

## **CS471**

### **Assignment 4**

**Issiah Deleon**

#### **Abstract**

The purpose of this lab is to demonstrate the utility of firewalls in creating, blocking, and redirecting network traffic, to analyze the limitations of mac address filtering, and to become more familiar with the Secure Shell Protocol (SSH) process.

#### **Introduction**

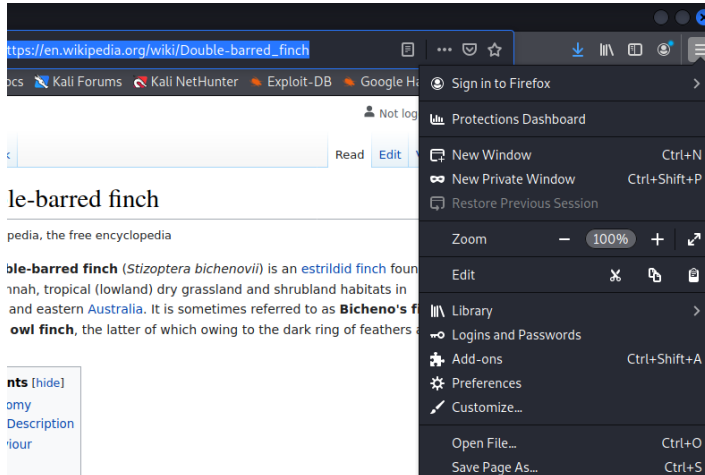
We will perform the demonstration on a virtual Kali Linux VirtualBox Machine which will be “attacking” a different virtual Ubuntu system. Within the Kali machine we will use the IPTables firewall to simulate ip blockages, network traffic redirections, and network traffic creations. We will use macchanger to demonstrate the inherent problems of using mac addresses as a single line of defense against attacks. Additionally we will use PuTTY emulating software on Windows to establish an SSH connection with our Kali Linux VM as a showcase of the SSH encryption and communication process. Traffic packets will be collected using WireShark to use as proof of successful connections or communications. Both the Kali and Ubuntu machines are set to “bridged adapter” modes to ensure that they exist on the same network.

#### **Summary**

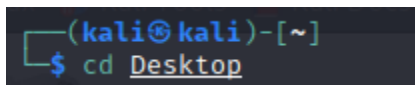
Begin by opening one Kali virtual machine and one Ubuntu virtual machine. The Kali machine will act as the attacking system. The Ubuntu machine will serve as the target system.

We will first create a convincing fake webpage using some simple python. On the Kali virtual machine, open a web browser and visit [this](#) website (it can be any website, but use this one for

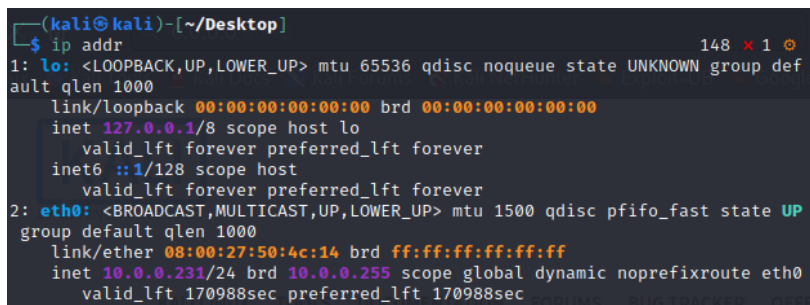
demo purposes). While on the website, click on Firefox's settings tab to the top right with the three horizontal bars. Click "Save Page As" and proceed to save the website to our Kali Linux Desktop.



Next, in our Kali virtual machine, open up a terminal window. Enter this command to change your directory to the Desktop, which contains our website's html file: **cd Desktop**.



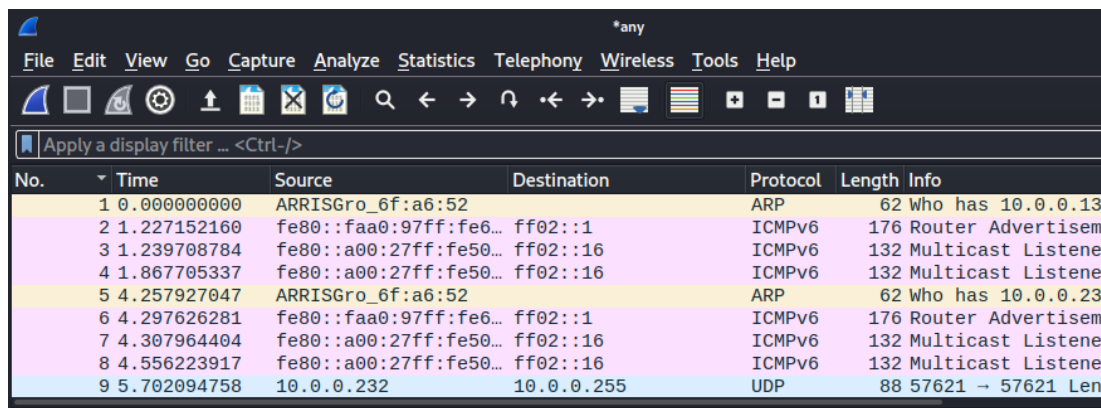
Before we continue, enter this command into your Kali machine to determine its IP address: **ip addr**. We will use the ip address listed under eth0: 10.0.0.231.



Next, enter this command into your Kali machine: `python2 -m SimpleHTTPServer 8089`. This will host a simple HTTP server on port 8089.

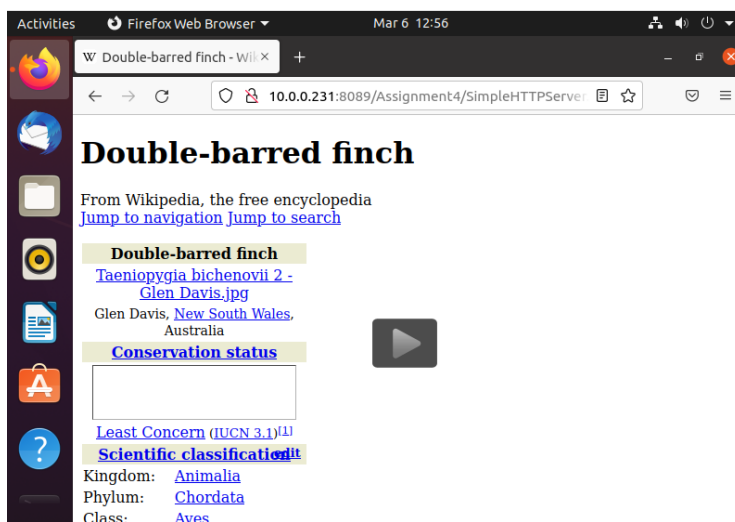
```
(kali㉿kali)-[~/Desktop]
$ python2 -m SimpleHTTPServer 8089
Serving HTTP on 0.0.0.0 port 8080 ...
127.0.0.1 - - [06/Mar/2022 15:46:21] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [06/Mar/2022 15:46:21] code 404, message File not found
127.0.0.1 - - [06/Mar/2022 15:46:21] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [06/Mar/2022 15:47:29] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [06/Mar/2022 15:47:29] code 404, message File not found
127.0.0.1 - - [06/Mar/2022 15:47:29] "GET /favicon.ico HTTP/1.1" 404 -
```

Now open Wireshark in your Kali machine and select “any” for packet capture type. Begin listening before we proceed. Wireshark will begin collecting packets.

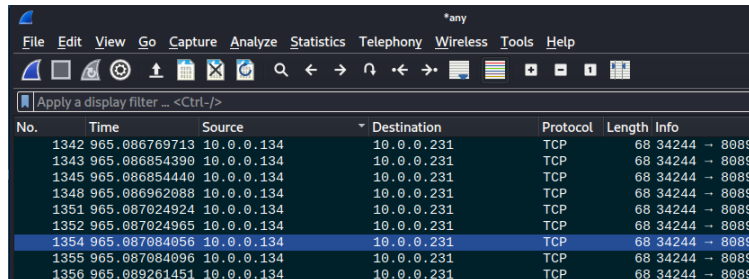


Now on our Ubuntu machine, open up a web browser window. In this window, type in the IP address of our Kali machine followed by the port number that we are hosting the fake website on: 10.0.0.231:8089.

We are now able to view the “fake” website on our Ubuntu machine.



Now if we revisit Wireshark on our Kali machine, we can filter by source and destination. The IP address of our source is 10.0.0.134, and the IP address of our destination is 10.0.0.231. We will capture packets with these source and destination IP addresses.



The image shows the Wireshark interface with a display filter applied. The packet list table is as follows:

No.	Time	Source	Destination	Protocol	Length	Info
1342	965.086769713	10.0.0.134	10.0.0.231	TCP	68	34244 → 8089
1343	965.086854390	10.0.0.134	10.0.0.231	TCP	68	34244 → 8089
1345	965.086854440	10.0.0.134	10.0.0.231	TCP	68	34244 → 8089
1348	965.086962088	10.0.0.134	10.0.0.231	TCP	68	34244 → 8089
1351	965.087024924	10.0.0.134	10.0.0.231	TCP	68	34244 → 8089
1352	965.087024965	10.0.0.134	10.0.0.231	TCP	68	34244 → 8089
1354	965.087084056	10.0.0.134	10.0.0.231	TCP	68	34244 → 8089
1355	965.087084096	10.0.0.134	10.0.0.231	TCP	68	34244 → 8089
1356	965.089261451	10.0.0.134	10.0.0.231	TCP	68	34244 → 8089

We will now manipulate blocked addresses. On our Kali machine, we will find the network subnet of Facebook by entering this command: `host -t a www.facebook.com`.

```
(kali㉿kali)-[~/Desktop]
$ host -t a www.facebook.com
www.facebook.com is an alias for star-mini.c10r.facebook.com.
star-mini.c10r.facebook.com has address 157.240.22.35
```

We will use this address to then find the entire network for the single host by entering this command: `whois 157.240.22.35 | grep CIDR`

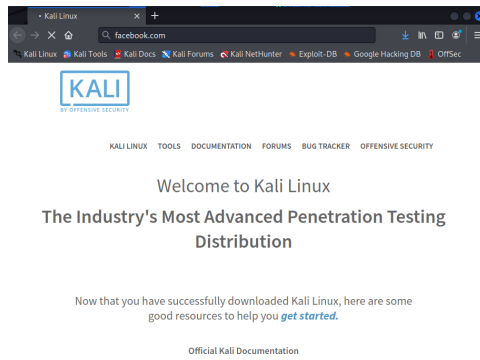
```
(kali㉿kali)-[~/Desktop]
$ whois 157.240.22.35 | grep CIDR
CIDR:      157.240.0.0/16
```

We will then use the entire network IP to drop all connections to this network: `sudo iptables -A`

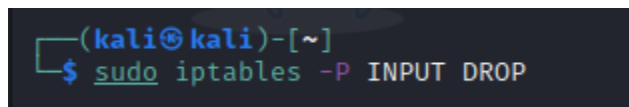
`OUTPUT -p tcp -d 157.240.0.0/16 -j DROP`

```
(kali㉿kali)-[~/Desktop]
$ sudo iptables -A OUTPUT -p tcp -d 157.240.0.0/16 -j DROP
[sudo] password for kali:
```

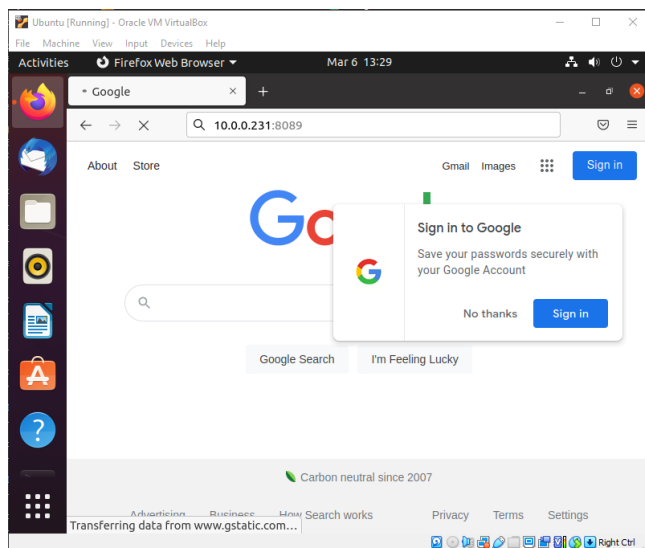
If we try to visit Facebook.com on our Kali machine, we notice that the website never loads.



Now if we wanted to block all incoming requests to port 8089, we could enter the following command: `sudo iptables -P INPUT DROP`.



We notice that if we attempt to revisit the fake webpage on our Ubuntu machine, it will never load



We have effectively blocked any inbound access to our Kali machine.

Next, remove the new rule we just created by entering: `sudo iptables -F`.

```
(kali㉿kali)-[~]  
$ sudo iptables -F
```

Before we move on to the next portion, take note of the fact that on our Kali Linux machine, there is an exchange of packets between the mac address of our Kali machine

(08:00:27:50:4c:14), and another device in pink.

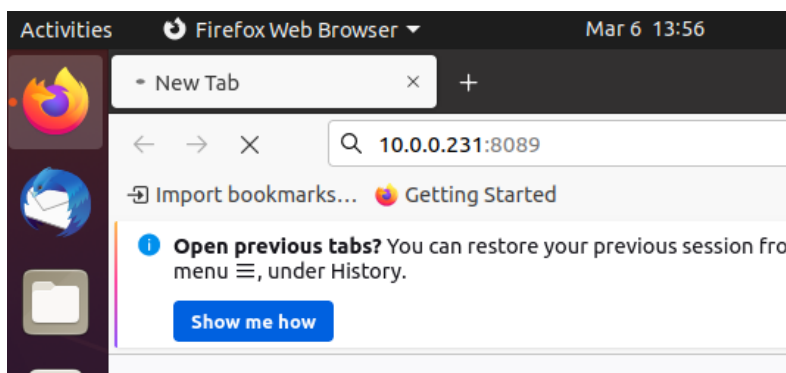
No.	Time	Source	Destination	Protocol	Length	Info
1089	341.039902151	2601:643:8300:59e0:...	2001:558:feed::1	DNS	119	Standard
1090	341.039926535	2601:643:8300:59e0:...	2001:558:feed::1	DNS	119	Standard
1091	341.343811823	fe80::a00:27ff:fe50...	ff02::16	ICMPv6	132	Multicas
1092	343.826666866	fe80::faa0:97ff:fe6...	ff02::1	ICMPv6	176	Router A
1093	343.835815161	fe80::a00:27ff:fe50...	ff02::16	ICMPv6	132	Multicas
1094	343.995802488	fe80::a00:27ff:fe50...	ff02::16	ICMPv6	132	Multicas
1095	344.228501595	10.0.0.231	75.75.75.75	DNS	87	Standard
1096	344.228527801	10.0.0.231	75.75.75.75	DNS	87	Standard
1097	344.242259222	75.75.75.75	10.0.0.231	DNS	141	Standard
1098	344.249326189	75.75.75.75	10.0.0.231	DNS	207	Standard

Now we will attempt to block by this mac address. On our Ubuntu system, open up a terminal window and enter the following command: `sudo iptables -A INPUT -m mac --mac-source`

`08:00:27:50:4c:14 -j DROP`. This will block inbound access from our Kali machine.

```
issiah@issiah-VirtualBox:~$ sudo iptables -A INPUT -m mac --mac-source 08:00:27  
:50:4c:14 -j DROP  
[sudo] password for issiah:
```

Now on our Ubuntu system if we attempt to visit the fake webpage that we created, it will still not be able to load.



We will now change the mac address of our Kali machine. To set it to a new random address, use the command: `sudo macchanger -r eth0`.

```
(kali㉿kali)-[~]
$ sudo macchanger -r eth0
[sudo] password for kali:
Current MAC: 08:00:27:50:4c:14 (CADMUS COMPUTER SYSTEMS)
Permanent MAC: 08:00:27:50:4c:14 (CADMUS COMPUTER SYSTEMS)
New MAC: 8e:ec:fd:b8:0d:30 (unknown)
(kali㉿kali)-[~]
```

We can now detect new packets from this new mac address: `8e:ec:fd:b8:ed:30`.

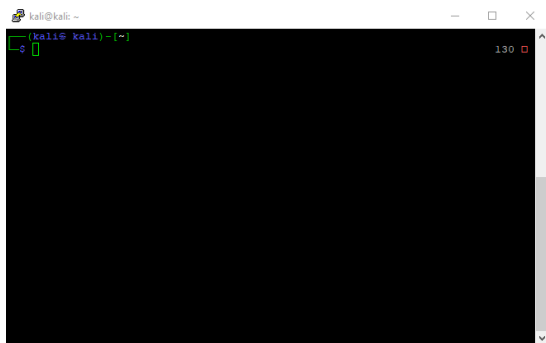
No.	Time	Source	Destination	Protocol	Length	Info
4424	1437.5736310...	2601:643:8300:59e0:...	2001:558:feed::1	DNS	113	Standard
4425	1437.5840982...	fe80::a00:27ff:fe50...	ff02::16	ICMPv6	132	Multicas
4426	1437.7602090...	fe80::a00:27ff:fe50...	ff02::16	ICMPv6	132	Multicas
4427	1438.9156326...	8e:ec:fd:b8:0d:30		ARP	44	Who has
4428	1439.9374985...	8e:ec:fd:b8:0d:30		ARP	44	Who has
4429	1440.5416007...	fe80::faa0:97ff:fe6...	ff02::1	ICMPv6	176	Router A
4430	1440.5524100...	fe80::a00:27ff:fe50...	ff02::16	ICMPv6	132	Multicas
4431	1440.8963488...	fe80::a00:27ff:fe50...	ff02::16	ICMPv6	132	Multicas
4432	1440.9640873...	8e:ec:fd:b8:0d:30		ARP	44	Who has
4433	1441.9176141...	2601:643:8300:59e0:...	2001:558:feed::1	DNS	127	Standard
4434	1441.9882742...	10.0.0.231	10.0.0.231	ICMP	135	Destinat

This demonstrates the vulnerability of a system from attacks that are composed of attackers manipulating their mac addresses to fool systems into “unrecognizing” them. Anyone can change their mac address, so blocking by mac address is not a perfect system for defense and security purposes.

For the following portion of the lab, we will establish an SSH connection with our Kali Linux VM and our host machine using PuTTY on Windows. First, in our Kali Linux machine, enter the following command in the terminal window: `sudo service ssh start`.

```
(kali㉿kali)-[~]
$ sudo service ssh start
```

This opens an ssh server and allows us to connect to our host machine using PuTTY. When opening PuTTY, we connect to the IP address corresponding to our Kali Linux VM (10.0.0.231), and use port number 22.



PuTTY opens successfully.

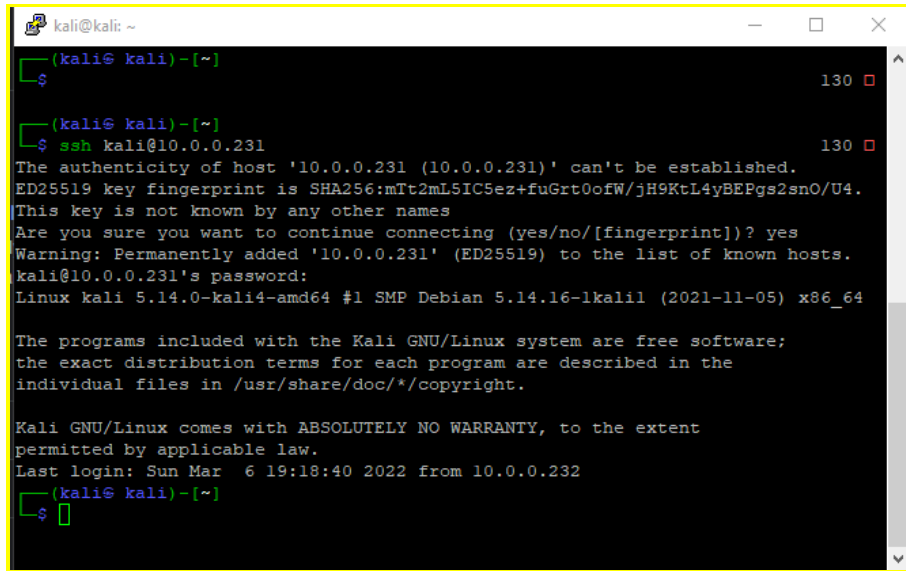
We also now take note of the fact that WireShark is picking up SSH packets. Our SSH connection has been successfully established.

A screenshot of the Wireshark network protocol analyzer interface. The packet list pane shows several captured packets. The selected packet is number 25, an SSH packet from 10.0.0.232 to 10.0.0.231. The packet details pane shows the structure of the SSH packet, including the 'SSH' protocol and 'Encrypted payload' field.

No.	Time	Source	Destination	Protocol	Length	Info
20	8.153086238	10.0.0.231	10.0.0.232	SSH	150	Server: Encrypted pa
21	8.194915594	10.0.0.232	10.0.0.231	TCP	60	59565 → 22 [ACK] Seq
22	8.301097684	10.0.0.232	10.0.0.231	SSH	118	Client: Encrypted pa
23	8.304062230	10.0.0.231	10.0.0.232	SSH	134	Server: Encrypted pa
24	8.344327153	10.0.0.232	10.0.0.231	TCP	60	59565 → 22 [ACK] Seq
25	8.498313628	10.0.0.232	10.0.0.231	SSH	118	Client: Encrypted pa

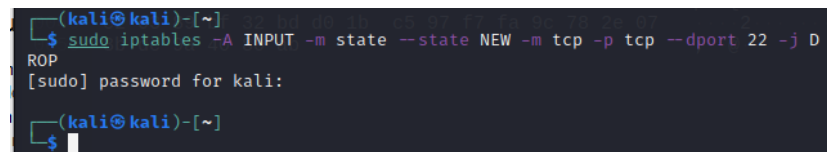


In PuTTY, proceed by entering the following command: `ssh kali@10.0.0.231`. We are prompted to enter the password of the Kali system, as well as confirming the continuation of the connection process after being warned about the lack of host authenticity.



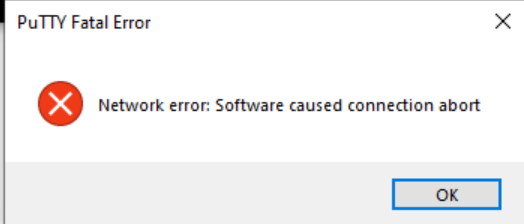
```
(kali@kali)~  
$  
(kali@kali)~  
$ ssh kali@10.0.0.231  
The authenticity of host '10.0.0.231 (10.0.0.231)' can't be established.  
ED25519 key fingerprint is SHA256:mTt2mL5ICSez+fuGrt0ofW/jH9KtL4yBEPgs2snO/U4.  
This key is not known by any other names  
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes  
Warning: Permanently added '10.0.0.231' (ED25519) to the list of known hosts.  
kali@10.0.0.231's password:  
Linux kali 5.14.0-kali4-amd64 #1 SMP Debian 5.14.16-1kali1 (2021-11-05) x86_64  
  
The programs included with the Kali GNU/Linux system are free software;  
the exact distribution terms for each program are described in the  
individual files in /usr/share/doc/*/copyright.  
  
Kali GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent  
permitted by applicable law.  
Last login: Sun Mar  6 19:18:40 2022 from 10.0.0.232  
(kali@kali)~  
$
```

Now use iptables to block port 22 traffic in our Kali machine: `sudo iptables -A INPUT -m state --state NEW -m tcp -p tcp --dport 22 -j DROP`



```
(kali@kali)~  
$ sudo iptables -A INPUT -m state --state NEW -m tcp -p tcp --dport 22 -j D  
ROP  
[sudo] password for kali:  
(kali@kali)~  
$
```

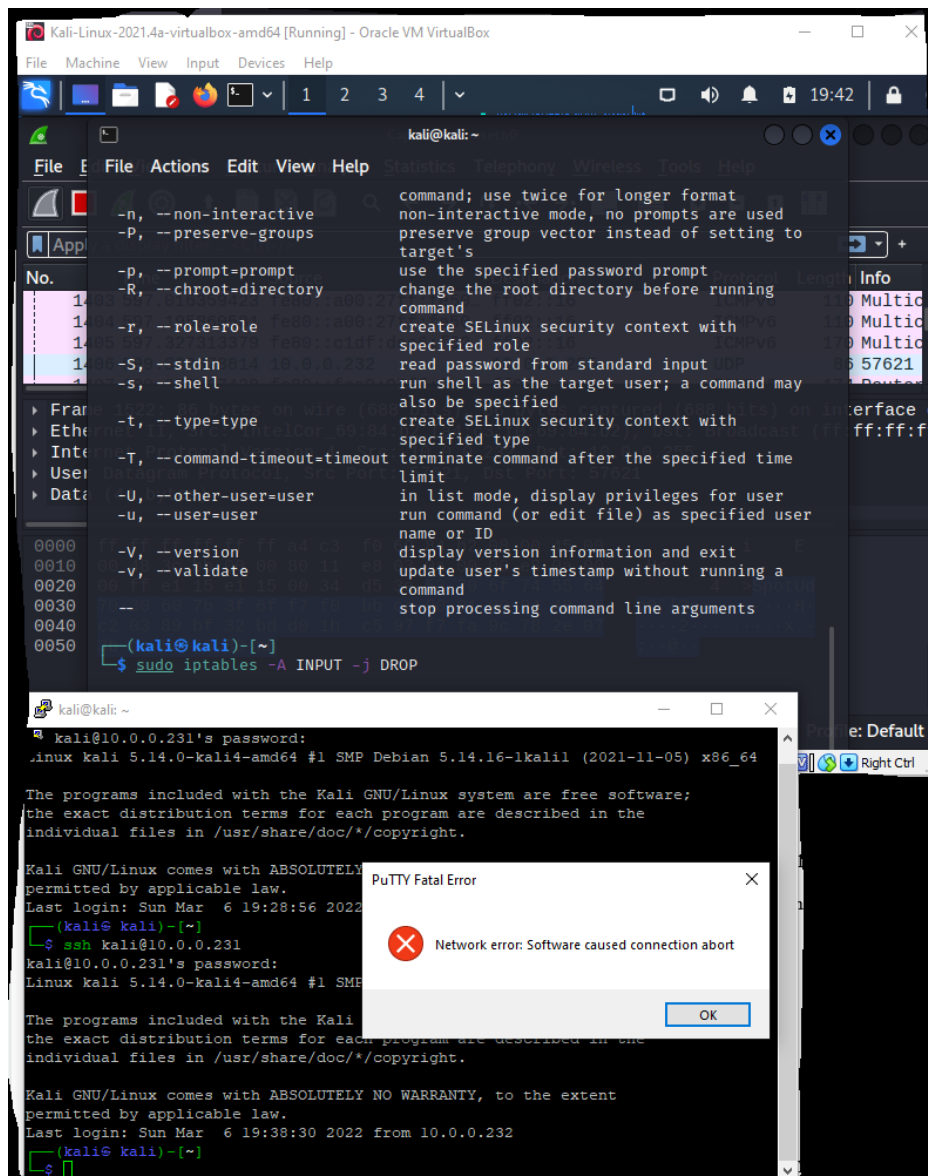
```
kali@kali: ~  
  
(kali@kali)-[~]  
$  
  
(kali@kali)-[~]  
$ ssh kali@10.0.0.231  
The authenticity of host '10.0.0.231 (10.0.0.231)' can't be established.  
ED25519 key fingerprint is SHA256:mTt2mL5IC5ez+fuGr  
This key is not known by any other names  
Are you sure you want to continue connecting (yes/no)  
Warning: Permanently added '10.0.0.231' (ED25519) to the list of known hosts.  
kali@10.0.0.231's password:  
Linux kali 5.14.0-kali4-amd64 #1 SMP Debian 5.14.16-  
  
The programs included with the Kali GNU/Linux system  
the exact distribution terms for each program are de  
individual files in /usr/share/doc/*/copyright.  
  
Kali GNU/Linux comes with ABSOLUTELY NO WARRANTY, to  
permitted by applicable law.  
Last login: Sun Mar  6 19:18:40 2022 from 10.0.0.231  
(kali@kali)-[~]  
$
```



A PuTTY Fatal Error dialog box is overlaid on the terminal window. The title bar reads "PuTTY Fatal Error". Inside the dialog, there is a red circle with a white "X" icon, followed by the text "Network error: Software caused connection abort". At the bottom right of the dialog is an "OK" button.

After this happens, we receive a fatal error in PuTTY as the connection is no longer possible.

We reestablished a connection through PuTTY after resetting our iptables rules on our Kali machine using the command: `sudo iptables -F`. We observe what happens to our SSH connection when blocking all traffic using the command: `sudo iptables INPUT -j DROP`. We encounter the same fatal error. The connection has ended.



We can block access to the webpage from a specific MAC address (blocking our ubuntu system) by entering the following command: `sudo iptables -A INPUT -m mac --mac-source`

`08:00:27:ea:f4:89.`

```
(kali@kali)-[~]  
$ sudo iptables -A INPUT -m mac --mac-source 08:00:27:ea:f4:89 -j DROP
```

As we can see, we are not able to visit our fake webpage.



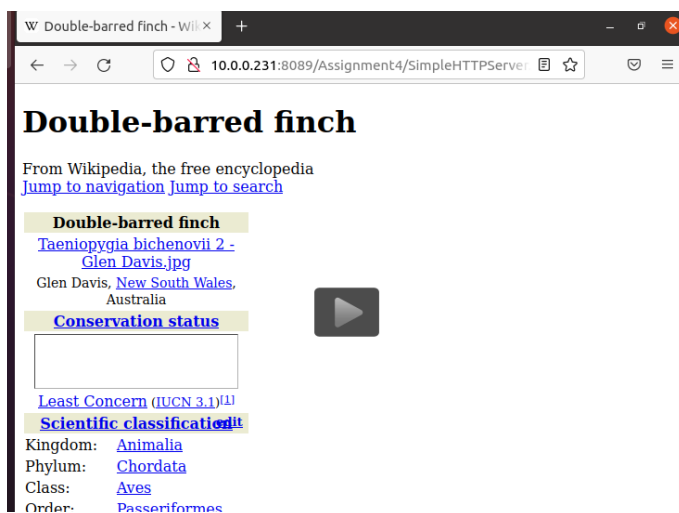
The connection has timed out

The server at 10.0.0.231 is taking too long to respond.

- The site could be temporarily unavailable or too busy. Try again in a few moments.
- If you are unable to load any pages, check your computer's network connection.
- If your computer or network is protected by a firewall or proxy, make sure that Firefox is permitted to access the Web.

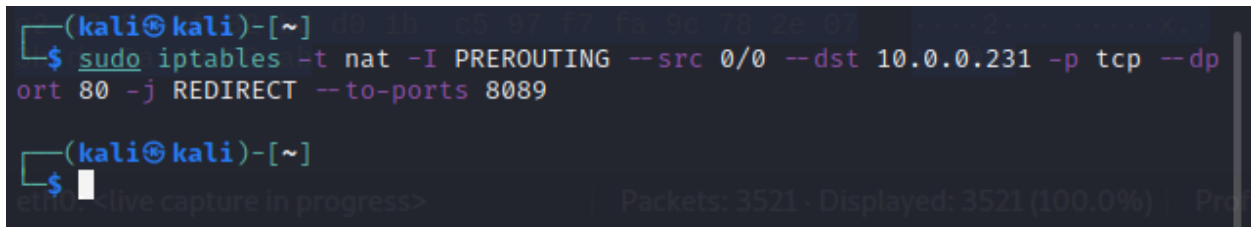
Try Again

Reset the rules in our Kali machine however: `sudo iptables -F`, we can view the website just fine.



Now if we want to redirect inbound requests for port 80 and send this traffic to our established python server on port 8089, we can enter the following command in our Kali Linux machine:

```
sudo iptables -t nat -I PREROUTING --src 0/0 --dst 10.0.0.231 -p tcp --dport 80 -j REDIRECT --to-ports 8089
```



Analysis

Some screenshots of the packets captured by Wireshark include:

No.	Time	Source	Destination	Protocol	Length	Info
1089	341.039902151	2601:643:8300:59e0::...	2001:558:feed::1	DNS	119	Standard
1090	341.039926535	2601:643:8300:59e0::...	2001:558:feed::1	DNS	119	Standard
1091	341.343811823	fe80::a00:27ff:fe50...	ff02::1	ICMPv6	132	Multicas
1092	343.826666866	fe80::faa0:97ff:fe6...	ff02::1	ICMPv6	176	Router A
1093	343.835815161	fe80::a00:27ff:fe50...	ff02::1	ICMPv6	132	Multicas
1094	343.995802488	fe80::a00:27ff:fe50...	ff02::1	ICMPv6	132	Multicas
1095	344.228561595	10.0.0.231	75.75.75.75	DNS	87	Standard
1096	344.228527801	10.0.0.231	75.75.75.75	DNS	87	Standard
1097	344.242259222	75.75.75.75	10.0.0.231	DNS	141	Standard
1098	344.249326189	75.75.75.75	10.0.0.231	DNS	207	Standard

Here we noticed our Kali machine communicating with other devices, identifying Kali by its mac address.

No.	Time	Source	Destination	Protocol	Length	Info
1089	341.039902151	2601:643:8300:59e0::...	2001:558:feed::1	DNS	119	Standard
1090	341.039926535	2601:643:8300:59e0::...	2001:558:feed::1	DNS	119	Standard
1091	341.343811823	fe80::a00:27ff:fe50...	ff02::1	ICMPv6	132	Multicas
1092	343.826666866	fe80::faa0:97ff:fe6...	ff02::1	ICMPv6	176	Router A
1093	343.835815161	fe80::a00:27ff:fe50...	ff02::1	ICMPv6	132	Multicas
1094	343.995802488	fe80::a00:27ff:fe50...	ff02::1	ICMPv6	132	Multicas
1095	344.228561595	10.0.0.231	75.75.75.75	DNS	87	Standard
1096	344.228527801	10.0.0.231	75.75.75.75	DNS	87	Standard
1097	344.242259222	75.75.75.75	10.0.0.231	DNS	141	Standard
1098	344.249326189	75.75.75.75	10.0.0.231	DNS	207	Standard

No.	Time	Source	Destination	Protocol	Length	Info
20	8.153086238	10.0.0.231	10.0.0.232	SSH	150	Server: Encrypted pa
21	8.194915594	10.0.0.232	10.0.0.231	TCP	60	59565 → 22 [ACK] Seq
22	8.301097684	10.0.0.232	10.0.0.231	SSH	118	Client: Encrypted pa
23	8.304062230	10.0.0.231	10.0.0.232	SSH	134	Server: Encrypted pa
24	8.344327153	10.0.0.232	10.0.0.231	TCP	60	59565 → 22 [ACK] Seq
25	8.498313628	10.0.0.232	10.0.0.231	SSH	118	Client: Encrypted pa

Here we noticed SSH packets. These SSH packets encrypt the transferred data using

asymmetric encryption. It utilizes the Diffie Hellman technique to negotiate key exchanges. We notice and save SSH key exchange packets to include with the assignment submission

## Conclusion

In conclusion, we have demonstrated the process of creating convincing fake web pages that could potentially be used for harm in the hands of an attacker. We manipulated Kali Linux firewall rules to demonstrate the interplay between firewall rules and network traffic access. We can block entire networks, single websites, ip addresses, ranges of ip addresses, mac addresses, or single hosts. These tools allow us to defend ourselves from malicious attacks. We also showcased the nature of mac address blocking, and showed how simply changing a mac address of an attacking system is enough to fool another system into unrecognizing this device. Finally, we established an SSH session between our host machine using PuTTY and our Kali VM using the terminal. We observed the transfer of SSH key exchange packets during this connection, as well as standard encrypted SSH packets. We manipulated the firewall rules to showcase how SSH sessions can be disrupted or prevented by blocking port numbers as well as mac addresses.

If we wanted to turn this process into an attack, we would need to complete some other steps. We would need to instruct our fake website to perform some action that constituted an attack. If we were to create a fake webpage and insert fake hyperlinks into it that tracked, collected, or manipulated the user's system or data in some way, we could constitute this as an attack.

IPTables can be used to improve security by blocking various avenues of network traffic. We can block by port number, ip address, a range of ip addresses, mac address, domains, or even entire networks such as the entire Facebook.com network. These tools can be integrated into a firewall system to add known malicious attackers to a blocked list so that our system never

accepts network traffic from them again. This is just one way among many that IPTables can be used to improve security.

Mac address filtering is the process of filtering network access/traffic by mac address. Each device has a unique and permanent mac address, and we can instruct our system to block network traffic associated with a specific mac address. However, we demonstrated in the lab the vulnerability of mac address filtering in that malicious users or individuals acting in bad faith can change their mac address using tools such as macchanger. Macchanger is a tool that allows users to easily manipulate their mac address, which would then allow the individual/entity to re-enter the network that they were previously banned from. This means mac address filtering is not a foolproof method of improving security.

Firewalls provide authentication between two parties by allowing an individual with a firewall to authenticate themselves with another anonymous entity that they would like to communicate with. Some firewalls have authentication services integrated, such as: authentication databases, certificate authentication methods, or two factor authentication. These authentication services are integrated into the firewall and ensure that our system is authenticated when it communicates with other anonymous entities over a network. Firewalls also provide forms of access control by determining who and what can connect to your system and exchange data. If your firewall has identified a malicious attacker's ip or mac address, the firewall can attempt to block the attacker and prevent them from accessing your network/system. A firewall will also provide data confidentiality because it is actively working as a line of defense against malicious individuals or entities. The firewall works to keep bad faith entities from accessing your data, system, hardware, or information that would otherwise be easily accessible and out in the open without the protection from a firewall. A firewall also provides data integrity, as it

ensures no outside actors are able to break into our network or system and manipulate data without your knowing. Occasionally attackers can slip by a firewall without the victim knowing, however firewalls operate as an active force against attacks and entities that wish to maliciously manipulate our data, serving as an active deterrent and defense against attacks that threaten the integrity of our data. Finally, firewalls do not typically offer a form of non-repudiation, as they are not necessarily keeping track of every single interaction taking place over a network. However it can be argued that firewalls do offer non-repudiation in that when an attack IS discovered by the firewall, it will provide proof and evidence of the attack/malicious action, allowing you to pinpoint your focus to the problem and present proof of an action if it causes your system/network/data harm.