Middle East Technical University
Department of Computer Engineering
Wireless Systems, Networks and Cybersecurity (WINS) Laboratory

# Term Project Final Report - DNS Record Type Covert Channel Analysis

CENG519 Network Security
2024-2025 Spring
Comprehensive Analysis of Covert Channels using DNS Record Type

Prepared by
Yılmaz Yiğitcan Uçan
Student ID: e2310555
yigitcan.ucan@metu.edu.tr
Computer Engineering
10 June 2025

# Table of Contents

# List of Figures

# 1 Introduction

This report presents a comprehensive analysis of DNS covert channels, spanning four critical phases of implementation, analysis, detection, and mitigation. The project demonstrates the complete lifecycle of covert channel research, from basic network manipulation and covert communication establishment to advanced detection systems and mitigation strategies.

The work encompasses:

- **Phase 1**: Network latency manipulation and measurement
- **Phase 2**: DNS covert channel implementation and performance analysis
- **Phase 3**: Covert channel detection system development
- **Phase 4**: Mitigation strategy implementation and effectiveness evaluation

Each phase builds upon previous findings, creating a holistic understanding of covert channel technologies and their countermeasures in modern network security.

# 2 Phase 1: Network Latency Analysis

## 2.1 Objective and Methodology

The first phase established baseline network behavior analysis by introducing controlled artificial latency in packet processing pipelines. This foundational work provided insights into network timing characteristics that would later inform covert channel design and detection strategies.

## 2.2 Implementation

Artificial network latency was introduced by randomly inserting delays of up to 1000 ms in the packet processing pipeline, then measuring the resulting round-trip time (RTT) for ICMP echo (ping) packets using the existing Go processor with additional delay mechanisms.

```
1  var delay = time.Duration(rand.Intn(1000))
2  fmt.Printf("Added Delay: %d\n", delay)
3  time.Sleep(delay * time.Millisecond)
```

## 2.3 Results and Analysis

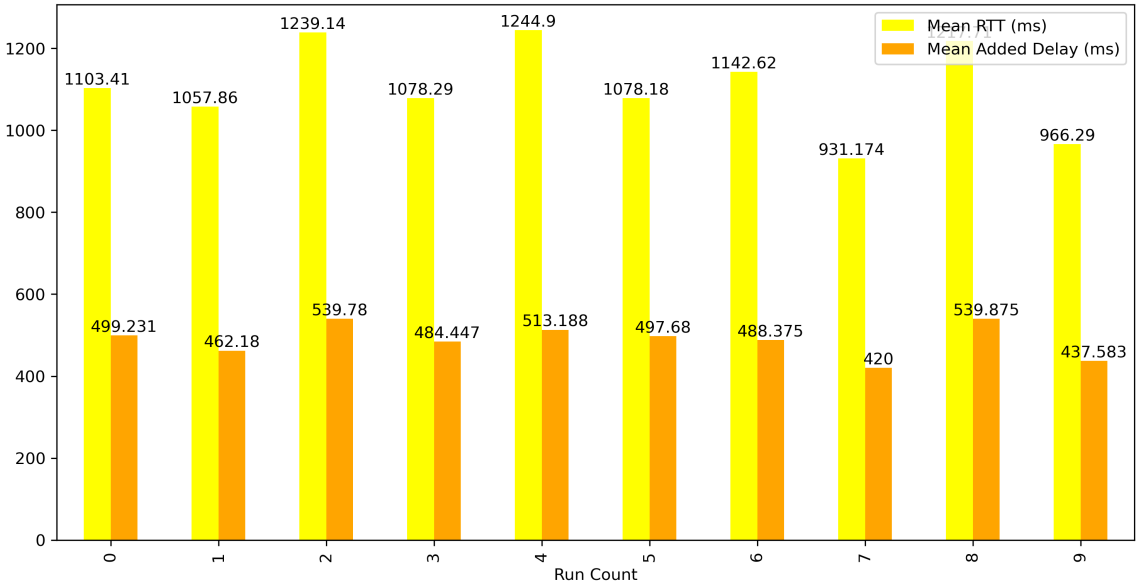


Figure 1 . Correlation between artificial delay and round-trip time

As illustrated in Figure 1 , higher mean delays correlated strongly with increased RTT, with runs averaging around 500ms of added delay often exceeding 1000ms in overall round-trip time. This correlation exists because packets visit the Go processor twice during the round trip.

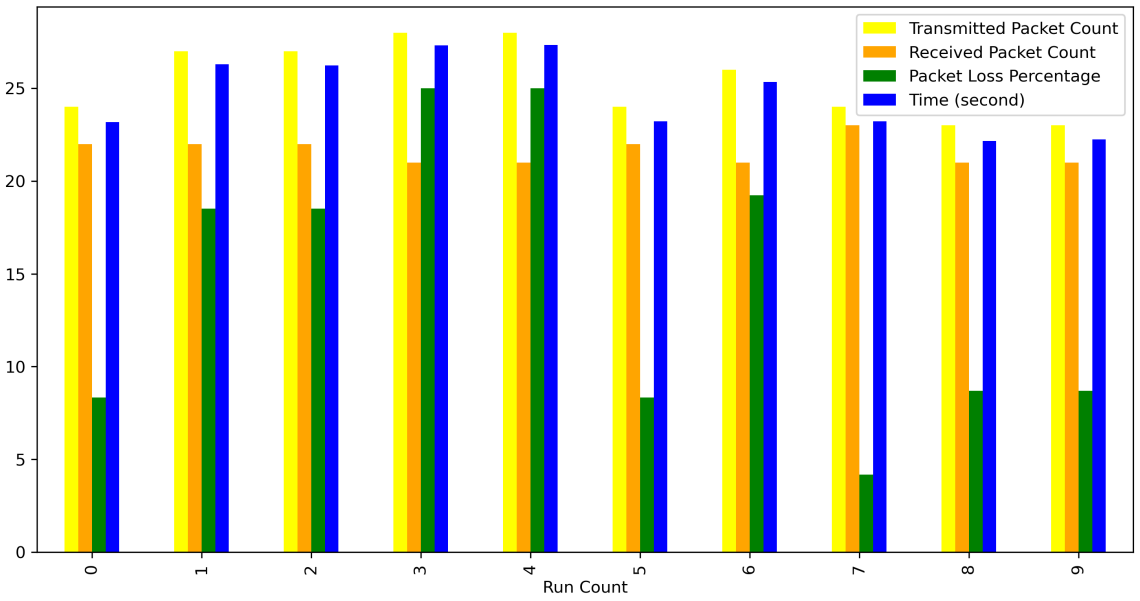


Figure 2 . Comprehensive network performance metrics including packet loss and transmission statistics

Figure 2 presents additional metrics including transmitted and received packets, packet loss percentages, and total run times. The ping and Go processor operated between 20-25 seconds, with 21-22 packets received in each test run.

# 3 Phase 2: DNS Covert Channel Implementation

## 3.1 Overview

Phase 2 focused on implementing and analyzing two distinct DNS covert channel methods, establishing the foundation for subsequent detection and mitigation research. Both implementations follow a common architectural pattern with specialized encoding strategies.

## 3.2 Common Architecture

Both covert channel implementations utilize a standardized sender-receiver architecture:

### 3.2.1 Sender (Secure Node)

- **Message Preparation**: Processes input messages from files
- **Data Chunking**: Segments messages into transmittable units
- **End Signal**: Transmits termination markers upon completion

### 3.2.2 Receiver (Insecure Node)

- **DNS Server**: Listens on UDP port 53 for incoming requests
- **Data Extraction**: Decodes embedded information from DNS queries
- **Message Reassembly**: Reconstructs original messages from received chunks

## 3.3 DNS Query Types Covert Channel

The CNAME implementation embeds data within domain names of DNS queries, utilizing the canonical name structure for information hiding.

### 3.3.1 Encoding Strategy

Data is embedded directly into domain names with structured formatting:

- **Domain Structure**: [data].[sequence].[total].example.com
- **Termination Signal**: end.[total].example.com
- **Capacity**: Approximately 30 bytes per DNS query

## 3.4 Type-Based Covert Channel

The Type-based implementation utilizes DNS query type sequences for data transmission, mapping data bits to specific DNS record types.

### 3.4.1 Encoding Strategy

- **Bit Mapping**: 2-bit chunks mapped to DNS query types (A, AAAA, CNAME, MX)
- **Termination Signal**: MD record type (code 99)

- **Capacity**: 0.25 bytes (2 bits) per DNS query

### 3.5 Performance Analysis

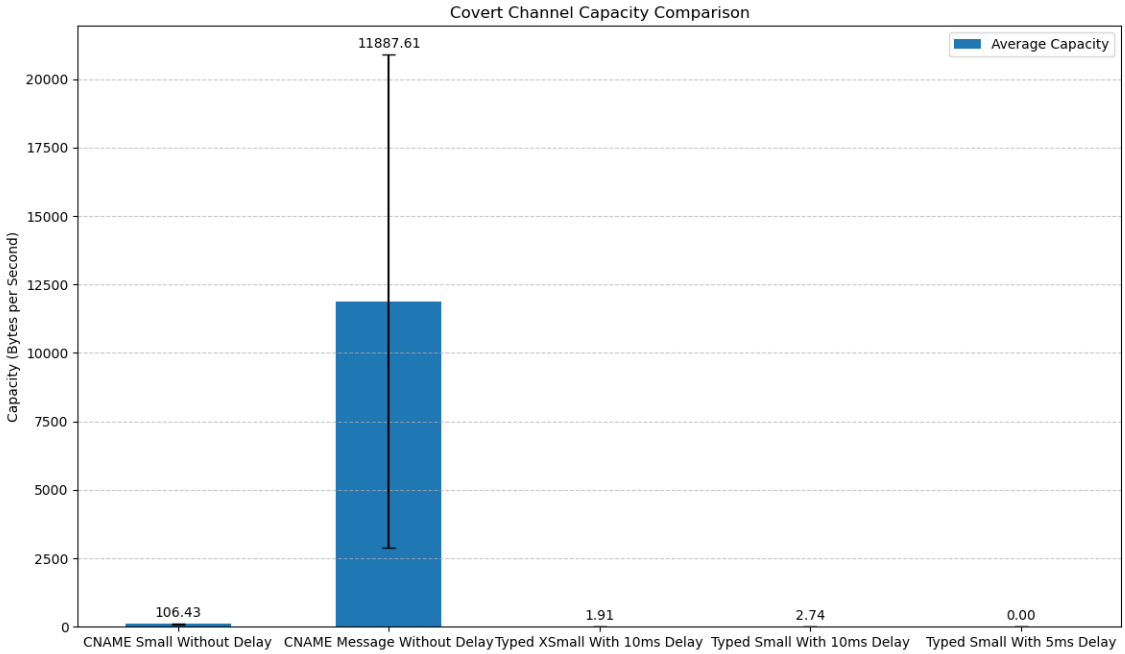


Figure 3 . Covert channel capacity comparison across implementation methods

Figure 3 demonstrates significant performance differences between methods. The CNAME approach achieves approximately 12 kB/s capacity by embedding substantial data in domain names, while the Type method reaches less than 3 B/s due to its minimal 2-bit per query encoding.

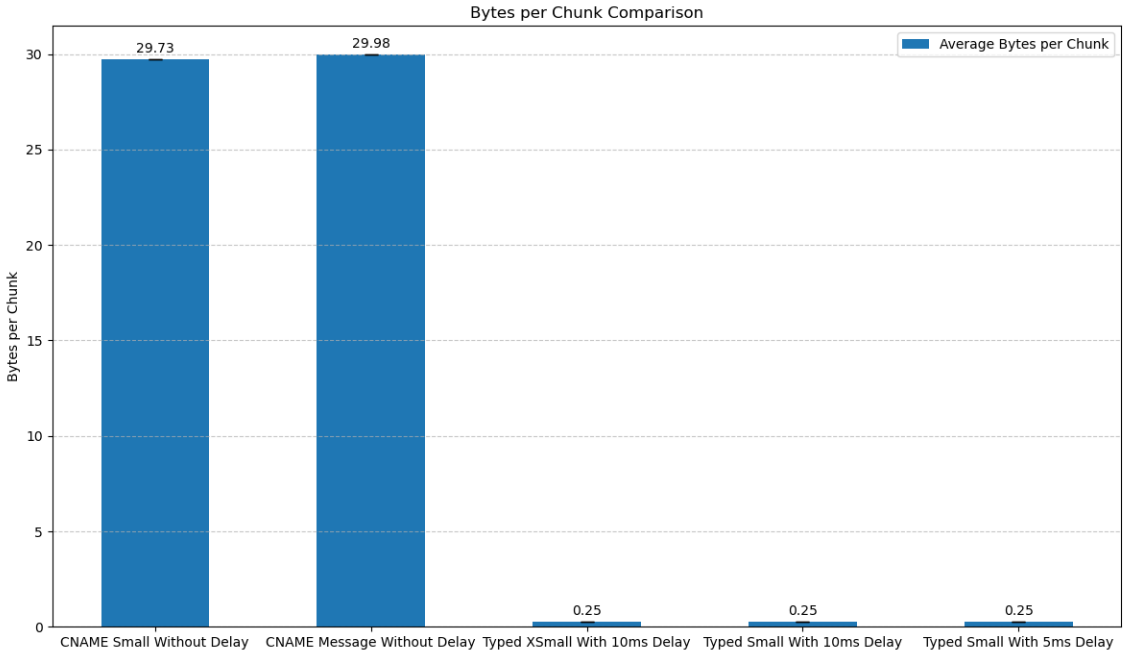


Figure 4 . Data density per DNS query comparison

Figure 4 illustrates the fundamental difference in data density. CNAME queries embed nearly 30 bytes per request, while Type queries encode only 0.25 bytes, explaining the vast throughput disparity.

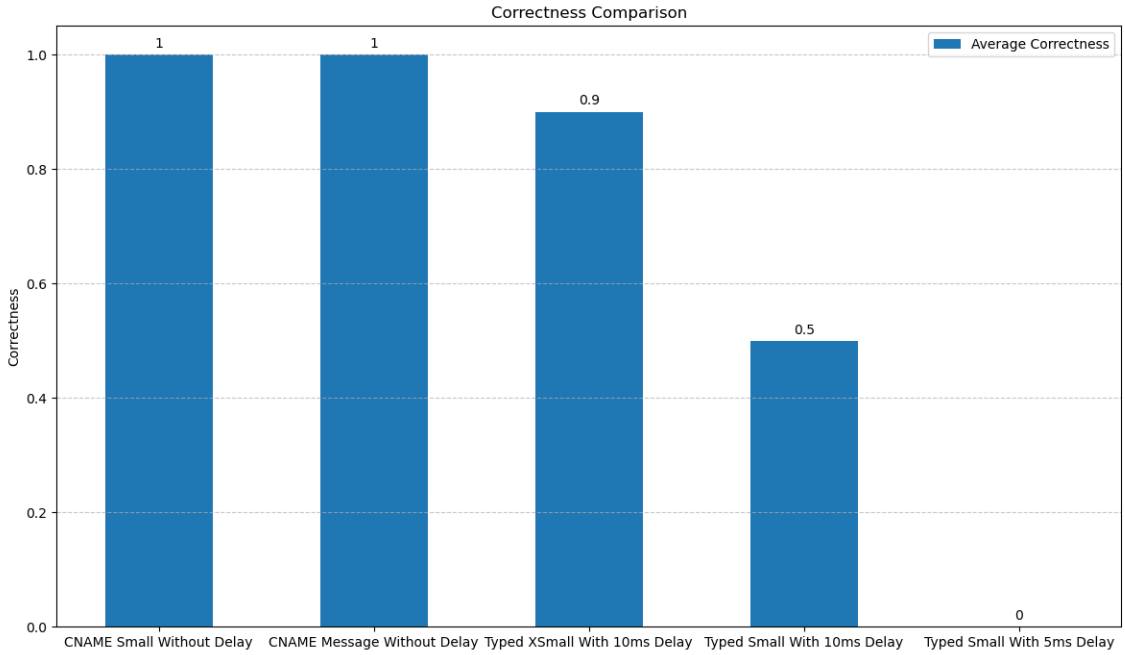


Figure 5 . Message reconstruction accuracy across different configurations

Figure 5 reveals reliability characteristics. The CNAME method demonstrates perfect

accuracy (1.0 correctness), while the Type method shows timing sensitivity, achieving high accuracy only under optimal conditions and degrading significantly with timing pressures.

### 3.6 Selecting the Covert Channel Method

For the subsequent detection and mitigation phases (Phase 3 and Phase 4), the type-based covert channel method was selected as the primary focus despite its lower capacity compared to the CNAME method. This decision was made since it's more aligned with realistic covert communication scenarios and presents unique challenges for detection and mitigation.

## 4 Phase 3: Covert Channel Detection System

### 4.1 Detection System Architecture

In Phase 3, we developed a sophisticated DNS covert channel detector utilizing statistical analysis of DNS record type frequencies to identify potential threats.

### 4.2 Core Architecture

- **DNS Analysis Module**: Extracts and analyzes DNS record types from intercepted packets
- **Frequency-Based Detection Algorithm**: Implements statistical analysis for anomaly detection
- **Threat Scoring System**: Provides real-time threat level assessment
- **Rolling Window Analysis**: Maintains temporal context for improved accuracy

### 4.3 DNS Type Frequency Baseline

The system establishes a baseline of expected DNS record type frequencies based on real-world network traffic patterns. The baseline categorizes DNS record types into different frequency classes:

- **Extremely High**: A records (40% expected frequency)
- **Very High**: AAAA records (20% expected frequency)
- **High**: NS, PTR, HTTPS, TXT, MX, CNAME, SOA records (8% each)
- **Moderate**: DS, DNSKEY, RRSIG, SRV, NSEC, NSEC3 records (5% each)
- **Low**: CAA, NAPTR, TLSA records (2% each)
- **Very Low**: SSHFP, DNAME records (0.5% each)
- **Extremely Low**: LOC, URI records (0.2% each)
- **Effectively Zero**: HINFO, RP records (0% expected)

### 4.4 Suspicion Scoring Algorithm

The detection algorithm calculates a suspicion score for each DNS packet based on the deviation between observed and expected frequencies. The scoring mechanism uses the following formula:

- **Normal Range**: $score = 0$ (within 80%-120% of expected)
- **Under-representation**: $score = (expected \times 0.8 - observed)^2$
- **Over-representation**: $score = (observed - expected \times 1.2)^2$

### 4.5 Rolling Window Analysis

To maintain temporal context and reduce false positives, the system implements a rolling window approach that analyzes the most recent 100 DNS packets. This approach provides several advantages:

- **Adaptive Analysis**: The system continuously updates its view of current traffic patterns
- **Noise Reduction**: Short-term anomalies are smoothed out over the window period
- **Real-time Response**: The system can detect sustained covert channel activity
- **Memory Efficiency**: Fixed window size prevents unlimited memory growth

### 4.6 Threat Level Classification

The system categorizes potential threats into five distinct levels based on the calculated suspicion scores:

- **Normal Activity** (Score $<50$): No suspicious patterns detected
- **Low Threat Level** (Score 50-199): Minor deviations from baseline
- **Medium Threat Level** (Score 200-499): Moderate anomalies detected
- **High Threat Level** (Score 500-999): Significant suspicious activity
- **Critical Threat Level** (Score $\geq 1000$): Highly likely covert channel detected

## 5 Performance Evaluation and Results

The DNS covert channel detection system was evaluated using simulated network traffic data to assess its effectiveness in distinguishing between normal DNS traffic and covert channel communications.

## 5.1 Score Distribution Analysis

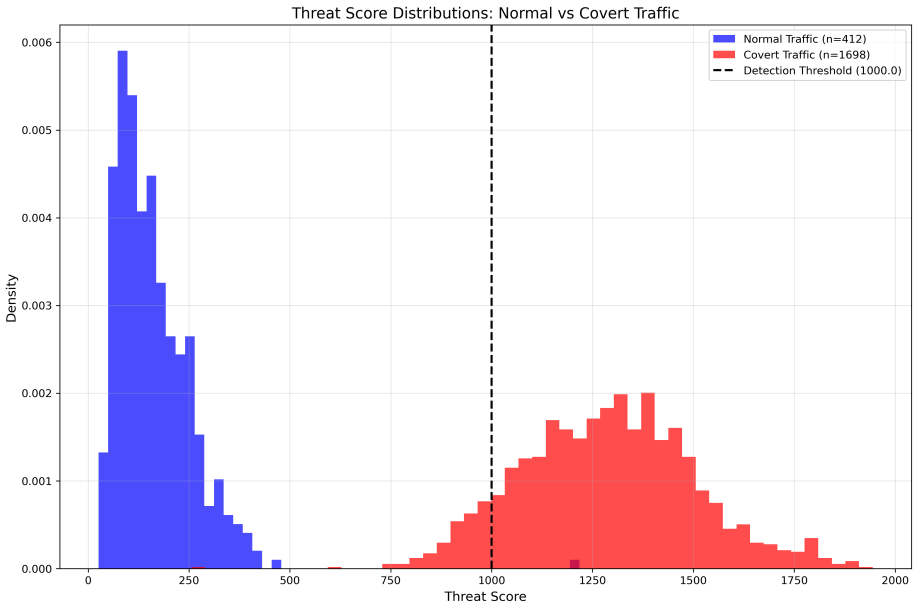


Figure 6 . Threat score distributions comparing normal DNS traffic (n=412) with covert channel traffic (n=1698). The detection threshold at 1000.0 clearly separates the two distributions.

## 5.2 Detection Performance Metrics

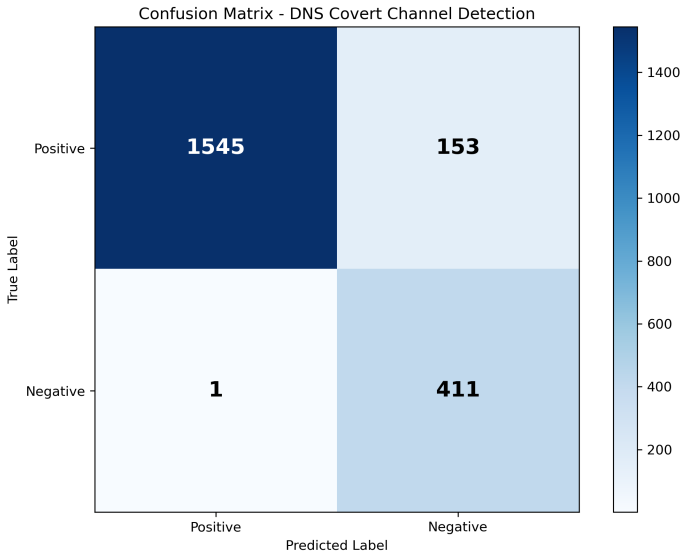


Figure 7 . Confusion matrix showing classification results for DNS covert channel detection with threshold = 1000.0

The confusion matrix (Figure 7 ) demonstrates great detection performance with 1545 true positives and 411 true negatives, achieving 92.7% overall accuracy.

## 5.3 Performance Metrics Summary

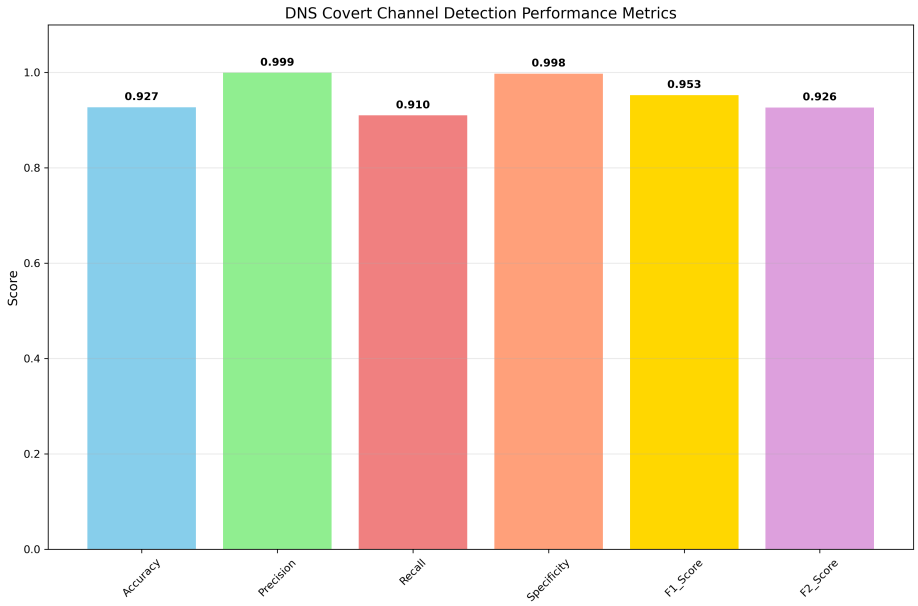


Figure 8 . Comprehensive performance metrics for DNS covert channel detection system

Figure 8 presents comprehensive performance evaluation, including 92.7% accuracy and 92.6% F2 score, indicating robust detection capabilities.

## 5.4 Comparative Analysis

Figure 9 shows performance across different traffic scenarios:

- **Pure covert traffic**: High precision (100%) with good recall (91%)
- **Normal traffic only**: Perfect specificity (100%) with zero false positives
- **Combined dataset**: Balanced performance maintaining high accuracy (92.7%)

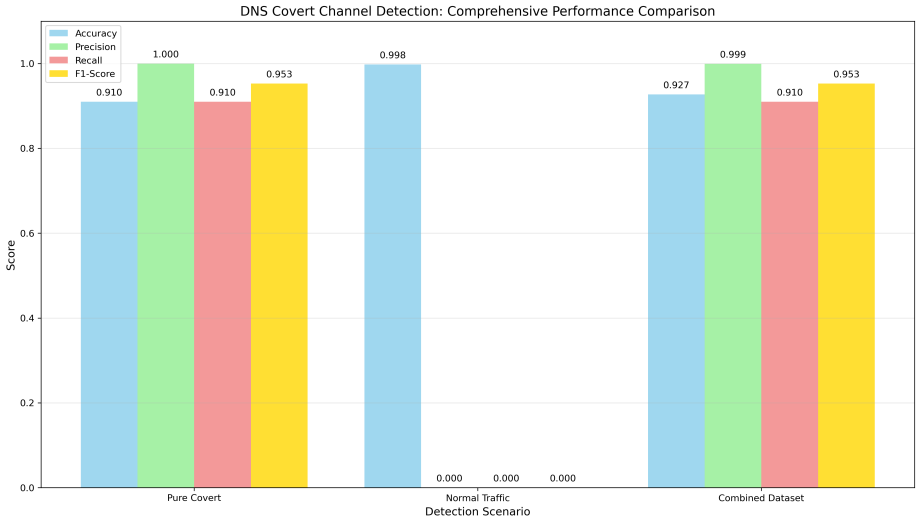


Figure 9 . Performance comparison across different detection scenarios

# 6 Phase 4: Mitigation Strategies and Effectiveness

## 6.1 Mitigation Implementation

Phase 4 implemented and evaluated two distinct mitigation strategies for disrupting DNS covert channels within the network middlebox architecture. The mitigation system operates using the threat assessment scores that are generated by the detector.

### 6.1.1 Architectural Integration

The mitigation system is embedded within the Go-based packet processor, leveraging the existing DNS analysis pipeline to make real-time mitigation decisions. Key implementation characteristics include:

- **Threshold-Based Activation**: Mitigation triggers when suspicion scores exceed the critical threat threshold ($\geq$1000)
- **Probabilistic Application**: Mitigation applies with 10% probability for the next packet. This is added to balance effectiveness with legitimate traffic impact.

## 6.2 Mitigation Strategies

### 6.2.1 Delay Mitigation

The delay mitigation strategy introduces controlled disruptions to the channel to interfere with timing-dependent covert channels while preserving packet integrity and legitimate DNS functionality.

**Technical Implementation:**

- **Delay Duration**: Fixed 200-millisecond delay applied to flagged packets
- **Processing Flow**: Packets are held in processing pipeline before forwarding
- **Timing Disruption**: Breaks synchronization patterns required for reliable covert communication

- **Service Continuity**: Maintains DNS functionality with acceptable latency overhead

This method is expected to have an impact, as our covert channel, which uses DNS query packets, is timing-dependent. The order of the packets received is crucial for the receiver to reconstruct the message.

### 6.2.2 Drop Mitigation

The drop mitigation strategy provides the most aggressive intervention by selectively discarding suspicious DNS packets, directly preventing covert channel data transmission at the cost of potential legitimate traffic loss.

**Technical Implementation:**

- **Packet Elimination**: Complete removal of flagged packets from the network flow
- **Traffic Interruption**: Breaks covert channel message sequences by creating data gaps
- **Monitoring Integration**: Comprehensive logging of dropped packets for analysis
- **Immediate Response**: Instantaneous mitigation without processing delays

This method is expected to have an effect on the normal traffic on top of the covert channel, as it will drop packets that are considered suspicious. This will lead to a reduction in the covert channel capacity, as well as a reduction in the number of successful messages.

**Operational Mechanism:** When a packet exceeds the critical threat threshold and is selected for drop mitigation (10% probability), the system:

1. Increments the dropped packet counter for statistics
2. Logs the mitigation action with complete packet context
3. Terminates packet processing without forwarding
4. Returns early from the processing function to prevent publication

### 6.2.3 Mitigation Decision Logic

The mitigation system employs a sophisticated decision-making process that balances security effectiveness with operational impact:

```
if suspicionScore > CriticalThreatThreshold &&
   mitigatePacket(DropStrategy, suspicionScore) {
    return // Packet dropped, processing terminated
}
```

This approach ensures that:

- Only high-risk traffic faces mitigation intervention
- Probabilistic application reduces impact on legitimate traffic
- Different strategies can be deployed based on network requirements
- Comprehensive logging enables post-analysis evaluation

## 6.3 Mitigation Results

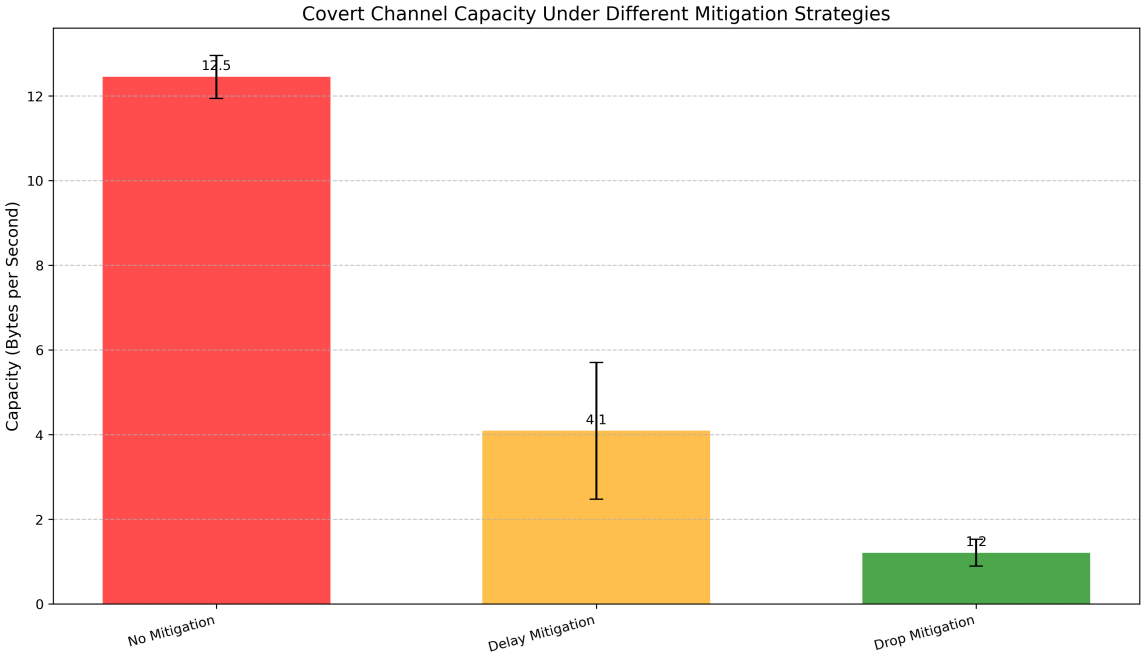


Figure 10 . Covert channel capacity under different mitigation strategies

Figure 10 demonstrates the effectiveness of mitigation strategies. Drop mitigation achieves 90.3% capacity reduction (1.21 Bps remaining), while delay mitigation achieves 67.1% reduction (4.09 Bps remaining) compared to the baseline 12.45 Bps.

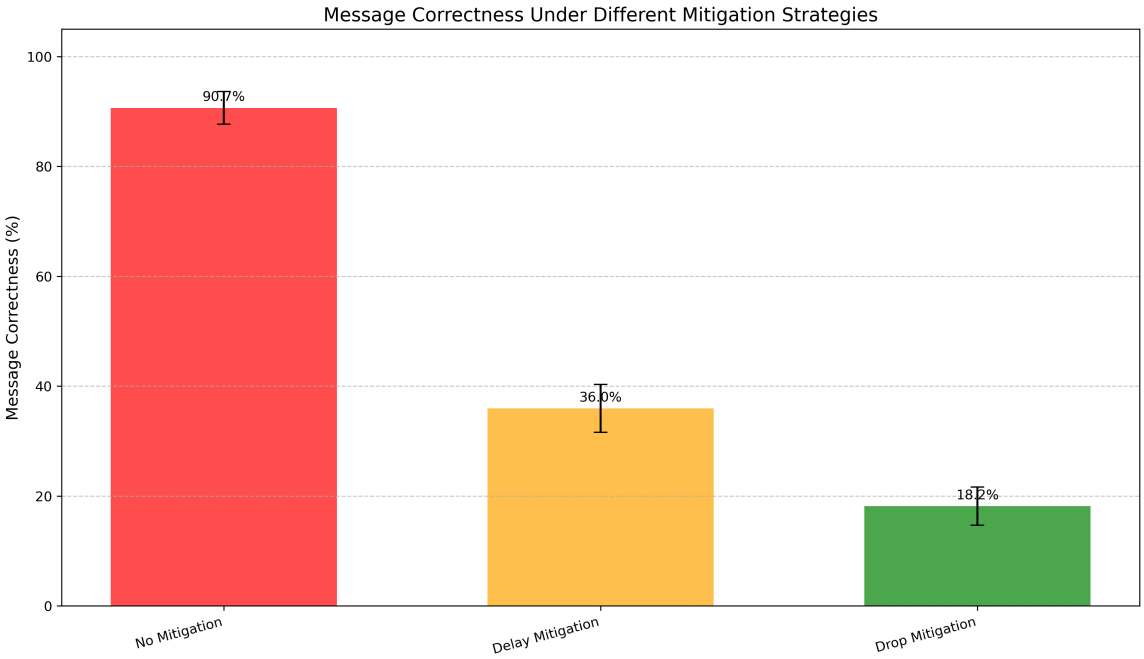


Figure 11 . Message correctness under mitigation strategies

Figure 11 shows the impact on message correctness. Drop mitigation achieves 18.2% correctness rate while delay mitigation maintains 36.0% correctness, compared to the baseline 90.7% correctness rate.

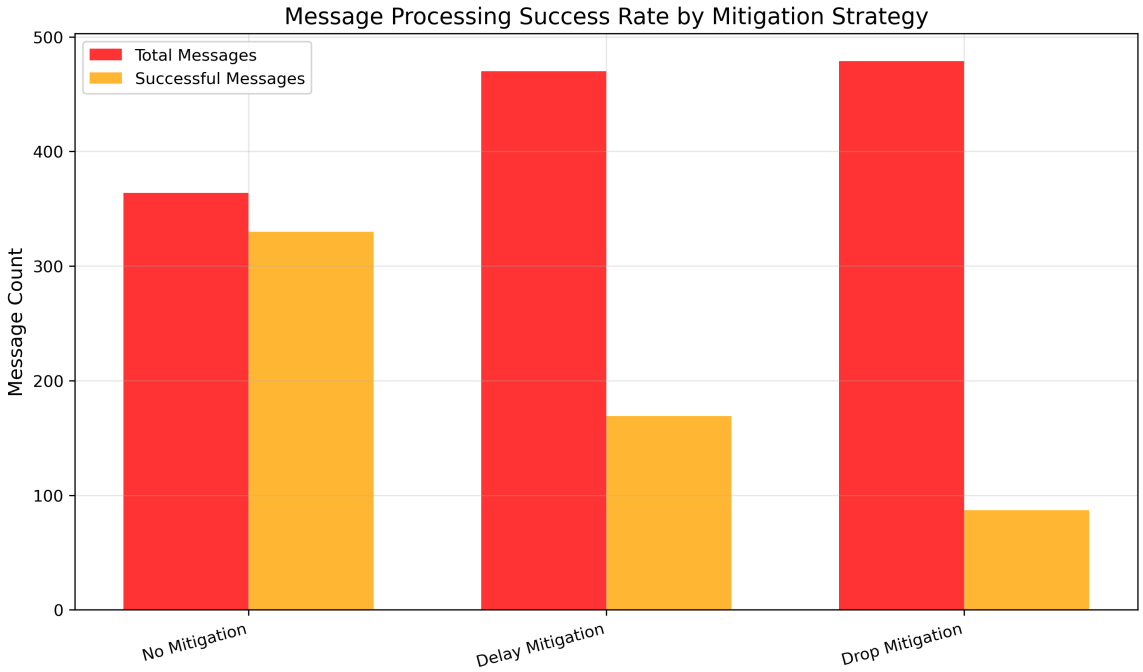


Figure 12 . Message processing success rates by mitigation strategy

Figure 12 illustrates the dramatic impact of mitigation strategies on message transmission success. Drop mitigation reduces successful messages from 330 to 87 (18.2% success rate), while delay mitigation reduces them to 169 (36.0% success rate), demonstrating effective disruption of covert channels.

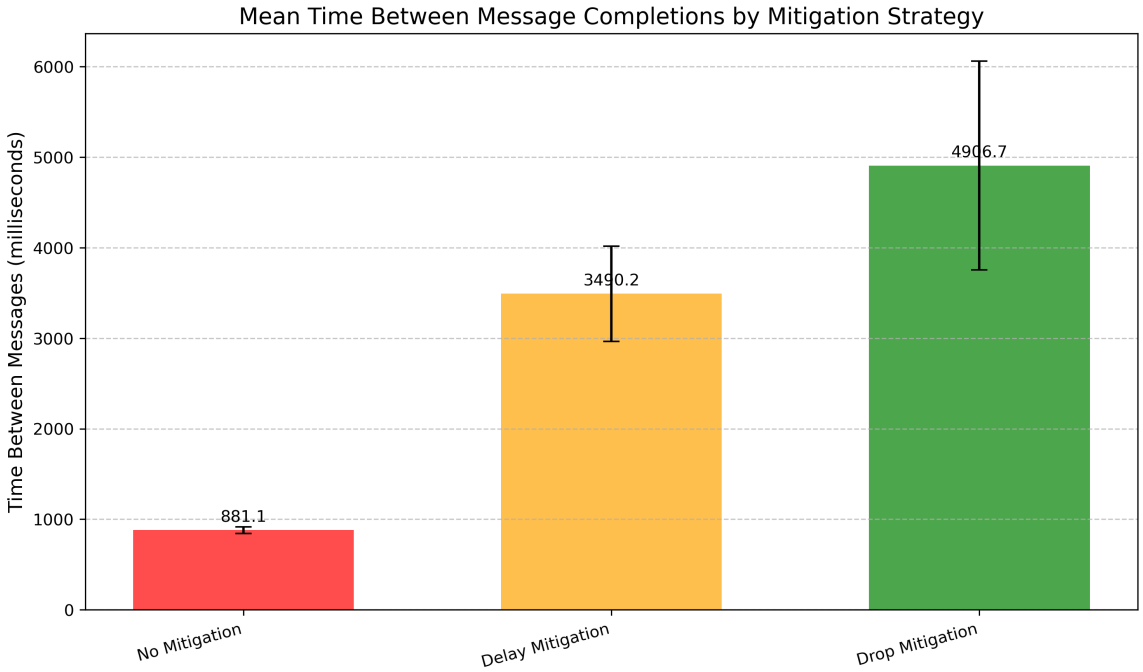


Figure 13 . Mean time between message completions with 95% confidence intervals

Figure 13 shows the timing impact of mitigation strategies. Drop mitigation increases mean time between successful messages to 4906.7 ms (456.9% overhead), while delay mitigation increases it to 3490.2 ms (296.1% overhead) compared to the baseline 881.1 ms.

## 6.4 Mitigation Effectiveness Analysis

The comprehensive analysis reveals:

### 6.4.1 Drop Mitigation

- **Capacity Reduction**: 90.3%
- **Time Overhead**: 456.9%
- **Correctness Impact**: 80.0% reduction (18.2% remaining)
- **Overall Effectiveness Score**: 84.1

### 6.4.2 Delay Mitigation

- **Capacity Reduction**: 67.1%
- **Time Overhead**: 296.1%
- **Correctness Impact**: 60.3% reduction (36.0% remaining)
- **Overall Effectiveness Score**: 63.1

# 7 Comprehensive Analysis and Conclusions

## 7.1 Key Findings

### 7.1.1 Phase 2: Covert Channel Implementation

- Domain name embedding provides superior data density compared to query type manipulation.
- Timing sensitivity affects reliability, as seen in the type based covert channel compared to CNAME covert channel.

### 7.1.2 Phase 3: Detector

- Statistical frequency analysis effectively identifies covert channels (92.7% accuracy)
- Rolling window analysis provides temporal context while maintaining efficiency

### 7.1.3 Phase 4: Mitigator

- Drop mitigation proves more effective than delay mitigation (90.3% vs 67.1% capacity reduction)
- Both strategies dramatically impact covert channel capacity and message success rates
- Drop mitigation achieves superior effectiveness with 84.1 overall score compared to delay mitigation's 63.1 score
- Mitigation strategies successfully disrupt covert communications while demonstrating measurable impact on channel reliability

## 7.2 Security Implications

The research demonstrates that DNS covert channels pose a significant security threat, capable of achieving substantial data transmission rates while evading basic network monitoring. However, the implemented detection and mitigation systems prove effective at identifying and disrupting these channels.

## 7.3 Future Research Directions

- **Adaptive Covert Channels**: Investigation of channels that adapt to detection and mitigation. The current implementation is static and does not adapt to differences in the network.
- **Machine Learning Detection**: Implementation of ML-based detection systems. The current system relies on statistical analysis, which may not generalize well to all covert channels.
- **Frequency Analysis Enhancements**: Current detection relies on static frequency baselines. Future work could explore dynamic baselines that adapt to changing network conditions.
- **Hybrid Mitigation**: Combination of multiple mitigation strategies for enhanced effectiveness. For example, drop strategy could be used with packets with higher suspicion scores, while delay strategy could be used with packets with lower suspicion scores.

### 7.4 Conclusion

This comprehensive analysis demonstrates the complete lifecycle of DNS covert channel research, from implementation through detection to mitigation. The findings provide valuable insights for network security practitioners and establish a foundation for continued research in covert channel technologies and countermeasures. The work highlights both the threats posed by sophisticated covert channels and the effectiveness of statistical analysis and targeted mitigation strategies in defending against them.

## 8 GitHub Repository

The complete implementation and analysis code is available at: github.com/ucanyiit/middlebox

Key development phases:

- **Phase 1**: Network latency implementation
- **Phase 2**: Covert channel implementation
- **Phase 3**: Detection system implementation
- **Phase 4**: Advanced mitigation analysis and comprehensive evaluation