LIST OF PROJECTS

1. Web Application Vulnerability Scanner

- **Objective:** Build a scanner to detect common web app vulnerabilities like XSS, SQLi, CSRF.
- **Tools:** Python, requests, BeautifulSoup, OWASP top 10 checklist, Flask
- **Mini Guide:**
  a. Use requests and BeautifulSoup to crawl input fields and URLs.
  b. Inject payloads for XSS, SQLi, etc., and analyze responses.
  c. Use regex or pattern matching for vulnerability detection.
  d. Create a Flask UI to manage scans and view results.
  e. Log each vulnerability with evidence and severity.
- **Deliverables:** Python-based scanner with web interface and detailed reports.

2. Personal Firewall using Python

- **Objective:** Develop a lightweight personal firewall that filters traffic based on rules.
- **Tools:** Python, scapy, iptables (Linux), Tkinter (GUI optional)
- **Mini Guide:**
  a. Use scapy to sniff incoming/outgoing packets.
  b. Define rule sets to block/allow IPs, ports, protocols.
  c. Log suspicious packets for audit.
  d. Optionally, use iptables to enforce rules on system level.
  e. Create GUI for live monitoring.
- **Deliverables:** CLI/GUI-based firewall with rule customization and logging.

3.Keylogger with Encrypted Data Exfiltration

- **Objective:** Build a proof-of-concept keylogger that encrypts logs and simulates exfiltration.
- **Tools:** Python, pynput, cryptography, base64
- **Mini Guide:**
  a. Capture keystrokes using pynput.
  b. Encrypt data using cryptography.fernet.
  c. Store logs locally with timestamp.
  d. Simulate sending to a remote server (localhost).
  e. Add startup persistence and kill switch.
- **Deliverables:** Encrypted keylogger PoC with ethical constraints and logs.

4. Password Strength Analyzer with Custom Wordlist Generator

- **Objective:** Build a tool to analyze password strength and generate custom wordlists.
- **Tools:** Python, argparse, NLTK, zxcvbn
- **Mini Guide:**
  a. Analyze user password using zxcvbn or custom entropy calculations.
  b. Allow user inputs (name, date, pet) to generate a custom wordlist.
  c. Include common patterns like leetspeak, append years.
  d. Export in .txt format for cracking tools.
  e. Add GUI with tkinter or CLI interface.
- **Deliverables:** Tool that evaluates password strength and exports attack-specific wordlists.

5. Secure File Storage System with AES

- **Objective:** Create a local file encryption/decryption system with AES-256.
- **Tools:** Python, cryptography, PyQt5 or CLI
- **Mini Guide:**
  a. Use AES (Fernet or manual key + IV).
  b. Allow upload, encrypt, and save with .enc extension.
  c. Store metadata (file name, time, hash) securely.
  d. Allow secure retrieval with decryption.
  e. Add hash verification to prevent tampering.
- **Deliverables:** AES-secured file storage app with integrity verification.

6. Ethical Phishing Simulation Platform

- **Objective:** Simulate phishing campaigns for educational/training purposes.
- **Tools:** Flask, Sendmail/Postfix (for SMTP), HTML/CSS, SQLite
- **Mini Guide:**
  a. Design customizable phishing templates.
  b. Send emails to test users (in a safe lab).
  c. Track clicks, inputs, and timestamps.
  d. Display analytics (open rate, success rate).
  e. Educate users post-campaign with best practices.
- **Deliverables:** Web-based phishing simulation and analytics dashboard.

7. Network Packet Sniffer with Alert System

- **Objective:** Build a real-time network traffic sniffer with anomaly detection.
- **Tools:** Python, scapy, SQLite, matplotlib
- **Mini Guide:**
    - a. Capture packets and log headers (IP, port, length, flags).
    - b. Detect anomalies (e.g., port scanning, flooding).
    - c. Store data in SQLite and display traffic summary.
    - d. Send alert on threshold breach (via email/log).
    - e. Optional: Add GUI for live traffic graph.
- **Deliverables:** CLI/GUI packet sniffer with anomaly alerting and database logs.

8. Linux Hardening Audit Tool

- **Objective:** Create a tool to audit a Linux system's security configuration.
- **Tools:** Bash or Python, os, subprocess
- **Mini Guide:**
    - a. Check firewall rules, unused services, SSH settings.
    - b. Verify permissions on key files (/etc/shadow, /etc/passwd).
    - c. Check for rootkit indicators.
    - d. Generate a score/report based on CIS benchmarks.
    - e. Recommend hardening actions.
- **Deliverables:** Script that generates system audit reports with compliance score.

9. SQL Injection Playground with Detection Engine

- **Objective**: Build a vulnerable app and an engine to detect SQLi in real-time.
- **Tools:** PHP/Flask, SQLite, Python detection tool
- **Mini Guide:**
    - a. Create a basic login/search page with intentional SQL flaws.
    - b. Build a Python script to inject and detect behavior (timeouts, errors).
    - c. Log successful injections and responses.
    - d. Show how parameterization can prevent SQLi.
    - e. Wrap into educational platform.
- **Deliverables:** Vulnerable app + SQLi detector with logs and defense example.

10. Secure Chat App with End-to-End Encryption

- Objective: Create a private chat application with E2EE using public-key cryptography.
- Tools: Python, Flask-SocketIO, RSA/AES from cryptography
- Mini Guide:
    a. Generate RSA keys per user and share public keys.
    b. Encrypt messages with AES, keys shared via RSA.
    c. Create real-time communication with Flask-SocketIO.
    d. Store chat logs encrypted on server (optional).
    e. Display messages decrypted only on client side.
- Deliverables: Secure chat app with E2EE and encrypted logs.

11. Log File Analyzer for Intrusion Detection

- Objective: Detect suspicious patterns in logs (Apache, SSH, etc.).
- Tools: Python, regex, pandas, matplotlib
- Mini Guide:
    a. Parse Apache and SSH logs.
    b. Identify brute-force, scanning, and DoS patterns.
    c. Visualize access patterns (by IP, time).
    d. Cross-reference with IP blacklist (public).
    e. Export incident reports.
- Deliverables: Python tool that processes logs, flags threats, and exports alerts.

12. Browser Extension to Block Trackers

- Objective: Build a privacy-focused extension to block known tracking scripts.
- Tools: JavaScript, Manifest v3, HTML/CSS
- Mini Guide:
    a. Maintain list of tracking domains (or use DuckDuckGo’s).
    b. Intercept requests via webRequest API.
    c. Block or redirect based on matches.
    d. Show a badge counter for blocked scripts.
    e. Add user-controlled whitelist/blacklist.
- Deliverables: Chrome/Firefox extension that blocks trackers and shows analytics.

13. Steganography Tool for Image/File Hiding

- **Objective:** Hide text or files inside images using steganography.
- **Tools:** Python, PIL, stepic, tkinter
- **Mini Guide:**
    - a.Convert message to binary and embed in image LSB.
    - b.Allow uploading image + hidden message.
    - c.Extract and decrypt (optional) from modified image.
    - d.Add drag-and-drop GUI.
    - e.Support image formats like PNG, BMP.
- **Deliverables:** GUI tool for embedding and extracting data from images.

14. HoneyPot Server to Detect Attack Patterns

- **Objective:** Deploy a honeypot to simulate vulnerable services and log attackers.
- **Tools:** Cowrie or custom Python scripts, SSH/FTP emulation
- **Mini Guide:**
    - a.Deploy honeypot on a VM.
    - b.Log connections, IPs, attempted commands.
    - c.Analyze log files for repeated attempts.
    - d.Use fail2ban to block real threats.
    - e.Visualize IP geolocation of attackers.
- **Deliverables:** Running honeypot + detailed logs + visual attack reports.

15. Cyber Threat Intelligence Dashboard

- **Objective:** Build a dashboard that aggregates real-time threat feeds.
- **Tools:** Flask/Django, VirusTotal API (free tier), AbuseIPDB, MongoDB
- **Mini Guide:**
    - a.Pull data from open CTI sources and APIs.
    - b.Display threat level, IOC (Indicators of Compromise), and trends.
    - c.Enable user to input IP/domains and verify against threat databases.
    - d.Visualize threat metrics over time.
    - e.Add tagging and export feature.
- **Deliverables:** Real-time CTI dashboard with threat lookup and visualizations.