Directory search

Overview

In this module you will be shown:

1. What a directory search tool is used for
2. Why it is useful to have multiple directory search tools
3. The major directory search tools
4. Alternate methods for directory searching

It is always useful to understand how to use multiple tools even if there purpose is the same or similar. This provide redundancy in your pen-testing arsenal. There are three main automated tools that are used for Directory searching:

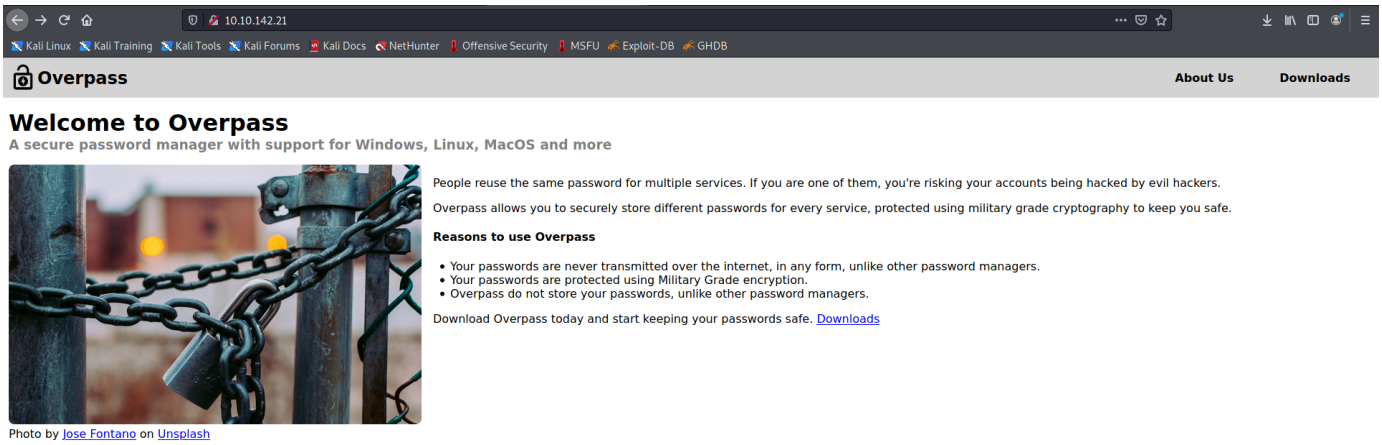
1. Gobuster
2. Dirbuster
3. Dirsearch

Some of these tools come pre-loaded on kali linux (I am working with kali 2020.4 currently). Others require an install prior to use, in this case we will go through installing `dirsearch`. We will also cover using each of these tools. But before that, what is the purpose of a directory searching tool and why are they useful.

> NOTE: If you want to follow along with any of the content I am using the `Overpass` machine from 'TryHackMe'. This machine is free to use.

What are directory search tools used for ?

When looking at a website of any description, more often then not there are directories you can see and directories you can not. For example, on the `Overpass` website there are a few tabs that you can click on which take you to particular sites, or directories hosted by the server.



Just by clicking the tabs we can find:

http://10.10.142.21/aboutus
http://10.10.142.21/downloads

These directories are what the website owner what the user to see. However, often for a website to function correctly it needs to access other files from a range of directories. The first place you should look is `robots.txt`.

> **Robots.txt**
>
> This is a bit of a tangent but useful non the less. A lot of websites have a file called `robots.txt` located at 'http://website/robots.txt'. The purpose of this file is to tell search engine crawlers which pages the crawler can or cant access from the site. The main purpose of this file is to prevent excess requests coming to the site. For a pen-tester it can provide potential hidden directories to explore. But this file will not always exist and by no means does it contain everything. An example would be:
>
> Kali Linux Kali Training Kali Tools
>
> ```
> User-agent: *
> Disallow: /wp-admin/
> Allow: /wp-admin/admin-ajax.php
> ```

In the case of `Overpass` the image of the lock needs to be located somewhere, but there is no link available to take the general user then. This is where an automated directory search tool can become extremely useful.

The tool will take a list of default and common words from a wordlist and check to see if the nominated address has those directories or files available. Once it checks it will provide:

1. Files/folders that can be seen but not accessed (400 response)
2. Files/folders that re-direct (300 response)
3. Files/folders that can be accessed (200 response)

Essentially, it brute-forces the directory names to see if it gets a hit.

Once a pen-tester knows what folders are hidden in the background they can be explored for additional information as part of enumeration.

Why have more that one tool

There are several reasons to have multiple directory searching tools. The most simple is preference, for a long time I personally used `dirbuster` as my 'go-to' tool as it has a gui, you can change the thread count whilst the scan is underway and its nice any user friendly. However, there are times when I want to do a simple and quick search, in this case `gobuster` might prove more useful. Whilst `gobuster` may not be as thorough it is simple and quick.

Finally, having access to multiple tools provides redundancy and options which is important to any attack. You may for some reason not have access to a gui that allows you to use `dirbuster` or you may be on a machine that does not have `dirsearch` installed and no access to the wider internet. The point is its good to try and get comfortable with a range of tools.

The Common tools

The three most common tools used for directory searching are Gobuster, Dirsearch and Dirbuster. We will go through each of these now in detail.

Gobuster

If you have already completed the lesson on installing `autorecon` you will already have `gobuster` installed. Otherwise it can be installed from the repository using:

`sudo apt-get install gobuster`

How is it Used

The basic `-h` flag will get you the following:

```
┌──(kali㉿kali)-[~]
└─$ gobuster -h
Usage:
  gobuster [command]

Available Commands:
  dir         Uses directory/file brutceforcing mode
  dns         Uses DNS subdomain bruteforcing mode
  help        Help about any command
  vhost       Uses VHOST bruteforcing mode

Flags:
  -h, --help              help for gobuster
  -z, --noprogress        Don't display progress
  -o, --output string     Output file to write results to (defaults to stdout)
  -q, --quiet             Don't print the banner and other noise
  -t, --threads int       Number of concurrent threads (default 10)
  -v, --verbose           Verbose output (errors)
  -w, --wordlist string   Path to the wordlist

Use "gobuster [command] --help" for more information about a command.
```

But to look at the help function for `dir` we can use the command `gobuster dir --help`. From this we can see a full range of flags we can use.

```
┌──(kali㉿kali)-[~]
└─$ gobuster dir --help
Uses directory/file brutceforcing mode

Usage:
  gobuster dir [flags]

Flags:
  -f, --addslash                     Apped / to each request
  -c, --cookies string               Cookies to use for the requests
  -e, --expanded                     Expanded mode, print full URLs
  -x, --extensions string            File extension(s) to search for
  -r, --followredirect               Follow redirects
  -H, --headers stringArray          Specify HTTP headers, -H 'Header1: val1' -H 'Header2: val2'
  -h, --help                         help for dir
  -l, --includelength                Include the length of the body in the output
  -k, --insecuressl                  Skip SSL certificate verification
  -n, --nostatus                     Don't print status codes
  -P, --password string              Password for Basic Auth
  -p, --proxy string                 Proxy to use for requests [http(s)://host:port]
  -s, --statuscodes string           Positive status codes (will be overwritten with statuscodesblacklist if set) (default "200,204,301,302,307,401,403")
  -b, --statuscodesblacklist string  Negative status codes (will override statuscodes if set)
      --timeout duration             HTTP Timeout (default 10s)
  -u, --url string                   The target URL
  -a, --useragent string             Set the User-Agent string (default "gobuster/3.0.1")
  -U, --username string              Username for Basic Auth
      --wildcard                     Force continued operation when wildcard found

Global Flags:
  -z, --noprogress        Don't display progress
  -o, --output string     Output file to write results to (defaults to stdout)
  -q, --quiet             Don't print the banner and other noise
  -t, --threads int       Number of concurrent threads (default 10)
  -v, --verbose           Verbose output (errors)
  -w, --wordlist string   Path to the wordlist
```

For directory searching the most basic usage is:

```
gobuster dir -u http://<host ip>/ -w
/usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt  -t 50
```

This will test all the words in our wordlist `/usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt` on the end of the url `http://10.10.142.21/`. The final `-t` flag is our thread count, this basically dictates how many attempts you want to run simultaneously.

By adding additional flags you are able refine your search further. You are able to set the status codes (200,204,301 etc) that you want to look for specifically, you may also want to ad `-r` to "follow redirects".

Real-world applications

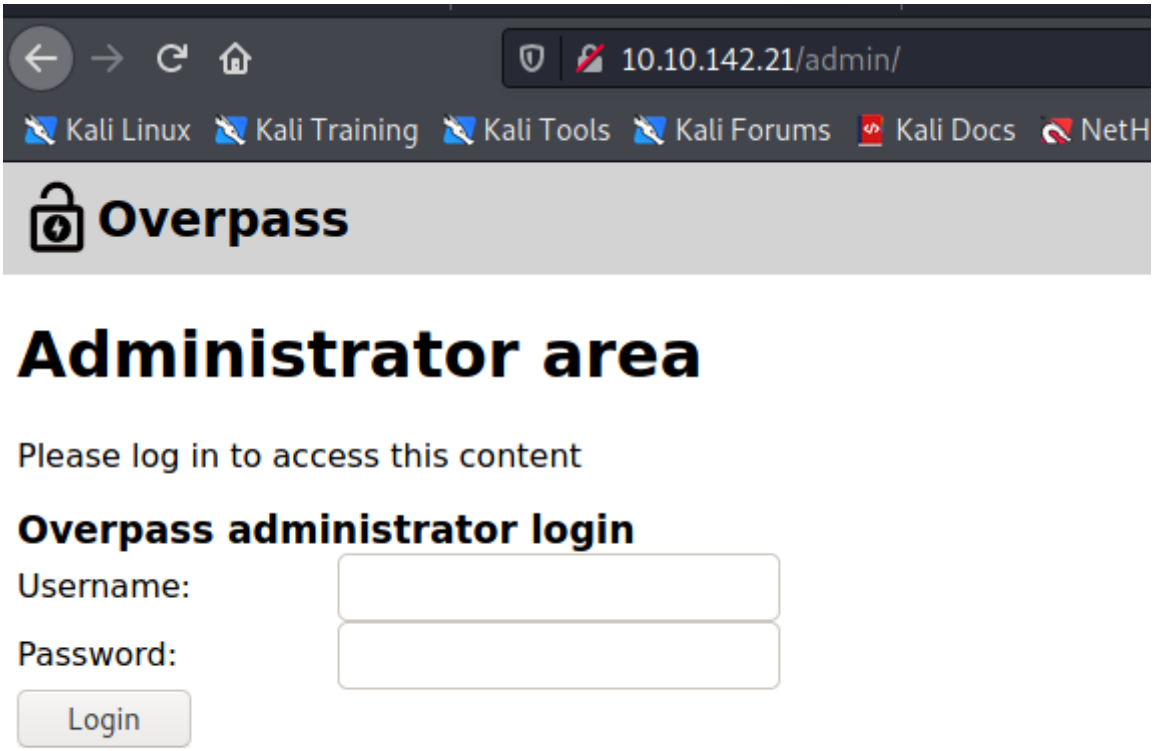
As a demonstration I have run this scan against the `Overpass` site.

```
┌──(kali㉿kali)-[~]
└─$ gobuster dir -u http://10.10.142.21/ -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -t 50
===============================================================
Gobuster v3.0.1
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@_FireFart_)
===============================================================
[+] Url:            http://10.10.142.21/
[+] Threads:        50
[+] Wordlist:       /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
[+] Status codes:   200,204,301,302,307,401,403
[+] User Agent:     gobuster/3.0.1
[+] Timeout:        10s
===============================================================
2021/02/09 00:31:49 Starting gobuster
===============================================================
/img (Status: 301)
/downloads (Status: 301)
/aboutus (Status: 301)
/admin (Status: 301)
/css (Status: 301)
/http%3A%2F%2Fwww (Status: 301)
/http%3A%2F%2Fyoutube (Status: 301)
/http%3A%2F%2Fblogs (Status: 301)
/http%3A%2F%2Fblog (Status: 301)
/**http%3A%2F%2Fwww (Status: 301)
Progress: 89934 / 220561 (40.78%)^C
[!] Keyboard interrupt detected, terminating.
===============================================================
2021/02/09 00:40:22 Finished
===============================================================
```

As you can see there are a number of directories that have come up that where not available from the site tabs.

If we explore our newly discovered `/admin` directory:

http://10.10.142.21/admin/



We find an administrator panel for the site and a potential step in exploitation.

Potential Issues

There are a few potential issues with any directory search. As you can see above there are a number of searches that seem a bit odd. These are 'False-Positives'. Due to the configuration of the site it is re-directing incorrect directories to a `404 error` page. As a result `gobuster` is seeing a re-direct `301 status`. The fix for this is to shape the search and limit the status codes.

The second potential issues is missing returns due to congestion. If we turn the thread-count up too high we can essentially 'DOS' the site we are trying to explore. If you are getting errors it is often a good idea to turn down the thread-count and see if they reduce.

The final potential issue for `gobuster` is its lack of ability to conduct a recursive search. If it identifies are request that is a folder, it will not conduct a search within that directory, it simply moves on. This particular capability is included in the other two searches however with `gobuster` you would need to go back and run a new search to explore any folders.

> **DOS - Denial Of Service**
> A Denial-of-Service (DoS) attack is an attack meant to shut down a machine or network, making it inaccessible to its intended users. DoS attacks accomplish this by flooding the target with traffic, or sending it information that triggers a crash. In both instances, the DoS attack deprives legitimate users (i.e. employees, members, or account holders) of the service or resource they expected.
>
> In our case an excessive thread count can mean that we inadvertently flood the target with requests. This means that a lot of our requests will never make it through and you will receive a lot of false-negatives.

Exercise

Additional flags

If you have access to `TryHackMe` run `gobuster` against the website getting only '200' status codes. Additionally, include any files with the extension `.css`

> HINT: You may need to let gobuster follow redirects as well.

You should end up with the following output:

```
===============================================================
Gobuster v3.0.1
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@_FireFart_)
===============================================================
[+] Url:            http://10.10.128.151/
[+] Threads:        100
[+] Wordlist:       /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
[+] Status codes:   200
[+] User Agent:     gobuster/3.0.1
[+] Extensions:     css
[+] Follow Redir:   true
[+] Timeout:        10s
===============================================================
2021/02/09 04:36:43 Starting gobuster
===============================================================
/main.css (Status: 200)
/downloads (Status: 200)
/img (Status: 200)
/aboutus (Status: 200)
/admin (Status: 200)
/css (Status: 200)
===============================================================
2021/02/09 04:57:40 Finished
===============================================================
```

```
/main.css (Status: 200)
/downloads (Status: 200)
/img (Status: 200)
/aboutus (Status: 200)
/admin (Status: 200)
/css (Status: 200)
```

Assessment

Nil

DirSearch

The next tool we will look at is `Dirsearch`. This is another command line tool which allows a more in depth directory search.

"As a feature-rich tool, dirsearch gives users the opportunity to perform a complex web content discovering, with many vectors for the wordlist, high accuracy, impressive performance, advanced connection/request settings, modern brute-force techniques and nice output."

`Dirsearch` is not installed by default on kali, if you have not already installed this program there are instructions below under "Installing Dirsearch".

How is it Used

For a full list of functions run `dirsearch --help` once installed. `Dirsearch` has a lot of functions which provides flexibility in your search, so make sure you get equated with the various flags. The

following is a basic use which we will break down:

```
dirsearch -u http://<Target IP>/ -w /usr/share/dirbuster/directory-list-2.3-medium.txt -e php,txt -x 403,404 -t 150 -r
```

- `-u` - url of the target
- `-w` - world list (there is a default wordlist which does not require a flag, but this is my preference list).
- `-e` - extensions to test for, separated by commas
- `-x` - status codes to be ignored, separated by commas
- `-t` - threads (same as discussed with `gobuster`)
- `-r` - recursive search (explore the folders)

The command above will check our medium list against the target as a recursive search, exploring each folder it finds. It will test for `.php` & `.txt` extensions. It will exclude results that return a 400 or 404 status code and it will be running 50 threads.

The `-e` flag

While most of the other flags are fairly straight forward the `-e` flag may require a bit more explanation. There are several considerations to keep in mind when using this flag, as selecting the wrong extensions can increase your scan time for no reason. For example it is unlikely that a site will use both `.php` and `.asp` or `.aspx` extensions on the same site as they use two different server side programming or scripting languages. If your site has `.php` extensions you would not include `.asp` or `.aspx` in your `-e` flag. Visa versa if you are looking at a webserver using ASP.NET then it is probably a waste of time including `.php` in your flag.

Have a look at default file type for the webserver, this will be key to selecting the right extensions.

One of the major benefits if `dirsearch` over `gobuster` is its ability to do recursive searches. Not only does it have a significant amount of flexibility and customisation available in the searches it can run.

> NOTE: If you are getting errors, it is likely you have the thread count too high and you should reduce it before continuing the search.

Real-world applications

For real-world application `dirsearch` is extremely useful as a background tool. You can set it running with the appropriate flags and it will search all files and folders to reveal the entire structure while you are enumerating else where. I have run this tool against `Overpass` so you can see how the search progresses through the file structure.

As `dirsearch` detects a folder, it will mark if for an additional search. Once it complete a search of the main directory, it will re-run the same wordlist against each of the discovered directories. This is where this program both shines and slows down.

```
┌──(kali㉿kali)-[~]
└─$ dirsearch -u http://10.10.86.44/ -w /usr/share/dirbuster/wordlists/directory-list-lowercase-2.3-medium.txt -e php,txt -x 403,404 -t 150 -r
/opt/dirsearch/thirdparty/requests/__init__.py:91: RequestsDependencyWarning: urllib3 (1.26.2) or chardet (4.0.0) doesn't match a supported version!
  warnings.warn("urllib3 ({}) or chardet ({}) doesn't match a supported "

  _|. _ _  _  _  _ _|_    v0.4.1
 (_||| _) (/_(_|| (_| )

Extensions: php, txt | HTTP method: GET | Threads: 150 | Wordlist size: 207628

Error Log: /opt/dirsearch/logs/errors-21-02-09_07-14-23.log

Target: http://10.10.86.44/

Output File: /opt/dirsearch/reports/10.10.86.44/_21-02-09_07-14-23.txt

[07:14:23] Starting:
[07:14:25] 301 -    0B  - /img  →  img/     (Added to queue)
[07:14:25] 301 -    0B  - /downloads  →  downloads/     (Added to queue)
[07:14:26] 301 -   42B  - /admin  →  /admin/     (Added to queue)
[07:14:26] 301 -    0B  - /aboutus  →  aboutus/     (Added to queue)
[07:14:27] 301 -    0B  - /css  →  css/     (Added to queue)
[07:15:09] 301 -    0B  - /http%3a%2f%2fwww  →  /http:/www
[07:16:13] 301 -    0B  - /http%3a%2f%2fyoutube  →  /http:/youtube
26.55% - Job: 1/6 - Last request to: 53219
```

As you can see it has added folders to a queue. Once it has completed the first round it will start searching the queue.

```
┌──(kali㉿kali)-[~]
└─$ dirsearch -u http://10.10.86.44/ -w /usr/share/dirbuster/wordlists/directory-list-lowercase-2.3-medium.txt -e php,txt -x 403,404 -t 150 -r
/opt/dirsearch/thirdparty/requests/__init__.py:91: RequestsDependencyWarning: urllib3 (1.26.2) or chardet (4.0.0) doesn't match a supported version!
  warnings.warn("urllib3 ({}) or chardet ({}) doesn't match a supported "

  _|. _ _  _  _  _ _|_    v0.4.1
 (_||| _) (/_(_|| (_| )

Extensions: php, txt | HTTP method: GET | Threads: 150 | Wordlist size: 207628

Error Log: /opt/dirsearch/logs/errors-21-02-09_07-14-23.log

Target: http://10.10.86.44/

Output File: /opt/dirsearch/reports/10.10.86.44/_21-02-09_07-14-23.txt

[07:14:23] Starting:
[07:14:25] 301 -    0B  - /img  →  img/     (Added to queue)
[07:14:25] 301 -    0B  - /downloads  →  downloads/     (Added to queue)
[07:14:26] 301 -   42B  - /admin  →  /admin/     (Added to queue)
[07:14:26] 301 -    0B  - /aboutus  →  aboutus/     (Added to queue)
[07:14:27] 301 -    0B  - /css  →  css/     (Added to queue)
[07:15:09] 301 -    0B  - /http%3a%2f%2fwww  →  /http:/www
[07:16:13] 301 -    0B  - /http%3a%2f%2fyoutube  →  /http:/youtube
[07:16:43] 301 -    0B  - /http%3a%2f%2fblogs  →  /http:/blogs
[07:16:46] 301 -    0B  - /http%3a%2f%2fblog  →  /http:/blog
[07:17:07] 301 -    0B  - /**http%3a%2f%2fwww  →  /%2A%2Ahttp:/www
[07:20:24] 301 -    0B  - /http%3a%2f%2fcommunity  →  /http:/community
[07:20:43] 301 -    0B  - /http%3a%2f%2fradar  →  /http:/radar
[07:21:26] 301 -    0B  - /http%3a%2f%2fjeremiahgrossman  →  /http:/jeremiahgrossman
[07:21:27] 301 -    0B  - /http%3a%2f%2fweblog  →  /http:/weblog
[07:21:28] 301 -    0B  - /http%3a%2f%2fswik  →  /http:/swik
[07:21:31] Starting: img/
[07:22:17] 301 -    0B  - /img/http%3a%2f%2fwww  →  /img/http:/www
[07:23:20] 301 -    0B  - /img/http%3a%2f%2fyoutube  →  /img/http:/youtube
[07:23:45] 301 -    0B  - /img/http%3a%2f%2fblogs  →  /img/http:/blogs
[07:23:49] 301 -    0B  - /img/http%3a%2f%2fblog  →  /img/http:/blog
[07:24:10] 301 -    0B  - /img/**http%3a%2f%2fwww  →  /img/%2A%2Ahttp:/www
[07:27:30] 301 -    0B  - /img/http%3a%2f%2fcommunity  →  /img/http:/community
[07:27:48] 301 -    0B  - /img/http%3a%2f%2fradar  →  /img/http:/radar
[07:28:31] 301 -    0B  - /img/http%3a%2f%2fjeremiahgrossman  →  /img/http:/jeremiahgrossman
[07:28:31] 301 -    0B  - /img/http%3a%2f%2fweblog  →  /img/http:/weblog
[07:28:33] 301 -    0B  - /img/http%3a%2f%2fswik  →  /img/http:/swik
[07:28:36] Starting: downloads/
[07:28:41] 301 -    0B  - /downloads/src  →  src/     (Added to queue)
[07:28:51] 301 -    0B  - /downloads/builds  →  builds/     (Added to queue)
[07:29:26] 301 -    0B  - /downloads/http%3a%2f%2fwww  →  /downloads/http:/www
13.19% - Job: 3/8 - Last request to: index-151
```

I can also see a potential loop is going to occur in this search if I let it finish. The search for `http%3a%2f%2fwww` has come back as a 301 response in all directories so far and will likely continue into a loop resulting in search for `/http%3a%2f%2fwww/http%3a%2f%2fwww/http%3a%2f%2fwww/http%3a%2f%2fwww...`

But you can see that as the search continues it has returned `/downloads/builds` & `/downloads/src`, these are returns which `gobuster` would not have found.

One flag that my be extremely useful in reducing results is `-R` (not to be confused with `-r`). This flag dictates the Maximum recursion depth of the search. This means that if there is a redirect that sends your search in circles you can prevent it from getting caught in an endless loop. The down side being you may not get to explore to then end of the file structure.

Potential Issues

This search is slower than `gobuster`, but really it makes up for this in functionality. However, this functionality can also become overwhelming at times, so it is important to practice this tool and understand the capabilities.

One of the big potential issues with `dirsearch` is again false-positives caused by redirects. Due to the ability to recursively search this can compound by creating an endless loop of false-positives within false-positives. This is where the `-R` flag becomes useful in allowing you to reduce the time spent on false-positives. You can also shape the status codes to try and reduce these false-positive responses.

Exercise

Installing Dirsearch

`DirSearch` does not come pre-installed on kali at this point in time. So we will go through the install process as an exercise. First you need to decide where you want to install `Dirsearch`. I like to install new tools like this in `/opt` depending on your OS and who you talk to, others deem `/usr/local` to be the appropriate directory. For this instance I will be using `/opt`.

1. **Navigate to install location.**

`cd /opt`

2. **Download or git clone the git repository.** We can do this using the `git clone` command.

   `sudo git clone https://github.com/maurosoria/dirsearch.git`

   ```
   ┌──(kali㉿kali)-[/opt]
   └─$ sudo git clone https://github.com/maurosoria/dirsearch.git
   [sudo] password for kali:
   Cloning into 'dirsearch'...
   remote: Enumerating objects: 85, done.
   remote: Counting objects: 100% (85/85), done.
   remote: Compressing objects: 100% (71/71), done.
   remote: Total 7273 (delta 49), reused 28 (delta 13), pack-reused 7188
   Receiving objects: 100% (7273/7273), 19.87 MiB | 6.29 MiB/s, done.
   Resolving deltas: 100% (4715/4715), done.
   ```

   > NOTE: `/opt` is owned by root so we need to use `sudo` to make changed to the directory or change the owner, I have simply used `sudo`.

3. **Move into the new directory.**

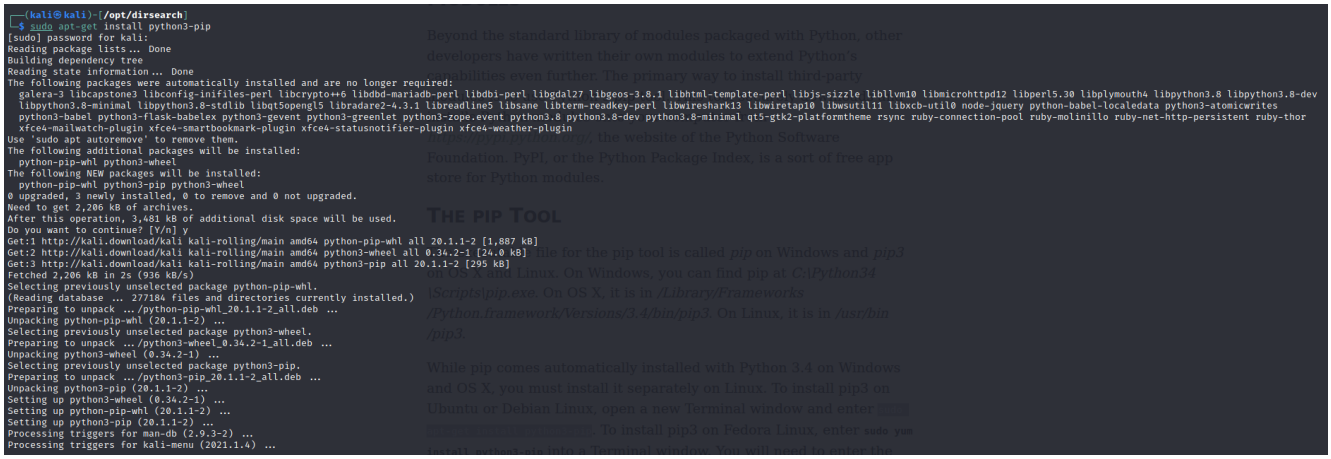
`cd dirsearch`

4. **Install requirements.txt**. To get the full functionality of `dirsearch` there are some additional requirements to be installed. The easiest way to do this is using `pip3`.

4a. **Installing pip3** (If you have already installed `pip3` go to '4b.')

The pip tool is used for installing 3rd party modules for python. It does not come pre-installed on Linux at this point in time. The command to install is:

`sudo apt-get install python3-pip`



4b. Once you have `pip3` installed the command is simply:

`pip3 install -r requirements.txt`

You can now run `dirsearch` for this directory using the following command:

`sudo python3 dirsearch.py --help`

5. **Adding an alias.** (optional)

If you have not used aliases before, they are a handy shortcut for running commands that might be long or annoying but used regularly. To add a permanent alias you need to modify your shell configuration file. This is the file that tells your shell how to behave, look, and any alterations.

The default shell for kali 2020.3 and greater is `zsh` so we will add our alias to `~/.zshrc`, being our users `zsh` config file.

Use your favourite text editor to open this file

`nano ~/.zshrc`

Scroll all the way to the end and add the following lines:

```
#My custom aliases
alias dirsearch="python3 /opt/dirsearch/dirsearch.py"
```

Finally run

`source ~/.zshrc`

to "refresh" the config. Alternatively, close and re-open your terminal. Now you can run `dirsearch` by simply typing `sudo dirsearch --help`

Assessment

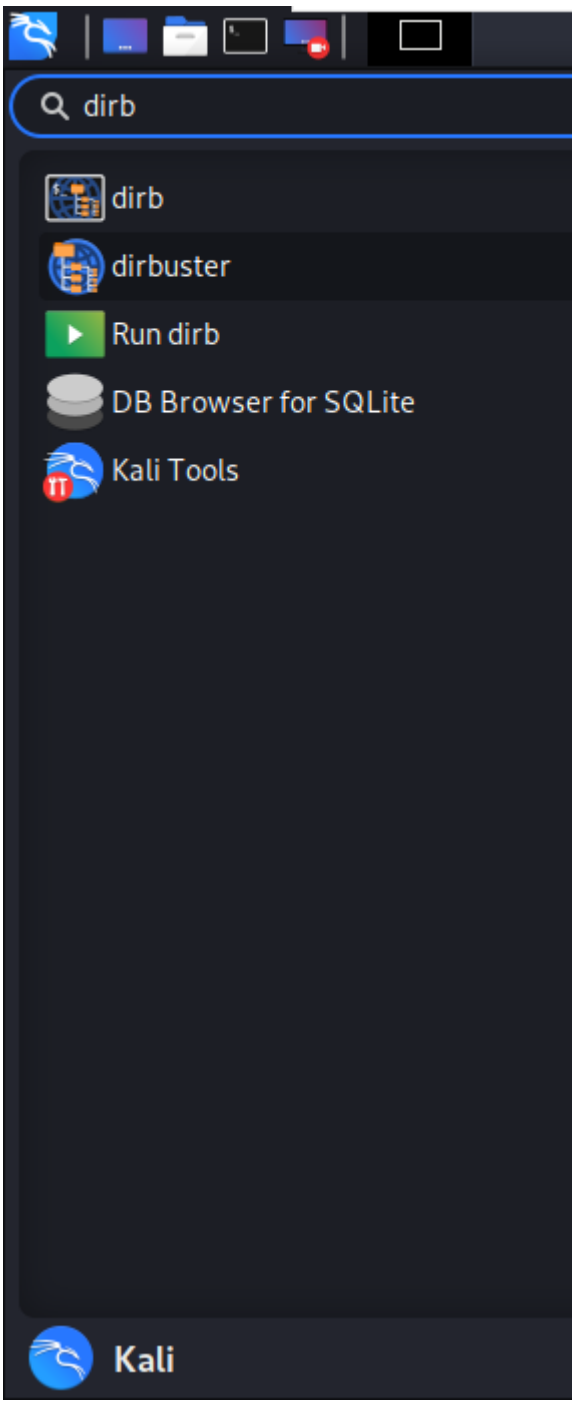
Nil

DirBuster

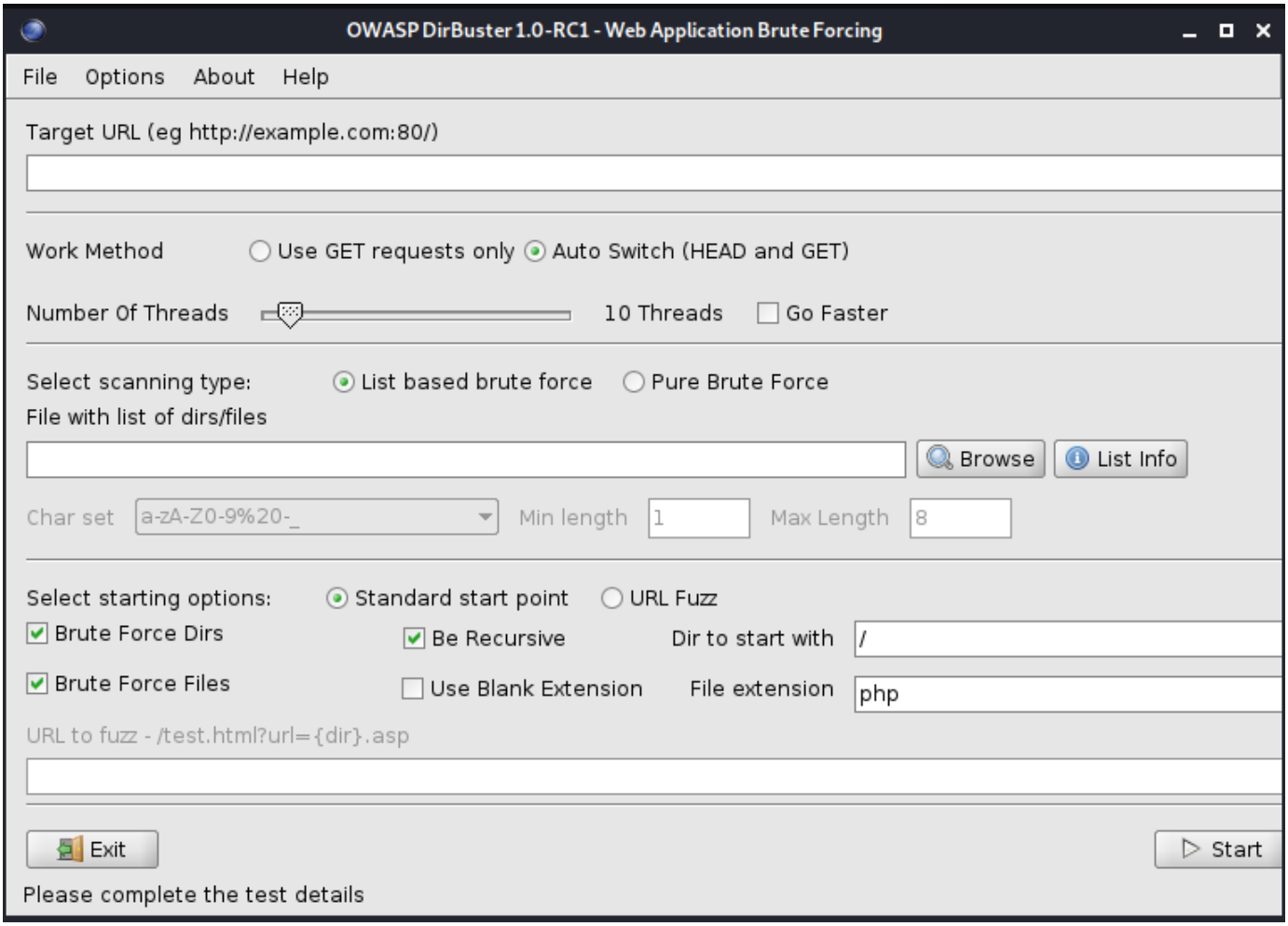
Dirbuster is the final directory search tool we will talk about. The major difference in this tool is that its has a Graphical User Interface (GUI). This provides additional levels of functionality but can also be cumbersome and require user intervention.

How is it Used

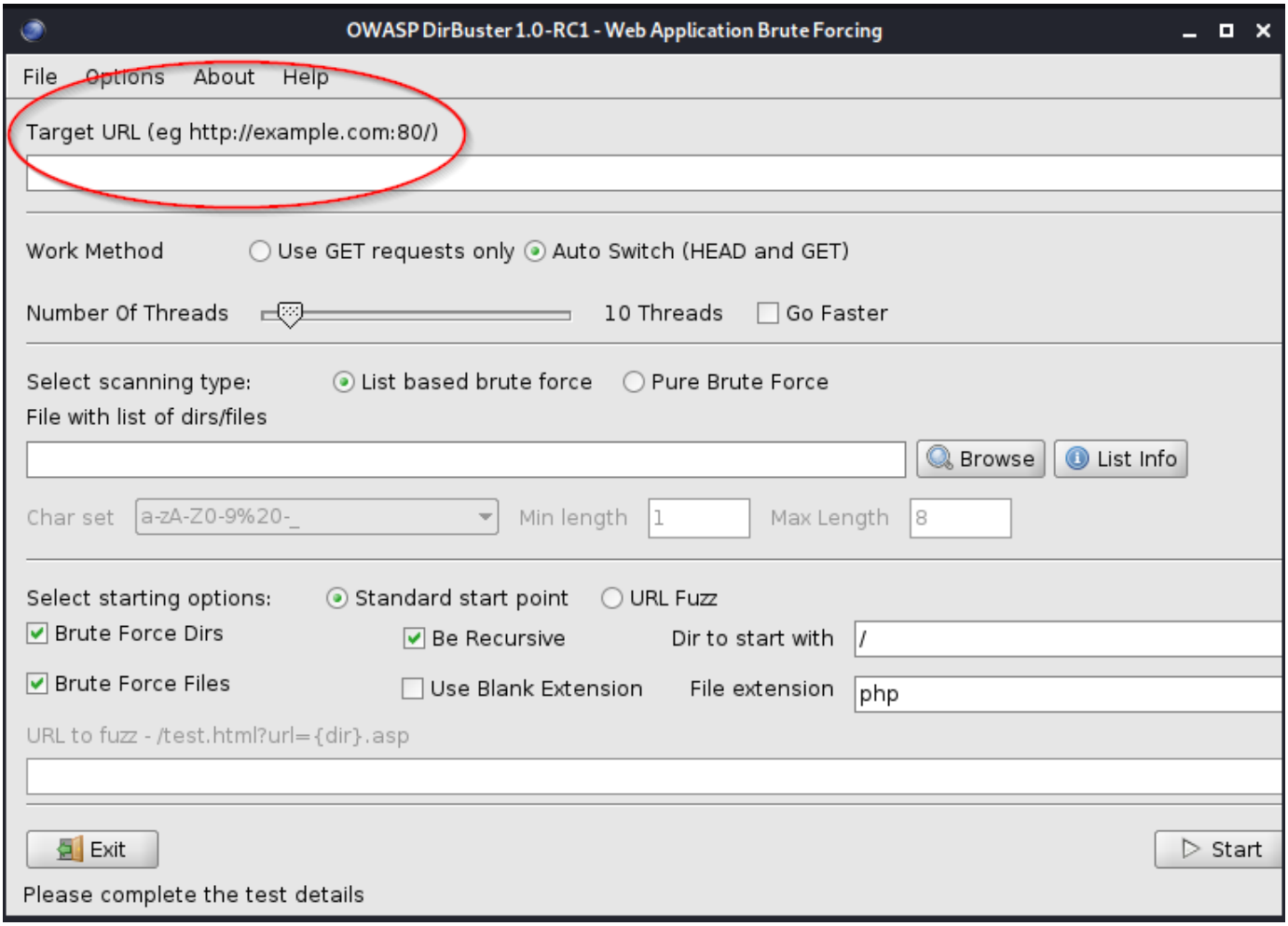
Dirbuster is installed on kali 2020.4 by default. You can access it from your applications tab:



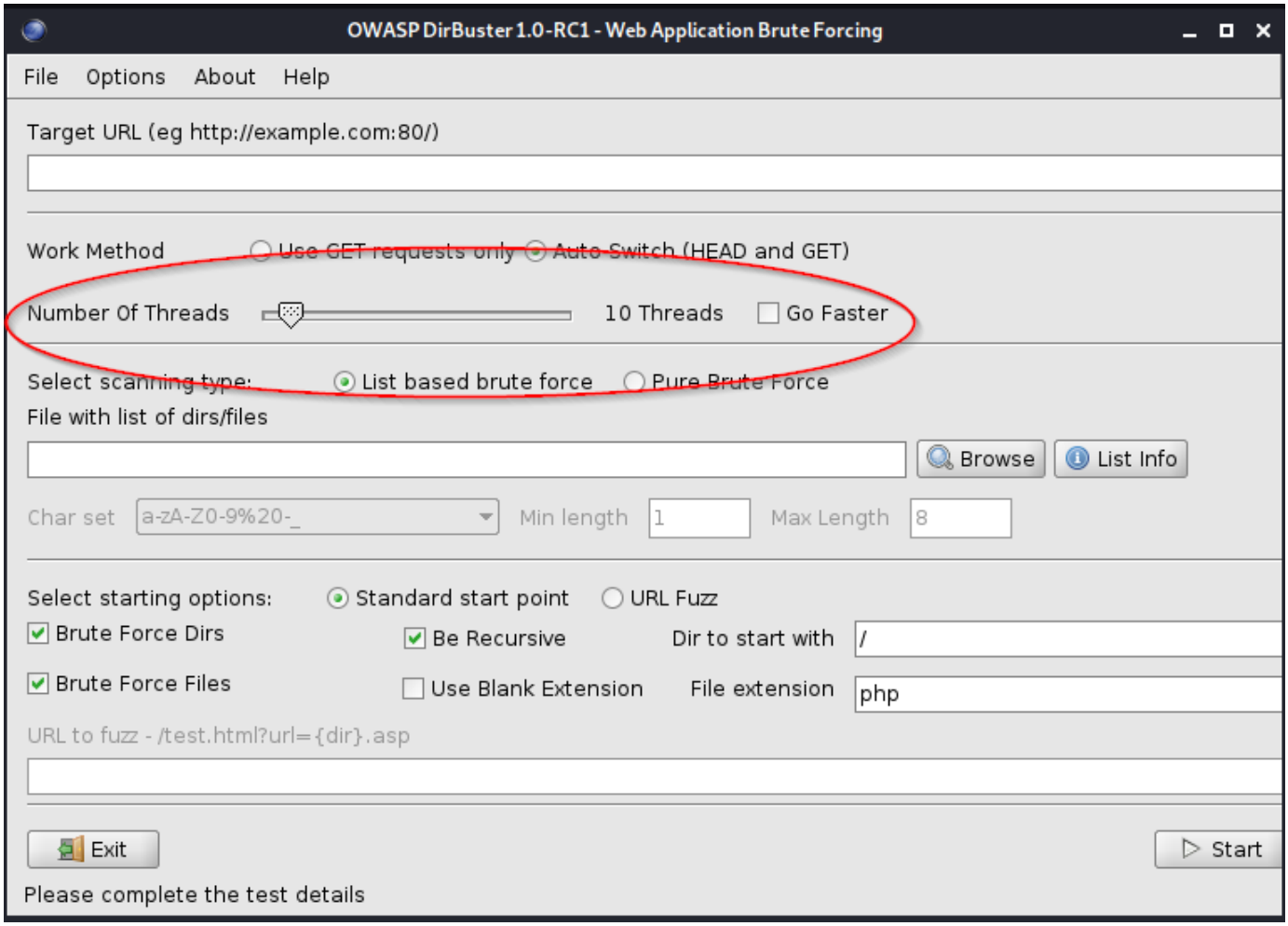
You will be presented with a fairly user friendly GUI:



First you will need to enter your website URL ensuring you use the port number. `http://<Target IP>:80/`

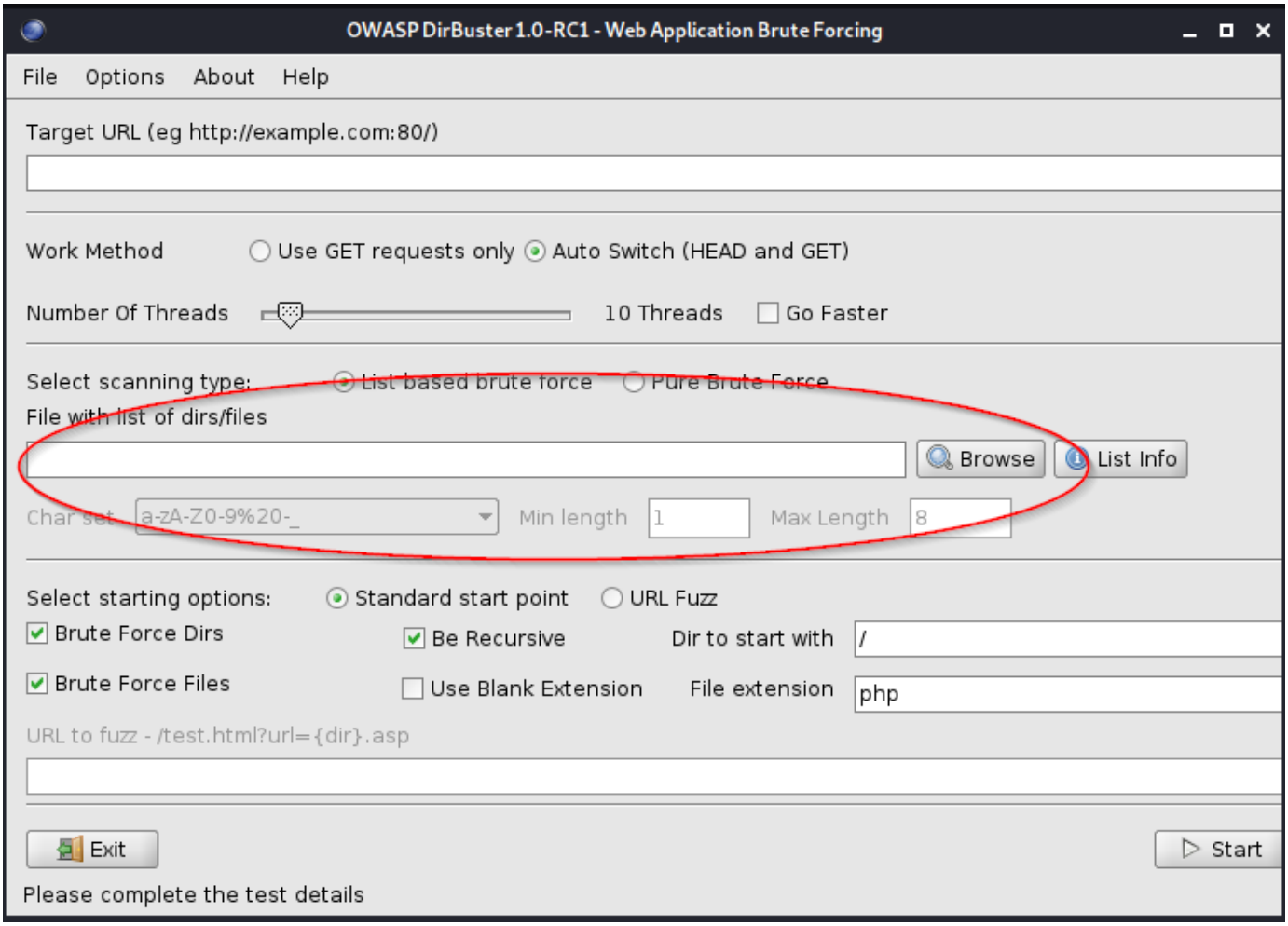


Next we will select our thread count. Remember we don't want to go so high with our thread count that we overload the site and create errors in our search.

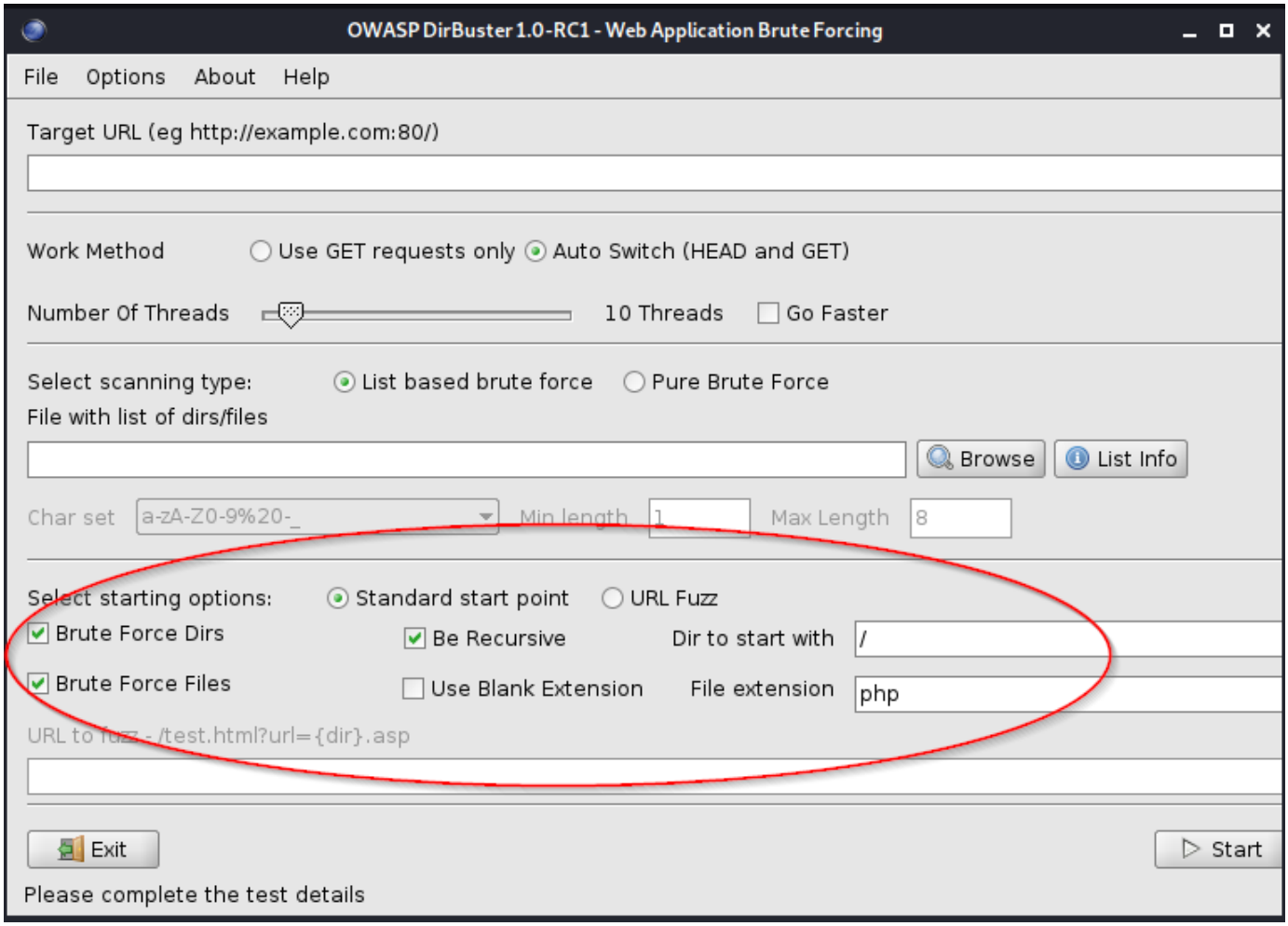


Now we can select our wordlist. I like to use `/usr/share/dirbuster/wordlists/directory-list-2.3-medium.txt` as it covers most things. Depending on what your targeting or the time you have you may find a better list for this, but as a default this should be your go to.

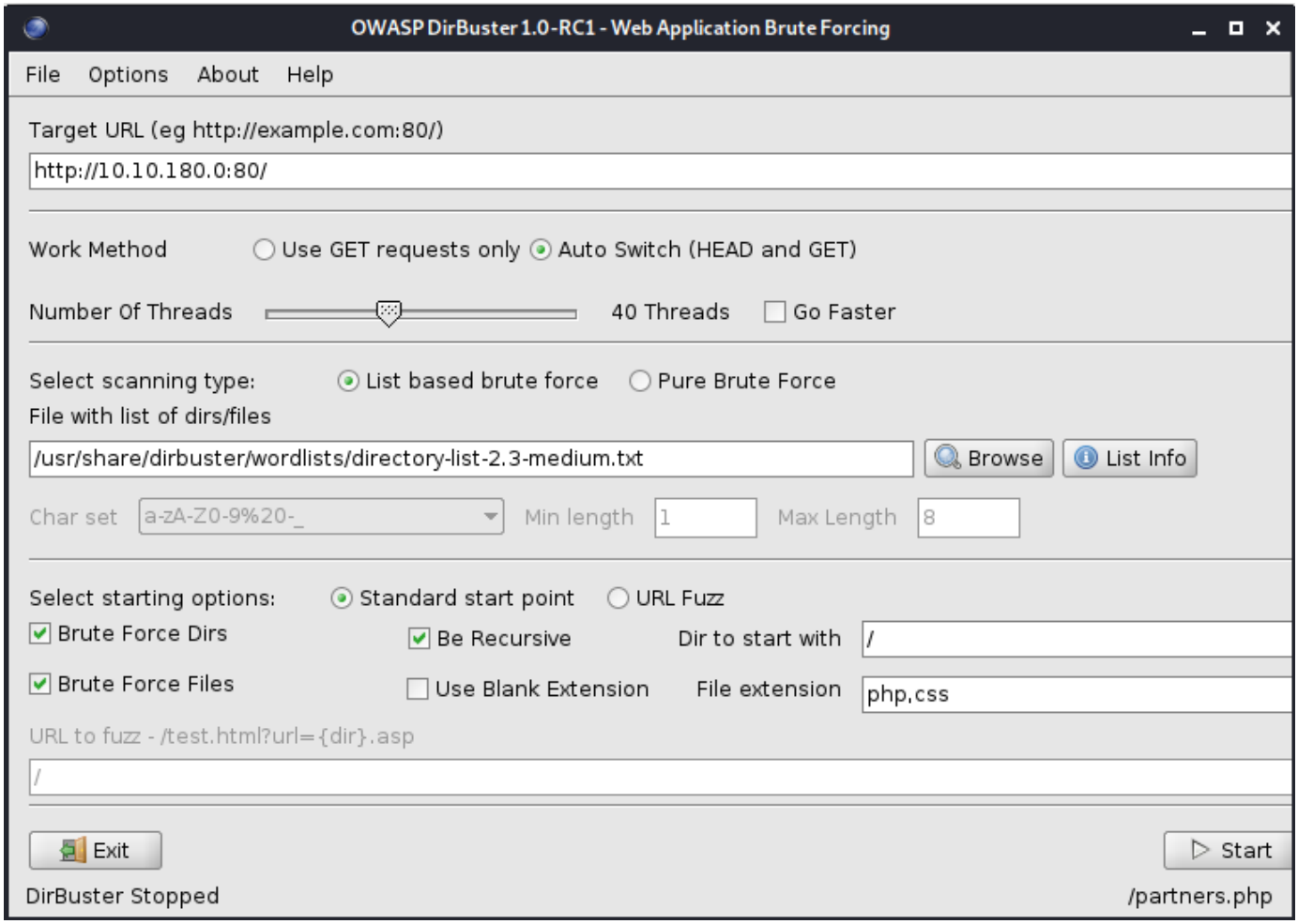
> NOTE: We do have an option of 'Pure Brute Force' this will take a very very long time if you do not have some specifics in mind.



Finally, we can select our options. These are the same as the additional flags from `dirbuster` or `gobuster`.

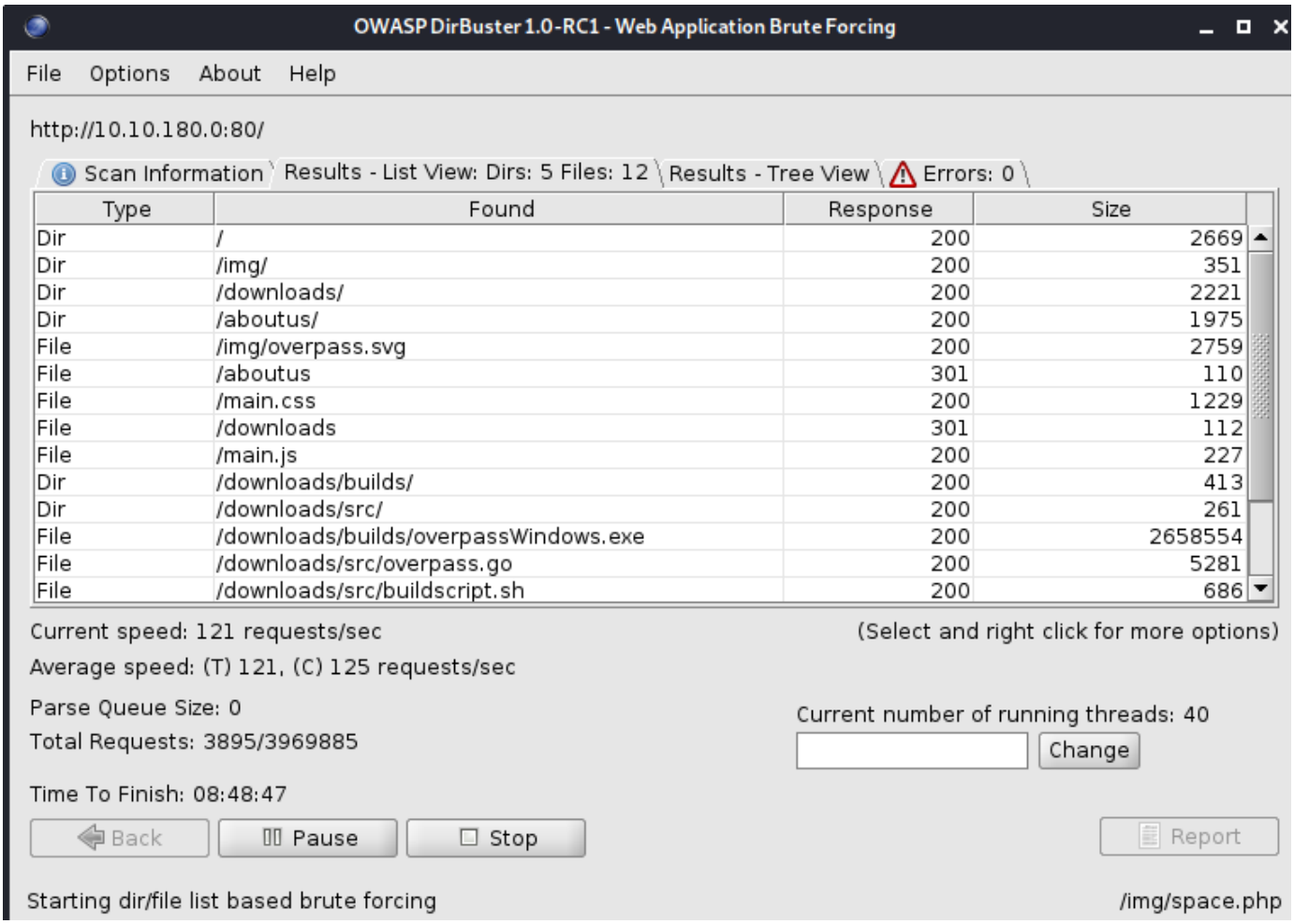


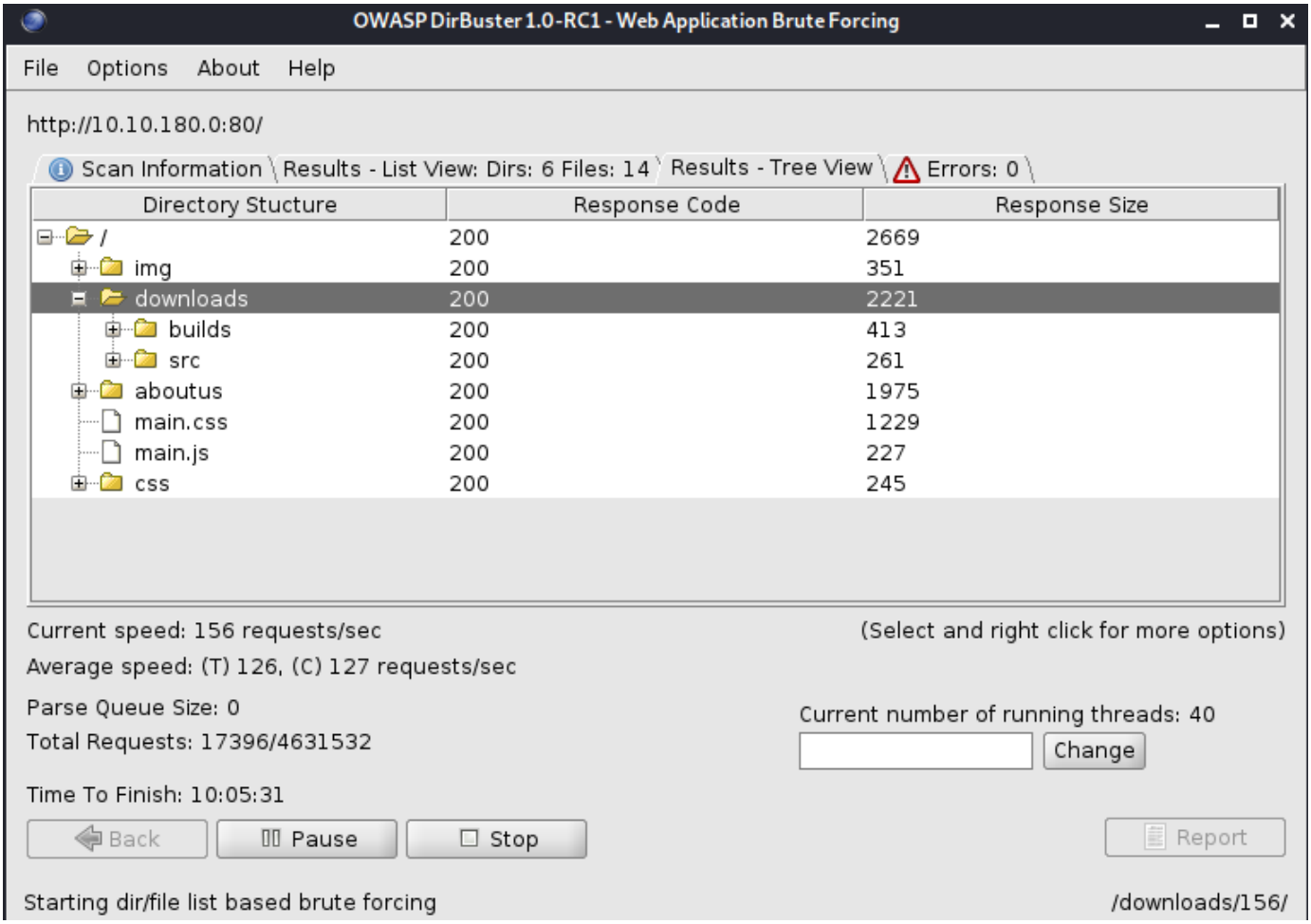
Your final set up might look something like this:



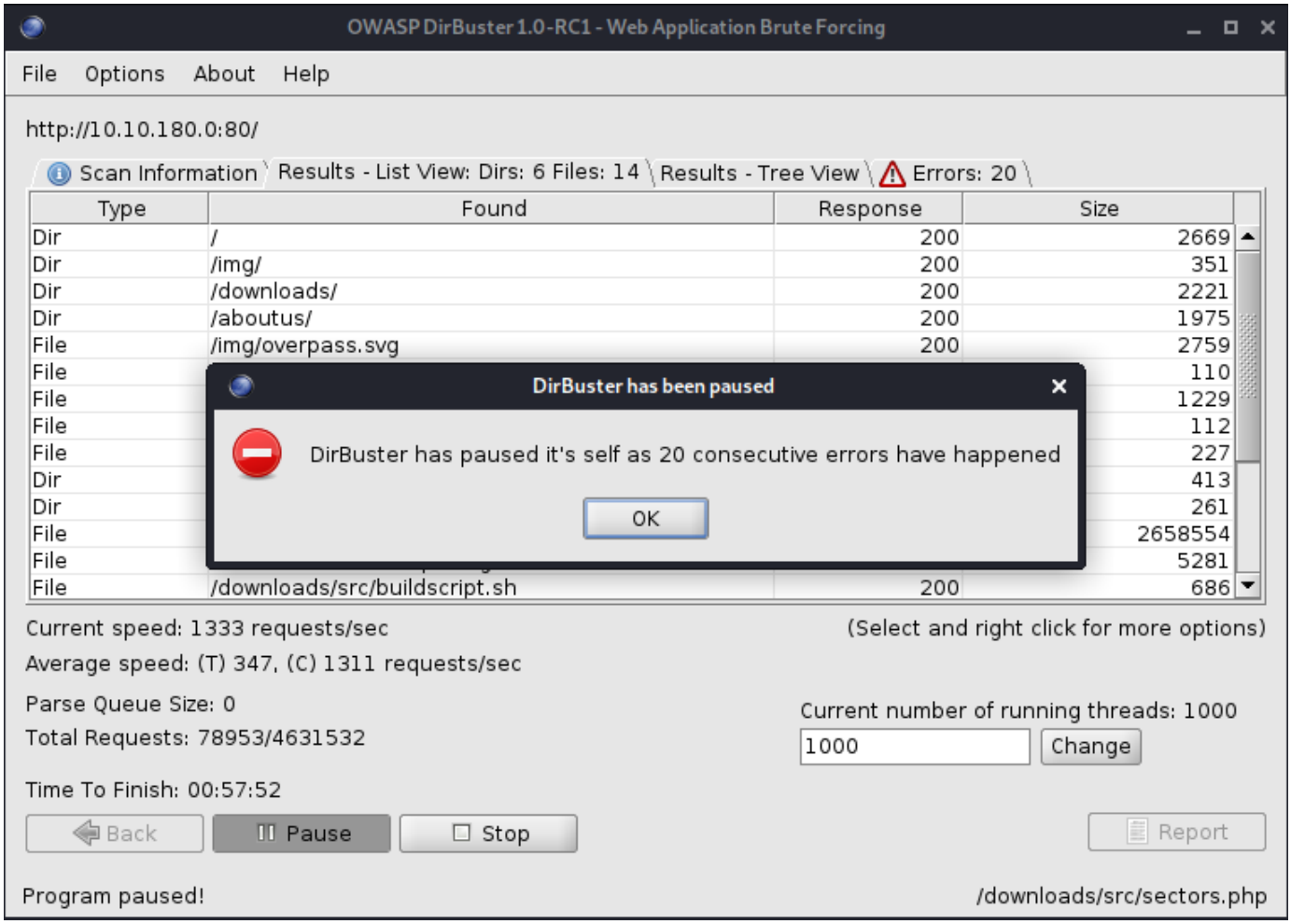
Real-world applications

Once you have `dirbuster` running the results are displayed in a tidy, easy to read manner. You can view the out put in two ways, as a list or a tree.





While `dirbuster` is running you may start to see errors populate. By default `dirbuster` will pause after 20 errors to let you make changes if required.



You can see above I have my thread count set to 1000, and it is attempting 1333 requests/second. To prevent further errors I can change the thread count to something a bit lower like 50 and un-pause it.

OWASP DirBuster 1.0-RC1 - Web Application Brute Forcing

File Options About Help

http://10.10.180.0:80/

Scan Information | Results - List View: Dirs: 6 Files: 14 | Results - Tree View | Errors: 89

Type	Found	Response	Size
Dir	/	200	2669
Dir	/img/	200	351
Dir	/downloads/	200	2221
Dir	/aboutus/	200	1975
File	/img/overpass.svg	200	2759
File	/aboutus	301	110
File	/main.css	200	1229
File	/downloads	301	112
File	/main.js	200	227
Dir	/downloads/builds/	200	413
Dir	/downloads/src/	200	261
File	/downloads/builds/overpassWindows.exe	200	2658554
File	/downloads/src/overpass.go	200	5281
File	/downloads/src/buildscript.sh	200	686

Current speed: 189 requests/sec (Select and right click for more options)

Average speed: (T) 205, (C) 157 requests/sec

Parse Queue Size: 0

Total Requests: 82183/4631532

Current number of running threads: 50

50 Change

Time To Finish: 08:02:56

Back Pause Stop Report

Program running again /downloads/Insurance/

In the error tab we can see all the requests that failed and why.

OWASP DirBuster 1.0-RC1 - Web Application Brute Forcing

File Options About Help

http://10.10.180.0:80/

Scan Information | Results - List View: Dirs: 6 Files: 14 | Results - Tree View | Errors: 89

Request	Error Message
http://10.10.180.0:80/img/home_off.php	ConnectTimeoutException The host did not accept the
http://10.10.180.0:80/img/previous/	ConnectTimeoutException The host did not accept the
http://10.10.180.0:80/downloads/src/337/	ConnectTimeoutException The host did not accept the
http://10.10.180.0:80/downloads/304.css	ConnectTimeoutException The host did not accept the
http://10.10.180.0:80/aboutus/dleimages/	ConnectTimeoutException The host did not accept the
http://10.10.180.0:80/downloads/school.css	ConnectTimeoutException The host did not accept the
http://10.10.180.0:80/css/cp1/	ConnectTimeoutException The host did not accept the
http://10.10.180.0:80/downloads/builds/bondage.php	ConnectTimeoutException The host did not accept the
http://10.10.180.0:80/css/Files/	ConnectTimeoutException The host did not accept the
http://10.10.180.0:80/downloads/src/soccer.php	ConnectTimeoutException The host did not accept the
http://10.10.180.0:80/aboutus/authentication.css	ConnectTimeoutException The host did not accept the
http://10.10.180.0:80/downloads/src/1006/	ConnectTimeoutException The host did not accept the
http://10.10.180.0:80/img/employers.php	ConnectTimeoutException The host did not accept the
http://10.10.180.0:80/downloads/src/webinars.css	ConnectTimeoutException The host did not accept the

Current speed: 149 requests/sec (Select and right click for more options)

Average speed: (T) 204, (C) 158 requests/sec

Parse Queue Size: 0

Total Requests: 84078/4631532

Current number of running threads: 50

50 Change

Time To Finish: 07:59:41

Back Pause Stop Report

Program running again /downloads/rt/

Potential Issues

If you are getting errors for any reason the program will pause. This means that there can be a bit of user interaction required, rather than a 'set and forget' like `dirsearch`. Additionally, there may be times when you don't have access to a GUI, in which case this is not a viable tool.

Exercise

Try running a `dirbuster` scan on any machine with a web site (that you have approval to use). Turn the threads up and down to create errors, and have a look at the different flags that can be used.

Assessment

Nil

Summary

These tools are one of the staples of pen-testing. You will use one or more of them when ever you are faced with a website or webapp, so take the time to get comfortable with how to use them.

There is nothing more frustrating than realising that the way forward was hidden in a directory that your scan error-ed over because your thread count was too high, or you only used the small list instead of medium.