Secure Software Design & Engineering(CY-321)

System Architecture & Secure Design

Anti-Phishing Browser Extension

Group Members

1. Shameer Awais (2022428)
2. Rooshan Riaz (2022506)
3. Naqi Raza (2022574)
4. M. Yasir (2022455)

Submission Date: 21/03/2025

Ghulam Ishaq Khan Institute of Engineering Sciences and Technology

Secure Architecture

1. **Browser Extension:** This is the primary deployment environment where the user interacts with the phishing detection system. It acts as the entry point for user interaction, phishing URL scanning, and alert generation.

2. **User Interface:**

    - **Alerts:** Notifies users of potential phishing threats detected.
    - **Settings & Privacy Settings:** Allow the user to customize extension behavior and control privacy preferences.
    - **Purpose:** Provides a user-friendly interface for managing the extension and responding to threats.

3. **Phishing Detection Engine:** This module handles the core phishing detection logic using machine learning (ML) and real-time intelligence.

    - **Google Safe Browsinjg API:**
        (a) Pulls real-time phishing data.
        (b) **Purpose:** Keeps detection up-to-date with emerging threats.
    - **Machine Learning Engine:**
        (a) Analyzes URLs for suspicious patterns.
        (b) **Purpose:** Serves as the checkpoint for detecting potential phishing links.

4. **Communication Layer:**

    This component handles secure data transmission and manages backend communications.

    - **Secure API:**
        (a) Central point for secure data exchange between browser and backend.
        (b) **Purpose:** Enforces encrypted communication and controls access.
    - **Rate Limiting:**
        (a) Limits the number of requests to prevent DoS attacks.
        (b) **Purpose:** Protects the API from abuse.

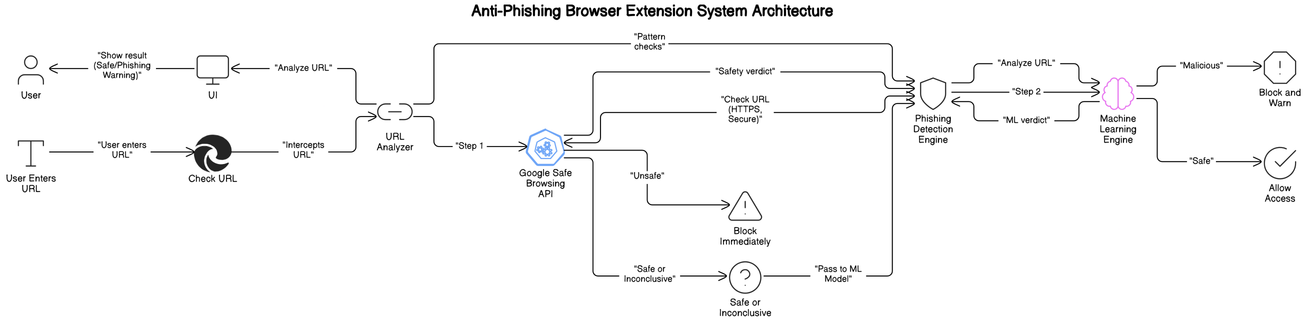

Figure 1: Architecture Diagram

Security Controls

The technical measures implemented in the architecture to safeguard the system from unauthorized access, data breaches, and attacks are as follows:

1. **HTTPS Communication:**

   The communication between client and server uses HTTPS.

   - **Purpose:** Prevents MITM (Man-in-the-Middle) attacks by encrypting transmitted data.

2. **Phishing Bypass:**

   (a) Addresses bypass attempts.

   (b) **Purpose:** Ensures robust detection even for obfuscated threats.

3. **Data Leakage Prevention:**

   (a) Controls access and protects sensitive information.

   (b) **Purpose:** Prevents unintended exposure of user data.

4. **Reverse Engineering Protection:**

   (a) Implements techniques to prevent binary inspection.

   (b) **Purpose:** Protects system logic.

5. **Rate Limiting:**

   Controls how frequently requests can be made to the system.

   - **Purpose:** Prevents DoS attacks and abuse of services.

Security Measures

The following are the strategic decisions and architectural practices incorporated to ensure a secure system design from the ground up.

1. **Code Obfuscation**

   - The source code of the browser extension is obfuscated.
   - **Purpose:** Makes reverse engineering difficult and protects intellectual property and logic.

2. **Input Sanitization**

   - A Sanitization Layer checks and cleans incoming data before it is processed by the ML inference engine.
   - **Purpose:** Protects against injection attacks and malformed input.

3. **Secure API Design**

   - APIs are untrusted by default, require validation, and communicate over secure channels.
   - Purpose: Mitigates API abuse, unauthorized access, and data leakage.

4. **User Consent Management**

   - Before any data is collected or processed, the user must provide consent.
   - **Purpose:** Complies with data privacy laws and ethical standards.