Secure File Upload & Encryption Web Portal

Cybersecurity Internship Task 3 : File Protection Using AES
By: Priya Kumari

INTRODUCTION

In this task, I developed a simple yet secure web-based file upload/download portal that encrypts files using AES (Advanced Encryption Standard). This helps ensure that files remain protected at rest and in transit, addressing common file-handling vulnerabilities.

The portal was built using Python Flask on Kali Linux, with PyCryptodome for AES encryption.

TOOLS & TECHNOLOGIES USED

- ✓ Flask – to create the web interface
- ✓ PyCryptodome – for AES encryption & decryption
- ✓ HTML/CSS – for building a simple frontend
- ✓ Kali Linux – development and testing environment
- ✓ Python Virtual Environment – to manage packages securely

FUNCTIONALITY IMPLEMENTED

1. Upload & Encrypt

- Users can upload any file (e.g., .txt, .pdf, .jpg)
- Upon upload, the file is stored in the encrypted_uploads/ folder.
- The backend reads the file and encrypts it using AES in EAX mode.
- Encrypted file is downloaded instantly with .enc extension.

2. AES Encryption Logic

- A 16-byte secret key was used.
- Each file was encrypted using a unique nonce and tag.

- Encryption flow:
  plaintext → AES encryption → ciphertext + nonce + tag → saved as .enc

3. Frontend (Web UI)

- Built using basic HTML forms.
- Page has two sections:
  - Upload to Encrypt
  - Download & Decrypt Instructions

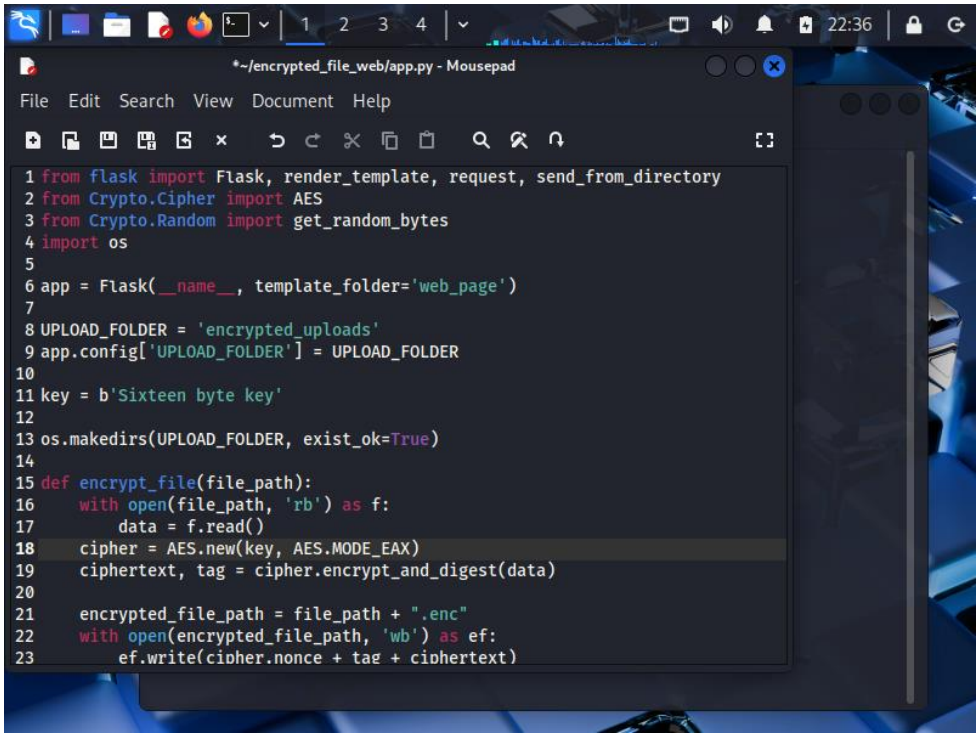
4. Directory Structure

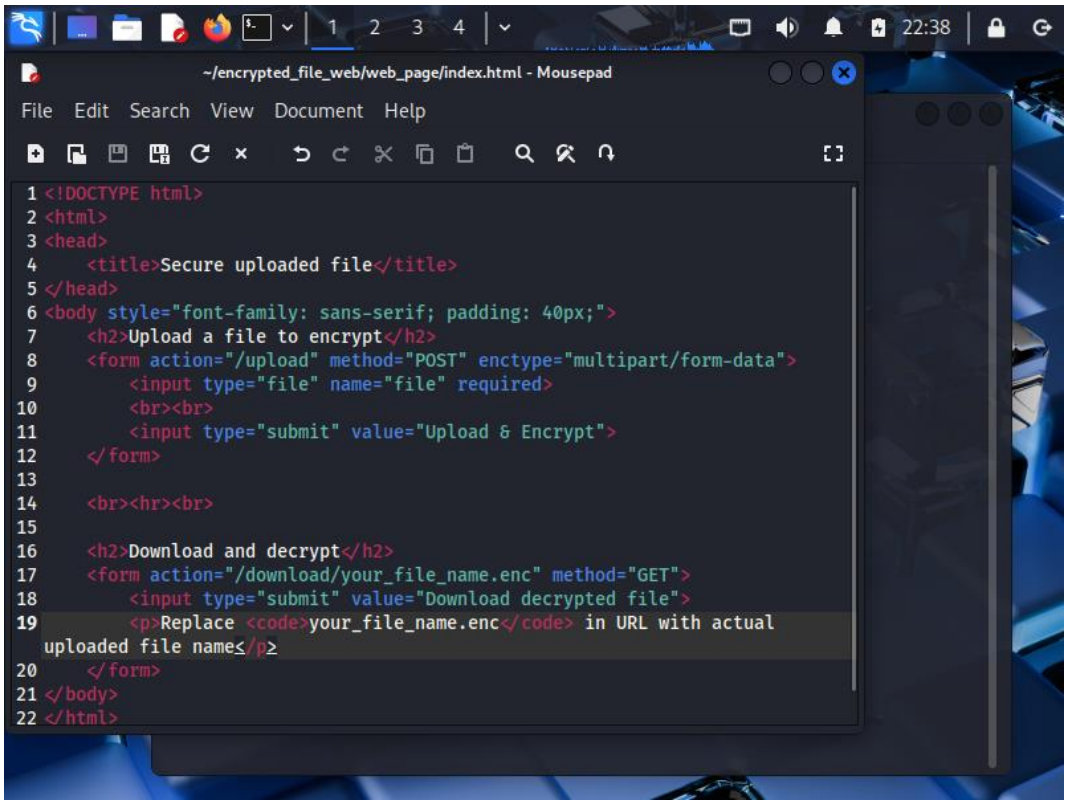
```
encrypted_file_web/
├── encrypted_uploads/      # Encrypted files stored here
├── web_page/               # HTML templates
│   └── index.html
├── app.py                  # Main backend script
└── secureenv/              # Python virtual environment
```

CHALLENGES FACED

- Faced ModuleNotFoundError: No module named 'Crypto' ➤ Resolved using Python virtual environment
- Flask wouldn't run initially because of incorrect template folder ➤ Fixed using: template_folder='web_page'
- Upload form gave 404 error ➤ Solved by aligning form action="/" to the correct route in Flask

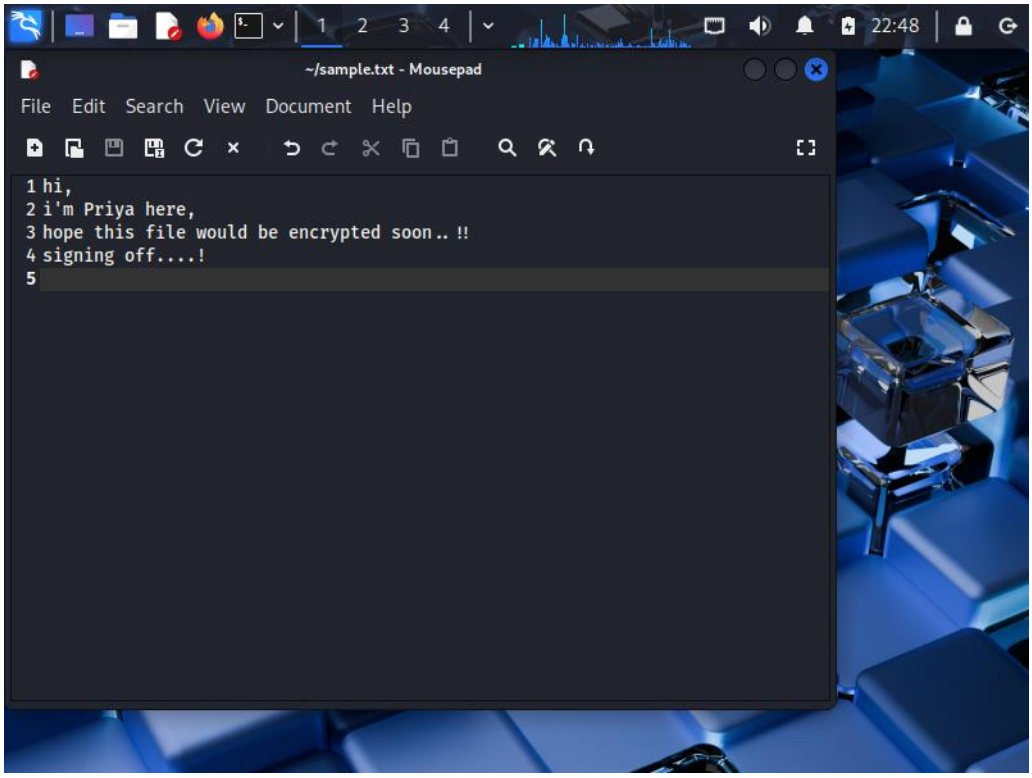
SCREENSHOTS

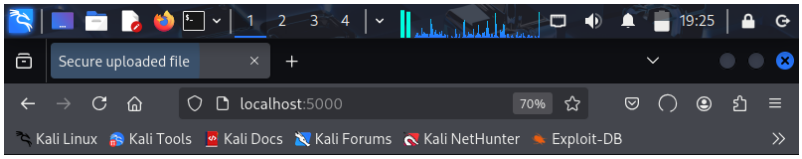

*~/encrypted_file_web/app.py - Mousepad
File Edit Search View Document Help
from flask import Flask, render_template, request, send_from_directory
from Crypto.Cipher import AES
from Crypto.Random import get_random_bytes
import os
app = Flask(__name__, template_folder='web_page')
UPLOAD_FOLDER = 'encrypted_uploads'
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
key = b'Sixteen byte key'
os.makedirs(UPLOAD_FOLDER, exist_ok=True)
def encrypt_file(file_path):
with open(file_path, 'rb') as f:
data = f.read()
cipher = AES.new(key, AES.MODE_EAX)
ciphertext, tag = cipher.encrypt_and_digest(data)
encrypted_file_path = file_path + ".enc"
with open(encrypted_file_path, 'wb') as ef:


~/encrypted_file_web/web_page/index.html - Mousepad
File Edit Search View Document Help
<!DOCTYPE html>
<html>
<head>
<title>Secure uploaded file</title>
</head>
<body style="font-family: sans-serif; padding: 40px;">
<h2>Upload a file to encrypt</h2>
<form action="/upload" method="POST" enctype="multipart/form-data">
<input type="file" name="file" required>

<input type="submit" value="Upload & Encrypt">
</form>

<hr>
<h2>Download and decrypt</h2>
<form action="/download/your_file_name.enc" method="GET">
<input type="submit" value="Download decrypted file">
Replace <code>your_file_name.enc</code> in URL with actual uploaded file name
</form>
</body>
</html>


1 2 3 4
22:48
~/sample.txt - Mousepad
File Edit Search View Document Help
1 hi,
2 i'm Priya here,
3 hope this file would be encrypted soon.. !!
4 signing off....!
5



Upload a file to encrypt

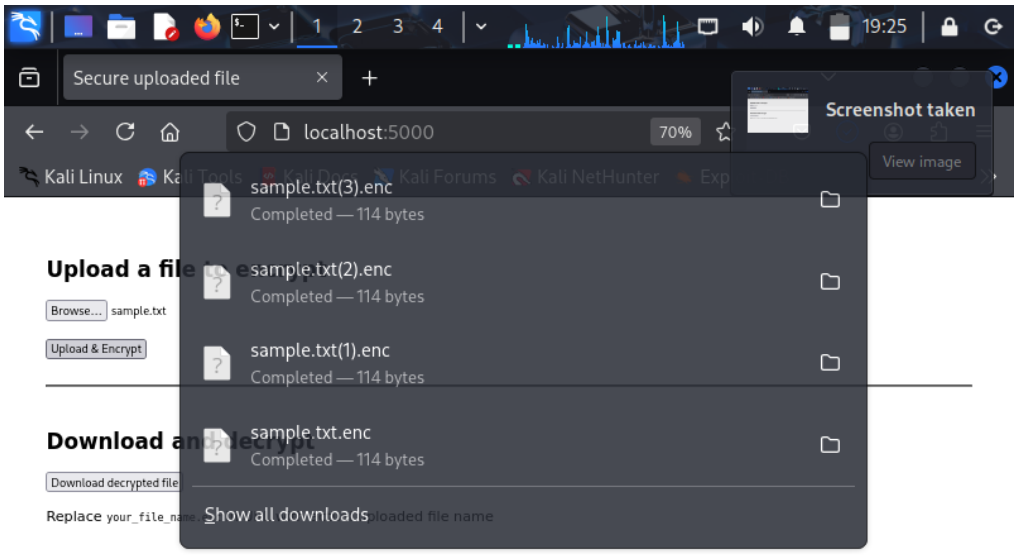
Browse... sample.txt

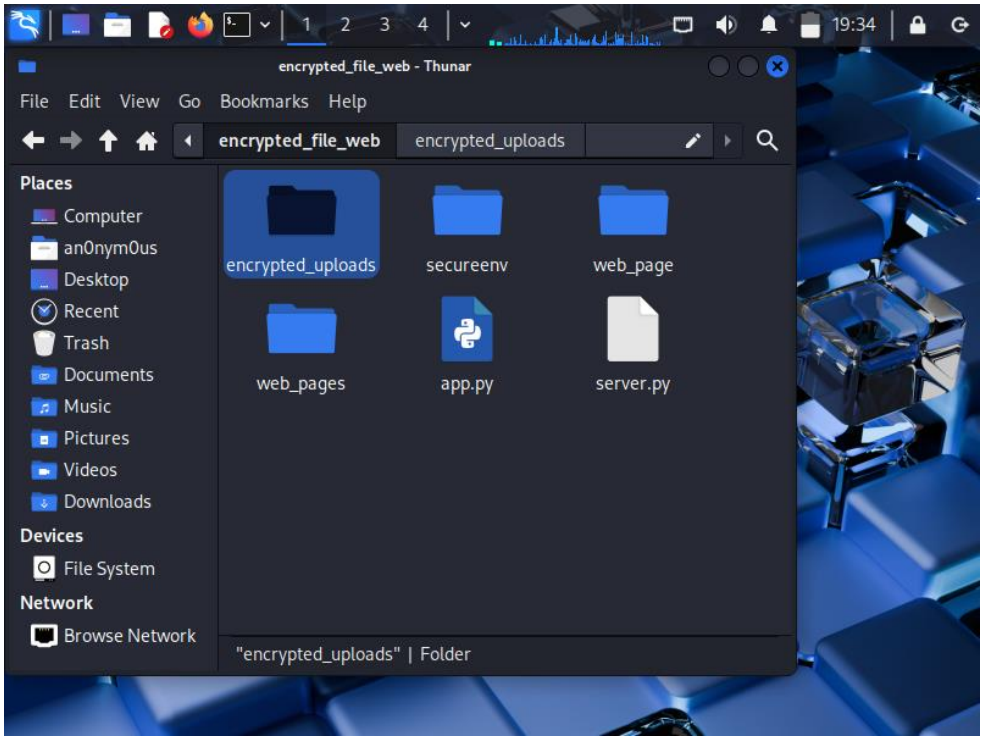
Upload & Encrypt

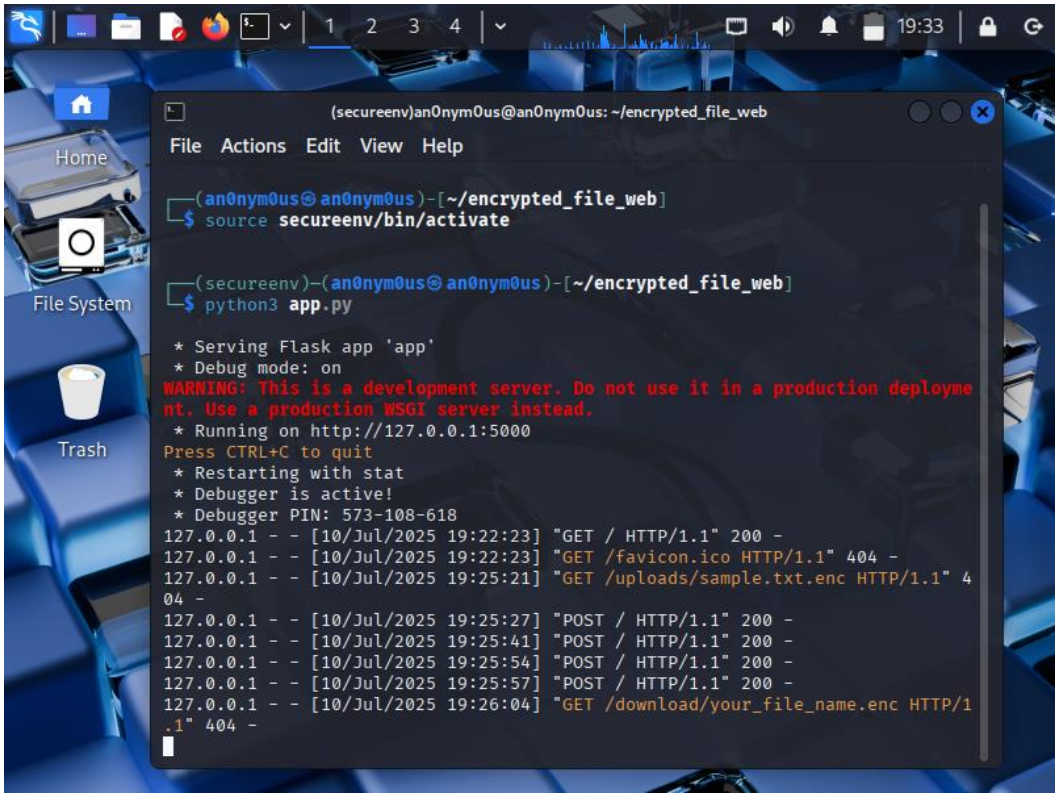
Download and decrypt

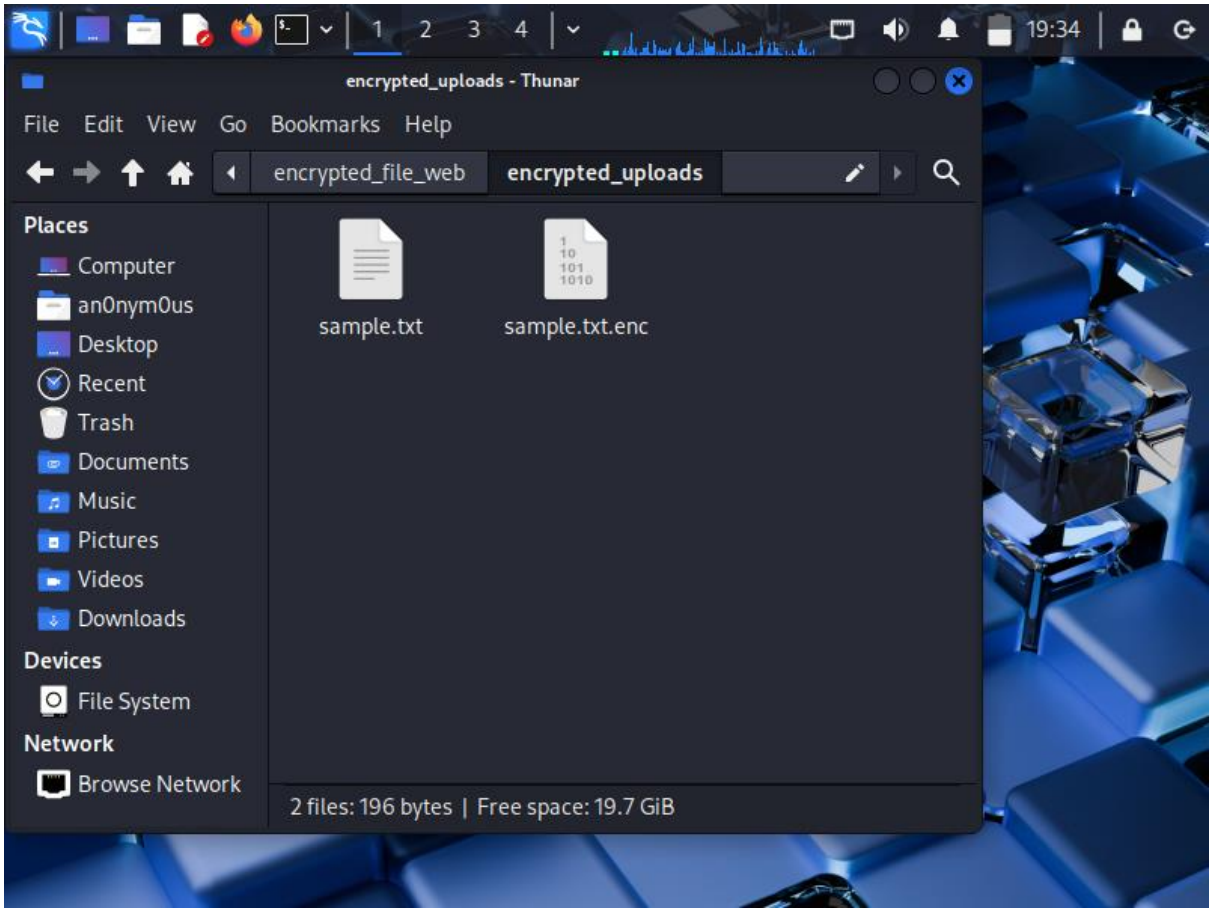
Download decrypted file

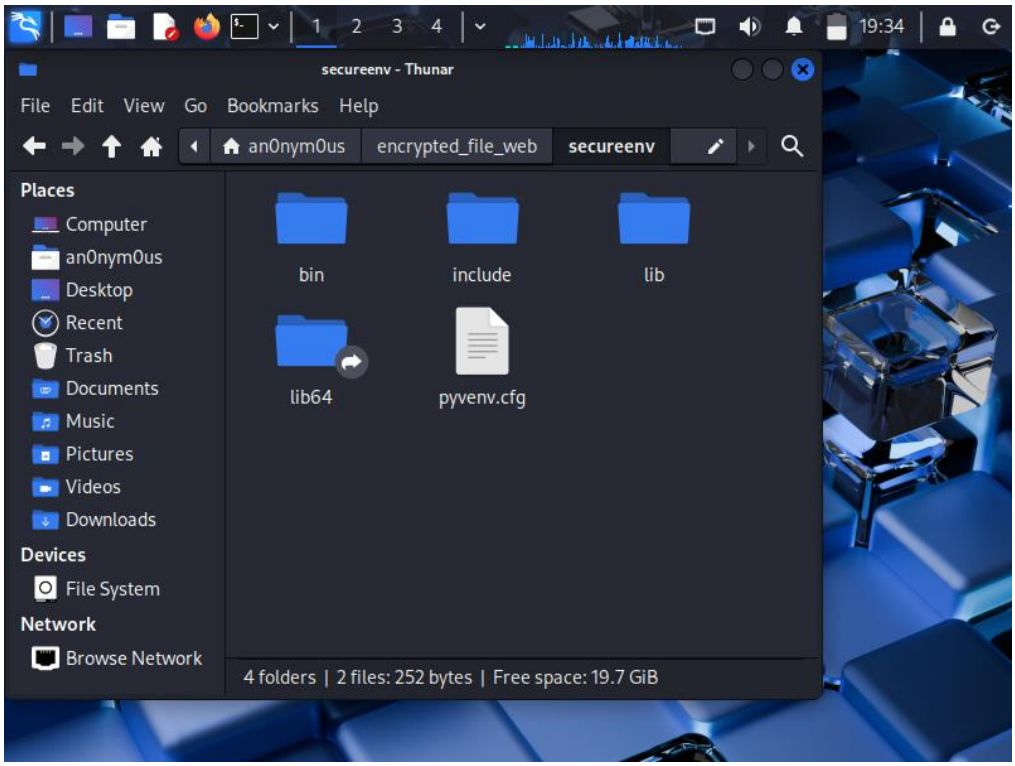
Replace `your_file_name.enc` in URL with actual uploaded file name




1 2 3 4
19:34
encrypted_file_web - Thunar
File Edit View Go Bookmarks Help
encrypted_file_web
encrypted_uploads
Places
Computer
an0nym0us
Desktop
Recent
Trash
Documents
Music
Pictures
Videos
Downloads
Devices
File System
Network
Browse Network
encrypted_uploads
secureenv
web_page
web_pages
app.py
server.py
"encrypted_uploads" | Folder


1 2 3 4
19:33
Home
File System
Trash
(secureenv)an0nym0us@an0nym0us: ~/encrypted_file_web
File Actions Edit View Help
(an0nym0us㉿an0nym0us)-[~/encrypted_file_web]
$ source secureenv/bin/activate
(secureenv)-(an0nym0us㉿an0nym0us)-[~/encrypted_file_web]
$ python3 app.py
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 573-108-618
127.0.0.1 - - [10/Jul/2025 19:22:23] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [10/Jul/2025 19:22:23] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [10/Jul/2025 19:25:21] "GET /uploads/sample.txt.enc HTTP/1.1" 404 -
127.0.0.1 - - [10/Jul/2025 19:25:27] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [10/Jul/2025 19:25:41] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [10/Jul/2025 19:25:54] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [10/Jul/2025 19:25:57] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [10/Jul/2025 19:26:04] "GET /download/your_file_name.enc HTTP/1.1" 404 -


1 2 3 4
19:34
encrypted_uploads - Thunar
File Edit View Go Bookmarks Help
encrypted_file_web
encrypted_uploads
Places
Computer
an0nym0us
Desktop
Recent
Trash
Documents
Music
Pictures
Videos
Downloads
Devices
File System
Network
Browse Network
sample.txt
sample.txt.enc
2 files: 196 bytes | Free space: 19.7 GiB


1 2 3 4
19:34
secureenv - Thunar
File Edit View Go Bookmarks Help
an0nym0us
encrypted_file_web
secureenv
Places
Computer
an0nym0us
Desktop
Recent
Trash
Documents
Music
Pictures
Videos
Downloads
Devices
File System
Network
Browse Network
bin
include
lib
lib64
pyvenv.cfg
4 folders | 2 files: 252 bytes | Free space: 19.7 GiB

KEY TAKEAWAYS

- Understood AES encryption implementation in real-world apps
- Learned to build a simple Flask-based web app
- Understood how files are encrypted/decrypted securely
- Improved my confidence in debugging & solving Python errors on Kali

WHAT CAN BE IMPROVED

- Add decryption feature to view decrypted content
- Use dynamic keys per user with secure storage
- Add user authentication for access control
- Improve UI with Bootstrap for responsiveness