TP- Phase 2

CENG 519 Network Security

Spring semester 2024/2025

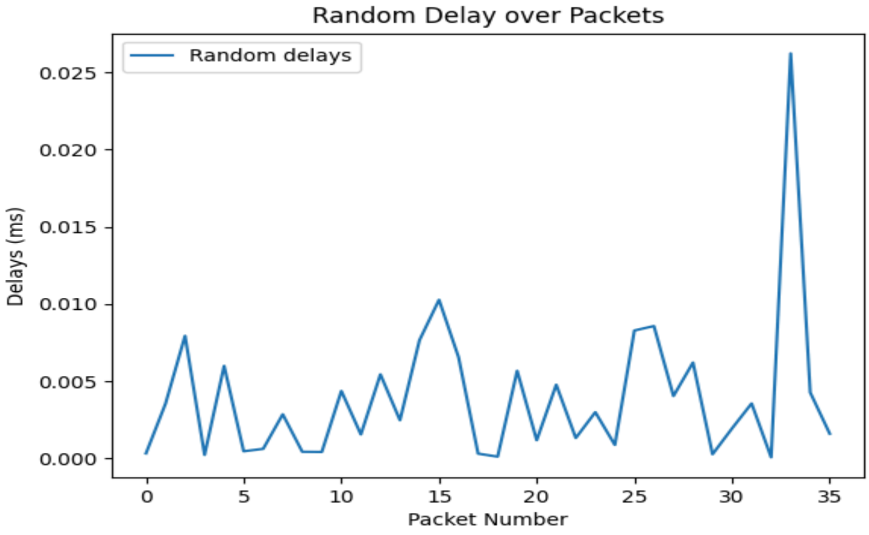
Ihssane EL JAZOULI

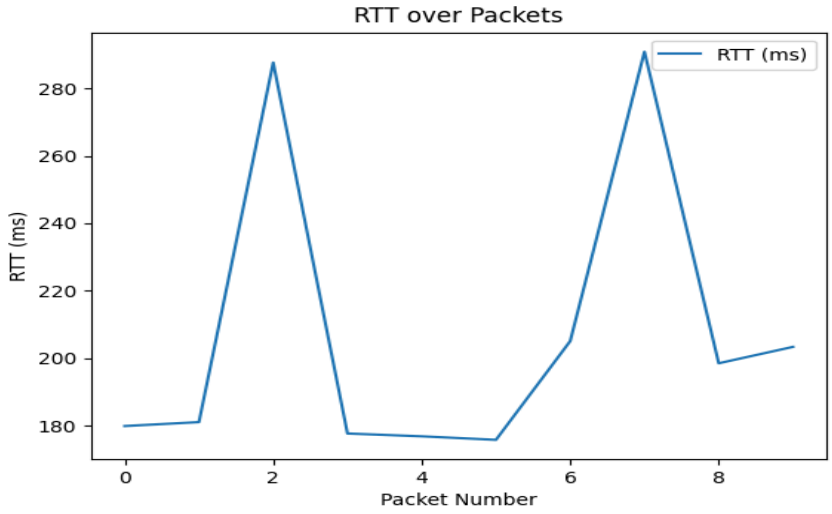
I. Introduction :

The objective of this phase is to demonstrate the feasibility of a **covert communication channel** through manipulation of **inter-packet delay** in ICMP traffic. The receiver extracts information that is hidden by the sender, who alters the time between ICMP echo requests, and uses it to rebuild the original message.
Before I proceed to execute the covert channel, I resolved some issues from Phase 1 and proceeded to execute the covert sender and receiver.

II. Phase 1 Fixes :

In the first phase, I made a few mistakes that affected the accuracy of my results. Most notably, I was mistakenly running the mim container, which generated excessive traffic and interfered with my measurements.
I also realized that my plots weren't being saved correctly. To fix this, I added the right command to save the generated plots as files. This allowed me to visualize and analyze the results more easily.
Here are the generated plots:





III. Phase 2 :

A. *Implementing covert channel via ICMP timing :*

For this phase, I implemented a covert communication system that encodes binary data into the timing between ICMP packets, using the same Docker-based environment as in Phase 1. Here's how each component works:

Sender Logic :

- Converts the message into binary (for example `"hi"` → `0110100001101001`)
- Sends ICMP echo requests (`ping`) to the `insec` container
- Encodes each bit using a delay:
    - **100 ms** for bit `0`
    - **300 ms** for bit 1

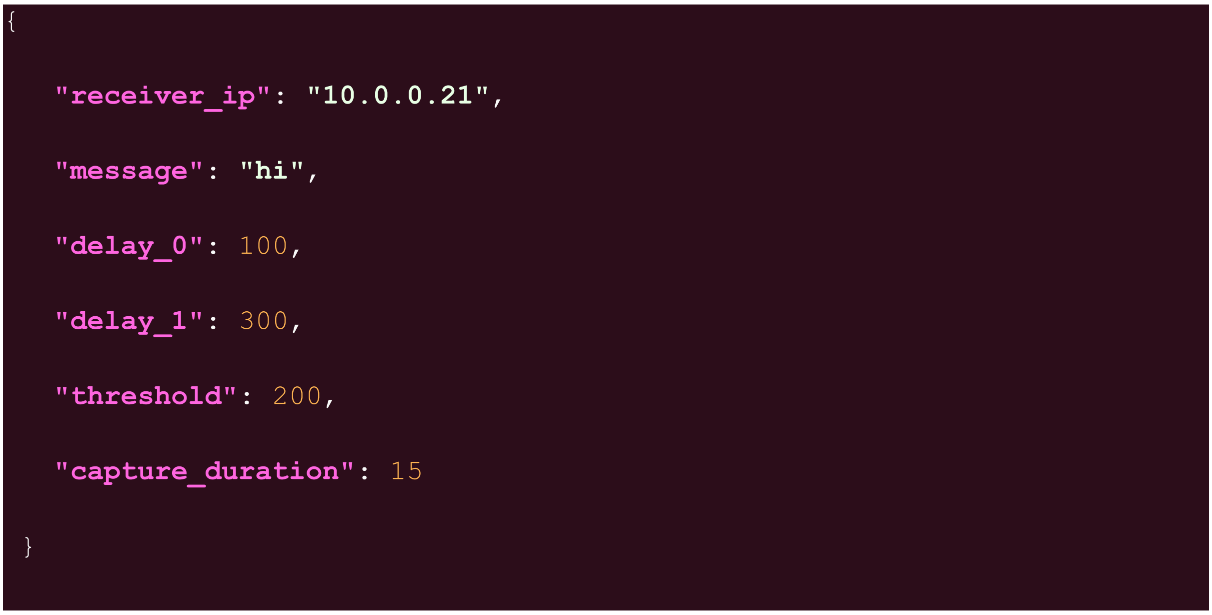
Receiver Logic :

- Listens for incoming ICMP echo requests
- Records timestamps of each packet
- Calculates the delay between consecutive packets
- Uses a threshold of **200 ms** to distinguish between `0` and 1

- Reconstructs the binary stream and decodes the original message

Configuration File:

Both sender and receiver read from a shared `config.json` file for flexibility. Here's an example configuration:



```
{

   "receiver_ip": "10.0.0.21",

   "message": "hi",

   "delay_0": 100,

   "delay_1": 300,

   "threshold": 200,

   "capture_duration": 15

 }
```

B. *Experimentation campaign :*

To evaluate the effectiveness and reliability of the covert channel implementation, I conducted an experimentation campaign where I varied parameters such as inter-packet delays (delay_0, delay_1) and decoding threshold values.
I noticed that if delays are too small, the decoded message at the end is wrong. This can be explained by the delay of the delay-processor (eg: delay_0: 30; delay_1: 50; threshold :40)
And also if the values are so close to each other (eg : delay_0: 200; delay_1: 250; threshold: 225)
In addition, capture duration is an issue. It is necessary to know the estimated length of the message before.
For these parameters (delay_0: 100; delay_1: 300; threshold :200) the results were always correct.

IV. Conclusion :

This phase demonstrates that **covert timing channels** can be effectively created using inter-packet delays in common protocols like ICMP. Even in environments without packet payload access, timing variations can leak information.