Algorithms and Data Structures Artifact Enhancement

Danielle Hoopes

Southern New Hampshire University

11-26-2017

Table of Contents

Abstract

This details the enhancement process for Algorithms and Data Structure artifact justifying its inclusion in the ePortfolio as outlined by the course objectives for CS-499 Computer Science Capstone course. The following enhancement demonstrate my ability to

https://github.com/dhoopes11/dhoopes11.github.io

https://dhoopes11.github.io/

Algorithms and Data Structures Artifact Enhancement

Artifact Description

I have chosen to implement structured PHP code that implements the hashing and salting user information. This is an extension of the Travel SNHU site that implements the majority of the security features for the site and improves the overall ability of the code to perform up to today's standards of cyber security.

Artifact Inclusion Justification

I selected this artifact because the inclusion of a user account system and the security surrounding it are benchmarks for today's development processes. Hashing a password creates a one-way highway of sorts, creating an alphanumeric value that has no decryption technique. Although they do not totally prevent their successful use, hashing and salting user passwords combat the use of hash tables and rainbow tables for cracking passwords. Rainbow Tables store a pre-computed hash values for each word in the dictionary and then compare each password hash to the real password hash until a match is found.

Salting a user's passwords is a technique that takes, for example, a dictionary word from the hash example and appends to it a 32-bit salt. The salted key is now the original password with an extra 32-bit key attached, making it exponentially harder to guess or match. Salting secures passwords from a rainbow attach as they have no unique identifying properties. Today's standard for hashing and salting passwords is bcrypt. Bcrypt is a hashing algorithm that slows

down an attack such that an attacker must deploy a large amount of hardware to even be able to iterate through possible password equivalents.

Planned Enhancements Review

1. Hash and Salt Passwords: PHP (using bycrpt)

```
2. <?php
3. $hash = password_hash($password, PASSWORD_DEFAULT);
```

The general workflow for account registration and authentication in a hash-based account system is as follows:

1. The user creates an account.
2. Their password is hashed and stored in the database. At no point is the plain-text (unencrypted) password ever written to the hard drive.
3. When the user attempts to login, the hash of the password they entered is checked against the hash of their real password (retrieved from the database).
4. If the hashes match, the user is granted access. If not, the user is told they entered invalid login credentials.
5. Steps 3 and 4 repeat every time someone tries to login to their account.

Crackstation.net

Enhancement Process Reflection

During the enhancement process I was able to add the security features to the site that are most important. I encountered some challenges in my research when deciding what hashing algorithm to use and key-stretching structures to implement. I did however settle on the today's standard: bcrypt. This process of hashing and salting passwords in the development of Travel SNHU is an important step to limit increased risk of theft, fraud, and abuse of user account information. Safeguarding all user accounts is one of the most useful ways to increase security around site data and prevent insider attacks.