

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 be 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 require that 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 always keep secure. 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 add the public key to the `authorized_keys` file in the user's `.ssh` directory 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. Windows 10 ver. 1803 (April 2018 update) and later now include `openssh` by default.

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.

SSH Lab

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

- Test SSH from the Windows host or VM to the Ubuntu VM using a username/password from the Ubuntu VM
- Create a public/private key pair on the Windows host or VM
- Append the public key in the `.ssh/authorized_keys` file of the Ubuntu user.
- Test SSH from the Windows host or VM to the Ubuntu VM using the key pair
- Disable username/password authentication for SSH on the Ubuntu VM.

Install SSH server (sshd) on the Ubuntu VM

Ubuntu server includes sshd by default, but the desktop version we are using does not. We can find the name of the software to install in a couple of ways. If we try to execute sshd, Ubuntu helpfully tells us how to install it.

(Note: If you installed ssh in an earlier lab and disabled it, you can re-enable it using `systemctl enable ssh`)

```
svgs@svgs-ubuntu-18:~$ sshd

Command 'sshd' not found, but can be installed with:

sudo apt install openssh-server
```

The more orthodox method is to search using apt.

```
svgs@svgs-ubuntu-18:~$ apt search sshd
Sorting... Done
Full Text Search... Done
fail2ban/bionic,bionic 0.10.2-2 all
  ban hosts that cause multiple authentication errors

libconfig-model-cursesui-perl/bionic,bionic 1.106-1 all
  curses interface to edit config data through Config::Model

libconfig-model-openssh-perl/bionic,bionic 1.238-1 all
  configuration editor for OpenSsh

libconfig-model-tkui-perl/bionic,bionic 1.365-1 all
  Tk GUI to edit config data through Config::Model

openssh-server/bionic-updates,bionic-security 1:7.6p1-4ubuntu0.1 amd64
  secure shell (SSH) server, for secure access from remote machines

python-logparser/bionic,bionic 0.4-1 all
  Python library for log parsing, tagging and analysis.

tinysshd/bionic 20180201-1 amd64
  Tiny SSH server - daemon
```

Install sshd.

```
sudo apt-get install openssh-server
```

Check that sshd is running with `systemctl status sshd`

```
svgs@svgs-ubuntu-18:~$ systemctl status sshd
● ssh.service - OpenBSD Secure Shell server
   Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enab
   Active: active (running) since Fri 2019-01-18 10:34:17 EST; 3min 42s ago
 Main PID: 3432 (sshd)
   Tasks: 1 (limit: 4634)
  CGroup: /system.slice/ssh.service
          └─3432 /usr/sbin/sshd -D

Jan 18 10:34:17 svgs-ubuntu-18 systemd[1]: Starting OpenBSD Secure Shell server.
Jan 18 10:34:17 svgs-ubuntu-18 sshd[3432]: Server listening on 0.0.0.0 port 22.
Jan 18 10:34:17 svgs-ubuntu-18 sshd[3432]: Server listening on :: port 22.
Jan 18 10:34:17 svgs-ubuntu-18 systemd[1]: Started OpenBSD Secure Shell server.
svgs@svgs-ubuntu-18:~$
```

We can verify that the server is listening on TCP port 22 with `ss -nat` (ss is the replacement for netstat.)

```
svgs@svgs-ubuntu-18:~$ ss -nat
State      Recv-Q    Send-Q    Local Address:Port    Peer Address:Port
LISTEN     0          128       127.0.0.53%lo:53      0.0.0.0:*
LISTEN     0          128       0.0.0.0:22            0.0.0.0:*
LISTEN     0           5        127.0.0.1:631         0.0.0.0:*
LISTEN     0          128       [::]:22               [::]:*
LISTEN     0           5        [::1]:631             [::]:*
```

SSH from your Windows host or VM to your Ubuntu 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 the command `ip address` to verify the IP address on Linux and the command `ipconfig` on Windows

```
svgs@svgs-ubuntu-18:~$ ip address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:67:62:40 brd ff:ff:ff:ff:ff:ff
    inet 192.168.183.141/24 brd 192.168.183.255 scope global dynamic noprefixroute ens33
        valid_lft 1190sec preferred_lft 1190sec
    inet6 fe80::fbf3:6296:7287:e2e4/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
```

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

VM.

```
PS C:\Users\John> ssh svgs@192.168.183.141
The authenticity of host '192.168.183.141 (192.168.183.141)' can't be established.
ECDSA key fingerprint is SHA256:EjYmNxvkOuGyryoLmP20jTSyvQK0Pgzc7CqEt3PvJPc.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.183.141' (ECDSA) to the list of known hosts.
svgs@192.168.183.141's password:
Permission denied, please try again.
svgs@192.168.183.141's password:
Welcome to Ubuntu 18.04.1 LTS (GNU/Linux 4.15.0-43-generic x86_64)

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

 * Canonical Livepatch is available for installation.
   - Reduce system reboots and improve kernel security. Activate at:
     https://ubuntu.com/livepatch

0 packages can be updated.
0 updates are security updates.
```

The SSH server gives its public key to the SSH client so the client knows that it is connecting to the correct server. 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` in Linux or in `C:\Users\[username]\.ssh\known_hosts` in Windows 10. 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 (the first time) 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.)

```
svgs@svgs-ubuntu-18:~$ ls /etc/ssh
moduli      ssh_host_ecdsa_key      ssh_host_ed25519_key.pub  ssh_import_id
ssh_config  ssh_host_ecdsa_key.pub  ssh_host_rsa_key
sshd_config ssh_host_ed25519_key    ssh_host_rsa_key.pub
svgs@svgs-ubuntu-18:~$
```

Once you have connected, experiment with some commands. The results should be the same as if you typed them from the Ubuntu 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 Ubuntu VM.

```
svgs@svgs-ubuntu-18:~$ uname -a
Linux svgs-ubuntu-18 4.15.0-43-generic #46-Ubuntu SMP Thu Dec 6 14:45:28 UTC 2018 x86_64 x86_64_ubuntu-18.04.1
svgs@svgs-ubuntu-18:~$ lsb_release
No LSB modules are available.
svgs@svgs-ubuntu-18:~$ hostnamectl
  Static hostname: svgs-ubuntu-18
            Icon name: computer-vm
            Chassis: vm
            Machine ID: e20014dac22c49d4a222491686cc7d7d
            Boot ID: 3d40f9e92c3a458ea67e5da97018df55
    Virtualization: vmware
    Operating System: Ubuntu 18.04.1 LTS
            Kernel: Linux 4.15.0-43-generic
    Architecture: x86-64
svgs@svgs-ubuntu-18:~$ hostname
svgs-ubuntu-18
```

When you are done, close the connection.

```
svgs@svgs-ubuntu-18:~$ exit
logout
Connection to 192.168.183.141 closed.
PS C:\Users\John>
```

Create a user key pair on the Windows host or VM

On the Windows host or 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. Do not create DSA or SSH v1 keys as they are not secure.

Important points to note:

- There are two versions of SSH, v1 and v2. Always use SSH v2. (Here's a video of Trinity shutting down a power grid using Nmap and an SSH v1 exploit in The Matrix Reloaded: <https://www.youtube.com/watch?v=0PxTAn4g20U>)
- Don't use `-t dsa` or `-t rsa1`, as DSA (`-t dsa`) keys have been deprecated and `-t rsa1` generates SSH v1 keys. More secure choices are `-t ecdsa` and `-t ed25519`, which provide. You can use `-t rsa`, but it is less secure and may be deprecated in the future; if the server only supports RSA keys, you'll have to use this however.
- The standard storage location for user SSH keys in Linux 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 Ubuntu VM is configured using the file, `/etc/ssh/sshd_config`. The easiest way to edit the file is to use `nano`.

```
svgs@svgs-ubuntu-18:~$ sudo nano /etc/ssh/sshd_config
```

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. Also edit PermitRootLogins to change it to no.

```
PermitRootLogin no
#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10

PubkeyAuthentication yes
#uncommented JY 1/18/19
# Expect .ssh/authorized_keys2 to be disregarded by default in future.
#AuthorizedKeysFile .ssh/authorized_keys .ssh/authorized_keys2
#AuthorizedPrincipalsFile none
```

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.

```
svgs@svgs-ubuntu-18:~$ sudo systemctl status sshd
● ssh.service - OpenBSD Secure Shell server
   Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enabled)
   Active: active (running) since Fri 2019-01-18 11:36:03 EST; 7s ago
     Process: 3979 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS)
    Main PID: 3980 (sshd)
      Tasks: 1 (limit: 4634)
     CGroup: /system.slice/ssh.service
             └─3980 /usr/sbin/sshd -D

Jan 18 11:36:03 svgs-ubuntu-18 systemd[1]: Starting OpenBSD Secure Shell server.
Jan 18 11:36:03 svgs-ubuntu-18 sshd[3980]: Server listening on 0.0.0.0 port 22.
Jan 18 11:36:03 svgs-ubuntu-18 sshd[3980]: Server listening on :: port 22.
Jan 18 11:36:03 svgs-ubuntu-18 systemd[1]: Started OpenBSD Secure Shell server.
```

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 from a Linux host is to use `ssh-copy-id`. It is explained in Option 1 of this link: <https://www.digitalocean.com/community/tutorials/initial-server-setup-with-Ubuntu-7>. Unfortunately, `ssh-copy-id` is not included with Windows so we will manually create and configure the `authorized_keys` file as in Option 2 of the link.

The key is now on the Ubuntu VM.

```
[john@john ~]$ ls .ssh
authorized_keys  known_hosts
[john@john ~]$ cat .ssh/authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQCaCSpWB8Dp2qEav9kU/xJlvJuuIS1ddwLcsjETmzXzo5lShzwXod9Mhydd3t
caoE+Qn3gyIkHWemMFskwsK+m137x88YwSaRQ66vMYsLodcwBDXogW/VPiLtvdhNWyI2Lm7fvAa7f1t/Foi4o9FrE5Bh3C4gU
VBk2Fj20LfRjnw/tj0AQEWcy+E96IaN2dxgsIhikEpL6rmai9hvaWfsR/1+zUhAZrlqRl14edFS1Buq7c91vxmVeG9CQ/kqKz
AHxYTS85WGxE9RrxrWlZSZA7CwyQl1rsx7NnDCdfuvS6gHWbxFr0ESkzQG3ALm4XaYU2YuwyY27sRlyF0eAhf root@kali
[john@john ~]$
```

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.

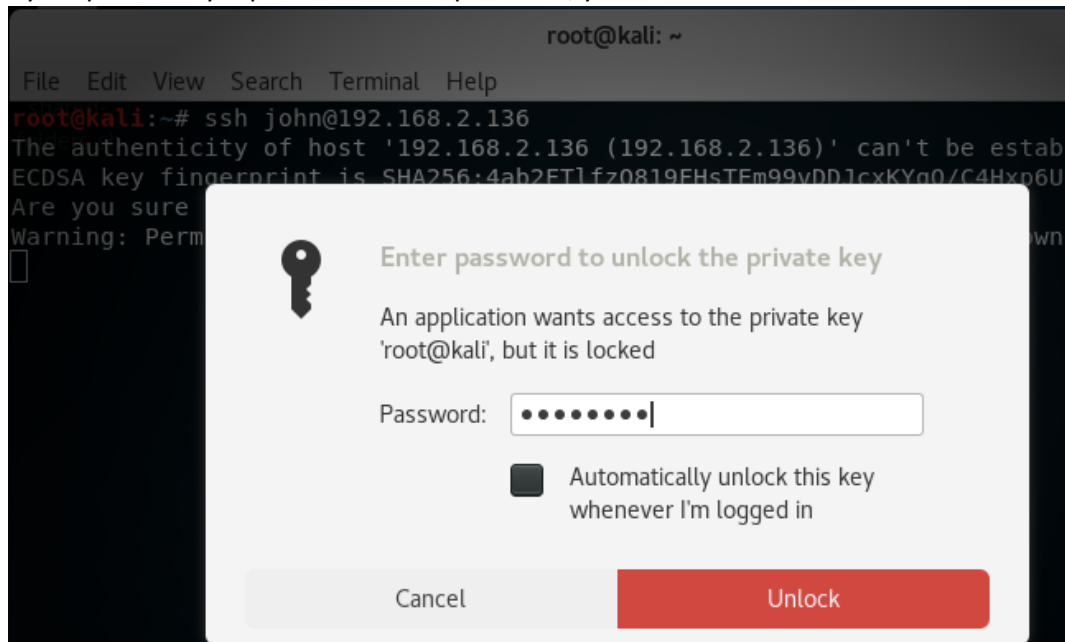
SSH to the server using a public key (Finally)

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

```
root@kali:~# ssh john@192.168.2.136
Last login: Wed Dec 13 18:44:20 2017 from 192.168.2.129
[john@john ~]$
```

The SSH client on the Windows host or VM used the private key in the .ssh directory to authenticate, and sshd on the Ubuntu 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 Ubuntu 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 Ubuntu VM to change the `PasswordAuthentication` parameter to `no`. Again, commenting changes to configuration files is a good thing.

```
[john@john ~]$ sudo nano /etc/ssh/sshd_config
# To disable tunneled clear text passwords, change to no here!
#PasswordAuthentication yes
#PermitEmptyPasswords no
PasswordAuthentication no
#jy changed PasswordAuthentication from yes to no 12/13/17
```

Be sure to save your changes to the `sshd_config` file, and then restart the `sshd` daemon.

```
[john@john ~]$ sudo systemctl restart sshd
[john@john ~]$
```

We can test this change by temporarily changing the name of the `authorized_keys` file in our user's `.ssh` directory on the Ubuntu 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!

```
root@kali:~# ssh john@192.168.2.136
john@192.168.2.136: Permission denied (publickey,gssapi-keyex,gssapi-with-mic).
root@kali:~#
```

Let's change the `authorized_keys.temp` file back to its correct name so SSH works again.

```
[john@john .ssh]$ ls -l
total 8
-rw-----. 1 john john 391 Dec 13 19:01 authorized_keys.temp
-rw-r--r--. 1 john john 175 Dec 13 11:39 known_hosts
[john@john .ssh]$ mv authorized_keys.temp authorized_keys
[john@john .ssh]$ sudo systemctl restart sshd
```

```
root@kali:~# ssh john@192.168.2.136
Last login: Thu Dec 14 09:06:05 2017
[john@john ~]$
```

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.

```
root@kali:~# ssh john@192.168.2.136 -i ~/.ssh/otherkey
Last login: Thu Dec 14 09:35:01 2017 from 192.168.2.129
[john@john ~]$ exit
logout
Connection to 192.168.2.136 closed.
```

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.

 svgs@localhost:~

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\John> ssh svgs@192.168.183.143
The authenticity of host '192.168.183.143 (192.168.183.143)' can't be established.
ECDSA key fingerprint is SHA256:VliHqiOI1L5RFxnfw9t47xGPugNZ6/1vAJ7fgGANKzE.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.183.143' (ECDSA) to the list of known hosts.
svgs@192.168.183.143's password:
Last login: Tue Jan 15 09:27:46 2019
[svgs@localhost ~]$ hostname
localhost.localdomain
[svgs@localhost ~]$ uname -a
Linux localhost.localdomain 3.10.0-957.1.3.el7.x86_64 #1 SMP Thu Nov 29 14:49:43 UTC 2018
[svgs@localhost ~]$
```