Brooklyn Dressel
Assignment 3

Abstract

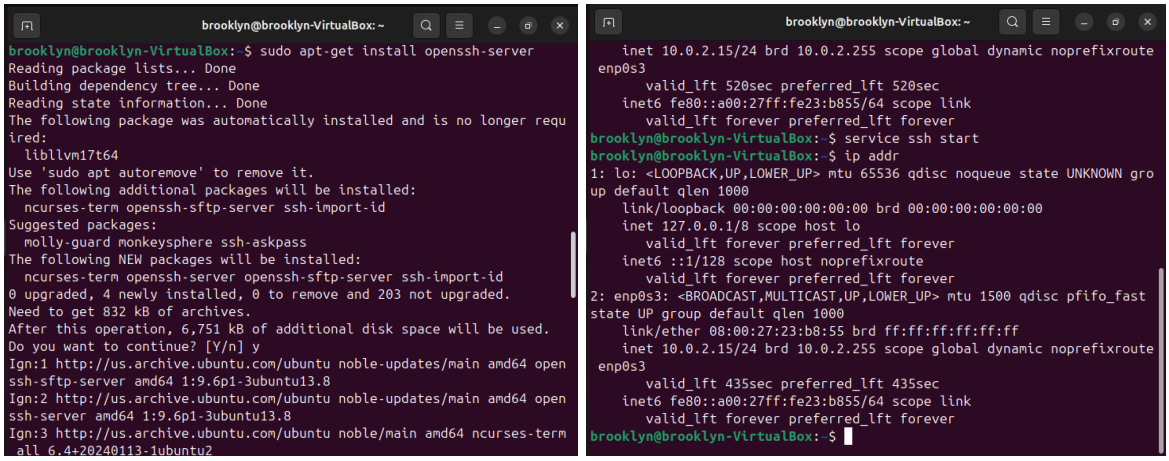
This project captures and analyzes packets using WireShark, captured from both attempted and successful brute force password attacks on an SSH server using Hydra. In this experiment the user's login is known and a list of roughly 300 common passwords are attempted, after roughly 250 the correct password is entered and Hydra returns a statement saying it is the correct password and the program terminates. Upon inspection of the packets you can see that the Diffie-Hellman key exchange algorithm is used and is clearly shown in plaintext. However, it is extremely difficult to tell the difference between successful password attempts and unsuccessful attempts. The main difference can be found in the size and length of the subsequent packets, with failed attempts having many more small packets sent back and forth. From this we can see both the vulnerabilities to brute force attacks for common passwords as well as show the strength and possible vulnerabilities of the encrypted communication.

Introduction

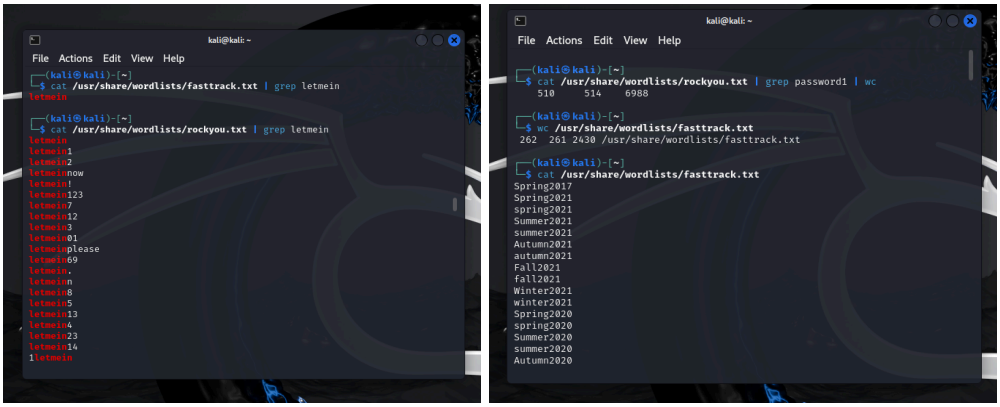
Used Kali Linux virtual machine and the applications provided by it, such as WireShark and Hydra. Public word lists are unzipped and used. Ubuntu is used to run the victim SSH server. Hydra is used to perform the brute force attack on Ubuntu using both a public and a personal word list. WireShark is used to capture the packets generated by this attack. Additionally, all commands used during this project are provided separately.

Summary of Results

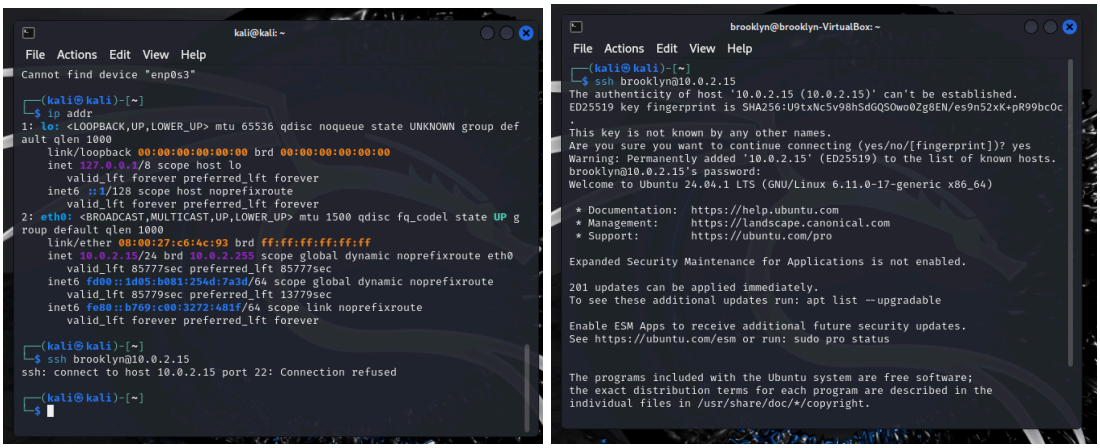
After a *long* struggle of trying to download and boot up my Ubuntu machine I finally was able to boot it up and run it from my hard drive. Once inside Ubuntu I was prompted to create a username and password, which I set to `brooklyn` and `letmein`, respectively. From this point I ran commands `4` and `5` to update Ubuntu and download the SSH server software. Once the server software was installed I ran command `6` to start up the server to open it to attacks. At this point I was prompted for my password, to which I entered `letmein` and the server started. In order to start a connection from another machine I used command `7` to find my ip address.



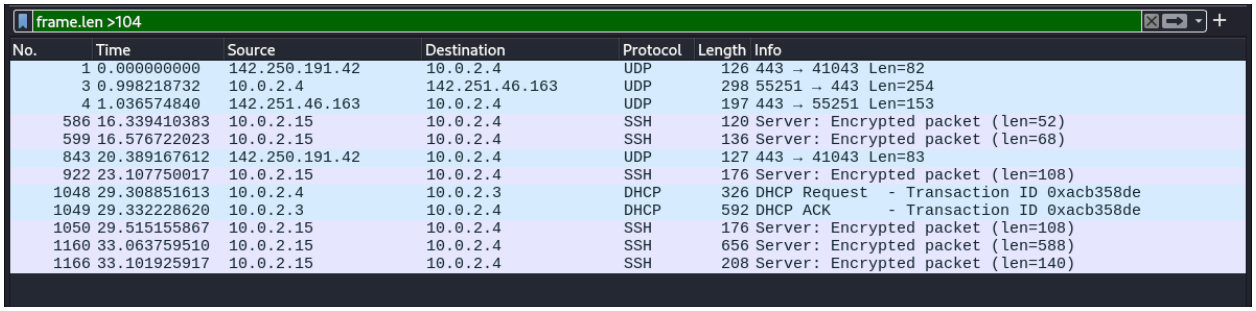
Next I started preparing and checking the public word lists inside Kali Linux. I do so by first running commands `1`, `2`, and `3` in order to unzip the public word list, find the word count, and finally check to see if the password `letmein` can be found in the word list. The results of the commands ran on this word list is shown below.



My next step was to ensure that the SSH server was up and running correctly and to check that I could log in remotely. At this point I had struggled a lot to get my network settings such that my two separate virtual machines could communicate with each other over the network. I had finally found that I could achieve this result using a NAT Network, this allowed both networks to get ip addresses and communicate. The screenshots below show my eventual success in obtaining a working ip address and logging into my ssh server with the correct password using command `8`.



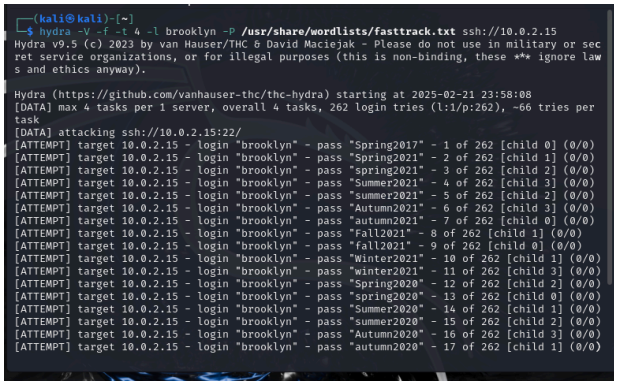
The packet capture shown below shows this successful login attempt from the Kali terminal. From the *many* packets captured, I filtered by the larger one as most of these were the same size. From this I believe that these packets at the bottom are showing the entry to the server upon the successful information provided.

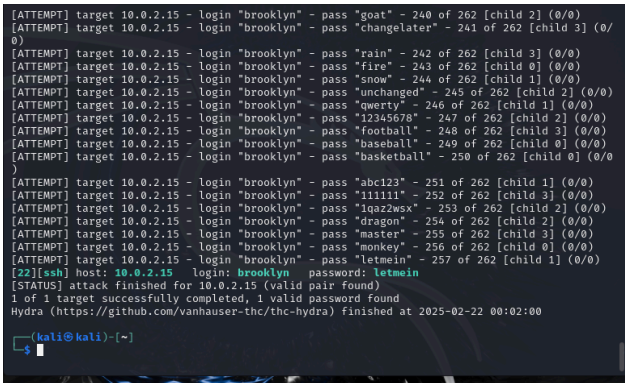


No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	142.250.191.42	10.0.2.4	UDP	126	443 → 41043 Len=82
3	0.998218732	10.0.2.4	142.251.46.163	UDP	298	55251 → 443 Len=254
4	1.036574840	142.251.46.163	10.0.2.4	UDP	197	443 → 55251 Len=153
586	16.339410383	10.0.2.15	10.0.2.4	SSH	120	Server: Encrypted packet (len=52)
599	16.576722023	10.0.2.15	10.0.2.4	SSH	136	Server: Encrypted packet (len=68)
843	20.389167612	142.250.191.42	10.0.2.4	UDP	127	443 → 41043 Len=83
922	23.107750017	10.0.2.15	10.0.2.4	SSH	176	Server: Encrypted packet (len=108)
1048	29.308851613	10.0.2.4	10.0.2.3	DHCP	326	DHCP Request - Transaction ID 0xacb358de
1049	29.332228620	10.0.2.3	10.0.2.4	DHCP	592	DHCP ACK - Transaction ID 0xacb358de
1050	29.515155867	10.0.2.15	10.0.2.4	SSH	176	Server: Encrypted packet (len=108)
1160	33.063759510	10.0.2.15	10.0.2.4	SSH	656	Server: Encrypted packet (len=588)
1166	33.101925917	10.0.2.15	10.0.2.4	SSH	208	Server: Encrypted packet (len=140)

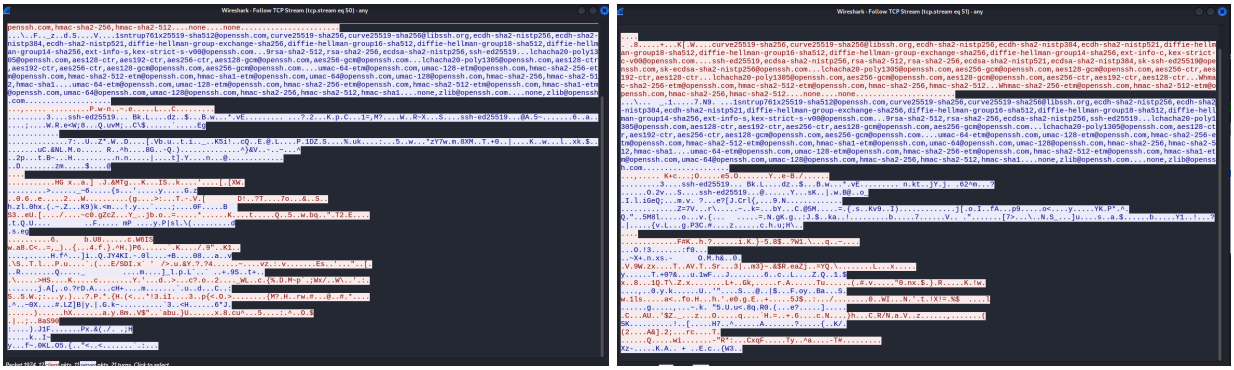
Now that I knew the server was running correctly, and confirmed the password, I created a new text file with the correct password to get a baseline of the packet captures when Hydra sends the attempts. In order to get this I used command `9`, I provided the correct username, the ip address, as well the text file with only the correct password. The results of the WireShark capture are shown below, the path of the packets is followed and displayed before, with the packets sent from the client in blue, and the packets sent from the server in red. These results will be compared to subsequent captures later.

Next I ran the code using a much longer word list, that was filled with 262 possible common passwords. I used command `10` to attempt every password in the word list. On attempt 257, the correct password was guessed and Hydra terminated the run and reported the correct guess.





The following two packet captures are followed from two separate communications. Both of these show failed password attempts. The large chunk of code at the top shows the Diffie-Hellman key exchange between the server and client (part of the packets are out of frame on the top of the left image). These packets have lots of information in plain text. The subsequent long chain of client-sent packets are believed to be the sending of the symmetrically encrypted password attempted. However, the long chain of shorter-length packets sent between client and server at the bottom are believed to be the server communicating the rejection of the password attempt to the client.



Compared to the following of the last exchange between Hydra and the SSH server, which we know was a successful password attempt. In this capture we can once again see the Diffie-Hellman key exchange at the top of the capture, and the long string sent by the client which is believed to be the symmetrically encrypted password attempt. However, the key difference between this and the previous two is the number of subsequent communications between the client and server. This image shows the packets sent between the two, after the password, to be nearly half the amount of the previously shown failed attempts. After capturing and analyzing all of these packet captures, I closed the SSH server on Ubuntu by running command `11`.

Conclusion

In conclusion, this assignment has shown the process of remote login. How the Diffie-Hellman key exchange algorithm is used to provide a set of symmetric keys to both the user and client for password attempts, helping to preserve authentication, access control, and data confidentiality. This serves to protect user's login information from unauthorized attackers gaining access to the user's confidential information. It is important to note that a key weakness to the Diffie-Hellman key exchange is the

man-in-the-middle attack. Since so much of the Diffie-Hellman information being so clearly identifiable (in plaintext) makes it a dangerous weakness if an attacker were to partition the network and control the key exchange, and would subsequently have access to all encrypted communications.

Since the keys exchanged using Diffie-Hellman are symmetric, there is no guarantee of who actually sent the packets, there is repudiation that could be solved using asymmetric cryptographic techniques such as digital signing.

Additionally, we cannot confirm whether or not there are any integrity checks performed on the packets being transmitted. A way to confirm integrity is through the process of hash maps, as we saw with our previous assignment which used md5sum to confirm the integrity of images. However, there is no way for us to confirm whether or not these packets we have captured are being checked for integrity.

Furthermore, this assignment shows the server's ability to provide obscure feedback, as it is incredibly difficult to tell which packets are successful logins versus unsuccessful logins.

This attack was shown to be successful. Hydra fairly quickly attempted numerous passwords and clearly identified the correct password. To which I would easily be able to use to then log in to the SSH server.

The advantages of this attack is that it doesn’t require too much effort or thought on the part of the attacker. There are large word lists publicly available, and given enough time and enough attempts the attacker has a fairly good chance of finding a match. However, the disadvantage is that it would become infinitely more difficult if you are not already aware of a valid username, as you would have to check all passwords against a username that might not even exist. Additionally, this attack can require a long wait time depending on the number of attempts you would have to check, and even then it is not guaranteed that the correct password is among your list.

Hydra can be used for many other applications, apart from an SSH server. Hydra can be used to brute force HTTP logins, given the proper request method (in the case of the SSH server it was the ip address). But you could hypothetically use Hydra to brute force Facebook user’s login information.

While Hydra is a powerful tool there are many ways to protect against brute force attacks. Such as much more complex passwords, making passwords longer, with various characters, as well as preventing commonly used passwords. Preventing users

reusing passwords for multiple accounts. Forcing users to change their passwords on a regular basis. Using another form of authentication in addition to the password, such as answers to security questions, smartcards, fingerprints, or dynamic biometrics.