Lecture 15

Authentication Stepped Up
Persistent User Data
Protected Content.

* Course logo spider web photograph from Morguefile openstock photograph by Gabor Karpati, Hungary.
First - Update on Hashing

- Previous Lecture covered
  - md5 using explicit salt
- There is a better way
  - The password hash and verify pair.
- Also, in this lecture
  - A step toward databases
  - Object information in spreadsheets
  - Users in comma-separated-value files
A Better Way

- md5 is getting tired
- Today – password_hash()
Blowfish Hashing

bcrypt

For the bcrypt file encryption utility, see Blowfish (cipher).

bcrypt is a key derivation function for passwords designed by Niels Provos and David Mazières, based on the Blowfish cipher, and presented at USENIX in 1999.[1] Besides incorporating a salt to protect against rainbow table attacks, bcrypt is an adaptive function: over time, the iteration count can be increased to make it slower, so it remains resistant to brute-force attacks even with increasing computation power.

The bcrypt function is the default password hash algorithm for BSD and other systems including some Linux distributions such as SUSE Linux.[2] The prefix "$2a$" or "$2b$" (or "$2y$") in a hash string in a shadow password file indicates that hash string is a bcrypt hash in modular crypt format.[3] The rest of the hash string includes the cost parameter, a 128-bit salt (base-64 encoded as 22 characters), and 184 bits of the resulting hash value (base-64 encoded as 31 characters).[4]

The cost parameter specifies a key expansion iteration count as a power of two, which is an input to the crypt algorithm.

For example, the shadow password record

$2a$10$N9qo8uLOickgx2ZMRZoMyeIjZAgcf17p921dGxad68LJZdL17lhWy$

specifies a cost parameter of 10, indicating 2^10 key expansion rounds. The salt is N9qo8uLOickgx2ZMRZoMye and the resulting hash is IjZAgcf17p921dGxad68LJZdL17lhWy. Per standard practice, the user's password itself is not stored.

There are implementations of bcrypt for Ruby, Python, C, C#, Perl, PHP, Java, JavaScript, and other languages.
password_hash

(PHP 5 >= 5.5.0)

password_hash — Creates a password hash

Description

```php
string password_hash ( string $password , integer $algo [, array $options ] )
```

password_hash() creates a new password hash using a strong one-way hashing algorithm. password_hash() is compatible with crypt(). Therefore, password hashes created by crypt() can be used with password_hash().

The following algorithms are currently supported:

- PASSWORD_DEFAULT - Use the bcrypt algorithm (default as of PHP 5.5.0). Note that this constant is designed to change over time as new and stronger algorithms are added to PHP. For that reason, the length of the result from using this identifier can change over time. Therefore, it is recommended to store the result in a database column that can expand beyond 60 characters (255 characters would be a good choice).
- PASSWORD_BCRYPT - Use the CRYPT_BLOWFISH algorithm to create the hash. This will produce a
Back to Authentication

- Where to store information?
- Up until now - hard coded info.
- This lecture we want info on disk.
- But how?
  - Separate database server (not yet)
  - Lightweight database system (not yet)
  - In a plain file (yes!)
- But how to format the file?
CT 310 Lecture 15 Authentication Example

The example here illustrates a series of important steps.

To begin learning from this example, review the structure of the include files: support, control then header. This is a relatively common sequence of steps and you may want to emulate this style when building your own password protected websites.

Just for example, note that attempting to load any page other than the login page without first successfully authenticating yourself results in an immediate redirect to the login page. This protection is accomplished using the **PHP header command** in order that the contents of the tiny site is protected until a user successfully logs into the site.

Next, this example uses the more recent PHP password hashing and verification protocols released in PHP 5.5. Also, notice there is an external support library included here for backward compatibility with older versions of PHP. See the [phpPasswordHashingLib](#) documentation for additional information.

In this example we also see the use of session variables to maintain state. Specifically, to keep track of whether a user is or is not authenticated. You should play with the connection between session variables and cookies placed in the clients browser when studying this example.

Finally, user and password data is stored in a persistent form; a comma separated variable (CSV) file. PHP has good support for CSV files and you should always ask yourself when faced with the need for persistent data whether a full database solution is necessary. In particular, this simple technique of storing data in spreadsheet files should be considered when:

1. A user may wish to edit persistent data in a spreadsheet program.
2. The amount of persistent data is small and fits easily in a single table.

Note also the heavy use of the **PHP header command** in order that the contents of the tiny site is protected until a user successfully logs into the site.

- **Example Site** Storing user data in a CSV file.

The above examples have the drawback that you cannot see the actual PHP code, so they can be [downloaded as a zip file](#).
Example 02: Learn

- Use of bcrypt in PHP
  - Paired functions hash and verify
- Persistent store of user data (CSV)
  - KISS principal sometimes applies.
- Protected pages, password control
  - Protection through redirects
  - The header command
Example 02: File Overview

admin.php: when logged in, see hash of alternative passwords.
control.php: always include, redirects to login unless login established.
footer.php: common page footer with links.
header.php: common header with title.
index.php: represents protected site content.
login.php: implement password login with redirect when successful.
logout.php: destroy session, redirect to login.
support.php: function library, mostly to support creation, reading, and writing of users file.
users.csv: list of site users and hashed passwords.
Issue: Data Persistence
RAM

Store user information as an array of User Object Instances

writeUsers()

readUsers()

Disk

Store user information as a comma separated variable spreadsheet
Define a User

Take note that if no user.csv file exists then a new default file is created.

This bootstraps the system and avoids the problem of formatting a file from scratch.
Step Away from PowerPoint

- Please study this example carefully!
- There is a lot to notice
  - Try deleting the users.csv file
  - Pay attention to redirects!
  - Study password hashing and verification
- Ask yourself:
  - Why no web interface to set password?
Oops – 3/3/17 Update

➢ There was a problem!

Never saw the error until now because of PHP output buffering.