NAME: kelvin Kimotho

LinkedIn: kelvin kimotho

Attacking Web Applications with Ffuf on HackTheBox

Here is my shareable link https://academy.hackthebox.com/achievement/1476251/54

Introduction

A **web applicatio**n is a software program that runs on a web server and can be accessed via a web browser.

- It uses web technologies such as HTML, CSS, JavaScript, and server-side languages (e.g., Python, PHP, Ruby).
- Web apps can perform various functions, from simple information display to complex interactions like online banking or social networking.

Top 10 Vulnerabilities (OWASP 2023)

- **Broken Access Control.** Inadequate enforcement of user permissions and roles.
- **Cryptographic Failures**. Poor implementation of cryptographic practices, leading to data exposure.
- **Injection**. Vulnerabilities allowing attackers to send malicious data (e.g., SQL, NoSQL, Command Injection).
- **Insecure Design**. Lack of security considerations in the design phase, leading to potential vulnerabilities.
- **Security Misconfiguration.** Improper configuration of security settings across applications, servers, and databases.
- **Vulnerable and Outdated Components**. Use of outdated libraries, frameworks, and other components that contain known vulnerabilities.
- **Identification and Authentication Failures.** Flaws in user authentication mechanisms, allowing unauthorized access.
- **Software and Data Integrity Failures.** Inadequate measures to ensure data integrity, leading to manipulation.

- **Security Logging and Monitoring Failures.** Insufficient logging and monitoring, making it hard to detect attacks.
- **Server-Side Request Forgery (SSRF)**. Vulnerabilities that allow an attacker to induce the server to make unintended requests.

Introduction to Fuzzing

- Our focus is on the ffuf tool for web fuzzing.
- ffuf is known for being common and reliable.

Topics Covered include,

- Fuzzing for Directories. Discovering hidden directories on a web server.
- Fuzzing for Files and Extensions. Identifying existing files and their extensions.
- Identifying Hidden Vhosts. Finding virtual hosts that may not be publicly accessible.
- Fuzzing for PHP Parameters. Testing PHP parameters for vulnerabilities.
- Fuzzing for Parameter Values. Exploring possible values for parameters to uncover hidden functionality.

Functionality of ffuf

- Automates the process of sending requests to a web server.
- Utilizes a list of potential paths/parameters to check for existence.
- A 200 response code indicates that the page exists, allowing for further manual exploration.

Web Fuzzing

Fuzzing is a testing technique that sends various inputs to a system to observe responses.

Examples include

- SQL Injection Fuzzing. Sending random special characters.
- Buffer Overflow Fuzzing. Sending incrementally longer strings.

Purpose of Fuzzing

- Web servers typically do not provide a directory of links and this necessitates the use of fuzzing to discover hidden pages.

Response Codes in Fuzzing

- 404 Not Found. Indicates a non-existent page.
- 200 OK. Indicates an existing page (e.g., /login).

Manual testing is inefficient and thus tools like ffuf automate the process.

- Tools can send hundreds of requests per second and analyze HTTP response codes to determine page existence.

Wordlists contain commonly used terms for directories and pages, similar to password dictionaries.

- They can reveal up to 90% of existing pages, though some may have unique names.

We can find popular and pre-defined wordlists are available on GitHub's SecLists.

These include various categories for different types of fuzzing, including common passwords.

- The specific wordlist for directory fuzzing is **directory-list-2.3-small.txt.**

Filtering Wordlist Entries

- The wordlist may contain copyright comments that clutter results so we use **-ic flag** in ffuf to ignore these lines.

Directory Fuzzing with ffuf

Our objective as penetration testers is to use ffuf to find hidden website directories.

Ffuf is pre-installed on some linux distributions but we can install it via **apt install ffuf -y.**

To check for help options we use the –h flag. **“ffuf -h”**

Key Options for ffuf

- **-u**: Is the target URL

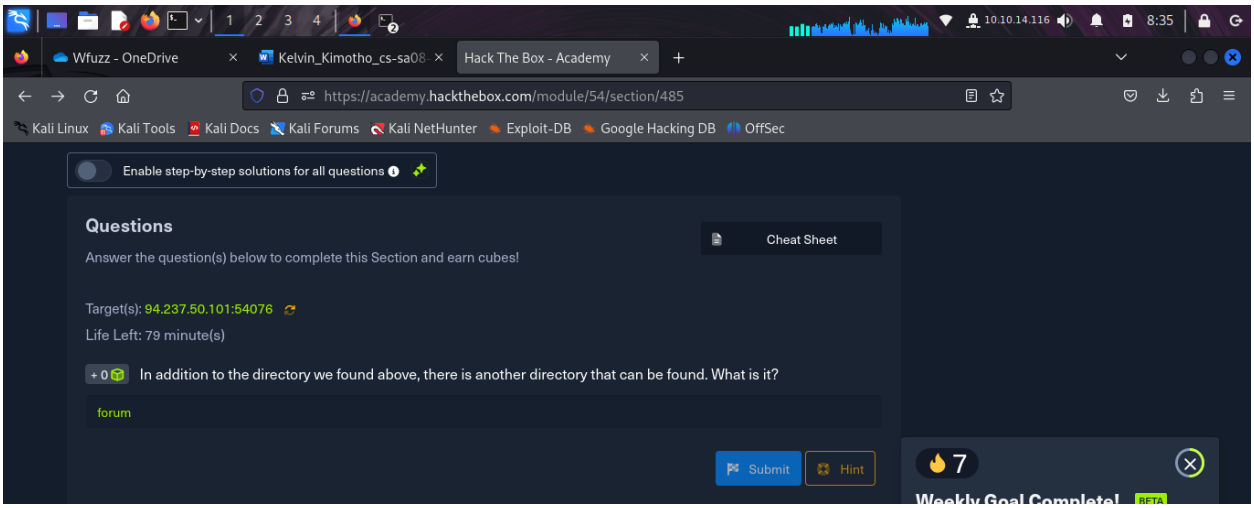
- **-w**: Wordlist file path and keyword (e.g., -w /path/to/wordlist:FUZZ)
- **-mc**: Match specific HTTP status codes
- **-fs**: Filter by response size
- **-o**: Write output to a file

The ffuf command format is " **ffuf -w <path_to_the_wordlist_we_intend_to_use>:FUZZ -u http://SERVER_IP:PORT/FUZZ"**

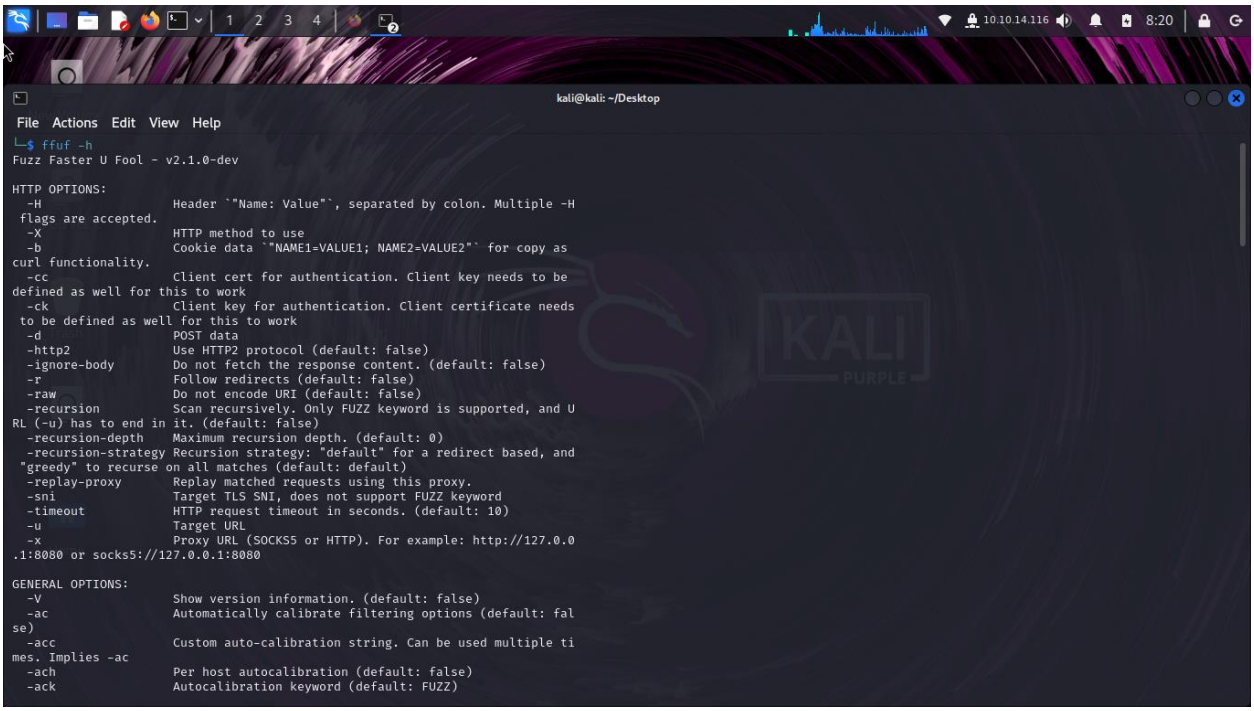
- ffuf can handle a high number of requests quickly (e.g., 90,000 requests in under 10 seconds).
- By increasing threads (e.g., -t 200) we can speed up the process but risks overwhelming the server or disrupting our connection.

Question: In addition to the directory we found above, there is another directory that can be found. What is it?

Answer: forum

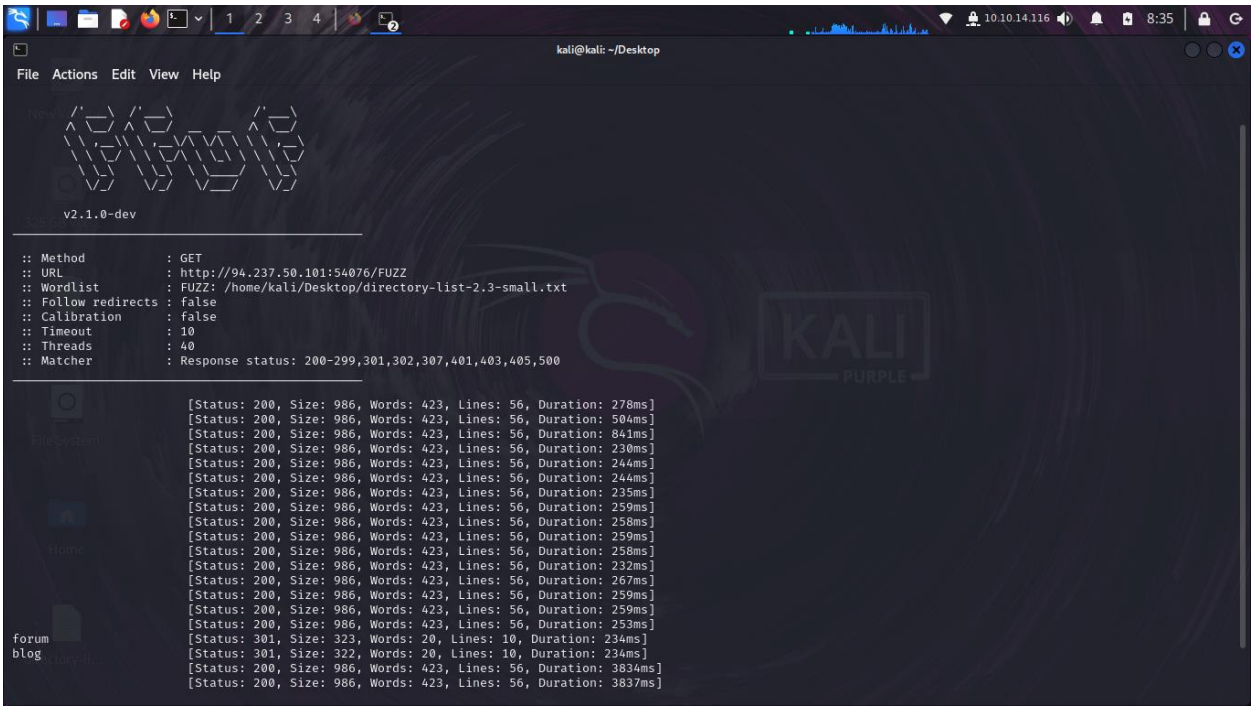


I started with interacting with the ffuf tool on my machine using **–h** flag to check the help options.



I downloaded a wordlist from the internet and proceeded to enumerating the target by running ”

ffuf -w directory-list-2.3-small.txt:FUZZ -u http://94.237.50.101:54076/FUZZ”



Page Fuzzing

Still using ffuf tool for web fuzzing using wordlists and keywords.

The aim here is to locate hidden pages on a web application. Discover hidden pages within the directory.

Extension Fuzzing

- Identify common file extensions (.html, .php, etc.) based on server type (Apache, IIS).
- We use Ffuf to fuzz extensions by replacing the FUZZ keyword with potential extensions.

Fuzzing Command Example " f**uf -w /path/to/web-extensions.txt:FUZZ -u http://SERVER_IP:PORT/blog/indexFUZZ"**

We use **index.*** as a common file to test various extensions.

Interpreting Results

- We can analyze HTTP response codes (200, 403) to determine accessible pages.
- Identify the server's language based on successful extensions (e.g., .php).

Finding PHP Files

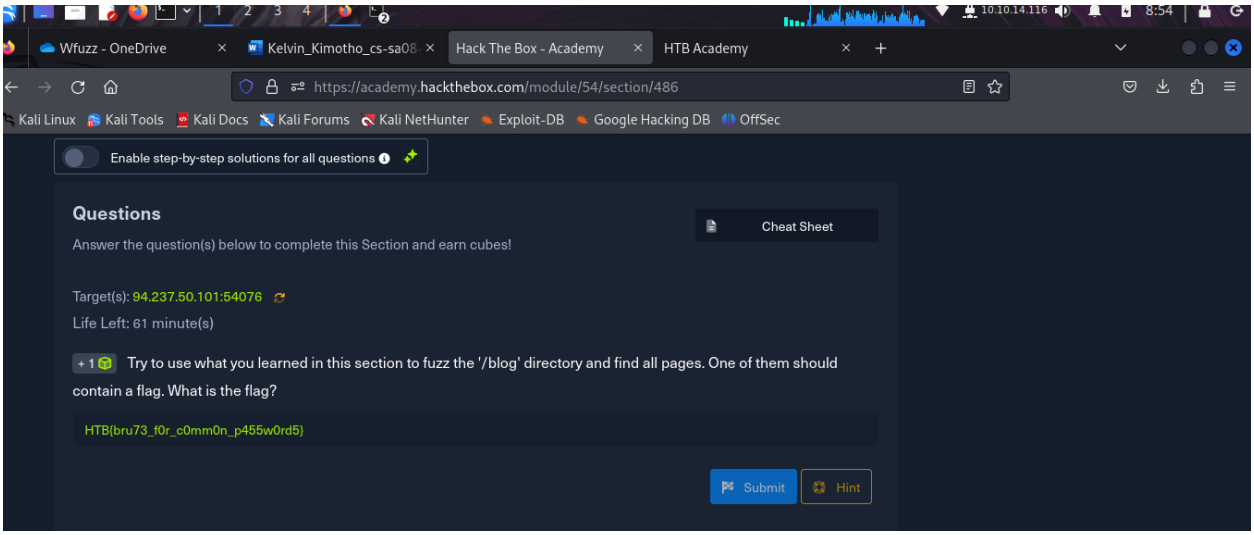
- We use a wordlist to fuzz for potential PHP filenames

" ffuf -w /path/to/directory-list.txt:FUZZ -u http://SERVER_IP:PORT/blog/FUZZ.php"

- Successful hits return HTTP 200 status, indicating accessible content.
- Empty pages can be identified, and content-rich pages should be explored further.

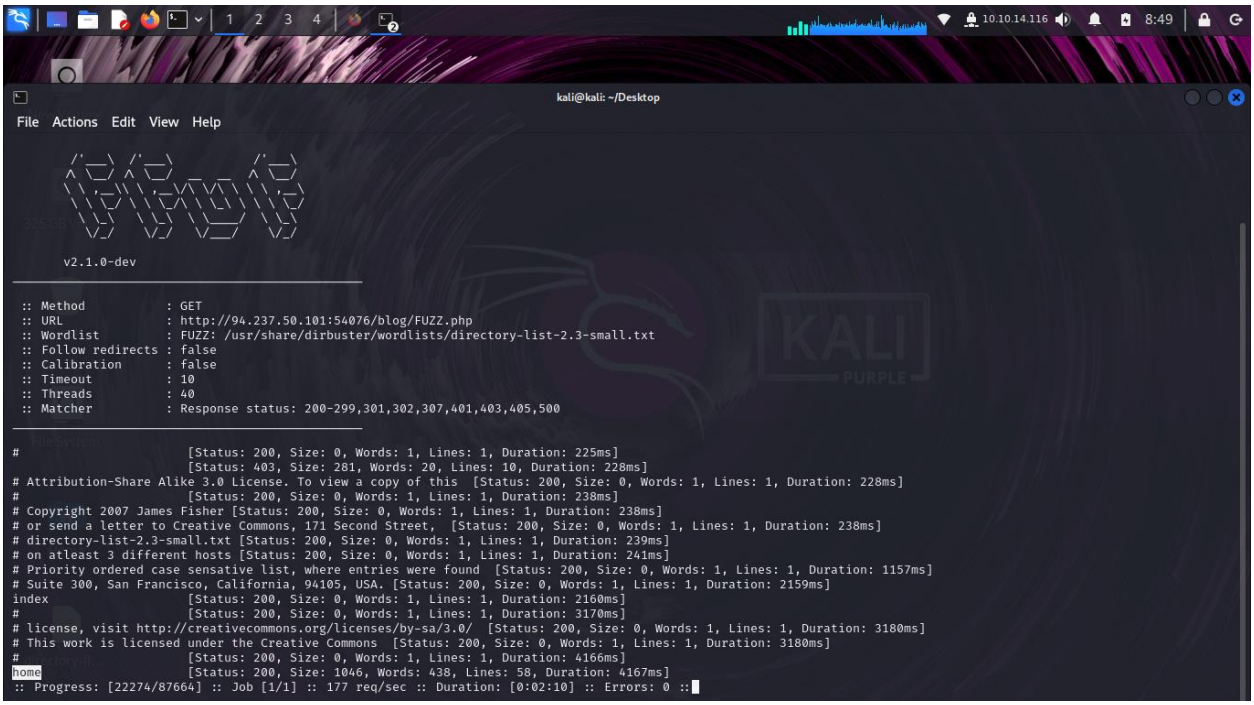
Question: Try to use what you learned in this section to fuzz the '/blog' directory and find all pages. One of them should contain a flag. What is the flag?

Answer: HTB{bru73_f0r_c0mm0n_p455w0rd5}

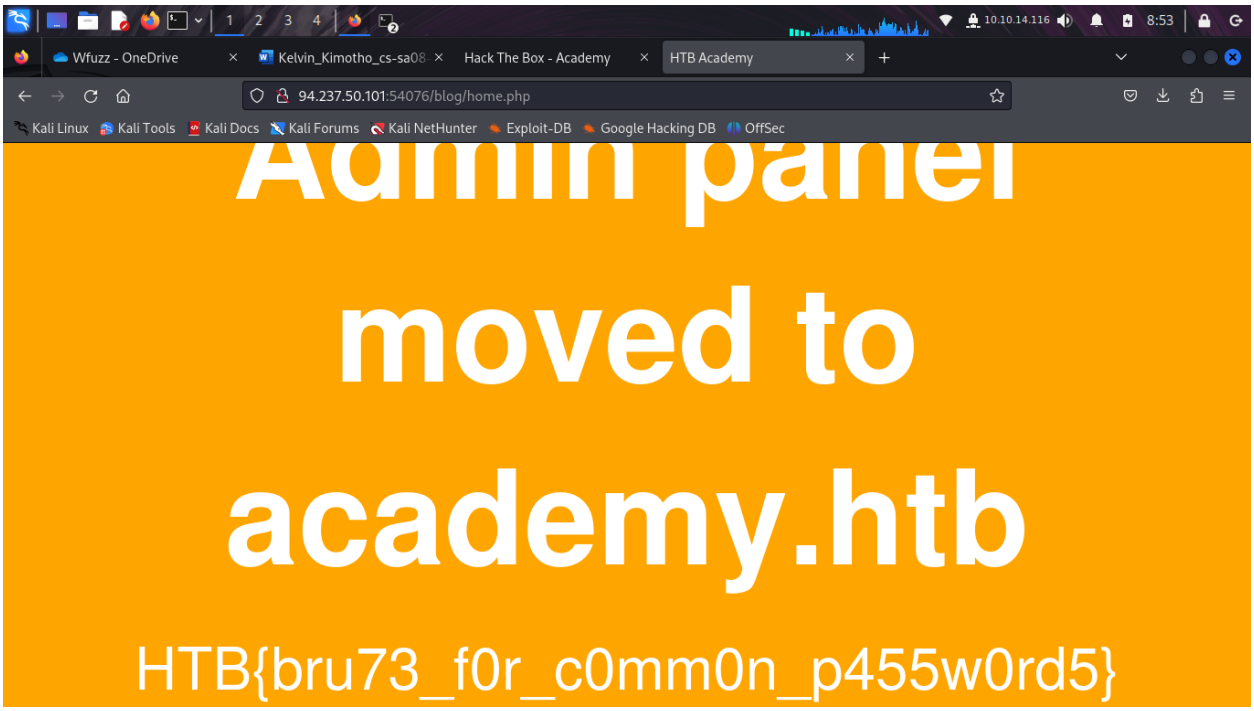


The goal Fuzz the **/blo**g directory to find all pages, with one containing a flag.

Here is the command i used " **ffuf -w /usr/share/dirbuster/wordlists/directory-list-2.3-small.txt:FUZZ -u http://94.237.50.101:54076/blog/FUZZ.php**"



I discovered several pages including **Index** which had **a size of zero meaning it had no content** , I went ahead and visited **home page**, atleast its size wasn't zero indicating it had some content.



And that's how i found the hidden flag.

Recursive Fuzzing with Ffuf

- This type of fuzzing automates scanning of directories and subdirectories.
- It is useful for websites with extensive directory trees.

It does the fuzzing process for multiple directories and their contents.

Recursive Flags in Ffuf

- -recursion: Enables recursive scanning.
- -recursion-depth: Limits the depth of recursion (e.g., -recursion-depth 1 only scans main and direct subdirectories).
- -e .php: Specifies the file extension for fuzzing.

We use **-v** to display full URLs for clarity.

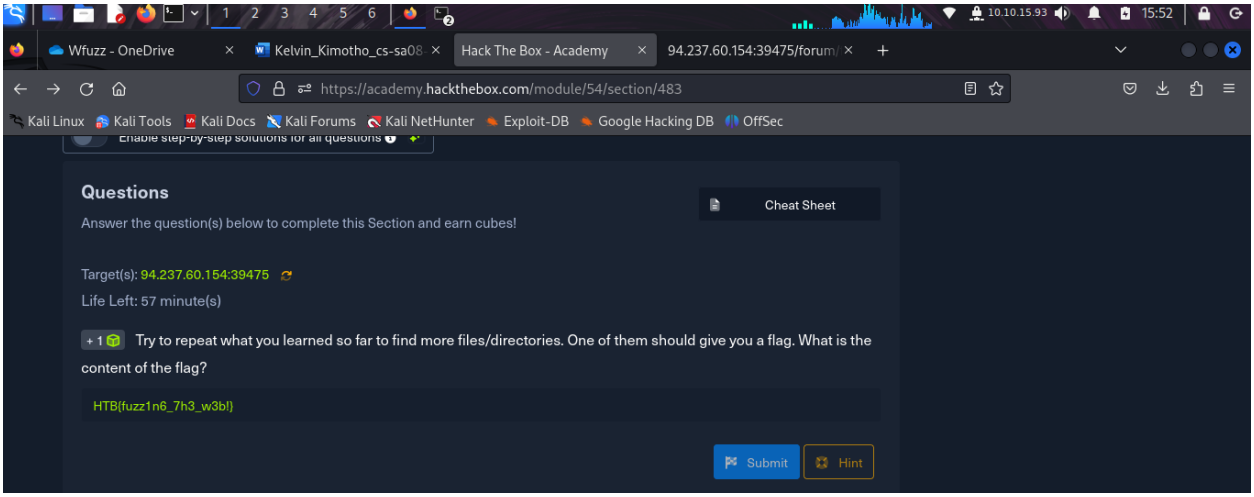
Example Command Format is

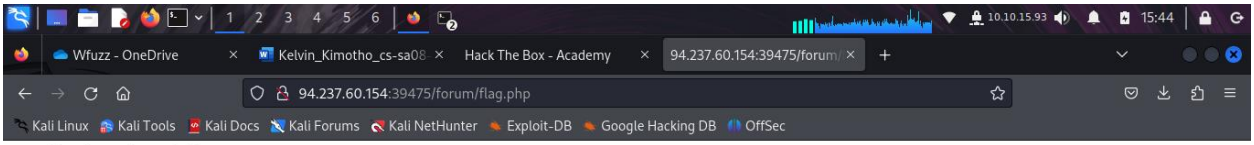
- “ffuf -w wordlist.txt:FUZZ -u http://SERVER_IP:PORT/FUZZ -recursion -recursion-depth 1 -e .php -v”

The scan results include various status codes and URLs identified during the fuzzing process.

Question: Try to repeat what you learned so far to find more files/directories. One of them should give you a flag. What is the content of the flag?

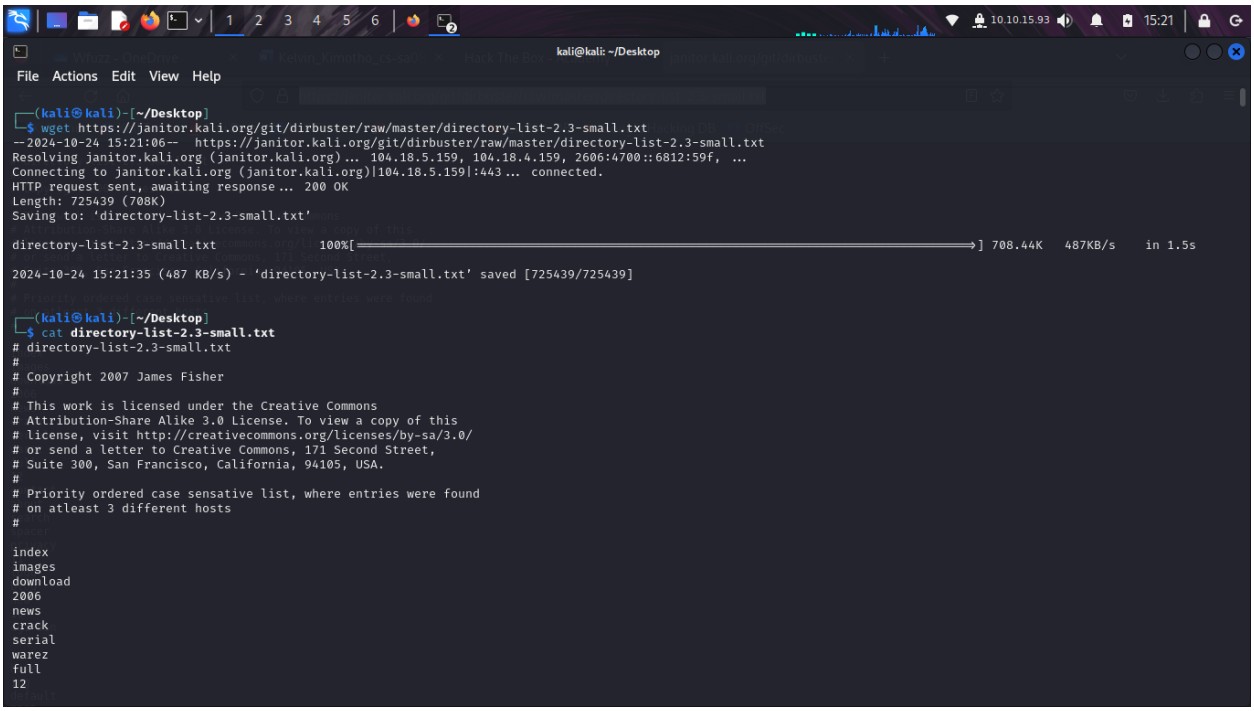
Answer: HTB{fuzz1n6_7h3_w3b!}





HTB{fuzz1n6_7h3_w3b!}

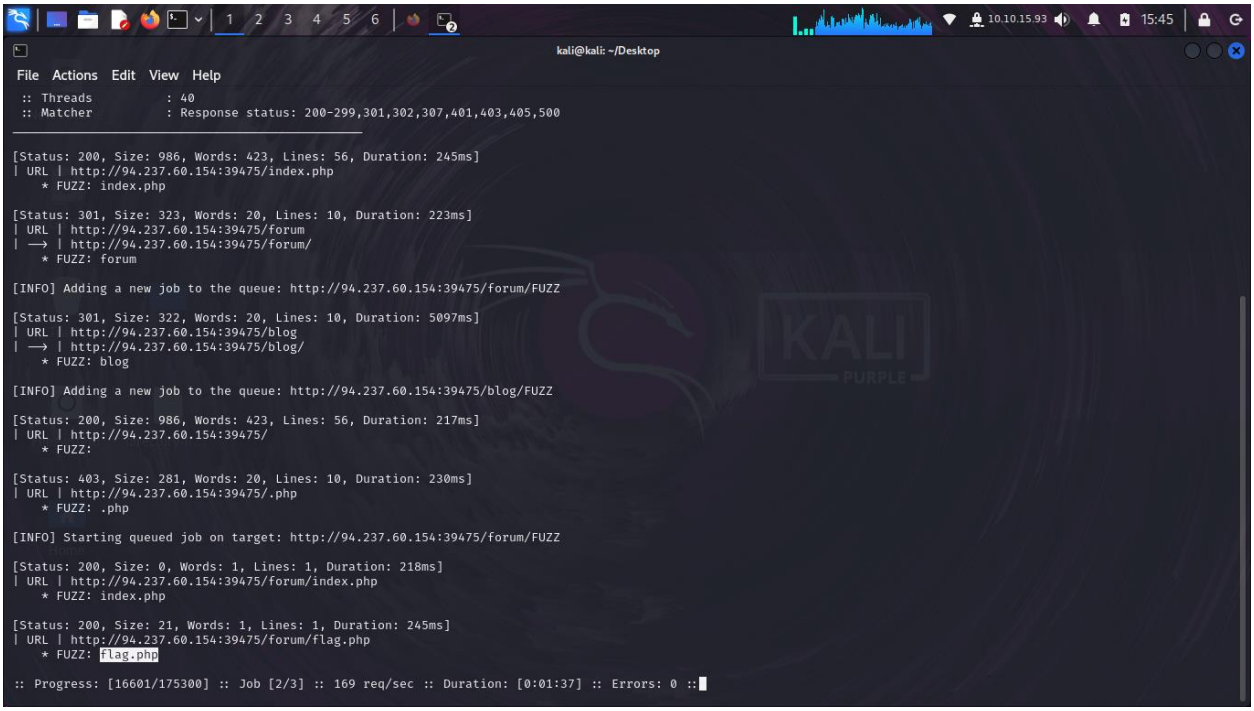
I first downloaded the "**directory-list-2.3-small.txt**" wordlist using **wget** tool.



Then ran the following command.

- “**ffuf -w directory-list-2.3-small.txt:FUZZ -u http://94.237.60.154:39475/FUZZ -recursion -recursion-depth 1 -e .php -v**”

I found a **flag.php** web page under the **forums** directory



Accessing Local DNS Records

By attempting to access the admin panel at academy.htb we encounter a connection error.

Reason for Connection Failure

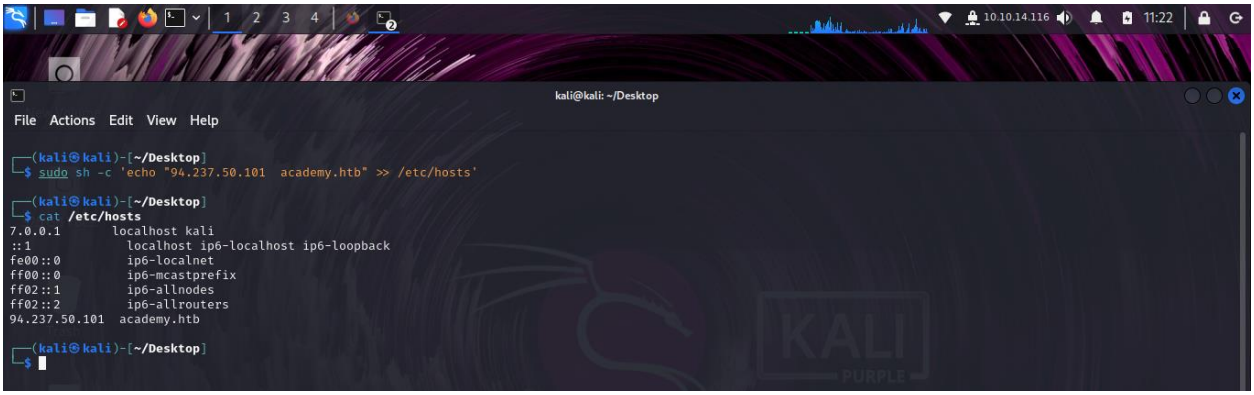
- academy.htb is a local domain, not publicly accessible.
- Browsers resolve URLs to IPs using the local /etc/hosts file and public DNS.
- Lack of entry for academy.htb leads to connection failure.

We need to add academy.htb or any other local domain to the local **/etc/hosts f**ile on our machinf using:

sudo sh -c 'echo "SERVER_IP academy.htb" >> /etc/hosts'

I added academy.htb on my machine.

- sudo sh -c 'echo "**94.237.50.101** academy.htb" >> /etc/hosts'



This allows our browser to recognize academy.htb or any other local domain we want to access.

Sub-domain Fuzzing with Ffuf

A **sub-domain** is a domain that exists under another domain (e.g., photos.google.com is a sub-domain of google.com).

The aim here is to identify potential sub-domains by checking for public DNS records that point to working server IPs.

Requirements for Fuzzing subdomains

Wordlist which is a list of common sub-domain names, found maybe in /opt/useful/seclists/Discovery/DNS/.

An example Command for Fuzzing

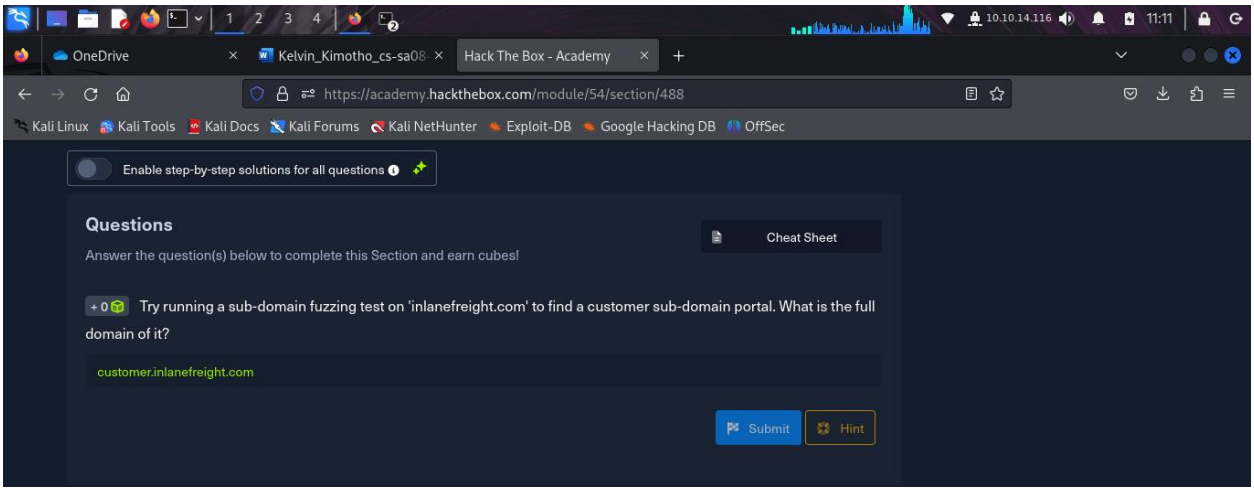
- “ffuf -w /path_to/subdomains-top1million-5000.txt:FUZZ -u https://FUZZ.domain/”

Results from such a scanning may include

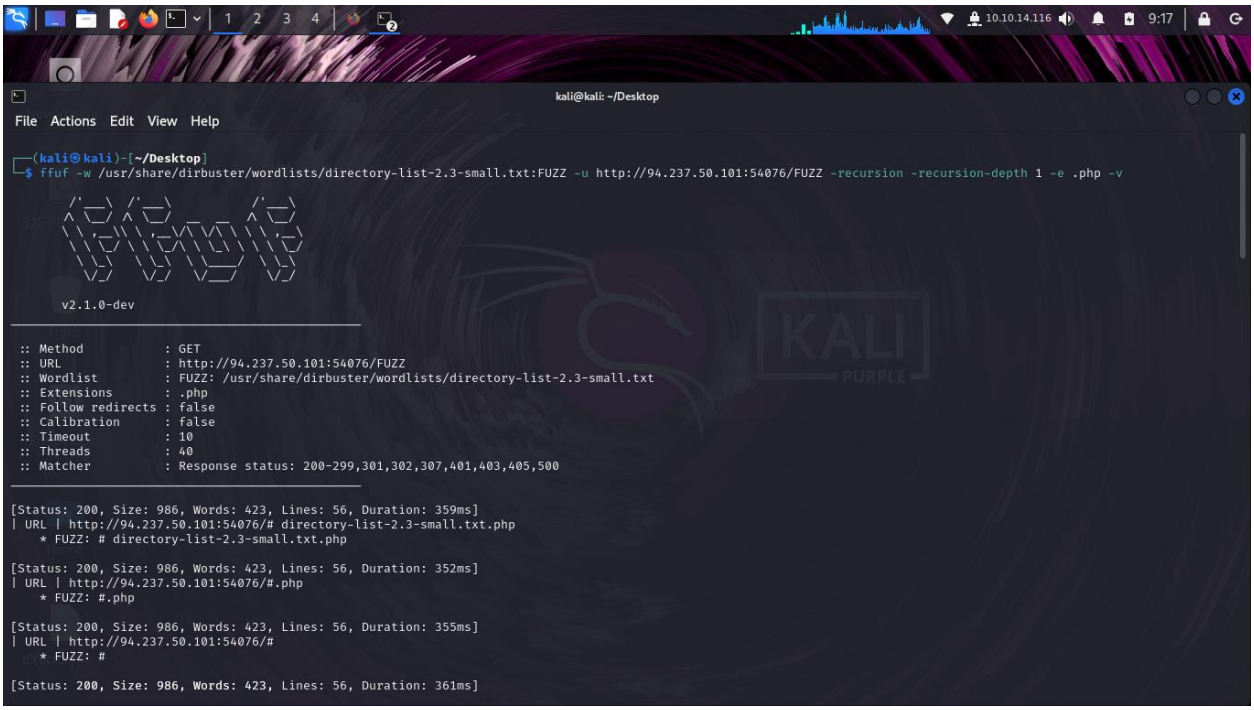
- Some sub-domains identified (e.g., support, blog, www).
- Various HTTP status responses indicating the existence of these sub-domains.

Question: Try running a sub-domain fuzzing test on 'inlanefreight.com' to find a customer sub-domain portal. What is the full domain of it?

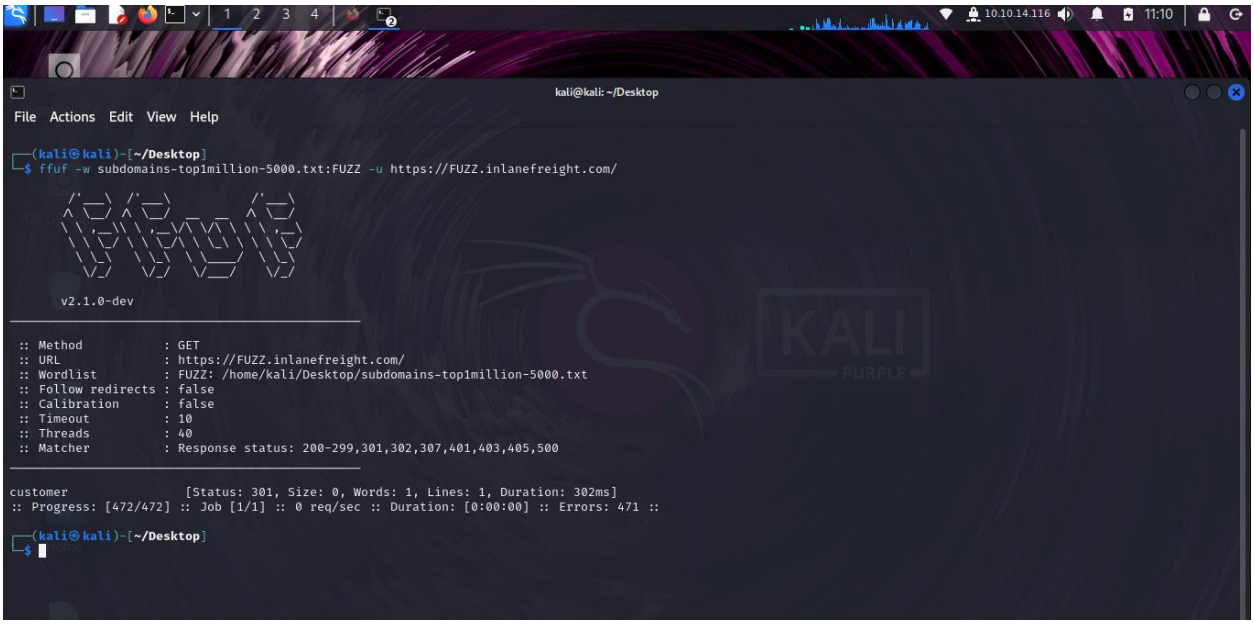
Answer: customer.inlanefreight.com



I didn't have the wordlist in my machine so i downloaded it first.



I went ahead and run the following command on the provided attack box “ **ffuf -w subdomains-top1million-5000.txt:FUZZ -u http://FUZZ.** i**nlanefreight.com/**”



I found **customer** a subdomain to **inlanefreight.Com** domain.

Vhost Fuzzing

Vhost fuzzing allows discovery of sub-domains without public DNS records.

- It is important for identifying both public and non-public sub-domains.

VHosts vs. Sub-domains

- **VHosts**: Sub-domains served on the same server with the same IP.
- May not have public DNS records.
- Non-public sub-domains can lead to connection failures if queried directly.

Traditional sub-domain fuzzing only identifies public records but VHost fuzzing leverages an existing IP to find hidden sub-domains.

VHost fuzzing uses HTTP headers, specifically the Host: header.

The command format is "**ffuf -w /path/to/wordlist:FUZZ -u http://Target_Domain:PORT/ -H 'Host: FUZZ.Target_Domain'**
A successful discovery of VHosts is indicated by varied response sizes when correct headers are sent.

Filtering Results

By default, ffuf filters results based on HTTP status codes, excluding 404 NOT FOUND responses.
Many responses can return a 200 status code, necessitating further filtering.

Filtering Options in Ffuf. Ffuf allows filtering based on:

- HTTP status codes (-fc)
- Response size (-fs)
- Number of lines in the response (-fl)
- Number of words in the response (-fw)
- Regular expressions (-fr)

We use ffuf -h to view options.

Since response sizes from valid VHosts are unknown, we filter out the known size of incorrect responses (900).

An example command with filtering

"**ffuf -w /path/to/wordlist:FUZZ -u http://academy.htb:PORT/ -H 'Host: FUZZ.academy.htb' -fs 900**"

The command focuses on responses with sizes other than 900, indicating potential valid VHosts. After executing, we may receive valid entries like

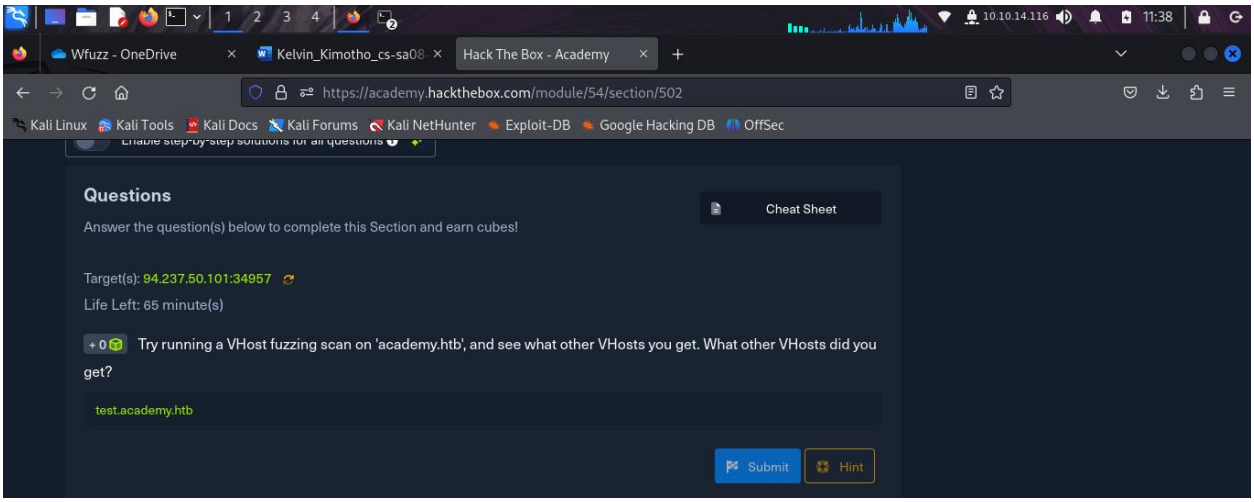
- admin with a response size of 0, indicating a different VHost.

We then add the discovered VHost (e.g., admin.academy.htb) to **/etc/hosts** to access it.

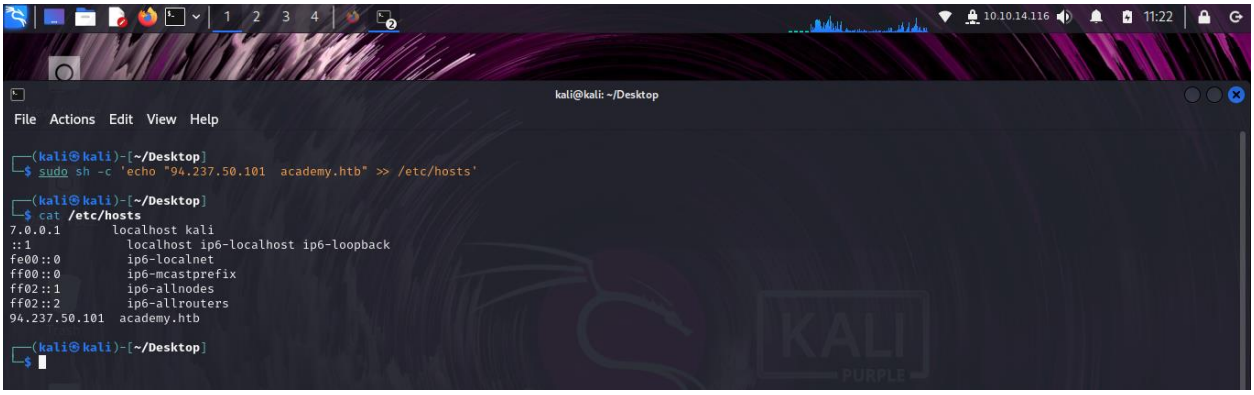
- We can confirm the VHost by visiting **https://subdomain:PORT/blog/page.php.**

Question: Try running a VHost fuzzing scan on 'academy.htb', and see what other VHosts you get. What other VHosts did you get?

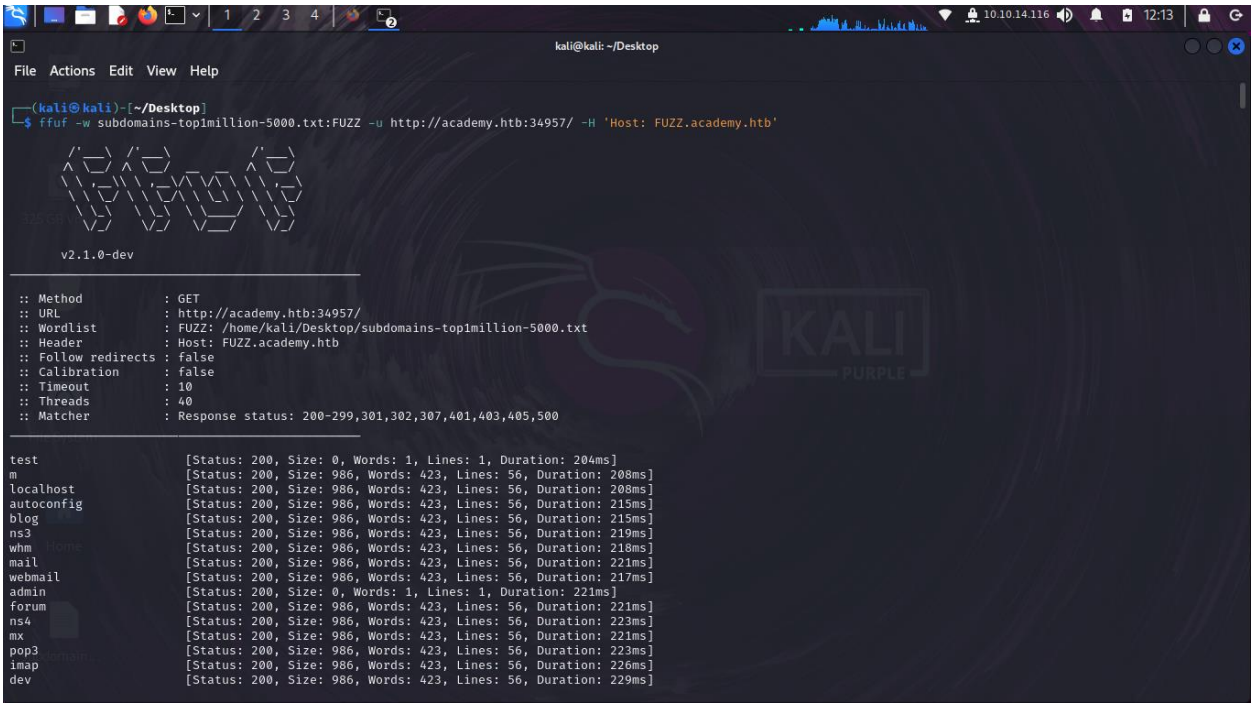
Answer: test.academy.htb



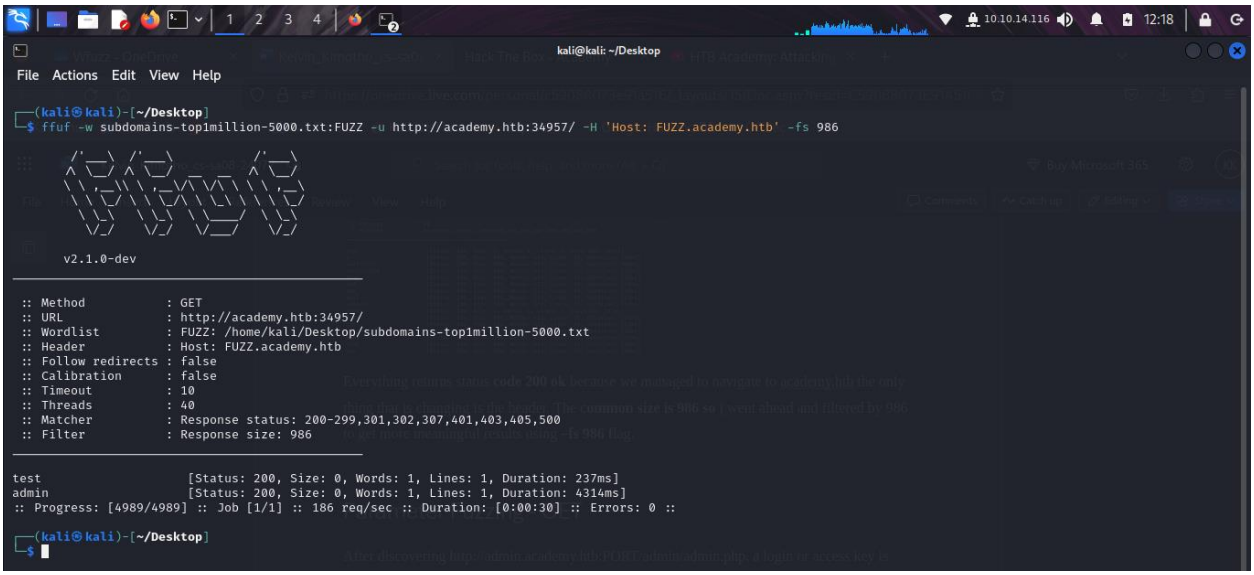
I first added the academy.htb host as an entry in our **/etc/hosts** file we *are fuzzing a private domain using the same IP address for every vhost under the domain.*



I ran the following command "**ffuf -w subdomains-top1million-5000.txt:FUZZ -u http://94.237.50.101:34957/ -H 'Host: FUZZ. 94.237.50.101' "**



Everything returns status **code 200 ok** because we managed to navigate to academy.htb the only thing that is changing is the header. The c**ommon size is 986 so** i went ahead and filtered by 986 to get more meaningful results using **–fs 986 f**lag.



I found **test** and admin but test was the answer.

Parameter Fuzzing - GET

Parameters can be passed via GET or POST requests to interact with a page.

Fuzzing for Parameters

- Fuzzing may reveal unpublished or insecure parameters that could be vulnerable.
- Focus on GET requests, where parameters follow a ? in the URL.

Fuzzing Method

We use ffuf to enumerate potential parameters, eg.

- “http://admin.academy.htb:PORT/admin/admin.php?FUZZ=key"

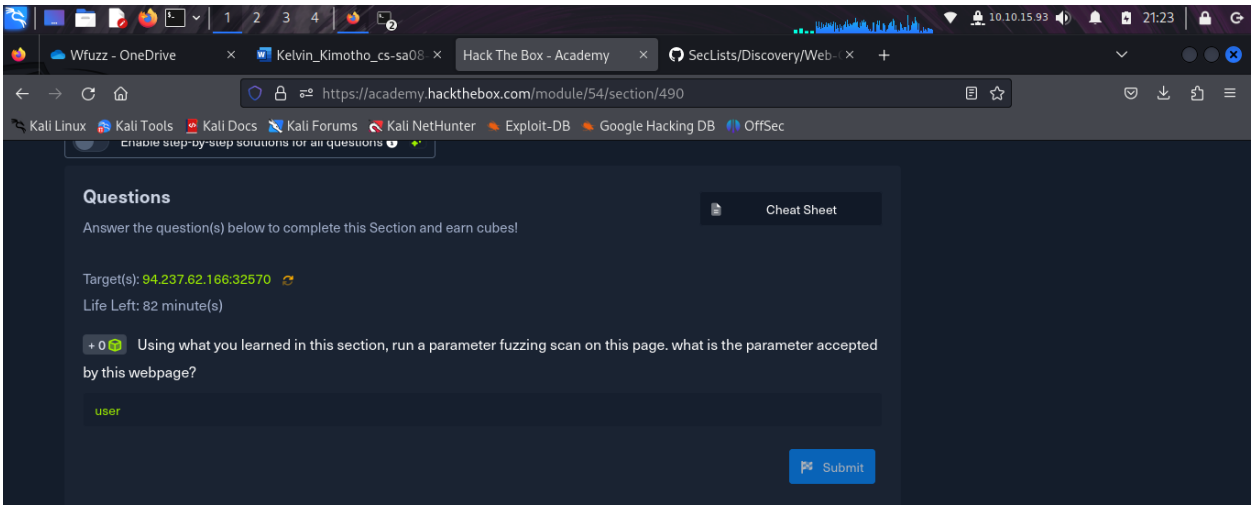
We can use the Burp Suite parameter names wordlist **“burp-parameter-names.txt”** for exxample.
This is how we run the ffuf command “ **ffuf -w burp-parameter-names.txt:FUZZ -u http://admin.academy.htb:PORT/admin/admin.php?FUZZ=key -fs xxx**”

We analyze the results while filtering out known response sizes. Example output may include

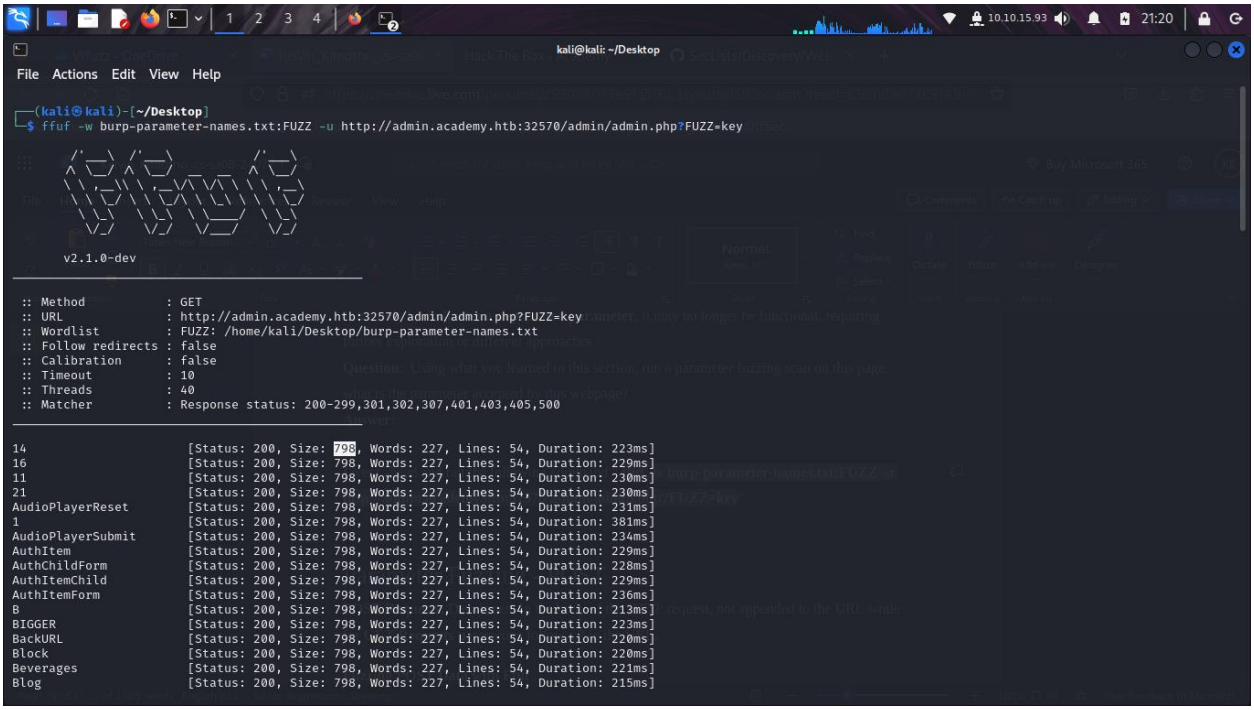
- Status code and response details that indicate if a valid parameter was identified.

We can verify the results Visiting the page with any discovered parameters to see if access to the flag is granted.

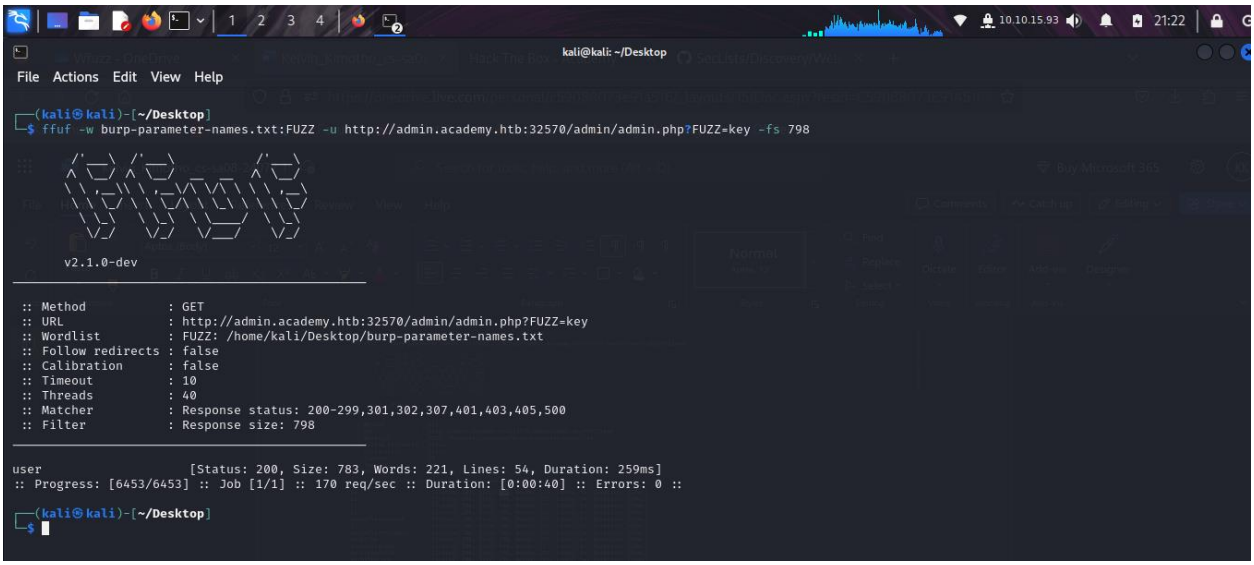
If the only hit **indicates a deprecated parameter**, it may no longer be functional, requiring further exploration or different approaches.
Question: Using what you learned in this section, run a parameter fuzzing scan on this page. what is the parameter accepted by this webpage?
Answer: user



I went ahead and ran the following command " **ffuf -w burp-parameter-names.txt:FUZZ -u http://admin.academy.htb:32570/admin/admin.php?FUZZ=key**"



798 was the common response size so i used it to filter"-**fs 798".**



I found a parameter **user.**

Parameter Fuzzing - POST

POST Requests. Data is sent in the body of the HTTP request, not appended to the URL while for **GET Requests p**arameters follow a ? in the URL.

Fuzzing POST Data with Ffuf

We use the -d flag to specify the data field. We then add -X POST to send POST requests.
For PHP applications, we set the content type to application/x-www-form-urlencoded using

- "-H 'Content-Type: application/x-www-form-urlencoded'"

We can execute the following command for example to fuzz POST parameters

- **"ffuf -w wordlist.txt:FUZZ -u http://admin.academy.htb:PORT/admin/admin.php -X POST -d 'FUZZ=key' -H 'Content-Type: application/x-www-form-urlencoded' -fs xxx"**

We then use curl to test identified parameters

- **"curl http://admin.academy.htb:PORT/admin/admin.php -X POST -d 'id=key' -H 'Content-Type: application/x-www-form-urlencoded' "**

Analyze the response, e.g., receiving "Invalid id!" indicates the parameter exists but needs valid input.

Value Fuzzing

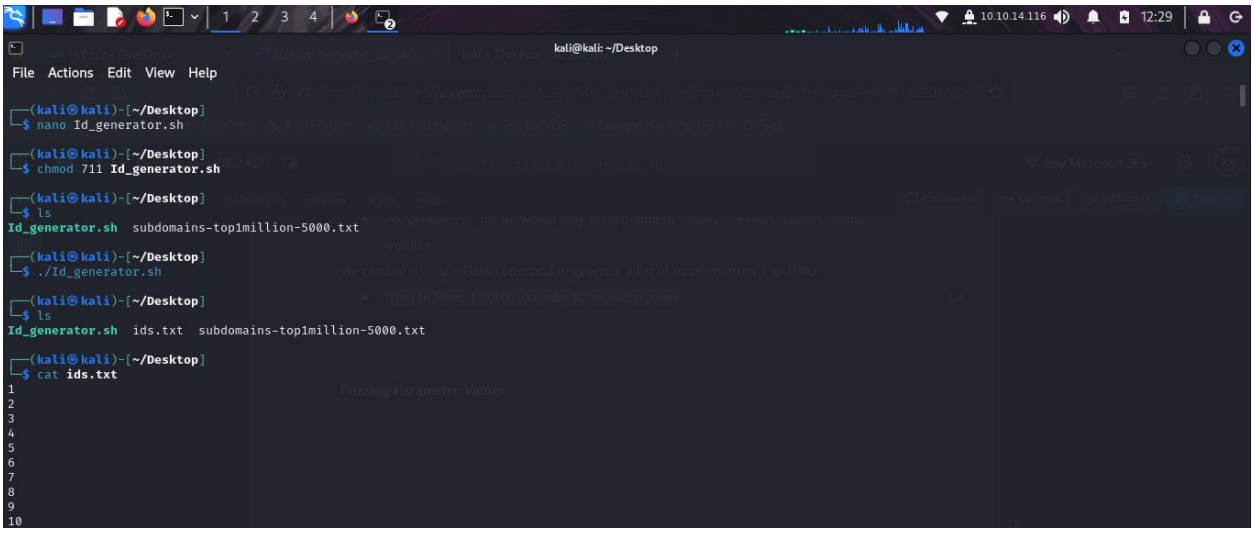
After identifying a working parameter, the next step is to fuzz its values.

Creating a Custom Wordlist

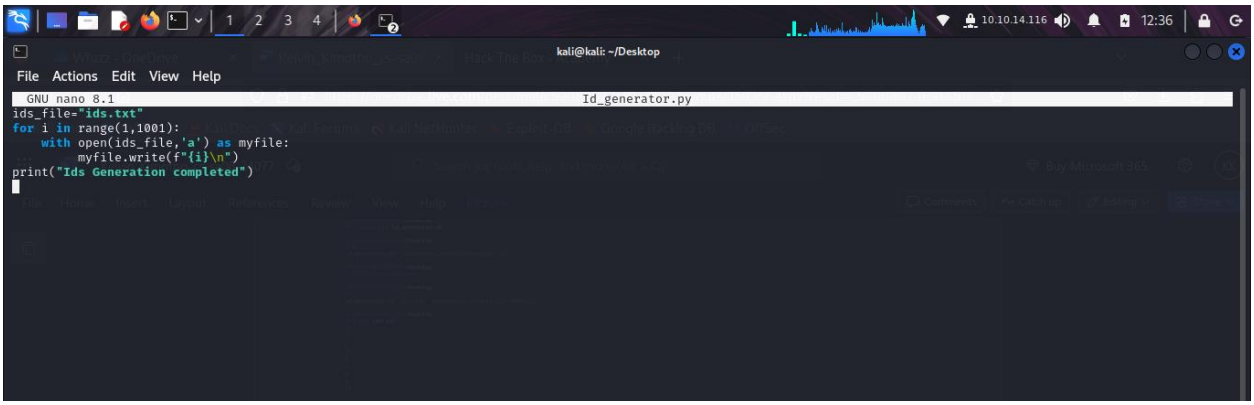
- Pre-made wordlists for specific values may not always be available, especially for custom parameters.
- For parameters like id, which may accept numeric values, we can create a custom wordlist.

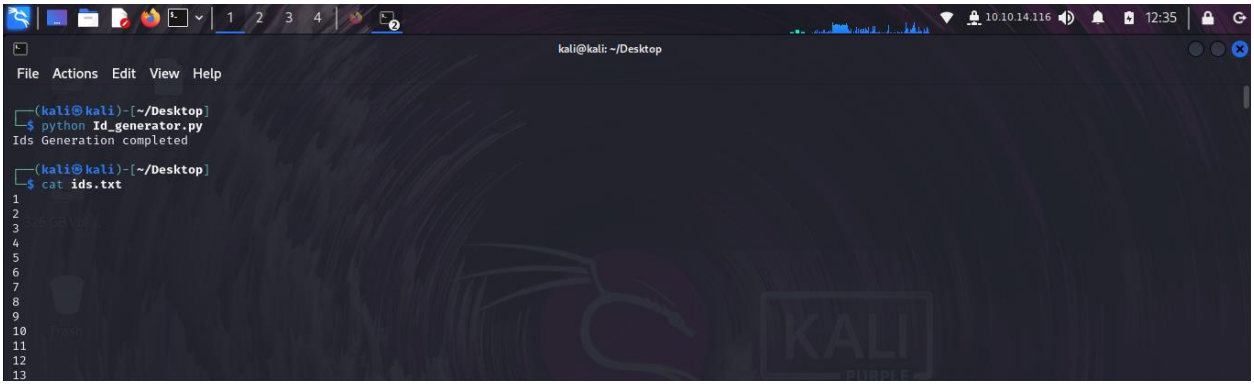
We can use a simple Bash command to generate a list of numbers from 1 to 1000:

- “for i in $(seq 1 1000); do echo $i >> ids.txt; done”



I also tried generating ids with a python script.





The ffuf command resembles previous commands but will place FUZZ in the value position, An example given is,

“**ffuf -w ids.txt:FUZZ -u http://admin.academy.htb:PORT/admin/admin.php -X POST -d 'id=FUZZ' -H 'Content-Type: application/x-www-form-urlencoded' -fs xxx**”

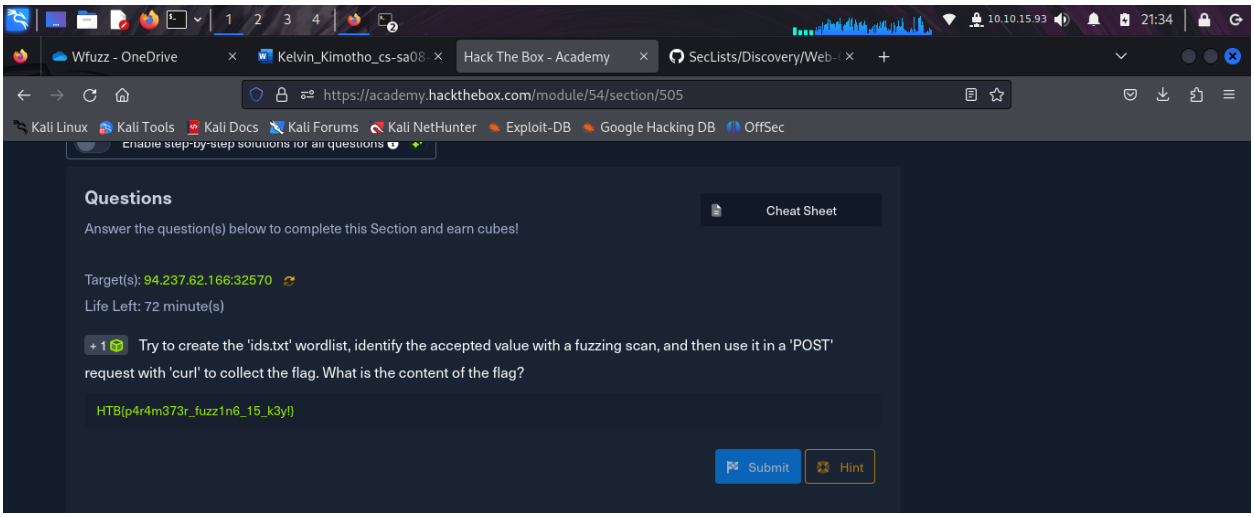
The fuzzing process should yield several responses, with at least one indicating a successful match for the parameter value.

Once a valid value is identified, we use curl to send a POST request with that value

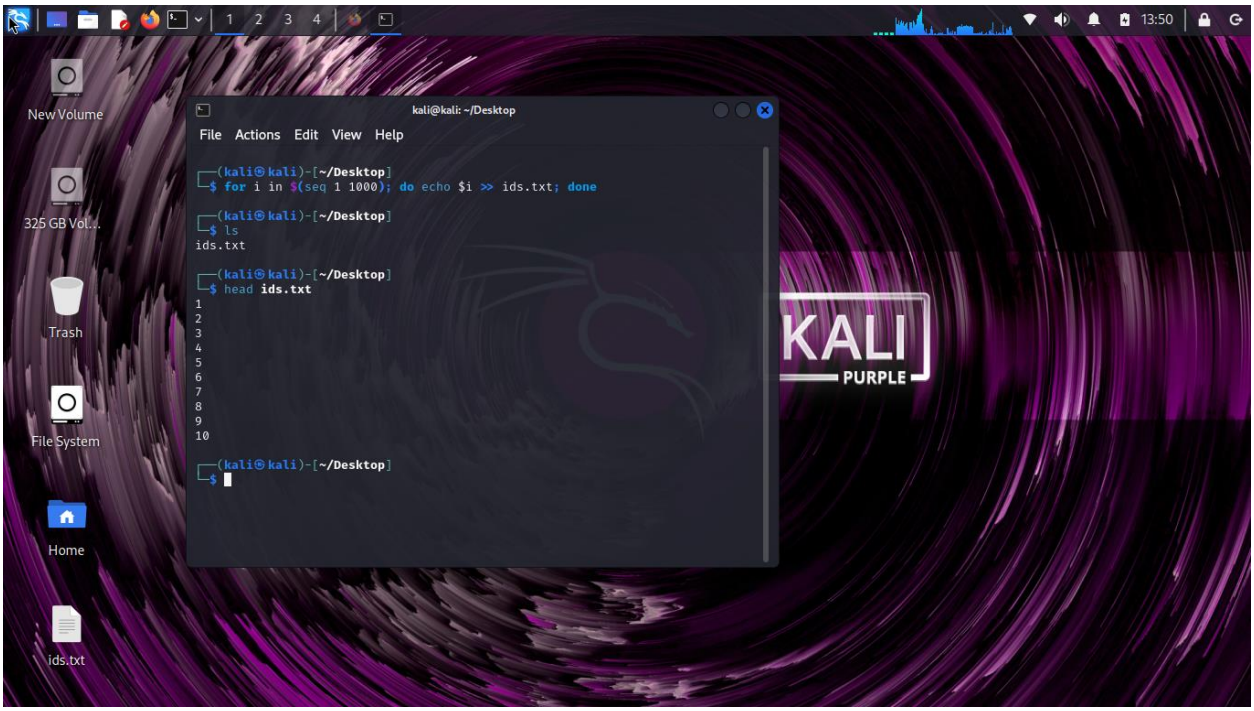
- **“ curl http://admin.academy.htb:PORT/admin/admin.php -X POST -d 'id=<valid_value>' -H 'Content-Type: application/x-www-form-urlencoded' “**

Question: Try to create the 'ids.txt' wordlist, identify the accepted value with a fuzzing scan, and then use it in a 'POST' request with 'curl' to collect the flag. What is the content of the flag?

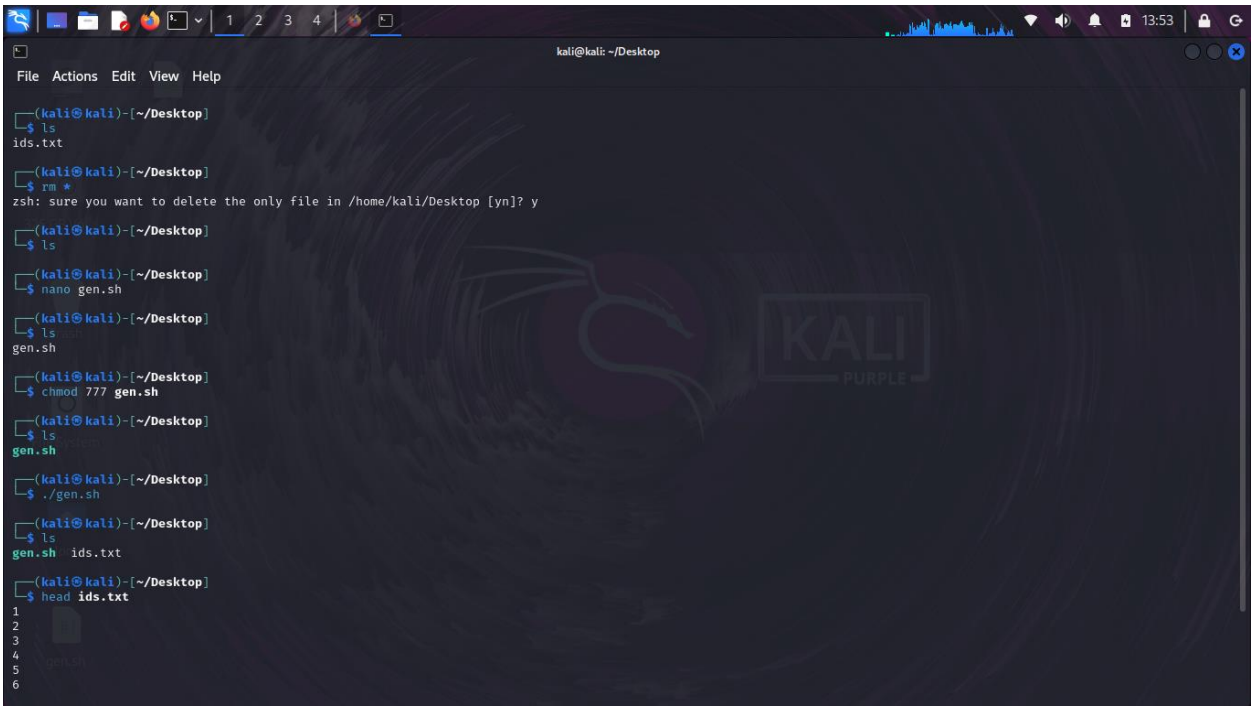
Answer: HTB{p4r4m373r_fuzz1n6_15_k3y!}



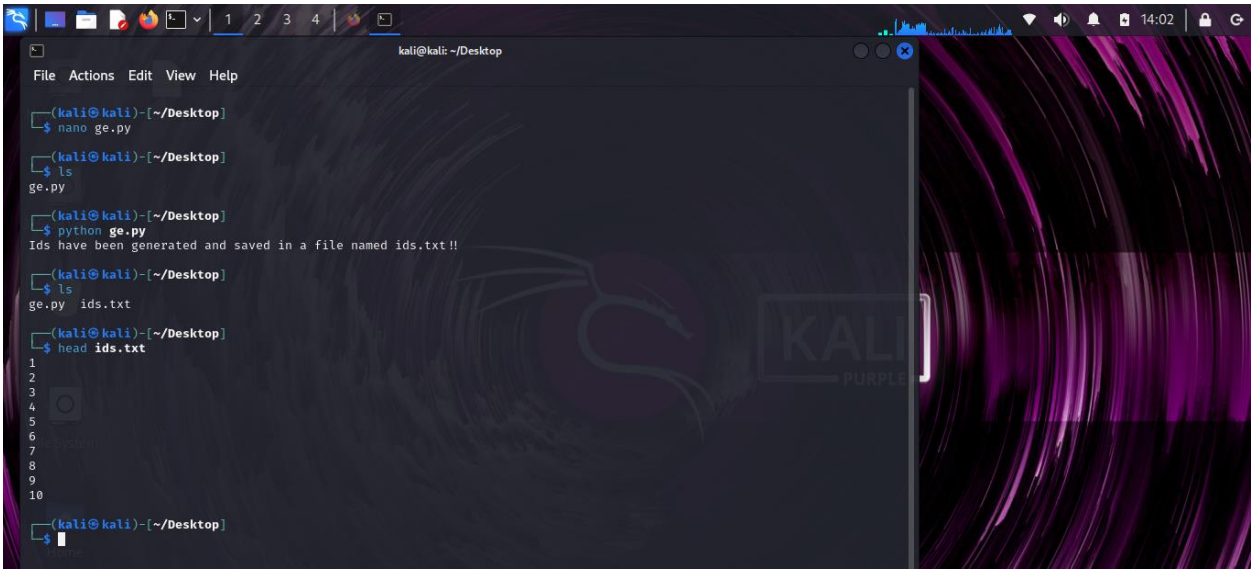
I first generated a ids wordlist using a bash script "f**or i in $(seq 1 1000); do echo $i >> ids.txt; done**"



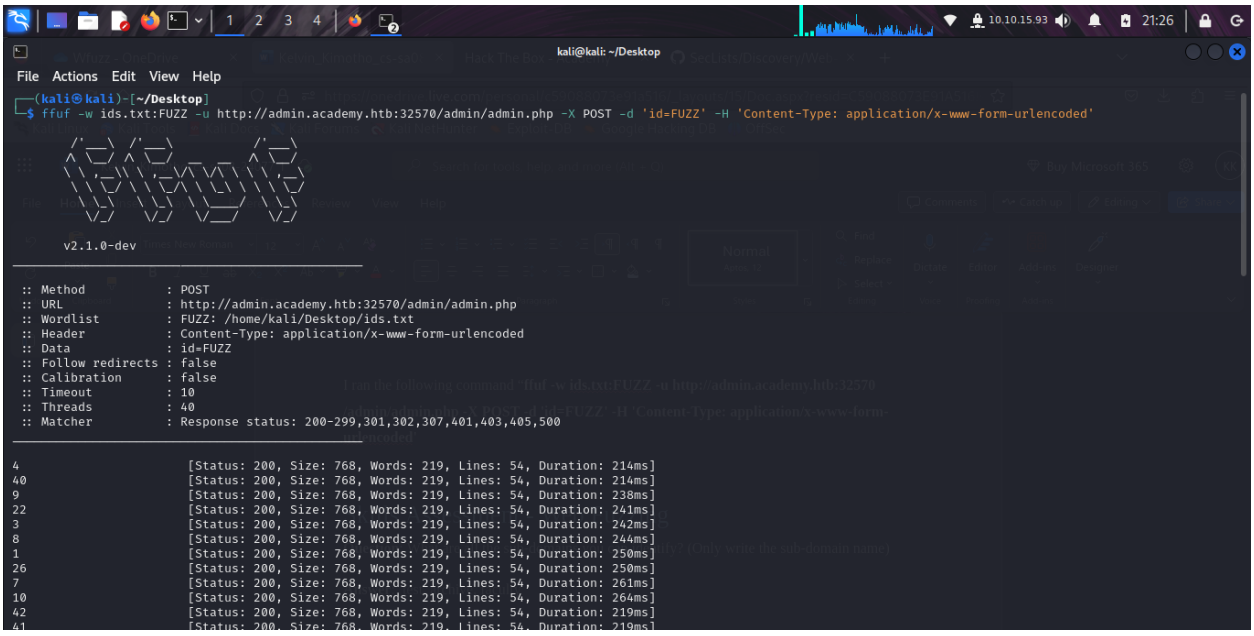
Saving the bash code in a bash script, adding execute privilege then running the script will also generate the same id list**.**



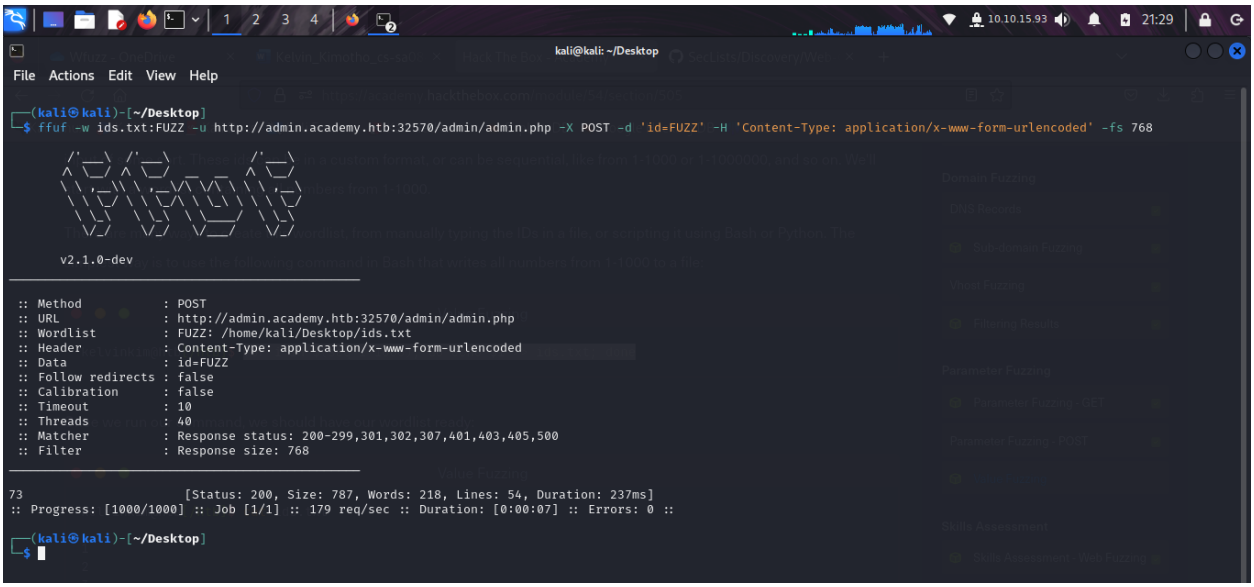
I also tried writing a python script to generate the ids and save them in a text file.



I ran the following command "**ffuf -w ids.txt:FUZZ -u http://admin.academy.htb:32570/admin/admin.php -X POST -d 'id=FUZZ' -H 'Content-Type: application/x-www-form-urlencoded'**

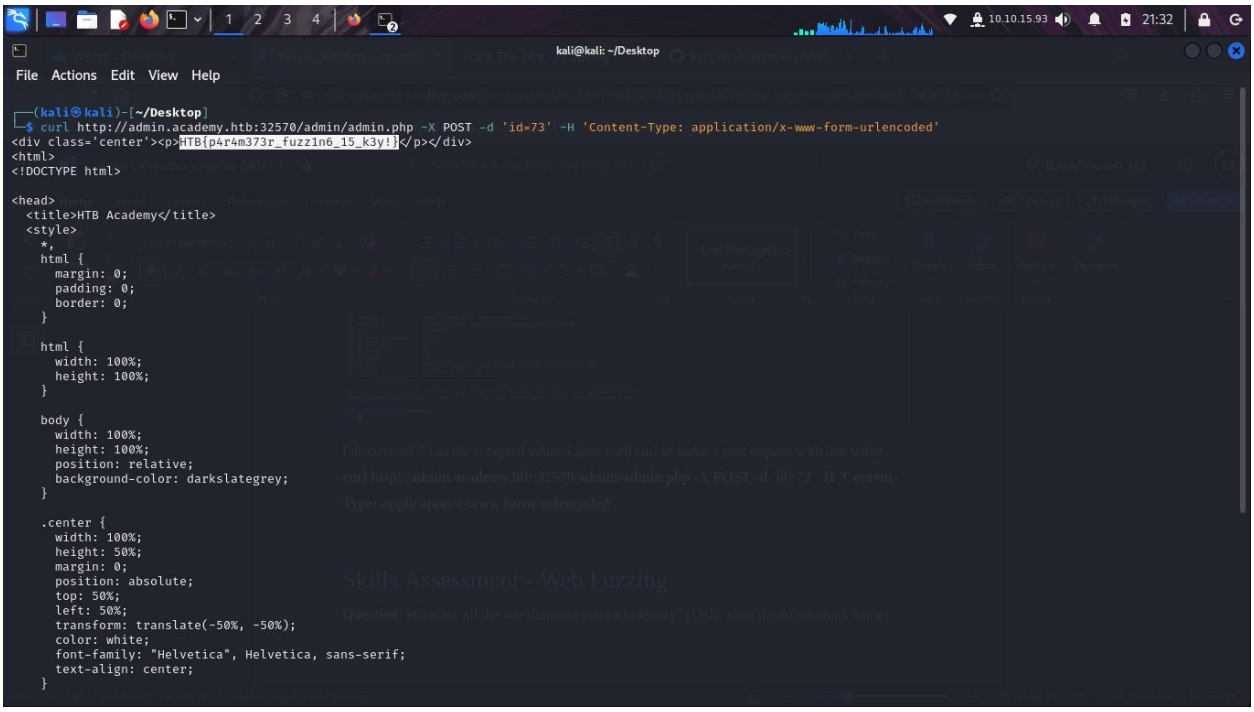


768 was the common response size so i used it to filter "**-fs 768".**



I discovered 73 as the accepted value. I then used curl to make a post request with that value.

curl http://admin.academy.htb:32570/admin/admin.php -X POST -d 'id=73' -H 'Content-Type: application/x-www-form-urlencoded'

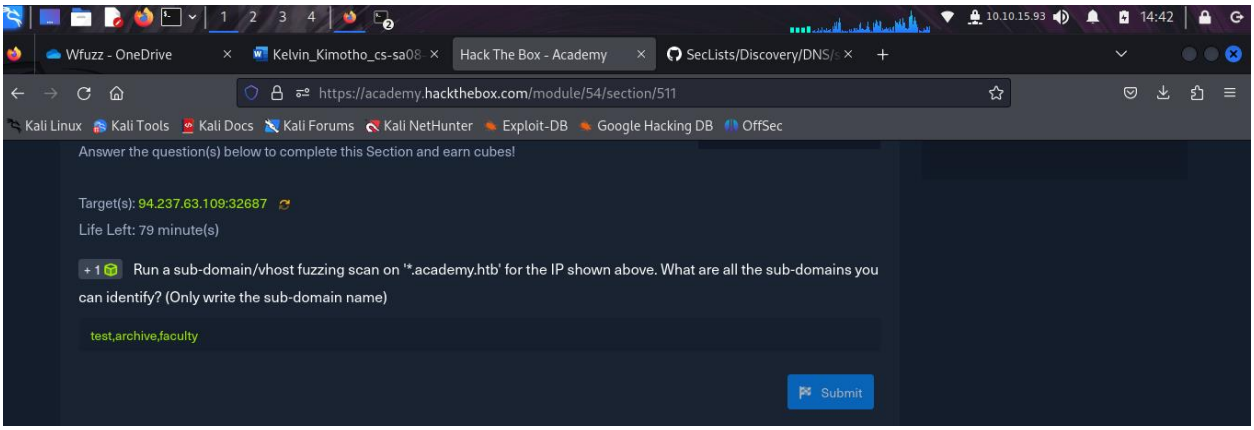


And that's how i captured the flag.

Skills Assessment - Web Fuzzing

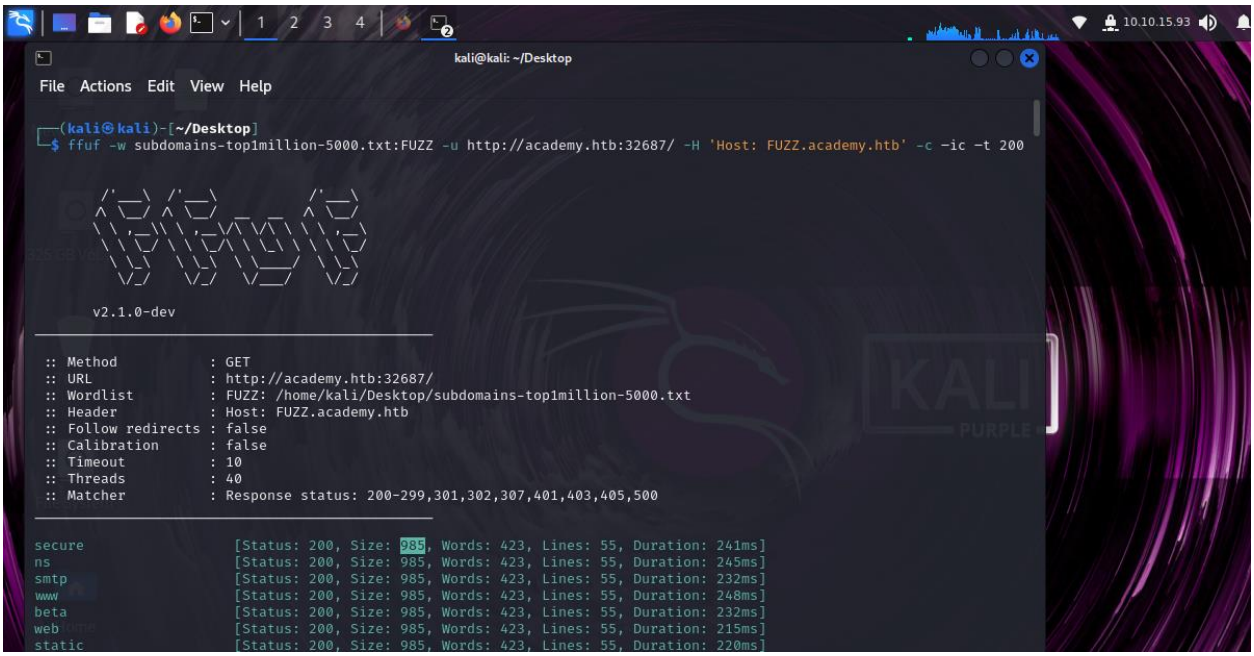
Question: What are all the sub-domains you can identify? (Only write the sub-domain name)

Answer: test, archive, faculty

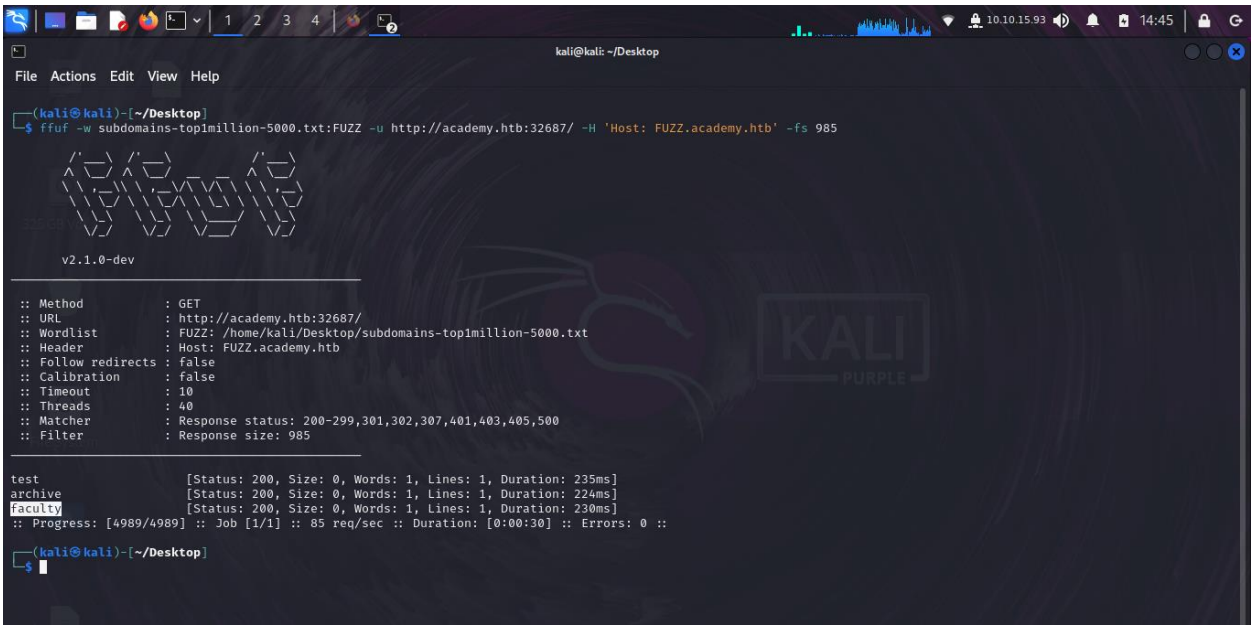


First i added academy.htb to my /**etc/hosts file "94.237.63 academy.htb"** .

"ffuf -w subdomains-top1million-5000.txt:FUZZ -u http://academy.htb:32687/ -H 'Host: FUZZ.academy.htb' -c –ic –t 200" is the command i ran.

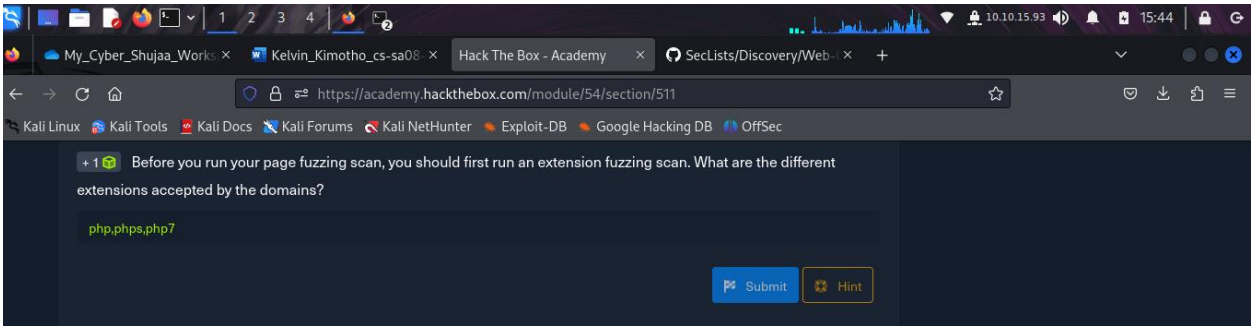


I added **–fs 985** to filter.



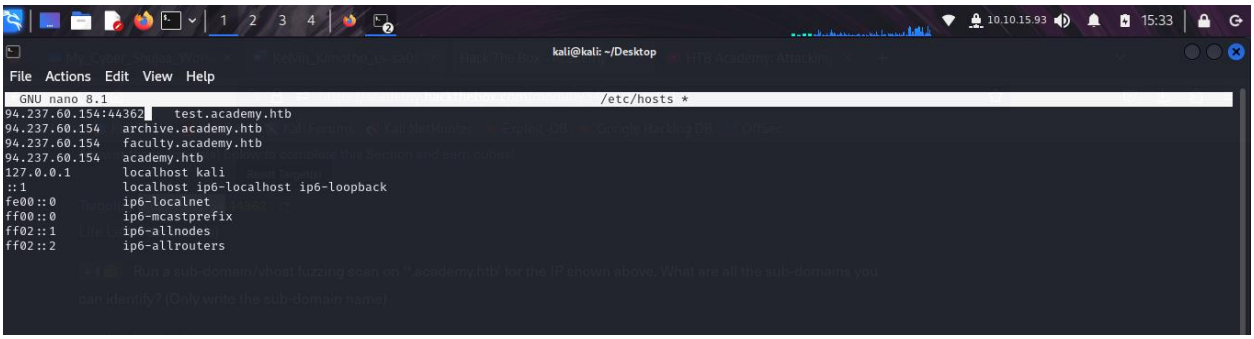
Question: Before you run your page fuzzing scan, you should first run an extension fuzzing scan. What are the different extensions accepted by the domains?

Answer: php,phps,php7

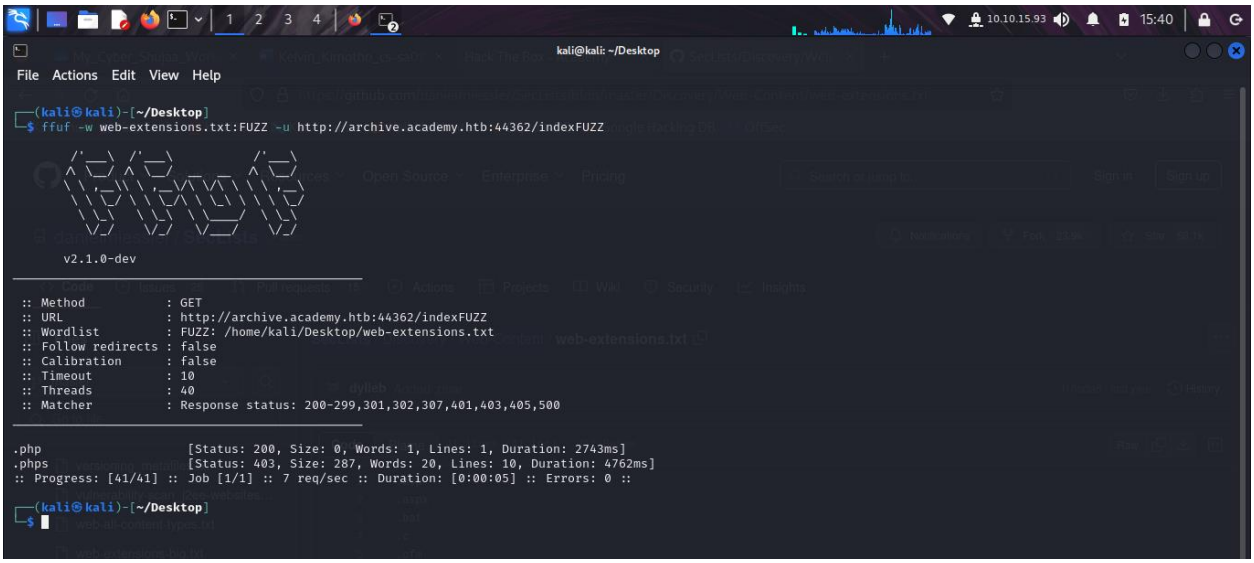


I first added the subdomains to the **/etc/hosts** file in my machine.

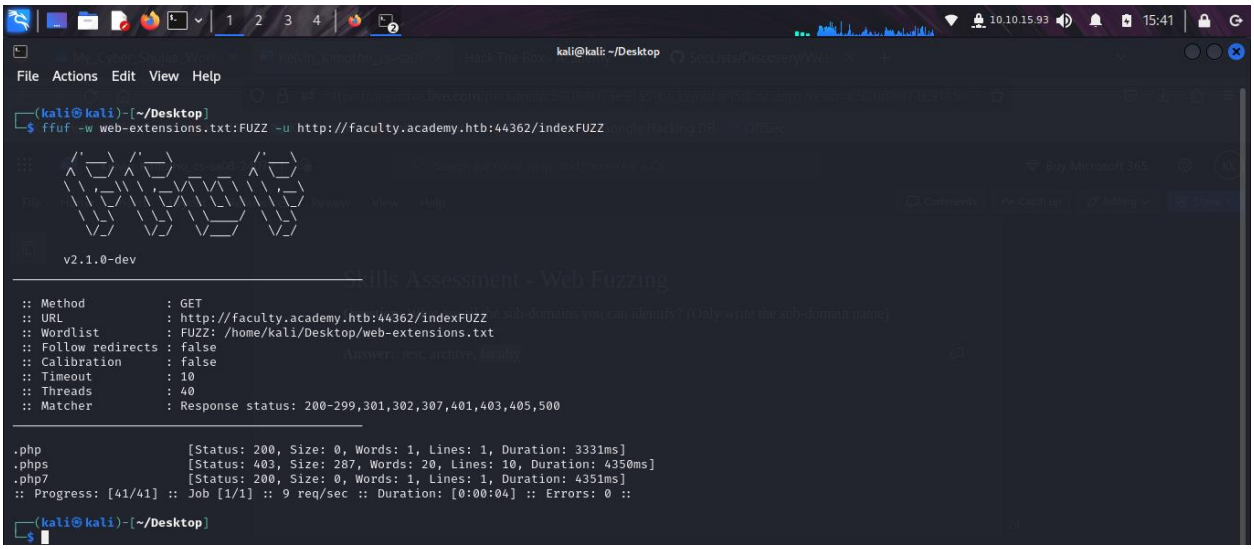
"IP **test.academy.htb, archive.academy.htb, faculty.academy.htb**"



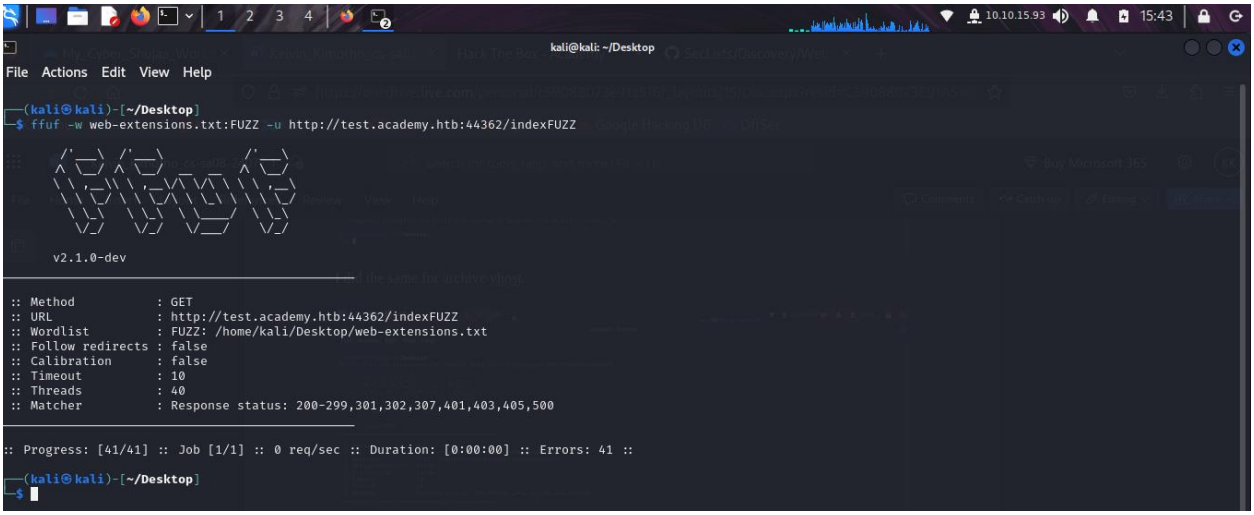
I then ran the following command" **ffuf -w web-extensions-big.txt:FUZZ -u http://archive.academy.htb:** 44362**/indexFUZZ**" for the **archive** vhost.



I did the same for archive vhost.

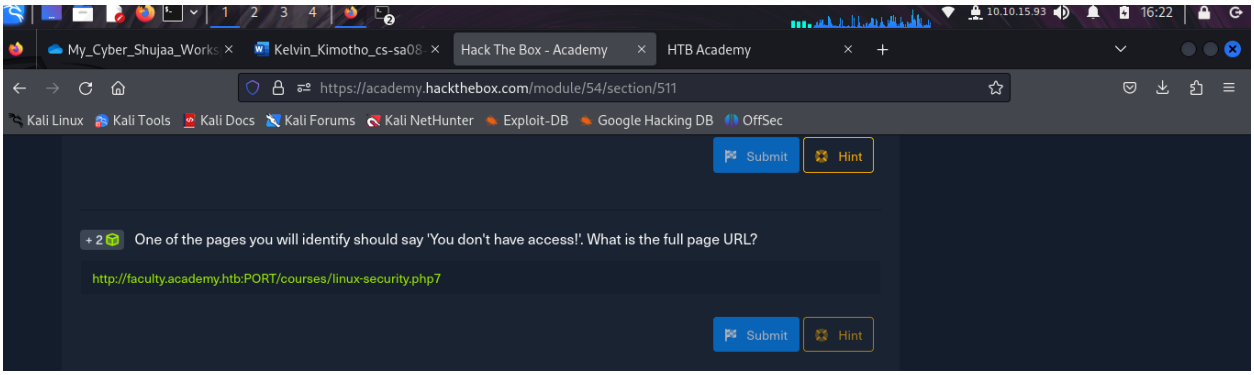


The test vhost revealed no extensions.



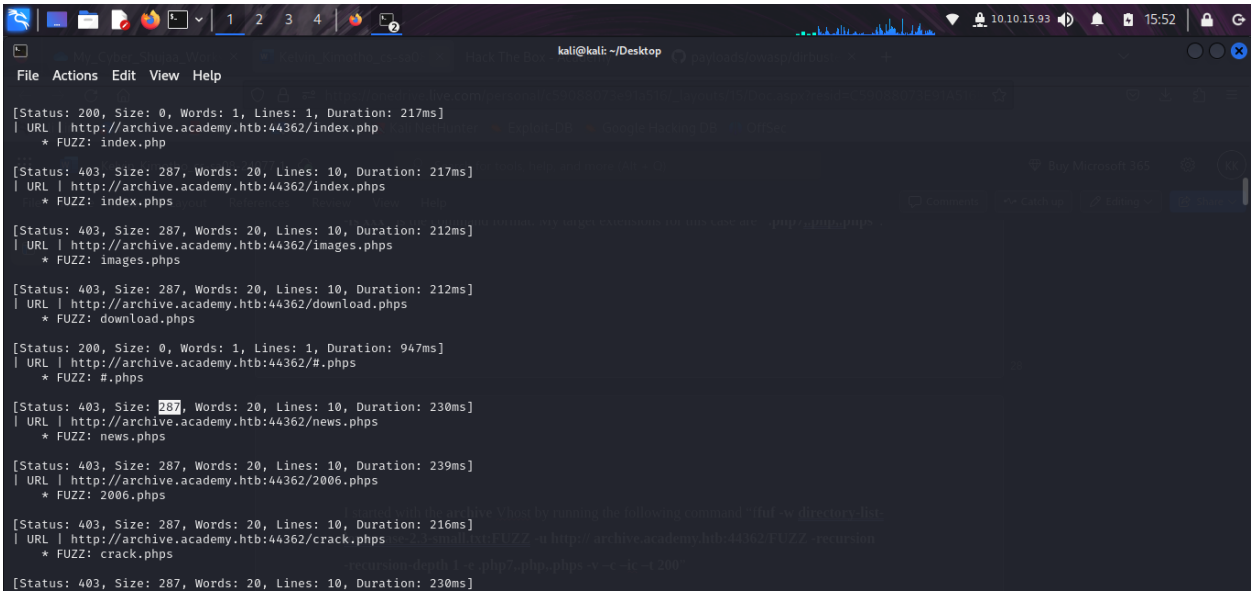
Question: One of the pages you will identify should say 'You don't have access!'. What is the full page URL?

Answer: http://faculty.academy.htb:44362/courses/linux-security.php7



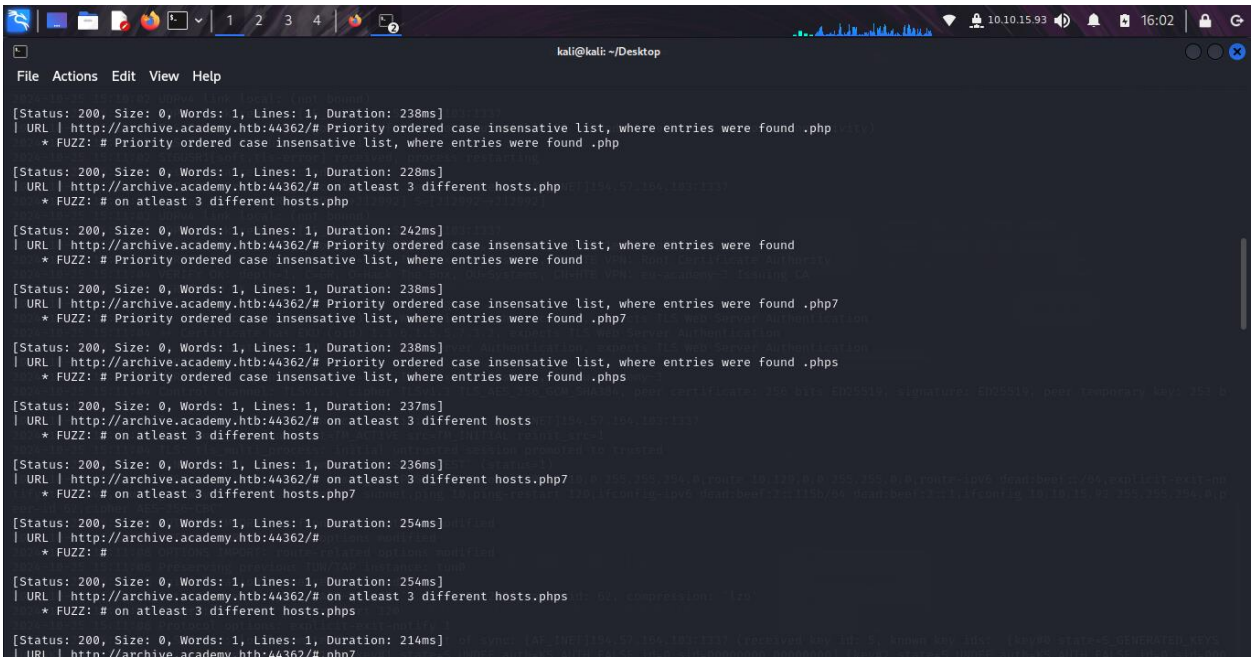
"ffuf -w directory-list-lowercase-2.3-small.txt:FUZZ -u http://vhost.academy.htb:PORT/FUZZ -recursion -recursion-depth 1 -e .ext1,.ext2,.ext3 -v -fs xxx" is the command format. My target extensions for this case are **".php7,.php,.phps**".

I started with the **archive** Vhost by running the following command "**ffuf -w directory-list-lowercase-2.3-small.txt:FUZZ -u http:// archive.academy.htb:44362/FUZZ -recursion -recursion-depth 1 -e .php7,.php,.phps -v –c –ic –t 200**"



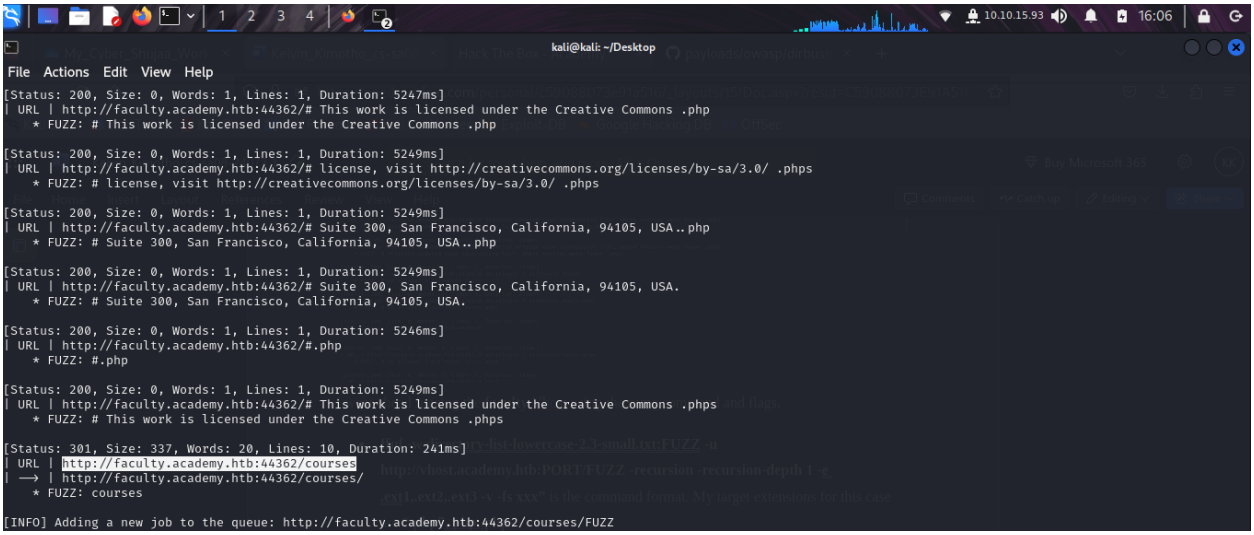
Size 287 was common so i filtered using it "-**fs 287**" since it was the common size.

I found nothing in archive Vhost.



I went ahead fuzzing the **faculty** Vhost using the same command and flags.

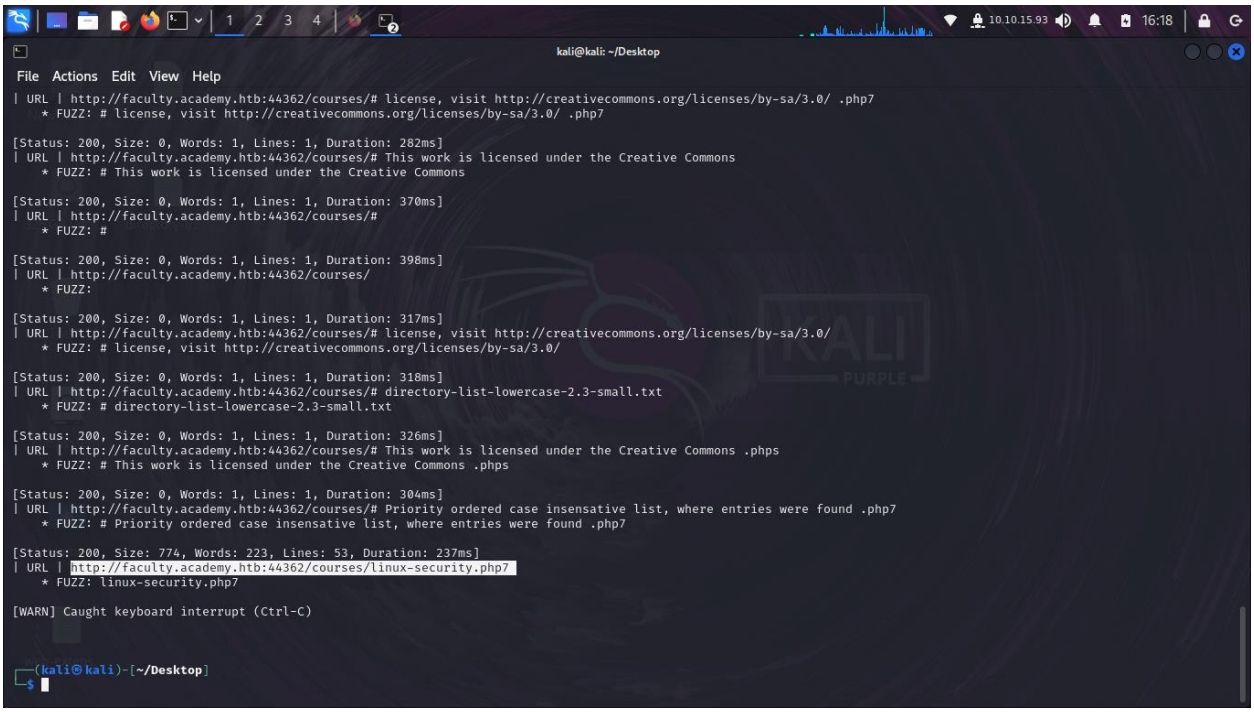
ffuf -w directory-list-lowercase-2.3-small.txt:FUZZ -u http://archive.academy.htb:44362/FUZZ -recursion -recursion-depth 1 -e .php7,.php,.phps -v –c –ic –t 200



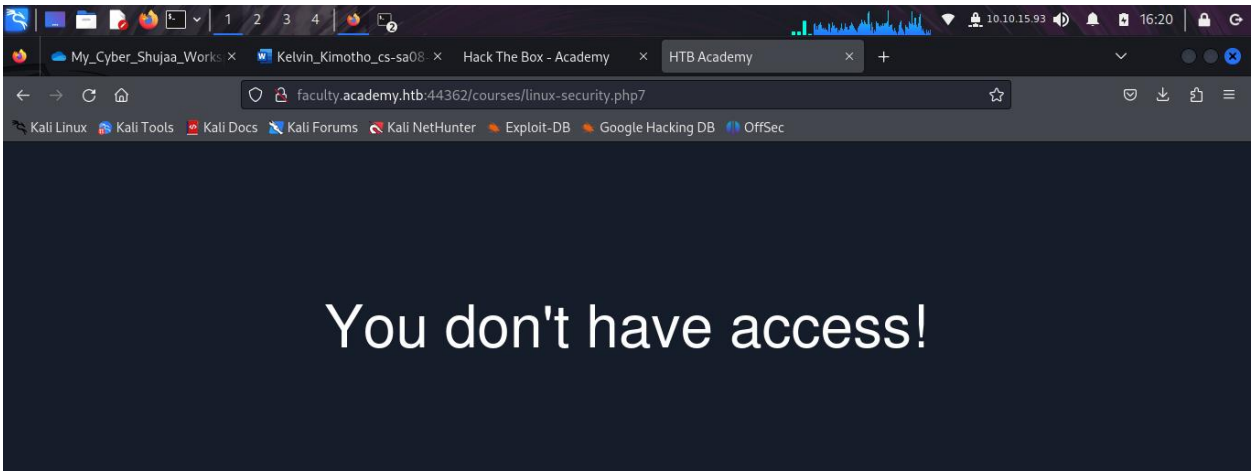
I discovered a courses directory in the archive vhost and i tried fuzzing it.

“**ffuf -w directory-list-lowercase-2.3-small.txt:FUZZ -u http://archive.academy.htb:44362/courses/FUZZ -recursion -recursion-depth 1 -e .php7,.php,.phps -v –c –ic –t 200 –fs 287 –t 1000”** -t 1000 for a 1000 threads.

In courses i found several files that matched the extension but **“linux-security.php7** was the unique one.

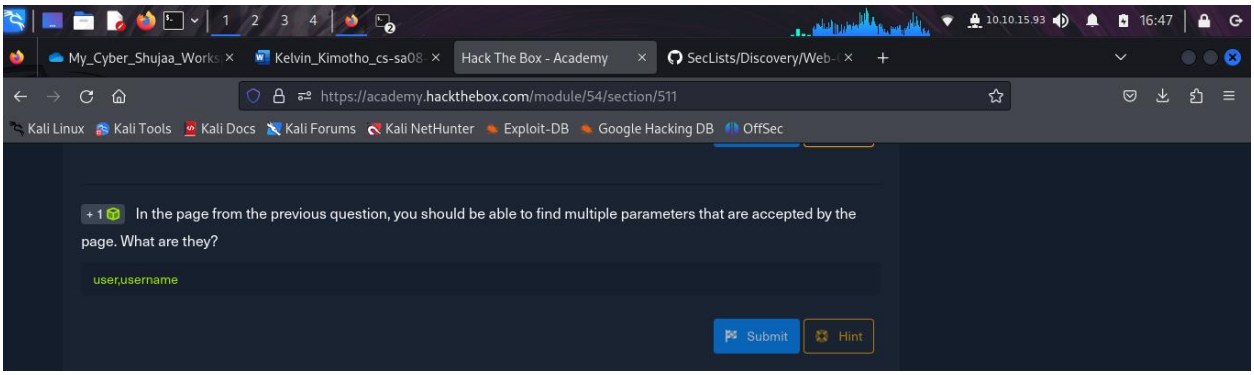


I visited the page**.**



Question: In the page from the previous question, you should be able to find multiple parameters that are accepted by the page. What are they?

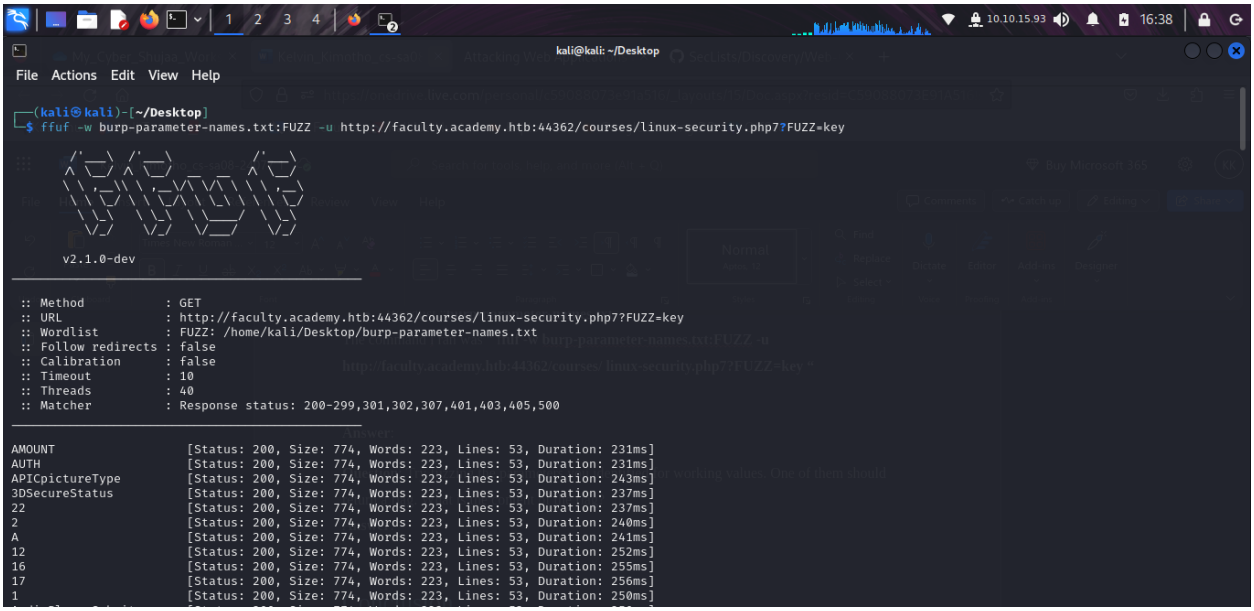
Answer: user, username



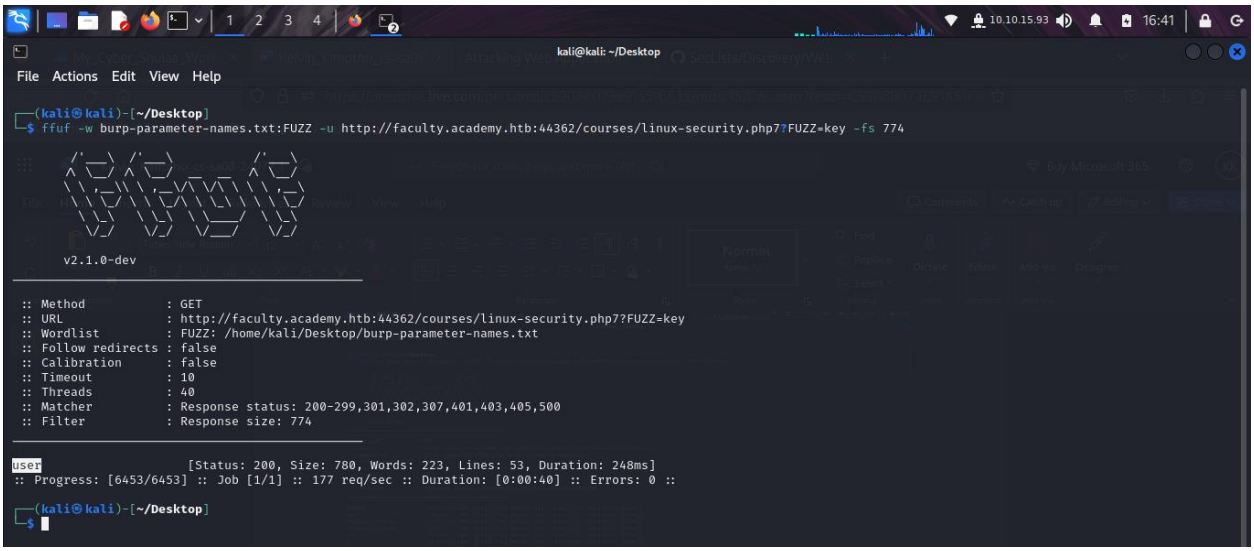
Here i did parameter fuzzing using command in "**ffuf -w wordlist.txt:FUZZ -u http://admin.academy.htb:PORT/admin/admin.php?FUZZ=key -fs xxx**" format for **GET parameter fuzzing**

- My target page was "**linux-security.php7"**
- Wordlist "**burp-parameter-names.txt**"

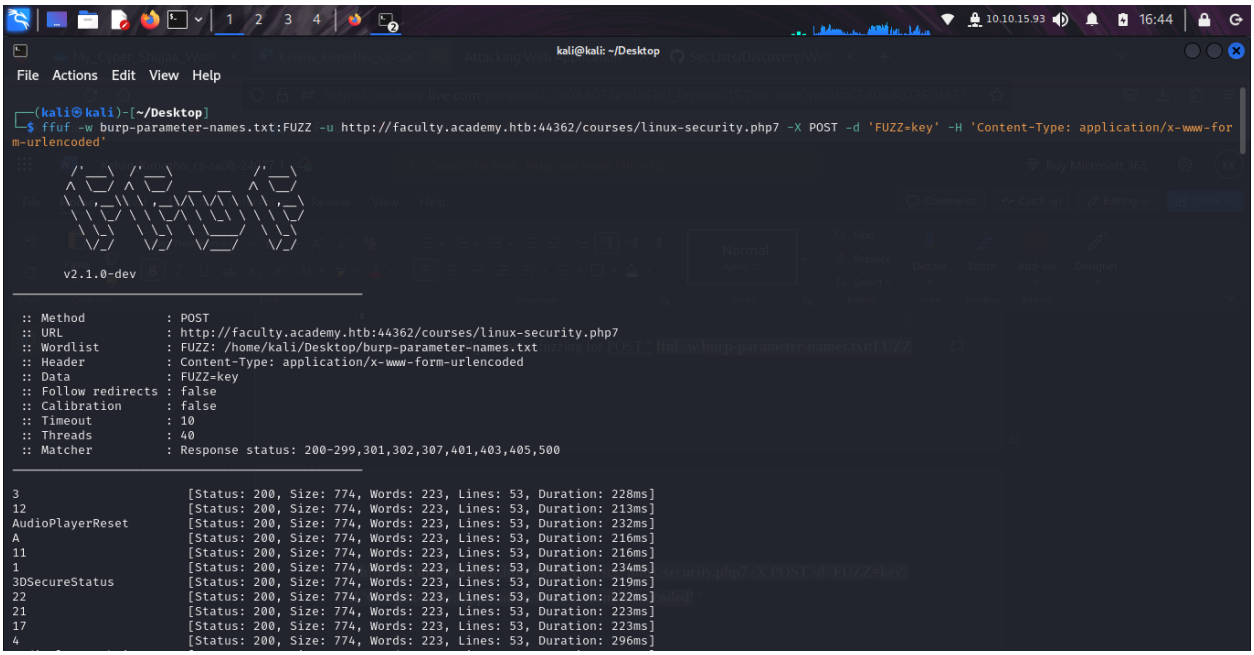
The command i ran was " **ffuf -w burp-parameter-names.txt:FUZZ -u http://faculty.academy.htb:44362/courses/ linux-security.php7?FUZZ=key "**



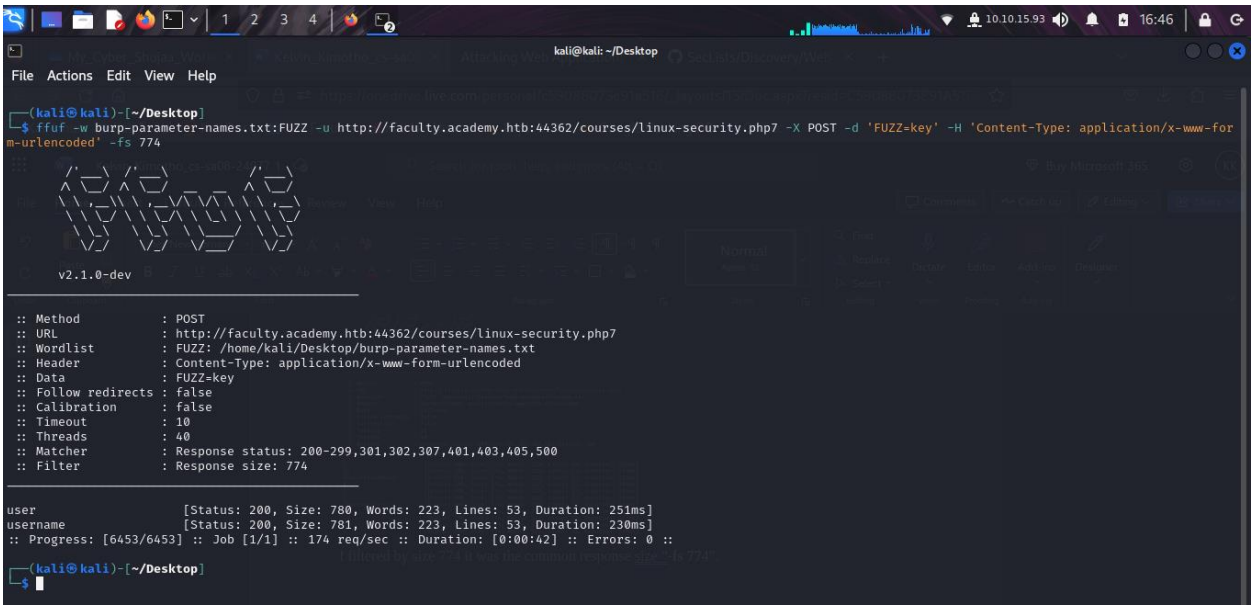
I filtered by **774** since it was the common size "-fs 774" and found a parameter " **user**".



I went ahead and tried parameter fuzzing for POST " **ffuf -w burp-parameter-names.txt:FUZZ -u http://faculty.academy.htb: 44362/courses/ linux-security.php7 -X POST -d 'FUZZ=key' -H 'Content-Type: application/x-www-form-urlencoded'** "



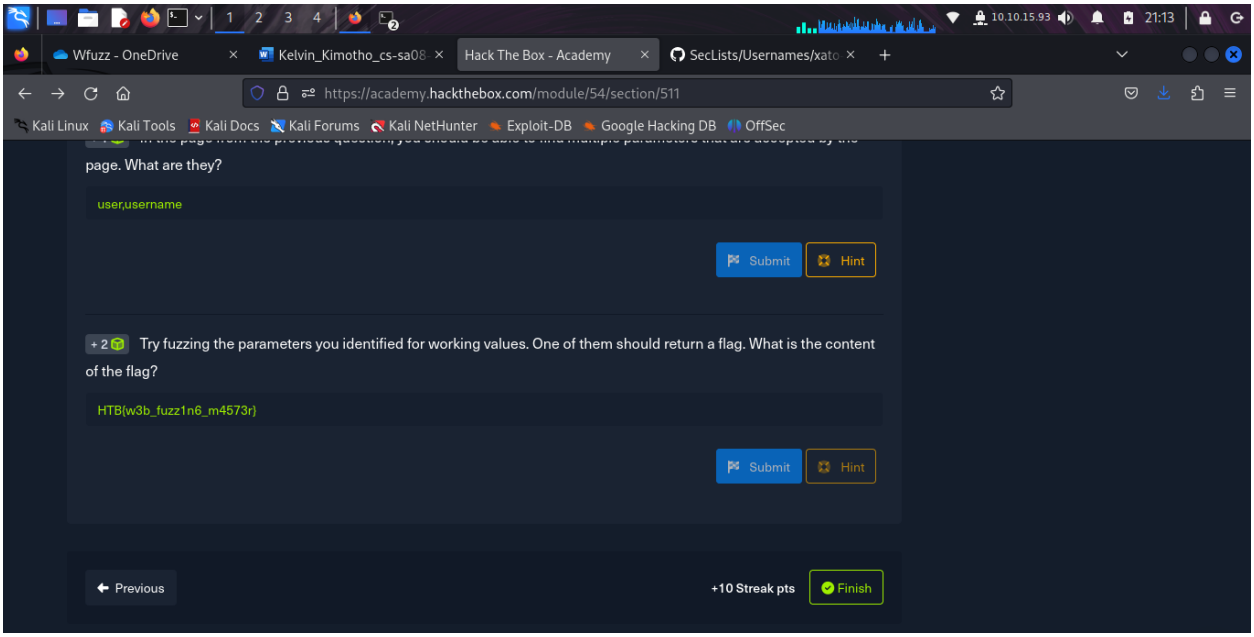
I filtered by size 774 it was the common response size "**-fs 774**".



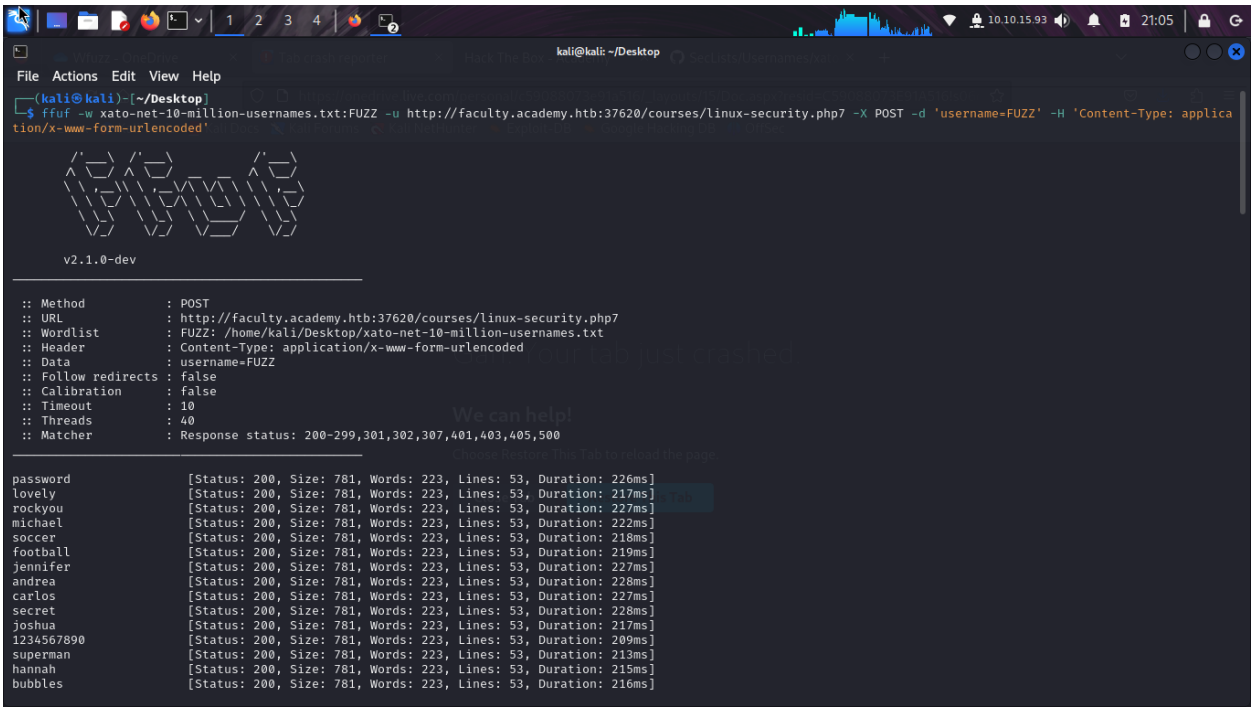
Here i found **user** and **username** as parameters.

Question: Try fuzzing the parameters you identified for working values. One of them should return a flag. What is the content of the flag?

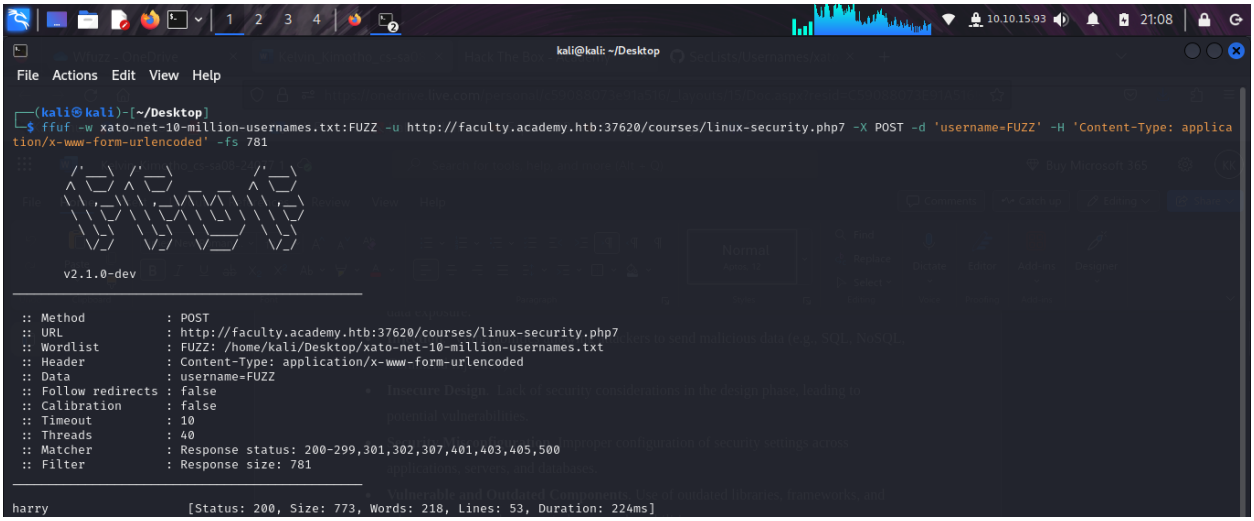
Answer: HTB{w3b_fuzz1n6_m4573r}



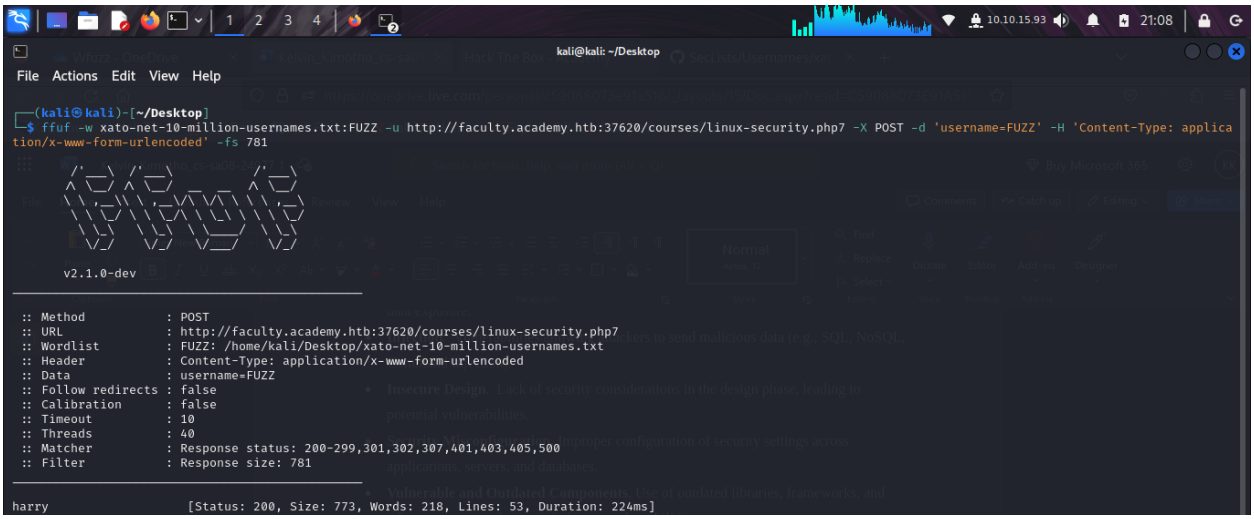
I downloaded a wordlist named “ato-net-10-million-usernames.txt” for common username and using the following command “**ffuf -w xato-net-10-million-usernames.txt:FUZZ -u http://faculty.academy.htb:** 37620**/courses/linux-security.php7 -X POST -d 'username=FUZZ' -H 'Content-Type: application/x-www-form-urlencoded**'”



I filtered by 781 since it was the common response size ”**-fs 781**”.

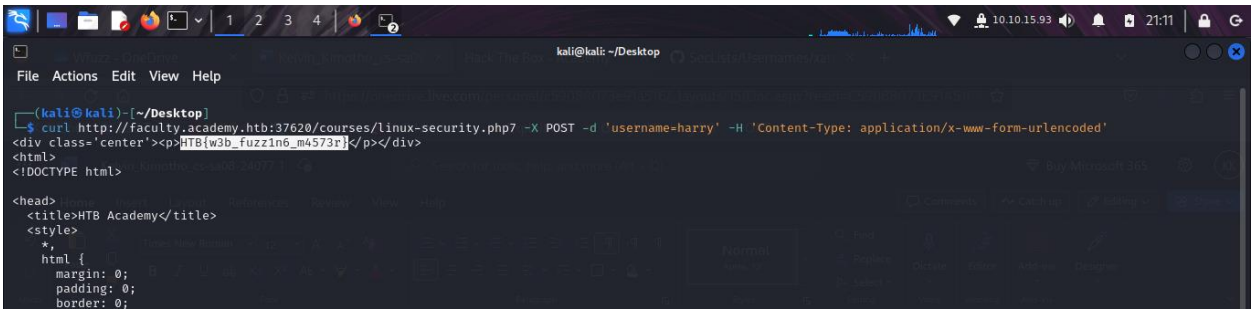


I found **harry** a user or a username parameter.

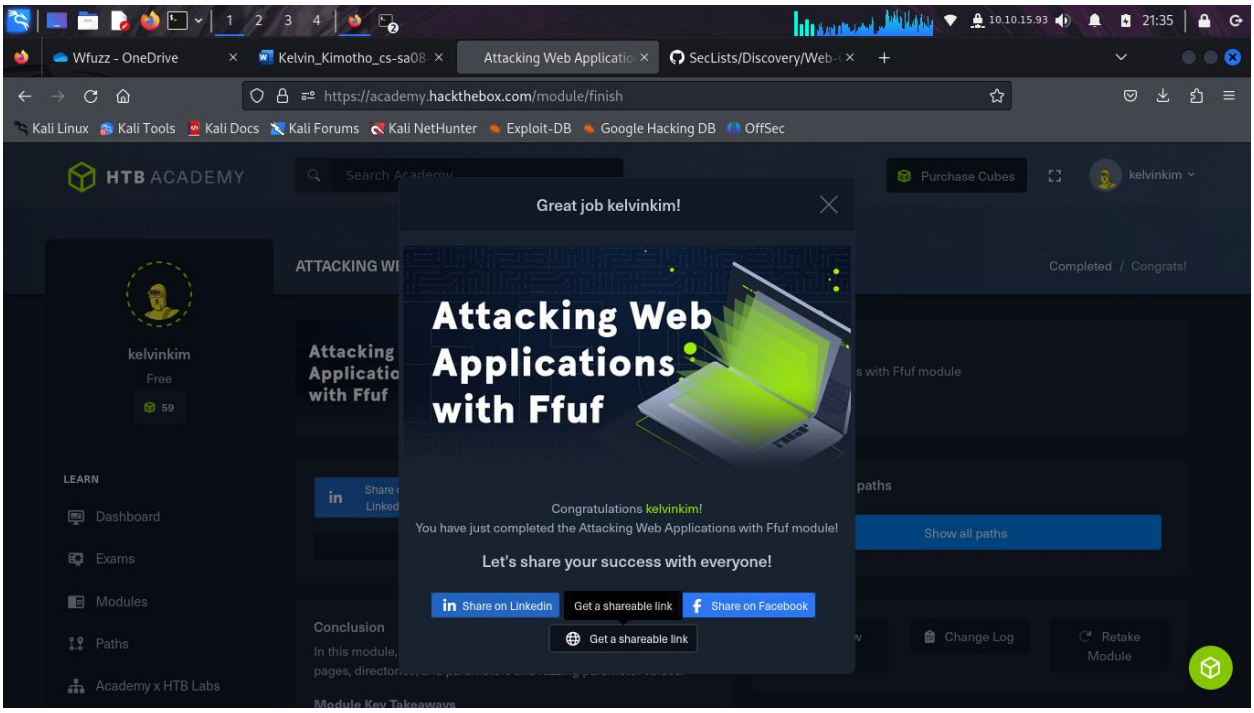


I went ahead and tried curl a tool for enumeration i started with **user** parameter.

“curl http://faculty.academy.htb:37620/courses/linux-security.php7 -X POST -d 'username=harry' -H 'Content-Type: application/x-www-form-urlencoded'”



Thats how i captured the hidden flag.



Conclusion

In summary, this comprehensive exploration of web application fuzzing using the ffuf tool has underscored the critical importance of identifying hidden resources and vulnerabilities within web applications. By conducting sub-domain, directory, and parameter fuzzing, I was able to uncover various elements of the application, including restricted areas and parameters that could be exploited. This hands-on approach demonstrated the utility of structured wordlists and automated scanning techniques in enhancing penetration testing effectiveness. Ultimately, the successful retrieval of a flag exemplified the practical benefits of fuzzing, emphasizing its role in improving web application security.