Basic Fuzzing

Directory Fuzzing

Now that we understand the concept of Web Fuzzing and know our wordlist, we should be ready to start using `ffuf` to find website directories.

Ffuf

`Ffuf` is pre-installed on your PwnBox instance. If you want to use it on your own machine, you can either use "`apt install ffuf -y`" or download it and use it from its GitHub Repo. As a new user of this tool, we will start by issuing the `ffuf -h` command to see how the tools can be used:

mayala@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. (default: false) -recursion-depth Maximum recursion depth. (default: 0) -u Target URL ...SNIP... MATCHER OPTIONS: -mc Match HTTP status codes, or "all" for everything. (default: 200,204,301,302,307,401,403) -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:KEYWORD' OUTPUT OPTIONS: -o Write output to file ...SNIP... EXAMPLE USAGE: Fuzz file paths from wordlist.txt, match all responses but filter out those with content-size 42. Colored, verbose output. ffuf -w wordlist.txt -u https://example.org/FUZZ -mc all -fs 42 -c -v ...SNIP...
```

As we can see, the `help` output is quite large, so we only kept the options that may become relevant for us in this module.

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 wordlist to a keyword 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:

```
mayala@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:

```
mayala@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:

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

Page Fuzzing

We now understand the basic use of `ffuf` through the utilization of wordlists and keywords. Next, we will learn how to locate pages.

Note: We can spawn the same target from the previous section for this section's examples as well.

Extension Fuzzing

In the previous section, we found that we had access to `/blog`, but the directory returned an empty page, and we cannot manually locate any links or pages. So, we will once again utilize web fuzzing to see if the directory contains any hidden pages. However, 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:

```
mayala@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.

Note: The wordlist we chose already contains a dot (.), so we will not have to add the dot after "index" in our fuzzing.

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

```
mayala@htb[/htb]$ ffuf -w /opt/useful/SecLists/Discovery/Web-Content/web-
extensions.txt:FUZZ -u http://SERVER_IP:PORT/blog/indexFUZZ /'___\ /'___\ /'___\
/\ \__/ /\ \__/ __ __ /\ \__/ \ \ ,__\\ \ ,__\/\ \/\ \ \ \ ,__\ \ \ \_/ \ \ \_/\
\ \_\ \ \ \ \_/ \ \_\ \ \_\ \ \____/ \ \_\ \/_/ \/_/ \/___/ \/_/ v1.1.0-git
________________________________________________ :: Method : GET :: URL :
http://SERVER_IP:PORT/blog/indexFUZZ :: Wordlist : FUZZ:
/opt/useful/SecLists/Discovery/Web-Content/web-extensions.txt :: Follow
redirects : false :: Calibration : false :: Timeout : 10 :: Threads : 5 ::
Matcher : Response status: 200,204,301,302,307,401,403
________________________________________________ .php [Status: 200, Size: 0,
Words: 1, Lines: 1] .phps [Status: 403, Size: 283, Words: 20, Lines: 10] ::
Progress: [39/39] :: Job [1/1] :: 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:

```
mayala@htb[/htb]$ ffuf -w /opt/useful/SecLists/Discovery/Web-Content/directory-
list-2.3-small.txt:FUZZ -u http://SERVER_IP:PORT/blog/FUZZ.php /'___\ /'___\
/'___\ /\ \__/ /\ \__/ __ __ /\ \__/ \ \ ,__\\ \ ,__\/\ \/\ \ \ \ ,__\ \ \ \_/ \
\ \_/\ \ \_\ \ \ \ \_/ \ \_\ \ \_\ \ \____/ \ \_\ \/_/ \/_/ \/___/ \/_/ v1.1.0-
git ________________________________________________ :: Method : GET :: URL :
http://SERVER_IP:PORT/blog/FUZZ.php :: 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
________________________________________________ index [Status: 200, Size: 0,
Words: 1, Lines: 1] REDACTED [Status: 200, Size: 465, Words: 42, Lines: 15] ::
Progress: [87651/87651] :: 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:

Admin panel moved

Recursive Fuzzing

So far, we have been fuzzing for directories, then going under these directories, and then fuzzing for files. However, 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`

Note: we can still use `.php` as our page extension, as these extensions are usually site-wide.

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:

```
mayala@htb[/htb]$ ffuf -w /opt/useful/SecLists/Discovery/Web-Content/directory-
list-2.3-small.txt:FUZZ -u http://SERVER_IP:PORT/FUZZ -recursion -recursion-
depth 1 -e .php -v /'___\ /'___\ /'___\ /\ \__/ /\ \__/ __ __ /\ \__/ \ \ ,__\\
\ ,__\/\ \/\ \ \ \ ,__\ \ \ \_/ \ \ \_/\ \ \_\ \ \ \ \_/ \ \_\ \ \_\ \ \____/ \
\_\ \/_/ \/_/ \/___/ \/_/ 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 ::
Extensions : .php :: Follow redirects : false :: Calibration : false :: Timeout
: 10 :: Threads : 40 :: Matcher : Response status: 200,204,301,302,307,401,403
________________________________________________ [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/index.php * FUZZ: index.php
[Status: 301, Size: 326, Words: 20, Lines: 10] | URL |
http://SERVER_IP:PORT/blog | --> | http://SERVER_IP:PORT/blog/ * FUZZ: blog
<...SNIP...> [Status: 200, Size: 0, Words: 1, Lines: 1] | URL |
http://SERVER_IP:PORT/blog/index.php * 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.