Network Protection for Energy Control Systems

CSP004_C_E

PRESENTATION BY:
DR. STEFAN SCHAUER
DR. ABDELKADER SHAABAN
AIT AUSTRIAN INSTITUTE OF TECHNOLOGY

Network Protection for Energy Control Systems

These slides outline the essential offensive tools that will be used in this course.

These tools are intended for use within this course to demonstrate how different tools can be employed for various cyberattack activities and address existing security weaknesses to avoid or mitigate related cyber risks. Therefore, all these practical activities are solely intended for educational purposes ONLY and not for any other malicious or unauthorized activities.

Prevention Tools

iptables

What is the iptables?

- **iptables** is a **command-line** utility for configuring the **Linux kernel firewall**.
- It allows **administrators** to define **rules** to control **incoming** and **outgoing network** traffic.
- These **rules** provide **security** by determining which **network packets** can pass through or be blocked.
- iptables **helps** protect **Linux systems** from **data breaches**, **unauthorized access**, and other network **security threats**.
- Administrators **use** iptables to **enforce** security **policies** and **safeguard** systems **against network-based** attacks.

Iptables Tutorial: Ultimate Guide to Linux Firewall (phoenixnap.com)

How Does iptables Work?

iptables uses rules to **determine** what to do with a **network packet**. The utility consists of the following components:

- **Tables**. Tables are **files** that group similar **rules**. A table consists of several rule **chains**.
- **Chains**. A chain is a string of **rules**. When a packet is **received**, **iptables finds** the appropriate **table** and **filters** it through the **rule chain** until it **finds** a **match**.
- **Rules.** A rule is a **statement** that defines the **conditions** for **matching** a packet, which is then sent to a **target**.
- **Targets.** A target is a **decision** of what to do with a packet. The packet is either **accepted**, **dropped**, or **rejected**.
  - **ACCEPT**. Allows the **packet to pass** through the **firewall**.
  - **DROP**. Discards the packet **without informing** the **sender**.
  - **REJECT**. **Discards** the packet and **returns** an error response to the sender.

Iptables Tutorial: Ultimate Guide to Linux Firewall (phoenixnap.com)

How Does iptables Work?

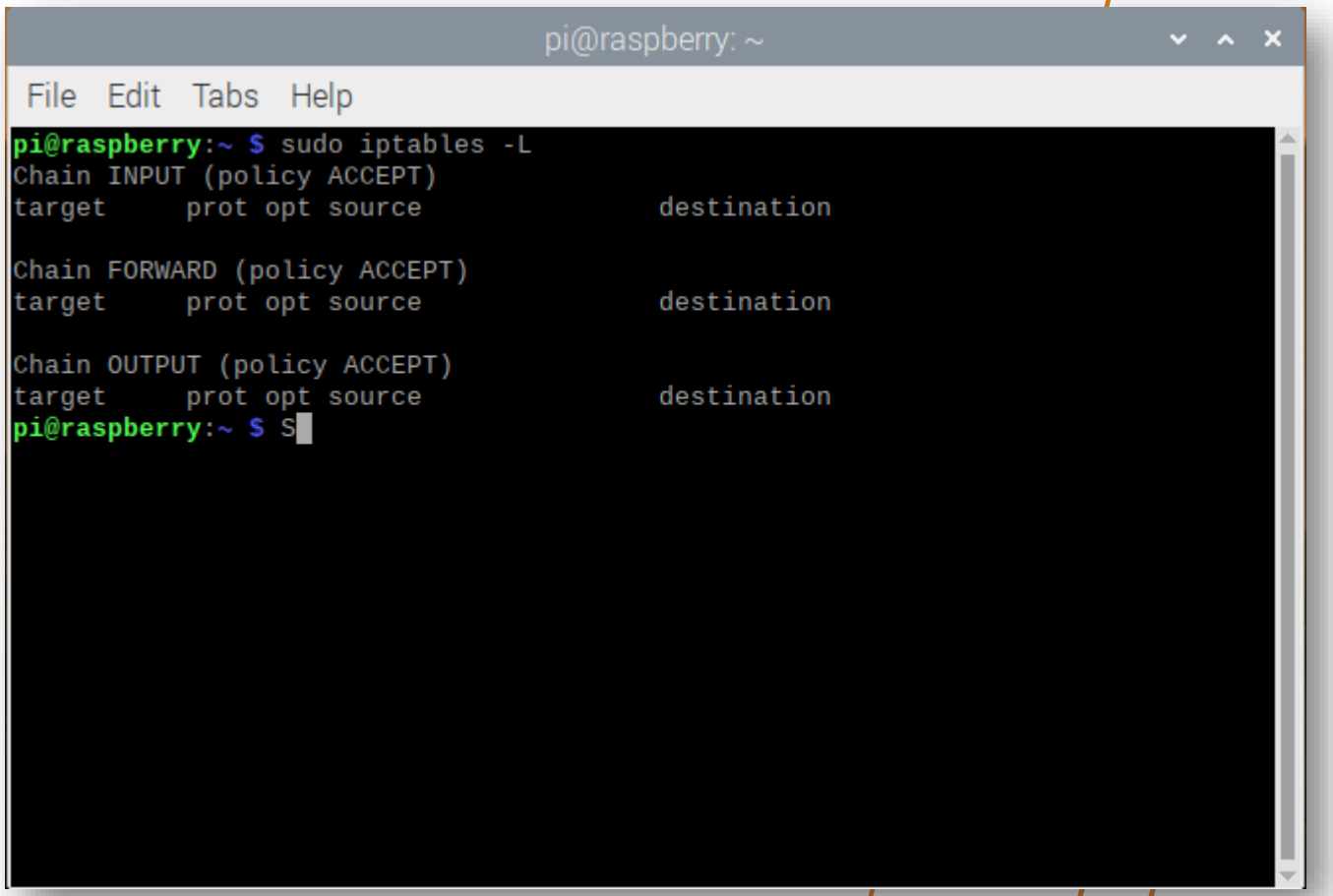
$ iptable <operations> <string> <conditions> <action> <action>

Main operations	Chain	Conditions	Actions
-A, --append: add the rule -I, --insert: insert at a specific position -D, --delete: delete the rule	INPUT OUTPUT FORWARD	-s, --source: the source IP address -d, --destination: the destination IP address -p, --protocol: the protocol to scan (tcp, udp, icmp, etc.) -sport, -dport: the source and destination ports -i, --in-interface: the incoming network interface (eth0, eth1,...) -o, --out-interface: the outgoing network interface (eth0, eth1,...)	ACCEPT: accept the network traffic DROP: drop the network traffic without notifying the sender of the deletion REJECT: drop the network traffic, notifying with ICMP the sender of the deletion

Examples on the iptables

SERVER

`sudo iptables -L`

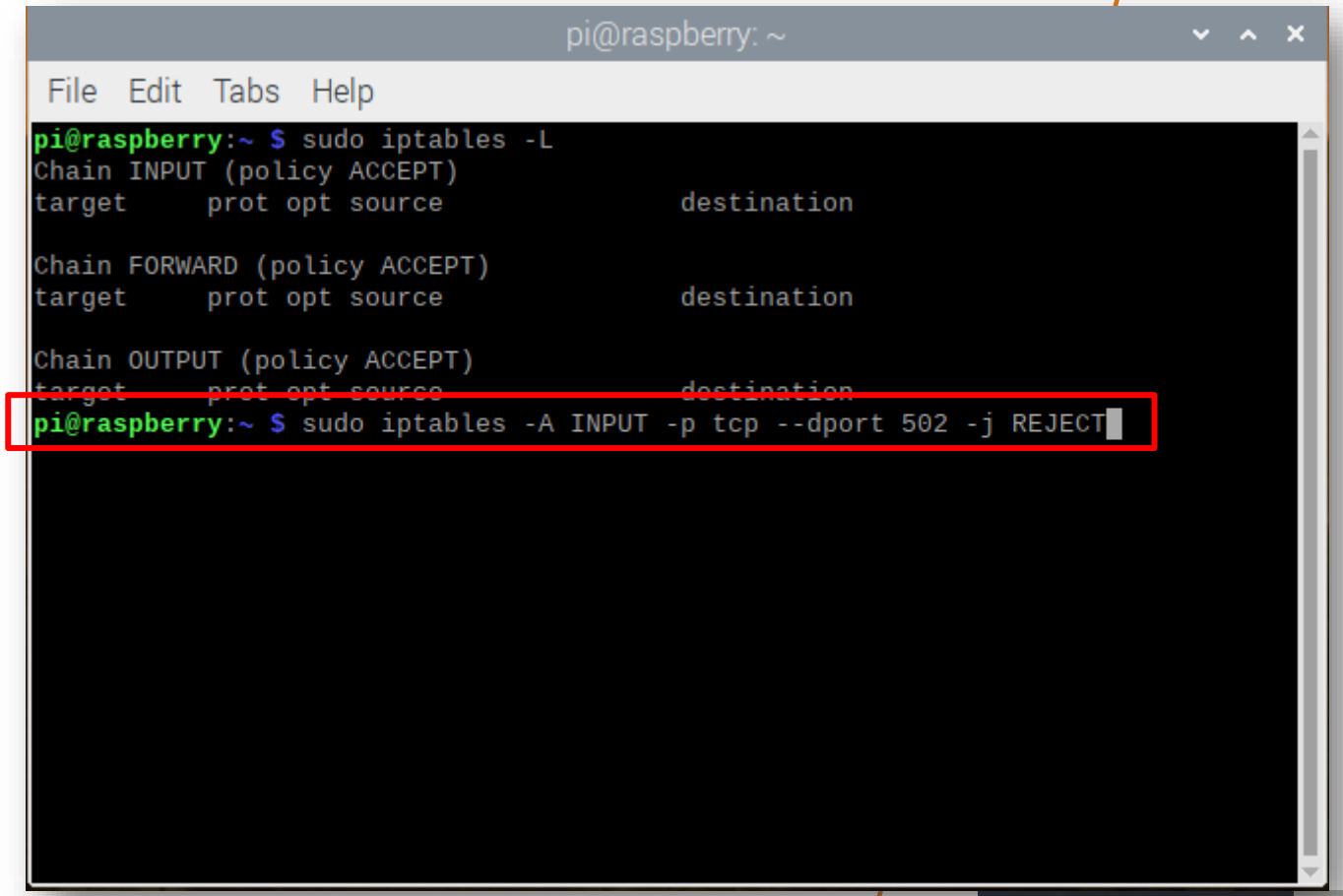


Examples on the iptables

SERVER

Drop all **incoming TCP** packets to the server on **port 502**.

```
sudo iptables -A INPUT -p tcp --dport 502 -j REJECT
```

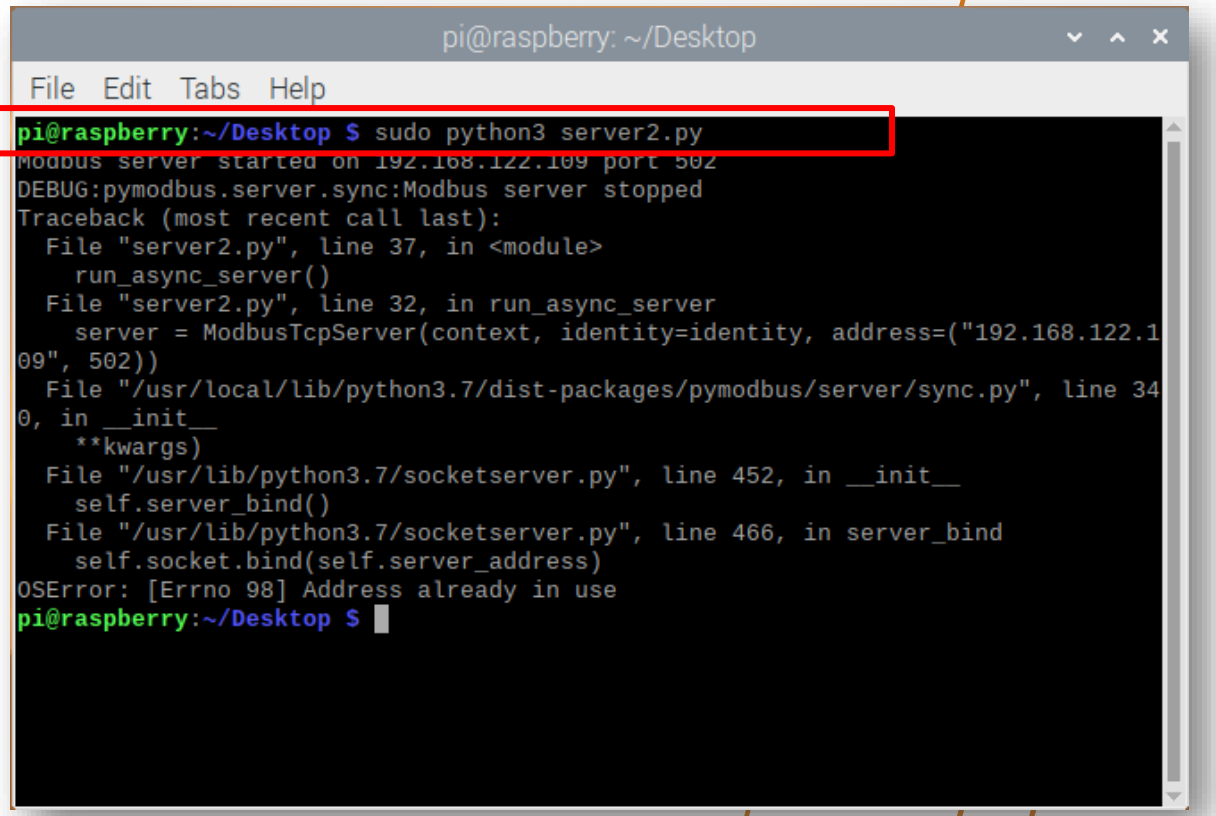


Examples on the iptables

SERVER

Drop all **incoming TCP** packets to the server on **port 502**.

Test the communication between the client and server



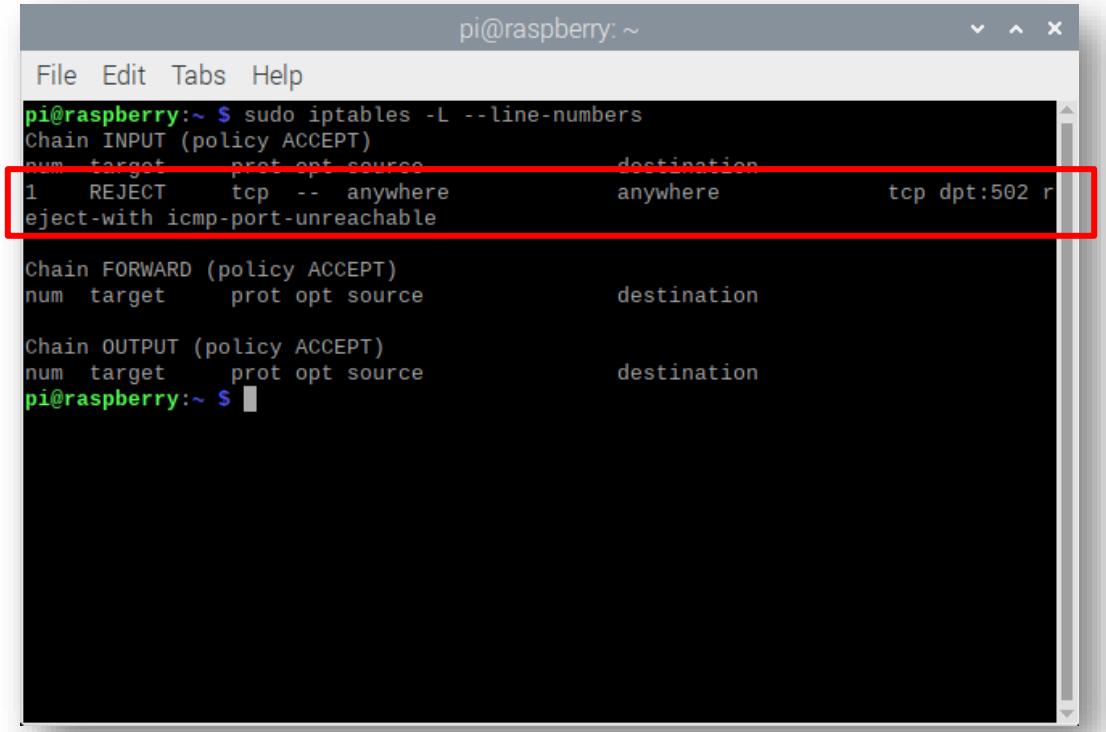
The port is already closed

Examples on the iptables

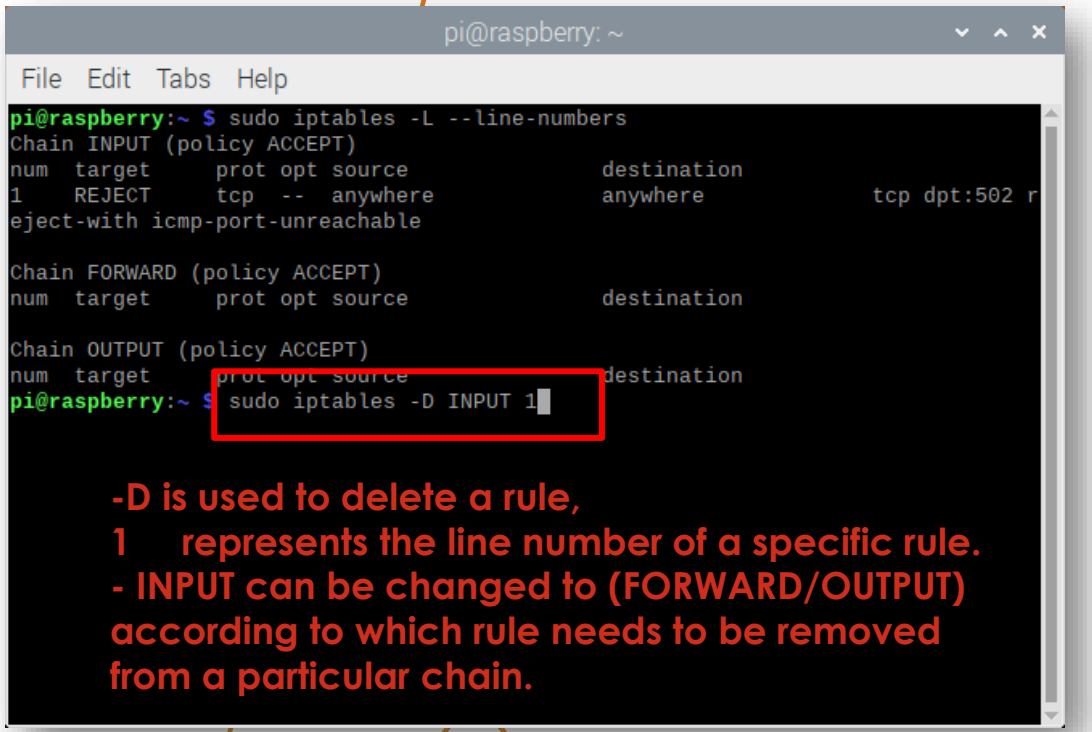
SERVER

Delete rules

`sudo iptables -L --line-numbers`



`sudo iptables -D INPUT 1`



-D is used to delete a rule,
1 represents the line number of a specific rule.
- INPUT can be changed to (FORWARD/OUTPUT) according to which rule needs to be removed from a particular chain.

Examples on the iptables

SERVER

REJECT VS **DROP** Targets

DROP: Discards the **packet without** notifying the **sender**. When the **receiver encounters** a **packet** that **matches** any **DROP rule**, the packet is **discarded**, and **no notification** is sent to the **sender**.

```
sudo iptables -A INPUT -p tcp --dport 502 -j DROP
```

Let's send a **TCP packet** from the **attacker device** using **Scapy**! Do you **remember** how **we did that?** 😊

On the Attacker Kali Linux

>> **P = IP(dst="192.168.122.109") / TCP(dport=502)**

Examples on the iptables

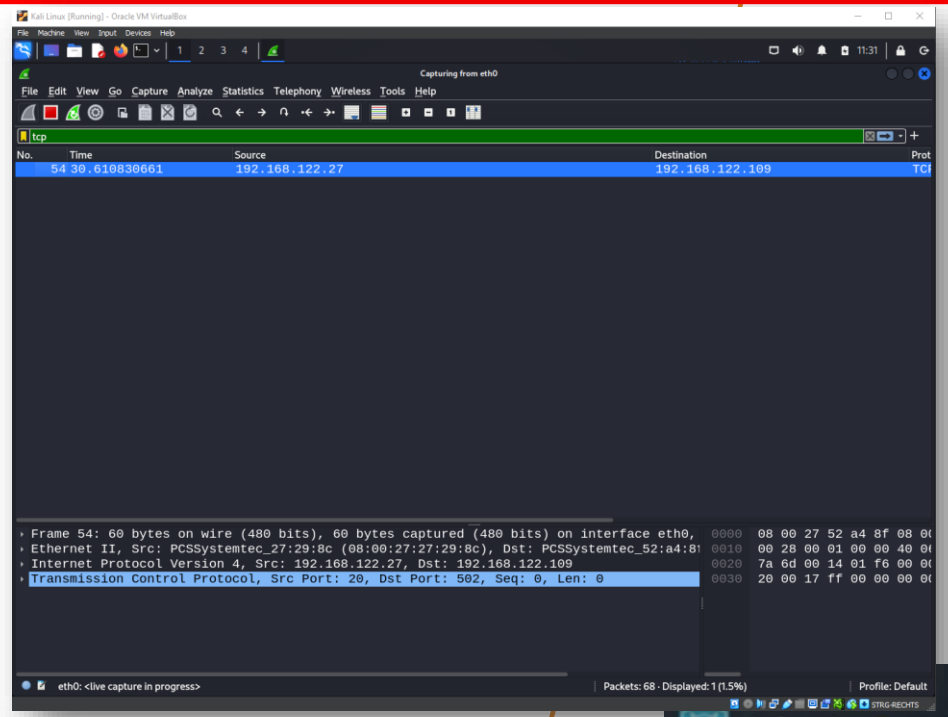
SERVER

REJECT VS **DROP** Targets

Start Wireshark on the Kali machine (Admin), and apply a filter to capture only TCP packets.

On the Attacker Kali Linux

>> **send(P)**



Only **one packet** has been **captured**. Typically, the **TCP protocol acknowledges receipt** when the **receiver** successfully **receives** a **TCP** packet and **responds** with a corresponding **acknowledgment**.

That is **not happening here**, which means that the **packet** has been **dropped** from the **receiver** (i.e., **server**), **and no notification is sent to the sender**

Examples on the iptables

SERVER

REJECT VS **DROP** Targets

REJECT: Discards the **packet with** notifying the **sender**. When the **receiver encounters** a **packet** that **matches** any **REJECT rule**, the packet is **discarded**, and **a notification** is sent to the **sender**.

```
sudo iptables -A INPUT -p tcp --dport 502 -j REJECT
```

Don't forget to delete the DROP rule ☺

Again, send a **TCP packet** from the **attacker device** using **Scapy**

On the Attacker Kali Linux

```
>> P = IP(dst="192.168.122.109") / TCP(dport=502)
```

Examples on the iptables

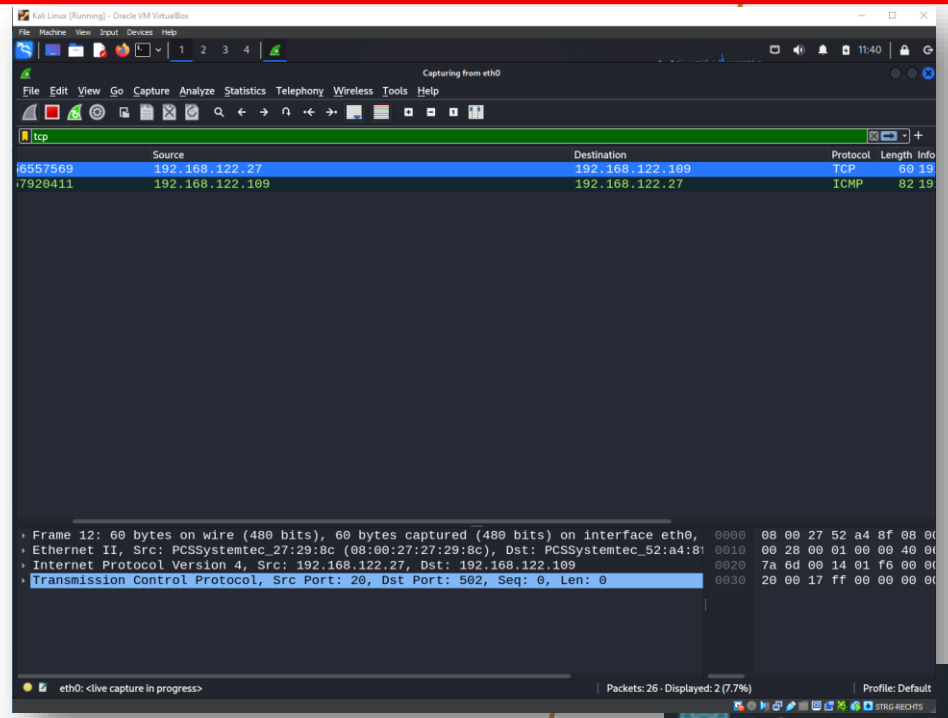
SERVER

REJECT VS **DROP** Targets

Start Wireshark on the Kali machine (Admin) and apply a filter to capture only TCP packets.

On the Attacker Kali Linux

>> **send(P)**



The TCP packet was **received** by the **server**, but instead of a **TCP acknowledgment** (e.g., from the server to the attacker), a notification was **sent** from the **server** to the **attacker** using **ICMP**, not **TCP**.

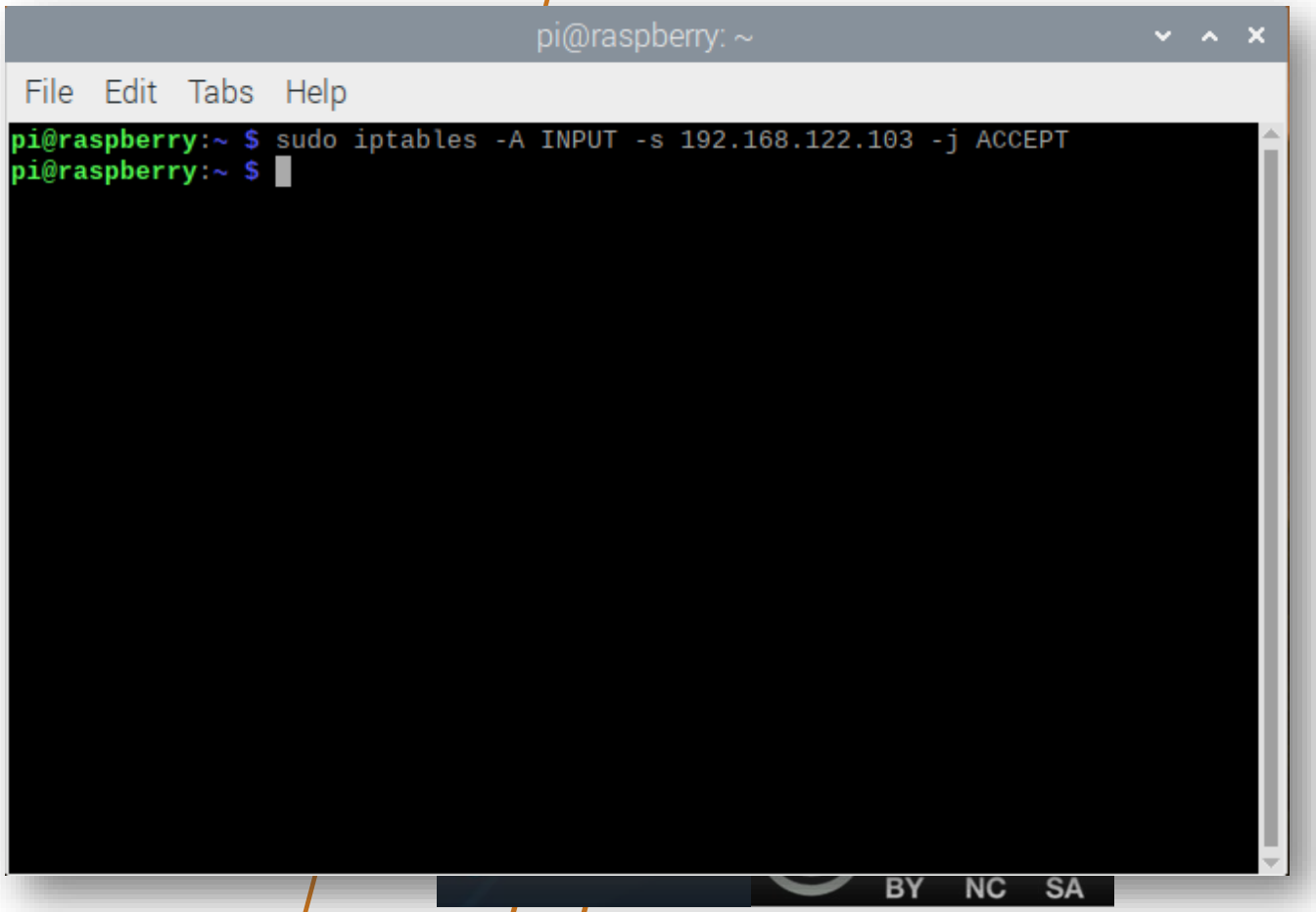
Examples on the iptables

SERVER

Allow all traffic from my authorized devices (only) in the network

1. **Allow** communication with **the specific IP**:

```
sudo iptables -A INPUT -s 192.168.122.103 -j ACCEPT
```



This **means** that I am **allowing** the **server** to accept **all packets** from the **client device** with the **IP ending** in **.103**.

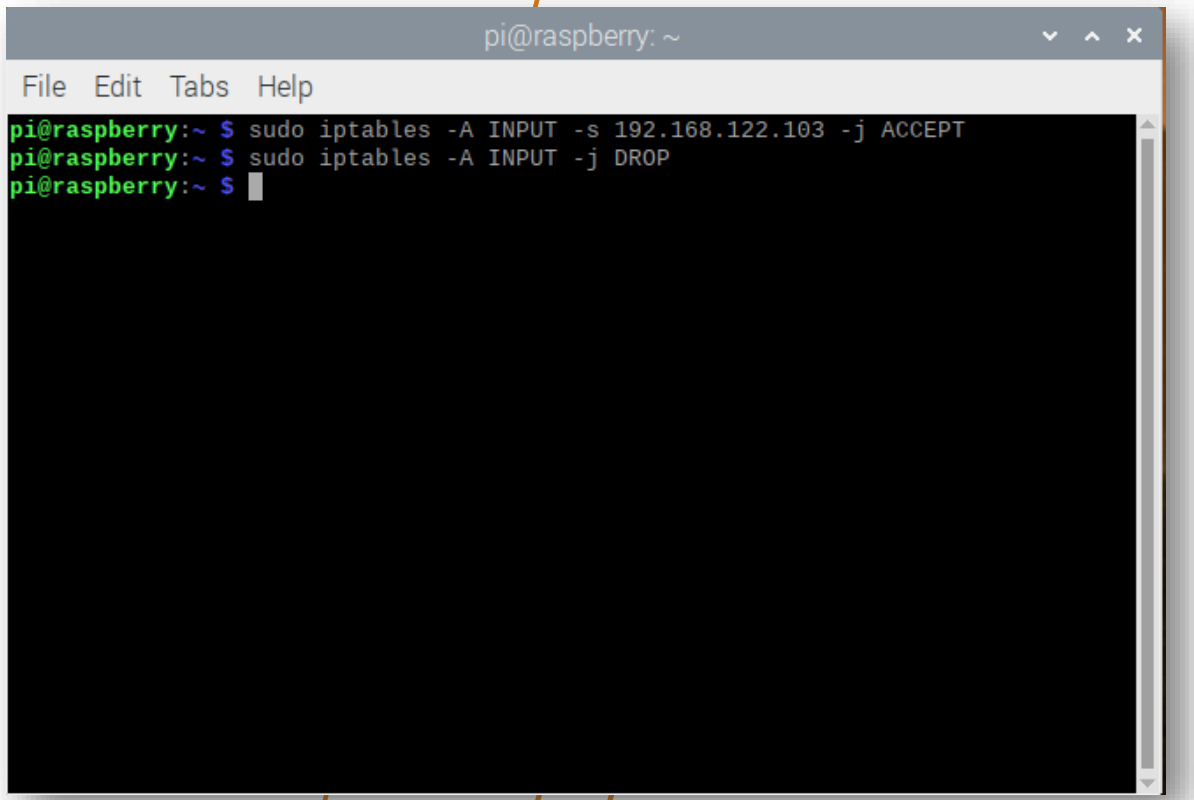
Examples on the iptables

SERVER

Allow all traffic from my authorized devices (only) in the network

2. **Block** all other **incoming** traffic:

```
sudo iptables -A INPUT -j DROP
```



This means that I am **dropping** all **incoming** packets.

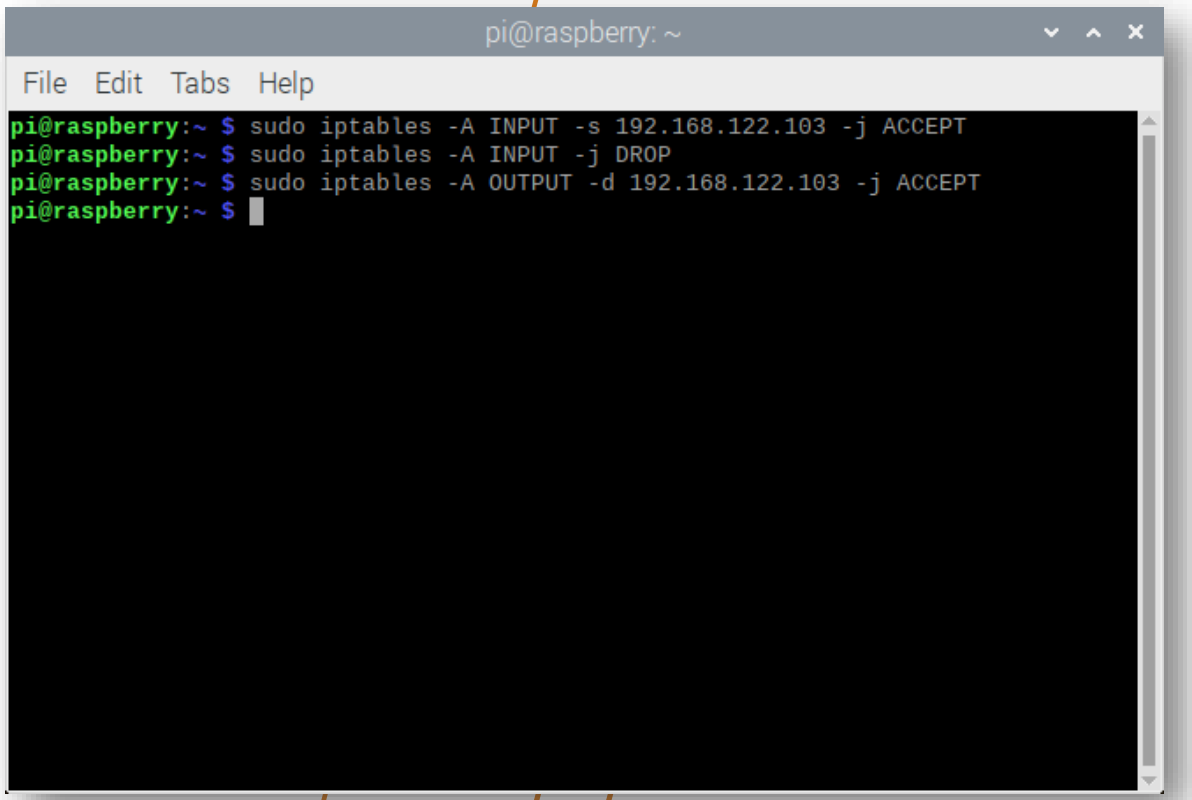
Examples on the iptables

SERVER

Allow all traffic from my authorized devices (only) in the network

3. **Allow outgoing** traffic to the **specific IP**:

```
sudo iptables -A OUTPUT -d 192.168.122.103 -j ACCEPT
```



This means that I am **allowing** all **outgoing packets** to the **client device** with the IP ending in **.103.**

The server is now configured to communicate only with the client device.

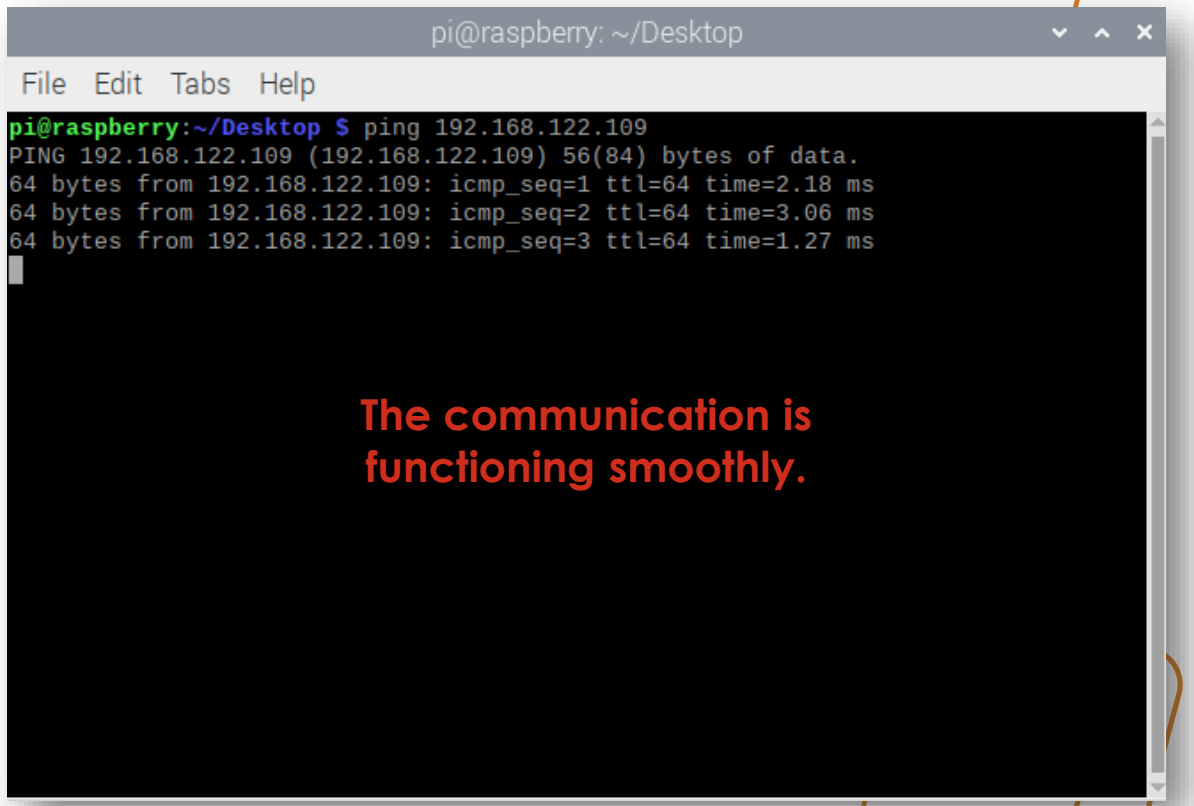
Let's TEST ;)

Examples on the iptables

SERVER

Allow all traffic from my authorized devices (only) in the network

Test the **communication** between the **client** and the **server** by using the **ping** command from the **client**.

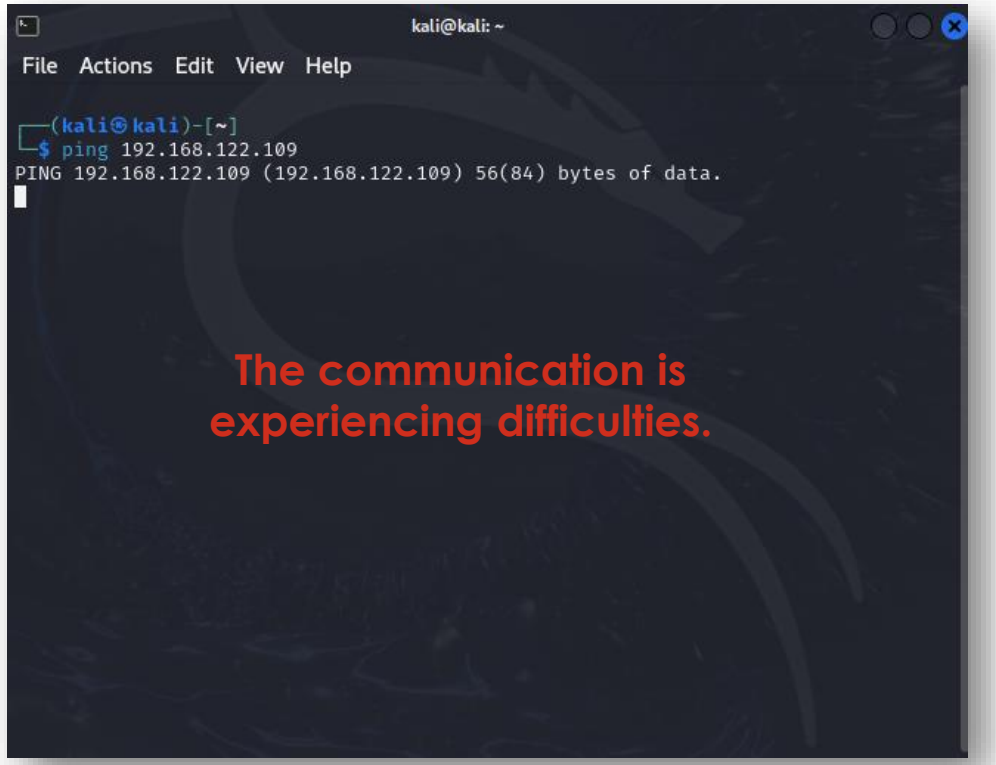


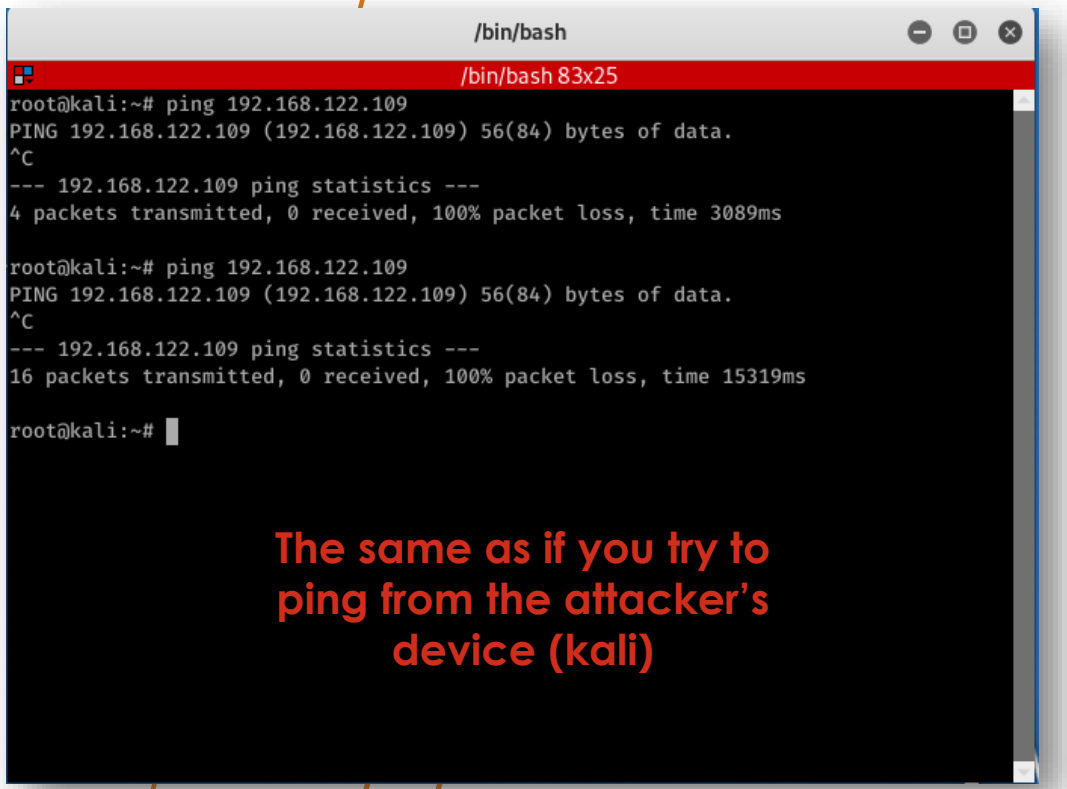
Examples on the iptables

SERVER

Allow all traffic from my authorized devices (only) in the network

Test the **communication** between the kali (**friendly admin**) and the **server** by using the ping command from the client.



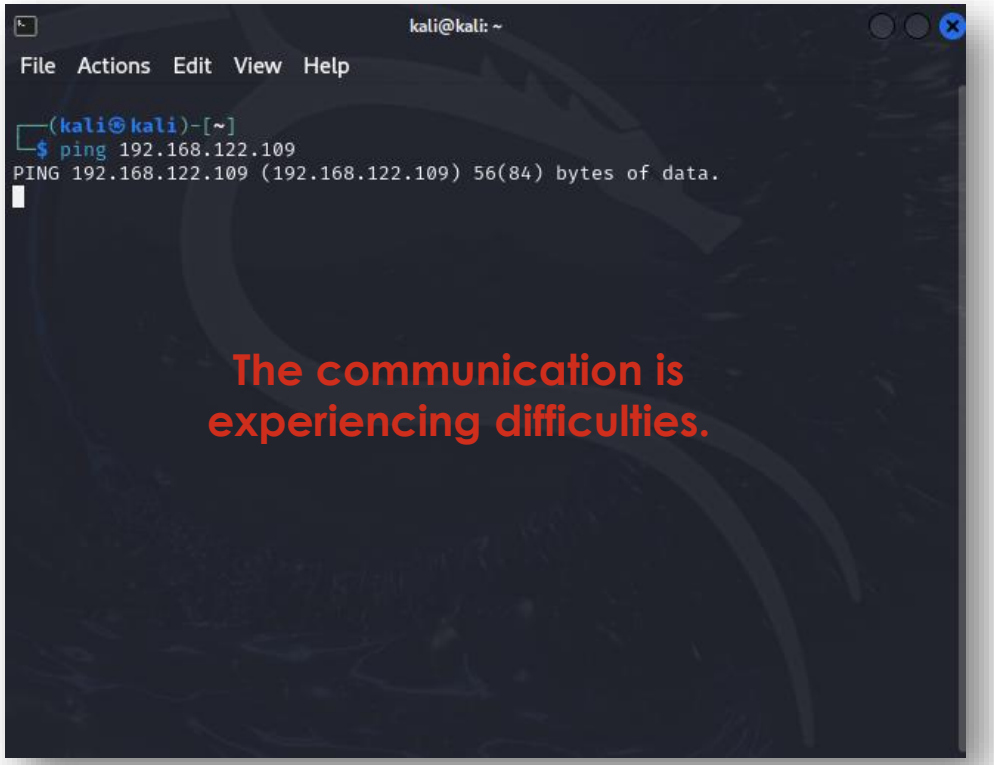


Examples on the iptables

SERVER

Allow all traffic from my authorized devices (only) in the network

Test the **communication** between the kali (**friendly admin**) and the **server** by using the ping command from the client.



Furthermore, it is **essential** to **add new rules** to the **server** to ensure **legitimate communication** with all **authorized devices** within the **network**. To prevent **unauthorized communication** with the **server.**

So, repeat the **previous** steps for **all authorized IPs** in the **network**.

Examples on the iptables

SERVER

Log dropped packets

1. **Use** the **LOG target** and **add a message prefix**:

```
sudo iptables -A INPUT -j LOG --log-prefix "Dropped: "
```

2. **Add** a rule to **drop** packets **after** logging:

```
sudo iptables -A INPUT -j DROP
```

3. To check **logs**, use the **dmesg command** to view system logs and **grep to filter the output**:

```
sudo dmesg | grep "Dropped"
```

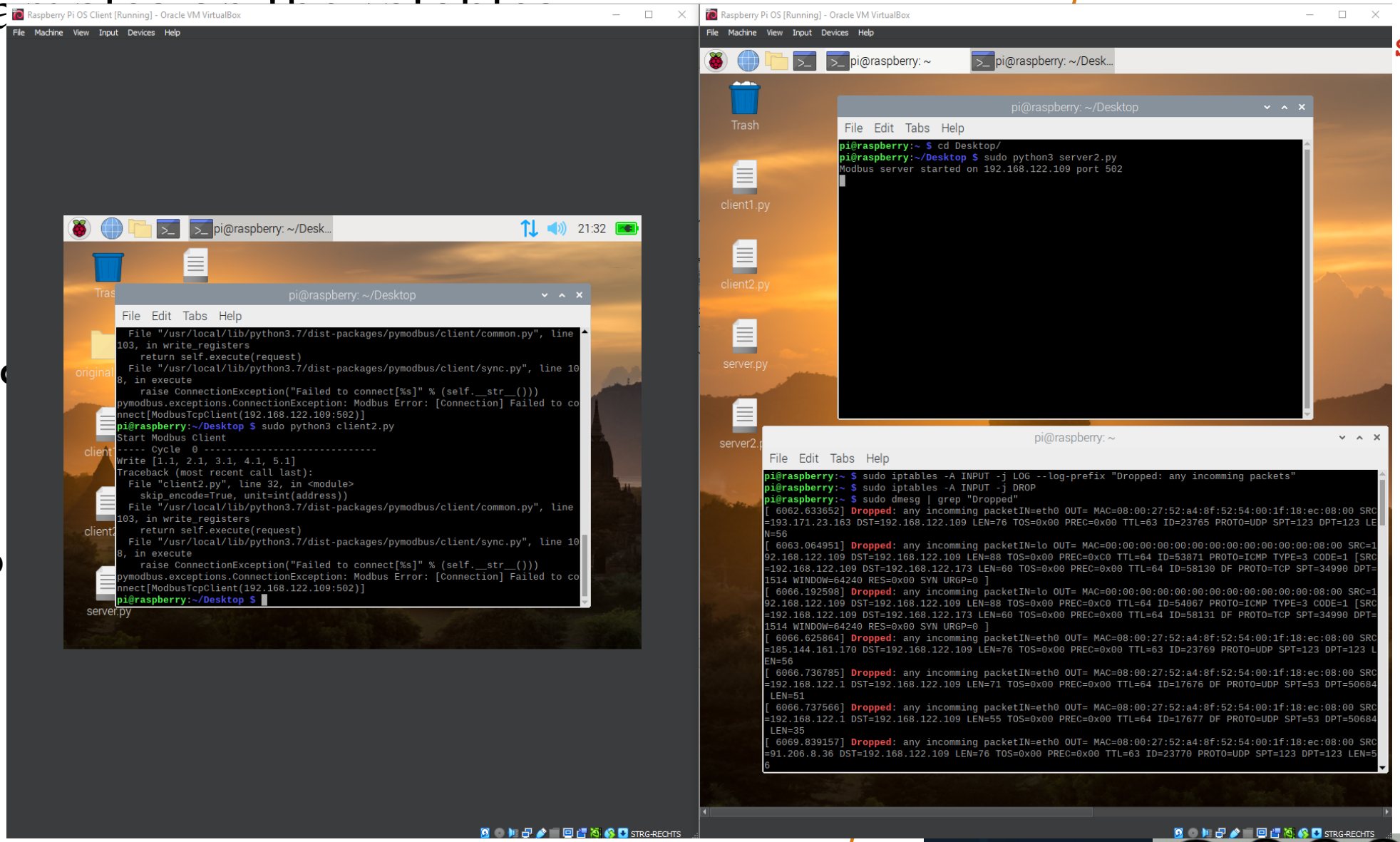
Exa

SERVER

1. Use

2. Ad

3. To



Iptables Tutorial: Ultimate Guide to Linux Firewall (phoenixnap.com)

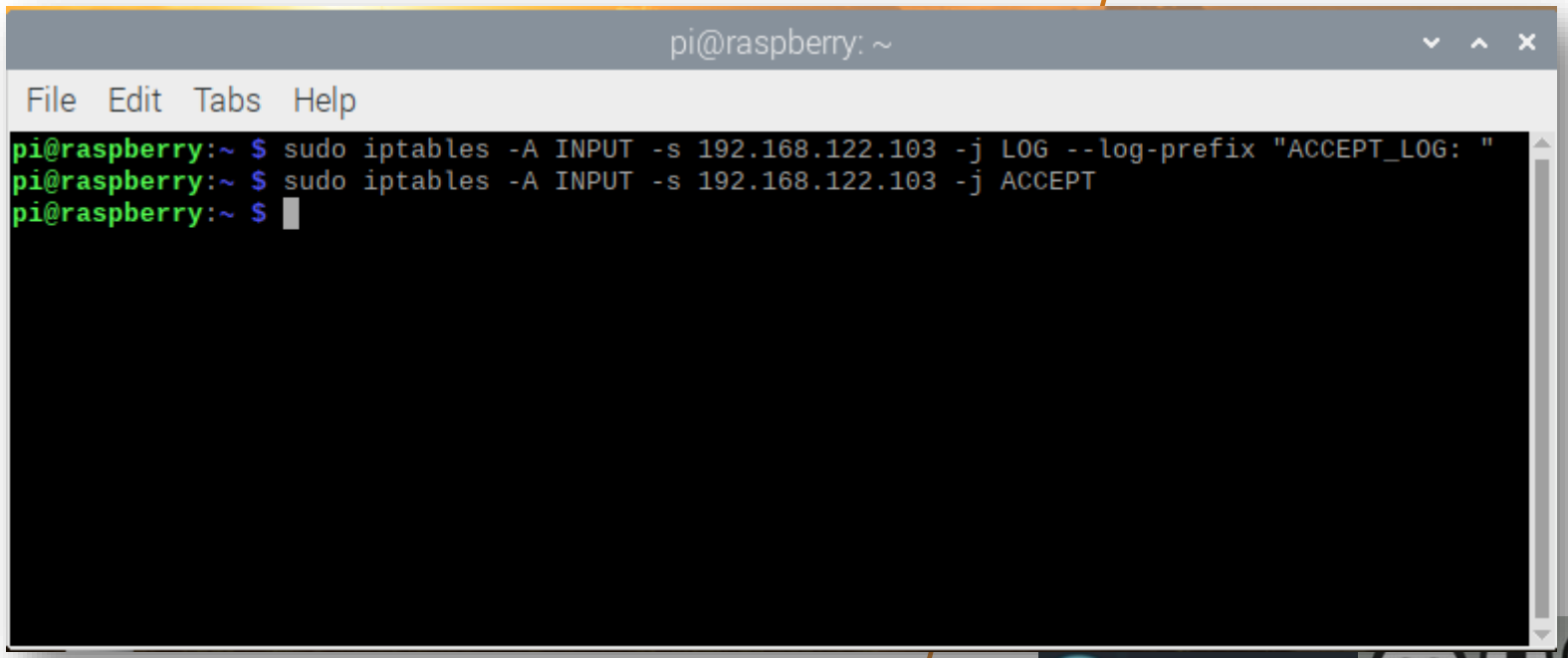
Examples on the iptables

SERVER

Log packets based on the example for accepting only packets from legitimate IPs.

1. Log **packets** from **legitimate IPs** before **accepting them**:

```
sudo iptables -A INPUT -s 192.168.122.103 -j LOG --log-prefix "ACCEPT_LOG: "

sudo iptables -A INPUT -s 192.168.122.103 -j ACCEPT
```



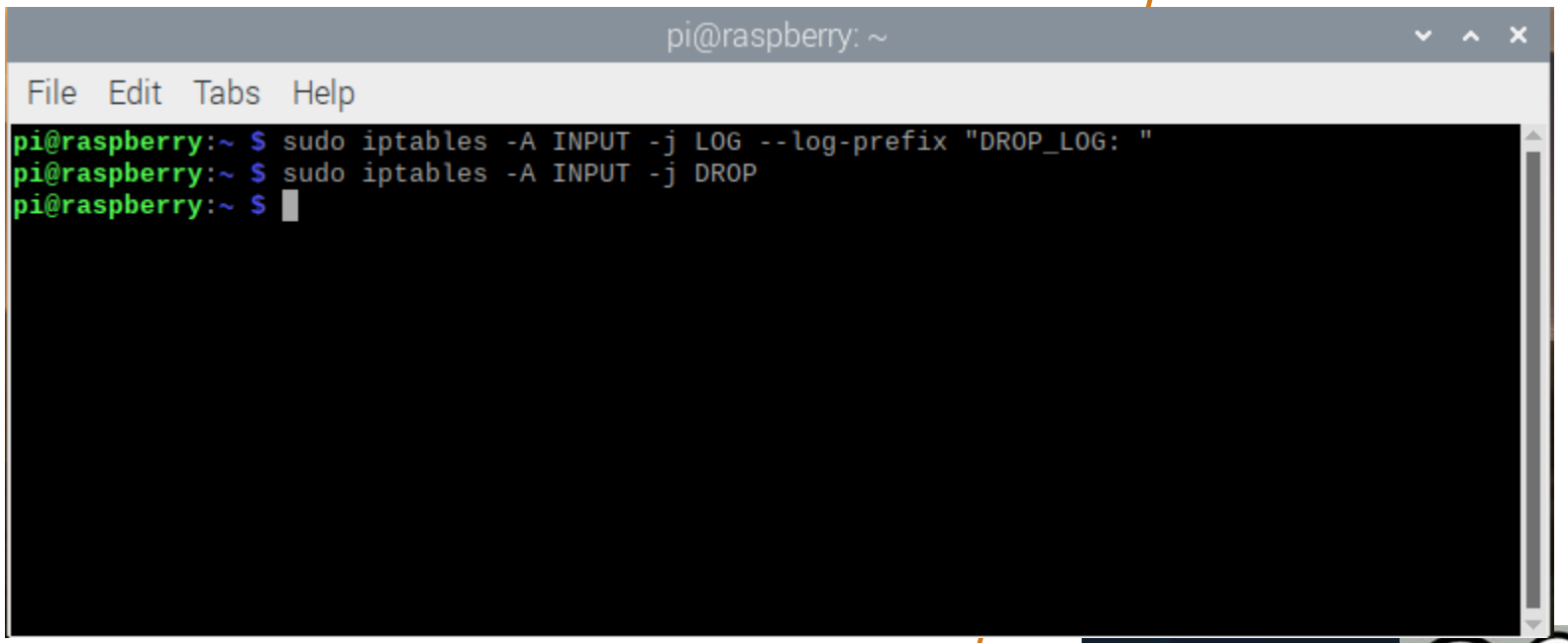
Examples on the iptables

SERVER

Log packets based on the example for accepting only packets from legitimate IPs.

2. Log **all other** traffic **before** dropping it:

```
sudo iptables -A INPUT -j LOG --log-prefix "DROP_LOG: "

sudo iptables -A INPUT -j DROP
```



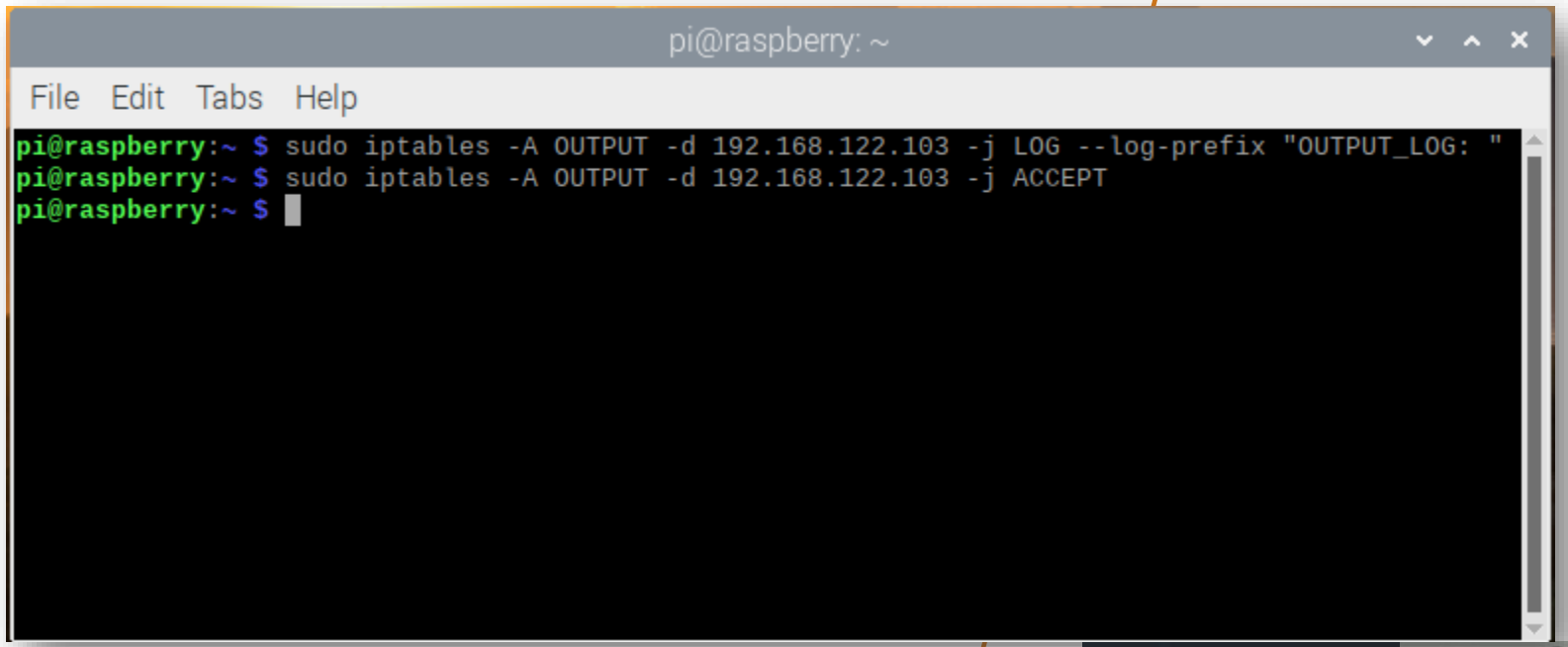
Examples on the iptables

SERVER

Log packets based on the example for accepting only packets from legitimate IPs.

2. Log **outgoing** packets to **a specific IP**:

```
sudo iptables -A OUTPUT -d 192.168.122.103 -j LOG --log-prefix "OUTPUT_LOG: "

sudo iptables -A OUTPUT -d 192.168.122.103 -j ACCEPT
```



Examples on the iptables

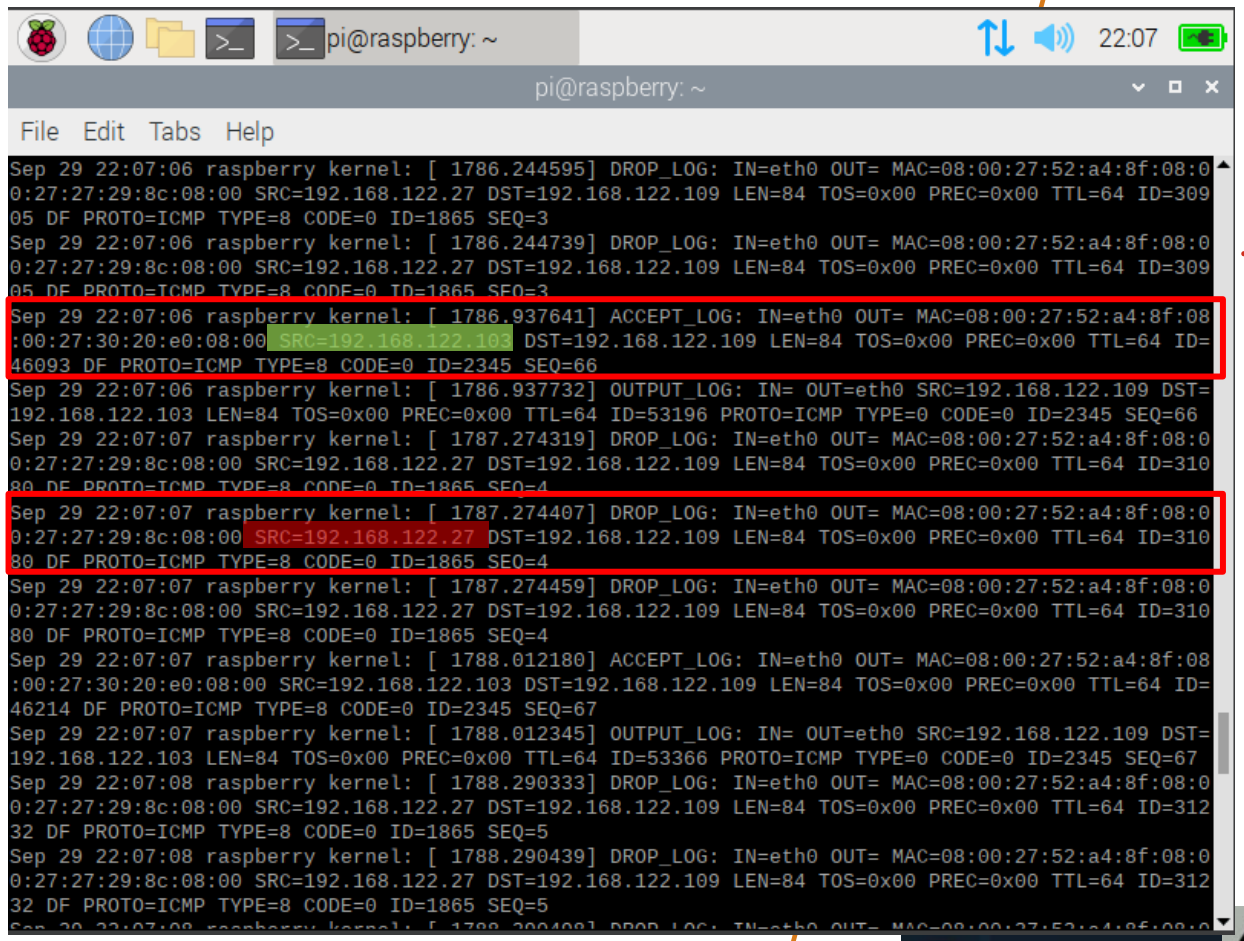
SERVER

Log packets based on the example for accepting only packets from legitimate IPs.

TEST logging:

`sudo tail -f /var/log/syslog`

Ping from the **client** with the **IP ending** in **.103** and from the **attacker** (**Kali**) with the IP ending in **.27.**



Accept packets from the client device.

Drop packets from the atatcker device.

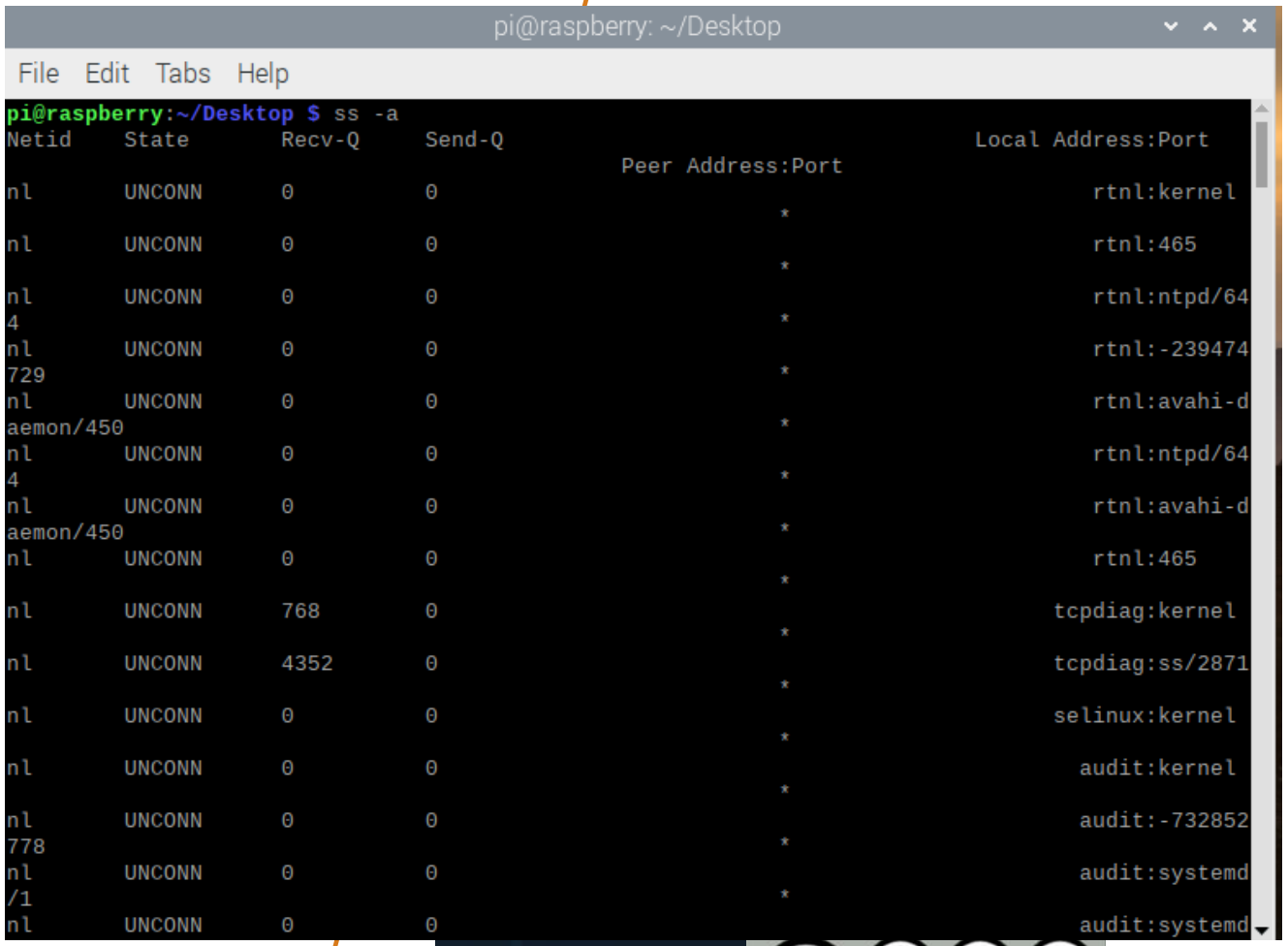
Investigation Tools

ss

Examples on SS

SS is used to dump socket statistics. It allows showing information similar to netstat. It can display more TCP and state information than other tools.

```
ss -a
```



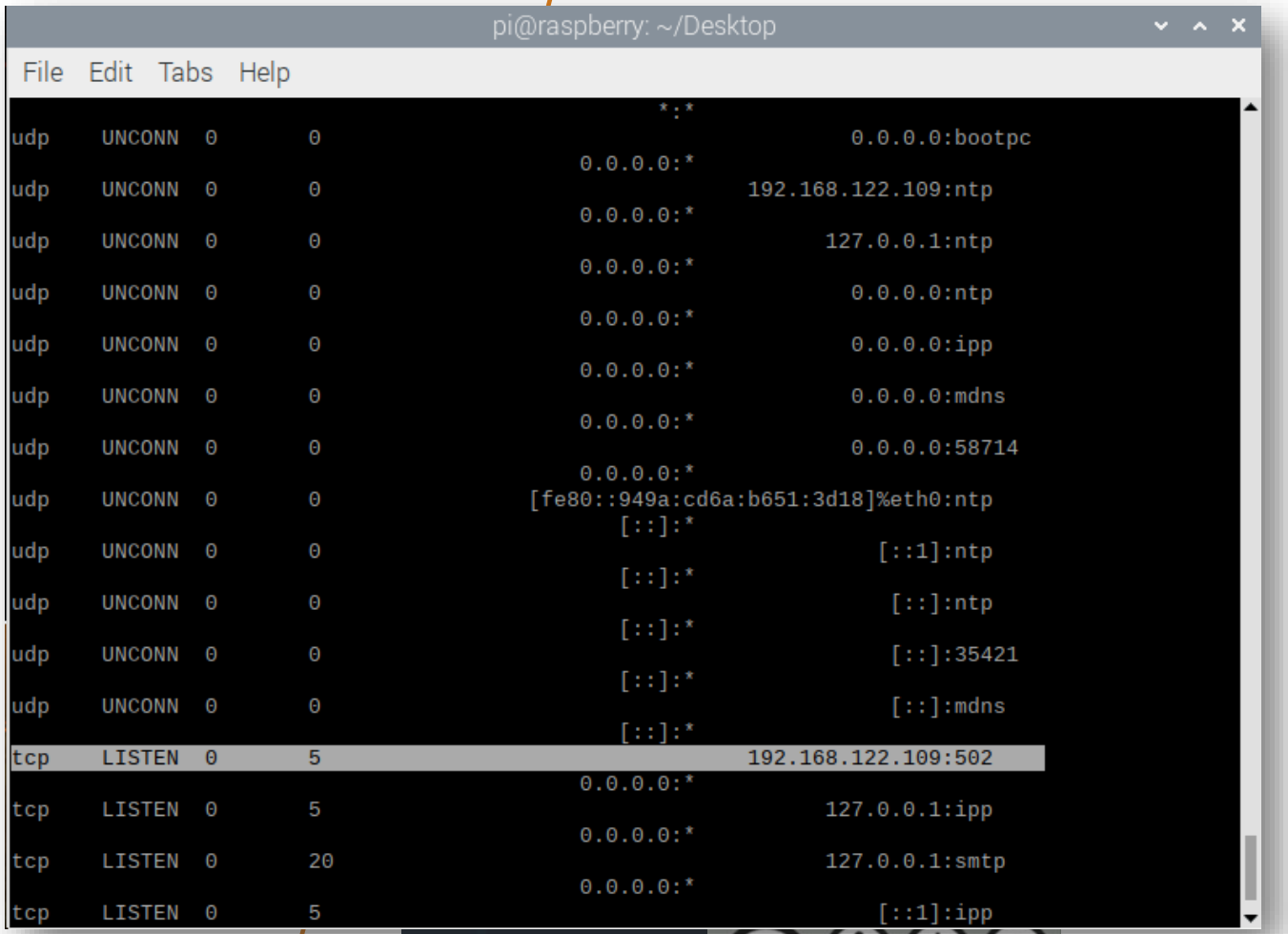
ss(8) - Linux manual page (man7.org)

Examples on SS

SS is used to dump socket statistics. It allows showing information similar to netstat. It can display more TCP and state information than other tools.

```
ss -a
```

If any port is open, the tool will detect it. For example, the port 502 is open, the tool will detect it as well



ss(8) - Linux manual page (man7.org)

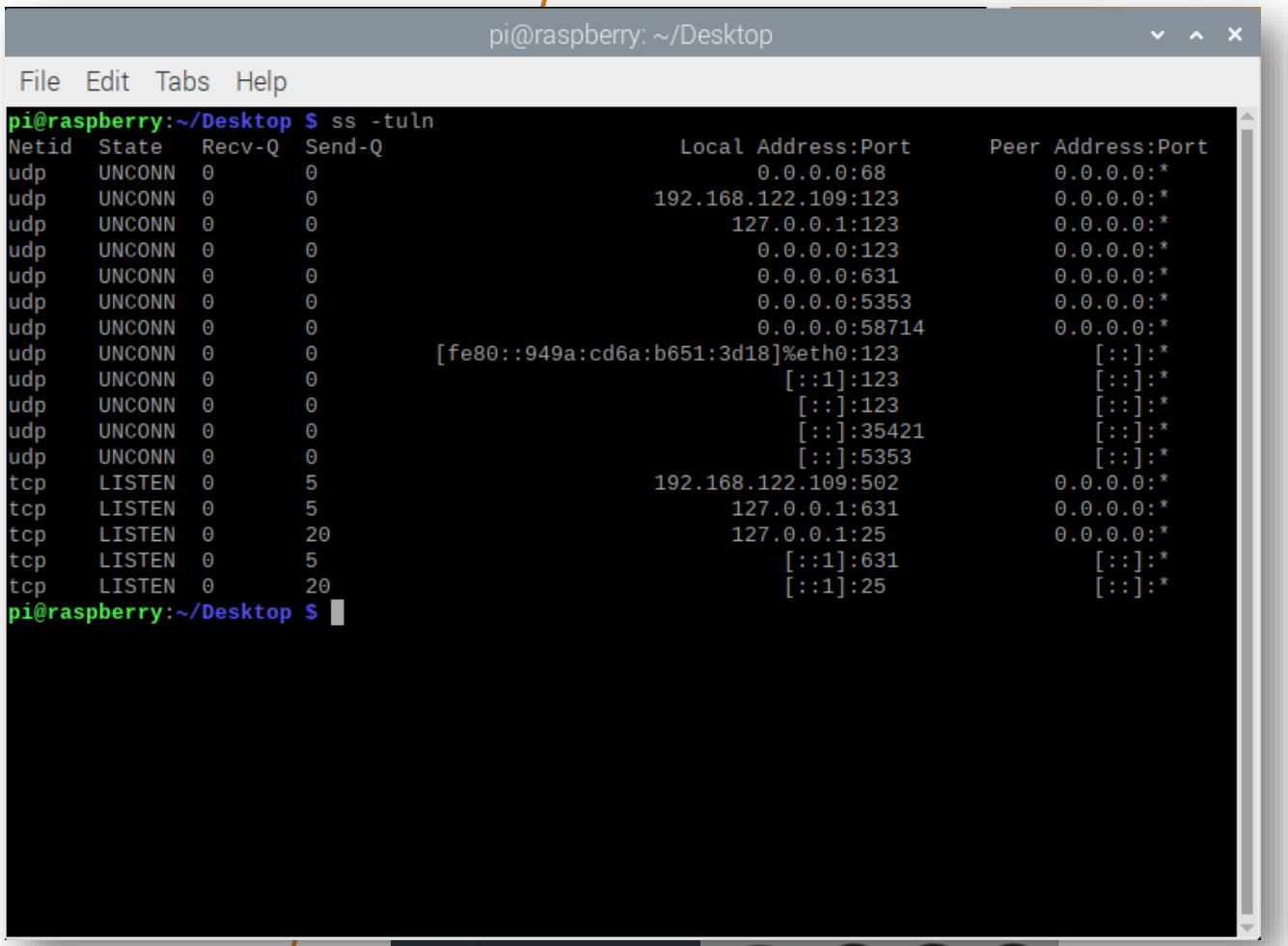
Examples on SS

SS is used to dump socket statistics. It allows showing information similar to netstat. It can display more TCP and state information than other tools.

List All TCP/UDP Sockets:

```
ss -tuln
```

```
-t: Display TCP sockets.
-u: Display UDP sockets.
-l: Show listening sockets.
-n: Show numerical addresses instead of resolving hostnames.
```



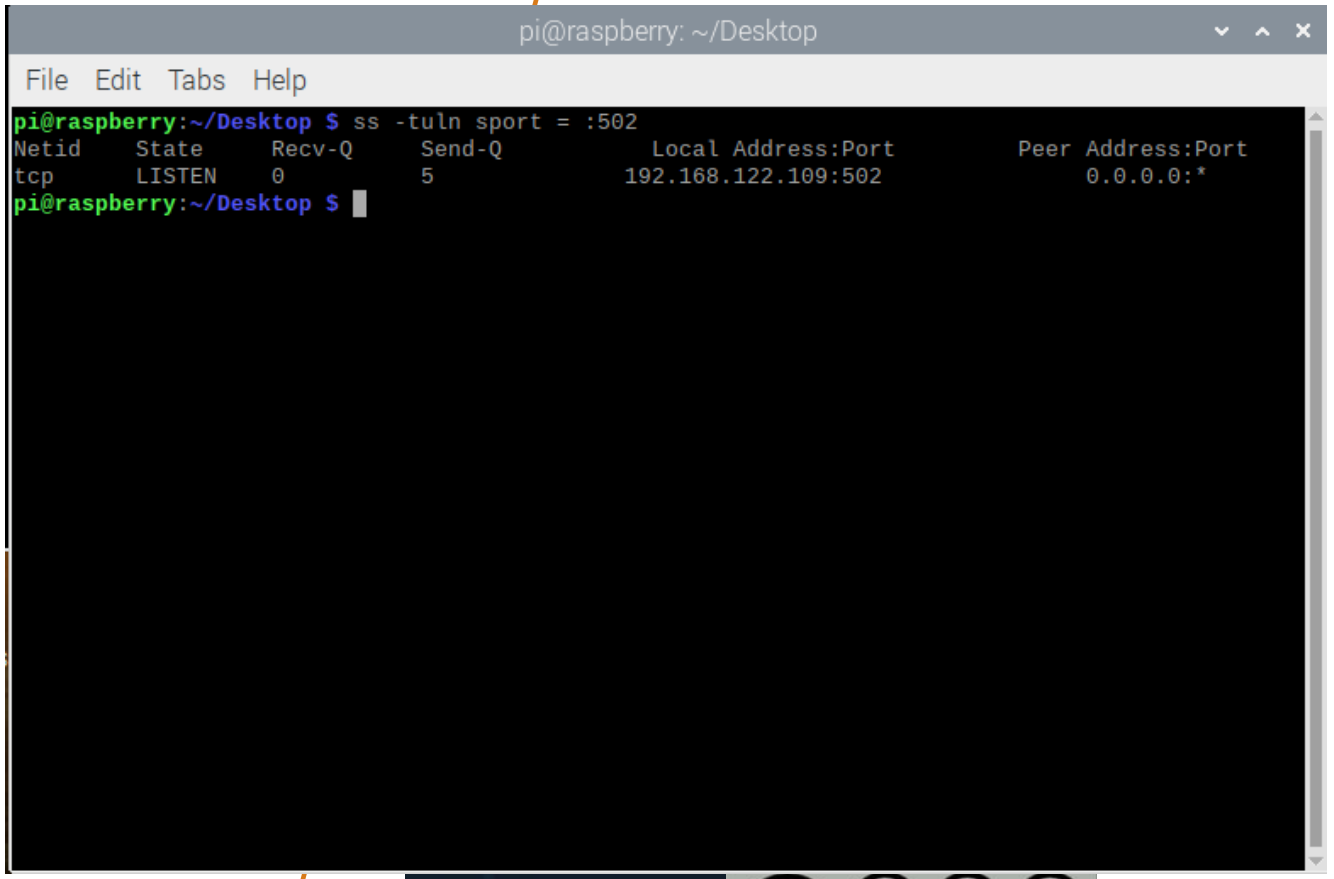
ss(8) - Linux manual page (man7.org)

Examples on SS

SS is used to dump socket statistics. It allows showing information similar to netstat. It can display more TCP and state information than other tools.

Filter by a Specific Port:

```
ss -tuln sport = :502
```

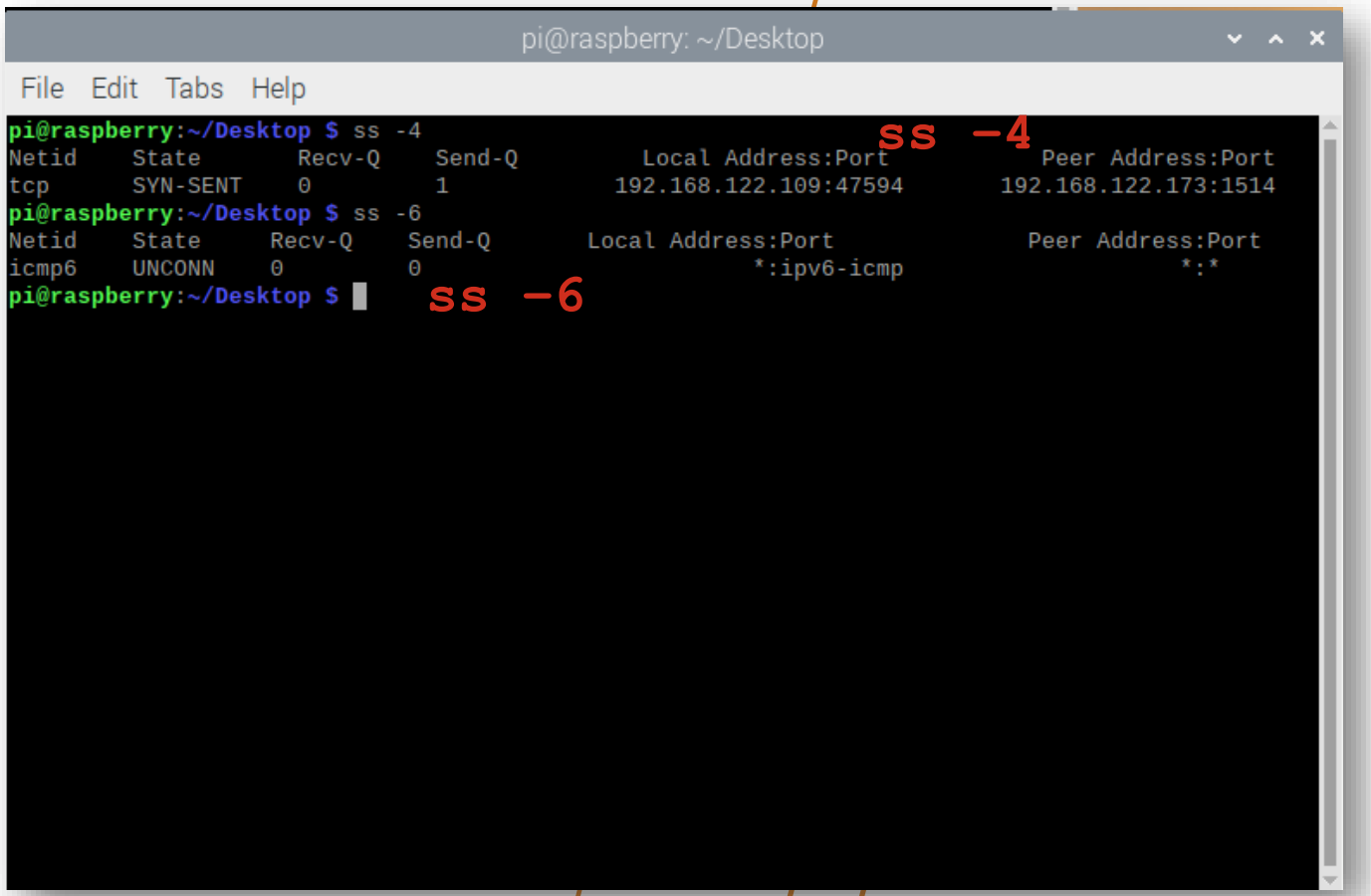


ss(8) - Linux manual page (man7.org)

Examples on SS

SS is used to dump socket statistics. It allows showing information similar to netstat. It can display more TCP and state information than other tools.

Show IPv4 or IPv6 Sockets:



ss(8) - Linux manual page (man7.org)

Investigation Tools

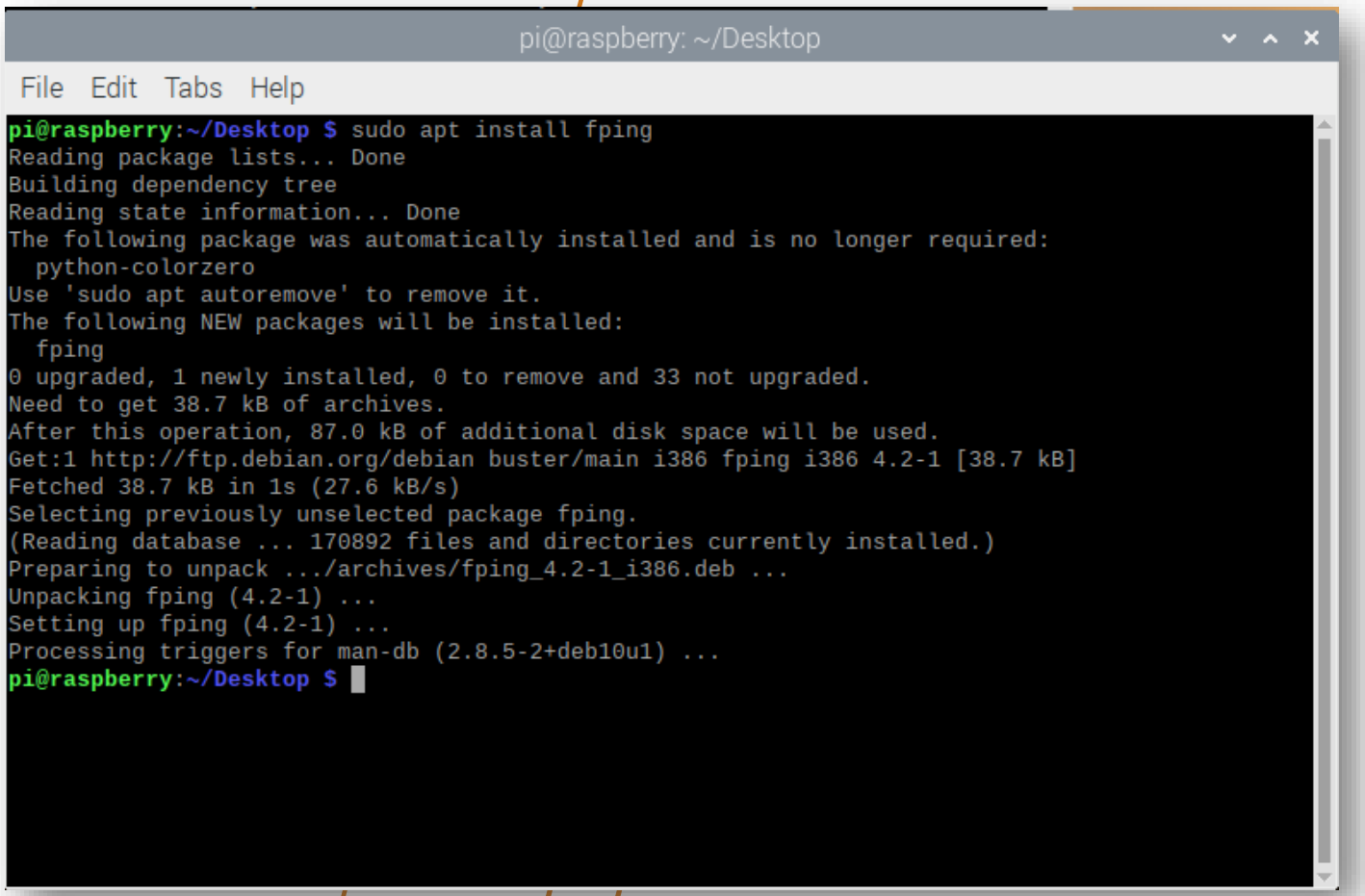
fping

Examples on fping

fping is a program to send ICMP echo probes to network hosts, similar to ping, but much better performing when pinging multiple hosts.

You may first need to install the fing on your linux

Sudo apt-get install fping



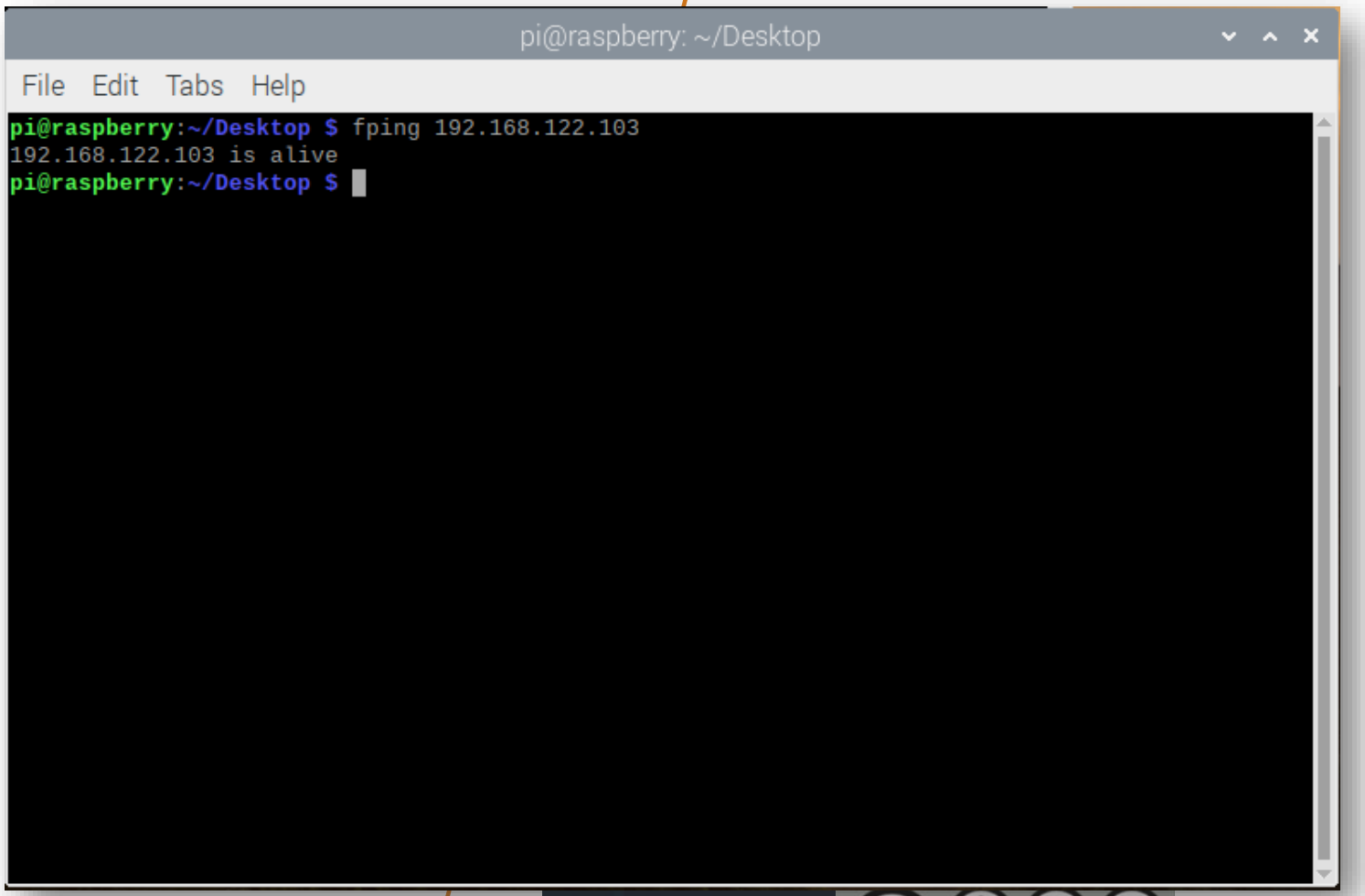
fping Homepage

Examples on fping

fping is a program to send ICMP echo probes to network hosts, similar to ping, but much better performing when pinging multiple hosts.

Ping a Single Host:

```
fping 192.168.122.103
```



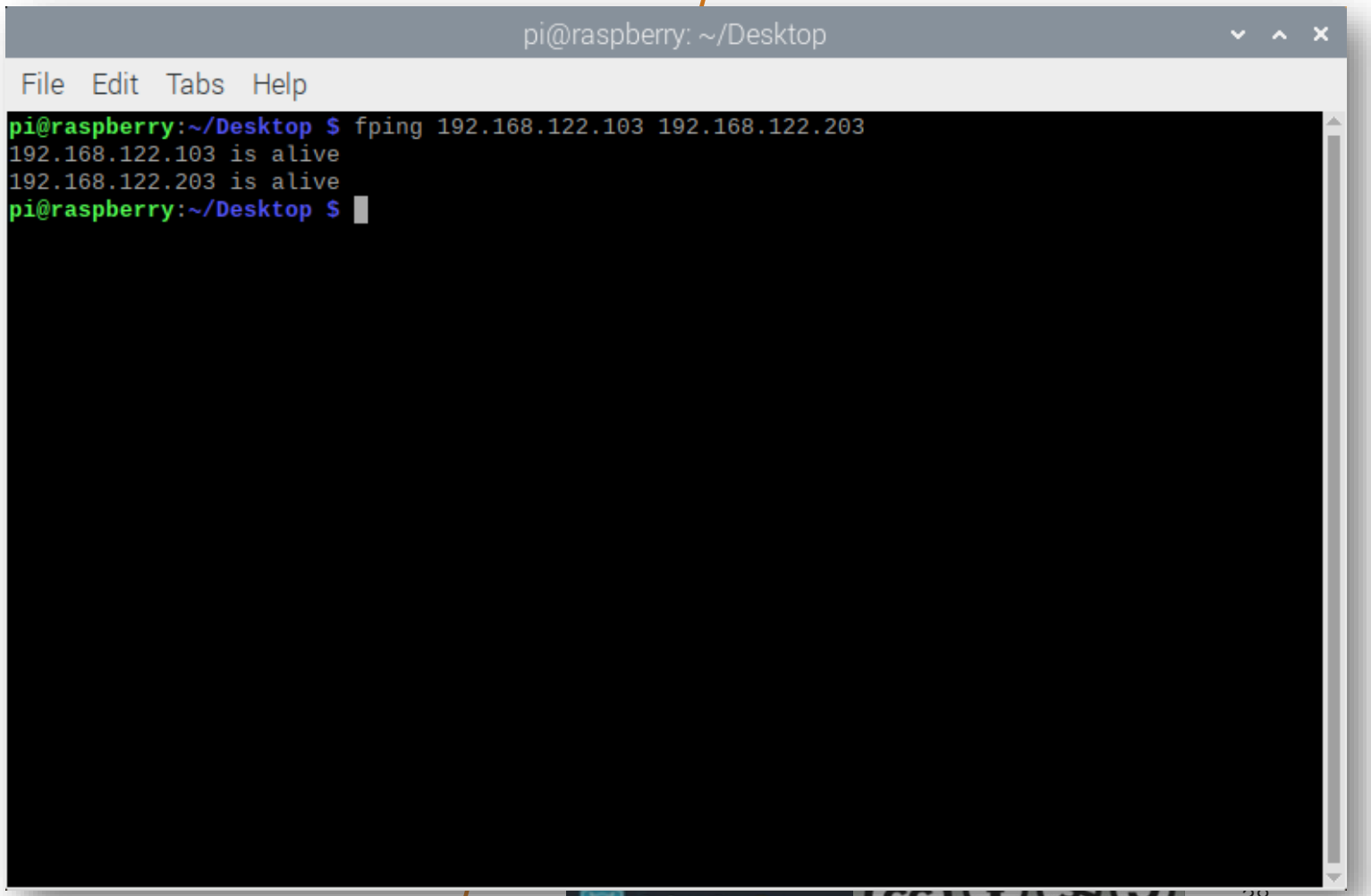
Examples on fping

fping is a program to send ICMP echo probes to network hosts, similar to ping, but much better performing when pinging multiple hosts.

Ping multiple hosts

```
fping 192.168.122.103
   192.168.122.203
```

I am attempting to ping both the client and the friendly Kali Linux.

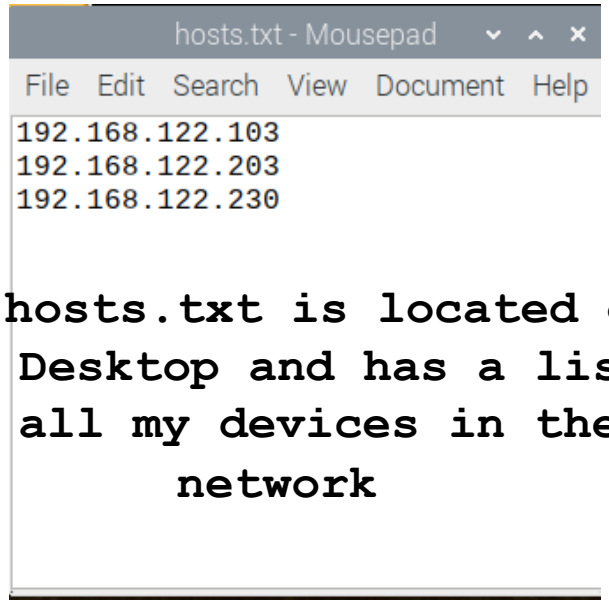


Examples on fping

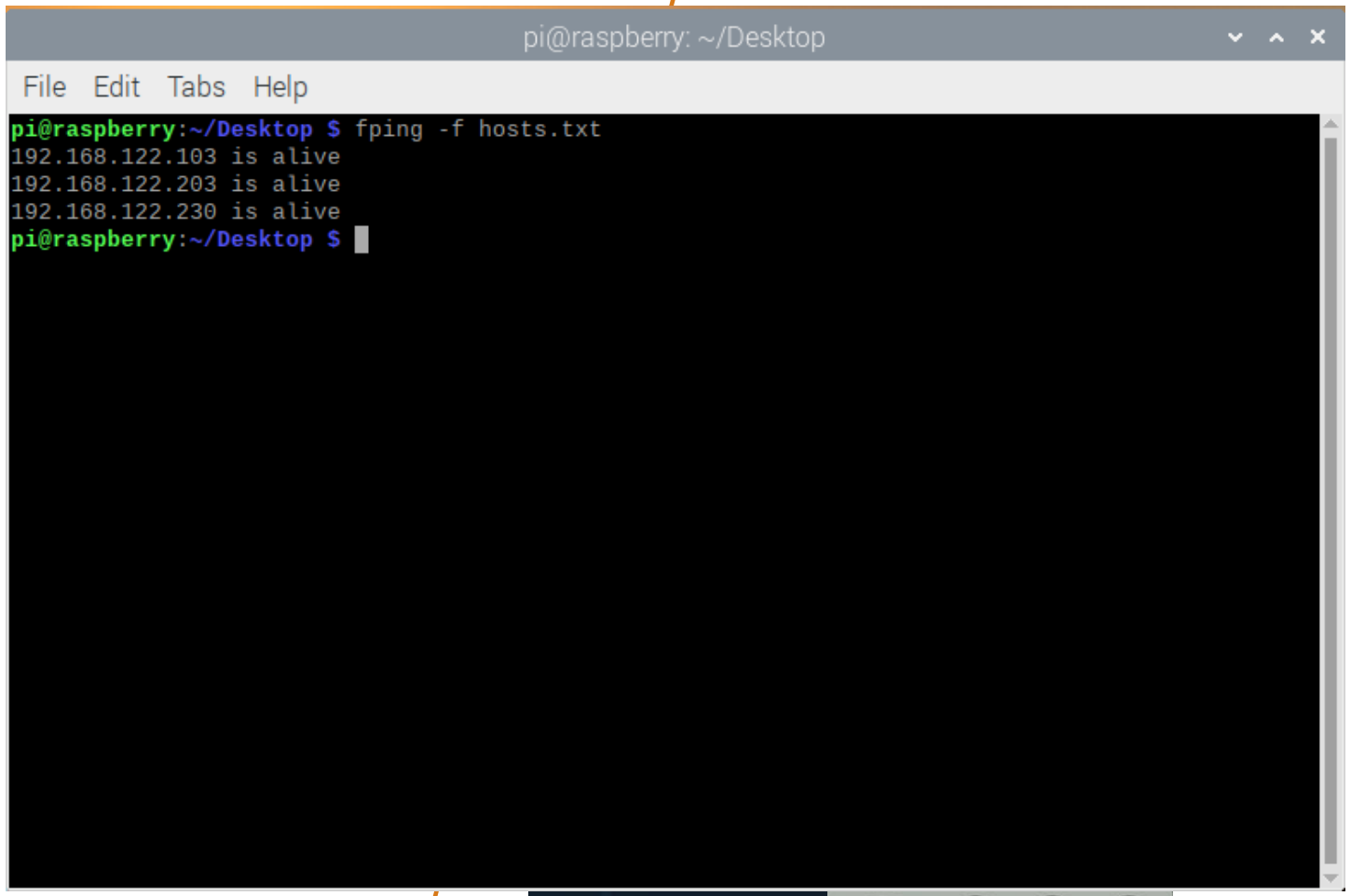
fping is a program to send ICMP echo probes to network hosts, similar to ping, but much better performing when pinging multiple hosts.

Ping multiple hosts from a file

```
fping -f hosts.txt
```



The hosts.txt is located on the Desktop and has a list of all my devices in the network



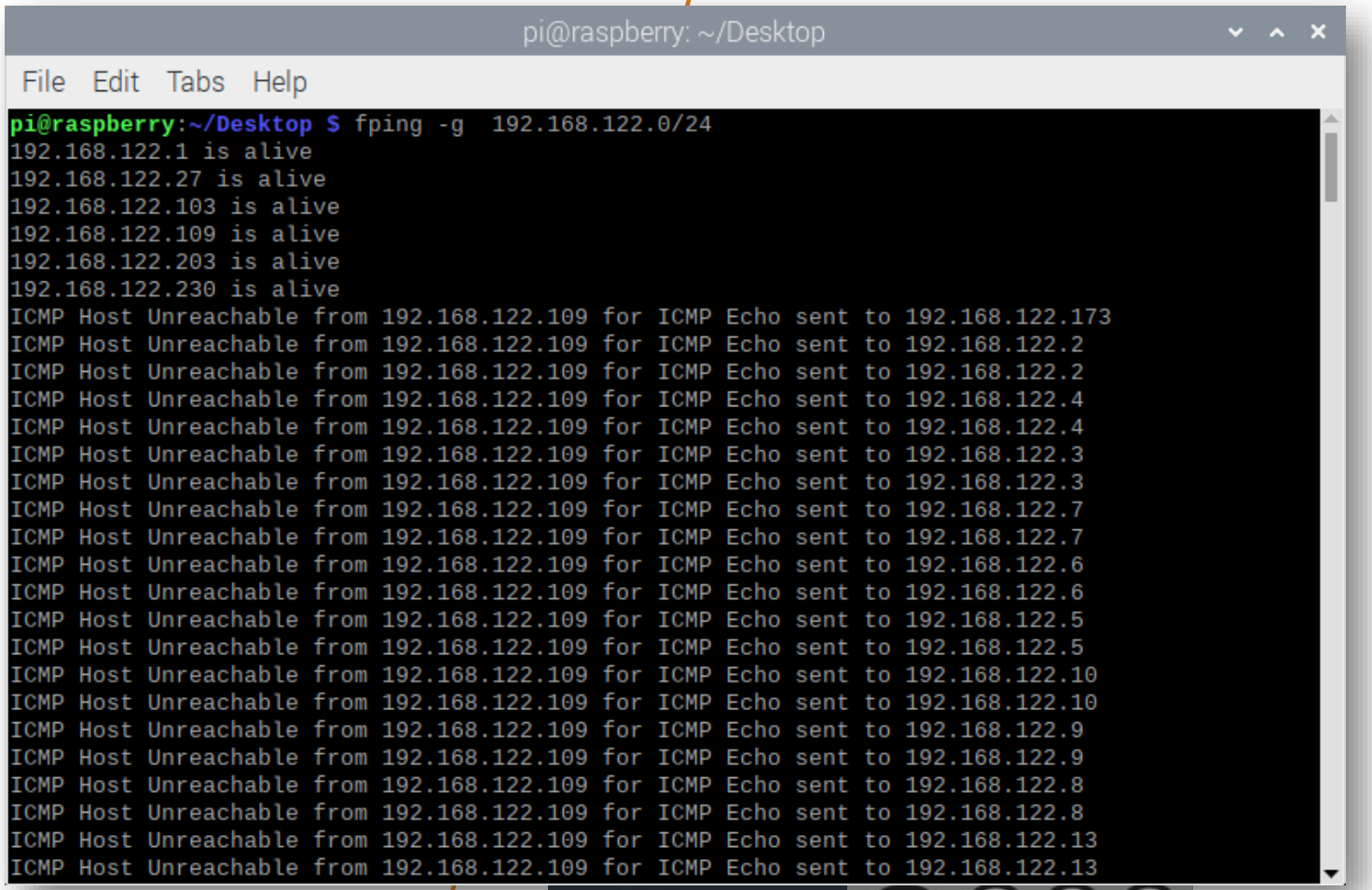
Examples on fping

fping is a program to send ICMP echo probes to network hosts, similar to ping, but much better performing when pinging multiple hosts.

Ping the entire subnet

```
fping -g 192.168.122.0/24
```

This can be used to perform a quick check in the network to detect any suspicious devices connected to it. For example, all active (alive status) devices should be known by the network administrator, but here, fping detects that there is a device with an IP ending in .27 that is unknown to the network, which could be recognized as an intruder.



Examples on fping

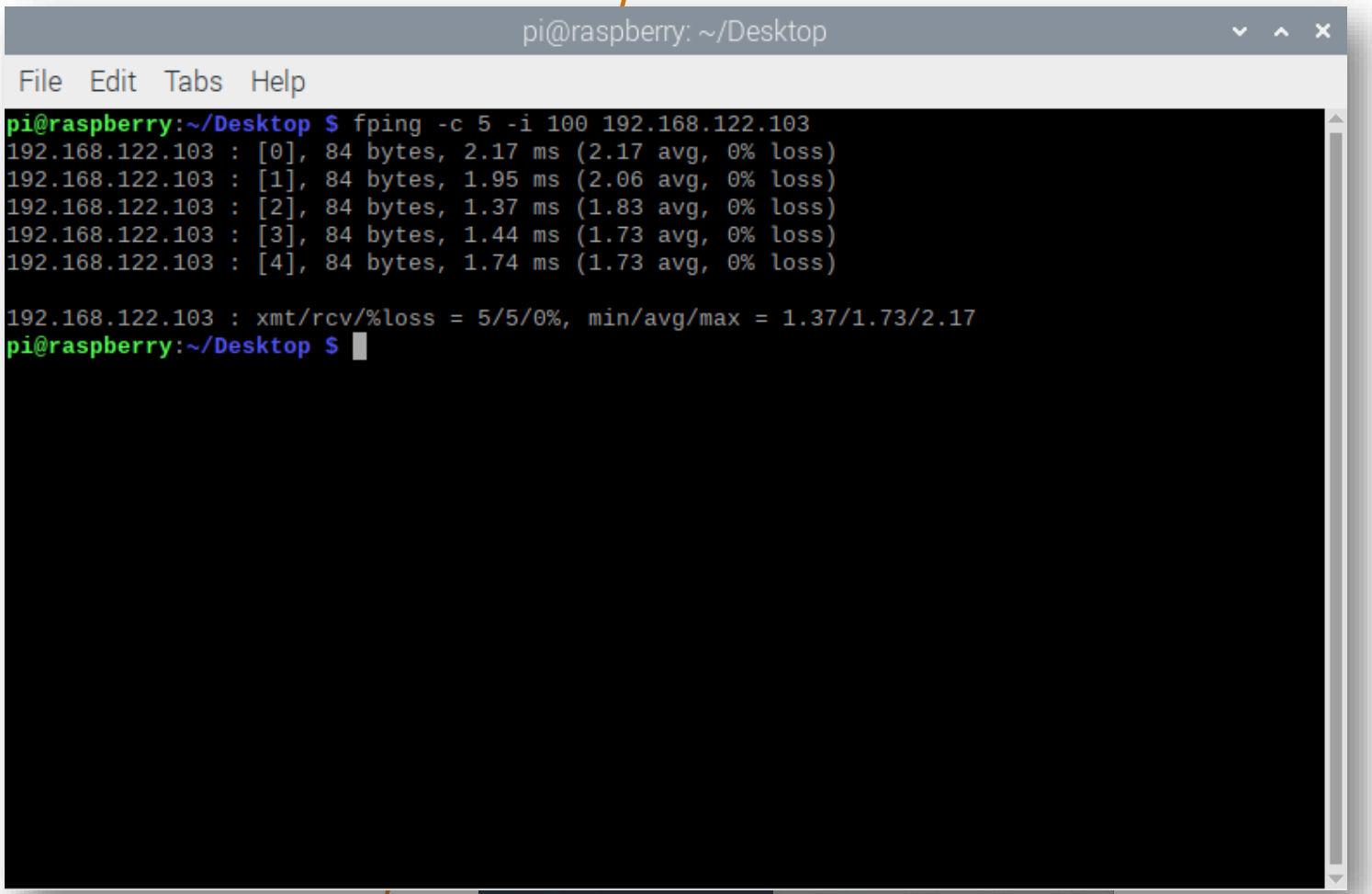
fping is a program to send ICMP echo probes to network hosts, similar to ping, but much better performing when pinging multiple hosts.

Rate of Pings

```
fping -c 5 -i 100  192.168.122.103
```

```
-c 5: Send 5 pings.
-i 100: Set the interval between
pings to 100 ms.
```

This command can be used to limit the rate of pings to avoid overwhelming the network.



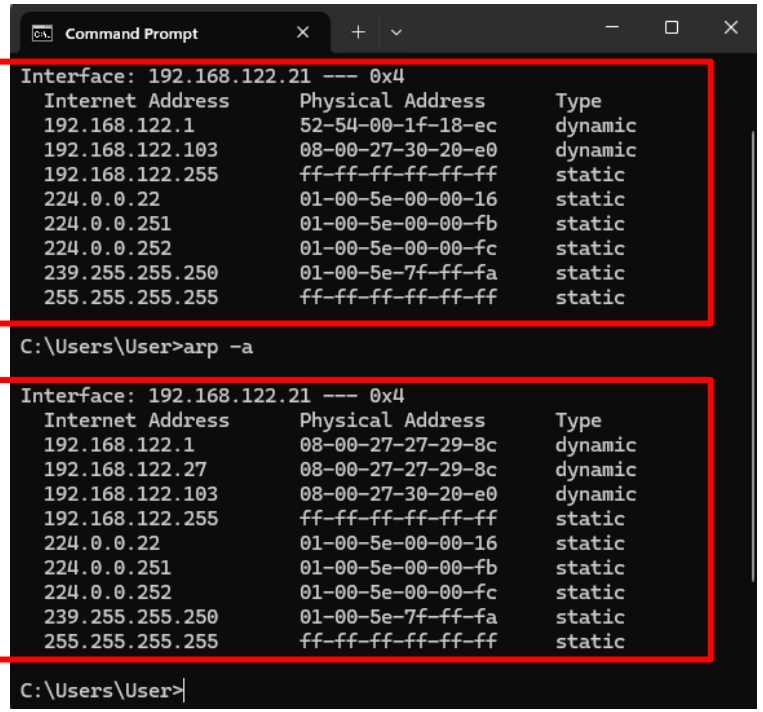
Detection of Cyberattacks

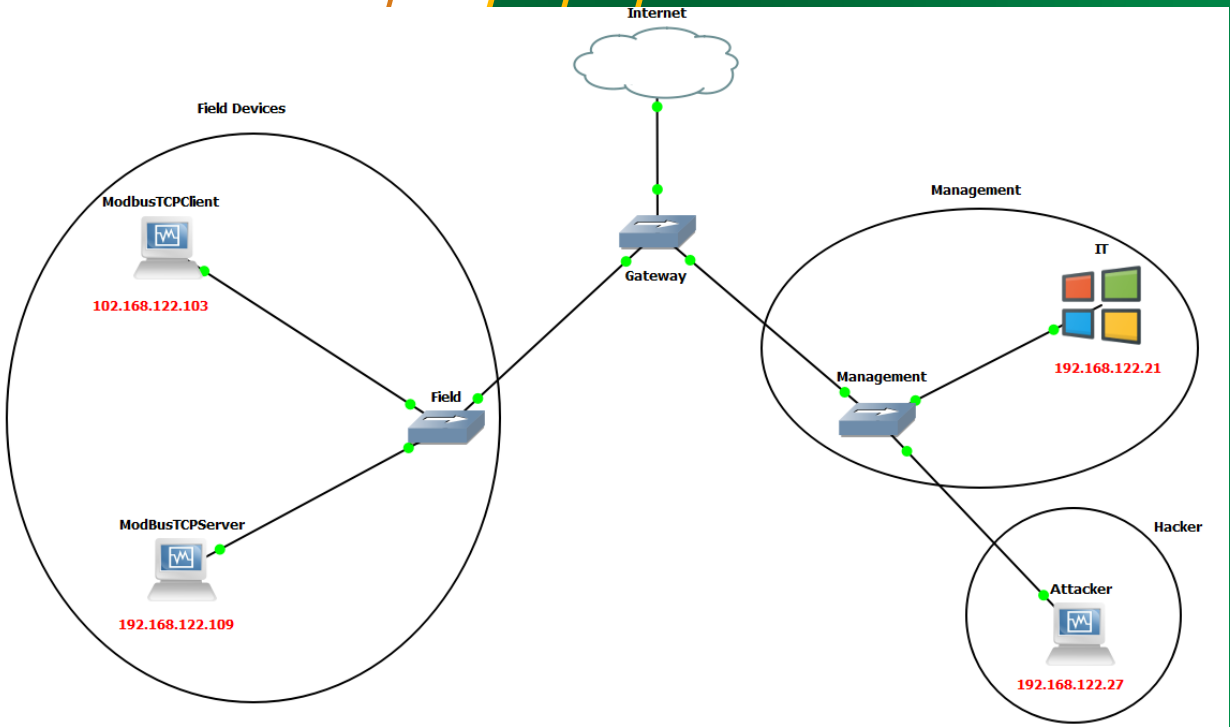
Detection of Cyberattacks

- Detecting **ARP Poisoning** Attacks on the network
- We consider the **Kali admin** machine to act as a network administration machine within the network.
- If any **suspicious activities** are occurring on the network, the network **administrator** should **detect** them. How?

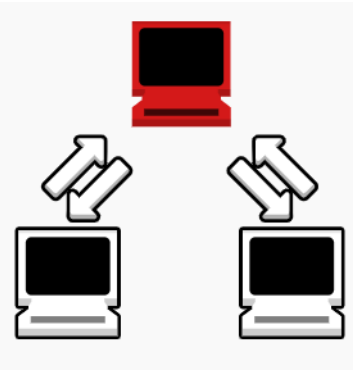
Before

After



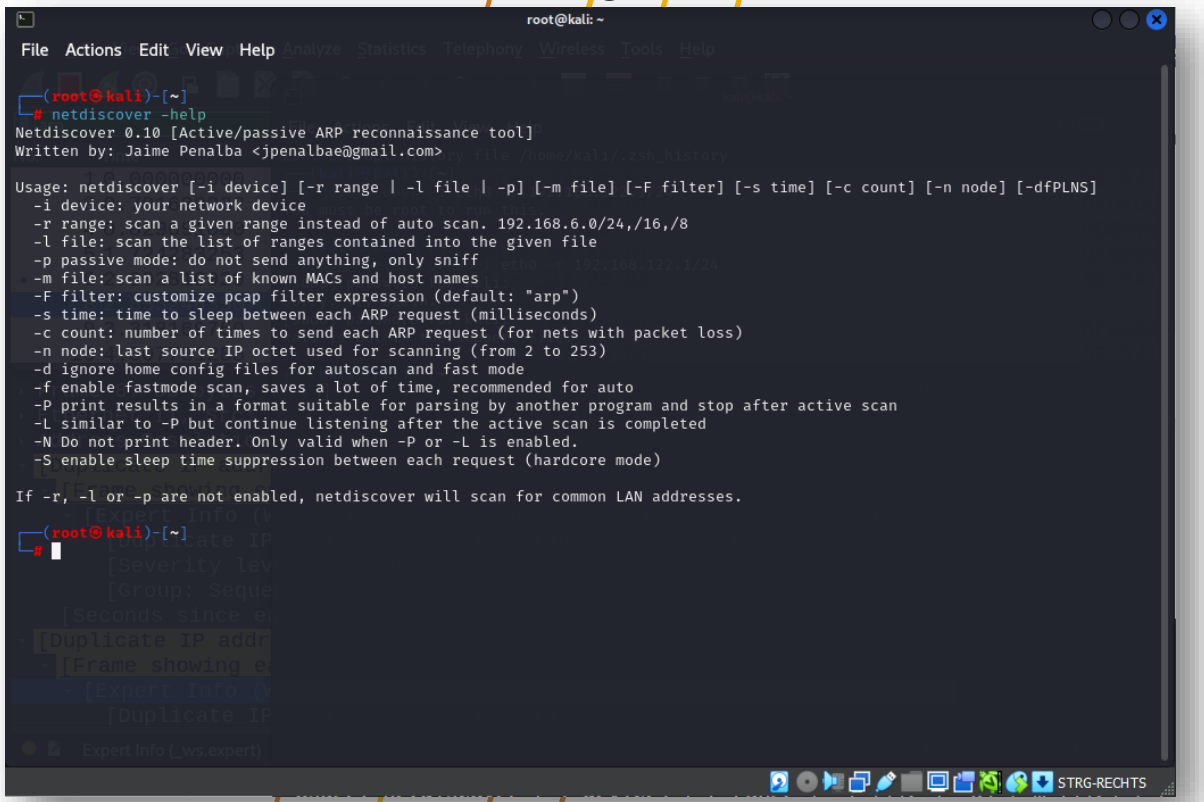


Detection of Cyberattacks

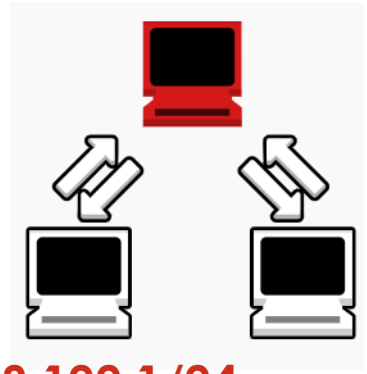


Netdiscover: is an **active/passive** address reconnaissance tool, mainly developed for those **wireless networks** without dhcp server, when you are wardriving. It can be also used on hub/switched networks.

Netdiscover can also be used to **inspect** your network **ARP traffic**, or find **network** addresses using auto-scan mode, which will scan for common local networks.



Detection of Cyberattacks



No ARP attack has been **initiated** in the network **against** the **server** with an IP ending in .**109.**

Arp -a

```
pi@raspberry: ~
File  Edit  Tabs  Help
pi@raspberry:~ $ arp -a
? (192.168.122.15) at 08:00:27:21:b1:d0 [ether] on eth0
_gateway (192.168.122.1) at 52:54:00:1f:18:ec [ether] on eth0
? (192.168.122.173) at <incomplete> on eth0
? (192.168.122.67) at 08:00:27:fa:75:14 [ether] on eth0
? (192.168.122.27) at 08:00:27:27:29:8c [ether] on eth0
pi@raspberry:~ $
```

Server

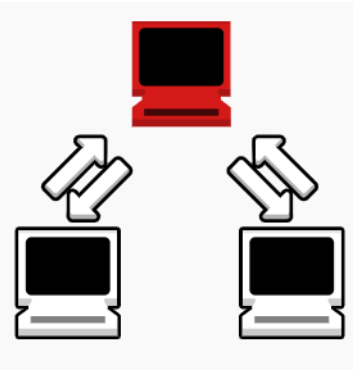
netdiscover –i eth0 –r 192.168.122.1/24

```
root@kali: ~
File  Actions  Edit  View  Help
Currently scanning: Finished!   |   Screen View: Unique Hosts

39 Captured ARP Req/Rep packets, from 3 hosts.   Total size: 2340

  IP            At MAC Address     Count     Len  MAC Vendor / Hostname
192.168.122.109 08:00:27:52:a4:8f     33    1980  PCS Systemtechnik GmbH
192.168.122.1   52:54:00:1f:18:ec      5     300  Unknown vendor
192.168.122.27  08:00:27:27:29:8c      1      60  PCS Systemtechnik GmbH
```

Admin Kali

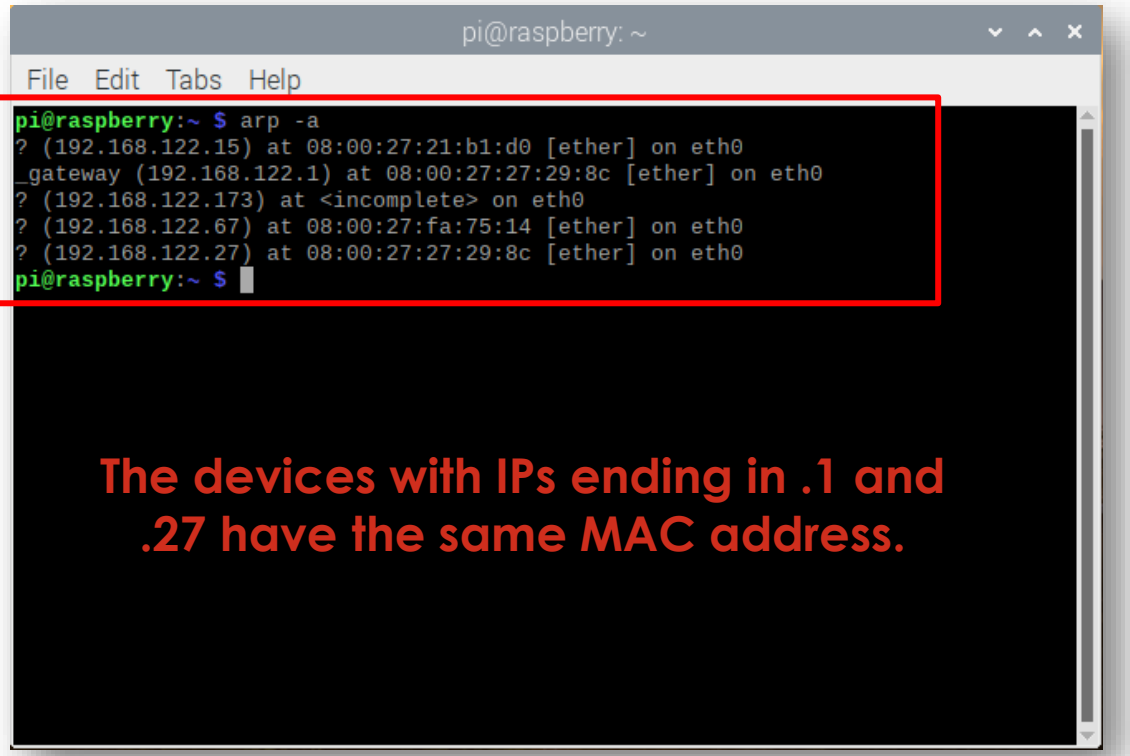
CyberSecPro

Detection of Cyberattacks



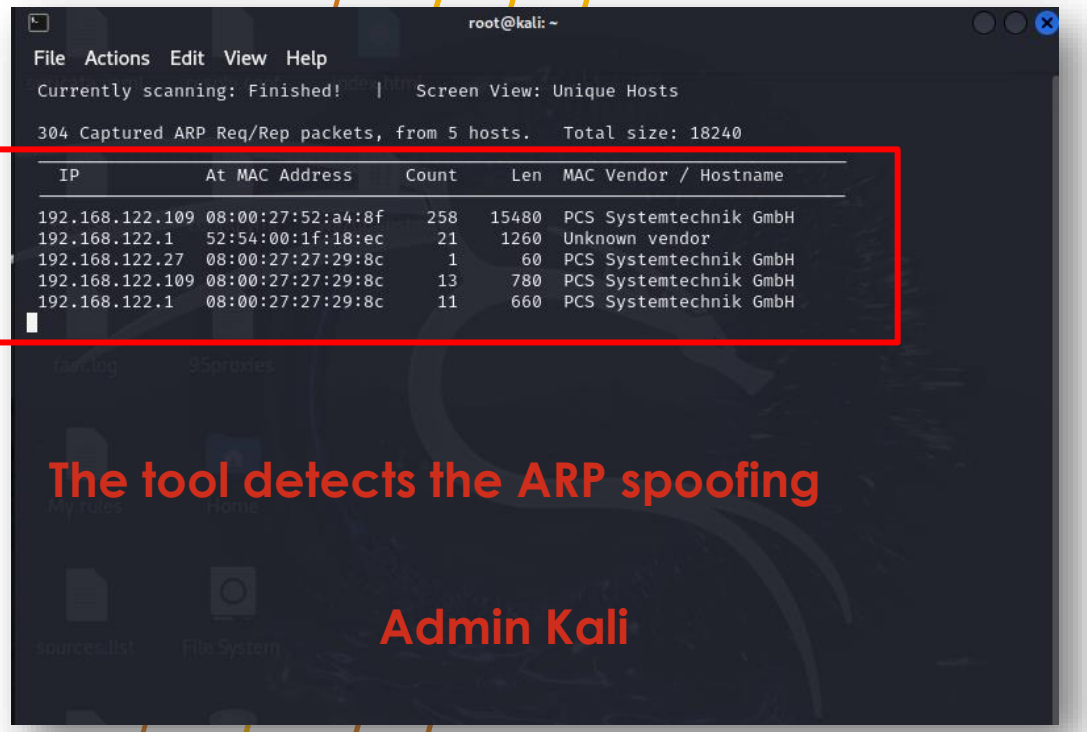
The ARP attack has been **initiated** from the attacker side (arpspoof or bettercap) in the network **against** the **server** with an IP ending in **.109.**

Arp -a



Server

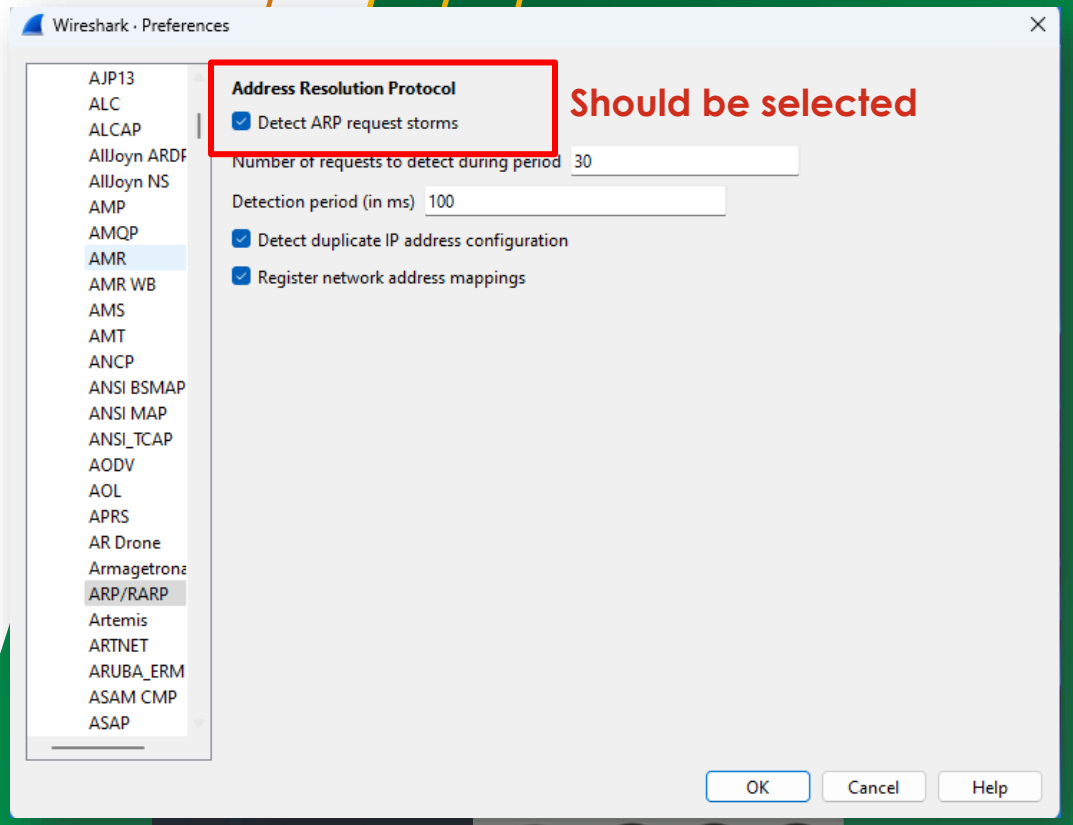
netdiscover –i eth0 –r 192.168.122.1/24



Detection of Cyberattacks

Detecting suspicious Activities in The Network using **Wireshark**

- Run **Wireshark**, and let the tool detect an **ARP request** packets through the network.

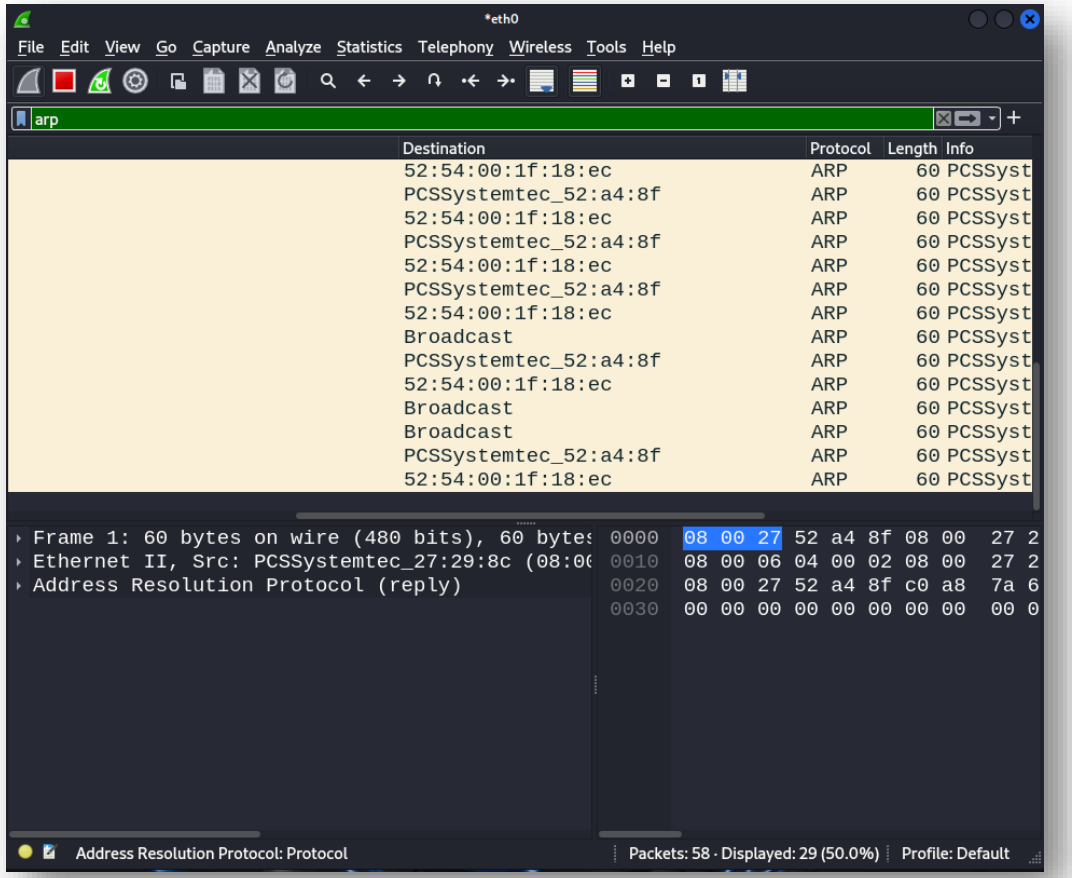


- Run **Wireshark**, and let the tool detect an **ARP request** packets through the network.
- To do so, do the following:
  - Select **preferences** from **Edit**, then **select Protocols→ ARP/RARP**
  - Keep the "**Detect ARP request storms**" selected. Then press **OK**

Detection of Cyberattacks

Detecting suspicious Activities in The Network using Wireshark

- Run the **Wireshark** tool on the **Kali** admin machine.



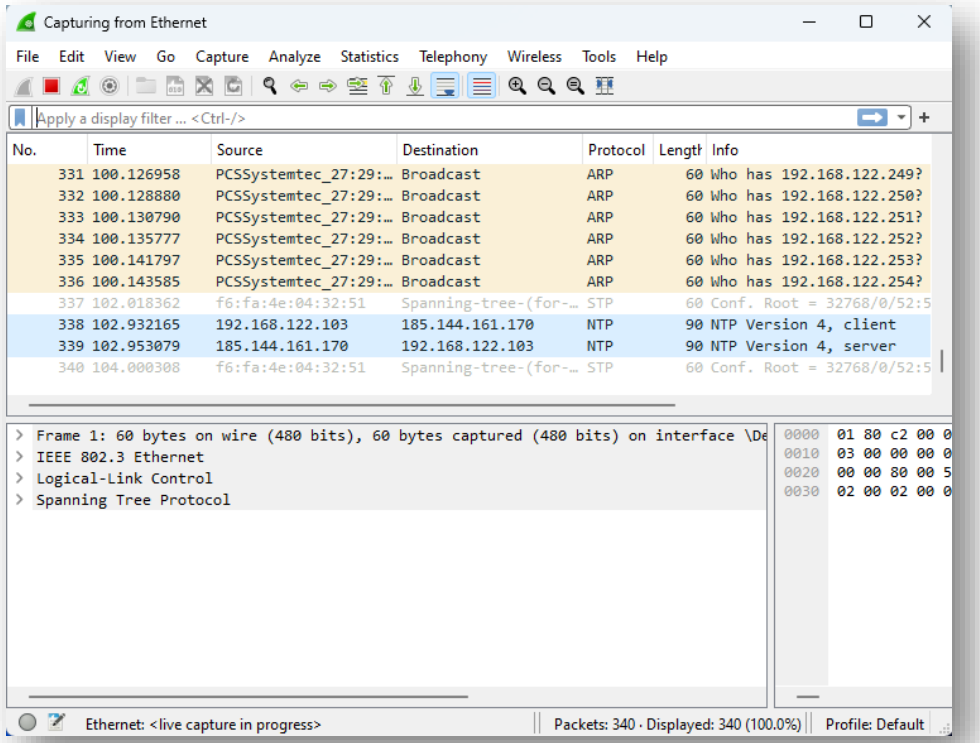
- Filter packets using the arp
- Wireshark start detecting ARP requests over the network

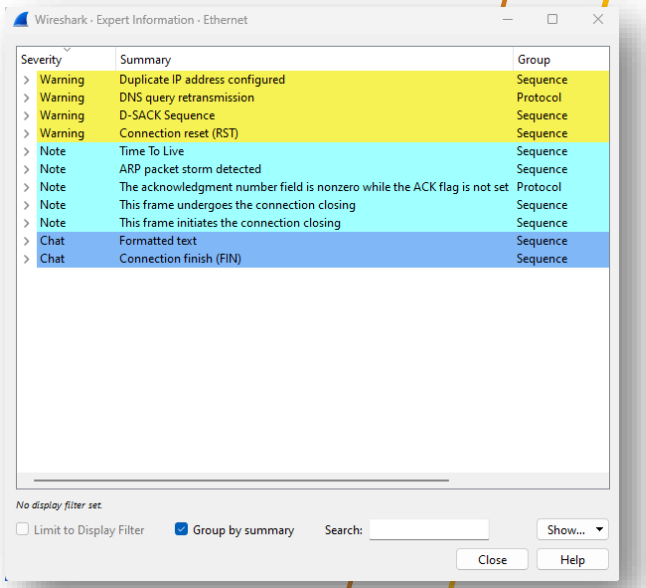
Detection of Cyberattacks

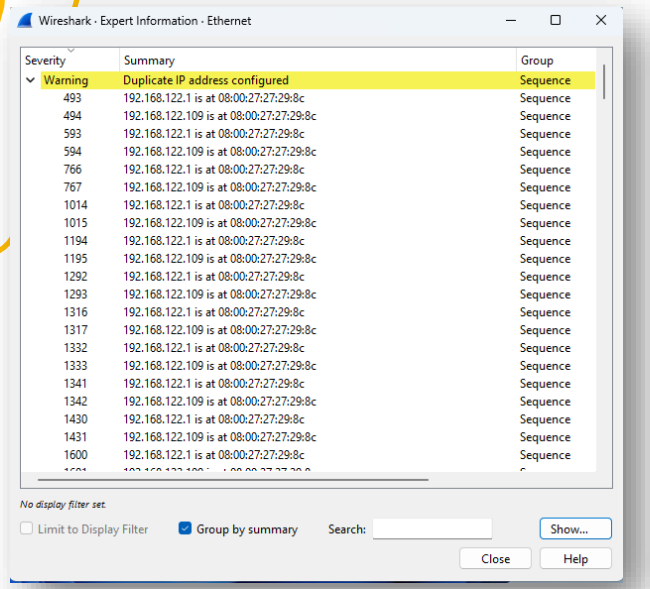
Detecting suspicious Activities in The Network using Wireshark

- Run the arb spoofing attack using any of the previously discussed too (**bettercap** or **arpspoof**)

Bettercap –iface eth0 –caplet spoof.cap







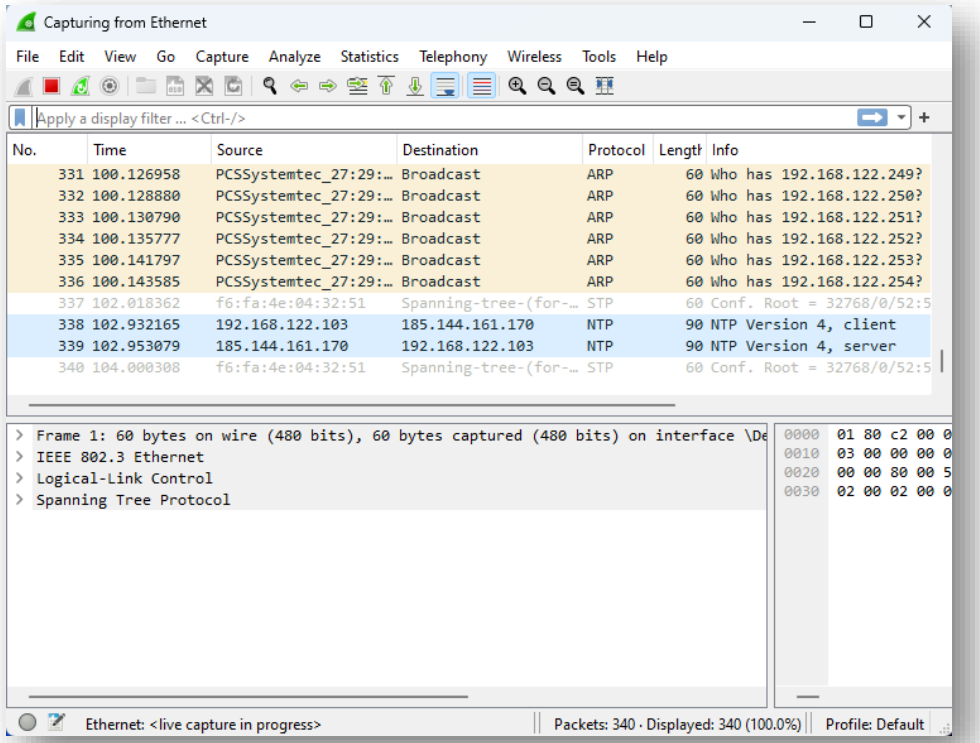
- Wireshark starts **detecting ARP poising** requests **over** the network.
- Formore **analysis**, select **Expert Infromation** from the **Analyze Menue**

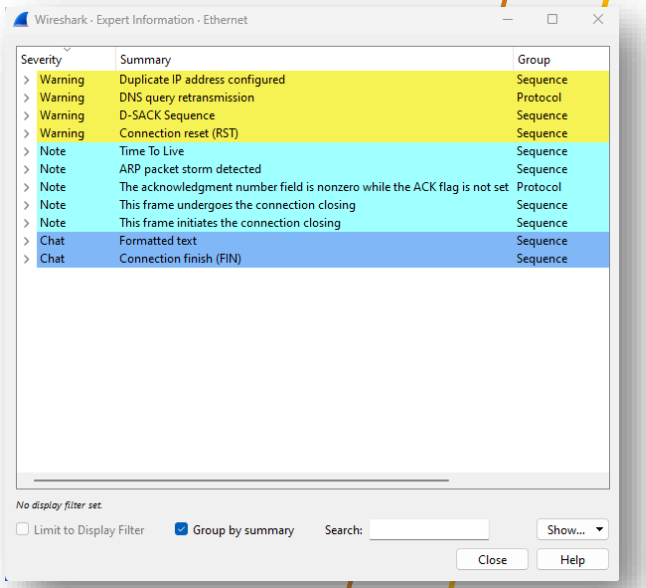
Detection of Cyberattacks

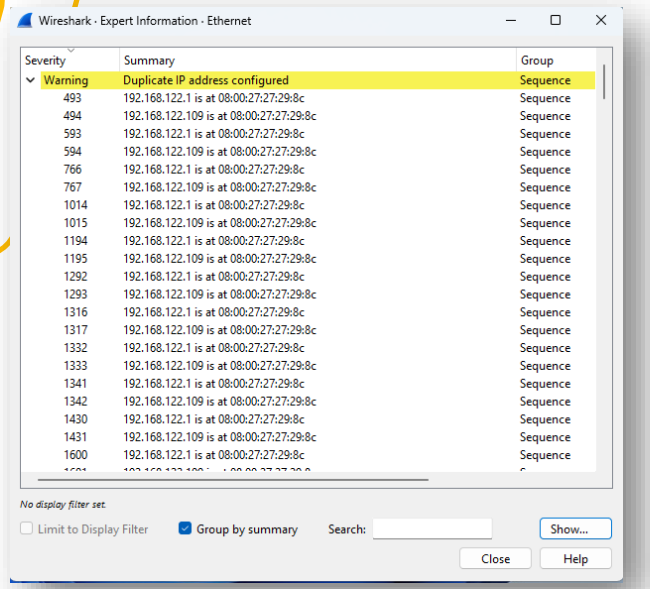
Detecting suspicious Activities in The Network using Wireshark

- Run the arb spoofing attack using any of the previously discussed too (**bettercap** or **arpspoof**)

Bettercap –iface eth0 –caplet spoof.cap







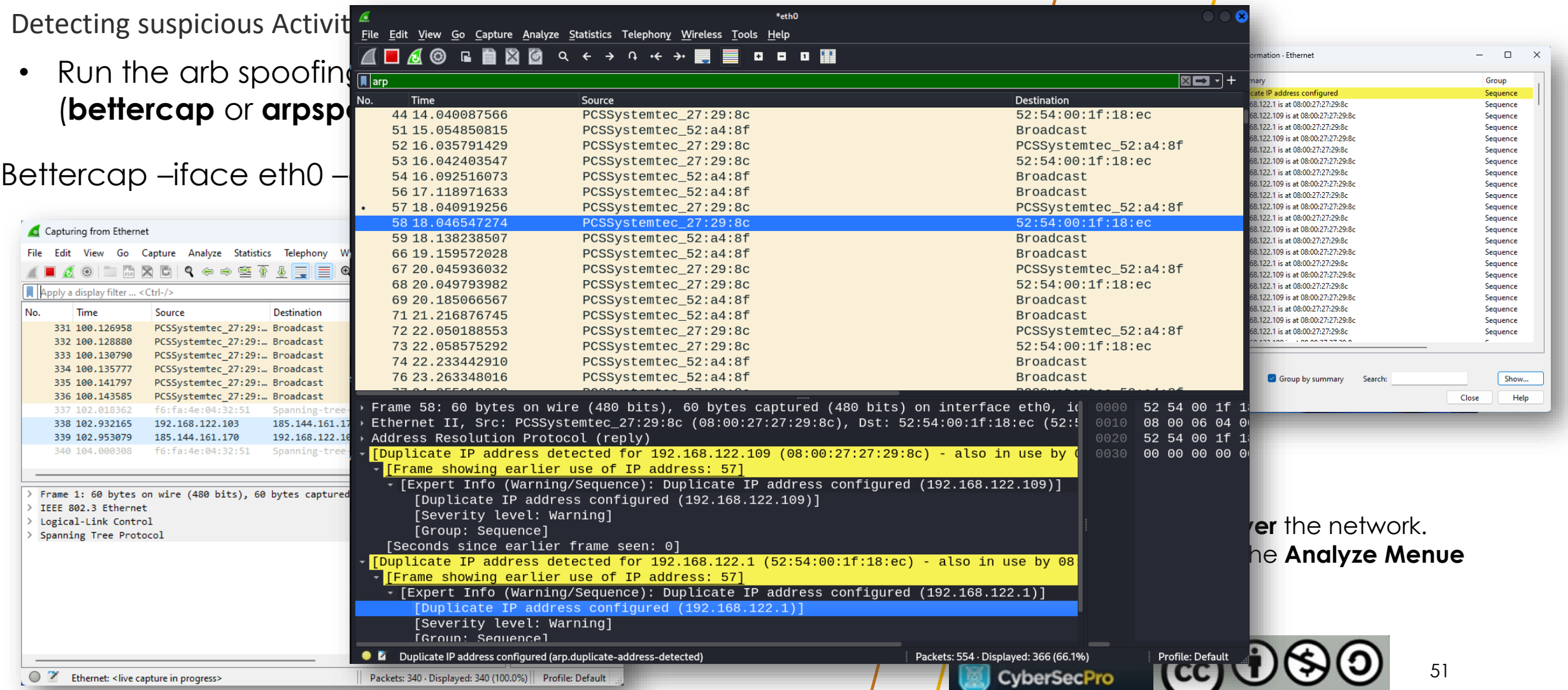
- Wireshark starts **detecting ARP poising** requests **over** the network.
- Formore **analysis**, select **Expert Infromation** from the **Analyze Menue**

Detection of Cyberattacks

Detecting suspicious Activit

- Run the arb spoofin
  (**bettercap** or **arpsp**

Bettercap –iface eth0 –



er the network.
he **Analyze Menue**

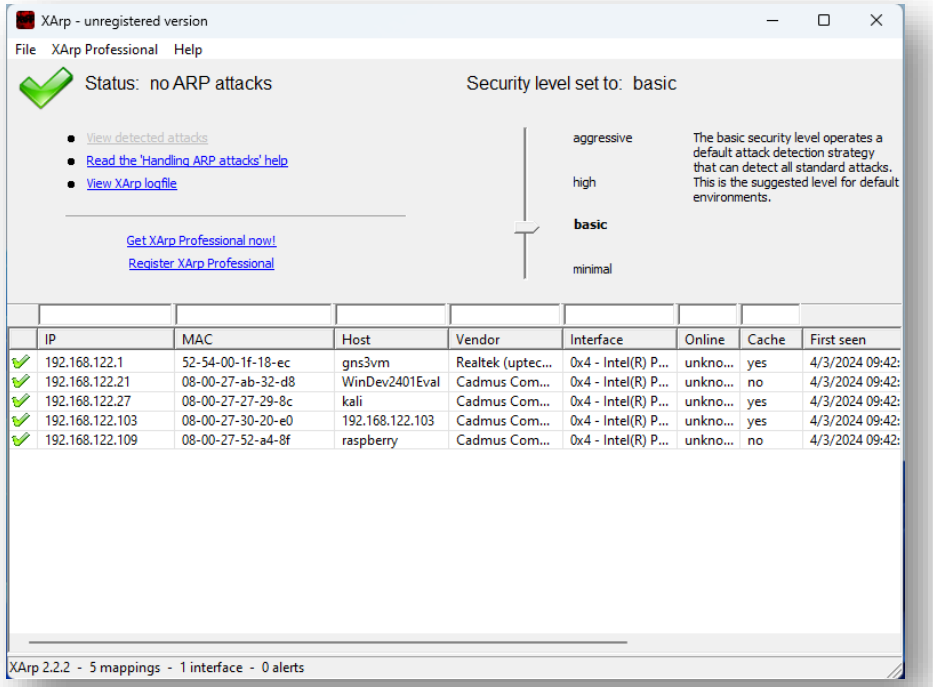
Detection of Cyberattacks

Windows OS

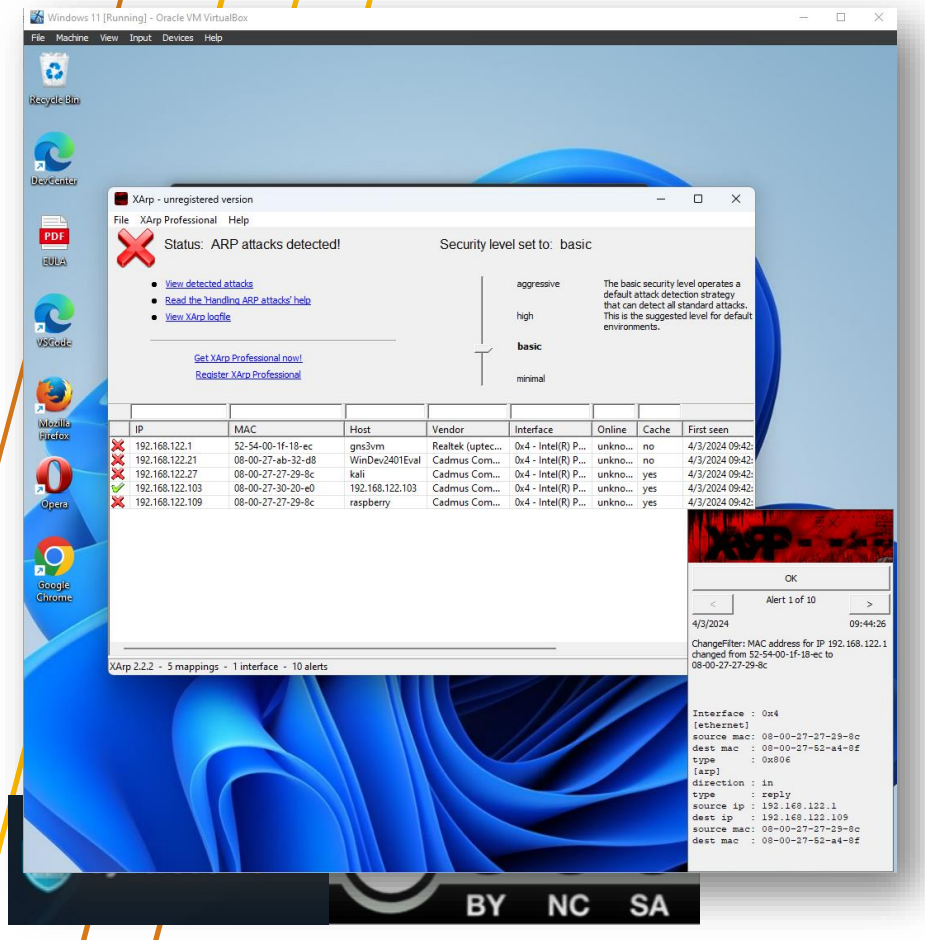
ARP Poisoning Attacks

- The "**arp -a**" command should be **executed manually** every time to **detect** such suspicious activities.
- The **xarp tool** can **automatically detect** any such activities.

Before

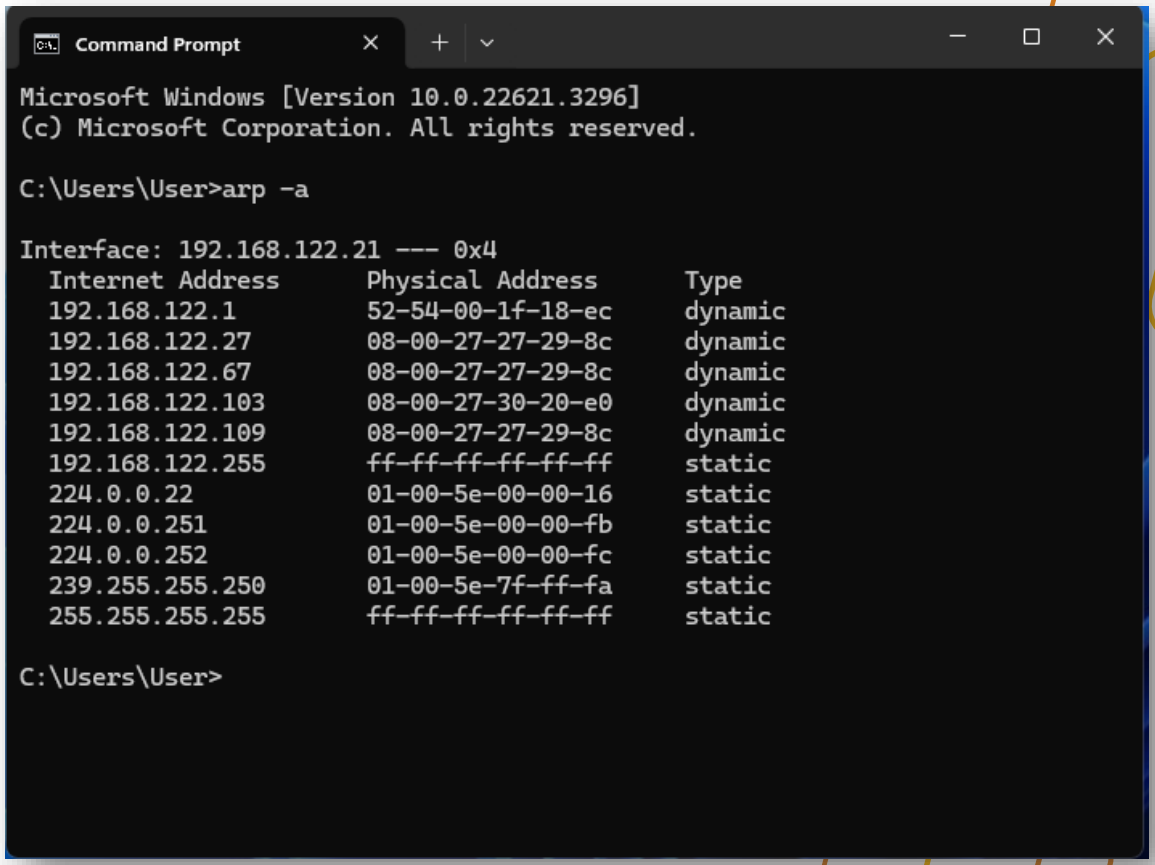


After



Prevention of ARP Poisoning Attack

Windows OS



"**Dynamic**" could be changed to "**static**," but when a **new device** is added to the network, you must configure it **manually**.

ARP Poisoning

Protection

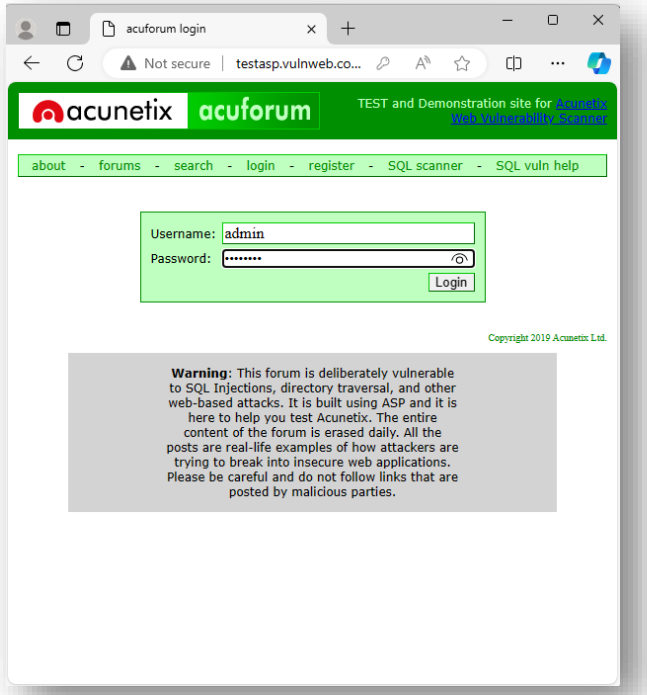
Prevention of MITM

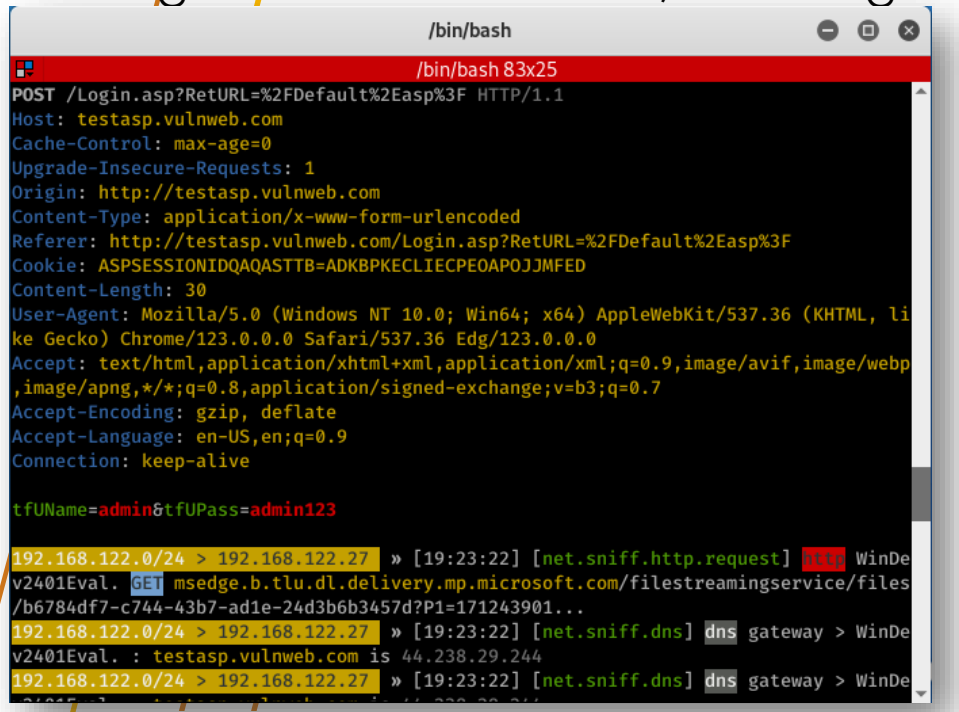
- Detection is not **considered** a **prevention** action.
- The previously discussed **detection approaches** were used only for **detecting ARP spoofing** attacks.
- To ensure network security, **encrypt traffic** using **HTTPS** (browser plugins).
- Most recent **versions** of **web browsers** support **services** that warn when visiting **HTTP websites**.
- Some plugins can redirect from **HTTP** to **HTTPS**, but hackers can still obtain information about your **insecure** communication.
- **VPN** is the **best solution** for **preventing** MITM attacks.

Protection
VPN

VPN for Prevention of MITM

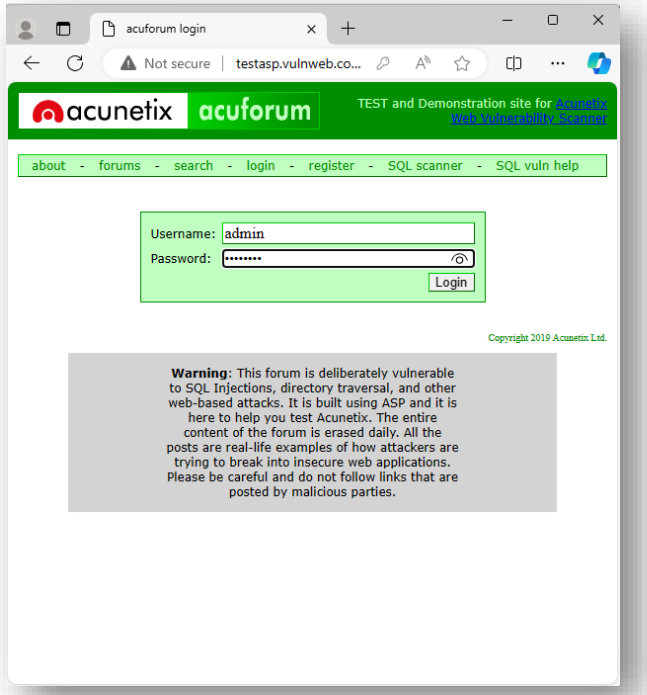
- Use any **preferred VPN** tool for your test.
- I use **Avira VPN**, the **free version**, for **testing**.
- There is a **free version available** with about **500** MB of **data stream**.
- Installed it on my **Windows machine**.
- **Before** turning the **VPN on**, visit **testasp.vulnweb.com** and **insert** your **login** credentials (**username** and **password**).
- At the **same time**, check **Bettercap** for **collecting** all visiting **website information**, including the **credentials**.

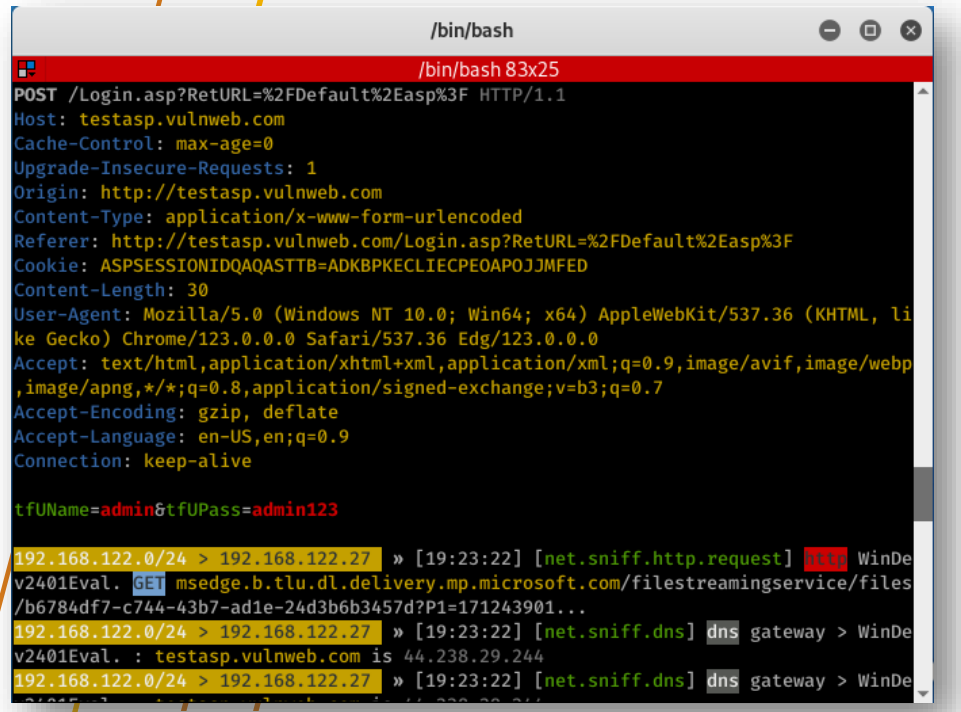




VPN for Prevention of MITM

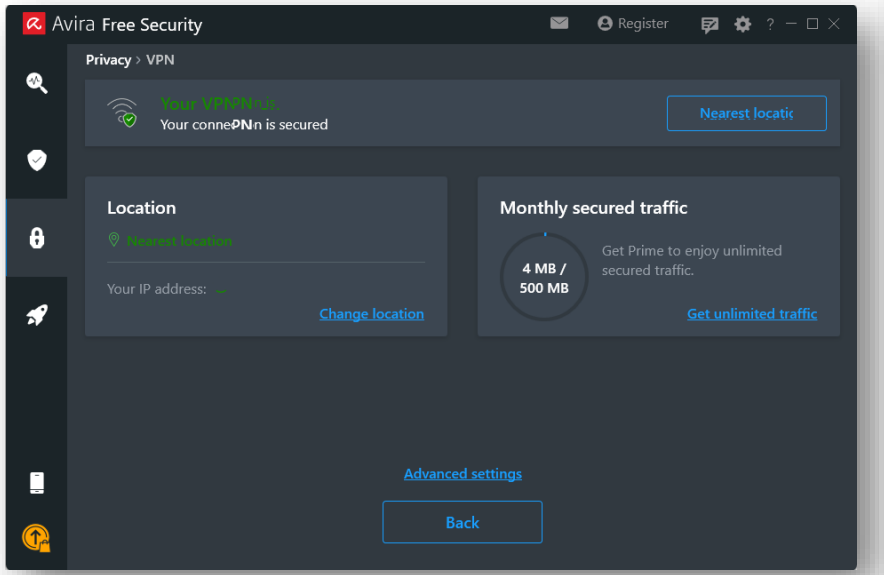
- Check the IP of the Windows device using the ipconfig

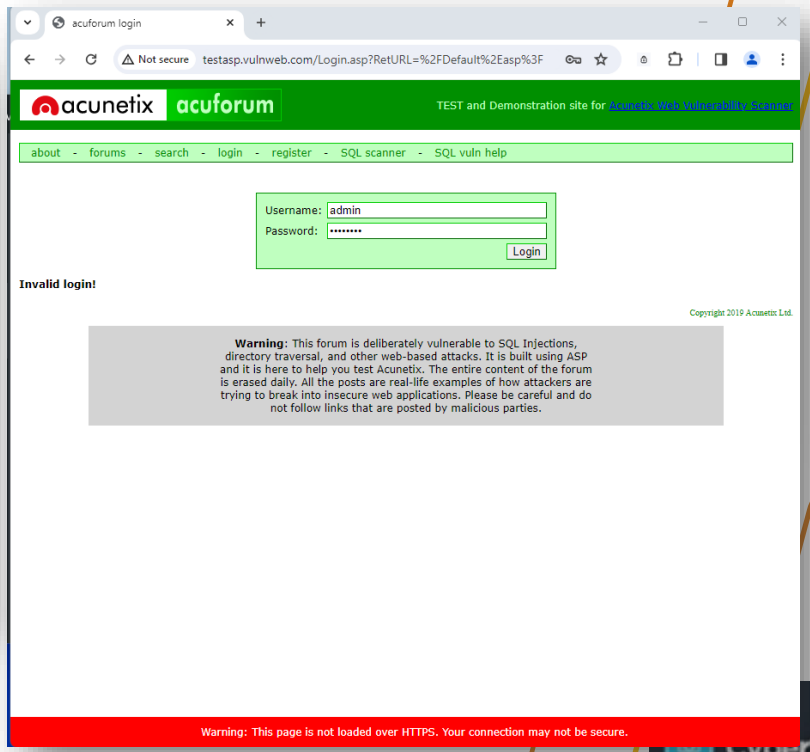


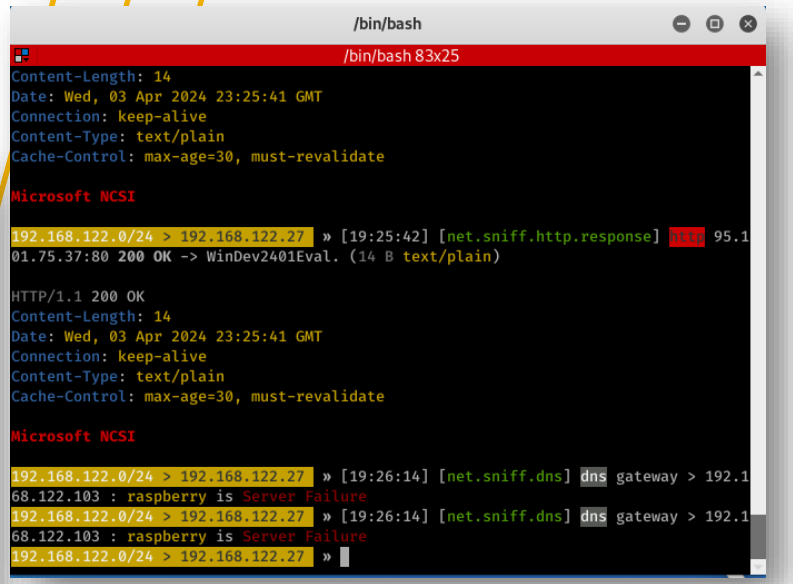


VPN for Prevention of MITM

- **Turn on** the **VPN** on your **Windows machine** (**victim** machine), and then reload the **same page** and **insert** your **access credentials**
- **Opserve** the **difference**







Thank you

Please send all questions to:
Abdelkader Shaaban,
abdelkader.Shaaban@ait.ac.at
Stefan Schauer
Stefan.Schauer@ait.ac.at