

# ***Introduction***

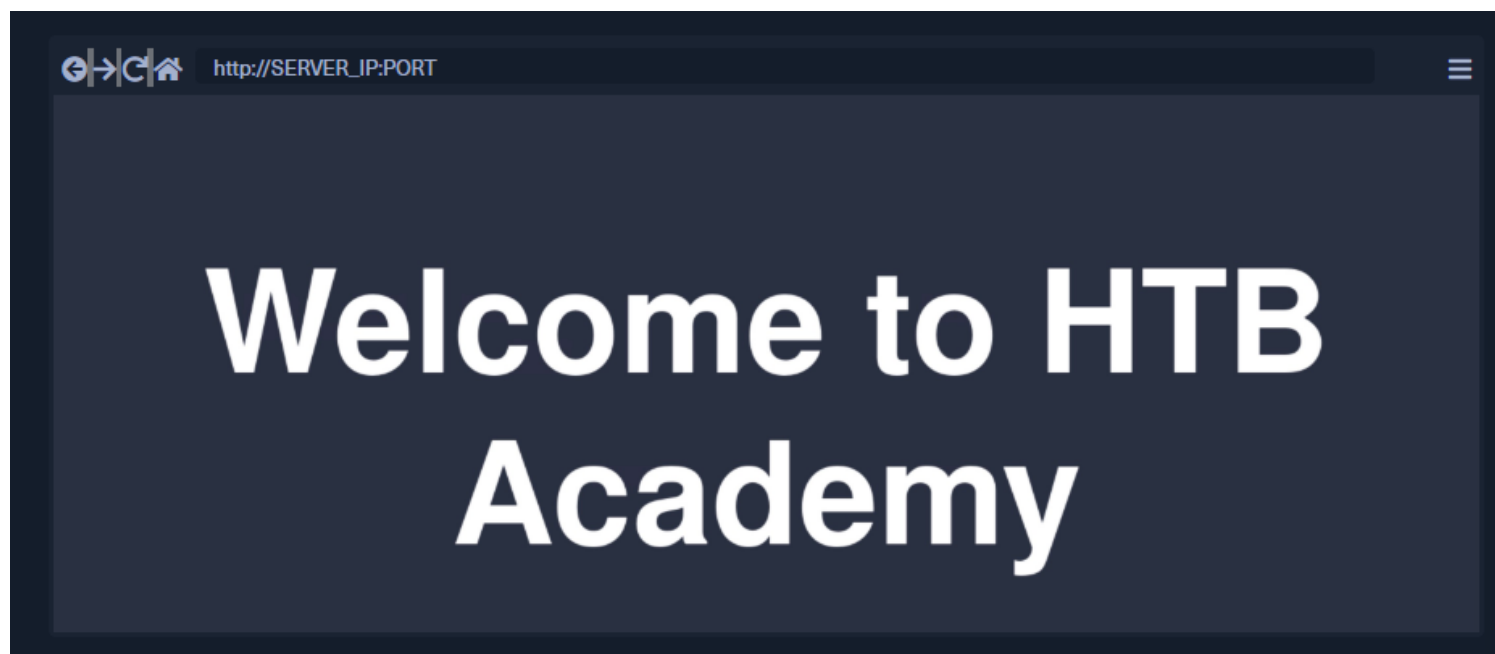
Tools such as `ffuf` provide us with a handy automated way to fuzz the web application's individual components or a web page. This means, for example, that we use a list that is used to send requests to the webserver if the page with the name from our list exists on the webserver. If we get a response code 200, then we know that this page exists on the webserver, and we can look at it manually.

- Fuzzing for directories
- Fuzzing for files and extensions
- Identifying hidden vhosts
- Fuzzing for PHP parameters
- Fuzzing for parameter values

## ***Web Fuzzing***

We will start by learning the basics of using `ffuf` to fuzz websites for directories.

We run the exercise in the question below, and visit the URL it gives us, and we see the following website:



The website has **no links to anything else**, nor does it give us any information that can lead us to more pages. So, it looks like our **only option is to 'fuzz' the website**.

## Fuzzing

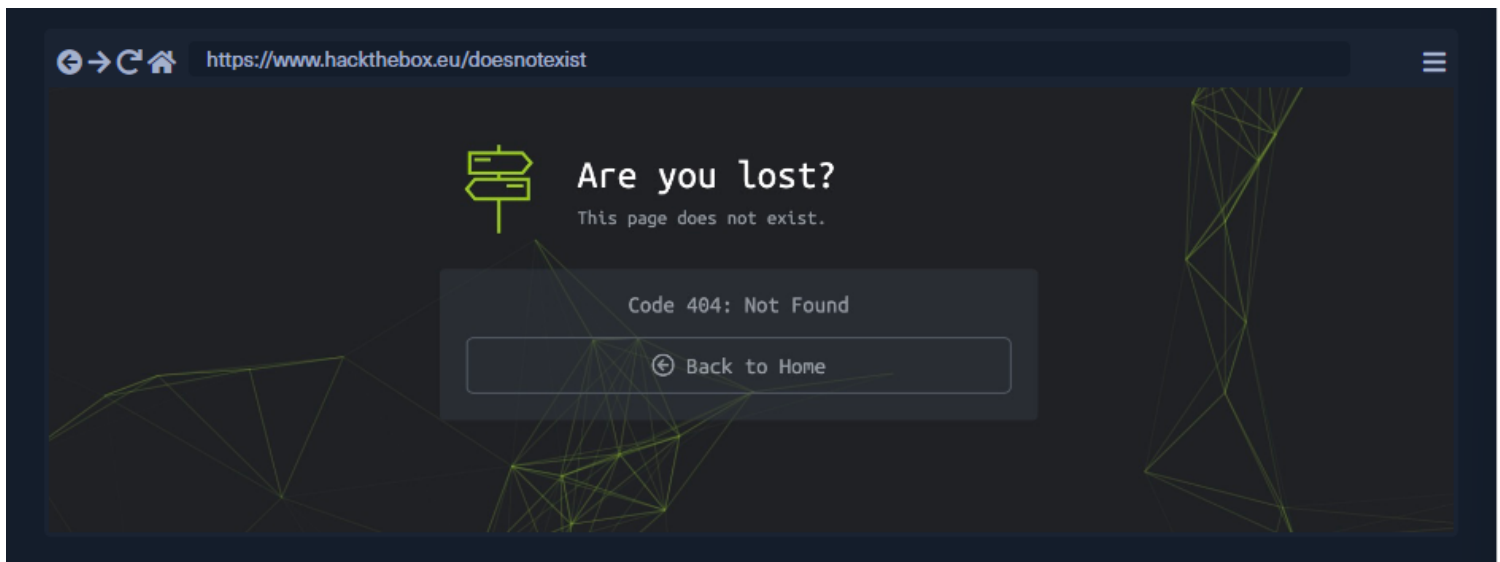
The term fuzzing refers to a **testing technique** that sends **various types of user input to a certain interface to study how it would react**.

If we were fuzzing for SQL injection vulnerabilities, we would be sending random special characters and seeing how the server would react. If we were fuzzing for a buffer overflow, we would be sending long strings and incrementing their length to see if and when the binary would break.

We usually utilize **pre-defined wordlists** of commonly used terms for each type of test for web fuzzing to see if the webserver would accept them.

This is done because web servers do not usually provide a directory of all available links and domains (unless terribly configured), and so we would have to check for various links and see which ones return pages.

For example, if we visit <https://www.hackthebox.eu/does not exist>, we would get an **HTTP code 404** Page Not Found, and see the below page:



However, if we visit a page that exists, like /login, we would get the login page and get an **HTTP code 200** OK, and see the below page:



This is the basic idea behind web fuzzing for pages and directories. Still, we cannot do this manually, as it will take forever. This is why we have tools that do this automatically, efficiently, and very quickly. Such tools send **hundreds of requests every second**, study the response HTTP code, and determine whether the page exists or not. Thus, we can quickly determine what pages exist and then manually examine them to see their content.

## Wordlists


To determine which pages exist, we should have a **wordlist containing commonly used words** for web directories and pages, very similar to a Password Dictionary Attack, which we will discuss later in the module. Though this will not reveal all pages under a specific website, as some pages are randomly named or use unique names, in general, this returns the majority of pages,

reaching up to 90% success rate on some websites.

We will not have to reinvent the wheel by manually creating these wordlists, as great efforts have been made to search the web and determine the most commonly used words for each type of fuzzing.

Some of the most commonly used wordlists can be found under the [GitHub SecLists repository](#), which categorizes wordlists under various types of fuzzing, even including commonly used passwords, which we'll later utilize for Password Brute Forcing

Within our PwnBox, we can find the entire SecLists repo available under /opt/useful/SecLists. The specific wordlist we will be utilizing for pages and directory fuzzing is another commonly used wordlist called [directory-list-2.3](#), and it is available in various forms and sizes. We can find the one we will be using under:



```
Vigneswar@htb[/htb]$ locate directory-list-2.3-small.txt
/opt/useful/SecLists/Discovery/Web-Content/directory-list-2.3-small.txt
```

Tip: Taking a look at this wordlist we will notice that it contains copyright comments at the beginning, which can be considered as part of the wordlist and clutter the results. We can use the following command to get rid of these lines with the `-ic` flag.

## ***Directory Fuzzing***

Ffuf

As a new user of this tool, we will start by issuing the `ffuf -h` command to see how the tools can be used:

```
Vigneswar@htb[/htb]$ ffuf -h
```

#### HTTP OPTIONS:

```
-H          Header `Name: Value`, separated by colon. Multiple -H flags are accepted.
-X          HTTP method to use (default: GET)
-b          Cookie data `NAME1=VALUE1; NAME2=VALUE2` for copy as curl functionality.
-d          POST data
-recursion  Scan recursively. Only FUZZ keyword is supported, and URL (-u) has to end in it. (de
-recursion-depth Maximum recursion depth. (default: 0)
-u          Target URL
```

...SNIP...

#### MATCHER OPTIONS:

```
-mc          Match HTTP status codes, or "all" for everything. (default: 200,204,301,302,307,401,
-ms          Match HTTP response size
```

...SNIP...

#### FILTER OPTIONS:

```
-fc          Filter HTTP status codes from response. Comma separated list of codes and ranges
-fs          Filter HTTP response size. Comma separated list of sizes and ranges
```

...SNIP...

#### INPUT OPTIONS:

...SNIP...

```
-w          Wordlist file path and (optional) keyword separated by colon. eg. '/path/to/wordlist
```

#### OUTPUT OPTIONS:

```
-o          Write output to file
```

...SNIP...

## Directory Fuzzing

As we can see from the example above, the main two options are **-w** for wordlists and **-u** for the URL. We can assign a keyword to a wordlist to refer to it where we want to fuzz. For example, we can pick our wordlist and assign the keyword **FUZZ** to it by adding **:FUZZ** after it:

```
Vigneswar@htb[/htb]$ ffuf -w /opt/useful/SecLists/Discovery/Web-Content/directory-list-2.3-small.txt:FUZZ
```

Next, as we want to be fuzzing for web directories, we can place the **FUZZ** keyword where the directory would be within our URL, with:

```
Vigneswar@htb[/htb]$ ffuf -w <SNIP> -u http://SERVER_IP:PORT/FUZZ
```

Now, let's start our target in the question below and run our final command on it:

```
Vigneswar@htb[/htb]$ ffuf -w /opt/useful/SecLists/Discovery/Web-Content/directory-list-2.3-small.txt:FUZZ

  /'___\  /'___\  /'___\
 /\  \_/\ /\  \_/\  \_/\
 \ \ ,__\ \ \ ,__\ \ \ ,__\
  \ \_/\  \ \_/\  \ \_/\
   \ \_/\  \ \_/\  \ \_/\
    \ \_/\  \ \_/\  \ \_/\

v1.1.0-git

-----

:: Method      : GET
:: URL         : http://SERVER_IP:PORT/FUZZ
:: Wordlist    : FUZZ: /opt/useful/SecLists/Discovery/Web-Content/directory-list-2.3-small.txt
:: Follow redirects : false
:: Calibration : false
:: Timeout     : 10
:: Threads    : 40
:: Matcher     : Response status: 200,204,301,302,307,401,403

-----

<SNIP>
blog [Status: 301, Size: 326, Words: 20, Lines: 10]
:: Progress: [87651/87651] :: Job [1/1] :: 9739 req/sec :: Duration: [0:00:09] :: Errors: 0 ::
```

We see that ffuf tested for almost **90k URLs in less than 10 seconds**. This speed may vary depending on your internet speed and ping if you used ffuf on your machine, but it should still be extremely fast.

We can even make it go faster if we are in a hurry by **increasing the number of threads to 200**, for example, with **-t 200**, but this is not recommended, especially when used on a remote site, as it may disrupt it, and **cause a Denial of Service**, or bring down your internet connection in severe cases. We do get a couple of hits, and we can visit one of them to verify that it exists:

We get an empty page, indicating that the directory does not have a dedicated page, but also shows that we do have access to it, as we do not get an HTTP code **404 Not Found** or **403 Access Denied**. In the next section, we will look for pages under this directory to see whether it is really empty or has hidden files and pages.

## Exercise

```
(vigneswar@vigneswar)-[~]
$ ffuf -w SecLists/Discovery/Web-Content/directory-list-2.3-small.txt:FUZZ -u http://94.237.59.185:49214/FUZZ -ic
```



```
:: Method      : GET
:: URL         : http://94.237.59.185:49214/FUZZ
:: Wordlist    : FUZZ: /home/vigneswar/SecLists/Discovery/Web-Content/directory-list-2.3-small.txt
:: Follow redirects : false
:: Calibration : false
:: Timeout     : 10
:: Threads    : 40
:: Matcher     : Response status: 200,204,301,302,307,401,403,405,500
```

```
[Status: 301, Size: 322, Words: 20, Lines: 10, Duration: 234ms]
* FUZZ: blog

[Status: 301, Size: 323, Words: 20, Lines: 10, Duration: 234ms]
* FUZZ: forum

[Status: 200, Size: 986, Words: 423, Lines: 56, Duration: 2376ms]
* FUZZ:
```

## Page Fuzzing

## Extension Fuzzing

before we start, we must find out what **types of pages** the website uses, like .html, .aspx, .php, or something else.

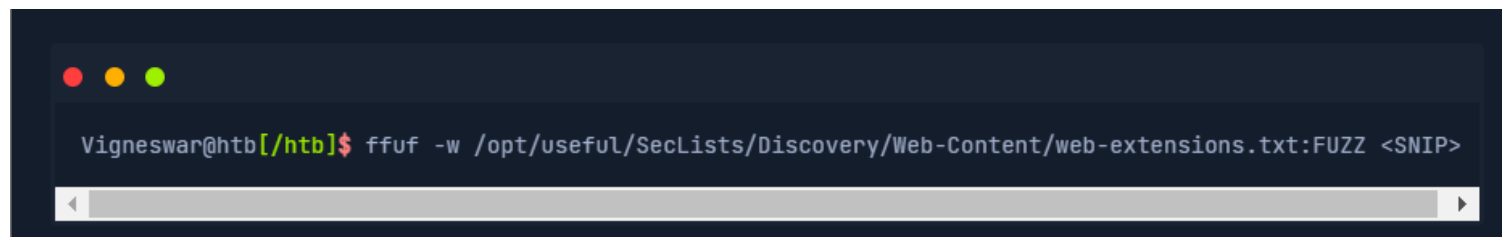
One common way to identify that is by finding the server type through the **HTTP response headers** and guessing the extension.

For example, if the server is apache, then it may be .php, or if it was IIS, then it could be .asp or .aspx, and so on.

This method is **not very practical**, though. So, we will again **utilize ffuf to fuzz the extension**, similar to how we fuzzed for directories.

Instead of placing the FUZZ keyword where the directory name would be, we would place it where the extension would be .FUZZ, and use a wordlist for common extensions.

We can utilize the following wordlist in SecLists for extensions:



```
Vigneswar@htb[/htb]$ ffuf -w /opt/useful/SecLists/Discovery/Web-Content/web-extensions.txt:FUZZ <SNIP>
```

Before we start fuzzing, we must specify which file that extension would be at the end of! We can always use **two wordlists** and have a **unique keyword** for each, and then do FUZZ\_1.FUZZ\_2 to fuzz for both.

However, there is one file we can always find in most websites, which is **index.\***, so we will use it as our file and fuzz extensions on it.



Now, we can rerun our command, carefully placing our **FUZZ** keyword where the extension would be after `index`:

```

t/useful/SecLists/Discovery/Web-Content/web-extensions.txt:FUZZ -u http://SERVER_IP:PORT/blog/indexFUZZ

/'---\
/\ \_/_/
\ \ ,--\
\ \ \_/_/
\ \ \_/_/
\ \ \_/_/
\ \ \_/_/
\ \ \_/_/

-----

SERVER_IP:PORT/blog/indexFUZZ
t/useful/SecLists/Discovery/Web-Content/web-extensions.txt

status: 200,204,301,302,307,401,403

-----

200, Size: 0, Words: 1, Lines: 1]
403, Size: 283, Words: 20, Lines: 10]
.] :: 0 req/sec :: Duration: [0:00:00] :: Errors: 0 ::

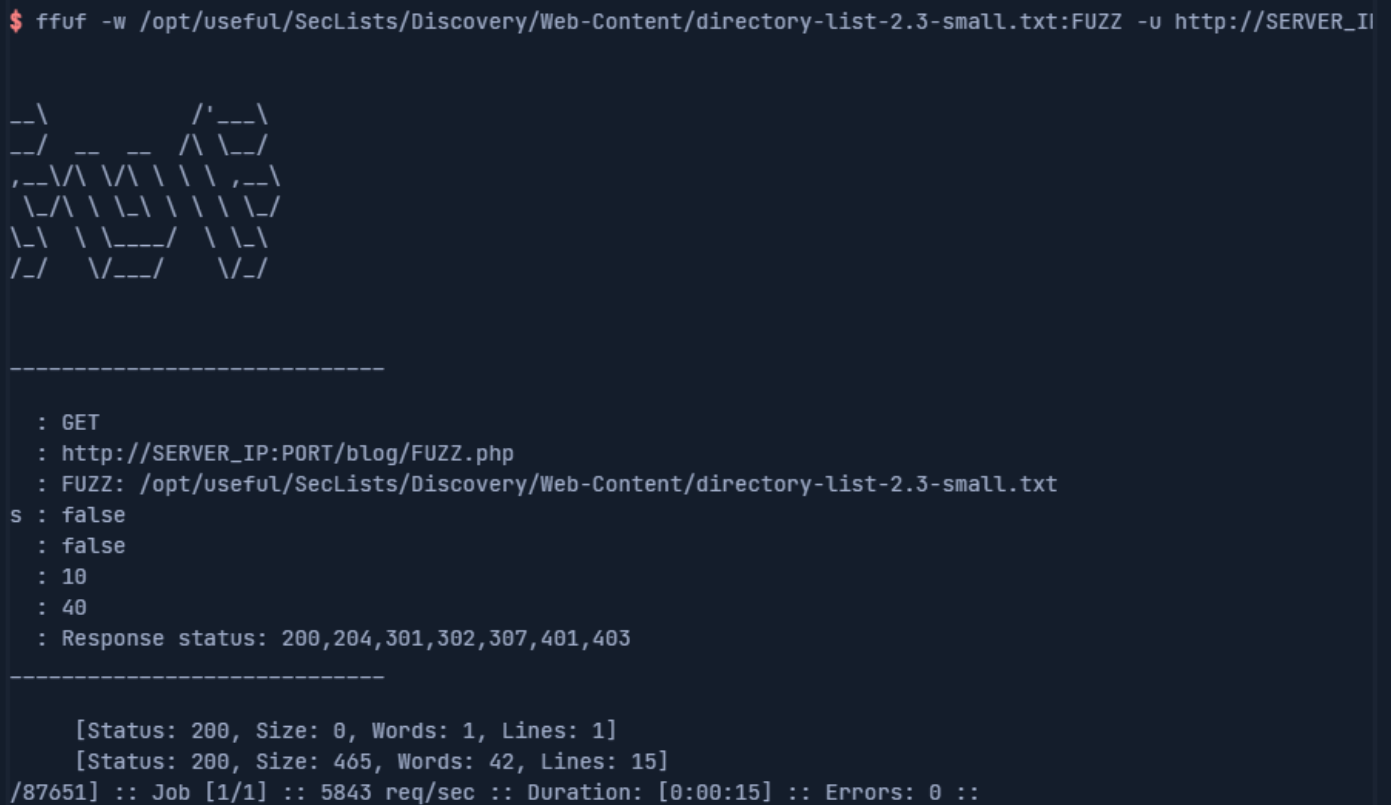
```

We do get a couple of hits, but only `.php` gives us a response with code 200. Great! We now know that this website runs on PHP to start fuzzing for PHP files.

## Page Fuzzing

We will now use the same concept of keywords we've been using with ffuf, use `.php` as the extension, place our **FUZZ keyword** where the `filename` should be, and use the same wordlist we used for fuzzing directories:

```
$ ffuf -w /opt/useful/SecLists/Discovery/Web-Content/directory-list-2.3-small.txt:FUZZ -u http://SERVER_IP:PORT/blog/FUZZ.php
```



```

-----
: GET
: http://SERVER_IP:PORT/blog/FUZZ.php
: FUZZ: /opt/useful/SecLists/Discovery/Web-Content/directory-list-2.3-small.txt
s : false
: false
: 10
: 40
: Response status: 200,204,301,302,307,401,403
-----

[Status: 200, Size: 0, Words: 1, Lines: 1]
[Status: 200, Size: 465, Words: 42, Lines: 15]
/87651] :: Job [1/1] :: 5843 req/sec :: Duration: [0:00:15] :: Errors: 0 ::

```

We get a couple of hits; both have an HTTP code 200, meaning we can access them. index.php has a size of 0, indicating that it is an empty page, while the other does not, which means that it has content. We can visit any of these pages to verify this:



## Exercise

1) Found the extension

```
(vigneswar@vigneswar)-[~]
$ ffuf -w SecLists/Discovery/Web-Content/web-extensions.txt:FUZZ -u http://94.237.59.185:49214/blog/indexFUZZ -ic

v2.0.0-dev

:: Method      : GET
:: URL         : http://94.237.59.185:49214/blog/indexFUZZ
:: Wordlist     : FUZZ: /home/vigneswar/SecLists/Discovery/Web-Content/web-extensions.txt
:: Follow redirects : false
:: Calibration : false
:: Timeout     : 10
:: Threads     : 40
:: Matcher     : Response status: 200,204,301,302,307,401,403,405,500

[Status: 403, Size: 281, Words: 20, Lines: 10, Duration: 3561ms]
* FUZZ: .phps

[Status: 200, Size: 0, Words: 1, Lines: 1, Duration: 4648ms]
* FUZZ: .php

:: Progress: [40/40] :: Job [1/1] :: 8 req/sec :: Duration: [0:00:04] :: Errors: 0 ::
```

## 2) Found pages

```
(vigneswar@vigneswar)-[~]
$ ffuf -w SecLists/Discovery/Web-Content/directory-list-2.3-small.txt -u http://94.237.59.185:49214/blog/FUZZ.php -ic

v2.0.0-dev

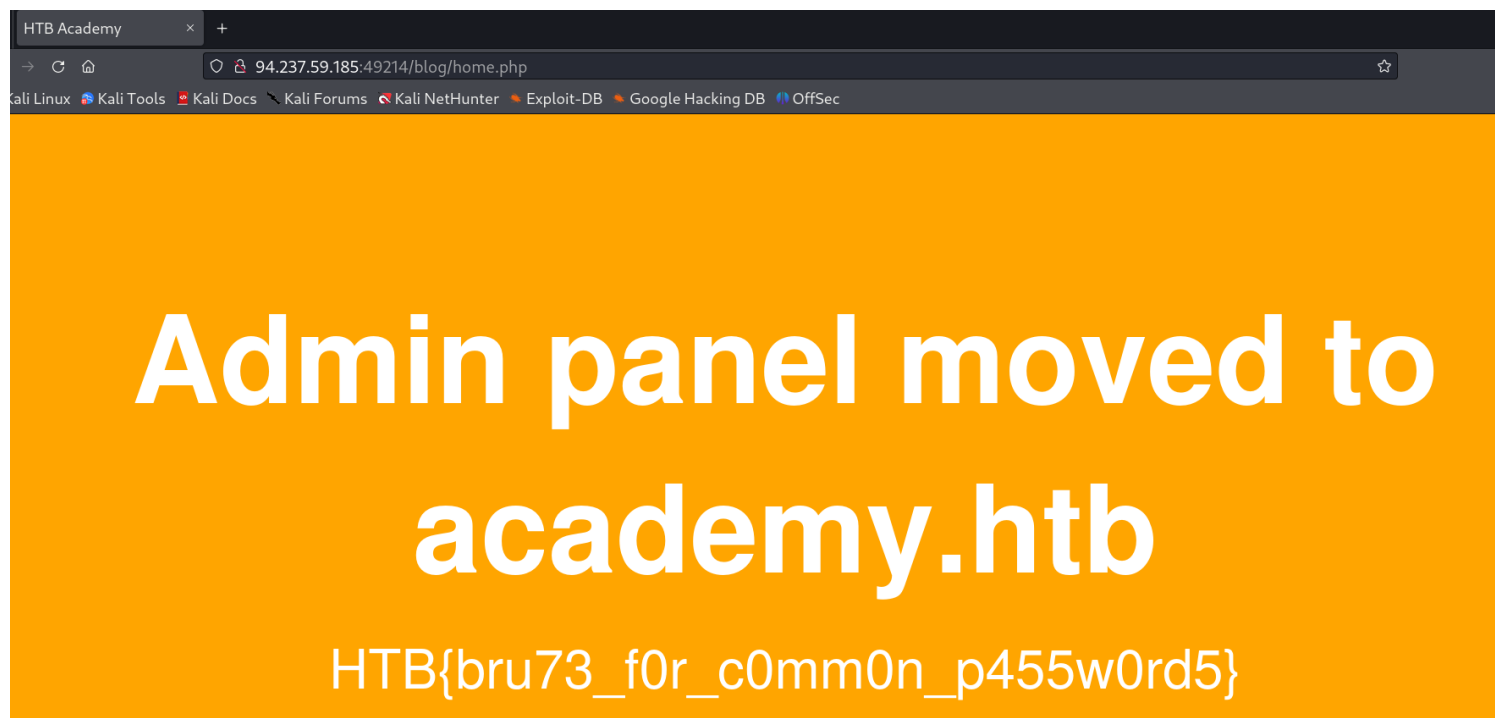
:: Method      : GET
:: URL         : http://94.237.59.185:49214/blog/FUZZ.php
:: Wordlist     : FUZZ: /home/vigneswar/SecLists/Discovery/Web-Content/directory-list-2.3-small.txt
:: Follow redirects : false
:: Calibration : false
:: Timeout     : 10
:: Threads     : 40
:: Matcher     : Response status: 200,204,301,302,307,401,403,405,500

[Status: 200, Size: 0, Words: 1, Lines: 1, Duration: 221ms]
* FUZZ: index

[Status: 200, Size: 1046, Words: 438, Lines: 58, Duration: 215ms]
* FUZZ: home

[Status: 403, Size: 281, Words: 20, Lines: 10, Duration: 235ms]
* FUZZ:
```

## 3) Got the flag



## ***Recursive Fuzzing***

If we had dozens of directories, each with their own subdirectories and files, this would take a very long time to complete. To be able to automate this, we will utilize what is known as **recursive fuzzing**.

### Recursive Flags

When we scan recursively, it automatically starts **another scan under any newly identified directories** that may have on their pages until it has fuzzed the main website and all of its subdirectories.

Some websites may have a **big tree of sub-directories**, like `/login/user/content/uploads/...etc`, and this will expand the scanning tree and may take a very long time to scan them all. This is why it is always advised to **specify a depth to our recursive scan**, such that it will not scan directories that are deeper than that depth. Once we fuzz the first directories, we can then pick the most interesting directories and run another scan to direct our scan better.

In ffuf, we can enable recursive scanning with the **-recursion flag**, and we can specify the depth with the **-recursion-depth flag**. If we specify `-recursion-depth 1`, it will only fuzz the main directories and their direct sub-directories. If any sub-sub-directories are identified (like `/login/user`, it will not fuzz them for pages). When using recursion in ffuf, we can **specify our extension with `-e .php`**

Finally, we will also add the **flag `-v` to output the full URLs**. Otherwise, it may be difficult to tell which `.php` file lies under which directory.

## Recursive Scanning

Let us repeat the first command we used, add the recursion flags to it while specifying `.php` as our extension, and see what results we get:

```
Discovery/Web-Content/directory-list-2.3-small.txt:FUZZ -u http://SERVER_IP:PORT/FUZZ -recursion -recursion-depth 1 -e .php -v
```

```
-----  
[Status: 200, Size: 986, Words: 423, Lines: 56] | URL | http://SERVER_IP:PORT/  
* FUZZ:  
  
[INFO] Adding a new job to the queue: http://SERVER_IP:PORT/forum/FUZZ  
[Status: 200, Size: 986, Words: 423, Lines: 56] | URL | http://SERVER_IP:PORT/inde  
* FUZZ: index.php  
  
[Status: 301, Size: 326, Words: 20, Lines: 10] | URL | http://SERVER_IP:PORT/blog  
* FUZZ: blog  
  
<...SNIP...>  
[Status: 200, Size: 0, Words: 1, Lines: 1] | URL | http://SERVER_IP:PORT/blog/inde  
* FUZZ: index.php  
  
[Status: 200, Size: 0, Words: 1, Lines: 1] | URL | http://SERVER_IP:PORT/blog/  
* FUZZ:  
  
<...SNIP...>
```

As we can see this time, the scan **took much longer**, sent almost **six times the number of requests**, and the **wordlist doubled in size** (once with `.php` and once without). Still, we got a large number of results, including all the results we previously identified, all with a single line of command.

## Exercise

1) Found the page with flag

```
(vigneswar@vigneswar)-[~]
$ ffuf -w SecLists/Discovery/Web-Content/directory-list-2.3-small.txt -u http://94.237.59.185:49214/forum/FUZZ -e .php -recursion -recursion-depth 1 -ic -v -t 100

v2.0.0-dev

:: Method      : GET
:: URL         : http://94.237.59.185:49214/forum/FUZZ
:: Wordlist     : FUZZ: /home/vigneswar/SecLists/Discovery/Web-Content/directory-list-2.3-small.txt
:: Extensions  : .php
:: Follow redirects : false
:: Calibration : false
:: Timeout      : 10
:: Threads     : 100
:: Matcher     : Response status: 200,204,301,302,307,401,403,405,500

[Status: 200, Size: 0, Words: 1, Lines: 1, Duration: 218ms]
| URL | http://94.237.59.185:49214/forum/index.php
* FUZZ: index.php

[Status: 200, Size: 0, Words: 1, Lines: 1, Duration: 1042ms]
| URL | http://94.237.59.185:49214/forum/
* FUZZ:

[Status: 403, Size: 281, Words: 20, Lines: 10, Duration: 4046ms]
| URL | http://94.237.59.185:49214/forum/.php
* FUZZ: .php

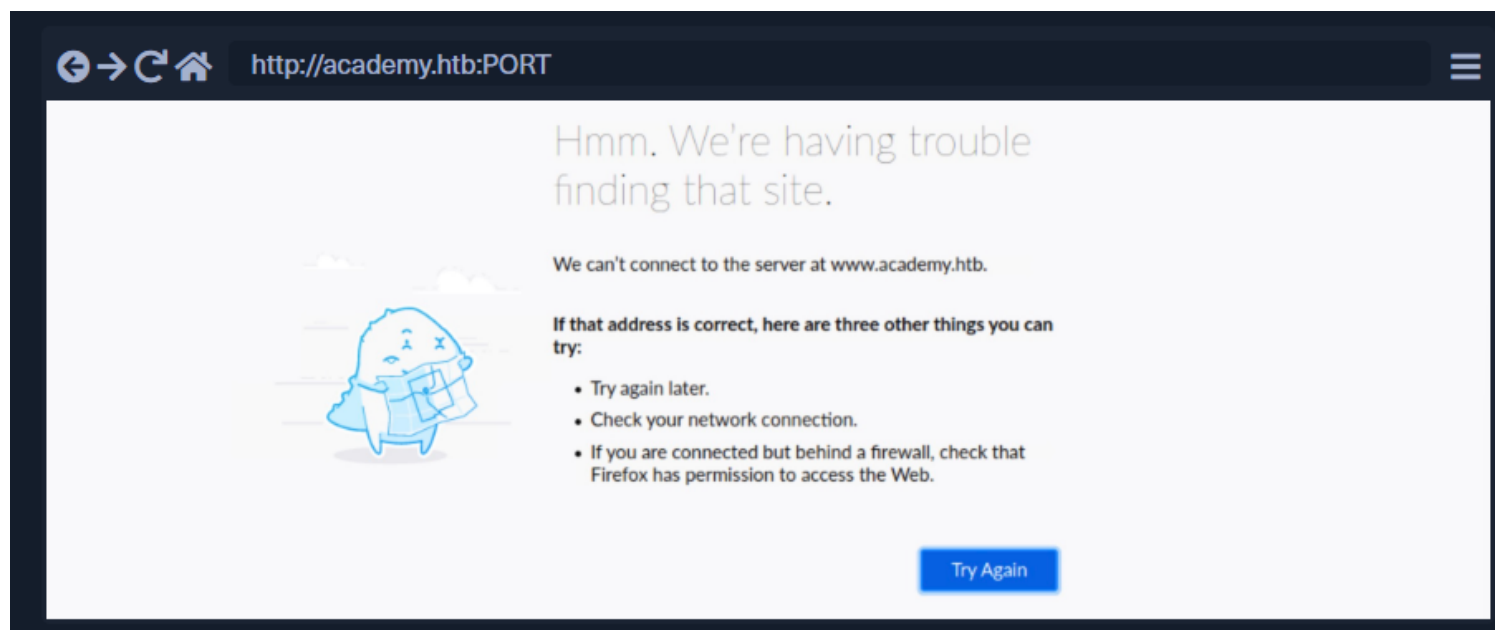
[Status: 200, Size: 21, Words: 1, Lines: 1, Duration: 294ms]
| URL | http://94.237.59.185:49214/forum/flag.php
* FUZZ: flag.php
```

2) got the flag

```
94.237.59.185:49214/f × +
← → ↻ 🏠 🔒 94.237.59.185:49214/forum/flag.php
🐧 Kali Linux 🌐 Kali Tools 📄 Kali Docs 🗣️ Kali Forums 🏠 Kali NetHunter 🔍 Exploit-DB 🔍 Google Hacking DB 🛡️ OffSec
HTB{fuzz1n6_7h3_w3b!}
```

## DNS Records

Once we accessed the page under `/blog`, we got a message saying **Admin panel moved to academy.htb**. If we visit the website in our browser, we get can't connect to the server at [www.academy.htb](http://www.academy.htb):

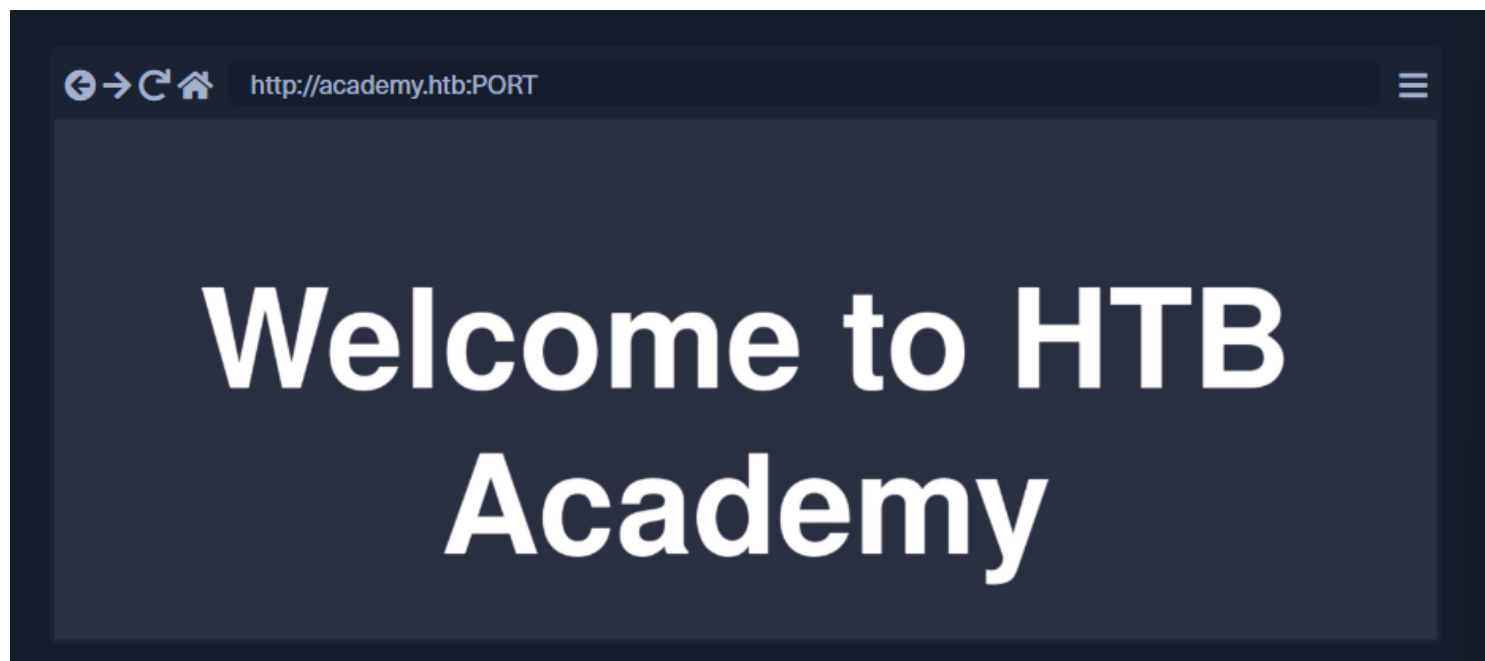


This is because the exercises we do are **not public websites** that can be accessed by anyone but **local websites within HTB**. Browsers only understand how to go to IPs, and **if we provide them with a URL**, they try to map the URL to an IP by looking into the local **/etc/hosts file** and the public DNS Domain Name System. If the URL is not in either, it would not know how to connect to it.

If we visit the IP directly, the browser goes to that IP directly and knows how to connect to it. But in this case, we tell it to go to **academy.htb**, so it looks into the local **/etc/hosts file** and doesn't find any mention of it. It asks the **public DNS** about it (such as Google's DNS 8.8.8.8) and does not find any mention of it, since it is not a public website, and eventually **fails to connect**. So, to connect to academy.htb, we would have to add it to **our /etc/hosts file**. We can achieve that with the following command:

```
Vigneswar@htb[/htb]$ sudo sh -c 'echo "SERVER_IP academy.htb" >> /etc/hosts'
```

Now we can visit the website (don't forget to add the PORT in the URL) and see that we can reach the website:



However, we get the same website we got when we [visit the IP directly](#), so academy.htb is the same domain we have been testing so far. We can verify that by visiting [/blog/index.php](#), and see that we can access the page.

When we run our tests on this IP, we did not find anything about admin or panels, even when we did a [full recursive scan](#) on our target. So, in this case, we start looking for [sub-domains under '\\*.academy.htb'](#) and see if we find anything, which is what we will attempt in the next section.

## ***Sub-domain Fuzzing***

Sub-domains

A sub-domain is any website [underlying another domain](#). For example, [https://photos.google.com](#) is the photos [sub-domain of google.com](#).

In this case, we are simply checking different websites to see if they exist by checking if they have a [public DNS record](#) that would redirect us to a working server IP. So, let's run a scan and see if we get any hits. Before we can start our scan, we need two things:

- A **wordlist**
- A **target**

Luckily for us, in the SecLists repo, there is a specific section for sub-domain wordlists, consisting



of common words usually used for sub-domains. We can find it in `/opt/useful/SecLists/Discovery/DNS/`. In our case, we would be using a shorter wordlist, which is `subdomains-top1million-5000.txt`. If we want to extend our scan, we can pick a larger list.

As for our target, we will use `inlanefreight.com` as our target and run our scan on it. Let us use ffuf and place the FUZZ keyword in the place of sub-domains, and see if we get any hits:

```
/opt/useful/SecLists/Discovery/DNS/subdomains-top1million-5000.txt:FUZZ -u https://FUZZ.inlanefreight.com/
```

```
-----  
[Status: 301, Size: 0, Words: 1, Lines: 1, Duration: 381ms]  
  * FUZZ: support  
  
[Status: 301, Size: 0, Words: 1, Lines: 1, Duration: 385ms]  
  * FUZZ: ns3  
  
[Status: 301, Size: 0, Words: 1, Lines: 1, Duration: 402ms]  
  * FUZZ: blog  
  
[Status: 301, Size: 0, Words: 1, Lines: 1, Duration: 180ms]  
  * FUZZ: my  
  
[Status: 200, Size: 22266, Words: 2903, Lines: 316, Duration: 589ms]  
  * FUZZ: www  
  
<...SNIP...>
```

We see that we do get a few hits back. Now, we can try running the same thing on `academy.htb` and see if we get any hits back:

```
Vigneswar@htb[/htb]$ ffuf -w /opt/useful/SecLists/Discovery/DNS/subdomains-top1million-5000.txt:FUZZ -u https://FUZZ.academy.htb/

v1.1.0-git

:: Method      : GET
:: URL         : https://FUZZ.academy.htb/
:: Wordlist    : FUZZ: /opt/useful/SecLists/Discovery/DNS/subdomains-top1million-5000.txt
:: Follow redirects : false
:: Calibration : false
:: Timeout     : 10
:: Threads    : 40
:: Matcher    : Response status: 200,204,301,302,307,401,403

:: Progress: [4997/4997] :: Job [1/1] :: 131 req/sec :: Duration: [0:00:38] :: Errors: 4997 ::
```

This means that there are **no public sub-domains** under academy.htb, as it does not have a public DNS record, as previously mentioned. Even though we did add academy.htb to our **/etc/hosts** file, we only added the main domain, so when ffuf is looking for other sub-domains, it will not find them in /etc/hosts, and will ask the public DNS, which obviously will not have them.

## Vhost Fuzzing

### Vhosts vs. Sub-domains

The key difference between VHosts and sub-domains is that a VHost is basically a 'sub-domain' served on the same server and has the same IP, such that a **single IP** could be serving **two or more different websites**.

**VHosts may or may not have public DNS records**

In many cases, many websites would actually have **sub-domains** that are **not public** and will not publish them in **public DNS records**, and hence if we visit them in a browser, we would fail to connect, as the public DNS would not know their IP.

Once again, if we use the sub-domain fuzzing, we would only be able to identify public sub-domains but will not identify any sub-domains that are not public.

This is where we utilize **VHosts Fuzzing** on an IP we already have. We will run a scan and test for scans on the same IP, and then we will be able to identify both public and non-public sub-domains and VHosts.

# Vhosts Fuzzing

To scan for VHosts, without manually adding the entire wordlist to our `/etc/hosts`, we will be fuzzing **HTTP headers**, specifically the **Host: header**. To do that, we can use the **-H flag** to specify a header and will use the FUZZ keyword within it, as follows:

```
Vigneswar@htb[/htb]$ ffuf -w /opt/useful/SecLists/Discovery/DNS/subdomains-top1million-5000.txt:FUZZ -u http://academy.htb:PORT/ -H 'Host: FUZZ.academy.htb'
```

```
      /\_/\   /\_/\   /\_/\
     /\_/\  /\_/\  /\_/\
    /\_/\ /\_/\ /\_/\ /\_/\
   /\_/\ /\_/\ /\_/\ /\_/\
  /\_/\ /\_/\ /\_/\ /\_/\
 /\_/\ /\_/\ /\_/\ /\_/\
/_/\ _/\_\/_\/_\/_\/_\/_/\
```

```
v1.1.0-git
```

---

```
:: Method           : GET
:: URL              : http://academy.htb:PORT/
:: Wordlist          : FUZZ:/opt/useful/SecLists/Discovery/DNS/subdomains-top1million-5000.txt
:: Header            : Host: FUZZ
:: Follow redirects  : false
:: Calibration       : false
:: Timeout           : 10
:: Threads           : 40
:: Matcher           : Response status: 200,204,301,302,307,401,403
```

---

```
mail2        [Status: 200, Size: 900, Words: 423, Lines: 56]
dns2         [Status: 200, Size: 900, Words: 423, Lines: 56]
ns3          [Status: 200, Size: 900, Words: 423, Lines: 56]
dns1         [Status: 200, Size: 900, Words: 423, Lines: 56]
lists        [Status: 200, Size: 900, Words: 423, Lines: 56]
webmail      [Status: 200, Size: 900, Words: 423, Lines: 56]
static       [Status: 200, Size: 900, Words: 423, Lines: 56]
web          [Status: 200, Size: 900, Words: 423, Lines: 56]
www1         [Status: 200, Size: 900, Words: 423, Lines: 56]
```

```
<...SNIP...>
```

We see that all words in the wordlist are returning 200 OK! This is expected, as we are **simply changing the header while visiting `http://academy.htb:PORT/`**. So, we know that we will always get 200 OK. However, if the VHost does exist and we send a correct one in the header, we should get a **different response size**, as in that case, we would be getting the page from that VHosts, which is likely to show a different page.

## Filtering Results

So far, we have not been using any filtering to our ffuf, and the results are automatically filtered by default by their HTTP code, which filters out code 404 NOT FOUND, and keeps the rest.

However, as we saw in our previous run of ffuf, we can get many responses with code 200. So, in this case, we will have to **filter the results based on another factor**, which we will learn in this section

## Filtering

**Ffuf** provides the option to match or filter out a specific HTTP code, response size, or amount of words. We can see that with **ffuf -h**:

```
Vigneswar@htb[/htb]$ ffuf -h
...SNIP...
MATCHER OPTIONS:
  -mc          Match HTTP status codes, or "all" for everything. (default: 20
  -ml          Match amount of lines in response
  -mr          Match regexp
  -ms          Match HTTP response size
  -mw          Match amount of words in response

FILTER OPTIONS:
  -fc          Filter HTTP status codes from response. Comma separated list o
  -fl          Filter by amount of lines in response. Comma separated list of
  -fr          Filter regexp
  -fs          Filter HTTP response size. Comma separated list of sizes and r
  -fw          Filter by amount of words in response. Comma separated list of
<...SNIP...>
```

In this case, we cannot use matching, as we don't know what the response size from other VHosts would be. We know the **response size of the incorrect results**, which, as seen from the test above, is 900, and we can filter it out with **-fs 900**. Now, let's repeat the same previous command, add the above flag, and see what we get:

```
Vigneswar@htb[/htb]$ ffuf -w /opt/useful/SecLists/Discovery/DNS/subdomains-t
```

```
/'___\  /'___\      /'___\
/\ \_/_/\ \ \_/_/  __  __  /\ \_/_/
\ \ ,__\ \ \ ,__\ \ \ \ \ \ \ ,__\
 \ \ \_/_ \ \ \_/_ \ \ \_/_ \ \ \_/_
  \ \ \_/_ \ \ \_/_ \ \ \_/_ \ \ \_/_
   \ \ \_/_ \ \ \_/_ \ \ \_/_ \ \ \_/_
```

v1.1.0-git

```
-----
:: Method          : GET
:: URL             : http://academy.htb:PORT/
:: Wordlist         : FUZZ: /opt/useful/SecLists/Discovery/DNS/subdomains-t
:: Header          : Host: FUZZ.academy.htb
:: Follow redirects : false
:: Calibration     : false
:: Timeout         : 10
:: Threads         : 40
:: Matcher         : Response status: 200,204,301,302,307,401,403
:: Filter          : Response size: 900
-----
```

<...SNIP...>

admin [Status: 200, Size: 0, Words: 1, Lines: 1]

:: Progress: [4997/4997] :: Job [1/1] :: 1249 req/sec :: Duration: [0:00:04]

We can verify that by visiting the page, and seeing if we can connect to it:

 https://admin.academy.htb:PORT/ 

We see that we can access the page, but we get an empty page, unlike what we got with academy.htb, therefore confirming this is **indeed a different VHost**. We can even visit <https://admin.academy.htb:PORT/blog/index.php>, and we will see that we would get a 404 PAGE NOT FOUND, confirming that we are now indeed on a different VHost.

## Exercise

```
(vigneswar@vigneswar)-[~]
$ ffuf -w SecLists/Discovery/DNS/subdomains-top1million-5000.txt -u http://94.237.53.115:31949 -H 'Host: FUZZ.academy.htb' -fs 986

      ^_/_/ ^_/_/ ^_/_/ ^_/_/
     /_/_/ /_/_/ /_/_/ /_/_/
    /_/_/ /_/_/ /_/_/ /_/_/
   /_/_/ /_/_/ /_/_/ /_/_/
  /_/_/ /_/_/ /_/_/ /_/_/
 /_/_/ /_/_/ /_/_/ /_/_/
/_/_/ /_/_/ /_/_/ /_/_/

v2.0.0-dev

:: Method      : GET
:: URL         : http://94.237.53.115:31949
:: Wordlist     : FUZZ: /home/vigneswar/SecLists/Discovery/DNS/subdomains-top1million-5000.txt
:: Header      : Host: FUZZ.academy.htb
:: Follow redirects : false
:: Calibration  : false
:: Timeout     : 10
:: Threads     : 40
:: Matcher     : Response status: 200,204,301,302,307,401,403,405,500
:: Filter      : Response size: 986

[Status: 200, Size: 0, Words: 1, Lines: 1, Duration: 3192ms]
* FUZZ: test

[Status: 200, Size: 0, Words: 1, Lines: 1, Duration: 4189ms]
* FUZZ: admin

:: Progress: [4989/4989] :: Job [1/1] :: 130 req/sec :: Duration: [0:00:39] :: Errors: 0 ::
```

## Parameter Fuzzing

**GET**

If we run a recursive ffuf scan on admin.academy.htb, we should find <http://admin.academy.htb:PORT/admin/admin.php>. If we try accessing this page, we see the following:



You don't have access to read the flag!

That indicates that there must be something that **identifies users** to verify whether they have access to read the flag. We did not login, nor do we have any cookie that can be verified at the backend. So, perhaps there is a key that we can pass to the page to read the flag. Such keys would usually be passed as a parameter, using either a **GET or a POST HTTP request**. This section will discuss how to fuzz for such parameters until we identify a parameter that can be accepted by the page.

**Tip:** Fuzzing parameters may expose unpublished parameters that are publicly accessible. Such parameters tend to be less tested and less secured, so it is important to test such parameters for the web vulnerabilities we discuss in other modules.

## GET Request Fuzzing

Similarly to how we have been fuzzing various parts of a website, we will use **ffuf** to enumerate parameters. Let us first start with fuzzing for **GET** requests, which are usually passed right after the URL, with a **?** symbol, like:

- `http://admin.academy.htb:PORT/admin/admin.php?param1=key`.

So, all we have to do is replace **param1** in the example above with **FUZZ** and rerun our scan. Before we can start, however, we must pick an appropriate wordlist. Once again, **SecLists** has just that in **/opt/useful/SecLists/Discovery/Web-Content/burp-parameter-names.txt**. With that, we can run our scan.

Once again, we will get many results back, so we will filter out the default response size we are getting.

```
Vigneswar@htb[/htb]$ ffuf -w /opt/useful/SecLists/Discovery/Web-Content/burp-para
```

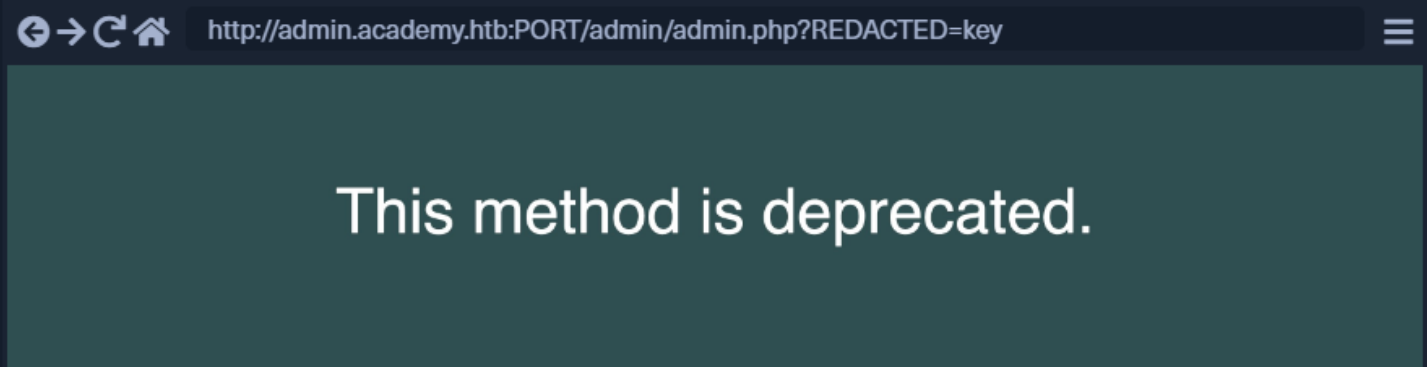
```
/'___\ /'___\ /'___\
/\___/\ /\___/\  __  __ /\___/\
\\ ,__\\ \\ ,__\\ \\ \\ \\ ,__\\
\\ \\___/ \\ \\___/ \\ \\___/ \\ \\___/
\\ \\_\\ \\ \\_\\ \\ \\___/ \\ \\_\\
\\_/_/ \\_/_/ \\_/_/ \\_/_/
```

```
v1.1.0-git
```

```
-----
:: Method      : GET
:: URL         : http://admin.academy.htb:PORT/admin/admin.php?FUZZ=key
:: Wordlist    : FUZZ: /opt/useful/SecLists/Discovery/Web-Content/burp-para
:: Follow redirects : false
:: Calibration : false
:: Timeout     : 10
:: Threads    : 40
:: Matcher    : Response status: 200,204,301,302,307,401,403
:: Filter     : Response size: xxx
-----
```



We do get a hit back. Let us try to visit the page and add this **GET** parameter, and see whether we can read the flag now:



As we can see, the only hit we got back has been **deprecated** and appears to be no longer in use.

## Exercise

```
(vigneswar@vigneswar)~$ ffuf -w SecLists/Discovery/Web-Content/burp-parameter-names.txt -u http://94.237.53.115:31949/admin/admin.php?FUZZ=key -H "Host: admin.academy.htb" -fs 798

  A'  A'  A'
 /  /  /
X  X  X
 \  \  \
  V  V  V

v2.0.0-dev

:: Method      : GET
:: URL         : http://94.237.53.115:31949/admin/admin.php?FUZZ=key
:: Wordlist    : FUZZ: /home/vigneswar/SecLists/Discovery/Web-Content/burp-parameter-names.txt
:: Header     : Host: admin.academy.htb
:: Follow redirects : false
:: Calibration : false
:: Timeout     : 10
:: Threads    : 40
:: Matcher    : Response status: 200,204,301,302,307,401,403,405,500
:: Filter     : Response size: 798

[Status: 200, Size: 783, Words: 221, Lines: 54, Duration: 236ms]
* FUZZ: user

:: Progress: [6453/6453] :: Job [1/1] :: 151 req/sec :: Duration: [0:00:51] :: Errors: 0 ::
```

***POST***

The main difference between POST requests and GET requests is that **POST requests are not passed with the URL** and cannot simply be appended after a ? symbol. POST requests are passed in the **data field within the HTTP request**. Check out the Web Requests module to learn more

about HTTP requests.

To fuzz the data field with ffuf, we can use the **-d flag**, as we saw previously in the output of ffuf -h. We also have to add **-X POST** to send POST requests.

Tip: In PHP, "POST" data "content-type" can only accept "application/x-www-form-urlencoded". So, we can set that in "ffuf" with "-H 'Content-Type: application/x-www-form-urlencoded'".

```
Vigneswar@htb[/htb]$ ffuf -w /opt/useful/SecLists/Discovery/Web-Content/burp-parameter-names.txt:FUZZ
```

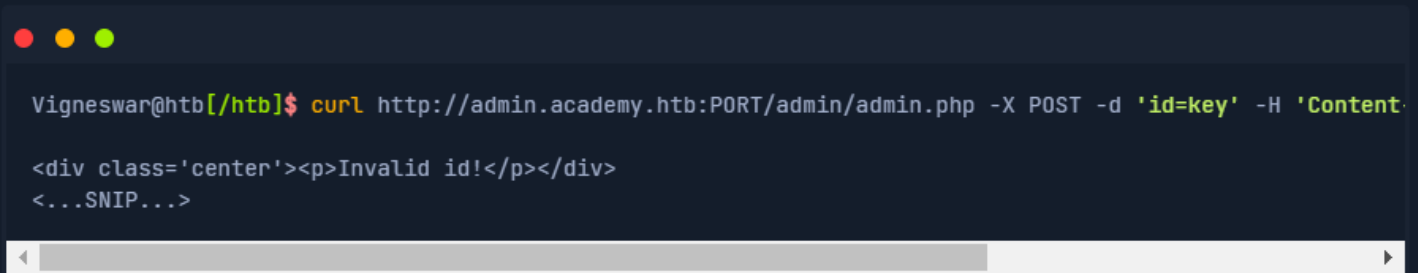
```
/'___\ /'___\ /'___\
/\___/\ \___/\ \___/\
\ \___\ \ \___\ \ \___\
\ \___\ \ \___\ \ \___\
\ \___\ \ \___\ \ \___\
\ \___\ \ \___\ \ \___\
\ \___\ \ \___\ \ \___\
```

v1.1.0-git

```
-----
:: Method      : POST
:: URL         : http://admin.academy.htb:PORT/admin/admin.php
:: Wordlist    : FUZZ: /opt/useful/SecLists/Discovery/Web-Content/burp-parameter-names.txt
:: Header     : Content-Type: application/x-www-form-urlencoded
:: Data       : FUZZ=key
:: Follow redirects : false
:: Calibration  : false
:: Timeout     : 10
:: Threads    : 40
:: Matcher     : Response status: 200,204,301,302,307,401,403
:: Filter     : Response size: xxx
-----
```

```
id [Status: xxx, Size: xxx, Words: xxx, Lines: xxx]
<...SNIP...>
```

As we can see this time, we got a couple of hits, the same one we got when fuzzing **GET** and another parameter, which is **id**. Let's see what we get if we send a **POST** request with the **id** parameter. We can do that with **curl**, as follows:

A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. The prompt is 'Vigneswar@htb[/htb]\$'. The command entered is 'curl http://admin.academy.htb:PORT/admin/admin.php -X POST -d 'id=key' -H 'Content-type: application/json''. The output is '<div class='center'><p>Invalid id!</p></div><...SNIP...>'.

```
Vigneswar@htb[/htb]$ curl http://admin.academy.htb:PORT/admin/admin.php -X POST -d 'id=key' -H 'Content-type: application/json'
<div class='center'><p>Invalid id!</p></div>
<...SNIP...>
```

As we can see, the message now says **Invalid id!**.

## Value

After fuzzing a working parameter, we now have to fuzz the **correct value** that would return the **flag content** we need. This section will discuss fuzzing for parameter values, which should be fairly similar to fuzzing for parameters, once we develop our wordlist.

### Custom Wordlist

When it comes to fuzzing parameter values, we may not always find a pre-made wordlist that would work for us, as each parameter would expect a certain type of value.

For some parameters, like usernames, we can find a pre-made wordlist for potential usernames, or we may create our own based on users that may potentially be using the website. For such cases, we can look for various wordlists under the **secLists** directory and try to find one that may contain values matching the parameter we are targeting. In other cases, like custom parameters, we may have to develop our own wordlist. In this case, we can guess that the **id** parameter can accept a number input of some sort. These ids can be in a custom format, or can be sequential, like from 1-1000 or 1-1000000, and so on. We'll start with a wordlist containing all numbers from 1-1000.

There are many ways to create this wordlist, from manually typing the IDs in a file, or scripting it using Bash or Python. The simplest way is to use the following command in Bash that writes all numbers from 1-1000 to a file:

```
[!bash!]$ for i in $(seq 1 1000); do echo $i >> ids.txt; done
```

Once we run our command, we should have our wordlist ready:

```
[!bash!]$ cat ids.txt
```

1  
2  
3  
4  
5  
6  
<...SNIP...>

Now we can move on to fuzzing for values.

## Value Fuzzing

Our command should be fairly similar to the `POST` command we used to fuzz for parameters, but our `FUZZ` keyword should be put where the parameter value would be, and we will use the `ids.txt` wordlist we just created, as follows:

```
[!bash!]$ ffuf -w ids.txt:FUZZ -u http://admin.academy.htb:PORT/admin/admin.php -X POST -d 'id=FUZZ' -H
```

```
/'___\  /'___\      /'___\  
/\ \_/\ /\ \_/\  __  __ /\ \_/  
\ \ ,__\ \ \ ,__\ \ \ \ \ \ ,__\  
\ \ \_/\ \ \ \_/\ \ \ \_/\ \ \_/  
\ \ \  \ \ \  \ \ \_\_\_\_/\ \ \\  
 \_/\   \_/\   \_/\_\_\_\_/\_/\
```

v1.0.2

## Exercise

1) Found the parameter

```
(vigneswar@vigneswar)-[~]
$ ffuf -w SecLists/Discovery/Web-Content/burp-parameter-names.txt -u http://94.237.53.115:31949/admin/admin.php -X POST -H "Content-Type: application/x-www-form-urlencoded" -H "Host: admin.academy.htb" -d FUZZ -fs 798

v2.0.0-dev

:: Method      : POST
:: URL         : http://94.237.53.115:31949/admin/admin.php
:: Wordlist     : FUZZ: /home/vigneswar/SecLists/Discovery/Web-Content/burp-parameter-names.txt
:: Header      : Content-Type: application/x-www-form-urlencoded
:: Header      : Host: admin.academy.htb
:: Data        : FUZZ
:: Follow redirects : false
:: Calibration  : false
:: Timeout     : 10
:: Threads     : 40
:: Matcher     : Response status: 200,204,301,302,307,401,403,405,500
:: Filter      : Response size: 798

[Status: 200, Size: 768, Words: 219, Lines: 54, Duration: 250ms]
* FUZZ: id

[Status: 200, Size: 783, Words: 221, Lines: 54, Duration: 231ms]
* FUZZ: user

:: Progress: [6453/6453] :: Job [1/1] :: 125 req/sec :: Duration: [0:00:57] :: Errors: 0 ::
```

## 2) Created wordslist

```
(vigneswar@vigneswar)-[~]
$ for i in {0..10000}; do echo $i >> ids.txt; done

(vigneswar@vigneswar)-[~]
$
```

## 3) Found the id

```
(vigneswar@vigneswar)-[~]
$ ffuf -w ids.txt -u http://94.237.53.115:31949/admin/admin.php -X POST -H "Content-Type: application/x-www-form-urlencoded" -H "Host: admin.academy.htb" -d "id=FUZZ" -fs 768

v2.0.0-dev

:: Method      : POST
:: URL         : http://94.237.53.115:31949/admin/admin.php
:: Wordlist     : FUZZ: /home/vigneswar/ids.txt
:: Header      : Content-Type: application/x-www-form-urlencoded
:: Header      : Host: admin.academy.htb
:: Data        : id=FUZZ
:: Follow redirects : false
:: Calibration  : false
:: Timeout     : 10
:: Threads     : 40
:: Matcher     : Response status: 200,204,301,302,307,401,403,405,500
:: Filter      : Response size: 768

[Status: 200, Size: 787, Words: 218, Lines: 54, Duration: 228ms]
* FUZZ: 73
```

## 4) Got the flag


```
(vigneswar@vigneswar)-[~]
$ curl http://94.237.53.115:31949/admin/admin.php -X POST -H "Content-Type: application/x-www-form-urlencoded" -H "Host: admin.academy.htb" -d "id=73"
<div class='center'><p>HTB{p4r4m373r_fuzz1n6_15_k3y!}</p></div>
<html>
<!DOCTYPE html>
```

You are given an online academy's IP address but have no further information about their website. As the first step of conducting a Penetration Test, you are expected to locate all pages and domains linked to their IP to enumerate the IP and domains properly.

Finally, you should do some fuzzing on pages you identify to see if any of them has any parameters that can be interacted with. If you do find active parameters, see if you can retrieve any data from them.

## 1) Found the subdomains

```
(vigneswar@vigneswar)-[~]
$ ffuf -w SecLists/Discovery/DNS/subdomains-top1million-5000.txt -u http://94.237.48.48:50040 -H 'Host: FUZZ.academy.htb' -fs 985 -t 100
```



```
v2.0.0-dev

:: Method      : GET
:: URL         : http://94.237.48.48:50040
:: Wordlist     : FUZZ: /home/vigneswar/SecLists/Discovery/DNS/subdomains-top1million-5000.txt
:: Header      : Host: FUZZ.academy.htb
:: Follow redirects : false
:: Calibration  : false
:: Timeout     : 10
:: Threads     : 100
:: Matcher     : Response status: 200,204,301,302,307,401,403,405,500
:: Filter      : Response size: 985

[Status: 200, Size: 0, Words: 1, Lines: 1, Duration: 300ms]
* FUZZ: test

[Status: 200, Size: 0, Words: 1, Lines: 1, Duration: 242ms]
* FUZZ: archive

[Status: 200, Size: 0, Words: 1, Lines: 1, Duration: 271ms]
* FUZZ: faculty

:: Progress: [4989/4989] :: Job [1/1] :: 57 req/sec :: Duration: [0:00:37] :: Errors: 0 ::
```

## 2) Found extensions

```

(vigneswar@vigneswar)-[~]
$ ffuf -w SecLists/Discovery/Web-Content/web-extensions.txt:FUZZ -u http://94.237.48.48:50040/indexFUZZ -H 'Host: faculty.academy.htb'

v2.0.0-dev

:: Method      : GET
:: URL         : http://94.237.48.48:50040/indexFUZZ
:: Wordlist     : FUZZ: /home/vigneswar/SecLists/Discovery/Web-Content/web-extensions.txt
:: Header      : Host: faculty.academy.htb
:: Follow redirects : false
:: Calibration : false
:: Timeout     : 10
:: Threads     : 40
:: Matcher     : Response status: 200,204,301,302,307,401,403,405,500

[Status: 200, Size: 0, Words: 1, Lines: 1, Duration: 341ms]
* FUZZ: .php7

[Status: 403, Size: 284, Words: 20, Lines: 10, Duration: 339ms]
* FUZZ: .phps

[Status: 200, Size: 0, Words: 1, Lines: 1, Duration: 338ms]
* FUZZ: .php

:: Progress: [40/40] :: Job [1/1] :: 0 req/sec :: Duration: [0:00:00] :: Errors: 0 ::

```

### 3) Enumerated for directories

```

(vigneswar@vigneswar)-[~]
$ ffuf -w SecLists/Discovery/Web-Content/directory-list-2.3-small.txt -u http://faculty.academy.htb:50040/FUZZ -ic -v

v2.0.0-dev

:: Method      : GET
:: URL         : http://faculty.academy.htb:50040/FUZZ
:: Wordlist     : FUZZ: /home/vigneswar/SecLists/Discovery/Web-Content/directory-list-2.3-small.txt
:: Follow redirects : false
:: Calibration : false
:: Timeout     : 10
:: Threads     : 40
:: Matcher     : Response status: 200,204,301,302,307,401,403,405,500

[Status: 200, Size: 0, Words: 1, Lines: 1, Duration: 231ms]
| URL | http://faculty.academy.htb:50040/
* FUZZ:

[Status: 301, Size: 337, Words: 20, Lines: 10, Duration: 241ms]
| URL | http://faculty.academy.htb:50040/courses
| → | http://faculty.academy.htb:50040/courses/
* FUZZ: courses

[Status: 200, Size: 0, Words: 1, Lines: 1, Duration: 283ms]
| URL | http://faculty.academy.htb:50040/
* FUZZ:

:: Progress: [87651/87651] :: Job [1/1] :: 131 req/sec :: Duration: [0:13:14] :: Errors: 0 ::

```

```

http://faculty.academy.htb:50040/courses/linux-security.php7/

```

## 4) Found the parameters

```
(vigneswar@vigneswar)-[~]
$ ffuf -w SecLists/Discovery/Web-Content/burp-parameter-names.txt:FUZZ -u http://faculty.academy.htb:48085/courses/linux-security.php7 -X POST -d 'FUZZ=key' -H 'Content-Type: application/x-www-form-urlencoded' -fs 774

KALI LINUX
"the quieter you become, the more you are able to hear"

v2.0.0-dev

:: Method      : POST
:: URL         : http://faculty.academy.htb:48085/courses/linux-security.php7
:: Wordlist     : FUZZ: /home/vigneswar/SecLists/Discovery/Web-Content/burp-parameter-names.txt
:: Header      : Content-Type: application/x-www-form-urlencoded
:: Data        : FUZZ=key
:: Follow redirects : false
:: Calibration : false
:: Timeout     : 10
:: Threads     : 40
:: Matcher     : Response status: 200,204,301,302,307,401,403,405,500
:: Filter      : Response size: 774

[Status: 200, Size: 780, Words: 223, Lines: 53, Duration: 227ms]
* FUZZ: user

[Status: 200, Size: 781, Words: 223, Lines: 53, Duration: 205ms]
* FUZZ: username

:: Progress: [6453/6453] :: Job [1/1] :: 176 req/sec :: Duration: [0:00:40] :: Errors: 0 ::
```

## 5) Found a working value

```
(vigneswar@vigneswar)-[~]
$ ffuf -w SecLists/Usernames/Names/names.txt -u http://faculty.academy.htb:48085/courses/linux-security.php7 -X POST -H "Content-Type: application/x-www-form-urlencoded" -d "username=FUZZ&user=FUZZ" -fs 781

KALI LINUX
"the quieter you become, the more you are able to hear"

v2.0.0-dev

:: Method      : POST
:: URL         : http://faculty.academy.htb:48085/courses/linux-security.php7
:: Wordlist     : FUZZ: /home/vigneswar/SecLists/Usernames/Names/names.txt
:: Header      : Content-Type: application/x-www-form-urlencoded
:: Data        : username=FUZZ&user=FUZZ
:: Follow redirects : false
:: Calibration : false
:: Timeout     : 10
:: Threads     : 40
:: Matcher     : Response status: 200,204,301,302,307,401,403,405,500
:: Filter      : Response size: 781

[Status: 200, Size: 773, Words: 218, Lines: 53, Duration: 195ms]
* FUZZ: harry
```

## 6) Got the flag

```
(vigneswar@vigneswar)-[~]
$ curl http://faculty.academy.htb:48085/courses/linux-security.php7 -X POST -H "Content-Type: application/x-www-form-urlencoded" -d "username=harry&user=harry"
<div class='center'><p>HTB{w3b_fuzz1n6_m4573r}</p></div>
<html>
<!DOCTYPE html>

<head>
<title>HTB Academy</title>
<style>
*,
html {
margin: 0;
padding: 0;
border: 0;
}

html {
width: 100%;
height: 100%;
}

body {
width: 100%;
height: 100%;
position: relative;
background-color: #151D28;
```