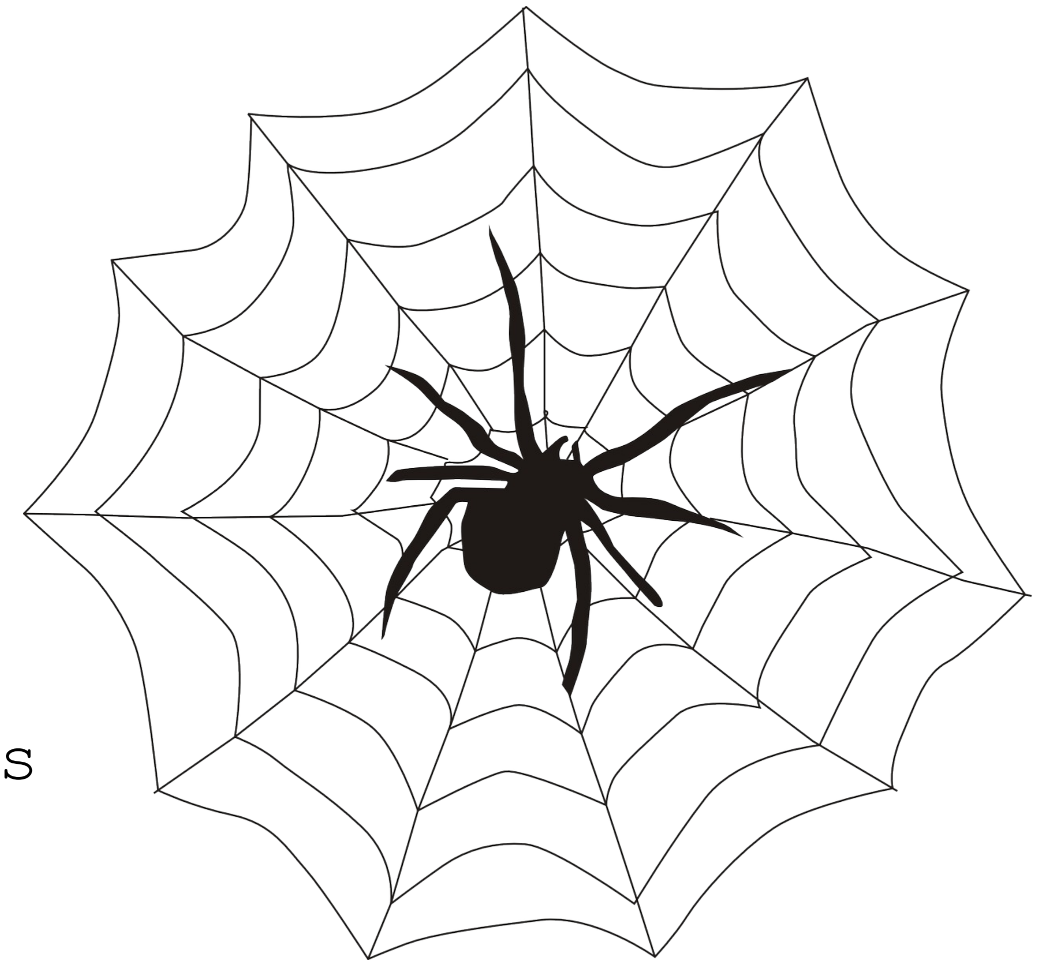
HackerFrogs Afterschool SQL Injection /w TryHackMe (Pt. 1)

Class:
Web Exploitation

Workshop Number:
AS-WEB-06

Document Version:
1.2

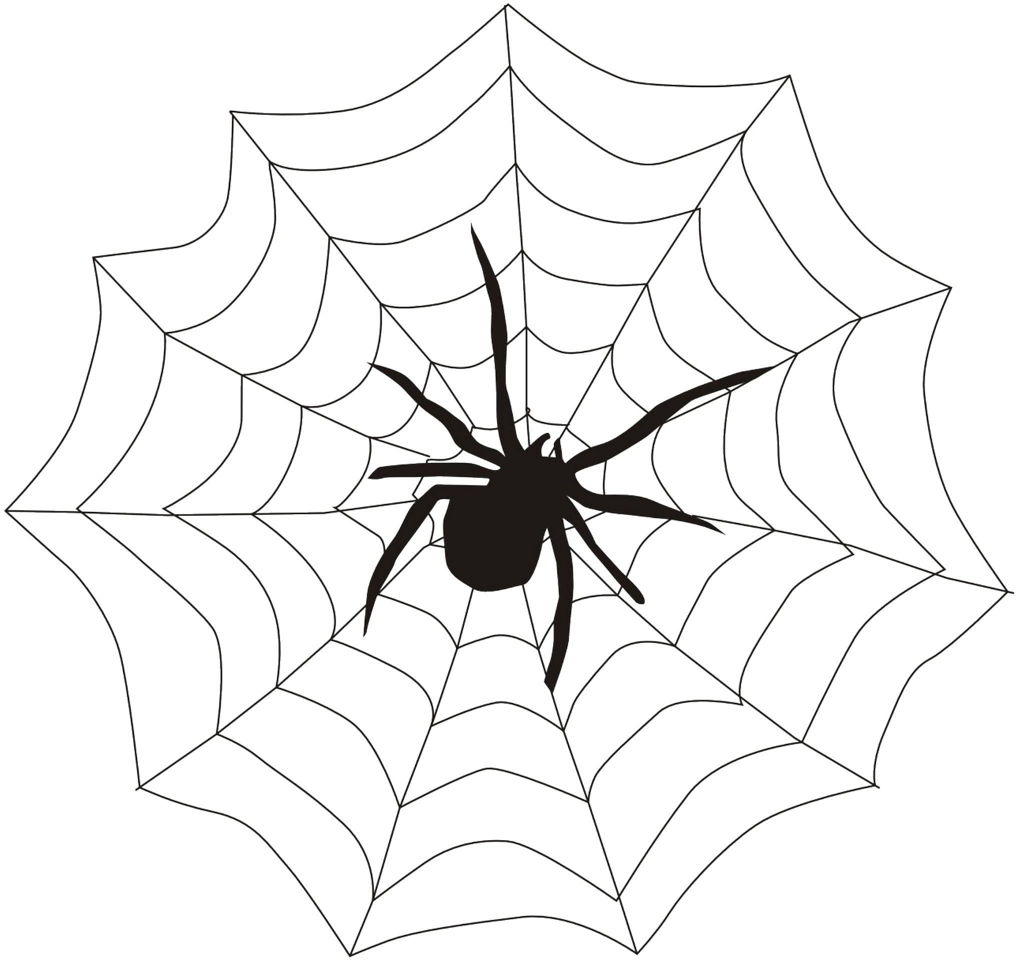
Special Requirements:
Completion of previous workshop, AS-WEB-05.
Registered account at tryhackme.com



What We Learned In The Previous Workshop

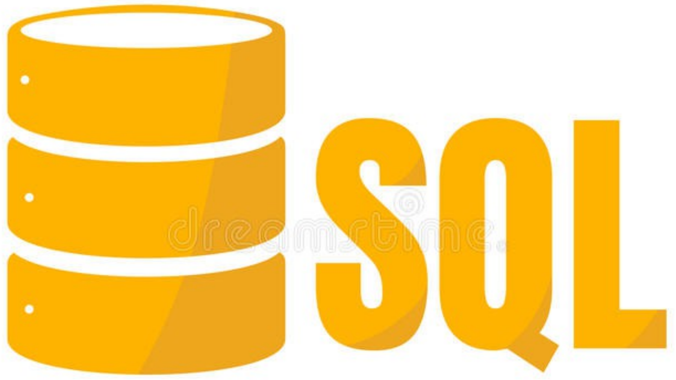
This is the sixth intro to web exploitation workshop.

In the previous workshop we learned about the following Web Exploitation concepts:



(S)tructured (Q)uery (L)anguage

Structured Query Language (SQL) is a programming language used for managing data held in relational databases.



(S)tructured (Q)uery (L)anguage

Simply put, SQL allows users to access and manage data contained in relational databases, which are common components in most web applications.

(S)tructured (Q)uery (L)anguage

In the previous workshop, we used SQL queries to retrieve relevant data on the SQL Murder Mystery Game and the SQL Zoo.

TryHackMe

Moving on to this workshop, we'll be learning SQL Injection exploits through the TryHackMe platform.

TryHackMe is a popular cybersecurity education platform with educational modules (called rooms) that span a great number of cybersecurity disciplines.

TryHackMe

In this workshop, we'll be making use of the one of TryHackMe's interactive rooms to learn more about SQL exploitation.

TryHackMe

But if we want use the TryHackMe platform, we must first have a valid user account for the website. If needed, signup for an account at the following URL:

https://tryhackme.com/signup

What is SQL Injection?

SQL Injection is a web app vulnerability where SQL commands are passed to the web app database through insecure user input.

What is SQL Injection?

SQL Injection vulnerabilities can lead to sensitive data exposure, admin account takeover, webserver takeover, and more.

Let's Go to the TryHackMe Room

To learn more about SQL injection, we'll be working through a TryHackMe room. In a web browser, login to the TryHackMe site, then navigate to the following URL:

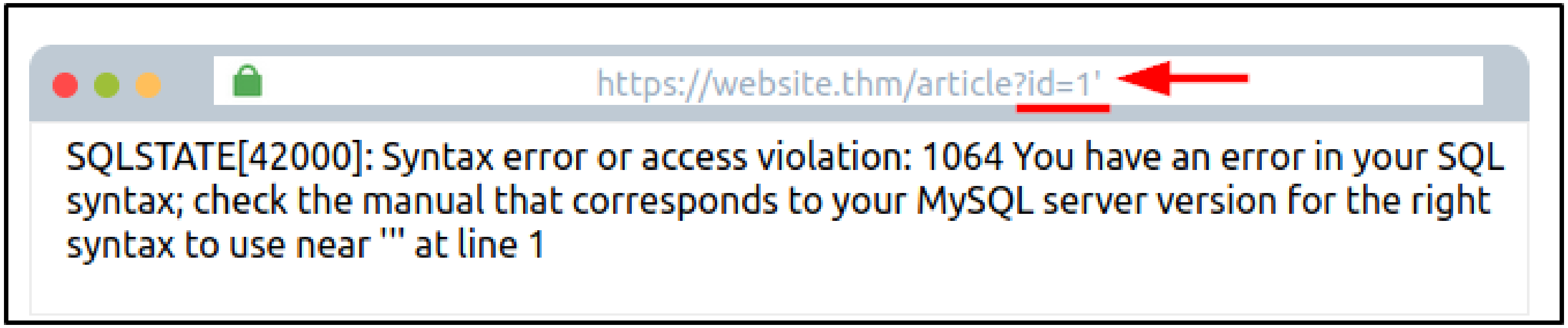
https://tryhackme.com/room/sqlinjectionlm

T4 - What is SQL Injection? SQL Login Query Example

```
select * from users
where username = 'admin'
and password = 'p4ssword';
```

In a web app, if this statement were to be supplied by the user, it would log the user in if the statement returned true.

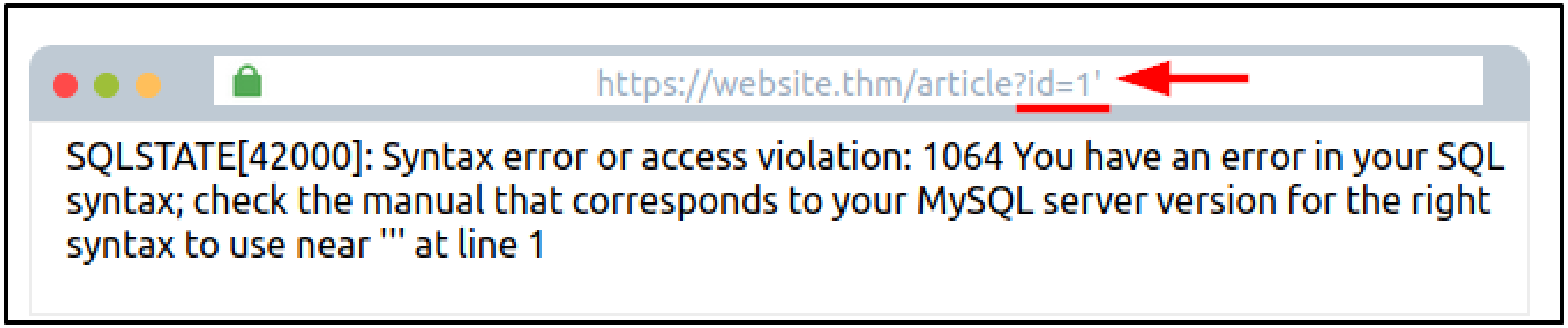
T5 - In-Band SQL Injection



```
SQLSTATE[42000]: Syntax error or access violation: 1064 You have an error in your SQL
syntax; check the manual that corresponds to your MySQL server version for the right
syntax to use near ''' at line 1
```

In-Band SQL Injection is SQL injection where the results of the injected query can be seen in the output of the web app. It is considered the easiest type of SQL injection to exploit.

T5 - In-Band SQL Injection
Error-Based Injection



```
SQLSTATE[42000]: Syntax error or access violation: 1064 You have an error in your SQL
syntax; check the manual that corresponds to your MySQL server version for the right
syntax to use near ''' at line 1
```

Error-Based SQL Injection is a type of In-Band SQL Injection, where SQL-related error messages are returned to webpage output.

T5 - In-Band SQLi
SQL Union Select Statements

```
select name,address,city,postcode
from customers
union select name,address,city,postcode
from suppliers;
```

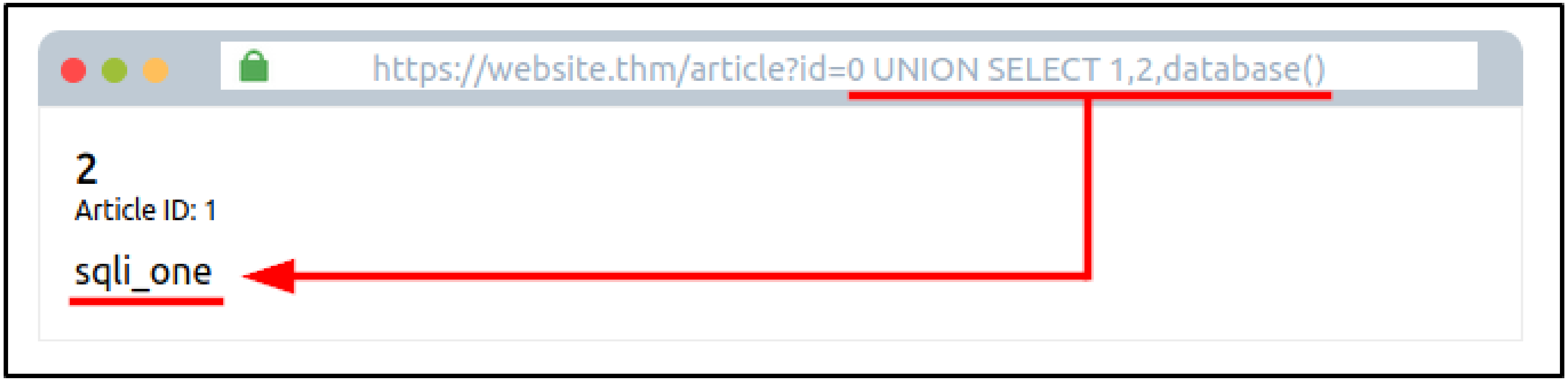
Union select statements allow rows to be returned from multiple tables at once,

T5 - In-Band SQLi
SQL Union Select Statements

```
select name,address,city,postcode
from customers
union select name,address,city,postcode
from suppliers;
```

provided that the number of columns returned from both queries is the same, and the type of data returned from each set of columns is also the same.

T5 - In-Band SQLi
Union-Based SQL Injection



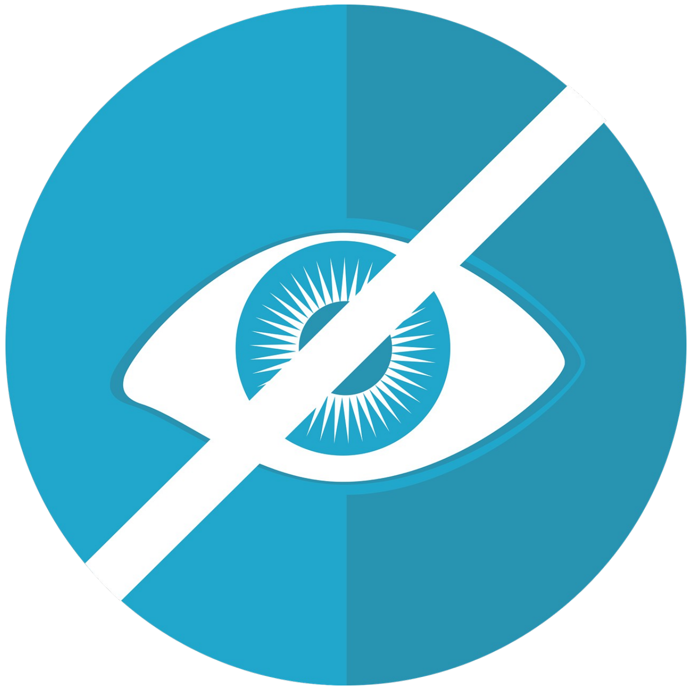
```
https://website.thm/article?id=0 UNION SELECT 1,2,database()
```

```
2
Article ID: 1
sqli_one
```

Union-Based SQL Injection is a type of In-Band SQL Injection which uses the SQL UNION operator with a SELECT statement in order to output additional information from the database.

T6- Blind SQLi – Authentication Bypass

Blind SQL Injection is SQL injection where the error messages resulting from improper SQL queries has been disabled, but the requests are going through nonetheless.



T6 - Blind SQLi – Authentication Bypass

Authorization Bypass is a type of Blind SQL Injection where insecurely written code allows the login of a user to a web app without the use of a proper username and / or password.

T6 - Blind SQLi – Authentication Bypass

```
' or 1=1 --
```

The most basic auth bypass SQL command is illustrated by the SQL command show above. Note that there is a space present in the command after the double dashes (--).

SQL Login Bypass String

' or 1=1 --

The single quote closes out the original query:

```
username = ''
```

SQL Login Bypass String

' or 1=1 --

If this portion is not included, the SQL database will return an error.

SQL Login Bypass String

' or 1=1 --

The next part, `or`, is what enables the bypass, because it enables us to create our own True statement.

SQL Login Bypass String

```
' or 1=1 --
```

Even if there is no matching username, because the or conditional is used, the overall query can return True if we supply the correct statement.

SQL Login Bypass String

```
' or 1=1 --
```

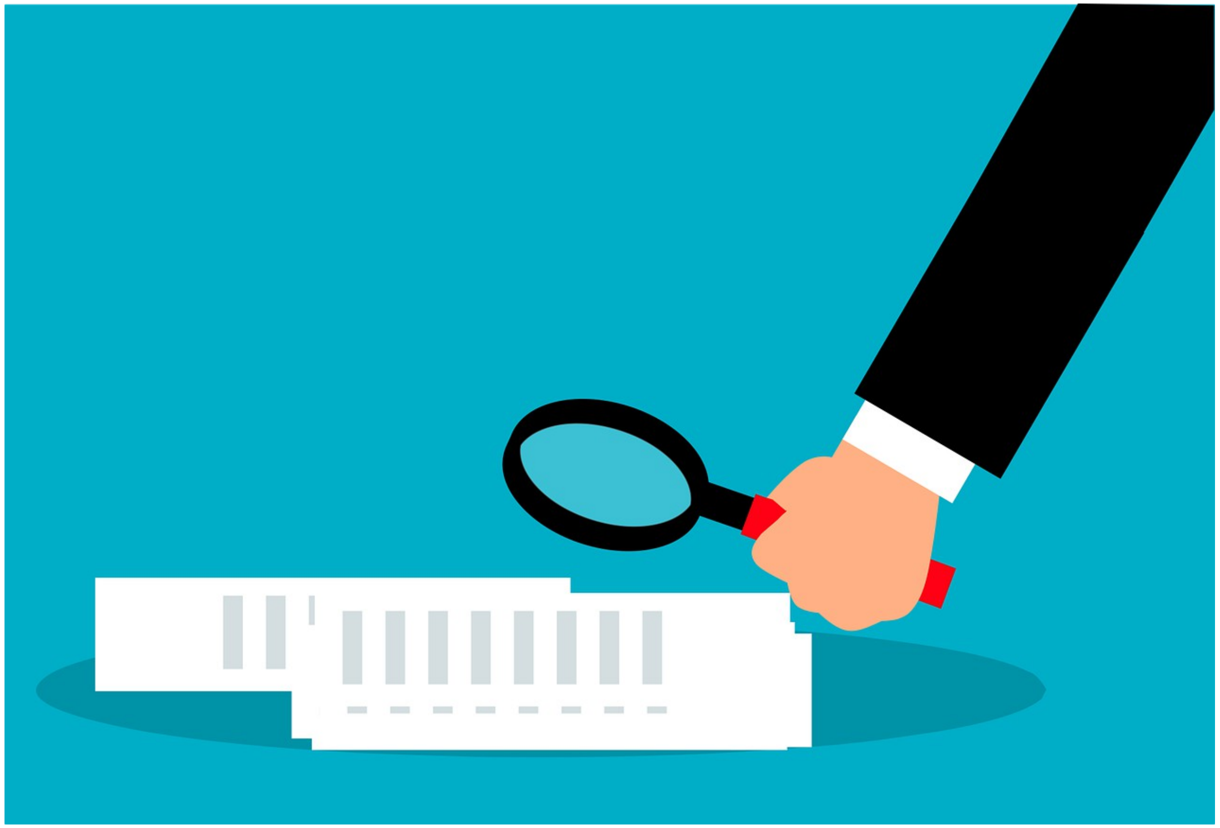
Which in this case is 1=1.

SQL Login Bypass String

```
' or 1=1 --
```

Then double dashes and space, which comments out rest of the original query.

Summary

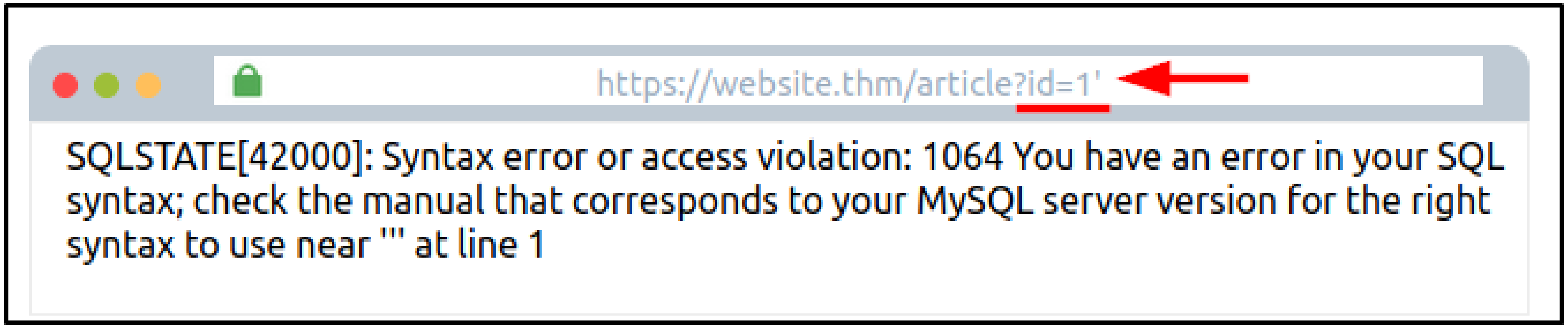


Let's review the SQL concepts we learned in today's workshop:

SQL Injection

SQL Injection vulnerabilities can lead to sensitive data exposure, admin account takeover, webserver takeover, and more.

Error-Based SQL Injection



```
SQLSTATE[42000]: Syntax error or access violation: 1064 You have an error in your SQL
syntax; check the manual that corresponds to your MySQL server version for the right
syntax to use near ''' at line 1
```

Error-Based SQL Injection is a type of In-Band SQL Injection where error messages from the database software are returned to the web app interface.

Useful SQL Commands (MySQL): Database Name

```
SELECT database()
```

Returns the name of the database used by the web app.

Useful SQL Commands (MySQL): Table Names

```
SELECT group_concat(table_name) from
information_schema.tables where
table_schema = 'DATABASE_NAME'
```

Returns all data in a single column with the **group_concat** function, and gets the names of all the tables contained in the specified database (DATABASE_NAME).

Useful SQL Commands (MySQL): Column Names

```
SELECT group_concat(column_name) from
information_schema.columns where
table_name = 'TABLE_NAME'
```

Using **group_concat** to return all results in a single column, returns all of the column names from a specified table (TABLE_NAME).

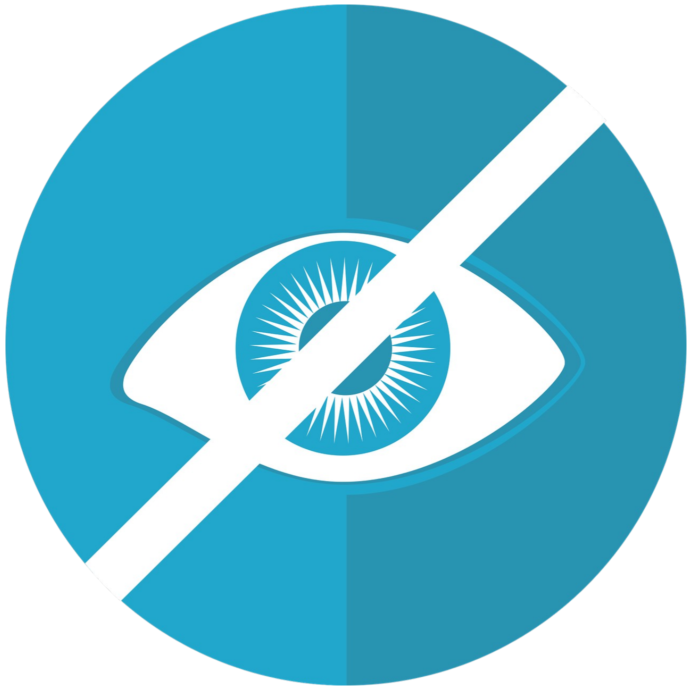
Useful SQL Commands (MySQL): Entry Dumping

```
SELECT group_concat(COL_NAME1, ' ',
COL_NAME2 SEPARATOR '<br>') FROM
TABLE_NAME
```

Using **group_concat** to return all results in a single column, returns all rows from the named columns (COL_NAME1,COL_NAME2) from the specified table (TABLE_NAME) separating the results from each row with an HTML linebreak.

Blind SQL Injection

Blind SQL Injection is SQL injection where the error messages resulting from improper SQL queries has been disabled, but the requests are going through nonetheless.



Blind SQL Injection: Auth Bypass

Authorization Bypass is a type of Blind SQL Injection where insecurely written code allows the login of a user to a web app without the use of a proper username and / or password.

Blind SQL Injection: Auth Bypass

```
' or 1=1 -- 
```

The most basic auth bypass SQL command is illustrated by the SQL command show above. Note that there is a space present in the command after the double dashes (--).

What's Next?

In the next web app hacking Workshops, we'll conclude our exploration of SQL injection with TryHackMe, covering Boolean and Time-Based SQL injection.

Extra Credit

Looking for more study material on this workshop's topics?

See this video's description for links to supplemental documents and exercises!

Until Next Time, HackerFrogs!