/:[0-9]+$//' | sort | uniq -c | sort -n
```

The result will give a number of active connections for each connected IP.

If web servers are under a DoS attack. For each IP, the one with 50-100 connections (or more) is most probably a slowloris attacker.

```
b1809707@b1809707-VirtualBox:~$ netstat -ntu -4 -6 | awk '/^tcp/{ print $5 }' |
sed -r 's/:[0-9]+$//' | sort | uniq -c | sort -n
  1 142.250.199.66
  1 142.250.204.66
  1 34.117.237.239
  1 34.120.208.123
  1 35.224.170.84
  1 35.244.181.201
  1 52.39.192.7
  2 172.217.25.4
  3 216.58.200.67
1000 10.0.2.15
```

[Step 5]: Viewing Apache error log file

```
cat /var/log/apache2/error.log
```

The result will give some symptoms.

The symptoms are always the same: “Server reached MaxRequestWorkers setting ...”. It’s how Slowloris prevents new connections from coming through.

```
b1809707@b1809707-VirtualBox:~$ cat /var/log/apache2/error.log
[Thu Nov 25 22:43:20.396642 2021] [mpm_event:notice] [pid 3501:tid 140687005740096] AH0048
9: Apache/2.4.41 (Ubuntu) configured -- resuming normal operations
[Thu Nov 25 22:43:20.396809 2021] [core:notice] [pid 3501:tid 140687005740096] AH00094: Co
mmand line: '/usr/sbin/apache2'
[Thu Nov 25 22:43:32.416181 2021] [mpm_event:error] [pid 3501:tid 140687005740096] AH00484
: server reached MaxRequestWorkers setting, consider raising the MaxRequestWorkers setting
```

VI. Problems and Solutions

1. Problems:

Slowloris is an application layer attack that operates by utilizing partial HTTP requests. The attack functions by opening connections to a targeted Web server and then keeping those connections open as long as it can.

Slowloris is not a category of attack but is instead a specific attack tool designed to allow a single machine to take down a server without using a lot of bandwidth. Unlike bandwidth-consuming reflection-based DDoS attacks such as NTP amplification, this type of attack uses a low amount of bandwidth, and instead aims to use up server resources with requests that seem slower than normal but otherwise mimic regular traffic. It falls in the category of attacks known as “low and slow” attacks. The targeted server will only have so many threads available to handle concurrent connections. Each server thread will attempt to stay alive while waiting for the slow request to complete, which never occurs. When the server’s maximum possible connections have been exceeded, each additional connection will not be answered and denial-of-service will occur.

A Slowloris attack occurs in these following steps:

- The attacker first opens multiple connections to the targeted server by sending multiple partial HTTP request headers.
- The target opens a thread for each incoming request, with the intent of closing the thread once the connection is completed. In order to be efficient, if a connection takes too long, the server will timeout the exceedingly long connection, freeing the thread up for the next request.
- To prevent the target from timing out the connections, the attacker periodically sends partial request headers to the target to keep the request alive. In essence saying, “I’m still here! I’m just slow, please wait for me.”
- The targeted server is never able to release any of the open partial connections while waiting for the termination of the request. Once all available threads are in use, the server will be unable to respond to additional requests made from regular traffic, resulting in denial-of-service.

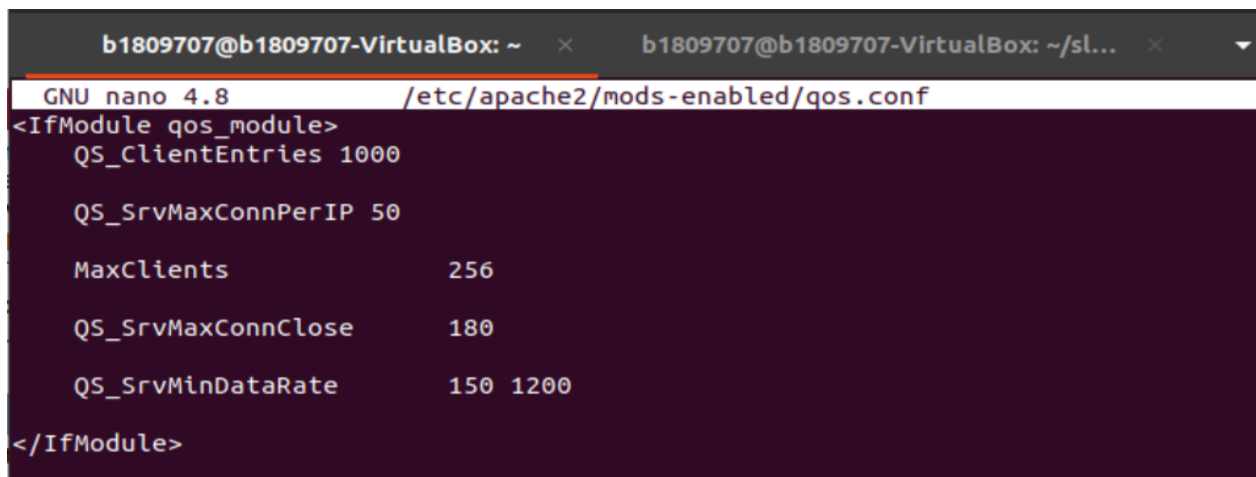
The key behind a Slowloris is its ability to cause a lot of trouble with very little bandwidth consumption.

2. Solutions:

2.1 Defending Slowloris DoS with mod_qos

Module qos gives some fine-grained opportunities to scale the number of used connections and to defend an attack according to bandwidth limits.

To describe shortly what we are doing is limiting the number of simultaneous inbound connections from a single IP. This will prevent automatically a user from creating more than multiple requests (specified at QS_SrvMaxConnPerIP) at the same time from the same device/network. However, the application has access from more users than the specified at this property, they will be able to access it as long as it's not a Slow HTTP request.

A screenshot of a terminal window with two tabs. The active tab is titled 'b1809707@b1809707-VirtualBox: ~' and shows the nano text editor editing the file '/etc/apache2/mods-enabled/qos.conf'. The editor's title bar says 'GNU nano 4.8'. The configuration content is as follows:

```
<IfModule qos_module>
  QS_ClientEntries 1000

  QS_SrvMaxConnPerIP 50

  MaxClients          256

  QS_SrvMaxConnClose   180

  QS_SrvMinDataRate    150 1200
</IfModule>
```

handles connections from up to 1000 different IPs
QS_ClientEntries 1000
will allow only 50 connections per IP
QS_SrvMaxConnPerIP 50
maximum number of active TCP connections is limited to 256
MaxClients 256
disables keep-alive when 70% of the TCP connections are occupied:
QS_SrvMaxConnClose 180
minimum request/response speed (deny slow clients blocking the server, ie. slowloris
keeping connections open without requesting anything):
QS_SrvMinDataRate 150 1200

When we running Slowloris tool, the result gives the Socket counting number drops from 500 to 100.

```
b1809707@b1809707-VirtualBox:~/slowloris/slowloris$ python3 slowloris.py 10.0.2.15 -s 500
[30-11-2021 18:41:35] Attacking 10.0.2.15 with 500 sockets.
[30-11-2021 18:41:35] Creating sockets...
[30-11-2021 18:41:35] Sending keep-alive headers... Socket count: 500
[30-11-2021 18:41:50] Sending keep-alive headers... Socket count: 500
[30-11-2021 18:42:05] Sending keep-alive headers... Socket count: 100
[30-11-2021 18:42:20] Sending keep-alive headers... Socket count: 100
```

2.2 Defending Slowloris DoS with load balancing

Slowloris involves an attacker making requests very slowly to tie up the connection slots. Contrary to other types of DoS, the volume of requests needed to make this attack successful is fairly low. However, as each request only sends one byte every few seconds, they can tie up many request slots for several minutes.

An HAProxy load balancer can hold a greater number of connections open without slowing down than most web servers.

We use Docker [6] to create a single HAProxy and two Apache containers

```
b1809707@b1809707-VirtualBox:~/lb/haproxy$ docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS           NAMES
4fbfaba3f39f   helloworld_apache_img_2            "httpd-foreground"      6 days ago    Up 9 minutes  80/tcp          helloworld_apache_con_2
961c83c52acd   helloworld_apache_img_1            "httpd-foreground"      6 days ago    Up 9 minutes  80/tcp          helloworld_apache_con_1
886576041baf   helloworld_haproxy_img             "docker-entrypoint.s..." 6 days ago    Up 9 minutes  0.0.0.0:80->80/tcp, :::80->80/tcp helloworld_haproxy_img_1
```

HAProxy Configuration

global

```
log /dev/log local0
log localhost local1 notice
maxconn 2000
daemon
```

defaults

```
log global
mode http
option httplog
option dontlognull
```

```
option http-buffer-request
maxconn 1950
retries 3
timeout connect 4s
timeout client 60s
# Don't queue requests too long if saturated.
timeout queue 60s
# Time we may wait for a response from the server.
timeout server 60s
timeout http-request 5s
```

frontend http-in

```
bind *:80
default_backend webservers
```

backend webservers

```
stats enable
stats auth admin:admin
stats uri /haproxy?stats
balance leastconn
option httpchk
option forwardfor
option http-server-close
server apache1 ${APACHE_1_IP}:${APACHE_EXPOSED_PORT} check
server apache2 ${APACHE_2_IP}:${APACHE_EXPOSED_PORT} check
```

In the global section, maxconn leaves enough headroom so that the server won't run out of memory even if all the connections are filled, per the sizing guide.

Inside the defaults section, maxconn value slightly under global that so that if an attack saturates one frontend, the others can still operate.

Inside the defaults section, the option timeout http-request cause HAProxy to respond to any clients that spend more than five seconds from the first byte of the request to the last with an HTTP 408 Request Timeout error. Normally, this only applies to the HTTP request and its headers and doesn't include the body of the request. However, with option http-buffer-request, HAProxy will store the request body in a buffer and apply the http-request timeout to it.

Inside the backend section, the balance leastconn mean the algorithm we use for balancing is least connection. Requests will be routed to the server that has the fewest connections to it.

2.3 Defending Slowloris DoS with Firewall

Linux operating systems come equipped with a very powerful, stateful packet filtering application or a firewall known as IPTABLES.

When a packet matches a rule, it is given a target, which can be another chain or one of these special values:

ACCEPT – will allow the packet to pass through.

DROP – will not let the packet pass through.

RETURN – stops the packet from traversing through a chain and tell it to go back to the previous chain.

Meaning of three chains:

INPUT – controls incoming packets to the server.

FORWARD – filters incoming packets that will be forwarded somewhere else.

OUTPUT – filter packets that are going out from the server.

To limit connections to port :80 from a single IP, we use the following iptables rule:
`iptables -I INPUT -p tcp --dport 80 -m connlimit --connlimit-above 50 --connlimit-mask 20 -j DROP`

```
b1809707@b1809707-VirtualBox:~$ sudo iptables -I INPUT -p tcp --dport 80 -m connlimit --connlimit-above 50 --connlimit-mask 20 -j DROP
```

The `--connlimit-above 50` will allow at most 50 connections. The `--connlimit-mask 20` groups IPs using that prefix length. Every IP from the same /20 network is subject to that 50 connection limit.

When we running Slowloris tool, the result give the Socket counting number is under 100.

```
b1809707@b1809707-VirtualBox:~/slowloris/slowloris$ python3 slowloris.py 10.0.2.15 -s 500
[16-12-2021 02:05:16] Attacking 10.0.2.15 with 500 sockets.
[16-12-2021 02:05:16] Creating sockets...
[16-12-2021 02:05:20] Sending keep-alive headers... Socket count: 43
[16-12-2021 02:05:39] Sending keep-alive headers... Socket count: 43
```

VII. Feelings and Opinions

In summary this project was a great exploration of a Web Server, its default installation and security posture at the install time. This project covered a lot of basic installation defaults and operating system configuration changes that should be made in order to make the server production ready. The project walked through various security controls, securing

common services and applications like mod_qos, HAProxy load balancer and also provided an insight to firewall configuration.

Even though the project is not a complete security solution to a Apache Web Server, it is a good starting place towards a secure web server. The security issues covered by the project are Slowloris DoS attack which when left vulnerable makes the server an easy target and the attackers may enjoy the vulnerabilities and compromise the server there by impacting business operations, confidentiality, integrity and availability of data and information contained in the server as well as pose a threat to consumer's personal information.

Thus in summary, this project presents a good starting point for security of Web Server. It includes a module real-time monitoring called mod_qos, as well as firewall (IPTABLES) configuration to protect the server, and ensure availability of web server and information served and contained by the load balancing system like HAProxy.

VIII. References

- [1] <https://cybersecuritynews.com/linux-firewall-iptables/#:~:text=A%20Linux%20firewall%20is%20a,rules%20on%20a%20Linux%20machine>
- [2] <https://whatis.techtarget.com/definition/Web-server>
- [3] <https://www.kaspersky.com/resource-center/threats/ddos-attacks>
- [4] <https://kemptechnologies.com/blog/load-balancing-and-ddos-attacks/#:~:text=Load%20balancers%20also%20add%20resiliency,exhaust%20resources%20and%20saturate%20links>
- [5] <https://www.techrepublic.com/article/ddos-attacks-increased-by-20-in-2020-meaning-everyone-should-consider-themselves-at-risk/>
- [6] <https://www.docker.com/resources/what-container>