Brute Force Defense

Jackson McCullough

1 Introduction

This was a simple brute force simulation using multiple tools to launch, log, and prevent future brute force attacks.

2 Setup

2.1 Kali(attacker)

The software **Hydra** was used to launch the brute force attack. The target was an Ubuntu virtual machine that had OpenSSH enabled. The attacker is brute forcing via SSH. A word list was used to attack the SSH and could therefore be posed as a dictionary attack. No scripts or anything will be injected into the Ubuntu machine, only password cracking.

2.2 Ubuntu(target)

First step was to install and enable **OpenSSH**, to widen some attack surface and give an attack vector for the attacker.
Next, **Splunk Forwarder** was installed on the ubuntu vm for logging and forwarding purposes. We added monitors to send data to a different computer that would be collecting the data on **port 9997**. Monitors were created to log the auth logs, so failed/successful password attempts, and logs from fail2ban. **Fail2ban** is the software being used for further prevention of a brute force attack once the initial attack is successful, since fail2ban will be used for recognizing patterns in logs and blocking certain IP's from logging in an x amount of times, essentially an IPS.

2.3 Windows(Splunk)

Nothing changed much on the windows machine. Splunk Enterprise was installed, configured to receive data on port 9997, and adjusted the firewall of the machine to actually be able to "talk to" the Ubuntu machine and accept the log data.

3 Attack

The attack was launched from kali. The first step was creating the word list, that was just random characters or dictionary words that were personally generated, then the correct password lingering somewhere in the document. Next was launching the attack, which is a simple command from hydra:

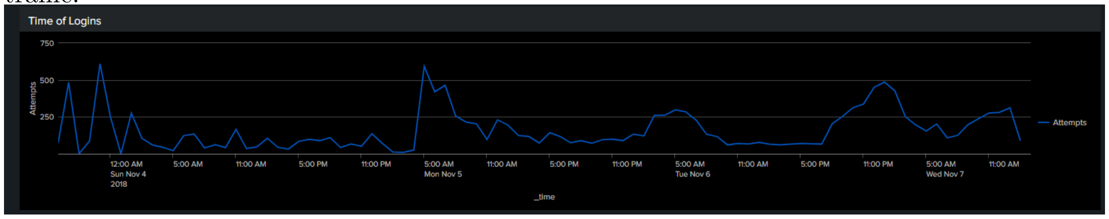
```
hydra -l ubuntu -P /home/lxvert/Desktop/passwords.txt -f ssh://[Ubuntu-IP-Address]
```

This launched the brute-force attack with the word list of about 300 words, and the filter "-f" just tells hydra to stop after the valid login credentials have been found(the correct password was in the word list).

4 Defense

There was no defense initially, even the password to ssh into this account was just "mysafepassword", with no multi-factor authentication or other type of authentication besides the weak password.
After the attack, visualizations were created in **Splunk** to help analyze the logs in a clear manner. There were also open-source datasets added that simulate a brute-force attack to get attacks from multiple places, and significantly higher volume.

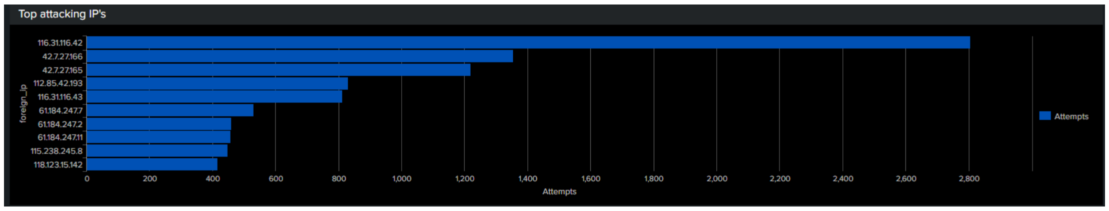
The first search query created was to find the the number of login attempts in congruence to the time(per hour). Displayed by a line chart, we could see the number of login attempts and the times they took place, along with the most traffic:



The query created to create this chart was:

```
index=main | eval _time = strptime(timestamp, "%a %b %d %H:%M:%S %Y")
| timechart span=1h count as Attempts
```

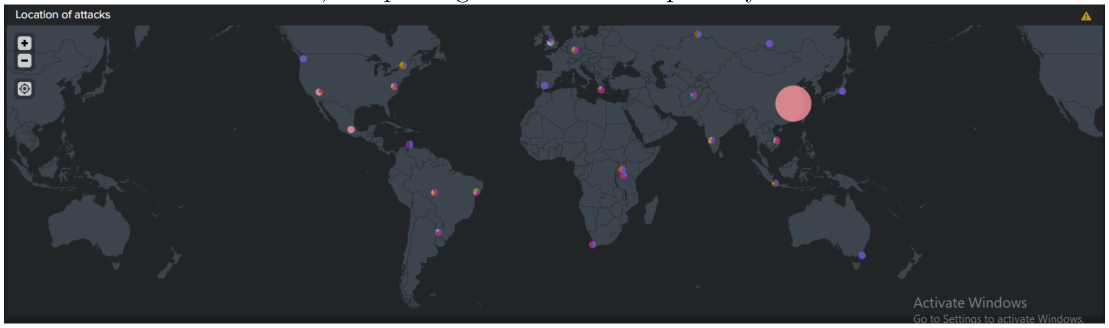
The next one created was to find the total login attempts based on the IP, to help gather information of particular IP's that may need to be blocked, or a certain subnet that needs to be blocked. The top ten IP addressed were displayed.



The query used to find and create this graph was:

```
index=main | stats count as Attempts by foreign_ip | sort - Attempts
| head 10
```

The final filter applied was finding the location of the logins, and the accounts that were trying to be accessed. If a search is created to find these, we could ban certain locations from logging in if it is completely useless for anyone to have access there. However, IP spoofing or VPN's could possibly counter this.



The query used was:

```
index=main | eval _time = strptime(timestamp, "%a %b %d %H:%M:%S %Y")
| iplocation foreign_ip | geostats count by username latfield=lat longfield=lon
```

4.1 Fail2ban

A pretty rudimentary configuration was made in fail2ban on the target machine in order to create a more strict login entry for ssh, as well as banning specific IP attempts. The **jail.local** file was configured to allow less retries and apply certain ban times. The purpose behind this is to allow login attempts, but if there is an odd amount of failures, a ban time would allow the security operations center to have time to recognize suspicious login attempts, giving them time to ban or take any steps needed. The following configurations were made for default login and ssh attempts.

Default Configurations:

```
bantime = 600
maxretry = 3
ignoreip = [Ubuntu IP address]
banaction = iptables-multiport
```

SSH specific configurations

```
enabled = true
```

```
port = ssh
filter = sshd
logpath = /var/log/auth.log
maxretry = 3
```

To ban certain IP's, it needs to be directly from the command-line interpreter, rather than set into jail.local.
There were multiple IP addresses that may have needed to be banned, so to do so the input would be:

```
sudo fail2ban-client set sshd banip [input IP Address]
```

5 Recommendations and Closing

The fail2ban software worked well with blocking, and the splunk logging helped exponentially with knowing specifically what to block or filter. A great upgrade or extra step of security would be introducing MFA.
The purpose of this was to apply tools and knowledge to a real-world scenario of a brute-force attack, and apply defensive measures and monitoring using splunk and fail2ban.