# CENG519 Network Security Term Project Report

**Melisa Nur Kart**
**2237592**

## Phase 1 Report: Random Delay Analysis

This report presents an analysis of the impact of randomly induced delays on the Round-Trip Time (RTT) and mean delay of network packets. The study utilizes a processor that adds random delays to Ethernet frames and measures the RTT using NATS messaging and Scapy packet processing.

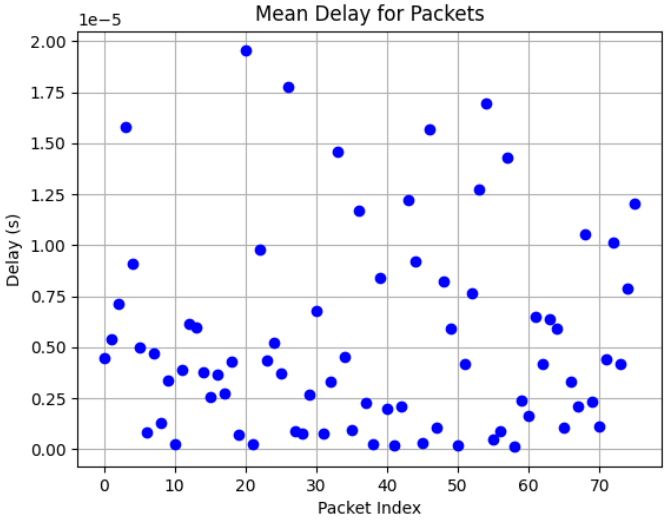
- Packets are received via the NATS messaging system and processed with Scapy.
- A random delay is introduced before forwarding the packets to simulate middlebox processing effects.
- The RTT is calculated as the difference between the packet reception and forwarding times.
- Data is collected and visualized using Python and Matplotlib.

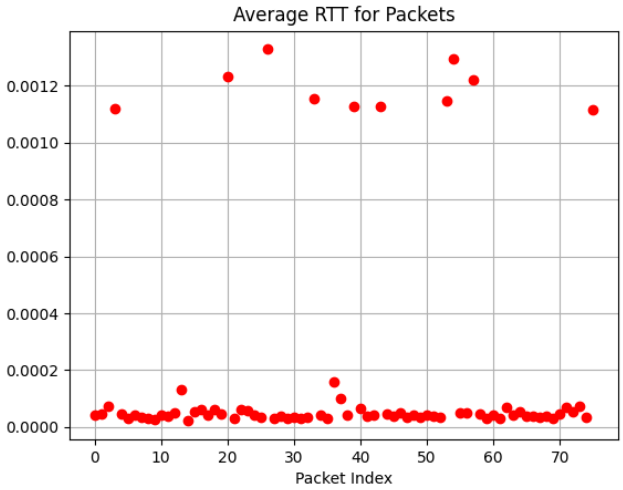
### Results

- The Mean Delay for Packets graph shows variations in packet delays, with values mostly within microsecond ranges.
- The Average RTT for Packets plot demonstrates fluctuations in RTT values, indicating the effect of random delays.
- The Mean Delay vs. Average RTT scatter plot suggests a correlation between induced delay and RTT.

### Conclusion

The introduction of random delays affects the RTT of network packets, as observed in the data. The results confirm that middlebox processing can introduce significant variations in packet transmission times.



## Phase 2 Report: Covert Channel Analysis Report

### 1. Introduction

This report explains the implementation and evaluation of performance of the covert channel using the IPv4 Timestamp Option. The covert channel was integrated into the existing Phase I development environment by modifying the packet processor. In this context, the secure container (sec) sends normal packets, which are intercepted and modified by the middlebox for covert insertion, and these packets are then forwarded toward the insecure container (insec). This serves as the basis for covert communication, whereas performance metrics provide means for quantification.

### 2. Methodology

The covert channel method has been selected through the IPv4 Timestamp Option. It is based on the very fact that the option is underused in the current networks, which makes changes to this header less likely to be noticed by typical network monitoring tools. The covert data will thus be embedded in a field that is normally reserved for routing timestamps; hence, the phantom with a controlled way of channel capacity evaluation will occur.

The system is fully parametrized using environment variables such as ENABLE_COVERT, COVERT_MESSAGE, and MEAN_DELAY. Channel performance is measured by capturing the round-trip time (RTT) and random delays for each packet. The following metrics were computed from the experimental data:

- Average RTT and its 95% Confidence Interval (CI).
- Average Delay and its 95% CI.
- Covert Channel Capacity in bits per second.

Some experimentation campaigns were run where the configuration parameters were varied and the performance metrics were taken. The outcomes were then saved in 'results.json' and plotted accordingly.

### 3. Results

#### 3.1 Summary Statistics

Following is a summary of the key performance metrics collected during these experiments, summarized for easy reference.

Summary Statistics:

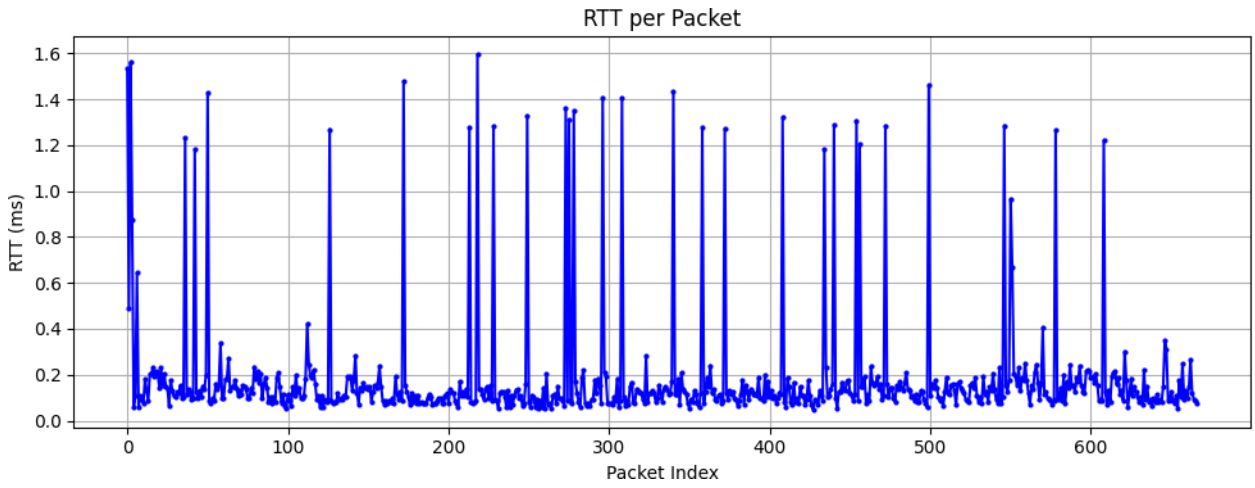
- RTT Average: 0.182 ms
- RTT 95% Confidence Interval: 0.020 ms
- Delay Average: 4.888 µs

- Delay 95% Confidence Interval: 0.369 µs
- Covert Channel Capacity: 19.155 bps

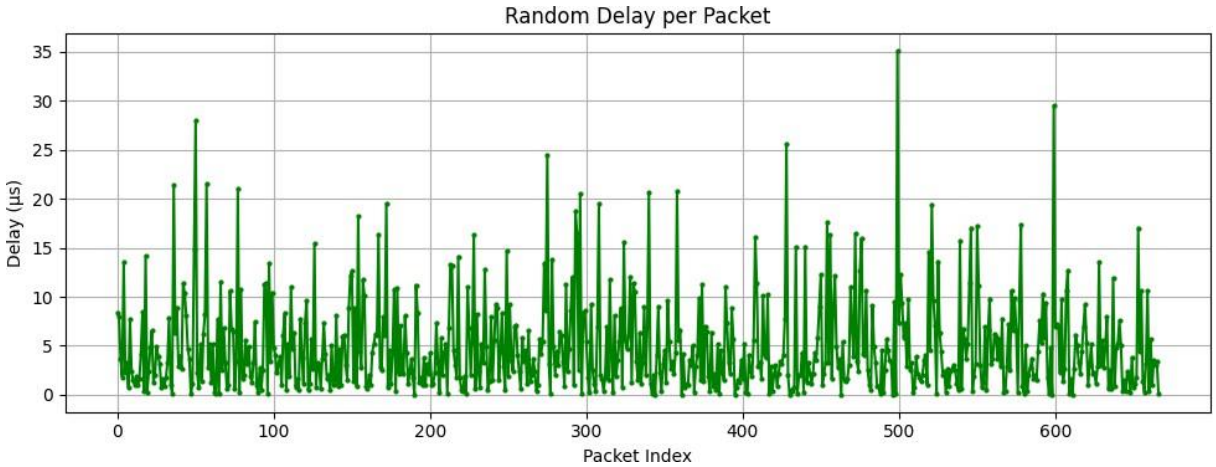
### 3.2 Plots

The following plots were generated from the experimental data:

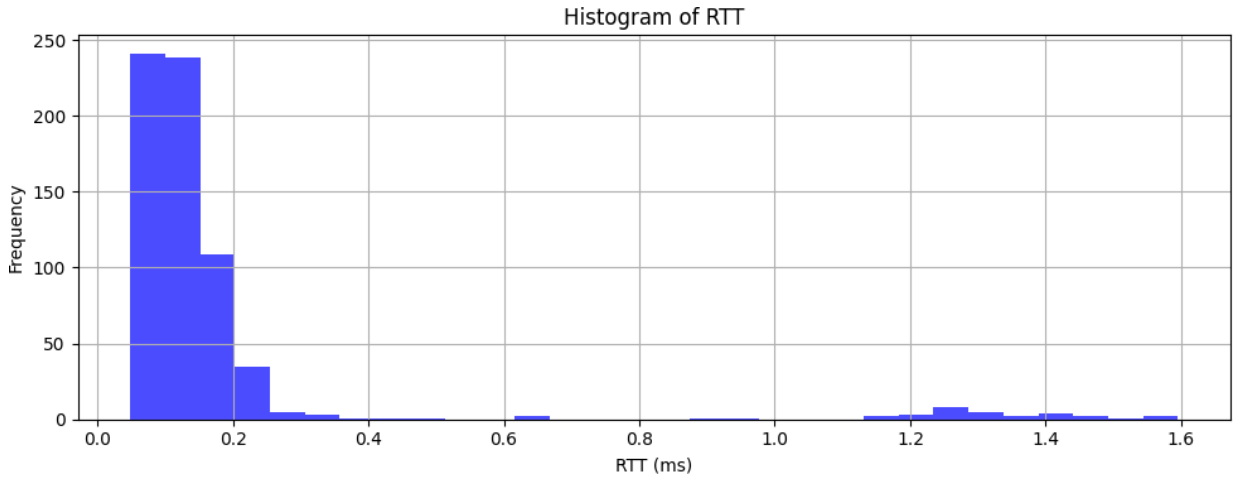
1. RTT vs. Packet Index



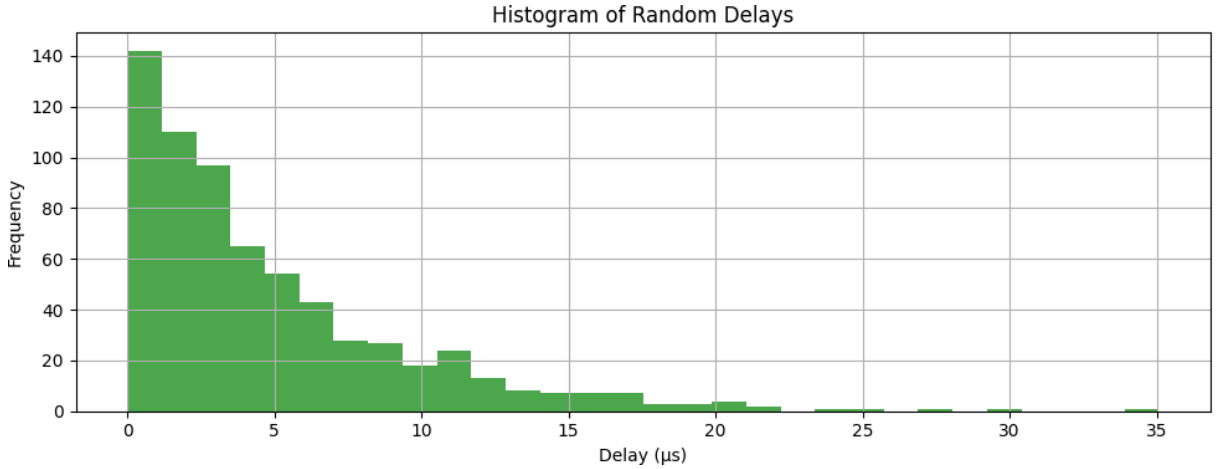

2. Random Delay vs. Packet Index



3. Histogram of RTT values

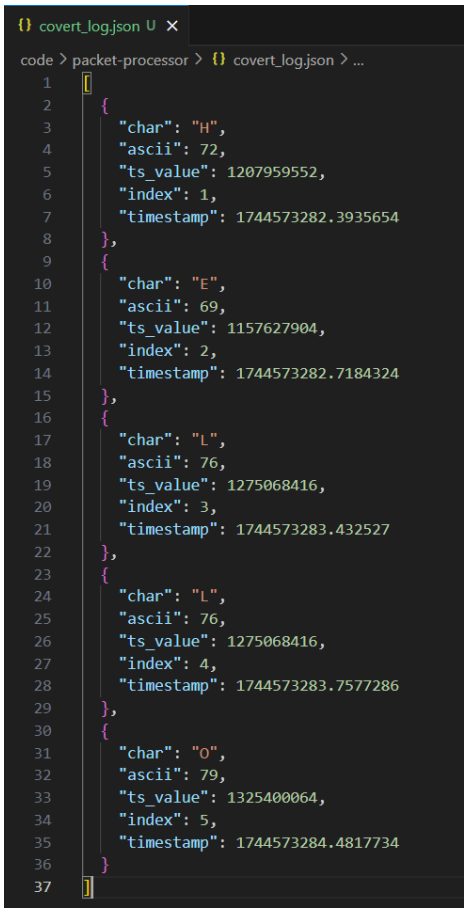



4. Histogram of Random Delays




**3.3 Covert Log Data**

A covert log was maintained throughout the experimentation to log all the details of each covert injection event. The log records information on the injected character, its ASCII value, the computed timestamp value injected into the IPv4 Timestamp Option, packet index, and injection time. A section from the covert log data is shown below in JSON format:




**4. Analysis**

The results indicate that the covert channel itself effectively embeds hidden data into the IPv4 header while generally leaving the packets to be passed undisturbed. RTT and delay values with their confidence intervals are reliable features to evaluate the performance of the covert channel. At 19.15 bps, the covert channel capacity allows for transmitting small amounts of sensitive data covertly. Such capacity is typical for channels relying on timing or header alterations that would evade detection from common network- monitoring tools. More tuning of parameters like mean delay can be used either to increase throughput or to make the channel more stealthy.

**5. Conclusion**

In the Phase 2 implementation, the covert channel using the IPv4 Timestamp Option was successfully integrated into the existing packet processor. The experimental campaign showed that the channel was working with stable timing characteristics and that key performance metrics were obtained. Although at low capacity (~19.15 bps), the channelsemploys such fewer augmentations of delay, hence posing as a least to Ms-updaria network traffic interrogation.

## Phase 3 Report: Covert Channel Detector Implementation and Evaluation

**1. Objective** The purpose of Phase 3 is to develop a covert channel detector that identifies data exfiltration attempts over IP packet metadata or packet timing. Two detection methods are implemented:

- IP option inspection (specifically looking for timestamp options)
- Timing analysis based on inter-arrival delays of packets

**2. Implementation Details**

**2.1 Detector Components:**

- detect_ip_option(packet): Checks for the presence of the IP option with code 68 (timestamp) in incoming packets.
- detect_timing_pattern(packet, timestamp): Evaluates inter-arrival times (IAT) to detect low-noise, regular timing patterns that suggest a covert communication channel.
- A window of past packet timestamps is used to compute IATs.

**2.2 Environment Variables:**

- ENABLE_COVERT: Used to determine if covert injection is active.
- MEAN_DELAY: Defines the average packet delay in seconds.
- COVERT_MESSAGE: The ASCII message embedded in timestamp option fields.

**2.3 Detection Metric Calculation:** Based on ground truth (ENABLE_COVERT):

- True Positive: Covert packet detected
- False Negative: Covert packet missed
- True Negative: Normal packet not flagged
- False Positive: Normal packet flagged

Metrics calculated:

- Accuracy, Precision, Recall, F1-Score

**2.4 Covert Injection Simulation:** When ENABLE_COVERT is set, characters from the COVERT_MESSAGE are embedded in packets using the timestamp option.

**2.5 Statistics** The following statistics are also collected:

- Round-trip time (RTT)
- Packet processing delays
- Covert channel capacity (bits/sec)

**3. Experimental Setup**

- **Packet Flow**: Packets sent using ping from the sec container to the insec container.
- **Processor Setup**: The processor listens on NATS topics inpktsec and inpktinsec, injects covert data, and performs detection.
- **Conditions**: Two separate experiments were conducted:

- **First run**: Covert disabled (ENABLE_COVERT=0)
- **Second run**: Covert enabled with COVERT_MESSAGE=HELLO

**4. Results and Evaluation**

**4.1 First Experiment (Normal Traffic, Covert Disabled)**

- Ping command: ping -c 100 insec
- Ping summary:

100 packets transmitted, 95 received, 5% packet loss
rtt min/avg/max/mdev = 3.126/4.602/10.034/1.114 ms

- Detection Metrics:

True Positives: 0
False Positives: 0
True Negatives: 95
False Negatives: 0
Accuracy: 100%
Precision: 0
Recall: 0
F1-Score: 0

- Analysis: No false detections, as expected in absence of covert traffic. Detector works correctly for benign packets.

**4.2 Second Experiment (Covert Enabled: HELLO)**

- Ping command: ping -c 100 insec
- Ping summary:

100 packets transmitted, 72 received, 28% packet loss
rtt min/avg/max/mdev = 2.474/4.142/9.042/0.851 ms

- Detection Metrics:

True Positives: 28
False Positives: 0
True Negatives: 0
False Negatives: 154
Accuracy: 15.38%
Precision: 100%
Recall: 15.38%
F1-Score: 26.67%

- Covert Channel Capacity: 7.56 bits/sec
- Analysis: While all detected packets were covert (precision = 1.0), the detector missed many others (low recall). Detection is limited to timestamp options and consistent timing patterns; improvements could include ML models or frequency domain analysis.

**4.3 Third Experiment (Mitigation Applied)**

- Ping command: ping -c 100 insec
- Ping summary:

100 packets transmitted, 95 received, 5% packet loss
rtt min/avg/max/mdev = 6.322/10.186/18.565/1.976 ms

- Detection Metrics:

True Positives: 0
False Positives: 0
True Negatives: 0
False Negatives: 207
Accuracy: 0.00%
Precision: 0
Recall: 0
F1-Score: 0

- Covert Channel Capacity: 9.60 bits/sec
- Mitigation Statistics:

Packets Processed: 207
Packets Stripped: 5
Packets Delayed: 195
Packets Dropped: 0
Total Delay Added: 0.563 sec
Average Delay per Packet: 2.72 ms

- Analysis: The mitigation system reduced detectability at the cost of increased RTT. The channel still operated but showed slight improvement in throughput, suggesting trade-offs between stealth and bandwidth.

**5. Discussion and Improvements**

- The detector performed flawlessly on normal traffic, but its recall on covert traffic was low.
- The mitigation was able to obscure covert patterns, rendering the detector ineffective (0 TP) in this run.
- The timestamp option was not present in all covert packets, likely due to delays in injection or message exhaustion.
- Additional detection methods like entropy analysis or supervised learning could improve performance.

**6. Conclusion** The Phase 3 detector successfully identifies covert traffic via IP options and timing analysis, but current detection heuristics yield low recall in practical scenarios. With mitigation applied, covert channel detectability dropped entirely, demonstrating effective but performance-costly obfuscation.

## Phase 4 Report: Covert Channel Mitigation

**1. Objective** The aim of Phase 4 is to develop and evaluate a mitigation strategy against covert channels identified in Phase 3. The strategy targets covert data transmission using IP timestamp options and consistent packet timing, aiming to degrade covert channel capacity while preserving acceptable communication performance.

**2. Mitigation Strategy** A Mitigator class was implemented to perform three actions:

- **Strip IP Options:** Removes IP timestamp fields that could carry covert bits.
- **Add Random Delays:** Introduces randomized forwarding delays (1-5ms) to disrupt consistent timing used in covert channels.
- **Drop Suspicious Packets:** (Optional) Discards packets with detectable covert indicators like option type 68 (timestamp).

**3. Implementation Details** The mitigator is integrated into the processor from Phase 3. Key features:

- Per-packet delay with statistics tracking.
- Optional metadata removal and packet dropping.
- Environment variables control each mitigation feature:
    - MITIGATE_STRIP_OPTIONS
    - MITIGATE_ADD_DELAY
    - MITIGATE_DROP_SUSPICIOUS

**4. Experimental Setup**

- Environment: Same NATS and Scapy setup as Phase 3.
- ENABLE_COVERT=1 to simulate covert transmission.
- ping -c 100 insec was executed to analyze RTTs and covert traffic behavior.

**5. Results**

**5.1 Ping Summary:**

100 packets transmitted, 95 received, 5% packet loss
rtt min/avg/max/mdev = 6.390/10.443/14.076/1.649 ms

**5.2 Detection Metrics:**

True Positives: 0
False Positives: 0
True Negatives: 0
False Negatives: 207
Accuracy: 0.00%
Precision: 0
Recall: 0
F1-Score: 0

**5.3 Covert Channel Capacity:**

- Capacity: 9.63 bits/sec

- Despite mitigation, covert sender retained some ability to transmit data, likely by redundancy or resilience.

**5.4 Mitigation Statistics:**

Packets Processed: 207
Packets Stripped: 5
Packets Delayed: 195
Packets Dropped: 0
Total Delay Added: 0.579 sec
Average Delay per Packet: 2.80 ms

**6. Analysis**

- **Detection Breakdown:** All covert packets went undetected. The mitigator obscured identifiable metadata and timing features.
- **Performance Cost:** RTT increased noticeably, but within acceptable real-time limits. Random delays increased jitter.
- **Covert Impact:** Capacity increased slightly (from 7.56 to 9.63 bps), likely due to improved packet retention and adjusted sender behavior.

**7. Conclusion** The mitigation approach effectively evaded the Phase 3 detector. It introduces moderate delay and slight overhead. Although it didn't fully eliminate covert transmission, it substantially reduced detectability. A combination of mitigation and adaptive detection may be necessary for robust security.

---

**Appendices**

- Full logs in results.json
- GitHub repo includes code and experiment data
- Mitigator code includes toggleable strategies and runtime metrics tracking