



EDUCATION AND TRAINING

CyberSecPro Training

We are creating cutting-edge education and training to advance competencies and professionalism in EU cybersecurity.

OUR VISION

Next level cybersecurity education and training



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.

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.

How Does iptables Work?

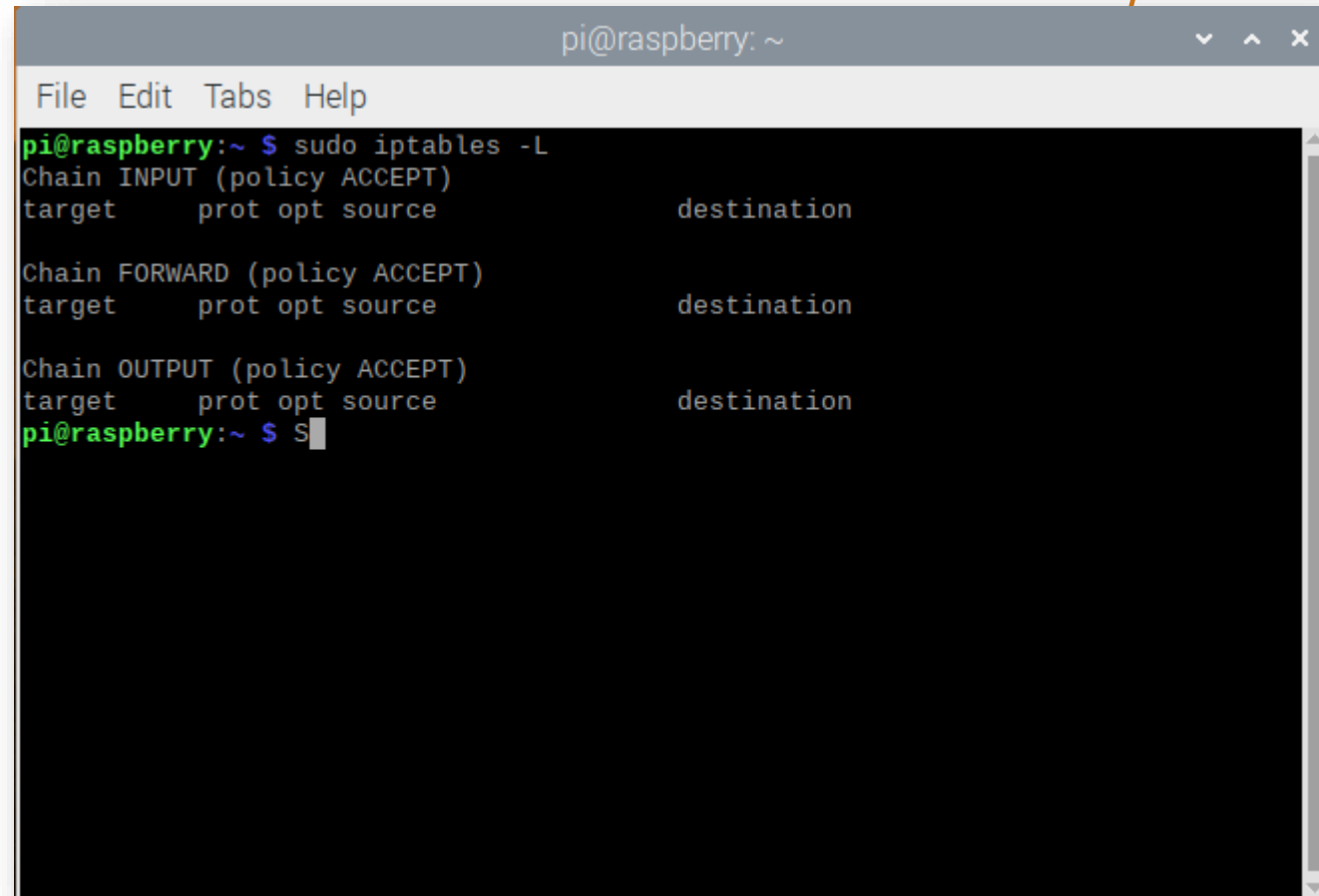
\$ iptable <operations> <string> <conditions> <action> <action>

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

Examples on the iptables

SERVER

```
sudo iptables -L
```



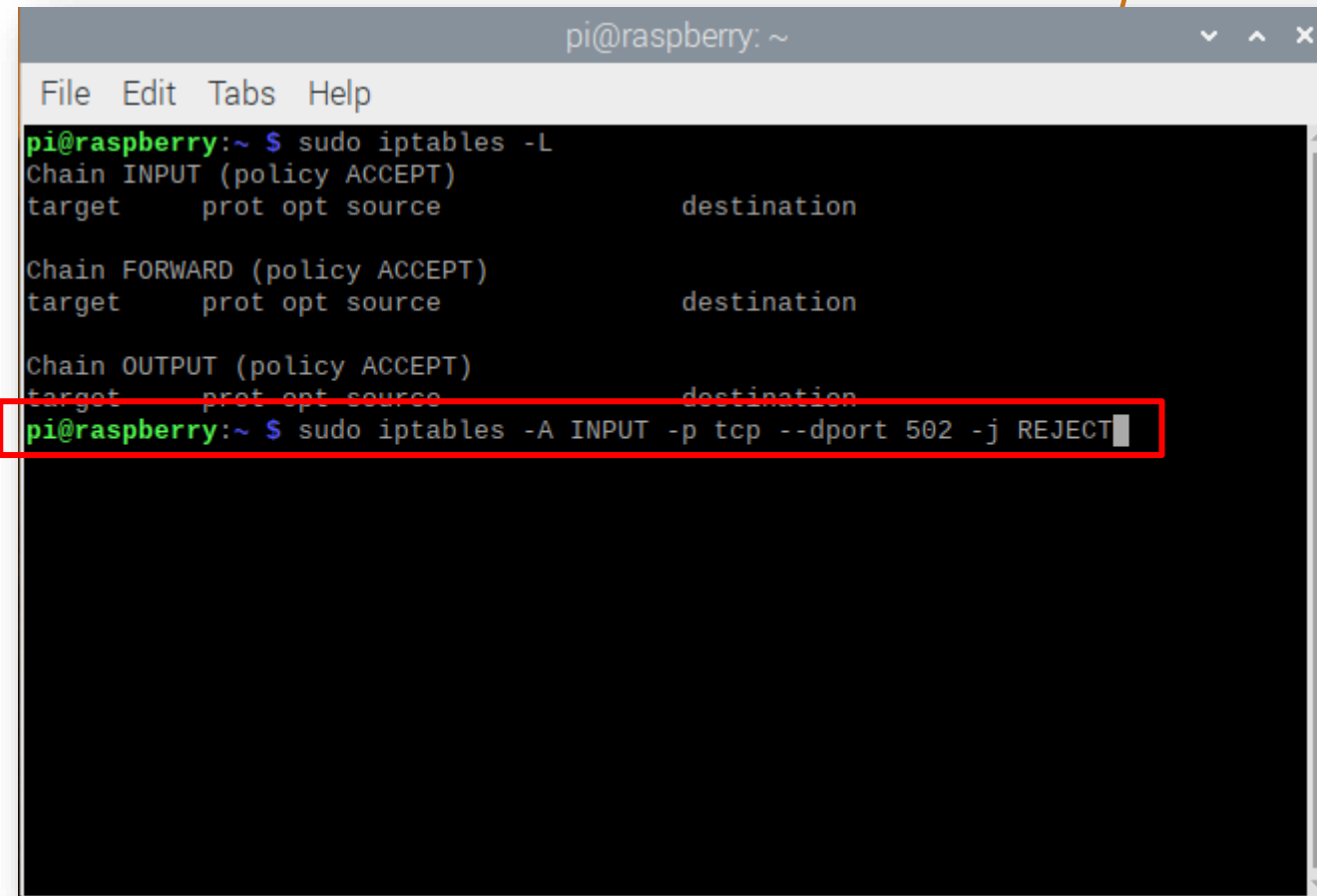
```
pi@raspberrypi: ~  
File Edit Tabs Help  
pi@raspberrypi:~$ sudo iptables -L  
Chain INPUT (policy ACCEPT)  
target      prot opt source      destination  
  
Chain FORWARD (policy ACCEPT)  
target      prot opt source      destination  
  
Chain OUTPUT (policy ACCEPT)  
target      prot opt source      destination  
pi@raspberrypi:~$ S
```

Examples on the iptables

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

SERVER

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



A terminal window titled 'pi@raspberrypi: ~' showing the output of the command 'sudo iptables -L'. The output lists three chains: INPUT, FORWARD, and OUTPUT, all with a policy of ACCEPT. Each chain has a table with columns for target, protocol, options, source, and destination. The INPUT chain is currently empty. Below the output, the command 'sudo iptables -A INPUT -p tcp --dport 502 -j REJECT' is entered and highlighted with a red box.

```
pi@raspberrypi:~ $ sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source                destination

Chain FORWARD (policy ACCEPT)
target     prot opt source                destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source                destination
pi@raspberrypi:~ $ sudo iptables -A INPUT -p tcp --dport 502 -j REJECT
```

Examples on the iptables

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

Test the communication between the client and server

SERVER

```
pi@raspberrypi: ~/Desktop
File Edit Tabs Help
pi@raspberrypi:~/Desktop $ sudo python3 server2.py
Modbus server started on 192.168.122.109 port 502
DEBUG:pymodbus.server.sync:Modbus server stopped
Traceback (most recent call last):
  File "server2.py", line 37, in <module>
    run_async_server()
  File "server2.py", line 32, in run_async_server
    server = ModbusTcpServer(context, identity=identity, address=("192.168.122.109", 502))
  File "/usr/local/lib/python3.7/dist-packages/pymodbus/server/sync.py", line 340, in __init__
    **kwargs)
  File "/usr/lib/python3.7/socketserver.py", line 452, in __init__
    self.server_bind()
  File "/usr/lib/python3.7/socketserver.py", line 466, in server_bind
    self.socket.bind(self.server_address)
OSError: [Errno 98] Address already in use
pi@raspberrypi:~/Desktop $
```

**The port is
already closed**

Examples on the iptables

Delete rules

`sudo iptables -L --line-numbers`

```

pi@raspberrypi:~$ sudo iptables -L --line-numbers
Chain INPUT (policy ACCEPT)
num target prot opt source destination
1 REJECT tcp -- anywhere tcp dpt:502 reject-with icmp-port-unreachable

Chain FORWARD (policy ACCEPT)
num target prot opt source destination

Chain OUTPUT (policy ACCEPT)
num target prot opt source destination
pi@raspberrypi:~$
  
```

SERVER

`sudo iptables -D INPUT 1`

```

pi@raspberrypi:~$ sudo iptables -L --line-numbers
Chain INPUT (policy ACCEPT)
num target prot opt source destination
1 REJECT tcp -- anywhere tcp dpt:502 reject-with icmp-port-unreachable

Chain FORWARD (policy ACCEPT)
num target prot opt source destination

Chain OUTPUT (policy ACCEPT)
num target prot opt source destination
pi@raspberrypi:~$ 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

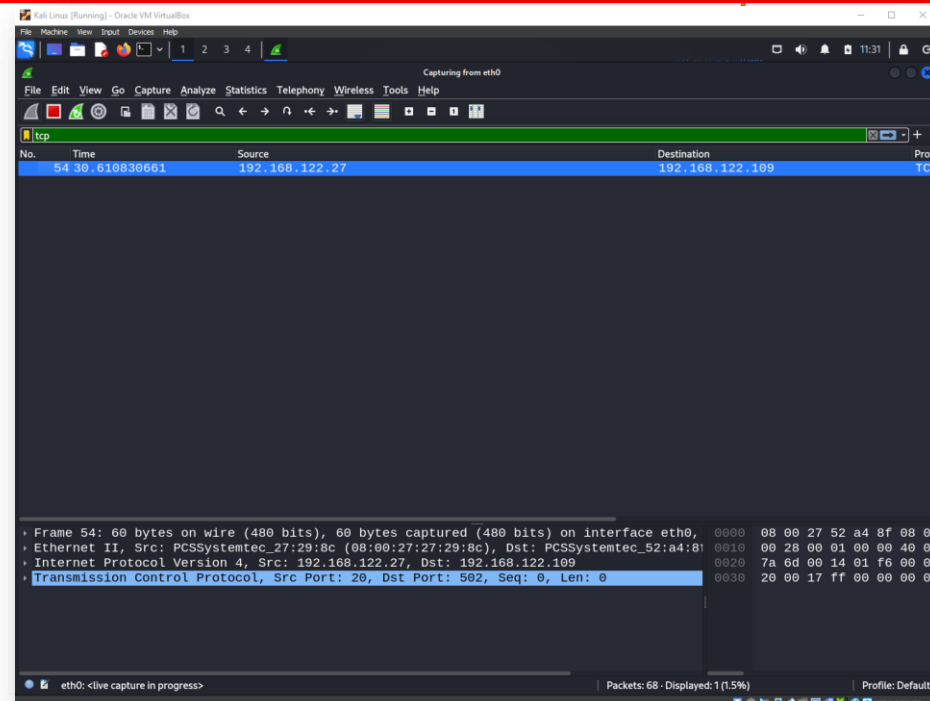
Start Wireshark on the Kali machine (Admin), and apply a filter to capture only TCP packets.

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

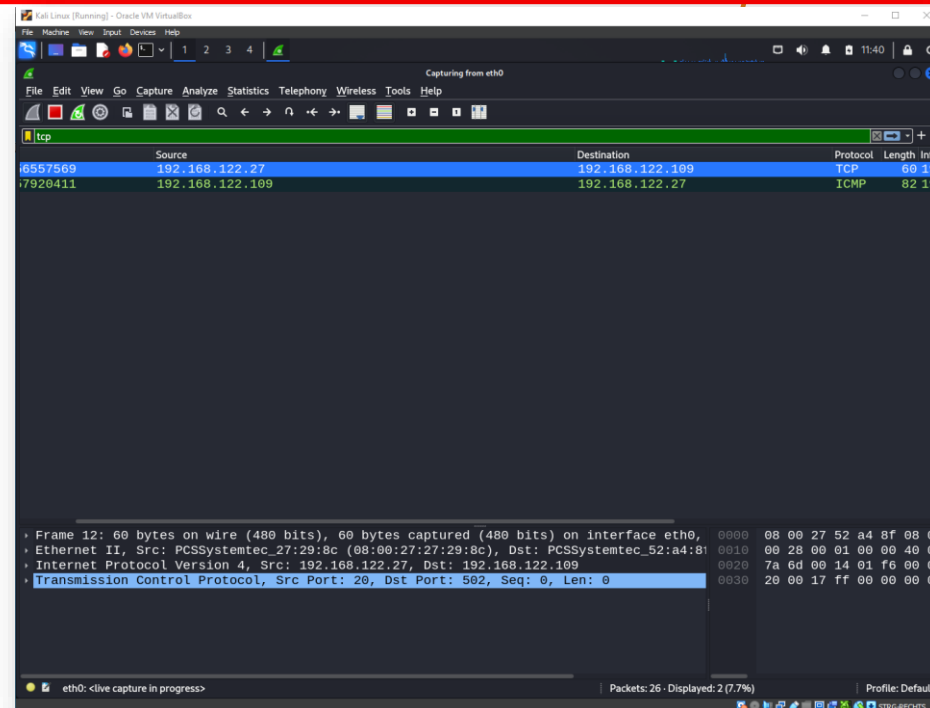
REJECT VS DROP Targets

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

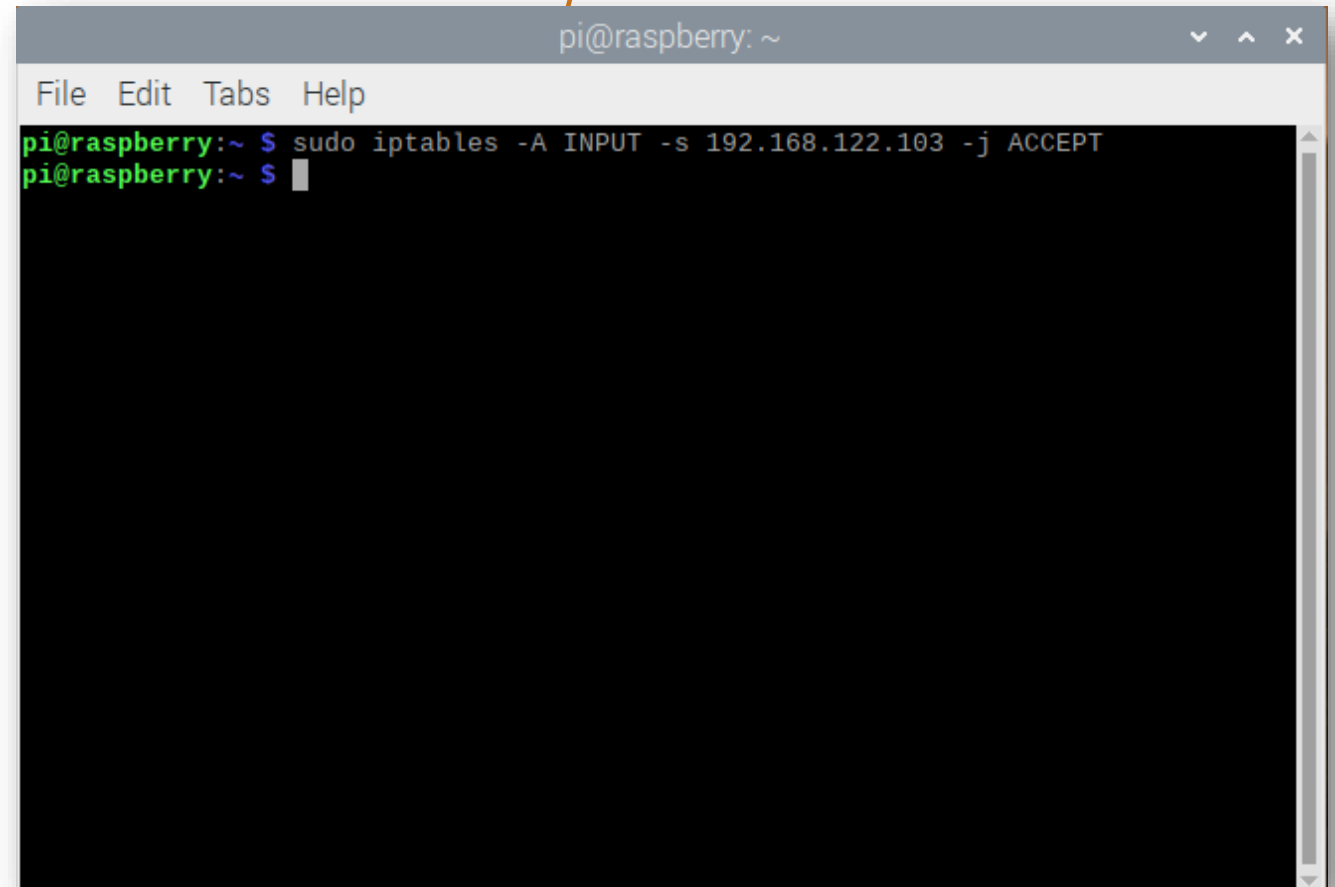
SERVER

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**.

A terminal window titled 'pi@raspberrypi: ~' with a menu bar containing 'File Edit Tabs Help'. The terminal shows the command 'sudo iptables -A INPUT -s 192.168.122.103 -j ACCEPT' being executed successfully, followed by a new prompt 'pi@raspberrypi: ~ \$'.

```
pi@raspberrypi: ~  
File Edit Tabs Help  
pi@raspberrypi:~ $ sudo iptables -A INPUT -s 192.168.122.103 -j ACCEPT  
pi@raspberrypi:~ $
```

Examples on the iptables

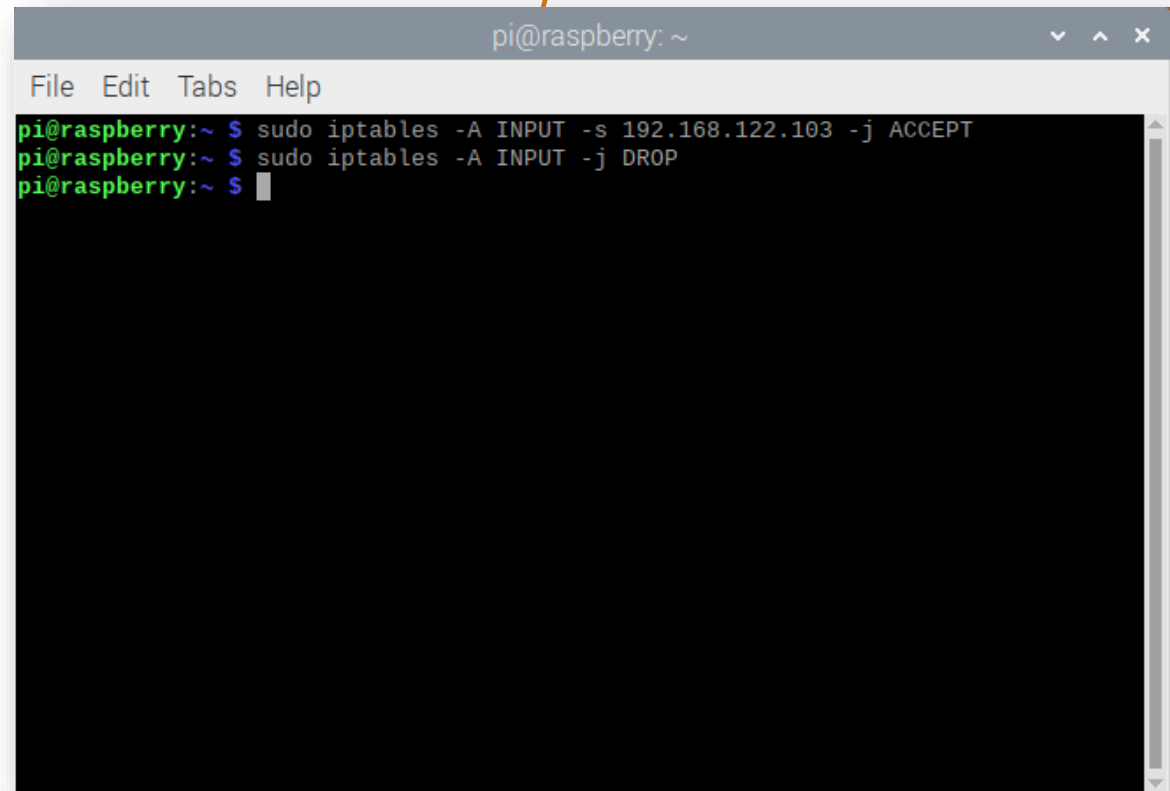
SERVER

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.



```
pi@raspberrypi: ~  
File Edit Tabs Help  
pi@raspberrypi:~$ sudo iptables -A INPUT -s 192.168.122.103 -j ACCEPT  
pi@raspberrypi:~$ sudo iptables -A INPUT -j DROP  
pi@raspberrypi:~$
```

Examples on the iptables

SERVER

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

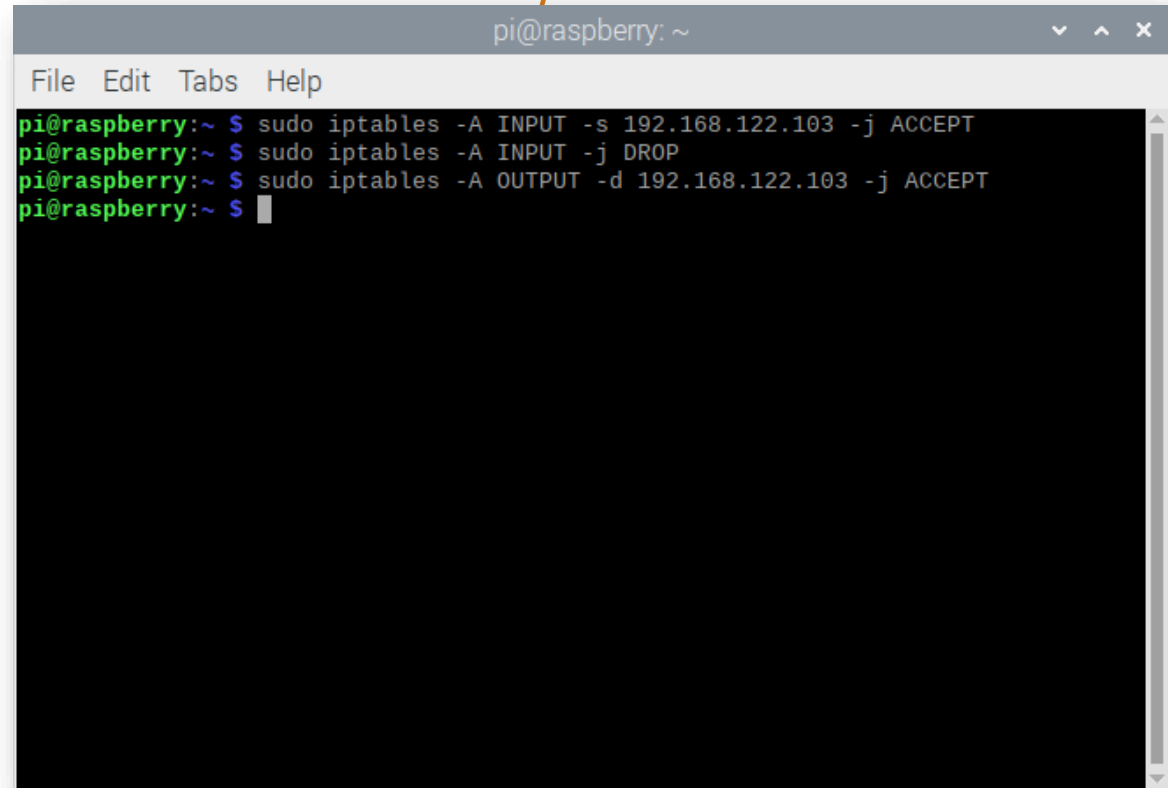
3. Allow outgoing traffic to the specific IP:

```
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 ;)



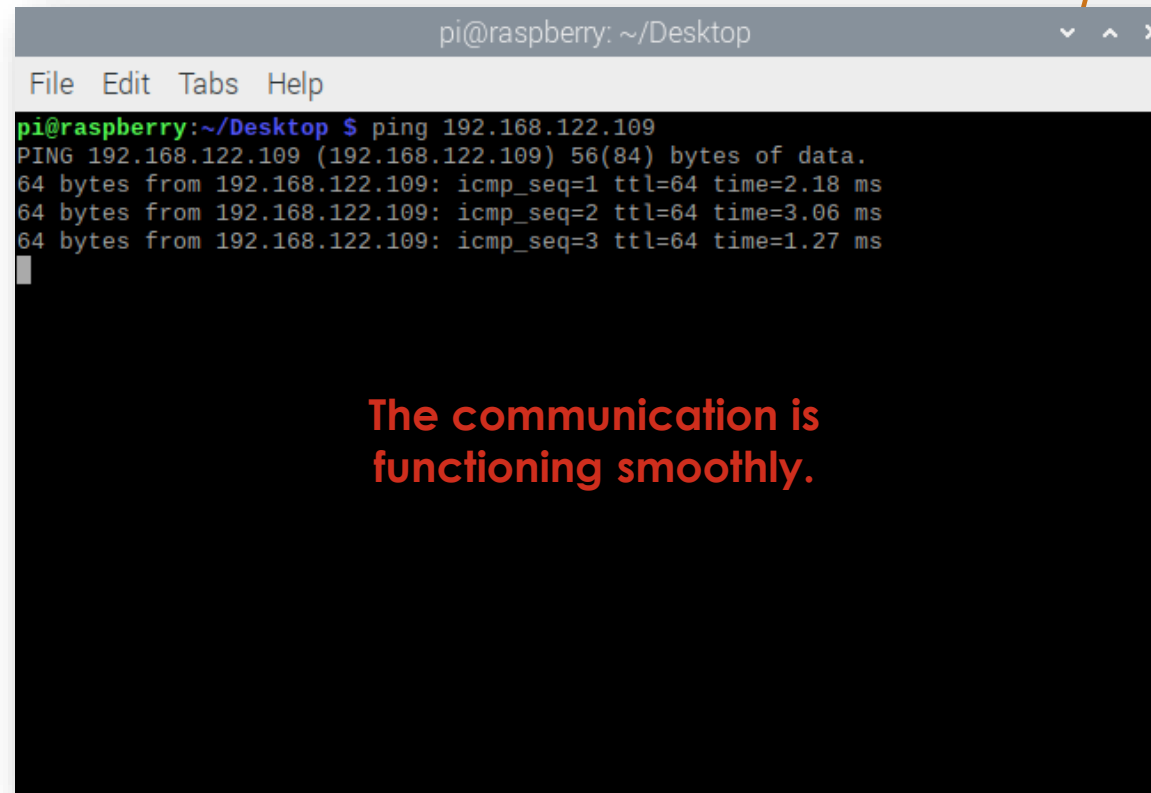
```
pi@raspberrypi: ~  
File Edit Tabs Help  
pi@raspberrypi:~$ sudo iptables -A INPUT -s 192.168.122.103 -j ACCEPT  
pi@raspberrypi:~$ sudo iptables -A INPUT -j DROP  
pi@raspberrypi:~$ sudo iptables -A OUTPUT -d 192.168.122.103 -j ACCEPT  
pi@raspberrypi:~$
```

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**.



```
pi@raspberrypi: ~/Desktop
File Edit Tabs Help
pi@raspberrypi:~/Desktop $ ping 192.168.122.109
PING 192.168.122.109 (192.168.122.109) 56(84) bytes of data:
64 bytes from 192.168.122.109: icmp_seq=1 ttl=64 time=2.18 ms
64 bytes from 192.168.122.109: icmp_seq=2 ttl=64 time=3.06 ms
64 bytes from 192.168.122.109: icmp_seq=3 ttl=64 time=1.27 ms
```

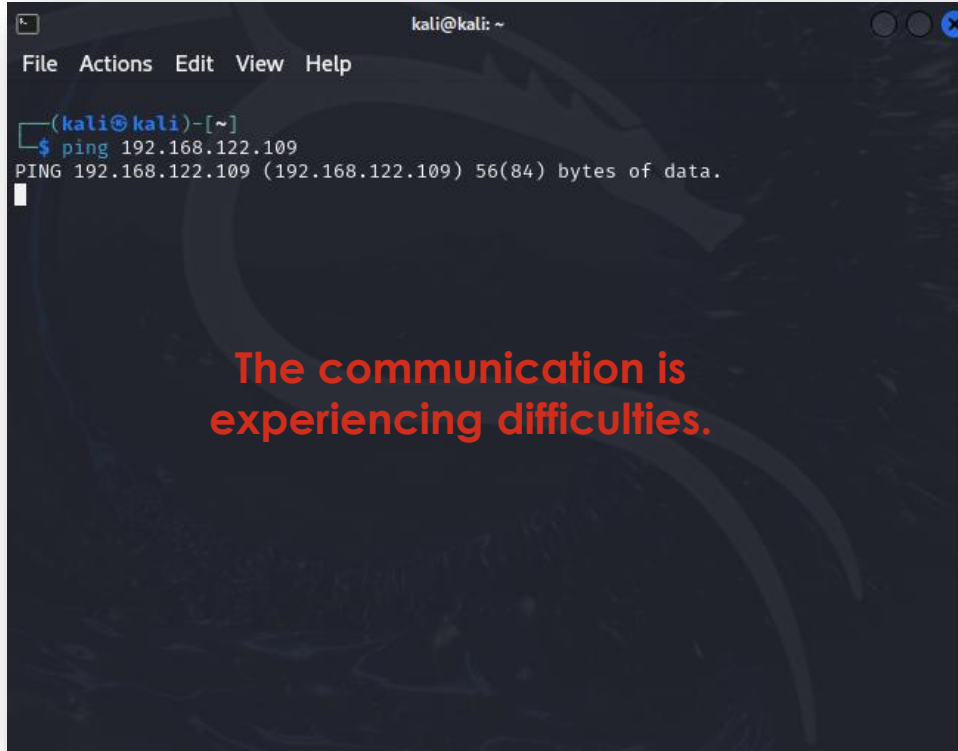
The communication is functioning smoothly.

Examples on the iptables

SERVER

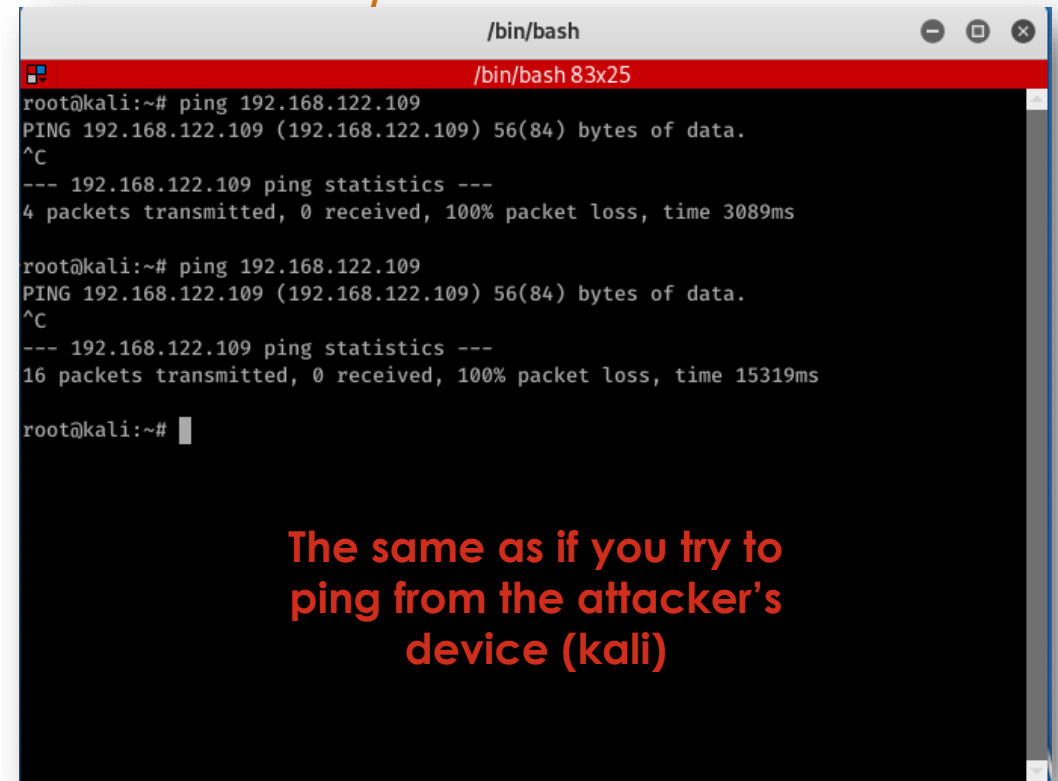
Allow all traffic from my authorized devices (only) in the network

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



```
kali@kali: ~  
File Actions Edit View Help  
(kali@kali)-[~]  
└─$ ping 192.168.122.109  
PING 192.168.122.109 (192.168.122.109) 56(84) bytes of data.  
|
```

The communication is experiencing difficulties.



```
/bin/bash  
root@kali:~# ping 192.168.122.109  
PING 192.168.122.109 (192.168.122.109) 56(84) bytes of data.  
^C  
--- 192.168.122.109 ping statistics ---  
4 packets transmitted, 0 received, 100% packet loss, time 3089ms  
  
root@kali:~# ping 192.168.122.109  
PING 192.168.122.109 (192.168.122.109) 56(84) bytes of data.  
^C  
--- 192.168.122.109 ping statistics ---  
16 packets transmitted, 0 received, 100% packet loss, time 15319ms  
  
root@kali:~# |
```

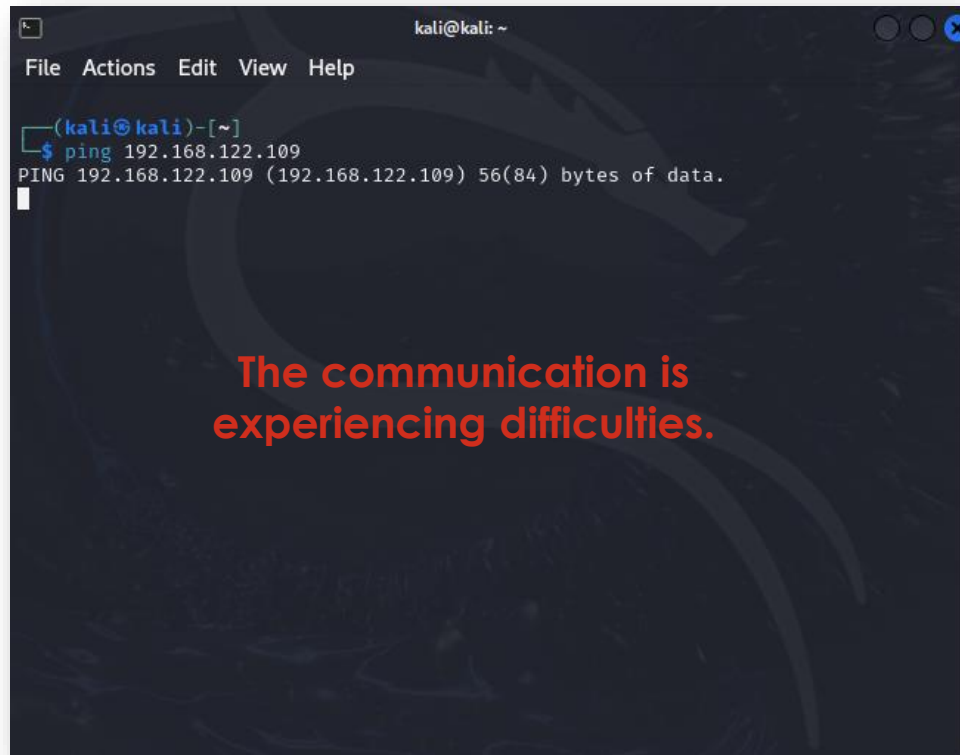
The same as if you try to ping from the attacker's device (kali)

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.



```
kali@kali: ~  
File Actions Edit View Help  
  
(kali@kali)-[~]  
└─$ ping 192.168.122.109  
PING 192.168.122.109 (192.168.122.109) 56(84) bytes of data.  
█
```

The communication is experiencing difficulties.

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"
```

Example 11.1.1

1. Use

2. Add

3. To

```
Raspberry Pi OS Client [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help

pi@raspberrypi: ~/Desktop
File Edit Tabs Help
-----
File "/usr/local/lib/python3.7/dist-packages/pymodbus/client/common.py", line 103, in write_registers
    return self.execute(request)
File "/usr/local/lib/python3.7/dist-packages/pymodbus/client/sync.py", line 108, in execute
    raise ConnectionException("Failed to connect[%s]" % (self.__str__()))
pymodbus.exceptions.ConnectionException: Modbus Error: [Connection] Failed to connect[ModbusTcpClient(192.168.122.109:502)]
pi@raspberrypi:~/Desktop $ sudo python3 client2.py
Start Modbus Client
----- Cycle 0 -----
Write [1.1, 2.1, 3.1, 4.1, 5.1]
Traceback (most recent call last):
  File "client2.py", line 32, in <module>
    skip_encode=True, unit=int(address))
  File "/usr/local/lib/python3.7/dist-packages/pymodbus/client/common.py", line 103, in write_registers
    return self.execute(request)
  File "/usr/local/lib/python3.7/dist-packages/pymodbus/client/sync.py", line 108, in execute
    raise ConnectionException("Failed to connect[%s]" % (self.__str__()))
pymodbus.exceptions.ConnectionException: Modbus Error: [Connection] Failed to connect[ModbusTcpClient(192.168.122.109:502)]
pi@raspberrypi:~/Desktop $
```

```
Raspberry Pi OS [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help

pi@raspberrypi: ~
pi@raspberrypi: ~/Desktop...

pi@raspberrypi: ~/Desktop
File Edit Tabs Help
-----
pi@raspberrypi:~ $ cd Desktop/
pi@raspberrypi:~/Desktop $ sudo python3 server2.py
Modbus server started on 192.168.122.109 port 502

pi@raspberrypi: ~
File Edit Tabs Help
-----
pi@raspberrypi:~ $ sudo iptables -A INPUT -j LOG --log-prefix "Dropped: any incoming packets"
pi@raspberrypi:~ $ sudo iptables -A INPUT -j DROP
pi@raspberrypi:~ $ sudo dmesg | grep "Dropped"
[ 6062.633652] Dropped: any incoming packetIN=eth0 OUT= MAC=08:00:27:52:a4:8f:52:54:00:1f:18:ec:08:00 SRC=193.171.23.163 DST=192.168.122.109 LEN=76 TOS=0x00 PREC=0x00 TTL=63 ID=23765 PROTO=UDP SPT=123 DPT=123 LEN=56
[ 6063.064951] Dropped: any incoming packetIN=lo OUT= MAC=00:00:00:00:00:00:00:00:00:00:00:00:00:08:00 SRC=192.168.122.109 DST=192.168.122.109 LEN=88 TOS=0x00 PREC=0x00 TTL=64 ID=53871 PROTO=ICMP TYPE=3 CODE=1 [SRC=192.168.122.109 DST=192.168.122.173 LEN=60 TOS=0x00 PREC=0x00 TTL=64 ID=58130 DF PROTO=TCP SPT=34990 DPT=1514 WINDOW=64240 RES=0x00 SYN URG=0 ]
[ 6066.192598] Dropped: any incoming packetIN=lo OUT= MAC=00:00:00:00:00:00:00:00:00:00:00:00:00:08:00 SRC=192.168.122.109 DST=192.168.122.109 LEN=88 TOS=0x00 PREC=0x00 TTL=64 ID=54067 PROTO=ICMP TYPE=3 CODE=1 [SRC=192.168.122.109 DST=192.168.122.173 LEN=60 TOS=0x00 PREC=0x00 TTL=64 ID=58131 DF PROTO=TCP SPT=34990 DPT=1514 WINDOW=64240 RES=0x00 SYN URG=0 ]
[ 6066.625864] Dropped: any incoming packetIN=eth0 OUT= MAC=08:00:27:52:a4:8f:52:54:00:1f:18:ec:08:00 SRC=185.144.161.170 DST=192.168.122.109 LEN=76 TOS=0x00 PREC=0x00 TTL=63 ID=23769 PROTO=UDP SPT=123 DPT=123 LEN=56
[ 6066.736785] Dropped: any incoming packetIN=eth0 OUT= MAC=08:00:27:52:a4:8f:52:54:00:1f:18:ec:08:00 SRC=192.168.122.1 DST=192.168.122.109 LEN=71 TOS=0x00 PREC=0x00 TTL=64 ID=17676 DF PROTO=UDP SPT=53 DPT=50684 LEN=51
[ 6066.737566] Dropped: any incoming packetIN=eth0 OUT= MAC=08:00:27:52:a4:8f:52:54:00:1f:18:ec:08:00 SRC=192.168.122.1 DST=192.168.122.109 LEN=55 TOS=0x00 PREC=0x00 TTL=64 ID=17677 DF PROTO=UDP SPT=53 DPT=50684 LEN=35
[ 6069.839157] Dropped: any incoming packetIN=eth0 OUT= MAC=08:00:27:52:a4:8f:52:54:00:1f:18:ec:08:00 SRC=91.206.8.36 DST=192.168.122.109 LEN=76 TOS=0x00 PREC=0x00 TTL=63 ID=23770 PROTO=UDP SPT=123 DPT=123 LEN=56
```

SERVER



Examples on the iptables

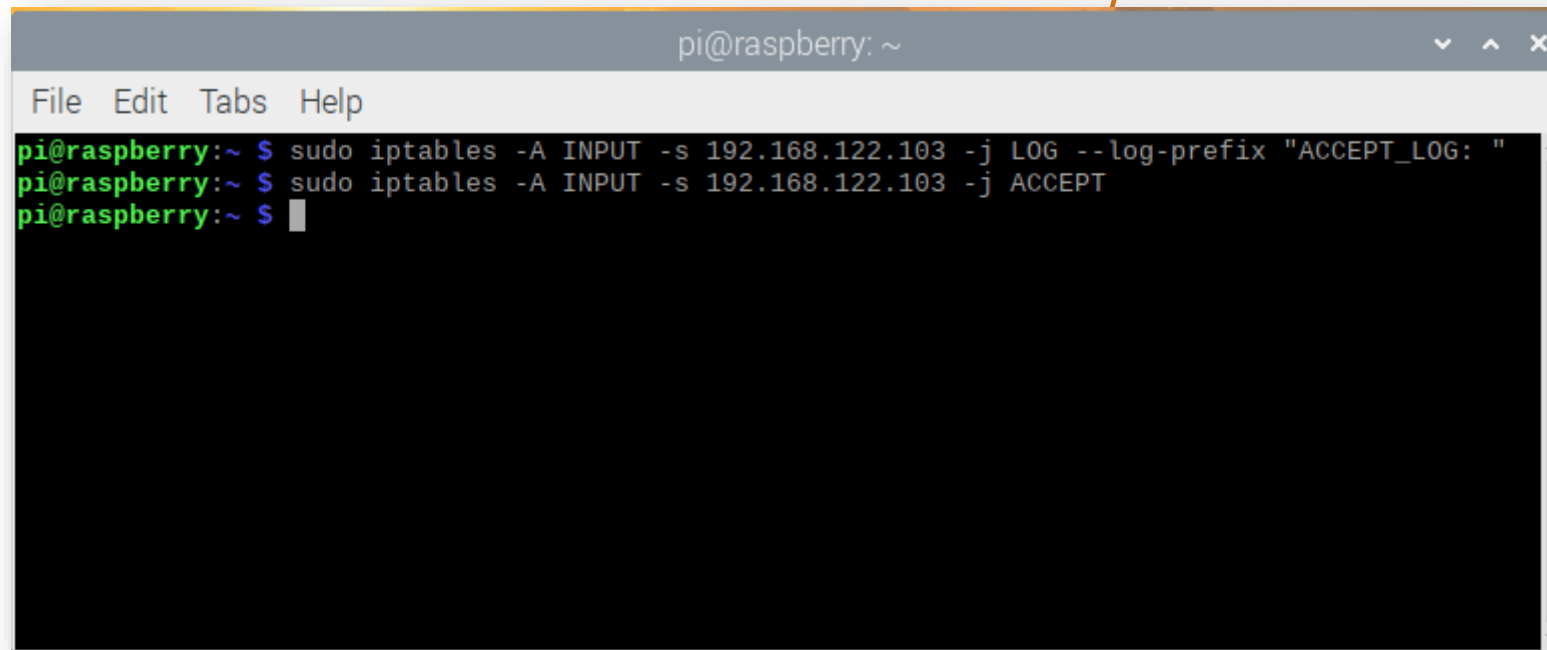
SERVER

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
```



```
pi@raspberrypi: ~  
File Edit Tabs Help  
pi@raspberrypi:~$ sudo iptables -A INPUT -s 192.168.122.103 -j LOG --log-prefix "ACCEPT_LOG: "  
pi@raspberrypi:~$ sudo iptables -A INPUT -s 192.168.122.103 -j ACCEPT  
pi@raspberrypi:~$
```

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
```



```
pi@raspberrypi: ~  
File Edit Tabs Help  
pi@raspberrypi:~$ sudo iptables -A INPUT -j LOG --log-prefix "DROP_LOG: "  
pi@raspberrypi:~$ sudo iptables -A INPUT -j DROP  
pi@raspberrypi:~$
```

Examples on the iptables

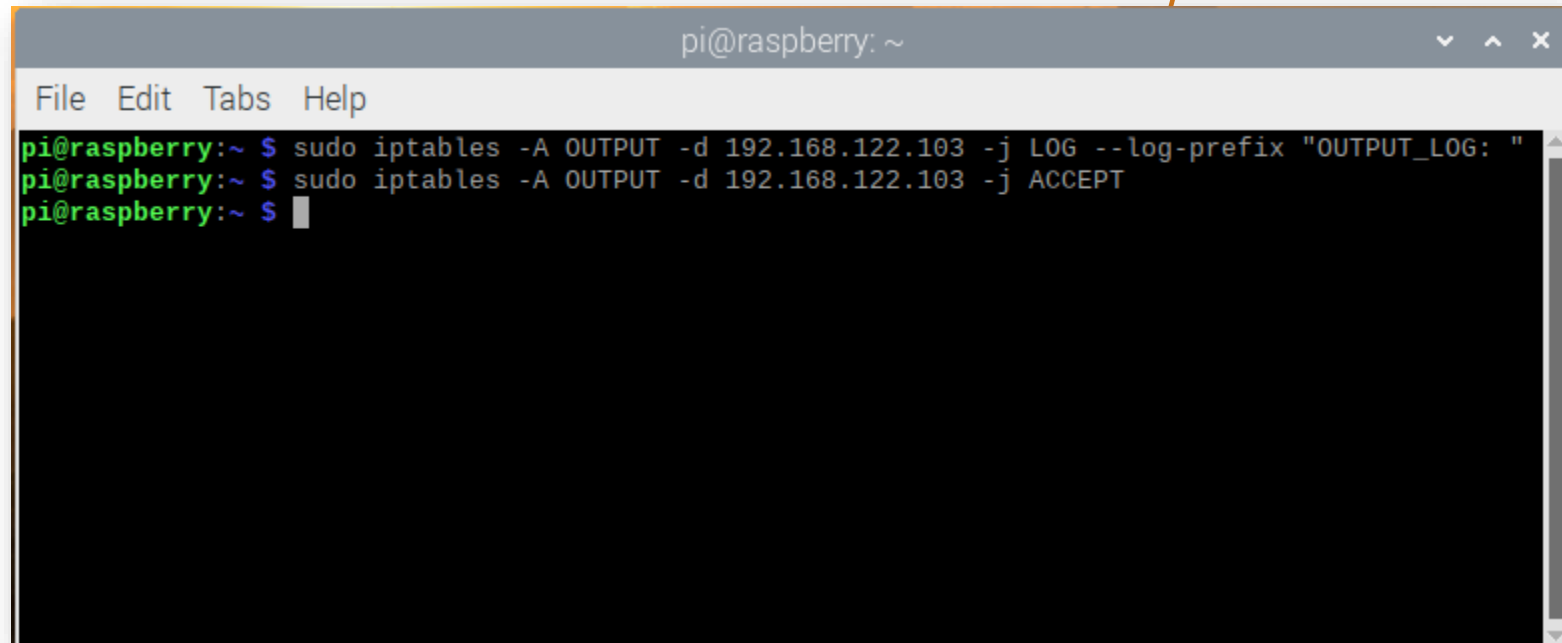
SERVER

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
```



```
pi@raspberrypi: ~  
File Edit Tabs Help  
pi@raspberrypi:~$ sudo iptables -A OUTPUT -d 192.168.122.103 -j LOG --log-prefix "OUTPUT_LOG: "  
pi@raspberrypi:~$ sudo iptables -A OUTPUT -d 192.168.122.103 -j ACCEPT  
pi@raspberrypi:~$
```

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
```

```

Sep 29 22:07:06 raspberry kernel: [ 1786.244595] DROP_LOG: IN=eth0 OUT= MAC=08:00:27:52:a4:8f:08:00:27:27:29:8c:08:00 SRC=192.168.122.27 DST=192.168.122.109 LEN=84 TOS=0x00 PREC=0x00 TTL=64 ID=30905 DF PROTO=ICMP TYPE=8 CODE=0 ID=1865 SEQ=3
Sep 29 22:07:06 raspberry kernel: [ 1786.244739] DROP_LOG: IN=eth0 OUT= MAC=08:00:27:52:a4:8f:08:00:27:27:29:8c:08:00 SRC=192.168.122.27 DST=192.168.122.109 LEN=84 TOS=0x00 PREC=0x00 TTL=64 ID=30905 DF PROTO=ICMP TYPE=8 CODE=0 ID=1865 SEQ=3
Sep 29 22:07:06 raspberry kernel: [ 1786.937641] ACCEPT_LOG: IN=eth0 OUT= MAC=08:00:27:52:a4:8f:08:00:27:30:20:e0:08:00 SRC=192.168.122.103 DST=192.168.122.109 LEN=84 TOS=0x00 PREC=0x00 TTL=64 ID=46093 DF PROTO=ICMP TYPE=8 CODE=0 ID=2345 SEQ=66
Sep 29 22:07:06 raspberry kernel: [ 1786.937732] OUTPUT_LOG: IN= OUT=eth0 SRC=192.168.122.109 DST=192.168.122.103 LEN=84 TOS=0x00 PREC=0x00 TTL=64 ID=53196 PROTO=ICMP TYPE=0 CODE=0 ID=2345 SEQ=66
Sep 29 22:07:07 raspberry kernel: [ 1787.274319] DROP_LOG: IN=eth0 OUT= MAC=08:00:27:52:a4:8f:08:00:27:27:29:8c:08:00 SRC=192.168.122.27 DST=192.168.122.109 LEN=84 TOS=0x00 PREC=0x00 TTL=64 ID=31080 DF PROTO=ICMP TYPE=8 CODE=0 ID=1865 SEQ=4
Sep 29 22:07:07 raspberry kernel: [ 1787.274407] DROP_LOG: IN=eth0 OUT= MAC=08:00:27:52:a4:8f:08:00:27:27:29:8c:08:00 SRC=192.168.122.27 DST=192.168.122.109 LEN=84 TOS=0x00 PREC=0x00 TTL=64 ID=31080 DF PROTO=ICMP TYPE=8 CODE=0 ID=1865 SEQ=4
Sep 29 22:07:07 raspberry kernel: [ 1787.274459] DROP_LOG: IN=eth0 OUT= MAC=08:00:27:52:a4:8f:08:00:27:27:29:8c:08:00 SRC=192.168.122.27 DST=192.168.122.109 LEN=84 TOS=0x00 PREC=0x00 TTL=64 ID=31080 DF PROTO=ICMP TYPE=8 CODE=0 ID=1865 SEQ=4
Sep 29 22:07:07 raspberry kernel: [ 1788.012180] ACCEPT_LOG: IN=eth0 OUT= MAC=08:00:27:52:a4:8f:08:00:27:30:20:e0:08:00 SRC=192.168.122.103 DST=192.168.122.109 LEN=84 TOS=0x00 PREC=0x00 TTL=64 ID=46214 DF PROTO=ICMP TYPE=8 CODE=0 ID=2345 SEQ=67
Sep 29 22:07:07 raspberry kernel: [ 1788.012345] OUTPUT_LOG: IN= OUT=eth0 SRC=192.168.122.109 DST=192.168.122.103 LEN=84 TOS=0x00 PREC=0x00 TTL=64 ID=53366 PROTO=ICMP TYPE=0 CODE=0 ID=2345 SEQ=67
Sep 29 22:07:08 raspberry kernel: [ 1788.290333] DROP_LOG: IN=eth0 OUT= MAC=08:00:27:52:a4:8f:08:00:27:27:29:8c:08:00 SRC=192.168.122.27 DST=192.168.122.109 LEN=84 TOS=0x00 PREC=0x00 TTL=64 ID=31232 DF PROTO=ICMP TYPE=8 CODE=0 ID=1865 SEQ=5
Sep 29 22:07:08 raspberry kernel: [ 1788.290439] DROP_LOG: IN=eth0 OUT= MAC=08:00:27:52:a4:8f:08:00:27:27:29:8c:08:00 SRC=192.168.122.27 DST=192.168.122.109 LEN=84 TOS=0x00 PREC=0x00 TTL=64 ID=31232 DF PROTO=ICMP TYPE=8 CODE=0 ID=1865 SEQ=5
Sep 29 22:07:08 raspberry kernel: [ 1788.290491] DROP_LOG: IN=eth0 OUT= MAC=08:00:27:52:a4:8f:08:00:27:27:29:8c:08:00 SRC=192.168.122.27 DST=192.168.122.109 LEN=84 TOS=0x00 PREC=0x00 TTL=64 ID=31232 DF PROTO=ICMP TYPE=8 CODE=0 ID=1865 SEQ=5

```

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 attacker 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

```

pi@raspberrypi: ~/Desktop
File Edit Tabs Help
pi@raspberrypi:~/Desktop $ ss -a
Netid      State      Recv-Q     Send-Q     Peer Address:Port      Local Address:Port
nl         UNCONN     0           0           *                    rtnl:kernel
nl         UNCONN     0           0           *                    rtnl:465
nl         UNCONN     0           0           *                    rtnl:ntpd/64
nl         UNCONN     0           0           *                    rtnl:-239474
nl         UNCONN     0           0           *                    rtnl:avahi-d
aemon/450  UNCONN     0           0           *                    rtnl:ntpd/64
nl         UNCONN     0           0           *                    rtnl:avahi-d
aemon/450  UNCONN     0           0           *                    rtnl:465
nl         UNCONN     768         0           *                    tcpdiag:kernel
nl         UNCONN     4352        0           *                    tcpdiag:ss/2871
nl         UNCONN     0           0           *                    selinux:kernel
nl         UNCONN     0           0           *                    audit:kernel
nl         UNCONN     0           0           *                    audit:-732852
nl         UNCONN     0           0           *                    audit:systemd
nl         UNCONN     0           0           *                    audit:systemd

```

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

```

pi@raspberrypi: ~/Desktop
File Edit Tabs Help
*:*
udp UNCONN 0 0 0.0.0.0:* 0.0.0.0:bootpc
udp UNCONN 0 0 0.0.0.0:* 192.168.122.109:ntp
udp UNCONN 0 0 0.0.0.0:* 127.0.0.1:ntp
udp UNCONN 0 0 0.0.0.0:* 0.0.0.0:ntp
udp UNCONN 0 0 0.0.0.0:* 0.0.0.0:ipp
udp UNCONN 0 0 0.0.0.0:* 0.0.0.0:mdns
udp UNCONN 0 0 0.0.0.0:* 0.0.0.0:58714
udp UNCONN 0 0 [fe80::949a:cd6a:b651:3d18]%eth0:ntp
udp UNCONN 0 0 [::]:* [::1]:ntp
udp UNCONN 0 0 [::]:* [::]:ntp
udp UNCONN 0 0 [::]:* [::]:35421
udp UNCONN 0 0 [::]:* [::]:mdns
tcp LISTEN 0 5 0.0.0.0:* 192.168.122.109:502
tcp LISTEN 0 5 0.0.0.0:* 127.0.0.1:ipp
tcp LISTEN 0 20 0.0.0.0:* 127.0.0.1:smtp
tcp LISTEN 0 5 0.0.0.0:* [::1]:ipp

```

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.

```

pi@raspberrypi: ~/Desktop
File Edit Tabs Help
pi@raspberrypi:~/Desktop $ ss -tuln
Netid State Recv-Q Send-Q Local Address:Port Peer Address:Port
udp UNCONN 0 0 0.0.0.0:68 0.0.0.0:*
udp UNCONN 0 0 192.168.122.109:123 0.0.0.0:*
udp UNCONN 0 0 127.0.0.1:123 0.0.0.0:*
udp UNCONN 0 0 0.0.0.0:123 0.0.0.0:*
udp UNCONN 0 0 0.0.0.0:631 0.0.0.0:*
udp UNCONN 0 0 0.0.0.0:5353 0.0.0.0:*
udp UNCONN 0 0 0.0.0.0:58714 0.0.0.0:*
udp UNCONN 0 0 [fe80::949a:cd6a:b651:3d18]::%eth0:123 [::]:*
udp UNCONN 0 0 [::1]:123 [::]:*
udp UNCONN 0 0 [::]:123 [::]:*
udp UNCONN 0 0 [::]:35421 [::]:*
udp UNCONN 0 0 [::]:5353 [::]:*
tcp LISTEN 0 5 192.168.122.109:502 0.0.0.0:*
tcp LISTEN 0 5 127.0.0.1:631 0.0.0.0:*
tcp LISTEN 0 20 127.0.0.1:25 0.0.0.0:*
tcp LISTEN 0 5 [::1]:631 [::]:*
tcp LISTEN 0 20 [::1]:25 [::]:*
pi@raspberrypi:~/Desktop $

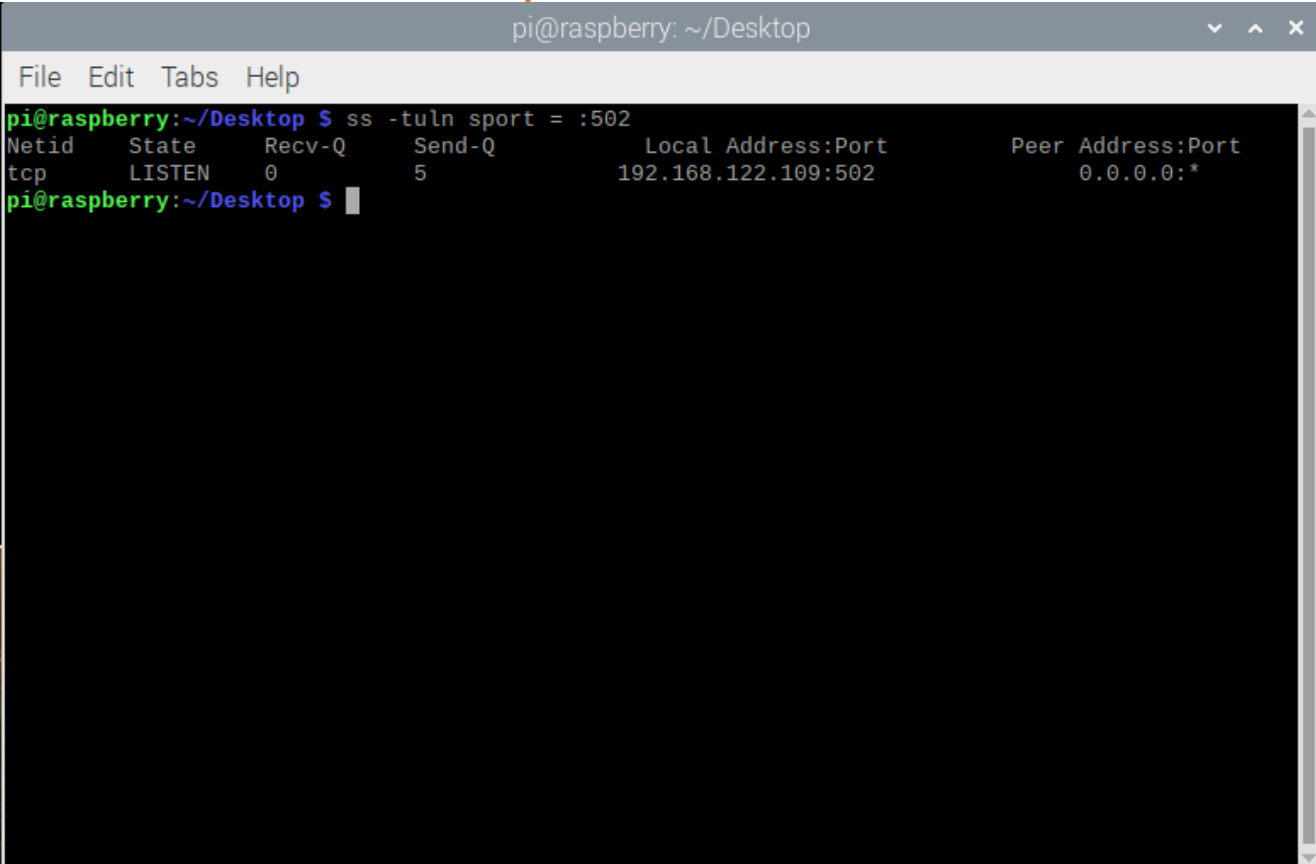
```

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
```



```
pi@raspberrypi: ~/Desktop
File Edit Tabs Help
pi@raspberrypi:~/Desktop $ ss -tuln sport = :502
Netid  State  Recv-Q  Send-Q  Local Address:Port  Peer Address:Port
tcp    LISTEN  0        5       192.168.122.109:502  0.0.0.0:*
pi@raspberrypi:~/Desktop $
```

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:

```

pi@raspberrypi: ~/Desktop
File Edit Tabs Help
pi@raspberrypi:~/Desktop $ ss -4
Netid  State      Recv-Q    Send-Q      Local Address:Port      Peer Address:Port
tcp    SYN-SENT  0         1           192.168.122.109:47594    192.168.122.173:1514
pi@raspberrypi:~/Desktop $ ss -6
Netid  State      Recv-Q    Send-Q      Local Address:Port      Peer Address:Port
icmp6  UNCONN    0         0           *:ipv6-icmp             *:*
```

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

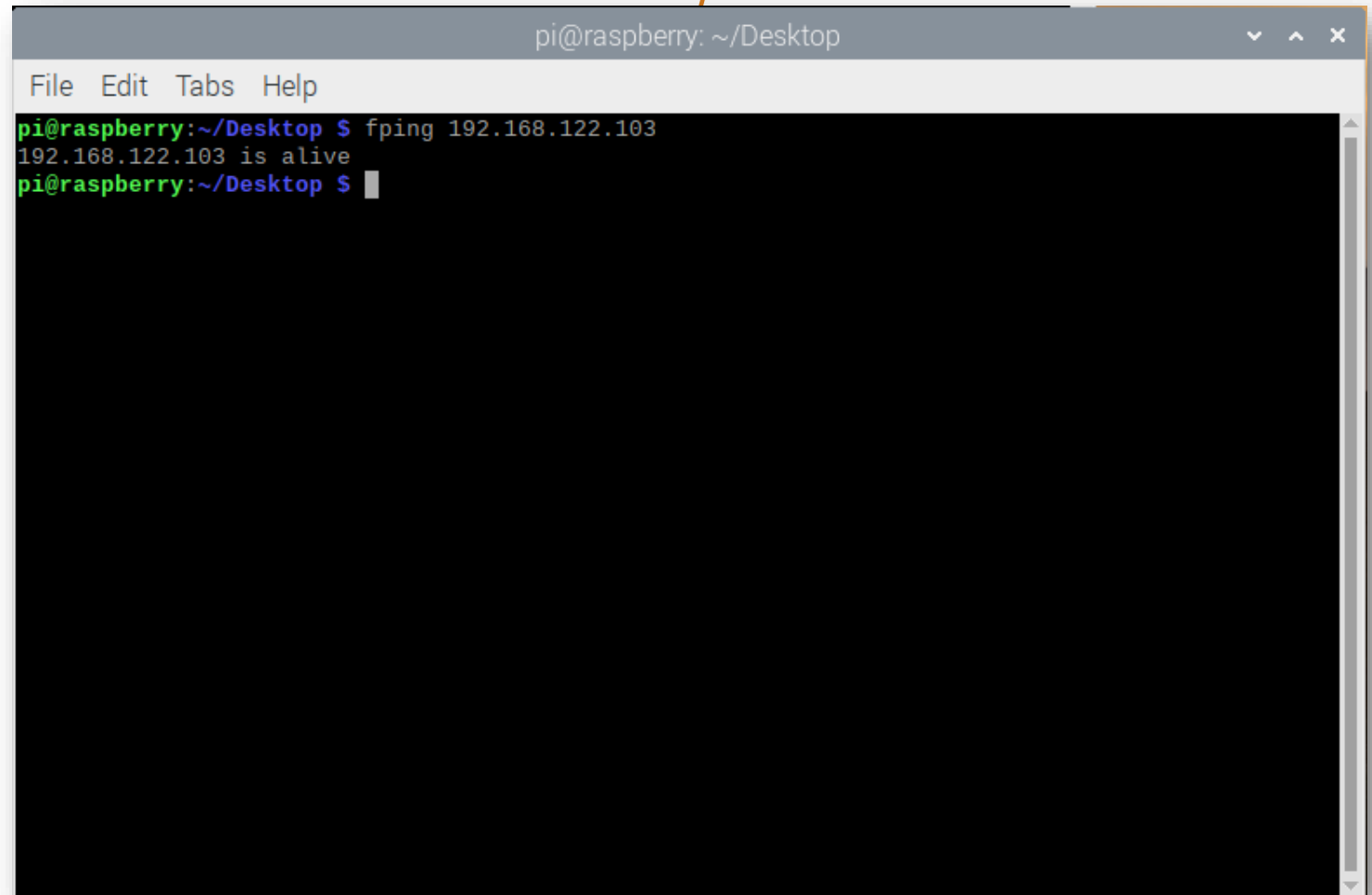
```
pi@raspberrypi: ~/Desktop
File Edit Tabs Help
pi@raspberrypi:~/Desktop $ sudo apt install fping
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer required:
  python-colorzero
Use 'sudo apt autoremove' to remove it.
The following NEW packages will be installed:
  fping
0 upgraded, 1 newly installed, 0 to remove and 33 not upgraded.
Need to get 38.7 kB of archives.
After this operation, 87.0 kB of additional disk space will be used.
Get:1 http://ftp.debian.org/debian buster/main i386 fping i386 4.2-1 [38.7 kB]
Fetched 38.7 kB in 1s (27.6 kB/s)
Selecting previously unselected package fping.
(Reading database ... 170892 files and directories currently installed.)
Preparing to unpack ../archives/fping_4.2-1_i386.deb ...
Unpacking fping (4.2-1) ...
Setting up fping (4.2-1) ...
Processing triggers for man-db (2.8.5-2+deb10u1) ...
pi@raspberrypi:~/Desktop $
```

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
```

A terminal window titled 'pi@raspberrypi: ~/Desktop' showing the execution of the 'fping' command. The prompt is 'pi@raspberrypi:~/Desktop \$'. The command entered is 'fping 192.168.122.103'. The output is '192.168.122.103 is alive'. The prompt returns to 'pi@raspberrypi:~/Desktop \$' with a cursor.

```
pi@raspberrypi:~/Desktop $ fping 192.168.122.103
192.168.122.103 is alive
pi@raspberrypi:~/Desktop $
```

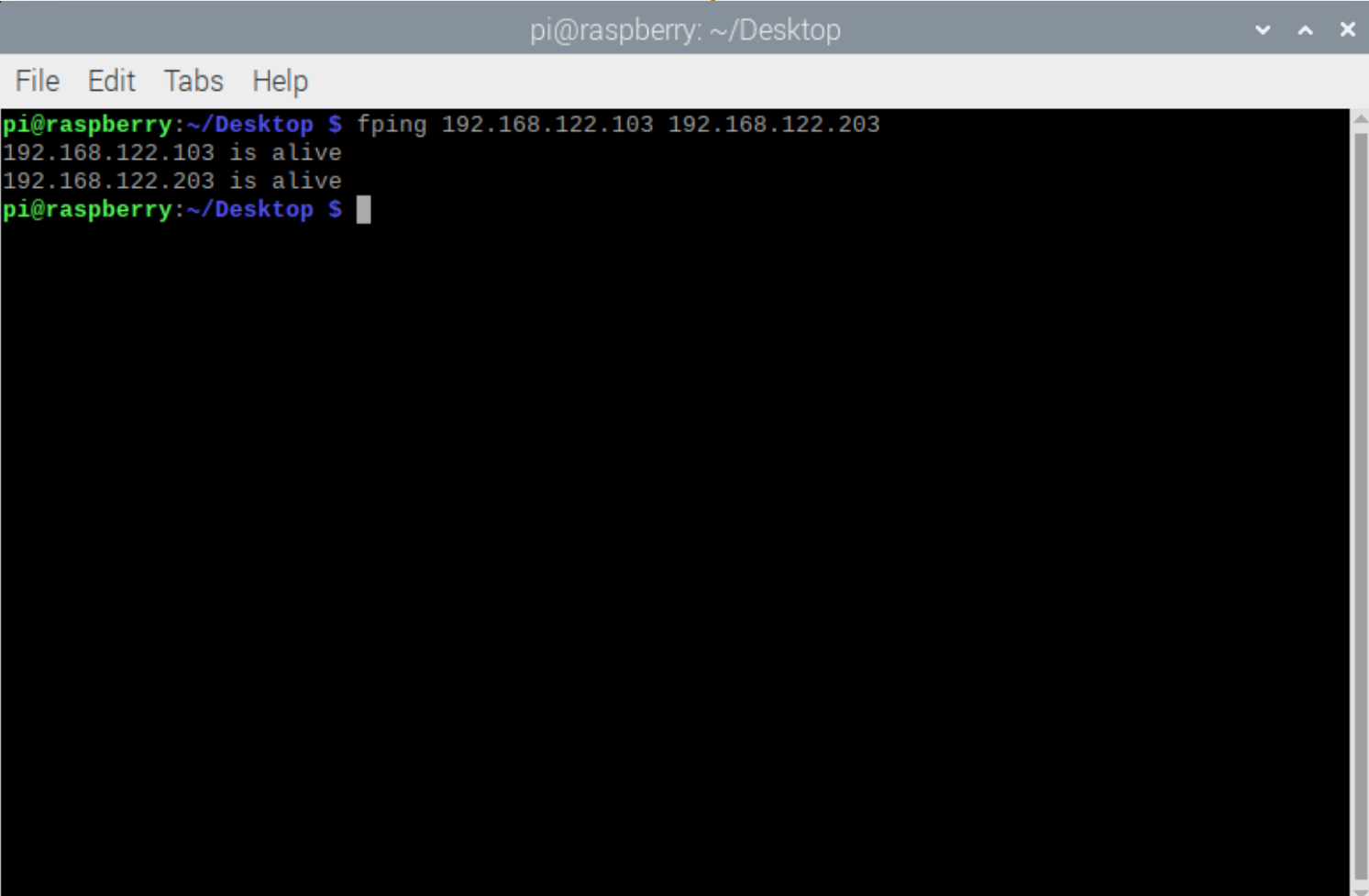
Examples on fping

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.

A terminal window titled 'pi@raspberrypi: ~/Desktop' with a menu bar containing 'File', 'Edit', 'Tabs', and 'Help'. The terminal shows the command 'fping 192.168.122.103 192.168.122.203' being executed. The output shows '192.168.122.103 is alive' and '192.168.122.203 is alive' on separate lines. The prompt 'pi@raspberrypi:~/Desktop \$' is visible at the end of the output.

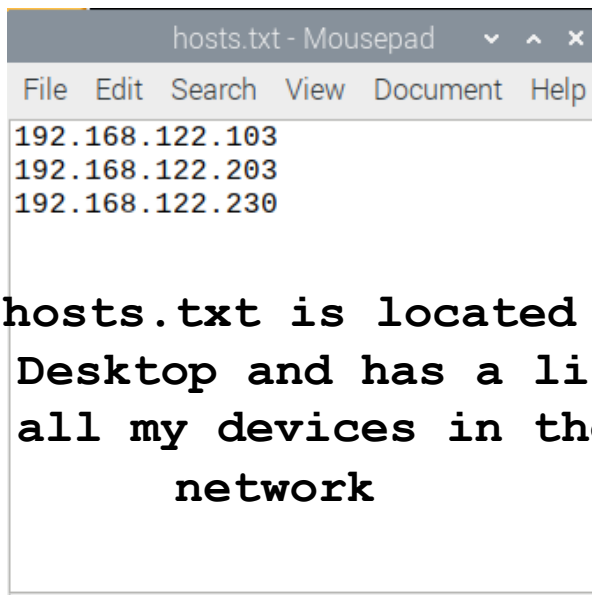
```
pi@raspberrypi:~/Desktop $ fping 192.168.122.103 192.168.122.203  
192.168.122.103 is alive  
192.168.122.203 is alive  
pi@raspberrypi:~/Desktop $
```

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.

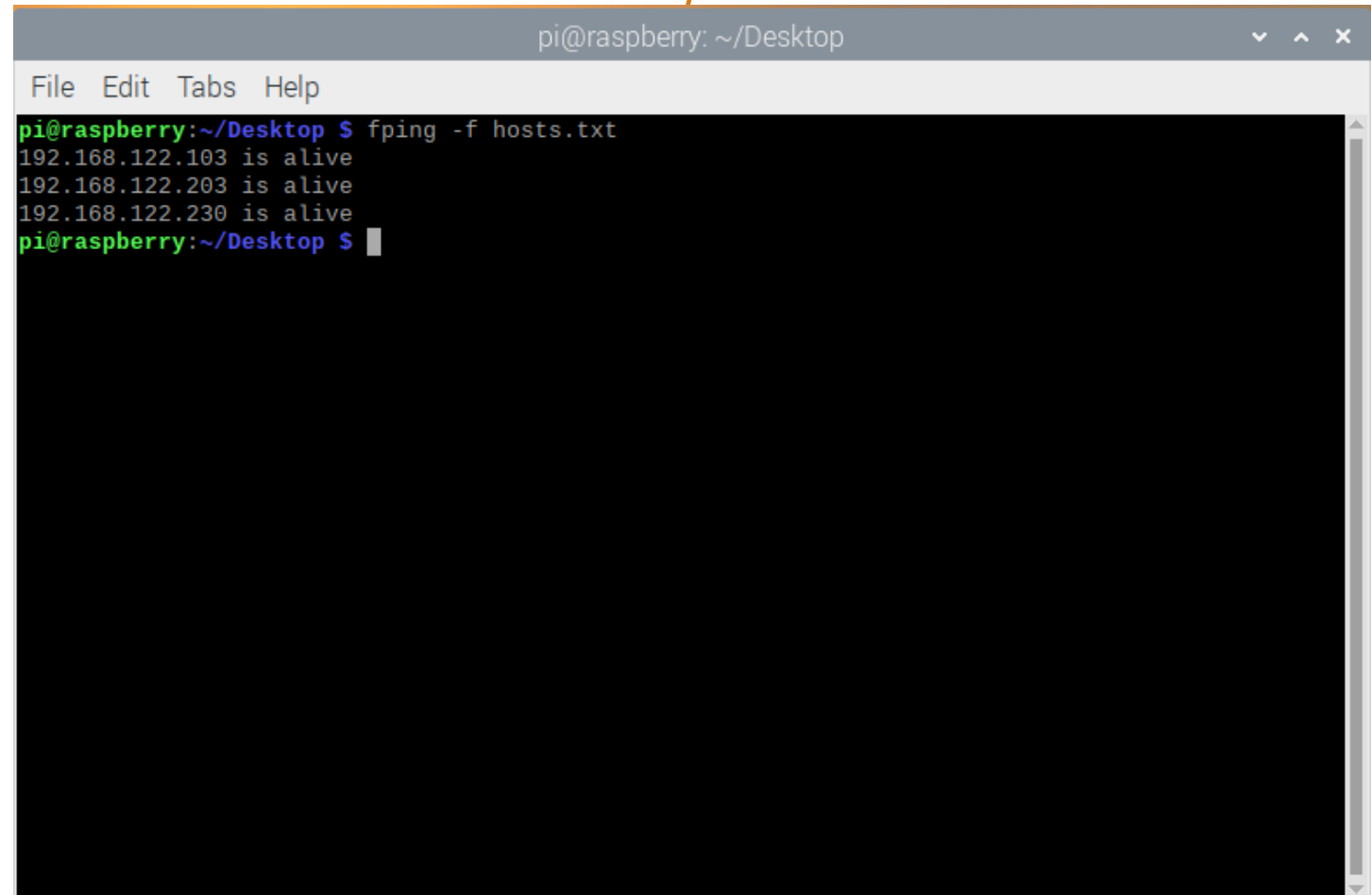
Ping multiple hosts from a file

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



```
hosts.txt - Mousepad
File Edit Search View Document Help
192.168.122.103
192.168.122.203
192.168.122.230
```

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



```
pi@raspberrypi: ~/Desktop
File Edit Tabs Help
pi@raspberrypi:~/Desktop $ fping -f hosts.txt
192.168.122.103 is alive
192.168.122.203 is alive
192.168.122.230 is alive
pi@raspberrypi:~/Desktop $
```

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.

```
pi@raspberrypi: ~/Desktop
File Edit Tabs Help
pi@raspberrypi:~/Desktop $ fping -g 192.168.122.0/24
192.168.122.1 is alive
192.168.122.27 is alive
192.168.122.103 is alive
192.168.122.109 is alive
192.168.122.203 is alive
192.168.122.230 is alive
ICMP Host Unreachable from 192.168.122.109 for ICMP Echo sent to 192.168.122.173
ICMP Host Unreachable from 192.168.122.109 for ICMP Echo sent to 192.168.122.2
ICMP Host Unreachable from 192.168.122.109 for ICMP Echo sent to 192.168.122.2
ICMP Host Unreachable from 192.168.122.109 for ICMP Echo sent to 192.168.122.4
ICMP Host Unreachable from 192.168.122.109 for ICMP Echo sent to 192.168.122.4
ICMP Host Unreachable from 192.168.122.109 for ICMP Echo sent to 192.168.122.3
ICMP Host Unreachable from 192.168.122.109 for ICMP Echo sent to 192.168.122.3
ICMP Host Unreachable from 192.168.122.109 for ICMP Echo sent to 192.168.122.7
ICMP Host Unreachable from 192.168.122.109 for ICMP Echo sent to 192.168.122.7
ICMP Host Unreachable from 192.168.122.109 for ICMP Echo sent to 192.168.122.6
ICMP Host Unreachable from 192.168.122.109 for ICMP Echo sent to 192.168.122.6
ICMP Host Unreachable from 192.168.122.109 for ICMP Echo sent to 192.168.122.5
ICMP Host Unreachable from 192.168.122.109 for ICMP Echo sent to 192.168.122.5
ICMP Host Unreachable from 192.168.122.109 for ICMP Echo sent to 192.168.122.10
ICMP Host Unreachable from 192.168.122.109 for ICMP Echo sent to 192.168.122.10
ICMP Host Unreachable from 192.168.122.109 for ICMP Echo sent to 192.168.122.9
ICMP Host Unreachable from 192.168.122.109 for ICMP Echo sent to 192.168.122.9
ICMP Host Unreachable from 192.168.122.109 for ICMP Echo sent to 192.168.122.8
ICMP Host Unreachable from 192.168.122.109 for ICMP Echo sent to 192.168.122.8
ICMP Host Unreachable from 192.168.122.109 for ICMP Echo sent to 192.168.122.13
ICMP Host Unreachable from 192.168.122.109 for ICMP Echo sent to 192.168.122.13
```

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.

```
pi@raspberrypi: ~/Desktop
File Edit Tabs Help
pi@raspberrypi:~/Desktop $ fping -c 5 -i 100 192.168.122.103
192.168.122.103 : [0], 84 bytes, 2.17 ms (2.17 avg, 0% loss)
192.168.122.103 : [1], 84 bytes, 1.95 ms (2.06 avg, 0% loss)
192.168.122.103 : [2], 84 bytes, 1.37 ms (1.83 avg, 0% loss)
192.168.122.103 : [3], 84 bytes, 1.44 ms (1.73 avg, 0% loss)
192.168.122.103 : [4], 84 bytes, 1.74 ms (1.73 avg, 0% loss)
192.168.122.103 : xmt/rcv/%loss = 5/5/0%, min/avg/max = 1.37/1.73/2.17
pi@raspberrypi:~/Desktop $
```

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

```

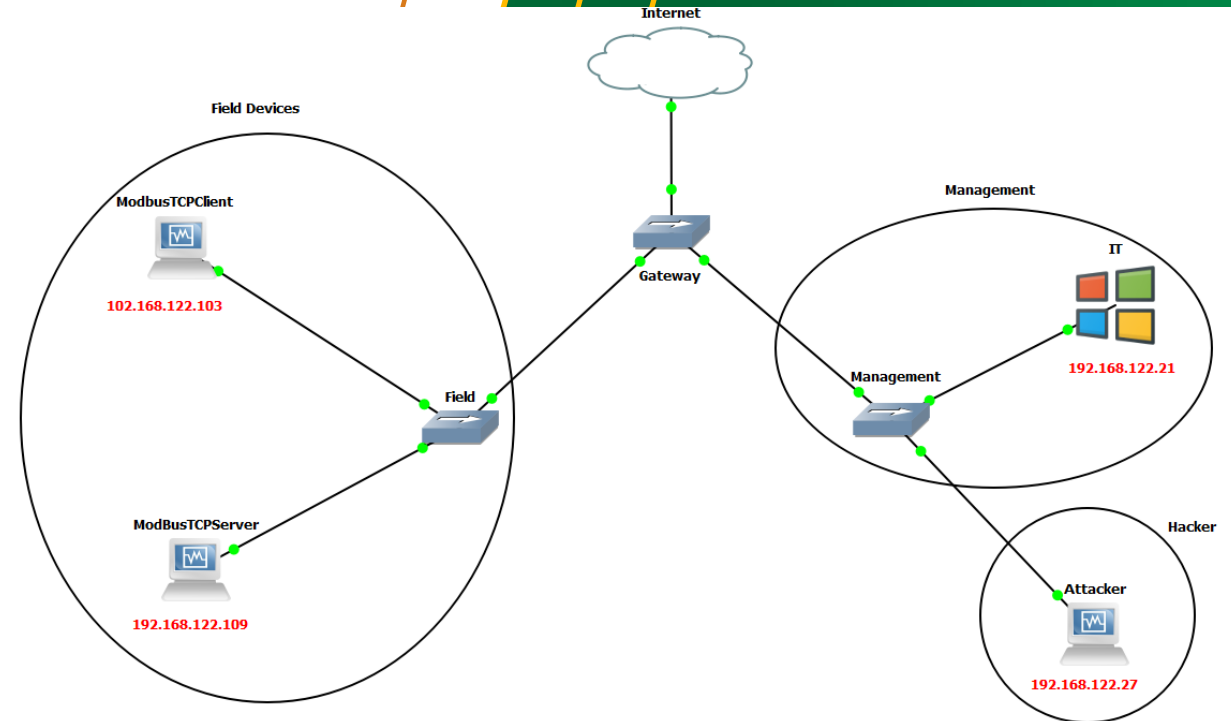
Command Prompt
Interface: 192.168.122.21 --- 0x4
Internet Address    Physical Address    Type
192.168.122.1      52-54-00-1f-18-ec  dynamic
192.168.122.103    08-00-27-30-20-e0  dynamic
192.168.122.255    ff-ff-ff-ff-ff-ff  static
224.0.0.22         01-00-5e-00-00-16  static
224.0.0.251        01-00-5e-00-00-fb  static
224.0.0.252        01-00-5e-00-00-fc  static
239.255.255.250    01-00-5e-7f-ff-fa  static
255.255.255.255    ff-ff-ff-ff-ff-ff  static

C:\Users\User>arp -a

Interface: 192.168.122.21 --- 0x4
Internet Address    Physical Address    Type
192.168.122.1      08-00-27-27-29-8c  dynamic
192.168.122.27     08-00-27-27-29-8c  dynamic
192.168.122.103    08-00-27-30-20-e0  dynamic
192.168.122.255    ff-ff-ff-ff-ff-ff  static
224.0.0.22         01-00-5e-00-00-16  static
224.0.0.251        01-00-5e-00-00-fb  static
224.0.0.252        01-00-5e-00-00-fc  static
239.255.255.250    01-00-5e-7f-ff-fa  static
255.255.255.255    ff-ff-ff-ff-ff-ff  static

C:\Users\User>
  
```

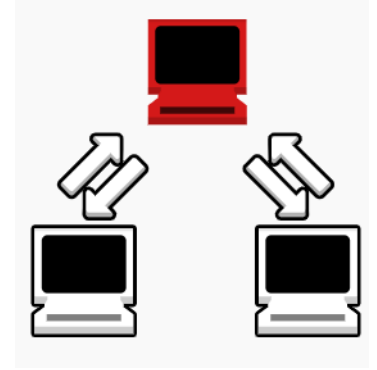
After



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, which will scan for common local networks.



```

root@kali: ~
File Actions Edit View Help Analyze Statistics Telephony Wireless Tools Help
(root@kali)-[~]
# netdiscover -help
Netdiscover 0.10 [Active/passive ARP reconnaissance tool]
Written by: Jaime Penalba <jpenalbae@gmail.com>

Usage: netdiscover [-i device] [-r range | -l file | -p] [-m file] [-F filter] [-s time] [-c count] [-n node] [--dfPLNS]
-i device: your network device
-r range: scan a given range instead of auto scan. 192.168.6.0/24,/16,/8
-l file: scan the list of ranges contained into the given file
-p passive mode: do not send anything, only sniff
-m file: scan a list of known MACs and host names
-F filter: customize pcap filter expression (default: "arp")
-s time: time to sleep between each ARP request (milliseconds)
-c count: number of times to send each ARP request (for nets with packet loss)
-n node: last source IP octet used for scanning (from 2 to 253)
-d ignore home config files for autoscan and fast mode
-f enable fastmode scan, saves a lot of time, recommended for auto
-P print results in a format suitable for parsing by another program and stop after active scan
-L similar to -P but continue listening after the active scan is completed
-N Do not print header. Only valid when -P or -L is enabled.
-S enable sleep time suppression between each request (hardcore mode)

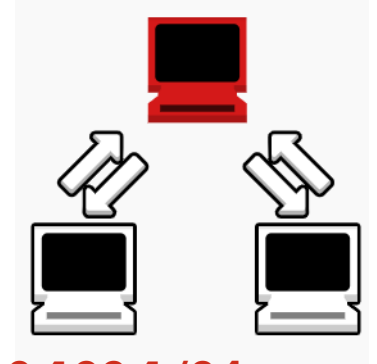
If -r, -l or -p are not enabled, netdiscover will scan for common LAN addresses.

(root@kali)-[~]
#

```

Detection of Cyberattacks

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



Arp -a

```

pi@raspberrypi:~$ 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@raspberrypi:~$
  
```

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

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

```

pi@raspberrypi: ~
File Edit Tabs Help
pi@raspberrypi:~$ arp -a
? (192.168.122.15) at 08:00:27:21:b1:d0 [ether] on eth0
_gateway (192.168.122.1) at 08:00:27:27:29:8c [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@raspberrypi:~$
  
```

The devices with IPs ending in **.1** and **.27** have the same MAC address.

Server

netdiscover -i eth0 -r 192.168.122.1/24

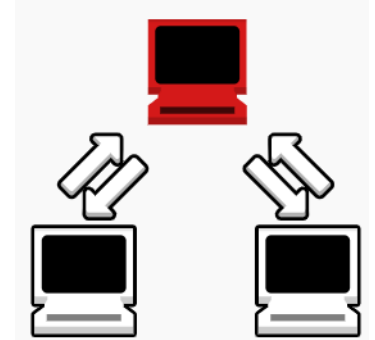
```

root@kali: ~
File Actions Edit View Help
Currently scanning: Finished! | Screen View: Unique Hosts
304 Captured ARP Req/Rep packets, from 5 hosts. Total size: 18240

IP           At MAC Address  Count  Len  MAC Vendor / Hostname
-----
192.168.122.109 08:00:27:52:a4:8f  258   15480 PCS Systemtechnik GmbH
192.168.122.1   52:54:00:1f:18:ec   21    1260  Unknown vendor
192.168.122.27  08:00:27:27:29:8c    1     60   PCS Systemtechnik GmbH
192.168.122.109 08:00:27:27:29:8c   13    780   PCS Systemtechnik GmbH
192.168.122.1   08:00:27:27:29:8c   11    660   PCS Systemtechnik GmbH
  
```

The tool detects the ARP spoofing

Admin Kali

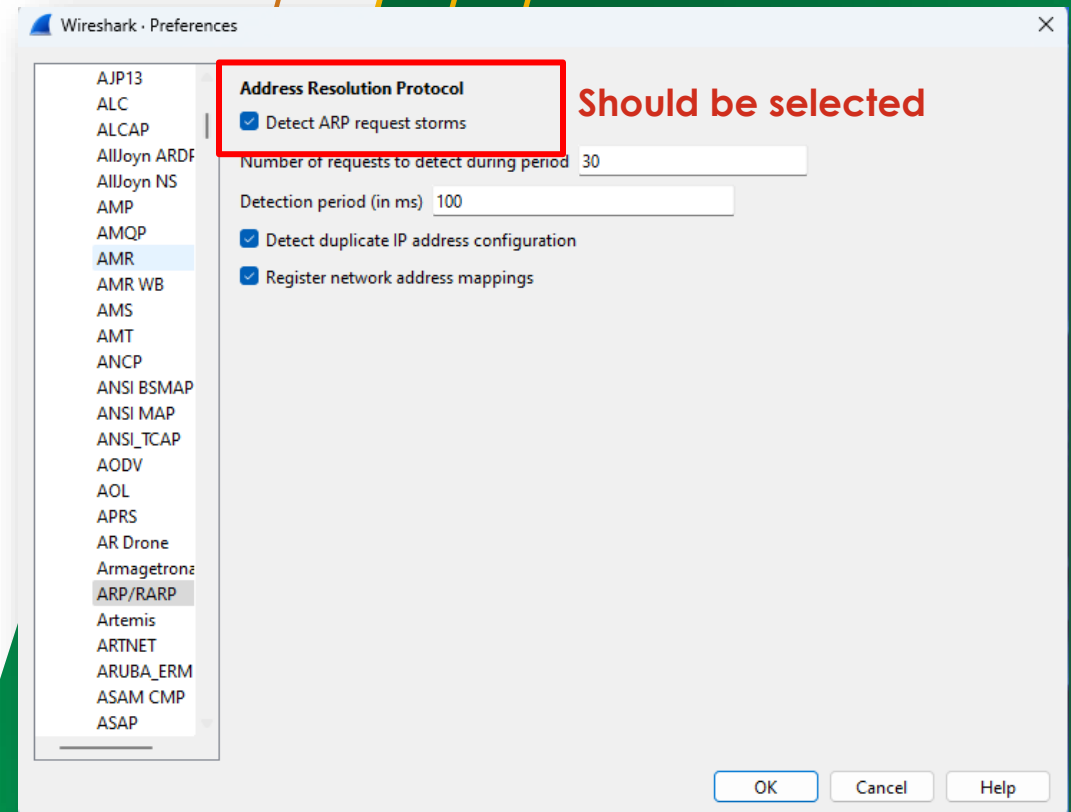


Detection of Cyberattacks

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

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

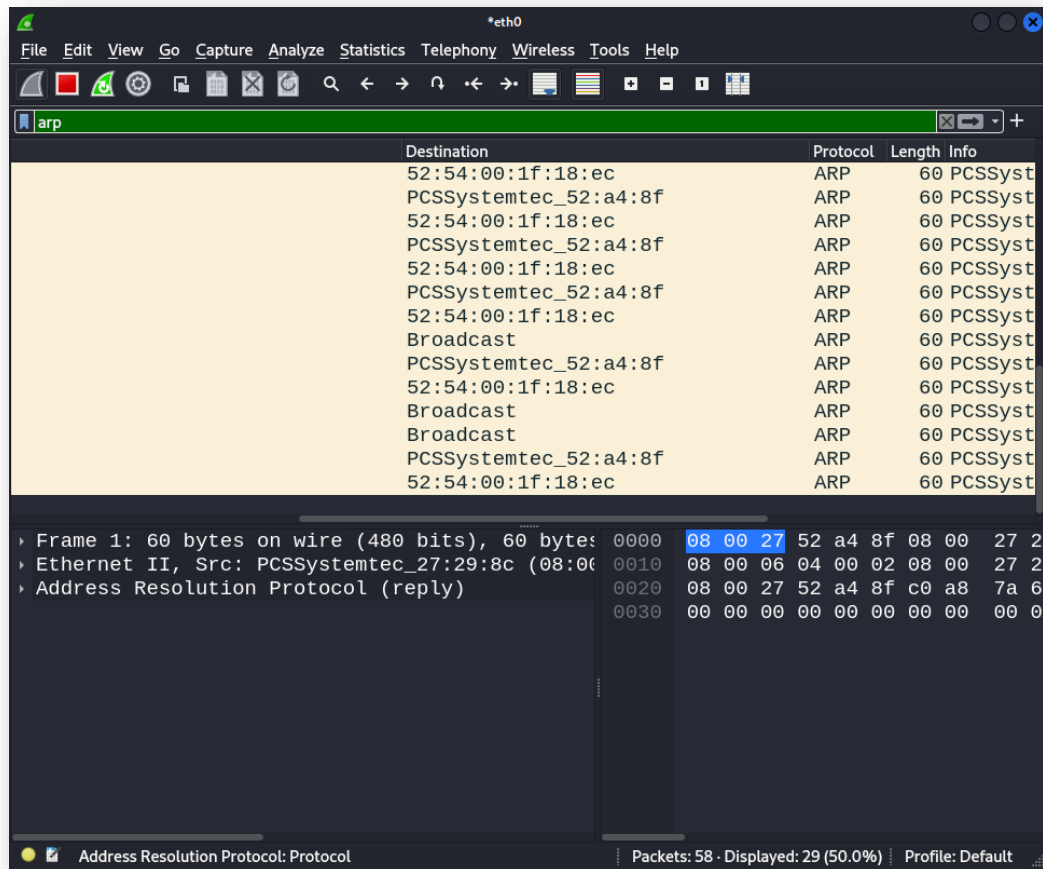
- Run **Wireshark**, and let the tool detect an **ARP request** packets through the network.
- To do so, do the following:
 - Select **preferences** from **Edit**, then **select Protocols** → **ARP/RARP**
 - Keep the “**Detect ARP request storms**” selected. Then press **OK**



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 arp spoofing attack using any of the previously discussed tools (**bettercap** or **arp spoof**)

Bettercap `-iface eth0 -caplet spoof.cap`

Wireshark packet capture window showing traffic on interface eth0. The packet list pane displays several ARP requests (No. 331-336) and NTP traffic (No. 338-339). The packet details pane shows the structure of the first frame: Ethernet II, Logical-Link Control, and Spanning Tree Protocol.

No.	Time	Source	Destination	Protocol	Length	Info
331	100.126958	PCSystemtec_27:29:...	Broadcast	ARP	60	Who has 192.168.122.249?
332	100.128880	PCSystemtec_27:29:...	Broadcast	ARP	60	Who has 192.168.122.250?
333	100.130790	PCSystemtec_27:29:...	Broadcast	ARP	60	Who has 192.168.122.251?
334	100.135777	PCSystemtec_27:29:...	Broadcast	ARP	60	Who has 192.168.122.252?
335	100.141797	PCSystemtec_27:29:...	Broadcast	ARP	60	Who has 192.168.122.253?
336	100.143585	PCSystemtec_27:29:...	Broadcast	ARP	60	Who has 192.168.122.254?
337	102.018362	f6:fa:4e:04:32:51	Spanning-tree-(for-...	STP	60	Conf. Root = 32768/0/52:5
338	102.932165	192.168.122.103	185.144.161.170	NTP	90	NTP Version 4, client
339	102.953079	185.144.161.170	192.168.122.103	NTP	90	NTP Version 4, server
340	104.000308	f6:fa:4e:04:32:51	Spanning-tree-(for-...	STP	60	Conf. Root = 32768/0/52:5

Wireshark Expert Information window showing detected anomalies. The list includes warnings for duplicate IP address configuration, DNS query retransmission, D-SACK Sequence, and connection reset (RST). Notes include time to live, ARP packet storm detected, and acknowledgment number field issues. Chat messages show formatted text and connection finish (FIN).

Severity	Summary	Group
Warning	Duplicate IP address configured	Sequence
Warning	DNS query retransmission	Protocol
Warning	D-SACK Sequence	Sequence
Warning	Connection reset (RST)	Sequence
Note	Time To Live	Sequence
Note	ARP packet storm detected	Sequence
Note	The acknowledgment number field is nonzero while the ACK flag is not set	Protocol
Note	This frame undergoes the connection closing	Sequence
Note	This frame initiates the connection closing	Sequence
Chat	Formatted text	Sequence
Chat	Connection finish (FIN)	Sequence

Wireshark Expert Information window showing a duplicate IP address warning. The list includes a warning for duplicate IP address configured and a sequence of subsequent packets from 192.168.122.109.

Severity	Summary	Group
Warning	Duplicate IP address configured	Sequence
	192.168.122.1 is at 08:00:27:27:29:8c	Sequence
	494 192.168.122.109 is at 08:00:27:27:29:8c	Sequence
	593 192.168.122.1 is at 08:00:27:27:29:8c	Sequence
	594 192.168.122.109 is at 08:00:27:27:29:8c	Sequence
	766 192.168.122.1 is at 08:00:27:27:29:8c	Sequence
	767 192.168.122.109 is at 08:00:27:27:29:8c	Sequence
	1014 192.168.122.1 is at 08:00:27:27:29:8c	Sequence
	1015 192.168.122.109 is at 08:00:27:27:29:8c	Sequence
	1194 192.168.122.1 is at 08:00:27:27:29:8c	Sequence
	1195 192.168.122.109 is at 08:00:27:27:29:8c	Sequence
	1292 192.168.122.1 is at 08:00:27:27:29:8c	Sequence
	1293 192.168.122.109 is at 08:00:27:27:29:8c	Sequence
	1316 192.168.122.1 is at 08:00:27:27:29:8c	Sequence
	1317 192.168.122.109 is at 08:00:27:27:29:8c	Sequence
	1332 192.168.122.1 is at 08:00:27:27:29:8c	Sequence
	1333 192.168.122.109 is at 08:00:27:27:29:8c	Sequence
	1341 192.168.122.1 is at 08:00:27:27:29:8c	Sequence
	1342 192.168.122.109 is at 08:00:27:27:29:8c	Sequence
	1430 192.168.122.1 is at 08:00:27:27:29:8c	Sequence
	1431 192.168.122.109 is at 08:00:27:27:29:8c	Sequence
	1600 192.168.122.1 is at 08:00:27:27:29:8c	Sequence

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

Detection of Cyberattacks

Detecting suspicious Activities in The Network using Wireshark

- Run the arp spoofing attack using any of the previously discussed tools (**bettercap** or **arp spoof**)

Bettercap `-iface eth0 -caplet spoof.cap`

Wireshark packet capture window showing traffic on interface eth0. The packet list pane displays several ARP requests (No. 331-336) and STP frames (No. 337-340). The packet details pane shows the structure of the selected frame (No. 338):

- Frame 1: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface eth0
- IEEE 802.3 Ethernet
- Logical-Link Control
- Spanning Tree Protocol

The packet bytes pane shows the raw data of the selected frame.

Wireshark Expert Information window showing detected anomalies. The table below summarizes the entries:

Severity	Summary	Group
Warning	Duplicate IP address configured	Sequence
Warning	DNS query retransmission	Protocol
Warning	D-SACK Sequence	Sequence
Warning	Connection reset (RST)	Sequence
Note	Time To Live	Sequence
Note	ARP packet storm detected	Sequence
Note	The acknowledgment number field is nonzero while the ACK flag is not set	Protocol
Note	This frame undergoes the connection closing	Sequence
Note	This frame initiates the connection closing	Sequence
Chat	Formatted text	Sequence
Chat	Connection finish (FIN)	Sequence

Wireshark Expert Information window showing a duplicate IP address warning. The table below summarizes the entries:

Severity	Summary	Group
Warning	Duplicate IP address configured	Sequence
	192.168.122.1 is at 08:00:27:27:29:8c	Sequence
	494 192.168.122.109 is at 08:00:27:27:29:8c	Sequence
	593 192.168.122.1 is at 08:00:27:27:29:8c	Sequence
	594 192.168.122.109 is at 08:00:27:27:29:8c	Sequence
	766 192.168.122.1 is at 08:00:27:27:29:8c	Sequence
	767 192.168.122.109 is at 08:00:27:27:29:8c	Sequence
	1014 192.168.122.1 is at 08:00:27:27:29:8c	Sequence
	1015 192.168.122.109 is at 08:00:27:27:29:8c	Sequence
	1194 192.168.122.1 is at 08:00:27:27:29:8c	Sequence
	1195 192.168.122.109 is at 08:00:27:27:29:8c	Sequence
	1292 192.168.122.1 is at 08:00:27:27:29:8c	Sequence
	1293 192.168.122.109 is at 08:00:27:27:29:8c	Sequence
	1316 192.168.122.1 is at 08:00:27:27:29:8c	Sequence
	1317 192.168.122.109 is at 08:00:27:27:29:8c	Sequence
	1332 192.168.122.1 is at 08:00:27:27:29:8c	Sequence
	1333 192.168.122.109 is at 08:00:27:27:29:8c	Sequence
	1341 192.168.122.1 is at 08:00:27:27:29:8c	Sequence
	1342 192.168.122.109 is at 08:00:27:27:29:8c	Sequence
	1430 192.168.122.1 is at 08:00:27:27:29:8c	Sequence
	1431 192.168.122.109 is at 08:00:27:27:29:8c	Sequence
	1600 192.168.122.1 is at 08:00:27:27:29:8c	Sequence

- Wireshark starts **detecting ARP poisoning** requests **over** the network.
- For more **analysis**, select **Expert Information** from the **Analyze Menu**

Detection of Cyberattacks

Detecting suspicious Activities

- Run the arp spoofing (bettercap or arpspoof)

Bettercap -iface eth0 -

No.	Time	Source	Destination
44	14.040087566	PCSSystemtec_27:29:8c	52:54:00:1f:18:ec
51	15.054850815	PCSSystemtec_52:a4:8f	Broadcast
52	16.035791429	PCSSystemtec_27:29:8c	PCSSystemtec_52:a4:8f
53	16.042403547	PCSSystemtec_27:29:8c	52:54:00:1f:18:ec
54	16.092516073	PCSSystemtec_52:a4:8f	Broadcast
56	17.118971633	PCSSystemtec_52:a4:8f	Broadcast
57	18.040919256	PCSSystemtec_27:29:8c	PCSSystemtec_52:a4:8f
58	18.046547274	PCSSystemtec_27:29:8c	52:54:00:1f:18:ec
59	18.138238507	PCSSystemtec_52:a4:8f	Broadcast
66	19.159572028	PCSSystemtec_52:a4:8f	Broadcast
67	20.045936032	PCSSystemtec_27:29:8c	PCSSystemtec_52:a4:8f
68	20.049793982	PCSSystemtec_27:29:8c	52:54:00:1f:18:ec
69	20.185066567	PCSSystemtec_52:a4:8f	Broadcast
71	21.216876745	PCSSystemtec_52:a4:8f	Broadcast
72	22.050188553	PCSSystemtec_27:29:8c	PCSSystemtec_52:a4:8f
73	22.058575292	PCSSystemtec_27:29:8c	52:54:00:1f:18:ec
74	22.233442910	PCSSystemtec_52:a4:8f	Broadcast
76	23.263348016	PCSSystemtec_52:a4:8f	Broadcast

over the network.
the **Analyze Menu**

Detection of Cyberattacks

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.

Windows OS

After

Before

XArp - unregistered version
File XArp Professional Help

✓ Status: no ARP attacks Security level set to: basic

- View detected attacks
- Read the 'Handling ARP attacks' help
- View XArp logfile

Get XArp Professional now!
Register XArp Professional

	IP	MAC	Host	Vendor	Interface	Online	Cache	First seen
✓	192.168.122.1	52-54-00-1f-18-ec	gns3vm	Realtek (uptec...	0x4 - Intel(R) P...	unkno...	yes	4/3/2024 09:42:
✓	192.168.122.21	08-00-27-ab-32-d8	WinDev2401Eval	Cadmus Com...	0x4 - Intel(R) P...	unkno...	no	4/3/2024 09:42:
✓	192.168.122.27	08-00-27-27-29-8c	kali	Cadmus Com...	0x4 - Intel(R) P...	unkno...	yes	4/3/2024 09:42:
✓	192.168.122.103	08-00-27-30-20-e0	192.168.122.103	Cadmus Com...	0x4 - Intel(R) P...	unkno...	yes	4/3/2024 09:42:
✓	192.168.122.109	08-00-27-52-a4-8f	raspberr...	Cadmus Com...	0x4 - Intel(R) P...	unkno...	no	4/3/2024 09:42:

XArp 2.2.2 - 5 mappings - 1 interface - 0 alerts

Windows 11 [Running] - Oracle VM VirtualBox

XArp - unregistered version
File XArp Professional Help

✗ Status: ARP attacks detected! Security level set to: basic

- View detected attacks
- Read the 'Handling ARP attacks' help
- View XArp logfile

Get XArp Professional now!
Register XArp Professional

	IP	MAC	Host	Vendor	Interface	Online	Cache	First seen
✗	192.168.122.1	52-54-00-1f-18-ec	gns3vm	Realtek (uptec...	0x4 - Intel(R) P...	unkno...	no	4/3/2024 09:42:
✗	192.168.122.21	08-00-27-ab-32-d8	WinDev2401Eval	Cadmus Com...	0x4 - Intel(R) P...	unkno...	no	4/3/2024 09:42:
✗	192.168.122.27	08-00-27-27-29-8c	kali	Cadmus Com...	0x4 - Intel(R) P...	unkno...	yes	4/3/2024 09:42:
✗	192.168.122.103	08-00-27-30-20-e0	192.168.122.103	Cadmus Com...	0x4 - Intel(R) P...	unkno...	yes	4/3/2024 09:42:
✗	192.168.122.109	08-00-27-27-29-8c	raspberr...	Cadmus Com...	0x4 - Intel(R) P...	unkno...	yes	4/3/2024 09:42:

XArp 2.2.2 - 5 mappings - 1 interface - 10 alerts

Alert 1 of 10
4/3/2024 09:44:26
ChangeFilter: MAC address for IP 192.168.122.1 changed from 52-54-00-1f-18-ec to 08-00-27-27-29-8c

```
Interface : 0x4
[ethernet]
source mac : 08-00-27-27-29-8c
dest mac : 08-00-27-52-a4-8f
type : 0x806
[arp]
direction : in
type : reply
source ip : 192.168.122.1
dest ip : 192.168.122.109
source mac : 08-00-27-27-29-8c
dest mac : 08-00-27-52-a4-8f
```

Prevention of ARP Poisoning Attack

Windows OS

```
Command Prompt
Microsoft Windows [Version 10.0.22621.3296]
(c) Microsoft Corporation. All rights reserved.

C:\Users\User>arp -a

Interface: 192.168.122.21 --- 0x4
Internet Address      Physical Address      Type
192.168.122.1         52-54-00-1f-18-ec    dynamic
192.168.122.27        08-00-27-27-29-8c    dynamic
192.168.122.67        08-00-27-27-29-8c    dynamic
192.168.122.103       08-00-27-30-20-e0    dynamic
192.168.122.109       08-00-27-27-29-8c    dynamic
192.168.122.255       ff-ff-ff-ff-ff-ff    static
224.0.0.22            01-00-5e-00-00-16    static
224.0.0.251           01-00-5e-00-00-fb    static
224.0.0.252           01-00-5e-00-00-fc    static
239.255.255.250       01-00-5e-7f-ff-fa    static
255.255.255.255       ff-ff-ff-ff-ff-ff    static

C:\Users\User>
```

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

ARP Poisoning Protection

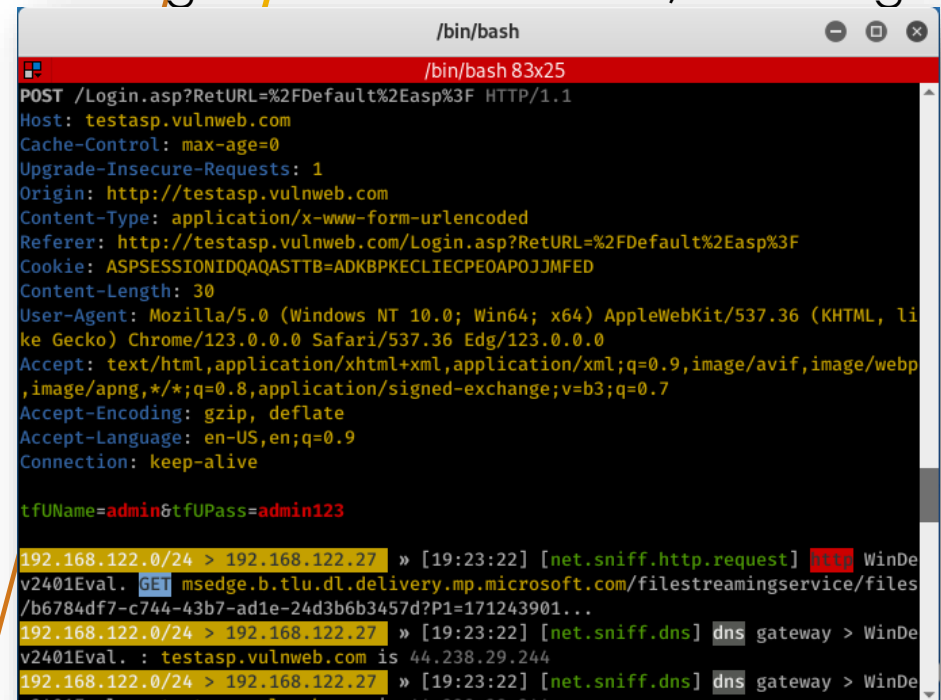
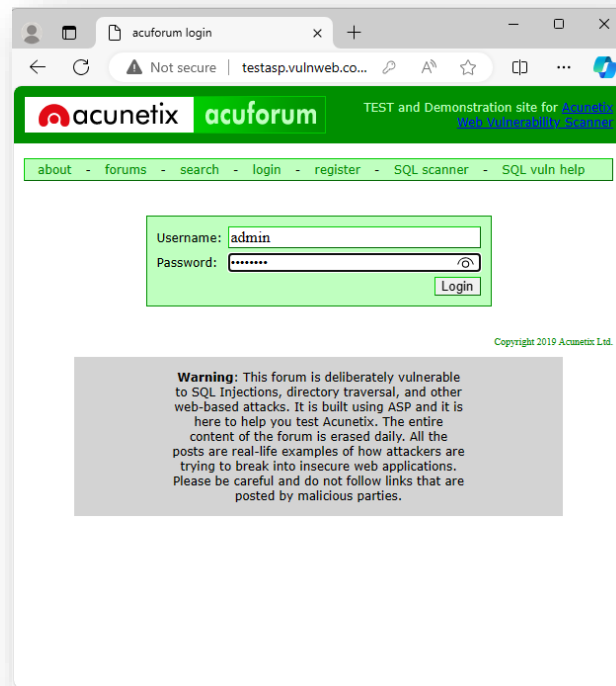
Prevention of MITM

- 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

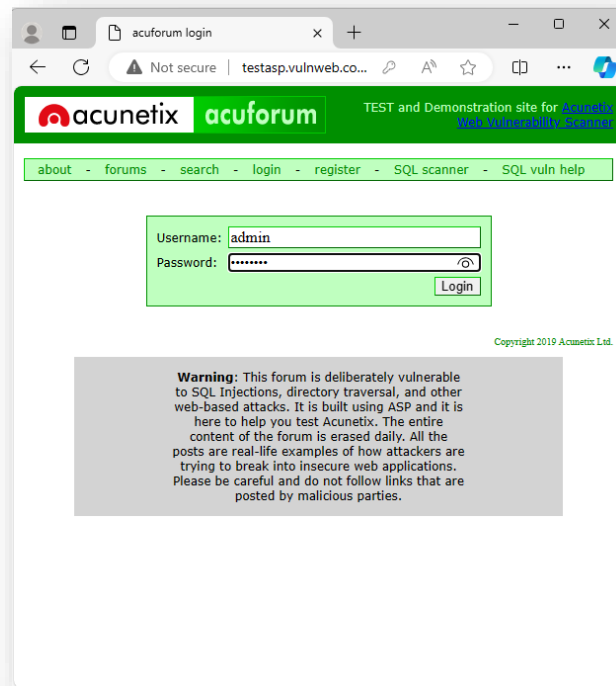
VPN for Prevention of MITM

- 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



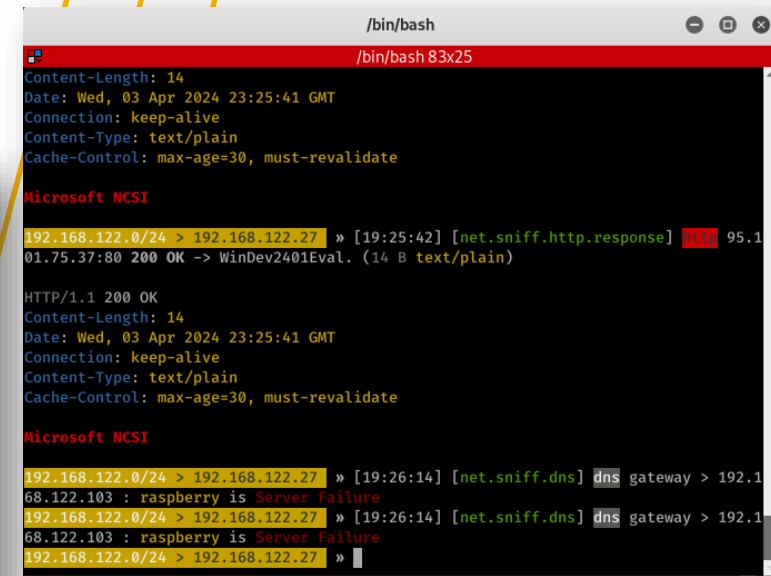
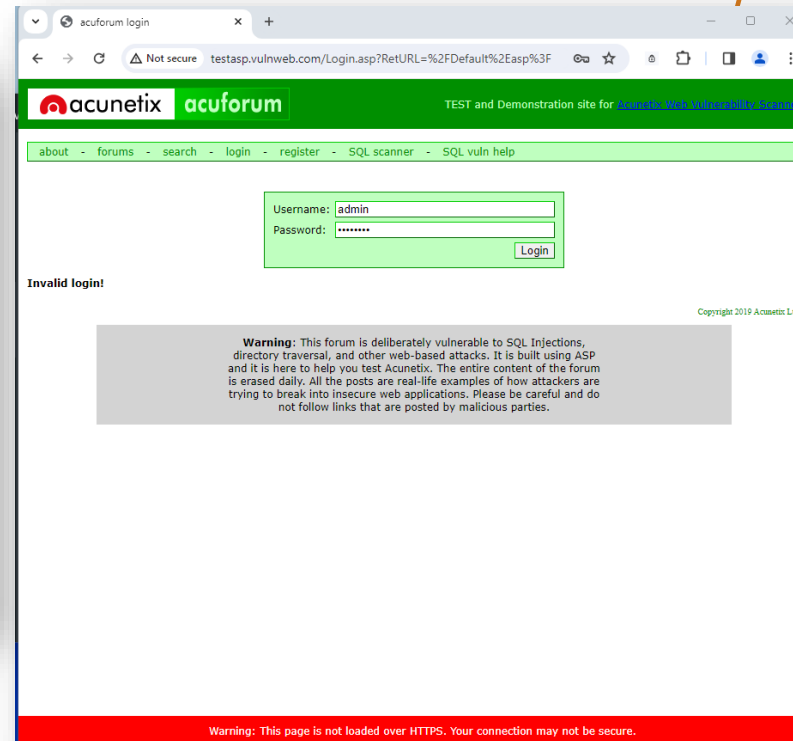
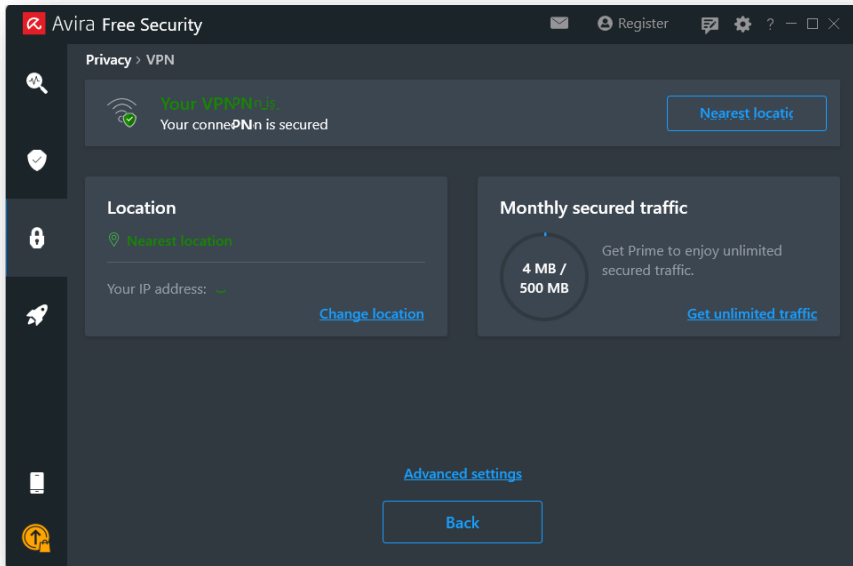
```
/bin/bash
/bin/bash 83x25
POST /Login.asp?RetURL=%2FDefault%2Easp%3F HTTP/1.1
Host: testasp.vulnweb.com
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin: http://testasp.vulnweb.com
Content-Type: application/x-www-form-urlencoded
Referer: http://testasp.vulnweb.com/Login.asp?RetURL=%2FDefault%2Easp%3F
Cookie: ASPSESSIONIDQAQASTTB=ADKBPKECLIECPEOAP0JJMFED
Content-Length: 30
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/123.0.0.0 Safari/537.36 Edg/123.0.0.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: keep-alive

tfUName=admin&tfUPass=admin123

192.168.122.0/24 > 192.168.122.27 » [19:23:22] [net.sniff.http.request] GET WinDe
v2401Eval. GET msedge.b.tlu.dl.delivery.mp.microsoft.com/filestreamingservice/files
/b6784df7-c744-43b7-ad1e-24d3b6b3457d?P1=171243901...
192.168.122.0/24 > 192.168.122.27 » [19:23:22] [net.sniff.dns] dns gateway > WinDe
v2401Eval. : testasp.vulnweb.com is 44.238.29.244
192.168.122.0/24 > 192.168.122.27 » [19:23:22] [net.sniff.dns] dns gateway > WinDe
```

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**
- Observe the **difference**



Thank you

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