



# Directory brute forcing

[Introduction](#)

[Attack strategies](#)

[Non recursive vs recursive scanning](#)

[Content discovery](#)

[Size does matter](#)

[Single target vs a list of targets](#)

[Parameter fuzzing, content discovery or directory brute forcing? HELP!](#)

[vHost brute forcing](#)

[Tools](#)

[BURP SUITE PRO: Burp suite content discovery](#)

[Wfuzz](#)

## Introduction

When we talk about directory brute forcing we are in essence trying to guess the directories of our target's webserver. We know that there is a webserver running and we might even have access to certain pages like /login.php which is guarding some juicy loot or we might just see that there is an IIS server running and we want to explore it some more. Whatever the case may be, we can approach this issue using several attack strategies.

This is something that we always do automated as trying to guess possibly millions of directories and check them manually can take quite a while as you might imagine. You might also be able to image that if i ask you to check 10 directories that it would take you a lot less time than checking 100000 directories. I bring this to your imagination because even though it's normal and logical, the same goes for automated scanners. The quality of your wordlist will determine the quality of your results but the same is true for the length of your wordlist determining the runtime of your attack.

## Attack strategies

## Non recursive vs recursive scanning

It does not matter what we want to fuzz, whether it be directories, content or even vhosts, when we scan non recursively, we are referring to whether or not the crawler should follow the links that it finds.

The crawler is the robot that will make the requests that we set it up to create based on our wordlists.

In non recursive scanning we do not allow this crawler to follow any links at all. We want the crawler to only make the requests that we tell it to and if it find a link we want the crawler to ignore that link.

Recursive scanning however will allow the crawler to follow these links that it finds. In recursive crawling we can also set the depth which will determine how deep the crawler will follow those links. Sometimes the crawler might follow a link and find even more links on that page, we can set it up to follow those links or just got 1 level deep.

## Content discovery

When we talk about content discovery we can either talk about adding content discovery to our methodology or only doing file discovery. We can also do this recursively or non-recursively but whatever option we pick, the type of content we are looking for will also play a factor in the runtime of our tools. If you are looking for image files for example, you might be looking for JPG files but you might also want to add PNG and GIF to the mix which will triple the runtime of our tools since it has to check every request three times.

When we fuzz for content discovery we can fuzz for several different things. I recommend that you have a specialised wordlist for every type of content because ofcourse fuzzing for pictures will probably require a different wordlist than fuzzing for documents.


- Pictures (jpg,png,gif,...)
- Scripts (js )
- Documents (xls,xlsx,doc,docx,pdf,...)
- Files/endpoints (asp, aspx, php , ...)
- Use your imagination!

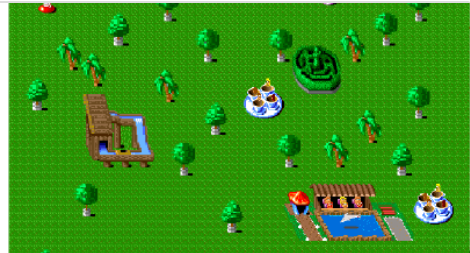
## Size does matter

We've talked about runtime several times before in this document and that has a reason. Runtime is going to be one of the determining factors of a successful attack. You can't have a good directory brute force if it runs for 6 years! So you might be wondering, okay what wordlist exactly do i use uncle? I say pick one but make it count!

heilla/SecurityTesting

Contribute to heilla/SecurityTesting development by creating an account on GitHub.

 <https://github.com/heilla/SecurityTesting/blob/master/wordlists/Collection%20of%20wordlists.md>



Make sure you pick a list that fits your target and if you can't find one then maybe you should make one.

There will be a video included in the course on what wordlists that i use.

## Single target vs a list of targets

Some of these tools will allow us to check a whole list of URLs and do directory brute forcing on that list instead just checking one target at a time. Even if the tool we use does not allow us to do this, a simple command can be all that it takes to feed a list to our tool instead of a single URL.

In this case, we really need to make sure our wordlists are not too big because every single scan will be repeated and increase the scantime. I do not recommend this and i recommend only doing targetted directory brute forcing attacks. Especially on slow computers this will be a huge hassle to complete and we do not want to frustrate ourselves any more than we need to.

## Parameter fuzzing, content discovery or directory brute forcing? HELP!

Besides the already known options of content discovery and directory brute forcing, there is also an option to perform parameter fuzzing. If you think about it, all that we are doing is just replacing a certain value in a URL up to this point, for example we can take this URL as input

```
https://example.com/FUZZ
```

And i can then replace the word FUZZ with every single word from my list and make the request. This can be either directories or files if i append a file extensions:

```
https://example.com/FUZZ.php
```

But what this also allows us to do is replace a certain GET parameter or POST value from our request with a value from our list:

```
https://example.com/index.php?FUZZ=1
```

And yes ladies and gentlemen, it can get even stranger, we can fuzz subdomains:

```
https://FUZZ.example.com/
```

## vHost brute forcing

vHost stands for virtual host. Virtual hosts are when multiple web applications are run on the same webserver. This is often done to save on having to setup and buy a separate server but it is not recommended at all, this has never been a valid reason to let anyone stop them to from doing anything though and as you can imagine it does happen in production environments.

An example of this can be when a company decides to host the production server and the UAT server on the same webserver but on a different vhost.

testing.google.com

www.google.com

Tidofjgdfg.hogfdpigdf.com

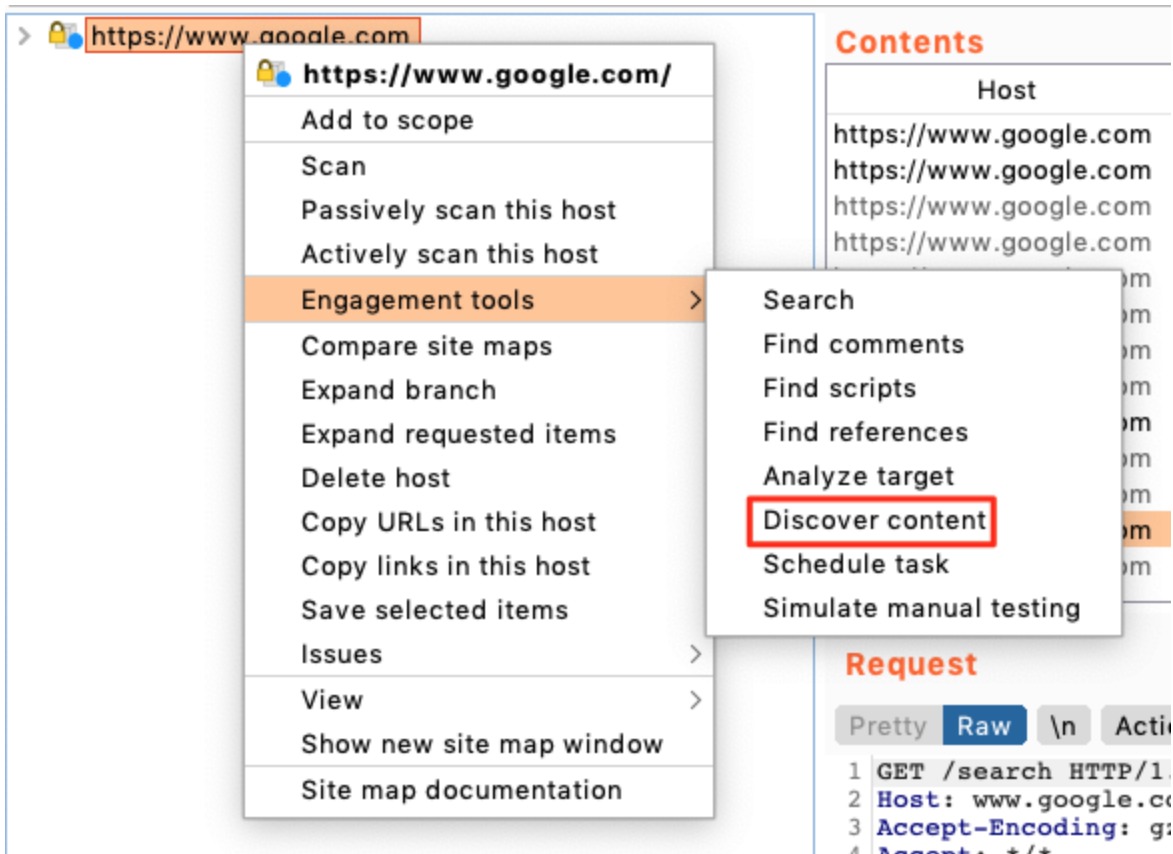
it doesn't matter as long as they all resolve to the same IP adress, it means they are configured in the /etc/hosts file of that host. I only mention them briefly in this course as i

think they are not as big of a security issue as people make them out to be in most situations as long as the separation of flows is handled properly but there might be others that differ from this opinion.

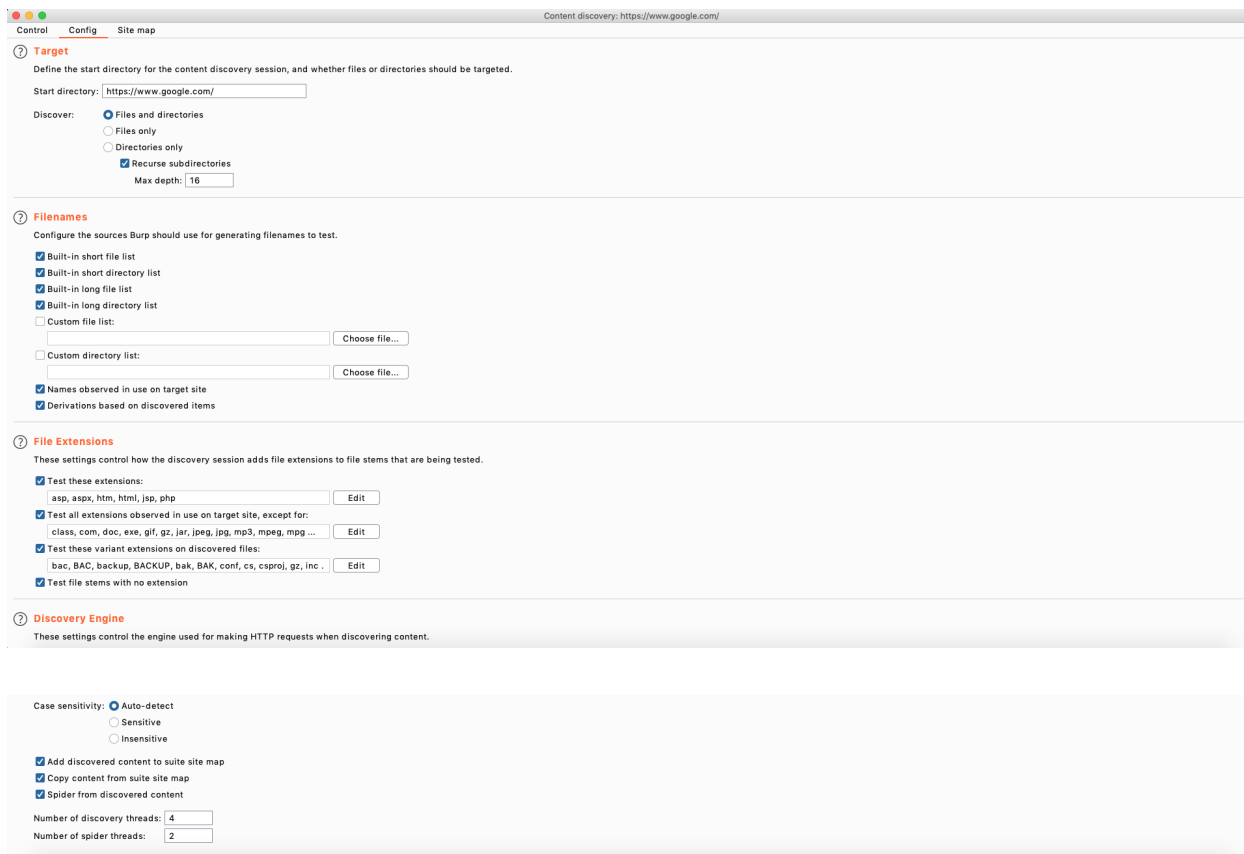
## Tools

### BURP SUITE PRO: Burp suite content discovery

Burp suite pro users have a range of engagement tools available to them, one of them and a very important one for that matter would be the content discovery tool. This is one of the most sophisticated spiders i ever found and it's the one i use most, however it is limited to content and directory brute forcing, it can not fuzz parameters.



Burp Suite Pro version content discovery



There are a lot of options in here that most people don't ever touch which is a big shame! Ofcourse the default options are fine, but i am a big proponent of tweaking your attack strategy for every target. This includes setting custom settings for our attack tools, blindly running the same tool against a range of different targets is not a good idea. Let's have a look at what we can tweak here.

- Start directory: We can play with this if we don't want to test the root of a website for example
- Discover: Here we can pick if we want to find only files, directories or both (Default option), The recursive subdirectories option allows us to set how deep the spider will crawl. By default if the spider has gone 16 links deep into a website it will stop but we can tweak this.
- Filenames: Burp suite has some filename lists built in by default and it will use the short and long list by default which makes the scan quite lengthy if the spider can go 16 levels deep. I usually start out with the small list and if i don't find what i need, i will go on to the big list, and ofcourse we can never forget about custom lists either. They are the diesel to our engines!

- File extensions is something you ALWAYS have to tune to your target, just open it, open wappalyzer(browser plugin) and see what server is running. Don't waste time trying to brute force ASPX servers with PHP files. Don't be a dumbass.
- Discovery engine: This will determine how the crawler engine will behave towards your target. The default options are good here unless you know for sure your target is case (in)sensitive. The number of threads determine how many children burp suite will spawn (that sounds brutal) to start crawling and discovering content. If your target requires you to keep it low and slow, ofcourse do change these values.

When everything is configured correctly, burp suite can start running.

## Wfuzz

Wfuzz is a fuzzer that provides us with a multitude of useful options. We will go over the most useful ones. We can do this either in the CLI in a command or we can import it into our python scripts but we will be focussing on the CLI option. It is important to know that wfuzz can do much more than just fuzzing. It has

- A payload generator
- Encoder/decoder
- Library for python

I will not explain the basic usage as it's self explanatory over at

<https://wfuzz.readthedocs.io/en/latest/user/basicusage.html#fuzzing-paths-and-files>

```
$ wfuzz -w wordlist/general/common.txt --hc 404 http://testphp.vulnweb.com/FUZZ
```

Where the `—hc` flag stands for which status code mean an error. If you see that you are constantly getting 500 error codes, wfuzz will not block them by default and you'd have to change the command

```
$ wfuzz -w wordlist/general/common.txt --hc 404,500 http://testphp.vulnweb.com/FUZZ
```