

DDOS ATTACKS

CECS 378 CYBERSECURITY PRINCIPLES

Professor: Louis Uuh

Date: 04/23/2021

SEMESTER PROJECT REPORT

Group 1:

1. Long Nguyen
2. Thanh Nguyen
3. Fiona Le
4. Dorothy Nguyen



Table Of Contents

What is DDOS Attack?	2
Setting up a Web Server	2
Introduction:	2
Method and Procedure:	2
Result:	5
Attack methods	5
UDP Flood using Socket in Python	5
Method idea:	5
Procedure:	5
Result:	7
ICMP (Ping) Flood using Scapy in Python	7
Introduction about Scapy module:	7
Idea of method:	8
Procedure:	8
Result:	10
SYN Flood using Scapy	10
Method idea:	10
Procedure:	11
Result:	12
Source code link	12
Ping of death	12
Abstract:	12
Introduction:	13
Method and Procedure:	13
Result:	14
Conclusion	17
Contribution	18
Works cited	19

What is DDOS Attack?

A Distributed Denial of Service (DDoS) attack is a result of multiple sources flooding the bandwidth or the resources of a victim machine (web server), which makes a website unavailable to users. Unlike DoS attack is implemented in one computer, a DDoS attack uses multiple devices to attack the target.

There are several ways to perform the DDoS attack. In our project, we use 4 different methods. All methods are programmed in Python.

I. Setting up a Web Server

1. Introduction:

In order to carry out the project of DDOS attack, we need to set up a webserver in which would be the control environment for us to test our attack on using either ping to death method, http flood and so on....

2. Method and Procedure:

One of the methods that we used to set up the server was using a virtual machine created using Virtual Box that will run a WAN server software. In this sense, the software will treat the virtual machine as physical hardware that is dedicated to run the server. However, the downside to this method when we did this was the physical machine that hosted the virtual machine of the team member who set up the server will have to be online 24/7. In other words, the person that is

hosting the server will need to keep his or her computer power on as long as there are others who need to work on it which is not an ideal solution.

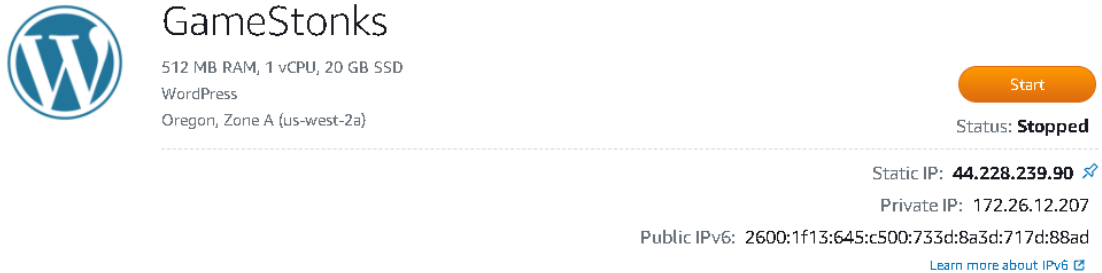
In addition, we tried using Amazon AWS service S3 in order to set up our server. In order to create a server on Amazon S3, the environment for the application that we will use will need to be specified; in this case, it can run Linux, Windows, Python, Java, ... After that, we will need to create an “instance” of our server. “Instances” in this sense are mini computers that are set up by using physical resources that are set aside by an Amazon server located at the location of our choice in this case would be Oregon. When setting up an “instance” of our server, we need to choose the amount of hardware will be dedicated to it and each of them have different cost depending on the amount of resources. Furthermore, we can also set up SSH connections to the server using the private key generated through Amazon AWS and download the private key.

However, using Amazon S3 service we were unable to set up a dummy website using some simple HTML files. The service only works well when it is deployed with the sample zip file provided by Amazon to set up a simple welcome page. When deploying a simple HTML file with the same package provided, the server health turns bad and reverts to the previous state which is the sample welcome page. In addition, we were unable to access the server command prompt despite having a private key.

Last but not least, after some research we switched service from S3 to Lightsail which provides us with our own private server with it's own dedicated resources. In order to create a private server using lightsail, we needed to choose the amount of hardware that would be dedicated to the server. We choose the least amount which is free; which gives us 500MB of RAM and 1 virtual CPU as well as 20GB of SSD memory space. Furthermore, we used Wordpress as our environment to run our dummy website. It is worth mentioning that we did

attempt to run Node.js server; however when the server runs we simply were not sure of how to deploy the website. In the process of creating the server, we were given a private key in order to connect and access the command line of our server. Using the private key and the server IP address, we can use our own SSH client to connect to the server in which case we used the SSH client provided by Amazon AWS Putty but any SSH client would work. However, we can also access the server just by signing in to Amazon AWS and there would be a webserver SSH set up for us. Then, we obtained a static IPv4 address in order to keep our IP address constant which is useful when we obtain a domain name. We then set up the dummy website using templates provided by WordPress in order to save time. After that, we obtained a free domain name through a website called “Namecheap.com” which requires a student email ending in .edu. We then linked the server IPv4 address with the domain name “ilikethestock.me” by creating a DNS zone and attaching the domain name to the server that we want and routing the domain name from the provider to Amazon AWS server. Up until this point however, our web page is still running using HTTP instead of HTTPS protocol. In order to change this, we needed to obtain a certificate which was also provided for free through Amazon. After obtaining the certificate through the server command line using the create certificate tool by providing the domain name and an email address. After the process is done, our website was automatically set up to use HTTPS protocol instead which is more secure.

3. Result:



GameStonks

512 MB RAM, 1 vCPU, 20 GB SSD
WordPress
Oregon, Zone A (us-west-2a)

[Start](#)

Status: **Stopped**

Static IP: **44.228.239.90** [↗](#)
Private IP: 172.26.12.207
Public IPv6: 2600:1f13:645:c500:733d:8a3d:717d:88ad
[Learn more about IPv6](#) [↗](#)

II. Attack methods

1. UDP Flood using Socket in Python

a. Method idea:

- This is one type of Volumetric Attacks. This type of attack will overload the capacity of the server with numerous User Datagram Protocol (UDP) packets.

b. Procedure:

- Step 01: Create a socket object by using ***Socket*** = ***socket.socket(socket.AF_INET, socket.SOCK_DGRAM)***
 - AF_INET refers to the address family ipv4 that is used as the transport mechanism.
 - SOCK_DGRAM refers to the User Datagram Protocol (UDP) which will be used as the default protocol. UDP is known as a connectionless protocol, which means it does not require any

handshake like TCP protocol, and does not care about resending the packets that were lost. That leads the flood attack to be executed faster and requires a support of few resources.

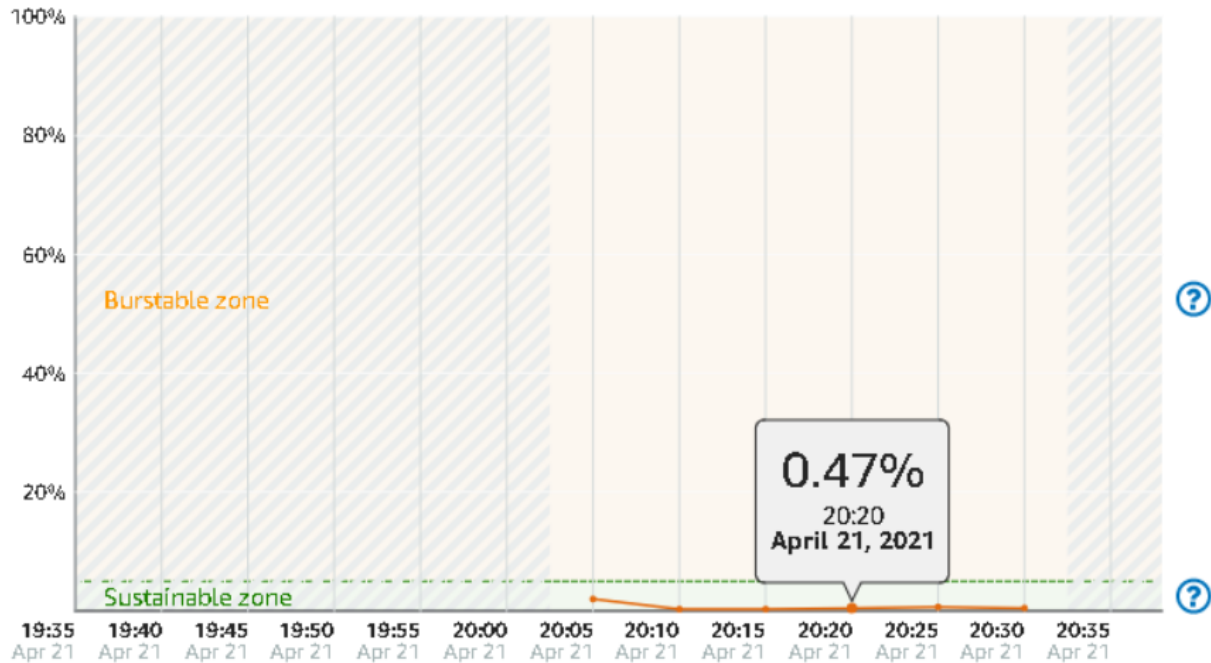
- Step 02: Ask the user to enter the IP address of the target and the port number where to send the packets to the target.
- Step 03: Run the endless loop to send packets to the target
- Step 04: In each loop, send packet to the target using the command line:

`socket.sendto(payload, (targetIP, targetPort))`

- targetIP - the address of the target
- targetPort - the port number
- payload - the packet will be sent to the target. This is created randomly using `payload = random._urandom(1024)`. This command creates a string of 1024 random bytes.
- socket.sendto() - This method initiates the connection to the target, then transmits UDP messages to it.

c. Result:

AVERAGE CPU UTILIZATION PER 5 MINUTES



Explain the result: The data went up from 0 to 0.47%. We had around 20 machines running the script to send UDP packets to the server. We need more devices to actually crack the target server.

2. ICMP (Ping) Flood using Scapy in Python

a. Introduction about Scapy module:

- Scapy is a module in Python, which is used for network packet manipulation. It allows users to generate, forge, send and receive, or decode packets on the networks to capture packets and analyze them, to match the requests and the replies, and more. It provides users with the ability of scanning, tracerouting, or discovery and attack networks.

- Scapy supports a wide number of network protocols (356 protocols) like TCP, DHCP,...
- Scapy is supported by both Python 2 and Python 3. To install the Scapy module into the Python library, users can run the command line `python -m pip install scapy`.

b. Idea of method:

- The attacker sends numerous ICMP Echo Request (ping) packets to a web server to overload the target capacity. We send the request and expect a response back from the server, so this attack uses ingoing bandwidth as well as outgoing bandwidth to receive messages and send responses. Then, it will slow down the target system.
- To do this task, we have to generate a packet which uses the IP layer to store the destination IP address, ICMP layer, and a payload which is the packet content. Then, we will endlessly send this packet to the victim machine.
- We will run the script on multiple computers at the same time to perform a DDoS attack.

c. Procedure:

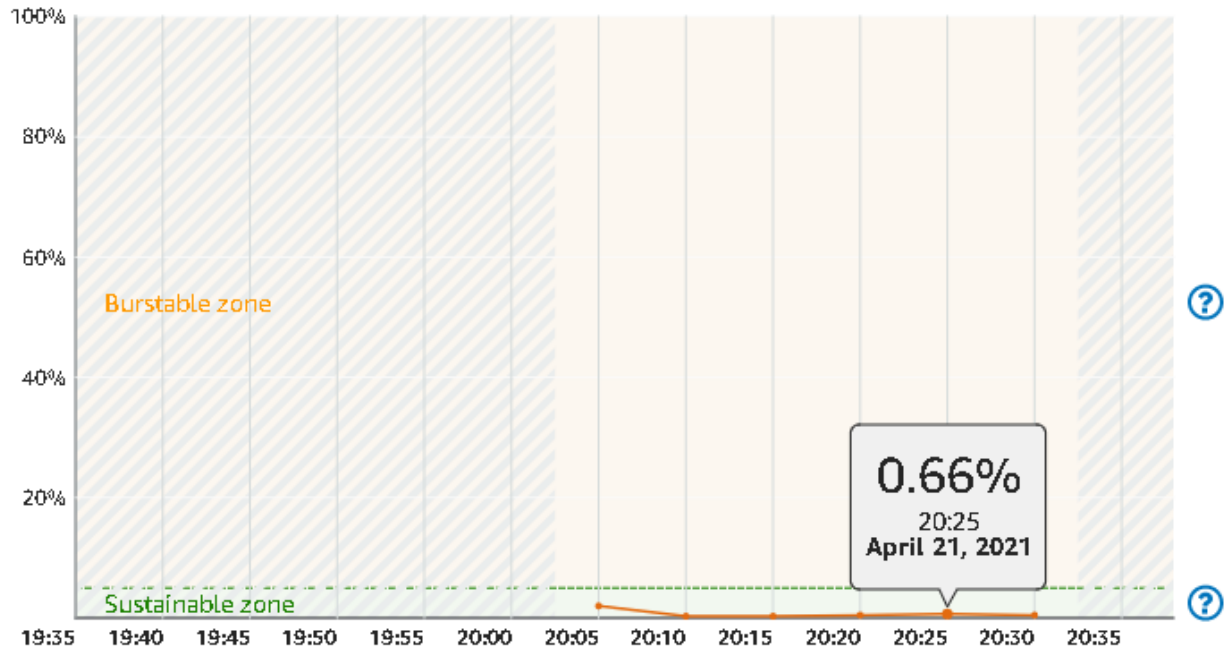
- Step #01: Request a user to enter the IP address of the target (web server). Then, store this IP address into the variable called *targetIP*.
- Step #02: We do the endless loop to to send packets to the server and attack it.

- Step #03: Create a packet: *packet = IP_Packet() / ICMP_Packet() / payload*
 - Create an IP packet: *IP_Packet = IP(dst = targetIP)*
 - Scapy will fill out the *src* address when it sends packets.
 - We need to define the *dst* address where packets are sent to.

In our case, this address is the IP address of the victim machine (web server).
 - Create an ICMP packet: *ICMP_Packet = ICMP()*
 - For the payload, we define it as *payload = "GROUP1" * 3000*
- Step #04: Sending packets to Server: *sr(packet)*
 - *sr()*: This function works at layer 3. It is used to send packets and receive the responses. It will return a couple of lists: a list of which packets were sent and its answers, and a list of unanswered packets.

d. Result:

AVERAGE CPU UTILIZATION PER 5 MINUTES



Result explanation: This method is our best method until now. Because the script sends packets and requires for the response, so it uses both ingoing and outgoing bandwidth. So that, the data is quite higher than other methods that we used. The data went up to 0.66%. We also run 21 devices for testing this script.

3. SYN Flood using Scapy

a. Method idea:

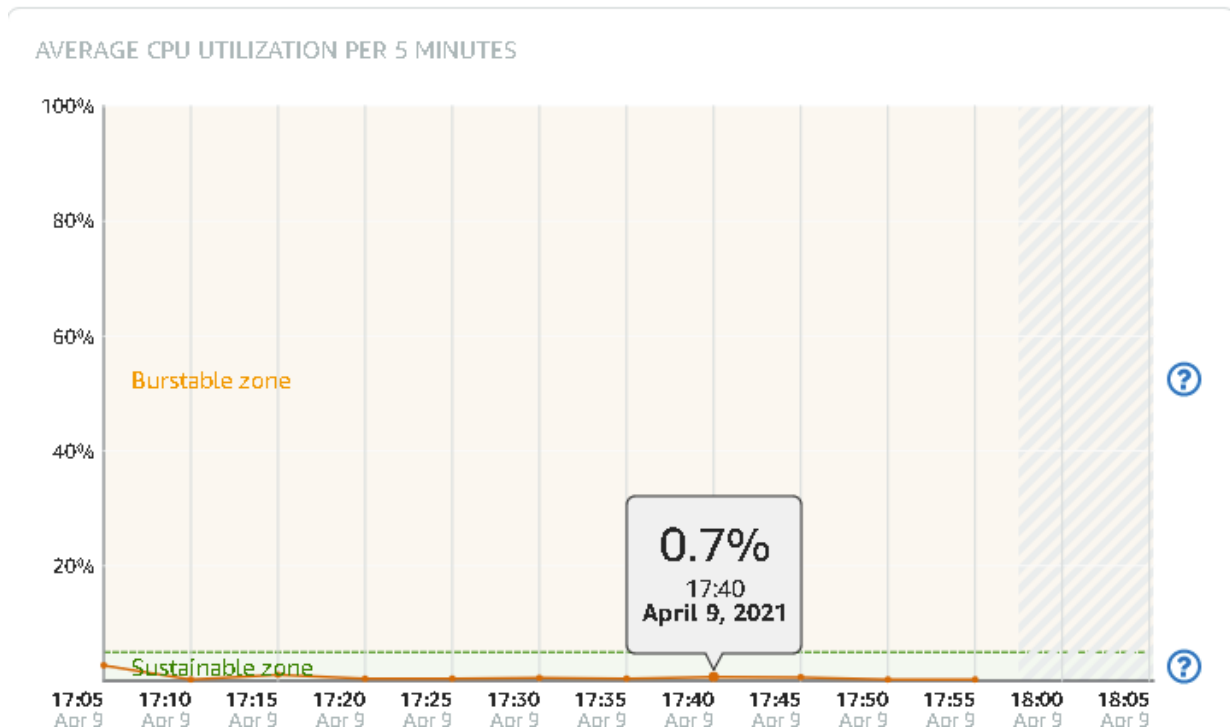
- In this attack method, clients will repeatedly send SYN packets to the server, then the server sends back with SYN-ACK. The attacker does not respond when the server expects an ACK packet. Therefore, it exhausts its resources waiting for the ACK packet.

- To perform the DDoS attack, the speed of sending packets needs to be faster than the time the target needs to process the request. That causes the target system to consume all available resources, then it cannot respond to other users.

b. Procedure:

- Enter the target IP address and create a random IP source.
- Create an packet to send to the server using: *packet = IP_Packet / TCP_Packet*
 - Create *IP_Packet = IP(src = sourceIP, dst = targetIP)*
 - *src* - the random IP, which is served as attacker
 - *dst* - the IP address of the target
 - Generate *TCP_Packet = TCP(sport = 443, dport = 443, flags = "S", seq = packetIP)*
 - The flags "S" means that Scapy will send a SYN packet to the server
 - *sport, dport*: the port that packet is sent from or to. We choose port 443 for both.
 - *seq* is the sequence number of the packet.
- Send packet to the target by the command: *send(packet)*
 - *send()*: This function also works at layer 3. It sends the packets and does not receive the response.

c. Result:



Result explanation: The data went from 0 to 0.7%. We used almost 25 devices to run this method, so the result is a bit higher than two other methods. The data went up shows that the server received the requests from the attacker and replied to these requests.

Source: <https://github.com/long237/CS378>

4. Ping of death

a. Abstract:

Ping is a command-line utility used to send data packages to a specific IP address on a network, and it returns a result that tells users how long it took to transmit that data and get a response. This behavior is to test to see if a networked device is reachable. So, my group uses the

Ping command to test whether the setup server receives requests from devices and the devices receive responses back from the server.

b. Introduction:

To get a successful ping, the ping command must get responses from the computer pinged back to the original operating computer after it sends a request over the network to a specific device, and it shows how long each packet took to make the round trip. In contrast, if the computer does not get any responses, there is no reply. Ping invention is inspired by the term used in sonar technology that sends out audible sound to find an object and then listens for the echo to return when the sound hits the object. Thus, the distance and location of the object can be determined by measuring the time and direction of the returning sound wave.

Ping command typically sends four or five echo requests by default. The result of each echo request is displayed, showing the following information:

1. Whether the request received a successfully respond
2. How many bytes were receive in response
3. Time to Live (TTL)
4. How long the response took to receive
5. Statistics about packet loss and round trip time.

c. Method and Procedure:

To test whether the server we set up in part 1 works successfully, we used 16 devices pinging to the server IP simultaneously and kept this process running continuously for 15 minutes. We monitored **the percentage of CPU utilization difference** during that period before and after running the ping process within 15 minutes.

- ❑ 16 computers (including local computers and virtual machines)

❑ Windows Command Prompts on Windows or Terminal on Ubuntu

- For Windows, we run command prompts as administration to ping.
- For Ubuntu, we run the terminal before starting to ping.

❑ Ping Command 1

- Ping Command is so simple, just one line syntax: `ping [targetedIP]`
- In this project, our server IP is **44.228.239.90**, so the Ping command we used is:

`ping 44.228.239.90`

❑ Ping Command 2

- Ping Command is so simple, just one line syntax: `ping <IP address> -f -l <packet size> -t`
- In this project, our server IP is **44.228.239.90**, so the Ping command we used is:

`ping 44.228.239.90 -f -l <packet size> -t`

- *-t is used to make the command line running continuously, on Windows while -t is not necessary on Ubuntu.*

❑ We have tried to Ping of Death **two times, using different ping command** and got the different results, which are shown in the Result part.

d. Result:

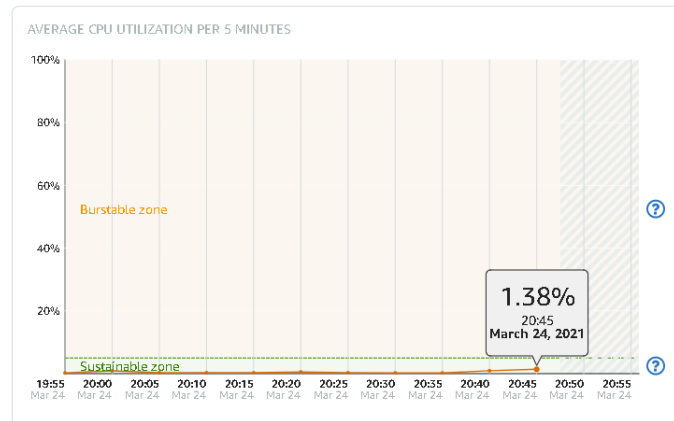
Trial 1:

Explanation: After running a ping request from all devices within 15 minutes, we stopped pinging and got the result in the capture below.

The data went from **0.2% to 1.38%**, which reflects the differences were 1.1%. The data shows that the setup server received requests from all devices, and all devices get responses from the

server. However, the data result is too low due to a lack of devices. Hence, we need more devices, at least 145 devices, to get 10%.

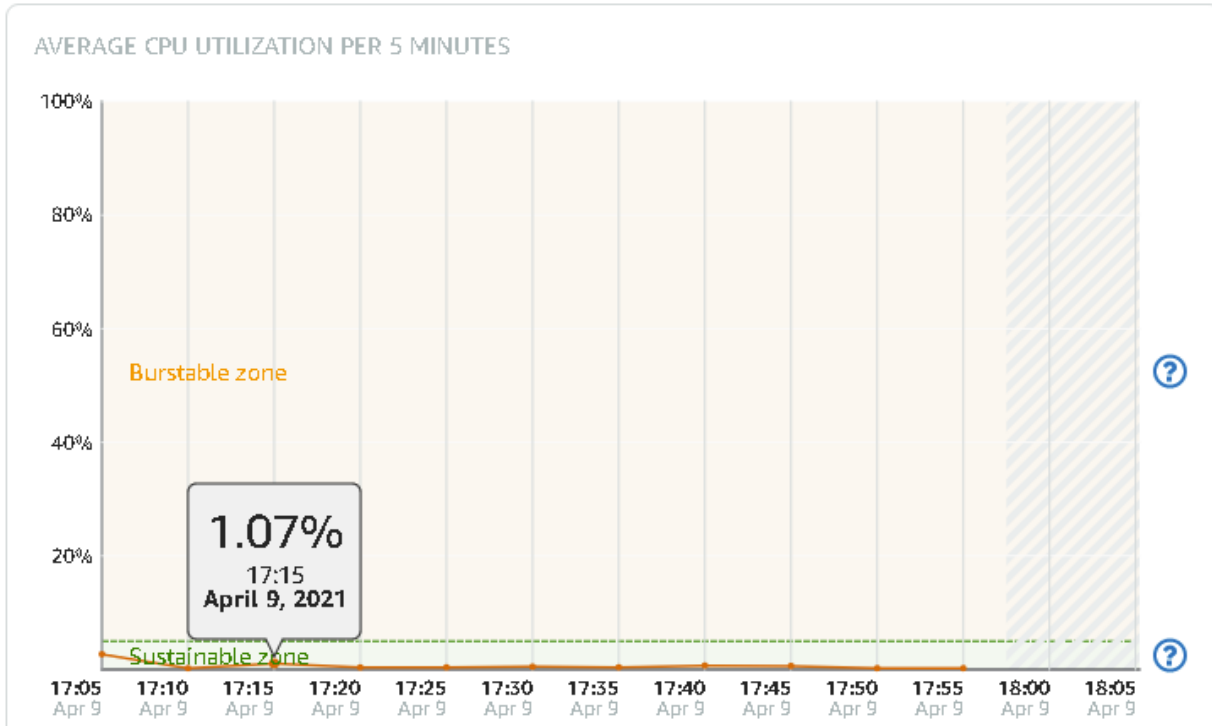
[Learn more about burst capacity](#)



Trial 2: Assigning packet size to increase the amount of dataset sending to the server.

Explanation: After running a ping request from all devices within 15 minutes, we stopped pinging and got the result in the capture below.

The data went from **0.2% to 1.07%**. The data shows that the setup server received requests from all devices, and all devices get responses from the server. However, the data result is too low due to a lack of devices. Hence, we need more devices, at least 145 devices, to get 10%.



III. Conclusion

In conclusion, the scripts that we wrote work since they increase the server resources usage which is observed through the CPU utilization graph provided by Amazon Lightsail service. However, the most machines that we used to run the attacks were limited to 20-25 machines due to limited resources. From the graph, the server CPU baseline utilization is 0.2%. Our most successful attack raised the utilization up to 1.38% which is an increase of 1.18%. That means each virtual machine that runs the script and sends a request to the server raises about 0.06% of CPU utilization. Which means that in order to overload the server we would need more than 1700 devices to send requests assuming that the CPU utilization scales linearly with the amount of machines we have.

Contribution

1. Long Nguyen:
 - a. Setting up the server on Lightsail, Amazon AWS
 - b. Setting up the website `ilikethestock.me`
 - c. Obtain domain name and SSL certificates
 - d. Testing the attack on the server by running scripts through virtual machines
 - e. Work on the report and the slide
2. Thanh Nguyen
 - a. Work on writing `DDOS.py`, `DDOS_Script.py`, `PingFlood.py`
 - b. Testing the attack on the server by running scripts through virtual machines
 - c. Work on the report and the slide
3. Fiona Le
 - a. Work on writing `DDOS_script.py`
 - b. Testing the attack on the server by running scripts through virtual machines
 - c. Work on the report (Ping of Death) and the slide
4. Dorothy Nguyen
 - a. Testing the attack on the server by running scripts through virtual machines
 - b. Work on the report and the slide

WORKS CITED

- [1] “Ethical Hacking - DDOS Attacks.” *Tutorialspoint*,
https://www.tutorialspoint.com/ethical_hacking/ethical_hacking_ddos_attacks.htm

- [2] LS. (1989). Retrieved April 23, 2021, from
https://lightsail.aws.amazon.com/ls/docs/en_us/articles/amazon-lightsail-quick-start-guide-wordpress

- [3] Bitnami WordPress stack for AWS Cloud. (2021, April 07). Retrieved April 23, 2021, from
<https://docs.bitnami.com/aws/apps/wordpress/>

- [4] LS. (1989). Retrieved April 23, 2021, from
https://lightsail.aws.amazon.com/ls/docs/en_us/articles/lightsail-how-to-create-dns-entry

- [5] Scapy usage, from <https://scapy.readthedocs.io/en/latest/usage.html>