Anthony Bauld
000754334

PHP & JavaScript
Sam Scott

PHP Final Project ~ Security

Code Security

Code injection, which is caused by processing incorrect data, is one of the simplest ways to compromise any code. An unknown user injects code into a vulnerable website, changing the execution path. This is especially terrible for the website I'm building since someone may add code that changes their balance or the outcome of a coinflip if it isn't checked properly. The answer to this problem is to use filter_input to sanitize the parameter before calling it from JavaScript. I'm not going to use POST or GET in my code because it opens up additional avenues for injection attacks and will also use filter_input to correctly return a value to prevent any run-time errors.

Example:

```
$win = filter_input(INPUT_GET, "win", FILTER_VALIDATE_INT);

$loss = filter_input(INPUT_GET, " loss ", FILTER_VALIDATE_INT);

  if($win === null && $loss === null) {

        $message .= "The outcome was not currently found.";

        return($win, $loss);

  }

  else if ($win === false && $loss === false) {

        $message .= " The outcome was not an integer.";

        return($win, $loss);

  }
```

Data Security

The database on my website should have the highest level of security because it stores all of the user information, including balances, which a hacker may manipulate using SQL injection. As my website pretends to be a *"casino"* since you wager your balance, it's vital that the database is properly secured, because in a real-life application, someone may empty a real casino's entire bankroll by manipulating their own balances within a SQL. The built-in password hash and password verify methods. This will create a hashed and salted version of a user password in the database, which is more secure than simply retaining the text in a string.

I'll employ HTML Elements correctly to prevent empty parameters and push them into their proper format. I'll utilize event listeners to catch errors that might otherwise go unnoticed, and I'll use HTTPS

(Hypertext Transfer Protocol Secure) for transferring sensitive data, such as user information, when retrieving it. *Since, CSUNIX already uses HTTPS, I won't need to buy an SSL certificate, but if I did, I'd get one and redirect website traffic using a .htaceess file in the home directory to keep the information private.* The last feature I'll add to my website is an AJAX request that will check if the user is correctly logged into the site on every page.