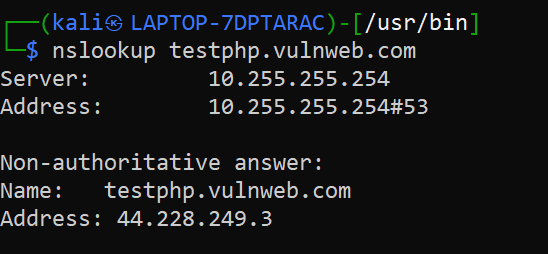
**CYBERSECURITY MAIN TASK-1**

- M.Manurajan 107124065

**Level- 1: Recon**

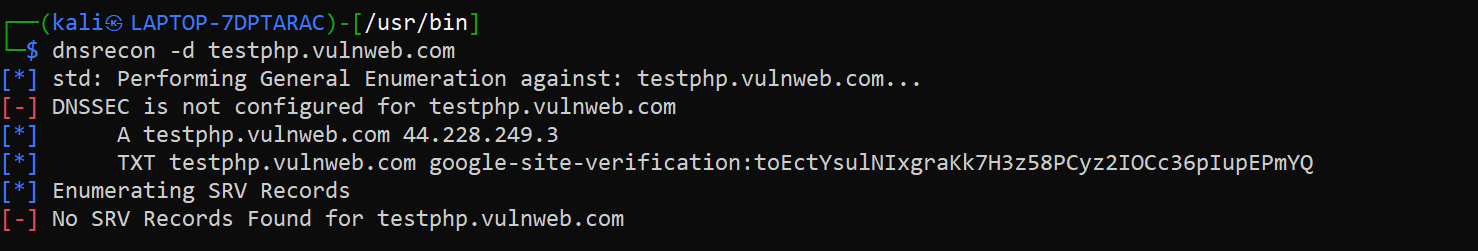
Perform recon on <http://testphp.vulnweb.com>

IP address: 44.288.249.3

Command: nslookup testphp.vulnweb.com

DNS info:

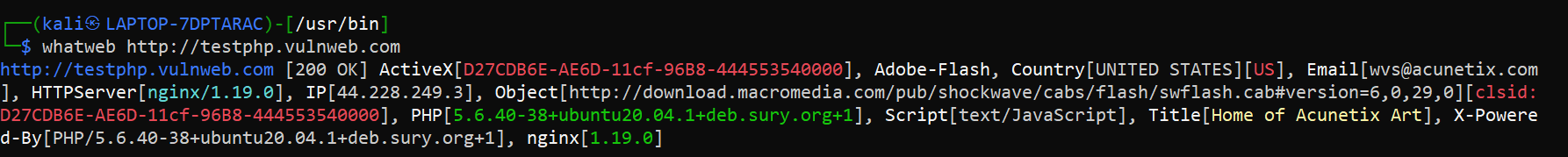
Tool used: dnsrecon

* The domain testphp.vulnweb.com is at 44.228.249.3.
* DNSSEC is not enabled.
* One TXT record found for Google site verification.
* No SRV records present.

Tech stack (server, CMS, etc.)

Tool used: whatweb

Command: whatweb http://testphp.vulnweb.com



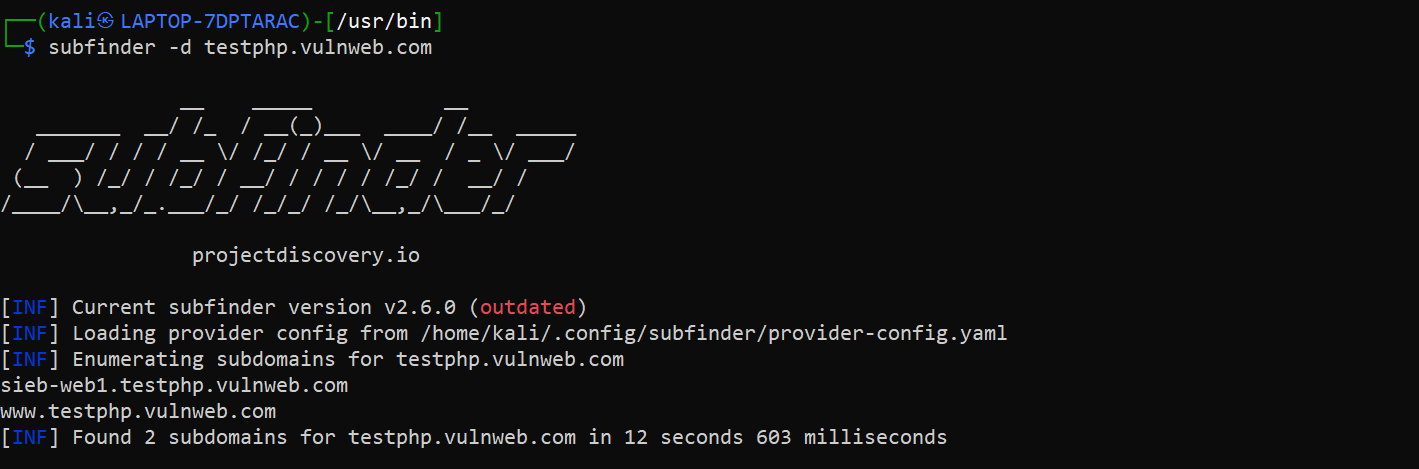
* Web server: nginx/1.19.0
* Backend: PHP 5.6.40 on Ubuntu 20.04
* Script: text/JavaScript
* Plugins Detected: Adobe Flash, ActiveX
* Title: “Home of Acunetix Art”
* Email Discovered: [wvs@acunetix.com](mailto:wvs@acunetix.com)
* Country: United States (US)

Subdomains (if any):

Tools used: subfinder

Command: subfinder -d testphp.vulnweb.com

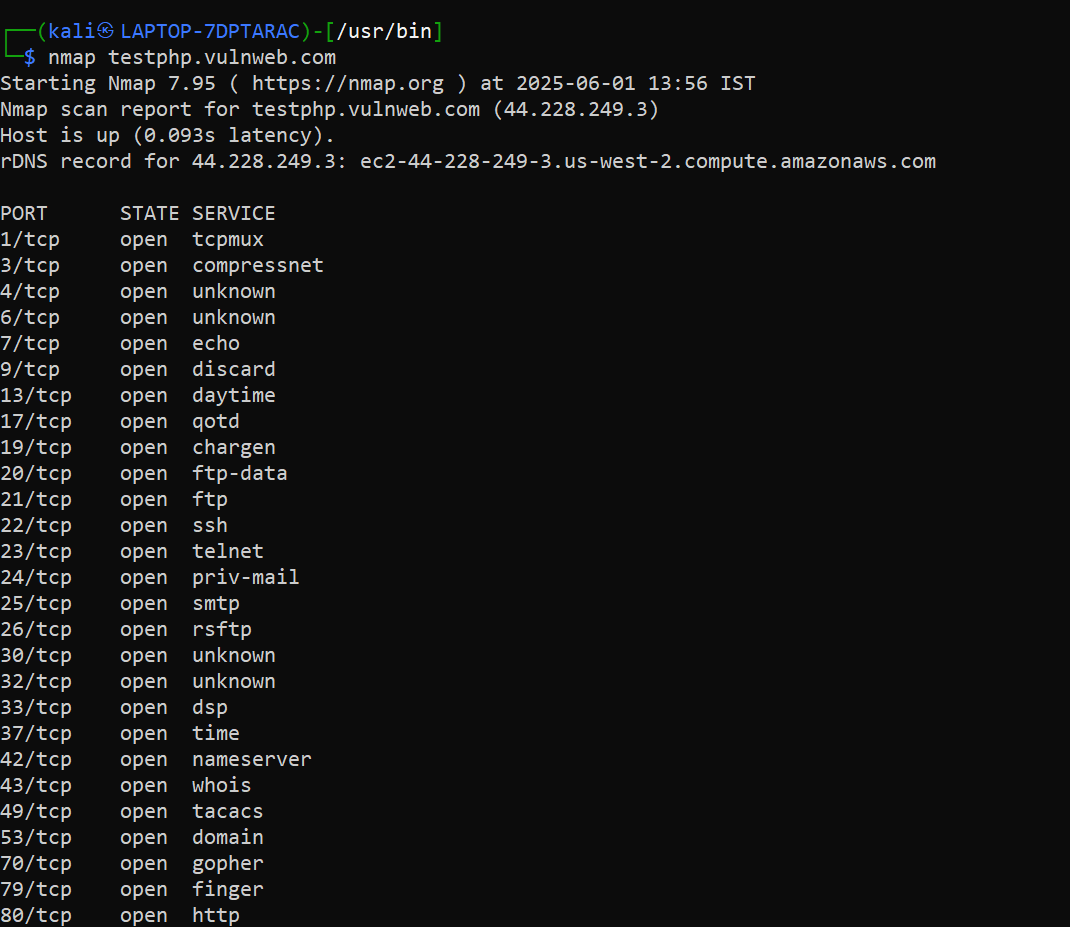
Subdomains: sieb-web1.testphp.vulnweb.com

 www.testphp.vulnweb.com

Open ports/services:

Tool used: nmap

Command: nmap testphp.vulnweb.com



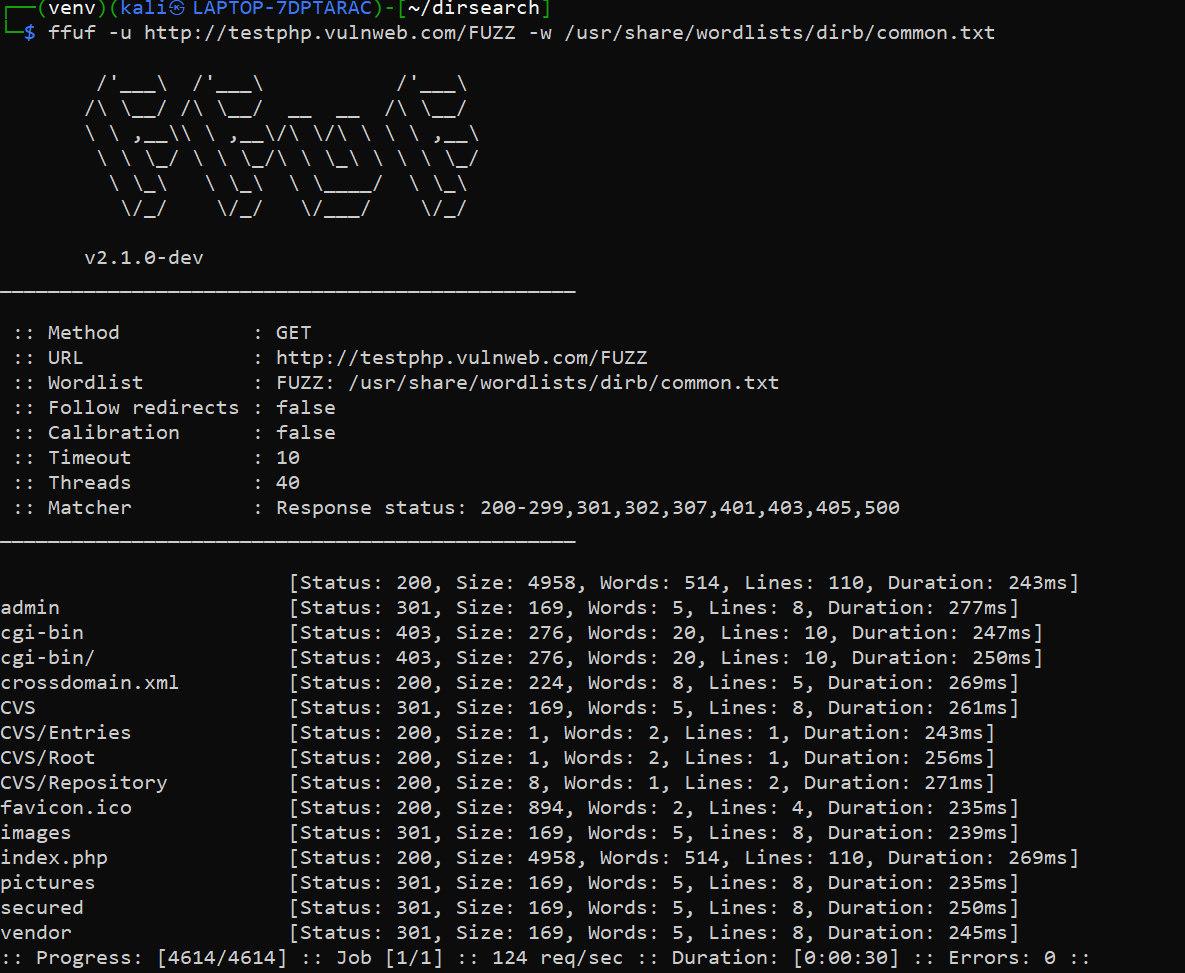




(rest refer here)

Directory structure:

Tools used: dirsearch

Command: ffuf -u http://testphp.vulnweb.com/FUZZ -w /usr/share/wordlists/dirb/common.txt

**Path Status**

/admin 301

/index.php 200

/secured 301

/images, /pictures 301

/vendor 301

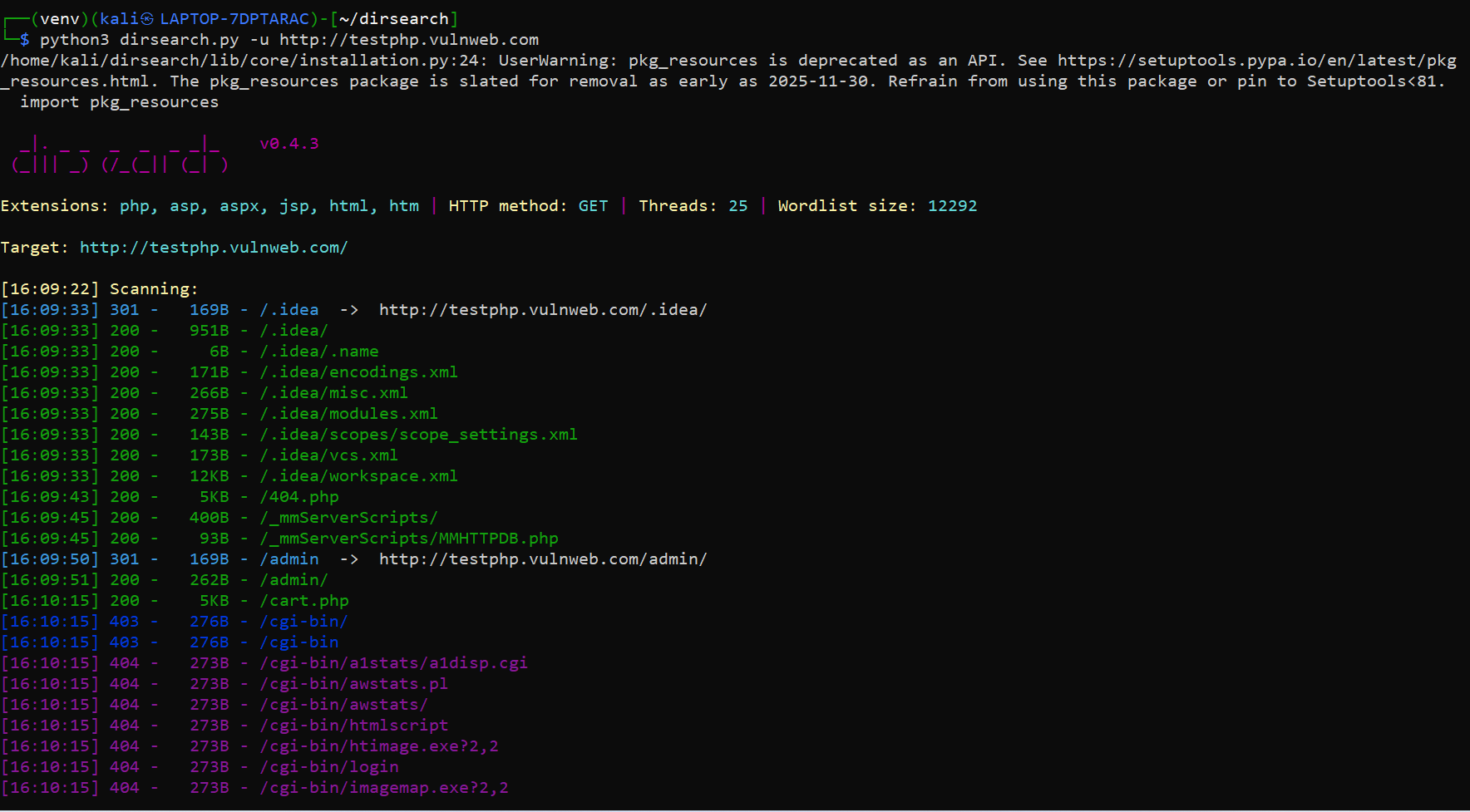
/CVS/ 301/200

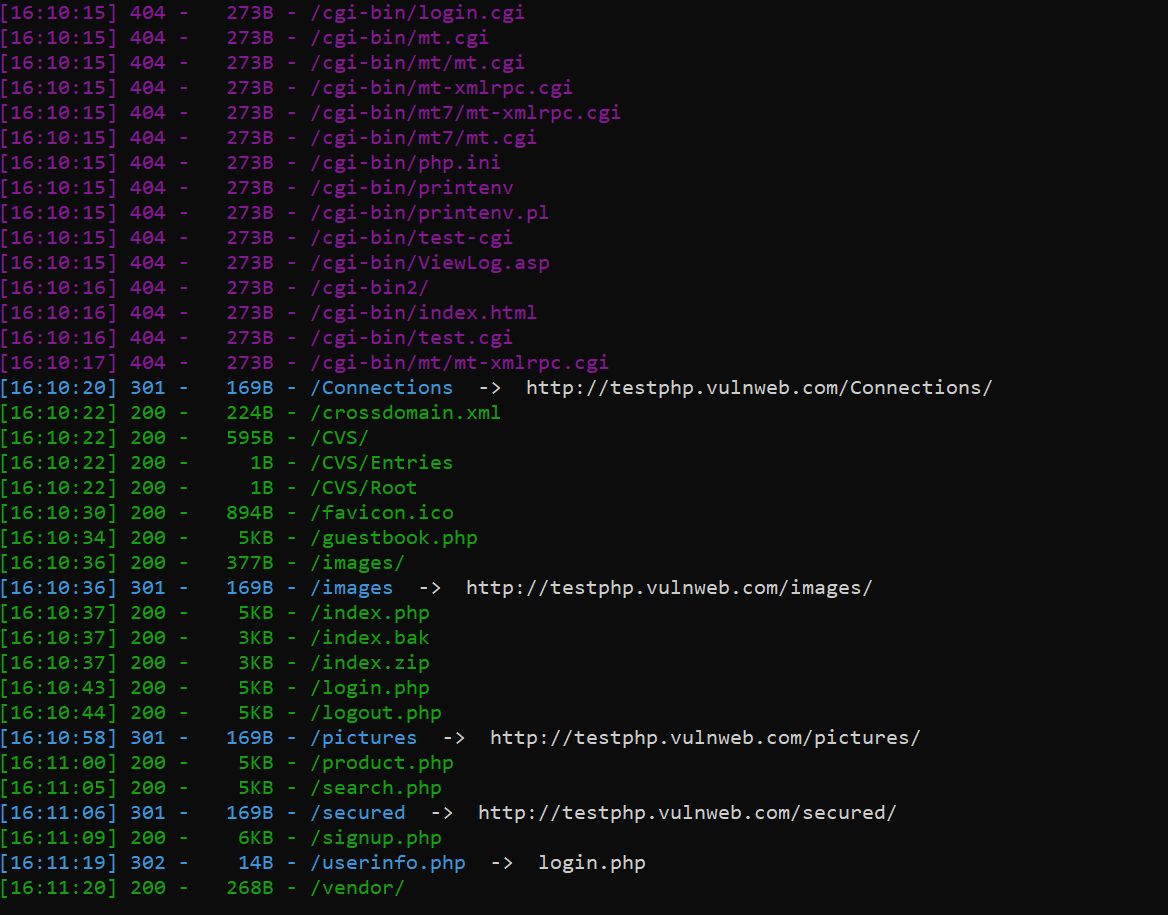
/crossdomain.xml 200

/favicon.ico 200

Tools used:dirsearch

Command: python3 dirsearch.py -u <http://testphp.vulnweb.com>





STATUS-200

/.idea/

/.idea/.name

/.idea/encodings.xml

/.idea/misc.xml

/.idea/modules.xml

/.idea/scopes/scope\_settings.xml

/.idea/vcs.xml

/.idea/workspace.xml

/404.php

/\_mmServerScripts/

/\_mmServerScripts/MMHTTPDB.php

/admin/

/cart.php

/crossdomain.xml

/CVS/

/CVS/Entries

/CVS/Root

/favicon.ico

/guestbook.php

/images/

/index.php

/index.bak

/index.zip

/login.php

/logout.php

/product.php

/search.php

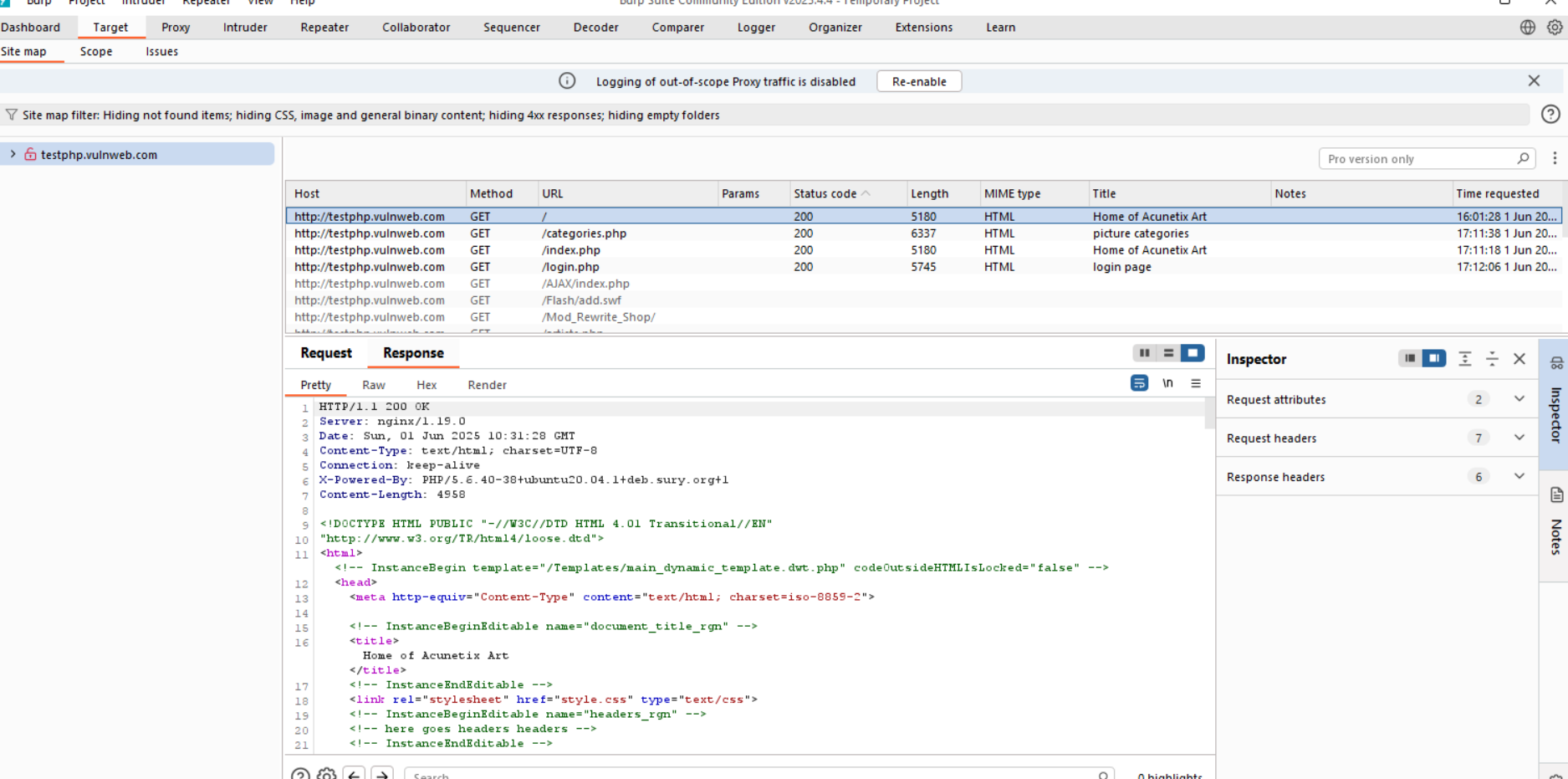
/signup.php

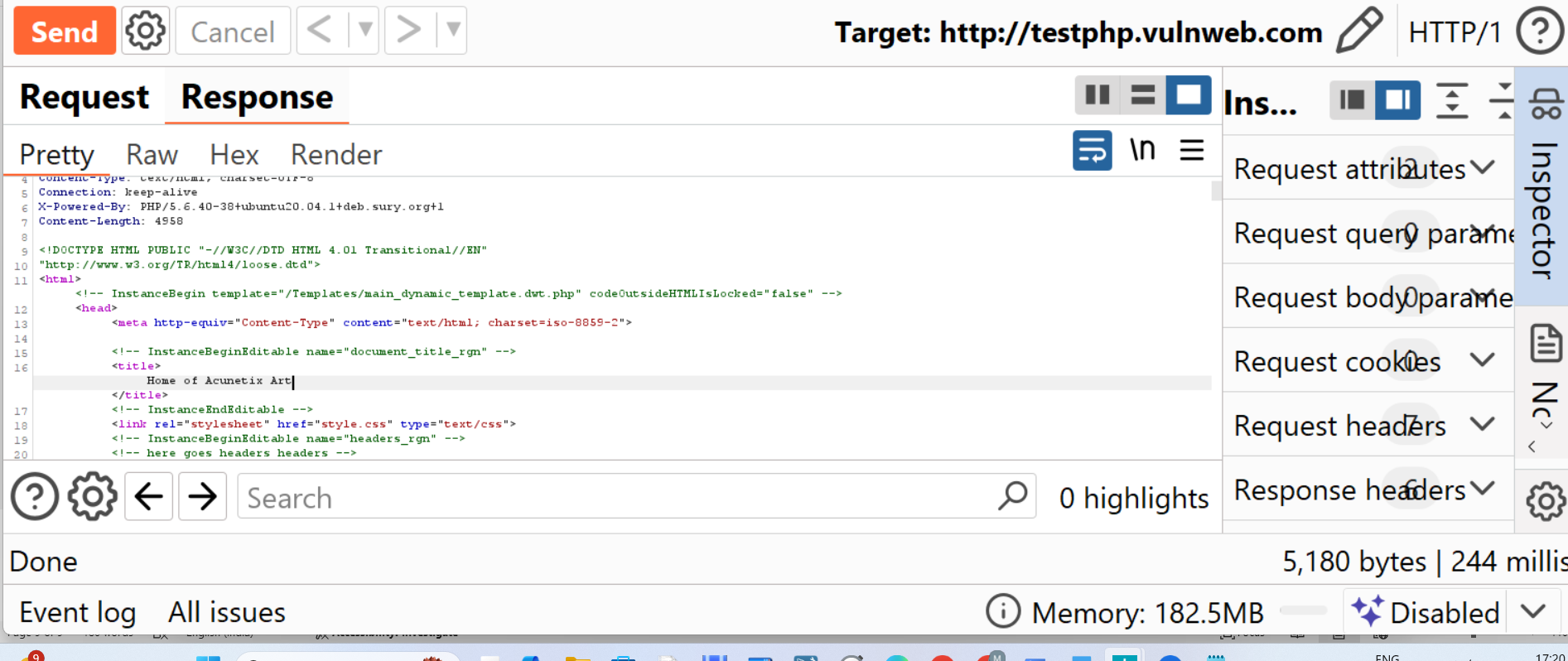
/vendor/

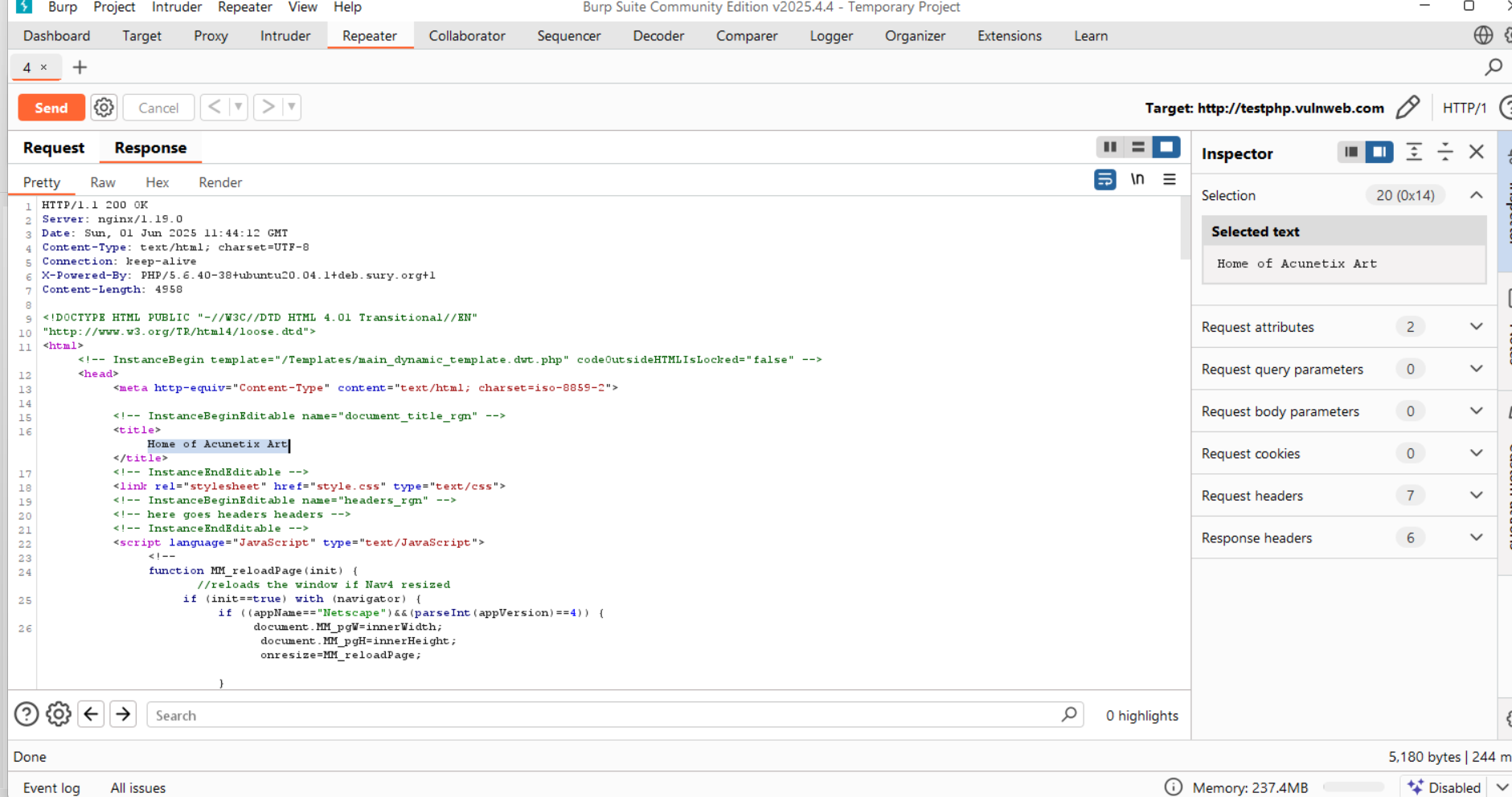
Page titles, parameters, forms:

Tools used: burp suite

Title: Home of Acunetix Art



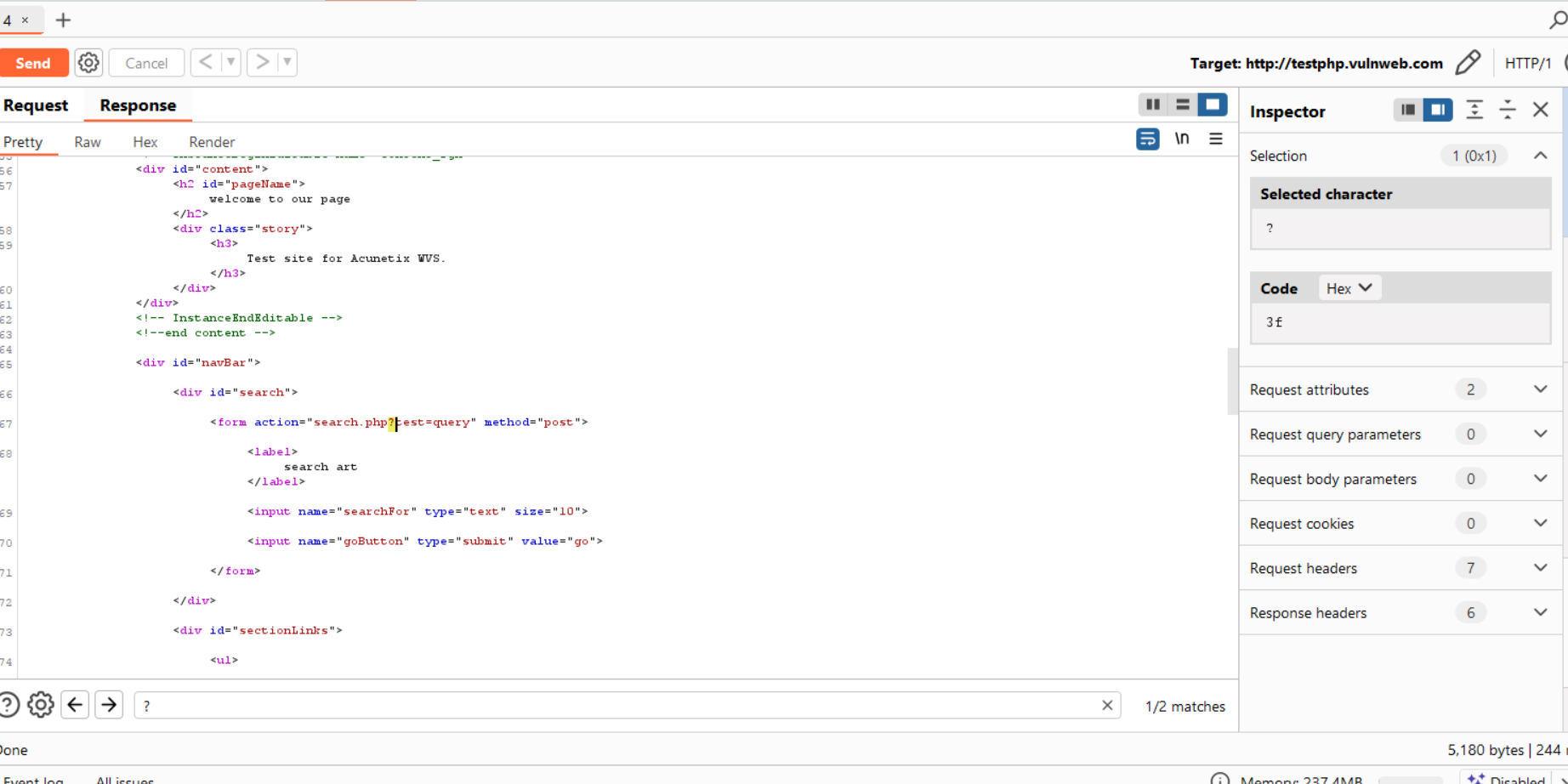




Parameters:

URL- search.php?test=query

Parameter- test



(zoom in for better view)

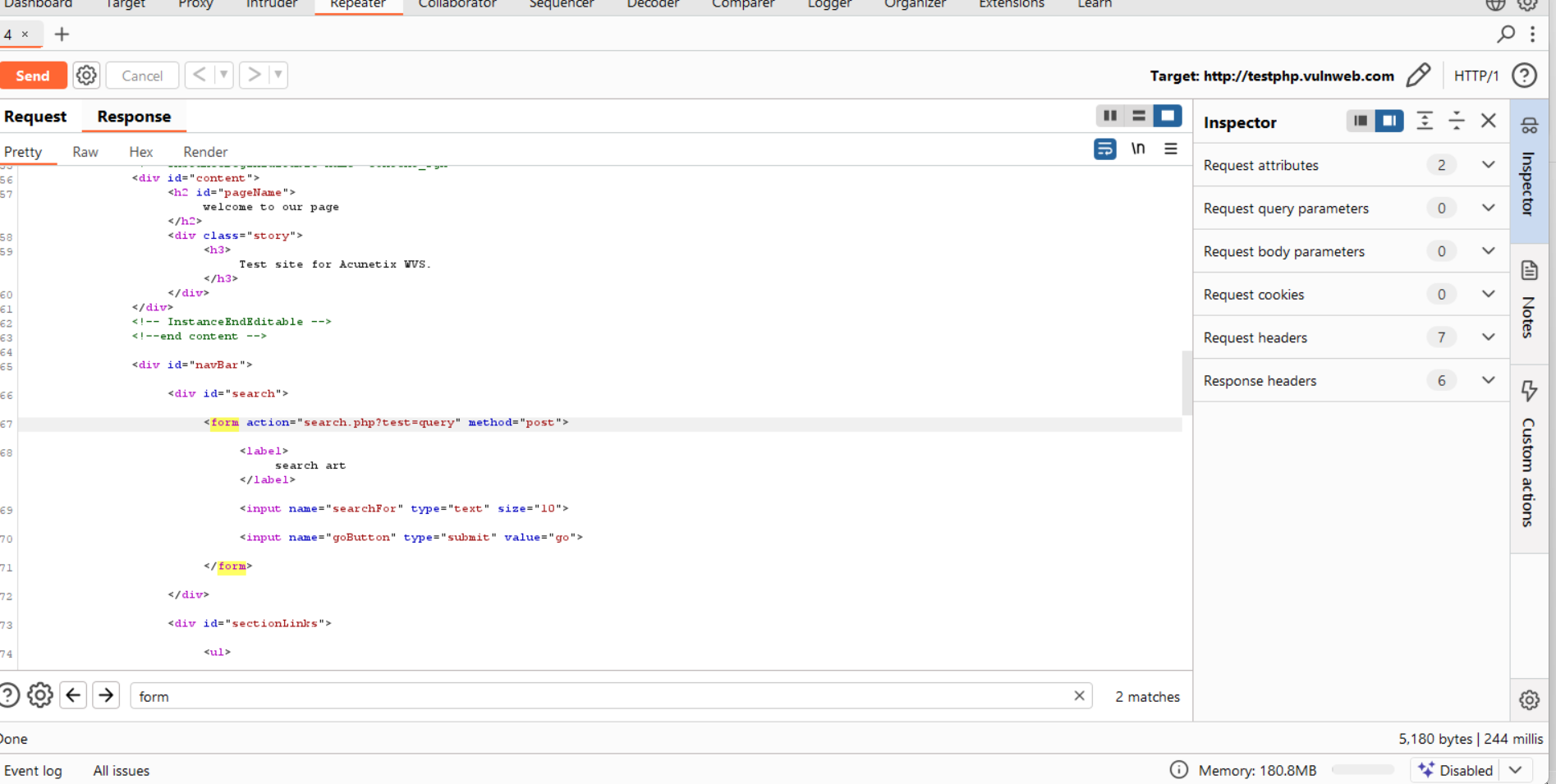
Forms:

Page: search.php?test=query

Method: Post

Input fields: name="searchFor" type="text" size="10"

name="goButton" type="submit" value="go"



**Identity difference between passive and active recon:**

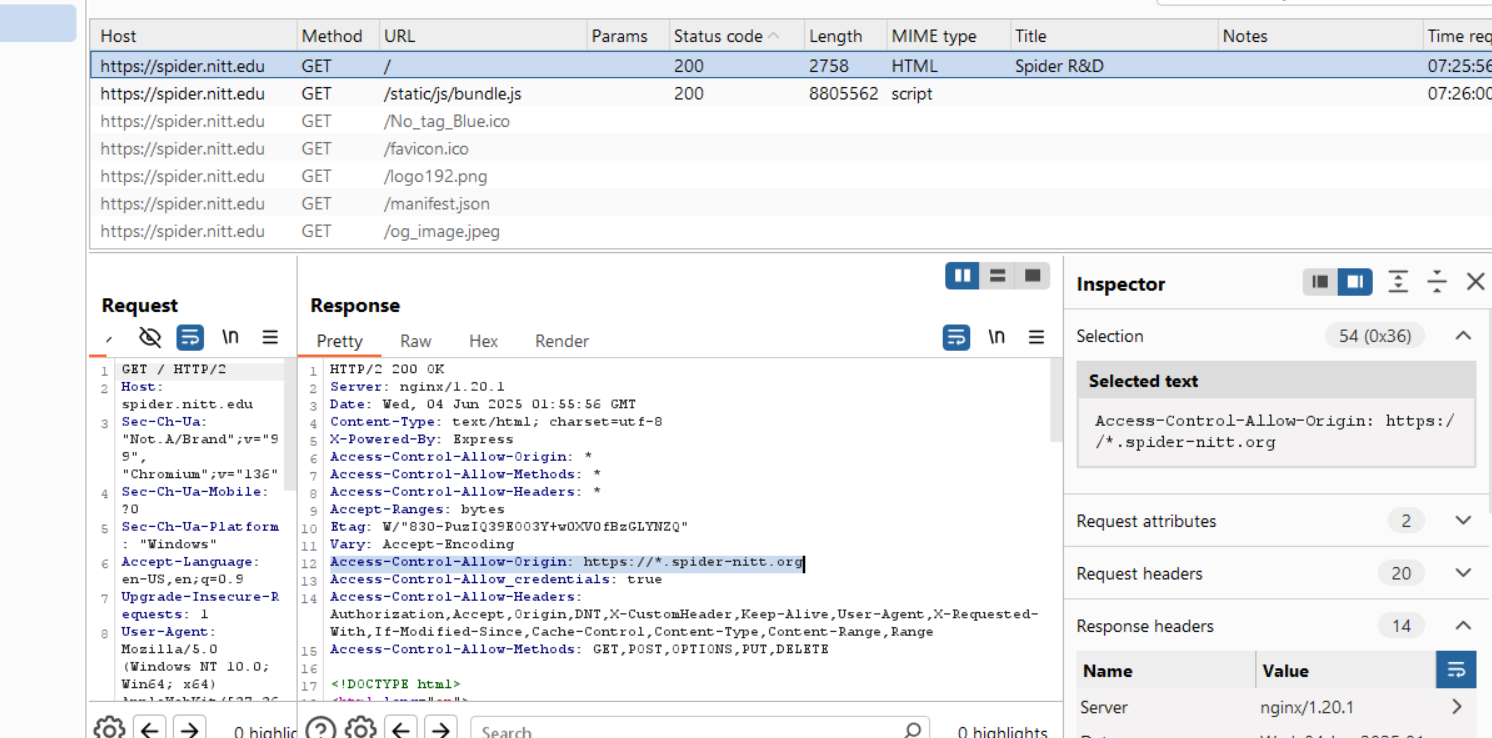
* The key difference between active and passive recon is that active recon gathers information with the target system or network directly while passive recon is that it gathers information with the target system or network from publicly available sources or without any direct interaction.
* In above documentation I have used various active and passive reconnaissance tools such as nslookup, whois and subfinder which are passive tools while dnsrecon, nmap, whatweb, fuff, dirsearch and burp suite which are active tools.

**Level 3: Live Recon & Exploitation**

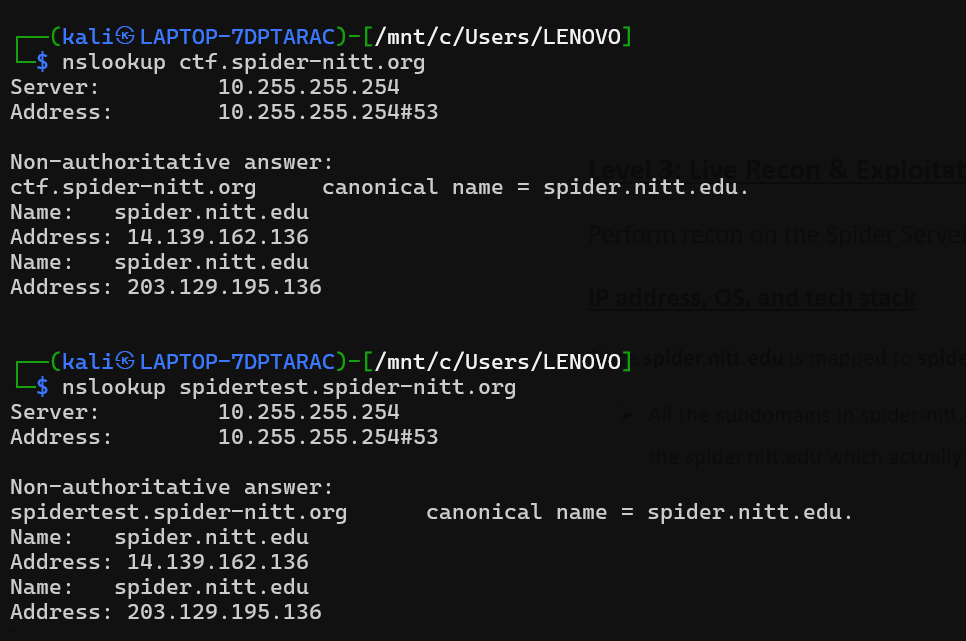
Perform recon on the Spider Server – **spider-nitt.org**

**IP address, OS, and tech stack**

Since **spider.nitt.edu** is mapped to **spider-nitt.org**

* All the subdomains in spider-nitt.org have the canonical domain which is the spider.nitt.edu which actually serves the content of the subdomains.
* Here in spider.nitt.edu Cross Origin Resource Sharing (CORS) header is present which allows any subdomain of spider-nitt.org to access resources over HTTPS requests.

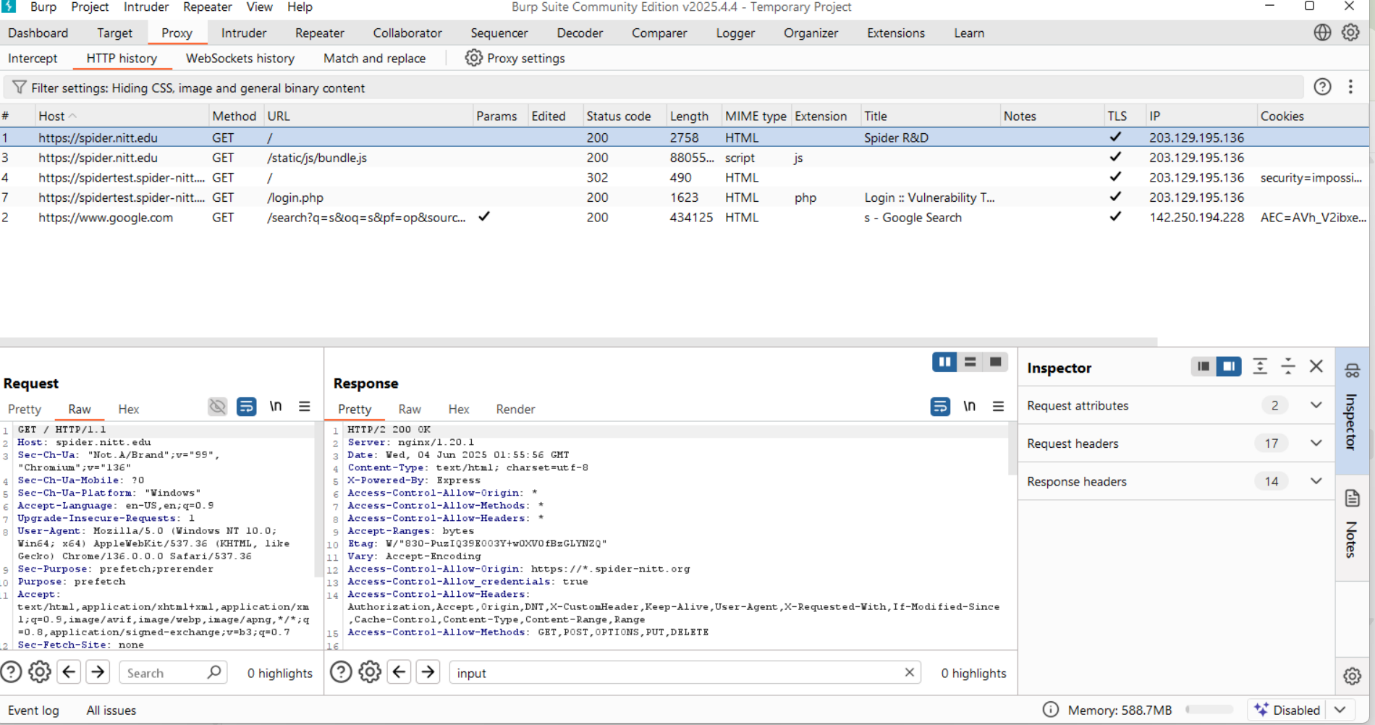
Tools used: nslookup, burpsuite, nmap

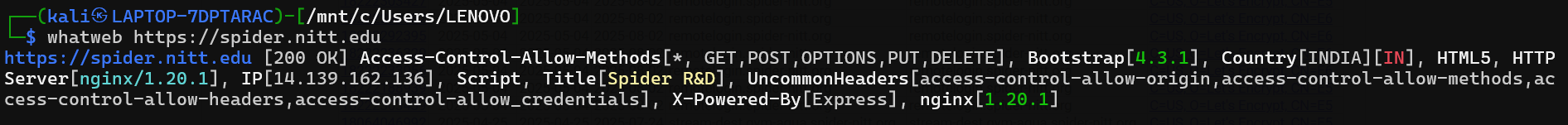


IP address: 14.139.162.136

203.129.195.136 (might be a backup server)

Tech stack:



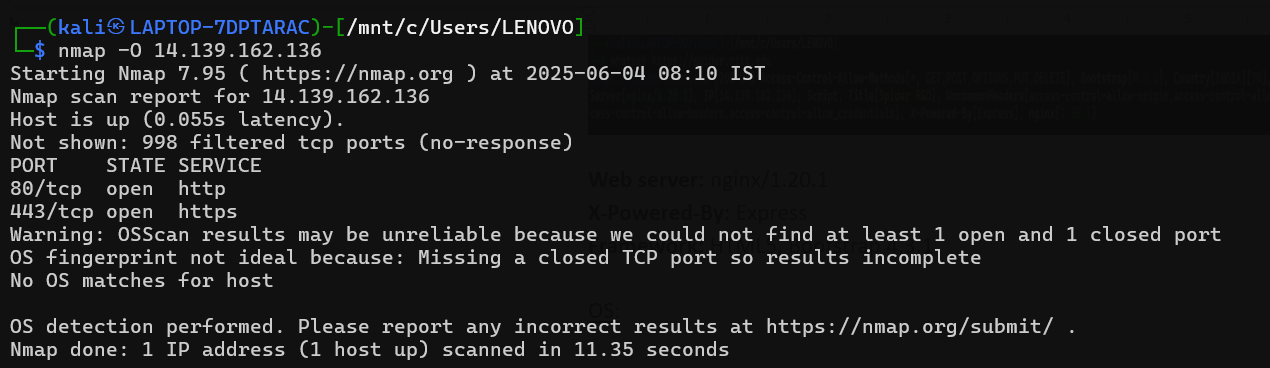


**Web server:** nginx/1.20.1

**X-Powered-By:** Express

**Framework:** HTML5, Bootstrap 4.3.1

OS:

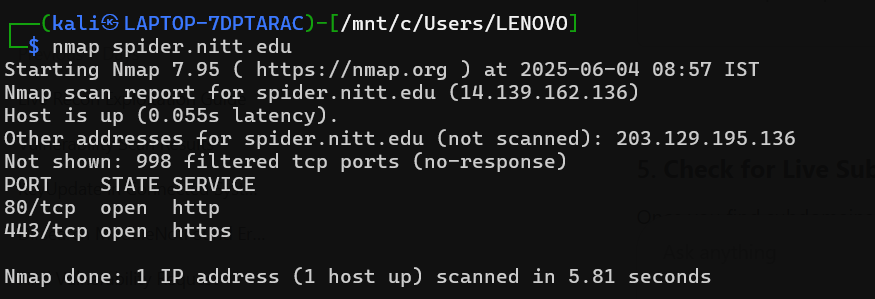


No OS matches found :( but mast likely to be Linux (Ubuntu)

**Services in use:**

Tools used: nmap

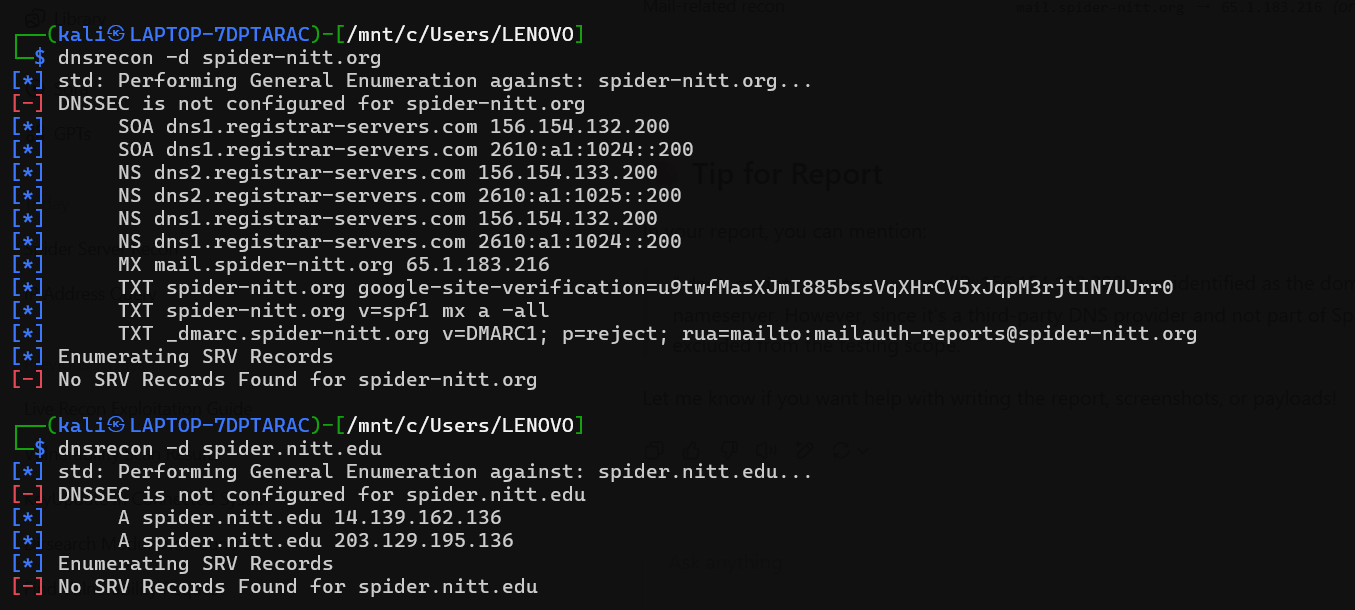
Command: nmap spider.nitt.edu



DNS INFO:

Tools used: dnsrecon

Command: dnsrecon -d spider-nitt.org



**Mail server record**: mail.spider-nitt.org → 65.1.183.216

**Name servers with IPv4 and IPv6**:

dns1.registrar-servers.com 156.154.132.200, 2610:a1:1024::200

dns2.registrar-servers.com 156.154.133.200, 2610:a1:1025::200

**SOA**: dns1.registrar-servers.com

**google-site-verification**=u9twfMasXJmI885bssVqXHrCV5xJqpM3rjtIN7UJrr0

**Hidden subdomains:**

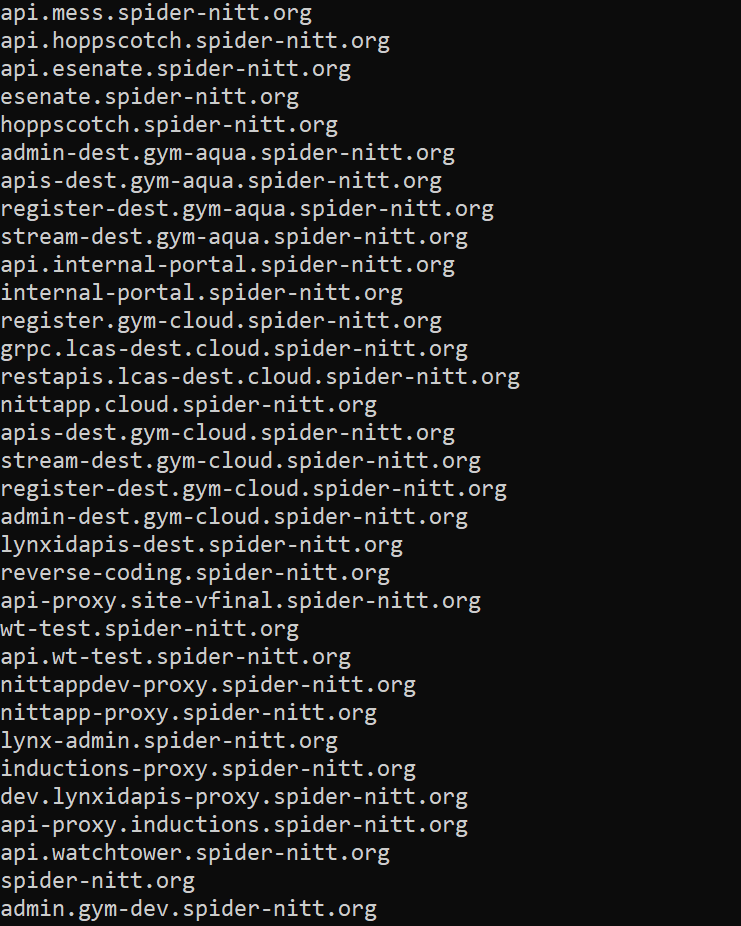
* Subdomains can be found using crt.sh or tools such as subfinder, assetfinder, etc...

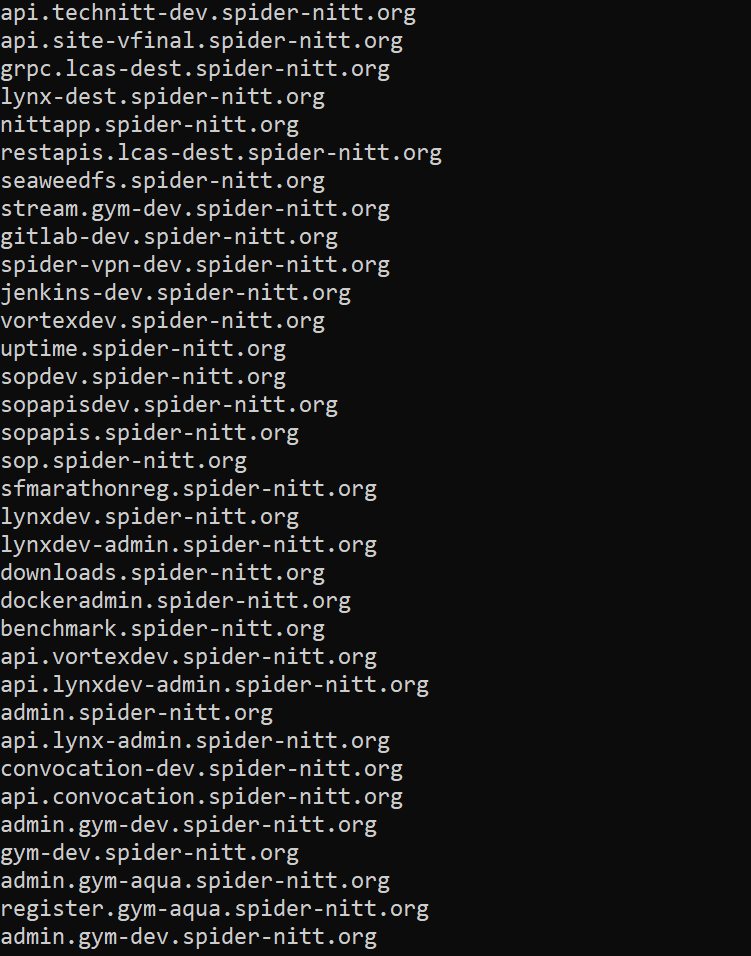
Tools used: crt.sh, assetfinder

* Assetfinder is an effective tool in finding subdomains.

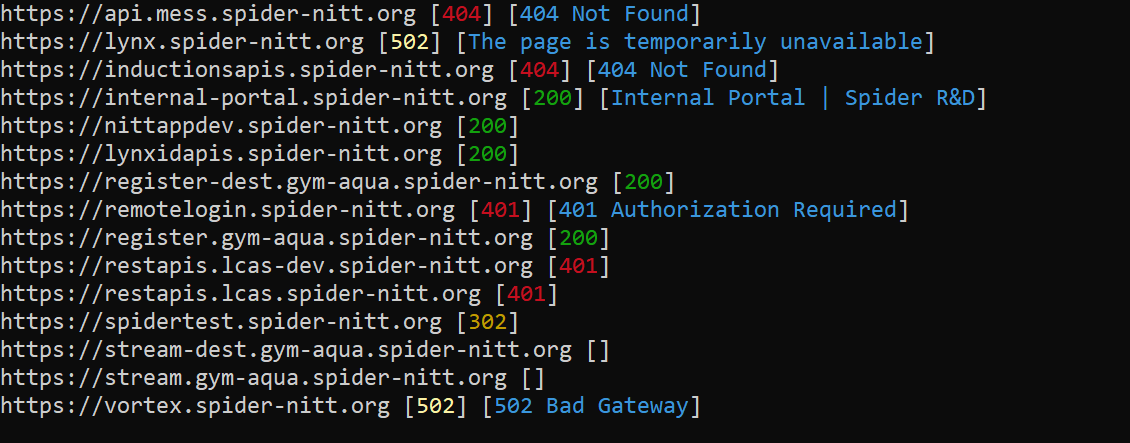
Command used: assetfinder - - subs-only spider-nitt.org



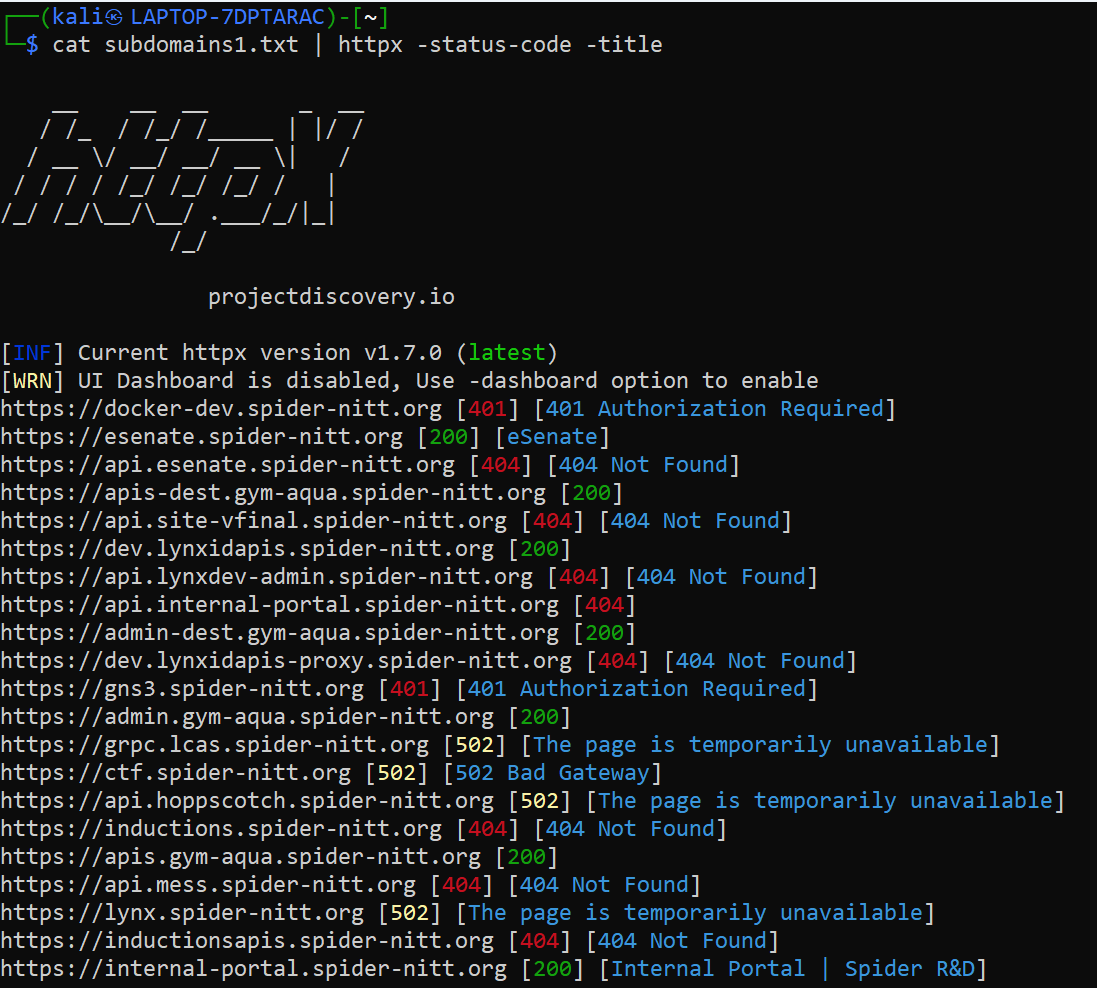






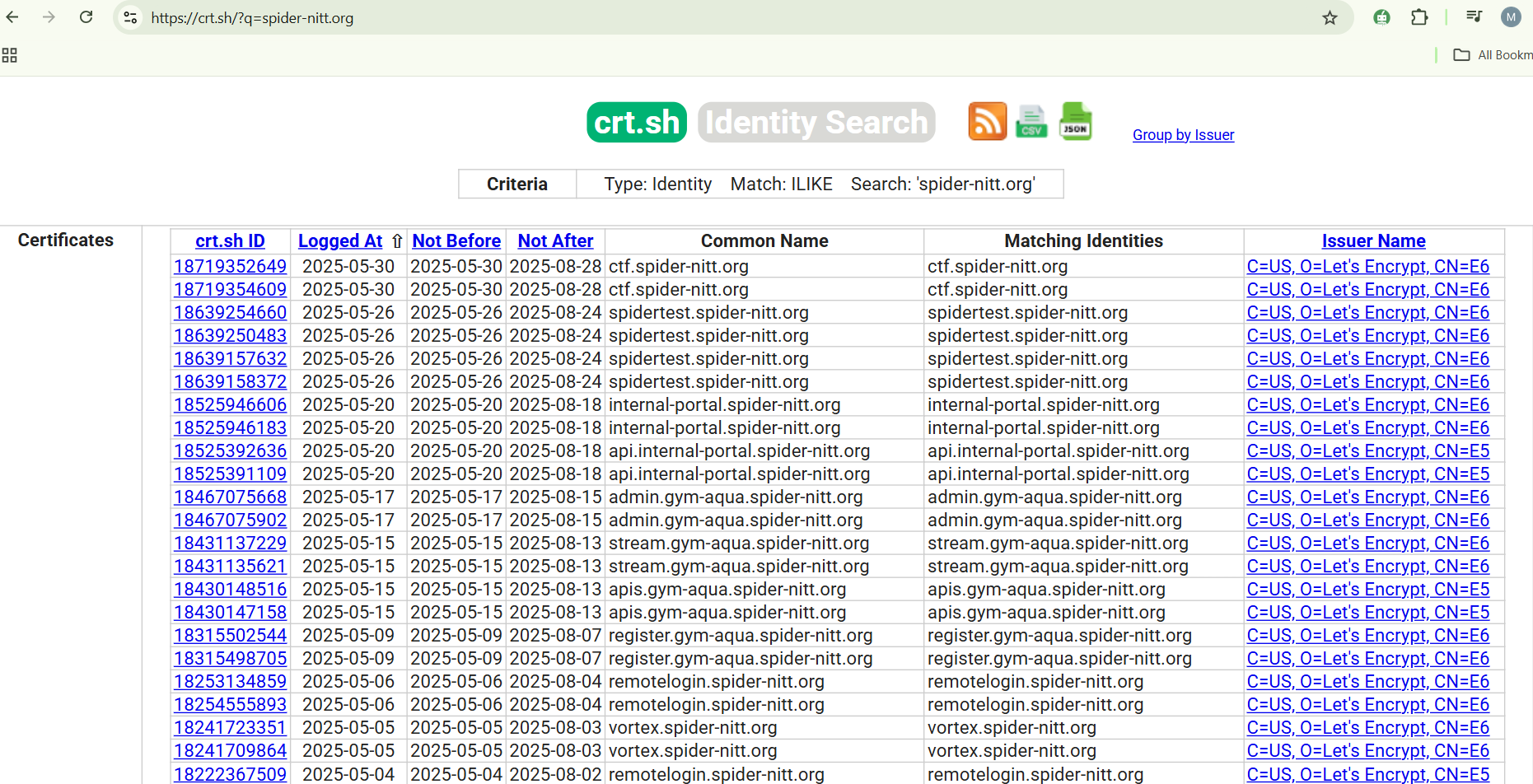
* Then using **httpx** tool we can find which subdomains are live and responding which are the active subdomains and the rest are inactive.

Command used: cat subdomains1.txt | httpx -status-code -title



* On looking at the live subdomains, there lies an interesting subdomain which is the <https://spidertest.spider-nitt.org> which hosts the vulnerable app which is the Damn Vulnerable Web Application(DVWA).
* Even in crt.sh, as we can see that **spidertest.spider-nitt.org** shows in the top since it is logged recently which made my searching easier.

<https://crt.sh/?q=spider-nitt.org>

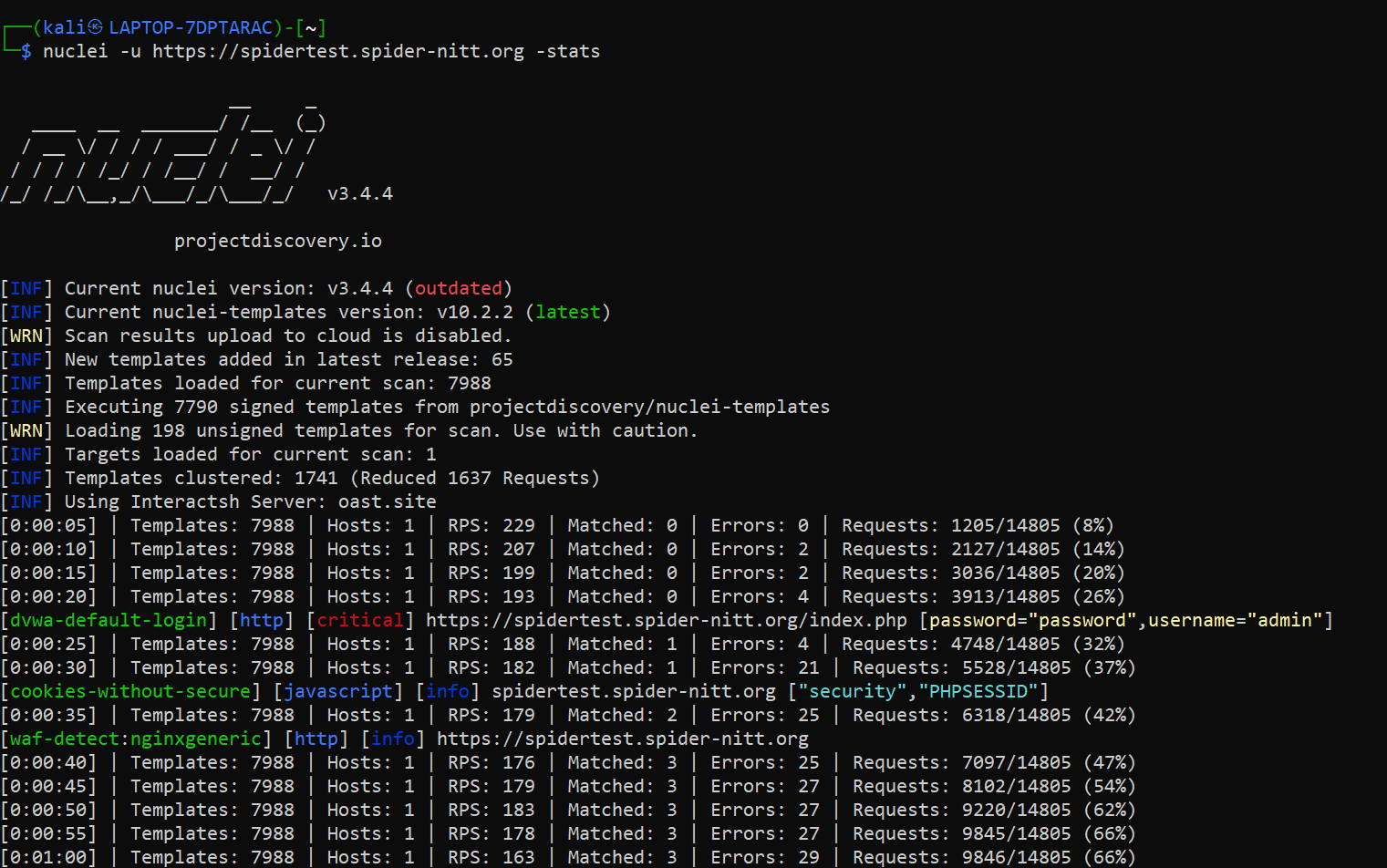


* So, since we have found that spidertest.spider-nitt.org has the vulnerable

app we’ll try exploiting it.

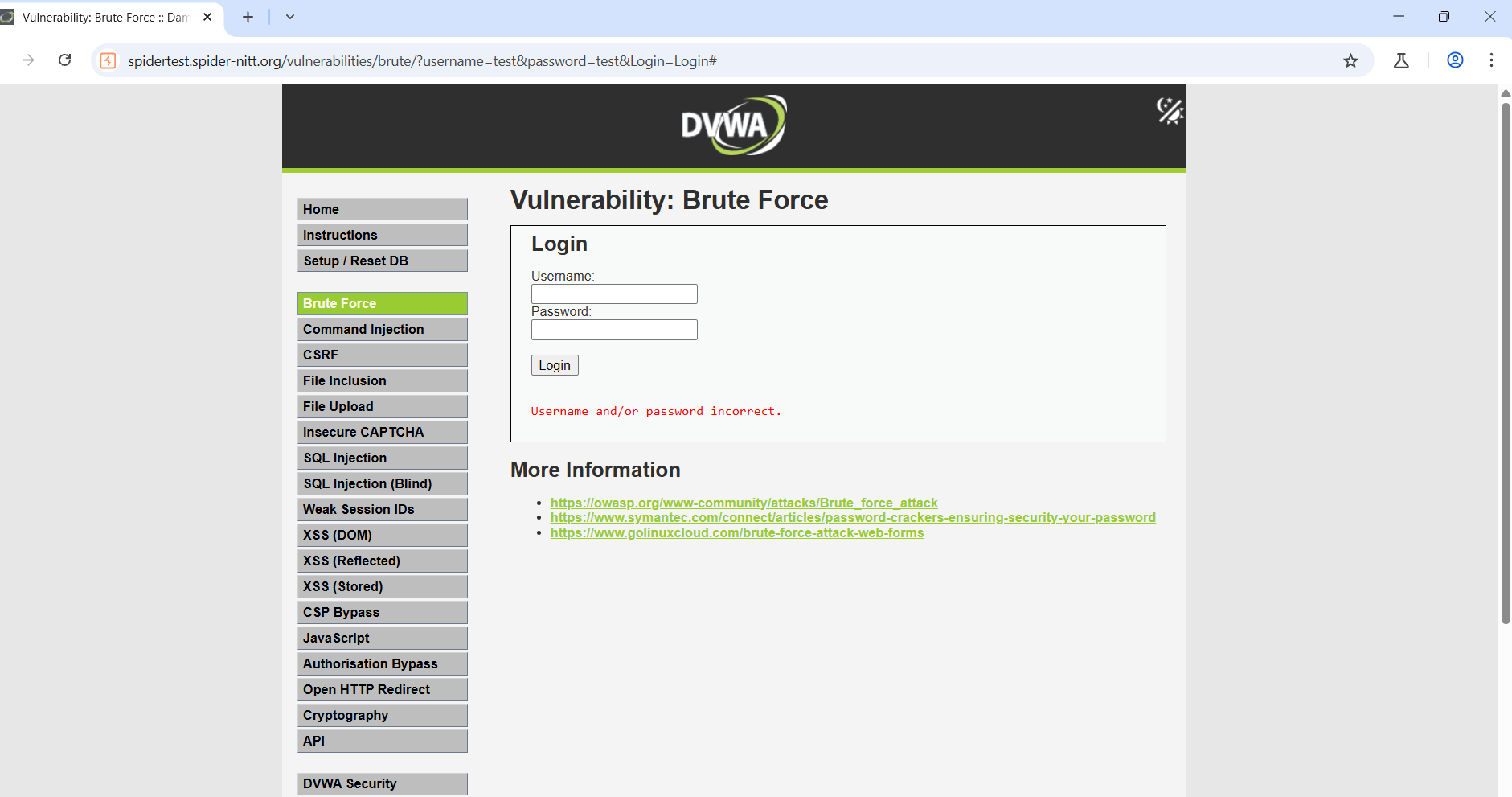
**EXPLOITATION:**

* I have used nuclei which is a vulnerable scanner and I have scanned it on the subdomain which has the vulnerable application. So now I got to know the login username and password.

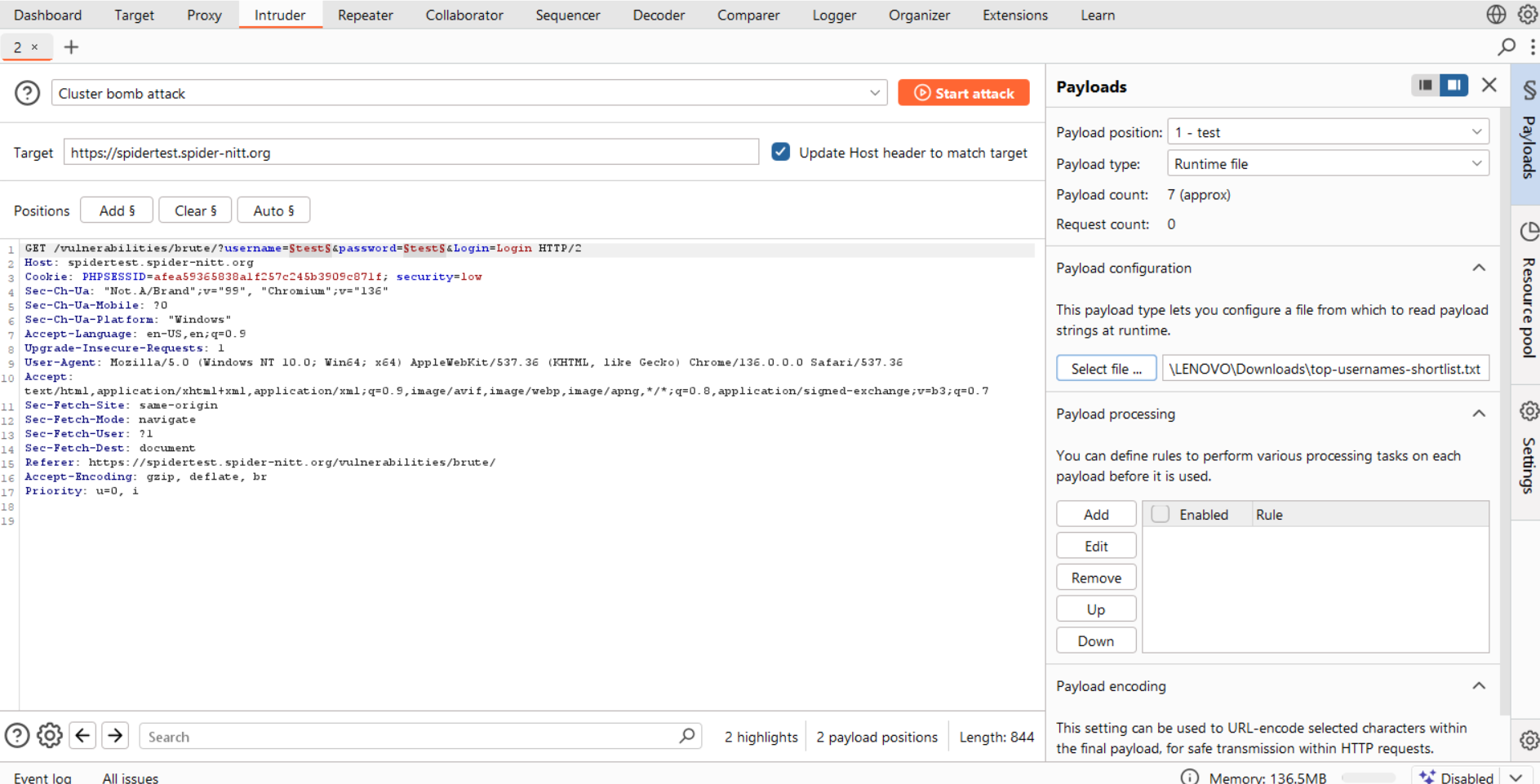


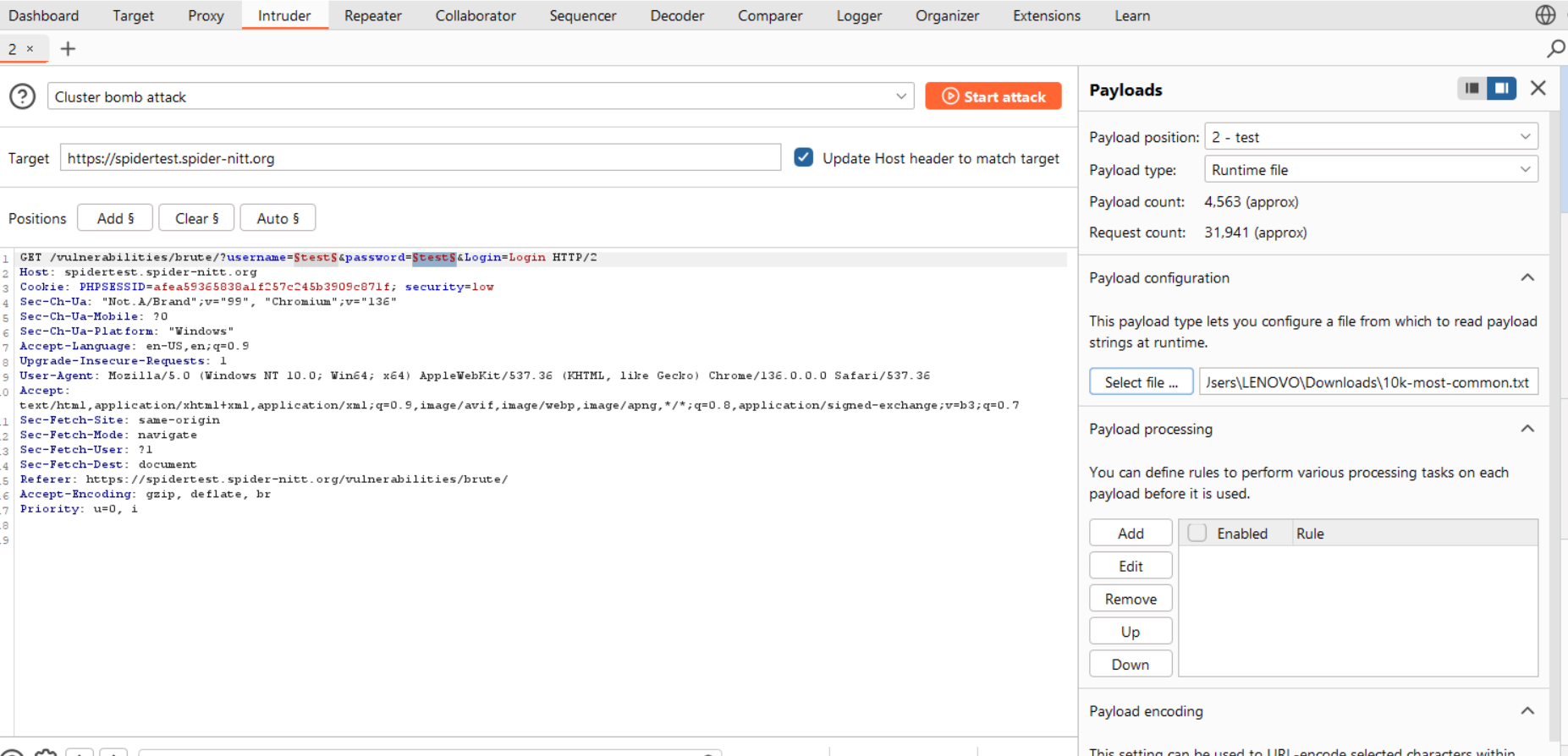
**Brute Force:**

* Now we have tried to login using a random username and password test. Since it’s the wrong username and password it won’t allow me. So I’ll brute force to find the username and password from the common wordlist of username and password.



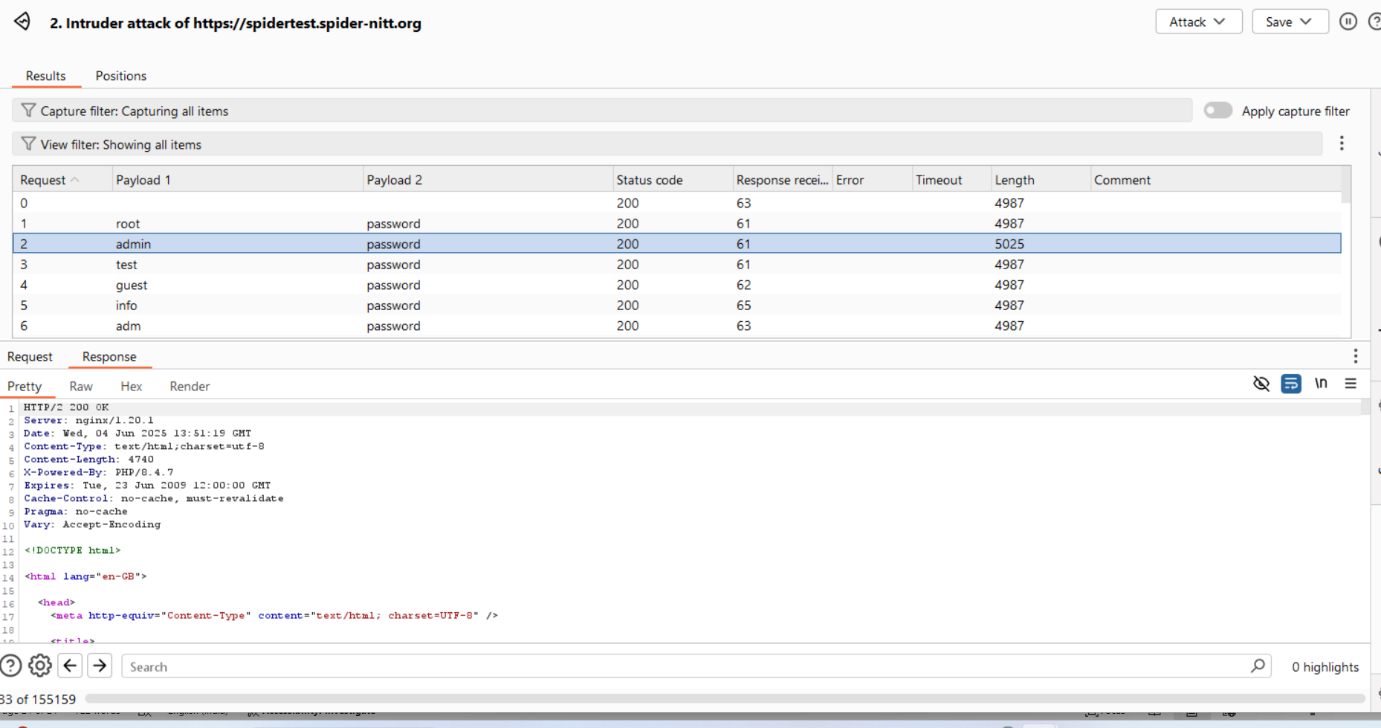
* I got the login response in burp suite. I’ll send this response to the intruder and start my cluster bomb attack on the target. I have downloaded payloads so I’ll be attaching in the intruder payloads and it ll start to check each and every credentials.

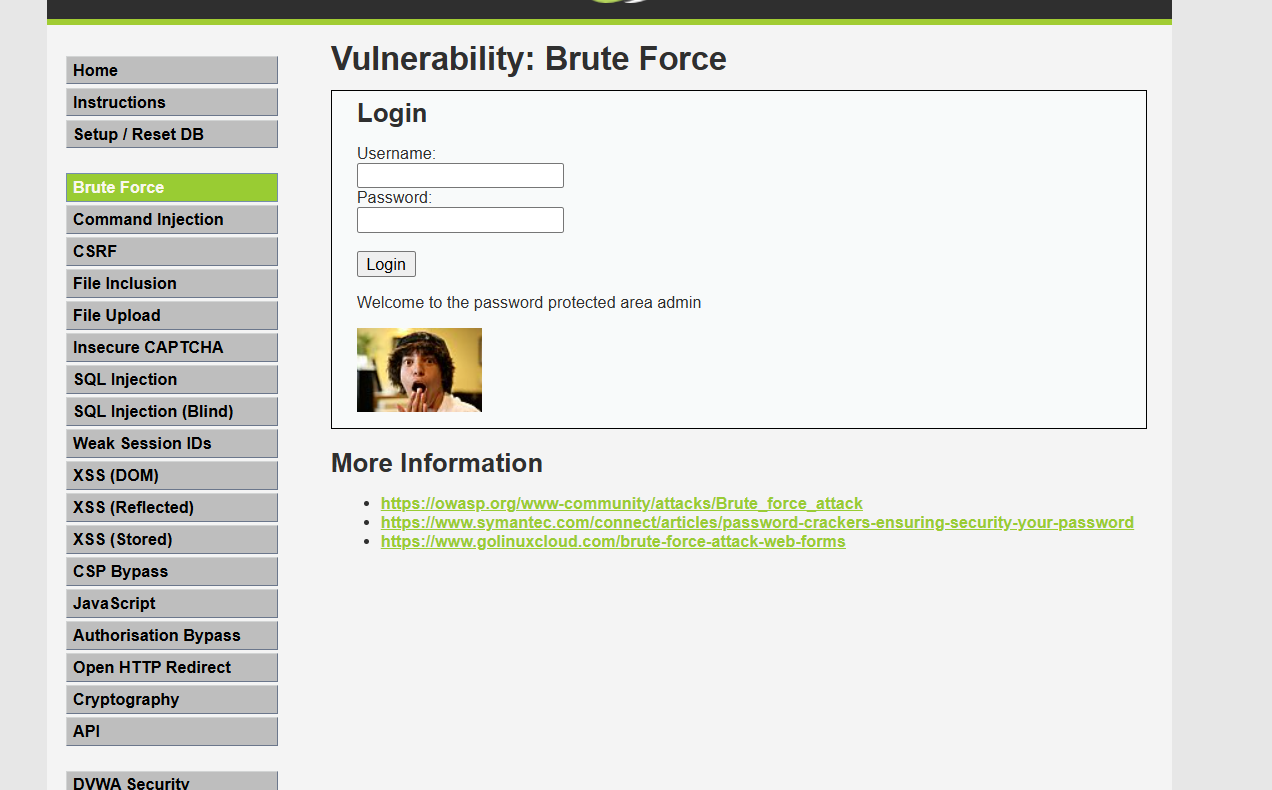




* Now that I have added the payloads which is

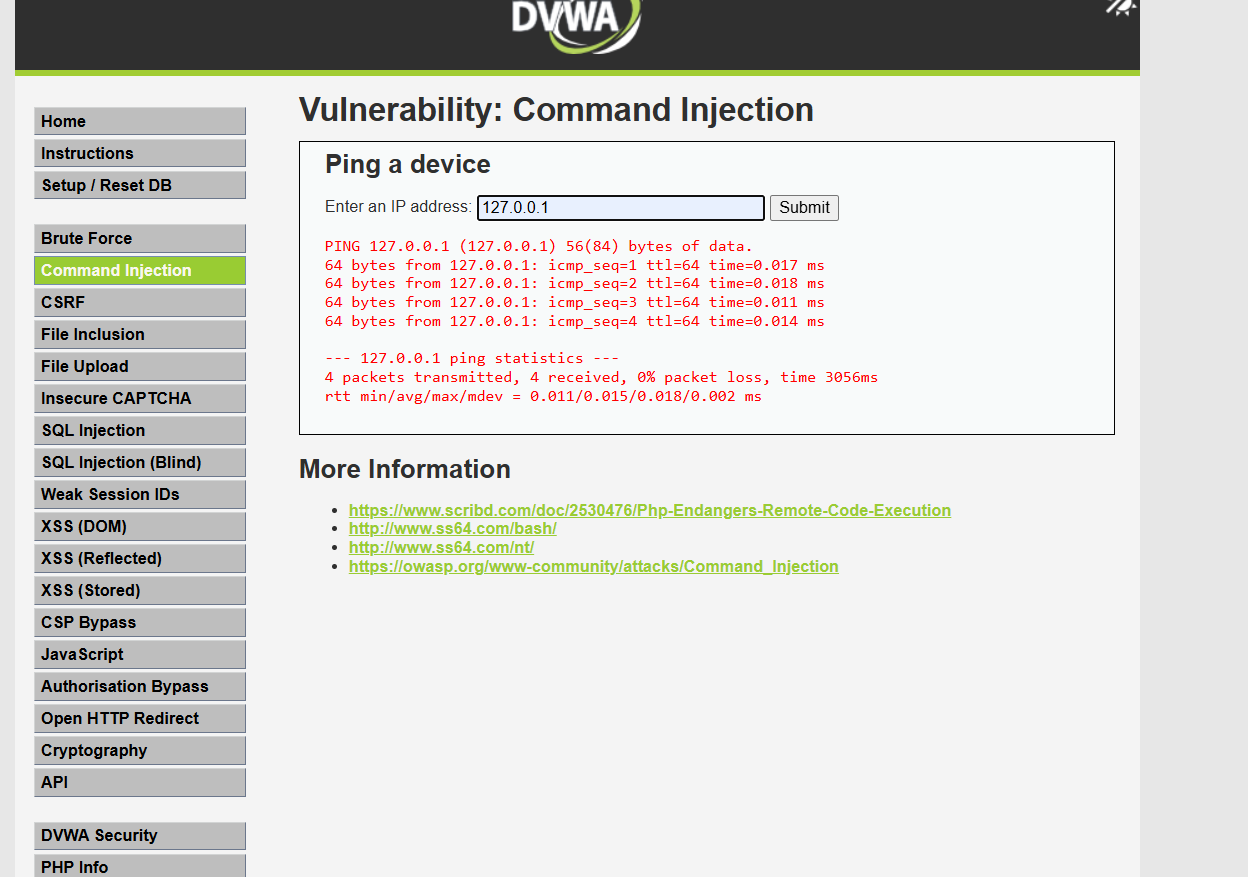
  

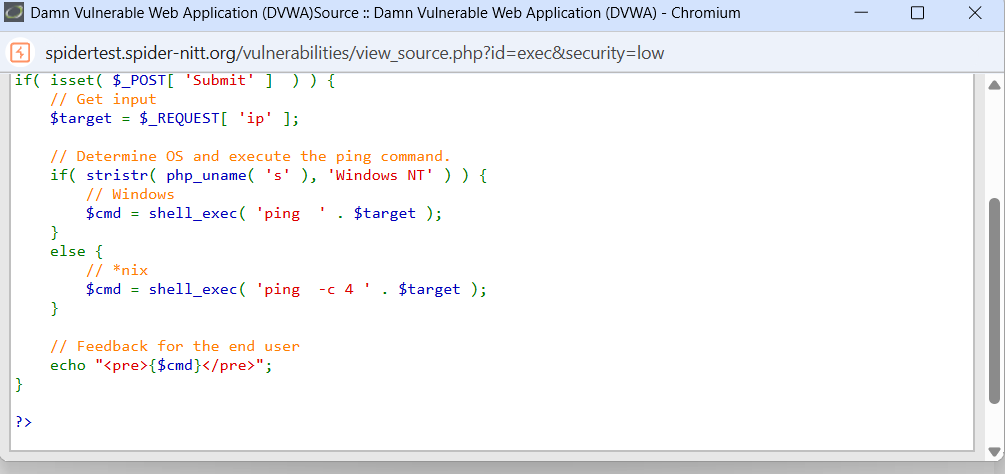
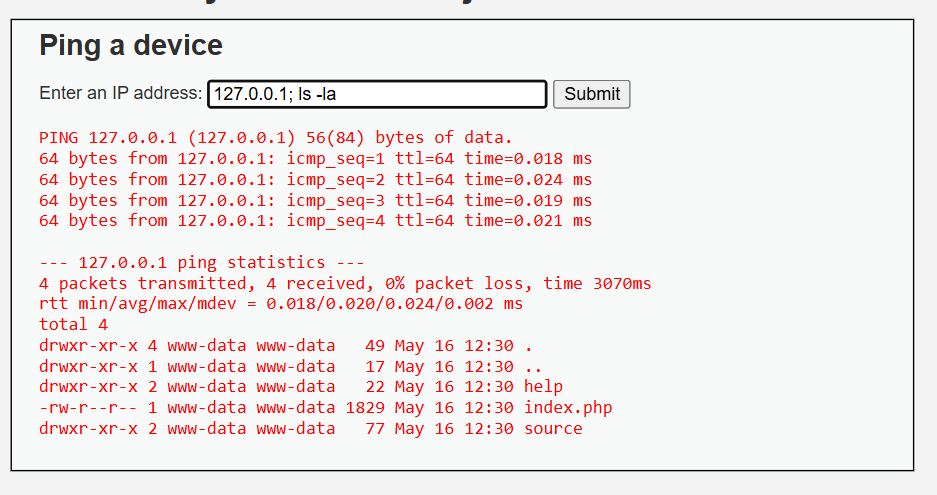


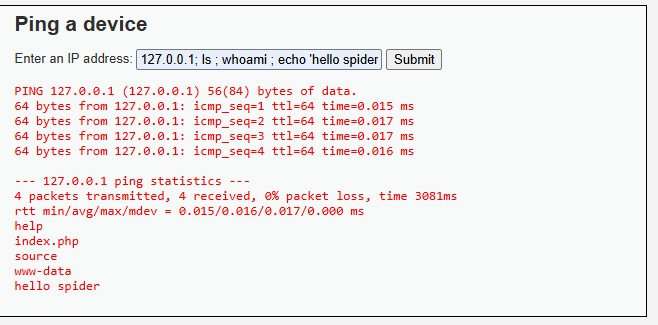
* Once we started attacking, we could find that the payload1-admin and payload2-password row alone has length 5025 and different from others which is the one we must be interested and it is the correct credentials we are looking for.
* Now we can login using the username and password.

**Command Injection:**

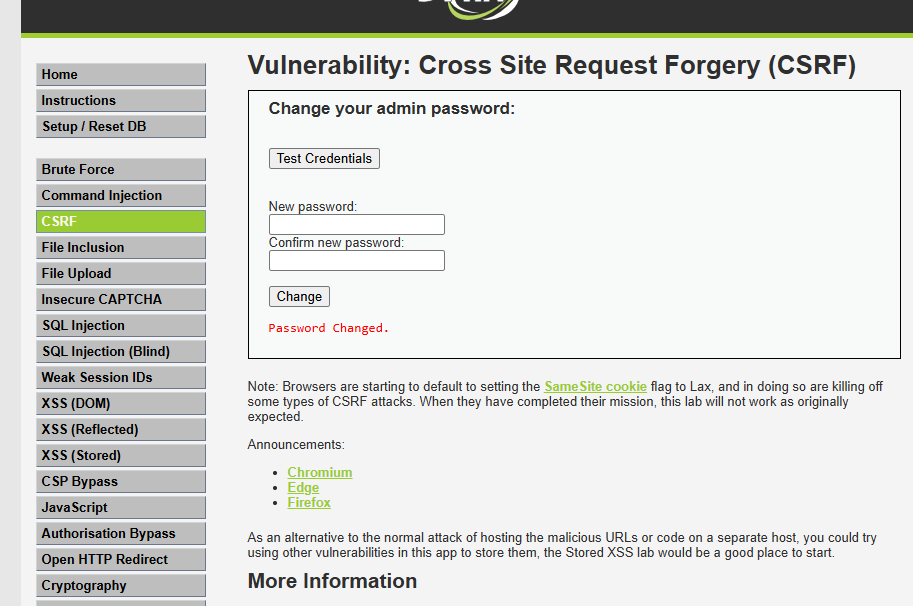
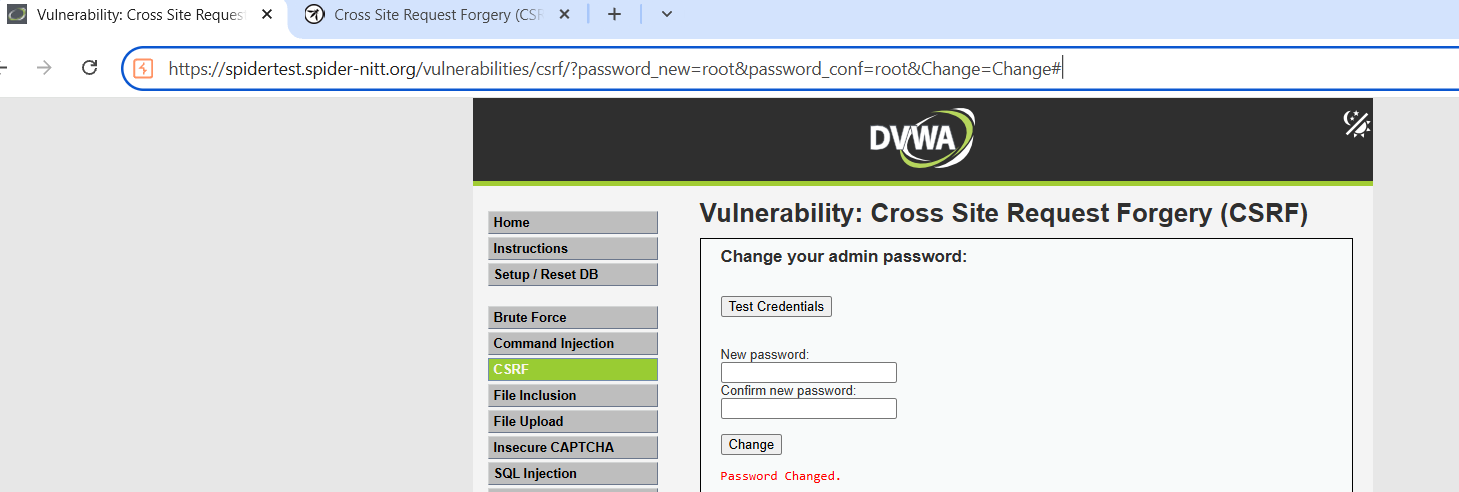
* The purpose of the command injection attack is to inject and execute commands specified by the attacker in the vulnerable application. It is possible to escape out of the designed command and executed unintentional actions. Now we’ll try to that.

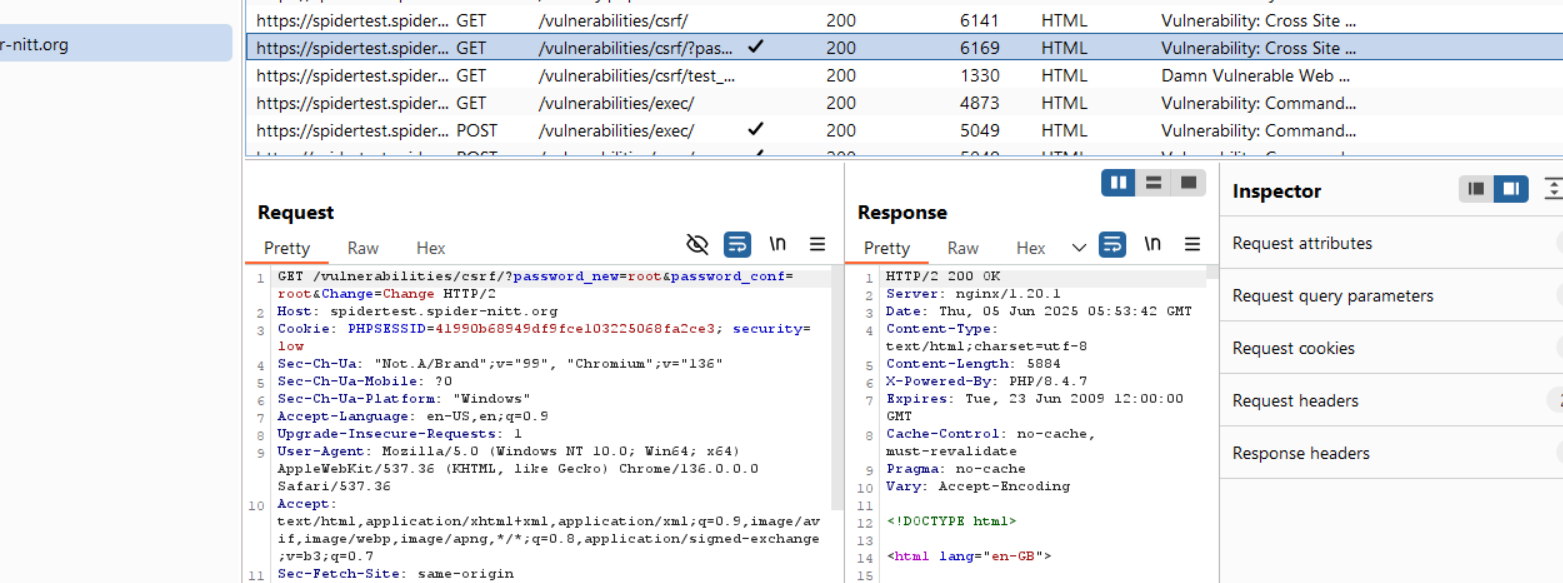


* We have tried pinging the loopback address and we got successfully pinged. Now we’ll add some commands to get some useful information.
* Since they are executing the commands directly, we can add ls at last to view files after the ip address using ‘;’ .
* We could see some directories help, source,etc and a file index.php. Now we’ll try to access it.
* In the above image we could see the contents of index.php and the command used here is **127.0.0.1; cat index.php**
* Some more commands we can add – input given: **127.0.0.1; ls ; whoami ; echo 'hello spider'.** This will ping to the ip and list the files and directories and tell the user name and print a message.



**CSRF:**

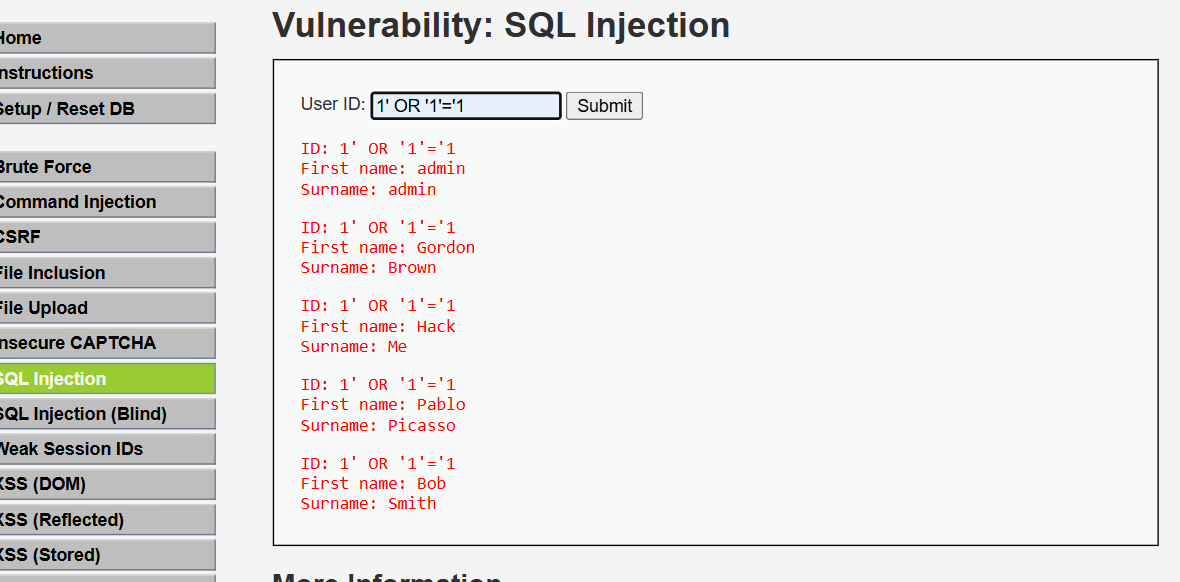
* CSRF is an attack that forces an end user to execute unwanted actions on a web application in which they are currently authenticated. So, we’ll try to change the password and a link can be created so that we can send this to the target via email or social apps and then if he clicks it, the action will be triggered.
* Now I have changed the password to root.
* We can see this in the URL. So that if a victim clicks this link his password will be changed according to the attacker’s wish.
* We can see the password has been change in burp suite too.



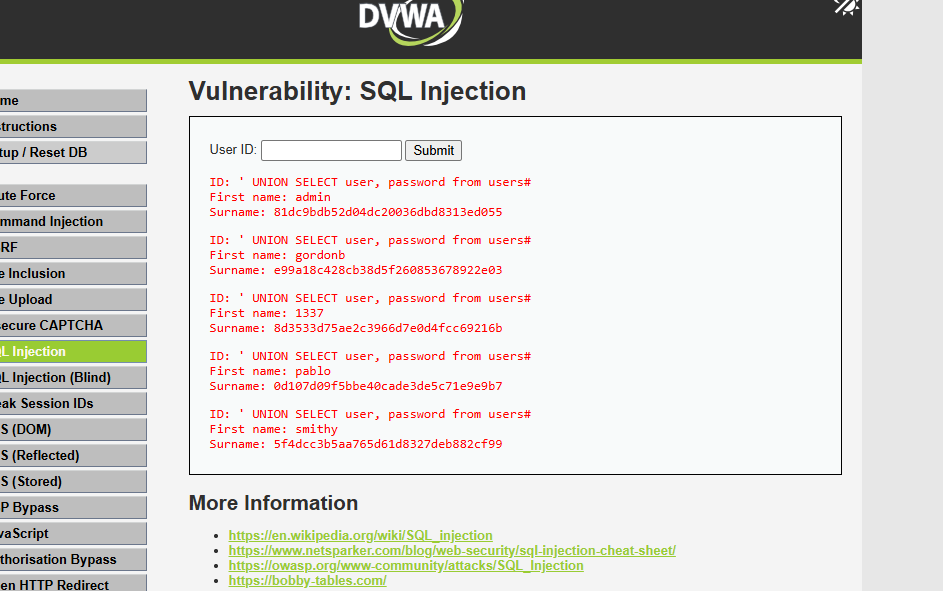
* If I give ‘password’ as the password it is not letting me in. so the new password has changed to root.

**SQL injection:**

* A SQL injection attack consists of insertion or "injection" of a SQL query via the input data from the client to the application. A successful SQL injection exploit can read sensitive data from the database, modify database data, etc...
* Now we can try some injections and try to retrieve some data.

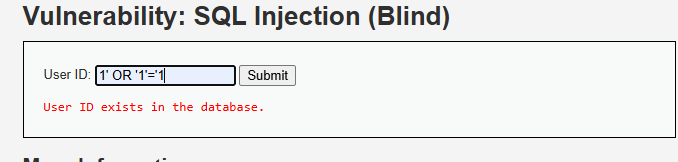
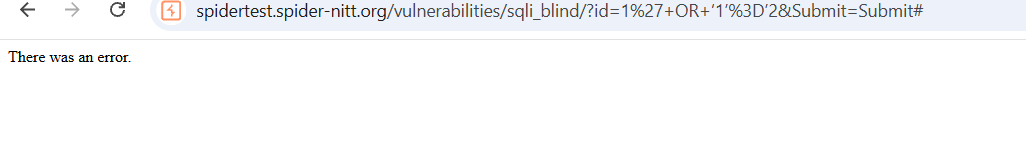


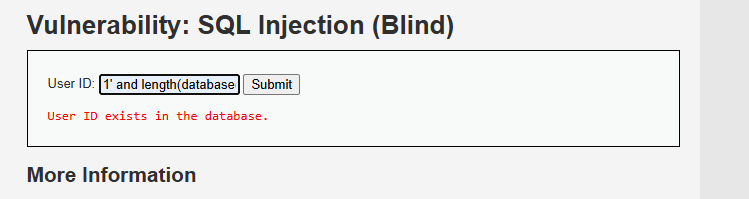
* On using the injection **1’ OR ‘1’=’1** we could not still able to retrive the passwords of the 5 users.
* Now we ll see the source page to find some clues.
* We could see the query which is so vulnerable and doesn’t have any constraints.
* If I give input as **‘ORDER BY 3#**, I’m getting error that unknown column so there are only 2 columns. Next, we’ll try some UNION based injections which is **‘UNION SELECT user, password FROM users#** the basic injection.



* And now we can see the passwords and username.

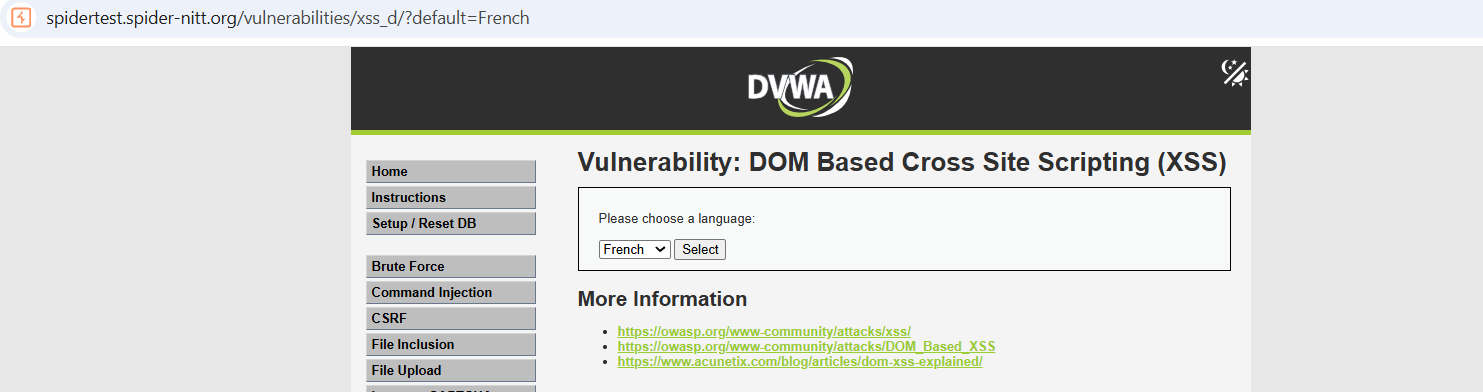
**SQL Injection (Blind):**

* Blind SQL (Structured Query Language) injection is a type of [SQL Injection](https://owasp.org/www-community/attacks/SQL_Injection) attack that asks the database true or false questions and determines the answer based on the applications response.
* If I tried to input **1’ OR ‘1’=’1** it shows user id exists in the database. This statement is true Boolean logic.
* But if I give input as a false logic such as **1’ OR ‘1’=’2** it shows error msg since it’s a false statement.
* On trying **1' and length(database())=4#** we get user id exists but if we try for other lengths, we will get missing user id which means that length of the database is 4.

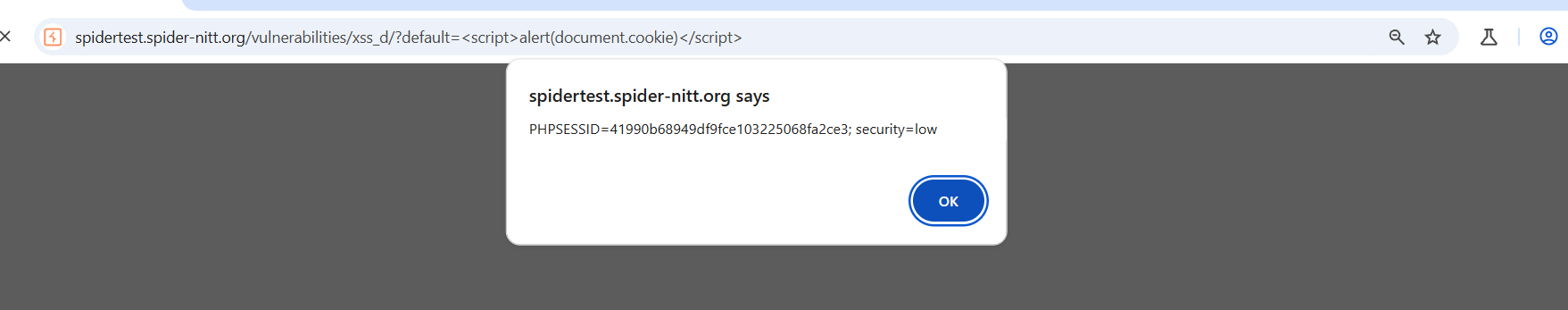


**XSS (DOM):**

* "Cross-Site Scripting (XSS)" attacks are a type of injection problem, in which malicious scripts are injected into the otherwise benign and trusted web sites. DOM based XSS is an XSS attack wherein the attack payload is executed as a result of modifying the DOM “environment” in the victim’s browser used by the original client-side script, so that the client-side code runs in an “unexpected” manner.
* Now I ll modify the URL to send it to the victim and when he opens the URL it will execute differently according to my modification. The below page is the normal page which contains language page options.

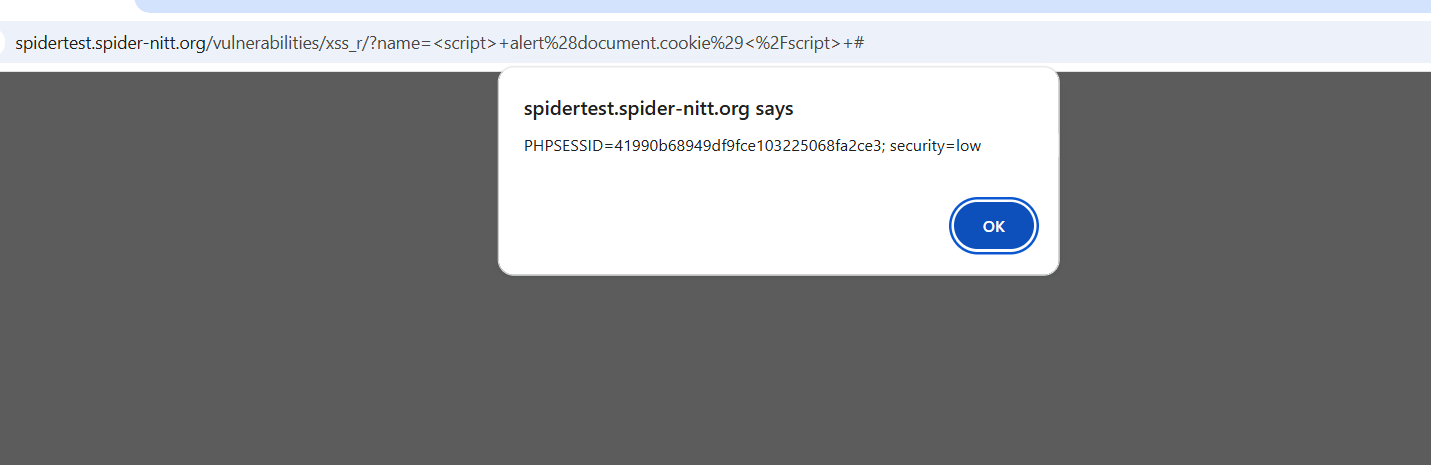


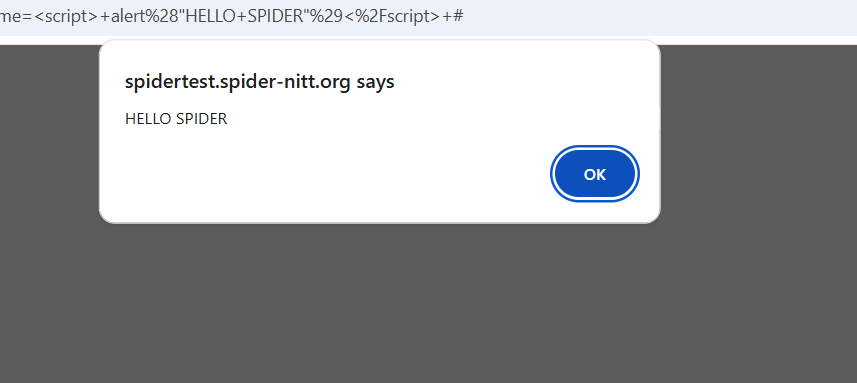
* When I change to URL by typing this **<script>alert(document.cookie)</script>** in place of **French**. It ll give the cookie set (php sessid and security level).



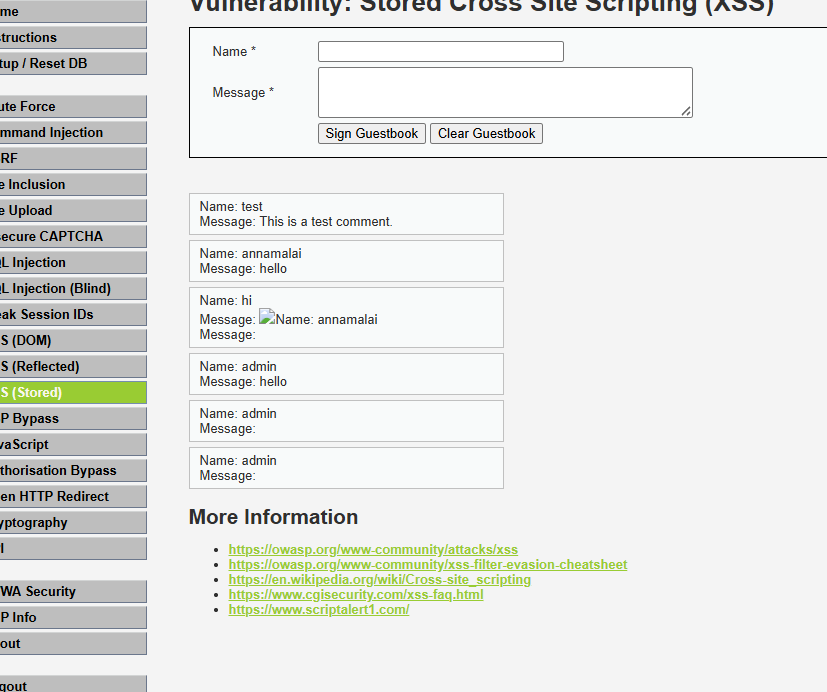
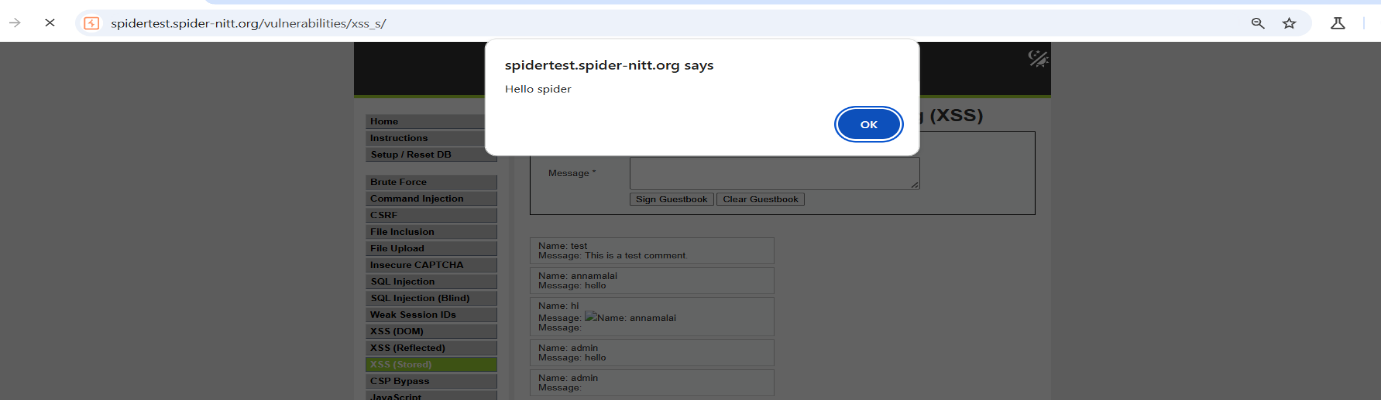
* Like wise we can exploit it and send the modified URL to user in which the website act according to our modification.

**XSS (Reflected):**

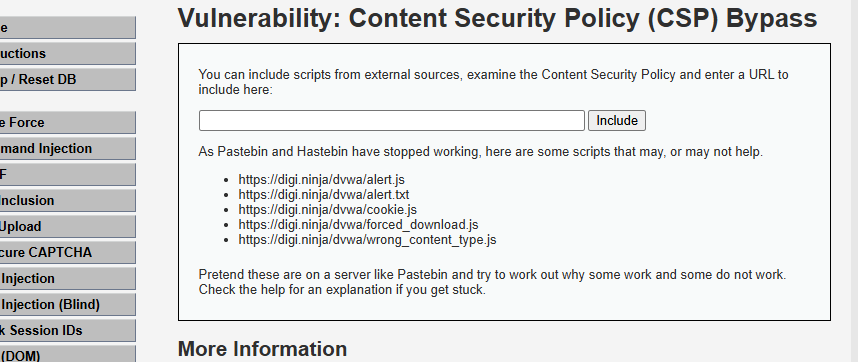
* Reflected attacks are those where the injected script is reflected off the web server, such as in an error message, search result, or any other response that includes some or all of the input sent to the server as part of the request.
* The objective of this to find the cookie of logged in user. So, we have seen in previous exploitation on how to get it. Same we’ll use in here.
* Now I’ll inject **<script> alert(document.cookie)</script>** in the search bar.
* We got the cookie of the current user.
* Now if I gave injection as **<script> alert(“HELLO SPIDER”)</script>. I’ll** get something like this-

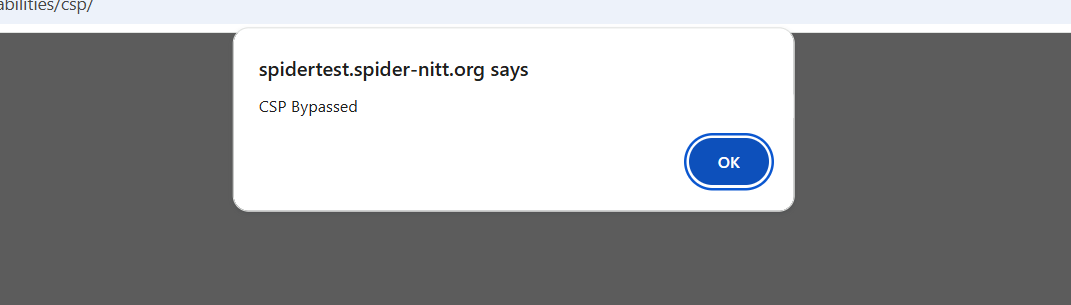


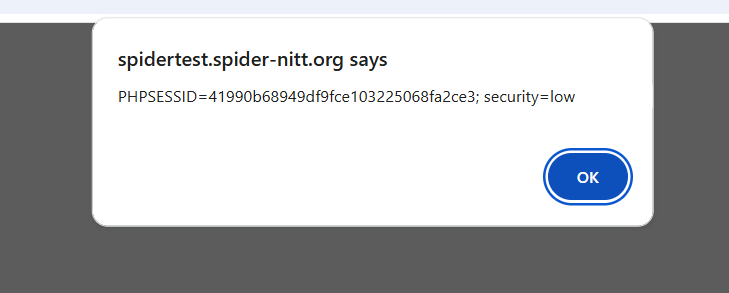
**XSS (Stored):**

* Stored attacks are those where the injected script is permanently stored on the target servers, such as in a database, in a message forum, visitor log, comment field, etc.
* Now I’m going to create a guestbook of name admin and message **<script>alert (“Hello spider”) </script>** which will get stored in that page and whenever I tried to visit the page it will show me the message ‘Hello spider’.
* Now if I try to visit the page it will show as

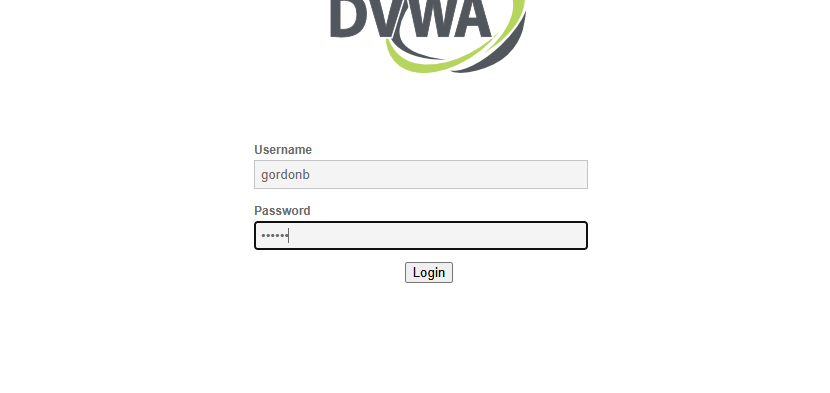
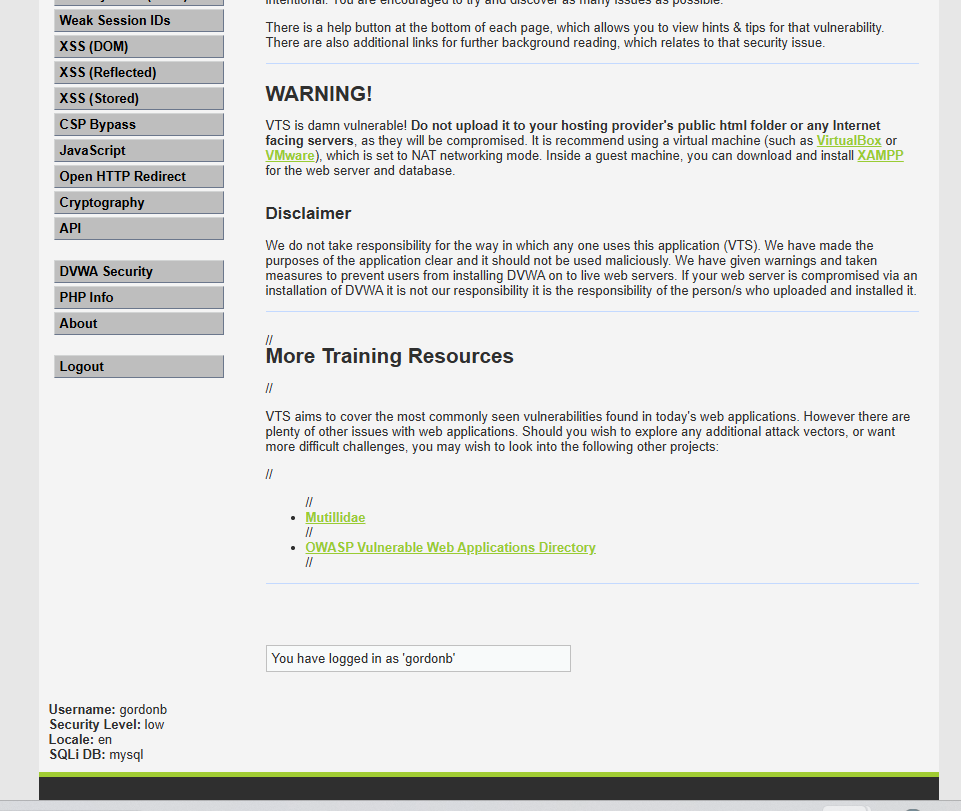
**CSP BYPASS:**

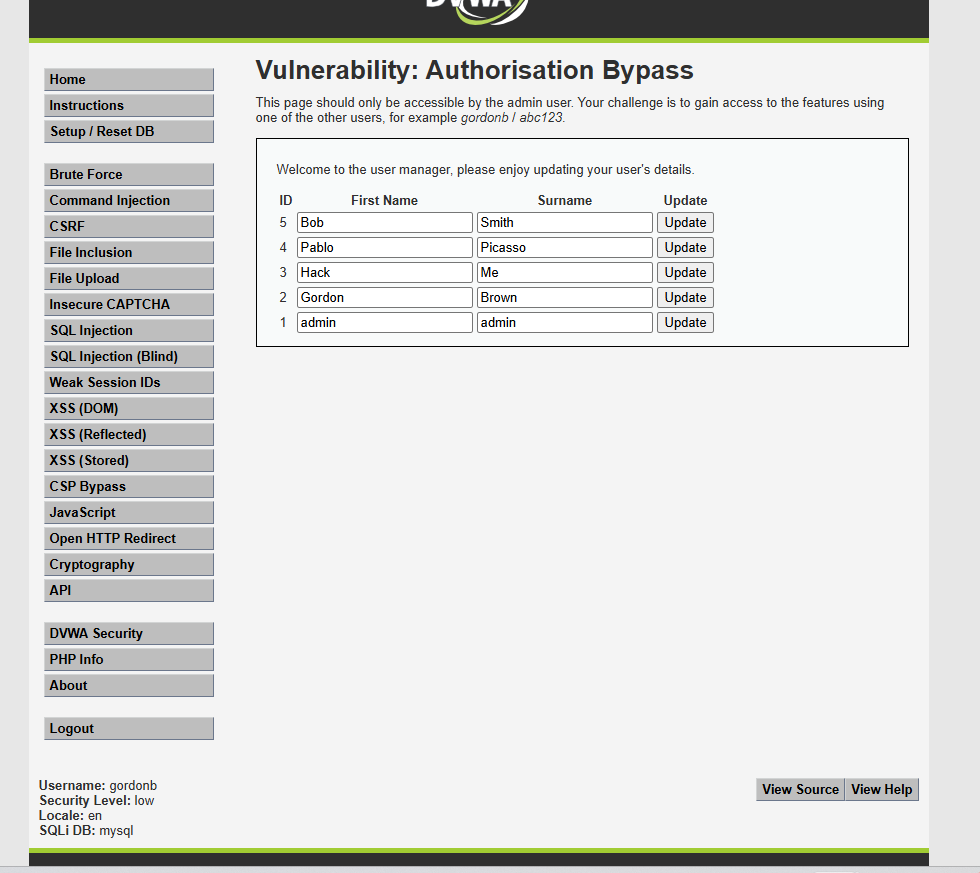
* Content Security Policy (CSP) is a feature that helps to prevent or minimize the risk of certain types of security threats. It consists of a series of instructions from a website to a browser, which instruct the browser to place restrictions on the things that the code comprising the site is allowed to do. Now I’ll include every URL and check for bypass.
* When I entered the first URL it showed output as-

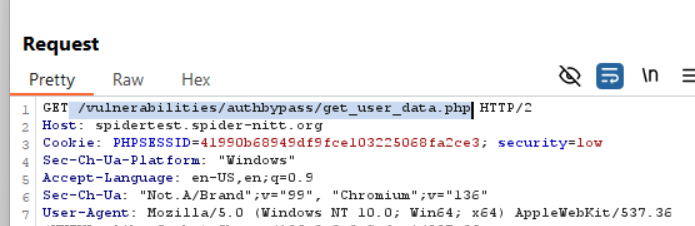


* Since alert.js is a normal java script file with correct headers.
* And now I ll include cookie.js which shows the cookie of the current user.
* Others won’t work such as alert.txt which has the wrong extension type, forced\_download.js as the name suggests it ll download something rather than execution.

**Authorisation Bypass:**

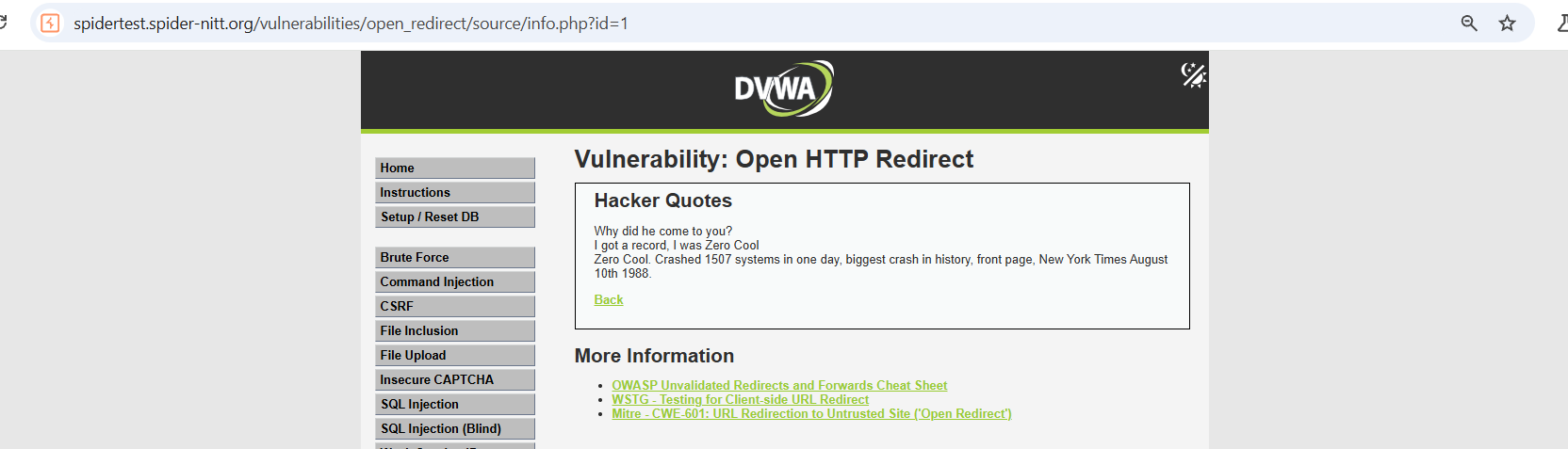
* An authentication bypass occurs when threat actors are able to bypass the authentication protocols an organization may have in place.
* My objective is to visit the authorisation bypass page in another user since this page is accessible only for admin.
* Now I’ll use another user to login which is gordnb and pass is abc123.
* We could not see that page once I logged in from another user.
* But in the URL if I type **/vulnerabilities/authbypass** instead **index.php** then I’ll get the required webpage.



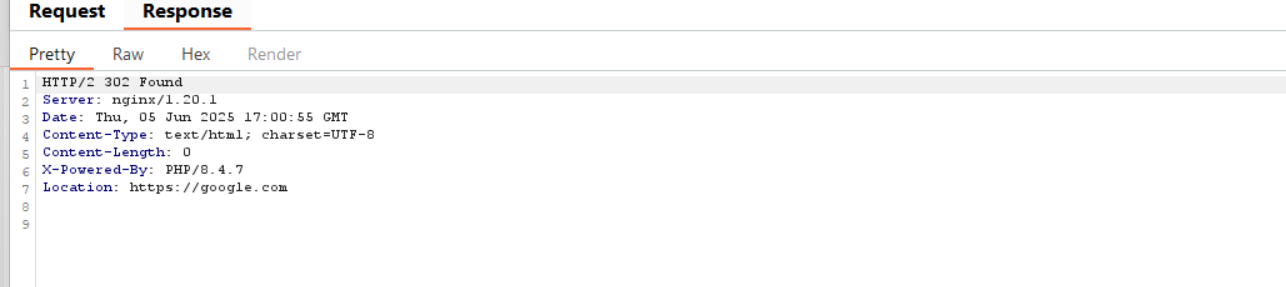


**Open HTTP Redirect:**

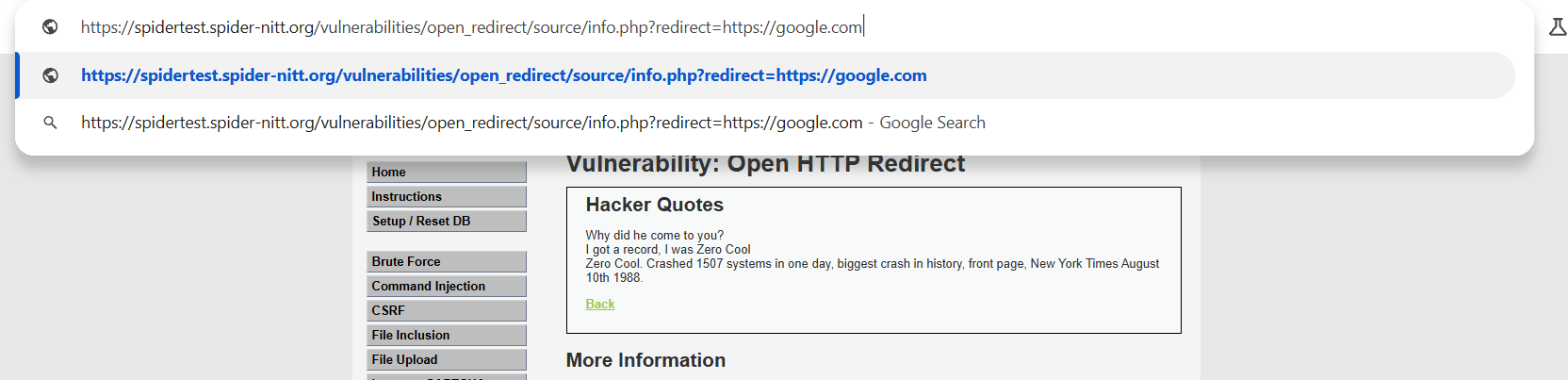
* An *open redirect* is a vulnerability that allows your website, web application, or API to be used as a tool to trick others into visiting malicious sites.
* Now we’ll try some redirects using burp suite. For that we need to find the open HTTP redirect page URL in burp suite and try to redirect to other websites by manipulating.

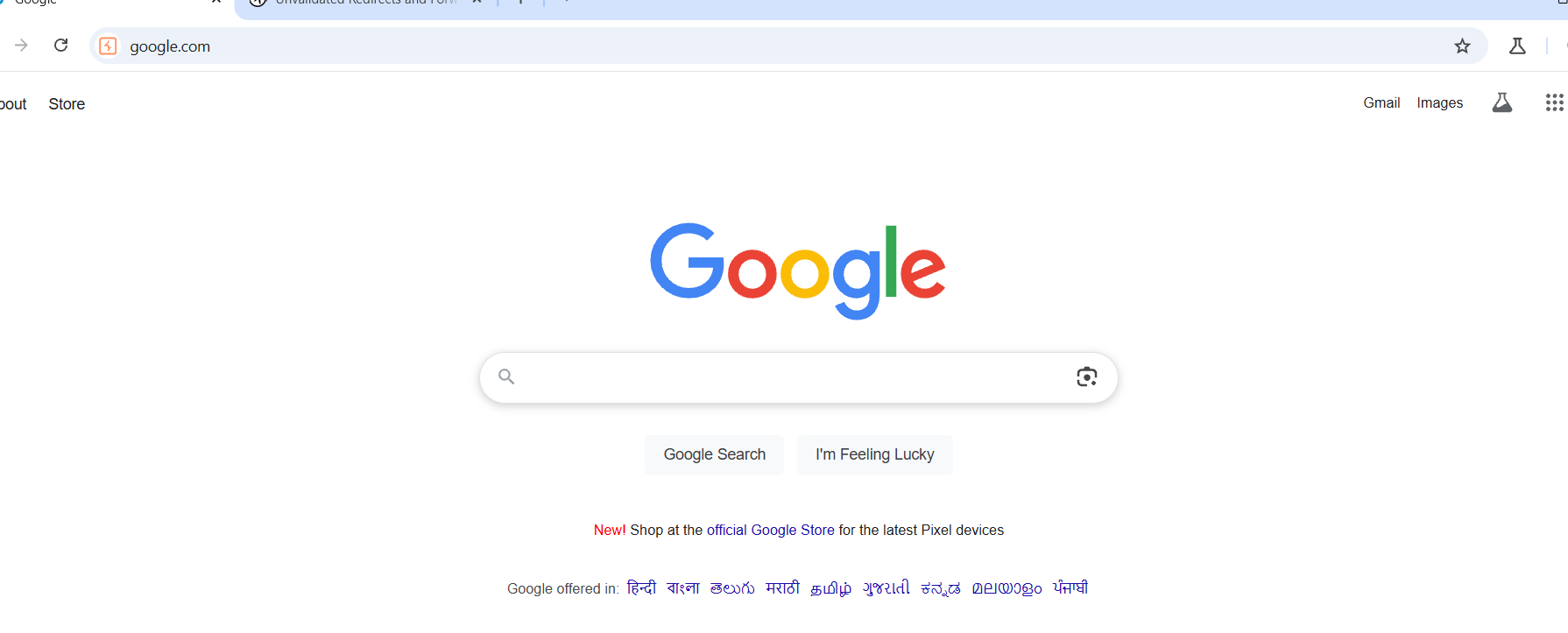


* This is the page and I’ll now try to redirect to google.com. I have opened this site in burp suite and I got to see its URL. Now I changed **id=1** to **redirect=https://google.com.** Now I send this and got a response as **302 FOUND.**



* So, once I got confirmation, I changed it in the websites URL and it redirected me into google.com.

****

****

* So now our open HTTP redirect page got redirected to google.com. Not only google we can also redirect to other pages as per needs.