Graded Assignment: Homework 4

University of Maryland Global Campus

SDEV 325 – Detecting Software Vulnerabilities

# Executive Summary

Homework 4 is focused on demonstrating porous defenses. I decided to work on Missing Encryption of Sensitive Data and Use of Hard-Coded Credentials. Unlike previous projects, I was able to use AWS for both vulnerabilities.

For the Missing Encryption of Sensitive Data, I successfully mitigated the code by encrypting the credentials that get created and stored on a text file. It required using a hash library. For the Use of Hard-Coded Credentials, information such as usernames, passwords, and, sometimes, cryptographic keys (which I did not use) are declared and initialized in the code. As a result, the information can be seen by anyone looking at the code. This was mitigated by prompting the user to create a username and password to get stored in a text file versus being initialized in the code.

# Example 1 – [CWE-798: Use of Hard-Coded Credentials]

## 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]

For this program, I am working with hard-coded credentials on Java. Based on the vulnerability name, user credentials are declared and initialized inside the code.

Text

Description automatically generated

## Analysis of the Vulnerability

Because the username and password are already declared in the credentials, it does not require the user to type them in. Per the photo, the credentials are pre-displayed on the screen. This vulnerability can be used by an attacker to bypass authentication.

Text

Description automatically generated

## Mitigation

For the mitigation, I decided to remove the hard-coded credentials that were originally declared in the code by prompting the user to input their username and password and then login. This username and password get stored in a text file. Notice in the following snapshot, the program does not display any credentials, but rather prompts the user to create their credentials.

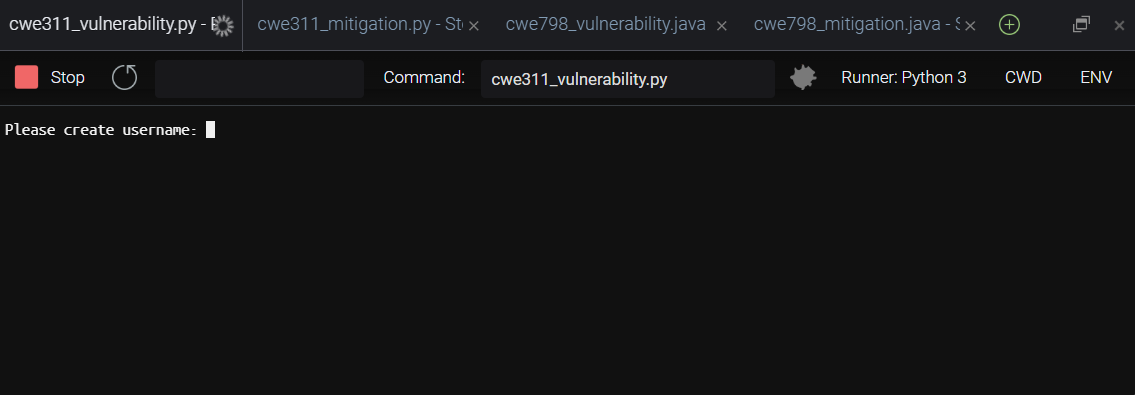
Text

Description automatically generated

# Example 2 – [CWE-311: Missing Encryption of Sensitive Data]

## Overview

This program uses Python to deal with missing encryption of sensitive data, in this case, credentials. Unlike the previous vulnerability, the credentials are not initialized inside the code.



## Analysis of the Vulnerability

[At least one paragraph describing the vulnerability in your example, to include code snippets of the vulnerable code and screenshots demonstrating the vulnerability in the application/program]

In this vulnerability, the username and password created by the user gets stored in a text file. However, the credentials may get compromised by an attacker because the credentials are visible in a text file. The snapshot below represents a text file displaying the username and password that the user created.

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

Description automatically generated

## Mitigation

For this mitigation, I used hashing to encrypt the credentials. With the use of the hashing library, the text file in which the credentials get stored will not directly display the username and password created. Instead, it will display a bunch of random characters from the ASCII. In this way, the attacker will not be able to see the credentials. On the next page, the screenshot represents the encrypted credentials.

Text

Description automatically generated

# References

Common weakness enumeration. (n.d.). *CWE-311: Missing Encryption of Sensitive Data*. CWE. Retrieved April 25, 2023, from https://cwe.mitre.org/data/definitions/311.html

Common weakness enumeration. (n.d.). *CWE-798: Use of Hard-Coded Credentials*. CWE. Retrieved April 25, 2023, from https://cwe.mitre.org/data/definitions/798.html

*Hashlib - secure hashes and message digests*. Python documentation. (n.d.). Retrieved April 25, 2023, from https://docs.python.org/3/library/hashlib.html#simple-hashing