Term Project Phase 2: Covert Channel

Selected Covert Channel: IP Identification Field Manipulation

1. Introduction

This phase focuses on implementing and evaluating a covert communication mechanism using the IP Identification (IP ID) field of IP packets. By embedding secret data in a low-level network field, the covert channel evades conventional detection mechanisms and simulates a realistic network steganography approach. The sender and receiver were deployed within the sec and insec containers, respectively, using the same Dockerized environment from Phase 1.

The covert message used for this experiment was "Secret: Operation Mincemeat", inspired by a real World War II British deception operation. This message was used in the World War II deception plan in which British intelligence used a dead body carrying misleading documents to deceive the Axis powers about the Allied invasion of Sicily.

2. System Components

We reused the architecture established in Phase 1 and included the following steps:

- MITM Container: Captures and relays packets between the secure and insecure networks using raw sockets and NATS.
- Python Processor: Adds programmable delays and forwards Ethernet frames via NATS.
- SEC Container: Implements the covert sender, encoding characters as IP ID fields and dispatching ICMP packets.
- INSEC Container: Hosts the covert receiver, which decodes received packets by extracting IP ID values.

3. Implementation Details

- Encoding: The sender uses Scapy to convert each character of the message into its ASCII value and sets it as the IP ID in an ICMP packet. To improve reliability, the packets are marked with a unique payload tag (CovertChannel). That was not mandatory, but to have a clear view of the results, we proceeded this way. The other packets received at the same time were out of ASCII range, and capturing too many packets was causing noise at the logs.
- Transmission: The message is sent at fixed intervals using a command-line configurable delay (0.5s to 2.0s). Also, within the test folder, the value can be configured by the user.

- Decoding: The receiver captures ICMP packets, filters only those containing the payload marker, and reconstructs the message by converting the IP ID values back to characters.

  Sample output:

```
Received packet with IP ID: 101 (character: e)
Received packet with modified IP ID (out of ASCII range): 59653
Received packet with IP ID: 97 (character: a)
Received packet with modified IP ID (out of ASCII range): 60070
Received packet with IP ID: 116 (character: t)
Received packet with modified IP ID (out of ASCII range): 60192
```

  In order to replicate the results, we added a test automation script (run_covert_tests.py):

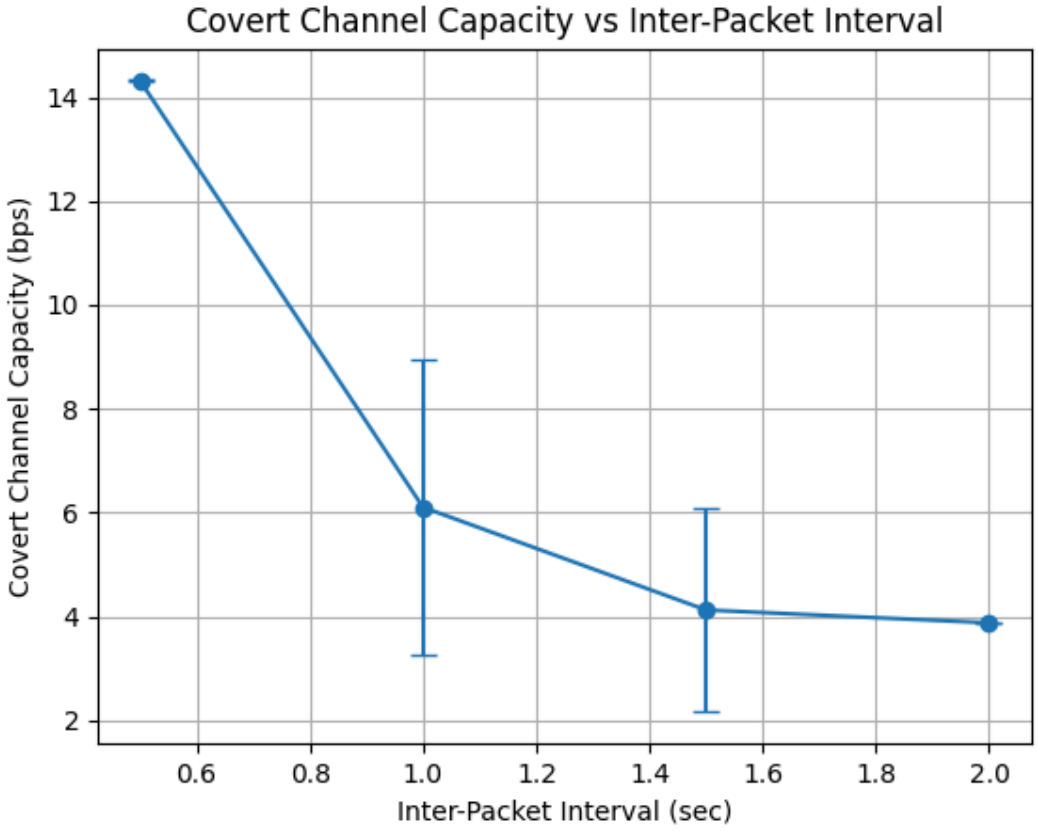
- Run multiple trials for each interval
- Log sender and receiver output
- Compute average channel capacity
- Calculate 95% confidence intervals
- Save results as CSV and visual plots

4. Results

The experiment was conducted over 5 trials per configuration using the message "Secret: Operation Mincemeat" (24 characters, 192 bits). Four different inter-packet intervals were tested: 0.5s, 1.0s, 1.5s, and 2.0s. The results can be seen in the Table.

Interval (sec)	Avg Capacity (bps)	Lower CI (bps)	Upper CI (bps)
0.5	14.32	14.30	14.33
1.0	6.11	3.26	8.95
1.5	4.13	2.16	6.09
2.0	3.88	3.88	3.88

The Figure visualizes the capacity decline with increasing delay. Faster intervals allow higher throughput, while longer delays offer stealth at the cost of performance.



5. Challenges

During the implementation, we encountered some challenges:

- Packet Filtering: Initial interference from unrelated ICMP traffic was mitigated by tagging packets with a known payload and refining the receiver's decoding logic. Otherwise directly going with the whole packet was creating some noises for the package to deliver.

- MITM Background Noise: Passive background traffic via MITM sometimes triggered false positives. Proper coordination between test steps minimized this impact it was

not completely over, but it worked in better results.

- Automation and Logging: Sender and receiver outputs were systematically logged per trial, which was hard to handle at the beginning of the misplaced packets.

- Channel Reliability: While the IP ID field is generally preserved, I am not sure that will be in this way if this trial is implemented in real-world deployments that work on real-world devices.

6. Conclusion

This phase of the project successfully demonstrated a functional covert channel using IP Identification field manipulation. We tested our implementation under different timing conditions and exhibited predictable trade-offs between throughput and stealth. We made some of the assumptions within the code, and the filtering worked out well. But the issue is still a doubt for me to include the covert channel key in the payload. We can just filter out the IP ID range, but to provide more clear visuality in the test, that was something considered to be implemented.

7. Repository

All implementation scripts, experimental logs, CSV data, and plots can be found in the public repository:

GitHub Repo: https://github.com/Alperemrehas/middlebox

Artifacts are organized under the TPPhase2_results/ directory.

Tests to run the implementation again are placed under the tests/ directory.

(Note: TTPhase1 was directly merged to main, the new development can be seen under the feature branch tpphase2-covert-channel-ip-field-manipulation. Before submitting the code merged to the main branch.)