

Table of Contents

1. Introduction	1
1.1 Aims & Objectives.....	1
2. Background	3
2.1 Covert Channel Analysis and data hiding in TCP/IP	3
2.2 Case study	3
2.3 Data Hiding & Covert TCP/IP	4
2.3.1 ARP.....	4
2.3.2 DHCP.....	4
2.3.3 TCP.....	5
3. Scenario	6
4. Demonstration.....	7
4.1 Sending Packets from Ubuntu to Kali Linux	7
4.2 Receiving Packets from Ubuntu in kali Linux.....	7
4.3 How It Works	8
5. Detection & Investigation	9
5.1 Message Send in Kali Linux	9
5.2 Message Received in Kali Linux.....	9
5.3 Investigation in Wireshark	10
6. Conclusion	13
References.....	13
Appendix	15
Details About Covert Channel	15
Digital Forensics & Investigation	15
Sender PC Python Scripts.....	16
Receiver PC Python Scripts	17
Working Platform in kali Linux	18
Working Platform in Ubuntu.....	19
Working Desktop in Kali Linux	19
Working Desktop in Ubuntu.....	20

Checking the IP address of Kali.....	20
Checking IP address of Ubuntu	21
Ping From kali Linux to Ubuntu	21
Ping from Ubuntu to Kali	22

Table of Figures

Figure 1 Covert Channel	1
Figure 2 ARP	5
Figure 3 DHCP	5
Figure 4 TCP	6
Figure 5 Sender PC.....	8
Figure 6 Receiver PC	8
Figure 7 Message Send	10
Figure 8 Message Received	10
Figure 9 Covert Channel Detection	11
Figure 10 Data Hidden in TCP Packet	12
Figure 11 Upon Filtering TCP Packets	12
Figure 12 Data Hidden in TCP Packet 1	13
Figure 13 Data Hidden in TCP Packets 2	13
Figure 14 Kali Linux.....	21
Figure 15 Ubuntu	21
Figure 16 Working area in kali Linux	22
Figure 17 Working area in Ubuntu	22
Figure 18 Checking IP address of kali Linux	23
Figure 19 Checking IP address of Ubuntu	23
Figure 20 Response from ubuntu to Kali	24
Figure 21 Response from Ubuntu to Kali	24

1. Introduction

Digital crime investigation is the process of looking into, evaluating, and obtaining sensitive forensic digital evidence from the attack's target networks to identify the attackers and their genuine intentions. The Internet or a local network might be used in this. To communicate between users, one might act as a covert channel.

The exchange of information or data across a hidden route is referred to as covert communication. A particular kind of computer attack or threat known as a covert channel that enables communication between distinct objects and processes that were previously prohibited from doing so due to system or network regulations (Prof. Claudio Cilli, CISA, CISM, CRISC, CGEIT, 2017).

[For furthermore details about Covert Channel Click Here.](#)

The messages and the information are sent through the help of TCP/IP as seen in as below figure:

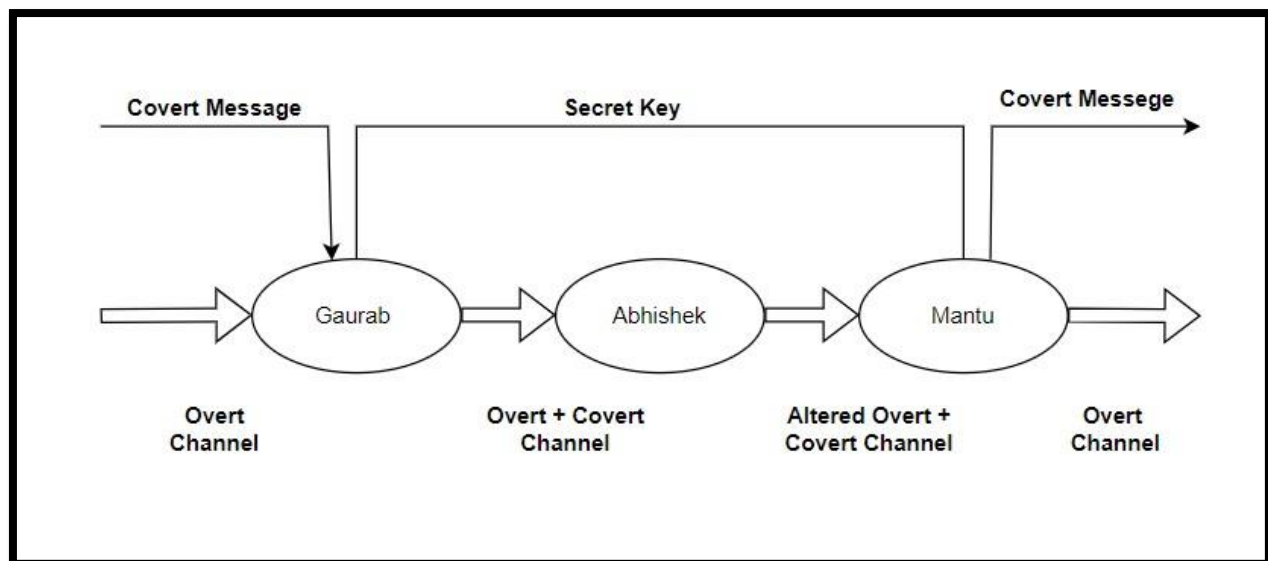


Figure 1 Covert Channel

1.1 Aims & Objectives

The purpose of this coursework is to explore and conduct studies on a digital crime scene utilizing interactive crime scene techniques and a variety of software tools.

The Objective of this Coursework are:

- ✦ Creating a crime scene and investigating
- ✦ To be familiar with the various case study types used in digital crime investigation.
- ✦ To apply an inquiry using a software application to circumvent security measures ✦
To protect your privacy.

2. Background

2.1 Covert Channel Analysis and data hiding in TCP/IP

Data hiding has taken on a new life in the digital world for the purpose of digital covert communication with the single goal of avoiding discovery. Additionally, by being aware of the forensic techniques used by ants to escape discovery, improper data-hiding techniques may be eliminated. This feature provides the groundwork for improving a data hiding approach with greater levels of confidence in the most important situations. The section is intended to cover more forensic and anti-forensic data hiding strategies. Further forensic and anti-forensic hiding data techniques will be covered in this area (Michael Raggo, Chet Hosmer, 2017).

Lampson recognized Covert Networks for the very first time as a route of communications that wasn't meant for any Simmons in 1973. an unconscious conduit through stenography. The use of this concept included the sharing of confidential material on how to escape between two prisoners via an open channel. This doesn't mean that any system that functions freely would continue to function under restrictions; it only means that any system would be impossible to leak information if contained (Lampson, 1993).

2.2 Case study

Covert Channel: A Penetration Testing Case Study from IBM X-Force Red

Firefox is a popular browser. Mozilla transmits and steals data to a packet. This satisfied our demand for a secret, segregated path inside the business. We won't have to worry with any public keys or distribution as it will explicitly encrypt and decode the packet for us in the web browser using the AES-GCM technique. The packet will stay private, including Mozilla, and won't be shared with any other group along the road. It will also escape being examined by transmission proxies that can disassemble Transport Layer Security (TLS). Using FFSend (Firefox) provides us additional benefit, even though firefox.com is a reliable domain for most organizational controls.

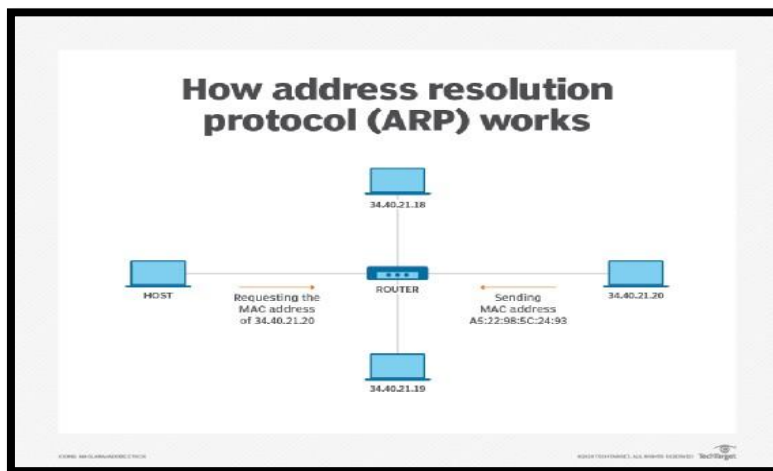
We also achieve the goal of limiting recurrent packet sampling by placing a strict onetime-only cap on the number of times our FFSend link may be examined once it is established, preventing firewall issues and the spread of vulnerabilities. Additionally, FFSend instantly destroys connections after 24 hours, this effectively transforms the path to our packet character if the target has not accessed it. Self-destruction is also offered by FFSend's application program interface (API), allowing orders to be placed for it after a connection has been received but even before its preset expiry time (Snezhkov, 2018).

2.3 Data Hiding & Covert TCP/IP

The Covert TCP/IP used in the projects are as below:

2.3.1 ARP

In a local area network, ARP is a method for translating a changing IP address to a fixed physical machine address (LAN). (Zydyk, 2022).

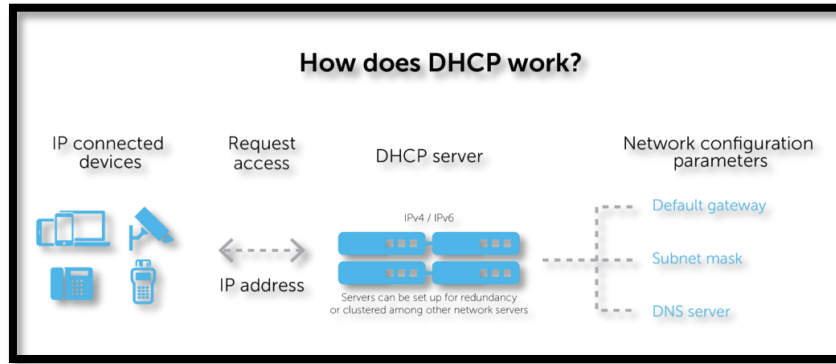


(Zydyk, 2022)

Figure 2 ARP

2.3.2 DHCP

An automated distribution and assignment of IP addresses, default gateways, and other network characteristics to client devices is performed by a DHCP server, a type of network server (Infoblox, 2023).

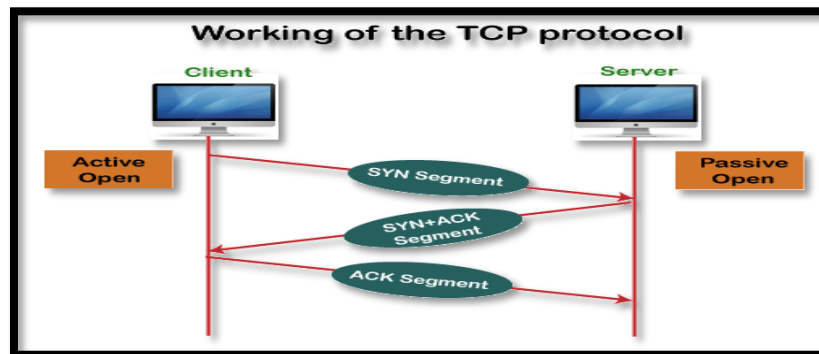


(Bluecat, 2023)

Figure 3 DHCP

2.3.3 TCP

TCP is a communications standard that enables application programs and computer devices to exchange messages across a network, goes by the name of this protocol (Fortinet, 2023).



(Javatpoint, 2023)

Figure 4 TCP

3. Scenario

The project's scenario is a simulation of network traffic. Here, two devices with distinct IP addresses will be able to communicate with one another by using their respective IP addresses and port numbers.

Any type of plain text communication that has not been encrypted in any way can be the hidden data. IP packets with covert data supporting channels.

In other words, packets with a secret message are sent to the network over a hidden channel. This packet is sniffed from a different PC to read the secret message that lies behind the Raw layer.

In this way, a covert channel is formed between two authorized wireless access points, and communication is conducted covertly by encrypting data in a TCP packet.

4. Demonstration

In the demonstration, messages sent across a covert channel are displayed using the sender and receiver's IP addresses.

[For Furthermore details about kali Linux and ubuntu configuration Click Here.](#)

4.1 Sending Packets from Ubuntu to Kali Linux

[Click Here, For the Python Scripts Made for Demonstration](#)

```

gaurab@gaurab-virtual-machine: ~/Desktop
[sudo] password for gaurab:
Enter Destination Address: 192.168.42.132
Enter Port Number: 800
Enter your message: By2024, This Mission Should Be Completed.
Begin emission:
Finished sending 1 packets.
Received 2 packets, got 1 answers, remaining 0 packets
Response
#### IP ####
version = 4
ihl = 5
tos = 0x0
len = 40
id = 0
flags = DF
frag = 0
ttl = 64
proto = tcp
chksum = 0x6478
src = 192.168.42.132
dst = 192.168.42.131
\options \
#### TCP ####
sport = 800
dport = 19114
seq = 0
ack = 42
dataofs = 5
reserved = 0
flags = RA
window = 0
chksum = 0x8b84
urgptr = 0
options = []
#### Padding ####
load = '\x00\x00\x00\x00\x00\x00'
  
```

Figure 5 Sender PC

4.2 Receiving Packets from Ubuntu in kali Linux

```

kali@kali: ~/Desktop
File Actions Edit View Help
(kali@kali)~$ sudo python3 Covert_Receive.py
[sudo] password for kali:
Enter interface: eth0
Enter IP address to listen: 192.168.42.131
Enter port to listen: 800
Listening on interface 'eth0' in port 800 for '192.168.42.131' ...
Source: 192.168.42.131
Message: By2024, This Mission Should Be Completed.
  
```

Figure 6 Receiver PC

4.3 How It Works

Here, the sender PC and receiver PC are both listed as authorized network devices.

However, the sender PC is producing a TCP Packet that conceals a message. Sender PC creates an IP packet with the receiver PC's IP address using Python and Scapy (a Python packet-creation library), then stacks it on top of a TCP packet with the userspecified destination port and adds a message. The network receives this packet.

Now, the receiver PC scans the traffic on the relevant interface. It will keep an eye out for packets with the sender PC's source IP and should be TCP packets on the sender PC's designated port. All of this was created using the Scapy sniff function and was written in Python. It is necessary to identify and do more research on this clandestine communication route between these devices.

5. Detection & Investigation

5.1 Message Send in Kali Linux

```

Enter your message: This message is for investigation
-----
Begin emission:
Finished sending 1 packets.
.*
Received 2 packets, got 1 answers, remaining 0 packets
-----
Response
-----
###[ IP ]###
version   = 4
ihl       = 5
tos       = 0x0
len       = 40
id        = 0
flags     = DF
frag      = 0
ttl       = 64
proto     = tcp
chksum    = 0x6478
src       = 192.168.42.132
dst       = 192.168.42.131
\options  \
###[ TCP ]###
sport     = 800
dport     = 28764
seq       = 0
ack       = 34
dataoffs  = 5
reserved  = 0
flags     = RA
window    = 0
chksum    = 0x65da
urgptr    = 0
options   = []
###[ Padding ]###
load      = '\x00\x00\x00\x00\x00\x00'

```

Figure 7 Message Send

5.2 Message Received in Kali Linux

```

kali@kali: ~/Desktop
File Actions Edit View Help
(kali@kali)-[~/Desktop]
$ sudo python3 Covert_Receive.py
[sudo] password for kali:
Sorry, try again.
[sudo] password for kali:

Enter interface: eth0
Enter IP address to listen: 192.168.42.131
Enter port to listen: 800

Listening on interface 'eth0' in port 800 for '192.168.42.131' ...

Source: 192.168.42.131
Message: This message is for investigation

```

Figure 8 Message Received

5.3 Investigation in Wireshark

When Wireshark is used to monitor network traffic, TCP may be seen being used to send packets between devices. The TCP of Wireshark shows the following exchange between two IP addresses. The messages used in this project's detection are shown in the image below.

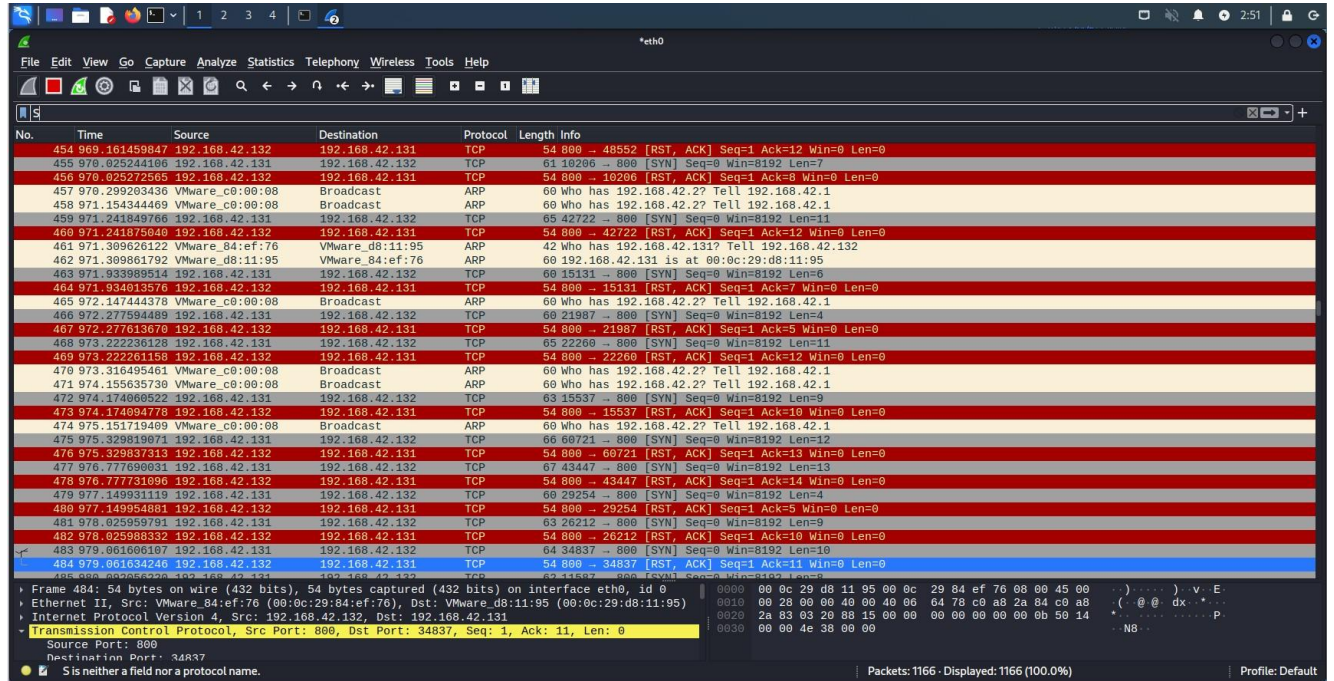


Figure 9 Covert Channel Detection

It is quite rare for devices to communicate in this way when the destination immediately responded with the RST flag. Now, given that this is only a demonstration, this is a feeble effort. In a real-world setting, things might not be as simple as this. Something strange was discovered after analyzing the packet, as illustrated below. The term "covert channel" refers to such a communication path between two devices.

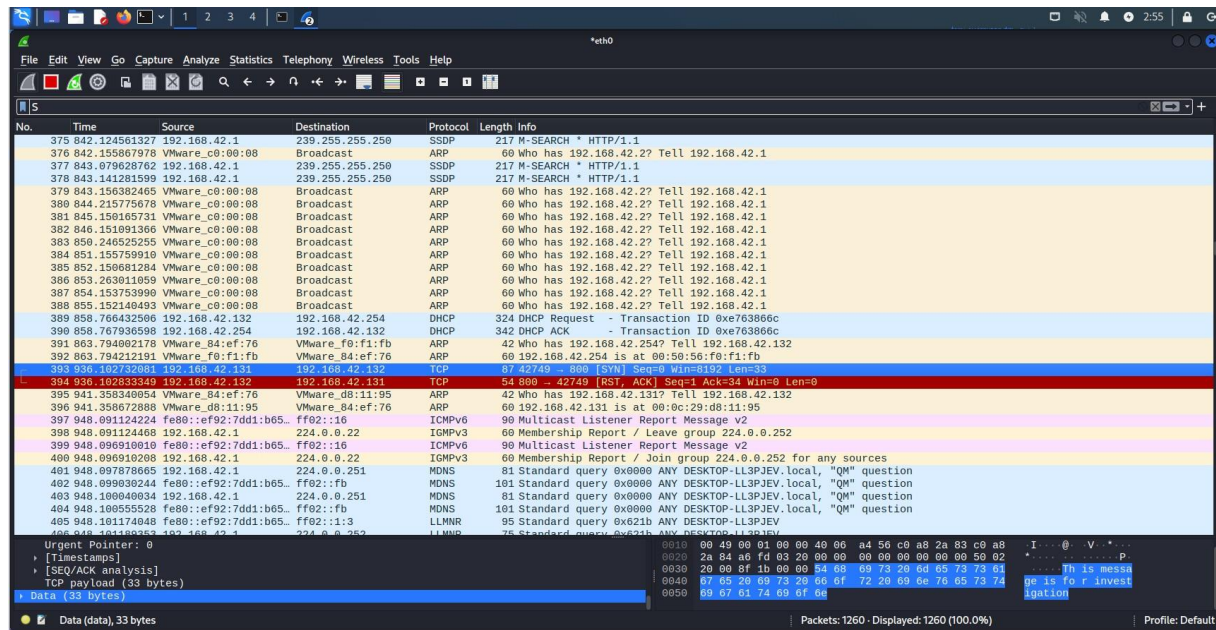


Figure 10 Data Hidden in TCP Packet

TCP Packet contains a secret, as seen in Figure 10. The message that was intercepted from Figure 7 is the same. However, these signals can occasionally be encrypted and remain unreadable to investigators. If you pay special attention, these packets are corresponding on port 800. Figure 11 depicts what occurs after TCP packets from port 800 are filtered.

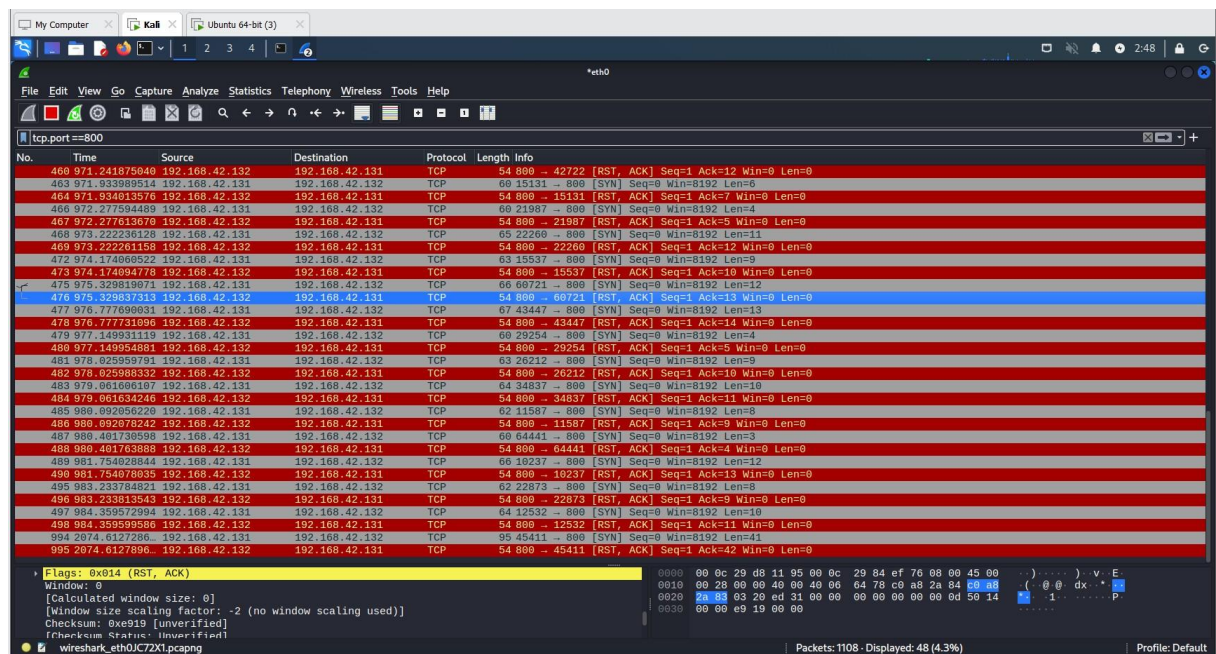


Figure 11 Upon Filtering TCP Packets

It detects such strange traffic. This is a method of detecting covert channels.

Figures 12 and 13 represent data that is hidden away in one of these packets.

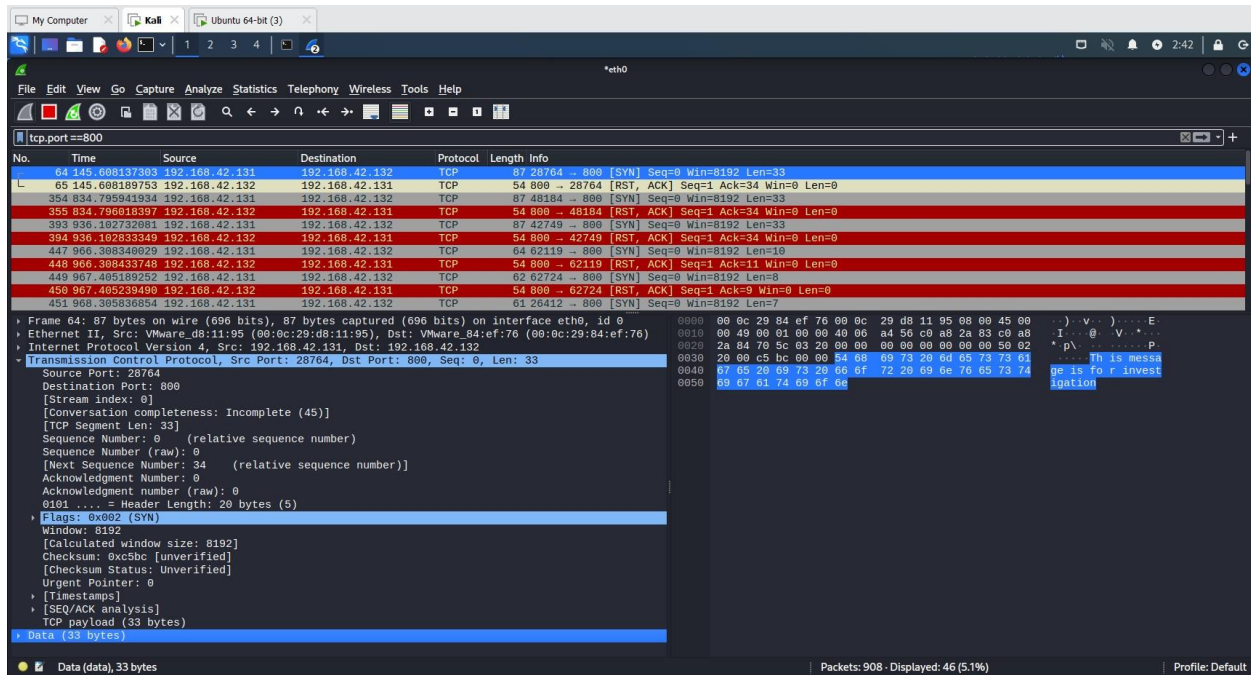


Figure 12 Data Hidden in TCP Packet 1

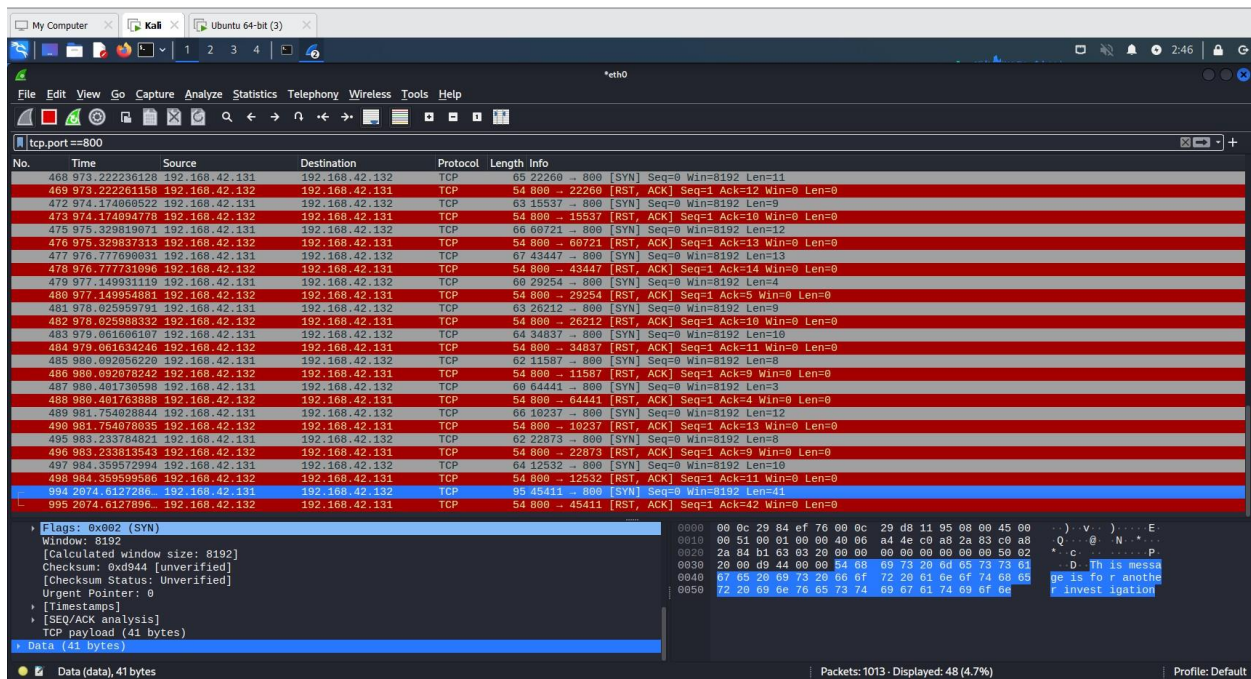


Figure 13 Data Hidden in TCP Packets 2

6. Conclusion

The project has been completed with demonstrations running on Kali Linux and Ubuntu. The focus of the study is on covert channel analysis and the correct identification and examination of data encryption in TCP/IP packets. Here, one computer acts as the transmitter while the other acts as the receiver, receiving the data and information that are being hidden from view from the sender's IP address. Since the receiver PC always responded to the SYN flag TCP packet with the RST flag, there was unpredictable communication behavior that was seen throughout the whole project while Wireshark sniffed or watched the data. After investigating the matter, additional covert communications were found by analyzing the detected packet. It is frequently advised to implement any systemic investigation management procedures. Such a hidden network must be detectable by an information security officer or expert in any firm. By questioning and speaking directly to the system's end user, these behaviors can be avoided. It is crucial to use a different strategy if it involves two computers.

References

Prof. Claudio Cilli, CISA, CISM, CRISC, CGEIT, 2017. *Understanding Covert Channels of Communication*. [Online]

Available at: <https://www.isaca.org/resources/news-and-trends/isaca-nowblog/2017/understanding-covert-channels-of-communication>

Bluecat, 2023. *What is DHCP? It assigns IP addresses dynamically*. [Online]

Available at: <https://bluecatnetworks.com/glossary/what-is-dhcp/> [Accessed 02 01 2023].

Fortinet, 2023. *What is Transmission Control Protocol TCP/IP?*. [Online]

Available at: <https://www.fortinet.com/resources/cyberglossary/tcp-ip> [Accessed 02 01 2023].

G.M. Marchetti, M. Mellia, and R. Sisto, 2019. Covert Channels and Data Hiding in TCP/IP. *EEE Communications Surveys & Tutorials*, 17(2019), p. 1456.

Infoblox, 2023. *What is a DHCP Server?*. [Online]

Available at: <https://www.infoblox.com/glossary/dhcp-server/> [Accessed 02 01 2023].

Javatpoint, 2023. *What is Transmission Control Protocol (TCP)?*. [Online]

Available at: <https://www.javatpoint.com/tcp> [Accessed 02 01 2023].

Lampson, B., 1993. *A Note on the Confinement Problem*, XXVI(2017), p. 613.

Mahendra Kumar Pandey, Girish Parmar, Rajeev Gupta, Afzal Sikande, 2018. *Non-blind Arnold scrambled hybrid image watermarking in YCbCr color space*. [Online]

Available at:

https://www.researchgate.net/publication/328079384_Nonblind_Arnold_scrambled_hybrid_image_watermarking_in_YCbCr_color_space [Accessed 30 12 2022].

Michael Raggo, Chet Hosmer, 2017. *Forensics and Anti-Forensics*. [Online]

Available at: <https://www.sciencedirect.com/topics/computer-science/data-hidingtechnique>

[Accessed 28 12 2022].

Mukesh Dalal, Mamta Juneja, 2021. *A survey on information hiding using video steganography*. [Online]

Available at: https://www.researchgate.net/figure/Basic-flowchart-ofsteganography_fig1_349182791 [Accessed 30 12 2022].

simplilearn, 2022. *What is Steganography? Types, Techniques, Examples & Applications*. [Online]

Available at: <https://www.simplilearn.com/what-is-steganography-article> [Accessed 29 12 2022].

Snezhkov, D., 2018. *Phish or Fox? A Penetration Testing Case Study From IBM XForce Red*. [Online]

Available at: <https://securityintelligence.com/phish-or-fox-a-penetration-testing-casestudy-from-ibm-x-force-red/> [Accessed 26 12 2022].

Zydyk, M., 2022. *Address Resolution Protocol (ARP)*. [Online]

Available at:

<https://www.techtarget.com/searchnetworking/definition/AddressResolution-Protocol-ARP> [Accessed 02 01 2023].

Appendix

Details About Covert Channel

Covert channel analysis refers to the study of methods and techniques used to transmit information covertly, or secretly, through a communication system. In the context of the TCP/IP (Transmission Control Protocol/Internet Protocol) suite, a covert channel is a means of communication that exploits the design or implementation of the protocol to transmit information in a manner that is not intended or visible to the system.

Data hiding is a subset of covert channel analysis, specifically focused on the practice of concealing data within other data to avoid detection. In the context of TCP/IP, data hiding techniques may be used to transmit information over the network without being detected by network security systems or other monitoring tools.

Covert channels and data hiding can be used for legitimate purposes, such as in the development of secure communication systems or for debugging and testing purposes. However, they can also be used for malicious purposes, such as to exfiltrate sensitive data, bypass security controls, or to evade detection by network administrators.

There are a variety of methods and techniques that can be used to create covert channels and hide data within TCP/IP communications. These may include manipulating the fields of the TCP and IP headers, using unconventional protocols or port numbers, or encoding data within the payload of legitimate traffic (G.M. Marchetti, M. Mellia, and R. Sisto, 2019).

Digital Forensics & Investigation

Technology utilization and dependence have increased, making market endeavors more susceptible to digital criminal scene than ever. The demonstration that was the subject of this article is a little illustration of what may go wrong if it were used on a larger scale. Someone or something must have a solid technological foundation and support if they are going to depend on technology.

It is now important to create rules and ethics separately for people's digital lives due to the nature of technological advancement. Digital forensics and investigation are not entirely new areas of cyber security, but they belong (or ought to belong) to the government's legal branch. Every crime, even those committed in the digital world, must be evaluated because the assailants' motivations and methods vary and tell various tales.

Sender PC Python Scripts

```
from scapy.all import IP, sr1, TCP import
random
```

```
def ip_address():
```

```
    # """ Get Destination IP Address. """
```

```
    host = input("Enter Destination Address: ")
```

```
    return host
```

```
def port_number():
```

```
    # """ Get Destination Port Number. """
```

```
    port = input("Enter Port Number: ")    return
```

```
    port
```

```
# IP Address and Port Number
```

```
print('-' * 100) host =
```

```
ip_address() port =
```

```
port_number() while
```

```
True:    try:
```

```
    # Parameters
```

```
    message = input("Enter your message: ")
```

```
    # Change IP Address
```

```
    if 'ip' in message:
```

```
        host = ip_address()
```

```
        continue    elif 'port' in
```

```
        message:        port =
```

```

port_number()
continue

    # TCP Packet    ip = IP(dst=host)    tcp =
TCP(sport=random.randint(10000, 65000), dport=int(port))/message
    # Stack the packets
packet = ip/tcp

    # Send the packet in the network
print('-' * 100)    sent = sr1(packet,
timeout=5)    try:
    # Received results
print('-' * 100)
print("Response")
print('-' * 100)
sent.show()    print('-'
* 100)    except:
    # Neglate NONE Type 'sent' if there is no reply.
pass    except
KeyboardInterrupt:
    print("")    print('-' *
100)    print('>>> Exiting ...
<<<')    print('-' * 100)
break    except Exception as
e:
    print("")
print('Exception:', e)

```

Receiver PC Python Scripts

```

from scapy.all import IP, Raw, sniff, TCP

```

```

def analyze(packet):
    """ Analyze the sniffed packets. """
    try:
        # Read only TCP packet
        if packet.haslayer(TCP):
            # Read only reply from sender PC in given port number
            if packet[IP].src == ip_address and packet[TCP].dport == int(port):
                print('Source:', packet[IP].src)
            print('Message:', packet[Raw].load.decode('utf-8'))
        print('-' * 100)
    except Exception as e:
        print(e)

# Parameters
print('-' * 100)
interface = input("Enter interface: ")
ip_address = input("Enter IP address to listen: ")
port = input("Enter port to listen: ")
print('-' * 100)

# Status
print("Listening on interface '{}' in port {} for '{}...'".format(interface, port, ip_address))
print('-' * 100)

# Sniff
sniffed = sniff(iface=interface, store=False, prn=analyze)

```

Working Platform in kali Linux

This is the working platform of one server where the messages and packets are received.

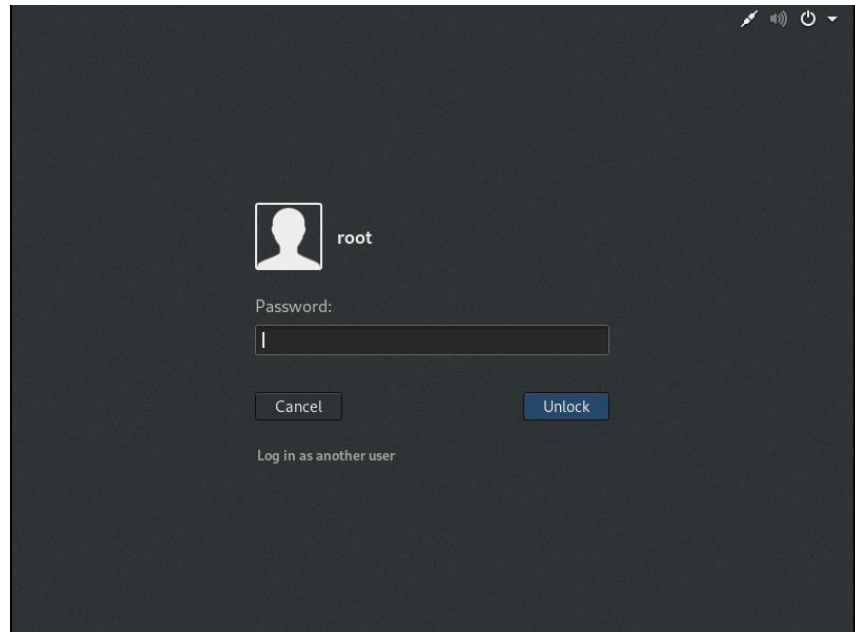


Figure 14 Kali Linux

Working Platform in Ubuntu

This is the server's Working platform, from which messages and packets are sent.

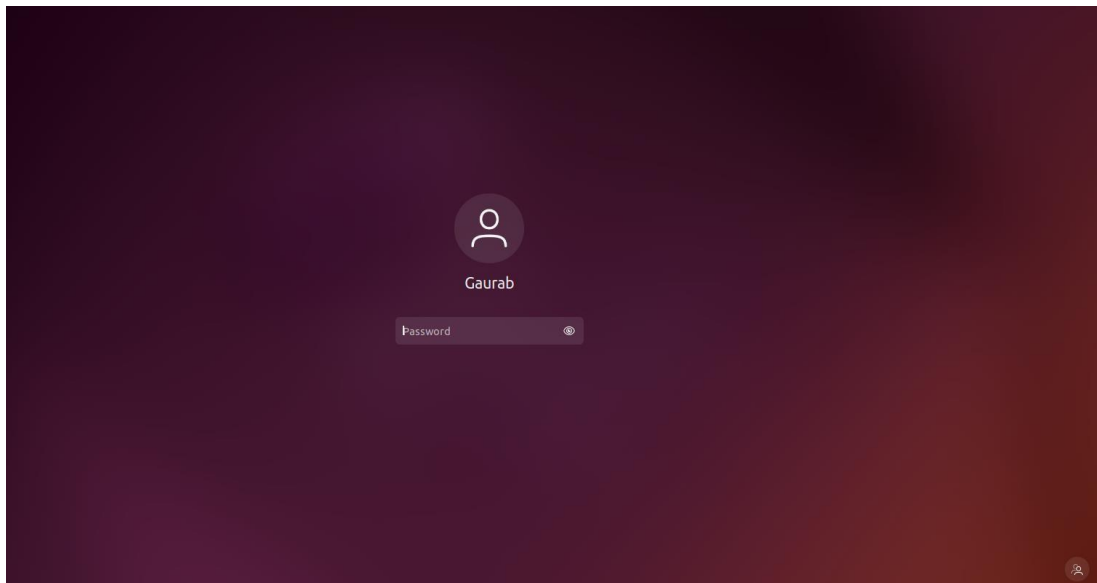


Figure 15 Ubuntu

Working Desktop in Kali Linux

This is the desktop area where the attack will be running

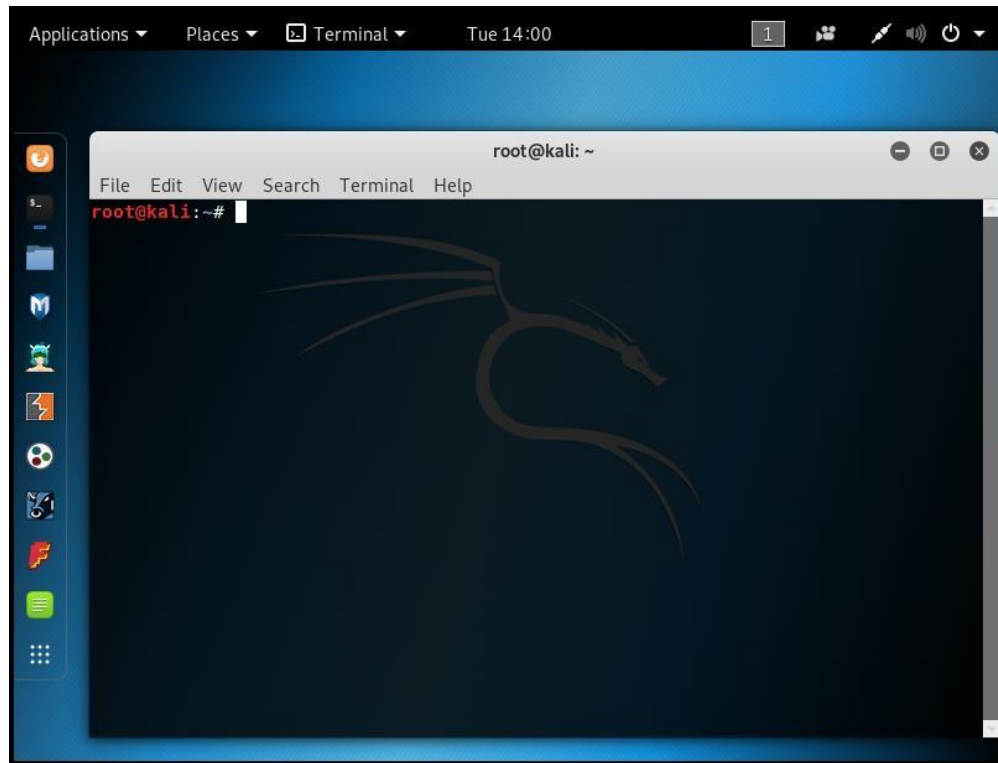


Figure 16 Working area in kali Linux

Working Desktop in Ubuntu

This is the desktop area where the attack will be running

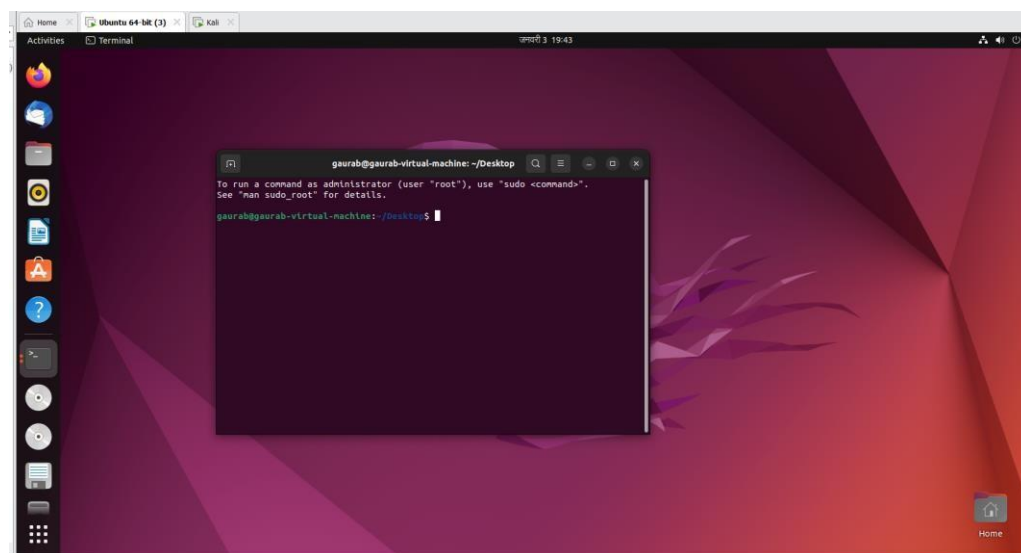
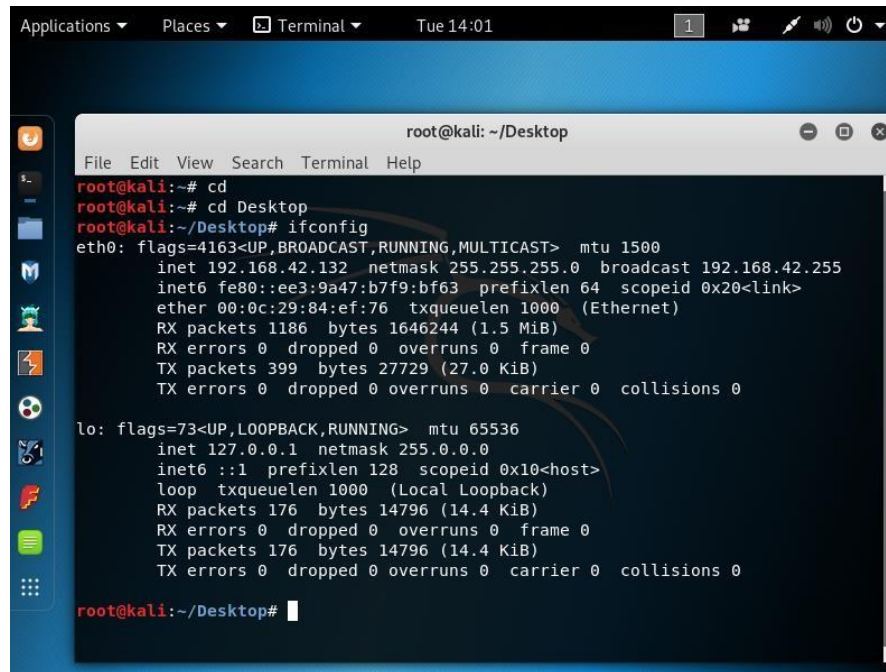


Figure 17 Working area in Ubuntu

Checking the IP address of Kali

The checking of ip is done in kali Linux inserting “ifconfig” command. Then by it get to know its Ip address is 192.168.42.132, netmask is 255.255.255.0 and its broadcast is 192.168.42.255.



```

root@kali:~/Desktop# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.42.132 netmask 255.255.255.0 broadcast 192.168.42.255
    inet6 fe80::ee3:9a47:b7f9:bf63 prefixlen 64 scopeid 0x20<link>
    ether 00:0c:29:84:ef:76 txqueuelen 1000 (Ethernet)
    RX packets 1186 bytes 1646244 (1.5 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 399 bytes 27729 (27.0 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 176 bytes 14796 (14.4 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 176 bytes 14796 (14.4 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

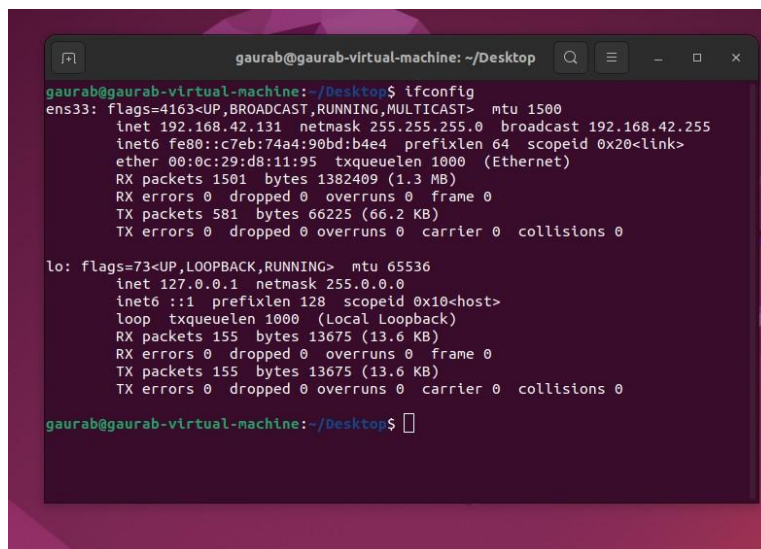
root@kali:~/Desktop#

```

Figure 18 Checking IP address of kali Linux

Checking IP address of Ubuntu

The checking of Ip is done in kali Linux inserting “ifconfig” command. Then by it get to know its Ip address is 192.168.42.131, netmask is 255.255.255.0 and its broadcast is 192.168.42.255.



```

gaurab@gaurab-virtual-machine:~/Desktop$ ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.42.131 netmask 255.255.255.0 broadcast 192.168.42.255
    inet6 fe80::c7eb:74a4:90bd:b4e4 prefixlen 64 scopeid 0x20<link>
    ether 00:0c:29:d8:11:95 txqueuelen 1000 (Ethernet)
    RX packets 1501 bytes 1382409 (1.3 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 581 bytes 66225 (66.2 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 155 bytes 13675 (13.6 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 155 bytes 13675 (13.6 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

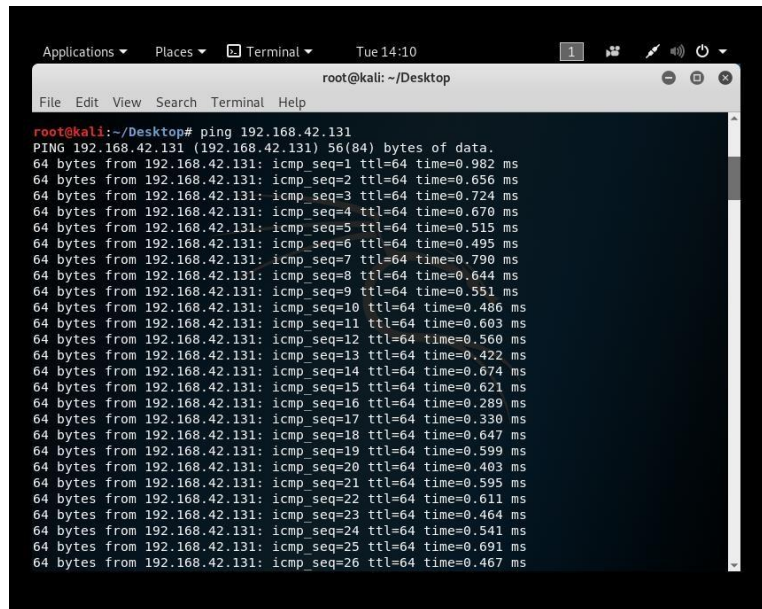
gaurab@gaurab-virtual-machine:~/Desktop$

```

Figure 19 Checking IP address of Ubuntu

Ping From kali Linux to Ubuntu

Here pinging is done from kali Linux to ubuntu if it is communicating or not with “ping 192.168.42.131” command. The response can be seen in below figure:

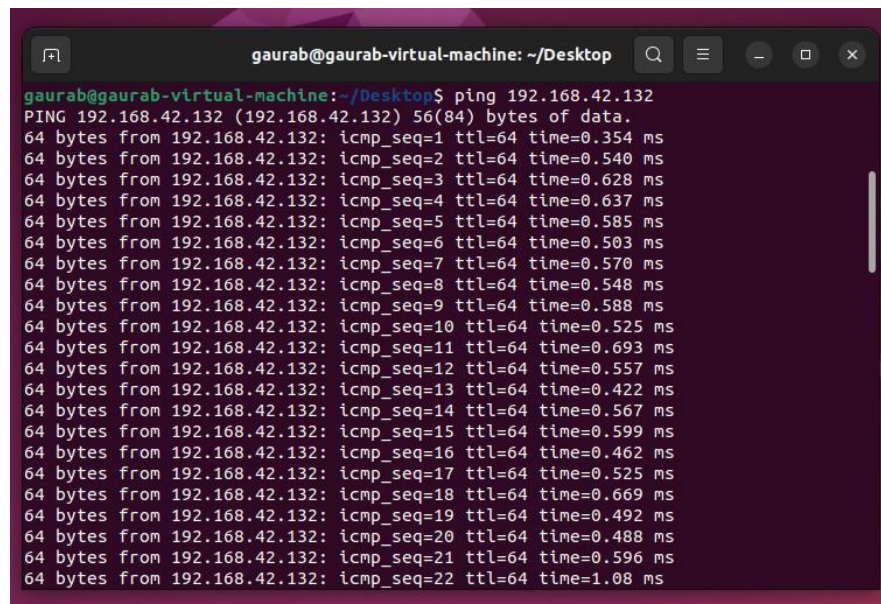
A terminal window titled 'root@kali: ~/Desktop' showing the output of a ping command. The command is 'ping 192.168.42.131'. The output shows 26 successful ping responses, each with a TTL of 64 and a time between 0.467 ms and 0.982 ms. The window has a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'.

```
root@kali: ~/Desktop
root@kali:~/Desktop# ping 192.168.42.131
PING 192.168.42.131 (192.168.42.131) 56(84) bytes of data.
64 bytes from 192.168.42.131: icmp_seq=1 ttl=64 time=0.982 ms
64 bytes from 192.168.42.131: icmp_seq=2 ttl=64 time=0.656 ms
64 bytes from 192.168.42.131: icmp_seq=3 ttl=64 time=0.724 ms
64 bytes from 192.168.42.131: icmp_seq=4 ttl=64 time=0.670 ms
64 bytes from 192.168.42.131: icmp_seq=5 ttl=64 time=0.515 ms
64 bytes from 192.168.42.131: icmp_seq=6 ttl=64 time=0.495 ms
64 bytes from 192.168.42.131: icmp_seq=7 ttl=64 time=0.790 ms
64 bytes from 192.168.42.131: icmp_seq=8 ttl=64 time=0.644 ms
64 bytes from 192.168.42.131: icmp_seq=9 ttl=64 time=0.551 ms
64 bytes from 192.168.42.131: icmp_seq=10 ttl=64 time=0.486 ms
64 bytes from 192.168.42.131: icmp_seq=11 ttl=64 time=0.603 ms
64 bytes from 192.168.42.131: icmp_seq=12 ttl=64 time=0.560 ms
64 bytes from 192.168.42.131: icmp_seq=13 ttl=64 time=0.422 ms
64 bytes from 192.168.42.131: icmp_seq=14 ttl=64 time=0.674 ms
64 bytes from 192.168.42.131: icmp_seq=15 ttl=64 time=0.621 ms
64 bytes from 192.168.42.131: icmp_seq=16 ttl=64 time=0.289 ms
64 bytes from 192.168.42.131: icmp_seq=17 ttl=64 time=0.330 ms
64 bytes from 192.168.42.131: icmp_seq=18 ttl=64 time=0.647 ms
64 bytes from 192.168.42.131: icmp_seq=19 ttl=64 time=0.599 ms
64 bytes from 192.168.42.131: icmp_seq=20 ttl=64 time=0.403 ms
64 bytes from 192.168.42.131: icmp_seq=21 ttl=64 time=0.595 ms
64 bytes from 192.168.42.131: icmp_seq=22 ttl=64 time=0.611 ms
64 bytes from 192.168.42.131: icmp_seq=23 ttl=64 time=0.464 ms
64 bytes from 192.168.42.131: icmp_seq=24 ttl=64 time=0.541 ms
64 bytes from 192.168.42.131: icmp_seq=25 ttl=64 time=0.691 ms
64 bytes from 192.168.42.131: icmp_seq=26 ttl=64 time=0.467 ms
```

Figure 20 Response from ubuntu to Kali

Ping from Ubuntu to Kali

Here ping is done from Ubuntu to Kali Linux if it is communicating or not with “**ping 192.168.42.132**” command. The response can be seen in below figure:

A terminal window titled 'gaurab@gaurab-virtual-machine: ~/Desktop' showing the output of a ping command. The command is 'ping 192.168.42.132'. The output shows 22 successful ping responses, each with a TTL of 64 and a time between 0.354 ms and 1.08 ms. The window has a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'.

```
gaurab@gaurab-virtual-machine: ~/Desktop
gaurab@gaurab-virtual-machine:~/Desktop$ ping 192.168.42.132
PING 192.168.42.132 (192.168.42.132) 56(84) bytes of data.
64 bytes from 192.168.42.132: icmp_seq=1 ttl=64 time=0.354 ms
64 bytes from 192.168.42.132: icmp_seq=2 ttl=64 time=0.540 ms
64 bytes from 192.168.42.132: icmp_seq=3 ttl=64 time=0.628 ms
64 bytes from 192.168.42.132: icmp_seq=4 ttl=64 time=0.637 ms
64 bytes from 192.168.42.132: icmp_seq=5 ttl=64 time=0.585 ms
64 bytes from 192.168.42.132: icmp_seq=6 ttl=64 time=0.503 ms
64 bytes from 192.168.42.132: icmp_seq=7 ttl=64 time=0.570 ms
64 bytes from 192.168.42.132: icmp_seq=8 ttl=64 time=0.548 ms
64 bytes from 192.168.42.132: icmp_seq=9 ttl=64 time=0.588 ms
64 bytes from 192.168.42.132: icmp_seq=10 ttl=64 time=0.525 ms
64 bytes from 192.168.42.132: icmp_seq=11 ttl=64 time=0.693 ms
64 bytes from 192.168.42.132: icmp_seq=12 ttl=64 time=0.557 ms
64 bytes from 192.168.42.132: icmp_seq=13 ttl=64 time=0.422 ms
64 bytes from 192.168.42.132: icmp_seq=14 ttl=64 time=0.567 ms
64 bytes from 192.168.42.132: icmp_seq=15 ttl=64 time=0.599 ms
64 bytes from 192.168.42.132: icmp_seq=16 ttl=64 time=0.462 ms
64 bytes from 192.168.42.132: icmp_seq=17 ttl=64 time=0.525 ms
64 bytes from 192.168.42.132: icmp_seq=18 ttl=64 time=0.669 ms
64 bytes from 192.168.42.132: icmp_seq=19 ttl=64 time=0.492 ms
64 bytes from 192.168.42.132: icmp_seq=20 ttl=64 time=0.488 ms
64 bytes from 192.168.42.132: icmp_seq=21 ttl=64 time=0.596 ms
64 bytes from 192.168.42.132: icmp_seq=22 ttl=64 time=1.08 ms
```

Figure 21 Response from Ubuntu to Kali

[Click Here to visit up](#)