NETWORKING IN LINUX

Spoofing a MAC ID

It is a real hassle for cable Internet customers that they are restricted to using a single machine for Internet access. If you want to plug your laptop in your friend's cable Internet connection, you have to call the service provider to refresh the MAC address.

The MAC address is permanent to the hardware and cannot be changed. Since we operate the hardware via the software abstraction layer, it is quite possible to do some software-level cheating for the network card's MAC ID. We can simply spoof it to some other MAC addresses.

You can obtain the original MAC ID from the ifconfig output. Mine is as follows:

```
 eth0      Link encap:Ethernet  HWaddr 00:1C:23:FB:37:22
```

Now, let's change the last part of the MAC ID from 22 to 23:

```
ifconfig eth0 hw ether 00:1C:23:FB:37:23
```

Now, run ifconfig again:

```
[root@gnubox slynux]# ifconfig eth0
eth0    Link encap:Ethernet  HWaddr 00:1C:23:FB:37:23
          inet addr:192.168.0.2  Bcast:192.168.0.255  Mask:255.255.255.0
          BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
          Interrupt:17
```

Easy enough? Well, let's consider the following instance: suppose you are in a Wi-Fi campus and the access to the wireless network is restricted using MAC addressing. You can simply look at your friend's laptop MAC ID and spoof it. Yes! You are now free to access the network. Have fun!

DNS (Domain Name Service)

DNS is responsible for name resolution. When you point your browser to www.google.com, it points to a server on the Internet. How does that happen? As you are aware, all networked computers are assigned with IP addresses. But how do you access the Web page hosted in Google's remote machine by simply typing a name like google.com?

That phenomenon is achieved using domain name resolution. There are some servers on the Web called name servers (or DNS) that resolve certain names to corresponding IP addresses, like google.com to its corresponding IP address, in our case. So, we should have the IP addresses of the DNS servers (generally provided by the Internet service provider) handy so that we don't have to remember everyone else's when we browse the Web. When we point our browser to google.com, it consults one of these name servers to find out the IP address and thus load the Web page. But where do we configure the IP addresses of these name servers?

If your network is configured with DHCP, there is no need to specify the name server explicitly. For static IPs, it is, however, necessary. We enter the DNS servers' IP addresses in the /etc/resolv.conf file. Mine looks like the following:

```
nameserver 208.67.222.222

nameserver 208.67.220.220
```

Note that you don't really need to use the DNS addresses provided by your ISP. For safety purposes, I use OpenDNS—the IP addresses are listed in the above snippet. You can learn more on why OpenDNS is a much safer bet at www.opendns.org.

SSH (Secure Shell)

SSH can be defined as the blood of *nix networks. SSH enables users and administrators to make remote logins to other machines that are connected through any kind of network. If you know the user name, password and IP address of another machine on the network, you can remotely log in to that machine and work on it as if you are actually working in front of that machine. The following is an example in which I'm authenticating to a system with the IP address of 192.168.1.3 as the user test:

```
[root@gnubox ~]# ssh test@192.168.1.3
The authenticity of host '192.168.1.3 (192.168.1.3)' can't be established.
RSA key fingerprint is 9f:61:ae:ac:8f:75:bb:3a:02:4a:f4:6c:7d:b9:0d:07.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.1.3' (RSA) to the list of known hosts.
test@192.168.1.3's password:
-sh-3.2$ echo I am on 192.168.1.3 Machine
I am on 192.168.1.3 Machine
-sh-3.2$
```

You can open the CD tray of the other machine, close the tray, shutdown, reboot the machine — depending on the privileges the user name you've logged in with, has.

sftp is an extension to the ssh protocol that helps us to use the SSH connection to transfer files between machines. The following is an example:

```
[root@localhost ~]# sftp test@192.168.1.3
Connecting to 192.168.1.3...
test@192.168.1.3's password:
sftp> ls
Desktop    Documents  Download   Music      Pictures   Public     Templates
Videos     a.out      test.bin     file.cpp   t.c
sftp> get t.c
Fetching /home/test/t.c to t.c
/home/test/t.c                                100%  239     0.2KB/s   00:00
sftp>
```

To download a file from the remote machine we use the get command, and to upload a file, we use put. In the above snippet you can see that I'm downloading a file named t.c using the get command, after logging in to the remote machine using sftp.

sshfs is another extension to SSH, which empowers you to mount directories on a remote machine as a filesystem to a specified mount point:

```
root@localhost ~]# sshfs test@192.168.1.3:/home/test /mnt/test

test@192.168.1.3's password:
```

In the above snippet, I'm mounting the home directory of the user 'test' on 192.168.1.3 to my local machine under the /mnt/test directory.

Proxy server configuration

Many of us on a college campus or office network access the Internet through a proxy server. How do you set the proxy server details in your shell environment? You can set the proxy for different protocols as follows:

```
export http_proxy="http://192.168.0.1:3128" ; // HTTP proxy

export ftp_proxy="192.168.0.1:3128" ; //FTP proxy
```

If you want these settings to be permanent, each time you log in add these lines to your ~/.bash_profile file.
That's all, folks! Hope you have enjoyed learning the secrets of networking. Happy hacking till we meet again!

Source : http://www.opensourceforu.com/2009/02/recipes-for-networking/