DAY4_advent_of_the_cyber

Introduction & Story:

We're going to be taking a look at some of the fundamental tools used in web application testing. You're going to learn how to use Gobuster to enumerate a web server for hidden files and folders to aid in the recovery of Elf's forums. Later on, you're going to be introduced to an important technique that is fuzzing, where you will have the opportunity to put theory into practice.

Our malicious, despicable, vile, cruel, contemptuous, evil hacker has defaced Elf's forums and completely removed the login page! However, we may still have access to the API. The sysadmin also told us that the API creates logs using dates with a format of YYYYMMDD.

What is Fuzzing?

To keep it simple, fuzzing can be argued as "fancy bruteforcing" to some degree. However, you can fuzz what you can't bruteforce. Fuzzing is using security tools to automate the input of data we provide into things such as websites or software applications. Fuzzing is an extremely effective process as computers can perform laborious actions like trying to find hidden files/folders, try different username and passwords much quicker then a human can (and is willing to do...)

Poorly built applications are often unable to handle data the way it is supposed to under intense load. Moreover, the data we're parsing to the application may be interpreted and executed (instead of being handled correctly i.e. system commands). We can use fuzzing to cause the application to trigger what's known as an error condition where this may be abused by a penetration tester or a bug bounty hunter.

An Introduction to Using Gobuster

Logically speaking, there are many pieces to a website that the average user doesn't see. They can be anything from a sitemap to a secret directory which contains important files. Unfortunately, this can cause developers to get a bit lazy, and not protect these directories, allowing anyone who finds out that they exist to steal the important data. gobuster is the tool that helps us discover these valuable directories if they exist. The idea behind the tool itself is simple, bruteforcing common paths to check if it's valid. Similar to how you would in your browser, albeit this tool is much, much quicker. Gobuster has three modes: `dir`, `vhost` and `dns`.

For the sake of today, we're going to be using gobuster in `dir` mode, as this is the most likely mode that you'll be using day-to-day. `dir` (short for directory) can be selected by using `gobuster dir <rest of command>`

Let's use the table below to illustrate how wordlists work:

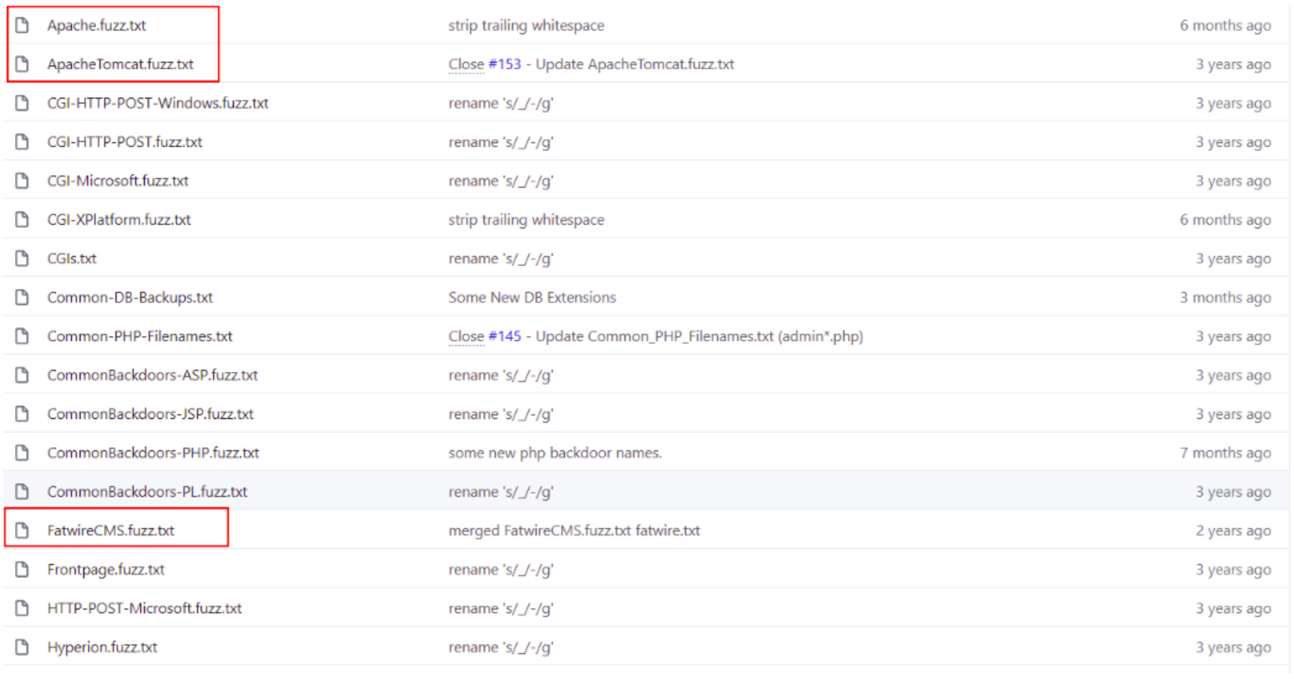
Original URL	Item in Wordlist	Final URL
http://example.com	backups	http://example.com/backups
http://loveucmnatic.thm	shepards	http://loveucmnatic.thm/shepards

Gobuster has a few other little tricks, it supports appending extensions which means you can bruteforce files as well. We can use another handy little chart to visualize it and show an example later on:

Original URL	Item in Wordlist	Specified Extension	Final URL
http://example.com	backup	php	http://example.com/backup.php
http://example.com	backup	txt	http://example.com/backup.txt
http://example.com	icecream	html	http://example.com/icecream.html

To find the data in the table above, we have used a command such as the following (assuming wordlist.txt had the words "backup" and "icecream"): `gobuster dir -u http://example.com -w wordlist.txt -x php,txt,html`.

Whilst Gobuster is considerably faster than alternatives such as Dirbuster on Kali Linux, it is still limited to the wordlists and options that you provide. The more appropriate your wordlist is to the target, the better your results will be. Wordlists such as SecLists have wordlists for specific applications and platforms. You can use the information gathered from enumerating to help determine what wordlist may be the most appropriate to use. Although, we will come onto refining those skills in a later day.



Apache.fuzz.txt	strip trailing whitespace	6 months ago
ApacheTomcat.fuzz.txt	Close #153 - Update ApacheTomcat.fuzz.txt	3 years ago
CGI-HTTP-POST-Windows.fuzz.txt	rename 's/_/-/g'	3 years ago
CGI-HTTP-POST.fuzz.txt	rename 's/_/-/g'	3 years ago
CGI-Microsoft.fuzz.txt	rename 's/_/-/g'	3 years ago
CGI-XPlatform.fuzz.txt	strip trailing whitespace	6 months ago
CGIs.txt	rename 's/_/-/g'	3 years ago
Common-DB-Backups.txt	Some New DB Extensions	3 months ago
Common-PHP-Filenames.txt	Close #145 - Update Common_PHP_Filenames.txt (admin*.php)	3 years ago
CommonBackdoors-ASP.fuzz.txt	rename 's/_/-/g'	3 years ago
CommonBackdoors-JSP.fuzz.txt	rename 's/_/-/g'	3 years ago
CommonBackdoors-PHP.fuzz.txt	some new php backdoor names.	7 months ago
CommonBackdoors-PL.fuzz.txt	rename 's/_/-/g'	3 years ago
FatwireCMS.fuzz.txt	merged FatwireCMS.fuzz.txt fatwire.txt	2 years ago
Frontpage.fuzz.txt	rename 's/_/-/g'	3 years ago
HTTP-POST-Microsoft.fuzz.txt	rename 's/_/-/g'	3 years ago
Hyperion.fuzz.txt	rename 's/_/-/g'	3 years ago

Gobuster itself works like any other Linux tool, meaning it has an online man page available here, which you can use as a reference if you want to learn more about the various options that gobuster supports. However, let's detail a few of the common options below:

Options	Description
-u	Used to specify which url to enumerate
-w	Used to specify which wordlist that is appended on the url path i.e "http://url.com/word1" "http://url.com/word2" "http://url.com/word3.php"
-x	Used to specify file extensions i.e "php,txt,html"

Recommended wordlist to use: big.txt

We have provided wordlists for you on the AttackBox located in "/usr/share/wordlists". This is also the default location for wordlists on pentesting distributions such as Kali Linux. To provide an example, "big.txt" is located at the file path "/usr/share/wordlists/dirb/big.txt". Take some time to explore other wordlists provided and think of the situation where they may be effective to use.

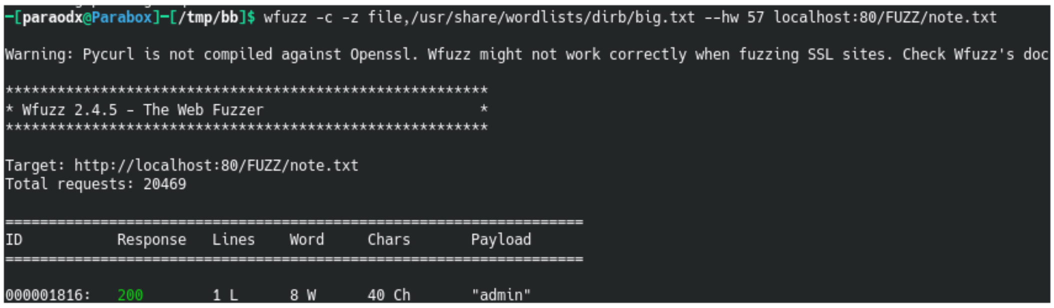
An Introduction to Using wfuzz

The premise behind wfuzz is simple. Occasionally you want a bit more information about how much data something within a web application returns. This could be anything from a file, a response code (i.e. 404 meaning the URL doesn't exist) or the parameters used in a form similar to the form you attacked in Day 2.

For example, let's say you are pentesting a note-taking application and you want to see if you can view notes by other users. One way you may want to achieve this is by *FUZZ*ing for usernames (with the knowledge that every valid user will have note.txt by default). Our wfuzz command would like the following: `wfuzz -c -z file,/usr/share/wordlists/dirb/big.txt localhost:80/FUZZ/note.txt`

Now wfuzz will query the webserver using the words iterated from the "big.txt" wordlist. To illustrate:

- Query #1: http://localhost/**admin**/note.txt
- Query #2: http://localhost/**administrator**/note.txt
- Query #3: http://localhost/**system**/note.txt



```
-[paraodx@Parabox]-[/tmp/bb]$ wfuzz -c -z file,/usr/share/wordlists/dirb/big.txt --hw 57 localhost:80/FUZZ/note.txt

Warning: Pycurl is not compiled against Openssl. Wfuzz might not work correctly when fuzzing SSL sites. Check Wfuzz's doc

********************************************************
* Wfuzz 2.4.5 - The Web Fuzzer                         *
********************************************************

Target: http://localhost:80/FUZZ/note.txt
Total requests: 20469

===================================================================
ID           Response   Lines    Word     Chars       Payload
===================================================================

000001816:   200        1 L      8 W      40 Ch       "admin"
```

Note how the "**FUZZ**" parameter is being replaced with the words from the wordlist. We'll outline some of the options that can be configured in wfuzz, however, it's worth knowing that will display results that are different to the parameters that we set. In the picture above we used the `--hw` option to hide all pages that have 57 words on them. Since wfuzz found a URL with only 8 words, it'll be displayed to us, as this is not 57 words.

- It is important to know that you can *FUZZ* any part of the URL, meaning that you can test any parameters if you don't know them as well.

As with any Linux-based tool, wfuzz also has a useful manpage here, that details some of the more advanced options available to you. Although, I have added some of the more useful options into the table below:

Options	Description
-c	Shows the output in color
-d	Specify the parameters you want to fuzz with, where the data is encoded for a HTML form
-z	Specifies what will replace FUZZ in the request. For example `-z file,big.txt`. We're telling wfuzz to look for files by replacing "FUZZ" with the words within "big.txt"
--hc	Don't show certain http response codes. I.e. Don't show **404** responses that indicate the file *doesn't* exist, or "**200**" to indicate the file *does* exist
--hl	Don't show for a certain amount of lines in the response
--hh	Don't show for a certain amount of characters

Let's bring this together and demonstrate some of these options. Let's say we wanted to fuzz an application on *http://shibes.thm/login.php* to find the correct credentials to the login form. After recalling our knowledge from Day 2, we know all about URL parameters! We can take a bit of a guess as to what parameters the login form may be using `username` and `password`, right? Worth a try! Our wfuzz command would look like so: `wfuzz -c -z file,mywordlist.txt -d “username=FUZZ&password=FUZZ” -u http://shibes.thm/login.php`

Where wfuzz will now iterate through the wordlist we provided and replace the "**FUZZ**" values specified in the "**username**" and "**password**" parameters.

Challenge

Deploy both the instance attached to this task (the green deploy button) and the AttackBox by pressing the blue "Start AttackBox" button at the top of the page. After allowing 5 minutes, navigate to the website (10.10.207.225) in your AttackBox browser.

It is up to you to decide if you wish to create the wordlist yourself or use a larger wordlist located in `/opt/AoC-2020/Day-4/wordlist` on the AttackBox. The wordlist is also available for download if you are using your own machine.

In summary, use the tools and techniques outlined in today's advent of cyber; search for the API, find the correct post and bring back Elf's forums!

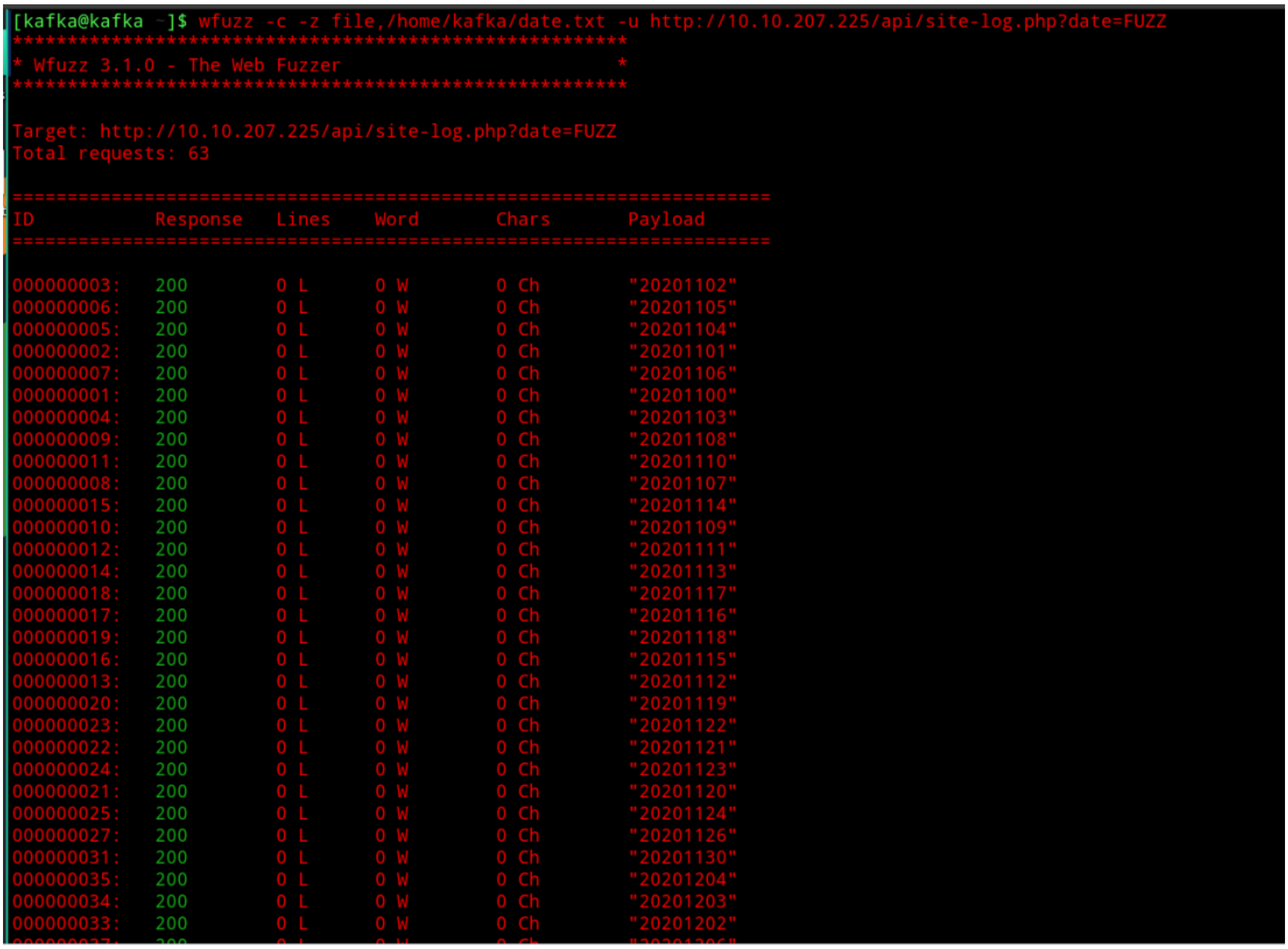
How to approach the challenge

Since we know there's theoretically an API directory we can use gobuster to enumerate the website and see if we can find anything. Then assuming we do find something, we should investigate it for interesting files. Let's say we then find what seems to hold the logs, we know we're searching by date, so we can infer that there's a good chance that we'll be using the date parameter to interact with the API. We also know that the API takes a date in the form of YYYYMMDD. A wordlist in that format can be found in the hint for this task, although if you want an extra challenge, you can try and build a wordlist in that format yourself.

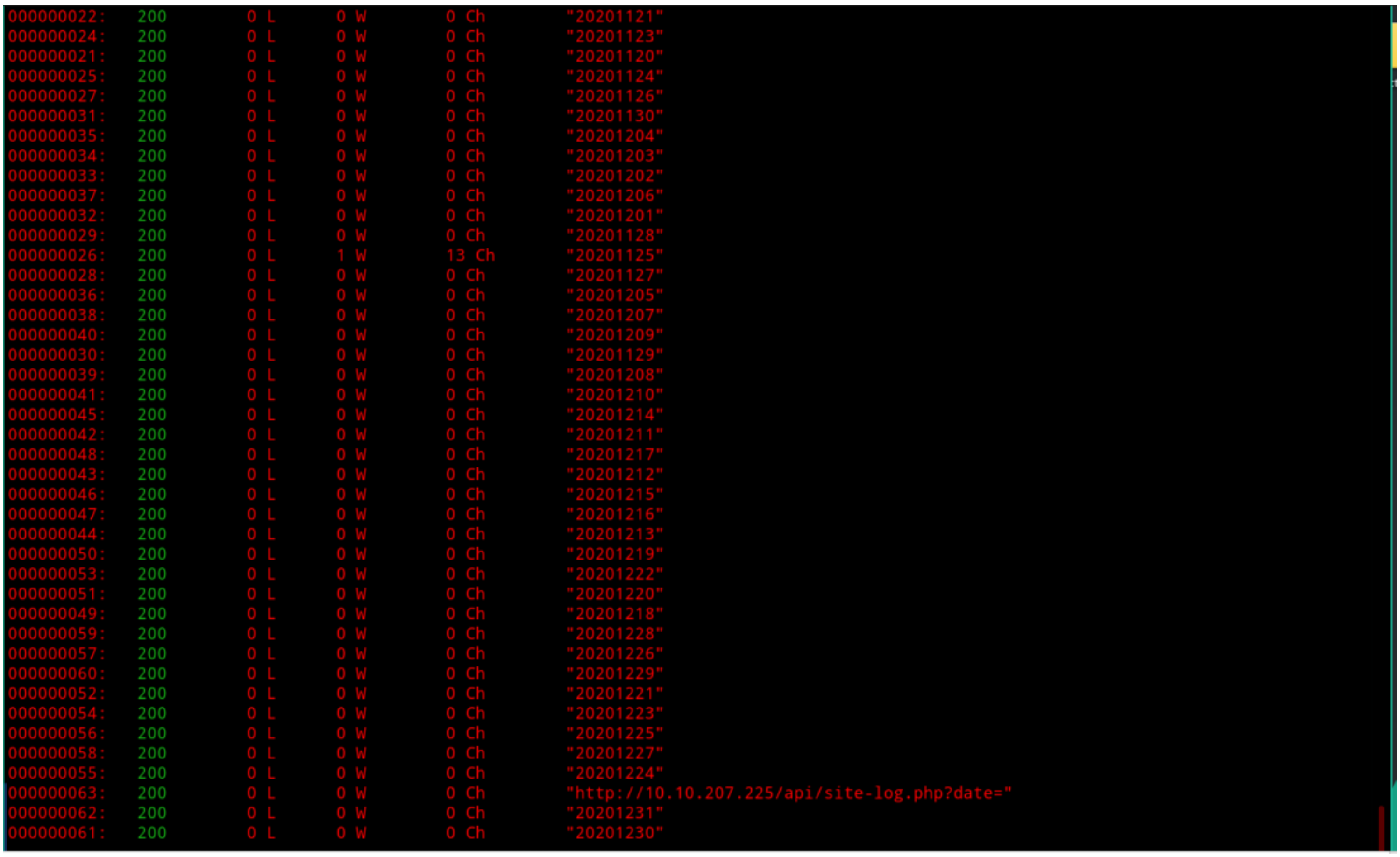
Finally, API's may not return data if the proper parameters aren't passed, so with that knowledge, we can use the options in wfuzz to filter out parameters that don't return anything.

With all that in mind, we should be able to get a flag.

Recommended Rooms:

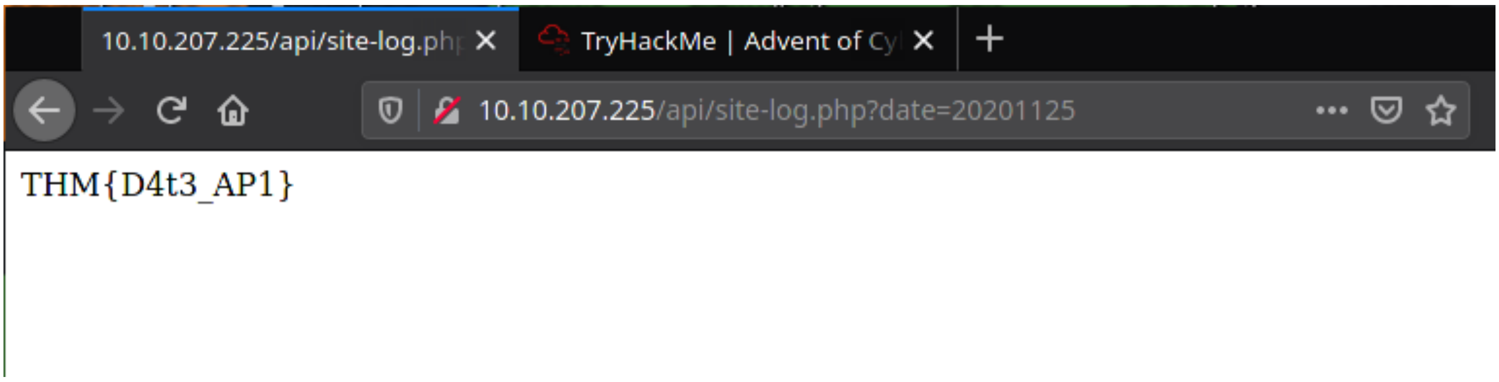


```
[kafka@kafka ~]$ wfuzz -c -z file,/home/kafka/date.txt -u http://10.10.207.225/api/site-log.php?date=FUZZ
********************************************************
* Wfuzz 3.1.0 - The Web Fuzzer                         *
********************************************************

Target: http://10.10.207.225/api/site-log.php?date=FUZZ
Total requests: 63

=====================================================================
ID           Response   Lines    Word       Chars       Payload
=====================================================================

000000003:   200        0 L      0 W        0 Ch        "20201102"
000000006:   200        0 L      0 W        0 Ch        "20201105"
000000005:   200        0 L      0 W        0 Ch        "20201104"
000000002:   200        0 L      0 W        0 Ch        "20201101"
000000007:   200        0 L      0 W        0 Ch        "20201106"
000000001:   200        0 L      0 W        0 Ch        "20201100"
000000004:   200        0 L      0 W        0 Ch        "20201103"
000000009:   200        0 L      0 W        0 Ch        "20201108"
000000011:   200        0 L      0 W        0 Ch        "20201110"
000000008:   200        0 L      0 W        0 Ch        "20201107"
000000015:   200        0 L      0 W        0 Ch        "20201114"
000000010:   200        0 L      0 W        0 Ch        "20201109"
000000012:   200        0 L      0 W        0 Ch        "20201111"
000000014:   200        0 L      0 W        0 Ch        "20201113"
000000018:   200        0 L      0 W        0 Ch        "20201117"
000000017:   200        0 L      0 W        0 Ch        "20201116"
000000019:   200        0 L      0 W        0 Ch        "20201118"
000000016:   200        0 L      0 W        0 Ch        "20201115"
000000013:   200        0 L      0 W        0 Ch        "20201112"
000000020:   200        0 L      0 W        0 Ch        "20201119"
000000023:   200        0 L      0 W        0 Ch        "20201122"
000000022:   200        0 L      0 W        0 Ch        "20201121"
000000024:   200        0 L      0 W        0 Ch        "20201123"
000000021:   200        0 L      0 W        0 Ch        "20201120"
000000025:   200        0 L      0 W        0 Ch        "20201124"
000000027:   200        0 L      0 W        0 Ch        "20201126"
000000031:   200        0 L      0 W        0 Ch        "20201130"
000000035:   200        0 L      0 W        0 Ch        "20201204"
000000034:   200        0 L      0 W        0 Ch        "20201203"
000000033:   200        0 L      0 W        0 Ch        "20201202"
```



```
000000022:   200        0 L      0 W      0 Ch       "20201121"
000000024:   200        0 L      0 W      0 Ch       "20201123"
000000021:   200        0 L      0 W      0 Ch       "20201120"
000000025:   200        0 L      0 W      0 Ch       "20201124"
000000027:   200        0 L      0 W      0 Ch       "20201126"
000000031:   200        0 L      0 W      0 Ch       "20201130"
000000035:   200        0 L      0 W      0 Ch       "20201204"
000000034:   200        0 L      0 W      0 Ch       "20201203"
000000033:   200        0 L      0 W      0 Ch       "20201202"
000000037:   200        0 L      0 W      0 Ch       "20201206"
000000032:   200        0 L      0 W      0 Ch       "20201201"
000000029:   200        0 L      0 W      0 Ch       "20201128"
000000026:   200        0 L      1 W      13 Ch      "20201125"
000000028:   200        0 L      0 W      0 Ch       "20201127"
000000036:   200        0 L      0 W      0 Ch       "20201205"
000000038:   200        0 L      0 W      0 Ch       "20201207"
000000040:   200        0 L      0 W      0 Ch       "20201209"
000000030:   200        0 L      0 W      0 Ch       "20201129"
000000039:   200        0 L      0 W      0 Ch       "20201208"
000000041:   200        0 L      0 W      0 Ch       "20201210"
000000045:   200        0 L      0 W      0 Ch       "20201214"
000000042:   200        0 L      0 W      0 Ch       "20201211"
000000048:   200        0 L      0 W      0 Ch       "20201217"
000000043:   200        0 L      0 W      0 Ch       "20201212"
000000046:   200        0 L      0 W      0 Ch       "20201215"
000000047:   200        0 L      0 W      0 Ch       "20201216"
000000044:   200        0 L      0 W      0 Ch       "20201213"
000000050:   200        0 L      0 W      0 Ch       "20201219"
000000053:   200        0 L      0 W      0 Ch       "20201222"
000000051:   200        0 L      0 W      0 Ch       "20201220"
000000049:   200        0 L      0 W      0 Ch       "20201218"
000000059:   200        0 L      0 W      0 Ch       "20201228"
000000057:   200        0 L      0 W      0 Ch       "20201226"
000000060:   200        0 L      0 W      0 Ch       "20201229"
000000052:   200        0 L      0 W      0 Ch       "20201221"
000000054:   200        0 L      0 W      0 Ch       "20201223"
000000056:   200        0 L      0 W      0 Ch       "20201225"
000000058:   200        0 L      0 W      0 Ch       "20201227"
000000055:   200        0 L      0 W      0 Ch       "20201224"
000000063:   200        0 L      0 W      0 Ch       "http://10.10.207.225/api/site-log.php?date="
000000062:   200        0 L      0 W      0 Ch       "20201231"
000000061:   200        0 L      0 W      0 Ch       "20201230"
```

using the parameter whose chars length is "13 Ch"



```
10.10.207.225/api/site-log.php?date=20201125
```

THM{D4t3_AP1}