Graded Assignment: Homework 5

Mark Kardash

University of Maryland Global Campus

SDEV325: Detecting Software Vulnerabilities

Executive Summary:

The two vulnerabilities chosen for this homework were CWE-759: Use of a One-Way Hash without a Salt, and CWE-307: Improper Restriction of Excessive Authentication Attempts. The first occurs when a storage system or program containing sensitive information (such as a password) does not provide a salt (a set of additional characters) in addition to the hash used in securing the information. The second, quite simply, is the absence of a proper restriction mechanism for when a user has attempted to log into a system too many times. In both cases, the system is seriously compromised, opening pathways to password-guessing attacks, brute-force attacks, and much more. To demonstrate them, I created two different username and password storage programs. I believe my source code works correctly, and I have managed to demonstrate its problems, although I didn’t quite demonstrate the first one correctly, as I had difficulties decoding the password.

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

Overview:

To demonstrate this vulnerability, I created a program taking a username and password from the user and storing them with a hash. The first (vulnerable) iteration of the program did not have a salt, while in the second (fixed) one, a salt was added. The program was written in Java.

Code working as expected:

Text

Description automatically generated

Analysis of the Vulnerability:

Use of a One-Way Hash without a Salt is a vulnerability occurring when, while able to hash a password, an application does not have a salt (An additional set of characters used in cryptography to protect passwords). Without a salt, the hash could be reversed, giving attackers access to the password, and therefore to data that the password protects.

Vulnerable Code:

Text

Description automatically generated

Graphical user interface, text

Description automatically generated

Vulnerable App Results:

Text

Description automatically generated

We have provided a username and successfully hashed a password. However, what happens if an attacker takes the hash, and uses a common decrypting website to decode it? Here, we used [Hex to Text | Convert Hexadecimal to Text Online (duplichecker.com)](https://www.duplichecker.com/hex-to-text.php)

The result is:

Graphical user interface, application

Description automatically generated

We have managed to convert the hex message to text. Even though ours did not match the password, there are more powerful tools that a hacker could use to fully decode the password.

Mitigation:

The way to mitigate this problem is to, of course, include a salt. A salt adds extra characters to the back of every password, so that it would not be able to be decoded.

Repaired Code:

Text

Description automatically generated

Text

Description automatically generated

A salt is now included, and the decoder would not recognize the hex when tasked with decrypting it.

Text

Description automatically generated

Again, we can see the website had a problem with decoding it, showing several letters and a couple of unintelligible rectangular symbols.

Graphical user interface, text, application

Description automatically generated

Example 2-CWE-307 – [Improper Restriction of Excessive Authentication Attempts]

Overview:

To demonstrate this vulnerability, I again created a Java program asking for a username and password, but having no set number of attempts. This exposes it to attacks.

Code works as expected:

Text

Description automatically generated

Analysis of the Vulnerability:

As previously explained, this vulnerability simply means not restricting the amount of login attempts for a website. A password guessing or brute-force attack is highly likely in such a situation, resulting in a data leak. Here, hackers can take all of the time they need to login into the system.

Vulnerable Code:

Text

Description automatically generated

Shape

Description automatically generated with medium confidence

Vulnerable Code Result:

Text

Description automatically generated

After several attempts and guesses, the attacker finally managed to access the program, and is now free to steal important information. This proves the program to be unsafe.

Mitigation:

To mitigate this problem, we simply restrict the number of login attempts by using “if” and “while” loops.

Repaired Code:

I also corrected the “Welcome to registration” mistake, as I realized the user is logging in rather than registering for the first time.

Text

Description automatically generated

Text

Description automatically generated

Repaired Code Result:

Text

Description automatically generated

References:

[CWE - CWE-307: Improper Restriction of Excessive Authentication Attempts (4.11) (mitre.org)](https://cwe.mitre.org/data/definitions/307.html)

[CWE - CWE-759: Use of a One-Way Hash without a Salt (4.11) (mitre.org)](https://cwe.mitre.org/data/definitions/759)

[Hex to Text | Convert Hexadecimal to Text Online (duplichecker.com)](https://www.duplichecker.com/hex-to-text.php)