TP report

CENG 519 Network Security

Spring semester 2024/2025

Ihssane EL JAZOULI

I. Introduction :

This project investigates covert communication techniques in network environments using ICMP inter-packet timing modulation and presents a heuristic-based detector. The project was divided into three phases:

1. **Phase 1: Delay Processor**
2. **Phase 2: Covert Channel Design**
3. **Phase 3: Covert Channel Detection**

Each phase involved design, implementation, experimentation, and reporting.

II. Phase 1:

Objective: Simulate timing manipulation by introducing random delays to network packets.

Implementation:

- A Python-based packet processor was developed.
- It subscribed to a NATS topic (`inpktsec`), added a random delay, and forwarded packets.
- Random delays followed a normal distribution.

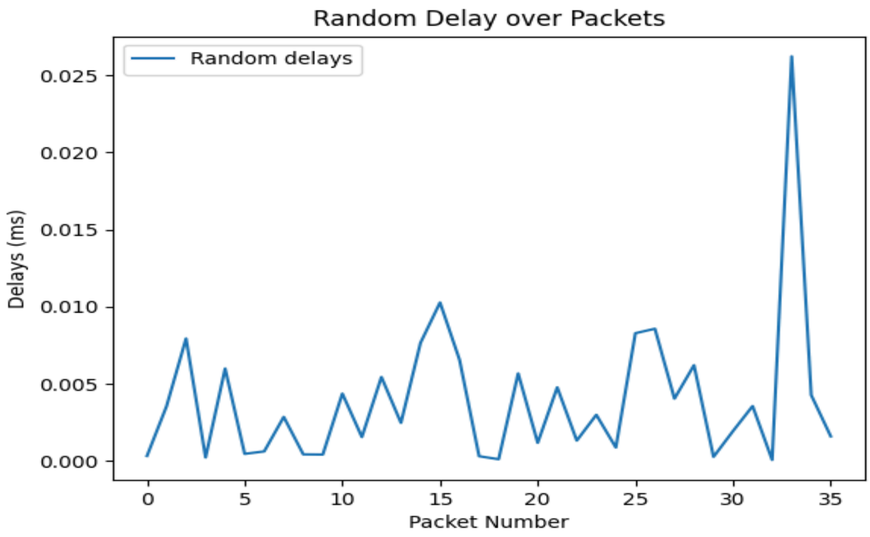
Experiment:

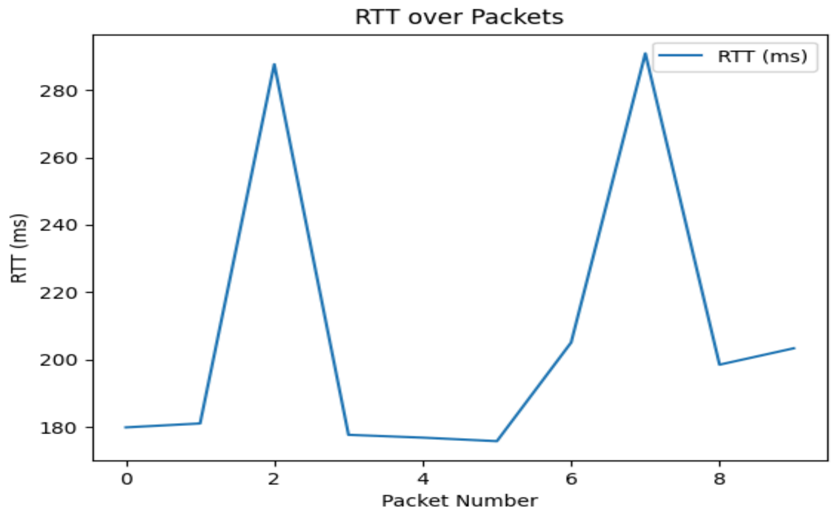
- RTTs were measured for different average delays.
- A graph showing average RTT vs. mean delay was plotted.

Outcome:

- RTTs were measured for different average delays.
- A graph showing average RTT vs. mean delay was plotted.

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. Phase 3 :

Objective: Detect timing-based covert channels by analyzing inter-packet delays.

Detector Logic:

- Sniffs ICMP packets using Scapy.
- Extracts timestamps and computes:
    - Mean delay
    - Standard deviation
    - Number of delay spikes (ΔIPD > 150ms)
- Covert activity is flagged if:
    - `std > 0.05` and `spikes >= 3`

Experiment Campaign:

- Conducted 10 experiments:
    - 5 with covert sender
    - 5 with normal sender

Results Summary:

- **Covert runs:**
    - Detected: 5
    - Not detected: 0
- **Normal runs:**
    - Detected as covert (False Positive): 2
    - Correctly not detected (True Negative): 3

Metrics:

- **True Positives (TP):** 5
- **False Positives (FP):** 2
- **True Negatives (TN):** 3
- **False Negatives (FN):** 0

Derived Metrics:

- **Accuracy:** (TP + TN) / Total = (5 + 3) / 10 = 0.80 (80%)
- **Precision:** TP / (TP + FP) = 5 / (5 + 2) = ~0.714
- **Recall:** TP / (TP + FN) = 5 / (5 + 0) = 1.0
- **F1 Score:** 2 * (P * R) / (P + R) = 2 * (0.714 * 1) / (0.714 + 1) = ~0.833
- **95% Confidence Interval for Accuracy (Wilson):** ~[0.52, 0.96]

V. Conclusion :

This project successfully demonstrated how inter-packet timing can be exploited to build a covert communication channel and how such channels can be detected using lightweight, interpretable statistical heuristics. The implemented detector showed a strong ability to identify covert traffic, with a recall of 100% and an F1 score of 83.3%. However, the presence of false positives suggests room for improving the detector's precision. Fine-tuning thresholds or using adaptive models could help balance sensitivity and specificity. Overall, the project offers a practical foundation for understanding timing-based covert channels and inspires further research into robust detection mechanisms.