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

   - **Threat Intelligence Integration:**
     (a) Pulls real-time phishing data feeds from external sources.
     (b) **Purpose:** Keeps detection up-to-date with emerging threats.
   - **URL Scanner:**
     (a) Analyzes URLs for suspicious patterns.
     (b) **Purpose:** Serves as the initial checkpoint for detecting potential phishing links.
   - **ML Engine:**
     (a) Uses trained ML models to classify URLs as safe or malicious.
     (b) **Purpose:** Provides intelligent detection using historical data.
   - **ML Inference:**
     (a) Applies lightweight ML models to infer phishing threats quickly.
     (b) **Purpose:** Reduces overhead during real-time use.
   - **Sanitization Layer:**
     (a) Filters and sanitizes input data to prevent injection or malformed input attacks.
     (b) **Purpose:** Ensures robust input validation before ML processing.
   - **ML Training:**
     (a) Continuously improves the ML engine using new data.
     (b) **Purpose:** Keeps detection models adaptive and resilient.

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.

- **Token-Based Authentication:**
  (a) Uses authentication tokens for verifying identity.
  (b) **Purpose:** Prevents unauthorized access.
- **Session Token Management:**
  (a) Manages user sessions securely.
  (b) **Purpose:** Protects against session hijacking.
- **Zero Trust Access:**
  (a) Treats all access attempts as untrusted by default.
  (b) **Purpose:** Enforces strict access control.
- **Logging with RBAC (Role-Based Access Control):**
  (a) Maintains audit logs with access controls.
  (b) **Purpose:** Supports traceability and accountability.
- **Rate Limiting:**
  (a) Limits the number of requests to prevent DoS attacks.
  (b) **Purpose:** Protects the API from abuse.

5. **Threat Modeling:**
   Identifying potential threats and integrates preventive mechanisms.

   - **Code Obfuscation:**
     (a) Makes source code harder to reverse engineer.
     (b) **Purpose:** Minimizes risks of exploitation.
   - **Phishing Bypass:**
     (a) Addresses bypass attempts.
     (b) **Purpose:** Ensures robust detection even for obfuscated threats.
   - **Data Leakage Prevention:**
     (a) Controls access and protects sensitive information.
     (b) **Purpose:** Prevents unintended exposure of user data.
   - **Reverse Engineering Protection:**
     (a) Implements techniques to prevent binary inspection.
     (b) **Purpose:** Protects system logic.

6. **Security Measures:**

   - **User Consent:**
     (a) Ensures the user is aware of data usage.
     (b) **Purpose:** Upholds privacy laws and ethical standards.

- **HTTPS Communication:**
  (a) Encrypts data during transmission.
  (b) **Purpose:** Secures the channel from man-in-the-middle (MITM) attacks.
- **Data Encryption:**
  (a) Encrypts sensitive information at rest and in transit.
  (b) **Purpose:** Protects confidentiality and integrity.

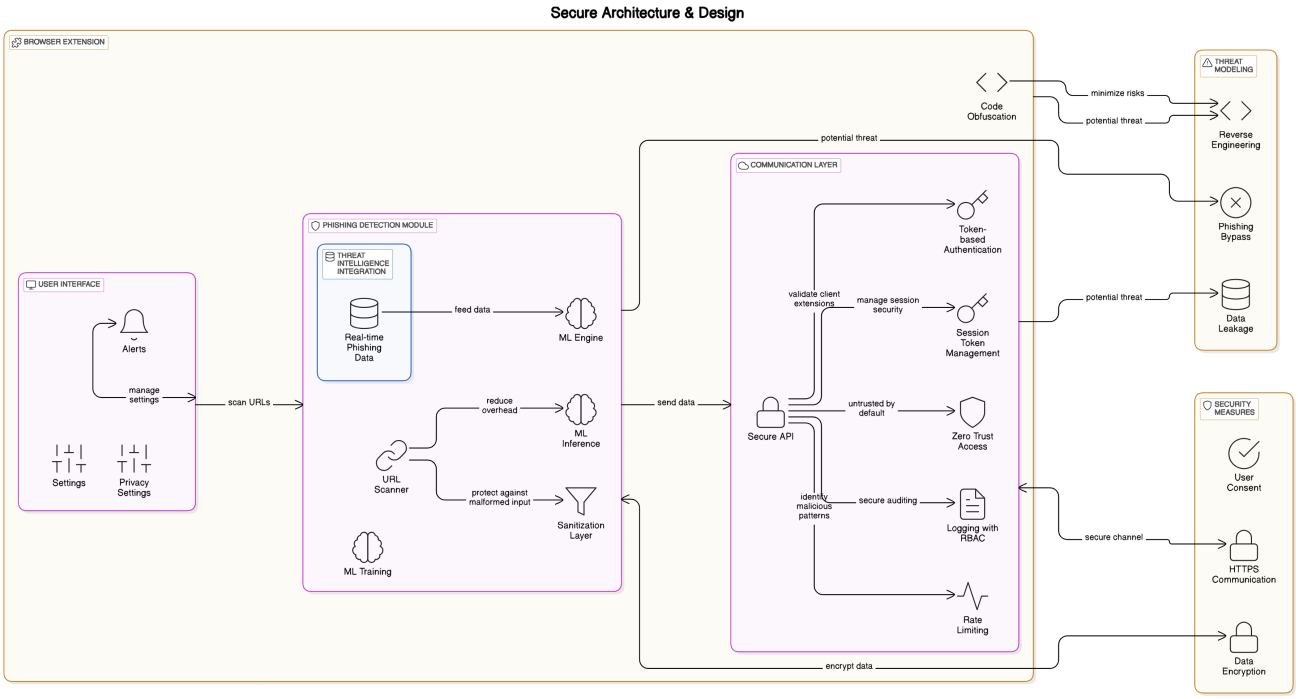

Figure 1: Architecture Diagram

Security Controls

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

1. **Authentication:**

   - **Token-Based Authentication:**
     Ensures that only authenticated users or browser extensions can access secure APIs. Each session or request must present a valid token.
     – **Purpose:** Prevent unauthorized access.
   - **Session Token Management:**
     Tokens are managed securely to avoid session hijacking and replay attacks.
     – **Purpose:** Maintains secure and valid user sessions.

2. **Encryption:**

- **Data Encryption:**
  All sensitive data is encrypted both in transit (using HTTPS) and at rest.
  - **Purpose:** Protects confidentiality and integrity of data.
- **HTTPS Communication:**
  The communication between client and server uses HTTPS.
  - **Purpose:** Prevents MITM (Man-in-the-Middle) attacks by encrypting transmitted data.

3. **Access Control:**

- **Zero Trust Access:**
  No user or system is trusted by default, even if they are inside the network perimeter.
  - **Purpose:** Restricts access unless explicitly granted.
- **Role-Based Access Control (RBAC):**
  Logging and administrative access are restricted based on roles.
  - **Purpose:** Ensures that users can only perform actions within their permission level
- **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.

(a) **Code Obfuscation**

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

(b) **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.

(c) **Threat Intelligence Integration**

- Feeds real-time phishing data into the system.
- **Purpose:** Proactively defends against known threats.

(d) **Machine Learning Inference & Overhead Reduction**

- Lightweight models used for real-time detection reduce performance impact.

- **Purpose:** Proactively defends against known threats.

(e) **Secure API Design**

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

(f) **Logging and Auditing**

- Logs are generated with RBAC enforcement for secure auditing.
- **Purpose:** Enables traceability of actions for incident response and compliance.

(g) **User Consent Management**

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