FUTURE_CS_03

Intern Name: Meghan Diwate
Date: 2nd October 2025

Abstract

This project demonstrates the development of a secure file sharing system with a hacker-themed interface. The system allows users to upload, encrypt, decrypt, and download files while maintaining confidentiality using AES-based encryption (Fernet). The project emphasizes ethical hacking principles, secure file handling, and real-time logging to simulate monitoring activity.

Introduction

With the increasing need for secure data transmission, organizations must ensure files are encrypted and accessible only to authorized users. This project simulates a secure ethical hacking environment, demonstrating how files can be safely handled, encrypted, and decrypted. It provides a terminal-style interface to mimic a hacker-friendly console while maintaining data security.

Objectives

- Develop a web application to upload and encrypt files for secure storage.
- Implement AES encryption using Python's Fernet to ensure confidentiality.
- Provide a terminal-style logging interface for monitoring file actions.
- Simulate a hacker-friendly ethical environment for cybersecurity learning.

Tools & Technologies Used

Component	Technology/Tool
Backend	Python Flask
Encryption	Cryptography (Fernet)
Frontend	HTML, CSS, JavaScript
Version Control	Git, GitHub
Environment	Windows OS, Virtual Environment (venv)

System Design & Architecture

Folder Structure:

```
Ethical_Hacking_File_Sharing/
├── app.py
├── secret.key
```

```
├── uploads/
├── templates/
│   └── index.html
└── static/
    └── style.css
```

System Flow: 1. File Upload → 2. Encryption → 3. Decryption → 4. Download → 5. Logging

Route Summary: | Route | Method | Functionality | |——-|——|————| | / | GET | Load home page with file list | | /encrypt | POST | Upload and encrypt file | | /decrypt | POST | Upload and decrypt file | | /download/ | GET | Download selected file |

Implementation

Frontend Design

index.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Hacker File Sharing System</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='style.css')
}}">
</head>
<body>
    <div class="terminal-header">[ Secure Ethical Hacking File System v1.0
]</div>

    <h1>Upload & Encrypt File</h1>
    <form action="/encrypt" method="post" enctype="multipart/form-data">
        <input type="file" name="file" required>
        <input type="submit" value="Encrypt & Upload">
    </form>

    <h1>Decrypt File</h1>
    <form action="/decrypt" method="post" enctype="multipart/form-data">
        <input type="file" name="file" required>
        <input type="submit" value="Decrypt & Upload">
    </form>

    <h1>Download Files</h1>
    <ul>
        {% for file in files %}
            <li><a href="/download/{{ file }}">{{ file }}</a></li>
        {% endfor %}
    </ul>
```

```
    <h2>System Logs</h2>
    <div id="log-panel" class="console">
        <p>[INFO] System initialized...</p>
        <p>[INFO] Waiting for file upload...</p>
    </div>

    <script>
        function addLog(message) {
            const logPanel = document.getElementById("log-panel");
            const p = document.createElement("p");
            p.textContent = `[${new Date().toLocaleTimeString()}]
${message}`;
            logPanel.appendChild(p);
            logPanel.scrollTop = logPanel.scrollHeight;
        }

        setInterval(() => {
            const messages = [
                "Monitoring upload directory...",
                "Scanning for suspicious activity...",
                "No threats detected.",
                "New connection request logged."
            ];
            addLog(messages[Math.floor(Math.random() * messages.length)]);
        }, 5000);
    </script>
</body>
</html>
```

CSS Design

style.css:

```
body {
    font-family: "Times New Roman", serif;
    background-color: #000;
    color: #00FF00;
    padding: 20px;
}
.terminal-header {
    background-color: #111;
    color: #00ff00;
    padding: 10px;
    border: 1px solid #00ff00;
    font-weight: bold;
    margin-bottom: 20px;
    text-align: center;
    box-shadow: 0 0 10px #00ff00;
}
h1, h2 {
```

```
    color: #00FF00;
    text-shadow: 0 0 5px #00FF00;
    margin-top: 20px;
}
form {
    background-color: #111;
    border: 1px solid #00FF00;
    padding: 15px;
    margin: 20px 0;
    box-shadow: 0 0 10px #00ff00;
}
input[type="file"] {
    background-color: #000;
    color: #00FF00;
    border: 1px solid #00FF00;
    padding: 5px;
    margin-right: 10px;
}
input[type="submit"] {
    background-color: #000;
    color: #00FF00;
    border: 1px solid #00FF00;
    padding: 5px 10px;
    cursor: pointer;
}
input[type="submit"]:hover {
    background-color: #00FF00;
    color: #000;
}
a {
    color: #00FF00;
    text-decoration: none;
}
a:hover {
    text-decoration: underline;
}
ul {
    list-style-type: none;
    padding-left: 0;
}
li {
    margin: 5px 0;
}
.console {
    background-color: #000;
    color: #00ff00;
    font-family: "Times New Roman", serif;
    border: 1px solid #00ff00;
    padding: 10px;
    height: 200px;
```

```
    overflow-y: auto;
    margin-top: 20px;
    box-shadow: 0 0 10px #00ff00;
}
.console p {
    margin: 0;
    font-size: 14px;
}
```

Backend Design

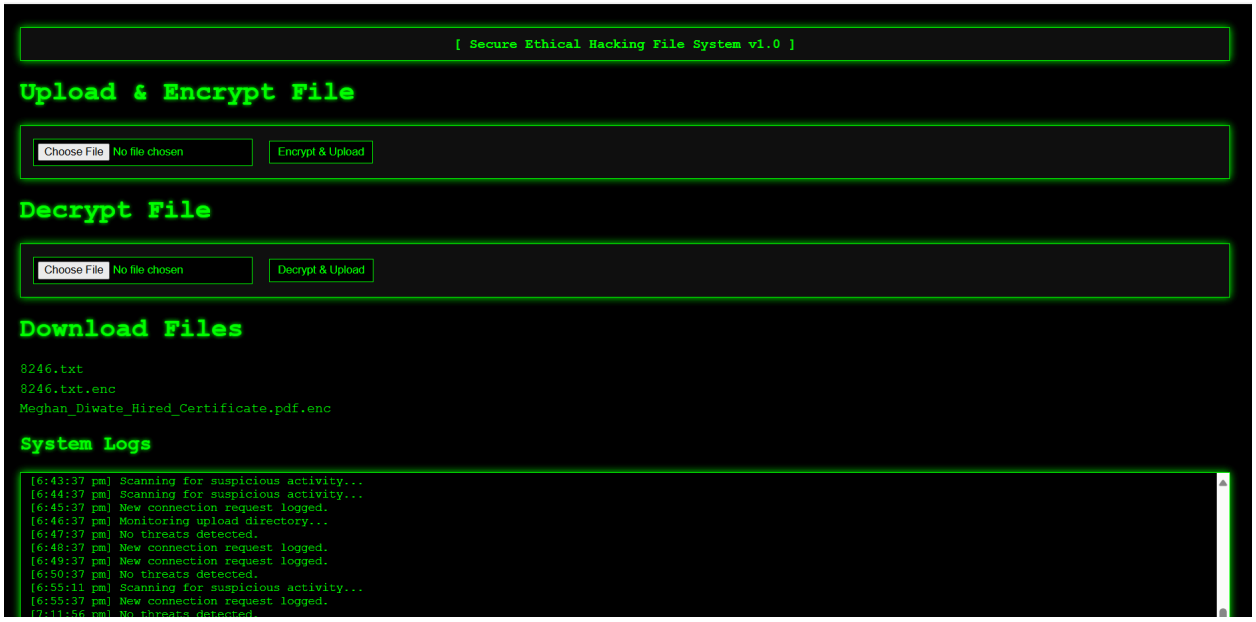
app.py:

```
from flask import Flask, render_template, request, send_from_directory,
redirect, url_for
from cryptography.fernet import Fernet
import os

app = Flask(__name__)
UPLOAD_FOLDER = "uploads"
KEY_FILE = "secret.key"

if not os.path.exists(UPLOAD_FOLDER):
    os.makedirs(UPLOAD_FOLDER)

if not os.path.exists(KEY_FILE):
    with open(KEY_FILE, "wb") as key_file:
        key_file.write(Fernet.generate_key())

with open(KEY_FILE, "rb") as key_file:
    key = key_file.read()

fernet = Fernet(key)

@app.route("/")
def index():
    files = os.listdir(UPLOAD_FOLDER)
    return render_template("index.html", files=files)

@app.route("/encrypt", methods=["POST"])
def encrypt_file():
    file = request.files["file"]
    filepath = os.path.join(UPLOAD_FOLDER, file.filename)
    file.save(filepath)

    with open(filepath, "rb") as f:
        data = f.read()

    encrypted_data = fernet.encrypt(data)
```

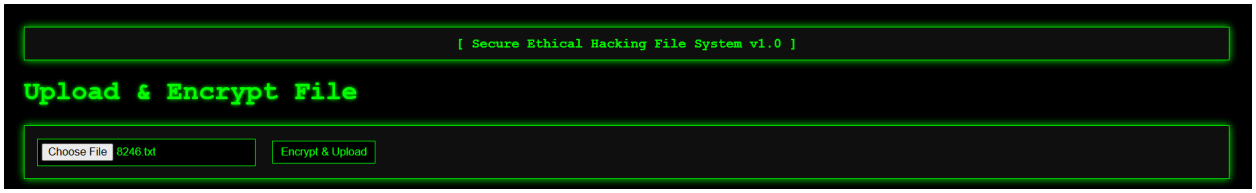
```
    encrypted_path = filepath + ".enc"

    with open(encrypted_path, "wb") as f:
        f.write(encrypted_data)

    os.remove(filepath)
    return redirect(url_for("index"))

@app.route("/decrypt", methods=["POST"])
def decrypt_file():
    file = request.files["file"]
    filepath = os.path.join(UPLOAD_FOLDER, file.filename)
    file.save(filepath)

    with open(filepath, "rb") as f:
        encrypted_data = f.read()

    decrypted_data = fernet.decrypt(encrypted_data)
    decrypted_path = filepath.replace(".enc", "")

    with open(decrypted_path, "wb") as f:
        f.write(decrypted_data)

    return redirect(url_for("index"))

@app.route("/download/<filename>")
def download_file(filename):
    return send_from_directory(UPLOAD_FOLDER, filename, as_attachment=True)

if __name__ == "__main__":
    app.run(debug=True)
```

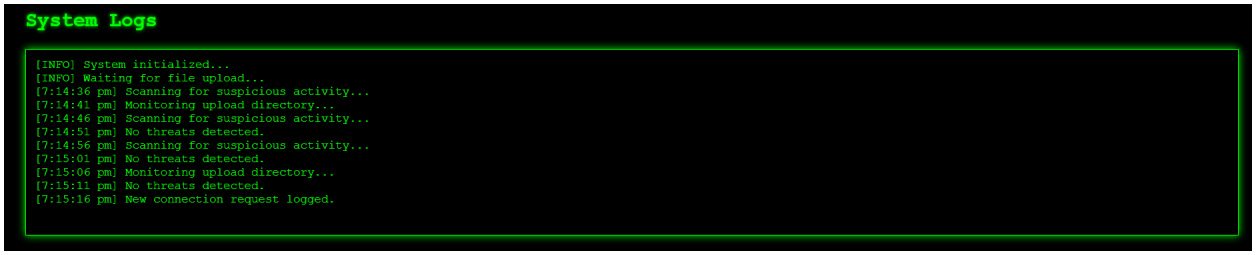
Screenshots

- Home Page



- File Upload & Encryption

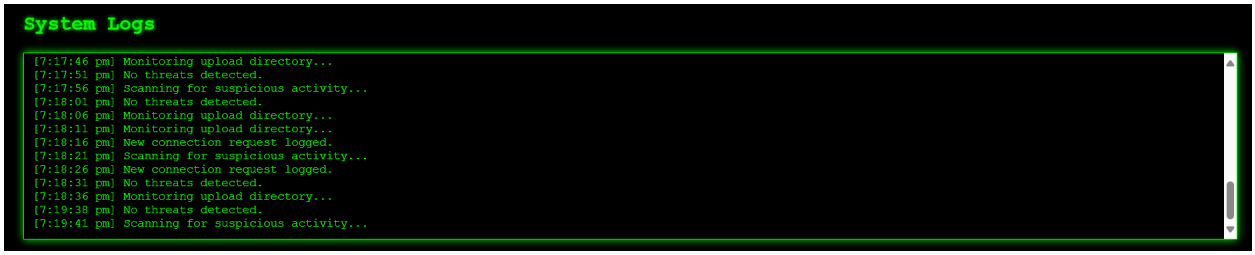




- File Decryption

- Download Section

- Console Logs



Challenges Faced

- PowerShell execution policy issues on Windows.
- Ensuring correct file encryption/decryption.
- Dynamic frontend-backend integration for file listing.

Learning Outcomes

- Implemented AES encryption and decryption in Python.
- Developed full-stack web application using Flask.
- Learned secure file handling practices.
- Simulated ethical hacking and real-time monitoring.

Conclusion

The Secure Ethical Hacking File Sharing System successfully demonstrates secure file handling, encryption, and logging. All objectives were achieved, emphasizing the importance of data confidentiality in cybersecurity.