STATIC APPLICATION SECURITY TESTING

DevSecOps Basics

What is Static Application Security Testing?

The Static Application Security Testing (SAST) is done to identify the possible vulnerabilities or security issues in non-running source code by using techniques likeTaint Analysis and Data Flow Analysis.

The following components are there in this phase:

- SAST tools i.e. Flawfinder, Graudit, Bandit, Spotbugs, SonarQube

People involved: Developers

External sources

- What is Static Code Analysis? https://owasp.org/www-community/controls/Static_Code_Analysis
- What is Taint Analysis? https://dzone.com/articles/what-is-taint-analysis-and-why-should-i-care
- What is Data Flow testing? https://www.testbytes.net/blog/data-flow-testing
- More Static Code Analysis techniques https://www.geeksforgeeks.org/types-of-static-analysis-methods/

Why is it important in DevSecOps?

The Static Application Security Testing phase can be used to identify security issues. For example, taint analysis can identify the variables that can handle the user input and check if vulnerability like buffer overflow can occur.

What will you learn in this section?

The user will learn to perform the following tasks

- Analyze the code of provided web applications for issues

Tools Covered

- Flawfinder
- Graudit
- Bandit
- Spotbugs
- SonarQube

Labs

- Flawfinder: Statically Scanning C code
  - A Kali machine is provided to the user with the FlawFinder installed on it. The source code for three sample web applications is provided in the home directory of the root user.
    Objective: Scan the source code using FlawFinder utility and find the security issues!
- Graudit: Hunting Sensitive Information
  - A Kali machine is provided to the user with Graudit installed on it. The source code for a sample web application is provided in the home directory of the root user.
    Objective: Scan the source code using Graudit utility and find the security issues!
- Bandit: Scanning Python Code for Issues
  - A Kali machine is provided to the user with Bandit installed on it. The source code for the web application is provided in the home directory of the root user.
    Objective: Scan the code using bandit utility and find the security issues!

root user.

Objective: Find bugs in the source code of web applications using the Spotbugs tool.

- SonarQube: Continuous Code Quality Monitoring
  - A Kali machine and a SonarQube server are provided in the lab. The SonarQube scanner client is installed on the Kali machine that will scan the web application and push the results to the SonarQube server machine where it can be accessed by the user in the form of reports.

    Objective: Use SonarQube to perform the static code analysis on the application and find issues!

Flawfinder: Statically Scanning C code

Start

Graudit: Hunting Sensitive Information

Start

Bandit: Scanning Python Code for Issues

Start

Spotbugs: Finding Bugs in Java Code

Start

SonarQube: Continuous Code Quality ...

Start

Brakeman: Finding Bugs in Ruby on Rails

Start

NodeJs Scan: Finding Bugs in NodeJs Code

Start

CodeSake dawn: Finding bugs in Ruby Code

Start

RIPS: Statically Scanning PHP Code

Start