2024

Multi-Layer Anomaly-Based Phishing Detection

THREAT MODELING REPORT

HARISH CHANDER KAUSHAL, NEERAJ BAIPUREDDY, NIMISH SRIVASTAV, RAJEEV MANDALAM

Table of Contents

1 Introduction

The purpose of this threat model document is to identify and mitigate potential security threats to SafeMail, a web-based network intrusion detection system (NIDS) POC designed to detect and alert organizations about phishing emails and potentially instant messages sent to internal inboxes. Unlike traditional detection systems, SafeMail uses heuristics from multiple layers in the network stack for an in-depth inspection of the sender's email data, TCP data, and IP data. It cross-references the organization's network security baseline configuration to determine whether an email has indicators of potential phishing activity, and subsequently marks the email as fraudulent or junk if necessary.

The goal of SafeMail is to accurately detect fraudulent emails using multi-layer inspection and to achieve a strong false-positive rate that competes with phishing detection systems focusing only on application-layer data. SafeMail can be implemented on top of an organization's existing email server, acting as a "proxy" system that evaluates emails based on the heuristics provided.

This document is intended for system architects, developers, security teams, and stakeholders. It is structured to provide an overview of the system, identify assets, threats, and vulnerabilities, assess risks, propose mitigations, and validate the effectiveness of these mitigations. It includes sections on system description, assets identification, threat and vulnerability identification, risk assessment, mitigation strategies, assumptions and dependencies, validation and verification, and a conclusion.

SafeMail is constructed with an algorithm-focused design to perform its core functionality. This design is detailed further in the system architecture section. The proof of concept emphasizes leveraging data points from each network layer to reduce false positives in phishing email detection, rather than focusing extensively on system architecture. Basic username-password-based authentication is used in the prototype.

SafeMail can be integrated with any internal email systems, such as Canvas, to enhance email security.

2 System Description

2.1 System Overview

SafeMail is a comprehensive web-based network intrusion detection system (NIDS) designed to safeguard organizations against phishing emails and potentially harmful instant messages directed towards internal inboxes. By utilizing a multi-layered approach, SafeMail conducts an in-depth inspection of data across various layers of the network stack. This approach allows SafeMail to perform a thorough analysis of sender's email data, TCP data, and IP data, effectively identifying and flagging potential phishing threats.

2.2 Architecture Diagram

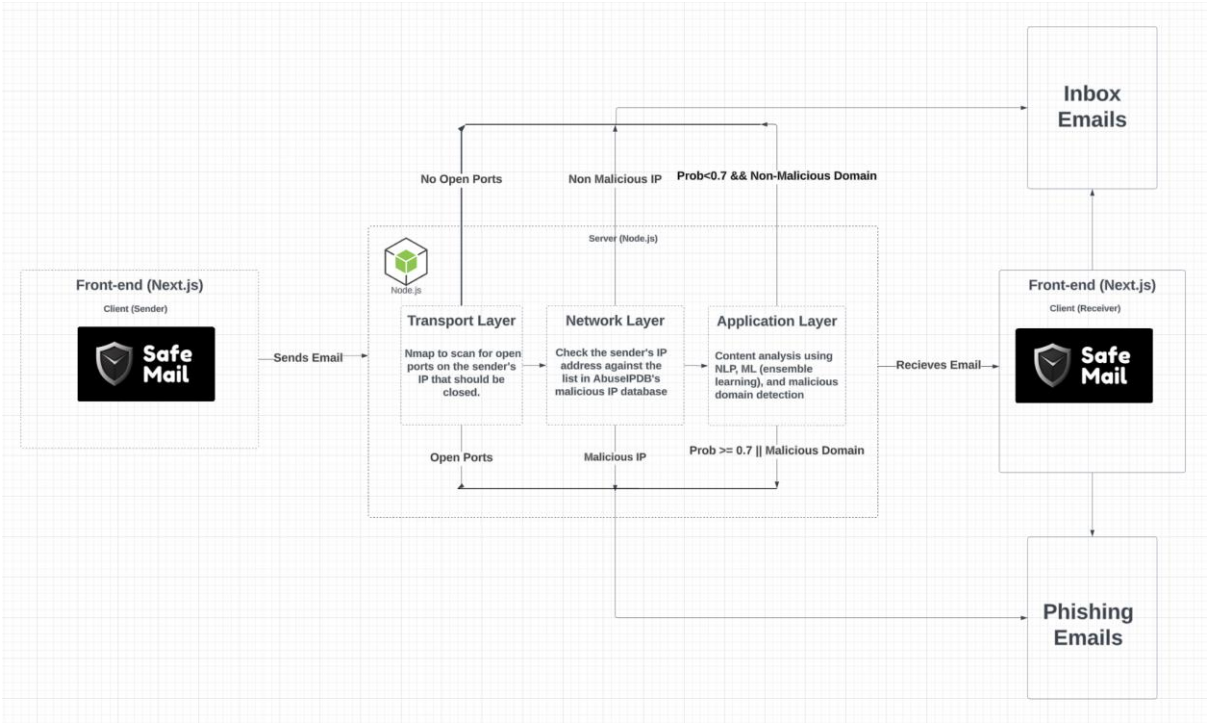


Figure 1: System Architecture

Figure 1 represents the architecture of SafeMail which is structured around multiple layers, each responsible for specific data points and analysis processes:

1. **Layer 3 - IP Address Verification:**
    - Client Side (Front-end, Next.js):
        - An API call is made to https://api.ipify.org/?format=json to retrieve the sender's IP address.
        - The retrieved IP address is included in the payload sent to the backend.

    - Server Side (Back-end, Node.js):
        - An API call is made to AbuseIPDB to check if the IP address is listed as malicious.
        - The IP address is cross-referenced with a baseline of high-risk IP addresses from MaxMind.

2. **Layer 4 - Port Scanning:**
    - Server Side (Back-end, Node.js):
        - The node-nmap library is used to perform an Nmap scan on the client IP.
        - The scan identifies open ports, which are then checked against a list of ports that should not be open and ports known for running malicious software.

3. **Layer 5 - Content and Domain Analysis:**
    - Content Analysis (Email Body):
        - A phishing email dataset is used to train three machine learning models: Random Forest, Naive Bayes, and Logistic Regression.
        - Natural Language Processing (NLP) techniques like tokenization are applied for cleaning the email content.
        - The three models are used in a hard voting classifier ensemble to determine if the email content is malicious.

    - Email Domain:
        - A regular expression is created from a static list of malicious domains.
        - Domains with excessive hyphens and digits are flagged as malicious.

3 Assumptions and Dependencies

Before we delve into the threat model, there are a few assumptions and dependencies that must be considered regarding the security of this Proof of Concept (PoC). The primary focus of this application was its functionality, with security considerations being secondary. Here are the assumptions that will be followed:

3.1 Assumptions

1. **Authentication already exists:**
    - The system architecture diagram presented earlier illustrates the application's functionality post-authentication.
    - It is assumed that the user of SafeMail has already been authenticated via single-factor authentication (username and password).

2. **SSL certification is already present:**
    - As a web application deployed within an organization’s internal network, it is assumed that the webpage is secured using HTTPS.
    - This security measure implies the presence of an SSL certification authorized by a Certificate Authority.

3. **Assumed IP detected by the system is real:**
    - It is assumed that the IP address of the email sender is a valid public-facing address that exists on the internet.
    - The PoC focuses on scenarios where the sender is external to the organization, using an external IP address auto-configured via DHCP.

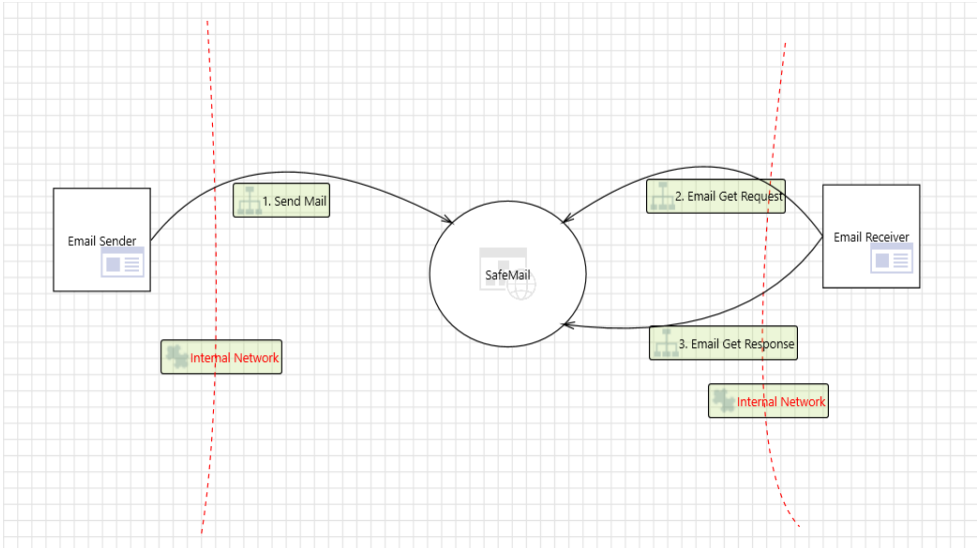
4. **Manually opened client ports:**
    - For Layer 4 inspection (Transport Layer), it is assumed that the sender’s ports were manually opened for testing purposes.
    - This manual configuration allows the node-nmap scan to run properly, and it is assumed to be present in the threat model.

3.2 Dependencies

1. **External APIs:**
   - The system relies on external APIs such as api.ipify.org, AbuseIPDB, and nodeNmap for various functionalities and data points.
   - The system heavily depends on these APIs to perform critical tasks and analyses.

4 Data Flow Diagrams

4.1 Level 0 DFD

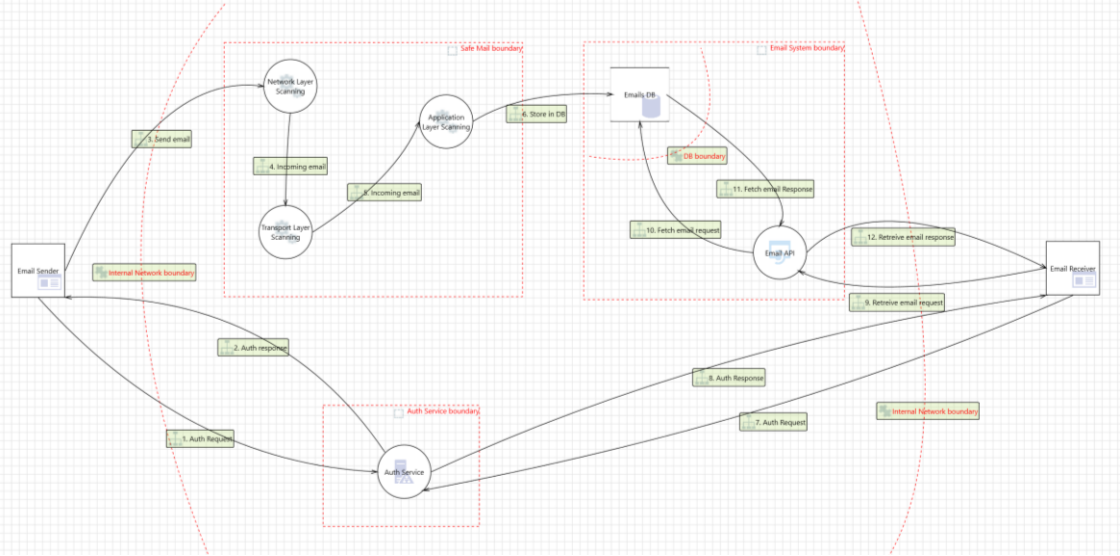


Now that we have established the foundation for SafeMail, we will now discuss the application's security posture and the corresponding threat model associated with it. Typically, a threat model will have 2 levels integrated into it: **Level 0 and Level 1**. A Level 0 threat model illustrates the application's data flow, vulnerabilities, and threats at a 50k foot view. From an organizational perspective, this type of threat model would be beneficial to show to top-level leadership or any part of the organization that does not have a proficient security background but requests to see a threat model for an application (whether it is internally-built or a COTS/GOTS app) that is about to be deployed into a production environment which has significant implications on the overall business.

For SafeMail, the Level 0 DFD breaks down the application into 3 major components: **the email sender, the SafeMail NIDS, and the email receiver**. In SafeMail's architecture, the assumption that we made for the app is that the sender is an external entity, meaning that they would not be authenticated or be allowed to authenticate into the organization's internal network. An example of this would be outsideuser@gmail.com sending an email to a Microsoft employee's internal inbox which we can say something like internalemployee@microsoft.com.

The sender sends the email to the receiver, and that email gets picked up by the SafeMail hback-end which is shown in the center of the diagram. If the email passes the multi-layer inspection from SafeMail, then that's where the next part of the data flow comes into play where the receiver requests and receives a response from the email server.

4.2 Level 1 DFD



In Level 1 threat model diagram, we can see that there are several new components now being presented to illustrate the overall threat landscape of SafeMail. First, we have an authentication server shown at the bottom of the diagram. As mentioned earlier, SafeMail uses single-factor authentication, so the server just takes the username and password as input and then requests the database for the password hash. The auth server is separated in its own trust boundary, which means that it is intended to be separated from the rest of the SafeMail components on the organization's network, and it has its own security controls in place to allow itself to be protected from unauthorized use or tampering.

The biggest vulnerabilities that we can identify in this level 1 threat model are the fact that:

- There is no checks in place to identify if the sender is coming from a genuine source address (IP Spoofing)
- There is no audit or logging system in place to record activity from the SafeMail application (Repudiation)
- External Dependencies such as APIs are used to run the datapoint inspections that SafeMail leverages for phishing detection (Dependencies DoS)
- Database does not encrypt emails (Information Disclosure)

5 Threat Identification

5.1 Threat Modeling Methodology

We have used STRIDE methodology to develop our threat model. The acronym STRIDE stands for six threat categories: **Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege**. The STRIDE methodology helps identify and categorize potential threats to SafeMail, ensuring comprehensive coverage of security aspects. By

systematically applying STRIDE, SafeMail can effectively detect and mitigate threats, enhancing the security and reliability of its phishing email detection capabilities. This structured approach aligns with the goal of SafeMail to accurately detect fraudulent emails while maintaining a low false-positive rate.

5.2 Threat List

- **Spoofing:** Spoofing refers to an attacker's attempt to impersonate another user or system entity to gain unauthorized access or privileges.
- **Tampering:** Tampering involves the unauthorized alteration of data or system components.
- **Repudiation:** Repudiation occurs when an entity denies performing an action, and there is no way to prove otherwise.
- **Information Disclosure:** Information disclosure involves the unauthorized exposure of sensitive information.
- **Denial of Service (DoS):** Denial of Service (DoS) attacks aim to disrupt the availability of a system or service.
- **Elevation of Privilege:** Elevation of privilege occurs when an attacker gains higher access levels or privileges than they are authorized to have.

6 Vulnerability Identification

6.1 Vulnerability List

Based on STRIDE methodology, following vulnerabilities were discovered during threat modeling:

		Threats	Severity
S	**Spoofing**	IP Spoofing	Critical
		Email Address/Header Spoofing	High
T	**Tampering**	Cross-Site Scripting (XSS)	Medium/High
		Email Header Injection	Critical/High
R	**Repudiation**	Logging	Critical
	Information	Absence of Email Encryption in Database	High

I	**Disclosure**	ML Model Exploitation	Critical
		NLP Evasion	Low/Medium
D	**Denial of Service**	Open Ports Exposure	High
		Nmap Scanning Misconfiguration	High
		External Dependencies	Critical
E	**Elevation of Privilege**	Server-Side Code Injection	High
		Basic Authentication	Critical

6.2 Vulnerability Analysis

In this section, we will discuss the vulnerabilities mentioned in the section 5.1.

- **Spoofing:**
  - **IP Spoofing -** Attackers can spoof IP addresses to bypass IP-based filtering, making malicious traffic appear legitimate.
  - **Email Address/Header Spoofing -** Same goes for Email Address/Header Spoofing, making fraudulent email appear legitimate.

- **Tampering:**
  - **Cross-Site Scripting (XSS) -** Attacker can use Stored/Blind XSS techniques to execute arbitrary scripts; Reflected XSS is also possible but we are considering it as Low priority.
  - **Email Header Injection -** Allow attackers to send emails to unintended recipients for phishing or spamming purposes. This can be done by injecting additional headers with arbitrary values into email headers without proper sanitization.

- **Repudiation:**
  - **Logging -** Absence of auditing/logging in the application leads security incidents be undetected.

- **Information Disclosure:**
  - **Absence of Email Encryption in Database -** The application doesn't analyze encrypted emails, as all the models are trained on the plain text dataset. Emails are stored in the database in plain text, prone to SQL Injection.
  - **ML Model Exploitation -** Dataset size used to train the ML model is small (150 samples) making it vulnerable to adversarial techniques to increase the number of false positives, hence defeating the purpose of this application.

- **NLP Evasion -** If an attacker gains access to the dataset, they can simply construct a phishing email text without those words to manipulate the model into detecting that text as legitimate.

- **Denial of Service (DoS):**
  - **Open Ports Exposure -** Open ports can be exploited by attackers to gain unauthorized access, launch DoS attacks, or deploy malware.
  - **Nmap Scanning Misconfiguration -** We are using Node Nmap in transport layer for scanning open ports. Incorrect or insufficient configuration can lead to incomplete scanning, leaving open ports unnoticed and exploitable.
  - **External Dependencies -** Project uses third-party APIs like ipify (Public IP Address API), Abuse-IPDB (community-based IP blacklist database). If any of the APIs are unavailable, it can lead to DoS.

- **Elevation of Privilege:**
  - **Server-Side Code Injection -** Spawning child processes for executing the python scripts from node.js backend. If user input is not sanitized, it can lead to command injection vulnerabilities.
  - **Basic Authentication -** Application uses simple password-based authentication, making it vulnerable to brute force/dictionary attacks.

7 Risk Assessment

7.1 Risk Criteria

- **Likelihood:**
  - **Low:** Unlikely to occur; required specific conditions or knowledge.
  - **Medium:** Possible to occur; could be exploited with some effort.
  - **High:** Likely to occur; easily exploitable or frequently targeted.

- **Impact:**
  - **Low:** Limited damage; minor inconvenience or exposure.
  - **Medium:** Moderate damage; affects some users or data, but not critical.
  - **High:** Severe damage; major disruption, significant data loss, or critical system compromise.

7.2 Risk Analysis

- **Spoofing:**
  - **IP Spoofing:**
    - **Likelihood:** Medium
    - **Impact:** Medium
    - **Justification:** IP-based filtering is common, but sophisticated attackers can bypass it.

  - **Email Address/Header Spoofing:**
    - **Likelihood:** Medium
    - **Impact:** Medium

        - **Justification:** Spoofed emails can bypass some email filters, leading to phishing attacks.

- **Tampering:**
    - **Cross-Site Scripting (XSS):**
        - **Likelihood:** High
        - **Impact:** High
        - **Justification:** XSS can lead to severe security breaches, including session hijacking.
    - **Email Header Injection:**
        - **Likelihood:** Medium
        - **Impact:** High
        - **Justification:** Allows attackers to conduct phishing or spam attacks, potentially affecting many users.

- **Repudiation:**
    - **Logging:**
        - **Likelihood:** Medium
        - **Impact:** High
        - **Justification:** Lack of logging makes it difficult to detect and investigate security incidents.

- **Information Disclosure:**
    - **Absence of Email Encryption in Database:**
        - **Likelihood:** High
        - **Impact:** High
        - **Justification:** Storing emails in plain text makes them vulnerable to SQL Injection and unauthorized access.

    - **ML Model Exploitation:**
        - **Likelihood:** High
        - **Impact:** Medium
        - **Justification:** Small dataset size increases the vulnerability to adversarial attacks, affecting model accuracy.

    - **NLP Evasion:**
        - **Likelihood:** Medium
        - **Impact:** Medium
        - **Justification:** If attackers know the dataset, they can craft messages to evade detection, reducing the effectiveness of the model.

- **Denial of Service (DoS):**
    - **Open Ports Exposure:**
        - **Likelihood:** High
        - **Impact:** High
        - **Justification:** Open ports are common targets for DoS attacks and unauthorized access.

- **Nmap Scanning Misconfiguration:**
  - **Likelihood:** Medium
  - **Impact:** Medium
  - **Justification:** Misconfigurations can lead to missed open ports, leaving the system vulnerable.

- **External Dependencies:**
  - **Likelihood:** Medium
  - **Impact:** High
  - **Justification:** Dependency on external APIs can cause service disruption if the APIs are unavailable.

- **Elevation of Privilege:**
  - **Server-Side Code Injection:**
    - **Likelihood:** High
    - **Impact:** High
    - **Justification:** Unsanitized inputs can lead to command injection, compromising the server.

  - **Basic Authentication:**
    - **Likelihood:** High
    - **Impact:** Medium
    - **Justification:** Simple password-based authentication is vulnerable to brute force attacks, potentially leading to unauthorized access.

7.3 Risk Matrix

Threat	Likelihood	Impact	Risk Level
Spoofing			
IP Spoofing	Medium	Medium	Moderate
Email Address/Header Spoofing	Medium	Medium	Moderate
Tampering			
Cross-Site Scripting (XSS)	High	High	Critical
Email Header Injection	Medium	High	High
Repudiation			

Logging	Medium	High	High
Information Disclosure			
Absence of Email Encryption	High	High	Critical
ML Model Exploitation	High	Medium	High
NLP Evasion	Medium	Medium	Moderate
Denial of Service (DoS)			
Open Ports Exposure	High	High	Critical
Nmap Scanning Misconfiguration	Medium	Medium	Moderate
External Dependencies	Medium	High	High
Elevation of Privilege			
Server-Side Code Injection	High	High	Critical
Basic Authentication	High	Medium	High

8 Mitigation Strategies

8.1 Existing Controls

- **Authentication**: Single-factor authentication is already in place.
- **SSL Certification**: Secure Socket Layer (SSL) certification is implemented.

8.2 Proposed Mitigations

- **IP Spoofing Mitigation:**
  - Implement IP validation mechanisms to verify the authenticity of the sender's IP address.

- **Logging and Auditing:**
  - Integrate comprehensive logging and auditing systems to monitor all activities and detect potential repudiation attempts.

- **Email Encryption:**
  - Encrypt emails stored in the database using advanced encryption techniques like Homomorphic Encryption (HE).

  - Perform homomorphic encryption to allow operations on the encrypted emails without decrypting them.

- **ML Model Improvement:**
  - Expand the training dataset for the machine learning model to improve accuracy and reduce vulnerability to adversarial attacks.

- **Dependency Management:**
  - Develop in-house APIs to minimize reliance on third-party services and ensure data integrity.

- **Enhanced Authentication:**
  - Implement Multi-Factor Authentication (MFA) to enhance security.

- **Web Application Firewall:**
  - Deploy a Web Application Firewall (WAF) to protect against web-based attacks.

- **Regular Vulnerability Scanning:**
  - Conduct regular vulnerability scanning and IP scanning using tools like nodeNmap to detect and mitigate potential threats.

- **Email Security Protocols:**
  - Utilize email security protocols such as DMARC, DKIM, and SPF to validate the authenticity of email sources.

- **NLP Evasion Countermeasures:**
  - Regularly update the Natural Language Processing (NLP) filters and models to identify and adapt to new phishing techniques.

- **API Deployment of ML Models:**
  - Currently, we are spawning child processes to run Python scripts for our ML models. This leads to higher memory utilization with increasing API calls. To mitigate this, we plan to deploy the ML model on a dedicated server and expose it through an API. This will streamline the processing, reduce memory consumption, and improve scalability.

8.3 Mitigation Plan

1. **Short-Term Actions (0-3 months):**
   a. Implement IP Validation mechanisms.
   b. Integrate logging and auditing systems.
   c. Implement email security protocols (DMARC, DKIM, SPF).

2. **Medium-Term Actions (3-6 months):**
   a. Expand the ML training set.
   b. Develop in-house APIs to reduce third-party dependency.

c. Deploy Web Application Firewall (WAF).
d. Begin deploying the ML models on a dedicated server and expose it through an API.

3. **Long-Term Actions (6-12 months):**
   a. Implement Multi-Factor Authentication (MFA).
   b. Encrypt emails stored in the database.
   c. Perform operations on encrypted emails using Homomorphic Encryption (HE).
   d. Regularly refine NLP filters and models to counter evasion attacks.
   e. Fully transition to using the API-exposed ML models to ensure efficient resource utilization.

9 Conclusion

The threat model analysis for SafeMail underscores the importance of a multi-layered security approach to effectively detect and mitigate phishing threats. By implementing robust mitigation strategies such as IP validation, comprehensive logging, email encryption, and deploying ML models via APIs, SafeMail can enhance its security posture significantly. Regular vulnerability scanning, enhanced authentication, and continuous updates to NLP filters ensure ongoing protection against evolving threats. The proposed mitigation plan provides a structured timeline to address vulnerabilities systematically, ensuring SafeMail remains a reliable and secure NIDS solution for organizations. This comprehensive approach not only improves security but also enhances the overall effectiveness of phishing detection.

Appendix A: Glossary of Terms

- **Threat Model:** A structured approach to identifying, analyzing, and mitigating potential security threats to a system.
- **STRIDE:** A methodology for categorizing threats into six categories: Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege.
- **Network Intrusion Detection System (NIDS):** A system designed to monitor network traffic and detect potential security threats or unauthorized access attempts.
- **Phishing:** A cybercrime technique that involves sending fraudulent emails or messages to trick users into revealing sensitive information or installing malware.
- **IP Spoofing:** A technique used by attackers to masquerade as a trusted IP address to gain unauthorized access or bypass security controls.
- **Cross-Site Scripting (XSS):** A security vulnerability that allows attackers to inject malicious scripts into web pages, potentially compromising user data or hijacking sessions.
- **SQL Injection:** A technique used by attackers to inject malicious SQL statements into application input fields, potentially exposing or manipulating data stored in databases.

- **Denial of Service (DoS):** A type of cyber-attack that aims to disrupt or disable a system or service by overwhelming it with traffic or requests, making it unavailable to legitimate users.

- **Multi-Factor Authentication (MFA):** A security mechanism that requires users to provide multiple forms of authentication, such as a password and a one-time code, to gain access to a system or service.

- **Homomorphic Encryption (HE):** A form of encryption that allows specific computations to be performed on encrypted data without decrypting it first, preserving data confidentiality.