[**Week 7**](https://learn.umuc.edu/d2l/le/news/330770/1360531/view)

Posted Dec 3, 2018 10:08 PM

Welcome to SDEV 300 Week 7.

Greetings!  The end of the class is fast approaching.  Do your best to not get caught up this week, as I cannot accept late work at the end of the semester since grades need to be entered within 72 hours of the end of class.  Also, I’m traveling Wednesday – next Monday, so I apologize in advance if it takes me longer than usual to respond to emails.

In this week’s content, you will be “putting it all together” in terms of what we covered in the class so far by connecting HTML, PHP, Databases, and Security.  Make sure you take the time to read through the provided links, which will demonstrate the basics of inserting, updating, and deleting data in a MySQL database from a PHP application.   Pay particular attention to how to use Prepared Statements in PHP.  These should be used in the majority of cases to prevent SQL Injection.  Actually, Prepared Statements are designed for other purposes as well.  Specifically they are used to execute the same statement repeatedly with high efficiency.  The side benefit is that, once a statement is “prepared”, the SQL is partially compiled by the DBMS, after which time the dynamic part of the query (the parameters) are passed to the statement and the complete statement is executed. Since the DBMS treats these as typed bound parameters, there’s no way to break out of that parameter by adding SQL code.  Since the SQL code is already set on the server, the input is treated as a value and not code to be interpreted.

Using prepared statements, you will have PHP code that looks like this for your SQL query:

$stmt = $mysqli->prepare("INSERT INTO test(id) VALUES (?)")

Then to execute with a particular value for ?

$stmt->bind\_param(“i”, $id)//the i indicates an int value is expected

If you were using a standard msqli query, the same statement would be:

 $conn->query("INSERT INTO test(id) VALUES (‘$id’))

In this case, if $id is a dynamic value we are bringing in from an outside source (i.e. a form), then we would need to process it first to avoid SQL injection:

$id = mysqli\_real\_escape\_string($id);

Note, we still want to process the values to be inserted for XSS injection before saving to the DB (and/or retrieving from the DB)

**Lab:**

The lab this week is quite extensive so please allow enough time.  This lab walks you through using Windows, Apache, MySQL and PHP (WAMP) to create simple, yet very powerful PHP applications connected to a MySQL database.  It also covers how to use prepared statements for SQL Injection prevention. You will create a simple online “game” application, complete with a user logon feature.

Make sure you review best practices on storing user passwords in a database. You will often find older examples with outdated methods that are easily defeated with rainbow tables.

<https://www.owasp.org/index.php/Password_Storage_Cheat_Sheet#Argon2_usage_proposal_in_PHP>

**Supplemental Reading:**

Prepared Statements PHP Manual:

<http://php.net/manual/en/mysqli.quickstart.prepared-statements.php>

[PHP Notes For Professionals](https://learn.umuc.edu/d2l/common/dialogs/quickLink/quickLink.d2l?ou=330770&type=coursefile&fileId=PHPNotesForProfessionals.pdf) – Chapter 58/59.

Here are last week’s SQL Injection links for your review: as well

SQL injection is really just another type of input validation vulnerability. It involves attempting to insert SQL into input that will be used in a DB query.

<https://motherboard.vice.com/en_us/article/aekzez/the-history-of-sql-injection-the-hack-that-will-never-go-away>

 SQL Injection Explained: <https://www.hacksplaining.com/exercises/sql-injection#/after-server-error>

<https://www.w3schools.com/sql/sql_injection.asp>

 SQL injection can be defeated by escaping any SQL strings (I.e. ‘ ) that are part of the input and using prepared statements instead of normal queries. The use of stored procedures can also help.

 SQL Injection Defense: <https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet#Defense_Option_1:_Prepared_Statements_.28with_Parameterized_Queries.29>

<https://websitebeaver.com/prepared-statements-in-php-mysqli-to-prevent-sql-injection>

**Infosec Article of the Week:**

How are passwords hacked? Well, it depends.  If a particular individual is being targeted, a phishing scheme, social engineering attack, or perhaps a brute force attack (on a application that has not implemented login rate limiting <https://code.tutsplus.com/tutorials/how-to-build-rate-limiting-into-your-web-app-login--cms-22133>)  is usually most effective.  In addition, referencing a list of known passwords for the user and testing for reuse is very effective due to password reuse. According to [naked security](https://nakedsecurity.sophos.com/2013/04/23/users-same-password-most-websites/), 55% of the net users use the same password for most of the websites! It implies that if the website storing your password in plain text gets compromised, hacker is not only able to gain access of your account on that website but all your social media, email, forums etc accounts in which you are using the same password (<https://www.geeksforgeeks.org/store-password-database/>) !

 So how do these lists of passwords come about?  Typically, some eCommerce, government, school, email, etc. service is hacked and the DB stolen.  Do to poor password storage practices, the Database containing encrypted passwords is decrypted.  Because the hacker now has the database offline and does not have to go through the controls of your web interface (which may have rate limiting code), an automated brute force attack is much easier and extremely fast.

Examples:

<https://krebsonsecurity.com/2018/12/what-the-marriott-breach-says-about-security/>

<https://money.cnn.com/2017/10/03/technology/business/yahoo-breach-3-billion-accounts/index.html>

Rainbow tables and Dictionary attacks are used to translate hashes to plaintext:

<https://en.wikipedia.org/wiki/Dictionary_attack>

<http://kestas.kuliukas.com/RainbowTables/>

<https://project-rainbowcrack.com/>

<https://web.cs.du.edu/~mitchell/forensics/information/pass_crack.html>

An improved way of implementing a hashed password storage is by using dynamic salt values and also using a modern algorithm like Argon2.  The reason this type of algorithm is better is that it’s designed to require more computational work than a normal hash algorithm.  As a result, it can be tuned to require a certain amount of work in order to derive the hash.  This makes it much harder to generate rainbow tables and use brute force attacks.