CST8285

Security Consideration & Working with Files

SQL Injection

- SQL injection is a method by which an attacker modifies an SQL query to circumvent desired behavior.
- For a quick explanation, here are some short videos that help explain it:
  - http://www.youtube.com/watch?v=FwIUkAwKzG8
  - http://www.youtube.com/watch?v=gll5wWoZ7aI
  - https://www.youtube.com/watch?v=PB7hWlqTSqs
  - https://www.youtube.com/watch?v=h-9rHTLHJTY

Preventing SQL Injection

- We could use the mysqli_real_escape_string function. View how this works here!
  - http://ca3.php.net/mysqli_real_escape_string
- This function is used to create a legal SQL string that you can use in an SQL statement
- Example

```
$unsafeData = " ' OR 1=1";

$safeData = $mysqli->real_escape_string($unsafeData);

$query = "SELECT * FROM users WHERE username = '$safeData'";
```

Preventing SQL Injection

- An easy way to prevent SQL injection attacks is to use **prepared statements** instead of adding variables to a query.
- Prepared statements send the parameterized query and the parameters separately to the MySQL server
- The parameters never become part of the query, so they can not change it

Storing passwords Properly

- In the Week 11 demo, we created a database table and stored passwords as plain text. In our lab we also stored email addresses.

  THIS IS WRONG!
- If an attacker gained access to the database, they would have a list of users, often with email addresses, and their password in plain text.
- Many people re-use passwords across multiple sites.

In the news

- Trump Hotel Properties – Credit Card Breach – July 2015
- Home Depot – 53 million email address stolen – Nov. 2014
- Adobe – 150 million passwords stolen – November 2013
- Cupid Media – 42 million (**plain text)** passwords stolen – January 2013
- LinkedIn – 6.4 million June 2012

Securing your data

- PHP has built in function to:
  - Use one-way encryption to create a hash of a plaintext password
  - Validate a password using both the supplied password and the hash

What is a hash?

- Hash functions are algorithms used to represent data of any length as data of a fixed length. The fixed length data is the hash
- A secure hash function will produce a unique hash for any unique data
- Hashes are one way encrypted. This means that you cannot determine the original data by reverse-engineering the hash
- You can test data against a given hash to determine if it was the source data used to create that hash

Where are hashes used?

- Common uses for hashes are
  - Storing passwords
  - Verifying file downloads
  - File identification (think Git)
  - Other data you consider sensitive

A note on password_hash() and password_verify()

- These are functions that are built into PHP 5.5
- Some of you may be running PHP 5.4
- If you are using PHP 5.4 please locate and download the password.php file in the Final Lab section on Bb
- This file must be included wherever your code uses these functions

Password_hash()

- Check out this quick article and example that covers the four PHP 5.5 functions;
  - http://www.sitepoint.com/hashing-passwords-php-5-5-password-hashing-api/
- Password hash takes a password, and an encryption type (represented by a constant), and produces a hash using the specified encryption type.
- Optionally, it takes two options parameters in an array:
  - Salt – The value of this option is added to the password string before it is hashed. This is done to prevent dictionary attacks. By default, password_hash() produces a random salt for you.
  - Cost – a value from 04 to 31 that determines how many times the password is hashed before it is returned. Default value is 10.

Password_hash()

- **Sample usage:**

```
$password = $_POST['pass'];
$passhash = password_hash($password,
PASSWORD_DEFAULT, [options]);
```

Password_hash()

- The first parameter is the value to be hashed
- The second parameter is a constant that identifies the type of hash to use
  - PASSWORD_DEFAULT – use the default PHP hashing mechanism (currently bcrypt). This constant will change according to best practices.
  - PASSWORD_BCRYPT – bcrypt.
- The third (optional) parameter is the options array
  - Ex $options = Array('cost'=>12);

Password_verify()

- Verifies a provided password against a given hash. If the password is correct, it will return the same hash.
- Accepts 2 parameters
  - $password – the password to be checked
  - $hash – the hash with which to verify the password.

Password_verify()

- Sample usage:

```
$password = 'password';
$hash =
'$2y$10$Ygsfv6q.N5FKoR248Fw1SeHpPxi1JcGbai
ACxn71PU17YiPoQfWK6';
if($hash == password_verify($password, $hash){
   true code to go here;
} else {
   false code to go here;
}
```

Transferring files in PHP

- There are many cases in which you will need to upload files to a website
  - Profile pictures, reports, documents, etc.
- PHP provides built-in functionality for handling file uploads.

The HTML form

- For a form to be able to handle file uploads, we need to set a new form attribute – **enctype**.
- enctype describes how the form data should be encoded when sent to the server
- By default, it is set to **application/x-www-form-urlencoded**
- For file uploads, it needs to be set to **multipart/form-data**

A new <input> type

- Files are uploaded using an <input> form element with the type **file.**
- When rendered, the **file** input will display a button that says ‘Browse’. When clicked, it will open a file picker for the user to select a file.
- The name of the file will be displayed beside the Browse button once selected.

The $_FILES variable

- In PHP, any files that are sent to the server are held in the **$_FILES** array.
- The $_FILES variable is an associative array, like $_GET and $_POST.
- To access a file, pass the name of the <input type="file"> element as the key.
- Ex: $_FILES['myFile']

The unseen magic

- When a file is sent to the server, it is saved in a temporary directory specified by the php.ini file. (In xampp, the default is C:\xampp\tmp)
- Files in the tmp directory are deleted once the script has finished executing.
- If you want to save the file, you need to do so specifically using the

move_uploaded_file()

- Takes two parameters:
  - The file name, and
  - The destination.
- The file name parameter is the tmp_name of the file.
  - Accessed using $_FILES['myFile']['tmp_name']
- The destination is the location on the file system where the file should be saved.
- Function returns true if move is successful, otherwise returns false.
- Examples can be found here;
  - http://php.net/manual/en/function.move-uploaded-file.php

file_exists($filename)

- Used to check if a file exists on the server's file system.
- Takes the filename (path + name) to check as a parameter
- Returns true if the file exists, otherwise returns false.