Penetration Testing

Comprehensive Guide on ffuf

March 28, 2021 By Raj Chandel

In this article, we will learn how we can use ffuf, which states for "Fuzz Faster U Fool", which is an interesting open-source web fuzzing tool. Since its release, many people have gravitated towards ffuf, particularly in the bug bounty scenario. So, let's dive into this learning process.

Table of Content

Introduction to ffuf

It is a professional command-line method for web fuzzing on a web server and the credit goes to the author (**@joohoi)**. Many people have gravitated towards ffuf since its release, especially in the bug bounty scene. While the bulk of this shift is possibly attributable to the herd mentality, a significant portion of the group has made the switch due to FFUF's tempo, versatility, and capacity to easily merge with external tooling.

Setup

It is a command-line program that runs in the Linux Terminal or the Windows Command Prompt. Upgrading from the source is not any more difficult than compiling from the source, with the exception of the inclusion of the **-u flag**.

```
go get -u github.com/ffuf/ffuf
```

Due to the fact we are using Kali Linux, we'll find ffuf in the apt repositories, allowing us to install by running the simple command.

```
apt install ffuf
```

After installing this tool, to get its working parameters and options all we need is just to use **[-h]** parameter for the help option.

```
ffuf -h
```

Input Options

These are parameters that help us to provide the required data for web fuzzing over a URL with the help of a world list.

1. Simple Attack

For the default attack, we need to use parameters **[-u]** for the target URL and **[-w]** to load a wordlist as shown in the image.

ffuf -u http://testphp.vulnweb.com/FUZZ/ -w dict.txt

After running the command, let's focus on the results.

- Firstly we noticed that it is by default running on **HTTP method** GET.
- The next things are **response code status** [200, 204, 301, 302, 307, 401, 403, 405}; it also shows the progression of our attack. At the end of the progress, we got our results.

2. Multiple Wordlists:

Sometimes one wordlist isn't sufficient to show us our desired results. In that case, we case put multiple wordlists at once to get better results. Only ffuf has the ability to run as many wordlists as per our need for attack.

Here I provided two dictionaries dict.txt as W1 & W2 as Dns.txt and fuff will read both dictionary simultaneously.

ffuf -u https://ignitetechnologies.in/W2/W1/ -w dict.txt:W1 -w dns_dict.txt:W2

3. Ignore Wordlist Comment and Silent:

Generally, the default wordlist might have some comments that can affect our result accuracy. In this case, we can use **[-ic]** parameter that can help us to get rid of that comment. Sometimes we need to be more focused on attack rather than tools banners for this kind of accuracy we need **[-s]** parameter which has the power to remove the banner of the tool.

```
ffuf -u http://testphp.vulnweb.com/FUZZ/ -w dict.txt
```

we can clearly see some comments are listed in the result when we have run above the command and after using **[-s]** & **[-ic]** parameters in the next command the comments and banner are removed.

```
ffuf -u http://testphp.vulnweb.com/FUZZ/ -w dict.txt -ic -s
```

4. Extensions:

We can search for a specific extension file on a web server with the help of **[-e]** parameter, all we need to just to specify the extension file along with **[-e]** parameter. To get these results we just need to follow the command.

```
ffuf -u http://192.168.1.12/dvwa/FUZZ/ -w dict.txt -e .php
```

5. Request | Request-Proto | Mode:

Burp Suite is an advanced framework for conducting web application security monitoring. Its different instruments act in agreement to help the testing process as a whole. A cluster bomb is a feature that uses several payload sets. For each given location, there is a different payload package. the attack goes through each payload package one by one, checking all potential payload variations.

There is a various parameter of this tool, which help to use this our scenario. Like **[-request]** parameter which can use our request in the attack, **[-request-proto]** parameter through which we can define our parameter, **[-mode]** parameter help us to define the mode of attack.

First of all, we use random credentials on our targeted URL page and set proxy up to capture its request in intercept mode on Burpsuite.

Now in the intercept tab of the Burpsuite, change our provided credential with **HFUZZ** and **WFUZZ**. Put HFUZZ in front of **uname** and WFUZZ in front of the **pass**. Then copy-paste this request in a text and name as per your desire. In our case, we named that brute.txt.

Now proceed towards the main attack, where **[-request]** parameter hold our request text file. **[-request-proto**] help us derive the http prototype **[-mode]** help us to derive us cluster bomb attack. The wordlists we use in these (users.txt and pass.txt) consist of SQL injections. Follow this command start attacking using these parameters.

```
ffuf -request brute.txt -request-proto http -mode clusterbomb -w users.txt:
```

as we can see in our attack results, we have successfully found out SQL injections working on that particular target.

Match Options

If we want ffuf to show only that data which is important in our web fuzzing data. Then it will help us to showcase only matched according to the parameter. Example: HTTP code, Lines, Words, Size and Regular Expressions.

1. Match HTTP Code:

To get an understanding of this parameter we need to consider a simple attack where we can see which HTTP codes are appearing in our results.

```
ffuf -u http://192.168.1.12/dvwa/FUZZ/ -w dict.txt
```

we can clearly see that it showing some 302 HTTP code along with 200 HTTP code.

If only need successful results like 200 HTTP code we just need to use **[-mc]** parameter along with our specific HTTP code. To use this parameter just follow the command.

```
ffuf -u http://192.168.1.12/dvwa/FUZZ/ -w dict.txt -mc 200
```

2. Match Lines:

Like the match code which we discussed earlier, it gives us the result for a specific-lines in a file with the help of **[-ml]** parameter. We can use this **[-ml]** parameter by specifying the lines we need in a file.

```
ffuf -u http://192.168.1.12/dvwa/FUZZ/ -w dict.txt -ml 15
```

3. Match Words:

Similarly, as the above functionalities match function it can provide us with a result with a specific word count. To get this result we need to use **[-mw]** parameter along specific words count we want in our results.

```
ffuf -u http://192.168.1.12/dvwa/FUZZ/ -w dict.txt -mw 53
```

4. Match Size:

Similarly, as the above functionalities match function it can provide us with a result with the size of the file. We can use **[-ms]** parameter along with the specific size count we want in our result.

```
ffuf -u http://192.168.1.12/dvwa/FUZZ/ -w dict.txt -ms 2929
```

Match Regular Expression:

It is the last of all match functions available in this tool. We are going to fuzz for LFI by matching the string with followed pattern “**root:x**” for the given dictionary.

We are using a URL that can achieve this functionality and by using **[-mr]** parameter we define the matching string “**root:x**”.

This our special wordlist looks like.

By using this wordlist, follow the below command to use **[-mr]** parameter in an attack scenario.

```
ffuf -u http://testphp.vulnweb.com/showimage.php?file=FUZZ -w dict2.txt -mr
```

Here we got HTTP to respond 200 for /etc/passwd for the given wordlist.

Filter Options

The Filter options are absolutely opposite to Match options. We can use these options to remove the unwanted from our web fuzzing. Example: HTTP Code, Lines, Words, Size, Regular Expressions.

1. Filter Code:

The **[-fc]** parameter need the specific HTTP status code we want to remove from the result.

```
ffuf -u http://192.168.1.12/dvwa/FUZZ/ -w dict.txt -fc 302
```

2. Filter Lines:

The **[-fl]** parameter has the ability to remove a specific length from our result or we can filter it out from our attack.

```
ffuf -u http://192.168.1.12/dvwa/FUZZ/ -w dict.txt -fl 26
```

3. Filter Size:

The **[-fs]** parameter has the ability to filter out the specified size is described by us during the command of the attack.

```
ffuf -u http://192.168.1.12/dvwa/FUZZ/ -w dict.txt -fs 2929
```

4. Filter Words

The **[-fw]** parameter has the ability to filter out the words count from results that we want to remove.

```
ffuf -u http://192.168.1.12/dvwa/FUZZ/ -w dict.txt -fw 83
```

5. Filter Regular Expression:

The parameter **[-fr]** we can remove a specific regular expression, here we try to exclude the log file from the output result.

```
ffuf -u http://192.168.1.12/dvwa/FUZZ/ -w dict.txt -fr "log"
```

General Options

These are the general parameters of this tool, which revolves around its general working on web fuzzing.

1. Custom Auto Calibration:

We know that the power of a computer or machine to automatically calibrate itself is known as auto-calibration. Calibration is the process of providing a measuring instrument with the information it requires to understand the context in which it will be used. When gathering data, calibrating a computer ensures its accuracy.

We can customize this feature according to our need with the help of **[-acc]** parameter. Which can't be used without **[-ac]** parameter for its customization.

```
ffuf -u http://192.168.1.12/dvwa/FUZZ/ -w dict.txt -acc -ac -fl 26 -ac -fs
```

2. Color:

Sometimes separation of colour creates extra attention to all details having in results. This **[-c]** parameter helps to create colour separation.

```
ffuf -u http://192.168.1.12/dvwa/FUZZ/ -w dict.txt -c
```

3. Maxtime For Task:

If you want to fuzz for a limited amount of time then you can choose **[-maxtime]** parameter. Follow the command to provide a timeslot.

```
ffuf -u http://192.168.1.12/dvwa/FUZZ/ -w dict.txt -maxtime 5
```

4, Maxtime For Job:

With the help of **[-maxtime-job]** parameter, we can put a time limit for a particular job. By using this command, we are trying to limit the time per job or request execution.

ffuf -u http://192.168.1.12/dvwa/FUZZ/ -w dict.txt -maxtime-job 2

5. Delay:

If we create a particular delay in each request offered by the attack. Through this feature, a request has a better opportunity to get better results. The **[-p]** parameter help us to achieve delay in those requests.

ffuf -u http://192.168.1.12/dvwa/FUZZ/ -w dict.txt -p 1

6. Request Rate:

We can create a separate request rate for each of our attack with the help of the **[-rate]** parameter. Through this parameter, we create our request per second as per our attack desired.

```
ffuf -u http://192.168.1.12/dvwa/FUZZ/ -w dict.txt -rate 500
```

7. Error Functions:

There are three parameters that support the Error function. The first parameter is **[-se]**, which is a spurious error. It states that the following request is genuine or not. The second parameter is **[-sf]**, it will stop our attack when more than 95% of requests occurred as an error. The third and final parameter is **[-sa]**, which is a combination of both the error parameter.

In our scenario, we are using **[-se]** parameter where it will stop our attack when our request is not real.

```
ffuf -u http://ignitetechnologies.in/W2/W1/ -w dict.txt:W1 -w dns_dict.txt:
```

8. Verbose Mode:

As we all know, the verbose mode is a feature used in many computers operating systems and programming languages that provide extra information on what the computer is doing and what drivers and applications it is loading at initialization. In programming, it produces accurate output for debugging purposes, making it easy to debug a program. There is a parameter called **[-v]** parameter.

```
ffuf -u http://192.168.1.12/dvwa/FUZZ/ -w dict.txt -v
```

9. Threads:

The **[-t]** parameter is used to speed up or slow down a process. By default, it is set on 40. if we want to pace up the process, we need to increase its number, vice versa to slow down process.

```
ffuf -u http://192.168.1.12/dvwa/FUZZ/ -w dict.txt -t 1000
```

Output Options

We save the performance of our attacks for the purposes of record-keeping, improved readability, and potential references. We use **[-o]** parameter to save our output, but we need to specify its format with **[-of]** parameter together.

1. Output Format in HTML:

We use **[-of]** parameter and this defining with an HTML format. By using the command, we can create our report in html.

```
ffuf -u http://192.168.1.12/dvwa/FUZZ/ -w dict.txt -o file.html -of html
```

Now after completion of this attack, we need to check our output file is up to that mark or not. As we can see that our file is successfully created.

2. Output Format in CSV:

Similarly, we just need to csv format along with **[-of]** parameter. Where csv is a comma-separated values, which file allows you to store data in a tabular format.

```
ffuf -u http://192.168.1.12/dvwa/FUZZ/ -w dict.txt -o file.csv -of csv
```

Now after completion of this attack, we need to check our output file is up to that mark or not. As we can see that our file is successfully created.

3. All Output Format:

Similarly, if we want all output format at once just use **[-of all]** parameter. Like json, ejson, html, md, csv, ecsv. Follow this command to generate all reports at once.

```
ffuf -u http://192.168.1.12/dvwa/FUZZ/ -w dict.txt -o output/file -of all
```

Now after completion of this attack, we need to check our output files is up to that mark or not. As we can see that our all files are successfully created.

HTTP Options

The options move around HTTP options, sometimes it required the details to run web fuzzing Like HTTP request, Cookie, HTTP header, etc.

1. Timeout:

Timeout act as a deadline for the event. The **[-timeout]** parameter help of established this feature with ease, follow this command to run this parameter.

```
ffuf -u http://192.168.1.12/dvwa/FUZZ/ -w dict.txt -timeout 5
```

2. Host Header:

If we want to perform fuzzing on subdomain, we can use **[-H]** parameter along with a domain name wordlist as given below in the command.

```
ffuf -u https://google.com -w dns_dict.txt -mc 200 -H "HOST: FUZZ.google.co
```

3. Recursion:

Recursion is the mechanism of repeating objects in a self-similar manner, as we all know. If a program requires you to access a function within another function, this is referred to as a recursive call of the function. By using **[-recursion]** parameter, we can achieve this functionality in our attacks.

```
ffuf -u "http://testphp.vulnweb.com/FUZZ/" -w dict.txt -recursion
```

3. Attack with Cookie:

Sometimes web fuzzing does not show the result on authenticated site without authentication. There is a **[-b]** parameter through which we can achieve your goal by providing a session cookie.

```
ffuf -u http://192.168.1.12/dvwa/FUZZ/ -w dict.txt -b "PHPSESSID:"7aaaa6d88
```

3. Replay-Proxy:

As you might be aware, there are speed restrictions when using the Intruder function in the free version of the Burp suite (Community Edition). The Intruder attack has been severely slowed, with each order slowing the attack even further.

In our case we are using Burp suite proxy to get results for evaluation in it. First, we have to establish a localhost proxy on port 8080.

Now use **[-replay-proxy]** parameter, which helps us to derive our local host proxy which we established in the previous step on port 8080 along with our attack.

```
ffuf -u http://192.168.1.12/dvwa/FUZZ/ -w dict.txt -replay-proxy http://127
```

This attack will show our results on two platforms. The first platform on the kali terminal and the second on the Burp suite HTTP history tab. Through these various techniques, we can better understand our target and our attack results.

Conclusion

The ffuf is often compared to tools like dirb or dirbuster, which, although accurate to certain extents, isn't a reasonable analogy. Although FFUF can be used to brute force files, its true strength lies in its simplicity, and a better comparative tool for FFUF would be anything like Burp Suite Intruder or Turbo Intruder.

Author: Shubham Sharma is a passionate Cybersecurity Researcher, contact **LinkedIn** and **Twitter**.

FACEBOOK TWITTER PINTEREST LINKEDIN

PREVIOUS POST
Comprehensive Guide to AutoRecon

NEXT POST
Wordlists for Pentester

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment *

Name *

Email *

Website

Notify me of follow-up comments by email.

Notify me of new posts by email.

Post Comment

Search

Search ... Search

Subscribe To Blog Via Email

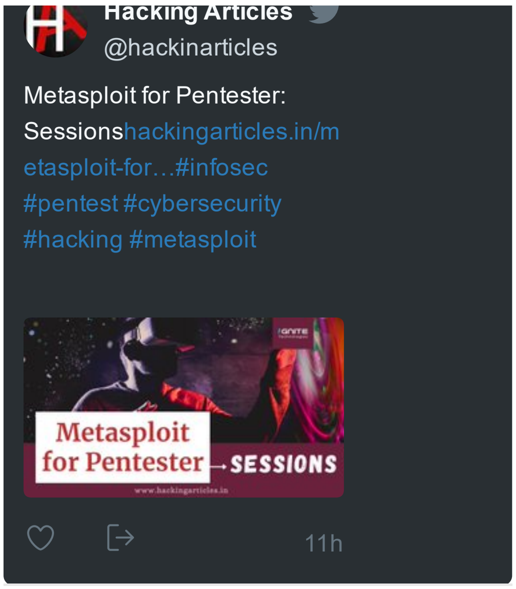
Email Address

Subscribe

Join Our Training Programs

IGNITE Technologies
JOIN OUR CYBER SECURITY TRAINING PROGRAMS
Enroll Today
www.ignitetechnologies.in

Follow Me On Twitter


Hacking Articles
@hackinarticles

Metasploit for Pentester: Sessionshackingarticles.in/metasploit-for…#infosec #pentest #cybersecurity #hacking #metasploit

11h

Categories

Cryptography & Stegnography

CTF Challenges

Cyber Forensics

Database Hacking

Footprinting

Hacking Tools

Kali Linux

Nmap

Others

Password Cracking

Penetration Testing

Pentest Lab Setup

Privilege Escalation

Red Teaming

Social Engineering Toolkit

Uncategorized

Website Hacking

Window Password Hacking

Wireless Hacking

Wireless Penetration Testing

Articles

Select Month ▼

You may like

MSSQL for Pentester: Abusing Linked Database

September 11, 2021

MSSQL for Pentester: Abusing Trustworthy

September 7, 2021