Linux SSH Public Key Authentication Lab
By Michael Ambeguia

Purpose: The purpose of this lab is to practice configuring SSH connections to use public key authentication. The default method used for SSH connections is to use passwords, but passwords are inherently weak and can be discovered. Public key authentication is more secure than password based SSH logins since there really is no way to brute force the keys. Public key authentication also requires a pair of keys so even if an attacker is able to brute force a public key it is useless without its corresponding private key. I will also harden asymmetric key authentication by changing some settings and configuring SSH to use multi-factor authentication.

Topics Covered:

1. How SSH Public Key Encryption protects SSH sessions
2. How to configure SSH to use Public Key Encryption
3. Relate SSH to the strengths of Public Key Encryption

Sections:

1. Introduction to SSH Public Key Authentication
2. Generating SSH Key Pairs
3. Send Ubuntu VM public key to RedHat Linux VM
4. Send RedHat Linux VM public key to Ubuntu VM
5. Testing SSH Connection
6. Securing Public Key Authentication Process

Section #1 Introduction to SSH Public Key Authentication:

1.1 How does SSH use Public Key Encryption?

SSH uses public key encryption during the authentication process between the SSH client and SSH server. There are 6 steps during the authentication process.

Step 1. The SSH client sends a request to the SSH server. During the request the client will also tell the server that it wants to use key-based authentication for the connection.

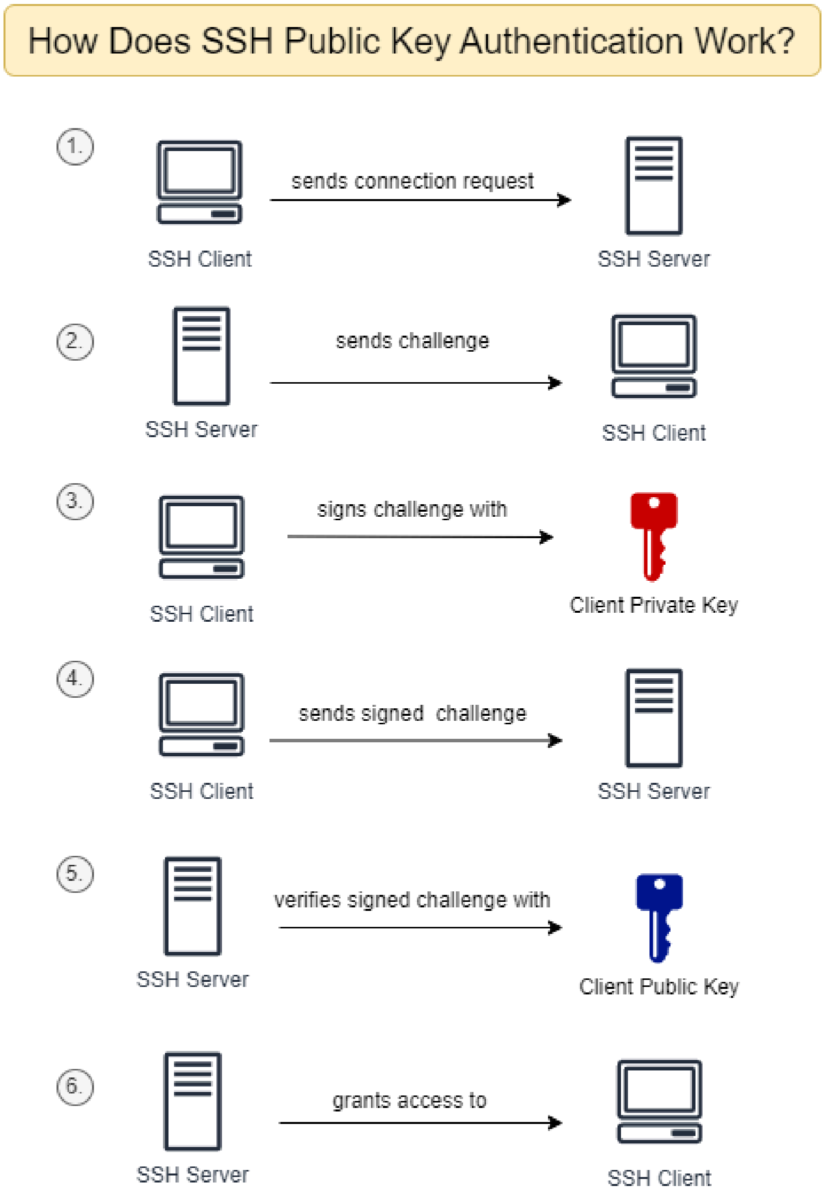
Step2. The SSH server sends the client a challenge. This challenge is a set of random characters that is encrypted using the client public key that is on the server machine.

Step3. The SSH client “solves” the challenge by decrypting it with its private key. It then signs this challenge using the private key as well.

Step 4. The SSH client sends the signed challenge back to the SSH server..

Step 5. The SSH server verifies the signed challenge using the client public key.

Step 6. If the signed challenge is verified, the client will gain access to the server.

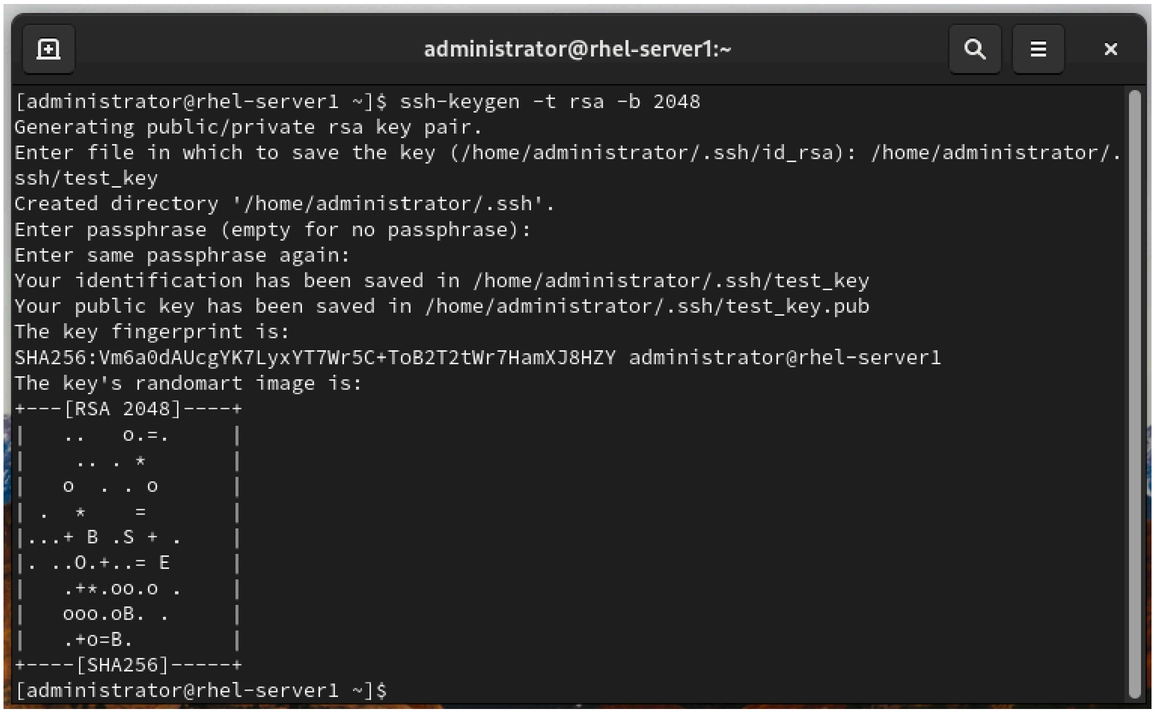


1.2 Why is Public Key authentication better?

Public key authentication is more secure than password based authentication for a variety of reasons. The first major reason is that passwords can be easily cracked or brute-forced. Attackers can simply try thousands of combinations until the password is found, and use it to log into your servers and do damage. The second reason is that password based authentication for SSH does not verify the authenticity of the user. The password might be correct, but the attacker could be logging in from an unknown host to the server. Since SSH public key authentication involves the use of digital signatures, the server can be certain that the login attempt is occurring from a valid host. This is because the client signs the challenge with their private key, and the server uses the public key to verify this. Only the client has the private key, thus proving that the client's identity is valid. The last reason is that there can be an element of multifactor authentication with public key authentication for SSH. The private keys are something you have, and these private keys can be protected by a passphrase, something you know. So even if a private key is compromised, the attacker will still need to know your passphrase. Additionally, the key pairs can be rotated and even revoked if a compromise is suspected.

Section #2 Generating SSH Key Pairs

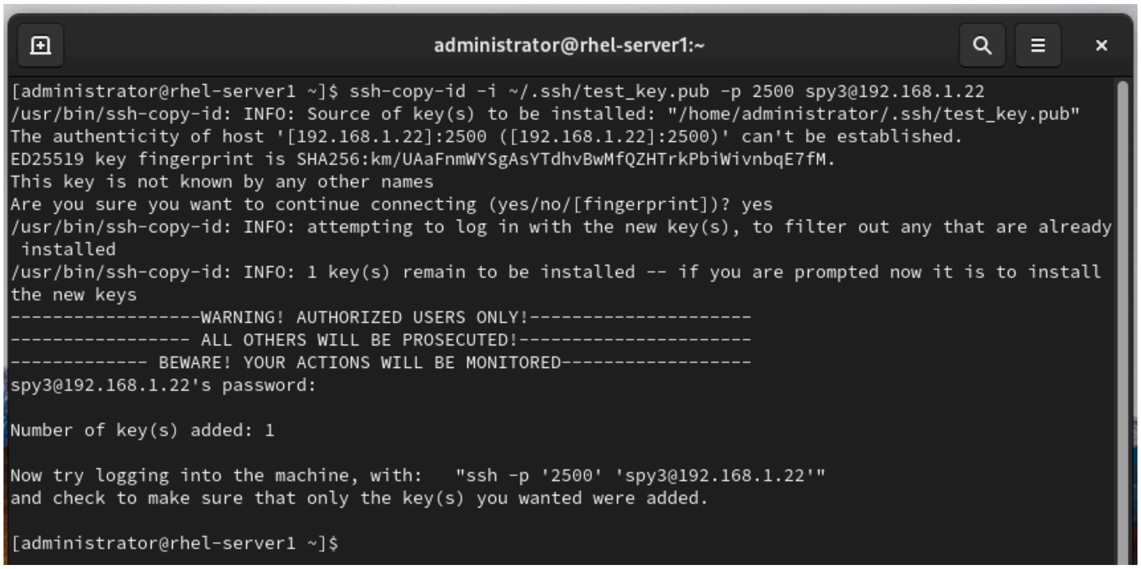
2.1. Generate a public/private key pair for the RHEL . Use the RSA algorithm and be sure to use a sufficiently large key size. Also be sure to set a passphrase for the key as well.



```
[administrator@rhel-server1 ~]$ ssh-keygen -t rsa -b 2048
Generating public/private rsa key pair.
Enter file in which to save the key (/home/administrator/.ssh/id_rsa): /home/administrator/.
ssh/test_key
Created directory '/home/administrator/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/administrator/.ssh/test_key
Your public key has been saved in /home/administrator/.ssh/test_key.pub
The key fingerprint is:
SHA256:Vm6a0dAUcgYK7LyxYT7Wr5C+ToB2T2tWr7HamXJ8HZY administrator@rhel-server1
The key's randomart image is:
+---[RSA 2048]----+
|   ..   o.=.     |
|    .. . *       |
|   o  . . o      |
| .  *    =       |
|...+ B .S + .    |
|. ..O.+..= E     |
|   .+*.oo.o .    |
|   ooo.oB. .     |
|   .+o=B.        |
+----[SHA256]-----+
[administrator@rhel-server1 ~]$
```

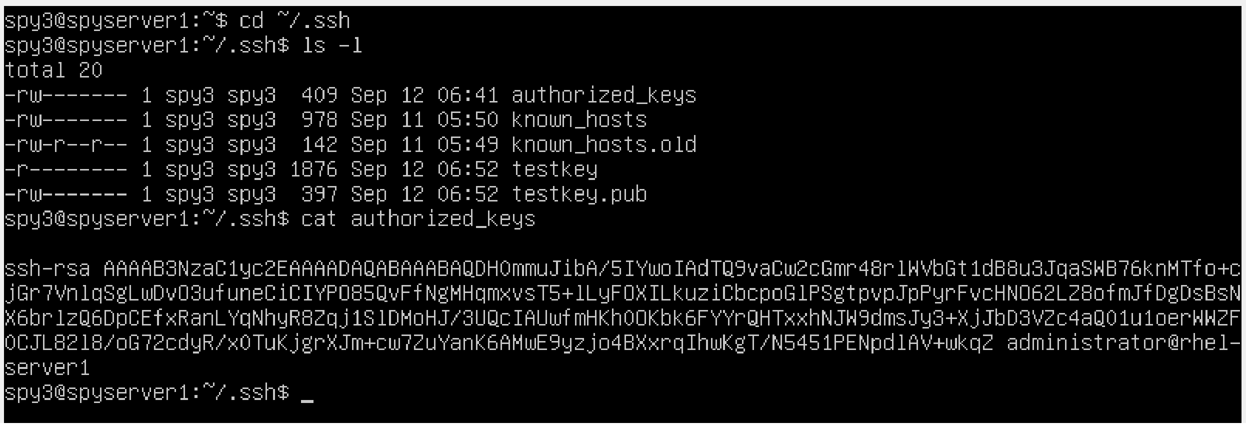
Section #4 Send RedHat Linux VM public key to Ubuntu VM:

4.1 Send the Ubuntu VM public key to the RedHat Linux VM using ssh-copy-id.



```
[administrator@rhel-server1 ~]$ ssh-copy-id -i ~/.ssh/test_key.pub -p 2500 spy3@192.168.1.22
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/administrator/.ssh/test_key.pub"
The authenticity of host '[192.168.1.22]:2500 ([192.168.1.22]:2500)' can't be established.
ED25519 key fingerprint is SHA256:km/UAaFnmWYSgAsYTdhvBwMfQZHTrkPbiWivnbqE7fM.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already
 installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install
the new keys
------------------WARNING! AUTHORIZED USERS ONLY!---------------------
----------------- ALL OTHERS WILL BE PROSECUTED!----------------------
------------- BEWARE! YOUR ACTIONS WILL BE MONITORED------------------
spy3@192.168.1.22's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh -p '2500' 'spy3@192.168.1.22'"
and check to make sure that only the key(s) you wanted were added.

[administrator@rhel-server1 ~]$
```

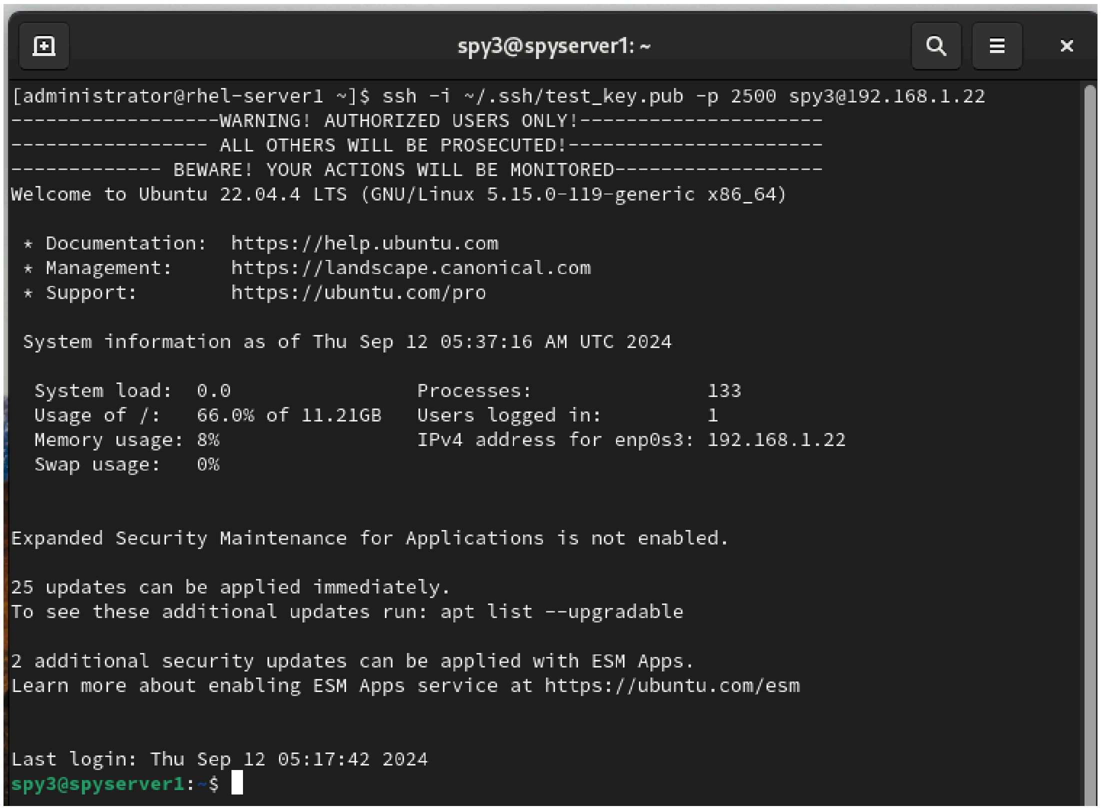
4.2 Verify on the Ubuntu VM that the RHEL VM public key is present in the ~/.ssh/authorized_keys file.



```
spy3@spyserver1:~$ cd ~/.ssh
spy3@spyserver1:~/.ssh$ ls -l
total 20
-rw------- 1 spy3 spy3  409 Sep 12 06:41 authorized_keys
-rw------- 1 spy3 spy3  978 Sep 11 05:50 known_hosts
-rw-r--r-- 1 spy3 spy3  142 Sep 11 05:49 known_hosts.old
-r-------- 1 spy3 spy3 1876 Sep 12 06:52 testkey
-rw------- 1 spy3 spy3  397 Sep 12 06:52 testkey.pub
spy3@spyserver1:~/.ssh$ cat authorized_keys

ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDHOmmuJibA/5IYwoIAdTQ9vaCw2cGmr48rlWVbGt1dB8u3JqaSWB76knMTfo+c
jGr7VnlqSgLwDvO3ufuneCiCIYPO85QvFfNgMHqmxvsT5+lLyFOXILkuziCbcpoGlPSgtpvpJpPyrFvcHNO62LZ8ofmJfDgDsBsN
X6brlzQ6DpCEfxRanLYqNhyR8Zqj1SlDMoHJ/3UQcIAUwfmHKhOOKbk6FYYrQHTxxhNJW9dmsJy3+XjJbD3VZc4aQ01u1oerWWZF
OCJL82l8/oG72cdyR/x0TuKjgrXJm+cw7ZuYanK6AMwE9yzjo4BXxrqIhwKgT/N5451PENpdlAV+wkqZ administrator@rhel-
server1
spy3@spyserver1:~/.ssh$ _
```

Section #5 Testing SSH Connection:

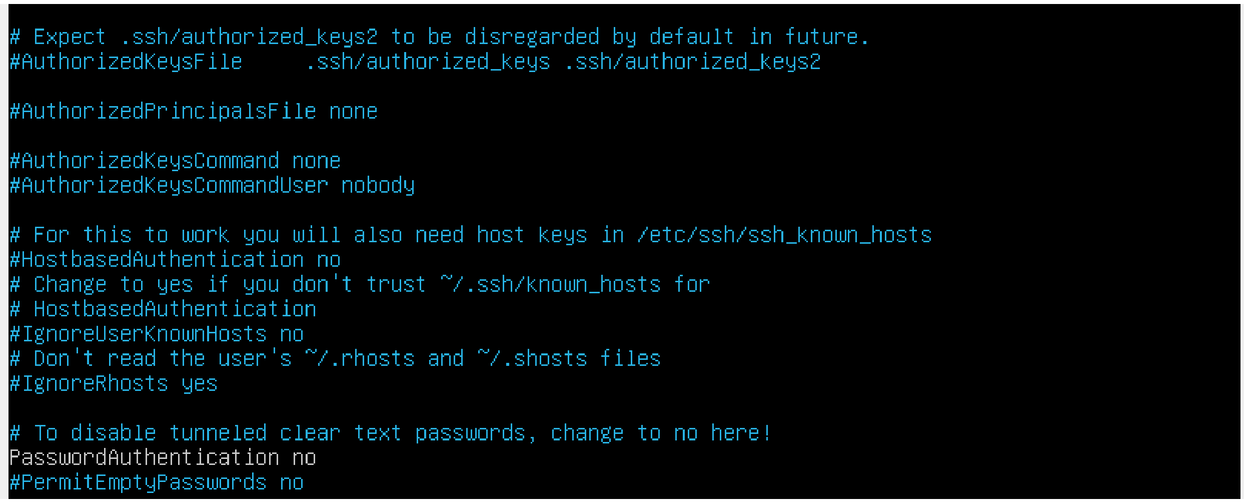
5.2 Test the SSH connection by connecting to the Ubuntu VM from the RedHat VM:



```
[administrator@rhel-server1 ~]$ ssh -i ~/.ssh/test_key.pub -p 2500 spy3@192.168.1.22
------------------WARNING! AUTHORIZED USERS ONLY!---------------------
----------------- ALL OTHERS WILL BE PROSECUTED!----------------------
------------- BEWARE! YOUR ACTIONS WILL BE MONITORED------------------
Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 5.15.0-119-generic x86_64)

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

 System information as of Thu Sep 12 05:37:16 AM UTC 2024

  System load:  0.0                Processes:               133
  Usage of /:   66.0% of 11.21GB   Users logged in:         1
  Memory usage: 8%                 IPv4 address for enp0s3: 192.168.1.22
  Swap usage:   0%


Expanded Security Maintenance for Applications is not enabled.

25 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

2 additional security updates can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm


Last login: Thu Sep 12 05:17:42 2024
spy3@spyserver1:~$
```

Section #6 Securing Public Key Authentication Process:

6.1. Edit the /etc/ssh/sshd_config file and change an additional setting:
I have already done most of the security configurations on the /etc/ssh/sshd_config file such as not permitting root login, changing the default ssh port, and not allowing challenge-based authentication. I just have to do one last change. I have to set the PasswordAuthentication option to no. Doing this would mean that for now the SSH server on my Ubuntu Server VM will only accept key-based authentication.



```
# Expect .ssh/authorized_keys2 to be disregarded by default in future.
#AuthorizedKeysFile     .ssh/authorized_keys .ssh/authorized_keys2

#AuthorizedPrincipalsFile none

#AuthorizedKeysCommand none
#AuthorizedKeysCommandUser nobody

# For this to work you will also need host keys in /etc/ssh/ssh_known_hosts
#HostbasedAuthentication no
# Change to yes if you don't trust ~/.ssh/known_hosts for
# HostbasedAuthentication
#IgnoreUserKnownHosts no
# Don't read the user's ~/.rhosts and ~/.shosts files
#IgnoreRhosts yes

# To disable tunneled clear text passwords, change to no here!
PasswordAuthentication no
#PermitEmptyPasswords no
```

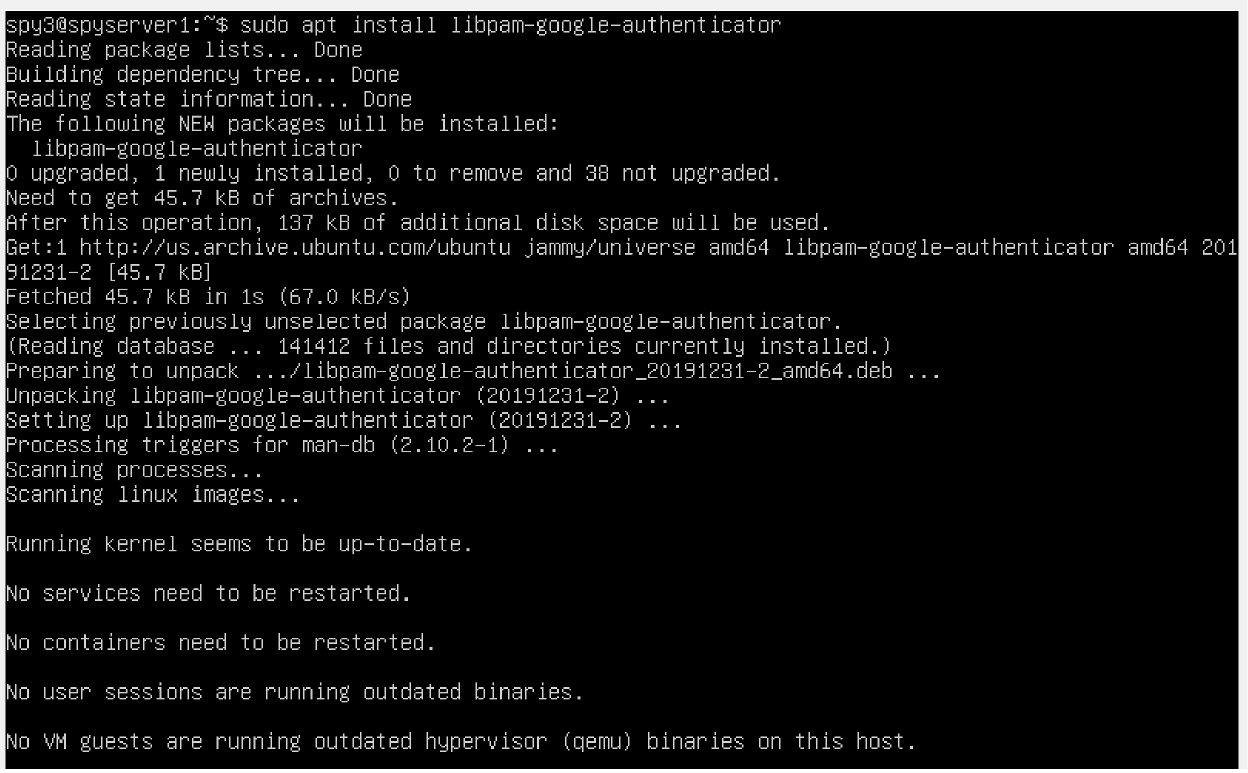
6.2. Have an ssh key rotation policy:

Having a key rotation policy is important since it reduces the risk of having a private key being compromised. During key rotation a new public/private key pair is created on the client device and is shared with the server. An efficient way to perform this task is to use a bash script that can automate the process of removing old ssh keys, creating the new key pairs, and establishing a new connection between the client and server through ssh.

6.3 Add MFA to the ssh process for extra security.

Steps:

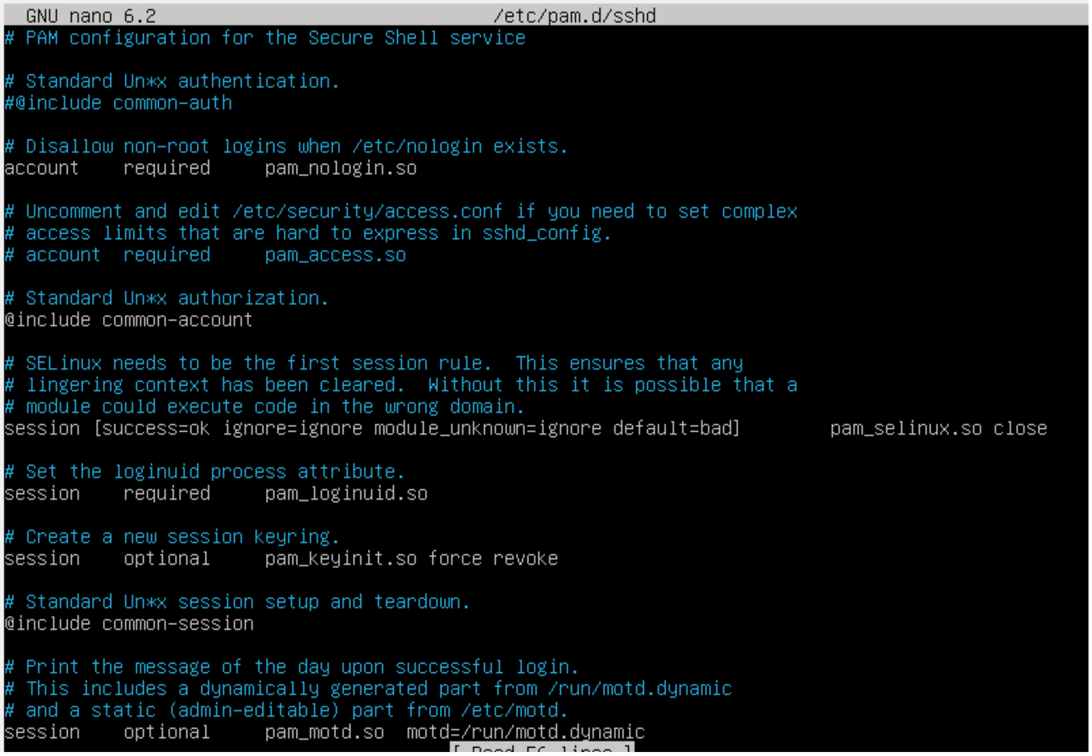
1. Install the libpam-google-authenticator package:



```
spy3@spyserver1:~$ sudo apt install libpam-google-authenticator
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  libpam-google-authenticator
0 upgraded, 1 newly installed, 0 to remove and 38 not upgraded.
Need to get 45.7 kB of archives.
After this operation, 137 kB of additional disk space will be used.
Get:1 http://us.archive.ubuntu.com/ubuntu jammy/universe amd64 libpam-google-authenticator amd64 201
91231-2 [45.7 kB]
Fetched 45.7 kB in 1s (67.0 kB/s)
Selecting previously unselected package libpam-google-authenticator.
(Reading database ... 141412 files and directories currently installed.)
Preparing to unpack .../libpam-google-authenticator_20191231-2_amd64.deb ...
Unpacking libpam-google-authenticator (20191231-2) ...
Setting up libpam-google-authenticator (20191231-2) ...
Processing triggers for man-db (2.10.2-1) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
```

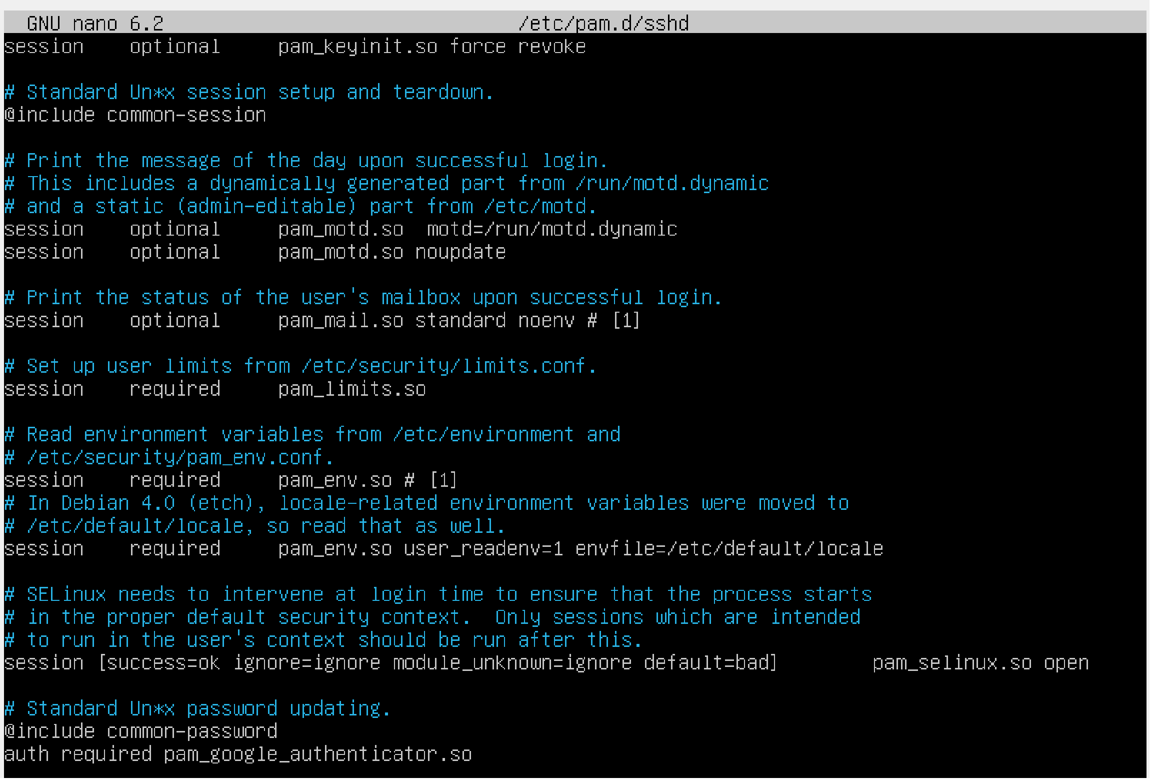
2. Configure PAM to allow SSH to use Google Authenticator

Comment out the include common-auth line under standard Unix authentication on the sshd PAM file. The reason you need to do this is you need to ensure that PAM will not prompt a user for their regular password.



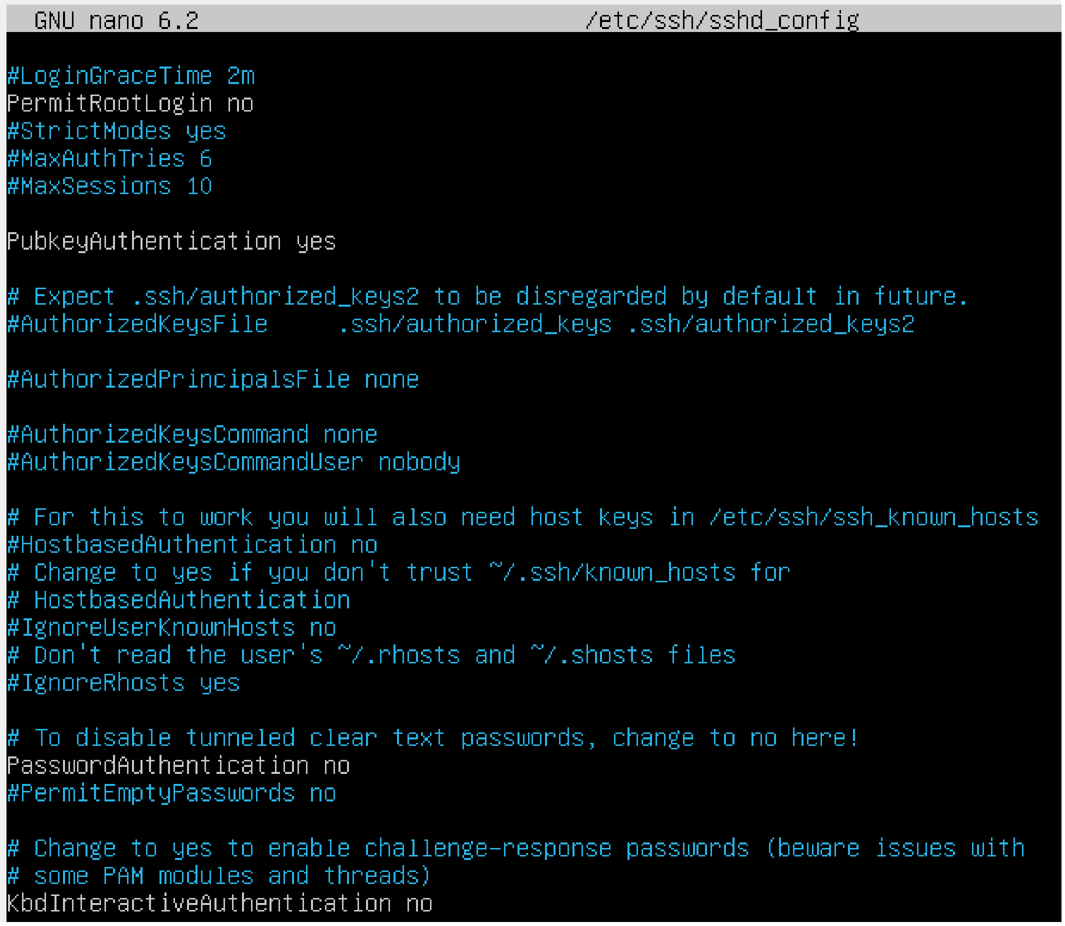
```
  GNU nano 6.2                                /etc/pam.d/sshd
# PAM configuration for the Secure Shell service

# Standard Un*x authentication.
#@include common-auth

# Disallow non-root logins when /etc/nologin exists.
account    required     pam_nologin.so

# Uncomment and edit /etc/security/access.conf if you need to set complex
# access limits that are hard to express in sshd_config.
# account  required     pam_access.so

# Standard Un*x authorization.
@include common-account

# SELinux needs to be the first session rule.  This ensures that any
# lingering context has been cleared.  Without this it is possible that a
# module could execute code in the wrong domain.
session [success=ok ignore=ignore module_unknown=ignore default=bad]        pam_selinux.so close

# Set the loginuid process attribute.
session    required     pam_loginuid.so

# Create a new session keyring.
session    optional     pam_keyinit.so force revoke

# Standard Un*x session setup and teardown.
@include common-session

# Print the message of the day upon successful login.
# This includes a dynamically generated part from /run/motd.dynamic
# and a static (admin-editable) part from /etc/motd.
session    optional     pam_motd.so  motd=/run/motd.dynamic
                                          [ Read 56 lines ]
```

Under the standard Unix password updating add auth required pam_google_authenticator.sos o that PAM knows that the one time passcode from Google Authenticator is needed.



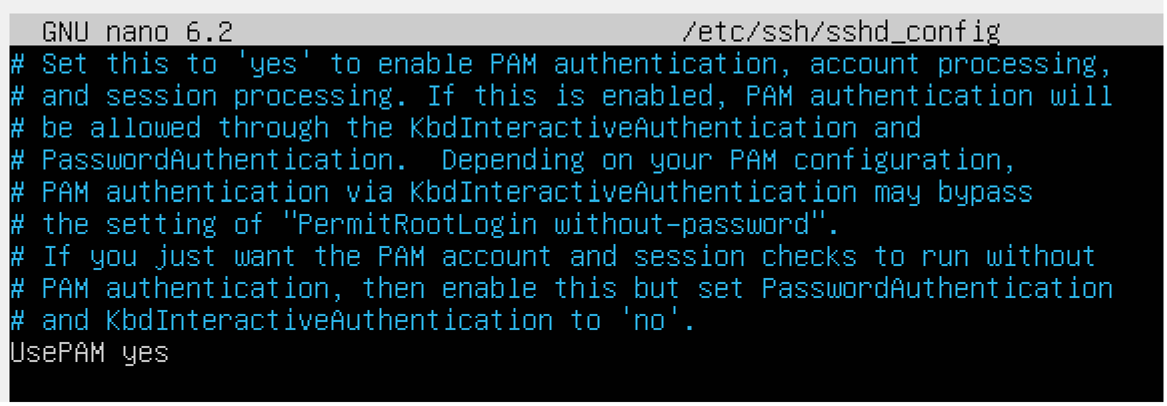
```
  GNU nano 6.2                                    /etc/pam.d/sshd
session    optional     pam_keyinit.so force revoke

# Standard Un*x session setup and teardown.
@include common-session

# Print the message of the day upon successful login.
# This includes a dynamically generated part from /run/motd.dynamic
# and a static (admin-editable) part from /etc/motd.
session    optional     pam_motd.so  motd=/run/motd.dynamic
session    optional     pam_motd.so noupdate

# Print the status of the user's mailbox upon successful login.
session    optional     pam_mail.so standard noenv # [1]

# Set up user limits from /etc/security/limits.conf.
session    required     pam_limits.so

# Read environment variables from /etc/environment and
# /etc/security/pam_env.conf.
session    required     pam_env.so # [1]
# In Debian 4.0 (etch), locale-related environment variables were moved to
# /etc/default/locale, so read that as well.
session    required     pam_env.so user_readenv=1 envfile=/etc/default/locale

# SELinux needs to intervene at login time to ensure that the process starts
# in the proper default security context.  Only sessions which are intended
# to run in the user's context should be run after this.
session [success=ok ignore=ignore module_unknown=ignore default=bad]        pam_selinux.so open

# Standard Un*x password updating.
@include common-password
auth required pam_google_authenticator.so
```

3. Modify the sshd_config file

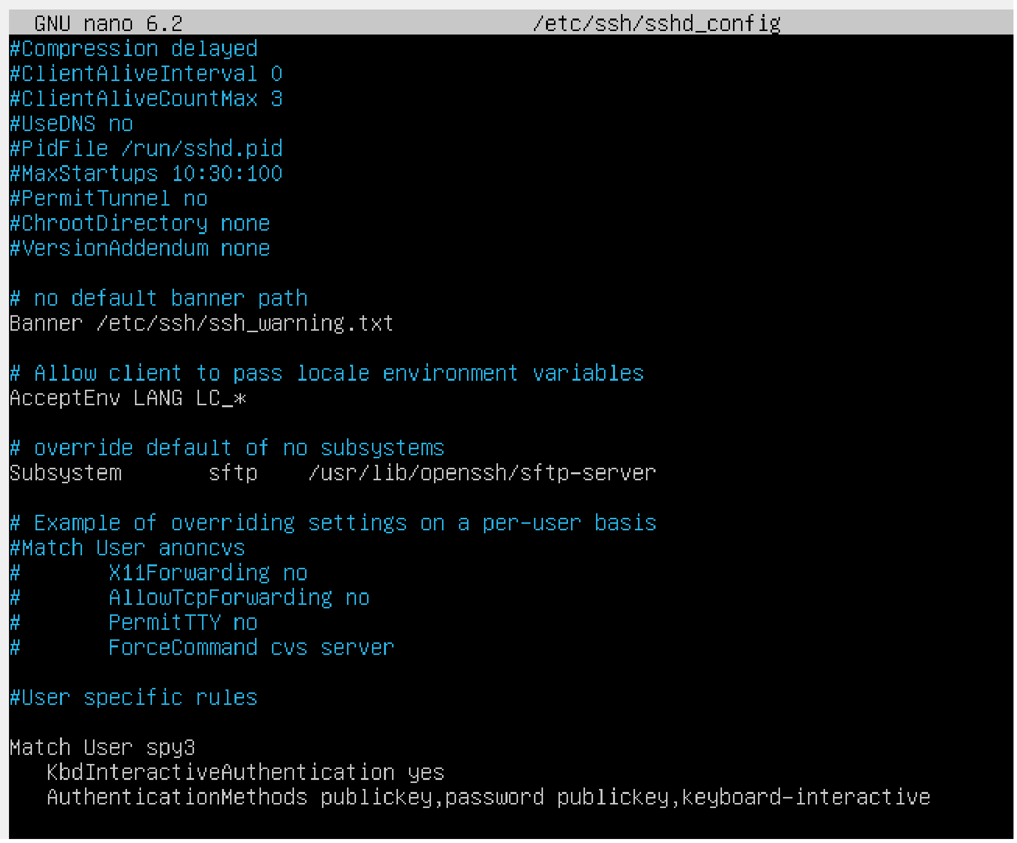


```
  GNU nano 6.2                        /etc/ssh/sshd_config

#LoginGraceTime 2m
PermitRootLogin no
#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10

PubkeyAuthentication yes

# Expect .ssh/authorized_keys2 to be disregarded by default in future.
#AuthorizedKeysFile     .ssh/authorized_keys .ssh/authorized_keys2

#AuthorizedPrincipalsFile none

#AuthorizedKeysCommand none
#AuthorizedKeysCommandUser nobody

# For this to work you will also need host keys in /etc/ssh/ssh_known_hosts
#HostbasedAuthentication no
# Change to yes if you don't trust ~/.ssh/known_hosts for
# HostbasedAuthentication
#IgnoreUserKnownHosts no
# Don't read the user's ~/.rhosts and ~/.shosts files
#IgnoreRhosts yes

# To disable tunneled clear text passwords, change to no here!
PasswordAuthentication no
#PermitEmptyPasswords no

# Change to yes to enable challenge-response passwords (beware issues with
# some PAM modules and threads)
KbdInteractiveAuthentication no
```

Since you need the /etc/pam/sshd file to work with SSH you would set UsePAM to yes.



```
  GNU nano 6.2                        /etc/ssh/sshd_config
# Set this to 'yes' to enable PAM authentication, account processing,
# and session processing. If this is enabled, PAM authentication will
# be allowed through the KbdInteractiveAuthentication and
# PasswordAuthentication.  Depending on your PAM configuration,
# PAM authentication via KbdInteractiveAuthentication may bypass
# the setting of "PermitRootLogin without-password".
# If you just want the PAM account and session checks to run without
# PAM authentication, then enable this but set PasswordAuthentication
# and KbdInteractiveAuthentication to 'no'.
UsePAM yes
```

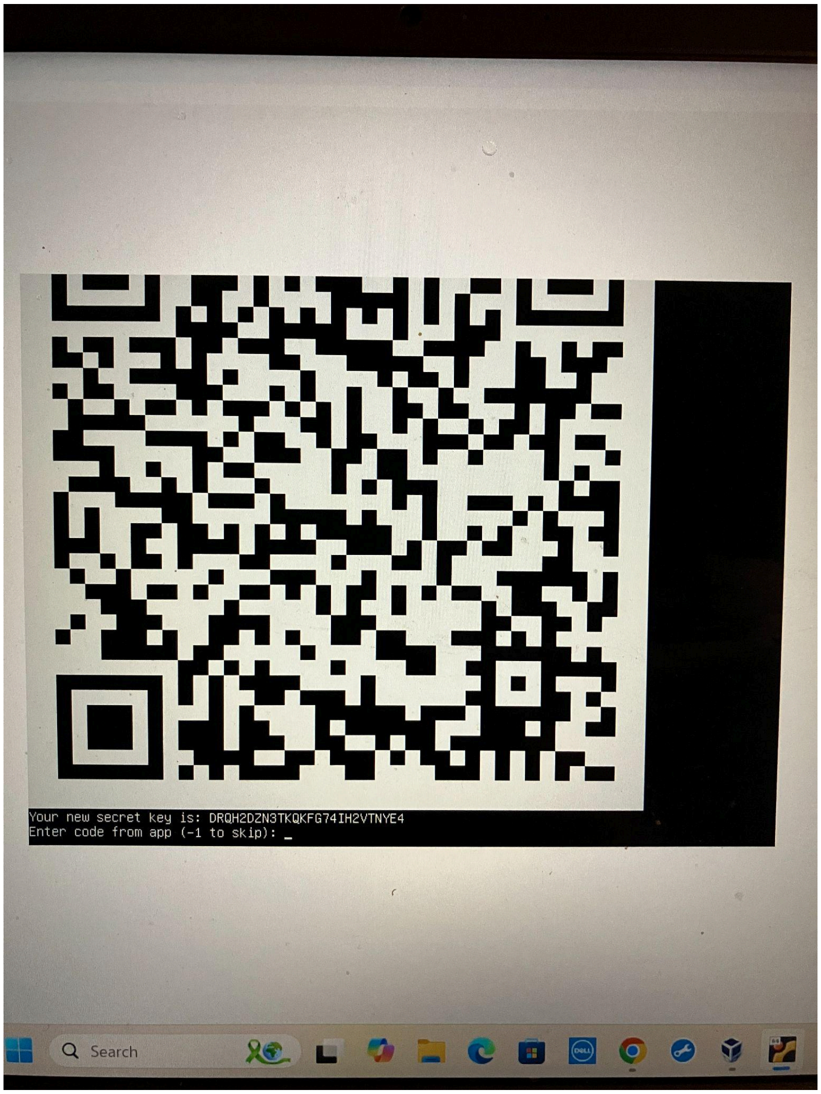


```
  GNU nano 6.2                            /etc/ssh/sshd_config
#Compression delayed
#ClientAliveInterval 0
#ClientAliveCountMax 3
#UseDNS no
#PidFile /run/sshd.pid
#MaxStartups 10:30:100
#PermitTunnel no
#ChrootDirectory none
#VersionAddendum none

# no default banner path
Banner /etc/ssh/ssh_warning.txt

# Allow client to pass locale environment variables
AcceptEnv LANG LC_*

# override default of no subsystems
Subsystem       sftp    /usr/lib/openssh/sftp-server

# Example of overriding settings on a per-user basis
#Match User anoncvs
#       X11Forwarding no
#       AllowTcpForwarding no
#       PermitTTY no
#       ForceCommand cvs server

#User specific rules

Match User spy3
   KbdInteractiveAuthentication yes
   AuthenticationMethods publickey,password publickey,keyboard-interactive
```

Under the user specific rules section I had to add the AuthenticationMethods. I want a private key to be used along with 2FA. 2FA requires you to enter the one time code so keyboard-interactive authentication is needed.

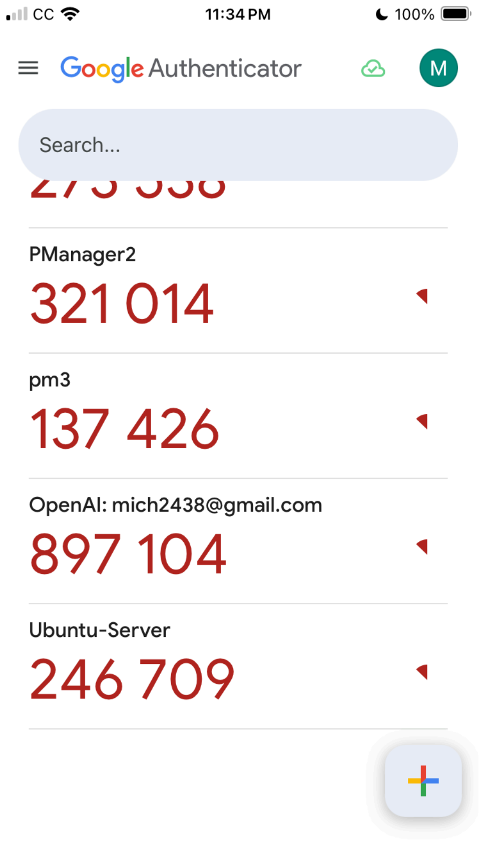
4. Configure Google Authenticator

4.1. Start the Google Authenticator app. You will see a QR code to add this device to your Google Authenticator app. If the QR code does not work the secret key can be used instead.



```
Your new secret key is: DRQH2DZN3TKQKFG74IH2VTNYE4
Enter code from app (-1 to skip): _
```

Now the Ubuntu-Server is linked to my Google Authenticator app.



PManager2
321 014

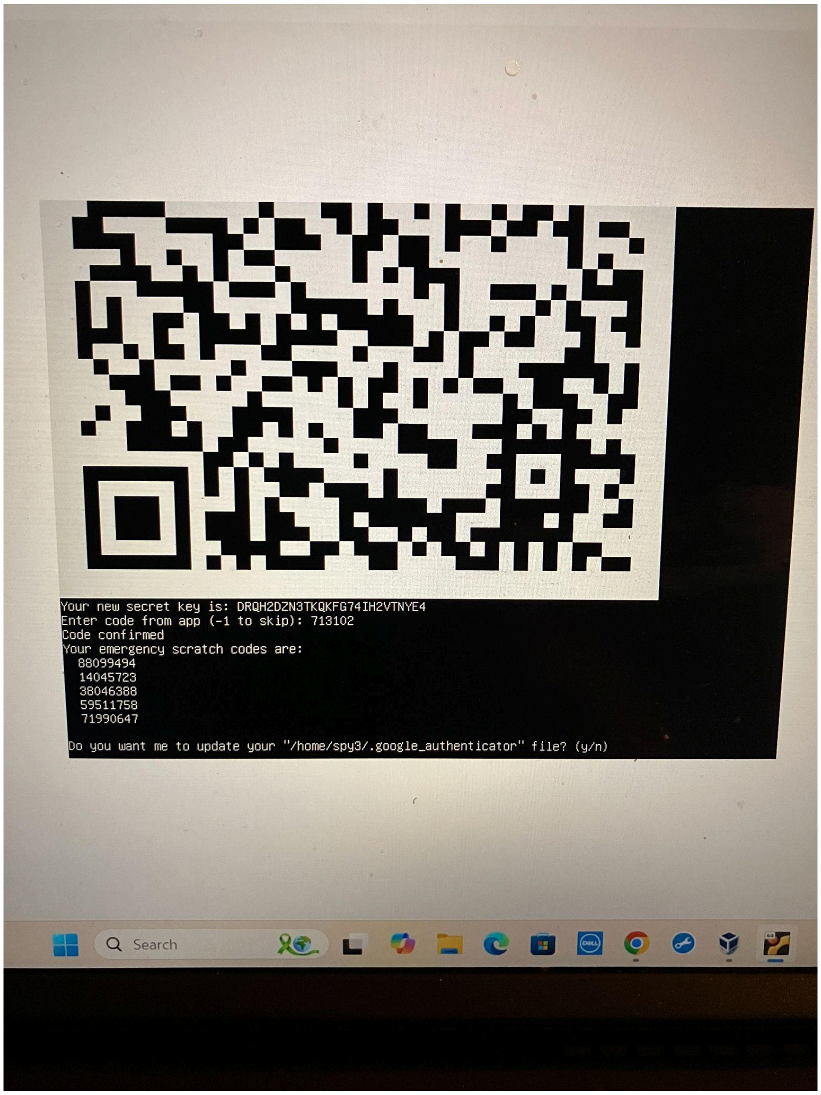
pm3
137 426

OpenAI: mich2438@gmail.com
897 104

Ubuntu-Server
246 709

4.2

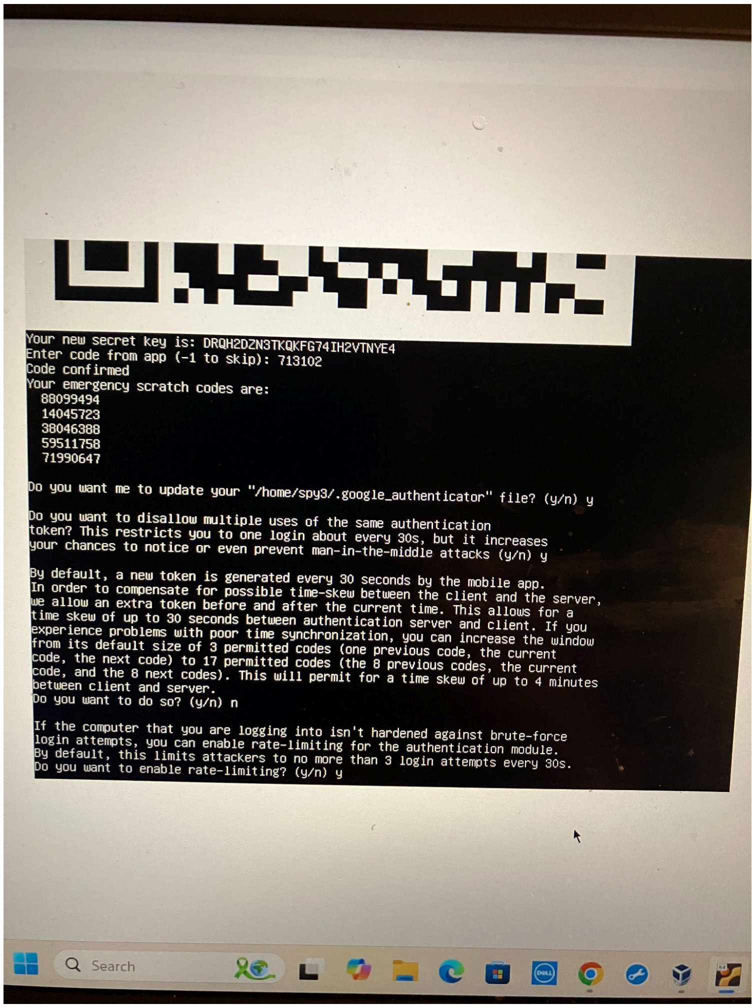
After the device is added you are prompted to enter a code from the Authenticator App.



```
Your new secret key is: DRQH2DZN3TKQKFG74IH2VTNYE4
Enter code from app (-1 to skip): 713102
Code confirmed
Your emergency scratch codes are:
  88099494
  14045723
  38046388
  59511758
  71990647

Do you want me to update your "/home/spy3/.google_authenticator" file? (y/n)
```

4.3.

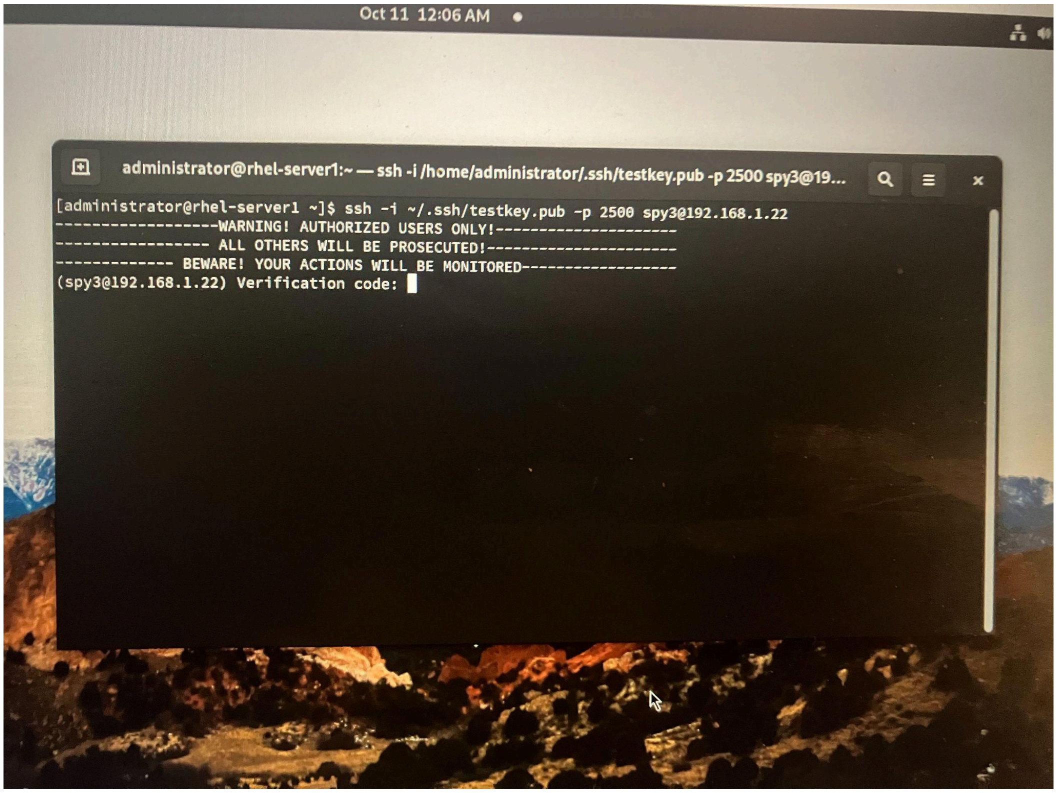
Set some settings for Google Authenticator.



```
Your new secret key is: DRQH2DZN3TKQKFG74IH2VTNYE4
Enter code from app (-1 to skip): 713102
Code confirmed
Your emergency scratch codes are:
  88099494
  14045723
  38046388
  59511758
  71990647

Do you want me to update your "/home/spy3/.google_authenticator" file? (y/n) y

Do you want to disallow multiple uses of the same authentication
token? This restricts you to one login about every 30s, but it increases
your chances to notice or even prevent man-in-the-middle attacks (y/n) y

By default, a new token is generated every 30 seconds by the mobile app.
In order to compensate for possible time-skew between the client and the server,
we allow an extra token before and after the current time. This allows for a
time skew of up to 30 seconds between authentication server and client. If you
experience problems with poor time synchronization, you can increase the window
from its default size of 3 permitted codes (one previous code, the current
code, the next code) to 17 permitted codes (the 8 previous codes, the current
code, and the 8 next codes). This will permit for a time skew of up to 4 minutes
between client and server.
Do you want to do so? (y/n) n

If the computer that you are logging into isn't hardened against brute-force
login attempts, you can enable rate-limiting for the authentication module.
By default, this limits attackers to no more than 3 login attempts every 30s.
Do you want to enable rate-limiting? (y/n) y
```

4.4. Test the connection.

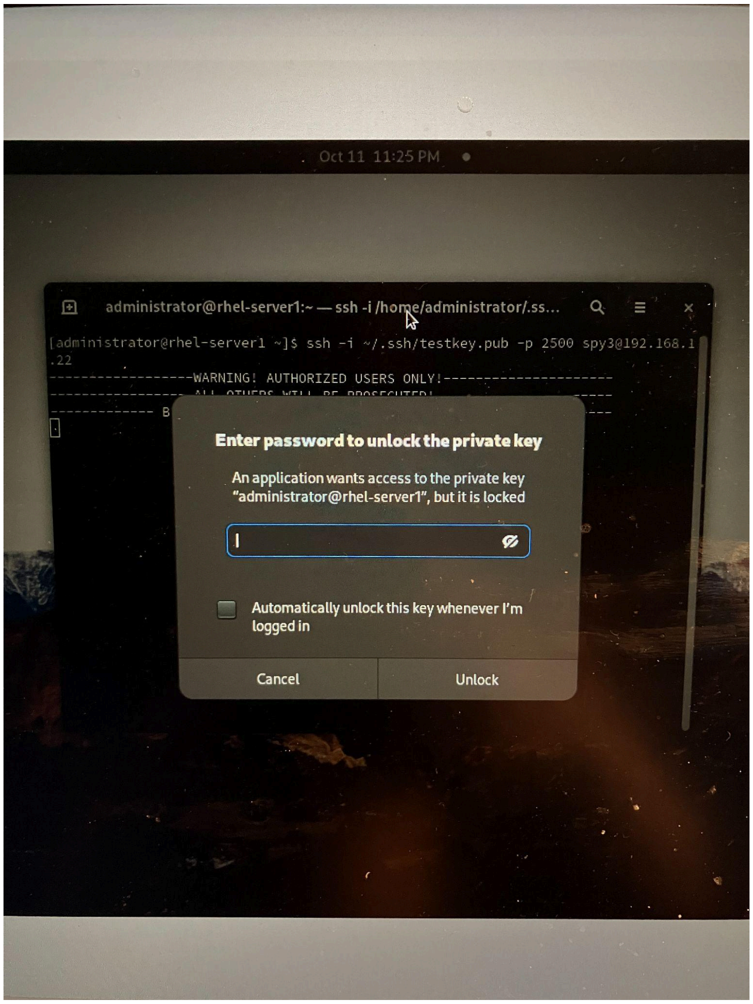
I am asked for the authenticator code first.



Oct 11 12:06 AM
administrator@rhel-server1:~ — ssh -i /home/administrator/.ssh/testkey.pub -p 2500 spy3@19...

```
[administrator@rhel-server1 ~]$ ssh -i ~/.ssh/testkey.pub -p 2500 spy3@192.168.1.22
-----------------WARNING! AUTHORIZED USERS ONLY!--------------------
----------------- ALL OTHERS WILL BE PROSECUTED!---------------------
------------- BEWARE! YOUR ACTIONS WILL BE MONITORED-----------------
(spy3@192.168.1.22) Verification code: 
```

After I enter the code I am prompted to enter the password to unlock my private key. I am using SSH to log into my Ubuntu server vm. Remember that I have my RHEL public key on the Ubuntu Server. In order to have a successful authentication I need to sign the ssh challenge using the RHEL private key.


Oct 11 11:25 PM
administrator@rhel-server1:~ — ssh -i /home/administrator/.ss...
Cancel
Unlock

```
[administrator@rhel-server1 ~]$ ssh -i ~/.ssh/testkey.pub -p 2500 spy3@192.168.1
.22
------------------WARNING! AUTHORIZED USERS ONLY!---------------------
-----------------ALL OTHERS WILL BE PROSECUTED!-----------------------
------------- B                                                 -----
```

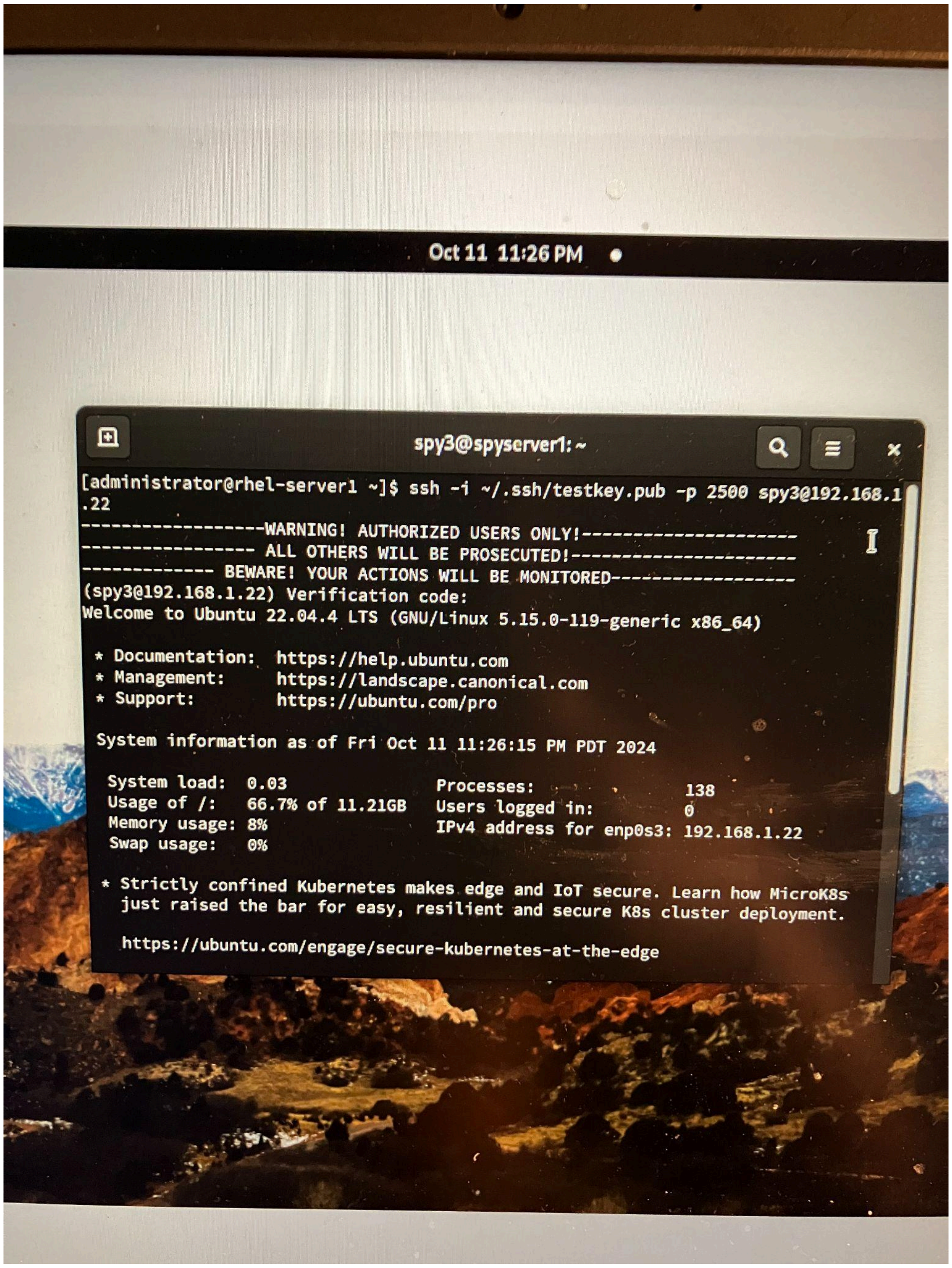
Enter password to unlock the private key

An application wants access to the private key "administrator@rhel-server1", but it is locked

□ Automatically unlock this key whenever I'm logged in

4.6

After the RHEL private key is unlocked ssh authentication worked and I am in!



```
[administrator@rhel-server1 ~]$ ssh -i ~/.ssh/testkey.pub -p 2500 spy3@192.168.1
.22
-----------------WARNING! AUTHORIZED USERS ONLY!---------------------
----------------- ALL OTHERS WILL BE PROSECUTED!---------------------
------------- BEWARE! YOUR ACTIONS WILL BE MONITORED-----------------
(spy3@192.168.1.22) Verification code:
Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 5.15.0-119-generic x86_64)

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

 System information as of Fri Oct 11 11:26:15 PM PDT 2024

  System load:  0.03               Processes:             138
  Usage of /:   66.7% of 11.21GB   Users logged in:       0
  Memory usage: 8%                 IPv4 address for enp0s3: 192.168.1.22
  Swap usage:   0%

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
   just raised the bar for easy, resilient and secure K8s cluster deployment.

   https://ubuntu.com/engage/secure-kubernetes-at-the-edge
```