Developing a Secure SDLC Framework

Integrating security throughout the software development lifecycle for DevSecOps

Brief overview of the project

This project focuses on developing a lightweight Secure Software Development Lifecycle (SDLC) framework using DevSecOps practices. A small Python application will be built as a case study, where threat modeling through the STRIDE approach will be applied to identify potential risks. Secure coding practices will then be implemented to address these threats. To ensure continuous security, a CI/CD pipeline will be designed using GitHub Actions, with Bandit integrated as a static analysis tool for automated vulnerability checks. The aim is to demonstrate a simple and practical framework that can be easily adopted by students and small teams for learning and implementation purposes.

Background of the Project

IN TODAY'S SOFTWARE INDUSTRY, SECURITY HAS BECOME A CRITICAL CONCERN AS APPLICATIONS ARE INCREASINGLY TARGETED BY CYBER THREATS. TRADITIONAL DEVELOPMENT APPROACHES OFTEN ADD SECURITY MEASURES AT THE END OF THE LIFECYCLE, WHICH MAKES IT COSTLY AND LESS EFFECTIVE. DEVSECOPS INTRODUCES A "SHIFT-LEFT" APPROACH, WHERE SECURITY IS INTEGRATED INTO EVERY STAGE OF DEVELOPMENT, FROM PLANNING TO DEPLOYMENT. HOWEVER, MOST AVAILABLE FRAMEWORKS ARE DESIGNED FOR LARGE ENTERPRISES AND RELY ON COMPLEX TOOLS THAT MAY NOT BE SUITABLE FOR STUDENTS OR SMALL TEAMS. THIS PROJECT ADDRESSES THIS GAP BY CREATING A LIGHTWEIGHT AND PRACTICAL SECURE SDLC FRAMEWORK THAT USES SIMPLE, OPEN-SOURCE TOOLS TO DEMONSTRATE HOW SECURITY CAN BE EMBEDDED IN THE DEVELOPMENT PROCESS IN AN ACCESSIBLE WAY.

Domain/Technology used

DOMAIN: SOFTWARE ENGINEERING & CYBERSECURITY

PROGRAMMING LANGUAGE: PYTHON (SAMPLE APPLICATION)

FOCUS: SECURE SDLC WITHIN A DEVSECOPS FRAMEWORK

CI/CD TOOL: GITHUB ACTIONS (AUTOMATED PIPELINE)

SECURITY TOOL: BANDIT (STATIC CODE ANALYSIS)

DIAGRAMMING TOOL: DRAW.IO (FOR THREAT MODELING)

APPROACH: LIGHTWEIGHT, EASY-TO-IMPLEMENT, STUDENT-FRIENDLY TECHNOLOGIES

Problem Statement

1. Traditional SDLC often adds security only at the end → costly and less effective

2. Existing DevSecOps frameworks are complex and enterprise-focused

3. Students and small teams lack lightweight, practical models

4. Limited use of automated security gates in small-scale CI/CD pipelines

5. Need for a simple, accessible framework to integrate security early in development

Objectives:-

- Perform threat modeling for a small Python application
- Apply secure coding practices to address identified risks
- Design and configure a CI/CD pipeline using GitHub Actions
- Integrate Bandit for automated security checks in the pipeline
- Demonstrate a lightweight Secure SDLC framework for students/small teams

Scope:-

- Development of a sample Python application with basic security measures
- Use of GitHub Actions + Bandit for automated pipeline security
- Focus on lightweight and accessible tools
- Out of scope: Large-scale enterprise deployment, advanced tools (SCA/DAST), detailed compliance mapping

METHODOLOGY / IMPLEMENTATION PLAN

1. REQUIREMENT & THREAT MODELING

DEFINE REQUIREMENTS OF THE SAMPLE PYTHON APP
PERFORM STRIDE-BASED THREAT MODELING WITH DIAGRAMS

. 2. DESIGN PHASE

INCORPORATE IDENTIFIED SECURITY MITIGATIONS INTO SYSTEM DESIGN

3. IMPLEMENTATION PHASE

DEVELOP PYTHON APPLICATION FOLLOWING SECURE CODING PRACTICES

4. PIPELINE INTEGRATION PHASE

CONFIGURE GITHUB ACTIONS FOR AUTOMATED BUILDS/TESTS
INTEGRATE BANDIT FOR STATIC SECURITY ANALYSIS

5. TESTING & EVALUATION PHASE

RUN PIPELINE WITH CODE CHANGES
VERIFY DETECTION OF VULNERABILITIES AND PIPELINE ENFORCEMENT

6. DEPLOYMENT & MAINTENANCE PHASE

DEPLOY SECURED APP IN A SIMPLE ENVIRONMENT
MAINTAIN THROUGH UPDATES AND RE-RUNNING PIPELINE CHECKS

Expected Outcome

1. A LIGHTWEIGHT SECURE SDLC FRAMEWORK SUITABLE FOR STUDENTS AND SMALL TEAMS

2. DEMONSTRATION OF THREAT MODELING (STRIDE) WITH DIAGRAMS AND MITIGATIONS

3. A WORKING CI/CD PIPELINE IN GITHUB ACTIONS

4. BANDIT INTEGRATED AS A SECURITY GATE TO DETECT VULNERABILITIES AUTOMATICALLY

5. CLEAR EVIDENCE THAT INSECURE CODE IS IDENTIFIED AND BLOCKED DURING DEVELOPMENT

WORK PLAN / TIMELINE

WEEK 1: REQUIREMENT GATHERING & PLANNING

WEEK 2: THREAT MODELING (STRIDE) AND DIAGRAMS

WEEK 3: DESIGN PHASE – INTEGRATE SECURITY CONSIDERATIONS

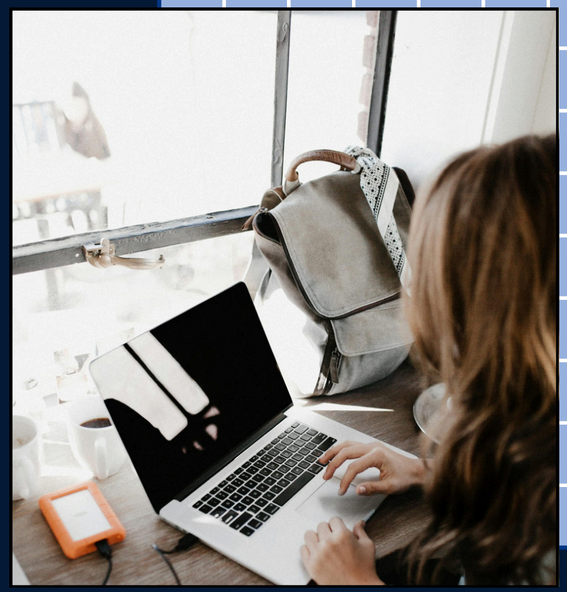
WEEK 4: IMPLEMENTATION – DEVELOP SAMPLE PYTHON APP

WEEK 5: CONFIGURE CI/CD PIPELINE WITH GITHUB ACTIONS

WEEK 6: INTEGRATE BANDIT FOR AUTOMATED SECURITY CHECKS

WEEK 7: TESTING & EVALUATION OF PIPELINE

WEEK 8: DEPLOYMENT, DOCUMENTATION & FINAL REPORT PREPARATION



These images illustrate key practices in securing the software development lifecycle, emphasizing collaboration, automation, and continuous security integration.

Thank you for your attention!