**Class 8**

In this exercise, we will accomplish the following tasks:

* Generate SSH public/private keys with a passphrase and using RSA encryption
* Use Puttygen to convert our private key to a .ppk file so that we can test it using Putty or WinSCP
* Test that we able to SSH using our private key
* Disable password authentication by editing /etc/ssh/sshd\_config
* SSH from one server to another without having to enter a password

**Generate a SSH public/private key pair**

Generating a SSH public/private key pair allows a user to login to a server with his/her own private key for extra security. This private key can also be protected (called a passphrase) in case somebody gets access to that private key.

First create the public/private key pair using:

**ssh-keygen –t rsa**

This will create two files in your (hidden) ~/.ssh directory (note that ~ means the home folder for that given user, most users are stored under /home by default but the root user is under /root) called **id\_rsa** (this is the private key) and **id\_rsa.pub** (this is the public key)

If you don't want to still be asked for a passphrase (which is basically a password to unlock a given public key) each time you connect, just press **enter** when asked for a passphrase when creating the key pair. It is up to you to decide whether or not you should add the passphrase protective encryption to your key when you create it. If you don't passphrase protect your key, then anyone gaining access to your local machine will automatically have ssh access to the remote server. Also, root on the local machine has access to your keys although one assumes that if you can't trust root (or root is compromised) then you're in real trouble. Encrypting the key adds additional security at the expense of eliminating the need for entering a password for the ssh server only to be replaced with entering a passphrase for the use of the key.

Now set permissions on the private key:

**chmod 700 ~/.ssh**

**chmod 600 ~/.ssh/id\_rsa**

Copy the public key (id\_rsa.pub) to the server and install it to the authorized\_keys list:

**cat id\_rsa.pub >> ~/.ssh/authorized\_keys**

Set file permissions on the server:

**chmod 600 ~/.ssh/authorized\_keys**

The above permissions are required if StrictModes is set to yes in **/etc/ssh/sshd\_config** (the default).

Restart the SSH service:

**service sshd restart**

**Side notes**: remember to navigate to a directory use the **cd** command (example cd /root) and then to list files that are hidden you can use **ls –hal** to see that the .ssh file is in fact there (.ssh is a hidden file).

**Using Puttygen to convert our file to a .ppk to test with Putty or WinSCP**

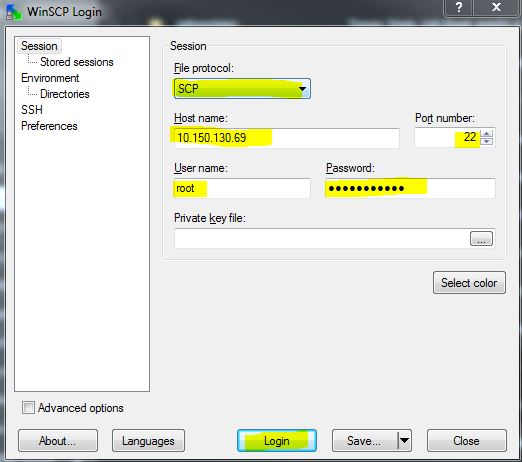
First download the following programs (WinSCP is the only that needs to be installed, rest can just be executed on the fly):

WinSCP: <http://winscp.net/eng/download.php> (click on installation package)

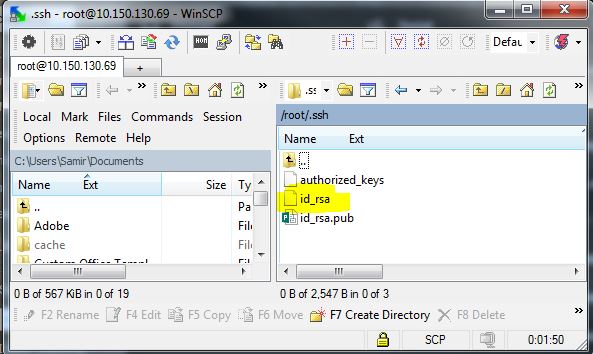
Putty: <http://the.earth.li/~sgtatham/putty/latest/x86/putty.exe>

Puttygen: <http://the.earth.li/~sgtatham/putty/latest/x86/puttygen.exe>

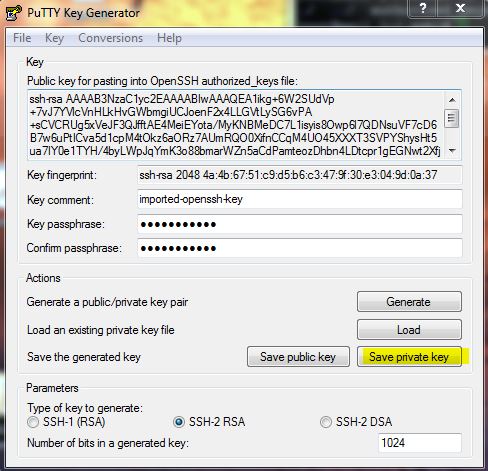
Connect to the server using WinSCP (note that you need to use the **SCP** protocol and port 22, enter the IP, user, and password)

****

Now go to the **/root/.ssh** directory and download the **id\_rsa** file (you can drag and drop it to the left or to your desktop)

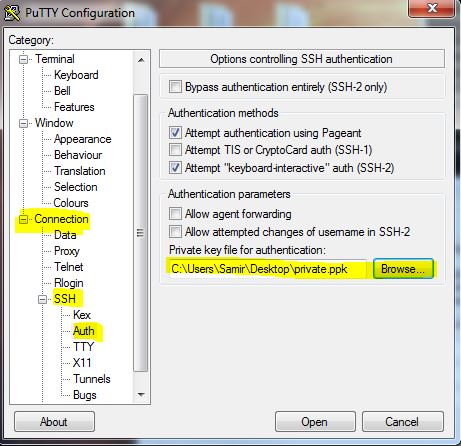


Open Putty Key Generator (Puttygen.exe) and hit “Load” then point to the id\_rsa file you have just download (it will ask you for the passphrase if you set one) and it should say “Successfully imported foreign key.” Now click on “Save private key” and give your private key a name.

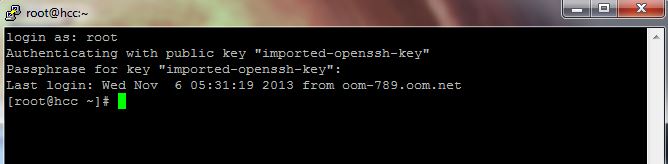


Open Putty and put your IP for the Host Name.

On the left menu, go under Connection -> SSH -> Auth and browse for your private key file



After you hit open, you should be able to SSH using your private key.



**Disable password authentication**

By having a private key, we can disable password authentication to our server all together by editing the **/etc/ssh/sshd\_config** file (change the password authentication from yes to no):

#Disable password authentication forcing use of keys

**PasswordAuthentication no**

Note: \*Do this once you are sure that you can successfully login to your server, otherwise you could get locked out!\*

**SSH from one server to another without needing a password**

Refer to this guide for information: <http://www.tecmint.com/ssh-passwordless-login-using-ssh-keygen-in-5-easy-steps/>

**Ifconfig** – UNIX version of ipconfig – to see what current IP configuration & network devices are set

**OLD for 6.5: /etc/sysconfig/network-scripts** – config files to specify the IP ranges & network interfaces

**New for 7.0: nmtui**  - this will open the network manager

**Netstat** – check incoming connections, good to see what ports are listening

Example: netstat –anl

**Iptables** is the default firewall for UNIX – you can use this to block IPs and much more

Recommend using this firewall (it’s free) to secure the server: <http://configserver.com/cp/csf.html>

Examples:

iptables -A INPUT -s 192.168.246.118 -j DROP

service iptables save

<http://www.cyberciti.biz/faq/how-do-i-block-an-ip-on-my-linux-server/>

**Firewalld** – this is the new default firewall for Centos 7 which replaces iptables. You can still disable firewalld and replace it with iptables if you are more comfortable with it. URL for doing that: <https://linuxize.com/post/how-to-install-iptables-on-centos-7/>

**DNS** – Domain Name System - big topic to look into. This is how domain names get translated into IPs. Basically the yellow book of the Internet. When we type google.com – DNS is used so that we can be directed to a server.

<http://en.wikipedia.org/wiki/Domain_Name_System>

the **dig** command is very useful for DNS troubleshooting. On our VPS, we must first install the command using yum (yum install bind-utils).

Service for DNS is called **bind** or **named**

examples of use:

**dig @ns1.google.com google.com** - this command looks up google.com on Google’s nameservers

**dig +trace google.com** – this command traces every network hop to get to Google.com.

**/etc/hosts** – This configuration file stores hostnames and IPs. It can be useful for local naming use when there is not DNS server available or public access. Back when the Internet was just a few machines (Arpanet), this file stored the IPs of all the machines it talked to – this is why DNS came about.

Windows’ version: http://www.timeatlas.com/term\_to\_learn/general/taking\_control\_with\_the\_hosts\_file#.UWSv2Rcp\_oI

**Resolver** (/etc/resolv.conf) – This is the configuration file where you can set the DNS resolvers, aka who serves DNS for your server when it does lookups. You could use public ones such as Google’s (8.8.8.8). Think of this as who provides you the yellow book service.

Note: in 6.5 you can use /etc/resolv.conf to clean up the config. You now use nmtui (Network Manager) for Centos 7.0

**Namebench**: Neat tool from Google to find out what would be optimal DNS resolvers depending on which network you are connected to: <http://code.google.com/p/namebench/> - if you use faster DNS resolvers, then you will have a faster surfing experience.

Change your resolvers in Windows: <http://www.mediacollege.com/computer/network/dns.html>

**Whois** – to dig up public information (if available) for a website name or IP address

Centos 6.5: yum install jwhois (you need the service to run the command)

Centos 7.0: yum install whois