Cybersecurity Internship Program

Case Study → Product Design → Securing the Product

1. Program Overview

This 45-hour, one month internship provides a comprehensive, lab-driven journey built around a single continuous workflow, starting with IT, OS, Active Directory, and networking foundations before moving into software lifecycle theory. Interns first master these core baselines, then identify a genuine daily-life problem through a top-down case study, design an innovative software product architecture with UI/UX wireframes and DFDs, and execute a dedicated cybersecurity project to secure that conceptualized product—using STRIDE threat modeling, Static Application Security Testing (SAST) with tools like SonarQube, Bandit, and ESLint, secure coding practices, and Dynamic Application Security Testing (DAST) / Vulnerability Assessment & Penetration Testing (VAPT) via Nmap, OWASP ZAP, Nikto, Burp Suite, and SQL map. This mirrors how security is applied in real organizations: built into a product from the design stage, not bolted on afterward.

2. Learning Objectives

By the end of the program, learners will be able to:

- Explain fundamental cybersecurity principles, operating systems, Active Directory, and networking baselines.
- Identify a real-world problem and translate it into a formal Software Requirement Specification (SRS) through a structured case-study methodology.
- Design a software product architecture, UI/UX wireframes, and Data Flow Diagrams (DFDs) with security integrated from the design stage (Security by Design).
- Apply secure coding practices and execute automated SAST using SonarQube, Bandit, and ESLint security rules.
- Perform STRIDE threat modeling, vulnerability scanning, and dynamic penetration testing on the target application environment.
- Produce professional security artifacts: SRS, UI/UX wireframes, DFDs, STRIDE threat model, VAPT report, and CWE-mapped developer remediation tickets.

3. Program Structure at a Glance

Week	Focus Area	Hours	Key Outcome
Week 1	IT Basics, Foundations of Cybersecurity & OS/AD Setup	10 hrs	Mastered core computer/network security baselines, operating systems, and Active Directory foundations.
Week 2	Networking, SSDLC, + Problem Identification (Case Study) & UI/UX, SFD. DFD Innovative Software Product Design, Security-by-Design (Threat Modeling)	12 hrs	Completed network diagnostics, software lifecycle theory, case study problem scoping, SRS, UI/UX wireframes, DFDs, and STRIDE threat modeling.
Week 3	Secure Coding, SAST & Ethical Hacking Recon/Scanning (DAST)	13 hrs	Deployed prototype targets, ran SAST scans with SonarQube/Bandit/ESLint, and executed reconnaissance/scanning via Nmap, OWASP ZAP, and Nikto.
Week 4	DAST Exploitation, Remediation, VAPT, Hardening, Reporting Capstone	10 hrs	Performed active web exploitation via Burp Suite/SQL map, wrote CWE-mapped developer tickets, compiled VAPT reports, and delivered final presentations.

4. Consolidated Course Syllabus (Modules + Outcomes)

Mod. No.	Module Title	Detailed Topics	Hours
1	**Cyber Fundamentals & Threat Landscape**	CIA Triad; Risk Management; Cyber Laws (Indian IT Act, GDPR); Hacker types/motives; Major Attack Types (Malware, Phishing, DoS, SQL Injection, XSS); Pen Testing Types; Vulnerability taxonomies (CVE, CWE, CVSS).	4
2	**Operating Systems & Active Directory**	OS comparison (Windows, Linux); Kali Linux VM installation; windows and Linux commands, Linux file permissions; Active Directory basics (Domains, Forests, Domain Controllers, Identity Management, Group Policies).	4
3	**Networking Essentials for Security**	OSI 7-layer and TCP/IP 4-layer models; IP addressing & subnetting; network traffic and packet structures; TCP 3-way handshake; well-known ports; DNS; SSL/TLS; CLI network diagnostics.	4
4	**Secure Software Lifecycle & AppSec Basics**	SDLC, SSDLC, DevSecOps principles; Introduction to SAST and DAST; Secure development principles.	3
5	**The Case Study & Requirement Gathering**	Case study method (top-down application of SDLC): identifying a daily-life problem, assessing stakeholder needs; drafting the Software Requirement Specification (SRS).	4
6	**Product Design & System Flow Diagrams**	Translating SRS requirements into technical design; UI/UX wireframes; system architecture; Creating System Flow Diagrams (SFD) and Data Flow Diagrams (DFDs).	4
7	**Threat Modeling Methodologies**	Threat modeling concepts (PASTA, DREAD, Trike); Deep dive into STRIDE (Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege).	3
8	**Practical STRIDE Threat Modeling**	Applying STRIDE to the student's own DFD; identifying system assets, drawing trust boundaries, categorizing potential threats, and defining mitigation requirements.	3
9	**Secure Coding & SAST Execution**	Secure Coding Practices (input validation, parameterized queries, output encoding); Environment setup; deploying target application prototypes; executing SAST with SonarQube, Bandit, and ESLint security rules.	4
10	**Ethical Hacking: Recon & Scanning (DAST)**	Overview of all 5 Ethical Hacking phases. Deep dive into Phase 1 (Reconnaissance: OSINT, WHOIS) and Phase 2 (Scanning). Utilizing Nmap, OWASP ZAP, and Nikto.	5
11	**DAST Exploitation & Gaining Access**	Deep dive into Phase 3 (Gaining Access). Intercepting web traffic with Burp Suite; testing against OWASP Top 10 vulnerabilities; automated exploitation with SQL map.	4
12	**Operational Remediation & VAPT Reporting**	Risk classification (CVSS); mapping vulnerabilities to CWE database entries; writing developer remediation tickets; compiling the comprehensive VAPT capstone report.	3

5. IT Basics & Security Foundation

Day	Duration	Activities
Day 1	2 hrs	**Cyber Fundamentals:** Importance of Cybersecurity; CIA Triad; Risk Management basics; Hacker types, motives, and cyber laws; Vulnerability taxonomies (CVE/CWE/CVSS).
Day 2	2 hrs	**Threat Landscape:** Malware types and OWASP Top 10 overview; Pen Testing Methodologies (Black-box, White-box). OS comparison; Kali Linux VM lab setup; Linux CLI navigation.
Day 3	2 hrs	**Operating Systems & AD:** OS comparison (Windows, Linux); Kali Linux VM installation; windows and Linux commands, Advanced Linux file permissions. Active Directory basics (Domains, Forests, Identity Management, Group Policies).
Day 4	2 hrs	**Networking Fundamentals (Part 1):** OSI & TCP/IP models, IP addressing, DNS; Network diagnostic CLI lab (ping, traceroute, netstat/ss).
Day 5	2 hrs	**Networking Fundamentals (Part 2):** TCP 3-way handshake deep dive, well-known ports, SSL/TLS packet structures, and advanced command-line troubleshooting.

6. SSDLC, Case Study & Product Design

Day	Duration	Activities
Day 6	2 hrs	**Lifecycle Theory:** Introduction to SDLC, SSDLC, DevSecOps, and SAST vs. DAST.
Day 7	2 hrs	**The Case Study:** Identifying a daily-life problem, assessing stakeholder needs, scoping features; drafting the Software Requirement Specification (SRS).
Day 8	2 hrs	**Product Design & UI/UX:** System architecture and tech stack selection; mapping UI/UX wireframes; Translating SRS into System Flow Diagrams (SFD) and Data Flow Diagrams (DFDs).
Day 9	2 hrs	**Threat Modeling Theory:** Overview of PASTA, DREAD, and Trike. Comprehensive breakdown of the STRIDE framework.
Day 10	2 hrs	**Practical STRIDE Workshop:** Apply STRIDE to the student's own DFD: list assets, draw trust boundaries, and catalog threats.

7. SAST, Secure Coding & Reconnaissance

Day	Duration	Activities
Day 11	2 hrs	**Secure Coding Principles:** Translating OWASP Top 10 into structural mitigations (input validation, parameterized queries, output encoding).
Day 12	2 hrs	**SAST Execution:** Prototype Deployment. Run automated SAST scanning tools (SonarQube, Bandit, ESLint) against the source directory and review findings.
Day 13	2 hrs	**Ethical Hacking Phases Overview & Recon:** Overview of all 5 phases. Deep dive Phase 1: Reconnaissance (passive/active recon, OSINT, WHOIS, DNS queries, Google Dorking).
Day 14	2 hrs	**Phase 2 Scanning:** Host and service discovery. Directory brute-forcing and vulnerability scanning using Nmap, Nikto, and OWASP ZAP.
Day 15	2 hrs	**Introduction to DAST:** Web application architecture review, HTTP request/response inspection, and configuring the Burp Suite proxy.

8. DAST Exploitation, Remediation & Capstone Reporting

Day	Duration	Activities
Day 16	2 hrs	**Phase 3 Gaining Access (DAST):** Testing for SQL Injection, XSS, and authentication flaws using Burp Suite and SQL map.
Day 17	2 hrs	**Advanced Exploitation & Concepts:** Advanced web payload testing, alongside a conceptual discussion of Phase 4 (Maintaining Access) & Phase 5 (Covering Tracks).
Day 18	2 hrs	**Operational Remediation:** Triage findings; map confirmed vulnerabilities to the CWE database. Write actionable developer remediation tickets based on Secure Coding Practices.
Day 19	2 hrs	**Report Compilation:** Writing the formal VAPT report (Executive Summary, Scope, Methodology, Detailed Findings, CVSS Risk Ratings, PoC Screenshots).
Day 20	2 hrs	**Final Capstone Demo & Presentation:** Interns present the end-to-end project: Case Study → Product Design (UI/UX & DFD) → STRIDE Threats → SAST/DAST Findings → Remediation Report.

9. How to Run the Securing Project, Step by Step

Once the prototype application is deployed and scanned, treat securing and testing it as its own scoped project with these steps:

- **Step 1 — Re-check the threat model:** revisit the STRIDE model against the deployed prototype or environment; note any gaps between the original design and implementation.

- **Step 2 — Define scope & rules of engagement:** determine which modules, APIs, or pages are in scope, strictly on a test or staging copy—never on live user data.
- **Step 3 — Reconnaissance & scanning:** map the application's attack surface; run automated SAST (SonarQube, Bandit, ESLint) and DAST tools (OWASP ZAP, Nmap, Nikto) to surface low-hanging vulnerabilities.
- **Step 4 — Manual testing against OWASP Top 10:** test for authentication bypass, SQL injection, XSS, and security misconfigurations using Burp Suite.
- **Step 5 — Severity rating:** classify each finding (Critical, High, Medium, Low) using CVSS risk ratings.
- **Step 6 — Operational Remediation:** map confirmed vulnerabilities to the CWE database and write actionable developer remediation tickets based on Secure Coding Practices.
- **Step 7 — Retest:** confirm how each fix closes the vulnerability based on CWE guidelines without breaking system functionality.
- **Step 8 — Report & document:** produce a professional VAPT report (findings, evidence, CVSS severity, CWE-mapped developer tickets) and a hardening checklist.

10. Suggested Tools

- **Reconnaissance/Scanning:** Nmap, OWASP ZAP, Nikto
- **Manual Testing & DAST:** Burp Suite (Community Edition), browser dev tools
- **Static/Code Analysis (SAST):** SonarQube, ESLint security plugins, Bandit (Python)
- **Data Flow Diagrams:** Draw.io for creating system architecture diagrams and DFDs during threat modeling.
- **Documentation:** Threat-modeling templates (STRIDE), CVSS calculators, CWE database references

11. Evaluation & Deliverables Checklist

- Approved problem statement & Software Requirement Specification (SRS) (Week 1)
- UI/UX wireframes, System Flow Diagram (SFD), Data Flow Diagram (DFD), and STRIDE threat model (Week 2)
- Deployed application prototype with automated SAST scan output and secure coding review (Week 3)
- VAPT report, CWE-mapped developer remediation tickets, hardening checklist, and final capstone demo (Week 4)
- Weekly mentor sign-off and progress evaluations

12. Program Assessment & Knowledge Quizzes

- **Quiz 1 (End of Week 1):** Assessment covering core cybersecurity principles, CIA triad, operating systems, Active Directory basics, and networking essentials.
- **Quiz 2 (End of Week 2):** Assessment covering SDLC/SSDLC lifecycle theory, case study scoping, UI/UX design, DFD mapping, and STRIDE threat modeling concepts.
- **Quiz 3 (End of Week 3):** Assessment covering Secure Coding Practices, automated SAST tool outputs (SonarQube, Bandit, ESLint), and ethical hacking recon/scanning phases.
- **Quiz 4 (End of Week 4):** Assessment covering DAST execution, vulnerability severity scoring (CVSS), CWE mapping, and operational remediation reporting.