**Name:** Pranav Dilip Hingankar

**Batch:** T6

**PRN:** 2020BTECS00040

Assignment 7 – Source code testing using tools

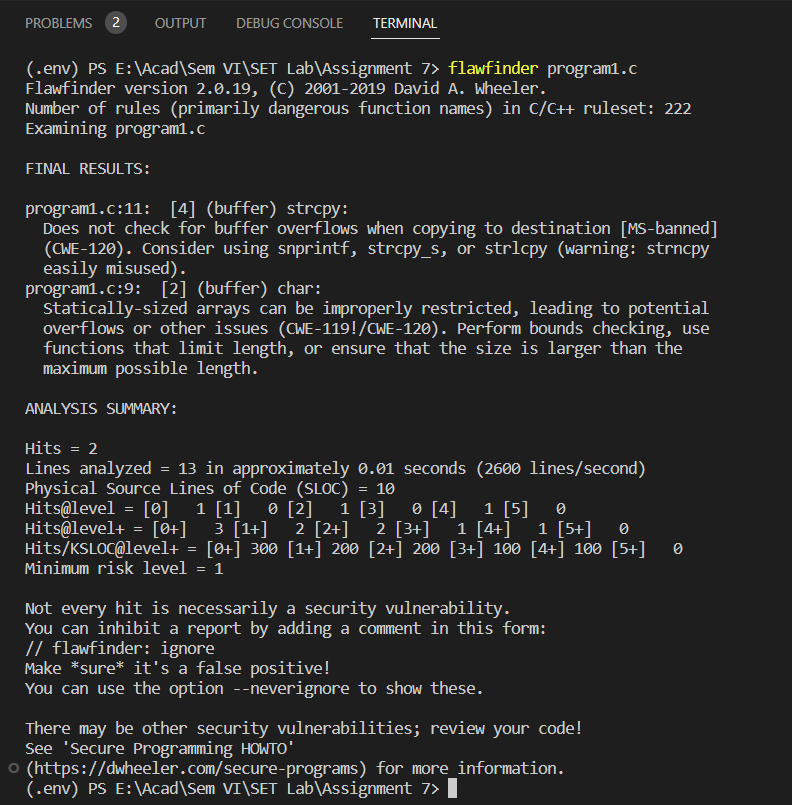
1. What is Source code analysis? What is its importance?

🡪

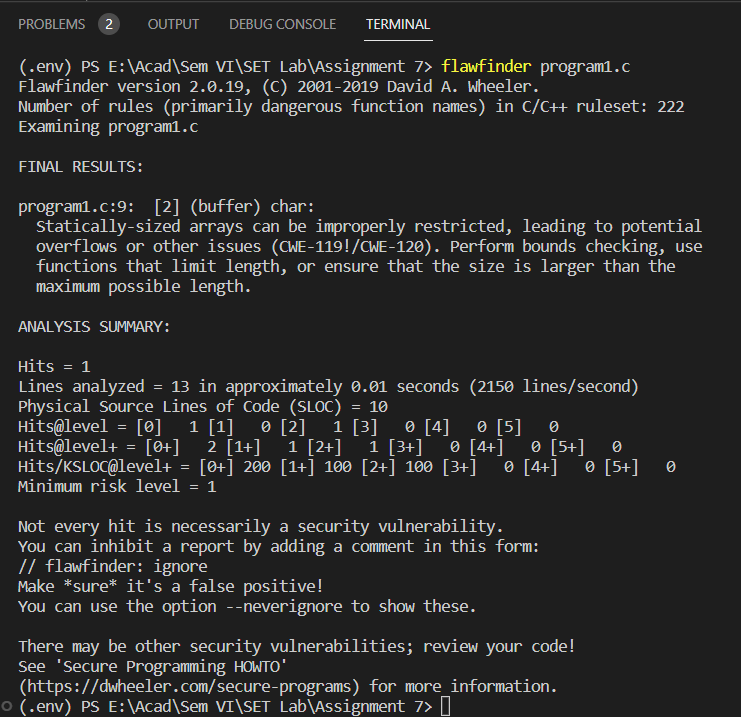
Source code analysis, also known as static code analysis, is the process of examining the source code of a software application without executing it, in order to identify potential issues, errors, vulnerabilities, and other quality-related aspects. The analysis is performed using automated tools that scan the codebase and detect patterns or structures that indicate potential problems.

1. Improved code quality: Source code analysis can help identify coding standards violations, code smells, and other issues that can degrade the quality of the codebase.
2. Increased security: Source code analysis can help identify security vulnerabilities in the codebase, such as potential buffer overflows, injection attacks, or other issues that can be exploited by attackers.
3. Enhanced maintainability: Source code analysis can help identify complex or hard-to-maintain code structures, making it easier for developers to update, refactor, or extend the codebase as needed.
4. Faster development: By catching potential issues early, source code analysis can help reduce the time and effort required for debugging and testing, allowing developers to focus on building new features or improving existing ones.
5. Below are some of the important open-source tools used in testing the source code, provide the information of below tools with respect to
6. VisualCodeGrepper
   * 1. Owner/developer: Developed by Joris van de Vis for CodeClimber.net.
     2. Developed in which language: Written in Python.
     3. Brief information/introduction: VisualCodeGrepper is an open-source static code analysis tool that searches for patterns of interest in source code using regular expressions. It can detect issues such as potential security vulnerabilities, coding standard violations, and code smells.
     4. Language support: Supports source code written in multiple languages including C, C++, C#, Java, JavaScript, PHP, Python, Ruby, and more.
     5. Advantages: Easy to install and use, customizable search patterns, support for multiple languages, fast scanning, and integration with external tools such as SonarQube and Jenkins.
     6. Disadvantages: Limited functionality compared to more advanced static code analysis tools, may produce false positives, and lack of comprehensive documentation.
7. Rips
   * 1. Owner/developer: Developed by Johannes Dahse.
     2. Developed in which language: Written in PHP.
     3. Brief information/introduction: RIPS is a free and open-source tool for detecting security vulnerabilities in PHP applications. It uses static code analysis techniques to detect issues such as SQL injection, cross-site scripting (XSS), and file inclusion vulnerabilities.
     4. Language support: Supports PHP source code.
     5. Advantages: Easy to install and use, comprehensive detection of security vulnerabilities, customizable analysis rules, and integration with popular development tools such as Jenkins and GitLab.
     6. Disadvantages: Limited language support, may produce false positives, and lack of support for other programming languages.
8. Brakeman
   * 1. Owner/developer: Developed by Justin Collins.
     2. Developed in which language: Written in Ruby.
     3. Brief information/introduction: Brakeman is an open-source static code analysis tool for Ruby on Rails applications. It can detect security vulnerabilities such as SQL injection, cross-site scripting (XSS), and more. It also generates reports that provide detailed information about the issues found.
     4. Language support: Supports Ruby on Rails source code.
     5. Advantages: Easy to install and use, comprehensive detection of security vulnerabilities, customizable analysis rules, and integration with popular development tools such as Jenkins and GitHub.
     6. Disadvantages: Limited language support, may produce false positives, and lack of support for other programming languages.
9. Flawfinder
   * 1. Owner/developer: Developed by David A. Wheeler.
     2. Developed in which language: Written in Perl.
     3. Brief information/introduction: Flawfinder is an open-source static code analysis tool that scans C/C++ code for potential security vulnerabilities such as buffer overflows, format string vulnerabilities, and more. It generates reports that provide detailed information about the issues found.
     4. Language support: Supports C and C++ source code.
     5. Advantages: Easy to install and use, comprehensive detection of security vulnerabilities, customizable analysis rules, and integration with popular development tools such as Jenkins and Git.
     6. Disadvantages: Limited language support, may produce false positives, and lack of support for other programming languages.
10. Bandit
    * 1. Owner/developer: Developed by OpenStack Security Group.
      2. Developed in which language: Written in Python.
      3. Brief information/introduction: Bandit is an open-source static code analysis tool that scans Python code for potential security vulnerabilities such as SQL injection, cross-site scripting (XSS), and more. It generates reports that provide detailed information about the issues found.
      4. Language support: Supports Python source code.
      5. Advantages: Easy to install and use, comprehensive detection of security vulnerabilities, customizable analysis rules, and integration with popular development tools such as Jenkins and GitHub.
      6. Disadvantages: Limited language support, false positives, limited functionality, Lack of support for complex applications.
11. Perform source code testing using **Flawfinder** for the code written in ‘c’ and ‘cpp’ language given below
12. Program1.c

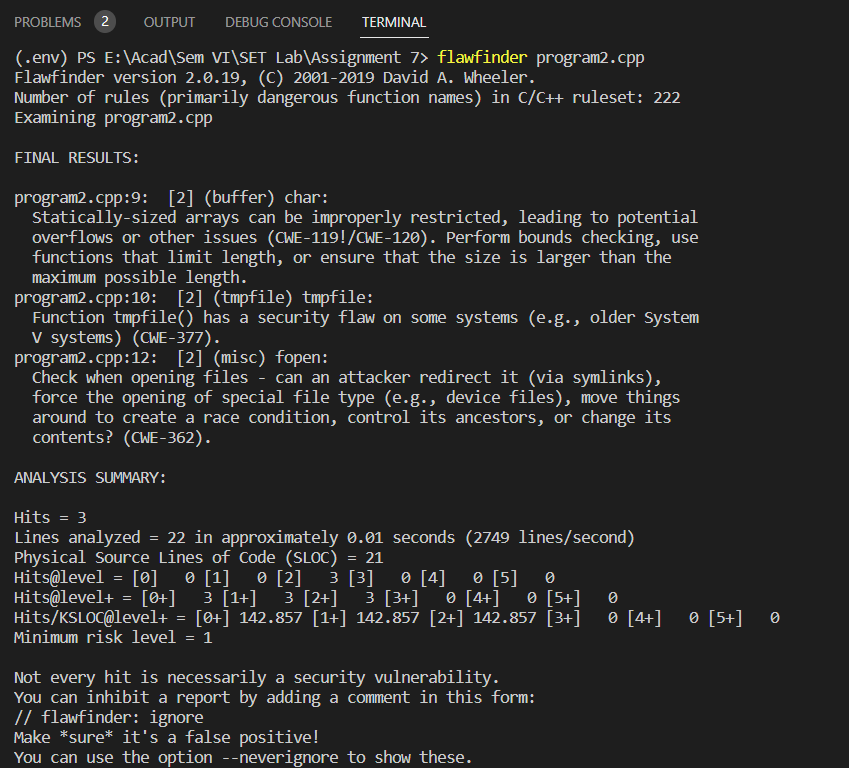
Initial hits - 2



After updating code as per suggestions hits reduced to 1

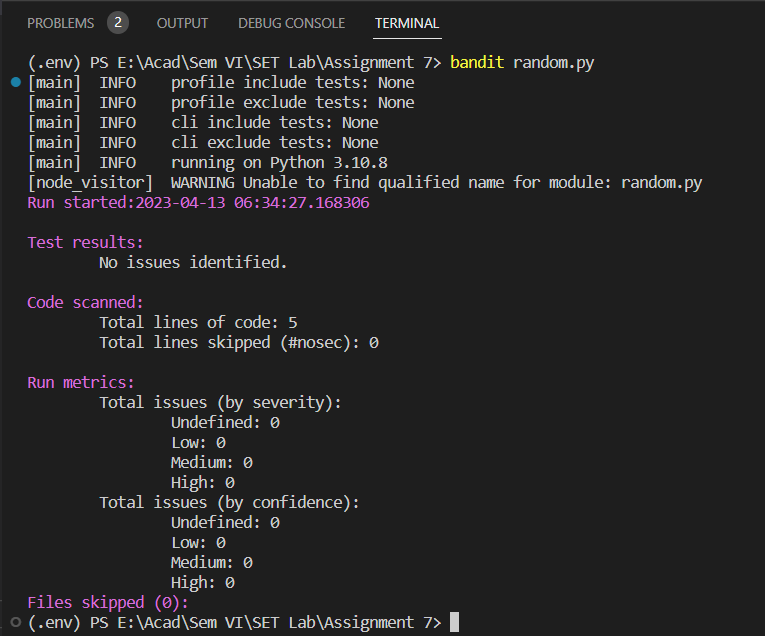


1. Program2.cpp



Q 4. Perform source code testing using **Bandit** for your code written in ‘python’ language (use your previous code) for any security flaws

After performing analysis create a report which will contain below points



a. Number of hits - 0

b. Potential risks - 0

c. Suggested alternatives for these risks

d. Updating the code as per suggestions

e. Re-execution of code after updating the changes.