Introduction to Web Vulnerabilities and Hardening

Cybersecurity

Web Vulnerabilities and Hardening

Class Objectives

By the end of today's class, we'll be able to:

- Explain how SQL queries execute CRUD operations in SQL databases.
- Use the SQL UNION operator to enumerate the number of entries within a web app's query.
- Create a malicious SQL injection payload using UNION to pull data from another table.
- Compare hash checksums.
- Explain reflected and stored XSS attacks.
- Execute a basic <script></script> cross-site script.
- Create a cross-site script payload that injects from the end of a JSON object.

Web Vulns Recap

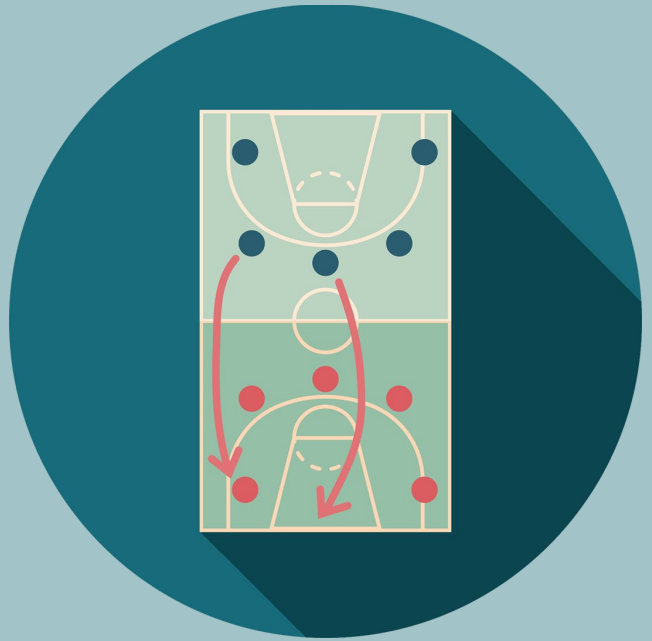
So far, we've covered the following web vulnerability concepts:

- The OWASP Top 10, a list of the most critical and commonly seen web application vulnerabilities.
- How to modify SQL queries with union and concat.
- SQL injection (SQLi) attacks
- Cross-site scripting (XSS) attacks

Web Vulns Recap

Remember, understanding injections is quintessential to learning about application security. It being one of the oldest but still relevant forms of attack.

Today’s lesson will focus mainly on SQL injections and cross-site scripts.



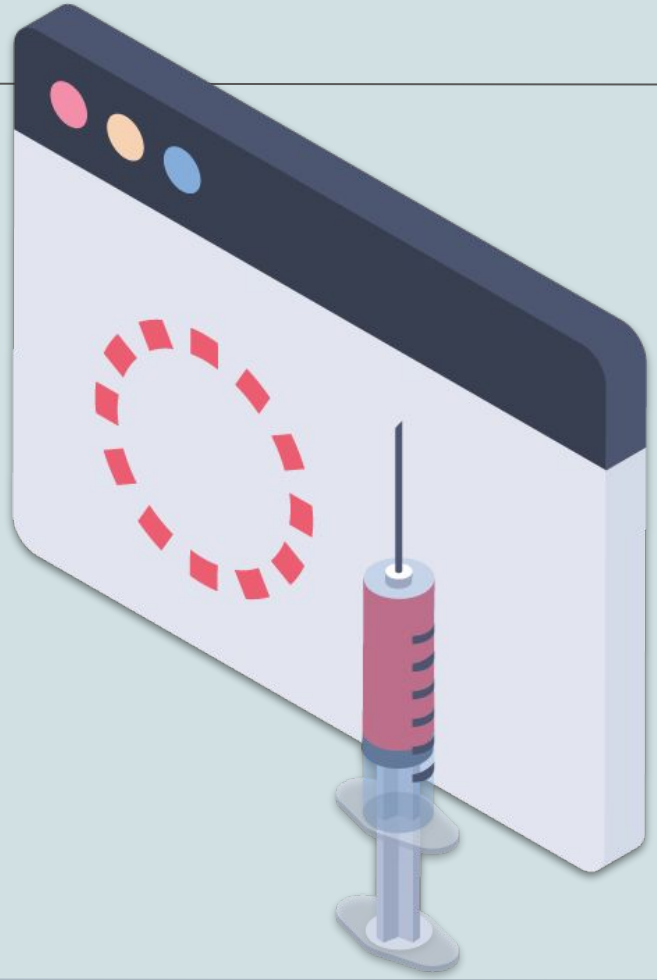
In today’s today's activities we use our local Vagrant machines to review and expand upon SQL injection and cross-site scripting attacks.

SQL Review

SQL Injections

SQL injection attacks are still prevalent, despite effective mitigation strategies, such as input sanitization.

This is especially true on websites and applications that rely on old and antiquated technology or deal with financially constrained budgets.



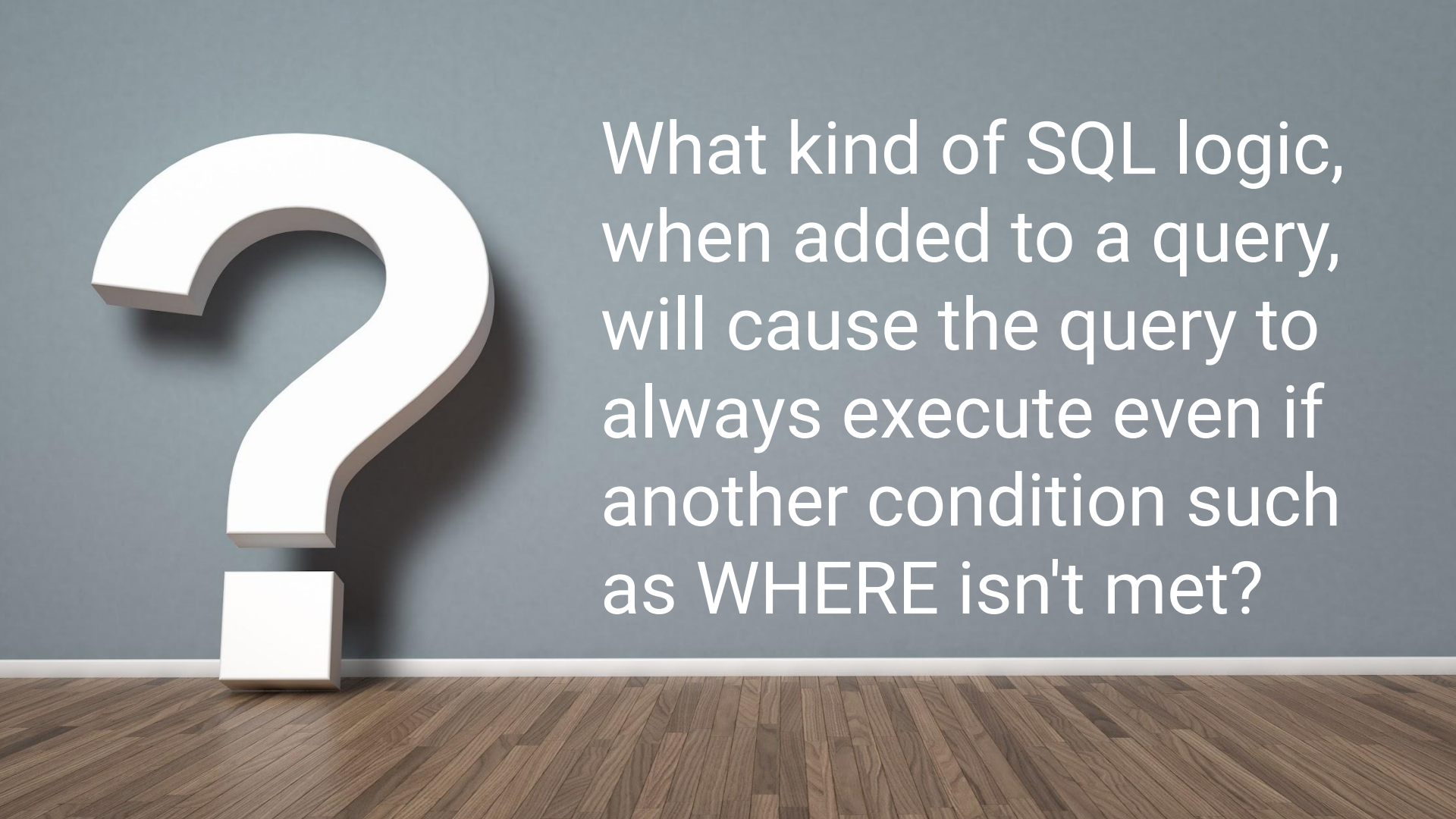
SQL Databases

An SQL database is where the data within a web application is stored.

A database's tables are structured in a column and row format, similar to a spreadsheet format, like Microsoft Excel.

- The SELECT statement is used to retrieve data, often in conjunction with FROM and WHERE to specify which table to look at and to set the matching condition of specific data to look for.

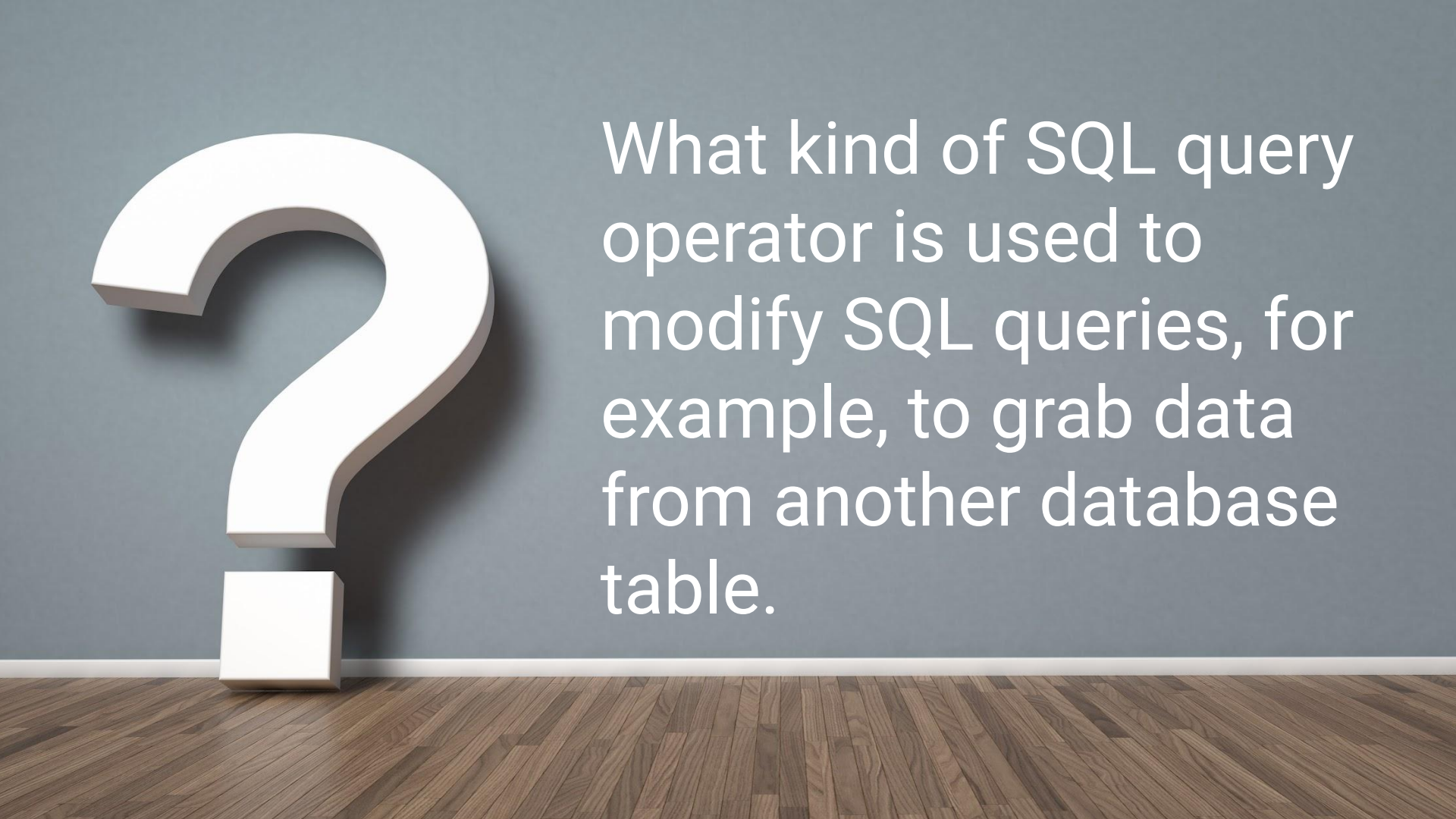
- For example: `SELECT username, id FROM users WHERE id = '123'`


What kind of SQL logic, when added to a query, will cause the query to always execute even if another condition such as WHERE isn't met?

Always True

SQL payloads with always true statements will cause an SQL query to execute, despite the other intended conditions.

```
SELECT * FROM popular WHERE actor='Will Smith' OR '1' = '1';
```


What kind of SQL query operator is used to modify SQL queries, for example, to grab data from another database table.

UNION

We've used the `UNION` SQL query operator to create SQL injections that retrieved data outside of the original query scope.

```
SELECT username, pin FROM users WHERE id = '12093' UNION
SELECT username, password FROM users WHERE '1' = '1'
```

- The UNION operator will force the database to retrieve the username and password from the employees database in place of the original query.

Using UNION SELECT for Column Enumeration

The union command can be altered to find out how many columns were used in the expected SQL query.

Title	Releases	Actor	Genre
Citizenfour	2014	Edward Snowden	Documentary

- For example, an SQL query may look similar to:

  ```
  'SELECT title, release, actor, genre FROM movies
  WHERE title LIKE %Citizen%
  ```

  with the following intended output. Recall that % represents a wildcard in SQL.

Using UNION SELECT for Column Enumeration

By using an SQL injection payload, we can interrupt this regular query with UNION in order to modify it to return the number of columns used in the original query

- The following format is commonly used for enumerating the number of values a query:

```
' union select NULL,NULL,NULL,etc.--
```

  - Where each NULL is a placeholder for the number of columns or values in the original query. Note that there is a space after the --.

Using UNION SELECT for Column Enumeration

For anything more or less than four NULL value placeholders, a site may return an error such as:

```
'Error: 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
```

- You will then either add or remove NULL's until the injection attempt no longer causes an error. When the right number of columns are found, the query should no longer error and should return blank data.

Using UNION SELECT for Column Enumeration

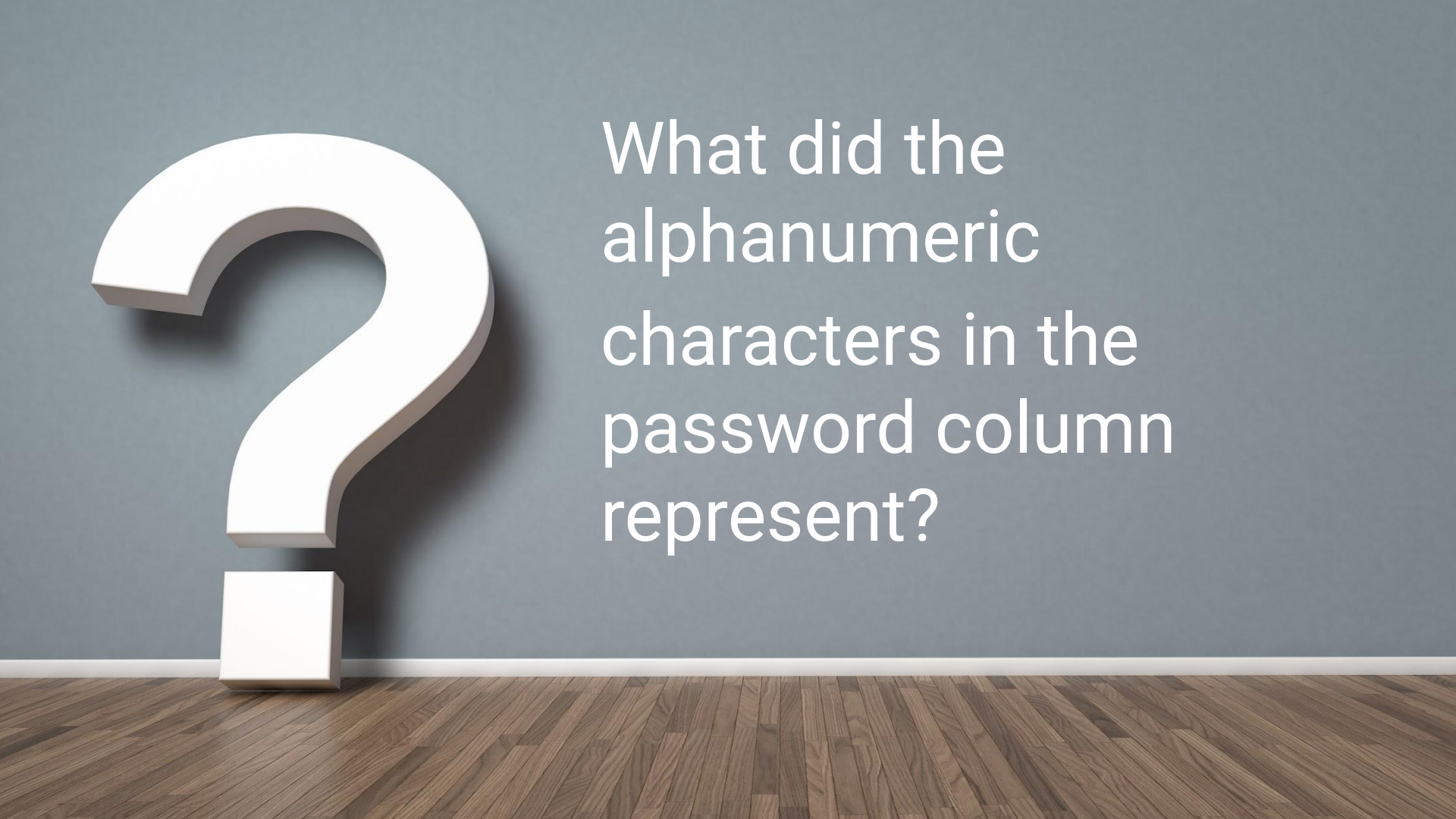
```
' union select NULL,NULL,NULL,NULL--
```

Title	Releases	Actor	Genre
Citizenfour	2014	Edward Snowden	Documentary

Using UNION SELECT for Column Enumeration

```
' union select NULL,username,password,NULL from employees--
```

Title	Releases	Actor	Genre
Citizenfour	2014	Edward Snowden	Documentary
admin	098f6bcd4621d373cade4e832627b4f6		
jsmith	5f4dcc3b5aa765d61d8327deb882cf99		
ckent	84d961568a65073a3bcf0eb216b2a576		
pparker	9f05aa4202e4ce8d6a72511dc735cce9		

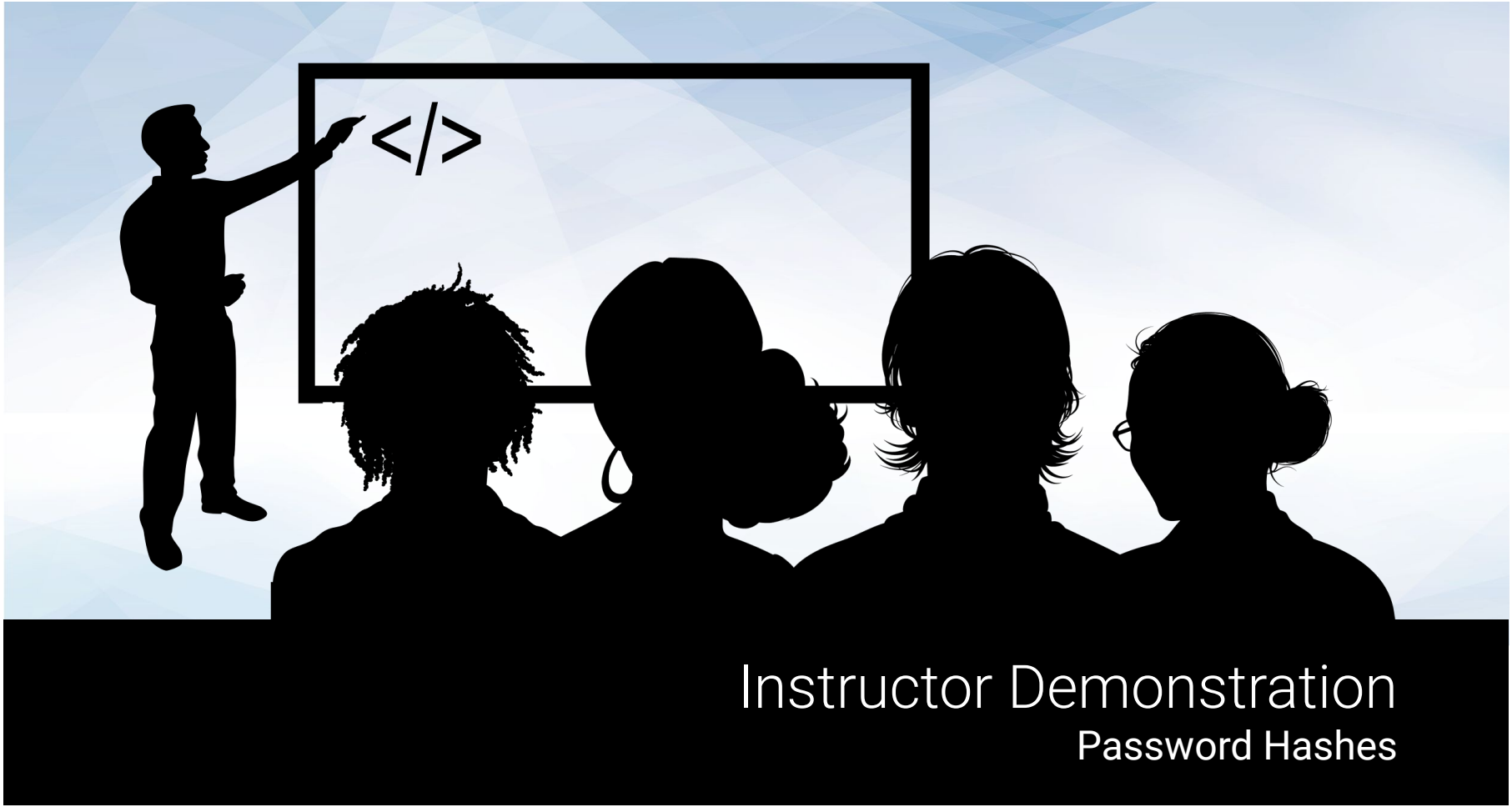

What did the alphanumeric characters in the password column represent?

Using UNION SELECT for Column Enumeration

The alphanumeric characters that appear in the password columns are called **hashes**.

- Whenever you create an account on a website and enter a new password, it isn't stored in plaintext for security reasons. Modern web applications will store a hashed version of their password in its database.

- **MoA hash** or **hash value** is a string of alphanumeric characters created by running what is known as a cryptographic hash function on data. The type of cryptographic hash function used will determine the number of alphanumeric characters that are used in the hash value.



Instructor Demonstration

Password Hashes

SQL Injections

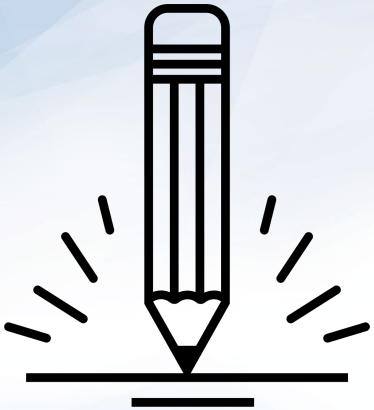
01 Use the web application as intended in order to establish a baseline of how the site is expected to function.

02 Use an always true statement to find out if we can even use an SQL injection. If we generate technical errors from the site, then we continue to the next step.

03 Enumerate the number of columns the query has by using union select NULL-- and adding or removing NULL's until no error occurs.

04 Modify our union select query, replacing the NULL's with the cross-table data that we're looking for.

05 If we find sensitive information, such as hashes, we'll try and run some tests to see if it matches what we believe the password is.



SQL Injection with bWAPP

In this activity, you will act as an Application Security Engineer tasked with testing a new web application by running SQL injection payloads on it.

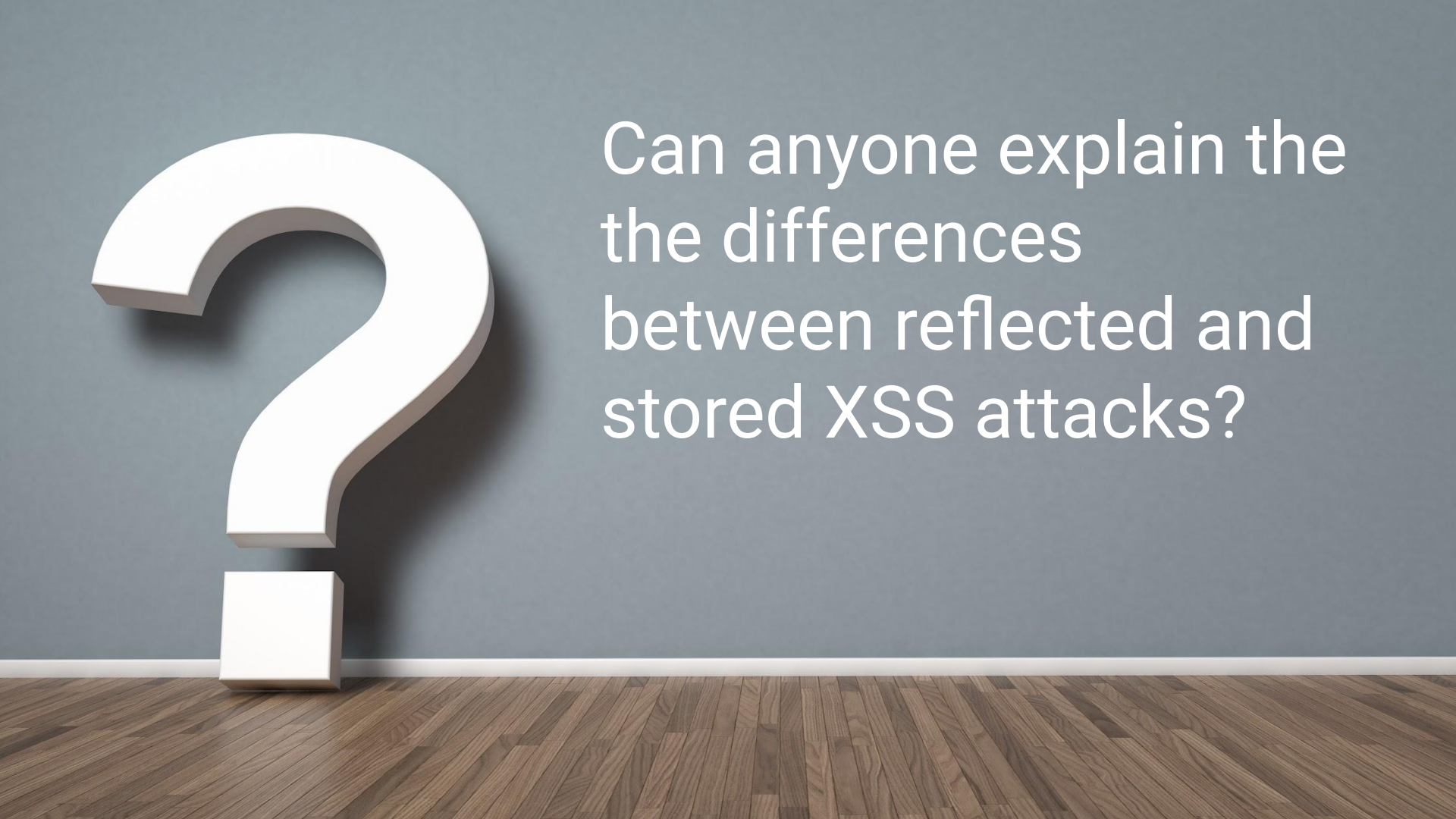
Suggested Time:
0:25

Time’s Up! Let’s Review.


Break

Testing XSS on Web Applications

Much like SQL injections, cross-site script attacks work from vulnerabilities in "injection" points, such as search fields and comment forms.


Can anyone explain the the differences between reflected and stored XSS attacks?

Can anyone explain the the differences between reflected and stored XSS attacks?

Reflected XSS attacks rely on the victim's interaction to work, such as a URL embedded in a socially engineered email.

Stored XSS attacks get stored within the web application, such as a posted comment within the comment section of a web page.

Simplified HTML XSS

Both a reflected and stored XSS executes on the frontend, within a client.

A web browser receives the frontend code such as HTML, CSS, and JavaScript within an HTTP response body from a server.

The HTML and CSS frontend code is responsible for displaying static content, the design and layout of the site, and embedding media.

JavaScript is used to extend functionality and create dynamic elements on a page.

Within that frontend code, the XSS attacks we're going to look at take advantage of what is inside of an HTML <script> block, where JavaScript is run from.

Simplified HTML XSS

A generalized HTML page containing a form:

```
<html>
      <head>
      </head>
      <body>
          <div>
              <form>
              <!-- This is the XSS injection point -->
              </form>
          </div>
      </body>
  </html>
```

Within the <form></form> tags contain the code that creates a form field for user input.

Simplified HTML XSS

```
<html>
    <head>
    </head>
    <body>
        <div>
     <form>
       <script>alert('Message!')</script> <!-- Attacker
        injection using <script> tags -->
     </form>
        </div>
    </body>
</html
```

When the page does not properly sanitize the form field, you can inject an HTML script block that contains JavaScript: <script>alert('Message')</script>.

Reflected XSS Review

The reflected XSS example is URL poisoning. When a user clicked on the link, it would execute malicious code:

- `192.168.50.128/webgoat.net/Content/ReflectedXSS.aspx?city=<script>alert("Hacked")</script>`

- In this example, an entire HTML <script> block is embedded with in the URL, and executes when the URL is clicked. Anything we put inside of the script block will execute when the user clicks it.

- When this script ran, a pop-up window shows up in the browser with a message that says "Hacked".

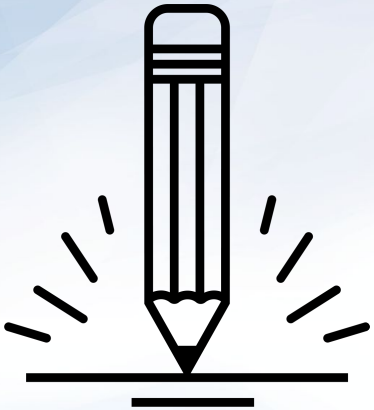
Stored XSS Review

A stored XSS attacks leverages that a web site will store data that is sent to it, usually through a POST request, such as someone filling out a comment field.

- Specifically, we can creating a comment in a vulnerable web application’s email and message forms.
- Within those message forms, we can created a JavaScript script block payload that looked like:

  `<script>alert(document.cookie)</script>.`

- When we hit the Save Comment button, a pop-up window appears, displaying the contents of a browser's cookie for this site.
- Then, a different user would see that same window pop up even though their message was completely normal.



SQL Injection with bWAPP

In this activity, you will act as an Application Security Engineer tasked with trying XSS techniques against a new web app.

Suggested Time:
0:30

Time’s Up! Let’s Review.


The End