Secure Healthcare Software Development

CSP009

PRESENTATION BY: STYLIANOS KARAGIANNIS (PDMFC, PORTUGAL)

Static and Dynamic Application Security Testing (SAST and DAST)

Discovering flaws

- **Static Application Security Testing (SAST)** tools analyze source code for vulnerabilities before deployment. Examples include CodeQL, SonarQube, Checkmarx, and Fortify Static Code Analyzer.
- SAST identifies vulnerabilities such as unvalidated input in a PACS server's search function, potentially leading to SQL injection (e.g., SELECT * FROM Patients WHERE Name = 'input').
- SAST analysis of a PACS server's source code may reveal vulnerabilities such as CVE-2019-5649, a SQL injection vulnerability in the patient search functionality. Remediation involves sanitizing inputs and using parameterized queries to prevent SQL injection attacks.

- **Dynamic Application Security Testing (DAST)** evaluates running applications for vulnerabilities by simulating attacks.
- Example tools include OWASP ZAP, Burp Suite, and Acunetix.
- DAST scans may uncover vulnerabilities like insufficient input validation in a HIS login form, potentially leading to XSS (e.g., <script>alert('XSS')</script>).

SAST in Practice Pt.1

GitHub Code Scanning

GitHub Code Scanning is a built-in SAST tool provided by GitHub, offering seamless integration with the development workflow.

Procedure

- Enable GitHub Code Scanning in the repository settings.
- Configure the scanning workflow to trigger scans automatically on code pushes or pull requests.
- Customize analysis settings and specify security rulesets tailored to healthcare application requirements.
- Review scan results within the GitHub interface, including identified vulnerabilities, severity levels, and affected code locations.
- Collaborate with team members to address issues directly within pull requests or issues.

SAST in Practice Pt.2

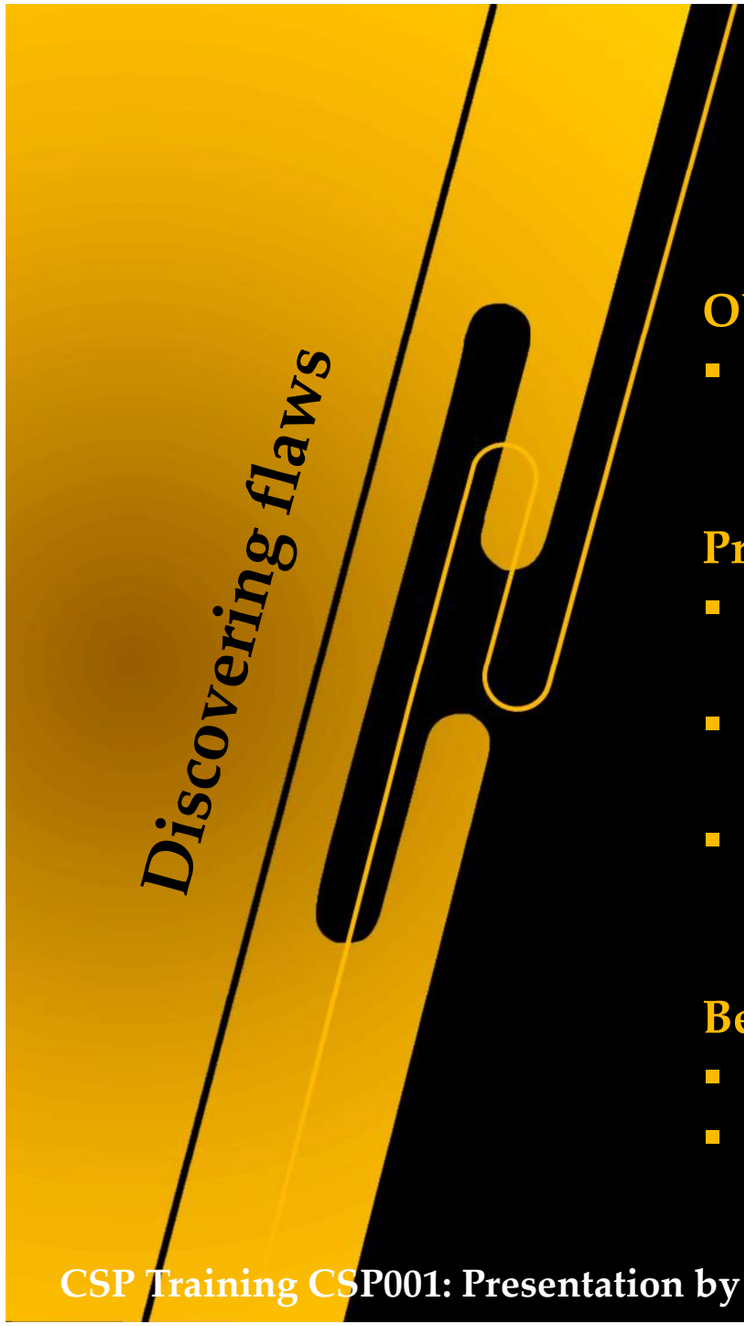
GitHub Code Scanning

Benefits

- Integration with GitHub repositories streamlines the development process, allowing developers to address security issues alongside code changes.
- Provides actionable insights into vulnerabilities, enabling developers to prioritize and remediate issues efficiently.
- Supports a wide range of programming languages and frameworks commonly used in healthcare software development.

Procedure

- Upon a code push or pull request, GitHub Code Scanning automatically triggers a scan of the codebase. The scan identifies vulnerabilities such as SQL injection, XSS, or insecure authentication methods within the source code.
- Detected vulnerabilities are reported within the GitHub interface, allowing developers to review and address them promptly.
- Developers collaborate to remediate vulnerabilities by making code changes and submitting pull requests for review.
- Once vulnerabilities are mitigated, code changes are merged back into the main branch, ensuring secure software deployment.



DAST in Practice Pt.1

Discovering flaws

OWASP ZAP (Zed Attack Proxy)

- OWASP ZAP is an open-source web application security scanner used to identify vulnerabilities in web applications.

Procedure

- Configure OWASP ZAP to act as a proxy between the browser and the target application.
- Set up automated scans or manually explore the application to identify vulnerabilities.
- Analyze scan results and generate reports detailing identified vulnerabilities.

Benefits

- Provides a user-friendly interface for both automated and manual testing.
- Offers advanced features for authentication testing and session management.

DAST in Practice Pt.2

DAST in Practice Pt.2

Burp Suite

Burp Suite is a widely used commercial DAST tool for web application security testing.

Procedure

- Configure Burp Suite to intercept and analyze HTTP requests and responses.
- Perform automated scans or manually explore the application to identify vulnerabilities.
- Utilize built-in tools for vulnerability confirmation and exploitation.

Benefits

- Offers extensive coverage of web vulnerabilities, including OWASP Top 10.
- Provides advanced features for penetration testing and vulnerability exploitation.

Common Weakness Enumeration (CWE)

CWE and Code Scanning

Common Weakness Enumeration (CWE) is a community-developed list of software and hardware weaknesses. CWE provides a common language for describing security vulnerabilities and weaknesses in software systems.

- Each CWE entry describes a specific type of weakness, such as buffer overflow, SQL injection, or cross-site scripting (XSS). CWE entries include detailed descriptions, examples, potential consequences, and mitigation strategies for addressing the weakness. Code scanning tools, such as GitHub Code Scanning, often use CWE as a reference to identify and classify vulnerabilities in source code.
- By mapping identified vulnerabilities to CWE entries, code scanning tools help developers understand the nature and severity of security issues in their codebases. CWE integration enables developers to prioritize remediation efforts based on the severity and impact of identified weaknesses.
- Code scanning reports typically include CWE identifiers alongside detected vulnerabilities, providing developers with actionable insights for improving code security.

Code Scanning Pt.3

GitHub Options Pt.1

Security

GitHub can identify the percentage of different programming languages used in a repository. GitHub analyzes the code within a repository and provides insights into the percentage of each programming language used.

- Security Policy: Define how users should report security vulnerabilities for this repository. By enabling a security policy, repository owners can define guidelines for reporting security vulnerabilities, enhancing collaboration and ensuring timely responses to potential threats.
- Security Advisories: View or disclose security advisories for this repository. GitHub allows the disclosure of security advisories related to the repository, ensuring transparency and enabling users to stay informed about potential security risks.
- Private Vulnerability Reporting: Allow users to privately report potential security vulnerabilities. Enabling private vulnerability reporting allows users to confidentially report security issues, fostering a secure environment for collaboration and prompt resolution of vulnerabilities.

Code Scanning Pt.3

GitHub Options Pt.2

Security

- Dependabot Alerts: Get notified when one of your dependencies has a vulnerability. Dependabot alerts notify repository owners about vulnerabilities in dependencies, helping them stay proactive in managing and updating dependencies to mitigate potential security risks.
- **Code Scanning Alerts:** Automatically detect common vulnerabilities and coding errors. GitHub's code scanning feature automatically detects common vulnerabilities and coding errors in the repository's codebase, empowering developers to identify and address potential security threats early in the development process.
- Secret Scanning Alerts: Get notified when a secret is pushed to this repository. Secret scanning alerts notify repository owners when sensitive information, such as access tokens or credentials, is accidentally pushed to the repository, helping prevent unauthorized access and data breaches.

Code Scanning Pt.1

GitHub Code Scanning - Part 1

Code scanning workflows in GitHub provide automated processes for integrating static code analysis into the software development lifecycle. Here's a description in bullets:

Configuration:

- Define code scanning workflows using YAML configuration files (e.g., .github/workflows/code-scanning.yml).
- Specify trigger conditions for initiating code scans, such as on push, pull request creation, or a scheduled basis.

Tool Selection:

- Choose the desired code scanning tool(s) to be used in the workflow.
- GitHub Code Scanning natively integrates with GitHub repositories, simplifying setup and configuration.

Workflow Steps:

- Define workflow steps to execute code scans using selected tools.
- Configure parameters such as scanning frequency, scanning depth, and scanning policies.

Code Scanning Pt.2

GitHub Code Scanning - Part 2

Scan Execution:

- Automated code scans are triggered based on defined workflow conditions.
- Scans analyze the source code for security vulnerabilities, coding errors, and other issues.

Results Analysis:

- Upon completion, scan results are generated and made available within the GitHub interface.
- Developers can access detailed reports highlighting identified vulnerabilities, their severity levels, and affected code locations.

Notification and Review:

- Optionally, configure notifications to alert developers and relevant stakeholders about scan results.
- Review scan findings within pull requests, issues, or dedicated dashboards, depending on the chosen workflow setup.

Code Scanning Pt.3

Code Scanning Pt.3

Resolution

Remediation and Mitigation:

- Collaborate with team members to address identified vulnerabilities promptly.
- Developers can prioritize and remediate issues directly within the GitHub interface, leveraging pull requests or issue tracking systems.

Continuous Improvement:

- Iterate on code scanning workflows based on feedback and evolving project requirements.
- Adjust scanning policies, tool configurations, and integration points to optimize code security practices.

Integration with CI/CD Pipelines:

- Seamlessly integrate code scanning workflows with existing CI/CD pipelines for comprehensive code validation and deployment automation.
- Ensure that code scanning results inform the software release process, preventing the introduction of vulnerabilities into production environments.

GitHub Workflows in Security Scanning

Definition

GitHub workflows in security scanning refer to automated processes or sequences of actions triggered by events in a GitHub repository, aimed at enhancing security by detecting vulnerabilities, coding errors, or sensitive information leaks in the codebase.

- GitHub workflows streamline security practices by automating security checks, ensuring code quality, and preventing security breaches before deployment.
- GitHub workflows can be configured to run various security scanning tools, such as CodeQL, to analyze code for potential vulnerabilities, security flaws, or compliance issues.
- CodeQL is a powerful static analysis tool provided by GitHub for identifying security vulnerabilities and coding errors in codebases. It allows developers to write custom queries to analyze code patterns and detect potential security risks.

Code Scanning using GitHub

Information

- **Continuous Scanning:** GitHub Code Scanning provides continuous analysis of code for vulnerabilities, ensuring proactive detection and mitigation of security risks.
- Code Scanning categorizes vulnerabilities into different severity levels. Identification of the relevant source-file and code-line where potential vulnerabilities were found.
- **Scheduled Scans:** Users can schedule the frequency of scans according to project requirements, allowing for regular and timely security checks on the codebase.
- **Details:** Extended explanation of the security issue, aiding in understanding and resolution.
- **CWE Matching:** Vulnerabilities are matched to Common Weakness Enumeration (CWE) standards, providing a standardized classification of weaknesses in software.

Thank you

Presenter: Stylianos Karagiannis (PDMFC, Portugal)

Please send all questions to:
stylianos.karagiannis@pdmfc.com