# Basic SSH Security

SSH is a powerful command line remote administration utility. It is also used to create secure tunnels for file transfer (scp, sftp), GUI remote administration, and many other tasks. Since SSH is powerful, it is often attacked. Any public IP address open to SSH on TCP port 22 will almost certainly scanned and attacked with basic brute force login attempts. Current scan data is available at <https://isc.sans.edu/port.html?port=22> and top passwords currently attempted are available here: <https://isc.sans.edu/ssh.html>. Please visit both those sites.

One method in use today to reduce exposure to SSH brute force password attempts is to change the port number to something different than 22. While this does help, it is more akin to “security through obscurity” than true security. Another method is to make users create long, difficult passwords. This works, but there always seem to be users that manage to evade requirements.

An effective way to secure SSH is to use public/private key pairs instead of usernames and passwords. The administrator of a machine using SSH only allows login by those users whose public key is saved on the server. The users authenticate with their private key, which they keep secure at all times. We will cover the cryptography of public and private key pairs in a later class.

In key-based authentication, someone must generate the key pair the user will use. If the users control the server running the SSH daemon, they will usually generate the key pair themselves. After generating the key pair, they save the private key (securely) on their client workstations and save a copy of the public key on the server. Some organizations may generate the key pair for the users, distribute the private key to the users and save the public key on the server. Larger organizations will have Certificate Authorities (CA) and a Public Key Infrastructure (PKI) to distribute and control certificates and keys.

Linux and most cloud services have SSH and the necessary tools built in, so we will discuss them first. For Windows we will discuss Putty and the new OpenSSH implementation in Windows 10.

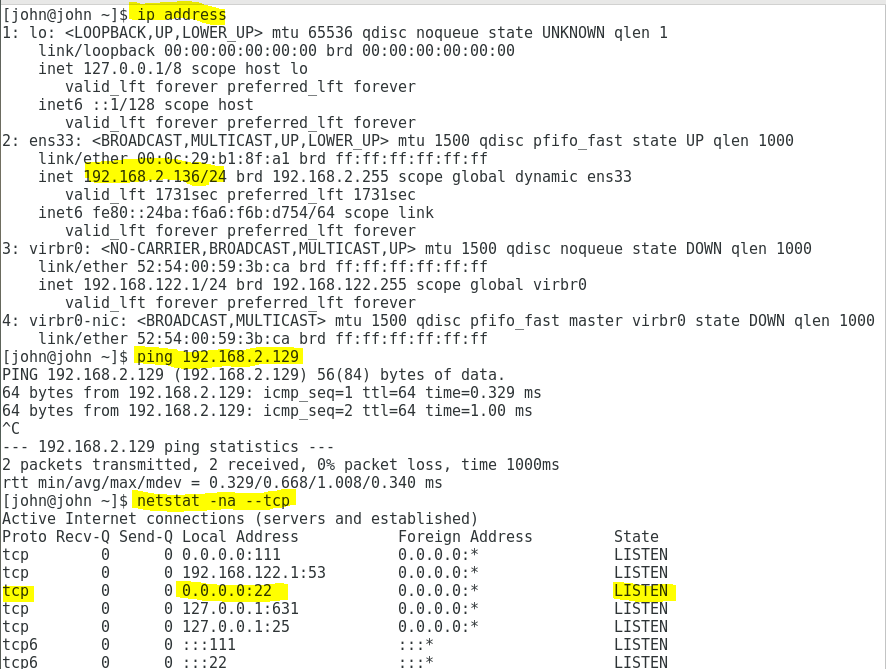
Please read this RedHat description of the SSH protocol: <https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/6/html/deployment_guide/ch-openssh#s1-ssh-protocol>. Note: On Linux, especially in configuration files, ssh refers to an SSH client and sshd refers to the SSH daemon, or server.

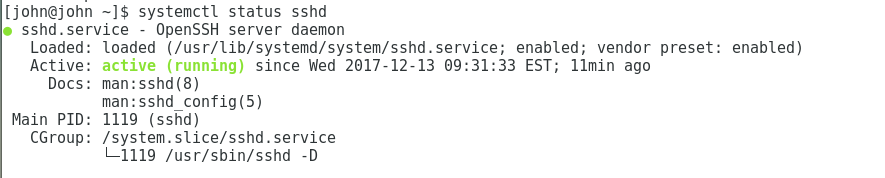
# Lab: SSH on Linux

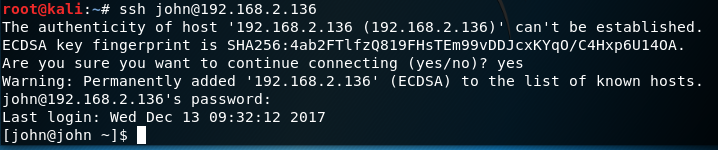
For this lab, we will practice SSH connections from our Kali VM (SSH client) to our CentOS VM (SSH server.) We will create the key pair on the Kali VM and then copy the public key to the CentOS VM, but it could just as easily be done the other way around. We will perform these steps:

* Test SSH from the Kali VM to the CentOS VM using a CentOS username/password
* Create a public/private key pair on the Kali VM
* Add the public key in the .ssh/authorized\_keys file of the CentOS user.
* Test SSH from the Kali VM to the CentOS VM using the key pair
* Disable username/password authentication for SSH on the CentOS VM.

## SSH from your Kali VM to your CentOS VM with username and password

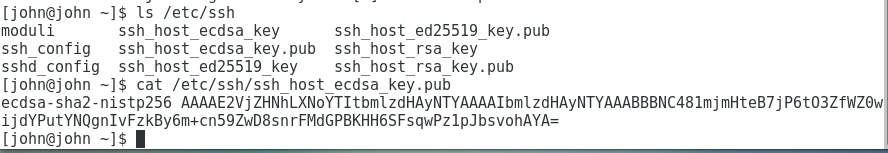
Make sure that your VMs are on the same VMware host network, that their IP addresses are on the same subnet, and that they can ping each other. You can use either the command ipconfig or the newer command ip address to verify IP addresses. Also use netstat on the CentOS VM to confirm that it is listening on TCP port 22. The screenshot below is from the CentOS VM. 

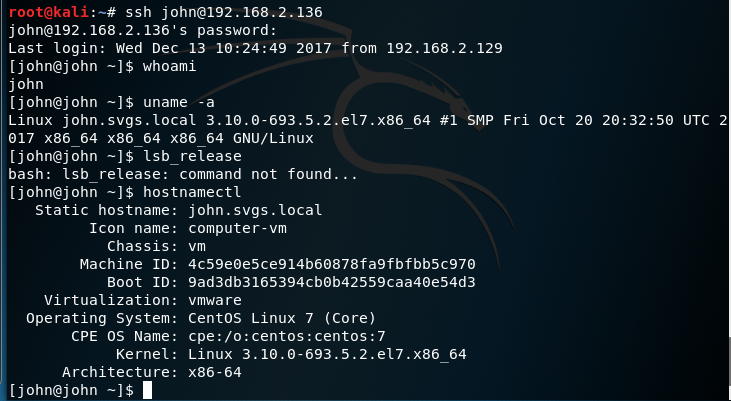
You can also verify that the sshd service is running with the command systemctl status sshd. 

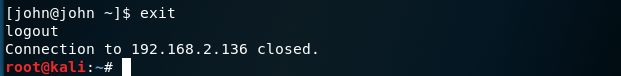
From your Kali VM, use ssh --help or man ssh study the syntax. Then create an SSH connection to the CentOS VM. The username should be that of a user that exists on the CentOS VM. 

The SSH client verifies the server’s public key by checking to see if it is listed in the file ~/.ssh/known\_hosts or /etc/ssh/known\_hosts. If the key is not in the list, the client gives you a warning that the authenticity of the server cannot be established. If you are certain that you are connected to the correct server, answer yes and the key will be added to the known\_hosts file.

Another method of verifying the host server would be for the owner of the server to make the fingerprints of the public keys available so you can check the fingerprint against the one in the warning. Or, they can make the public keys available so that you can add them to the known\_hosts file yourself. (The server key pairs are kept in /etc/ssh. The ones ending in .pub are the public keys.)



Once you have connected, experiment with some commands. The results should be the same as if you typed them from the CentOS console. (In the screenshot, uname -a, lsb\_release, hostnamectl, and hostname are commands that can show information about the Linux host. This just demonstrates that we really are executing commands on the CentOS VM.

When you are done, close the connection. 

## Create a user key pair on the Kali VM

On the Kali VM, create a public/private key pair for your user with the procedure shown here: <https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/6/html/deployment_guide/s2-ssh-configuration-keypairs>. Only create the first key, RSA for SSH v2. Do not create DSA or SSH v1 keys.

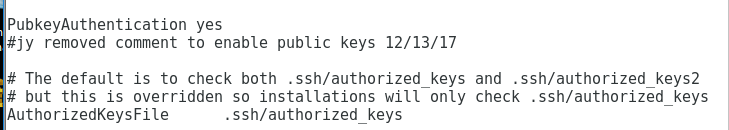
Important points to note:

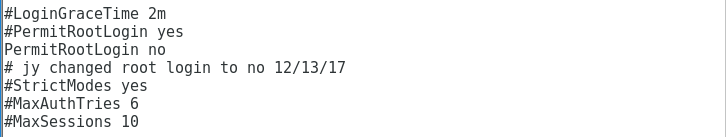
* There are two versions of SSH, v1 and v2. Always use SSH v2. (Here’s a video of Trinity using Nmap and an SSH v1 exploit in The Matrix Reloaded: <https://www.youtube.com/watch?v=0PxTAn4g20U>)
* Use RSA (-t rsa) keys, as DSA (-t dsa) keys have been deprecated and -t rsa1 generates SSH v1 keys. Other possibilities are -t ecdsa and -t ed25519, which provide somewhat more security than rsa providing the server supports them.
* The standard storage location for user SSH keys in Linix is ~/.ssh/
* The .ssh directory should always be rwx------, full rights for the user and no rights for anyone else. That’s what the step chmod 700 ~/.ssh did. If someone else could read your keys, they could impersonate you.
* The passphrase you enter during key generation protects the private key so that only you can use it. Other than allowing the key to be used, it does not affect authentication.
* The root user should never be allowed to login via SSH; always log in with a standard user and use su or sudo when root privileges are needed.

## Configuring the SSH Server for public keys

The SSH daemon, sshd, on the CentOS VM is configured using the file, /etc/ssh/sshd\_config. The easiest way to edit the file is to use nano.

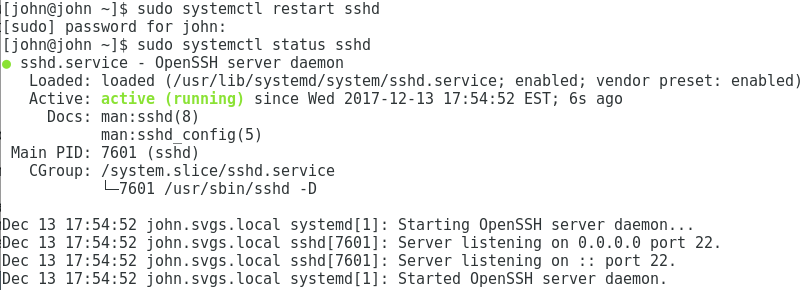


To allow sshd to authenticate public keys, we need to change one line in sshd\_config. The comment in front of “PubkeyAuthentication yes” needs to be removed. Also note that the AuthorizedKeysFile setting will cause the SSH daemon to look for public keys in the users’ .ssh/authorized\_keys file. It is a good idea to add comments of your own whenever you change configuration files so you can trace the changes later on. 

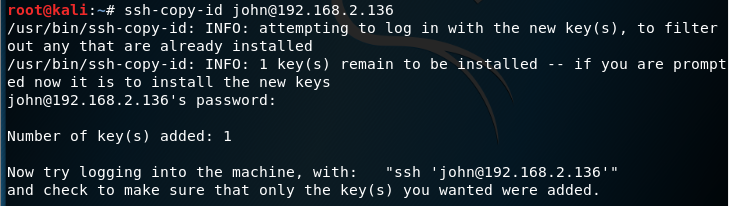
Also edit PermitRootLogins to change it to no. 

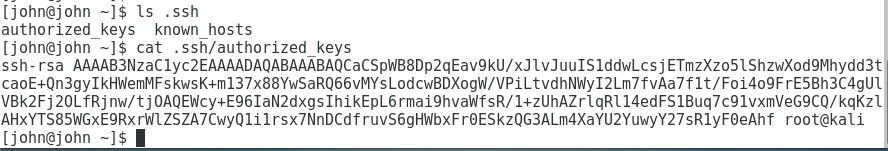
You can see explanations and defaults for the sshd\_config parameters by running man sshd\_config

Once you have saved the changes to sshd\_config, you will need to restart sshd.



Copy the public key to the server

Once sshd is configured, you still need to enter a copy of the user’s public key in the user’s ~/.ssh/authorized\_keys file. One way to do that is to use ssh-copy-id. It explained in Option 1 of this link: <https://www.digitalocean.com/community/tutorials/initial-server-setup-with-centos-7>. To use this method, you must be able to SSH from your client to the server using username/password authentication (unless some other key is already on the server.) You may ask, why are we using password authentication when the objective is to turn that off? Good question. Using ssh-copy-id is the easiest option, and reasonable during initial configuration before the server is connected to the Internet. Feel free to manually create and configure the authorized\_keys file as in Option 2 of the link. It may be easier to try ssh-copy-id first, and then view the authorized\_keys file to see the format before you edit it manually.

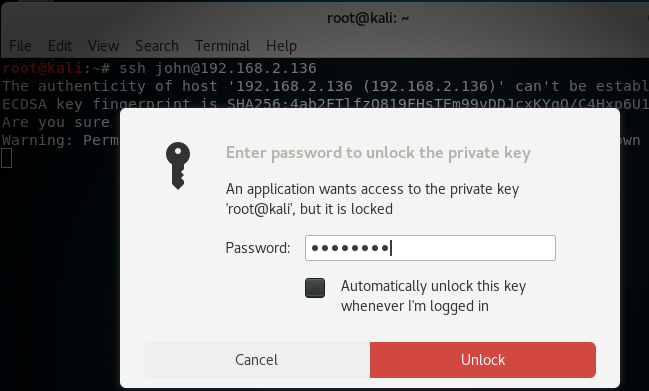
The key is now on the CentOS VM. 

Note: On servers with many users, the sysadmin may want tighter control over the keys that allow SSH access. In that case, they will prevent sshd from using keys in users’ .ssh directories by setting the AuthorizedKeysFile parameter in sshd\_config to none. Then, sshd will only look for public keys in /etc/ssh/authorized\_keys, which requires root privileges to access. Also, the sysadmin can create a group (allowssh, for example) and set the AllowGroups parameter to the group name.

## SSH to the server using a public key (Finally)

Now, when you SSH from the Kali VM to the CentOS VM, you should be authenticated automatically. If you did not put a password on your private key, you will see this: 

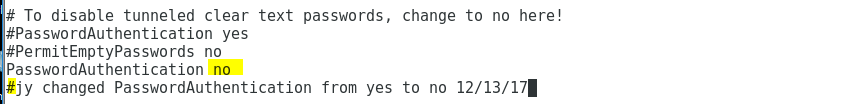
The SSH client on the Kali VM used the private key in the .ssh directory to authenticate, and sshd on the CentOS VM accepted the authentication because it has the public key.

If your private key is protected with a password, you will see this: 

Once you enter the password to unlock the private key, you should immediately be logged in to the CentOS VM through SSH.

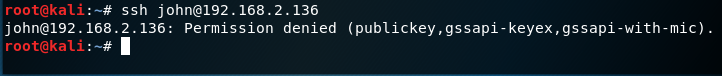
## Disable Password Authentication

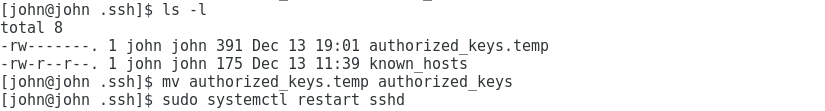
Finally, we need to disable password authentication so that only public key authentication is allowed. We need to edit /etc/ssh/sshd\_config on the CentOS VM to change the PasswordAuthentication parameter to no. Again, commenting changes to configuration files is a good thing.



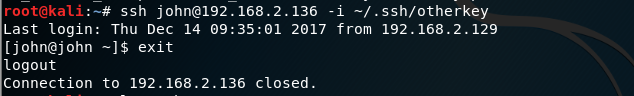
Be sure to save your changes to the sshd\_config file, and then restart the sshd daemon.

We can test this change by temporarily changing the name of the authorized\_keys file in our user’s .ssh directory on the CentOS VM. Since the sshd daemon will not find an authorized key, it should deny authentication and not allow us to authenticate with a password.

It worked! 

Let’s change the authorized\_keys.temp file back to its correct name so SSH works again.  

## Multiple Keys

You may have access to several servers, each with its own key. In that case, you can specify which key the SSH client should use with the -i (identity) switch. 

# Windows SSH Clients

The most popular SSH Client for Windows is Putty, available at <https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html>. However, openssh is now available in Windows 10 ver. 1803 (April 2018 update) and later in PowerShell. It works just like the Linux version.  
