Sairam Soundararajan

Graded Assignment: Homework 3

University of Maryland Global Campus

SDEV 325 – Detecting Software Vulnerabilities

# Executive Summary

Homework 3 is focused on demonstrating risky resource management. I decided to test out two of the top 25 vulnerabilities, such as Integer Overflow or Wraparound and XSS (Cross Site Scripting). While I was able to use AWS Cloud 9 for the Integer Overflow, I could only use Pycharm for Cross Site Scripting since I had trouble viewing the sites on AWS Cloud 9. Because I had already done path traversal, a risky resource management vulnerability, for the last assignment, I decided to do cross site scripting in exchange.

For the Integer Overflow/Wraparound, I successfully mitigated the code by slight alteration of the for loop. It did not require much modification to handle the vulnerability. For the Cross Site Scripting, websites using Python Flask are only vulnerable if auto-escaping is disabled. So rather than figuring out how to mitigate the vulnerability, I had to figure out how to make the code vulnerable in order to understand how hackers can exploit a website.

# Example 1 – CWE-190: Integer Overflow or Wraparound

## Overview

For this vulnerability, I used Java. This application performs a calculation that produces an integer overflow, which means the value exceeds the max possible value of an integer. This program specifically detects an overflow in order to know how to handle the overflow.



## Analysis of the Vulnerability

In the vulnerable code, the program displays an integer value that is beyond the maximum value of the range of a typical integer. This value is too large and can possibly result in security consequences.

Text

Description automatically generated

## Mitigation

In the mitigated code, the value is not incremented as a part of the conditions of the for loop and is instead initialized to add an exact value inside of the loop. With this minor change, the program displays an integer overflow exception and causes the test run to terminate.

Text

Description automatically generated

# Example 2 – CWE-79: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')

## Overview

For this vulnerability, I used Python. Because I could not run the website on AWS Cloud 9, I used Pycharm. This program runs a website in which hackers may exploit by typing in malicious code as the user input.

Text

Description automatically generated

## Analysis of the Vulnerability

In the vulnerable code, I disabled auto-escaping. The exclusion of auto-escaping causes the website to be prone to XSS attacks where hackers can input any code to change or alter the website. One of the ways the code is exploited is by typing in a piece of code where a dialogbox occurs showing the malicious user’s message.

Graphical user interface, text, application, chat or text message

Description automatically generated

## Mitigation

Because Flask uses render\_template and auto-escaping, the website is secure and thus, mitigated by default. It is only when we disable auto-escaping that the code is then vulnerable to XSS attacks. Unlike the vulnerable code, this code will display the malicious actor’s message as just an ordinary input, not having any negative effect on the website.

Graphical user interface, text, application, email

Description automatically generated

# References

*Cross-site Scripting*. CWE. (n.d.). Retrieved April 10, 2023, from https://cwe.mitre.org/data/definitions/79.html

*Integer Overflow or Wraparound*. CWE. (n.d.). Retrieved April 10, 2023, from https://cwe.mitre.org/data/definitions/190.html

Krimgen, W. by: M. (2021, May 22). *Overflow and underflow in Java*. Baeldung. Retrieved April 10, 2023, from https://www.baeldung.com/java-overflow-underflow

*Security considerations*. Security Considerations - Flask Documentation (1.1.x). (n.d.). Retrieved April 11, 2023, from https://flask.palletsprojects.com/en/1.1.x/security/#xss

*XSS prevention for Flask*. Semgrep. (n.d.). Retrieved April 11, 2023, from https://semgrep.dev/docs/cheat-sheets/flask-xss/