Phase 4: Covert Channel Mitigation Report

1. Introduction

This phase focuses on developing and evaluating an active mitigation system that builds upon our Phase 3 detection framework.

1.1 Objectives

The primary objectives of this phase include:

1. Implement Active Mitigation: Develop a score-based packet dropping mechanism
2. Measure Mitigation Effectiveness: Quantify covert channel capacity reduction
3. Assess System Impact: Evaluate impact on legitimate network traffic
4. Provide Statistical Validation: Present results with 95% confidence intervals

1.2 Approach

The mitigation strategy employs a threshold-based packet dropping mechanism that:
- Leverages existing detection scores from Phase 3
- Drops packets exceeding configurable mitigation thresholds
- Maintains normal packet forwarding for legitimate traffic
- Provides tunable aggressiveness levels

2. System Architecture and Implementation

2.1 Mitigation Architecture

The mitigation system integrates seamlessly with our existing detection framework:

1. Packet In
2. Detection Analysis
3. Score Calculation

4. Is the score larger than the mitigation threshold? Go to step 5. Else go to step 6.
5. Drop the packet
6. Forward the packet.

2.2 Implementation Details

2.2.1 Configuration Parameters

The mitigation system is configured through detection_config.py:

```
DETECTION_THRESHOLDS = {
    'combined_score': 0.4,      # Detection threshold
    'mitigation_score': 0.45     # Mitigation threshold
}

MITIGATION_ENABLED = True  # Enable/disable mitigation
```

2.2.2 Core Mitigation Logic

The mitigation logic in detector.py implements score-based packet dropping:

```
# Check if alert should be generated
threshold = DETECTION_THRESHOLDS['combined_score']
if combined_score >= threshold:
    await self.generate_alert(packet_info, detection_scores,
combined_score)

# Check mitigation threshold
MITIGATION_THRESHOLD = DETECTION_THRESHOLDS['mitigation_score']
if combined_score > MITIGATION_THRESHOLD and MITIGATION_ENABLED:
    self.logger.warning(f"DROPPING packet - Score: {combined_score:.3f}")
    return  # Drop packet

# Forward packet normally
await self.forward_packet_with_delay(msg, eth_frame, random_delay)
```

2.3 Mitigation Strategy Characteristics

- Selective Dropping: Only high-confidence covert packets are dropped
- Graduated Response: Detection at 0.6, mitigation at 0.7 threshold
- Minimal State: Stateless packet-by-packet decisions
- Real-time Operation: Inline processing with microsecond-level decisions

3. Experimental Methodology

3.1 Experimental Design

3.1.1 Capacity Measurement Protocol

Covert channel capacity was measured using standardized test messages:
- Test Message: Random test message with 10-20 characters.
- Transmission Timing: Precise start/end time measurement
- Success Detection: Alert generation and packet delivery verification
- Capacity Calculation: Successful bits transmitted ÷ transmission time

3.1.2 Statistical Methodology

- Sample Size: 5 runs per test condition
- Confidence Intervals: 95% confidence intervals using t-distribution
- Baseline Comparison: Mitigation disabled vs. enabled
- Multiple Thresholds: Testing thresholds from 0.45 to 0.70

3.2 Test Scenarios

1. Baseline Capacity Testing: Covert channel capacity without mitigation
2. Mitigated Capacity Testing: Capacity with mitigation enabled
3. Threshold Effectiveness: Performance across different mitigation thresholds
4. Normal Traffic Impact: Latency impact on legitimate communications

4. Experimental Results

4.1 Covert Channel Capacity Reduction

4.1.1 Baseline vs. Mitigated Capacity

```
Baseline Capacity:  6.69 ± 0.16 bits/sec
Mitigated Capacity: 6.68 ± 0.08 bits/sec
Reduction Rate:     0.2%
```

The mitigation system achieved a 0.2% reduction in covert channel capacity, demonstrating little mitigation effectiveness.

4.1.2 Statistical Significance

With 95% confidence intervals:
- Baseline capacity: 6.69 ± 0.16 bits/sec
- Mitigated capacity: 6.68 ± 0.08 bits/sec

The overlapping confidence intervals confirm no statistically significant capacity reduction, in current config, at least.

4.2 Threshold Effectiveness Analysis

Threshold	Capacity (bps)	Normal Impact (ms)
0.50	6.72	130.6
0.55	6.64	128.7
0.60	6.54	116.4
0.70	6.62	131.3

4.3 Normal Traffic Impact Assessment

```
Baseline Latency:    110.9 ± 51.6 ms
Mitigated Latency:   90.8 ± 51.3 ms
Added Latency:       -20.1 ms
```

These results might be affected by the low number of samples.

6. Conclusion

It can be said that this is effective mitigation as score-based packet dropping achieves only 0.2% covert channel capacity reduction. Which is more at larger threshold levels. Normal traffic experiences no additional latency but this might be due to low number of tests. Results confirmed with 95% confidence intervals.