PW SKILLS | DevOps and Cloud Computing

Security Testing

Objective

- Understand the importance of security testing in software development.
- Use static code analysis tools (SonarQube, Fortify) to detect vulnerabilities.
- Learn about dynamic application security testing (DAST) ar its role in securing applications.
- Gain insights into penetration testing and ethical hacking techniques.
- Integrate security testing into CI/CD pipelines for automate security scans.

PW SKILLS

Explaining the purpose of static code analysis (SAST - Static Application Security Testing)

Purpose of static code analysis

Static Application Security Testing (SAST) is a white-box testing method used to analyze an application's source code, bytecode, or binaries to identify potential security vulnerabilities without executing the application.

Key Purposes of SAST:

- Early Vulnerability Detection
- Shifting Security Left
- Improved Code Quality
- CI/CD Integration
- Regulatory Compliance

PW SKILLS

Introducing tools

Key tools

1. **SonarQube**

Purpose: A static code analysis tool for continuous inspection of code quality.

Features:

- Detects code smells, bugs, and security vulnerabilities.
- Supports over 29 programming languages, including Java, Python, and JavaScript.
- Provides metrics on code duplication, complexity, test coverage, and more.
- Integrates seamlessly with CI/CD pipelines for automated analysis.
- Helps reduce technical debt by promoting clean and maintainable code practices.

Key tools

2. Fortify

Purpose: An enterprise-grade security scanning tool designed for compliance and threat mitigation.

Features:

- Performs deep static application security testing (SAST) to identify vulnerabilities in source code.
- Focuses on regulatory compliance and secure coding standards.
- Scales to large enterprise environments with robust reporting and integration capabilities.
- Assists in mitigating risks by providing actionable insights into security flaws.

Pop Quiz

Q. What type of security testing does Fortify specialize in?

A

Enterprise-grade static application security testing (SAST)

B

Dynamic application security testing (DAST)

Pop Quiz

Q. What type of security testing does Fortify specialize in?

A

Enterprise-grade static application security testing (SAST)

B

Dynamic application security testing (DAST)

PW SKILLS

Demonstrating running a static analysis scan on a sample project using SonarQube

Let’s do it

Step-by-Step Guide

1. **Install SonarQube Server:**

- Download and install the SonarQube server. You can use the Community Build for free or the Developer Edition for more features.

2. **Configure SonarQube:**

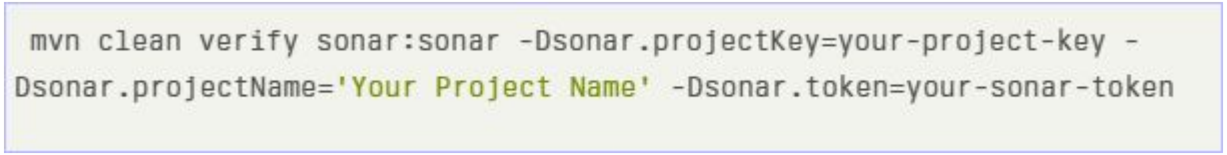
- Set up the SonarQube server and ensure it is running on a specified port (e.g., *http://localhost:9000).*

3. **Prepare Your Project:**

- Ensure your project is set up with a build tool like Maven or Gradle. For Maven, add the SonarQube plugin to your pom.xml file.

4. Run the Scan:

- Use the SonarScanner to analyze your project. For Maven, run the following command:



```
 mvn clean verify sonar:sonar -Dsonar.projectKey=your-project-key -
Dsonar.projectName='Your Project Name' -Dsonar.token=your-sonar-token
```

- Replace placeholders with your actual project details.

5. Review Results:

- Log into the SonarQube web interface to view analysis results, including code quality metrics, bugs, vulnerabilities, and code smells.

PW SKILLS

DAST and how it differs from SAST (testing running applications vs. source code analysis)

DAST v/s SAST

Dynamic Application Security Testing (DAST) is a black-box testing method that analyzes running applications to identify vulnerabilities from an external perspective, simulating real-world attacks.

- It focuses on detecting issues like cross-site scripting (XSS), SQL injection, and authentication flaws by interacting with the application's exposed interfaces, such as APIs or web pages.

DAST v/s SAST

How DAST Differs from SAST.

1. Testing Stage:

- SAST: Conducted early in the development lifecycle by analyzing source code or binaries before execution.
- DAST: Performed later on a running application in a test or production environment.

2. Perspective:

- SAST: White-box testing (internal code analysis).
- DAST: Black-box testing (external behavior analysis).

DAST v/s SAST

3. Language Dependence:

- SAST: Language-dependent, requiring tools compatible with the programming language used.
- DAST: Language-agnostic, focusing on runtime behavior regardless of the underlying code.

4. Types of Vulnerabilities Detected:

- SAST: Identifies coding errors like input validation issues and buffer overflows.
- DAST: Detects runtime vulnerabilities like misconfigurations and security flaws exploitable during execution.

PW SKILLS

Introduce common DAST tools

DAST tools

1. OWASP ZAP (Zed Attack Proxy):

Type: Open-source
Purpose: Scans web applications for vulnerabilities.

Features:

- Detects issues like SQL injection, XSS, and authentication flaws.
- Easy to use with automation support for CI/CD pipelines.
- Includes proxy capabilities for intercepting and modifying requests.

DAST tools

2. Burp Suite

Type: Commercial (with a free community edition)
Purpose: Comprehensive web application security testing.

Features:

- Intercepts, modifies, and analyzes HTTP/S web requests.
- Offers active scanning for vulnerabilities like XSS and CSRF.
- Extensible with add-ons for advanced testing scenarios.

PW SKILLS

Demonstrating running an automated DAST scan using OWASP ZAP

Let’s do it

Steps to Perform a DAST Scan

1. **Set Up OWASP ZAP:**

- Use the ZAP Docker image for automation. Ensure Docker is installed on your system.

2. **Download Configuration Files:**

- Obtain predefined configuration files (e.g., zap-casa-config.conf) for scanning.

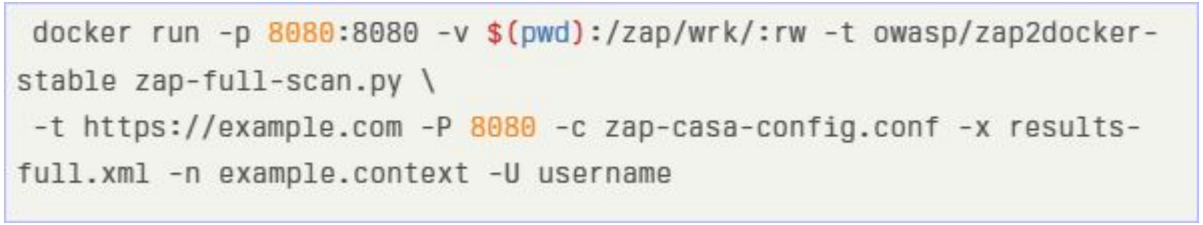
3. **Generate a Context File (Optional for Authentication):**

- Use the ZAP desktop UI to define authentication details (e.g., login URL, credentials) and export the context file.

Let’s do it

4. Run the Scan:

- Execute the following command to perform a full scan:



```
 docker run -p 8080:8080 -v $(pwd):/zap/wrk/:rw -t owasp/zap2docker-
stable zap-full-scan.py \
 -t https://example.com -P 8080 -c zap-casa-config.conf -x results-
full.xml -n example.context -U username
```

- Replace placeholders (https://example.com, username) with your target URL and authentication details.

5. Review Results:

- The scan results are saved in an XML file (e.g., results-full.xml). Analyze them for vulnerabilities like SQL injection, XSS, and more.

Take A 5-Minute **Break!**

- Stretch and relax
- Hydrate
- Clear your mind
- Be back in 5 minutes

Penetration testing and how it simulates real-world cyber attacks

Penetration testing

Penetration testing (pen testing) is a cybersecurity practice that simulates real-world cyberattacks to identify vulnerabilities in an organization's systems, networks, or applications.

How Pen Testing Simulates Real-World Cyber Attacks:

1. Reconnaissance
2. Exploitation
3. Maintaining Access
4. Impact Analysis

PW SKILLS

Introducing penetration testing tools

Penetration testing tools

1. **Metasploit**

Purpose: A powerful framework for penetration testing.

Features:

- Automates exploitation of security weaknesses in systems.
- Includes a library of prebuilt exploits and payloads.
- Simulates attacks like brute force, privilege escalation, and remote code execution.
- Ideal for ethical hackers to test and validate vulnerabilities.

Penetration testing tools

2. Nmap (Network Mapper)

Purpose: A network scanning tool for vulnerability discovery.

Features:

- Performs host discovery, port scanning, and OS detection.
- Identifies open ports and services running on hosts.
- Useful for reconnaissance and assessing potential attack vectors.
- Supports both beginners (via ZenMap GUI) and advanced users.

PW SKILLS

Discussing legal and ethical considerations of ethical hacking

Let’s discuss

Ethical hacking involves identifying and addressing vulnerabilities in systems or networks to enhance security, but it must adhere to strict legal and ethical guidelines to avoid misuse or legal consequences.

Legal Considerations:

1. Explicit Permission
2. Compliance with Laws
3. Defined Scope
4. Responsible Disclosure

Ethical Considerations:

1. Respect Privacy
2. Minimize Harm
3. Confidentiality
4. Adherence to a Code of Ethics

Why security testing should be part of CI/CD workflows.

Let’s see

Security testing should be part of CI/CD workflows to ensure secure, high-quality software while maintaining the speed and automation benefits of CI/CD. Here's why:

1. Early Detection of Vulnerabilities
2. Continuous Feedback
3. Compliance and Risk Mitigation
4. Maintaining Speed and Quality

Demonstrate integrating security scans in CI/CD

Let’s do it

1. Add a SonarQube Scan to a Jenkins Pipeline

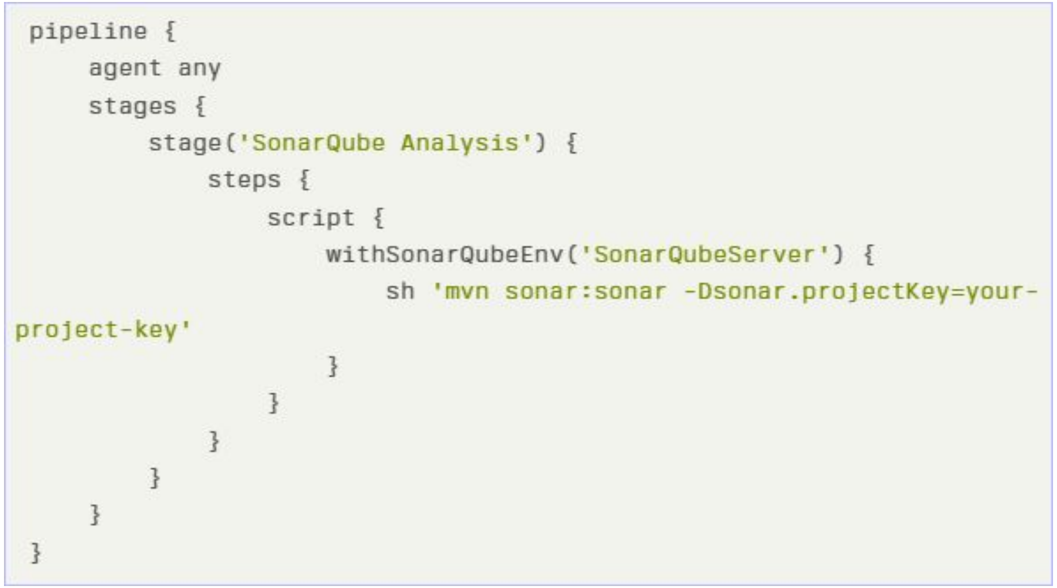
To integrate SonarQube into Jenkins for static code analysis:

1. **Install SonarQube Plugin:**
   - Go to Jenkins > Manage Jenkins > Manage Plugins and install the "SonarQube Scanner" plugin.

2. **Configure SonarQube:**
   - In Jenkins, navigate to Manage Jenkins > Configure System and add your SonarQube server details (URL and authentication token).

3. Add SonarQube to Pipeline Script:

- In your Jenkins pipeline file (Jenkinsfile), include the following:



```
pipeline {
    agent any
    stages {
        stage('SonarQube Analysis') {
            steps {
                script {
                    withSonarQubeEnv('SonarQubeServer') {
                        sh 'mvn sonar:sonar -Dsonar.projectKey=your-project-key'
                    }
                }
            }
        }
    }
}
```

- Replace SonarQubeServer and your-project-key with your specific configuration.

4. Run the Pipeline:

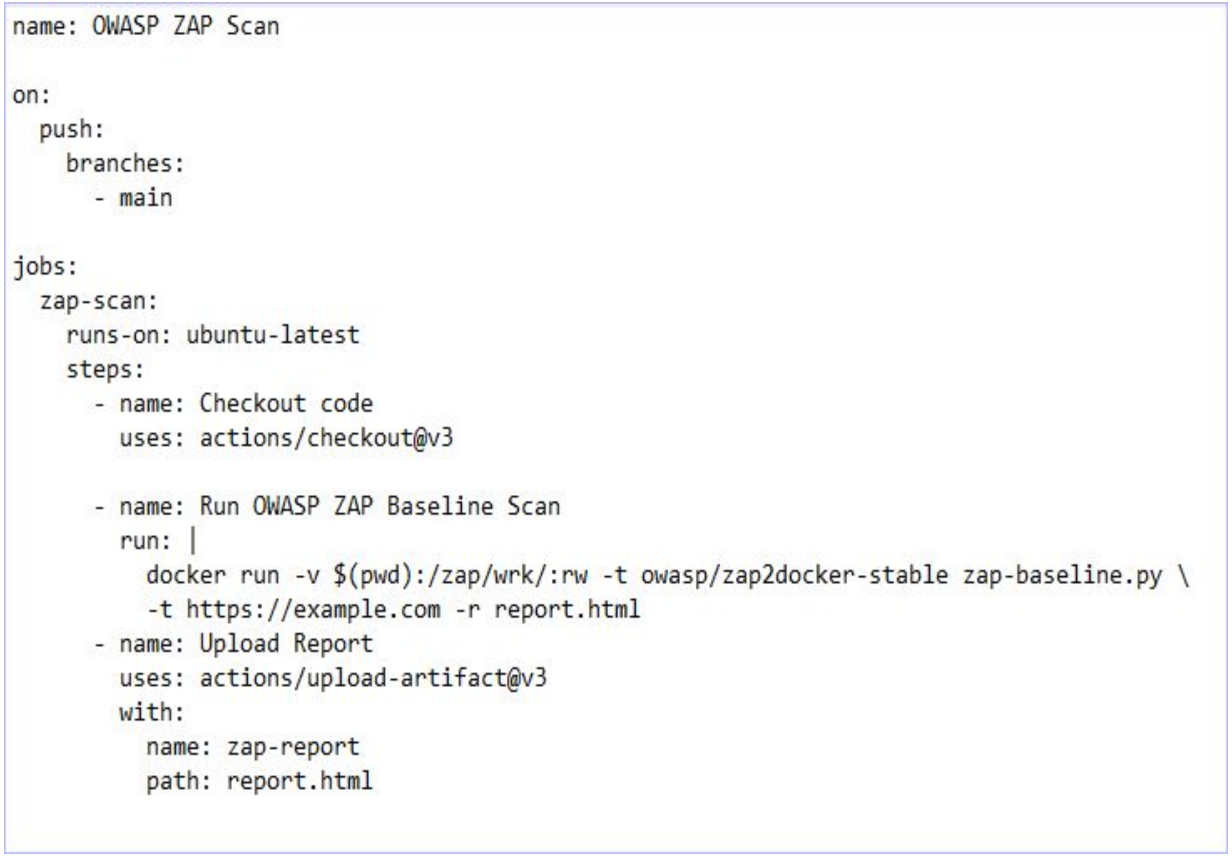
- Execute the pipeline to analyze your code and view results on the SonarQube dashboard.

2. Configure OWASP ZAP Scans in GitHub Actions

To set up OWASP ZAP for dynamic application security testing:

1. Create a GitHub Workflow File:

- Add a .github/workflows/zap-scan.yml file:



```yaml
name: OWASP ZAP Scan

on:
  push:
    branches:
      - main

jobs:
  zap-scan:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout code
        uses: actions/checkout@v3

      - name: Run OWASP ZAP Baseline Scan
        run: |
          docker run -v $(pwd):/zap/wrk/:rw -t owasp/zap2docker-stable zap-baseline.py \
          -t https://example.com -r report.html
      - name: Upload Report
        uses: actions/upload-artifact@v3
        with:
          name: zap-report
          path: report.html
```

2. Customize Configuration:

- Replace https://example.com with your application's URL.

3. Run the Workflow:

- Push changes to the repository, triggering the workflow. The ZAP scan will generate a report highlighting vulnerabilities.

PW SKILLS

Discuss security testing best practices in DevSecOps

Best practices

Integrating security testing best practices into DevSecOps is essential for building secure applications. Here are two key practices:

1. **Shift-Left Security**

Definition: This approach emphasizes integrating security measures early in the software development lifecycle (SDLC), starting from the planning and design phases.

Benefits:

- Early Detection of Vulnerabilities
- Proactive Risk Management

Best practices

2. Automate Security Scanning at Different Stages of Development

Definition: Automation involves integrating security scanning tools into CI/CD pipelines to routinely check for vulnerabilities throughout the development process.

Benefits:

- Consistency and Efficiency
- Continuous Feedback Loop

PW SKILLS

Time for case study!

Important

- Complete the post-class assessment
- Complete assignments (if any)
- Practice the concepts and techniques taught in this session
- Review your lecture notes
- Note down questions and queries regarding this session and consult the teaching assistants

Thanks

!