UDC 004.912

Comparison of Tools for Web-Application Brute Forcing

Oleksii Chalyi[1], Myhailo Kolomytsev[1]

[1] *National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute" Institute of Physics and Technology, 37, Prosp. Peremohy, Kyiv, 03056, Ukraine*

Abstract

For a long time, threat of web-application authentication break remained a problem not only for users but also for business. This threat still exists, since broken authentication provides a blackhat with full access to accounts of users and business data. This article analyzes software tools for breaking authentication as well as it defines time required for breaking-in depending on different conditions. Based on the results of the analysis, the fastest tool was determined. In order to complete this analysis and determine the fastest tool, own web-application was created.

Keywords: tools comparison, software analysis, web-application, authentication, brute force, dictionary attack, Hydra, Wfuzz, Patator, Legion, Kali Linux

Introduction

In 21th century, technologies are becoming more powerful and more functional. As a result, demand to obtaining secret and personal information in unsanctioned way appears. According to cyberattacks statistics, their main objective is stealing money or very costly and important information.

Money is economics category, which might force anyone to unacceptable actions. It is possible to get money illegally in two ways. Direct - means getting access to the user's bank account. Or Indirect – to force victim to pay using blackmail upon breaking into account and getting some secret information. Both these ways are part of Broken Authentication, also known as Identification and Authentication Failures, attack on users accounts. This attack take 7-th place in OWASP TOP – 10:2021 ranking [1]. The most widespread attacks in this category are bruteforce and dictionary attacks on users accounts.

The purpose of survey was determining the fastest tools for web-application user accounts credentials brute forcing.

The solution consists from the following:

- Modern tools analysis for brute forcing web-application user accounts credentials
- Experiments plan determining
- Conducting experiments and analysis of results

1. Analysis of modern tools for brute forcing of web-application user accounts credentials

There are a lot of tools for brute forcing web-application user accounts credentials. These tools are usually free, in open access and working on different operating systems. But is there a difference between them? We will investigate four tools that are able to brute force logins, passwords and login-password pair. All of them are free and can be easily installed in any Linux system. This is the list of the tools:

- Hydra
- Wfuzz
- Patator
- Legion

1.1. Hydra

Hydra is the first tool analyzed. According to the official Hydra website [2] this tool was developed in 2001 by van Hauser, the founder of THC (The Hacker's Choice) company. Despite the fact that it was created in 2001, it is still actively used nowadays, because it is really powerful software. On GitHub, the first upload was made in 2016 with Hydra v8.2. This tool is free access and you can see all its possibilities on its official website. It is free and can be installed by anyone from the Internet. Hydra is pre-installed in Linux distributives that have information security direction. For example, Hydra is pre-installed in Kali Linux and Parrot Security OS. Moreover, it can be installed in any Unix system, due to officially stated compatibility.

Hydra is cross platforming. This means that we can install it in Windows, MacOS as well as in Linux systems. Moreover, it can even be installed in Android and Iphone mobile systems.

In total Hydra supports 58 protocols, but if different versions of SSH and SNMP are included that it supports 61 protocols. Most important protocols for us are the following:

- FTP
- SSH
- Telnet
- HTTPS-POST-FORM
- HTTP-POST-FORM

Hydra has GUI interface, also known as software XHydra. We can use these tools not only for brute forcing web-application user's accounts credentials but also for breaking authentication in other supported protocols. Hydra official website presents a disclaimer about its usage, it must be used only for legal purposes. We are following these rules, since testing the software on our own web-application.

1.1.1. Finding password with login available

In Figure 1 we can see how we can find passwords when we know login. For this we need to use the following parameters:

1. -l to set login that we know
2. -P to set folder with password wordlists
3. -t to set number of threads
4. -f to stop when the first password will be found
5. http-post-form to set http website ip
6. -m to set additional settings

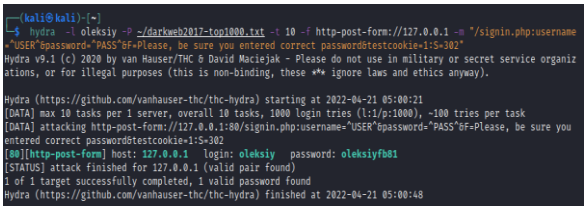

Figure 1: Using Hydra for finding the password

Here we can see that it took 27 seconds to find a password.

1.1.2. Finding login with password available

In Figure 2 we can see how we can find logins when we know password. For this, we need use the following parameters:

1. -L to set folder with login wordlists
2. -P to set password that we know
3. -t to set number of threads
4. -f to stop when the first login will be found
   http-post-form to set http website ip
5. -m to set additional settings

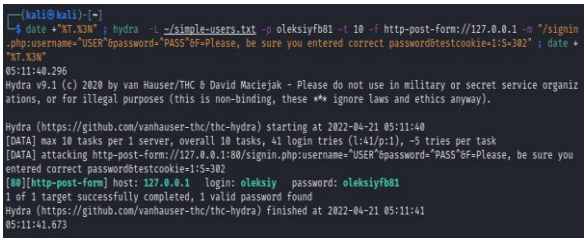

Figure 2: Using Hydra for login finding

Here we can see, that it took 1,377 seconds to find a login.

1.1.3. Finding login-password pair

In Figure 3 we can see how we can find pair login-password. For this, we need use the following parameters:

1. -L to set folder with login wordlists
2. -P to set folder with password wordlists
3. -t to set number of threads
4. http-post-form to set http website ip
5. -m to set additional settings

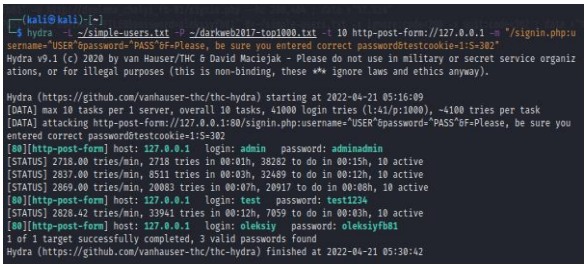

Figure 3: Using Hydra for login-password pair finding

Here we can see that it took 873 seconds to find all possible pairs of login-passwords. In this case we did not use -f parameter, because we wanted to find all pairs.

1.2. Wfuzz

Wfuzz was created in 2011. On GitHub Wfuzz V.1.4 was uploaded in 2014. This tool has really good documentation [5]. Xavi Mendez, the author of Wfuzz, concentrated the main purpose of this tool to brute force web-application user's accounts credentials.

Wfuzz is actively updated and developed. Besides it has customer support possibility. This tool is pre-installed in Kali Linux, however it is located not in password attacks sections, but in web application analysis. Unlike Hydra, Wfuzz is working only in Unix systems.

1.2.1. Finding password with login available

In Figure 4 we can see how we can find passwords when we know login. For this we need to use the following parameters:

1. -c for colorful output
2. -z to specify FUZZ keyword payload
3. -d to set input data
4. -u to set url
5. -hc to avoid response output

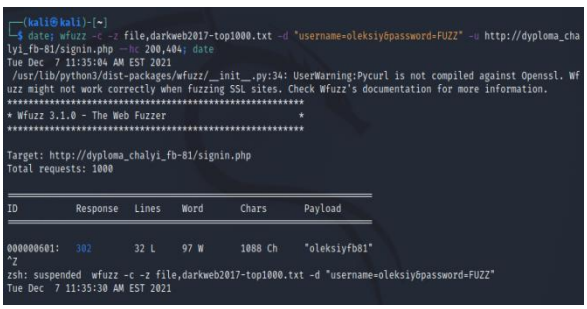

Figure 4: Using Wfuzz for password finding

Here we can see that it took 26 seconds to find a password. Wfuzz does not has stop command unlike Hydra.

1.2.2. Finding login with password available

In Figure 5 we can see how we can find logins when we know password. For this we need to use the following parameters:

1. -c for colorful output
2. -z to specify FUZZ keyword payload
3. -d to set input data
4. -u to set url
5. -hc to avoid response output

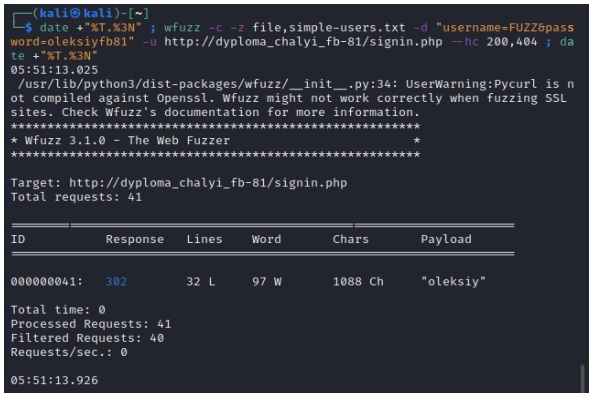

Figure 5: Using Wfuzz for login finding

Here we can see, that it took 0,901 seconds to find a login.

1.2.3. Finding login-password pair

In Figure 6 we can see how we can find pair login-password. For this, we need to use the following parameters:

1. -c for colorful output
2. -z to specify FUZZ keyword payload
3. -d to set input data
4. -u to set url
5. -hc to avoid response output

Here we can see that it took 200 seconds to find all possible pairs of login-passwords. In this case we used two -z parameters, because we wanted to find all pairs.

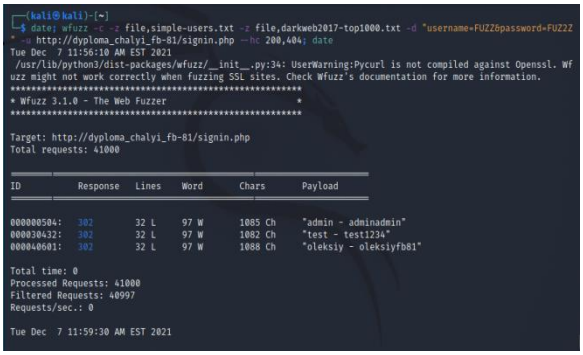

Figure 6: Using Wfuzz for finding for finding pair login-password

1.3. Patator

The third tool is called Patator. It also has no GUI as Wfuzz. Unlike previous software, Patator has no official website, only GitHub page [6]. Patator was created by a person, whose twitter account is @lanjelot. According to GitHub, the first version 0.1 was developed in 2011. The latest version at this moment 0.9 was released in 2020.

Unlike Wfuzz, Patator can work not only with HTTP and HTTPS but also support other protocols. In total, Patator supports 28 protocols, including SSH, Telnet, FTP and HTTP. The biggest advantage of this tool is that it can show used time and even in microseconds. Moreover Patator has a lot of additional parameters. Patator officially works only in Unix systems. It is pre-installed in Kali Linux. Using PyInstaller it can be installed in Windows.

Patator is not as popular as Hydra and Wfuzz. But its functionality and possibilities are not fewer than of aforcited . Patator perfectly complets tasks and this we will see in comparison Table 1.

1.3.1. Finding password with login available

In Figure 7 we can see how we can find passwords when we know login. For this, we need to use the following parameters:

1. http_fuzz for http protocol
2. url to specify url address
3. method to set POST or GET method
4. body to specify which data will be brute forced
5. 0 to set password wordlist directory
6. -x ignore:code to avoid response output
7. -x quit:code to stop when the first password will be found

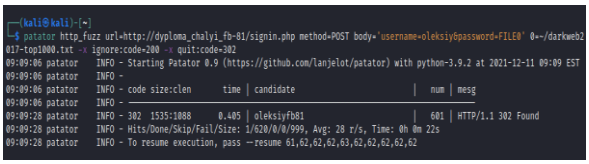

Figure 7: Using Patator for password finding

Here we can see that it took 22 seconds to find a password. Unlike Wfuzz, Patator has stop command just like Hydra has.

1.3.2. Finding login with password available

In Figure 8 we can see how we can find logins when we know password. For this, we need to use the following parameters:

1. http_fuzz for http protocol
2. url to specify url address
3. method to set POST or GET method
4. body to specify which data will be brute forced
5. 0 to set logins wordlist directory
6. -x ignore:code to avoid response output
7. -x quit:code to stop when the first password will be found

Figure 8: Using Patator for login finding

Here we can see that it took 0,568 seconds to find a login.

1.3.3. Finding login-password pair

In Figure 9 we can see how we can find pair login-password. For this, we need to use the following parameters:

1. http_fuzz for http protocol
2. url to specify url address
3. method to set POST or GET method
4. body to specify which data will be brute forced
5. 0 to set logins wordlist directory
6. -x ignore:code to avoid response output
7. -x quit:code to stop when the first password will be found

Figure 9: Using Patator for finding pair login-password

Here we can see that it took 184 seconds for all possible login-passwords pairs finding. In this case we did not use -x quit:code parameter, because we wanted to find all pairs.

1.4. Legion

Legion is the first software in our comparison that has only GUI. It looks a lot like updated Sparta, which was earlier pre-installed in Kali Linux. However, Legion is not updated with Sparta, because they have different authors. Since Kali Linux 2019.4 version, Sparta was deleted from Kali Linux tools list due to old version of Python using. According to GitHub [7] Legion was developed in 2020. GoVanguard, the developers of Legion, were encouraged by Sparta, because Legion has a lot of similar features.

Legion consists a lot of functionality of other tools like Nmap, Nikto, Shodan, Hydra and others. According to this, Legion is a very powerful tool for whitehats. In fact, we can see in file /usr/share/legion/app/setting.py that Legion supports 61 protocols as Hydra support.

For now, Legion works only on Unix systems. However, due to actively development of this software, it might take not so much time to port it to Windows and MacOs.

1.4.1. Finding password with login available

In Figure 10 we can see how we can find passwords when we know login in Legion software. For this, we need to use the same parameters as we used in Hydra. It is necessary to write IP and Port in their sections. Also it is important to choose the right Service. We could also set numbers of threads in Threads button.

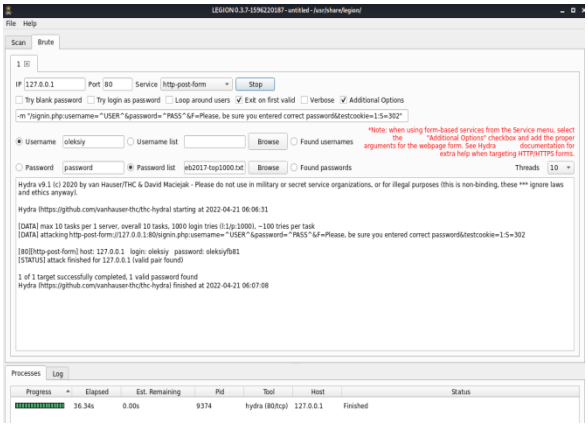

Figure 10: Using Legion for password finding

Here we can see that it took 36,34 seconds to find a password.

1.4.2. Finding login with password available

In Figure 11 we can see how we can find login when we know password in Legion software. For this, we need to use the same parameters as we used for password finding. We need to set Username list, set its wordlist file and write password that we know.

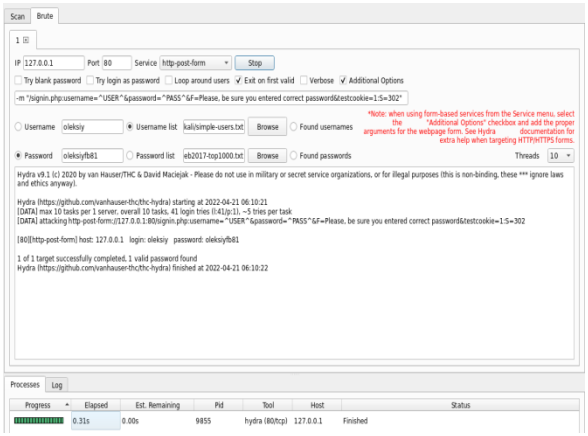

Figure 11: Using Legion for finding the login

Here we can see that it took 0,31 seconds to find a login.

1.4.3. Finding login-password pair

In Figure 12 we can see how we can find login-password pair in Legion software. For this, we need to select Username list to set logins wordlist and Password list to set passwords wordlist.

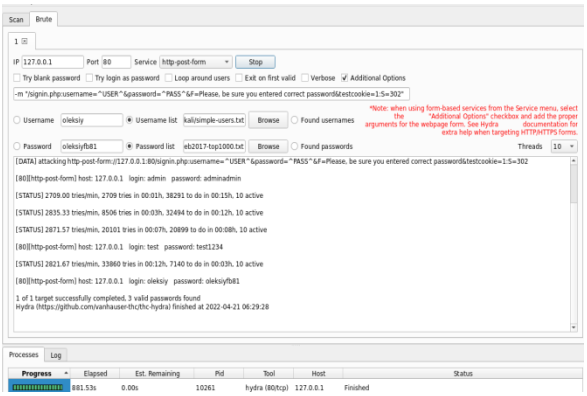

Figure 12: Using Legion for finding login-password pair

Here we can see that it took 881,53 seconds for finding all possible login-password pairs.

2. Presentation of research results

In this section the fastest tool for brute force in different cases is defined. You can see the syntax in Figure 1-12 or in official tools websites, that you can find in references. Experiments were carried out in own web-application in own virtual machine and on own localhost.

2.1. Plan of experimental research

To determine the fastest tool, three experimental researches were carried out, these you can see in Figures 1-12 for each software:

1. Finding password with login available
2. Finding login with password available
3. Finding login-password

All tools were tested in working pair with 10 threads. As the dictionaries, were used the following:

- darkweb2017-top1000.txt dictionary [3] with 1000 most popular passwords
- simple-users.txt dictionary [4] with 41 most popular logins

In third research both files were used to find all possible login-password pairs. The combination of words to check was the multiple of 1000 most popular passwords and 41 most popular logins, in total 41000 words to use. Three experiments were conducted not only to determine the fastest tool, but also to see how these tools will complete their tasks in different cases with different volumes of required combinations.

2.2. Analysis of experimental results

After conducting all experimental researches in created web-application, we are able to create input data Table 1.

Table 1
Tools experimental results

Tool	Password finding time	Login finding time	Login-password pair finding time
Hydra	27 s	1,377 s	873 s
Wfuzz	26 s	0,901 s	200 s
Patator	22 s	0,568 s	184 s
Legion	36,34 s	0,31 s	881,53 s

For better understanding of the data, it is required to convert input data in 5-points ranking system. This will help us to identify in which case which tool works the best. To convert the data we will use formula (1).

$$R = (5*Rb) / Rc \quad (1)$$

Here R – Tool points in 5-points ranking system

Rb – The best result from all tools in certain cases

Rc – Current tool result

After converting all data, we can build Table2

Table 2
Tools marks in 5-points ranking system

Tool	Finding password time	Finding login time	Finding pair login-password
Hydra	4,07	1,13	1,05
Wfuzz	4,23	1,72	4,6
Patator	5	2,73	5
Legion	3,03	5	1,04

Patator shows the best result in finding password and login-password pairs. Legion shows the best result in finding login. Wfuzz demonstrated a pretty good result for login-passwords pair finding. Hydra shows better then Legion result for password and login-password pairs finding.

Now, we are able to build Table3 to determine which tool is better to use in brute forcing user's account credentials in web-application.

Table 3
Final results

Tool	Sum	Result
Hydra	4,07+ 1,13+1,05	6,25
Wfuzz	4,23+1,72+4,6	10,55
Patator	5+2,73+5	12,73
Legion	3,03+5+1,04	9,07

Here we can see the leaders list:

1. Patator – 12,73
2. Wfuzz – 10,55
3. Legion – 9,07
4. Hydra – 6,25

3. Acknowledgements

I am really thankful to my Mom and Grandma for their support and for helping with preparation of this article. They encouraged me to do this article and gave interesting ideas.

I am also thankful to Mr. Myhailo Kolomytsev for his working with me, helping in updating and improving this article.

Conclusions

To sum up, there were four tools for finding user's account credentials in web-application researched, namely Hydra, Wfuzz, Patator, Legion. Besides, tools, basics characteristics and possibilities of usage in different operating systems were determined. Three experiments conducted for finding user's accounts credentials in web-application: password finding, login finding and login-password pairs finding.

All tools were tested in the same conditions and in the same environment. After gathering information from the experiments input table was created, which shows time that was required in each of experiments. Furthermore the results were converted in a 5 points system to determine the best tool in each case. Patator shows the best result in password and login-password pairs finding, whilst Legion shows the best result in login finding.

With these results using, information security specialist are able to choose among tools that were researched, to estimate potency of protection of user's accounts credentials in web-application and to develop protection tools from corresponding attacks.

References

[1] OWASP Top 10 team, OWASP Top 10 Web Application Security Risks, 2021, URL: https://owasp.org/Top10/

[2] van Hauser, THC-Hydra, version 9.4, 2022, URL: https://github.com/vanhauser-thc/thc-hydra

[3] g0tmi1k, Wordlist with 1000 most popular passwords darkweb2017-top1000.txt, 2019, URL: https://github.com/danielmiessler/SecLists/blob/master/Passwords/darkweb2017-top1000.txt

[4] 1N3, Wordlist with popular usernames simple-users.txt, 2021, URL: https://github.com/1N3/BruteX/blob/master/wordlists/simple-users.txt

[5] Xavi Mendez, Wfuzz Documentation, Release 2.1.4, 2020, URL: https://buildmedia.readthedocs.org/media/pdf/wfuzz/latest/wfuzz.pdf

[6] Lanjelot, Patator, version 0.9, 2020, URL: https://github.com/lanjelot/patator

[7] GoVanguard, Legion, 2020, URL: https://govanguard.com/legion/