🕵️ Content Identifiers

Intro

Once the passive recognition is done, expanding the attack surface by identifying subdomains and we have done a semi-passive recognition, identifying technologies of the website or the subdomains, what we have left to do is to do an active **recognition.**

It is possible that with the previous phases we have found some things, having identified the subdomains that we could attack individually and it is also possible that we have identified technologies that are vulnerable to attacks from those applications and focus on the *exploitation phase.*

But it is also possible that due to circumstance we have not found anything regarding subdomains and technologies, so we have to go further.

In a web application there are sections that are easily accessible to the public. But also, there are sections that are no longer referenced on the website, either due to obsolescence or that have simply been forgotten, but **that are still operating there.**

This is when the techniques come in that are more intrusive and that will Brute Force the resources of those applications. And for this there are several tools

⚙️ Tools to identify content

🗄️🔫 Dirbuster

Intro

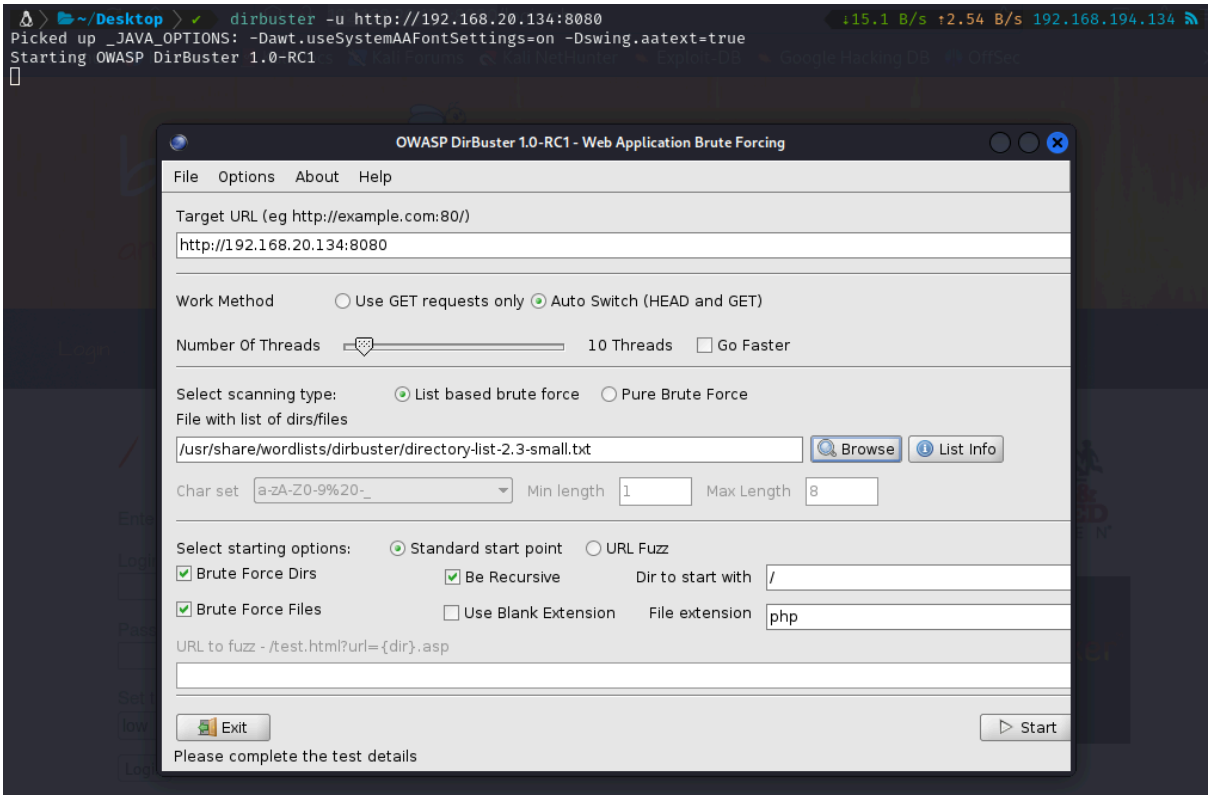
This tool comes by default in 🐉Kali. This works in the following way, we are going to pass it to the *url* of the *domain or subdomain* and this will perform brute force on a series of sections that a dictionary has registered and uploaded. These being the ones found in the directory **usr/share/wordlist/dirbuster.**

Dirbuster in action

To put to work **working** just use the following command

dirbuster -u http://target.website

If we do not indicate a default dictionary, it will not select through the graphical interface which dictionary it is going to use. This would look like this



From Browser we can indicate the dictionary that we want to use, which is located in the directory that we previously specified. In the image we already placed one.

And the result would be this:

Now here we can see the different directories that it has not found **working** And when we stop the scanning in the graphical interface we can immediately generate a report by clicking on **report** and then place it in the directory we want on our attacking machine, click on **Generate Report** and, done!

🏃‍♂️🔫 Gobuster

Intro

This is a tool just like Dirbuster, it will allow us to brute force the directories and files of the web application that we have as our target. This also allows you to perform brute force on *subdomains, virtual host y Buckets de amazon*.

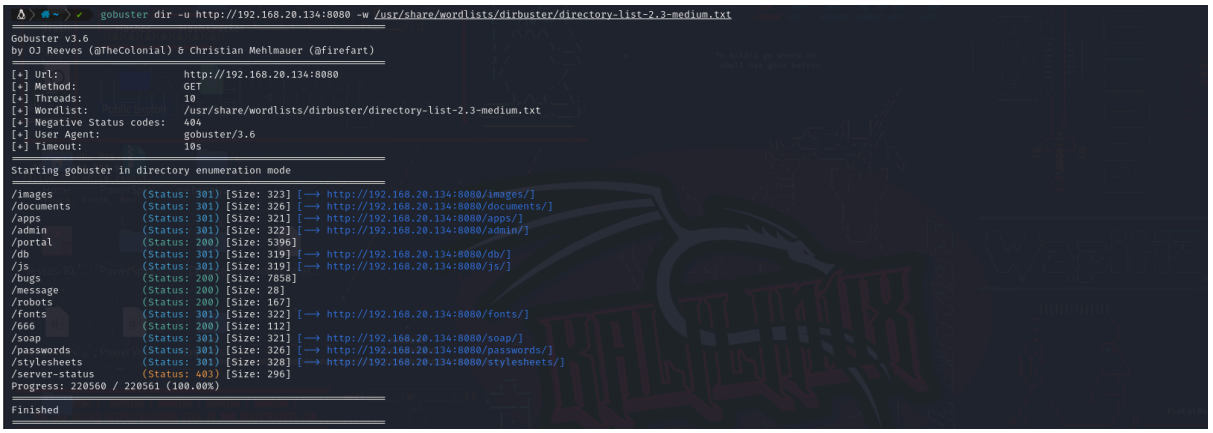
Go Buster comes pre-installed in 🐉 Kali and 🦜 Parrot, but in case we don't have it installed we just have to do a **sudo apt install gobuster,** making sure we have all the repositories updated, and we will have it available

Gobuster in action 🏃‍♂️

Go buster is useful for many things but focusing on our issue **gobuster** used with the command:

gobuster dir -u http://target.website -w /usr/share/wordlists/wordlisttouse.txt

And the result would be something like this:



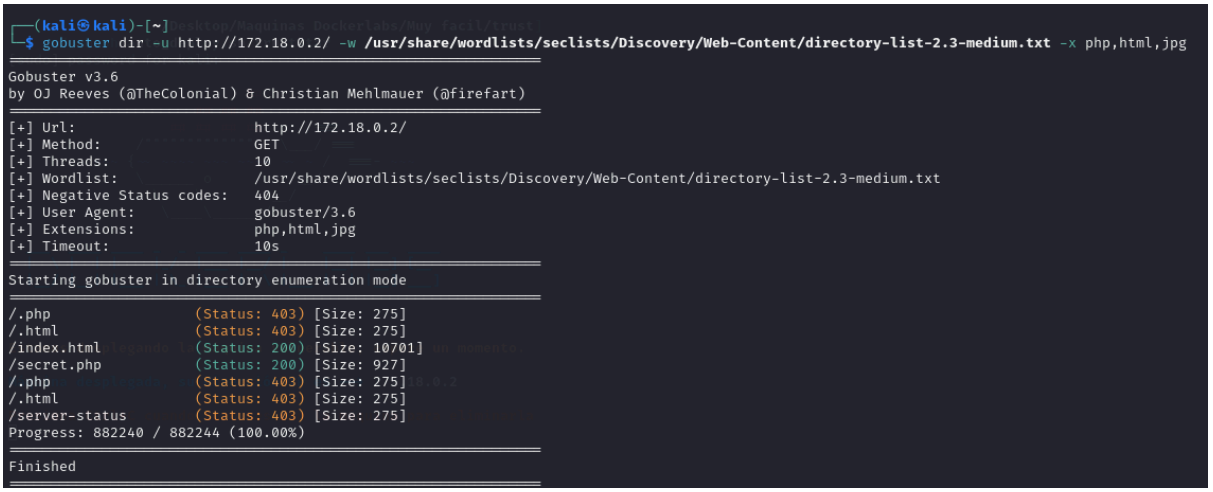
We can open all these directories in the browser from the terminal. Once these directories are located, we could start searching and seeing interesting things, **from gathering more information** or even, **compromise a website** with the *information* that *is exposed* in one of those directories without having to carry out an attack.

🔎🤖 Searches with technologies

With gobuster we can also do more specific searches using the command **-x** and passing the technologies that we have identified in that web application. The command would be something like this

gobuster dir -u http://target.website -w /usr/share/wordlists/wordlisttouse.txt -x php,html,etc

And the result on the screen would be something like this:

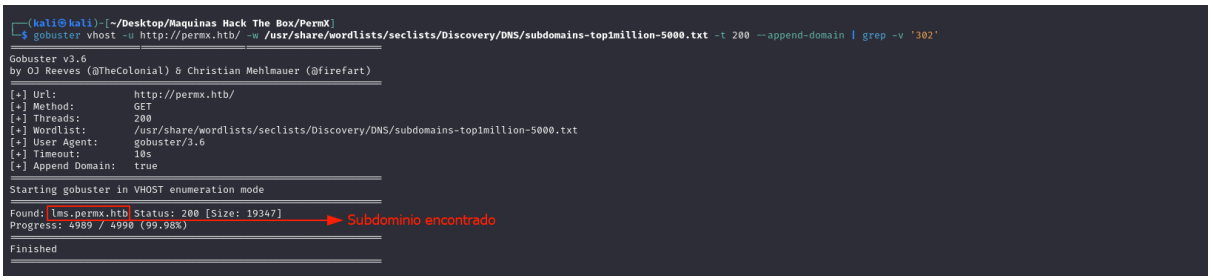


👥Subdomain identification

With **gobuster** We can also identify subdomains on a target website, which would increase the attack surface on said target.

We can do this with the command:

gobuster vhost -u http://target.website/ -w /usr/share/wordlists/wordlisttouse.txt -t 200 --append-domain | grip -v '302'



Something important to mention is that in the example shown the parameter is used **vhost -in** because it is a virtual machine. If it is a real host we will use the parameter **dns -d**

📋🔫Dirb

This is another tool for identifying content and directory browsing that will work via Terminal. This tool is much simpler than **Dibuster** and **Gobuster** so its time is going to be more limited with respect to what we can do with it.

Comes pre-installed on systems **Kali Linux** 🐉 and **Parrot** 🦜

Dirb in action

As we already mentioned, using Dirb is going to be quite simple, we just have to call the tool followed by the URL we want to scan.

dirb http://target.website

```
┌──(root㉿INE)-[~]
└─# dirb http://target2.ine.local/

-----------------
DIRB v2.22
By The Dark Raver
-----------------

START_TIME: Mon Dec 23 18:20:07 2024
URL_BASE: http://target2.ine.local/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

-----------------

GENERATED WORDS: 4612

---- Scanning URL: http://target2.ine.local/ ----
+ http://target2.ine.local/index.php (CODE:301|SIZE:0)
+ http://target2.ine.local/server-status (CODE:403|SIZE:282)
==> DIRECTORY: http://target2.ine.local/wp-admin/
==> DIRECTORY: http://target2.ine.local/wp-content/
==> DIRECTORY: http://target2.ine.local/wp-includes/
+ http://target2.ine.local/xmlrpc.php (CODE:405|SIZE:42)
```

As we can see in the previous example, unlike **Dirbuster** and **Gobuster** here we did not have to place a wordlist to do the scanning. And it is that **dirb** will use a default wordlist, specifically **common.txt,** which, as its name indicates, is used to search for the most common directories that usually exist in a web application.

But of course this would not greatly limit what we can do with this tool in what we can do to find directories, what can we do then? Well, we can use other wordlists by adding the dictionary path to the previous command.

dirb http://target.website /route/of/wordlist/wordlist.txt

```
┌──(root㉿INE)-[~]
└─# dirb http://target2.ine.local/wp-content/plugins/ /usr/share/nmap/nselib/data/wp-plugins.lst

-----------------
DIRB v2.22
By The Dark Raver
-----------------

START_TIME: Mon Dec 23 18:23:15 2024
URL_BASE: http://target2.ine.local/wp-content/plugins/
WORDLIST_FILES: /usr/share/nmap/nselib/data/wp-plugins.lst

-----------------

GENERATED WORDS: 50546

---- Scanning URL: http://target2.ine.local/wp-content/plugins/ ----
==> DIRECTORY: http://target2.ine.local/wp-content/plugins/akismet/
==> DIRECTORY: http://target2.ine.local/wp-content/plugins/duplicator/

---- Entering directory: http://target2.ine.local/wp-content/plugins/akismet/ ----

---- Entering directory: http://target2.ine.local/wp-content/plugins/duplicator/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
    (Use mode '-w' if you want to scan it anyway)

-----------------
END_TIME: Mon Dec 23 18:23:50 2024
DOWNLOADED: 101092 - FOUND: 0
```

Wordlist agregada

📝Additional notes:

Note I: Also, something that won't help a lot with these tools is to have the dictionary **SecList** installed. This is the most influential and important dictionary in the world of Hacking, due to the number of dictionaries it has. This is installed with a simple **sudo apt install seclists,** or we can download it directly from the GitHub repository.

Note I: we can see all the dictionaries we have in Kali, even **SecList**, with the command **wordlists.**