Sairam Soundararajan

Graded Assignment: Homework 2

University of Maryland Global Campus

SDEV 325 – Detecting Software Vulnerabilities

# Executive Summary

Homework 2 is focused on demonstrating insecure interaction between components. I chose to test out two of the top 25 vulnerabilities, such as Command Line Injection and Path Traversal. I was not able to use AWS Cloud 9 to work on the examples due to the need to access or navigate files on my PC. So, I used Pycharm to test path traversal on Python, while I used IntelliJ to test command injection on Java.

For the path traversal, I added functions in python where the program requires the user to type in the path for the file with the file name included. This will validate the path if the actual path matches the expected path and will invalidate it if they don’t. For the command injection, I added an extra layer of security by utilizing another java built-in library along with another try-catch statement in order to reduce vulnerability in the code.

# Example 1 – [CWE-77 and Improper Neutralization of Special Elements used in a Command ('Command Injection')]

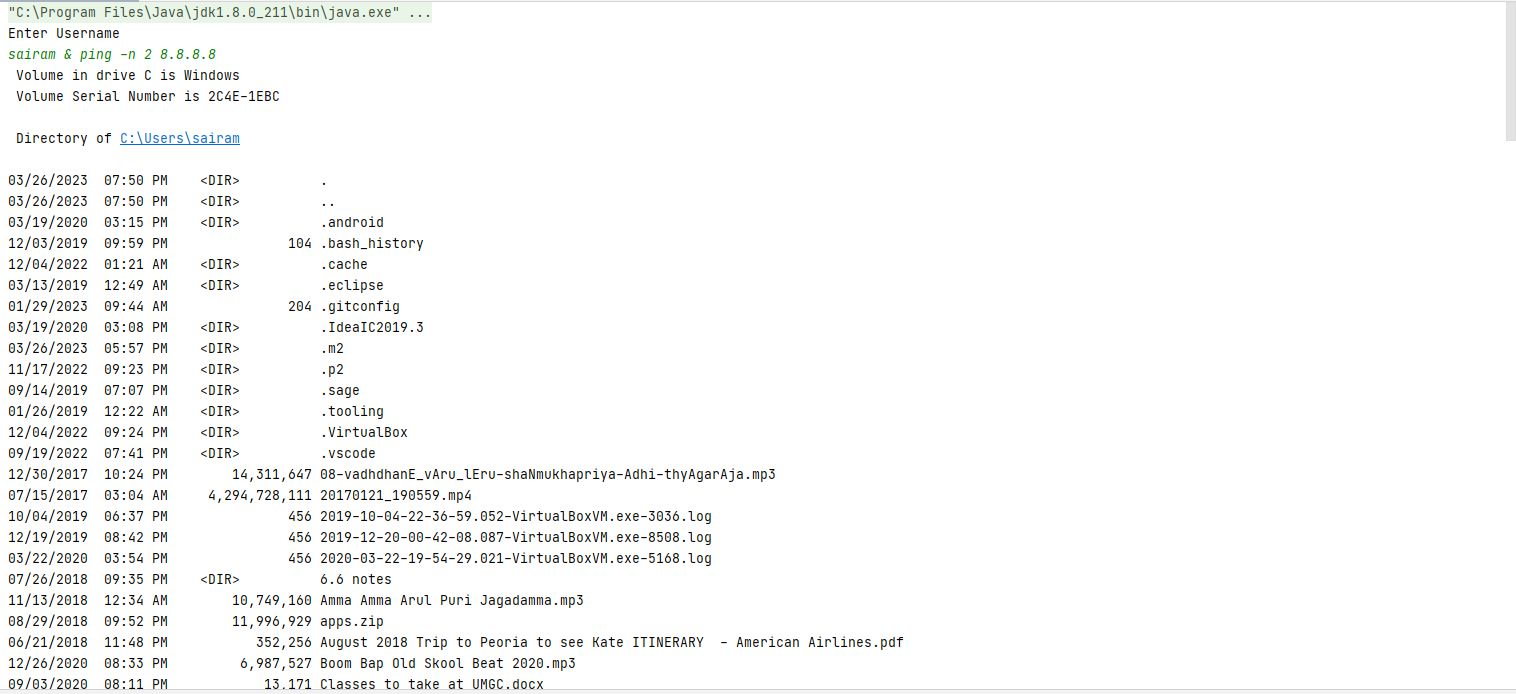
## Overview

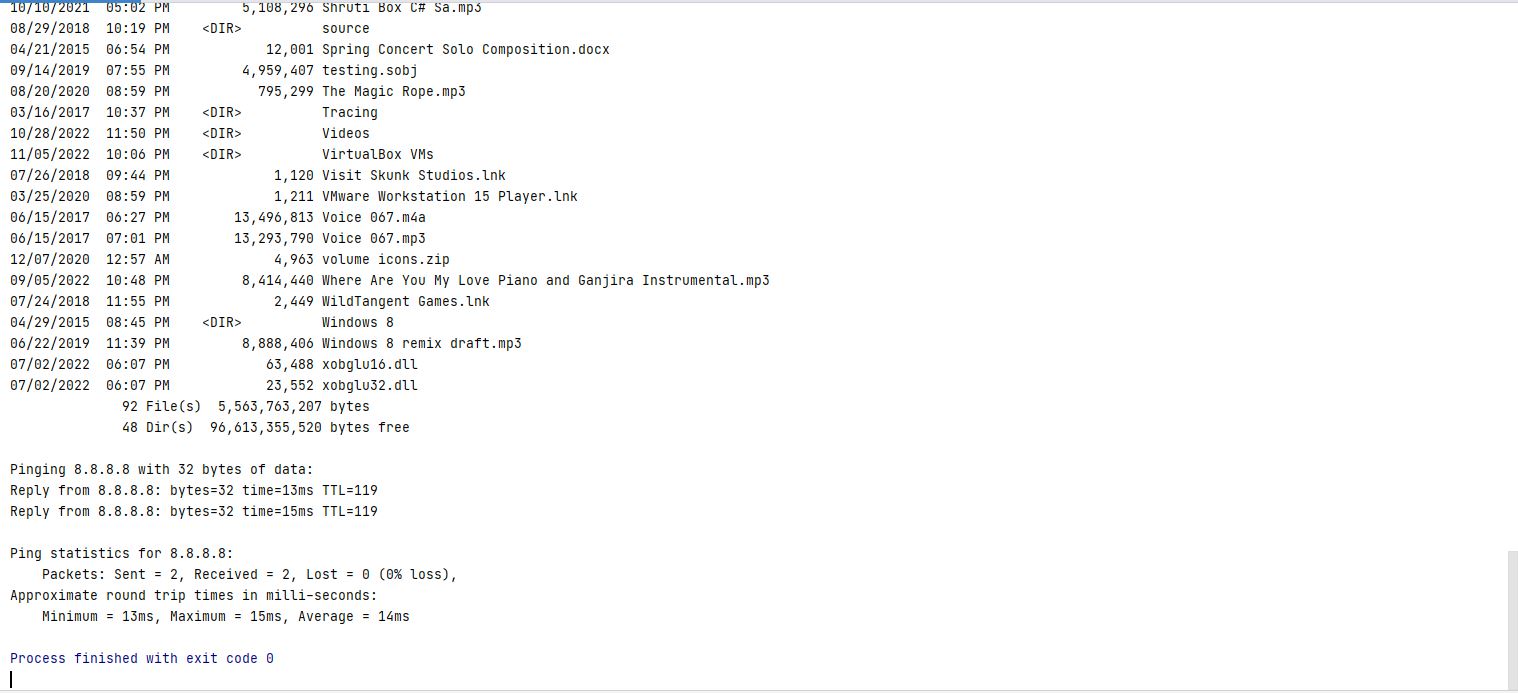
In Java, the program uses Command Injection, where a malicious user attempts to execute system commands on the system hosting the application. Specifically, the program allows you to enter a username so you can view all the files the user has while seeing how much storage is used and left.



## Analysis of the Vulnerability

In the vulnerable code, a malicious actor may type in the username followed by a sequence of characters in order to ping any files on the system. Additionally, they can also delete files or shutdown the system.





## Mitigation

With the mitigation code, I added an extra layer of security by using another library of Java rather than system commands. Alongside, I added more try-catch statements for the program to display an error message when attempting to alter another user’s files. As a result, the malicious actor is unable to alter the other user’s files in any way.



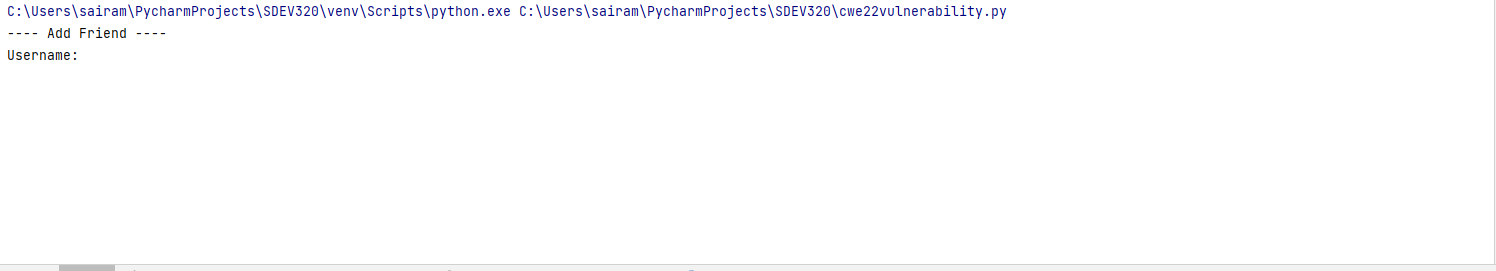


# Example 2 – [CWE-22 and Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')]

## Overview

[A paragraph describing your example, what the application does, what language the application is written in, and you may include a screenshot of the application’s startup screen]

In Python, the program focuses on Path Traversal where the malicious user may try to traverse a directory. A user can type in a name of a text file to access it.

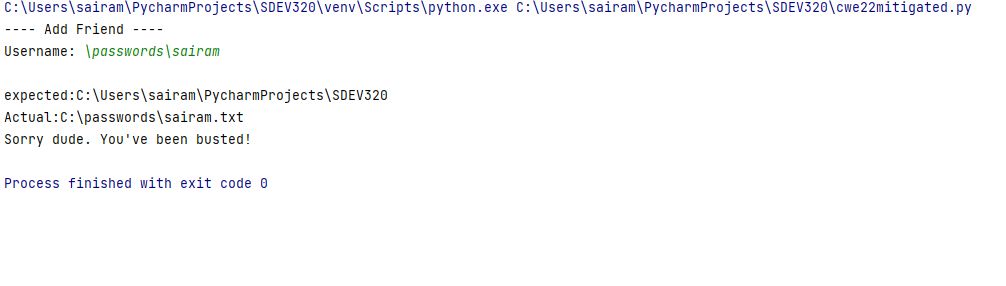


## Analysis of the Vulnerability

In the vulnerable code, not only is a user able to access the file but is also able to see what is written in the text file. This allows the malicious actor to potentially invade another user’s personal/confidential information. Subsequently, they may ambush the user in a financial or life-threatening way. When I type in my name, you can see personal information about me.



## Mitigation

In the mitigated code, however, the program requires the user to type in a path where the file may be located. Based on the input, the program will compare the user’s path (actual path) to the expected path (specified in the code). Unlike the vulnerable code, the mitigated code displays the expected path vs what the user typed in, following a message saying “Sorry dude. You’ve been busted!”

# References

*Command injection in java: Examples and prevention*. StackHawk. Retrieved March 28, 2023, from https://www.stackhawk.com/blog/command-injection-java/

*Command Injection*. CWE. (n.d.). Retrieved March 28, 2023, from https://cwe.mitre.org/data/definitions/77.html

*Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')*. CWE. (n.d.). Retrieved March 28, 2023, from https://cwe.mitre.org/data/definitions/22.html

Guy, A. (2019, June 4). *Preventing directory traversal in Python*. Guy Rutenberg. Retrieved March 28, 2023, from https://www.guyrutenberg.com/2013/12/06/preventing-directory-traversal-in-python/