12th Project

🔐 12th Project – Web Security Simulation Project

👥 Team Structure

- Each team will consist of **5 students**.
- Team roles:
  - **1 Offensive Security Specialist** (Attacker)
  - **1 System Administrator** (Server/Network setup)
  - **3 Log Analysts/Defenders** (Use Splunk, monitor security, detect attacks)

🗓️ Project Timeline Overview

- **Weeks 10–12**:
  - Teams **build and secure their web servers locally** using VMs.
  - Prepare Splunk VM for live log analysis.
  - Develop attack tools/scripts and hide security flags.
- **Week 13 (On-Campus)**:
  - **All teams gather in college** for the hacking simulation.
  - Teams will **scan the network** to discover others' web servers.
  - Launch attacks and **analyze logs in real-time** using Splunk.
  - Collect evidence and prepare final reports.

🛡️ Defender Role – Local Web Server Creation

1. Web Server Setup

- **Platform**: Virtual Machine (VM) on Kali Linux or any Linux distro.

- **Web Server**: Apache (or any preferred)
- **Programming Language**: PHP (or your choice)
- **Database**: MySQL (or your choice)

Each student/team will set up:

- **One Web Server VM**
- **One Splunk VM**

> Both can be hosted on the same or separate laptops.

2. Required Features

- Homepage
- Login page
- Contact form
- **At least one script vulnerable to SQL Injection or XSS**
- Flag embedded in the format: `flag_{xxxx-xxxx-xxxx}` inside vulnerable code

3. Deployment & Hardening

- **No need for public hosting** – all hosted locally
- Disable directory listing
- Remove unused Apache modules
- File permission hardening
- Basic input validation
- Use HTTPS (self-signed certificate usage is a **bonus**)

🧨 Attacker Role – Offensive Testing

1. Network Scanning

- Scan college LAN to find other teams' web servers (e.g., using `nmap`)

2. Attack Execution

Choose from:

- SQL Injection (to extract flag)
- XSS
- Brute Force
- Directory Traversal
- Malicious File Upload
- Others (creative attacks welcome)

> Target is to capture flags from other teams' vulnerable apps.

3. Tools

- `sqlmap`, `Burp Suite`, `Hydra`, `Nikto`, `WFuzz`, `OWASP ZAP`, etc.

📊 Defender Role – Splunk & Real-Time Monitoring

1. Log Collection

- Monitor logs in:
  - `/var/log/apache2/access.log`
  - `/var/log/apache2/error.log`
- Forward logs to Splunk
- Use Splunk dashboards and queries to:
  - Detect SQLi, XSS, brute force, etc.
  - Trace IPs and actions of attackers
  - Identify stolen flag attempts

🚩 Capture-the-Flag Integration (CTF Style) Bonus

- Each team hides **1–3 unique flags** (`flag_{1234-5678-9012}` format) in vulnerable code (like in SQL database, query results, or HTML).
- Other teams must:
  - Find and exploit the vulnerability
  - Extract and **screenshot the flag**
  - Document the method used

✅ Final Deliverables

🛡️ Website & Infrastructure

- Web server VM configuration
- Splunk VM setup and dashboard
- Screenshot of vulnerable code with embedded flag
- Evidence of hardening and defense measures

💥 Offensive Report

- Attacks performed
- Tools used
- Flags captured
- Screenshots and logs of success/failure

🔍 Log Analysis Report

- Attack traces in Splunk
- IPs, payloads, methods detected
- Response actions (block IP, patch, etc.)
- Improvements implemented post-attack

📌 Important Rules

- No copying code or reports between teams
- Each team must submit **original code and configurations**
- Team members must perform **distinct roles**
- Submit logs, screenshots, and documentation as proof