Ensuring Linux VM Access After Rotating Keys, System “Updates,” And the Rest

1. The Problem

You have been able to connect to your Linux virtual machine (VM) until one time you try to connect with a terminal via SSH (on a Mac) or PuTTY (on Windows), you get an error similar to

> Network error: Connection refused
> Server sent public key

You may still be able to connect using the VMware Horizon Client, but you know that has availability and reliability issues. You really need to be able to connect with a terminal.

There can be many causes, each with their own fix. Try to determine which scenario applies to you and take the appropriate action.

2. Can’t Connect After Rotating Your Internal Key

2.1. The Reason

Keys at Akamai expire every 120 days. You received notification that your internal key is about to expire and before the current key expired, you created a new key, uploading the public key to the system. A short time later you verified the new key using the instructions given to you in the reply from the NOCC. All is well.

You have most likely rebooted your Mac or Windows laptop and added only your new internal key to your session or removed your old key from your session when you added the new key.

The VM is configured to accept *public key authentication*, allowing permanent access with your internal key, no password required. Your internal public key is used as an *authorized key* stored in file `~/.ssh/authorized_keys` on the VM. In the most basic setup, the `authorized_keys` file contains a copy of your internal public key but there is no automated way to update this when you rotate your internal key. This means as soon as you attempt to connect to your VM without your old internal key added to your laptop’s session (or the old key has expired), the VM will refuse access.

> ⚠ Access via the VMware Horizon Client (needed for X Window System (X11), a GUI Linux desktop on your laptop) uses password authentication, so access here is unaffected.

2.2. The Solution

You need to place your new internal public key into file `authorized_keys`.

Verify your `authorized_keys` file contains only a copy of your old internal public key by running the `diff` command to compare the files. Note that you may need to specify the path of your internal public key if it is located in a subdirectory of the `.ssh` directory. **No output from the command means the files are identical.** If they are different, you will see something like the following:

Bash terminal

```
$ cd ~/.ssh
$ diff -q <internal_key_file>.pub authorized_keys
Files <internal_key_file>.pub and authorized_keys differ
```

Files are identical
Change to your SSH keys directory then copy your internal public key to the `authorized_keys` file. This requires super user privileges and you will be prompted for the password. Then re-run the `diff` command to compare the files to verify they are now the same.

Bash terminal

```
$ sudo cp <internal_key_file>.pub authorized_keys
$ diff -q <internal_key_file>.pub authorized_keys
```

You are done and should now be able to connect to your VM from a terminal (PuTTY in Windows).
Files are different
Change to your SSH keys directory then append your internal public key to the authorized_keys file. This requires super user privileges and you will be prompted for the password.

> ⚠ You may have to become the root user to append your new key instead of only running the command with `sudo`. When you become the root user, the terminal prompt will change from $ (or % in the Z shell) to #. When you are done appending, you must run the `exit` command to return to your normal account.

Bash terminal

```
$ sudo su
[sudo] password for <username>:
# cat <internal_key_file>.pub >> authorized_keys
# exit
$
```

You are done and should now be able to connect to your VM from a terminal (PuTTY in Windows).

2.3. For Advanced Users

Public key authentication is defined in the SSH daemon configuration file `/etc/ssh/sshd_config` by the following parameters:

```
RSAAuthentication yes
PubkeyAuthentication yes
AuthorizedKeysFile %h/.ssh/authorized_keys
PasswordAuthentication no
```

You can force the VM to use password authentication, eliminating the problem of internal key rotation, but always requiring a password by changing `PasswordAuthentication` from `no` to `yes` and adding the line

```
AllowedUsers <your_username>
```

To incorporate changes to `/etc/ssh/sshd_config` you must reload the `systemd` daemon and restart the `ssh` service:

Bash terminal

```
$ sudo systemctl daemon-reload
$ sudo systemctl restart ssh.service
```

> ⚠ **CAUTION**
>
> **If your `authorized_keys` file contains other keys (which is possible), editing this file to delete the old internal public key is fraught with peril and you do so at your own risk!**

But if you are a risk taker, after appending your new key to the file, you should make a backup copy of `authorized_keys` before editing, as the super user. [1]
(# is the root user prompt, not a comment):

Bash terminal

```
$ sudo su
[sudo] password for <username>:
# cat <internal_key_file>.pub >> authorized_keys
# cp authorized_keys authorized_keys_`date +%F`
# vim authorized_keys
# exit
$
```

> ⚠ **IMPORTANT TIP**
>
> In new versions of key verification, the `<newline>` character at the end of the public key file will throw an error, but the `<newline>` is needed to separate entries in the `authorized_keys` file if you append. To append the `<newline>` first, use
>
> `echo "" >> authorized_keys`
>
> after the `sudo su` command.

3. Can't Connect After System Package Updates

3.1. The Reason

After updating your Linux machine with the `sudo apt upgrade` command followed by a reboot, you are subsequently unable to connect to your virtual machine with `ssh` from a terminal on another machine that was previously authorized via your internal key. On a Windows, this means you cannot connect with a PuTTY terminal. However you can still connect to your VM with the VMware Horizon client either from the application or the website at https://mydesktop.corp.akamai.com/.

There is much debate about the exact ultimate cause, but the consensus seems to be problems with `systemd` on certain Ubuntu kernels not correctly creating a required temporary directory `/var/run/sshd` which is a symbolic link to `/run/sshd`.

> ⚠ The SSH server on your VM uses your internal key to authenticate, therefore ensure your internal key is loaded into your session on your local machine before attempting to connect to your VM

3.2. The Solution

First determine is the SSH server is running with the command `ps -ef | grep sshd`.

If the only process returned is the `grep` command, the `sshd` is not running:

```
<username> 15795 22724  0 10:00 pts/   00:00:00 grep sshd
```

Something similar to the below process in addition to the above indicates `sshd` is running:

```
root       22724     1  0 10:00 ?      00:00:00 /usr/sbin/sshd -D
```

Attempting to start the SSH server from the terminal fails. The below example shows what starting `sshd` failing looks like with two commands, `systemctl` status and `journalctl` run to help troubleshoot the problem.

Bash terminal

```
$ sudo systemctl start ssh
Job for ssh.service failed because the control process exited with error code. See "systemctl status ssh.
service" and "journalctl -xe" for details.

$ sudo systemctl status ssh.service
â— ssh.service - OpenBSD Secure Shell server
   Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enabled)
   Active: failed (Result: start-limit-hit) since Thu 2099-01-01 10:30:00 EDT; 30s ago
  Process: 9099 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=255)

Jan 01 10:29:40 systemd[1]: ssh.service: Control process exited, code=exited status=255
Jan 01 10:29:40 systemd[1]: Failed to start OpenBSD Secure Shell server.
Jan 01 10:29:40 systemd[1]: ssh.service: Unit entered failed state.
Jan 01 10:29:40 systemd[1]: ssh.service: Failed with result 'exit-code'.
Jan 01 10:29:40 systemd[1]: ssh.service: Service hold-off time over, scheduling restart.
Jan 01 10:29:40 systemd[1]: Stopped OpenBSD Secure Shell server.
Jan 01 10:29:40 systemd[1]: ssh.service: Start request repeated too quickly.
Jan 01 10:29:40 systemd[1]: Failed to start OpenBSD Secure Shell server.
Jan 01 10:29:40 systemd[1]: ssh.service: Unit entered failed state.
Jan 01 10:29:40 systemd[1]: ssh.service: Failed with result 'start-limit-hit'.

$ sudo journalctl -xe
Jan 01 11:19:33 systemd[1]: ssh.service: Unit entered failed state.
Jan 01 11:19:33 systemd[1]: ssh.service: Failed with result 'exit-code'.
Jan 01 11:19:33 systemd[1]: ssh.service: Service hold-off time over, scheduling restart.
Jan 01 11:19:33 systemd[1]: Stopped OpenBSD Secure Shell server.
-- Subject: Unit ssh.service has finished shutting down
-- Defined-By: systemd
-- Support: http://lists.freedesktop.org/mailman/listinfo/systemd-devel
--
-- Unit ssh.service has finished shutting down.
Jan 01 11:19:33 systemd[1]: Starting OpenBSD Secure Shell server...
-- Subject: Unit ssh.service has begun start-up
-- Defined-By: systemd
-- Support: http://lists.freedesktop.org/mailman/listinfo/systemd-devel
--
-- Unit ssh.service has begun starting up.
Jan 01 11:19:33 sshd[12266]: Missing privilege separation directory: /var/run/sshd
Jan 01 11:19:33 systemd[1]: ssh.service: Control process exited, code=exited status=255
Jan 01 11:19:33 systemd[1]: Failed to start OpenBSD Secure Shell server.
-- Subject: Unit ssh.service has failed
-- Defined-By: systemd
-- Support: http://lists.freedesktop.org/mailman/listinfo/systemd-devel
--
-- Unit ssh.service has failed.
```

The key is in line 37, `Missing privilege separation directory: /var/run/sshd`.

Next, create that directory. Note it should be owned by *root* and `/var/run/sshd` is a symbolic link to `/run/sshd`, Then start the ssh server.

Bash terminal

```
$ sudo mkdir /run/sshd
$ sudo systemctl start ssh
$
```

You can verify the SSH server is running with `ps -ef | grep sshd`. If the SSH server is running, you will now be able to SSH to the VM from a terminal on your local machine.

4. References

Public Key Authorization at SSH.COM.

Ask Ubuntu: *How to Resolve "service start-limit-hit"*.

Ask Ubuntu: *SSH Server stops working after reboot, caused by missing /var/run/sshd*.

Server Fault: *Why am I missing /var/run/sshd after every boot?*.

---

1. The `su` command is disabled in Ubuntu so you must instead run `sudo su`.