Graded Assignment: Homework 5

Sairam Soundararajan

University of Maryland Global Campus

SDEV 325 – Detecting Software Vulnerabilities

# Executive Summary

For this homework, the focus was on demonstrating porous defenses even further. The two vulnerabilities I chose were CWE-759: Use of a One-Way Hash without a Salt and CWE-327: Use of a Broken or Risky Cryptographic Algorithm. I used AWS Cloud9 for both vulnerabilities.

For Risky Crptographic Algorithm, I successfully mitigated the code just by using a different hashing algorithm. For the Use of One-Way Hash without salt, I had to choose a more secure hashing algorithm and add a salt to the code for better security.

# Example 1 – [CWE-759: Use of a One-Way Hash without a Salt]

## Overview

For this program, I am dealing with a one-way hash without using a salt. This makes it easier for attackers to pre-calculate the hash value using dictionary attack techniques. I used Python to work on this vulnerability.

## A picture containing text, screenshot, font, black Description automatically generated

## Analysis of the Vulnerability

In the vulnerable code, I worked with hashlib, .encode, .update, and .hexdigest. To simply put, these are what would be used originally before realizing that we would need to use a salt. This leaves the code vulnerable for attackers to get access to the username and password.

A picture containing text, screenshot, font, black

Description automatically generated

## Mitigation

For the mitigation, I used a salt to keep the username and password more secure and protected. I also used pbkdf2\_hmac as a part of using salt. The parameters inside take in utf-8, salt, and an integer value. Notice that it is harder to decode the following compared to the previous visual.

A black screen with white text

Description automatically generated with low confidence

# Example 2 – [CWE-327: Use of a Broken or Risky Cryptographic Algorithm]

## Overview

For Risky Cryptographic Algorithm, I used Java. The main objective behind this vulnerability is knowing which cryptographic algorithm is most secure to use. SHA stands for Secure Hashing Algorithm.



## Analysis of the Vulnerability

In the vulnerable code, I used SHA-1 to test how secure it is. SHA-1 is reported to be a very insecure hashing algorithm, as its support ended a long time ago. One thing to bear in mind is that its certificates are not reliable. Hash generated by SHA-1 is generally weak and, there, not used extensively these days.

A picture containing text, screenshot, font, black

Description automatically generated

## Mitigation

For the mitigation, I used SHA-256 instead of SHA-1. SHA-256 is a type of SHA-2 that generates 256-bit hash value. It is more secure than SHA-1 because it has improved security. Hashes generated by SHA-256 are stronger compared to SHA-256, which is why it is largely used among individuals.

A picture containing text, screenshot, font

Description automatically generated

References

*CWE - CWE-327: Use of a Broken or Risky Cryptographic Algorithm (4.1)*. (n.d.). Cwe.mitre.org. https://cwe.mitre.org/data/definitions/327.html

*CWE - CWE-759: Use of a One-Way Hash without a Salt (4.5)*. (n.d.). Cwe.mitre.org. https://cwe.mitre.org/data/definitions/759.html

*Difference between SHA1 and SHA2*. (2020, May 22). GeeksforGeeks. https://www.geeksforgeeks.org/difference-between-sha1-and-sha2/