Data Integrity Final Project

BY

Ibrahim Sameh 2205002
Ahmed Hesham 2205012
Ahmed Mohamed 2205134
Ahmed Fawzy 2205156
Ahmed Mourad 2205010

Encryption and Authentication Flow Report

Overview

The system is a Flask-based secure document management platform. It supports:
- User authentication via email/password and OAuth (GitHub, Auth0)
- Two-Factor Authentication (2FA) using TOTP
- AES encryption for file confidentiality
- HMAC for file integrity
- RSA digital signatures for authenticity and non-repudiation

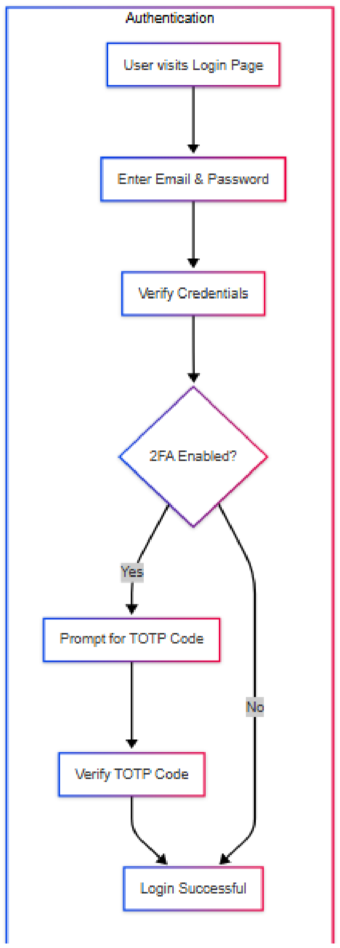
1. Authentication Flow

A. Local Email/Password Login

1. Signup:
- User provides email and strong password (min 12 chars, with complexity).
- Email and hashed password (pbkdf2:sha256) are stored in the database.
- RSA key pair is generated and stored (PEM format) for future document signing.

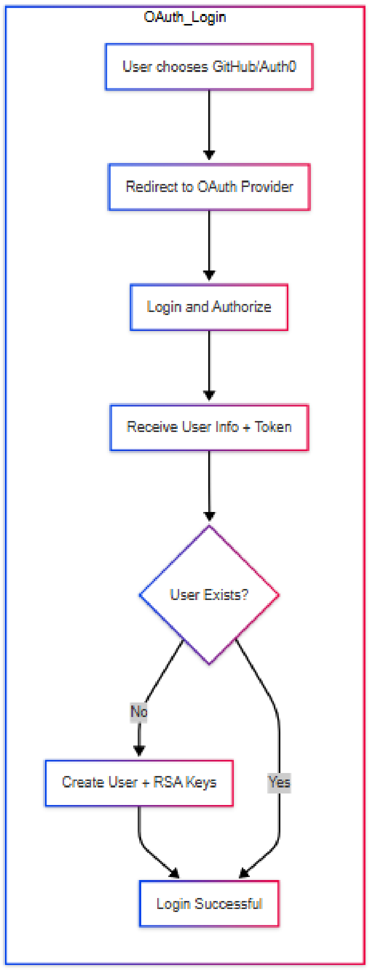
2. Login:
- Password is verified using check_password_hash.
- If 2FA is enabled, user is redirected to verify TOTP code.

3. Two-Factor Authentication (2FA):
- Implemented using pyotp.
- Users can scan a QR code with an authenticator app.
- Upon successful TOTP validation, session['2fa_verified'] is set.



B. OAuth Login (GitHub / Auth0)

1. OAuth Flow:
- Users can log in using their GitHub or Auth0 account.
- After redirection and token exchange, user data (especially email) is retrieved.
- If new, a user record is created in the database with RSA key pair.
- No local password is set for OAuth users unless manually assigned.



2. Encryption Flow (File Upload & Download)

A. Upload (Confidentiality & Integrity)

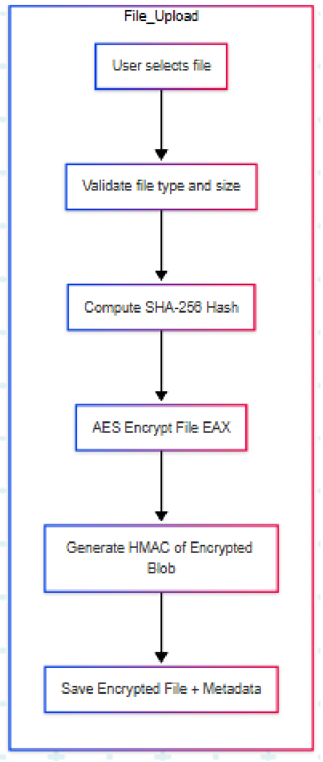
1. File Validations:
- Allowed extensions: .pdf, .docx, .txt
- Max size: 16 MB

2. Hashing:
- Original file SHA-256 hash is computed and stored (file_hash)

3. Encryption (AES-256):
- AES in EAX mode is used (Crypto.Cipher.AES)
- nonce (16 bytes), tag (16 bytes), and ciphertext are concatenated

4. HMAC (Integrity):
- An HMAC-SHA256 is generated on the full encrypted blob using a 32-byte key.
- Stored in file_hmac

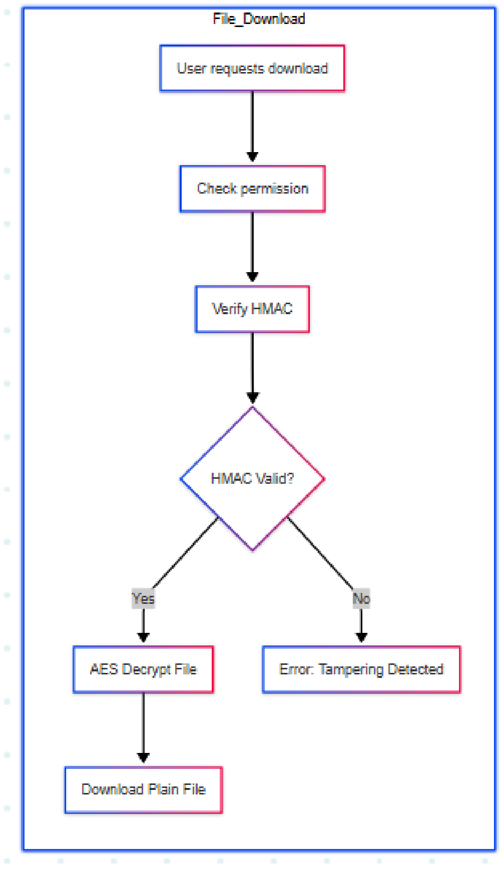
5. Storage:
- The encrypted file is saved with a generated filename.
- Metadata is saved in the Document table.


File_Upload
User selects file
Validate file type and size
Compute SHA-256 Hash
AES Encrypt File EAX
Generate HMAC of Encrypted Blob
Save Encrypted File + Metadata

B. Download (Decryption & Integrity Check)

1. Authorization: Only owners/admins can download.
2. HMAC Verification:
- Recalculate HMAC on encrypted blob and compare with stored file_hmac.
- Prevents tampering or corruption.

3. Decryption:
- Extract nonce and tag from the encrypted file.
- Decrypt with AES using stored encryption key.
- cipher.decrypt_and_verify(ciphertext, tag) ensures authenticity.



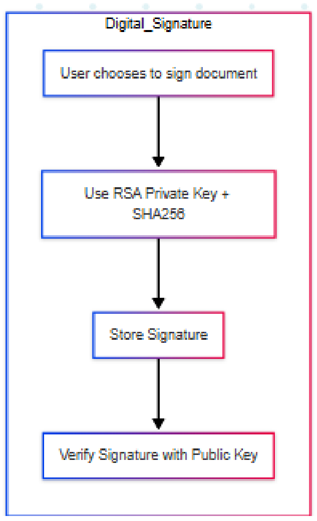
3. Digital Signature Flow (Authenticity & Non-repudiation)

A. Signing:

- After upload, users can digitally sign the encrypted file.
- RSA private key is used to sign using PSS padding and SHA-256 hash.
- Signature is stored as a hexadecimal string.

B. Signature Verification:

- Anyone with access (user/admin) can verify a signature using:
  * Encrypted data
  * Stored signature
  * User's public key
- Confirms the file wasn't altered and was signed by the key owner.

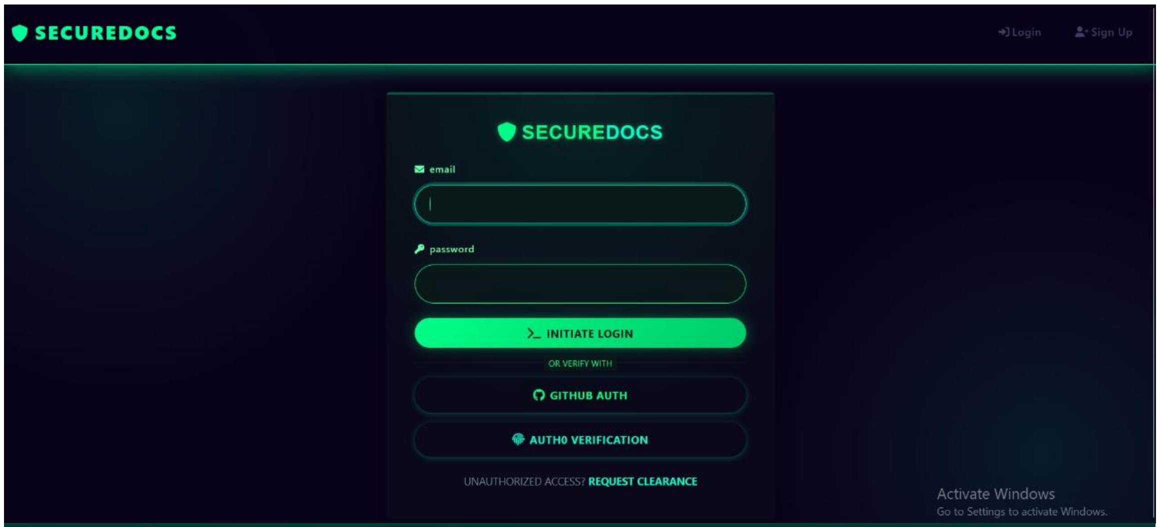

Digital_Signature
User chooses to sign document
Use RSA Private Key + SHA256
Store Signature
Verify Signature with Public Key

4. Session & Security Controls

- Session Timeout: 15 minutes (PERMANENT_SESSION_LIFETIME)
- Cookie Security: SESSION_COOKIE_SECURE, HttpOnly, and SameSite settings
- Error Handling & Logging: All access violations and failures are logged via log_action.

Screen Shots of Implemented features:

-Login&Signup


SECUREDOCS

Login

Sign Up

SECUREDOCS

email

password

INITIATE LOGIN

OR VERIFY WITH

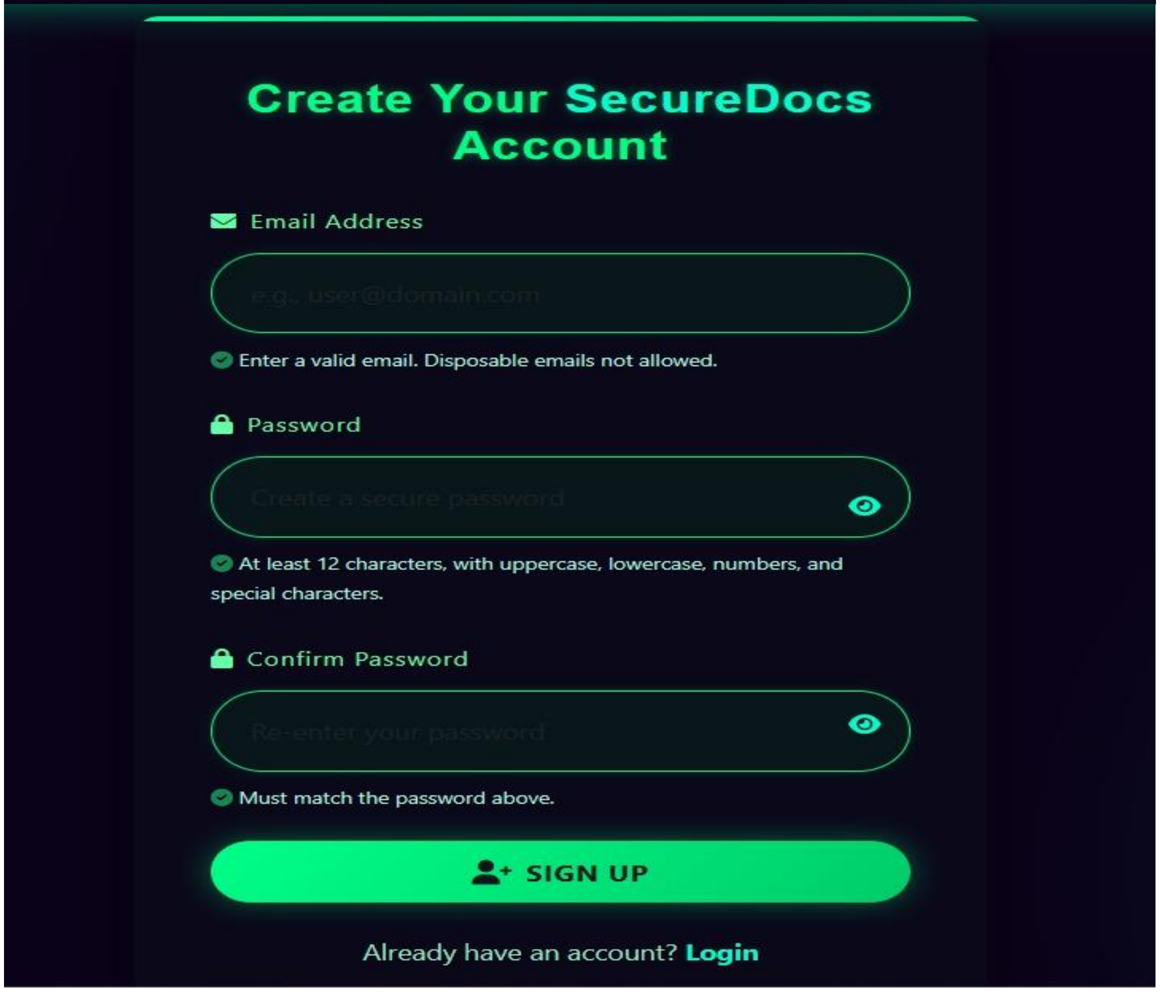
GITHUB AUTH

AUTH0 VERIFICATION

UNAUTHORIZED ACCESS? **REQUEST CLEARANCE**

Activate Windows

Go to Settings to activate Windows.


Create Your SecureDocs Account

Email Address

e.g., user@domain.com

Enter a valid email. Disposable emails not allowed.

Password

Create a secure password

At least 12 characters, with uppercase, lowercase, numbers, and special characters.

Confirm Password

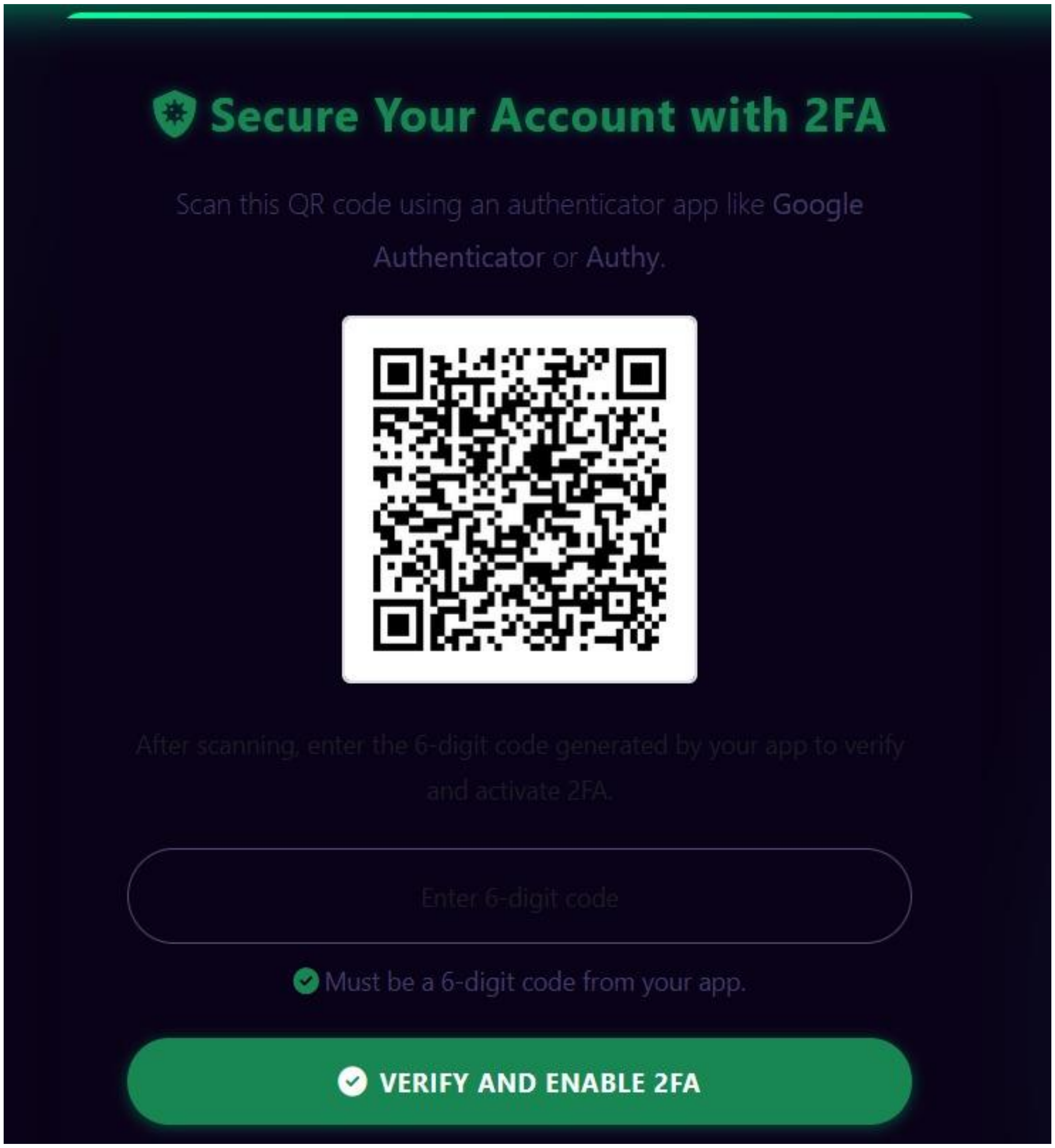
Re-enter your password

Must match the password above.

SIGN UP

Already have an account? **Login**

-2FA



Secure Your Account with 2FA

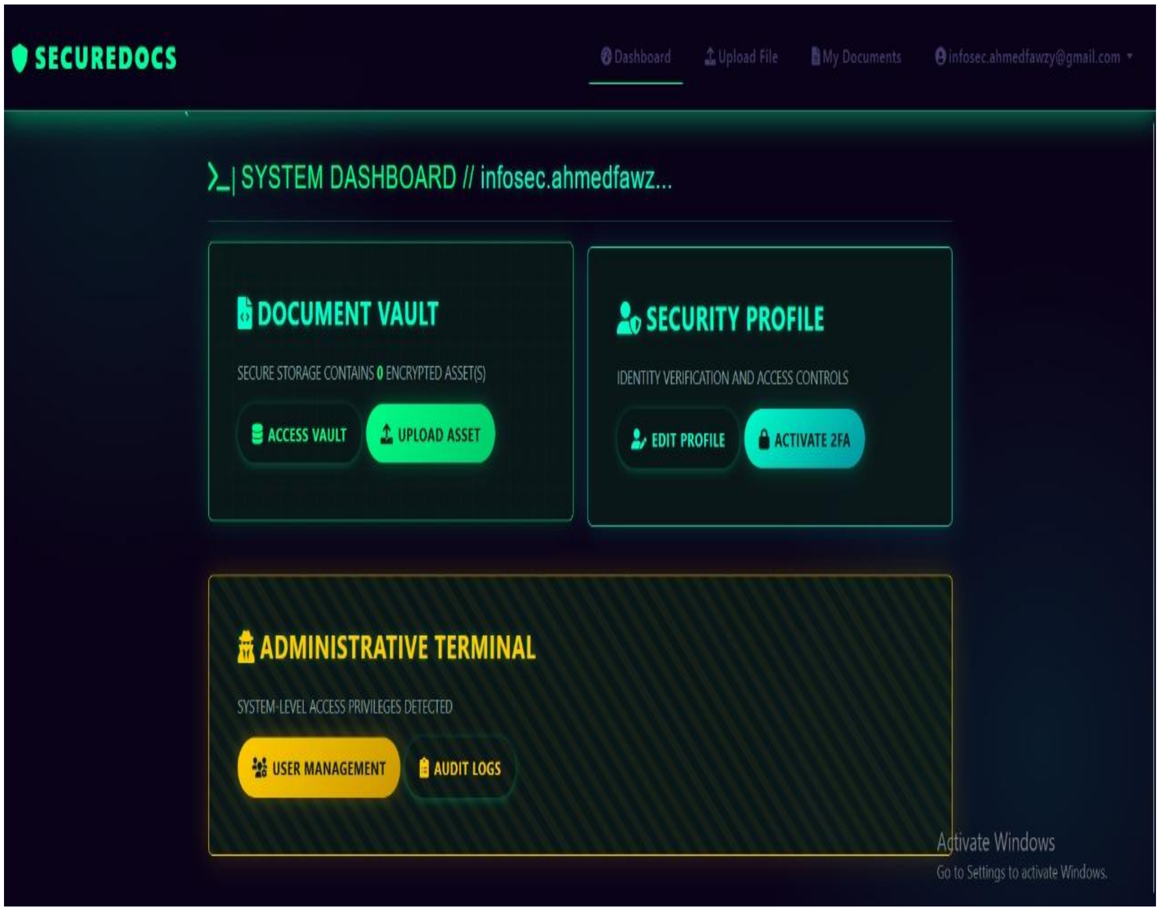
Scan this QR code using an authenticator app like **Google Authenticator** or **Authy**.

After scanning, enter the 6-digit code generated by your app to verify and activate 2FA.

Enter 6-digit code

✓ Must be a 6-digit code from your app.

-Dashboard



>_| SYSTEM DASHBOARD // infosec.ahmedfawz...

DOCUMENT VAULT

SECURE STORAGE CONTAINS 0 ENCRYPTED ASSET(S)

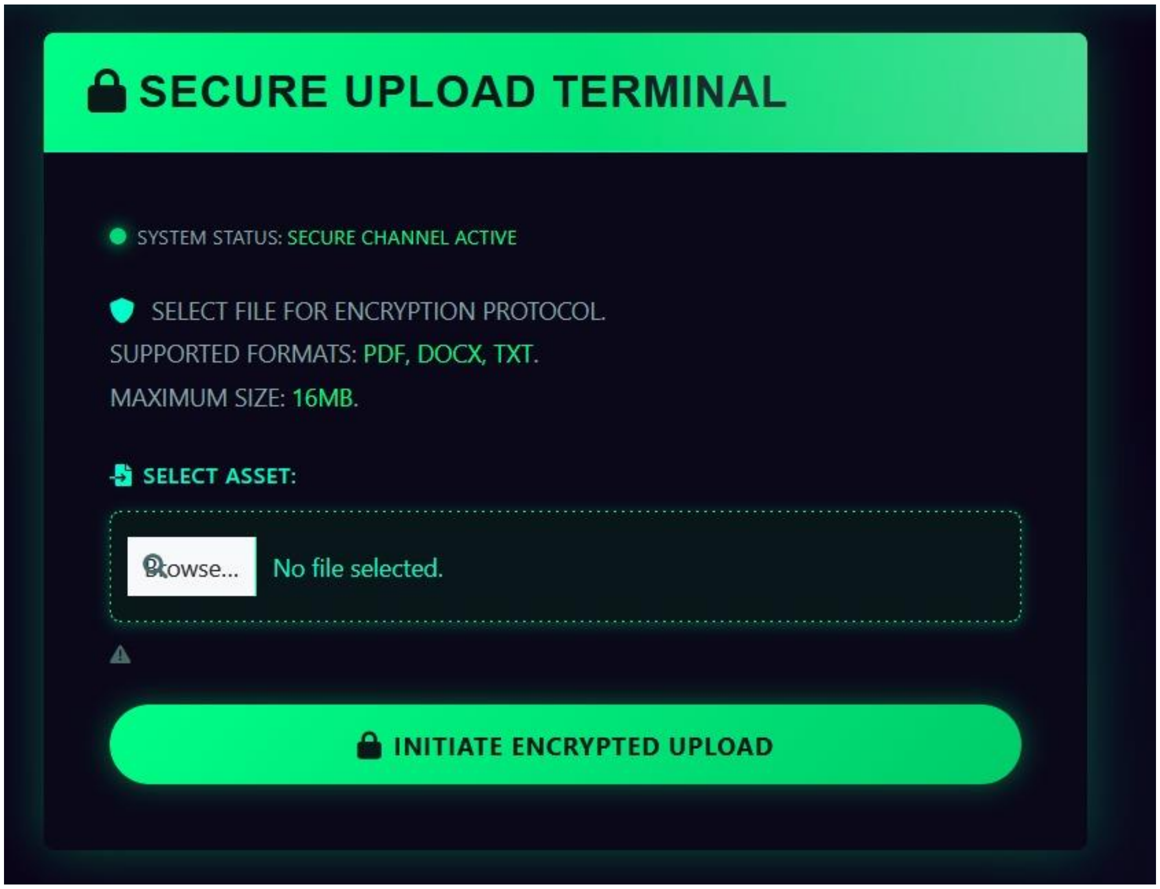
SECURITY PROFILE

IDENTITY VERIFICATION AND ACCESS CONTROLS

ADMINISTRATIVE TERMINAL

SYSTEM-LEVEL ACCESS PRIVILEGES DETECTED

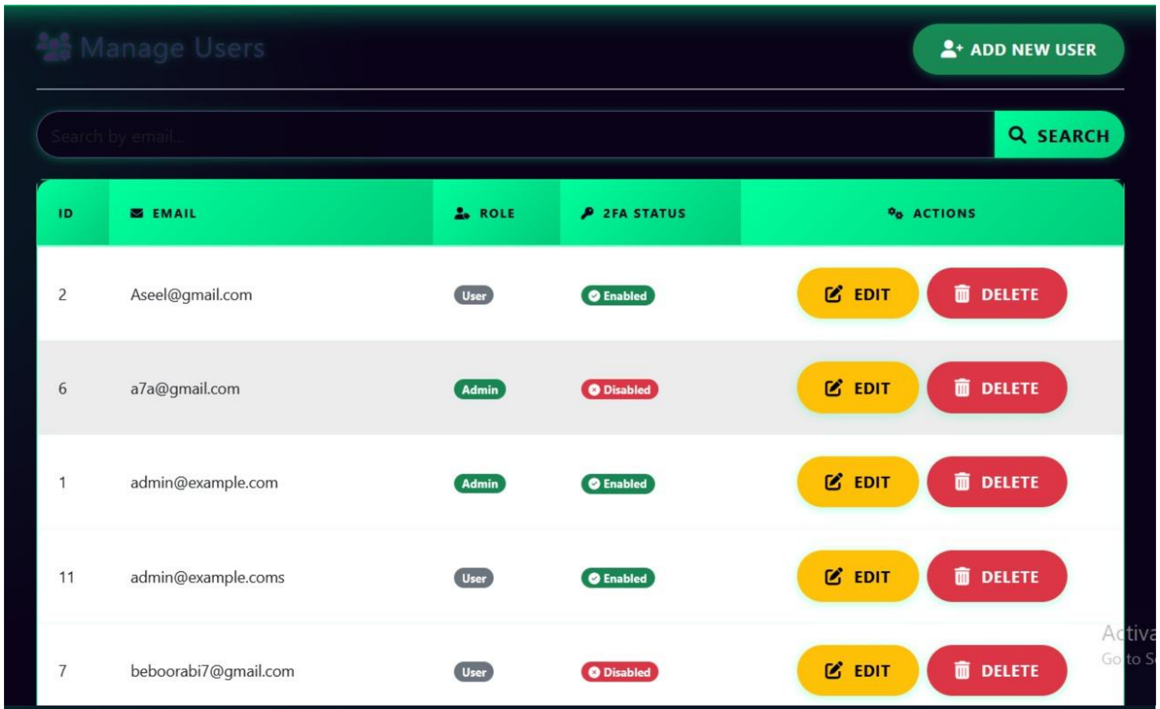
-Upload Terminal


SECURE UPLOAD TERMINAL

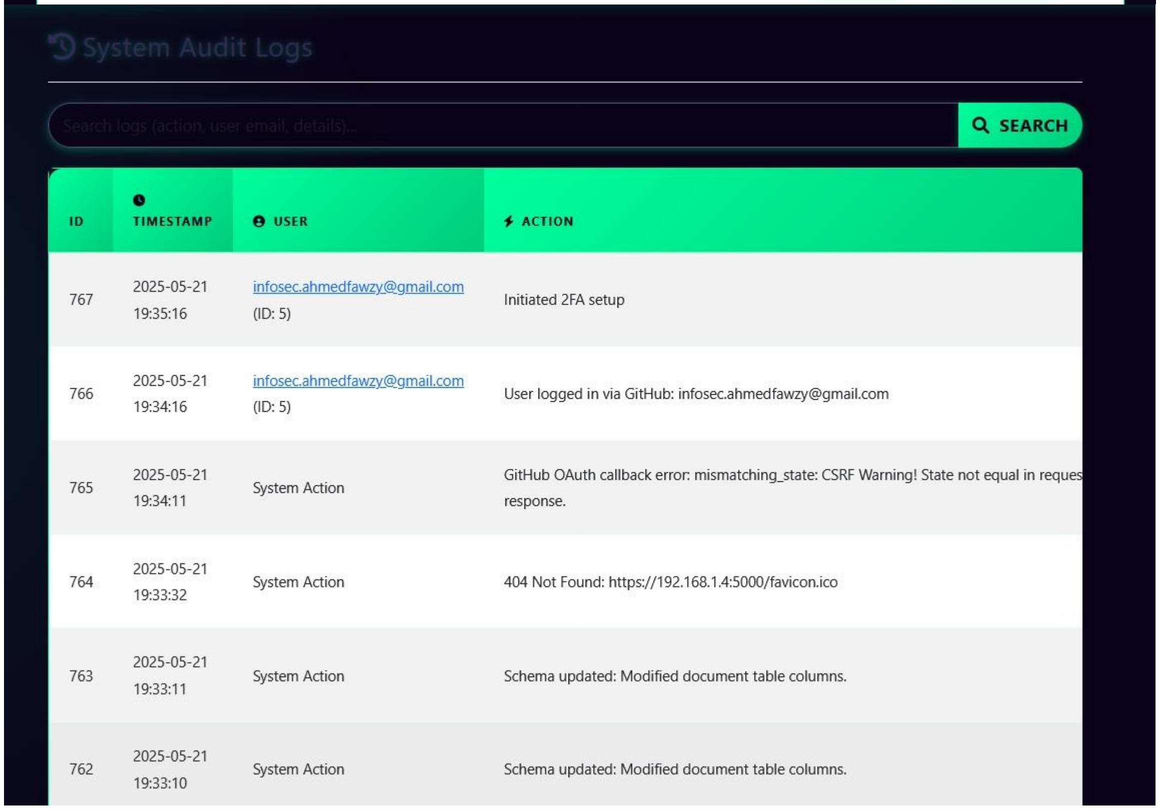
SYSTEM STATUS: SECURE CHANNEL ACTIVE

SELECT FILE FOR ENCRYPTION PROTOCOL.
SUPPORTED FORMATS: PDF, DOCX, TXT.
MAXIMUM SIZE: 16MB.

Admin User Manegemnt & Logs:

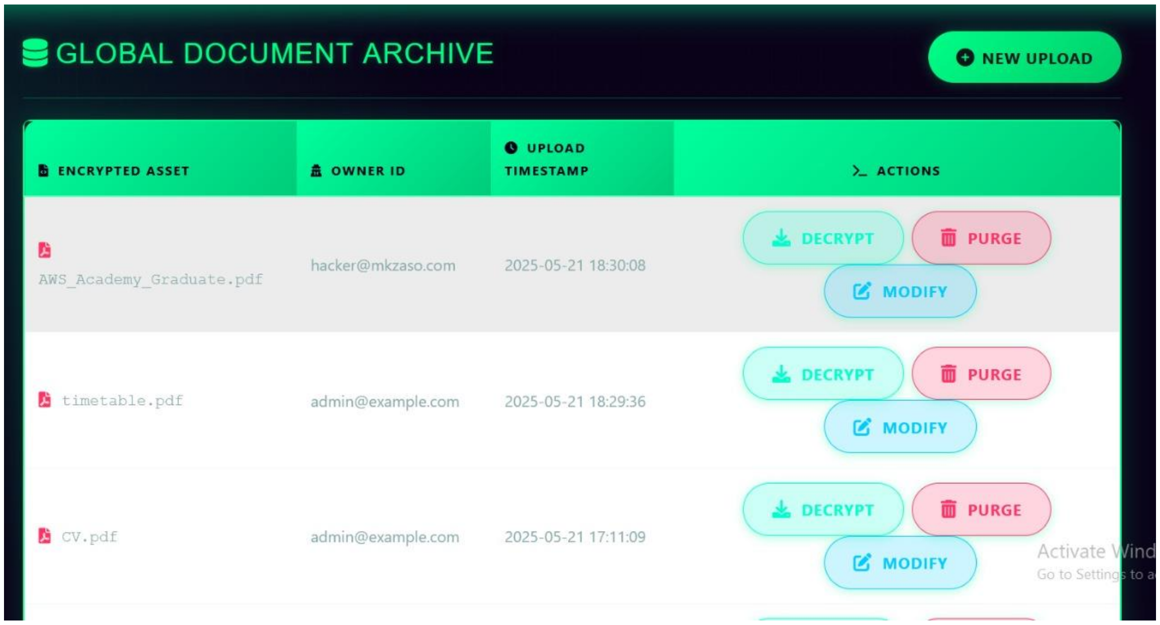


ID	EMAIL	ROLE	2FA STATUS	ACTIONS
2	Aseel@gmail.com	User	Enabled	EDIT DELETE
6	a7a@gmail.com	Admin	Disabled	EDIT DELETE
1	admin@example.com	Admin	Enabled	EDIT DELETE
11	admin@example.coms	User	Enabled	EDIT DELETE
7	beboorabi7@gmail.com	User	Disabled	EDIT DELETE



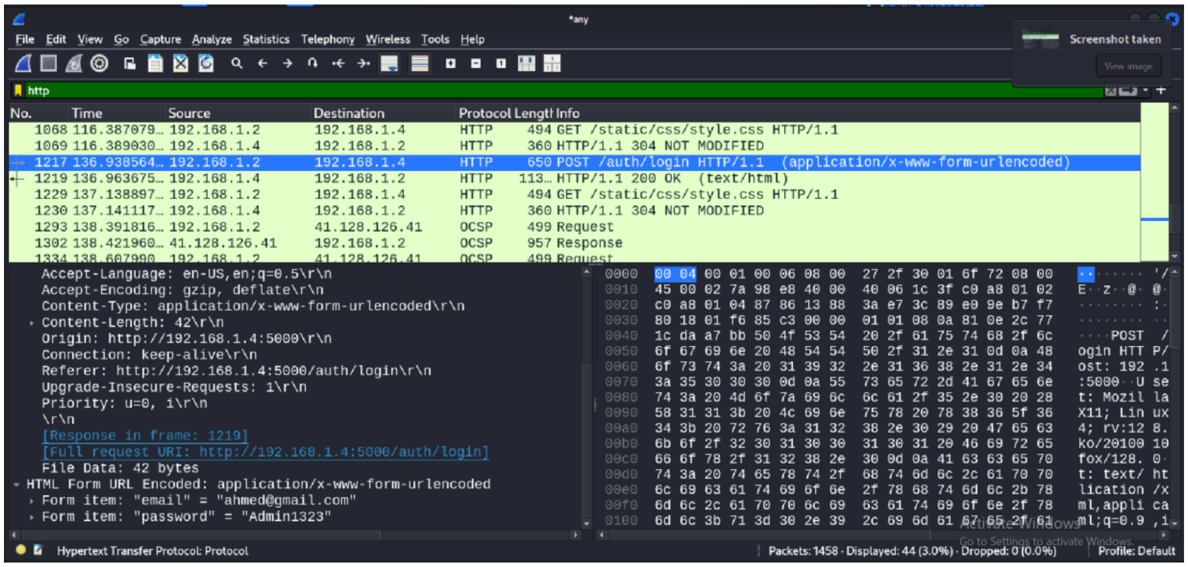
ID	TIMESTAMP	USER	ACTION
767	2025-05-21 19:35:16	infosec.ahmedfawzy@gmail.com (ID: 5)	Initiated 2FA setup
766	2025-05-21 19:34:16	infosec.ahmedfawzy@gmail.com (ID: 5)	User logged in via GitHub: infosec.ahmedfawzy@gmail.com
765	2025-05-21 19:34:11	System Action	GitHub OAuth callback error: mismatching_state: CSRF Warning! State not equal in reques response.
764	2025-05-21 19:33:32	System Action	404 Not Found: https://192.168.1.4:5000/favicon.ico
763	2025-05-21 19:33:11	System Action	Schema updated: Modified document table columns.
762	2025-05-21 19:33:10	System Action	Schema updated: Modified document table columns.

-Download



ENCRYPTED ASSET	OWNER ID	UPLOAD TIMESTAMP	ACTIONS
AWS_Academy_Graduate.pdf	hacker@mkzaso.com	2025-05-21 18:30:08	DECRYPT PURGE MODIFY
timetable.pdf	admin@example.com	2025-05-21 18:29:36	DECRYPT PURGE MODIFY
CV.pdf	admin@example.com	2025-05-21 17:11:09	DECRYPT PURGE MODIFY

analyzing HTTP traffic using wireshark



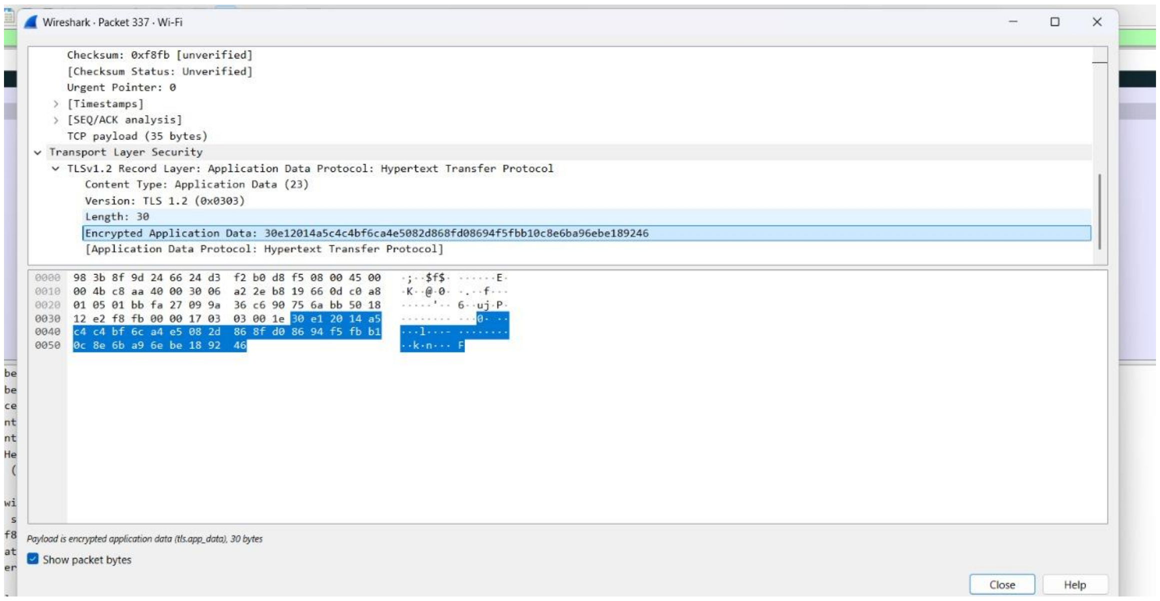
*any
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Screenshot taken
View image
http

```
No.  Time          Source          Destination     Protocol Length Info
1068 116.387079… 192.168.1.2     192.168.1.4     HTTP     494 GET /static/css/style.css HTTP/1.1
1069 116.389030… 192.168.1.4     192.168.1.2     HTTP     360 HTTP/1.1 304 NOT MODIFIED
1217 136.938564… 192.168.1.2     192.168.1.4     HTTP     650 POST /auth/login HTTP/1.1  (application/x-www-form-urlencoded)
1219 136.963675… 192.168.1.4     192.168.1.2     HTTP     113… HTTP/1.1 200 OK  (text/html)
1229 137.138897… 192.168.1.2     192.168.1.4     HTTP     494 GET /static/css/style.css HTTP/1.1
1230 137.141117… 192.168.1.4     192.168.1.2     HTTP     360 HTTP/1.1 304 NOT MODIFIED
1293 138.391816… 192.168.1.2     41.128.126.41   OCSP     499 Request
1302 138.421960… 41.128.126.41   192.168.1.2     OCSP     957 Response
1334 138.607990  192.168.1.2     41.128.126.41   OCSP     499 Request
```

```
    Accept-Language: en-US,en;q=0.5\r\n
    Accept-Encoding: gzip, deflate\r\n
    Content-Type: application/x-www-form-urlencoded\r\n
  › Content-Length: 42\r\n
    Origin: http://192.168.1.4:5000\r\n
    Connection: keep-alive\r\n
    Referer: http://192.168.1.4:5000/auth/login\r\n
    Upgrade-Insecure-Requests: 1\r\n
    Priority: u=0, i\r\n
    \r\n
    [Response in frame: 1219]
    [Full request URI: http://192.168.1.4:5000/auth/login]
    File Data: 42 bytes
˅ HTML Form URL Encoded: application/x-www-form-urlencoded
  › Form item: "email" = "ahmed@gmail.com"
  › Form item: "password" = "Admin1323"
```

```
0000  00 04 00 01 00 06 08 00  27 2f 30 01 6f 72 08 00   ········ '/
0010  45 00 02 7a 98 e8 40 00  40 06 1c 3f c0 a8 01 02   E··z··@· @·
0020  c0 a8 01 04 87 86 13 88  3a e7 3c 89 e0 9e b7 f7   ········ :·
0030  80 18 01 f6 85 c3 00 00  01 01 08 0a 81 0e 2c 77   ········ ··
0040  1c da a7 bb 50 4f 53 54  20 2f 61 75 74 68 2f 6c   ····POST  /
0050  6f 67 69 6e 20 48 54 54  50 2f 31 2e 31 0d 0a 48   ogin HTT P/
0060  6f 73 74 3a 20 31 39 32  2e 31 36 38 2e 31 2e 34   ost: 192 .1
0070  3a 35 30 30 30 0d 0a 55  73 65 72 2d 41 67 65 6e   :5000··U se
0080  74 3a 20 4d 6f 7a 69 6c  6c 61 2f 35 2e 30 20 28   t: Mozil la
0090  58 31 31 3b 20 4c 69 6e  75 78 20 78 38 36 5f 36   X11; Lin ux
00a0  34 3b 20 72 76 3a 31 32  38 2e 30 29 20 47 65 63   4; rv:12 8.
00b0  6b 6f 2f 32 30 31 30 30  31 30 31 20 46 69 72 65   ko/20100 10
00c0  66 6f 78 2f 31 32 38 2e  30 0d 0a 41 63 63 65 70   fox/128. 0·
00d0  74 3a 20 74 65 78 74 2f  68 74 6d 6c 2c 61 70 70   t: text/ ht
00e0  6c 69 63 61 74 69 6f 6e  2f 78 68 74 6d 6c 2b 78   lication /x
00f0  6d 6c 2c 61 70 70 6c 69  63 61 74 69 6f 6e 2f 78   ml,appli ca
0100  6d 6c 3b 71 3d 30 2e 39  2c 69 6d 61 67 65 2f 61   ml;q=0.9 ,i
```

Hypertext Transfer Protocol: Protocol
Packets: 1458 · Displayed: 44 (3.0%) · Dropped: 0 (0.0%)
Profile: Default



```
    Checksum: 0xf8fb [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
  > [Timestamps]
  > [SEQ/ACK analysis]
    TCP payload (35 bytes)
v Transport Layer Security
  v TLSv1.2 Record Layer: Application Data Protocol: Hypertext Transfer Protocol
      Content Type: Application Data (23)
      Version: TLS 1.2 (0x0303)
      Length: 30
      Encrypted Application Data: 30e12014a5c4c4bf6ca4e5082d868fd08694f5fbb10c8e6ba96ebe189246
      [Application Data Protocol: Hypertext Transfer Protocol]

0000  98 3b 8f 9d 24 66 24 d3  f2 b0 d8 f5 08 00 45 00   ·;··$f$· ······E·
0010  00 4b c8 aa 40 00 30 06  a2 2e b8 19 66 0d c0 a8   ·K··@·0· ·.··f···
0020  01 05 01 bb fa 27 09 9a  36 c6 90 75 6a bb 50 18   ·····'·· 6··uj·P·
0030  12 e2 f8 fb 00 00 17 03  03 00 1e 30 e1 20 14 a5   ········ ···0· ··
0040  c4 c4 bf 6c a4 e5 08 2d  86 8f d0 86 94 f5 fb b1   ···l···- ········
0050  0c 8e 6b a9 6e be 18 92  46                        ··k·n··· F
```

Payload is encrypted application data (tls.app_data), 30 bytes

HTTPS (HyperText Transfer Protocol Secure) is the secure version of HTTP, which adds encryption using **TLS (Transport Layer Security)** to protect data exchanged between a client and a web server.

1. Encryption (Confidentiality)

- All data sent between the client and the server is **encrypted**, so even if an attacker intercepts the traffic ,they will only see **random, unreadable data**.

2. Authentication (Trust)

- During the TLS handshake, the server presents an **SSL/TLS certificate** to the client.
- This certificate is issued by a **trusted Certificate Authority (CA)** and proves that the server is **legitimate**.

3. Data Integrity

- HTTPS ensures that the data cannot be **modified or tampered with** during transmission.
- Any change in the data by a third party will be **detected** by the TLS protocol, and the connection will be terminated.