Software Engineering Tools Lab

Assignment No-7

Module 5- Source code testing using tools

PRN: 2020BTECS00079

Name: Saurabh Khadsang

Q1.What is Source code analysis? What is its importance?

Source code analysis (also known as static code analysis or source code scanning) is the process of examining source code to identify potential vulnerabilities, bugs, or other issues. This is typically done using automated tools that analyze the code for patterns and common mistakes, such as memory leaks, buffer overflows, or insecure input validation.

The importance of source code analysis lies in its ability to detect potential security vulnerabilities and other issues early in the software development lifecycle. By identifying these issues early, developers can fix them before the software is deployed, reducing the risk of security breaches and other problems down the line.

Source code analysis can also help to improve code quality and maintainability, by identifying areas of the code that are hard to read, poorly structured, or likely to cause errors or performance issues.

Q 2. Below are the some important open source tools used in testing the source code,

provide the information of below tools with respect to

a. Owner/ developer

b. Developed in which language

c. Brief information/introduction

d. Language support (applicable for source code written in language)

e. Advantages

f. Disadvantages

Source code analysis tools-

I. VisualCodeGrepper

a. Owner/ Developer: VisualCodeGrepper is an open-source tool developed by Nick Lehmann.

b. Developed in which language: VisualCodeGrepper is written in Python.

c. Brief information/ introduction: VisualCodeGrepper is a command-line tool used for source code analysis and pattern matching. It scans source code files and searches for specific patterns, such as regular expressions or keywords, to identify potential vulnerabilities or other issues. VisualCodeGrepper can be customized to support multiple programming languages and can output results in various formats, including CSV, XML, and JSON.

d. Language support: VisualCodeGrepper supports source code written in multiple programming languages, including C, C++, Java, Python, Ruby, Perl, and more.

e. Advantages: VisualCodeGrepper is a powerful and flexible tool that can be customized to suit a wide range of source code analysis needs. It can quickly identify potential vulnerabilities and other issues, allowing developers to fix them before the software is deployed. Additionally, it can support multiple programming languages, making it a useful tool for teams working with different languages.

f. Disadvantages: VisualCodeGrepper is a command-line tool, which may not be as user-friendly as some graphical user interface (GUI) tools. It requires some knowledge of the command line and may be more challenging to set up and configure than some other tools. Additionally, while it can support multiple programming languages, it may not be as specialized for any one language as some other tools that are specifically designed for that language.

II. Rips

a. Owner/ Developer: RIPS is a software tool developed by RIPS Technologies GmbH, a company based in Bochum, Germany.

b. Developed in which language: RIPS is written in PHP.

c. Brief information/ introduction: RIPS is a web-based tool used for source code analysis and vulnerability detection in PHP applications. It scans source code files for potential vulnerabilities, such as SQL injection, cross-site scripting (XSS), and file inclusion vulnerabilities. It also provides a dashboard that allows developers to manage and track their findings.

d. Language support: RIPS is specifically designed to analyze PHP applications.

e. Advantages: RIPS is a specialized tool for PHP applications, which makes it particularly effective at detecting PHP-specific vulnerabilities. It can scan large codebases quickly and provides detailed reports that allow developers to prioritize and address issues efficiently. Additionally, it can be integrated into the development process, allowing developers to catch vulnerabilities early in the development cycle.

f. Disadvantages: Since RIPS is specifically designed for PHP applications, it may not be as effective at detecting vulnerabilities in other programming languages. Additionally, while it provides detailed reports, some users may find the interface less user-friendly than other tools. Finally, being a web-based tool, it requires a server to host it, which may not be suitable for all organizations.

III. Brakeman

a. Owner/ Developer: Brakeman is an open-source tool developed by Justin Collins.

b. Developed in which language: Brakeman is written in Ruby.

c. Brief information/ introduction: Brakeman is a security analysis tool for Ruby on Rails applications. It scans source code files for potential security vulnerabilities, such as SQL injection, cross-site scripting (XSS), and cross-site request forgery (CSRF) attacks. It provides detailed reports on potential vulnerabilities, allowing developers to prioritize and address issues efficiently.

d. Language support: Brakeman is specifically designed to analyze Ruby on Rails applications.

e. Advantages: Brakeman is a specialized tool for Ruby on Rails applications, making it particularly effective at detecting vulnerabilities specific to that language and framework. It can scan large codebases quickly and provides detailed reports that allow developers to prioritize and address issues efficiently. Additionally, it can be integrated into the development process, allowing developers to catch vulnerabilities early in the development cycle.

f. Disadvantages: Since Brakeman is specifically designed for Ruby on Rails applications, it may not be as effective at detecting vulnerabilities in other programming languages. Additionally, some users may find the interface less user-friendly than other tools, and it may not be suitable for organizations that do not use Ruby on Rails.

IV. Flawfinder

a. Owner/ Developer: Flawfinder is an open-source tool developed by David A. Wheeler.

b. Developed in which language: Flawfinder is written in Python.

c. Brief information/ introduction: Flawfinder is a static analysis tool used for identifying potential security vulnerabilities in source code. It scans source code files for common programming errors that could lead to vulnerabilities, such as buffer overflows, format string vulnerabilities, and SQL injection vulnerabilities. It provides detailed reports on potential vulnerabilities, allowing developers to prioritize and address issues efficiently.

d. Language support: Flawfinder can be used to analyze source code written in various programming languages, including C, C++, Java, and Python.

e. Advantages: Flawfinder is a lightweight and easy-to-use tool that can be used to analyze source code written in multiple programming languages. It provides detailed reports that allow developers to prioritize and address issues efficiently. Additionally, it can be integrated into the development process, allowing developers to catch vulnerabilities early in the development cycle.

f. Disadvantages: Flawfinder is a general-purpose tool that may not be as effective at detecting language- or framework-specific vulnerabilities. It may also generate false positives or miss some vulnerabilities, which could lead to security risks if not properly addressed. Additionally, some users may find the interface less user-friendly than other tools.

V. Bandit

a. Owner/ Developer: Bandit is an open-source tool developed by the OpenStack Security Project.

b. Developed in which language: Bandit is written in Python.

c. Brief information/ introduction: Bandit is a security analysis tool used for identifying potential security vulnerabilities in Python applications. It scans source code files for potential security issues, such as hard-coded passwords, SQL injection vulnerabilities, and cross-site scripting (XSS) vulnerabilities. It provides detailed reports on potential vulnerabilities, allowing developers to prioritize and address issues efficiently.

d. Language support: Bandit is specifically designed to analyze Python applications.

e. Advantages: Bandit is a specialized tool for Python applications, making it particularly effective at detecting vulnerabilities specific to that language. It can scan large codebases quickly and provides detailed reports that allow developers to prioritize and address issues efficiently. Additionally, it can be integrated into the development process, allowing developers to catch vulnerabilities early in the development cycle.

f. Disadvantages: Since Bandit is specifically designed for Python applications, it may not be as effective at detecting vulnerabilities in other programming languages. Additionally, some users may find the interface less user-friendly than other tools, and it may not be suitable for organizations that do not use Python.

Q 3. Perform source code testing using Flawfinder for the code written in 'c' and

‘cpp’ language given below

Link- https://github.com/sidp1991/SETAssignment

Note-use files program1.c and program2.cpp present on above link.

After performing analysis create a report which will contain below points

a. Number of hits

b. Potential risks

c. Suggested alternatives for these risks

d. Updating the code as per suggestions

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

```
PS F:\z.DOdo\WCE Assignment\Sem 6\SAT\7\SETAssignment> flawfinder program1.c program2.cpp
Flawfinder version 2.0.19, (C) 2001-2019 David A. Wheeler.
Number of rules (primarily dangerous function names) in C/C++ ruleset: 222
Examining program1.c
Examining program2.cpp

FINAL RESULTS:

program1.c:11:  [4] (buffer) strcpy:
  Does not check for buffer overflows when copying to destination [MS-banned]
  (CWE-120). Consider using snprintf, strcpy_s, or strlcpy (warning: strncpy
  easily misused).
program1.c:9:  [2] (buffer) char:
  Statically-sized arrays can be improperly restricted, leading to potential
  overflows or other issues (CWE-119!/CWE-120). Perform bounds checking, use
  functions that limit length, or ensure that the size is larger than the
  maximum possible length.
program2.cpp:9:  [2] (buffer) char:
  Statically-sized arrays can be improperly restricted, leading to potential
  overflows or other issues (CWE-119!/CWE-120). Perform bounds checking, use
  functions that limit length, or ensure that the size is larger than the
  maximum possible length.
program2.cpp:12:  [2] (misc) fopen:
  Check when opening files - can an attacker redirect it (via symlinks),
  force the opening of special file type (e.g., device files), move things
  around to create a race condition, control its ancestors, or change its
  contents? (CWE-362).

ANALYSIS SUMMARY:

Hits = 4
Lines analyzed = 38 in approximately 0.01 seconds (4266 lines/second)
Physical Source Lines of Code (SLOC) = 29
Hits@level = [0]   1 [1]   0 [2]   3 [3]   0 [4]   1 [5]   0
Hits@level+ = [0+]   5 [1+]   4 [2+]   4 [3+]   1 [4+]   1 [5+]   0
Hits/KSLOC@level+ = [0+] 172.414 [1+] 137.931 [2+] 137.931 [3+] 34.4828 [4+] 34.4828 [5+]   0
Minimum risk level = 1

Not every hit is necessarily a security vulnerability.
You can inhibit a report by adding a comment in this form:
// flawfinder: ignore
Make *sure* it's a false positive!
You can use the option --neverignore to show these.

There may be other security vulnerabilities; review your code!
See 'Secure Programming HOWTO'
(https://dwheeler.com/secure-programs) for more information.
PS F:\z.DOdo\WCE Assignment\Sem 6\SAT\7\SETAssignment>
```

a. Number of hits: Flawfinder found a total of 4 hits in the code - 1 hit with level 0, 3 hits with level 2.

b. Potential risks: The hits indicate potential security risks that could be exploited by attackers. The risks include:

1. The use of the "strcpy" function in program1.c without checking for buffer overflows could lead to a buffer overflow vulnerability (CWE-120).
2. The use of statically-sized arrays in program1.c and program2.cpp could lead to potential buffer overflows or other issues (CWE-119!/CWE-120).

3. The use of the "fopen" function in program2.cpp without proper checks could lead to file handling vulnerabilities (CWE-362).

c. Suggested alternatives for these risks:

1. Instead of using "strcpy", use "snprintf", "strcpy_s", or "strlcpy" to copy strings, which provide additional checks for buffer overflows.
2. Instead of using statically-sized arrays, perform bounds checking, use functions that limit length, or ensure that the size is larger than the maximum possible length.
3. When using the "fopen" function, ensure that the file being opened is trusted and that the path is secure. Use the "fopen_s" function, which provides additional security checks.

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

// Driver code
int main()
{
    char *temp = (char*)malloc(6*sizeof(char));
    char str[] = "hello";
    strncpy_s(temp, 6, str, 5);
    temp[5] = '\0';
    printf("%s", temp);
    free(temp);
    return 0;
}
```

```
PS F:\z.DOdo\WCE Assignment\Sem 6\SAT\7\SETAssignment> flawfinder program1.c
Flawfinder version 2.0.19, (C) 2001-2019 David A. Wheeler.
Number of rules (primarily dangerous function names) in C/C++ ruleset: 222
Examining program1.c

FINAL RESULTS:


ANALYSIS SUMMARY:

No hits found.
Lines analyzed = 31 in approximately 0.01 seconds (4750 lines/second)
Physical Source Lines of Code (SLOC) = 13
Hits@level = [0]   1 [1]   0 [2]   0 [3]   0 [4]   0 [5]   0
Hits@level+ = [0+]   1 [1+]   0 [2+]   0 [3+]   0 [4+]   0 [5+]   0
Hits/KSLOC@level+ = [0+] 76.9231 [1+]   0 [2+]   0 [3+]   0 [4+]   0 [5+]   0
Minimum risk level = 1

There may be other security vulnerabilities; review your code!
See 'Secure Programming HOWTO'
(https://dwheeler.com/secure-programs) for more information.
PS F:\z.DOdo\WCE Assignment\Sem 6\SAT\7\SETAssignment>
```

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

b. Potential risks

c. Suggested alternatives for these risks

d. Updating the code as per suggestions

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

random.py

```python
import random
randomlist = random.sample(range(1, 9),8)
list=['ARS','SDP','KPK','MAS','ASP','PDM','SSS','SSR']

print(list)
print(randomlist)
```

```
PS F:\z.DOdo\WCE Assignment\Sem 6\SAT\7\SETAssignment> bandit .\random.py
[main]  INFO    profile include tests: None
[main]  INFO    profile exclude tests: None
[main]  INFO    cli include tests: None
[main]  INFO    cli exclude tests: None
[main]  INFO    running on Python 3.7.1
Run started:2023-04-22 19:13:33.792133

Test results:
        No issues identified.

Code scanned:
        Total lines of code: 5
        Total lines skipped (#nosec): 0

Run metrics:
        Total issues (by severity):
                Undefined: 0
                Low: 0
                Medium: 0
                High: 0
        Total issues (by confidence):
                Undefined: 0
                Low: 0
                Medium: 0
                High: 0
Files skipped (0):
```