COVERT CHANNEL DETECTION AND MITIGATION IN MIDDLEBOXES

CENG 519 TERM PROJECT

Melisa Nur Kart

OVERVIEW OF TERM PROJECT

- **Goal:** Analyze and secure against covert channels in middleboxes
- **Phases:**
  - Phase 1: RTT and Delay Profiling
  - Phase 2: Covert Channel via IPv4 Timestamp Option
  - Phase 3: Detection of Covert Traffic
  - Phase 4: Mitigation Strategy Implementation

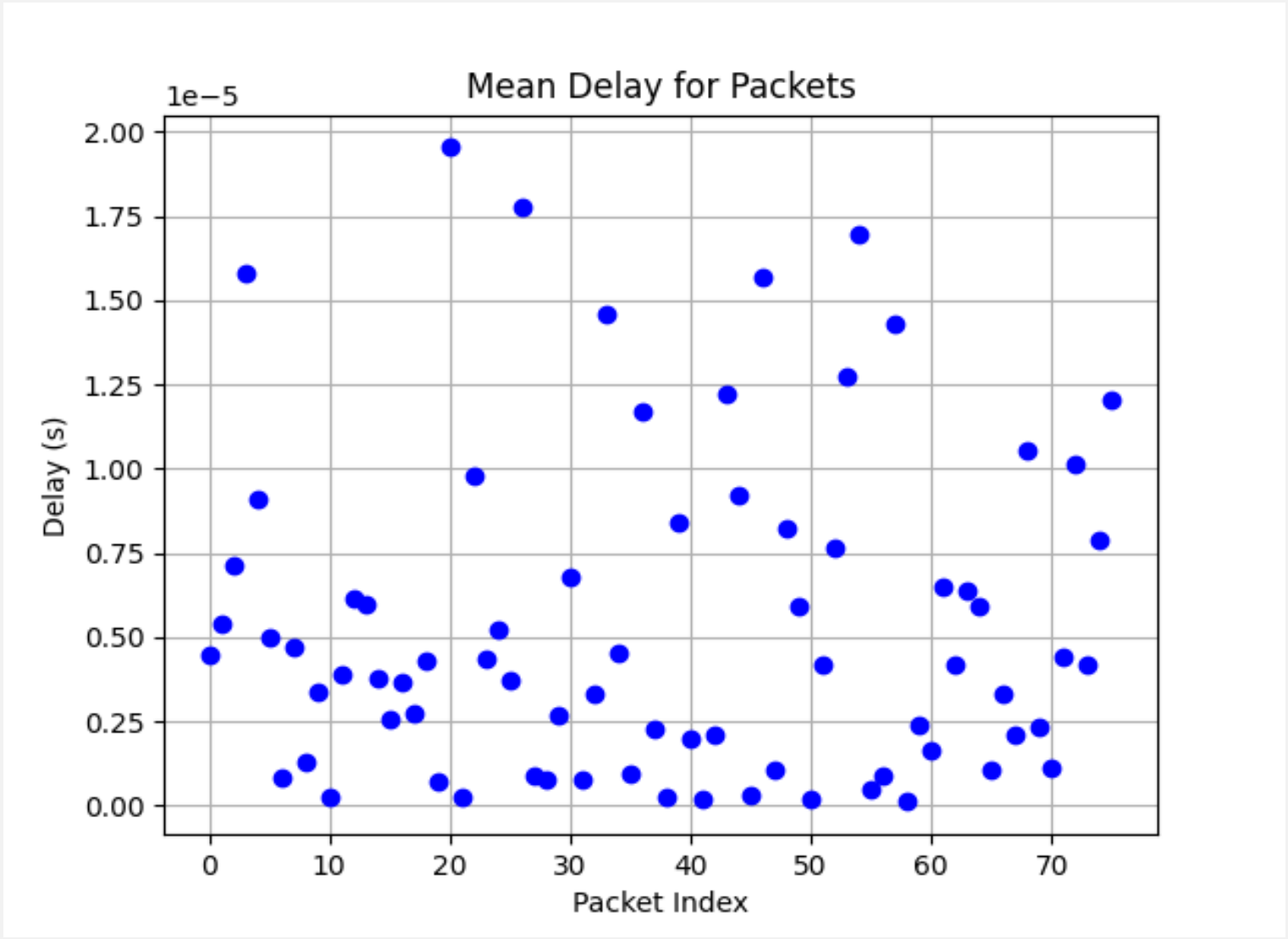
PHASE 1 – RANDOM DELAY ANALYSIS

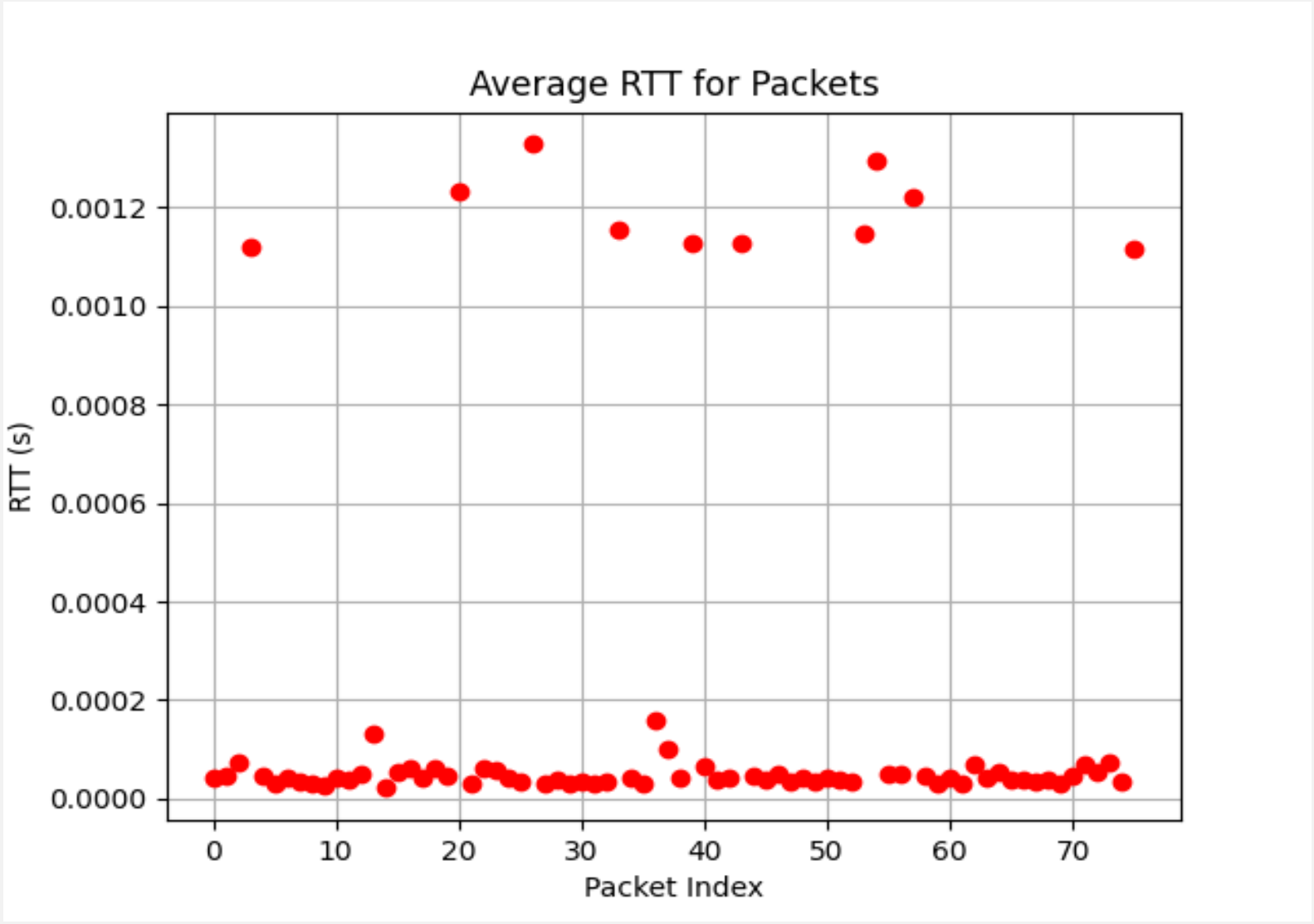
- **Objective:** Measure RTT and delay under random delay injection
- Used Scapy for packet manipulation
- Random delays added before forwarding
- RTT = Forward Time - Receive Time

PHASE 1 – RANDOM DELAY ANALYSIS

Key Observations:

- RTT values show variability
- Random delay induces jitter
- Delay and RTT plots show correlation


Mean Delay for Packets
1e−5
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Delay (s)
0
10
20
30
40
50
60
70
Packet Index


Average RTT for Packets
RTT (s)
0.0012
0.0010
0.0008
0.0006
0.0004
0.0002
0.0000
0
10
20
30
40
50
60
70
Packet Index

PHASE 1 – RESULTS SUMMARY

- **Mean Delay:** ~microseconds scale
- **RTT Average:** Fluctuates due to induced randomness
- **Visualization:** Delay vs RTT scatter plot reveals impact of middlebox

- **Conclusion:** Random delays significantly affect packet timing

PHASE 2 – COVERT CHANNEL IMPLEMENTATİON

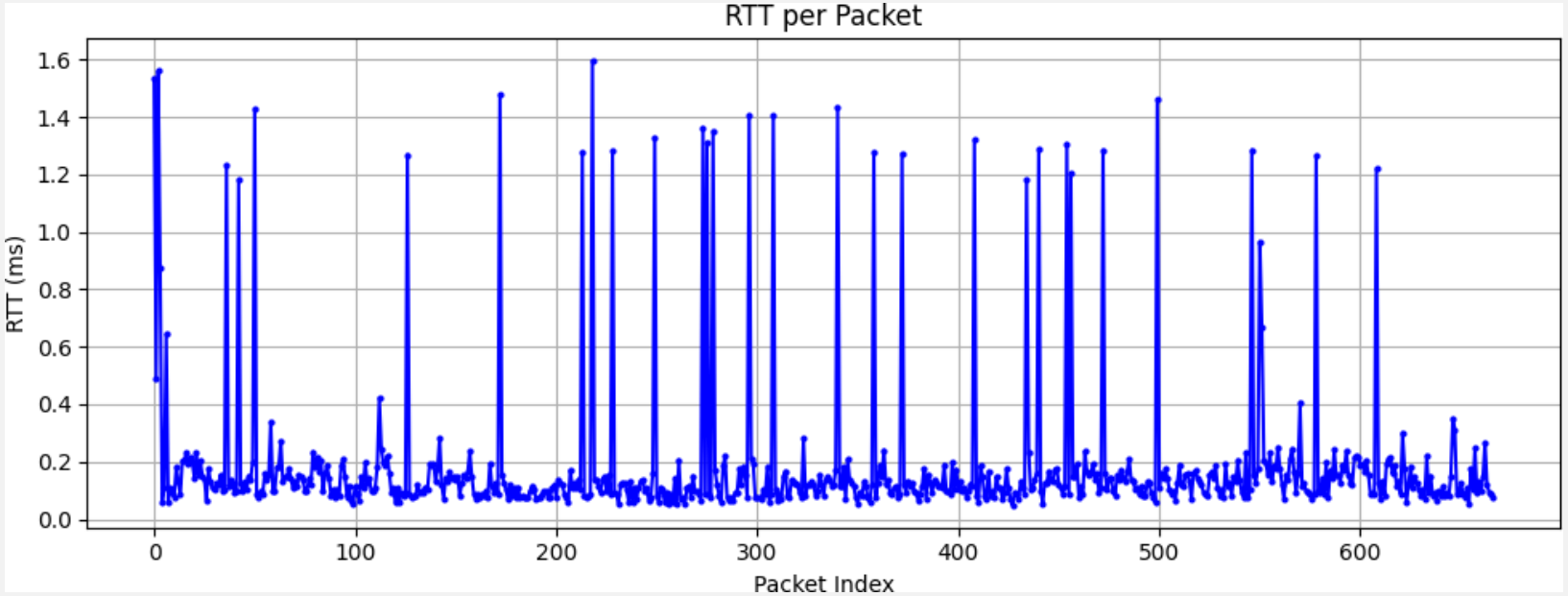
- **Objective:** Embed hidden message in IPv4 timestamp options
- **Method:**
- Use IP Option code 68 (Timestamp)
- ASCII chars encoded into timestamp value
- Injected into selected packets from `sec to insec`
- Parametrized with:
  - `ENABLE_COVERT`
  - `COVERT_MESSAGE`
  - `MEAN_DELAY`

PHASE 2 – METRICS AND RESULTS

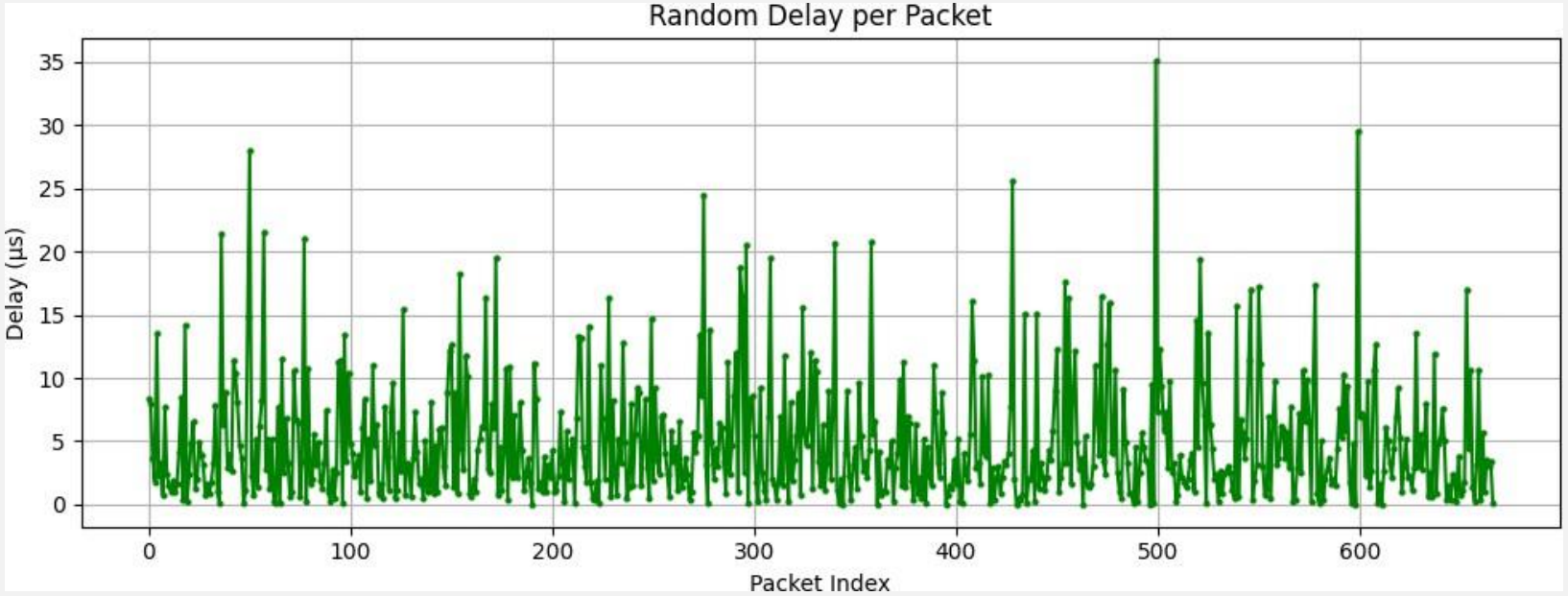
- **Performance Metrics:**
  - **RTT Avg:** 0.182 ms (CI 0.020 ms)
  - **Delay Avg:** 4.888 µs (CI 0.369 µs)
  - **Channel Capacity:** 19.15 bps
- **Covert Log:** JSON format with:
  - Injected char
  - ASCII value
  - Timestamp option field
  - Injection timestamp


```
[
  {
    "char": "H",
    "ascii": 72,
    "ts_value": 1207959552,
    "index": 1,
    "timestamp": 1749588246.2201784
  },
  {
    "char": "E",
    "ascii": 69,
    "ts_value": 1157627904,
    "index": 2,
    "timestamp": 1749588247.255361
  },
  {
    "char": "L",
    "ascii": 76,
    "ts_value": 1275068416,
    "index": 3,
    "timestamp": 1749588248.3049314
  },
  {
    "char": "L",
    "ascii": 76,
    "ts_value": 1275068416,
    "index": 4,
    "timestamp": 1749588249.3450284
  },
  {
    "char": "O",
    "ascii": 79,
    "ts_value": 1325400064,
    "index": 5,
    "timestamp": 1749588250.3747554
  }
]
```

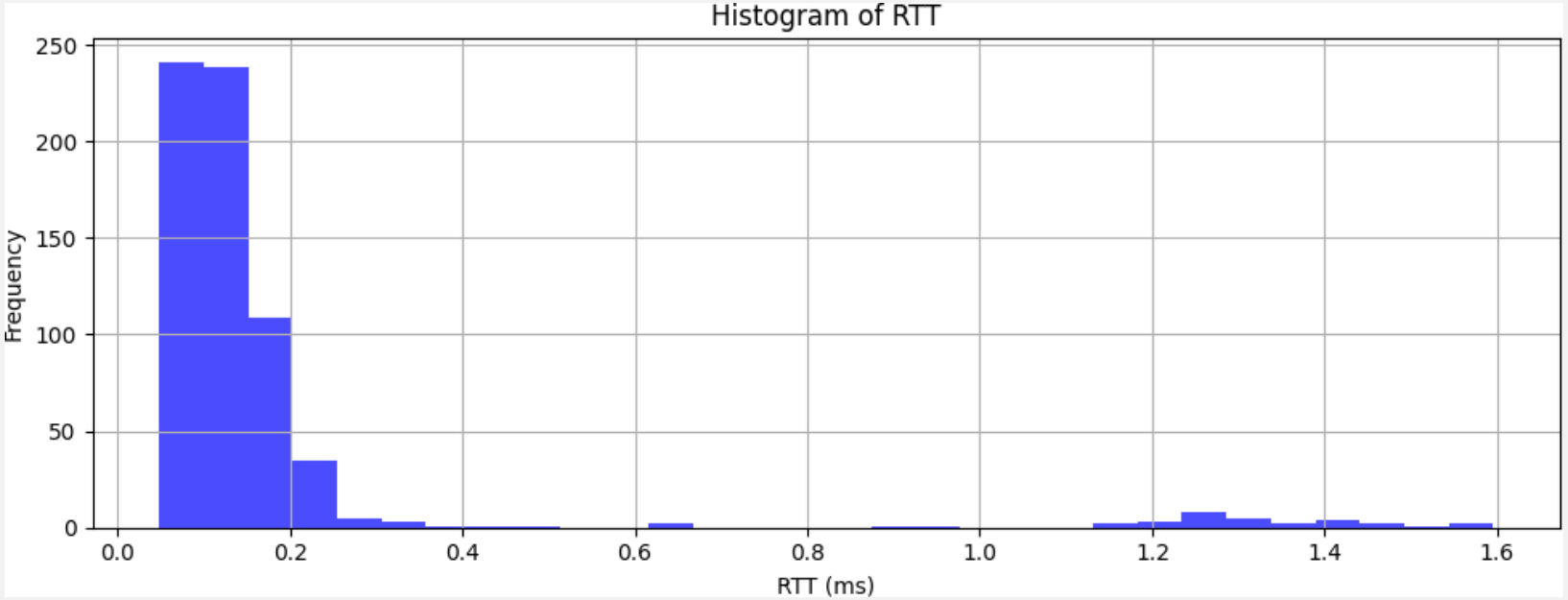
RTT VS. PACKET INDEX



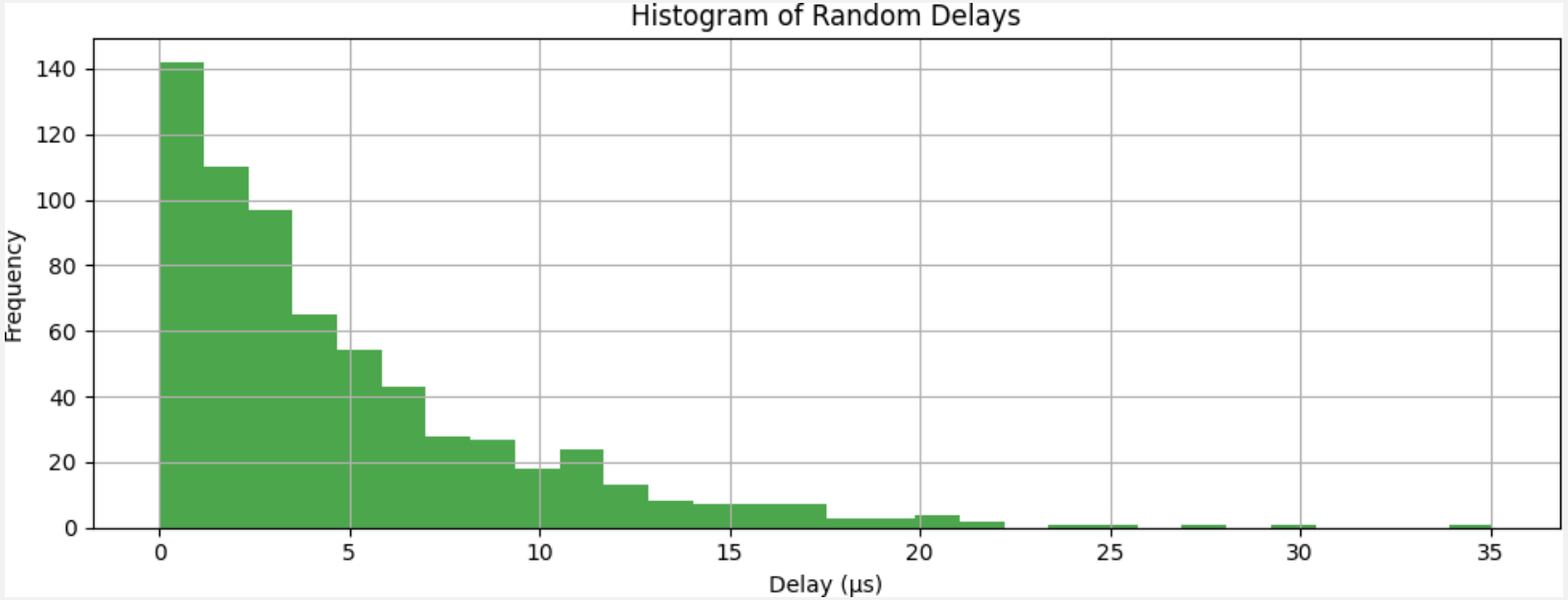
RANDOM DELAY VS. PACKET INDEX



HISTOGRAM OF RTT VALUES



HISTOGRAM OF RANDOM DELAYS



PHASE 2 – PLOTS AND ANALYSIS

- RTT over time
- Delay over time
- Histograms of RTT and delay
- Scatter of Mean Delay vs RTT

Conclusion:

- IPv4 timestamp enables low-capacity covert channel
- Delay tuning can optimize throughput vs stealth

PHASE 3 – DETECTOR DESIGN

Objective: Detect covert channel traffic

Techniques:

- Check for IP Option 68 (timestamp)
- Analyze inter-arrival timing patterns
- Use sliding window for IAT calculation

Ground Truth via `ENABLE_COVERT`

- Classify packets as TP, FP, TN, FN
- Compute Accuracy, Precision, Recall, F1-Score

PHASE 3 – EXPERIMENTAL CAMPAIGNS

Experiment 1: Normal traffic (covert disabled)

- 100 pings, 95 received
- All packets correctly classified as benign
- **Accuracy:** 100%

Experiment 2: Covert traffic (HELLO message)

- 100 pings, 72 received
- **TP:** 28, **FN:** 154
- **Recall:** 15.38%, **Precision:** 100%, **F1:** 26.67%

PHASE 3 – ANALYSIS

- High precision, poor recall
- Detector misses many covert packets
- Timestamp option not in all packets
- Possible improvements:
  - Frequency domain analysis
  - Entropy or ML-based detection

PHASE 4 – MITIGATION STRATEGY

- **Goals:**
- Decrease covert channel capacity
- Obfuscate detectability

PHASE 4 – MITIGATION STRATEGY

- **Actions:**
- Strip IP Options
- Add random delay (1–5ms)
- Drop suspicious packets (optional)
- Configurable via env vars:
  - `MITIGATE_STRIP_OPTIONS`
  - `MITIGATE_ADD_DELAY`
  - `MITIGATE_DROP_SUSPICIOUS`

PHASE 4 – MITIGATION RESULTS

Experiment: Covert traffic with mitigation

- 100 pings, 95 received
- **RTT Avg:** 10.44 ms
- **Capacity:** 9.63 bps (down from 19.15)

Detection: 0 TPs **Mitigation Stats:**

- Processed: 207, Delayed: 195
- Delay Added: 0.579s
- Avg delay: 2.8 ms

PHASE 4 – FINAL ANALYSIS

- Mitigation broke pattern detection
- RTT increased, jitter grew
- Some covert traffic passed undetected
- Tradeoff: Detection evasion vs. network performance

FINAL THOUGHTS

Key Takeaways:

- IPv4 timestamp is a viable covert channel
- Detection via heuristic patterns is limited
- Mitigation is effective but costly

REFERENCES & APPENDICES

- Code and data in GitHub repo
- Full logs in `results.json` and `covert_log.json`