SQL injection UNION attacks

When an application is vulnerable to SQL injection and the results of the query are returned within the application's responses, the `UNION` keyword can be used to retrieve data from other tables within the database. This results in an SQL injection UNION attack.

The `UNION` keyword lets you execute one or more additional `SELECT` queries and append the results to the original query. For example:

```
SELECT a, b FROM table1 UNION SELECT c, d FROM table2
```

This SQL query will return a single result set with two columns, containing values from columns `a` and `b` in `table1` and columns `c` and `d` in `table2`.

For a `UNION` query to work, two key requirements must be met:

- The individual queries must return the same number of columns.
- The data types in each column must be compatible between the individual queries.

To carry out an SQL injection UNION attack, you need to ensure that your attack meets these two requirements. This generally involves figuring out:

- How many columns are being returned from the original query?
- Which columns returned from the original query are of a suitable data type to hold the results from the injected query?

Determining the number of columns required in an SQL injection UNION attack

When performing an SQL injection UNION attack, there are two effective methods to determine how many columns are being returned from the original query.

The first method involves injecting a series of `ORDER BY` clauses and incrementing the specified column index until an error occurs. For example, assuming the injection point is a quoted string within the `WHERE` clause of the original query, you would submit:

```
' ORDER BY 1--
' ORDER BY 2--
' ORDER BY 3--
etc.
```

This series of payloads modifies the original query to order the results by different columns in the result set. The column in an `ORDER BY` clause can be specified by its index, so you don't need to know the names of any columns. When the specified column index exceeds the number of actual columns in the result set, the database returns an error, such as:

```
The ORDER BY position number 3 is out of range of the number of items in the select l
```

The application might actually return the database error in its HTTP response, or it might return a generic error, or simply return no results. Provided you can detect some difference in the application's response, you can infer how many columns are being returned from the query.

The second method involves submitting a series of `UNION SELECT` payloads specifying a different number of null values:

```
' UNION SELECT NULL--
' UNION SELECT NULL,NULL--
' UNION SELECT NULL,NULL,NULL--
etc.
```

If the number of nulls does not match the number of columns, the database returns an error, such as:

```
All queries combined using a UNION, INTERSECT or EXCEPT operator must have an equal n
```

Again, the application might actually return this error message, or might just return a generic error or no results. When the number of nulls matches the number of columns, the database returns an additional row in the result set, containing null values in each column. The effect on the resulting HTTP response depends on the application's code. If you are lucky, you will see some additional content within the response, such as an extra row on an HTML table. Otherwise, the null values might trigger a different error, such as a `NullPointerException`. Worst case, the response might be indistinguishable from that which is caused by an incorrect number of nulls, making this method of determining the column count ineffective.

LAB	PRACTITIONER **SQL injection UNION attack, determining the number of columns returned by the query** »	✓ Solved

Note

- The reason for using `NULL` as the values returned from the injected `SELECT` query is that the data types in each column must be compatible between the original and the injected queries. Since `NULL` is convertible to every commonly used data type, using `NULL` maximizes the chance that the payload will succeed when the column count is correct.
- On Oracle, every `SELECT` query must use the `FROM` keyword and specify a valid table. There is a built-in table on Oracle called `dual` which can be used for this purpose. So the injected queries on Oracle would need to look like:

```
' UNION SELECT NULL FROM DUAL--
```

- The payloads described use the double-dash comment sequence `--` to comment out the remainder of the original query following the injection point. On MySQL, the double-dash sequence must be followed by a space. Alternatively, the hash character `#` can be used to identify a comment.

For more details of database-specific syntax, see the SQL injection cheat sheet.

Finding columns with a useful data type in an SQL injection UNION attack

The reason for performing an SQL injection UNION attack is to be able to retrieve the results from an injected query. Generally, the interesting data that you want to retrieve will be in string form, so you need to find one or more columns in the original query results whose data type is, or is compatible with, string data.

Having already determined the number of required columns, you can probe each column to test whether it can hold string data by submitting a series of `UNION SELECT` payloads that place a string value into each column in turn. For example, if the query returns four columns, you would submit:

```
' UNION SELECT 'a',NULL,NULL,NULL--
' UNION SELECT NULL,'a',NULL,NULL--
' UNION SELECT NULL,NULL,'a',NULL--
' UNION SELECT NULL,NULL,NULL,'a'--
```

If the data type of a column is not compatible with string data, the injected query will cause a database error, such as:

```
Conversion failed when converting the varchar value 'a' to data type int.
```

If an error does not occur, and the application's response contains some additional content including the injected string value, then the relevant column is suitable for retrieving string data.

LAB	PRACTITIONER **SQL injection UNION attack, finding a column containing text** »	Not solved

Using an SQL injection UNION attack to retrieve interesting data

When you have determined the number of columns returned by the original query and found which columns can hold string data, you are in a position to retrieve interesting data.

Suppose that:

- The original query returns two columns, both of which can hold string data.
- The injection point is a quoted string within the `WHERE` clause.
- The database contains a table called `users` with the columns `username` and `password`.

In this situation, you can retrieve the contents of the `users` table by submitting the input:

```
' UNION SELECT username, password FROM users--
```

Of course, the crucial information needed to perform this attack is that there is a table called `users` with two columns called `username` and `password`. Without this information, you would be left trying to guess the names of tables and columns. In fact, all modern databases provide ways of examining the database structure, to determine what tables and columns it contains.

LAB	PRACTITIONER	Not solved
	SQL injection UNION attack, retrieving data from other tables »	

Read more

Examining the database in SQL injection attacks »

Retrieving multiple values within a single column

In the preceding example, suppose instead that the query only returns a single column.

You can easily retrieve multiple values together within this single column by concatenating the values together, ideally including a suitable separator to let you distinguish the combined values. For example, on Oracle you could submit the input:

```
' UNION SELECT username || '~' || password FROM users--
```

This uses the double-pipe sequence `||` which is a string concatenation operator on Oracle. The injected query concatenates together the values of the `username` and `password` fields, separated by the `~` character.

The results from the query will let you read all of the usernames and passwords, for example:

```
...
administrator~s3cure
wiener~peter
carlos~montoya
...
```

Note that different databases use different syntax to perform string concatenation. For more details, see the SQL injection cheat sheet.

LAB	PRACTITIONER	Not solved
	SQL injection UNION attack, retrieving multiple values in a single column »	