ENCRYPTION FILE SHARING SYSTEM WITH INTRUSION ALERTS

I. Executive Summary

Key Findings

This project demonstrates a secure file sharing system using encryption, with integrated intrusion detection alerts. The system enhances data confidentiality and actively monitors unauthorized access attempts.

Security Risk Assessment Results

Risks include unauthorized access, data interception, and insufficient user authentication. However, these risks are mitigated using encryption techniques and active alert mechanisms.

Incident Summary

No live incidents occurred during testing, but simulated unauthorized access attempts were successfully detected and logged with alerts.

Overall Security Posture

The project showcases a robust security posture with strong encryption and intrusion detection capabilities, but lacks real-time scalability and advanced behavioral analytics.

II. Project Scope and Objectives

This project involves developing a secure file sharing system in Python, incorporating AES encryption for data confidentiality and a basic intrusion detection alert mechanism.

Objectives include:
- Encrypting files before transmission.
- Ensuring secure file transfer.
- Detecting and alerting unauthorized access attempts.
- Logging events for audit purposes.

III. Detailed Findings

Network Infrastructure

Since the system is hosted locally and shared via GitHub Pages, network risks were minimal. However, secure transport protocols were considered.

Applications and Services

The Python-based application was reviewed for vulnerabilities like weak encryption, poor exception handling, and lack of session control.

Endpoints

Simulated endpoints were tested for malware and unauthorized access; the system responded effectively to intrusion attempts.

Cloud Security

Though not fully cloud-integrated, files were shared through GitHub Pages with controlled access.

Data Security

AES encryption secures files before sharing. Access is validated before decryption, preventing unauthorized disclosure.

Security Awareness

Users were provided with documentation and training on secure file usage, encryption awareness, and recognizing intrusion alerts.

IV. Risk Assessment

Vulnerability Assessment

Identified risks include:
- Brute-force attacks
- Local file interception
- Insecure temporary file storage

Threat Modeling

Threats include:
- Insider threats
- Malware injection
- Unauthorized file downloads

Risk Prioritization

High-priority risks included unencrypted transmission and lack of access validation. These were mitigated through AES encryption and access checks.

Mitigation Strategies

Implemented AES-256 encryption, input validation, intrusion logs, and time-stamped event alerts.

V. Incident Summary

Incident Analysis

Test intrusions triggered alerts and logs, confirming detection system functionality.

Incident Response

The system logged IP, timestamp, and action type for every suspicious request. Alerts were triggered and recorded in log files.

Lessons Learned

Logging and alerting are effective, but future iterations should support real-time blocking and external notifications.

VI. Recommendations

Improve Security Controls

Include SSL/TLS, implement stronger hashing for passwords, and enhance session management.

Enhance Security Awareness

Educate users on secure file handling, phishing threats, and password hygiene.

Improve Incident Response

Introduce real-time alerts (email/SMS) and use third-party intrusion detection frameworks.

Implement Security Policies

Define access control policies, secure file retention rules, and incident management procedures.

Continuous Improvement

Automate periodic vulnerability scans, regularly update libraries, and perform annual penetration testing.

VII. Appendices

Detailed Vulnerability Reports

Simulated tests revealed no critical vulnerabilities but highlighted a need for multi-user support and real-time defenses.

Risk Assessment Matrix

Risks were rated based on likelihood and impact, with mitigation strategies tailored accordingly.

Incident Response Plan

The plan includes detection (logging & alerts), containment (access lock), and recovery (manual override and audit trails).

Supporting Documents

Repository:
https://veturisaikishor.github.io/Encryption_file_sharing_system_with_intrusion_alerts_hackculprit-HCIN1403004/
Tools: Python, Flask, AES Library
Organization: Hack Culprit