Cybersecurity Internship – Task 1 Report

Intern Name: Priteshsinh Ashokkumar Chavda

Internship Period: September 2025

Organization: Redynox

# Table of Contents

1. Introduction  
2. Task Overview  
 - Objective  
 - Tools and Methods  
 - Steps Taken  
 - Challenges and Solutions  
 - Results and Outcomes  
3. Conclusion  
4. Closing Remarks  
5. Figures / Screenshots

## 1. Introduction

This report documents my initial internship assignment: a learning of the fundamentals of network security. The goal was to capture network traffic, inspect it with Wireshark, and configure simple firewall rules with Windows Defender in order to learn how network protection functions.

## 2. Task Overview

Objective of the Task

To capture and inspect network traffic (HTTP, DNS, ICMP) and configure a firewall to deny unwanted connections.

Tools and Methods Used

- Wireshark

- Windows Defender Firewall

- Kali Linux

- VirtualBox

## Steps:

Configured virtual lab (Internal Network), recorded packets, imposed filters, inspected suspicious traffic.

## Steps Taken to Fulfill the Task

1. Configured two VirtualBox VMs: Kali Linux and Windows 10, both in Internal Network.

2. Turned on Windows Defender Firewall for Windows VM.

3. Launched Wireshark on Windows VM, initiated packet capture on appropriate interface.

4. On Kali VM:

- Utilized ping to create ICMP traffic.

- Launched browser → accessed example.com to induce HTTP/DNS traffic.

5. Stopped capture after 1 minute.

6. Entered display filters in Wireshark (http, dns, icmp).

Problems Encountered and How I Overcame Them

- Initially Wireshark did not capture anything → Solved by running as Administrator.

- Internal network misconfigured → Corrected proper "Internal Network" adapter in VirtualBox settings.

## 3. Conclusion

This exercise taught me the basics of packet capture and analysis. I was taught to use display filters in Wireshark, identify typical vs. suspicious traffic, and set up an elementary firewall.

## 4. Conclusion

I am grateful to Redynox for the opportunity to intern here. This exercise provided me with actual hands-on training to secure a network. In future internships, I recommend incorporating lessons on automating the analysis of traffic.

## 5. Results and Consequences

## (1) HTTP :

## (A) HTTP Traffic Analysis – Suspicious Behavior Identified

In the packet capture session with Wireshark, I used the HTTP display filter (`http`) to analyze HTTP requests originated from my lab network.

## ✅ Sensed Suspicious HTTP Request:

• Source IP: 10.0.2.15 (Windows 10 VM)

• Destination IP: 146.75.122.172

• HTTP Method: GET

• Request URI:

/d/msdownload/update/software/defu/2025/08/am\_base\_61aadaa3518966f31280e971050ba6ca9749ec1f.exe

• Protocol: HTTP

• User-Agent: Microsoft-Delivery-Optimization/1.0.0.0

## ⚠ Why This Seems Suspicious:

1.The URL contains a lengthy executable filename (`am\_base\_61aadaa3518966f31280e971050ba6ca9749ec1f.exe`) which seems out of place and randomly generated.

2. The request is attempting to download an `.exe` file over HTTP from an external IP, which is rare for regular traffic, particularly in a controlled lab network.

3. The source IP is within the private IP range (`10.0.2.15`), usually allocated in VirtualBox internal networks — suggesting this request most probably originated from the lab VM.

4. Legitimate software updates usually employ secure HTTPS connections, not plain HTTP.

## ✅ My Opinion and Analysis:

 This HTTP GET request might signal a download of possibly malicious payload from a remote server as part of a test run or misconfigured application.

 Since the file is executable and the connection is unencrypted (no HTTPS), this poses a huge security threat in reality.

## ✅ Recommended Mitigation:

• Use strict firewall rules to prevent outgoing HTTP connections for unfamiliar applications.

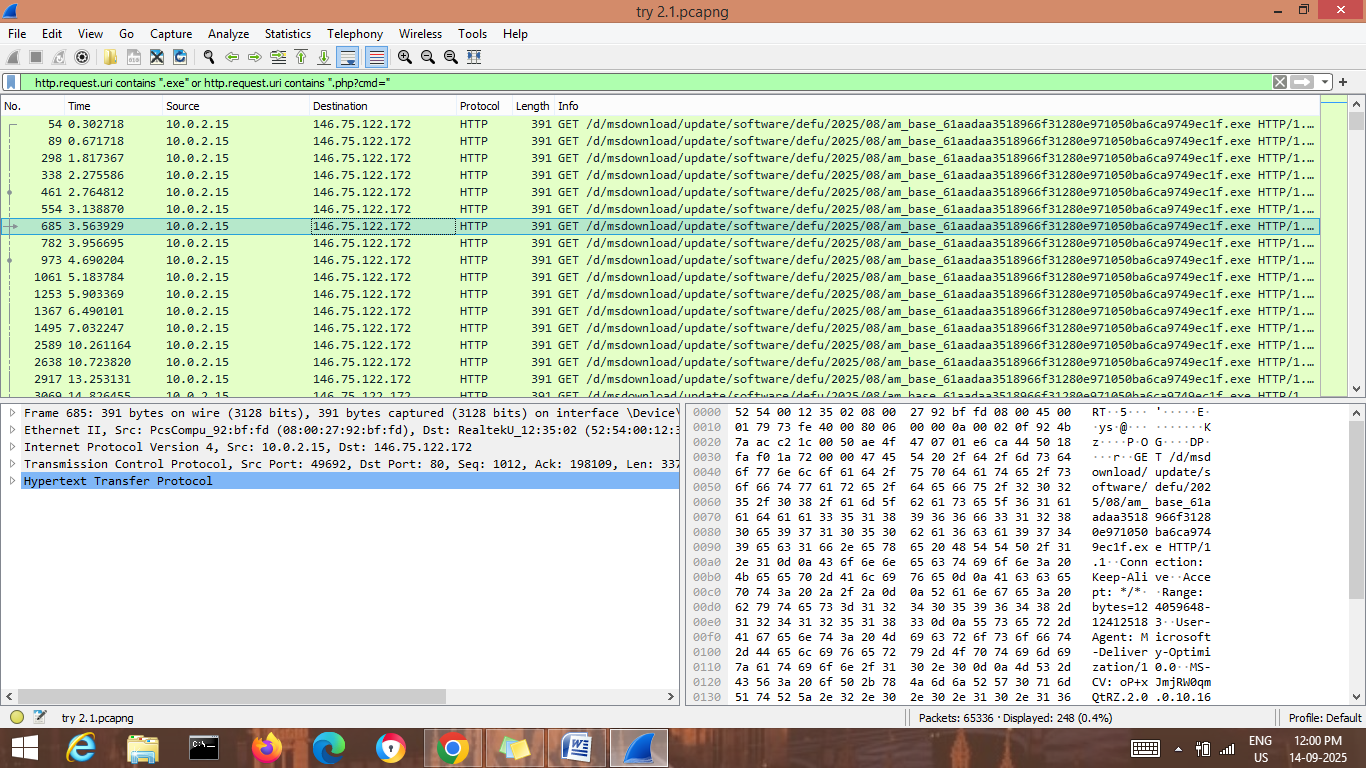
• Use HTTPS only for downloads and updates of software.

• Utilize endpoint security products to scan downloaded executables prior to execution.

• Look for repeated attempts to hit such suspicious URLs.

## Screenshot attached:

## 

Figure 1 – Suspicious HTTP GET request seen in Wireshark 

## (B) HTTP Traffic Analysis – 404 Errors Observed

I used the HTTP response code filter (`http.response.code == 404`) in the packet capture session in Wireshark to inspect failed HTTP requests made within the lab environment.

## ✅ Observed HTTP Response Behavior:

• Source IP: 44.228.249.3

• Destination IP: 10.0.2.15 (Windows 10 VM)

• Protocol: HTTP/1.1

• Response Code: 404 Not Found

• Content Type: text/html

• Web Server: nginx/1.19

## ⚠ Why This Looks Suspicious or Important:

1. Several `404 Not Found` responses were received from the external host (`44.228.249.3`), which means that the client was trying to access web resources that were not present on the server.

2. The occurrence of successive failed requests could indicate:

• An incorrectly configured or old client application attempting to download non-existent resources.

• Possible reconnaissance action where a hacker probes for files or endpoints.

3. The target (`10.0.2.15`) is an internal IP within VirtualBox, verifying the traffic came from the lab machine.

4. Genuine services will keep such repeated failed requests to a minimum — too many `404` responses can be an indicator of something unusual or suspicious.

## ✅ My Opinion and Analysis:

The frequent `404` HTTP returns reflect failed attempts to access specific files or pages. Though this may be innocuous (e.g., stale links), it can also be indicative of automated scanning or probing for resources by malware, test scripts, or poorly configured software. In a production environment, such traffic would warrant further investigation to see if the requests were benign or malicious.

## ✅ Recommended Mitigation:

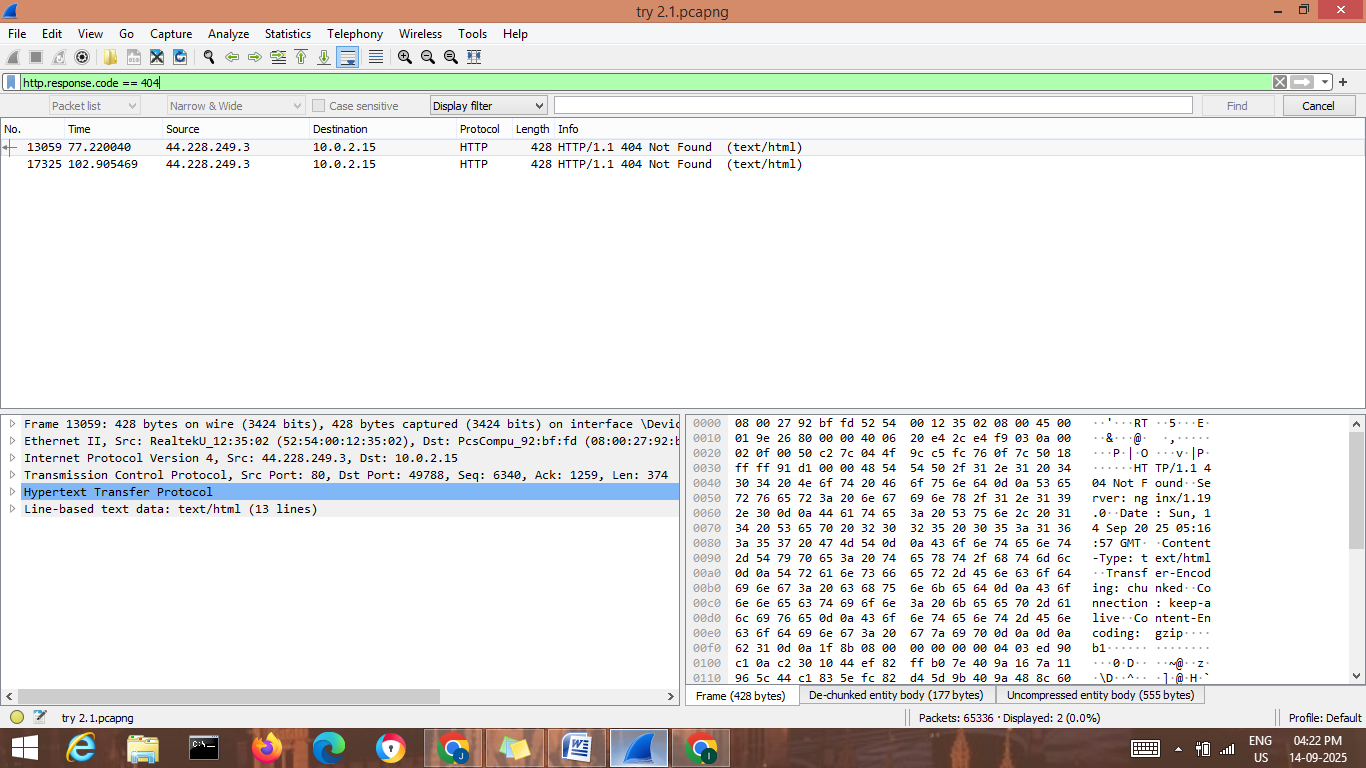
•Inspect outgoing HTTP requests for anomalous or frequent `404` messages since they can expose reconnaissance or malware activity.

•Enforce tight proxy and firewall rules to restrict entry to untrusted sites.

•Employ Intrusion Detection Systems (IDS) to identify frequent failed requests, which may signify malicious traffic.

•Install endpoint security software with the latest updates to resist unauthorized requests from being executed.

## Screenshot included:

Figure 2 – HTTP 404 Not Found responses seen in Wireshark.

**(2) DNS :-**

**(A) DNS Traffic Analysis – Failed DNS Resolutions Detected**

While performing the packet capture activity in Wireshark, I used the display filter "dns.flags.rcode != 0" This display filter displays DNS responses with an error code (e.g., `NXDOMAIN` for non-existent domain, `SERVFAIL` for server failure). These kinds of responses reflect problems with DNS resolution or efforts to connect to non-existent domains, and these may be a component of suspicious or malicious activity.

## ✅ Observed DNS Behavior:

•Source IP:192.168.142.204

•Destination IP:10.0.2.15(Windows 10 VM)

•Protocol: DNS (UDP Port 53)

•Error Type: No such name(NXDOMAIN)

•Queried Domain(s):

vellora.mi

HTTPS vellora.mi

SOA a.root-servers.net

## ⚠ Why This Looks Suspicious or Important:

1.The recurring `NXDOMAIN` replies imply the client attempting to resolve non-existent domains.

2.Malware tends to utilize Domain Generation Algorithms (DGAs) to create random or non-existent domains, trying to access its Command & Control (C2) server.

3.Valid applications typically query valid domains — failed repeated lookups (`vellora.mi`) can imply:

• A misconfigured or deprecated application.

• Malware trying to connect to a C2 infrastructure.

• Lab environment probing or testing activity.

4. Having abnormal top-level domains (for example, `.mi`) raises suspicion because they are not typically used.

The unsuccessful DNS lookups to unusual and suspicious domains are a strong indication of potential malicious or abnormal activity. Though this may be innocent (e.g., a test or typo domain), within real-world security monitoring it would need to be looked at closer to ensure no malware was beaconing out to dynamically generated domains.

## ✅ Recommended Mitigation:

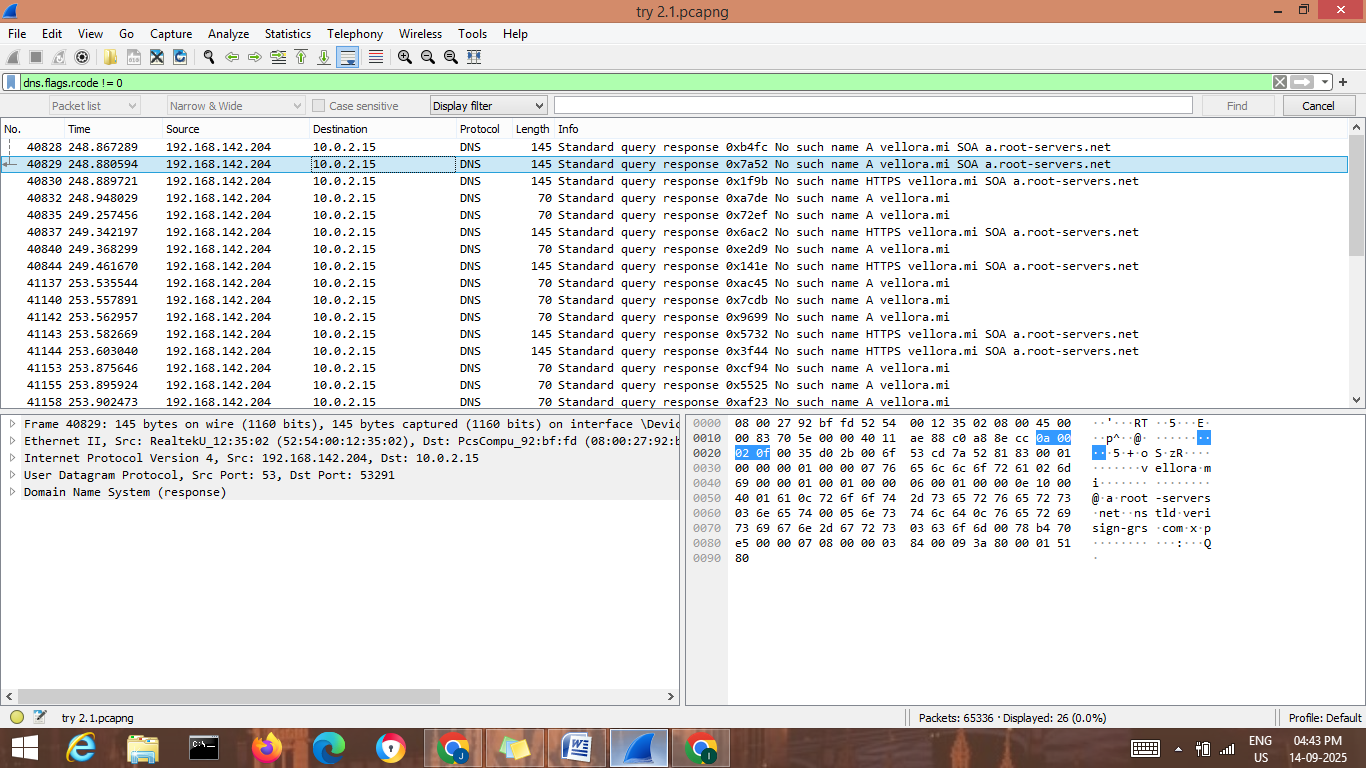
1.Enable DNS logging and monitoring to monitor unsuccessful lookups for strange or suspicious domains.

2. Utilize threat intelligence feeds to identify if the queried domains (`vellora.mi`) are linked to known malicious campaigns.

3. Utilize DNS filtering solutions (such as Cisco Umbrella, Quad9, or Pi-hole) to prevent access to suspicious domains.

4. Check for strange processes on endpoints that are producing large numbers of failed DNS queries.

## Screenshot provided:

Figure 3 – DNS NXDOMAIN responses seen in Wireshark

**B) DNS Traffic Analysis – Access to Web Domain That Is Vulnerable**

During the packet capturing session in Wireshark, DNS query and response about `www.vulnweb.com` were seen. This web domain is a popular security test site (which is maintained for security training and pen-testing purposes).

## ✅ DNS Behavior Seen:

• Source IP: 10.0.2.15(Windows 10 VM)

• Destination IP (DNS Resolver): 192.168.142.204

• Queried Domain: www.vulnweb.com

• DNS Response: A record → 44.228.249.3 (resolved successfully)

## ⚠ Why This Looks Suspicious or Important:

1. `www.vulnweb.com` is not a typical browsing location; it is made deliberately vulnerable for web security testing.

2. Requests to such domains suggest:

• Lab penetration testing activity (expected in training labs).

• Or, in a live network, potential malicious activity, where an intruder or malware is trying to exploit/test a vulnerable application.

3. The domain resolved successfully to an IP (`44.228.249.3`), which means the client was successful in reaching the host.

4. Under enterprise security monitoring, unusual queries to such recognized vulnerable domains would be a serious red alert.

## ✅ My Opinion and Analysis:

The ensuing DNS traffic clearly indicates intentional security probing within the lab. Harmless in the context of a training setup, the same action within a production corporate network would be either unauthorized penetration testing or malicious probing to probe vulnerabilities.

## ✅ Suggested Mitigation:

1. In testing labs: Record the activity as part of training and keep it segregated from production systems.

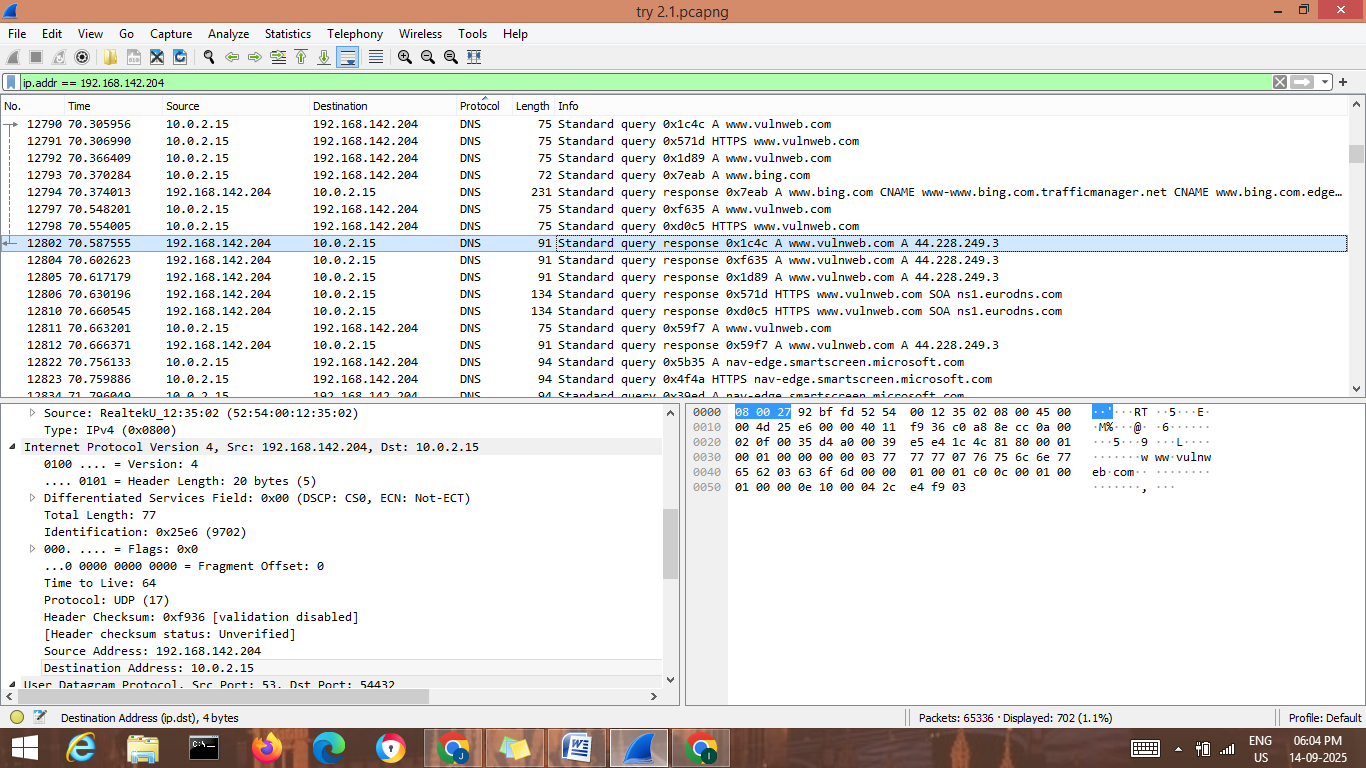
2. In live networks:

• Flag and alert on DNS requests to known vulnerable sites (`vulnweb.com`).

• Inspect the process/user making the request.

• Implement DNS filtering to prevent user endpoints from accessing test/malicious sites.

## Screenshot attached:

Figure 4 – DNS resolution of `www.vulnweb.com` captured in Wireshark

**(c) ICMP :**

•Capture tool: Wireshark (ICMP filter used — top left indicates `icmp`).

•Source IP (ping source): `10.0.2.15` (probably a VM).

•Destinations seen: `142.251.42.36` and `23.33.54.93` (example remote IPs — at least one is in the `142.251.42.0/24` block, frequently Google/Bing).

•Protocol: ICMP (Echo (ping) request / reply).

## Packet details:

* ICMP Echo request and reply pairs.
* ID field = `0x0001` in many frames.
* Increasing sequence numbers: `seq=1/256`, `seq=2/512`, `seq=3/768`, etc. (Info column displays seq and reply pairing).
* Displayed TTL values: request `ttl=128`, reply `ttl=117` (reply TTL smaller → different OS/hop counts or NAT).
* Payload content (hex/ASCII pane): visible ASCII pattern — readable sequence of characters (`abcdef.`, `ghijklmn.`, etc). This indicates the ICMP payload includes structured ASCII data (not random).

Frame sizes: 74 bytes on wire (standard ping size + payload).

## Potentially suspicious :

•Repeated ICMP Echo payloads with structured ASCII might be harmless,' or ' it might be ICMP-based covert channel / tunneling (attacker encoding information in ICMP payloads to exfiltrate information or issue command-and-control).

•Indicators of concern:

oNon-random ASCII payload repetition in numerous packets.

oMuch traffic to external IP addresses (potential persistent channel).

oAlways `id=0x0001` with monotonically incrementing sequence numbers and repeated pattern — typical of automated tool or tunneling session.

## Observed Outcomes & Evidence:

1.Echo requests and responses are pairing up properly — connectivity is established between `10.0.2.15` and the remote IP addresses.

2.Console / hex window verifies ASCII data (`abcdef.`), not encrypted or compressed data.

3.Reply TTL is smaller than request TTL— signifies reply went through more hops or comes from a host with smaller initial TTL (or some NAT/inspection device changed TTL).

4.Packet sizes (74 bytes) indicate small sized payloads for command/control or lightweight exfil (chunked).

## Why this might be harmful (impact)

1.ICMP tunneling / hidden channel: ICMP payloads can be utilized to encapsulate arbitrary data. Attackers exploit ICMP to bypass firewalls that permit ping but prevent higher-layer traffic.

2.Data exfiltration: Binary/text within ICMP payloads can transport stolen data out of a network in little pieces.

3.Command & Control (C2): Certain malware employs ICMP to get commands and provide output.

4.Hard to detect unless payload inspection is done: Numerous networks overlook ICMP payloads; basic signatureless tunneling can sneak past detection.