ECS Lab Report

SQL Injection

The

Testing Environments:

The testing environments were of two types; one that was to simulate a login portal, while the other kind was intentionally vulnerable web application that was designed with such attacks in mind. The self-designed testing environments include:

* Login and register portal system, with barebones PHP code which did not use security mechanisms of any kind.
* Improved login and register portal system with backend-sanitized input.
* Further improved login and register portal system with hashing passwords and optionally the username too.
* Maximally improved login and register system utilizing PHP’s password encryption method to encrypt the password and the password verify function to verify the password, along with the username stored in hashed format.

A set of SQL injection queries were run against each of the above mentioned testing environments. The tests included:

* Blind SQL injection to bypass login verification
* SQL injection to extract data out of user credentials table

Upon testing on the self-built testing environments, the SQL injection queries yielded the following results:

On the login system utilizing barebones code, blind SQL injection techniques were successfully able to bypass login verification and provide authorized access to the resource.

On the login system utilizing sanitized input fields, some blind SQL injection techniques failed to execute due to the presence of characters like whitespace.

On the login system utilizing hashed passwords, all blind SQL injection techniques failed, due to the fact that the fields were being hashed and thus could not help provide a bypass mechanism to the login system.

On the login system utilizing PHP’s password encryption method, all blind SQL injection attacks failed, since the cleartext password was encrypted and salted, to form the ciphertext which was later used with the password\_verify() function of PHP, which automatically handles special characters and escape characters. Blind SQL injection on the username field would also not work, and this can be attributed to two cases:

If the username was stored and retrieved in a hashed format, then escaping the query wouldn’t work since the query is being run with the hash of the username.

Even if the username was not stored and retrieved in a hashed format, then escaping the query would work, but still the password verification would not occur, and thus bypass becomes unattainable.

The SQL injection techniques to generate/extract data out of the user credentials table would not produce the intended effect on any variant of the login systems, since the login systems were designed with it needing to only verify the username and password combination, and not return any value back.

The second testing environment was the