Alexandria National University

Report: A Secure Document Vault with Authentication, Integrity, and Encryption

Prepared by:

Abdallah Mohammed Hejazi 2205228

Ziad Mahmoud Abdelgwad 2205021

Mohamed Hany Khattab 2205053

Ibrahim Hassan Ibrahim 2205001

Youssef saeed thabet 2205064

Alexandria National University

1. Introduction

This report provides an analysis of the GitHub repository titled "A Secure Document Vault with Authentication, Integrity, and Encryption" by Abdallah M. Hegazy and his team. The project implements a secure document storage system that ensures confidentiality, integrity, and authentication. Below, we break down the encryption and authentication flow, followed by an explanation of each implemented feature.

2. Encryption and Authentication Flow

A. Authentication Flow

1. **User Registration:**

- - A user provides a username and password.
- - The system hashes the password using a secure hashing algorithm (e.g., bcrypt, PBKDF2) and stores it in the database.
- - A unique salt is generated for each user to prevent rainbow table attacks.

2. **User Login:**

- - The user enters their credentials (username and password).
- - The system retrieves the stored hash and salt from the database.
- - The entered password is hashed with the stored salt and compared against the stored hash.
- - If they match, the user is authenticated and granted access; otherwise, access is denied.

3. **Session Management:**

- - Upon successful authentication, a session token (e.g., JWT) is generated and stored securely.
- - The token is sent to the client and must be included in subsequent requests for authorization.

Alexandria National University

B. Encryption Flow

1. **File Upload:**

- - The user selects a file to upload.
- - A symmetric encryption key (e.g., AES-256) is generated for the file.
- - The file is encrypted using this key.

2. **Key Management:**

- - The symmetric key is then encrypted using the user's public key (asymmetric encryption, e.g., RSA).
- - The encrypted symmetric key is stored in the database, while the encrypted file is stored in the vault.

3. **File Download/Decryption:**

- - The user requests a file.
- - The system retrieves the encrypted symmetric key and decrypts it using the user's private key.
- - The decrypted symmetric key is then used to decrypt the file.

4. **Integrity Verification (Optional):**

- - A cryptographic hash (e.g., SHA-256) of the original file is stored.
- - Upon retrieval, the decrypted file is hashed again and compared to the stored hash to ensure no tampering occurred.

3. Explanation of Implemented Features

A. User Authentication

- - **Secure Password Storage:** Passwords are hashed with a salt to prevent brute-force and rainbow table attacks.

- - **Session Tokens:** JSON Web Tokens (JWT) or similar mechanisms ensure secure session management.
- - **Rate Limiting & Lockout:** Prevents brute-force attacks by limiting login attempts.

B. File Encryption & Decryption

- - **Symmetric Encryption (AES-256):** Used for encrypting files due to its speed and security.
- - **Asymmetric Encryption (RSA):** Used to securely store the symmetric key by encrypting it with the user's public key.
- - **Key Management:** Ensures that even if the database is compromised, encrypted files remain secure without the user's private key.

C. Data Integrity Protection

- - **Hash Verification (SHA-256):** Files are hashed before storage, and the hash is verified upon retrieval to detect tampering.
- - **Digital Signatures (Optional):** Ensures that files are not altered by verifying a signature generated with the user's private key.

D. Secure File Storage & Access Control

- - **Role-Based Access Control (RBAC):** Differentiates between admin and regular users, restricting file access accordingly.
- - **Secure File Deletion:** Ensures files are permanently erased using secure deletion methods.

E. Audit Logging

- - **Access Logs:** Records all file access, modifications, and login attempts for security monitoring.
- - **Anomaly Detection:** Alerts administrators of suspicious activities (e.g., multiple failed login attempts).

Alexandria National University

F. Secure Transmission (HTTPS/TLS)

- - All communications between the client and server are encrypted using TLS to prevent man-in-the-middle attacks.

4. Conclusion

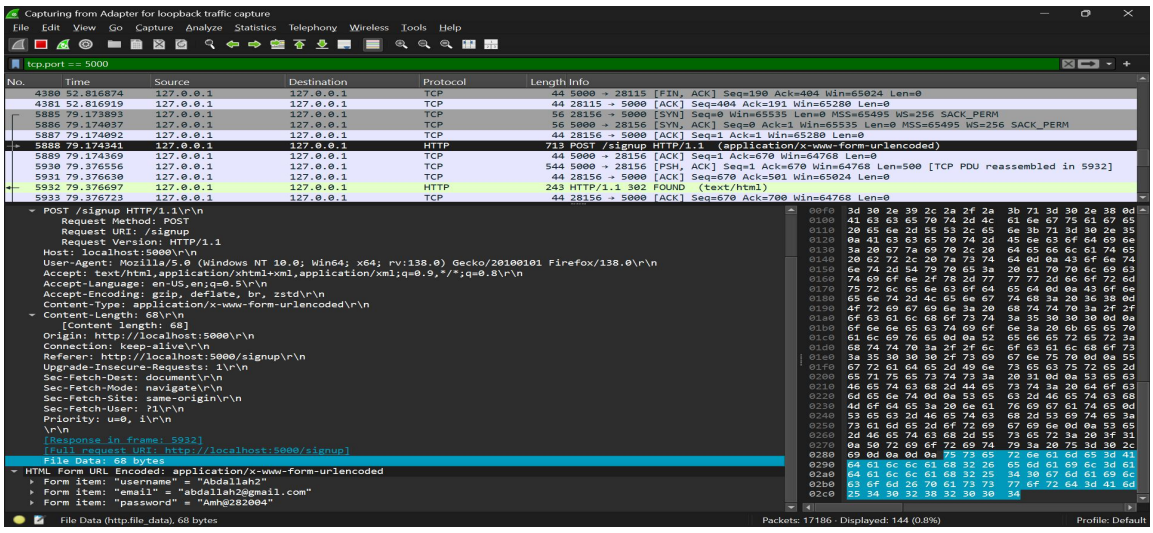
This project provides a robust and secure document vault by combining multiple security mechanisms: Strong **authentication** with salted password hashing and session management, **encryption** using both symmetric (AES) and asymmetric (RSA) techniques, **integrity checks** via hashing and optional digital signatures, and **access control** and **audit logging** for accountability.

By following these best practices, the system ensures that documents remain confidential, unaltered, and accessible only to authorized users.

5. MIMA Simulation: Security Analysis Over HTTP vs HTTPS

A. Over HTTP

The data over HTTP is sent as plaintext, making it visible to attackers. For example, during user signup, captured traffic reveals:



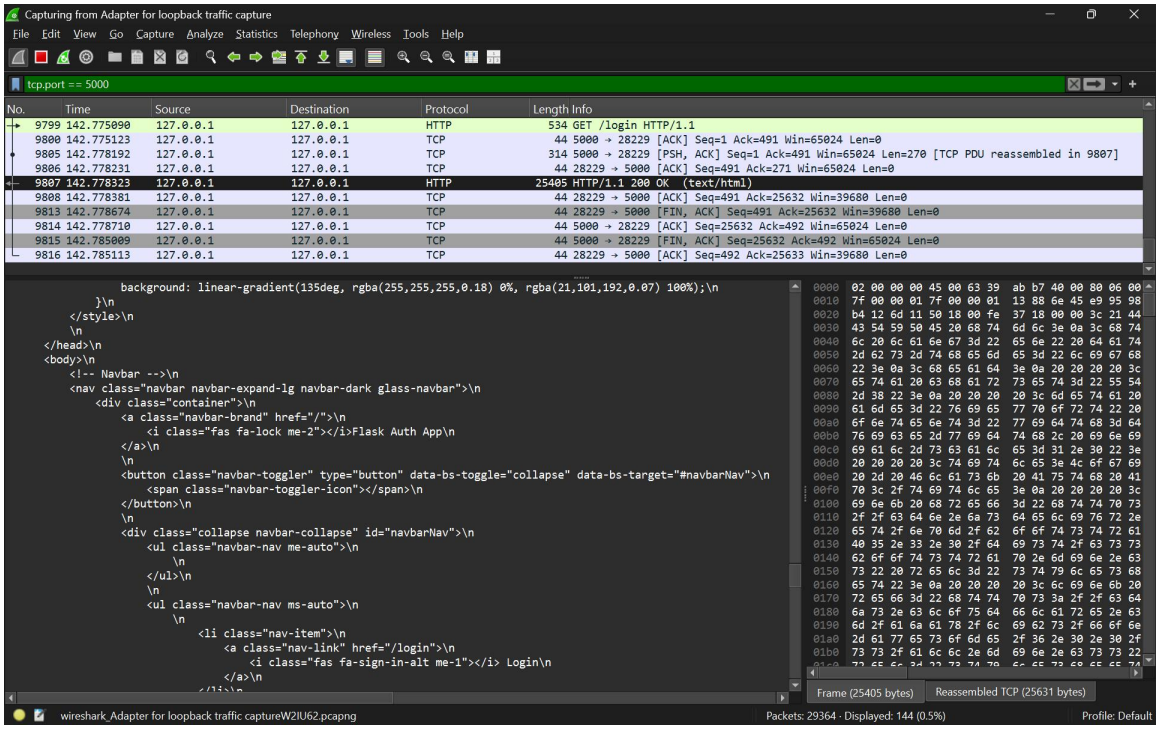
Alexandria National University

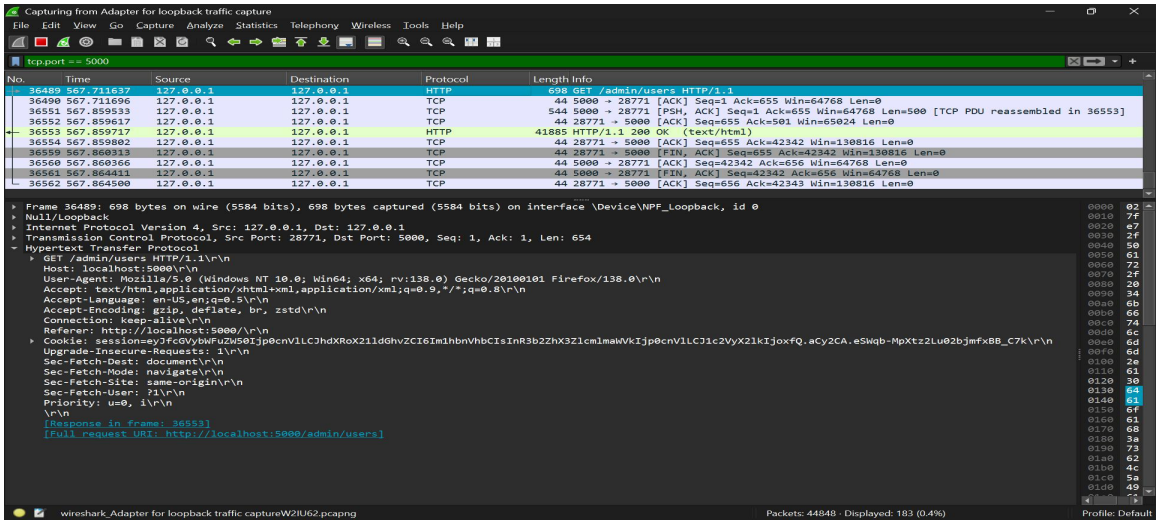
- - Username: Abdallah2

  - Email: abdallah2@gmail.com

  - Password: Amh@282004

- The server response also exposes the website's base code and session keys, enabling session hijacking attacks.

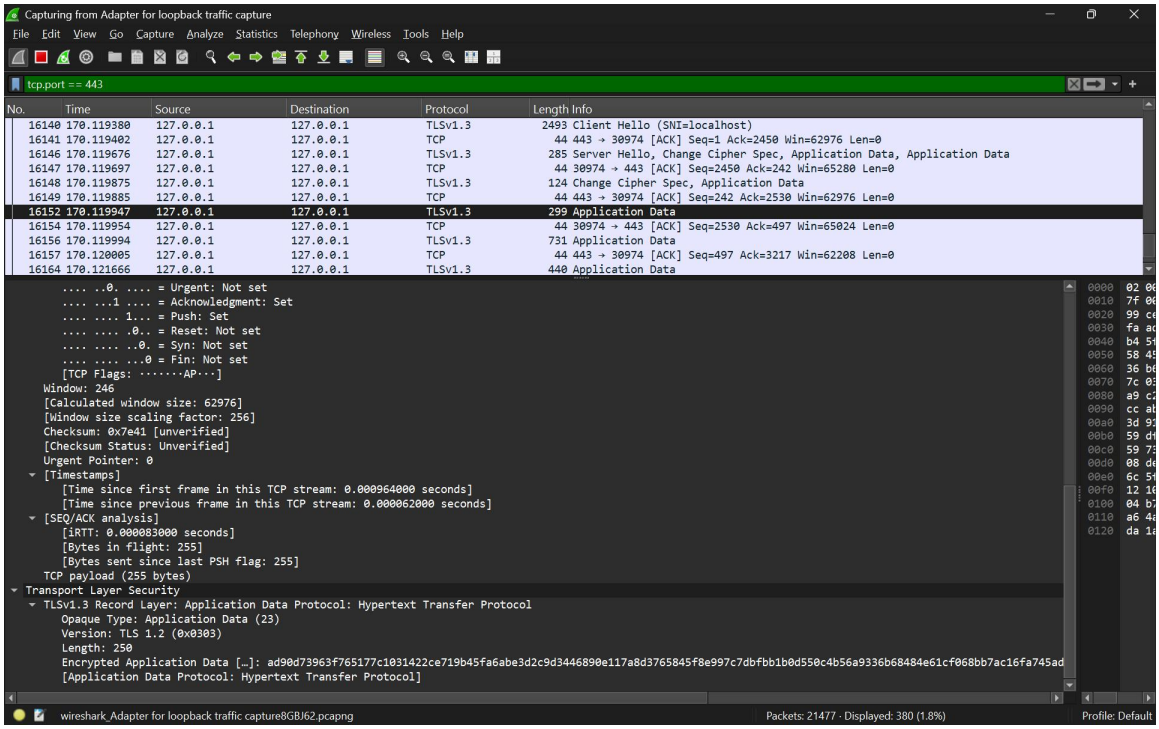




Alexandria National University

B. Over HTTPS

HTTPS encrypts data using TLS (e.g., TLS v1.3), hiding application data during transmission. This ensures secure registration and prevents eavesdropping.



- Key observations:
- - No plaintext credentials are exposed.

  - Session keys and responses are encrypted.

  - TLS v1.3 provides modern cryptographic protections.