

SSH Attack and Defense

Custom Project Experiment

Peyton Robinson

10/11/2025

Table of contents

Table of contents	2
Summary.....	2
Overview	4
Purpose	4
Scope	4
Attack and Defense Timeline.....	5
Timeline Overview	5
Phase 1: Initial Attack	6
Step 1: Deploy Kali Linux virtual attack machine	6
Step 2: Check port 22 availability with Nmap	6
Step 3: Brute force port 22 with Metasploit.....	8
Phase 2: Defense	10
Step 1: setup Iptable rules	10
Step 2: Add authentication logs into Splunk	11
Step 3: Create Splunk alerts for SSH login attempts	13
Phase 3: Test Attack	13
Step 1: Brute force port 22	13
Step 2: View port 22 with Nmap after attack	13
Step 2: Monitor SSH login alerts in Splunk	14
Outcome	15
Lessons Learned.....	15
Recommendations.....	15
References	17

Summary

Because SSH is a popular attack vector, I practiced hardening SSH with this custom project in three phases: initial attack, defense, and test attack. The following were tools used in the project:

1. Linux target and attack machine
2. Splunk Enterprise
3. Iptables
4. Nmap and Metasploit

The timeline gives a detailed run-through of how I initiated attacks and built defenses, including screenshots for more context.

I learned a lot about how Splunk and firewalls can be used to protect SSH connections from brute force attacks; however, disabling root access and enabling Multi-Factor Authentication (MFA) would further harden SSH connections. For future projects, I aim to use AI more and to watch more video tutorials instead of reading mostly articles.

Overview

Purpose

The SSH protocol is a popular attack vector, and misconfigurations can lead to malicious connections. Once a connection is established for the attacker, a plethora of other attacks could take place: privilege escalation, lateral movement, and data exfiltration to name a few.

I created this custom attack and defense project to learn about Splunk and firewall defenses against brute force attacks on the SSH protocol.

Scope

The relevant components are the following:

1. Linux Host device
2. Kali Linux attack virtual machine
3. Splunk Enterprise
4. Iptables
5. Nmap
6. Metasploit

The attack is only to attempt to establish a SSH connection into the host computer on port 22. The attack does not go any further, whether getting into the host computer or not. Defense is only to rate-limit SSH logins from the attacker IP and set up alerts in Splunk.

Attack and Defense Timeline

Timeline Overview

The timeline below is a summary of the steps taken in the project.

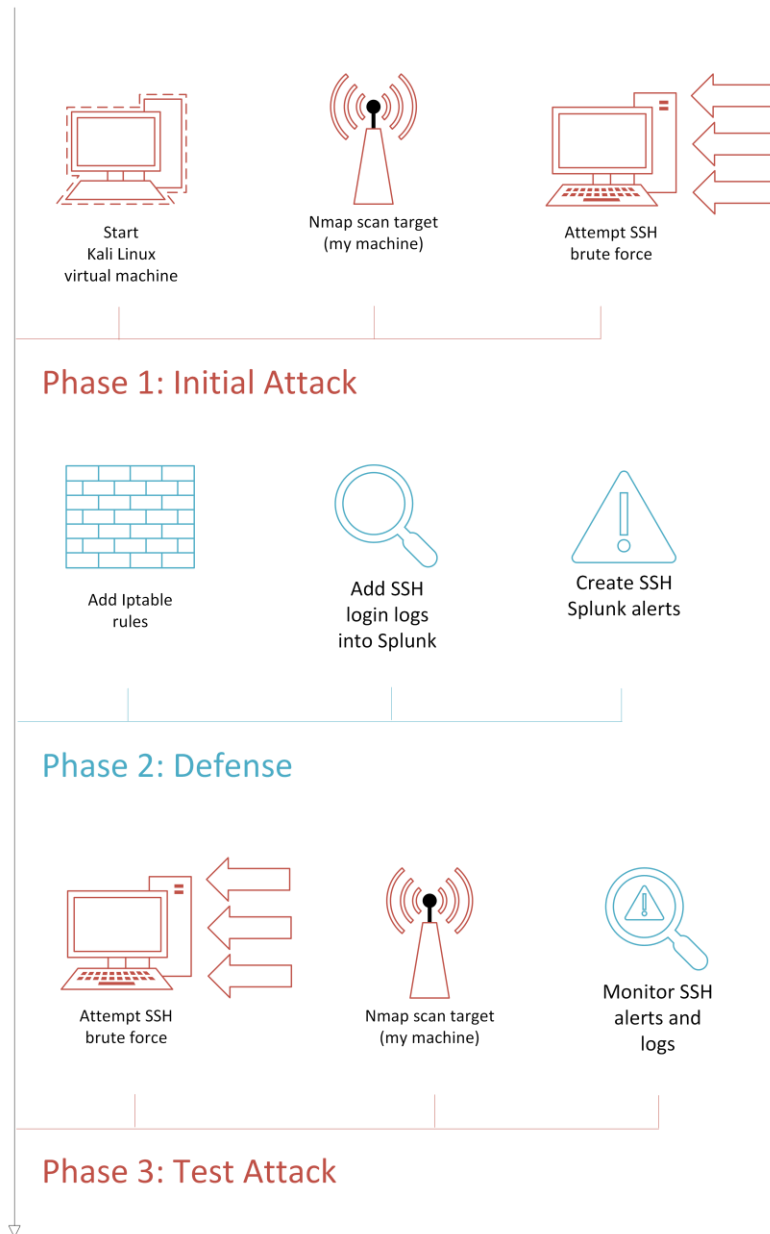


Figure 1 - Microsoft Visio timeline overview of events and phases of attack and defense

Phase 1: Initial Attack

Step 1: Deploy Kali Linux virtual attack machine

To initiate the attack, I started up my Kali Linux virtual machine on VirtualBox. I have experimented with this machine in my home lab before, so everything is up to date.

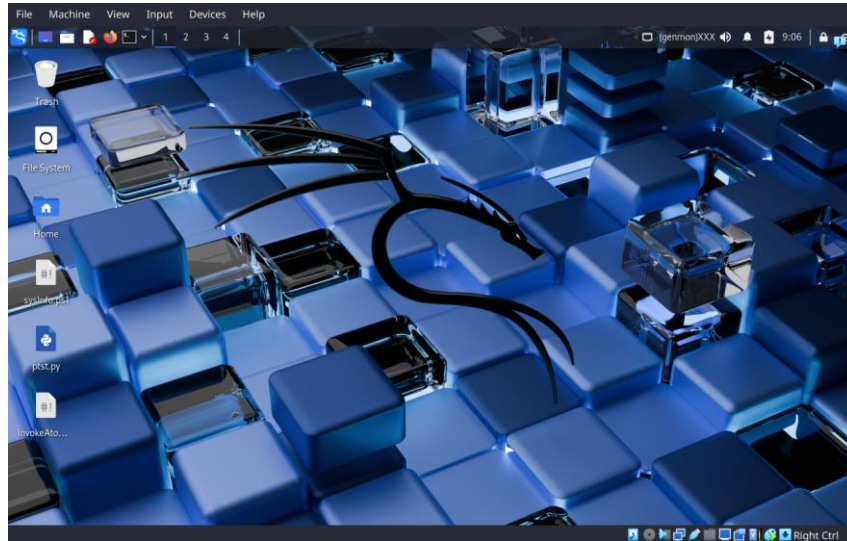


Figure 2 – My Kali Linux home screen.

Step 2: Check port 22 availability with Nmap

Before using Nmap, I installed and enabled SSH on the target machine, which is my personal computer. I used Nmap in the terminal to get information about port 22 on the target machine. I also requested to get machine information by passing the `-A` flag, and Nmap thought the host was a virtual machine. It is because I bridged my network adapter to the virtual machine, so it is scanning itself in a way.

```
File Machine View Input Devices Help
mb@kali: ~
Session Actions Edit View Help
Nmap done: 1 IP address (1 host up) scanned in 17.56 seconds

(mb@kali)~$ nmap -T2 -A -p 22
Starting Nmap 7.95 ( https://nmap.org ) at 2025-10-09 09:08 EDT
Nmap scan report for [redacted]
Host is up (0.00048s latency).

PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 9.6p1 Ubuntu 3ubuntu13.14 (Ubuntu Linux; protocol 2.0)
|_ ssh-hostkey:
|_ 256 [redacted] (ECDSA)
|_ 256 [redacted] (ED25519)
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Device type: VoIP adapter|bridge|general purpose|printer
Running (JUST GUESSING): AT&T embedded (98%), Oracle Virtualbox (94%), Slirp (94%), QEMU (94%), Konica Minolta embedded (89%), GNU Hurd (88%)
OS CPE: cpe:/a:oracle:vm_virtualbox cpe:/a:danny_gasparovski:slirp cpe:/a:qemu:qemu cpe:/h:konicaminolta:7035 cpe:/o:gnu:hurd
Aggressive OS guesses: AT&T BGW210 voice gateway (98%), Oracle Virtualbox Slirp NAT bridge (94%), QEMU user mode network gateway (94%), Konica Minolta 7035 printer (89%), GNU Hurd 0.3 (88%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 2 hops
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE (using port 22/tcp)
HOP RTT ADDRESS
1 0.26 ms [redacted]
2 0.51 ms [redacted]

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 28.63 seconds

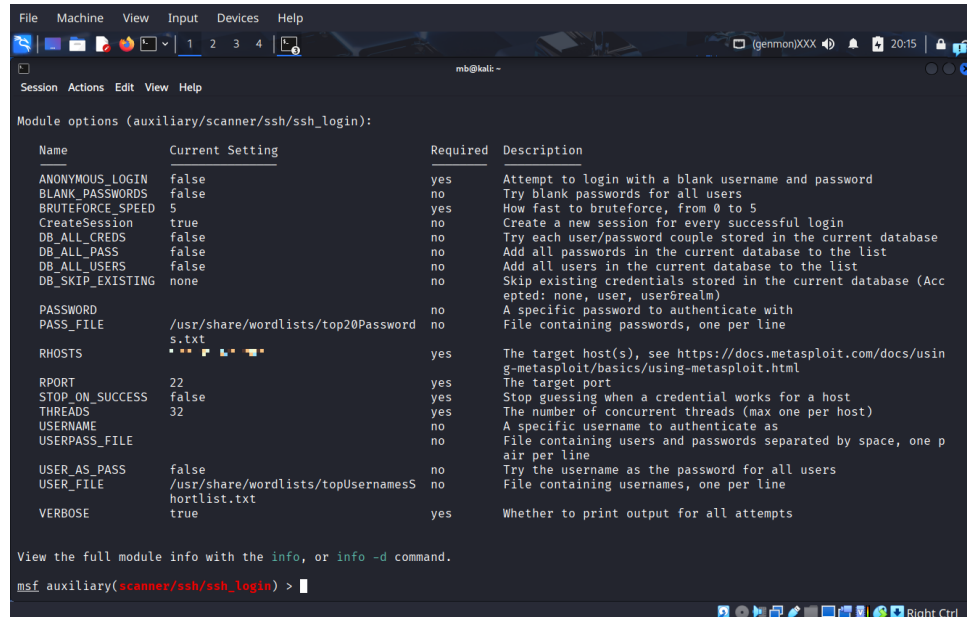
(mb@kali)~$
```

Figure 3 – Nmap scan result of the target machine

Metasploit is a common penetration testing tool for automating attacks on devices. I used its ability to brute force ssh connections to test my machine.



Afterward, I switched to the `ssh_login` tool and configured the options by setting up the host IP, password list file, username list file, number of threads to 32, and its verbosity to true. To make the experiment faster, I created custom password and username wordlists, containing only the most popular passwords and usernames.



The screenshot shows a Metasploit terminal window with the `ssh_login` module selected. The terminal displays a table of module options, their current settings, whether they are required, and their descriptions. The options include `ANONYMOUS_LOGIN`, `BLANK_PASSWORDS`, `BRUTEFORCE_SPEED`, `CreateSession`, `DB_ALL_CREDS`, `DB_ALL_PASS`, `DB_ALL_USERS`, `DB_SKIP_EXISTING`, `PASSWORD`, `PASS_FILE`, `RHOSTS`, `RPORT`, `STOP_ON_SUCCESS`, `THREADS`, `USERNAME`, `USERPASS_FILE`, `USER_AS_PASS`, `USER_FILE`, and `VERBOSE`. The `VERBOSE` option is set to `true`. The terminal also shows the prompt `msf auxiliary(scanner/ssh/ssh_login) >`.

Name	Current Setting	Required	Description
ANONYMOUS_LOGIN	false	yes	Attempt to login with a blank username and password
BLANK_PASSWORDS	false	no	Try blank passwords for all users
BRUTEFORCE_SPEED	5	yes	How fast to bruteforce, from 0 to 5
CreateSession	true	no	Create a new session for every successful login
DB_ALL_CREDS	false	no	Try each user/password couple stored in the current database
DB_ALL_PASS	false	no	Add all passwords in the current database to the list
DB_ALL_USERS	false	no	Add all users in the current database to the list
DB_SKIP_EXISTING	none	no	Skip existing credentials stored in the current database (Accepted: none, user, user@realm)
PASSWORD		no	A specific password to authenticate with
PASS_FILE	/usr/share/wordlists/top20Password.txt	no	File containing passwords, one per line
RHOSTS		yes	The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
RPORT	22	yes	The target port
STOP_ON_SUCCESS	false	yes	Stop guessing when a credential works for a host
THREADS	32	yes	The number of concurrent threads (max one per host)
USERNAME		no	A specific username to authenticate as
USERPASS_FILE		no	File containing users and passwords separated by space, one pair per line
USER_AS_PASS	false	no	Try the username as the password for all users
USER_FILE	/usr/share/wordlists/topUsernames.txt	no	File containing usernames, one per line
VERBOSE	true	yes	Whether to print output for all attempts

View the full module info with the `info`, or `info -d` command.

`msf auxiliary(scanner/ssh/ssh_login) >`

Figure 5 – Metasploit `ssh_login` options

I started the attack, and it began passing the passwords and usernames; however, the attack was very slow, even though the thread count was high, and the VM had plenty of computing power. After some research, SSH comes with built-in brute force mitigations by retry-limiting logins. Despite this, I still wanted to harden my system with more security controls. I also knew that these tests would not get into the machine because I knew the username and password.

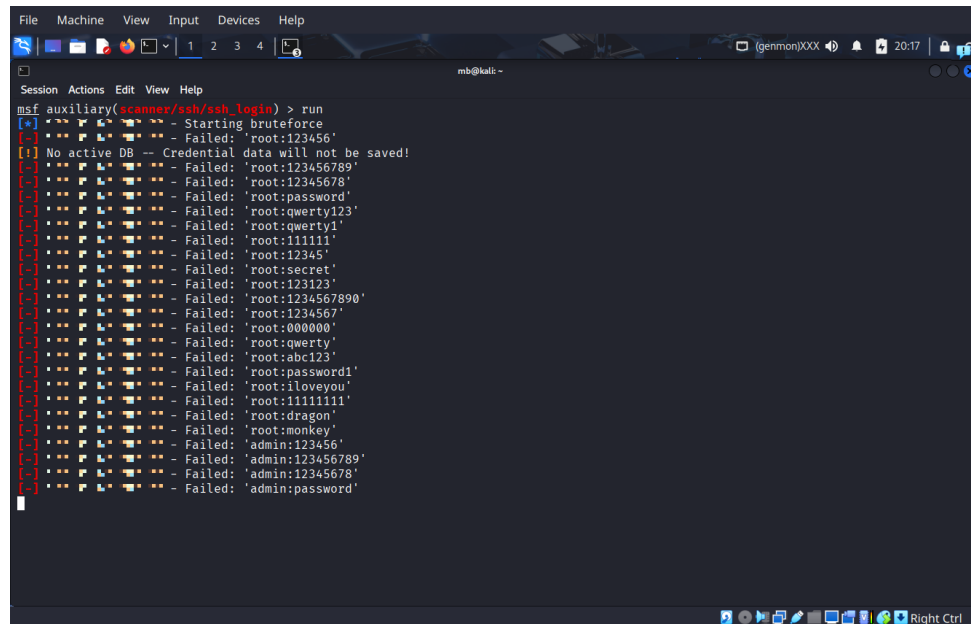


Figure 6 – running the brute force attack on the Kali Linux VM

Now that the initial test attack has finished, it was time to harden my system with Iptables and Splunk.

Phase 2: Defense

Step 1: setup Iptable rules

Switching to the host machine, I opened the terminal and added these iptable rules:

```
sudo iptables -I INPUT -p tcp --dport 22 -m state --state NEW -m
recent --set
```

```
sudo iptables -I INPUT -p tcp --dport 22 -m recent -update -seconds 60 --hitcount 4 -j DROP
```

I used Gemini 2.5 flash to learn how to use Iptable syntax, and what each part of the command means. I used the flag “-I” to ensure the rules are inserted to the top of the rule list based on how firewall rules behave. Below is a snapshot of my terminal entering these Iptable rules.



Figure 7 – adding Iptable rules

To check if the rules were added, I used the following command:

```
sudo iptables -L INPUT --line-numbers
```

The output is below.

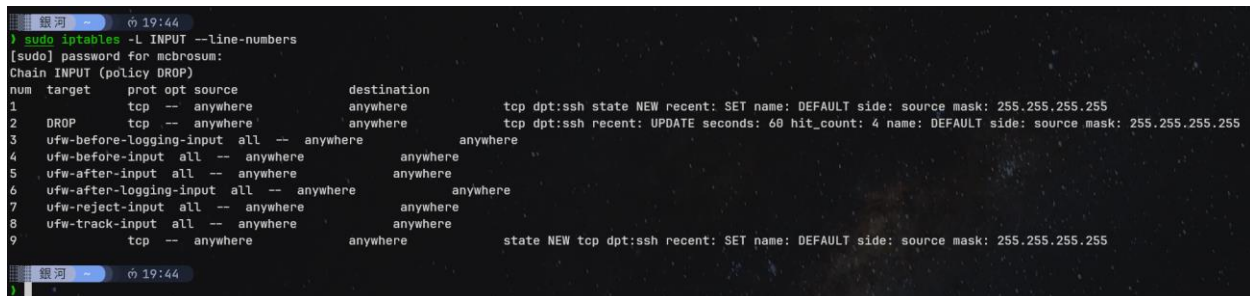


Figure 8 – Iptable output in the terminal

Next is setting up Splunk monitoring.

Step 2: Add authentication logs into Splunk

I used Splunk before this project, so I was familiar with the basics of the UI; however, I had to study more about the search syntax. Gemini 2.5 flash came in handy to learn how to search logs. I also followed a Splunk guide for searching SSH logs [1].

Below is my first search for invalid SSH login attempts:

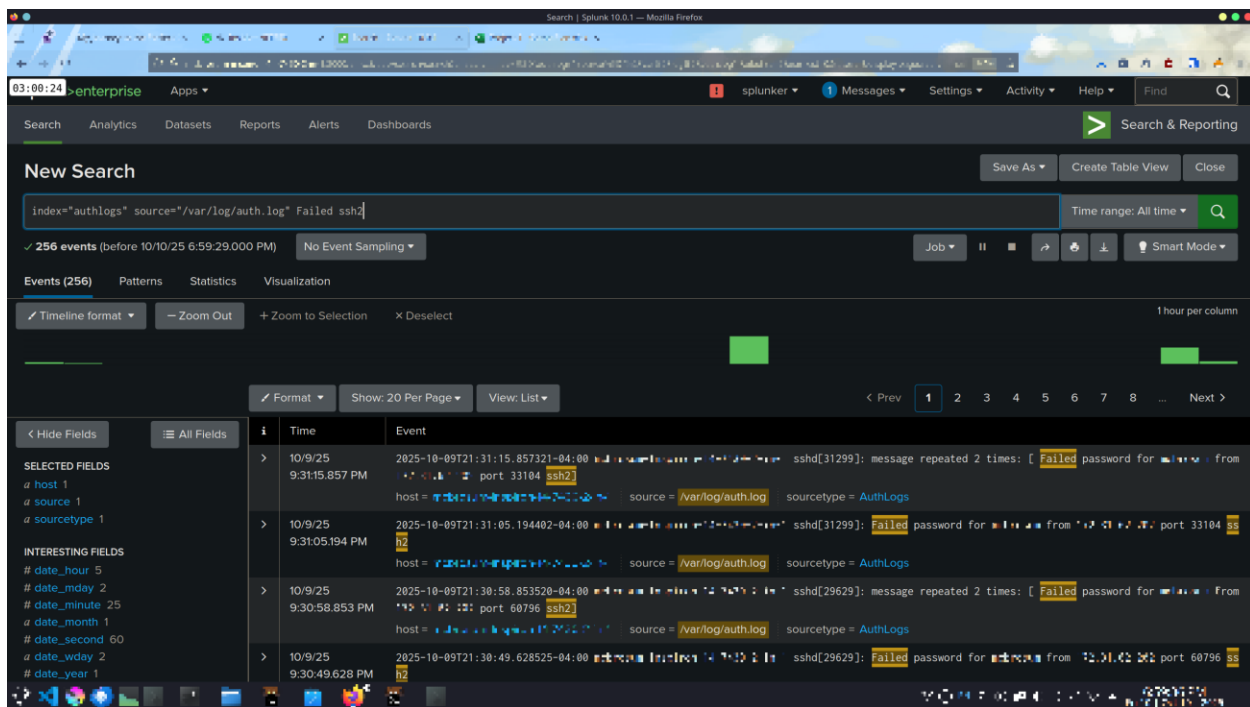


Figure 9 – Splunk search results with “Failed” and “ssh2” keywords.

After finding the correct logs, I extracted a field and named it “SSHInvalidUser” by using regex. Adding this field to the search result creates this chart:

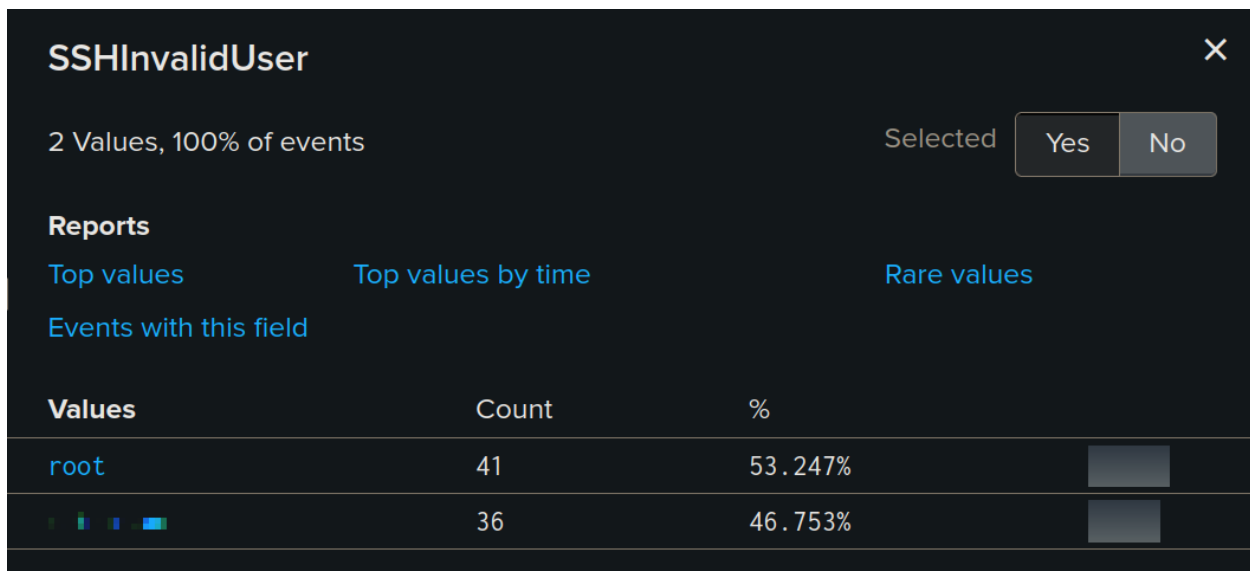
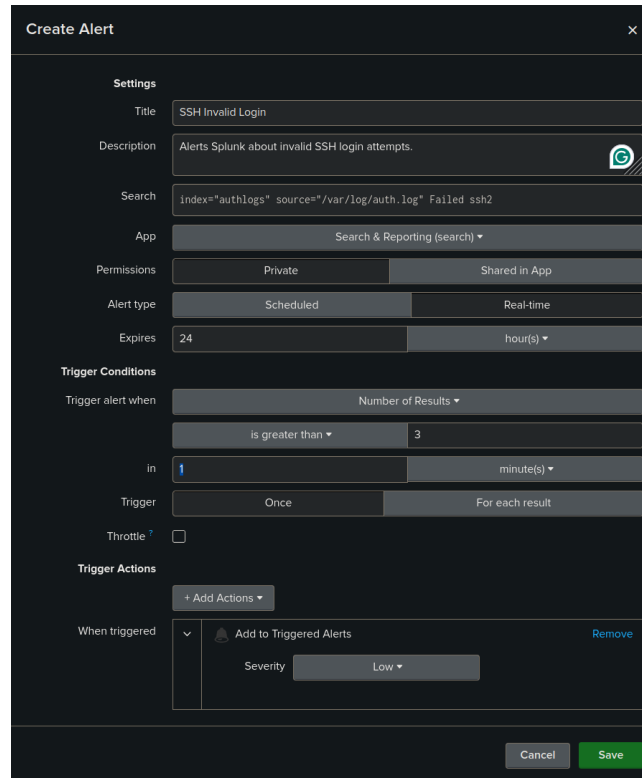


Figure 10 – chart showing invalid users from the SSH attack.

Step 3: Create Splunk alerts for SSH login attempts

Once I found the correct logs, I created real-time alerts using the SSH log search query. Below is me setting alerts up:



The screenshot shows the 'Create Alert' interface in Splunk. The 'Settings' section includes: Title 'SSH Invalid Login', Description 'Alerts Splunk about invalid SSH login attempts.', Search query 'index="authlogs" source="/var/log/auth.log" Failed ssh2', App 'Search & Reporting (search)', Permissions 'Private', Alert type 'Real-time', and Expires '24 hour(s)'. The 'Trigger Conditions' section shows 'Trigger alert when' set to 'Number of Results' 'is greater than' '3' 'in' '1 minute(s)'. The 'Trigger' is set to 'Once'. The 'Throttle' checkbox is unchecked. The 'Trigger Actions' section shows 'When triggered' with a dropdown menu containing 'Add to Triggered Alerts' and 'Remove'. The 'Severity' is set to 'Low'. At the bottom are 'Cancel' and 'Save' buttons.

Figure 10 – setting up real-time SSH Splunk alerts.

Now that my defenses are set up, I can initiate a test attack.

Phase 3: Test Attack

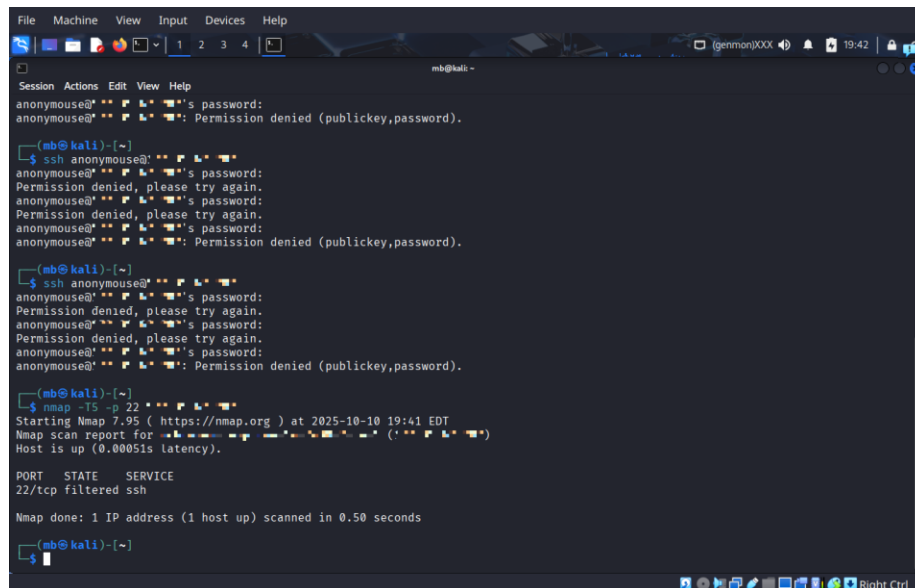
Step 1: Brute force port 22

Using the same Kali Linux virtual machine and method as before, I attempted the same attack on the target machine. This time, I inputted random passwords instead of using Metasploit because I wanted to trigger the Splunk alerts and see if the firewall drops my packets.

Step 2: View port 22 with Nmap after attack

Going back to the attack machine, I ran another Nmap scan after the attack. Success! The state is “filtered,” meaning that the attack machine’s packets are dropped toward the

target machine. The attack will have to wait one minute before they can make four more attempts.



```
File Machine View Input Devices Help
mb@kali ~
Session Actions Edit View Help
anonymouse@* * * * *'s password:
anonymouse@* * * * *: Permission denied (publickey,password).

(mb@kali)~[~]
$ ssh anonymouse@* * * * *
anonymouse@* * * * *'s password:
Permission denied, please try again.
anonymouse@* * * * *'s password:
Permission denied, please try again.
anonymouse@* * * * *'s password:
Permission denied (publickey,password).

(mb@kali)~[~]
$ ssh anonymouse@* * * * *
anonymouse@* * * * *'s password:
Permission denied, please try again.
anonymouse@* * * * *'s password:
Permission denied, please try again.
anonymouse@* * * * *'s password:
Permission denied (publickey,password).

(mb@kali)~[~]
$ nmap -s -p 22 * * * * *
Starting Nmap 7.95 ( https://nmap.org ) at 2025-10-10 19:41 EDT
Nmap scan report for * * * * * ( * * * * *)
Host is up (0.00051s latency).

PORT      STATE      SERVICE
22/tcp    filtered  ssh

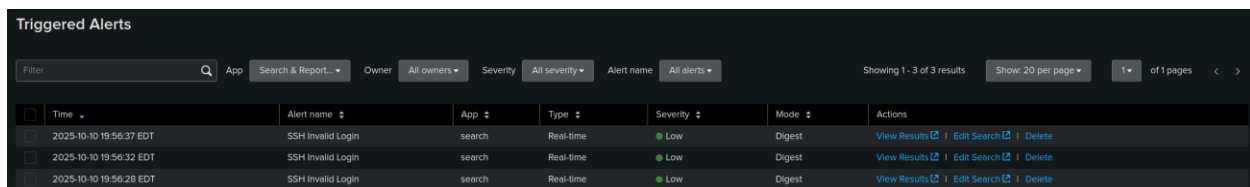
Nmap done: 1 IP address (1 host up) scanned in 0.50 seconds

(mb@kali)~[~]
$
```

Figure 12 – An Nmap scan showing a filtered state on port 22.

Step 2: Monitor SSH login alerts in Splunk

While the attack was live, I observed alerts on Splunk. Success! Splunk generated some alerts based on failed SSH logins.



Triggered Alerts						
Filter	App	Search & Report...	Owner	All owners	Severity	All severity
			Alert name	All alerts	Showing 1 - 3 of 3 results	
2025-10-10 19:56:37 EDT	SSH Invalid Login	search	Real-time	Low	Digest	View Results Edit Search Delete
2025-10-10 19:56:32 EDT	SSH Invalid Login	search	Real-time	Low	Digest	View Results Edit Search Delete
2025-10-10 19:56:28 EDT	SSH Invalid Login	search	Real-time	Low	Digest	View Results Edit Search Delete

Figure 11 – A view of custom Splunk alerts.

This concludes the experiment.

Outcome

As my first project, it was enjoyable to test my machine and harden it. There was a lot that I learned, and I look forward to performing more experiments in the future.

Lessons Learned

The following is a summary of what I learned in this project:

1. Splunk
 - a. How to add data specify which files to monitor
 - b. How to search for logs using regex, specifying the source
 - c. How to extract fields from logs
 - d. How to create and view alerts
2. SSH
 - a. SSH's built in brute force protection
 - b. How to download and start SSH
3. Virtual Machine
 - a. How the virtual machine gets Wi-Fi connection implicitly through the host
4. Iptables
 - a. Overall syntax
 - b. Listing rules
 - c. Creating new rules

I used online resources and Gemini Pro to learn and practice for this project.

Recommendations

For further hardening of my SSH connections, I could disable root login and require Multi-Factor Authentication (MFA). Disabling root login would prevent attackers from immediately entering a privileged shell, and MFA would further mitigate brute-force attacks.

For future projects, I will use AI more for learning how to use tools, and online video tutorials on how to implement security controls. I was reading a lot of articles that did not directly apply to my situation and extended the project work time.

References

[1]

D. Miessler, "Monitoring SSH Bruteforce Attempts Using Splunk," *Danielmiessler.com*, 2017. <https://danielmiessler.com/blog/monitoring-ssh-bruteforce-attempts-using-splunk> (accessed Oct. 11, 2025).