Concordia Institute for Information System Engineering(CIISE)

INSE 6961 – Graduate Seminar in Information and Systems Engineering

Graduate Seminar Report-3

Udemy Course: Learning Ethical Hacking from Scratch

Section Covered: Website Hacking - SQL Injection Vulnerabilities

Total Time covered for this Segment: 60 Mins

Submitted to:

Professor Ayda Basyouni

Submitted By:

Sai Chandra Sekhar Reddy Dwarampudi - 40189233

What is SQl?

Every application created I the present world requires some data to be stored in some place. This data is retrieved when the user requests data, updates when user wants to add some data or delete the data as per the user request. It is basically a server which is called database and there should be a language to perform these create, update, and delete operations on the database which is called the structured query language (SQL). The data is usually represented in the form of tables (rows and columns) in the database. It is one of the main components which is embedded in every application as a back end, the storage point of all data in the form of tables.

What is SQL injection and why so dangerous?

SQL injection is a code injection method that has the potential to completely ruin your database. It is the most frequent online hacking tactic. In the SQL injection web page input is used to insert malicious code into SQL queries which makes the adversary gain access to the sensitive information in the database. The adversary can further exploit the system by logging in as admin, reading files outside the root, changing permissions, and uploading various malicious files. Files outside the www route can also be read by performing SQL injection. These can be used to update the scripts of php.

This attack can be automated by the attacker by using tools like SQL Map or try to insert SQL queries manually. Once the adversary can get into the database of SQL, they will be able to access all the data of the application and user which very dangerous and may result in data breach. Overall, SQL injection is one of the most dangerous and common vulnerability that most applications have which is a very good chance for an attacker to break into the system.

How to check whether the application is vulnerable to SQL injection?

The communication between the front-end webpage and the backend database of an application is mainly done using two methods GET and POST.

- **Checking whether a post webpage is vulnerable to the attack**

  In this example we are using a login page with username and password to perform the SQL injection. There are two ways to check the first one is by giving correct username and correct or wrong password along with the malicious SQL code we want to insert for example username as admin and password as aaa' or 1=1#'. The second one by giving the username as admin' #' and any text in password. This makes us to directly login to the admin page of the database.

- **Checking whether a get webpage is vulnerable to the attack**

  To check this, we are using a webpage which is to get the details of a particular user from the database. Whenever we are using the get method all the details like username, password, id and page information are available in clear form in the URL. This can be done

in two ways, one by using the URL and the other by using the text boxes in the webpage. By using the URL like page=user- info.php&username=zaid&password=123456... here to exploit by URL we need to just add ‘ order by 1%23 to the able URL after the username=zaid. Here %23 specifies the # (comment). The password present after the # is considered as a comment and will not be considered and the details are given directly. In the similar way in the text box of the username we can specify the username as zaid ‘order by 1#. Upon successful execution we can get the details of the zaid without even entering the correct password.

Reading database Information

By using the above method to exploit the get method we can retrieve the database credentials by using which the webpage is connected to the database. First to identify the number of columns that are selected by default and the amount of the data that is retrieved to show on this page. We will be using the order by command for this purpose. We are running the order by command up to 100000 to check whether we get any output or not. We got an error stating the columns exceed more than the specified range. Now let’s change the order to some tens and identify the output and we get an error and finally when we change the order to five, we get no error. We can identify that there are five columns which are retrieved to display in the page. Now let’s create our own customize select statement to retrieve the fields that we require. We'll use this format to choose items normally, but because we're attempting to perform a lot of selections from the URL, we'll have to use a union first and then say select. Then we must visualize what is happening with this app. We now know that this web application is selecting five records. As a result, five columns were selected.

We're now carrying out the commands one, two, three, four, and five. So, let's see what happens if we execute this and it combines that selection with another and shows us something else, and as you can see here, we're only seeing two, three, and four, which means that whatever value you put in a number two, three, or four, whatever you want to select, if you put it in there, it'll be displayed in this page in this specific location, and you can also see that you have results for two. Now to get the database details instead of number two replace it with the database(), number three with the user() and similarly number four with the version(). Now the statement becomes union select 1,database(), user(), version(),5. By using this statement we can get the details of the credentials the webpage is using to connect to the database. When we run the above command in the URL we get the username as owasp10, the seond option the databse we are looking for. We logged into the system with the root previleges which can be identified from the password as root@localhost. From the signature 5.0.51a-3-ubuntu5 we can specify the database version used by the webpage.

Discovering the Database Tables

Now from the previous analysis, we got to know that the target database in the owasp10. Now we try to get the tables that are present in the target database. From the above union statement we are

changing the databse() to table_name and the other two values user() and version() to null respectively. Now we are slectiong this statement from the database called information schema and we are selcting the table from tables. So the final statement becomes union select 1,table_name,null,null,5 from information_schema.tables. Here we are selecting column table_name from the database called information schema from table 'tables'. When we execute this we get 237 records and we get all the tbles that we have access to. We can also see from other web applications but in practical we can only see databases from the current application only. If we add where clause ' where table_schema = 'owsap10' which is the current database and execute it we only get the tables related our current database.

Extracting sensitive information from the Database

Now let's see how we can get all the accounts details related to a particular table. Consider the example of the accounts table. Here we are interested to get all the different accounts in the accounts table. Since we don't know the value of the column names in the accounts table in the above command change table_name to column_name , table 'tables' to columns and table_name from owsap10 to accounts. When you execute this, you will get column names present in the accounts table which are username, password, is_admin. Now the command becomes union select 1,username, password, is_admin,5 from accounts gives all the usernames and passwords along with whether the user is admin or not. You can use this credential to login as admin which can be used to upload malicious php scripts and bring down the whole application.

Prevent SQL injection vulnerabilities:

Filters, blacklist, whitelists can be used to prevent SQL injections which are not hundred percent able to block the attack. The best way is to code the application in such a way that whenever the adversary enters some malicious code in the test box the application should be able to recognize and prevent the user from the adversary from entering into the application. Using the parameterized statements which are present in most of the applications is the best way. If we considered the entire username till the # the problem is solved. In php we can use prepare statement by preparing the application to search for the column username in the accounts table and passing all the values till the # as username. This makes the web application to consider the entire thing as a value and it will try to check the username as the entire value which cannot be found thus preventing the user to login.

Another best way is to give minimum privileges to a user based on the work he/she will be doing. For example, if a employee has to only insert and update the data in the database give only those permissions to him eliminating any other permissions which can be exploited by the adversary. We can also use RBAC, ACL concepts which can prevent the data flow to the lower stages which is very much required in an organization.

Reference: https://concordia.udemy.com/course/learn-ethical-hacking-from-scratch/learn/lecture/5308842#overview

Course Instructor: Zaid Sahib (Ethical Hacker, Computer Scientist and CEO of zSecurity)