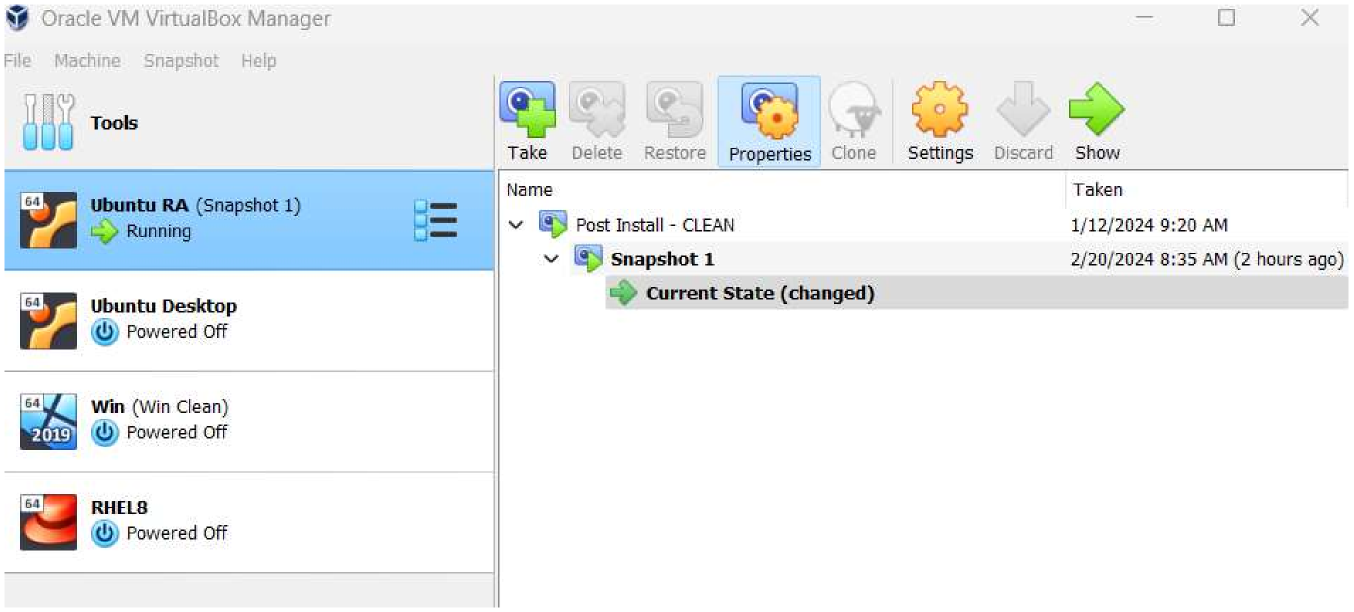
Install 2FA on Ubuntu Server

By: Chris Gaynor

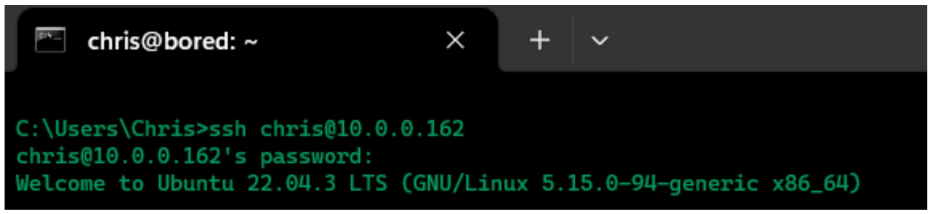
Installing 2FA on Linux VM (Ubuntu 22.04.3)

Recommend starting with snapshot and I then remote in from a Windows Host


Oracle VM VirtualBox Manager
File Machine Snapshot Help
Tools
Take Delete Restore Properties Clone Settings Discard Show

Ubuntu RA (Snapshot 1) Running
Ubuntu Desktop Powered Off
Win (Win Clean) Powered Off
RHEL8 Powered Off

Name	Taken
Post Install - CLEAN	1/12/2024 9:20 AM
Snapshot 1	2/20/2024 8:35 AM (2 hours ago)
Current State (changed)	

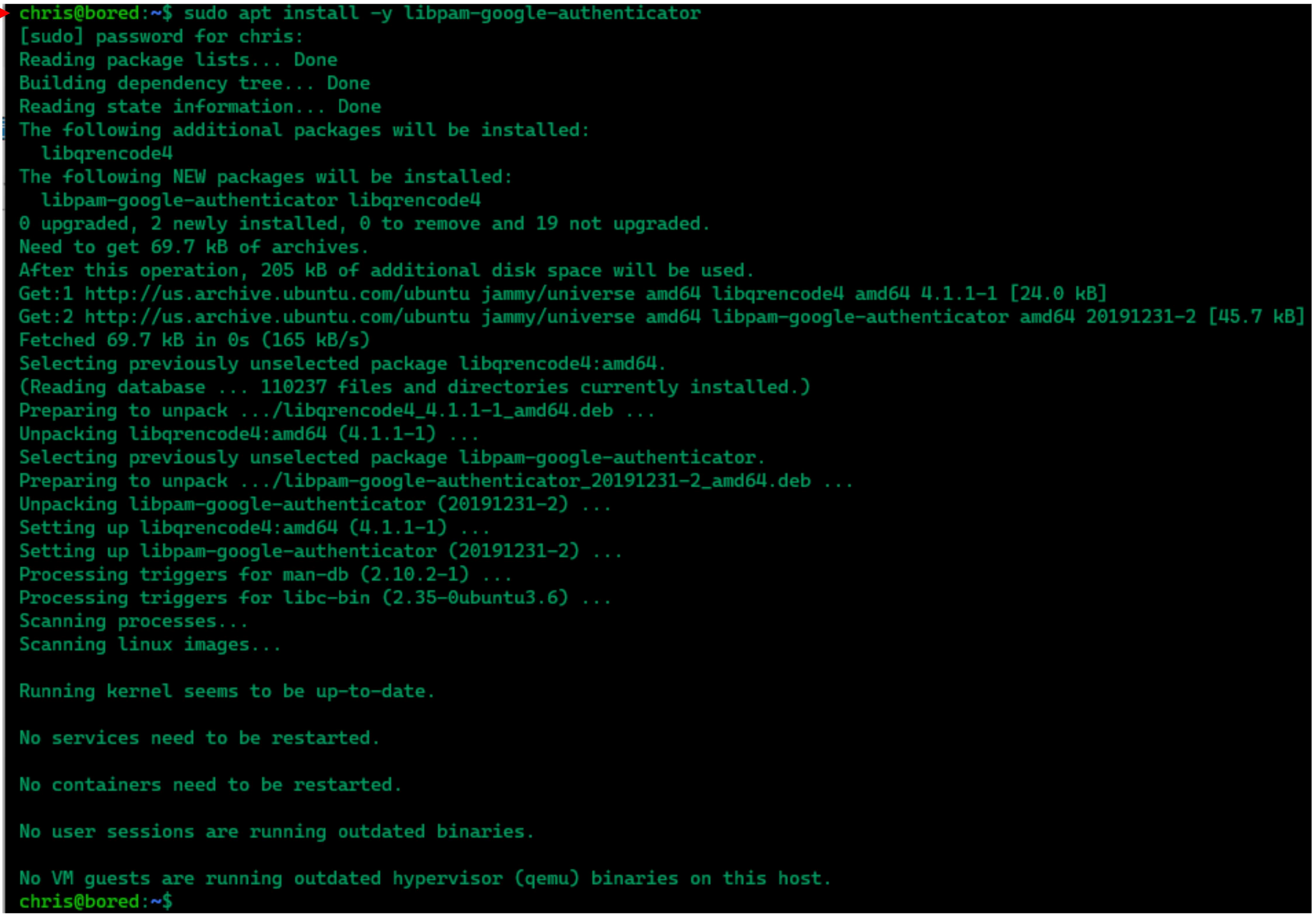


```
C:\Users\Chris>ssh chris@10.0.0.162
chris@10.0.0.162's password:
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.0-94-generic x86_64)
```

*sudo apt install -y libpam-google-authenticator

The first thing we will do is to install the Google Authenticator application

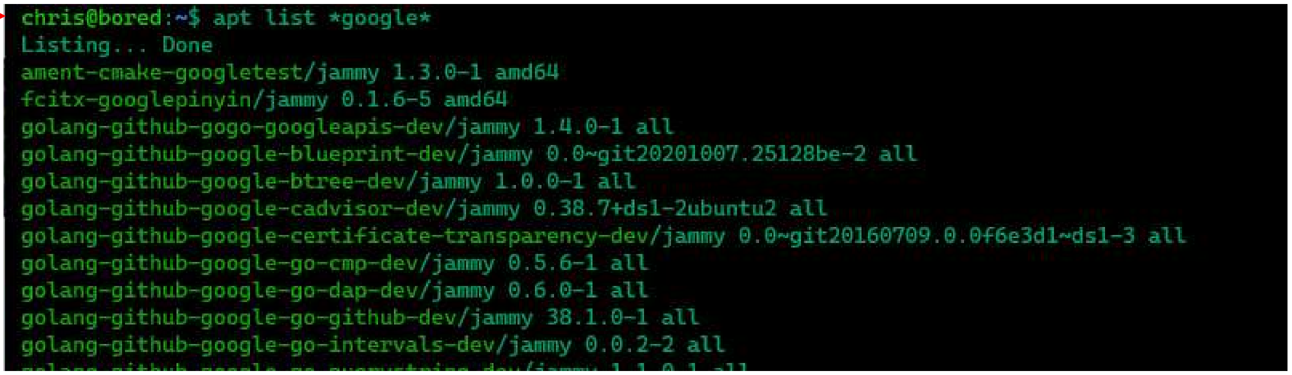
We will download from the default Ubuntu package repository



```
chris@bored:~$ sudo apt install -y libpam-google-authenticator
[sudo] password for chris:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libqrencode4
The following NEW packages will be installed:
  libpam-google-authenticator libqrencode4
0 upgraded, 2 newly installed, 0 to remove and 19 not upgraded.
Need to get 69.7 kB of archives.
After this operation, 205 kB of additional disk space will be used.
Get:1 http://us.archive.ubuntu.com/ubuntu jammy/universe amd64 libqrencode4 amd64 4.1.1-1 [24.0 kB]
Get:2 http://us.archive.ubuntu.com/ubuntu jammy/universe amd64 libpam-google-authenticator amd64 20191231-2 [45.7 kB]
Fetched 69.7 kB in 0s (165 kB/s)
Selecting previously unselected package libqrencode4:amd64.
(Reading database ... 110237 files and directories currently installed.)
Preparing to unpack .../libqrencode4_4.1.1-1_amd64.deb ...
Unpacking libqrencode4:amd64 (4.1.1-1) ...
Selecting previously unselected package libpam-google-authenticator.
Preparing to unpack .../libpam-google-authenticator_20191231-2_amd64.deb ...
Unpacking libpam-google-authenticator (20191231-2) ...
Setting up libqrencode4:amd64 (4.1.1-1) ...
Setting up libpam-google-authenticator (20191231-2) ...
Processing triggers for man-db (2.10.2-1) ...
Processing triggers for libc-bin (2.35-0ubuntu3.6) ...
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.
chris@bored:~$
```

Curious about what other packages are out there? The command apt list will show them all, but a bit much to view this way.

*However, I was curious so I checked into what other google related packages were available. I used the cmd: apt list *google**

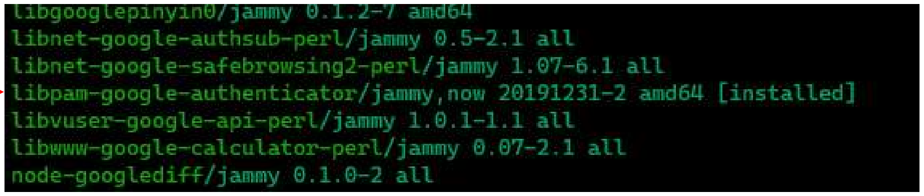


```
chris@bored:~$ apt list *google*
Listing... Done
ament-cmake-googletest/jammy 1.3.0-1 amd64
fcitx-googlepinyin/jammy 0.1.6-5 amd64
golang-github-gogo-googleapis-dev/jammy 1.4.0-1 all
golang-github-google-blueprint-dev/jammy 0.0~git20201007.25128be-2 all
golang-github-google-btree-dev/jammy 1.0.0-1 all
golang-github-google-cadvisor-dev/jammy 0.38.7+ds1-2ubuntu2 all
golang-github-google-certificate-transparency-dev/jammy 0.0~git20160709.0.0f6e3d1~ds1-3 all
golang-github-google-go-cmp-dev/jammy 0.5.6-1 all
golang-github-google-go-dap-dev/jammy 0.6.0-1 all
golang-github-google-go-github-dev/jammy 38.1.0-1 all
golang-github-google-go-intervals-dev/jammy 0.0.2-2 all
golang-github-google-go-querystring-dev/jammy 1.1.0-1 all
```

.

.

.

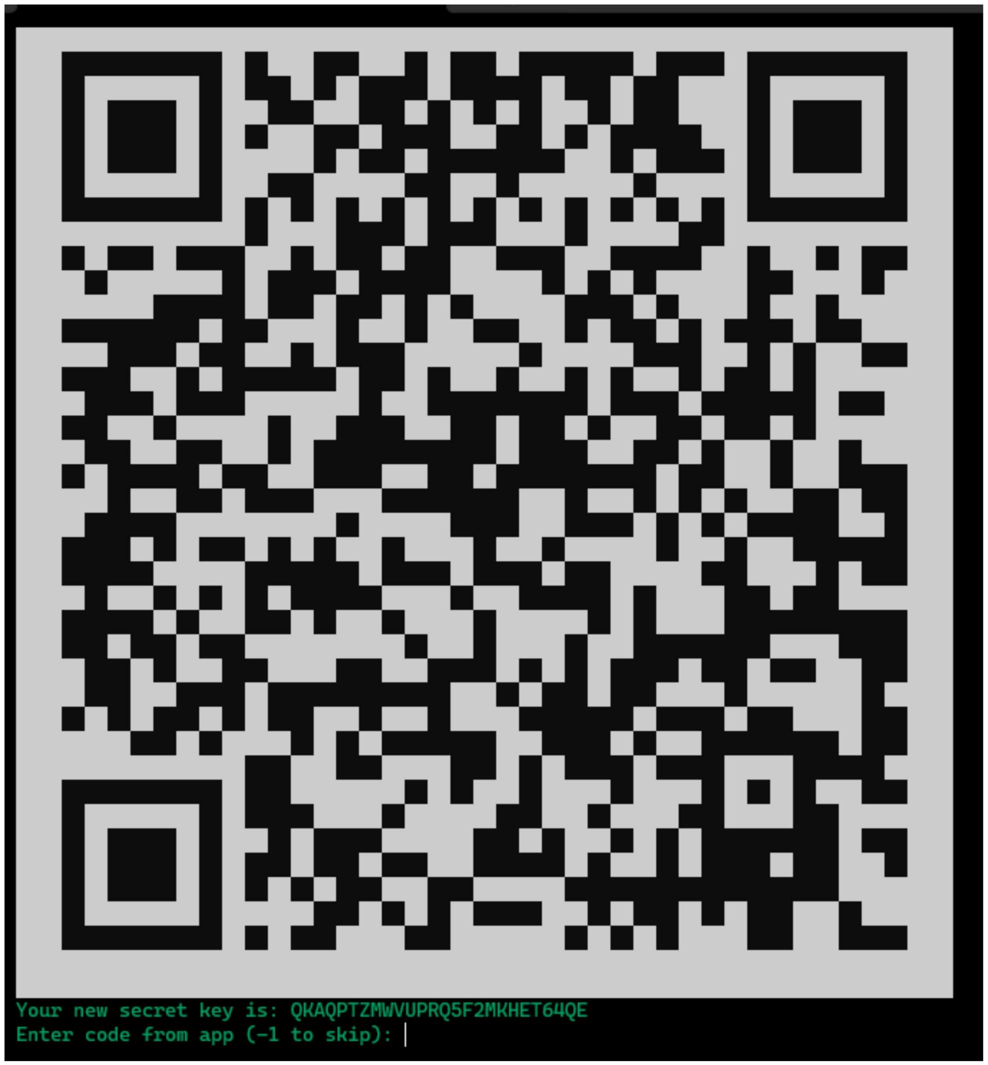


```
libgooglepinyin0/jammy 0.1.2-7 amd64
libnet-google-authsub-perl/jammy 0.5-2.1 all
libnet-google-safebrowsing2-perl/jammy 1.07-6.1 all
libpam-google-authenticator/jammy,now 20191231-2 amd64 [installed]
libvuser-google-api-perl/jammy 1.0.1-1.1 all
libwww-google-calculator-perl/jammy 0.07-2.1 all
node-googlediff/jammy 0.1.0-2 all
```

Still a fairly long list but I saw what was offered and the package we just installed.

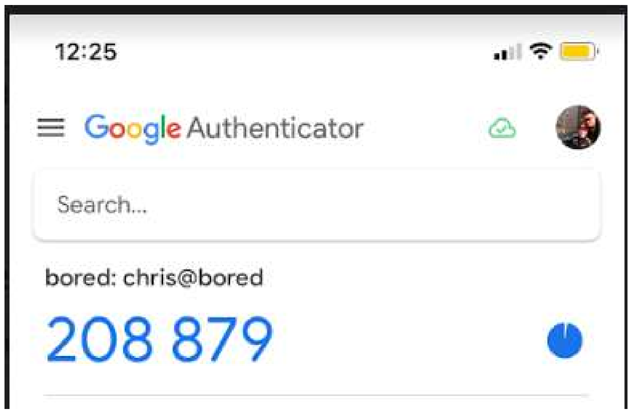
Next, lets run the package by inputing Google authenticator into the command line. You should next see a large QR code.

Scan using the google authenticator app on your phone/device



```
Your new secret key is: QKAQPTZMWVUPRQ5F2MKHET64QE
Enter code from app (-1 to skip):
```

It will then provide you with a code. You will use this code next to confirm you associated properly

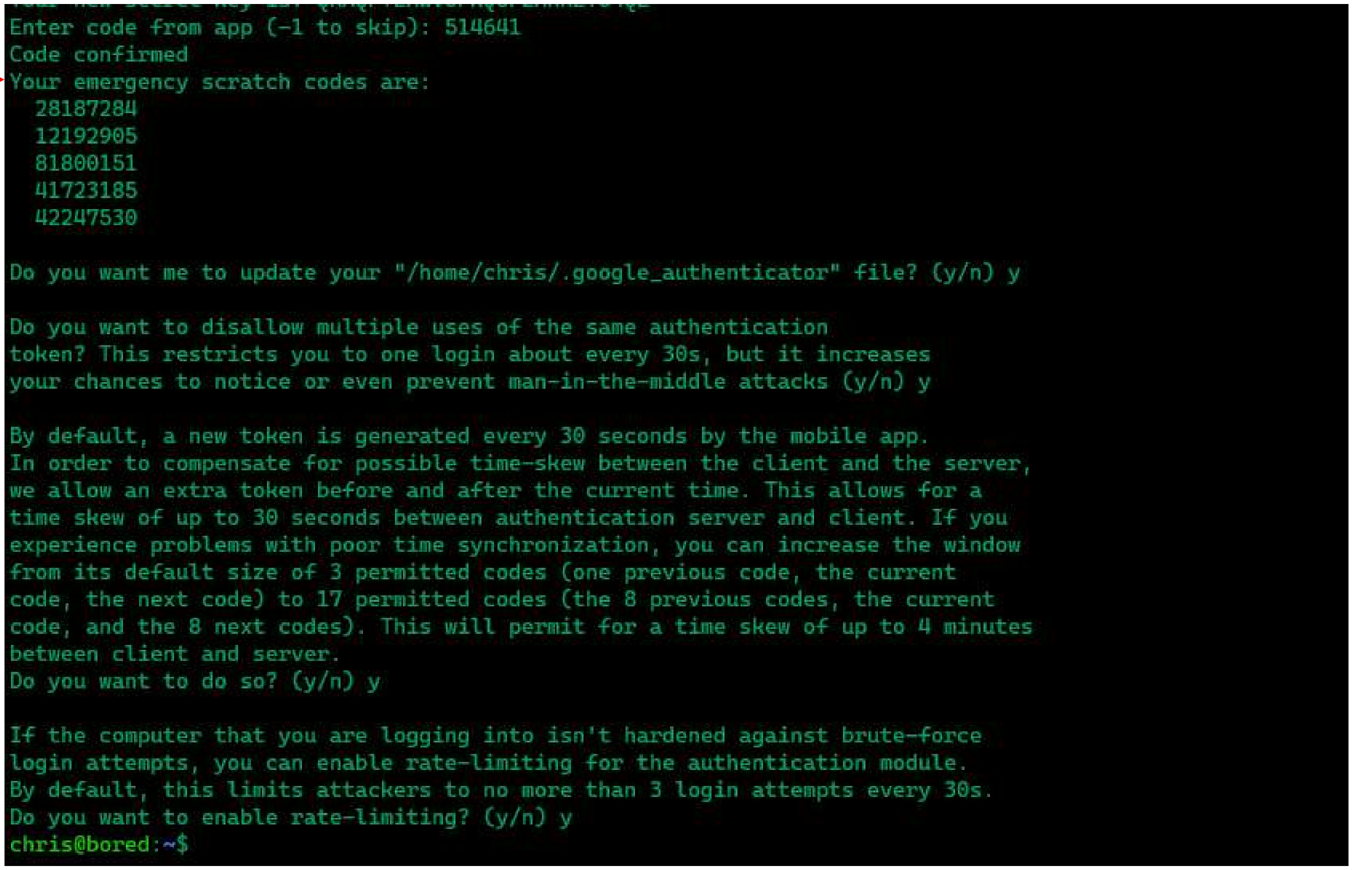


bored: chris@bored

208 879

Once confirmed, you will be provided 5 emergency codes in case you lose access to the authenticator app

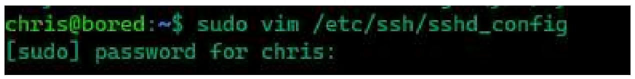
Then continue by answering “y” to the next few questions.



```
[illegible]
Enter code from app (-1 to skip): 514641
Code confirmed
Your emergency scratch codes are:
  28187284
  12192905
  81800151
  41723185
  42247530

Do you want me to update your "/home/chris/.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) y

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
chris@bored:~$
```

Next we need to modify the ssh daemon to use the google authenticator. The path for this is going to be /etc/ssh/sshd_config and lets use Vim or “vi” to make the edits. Not only because we have skill but also this is the default editor and you may not always have Nano or be able to install it (air gap system).

Input: sudo vim /etc/ssh/sshd_config



```
chris@bored:~$ sudo vim /etc/ssh/sshd_config
[sudo] password for chris:
```

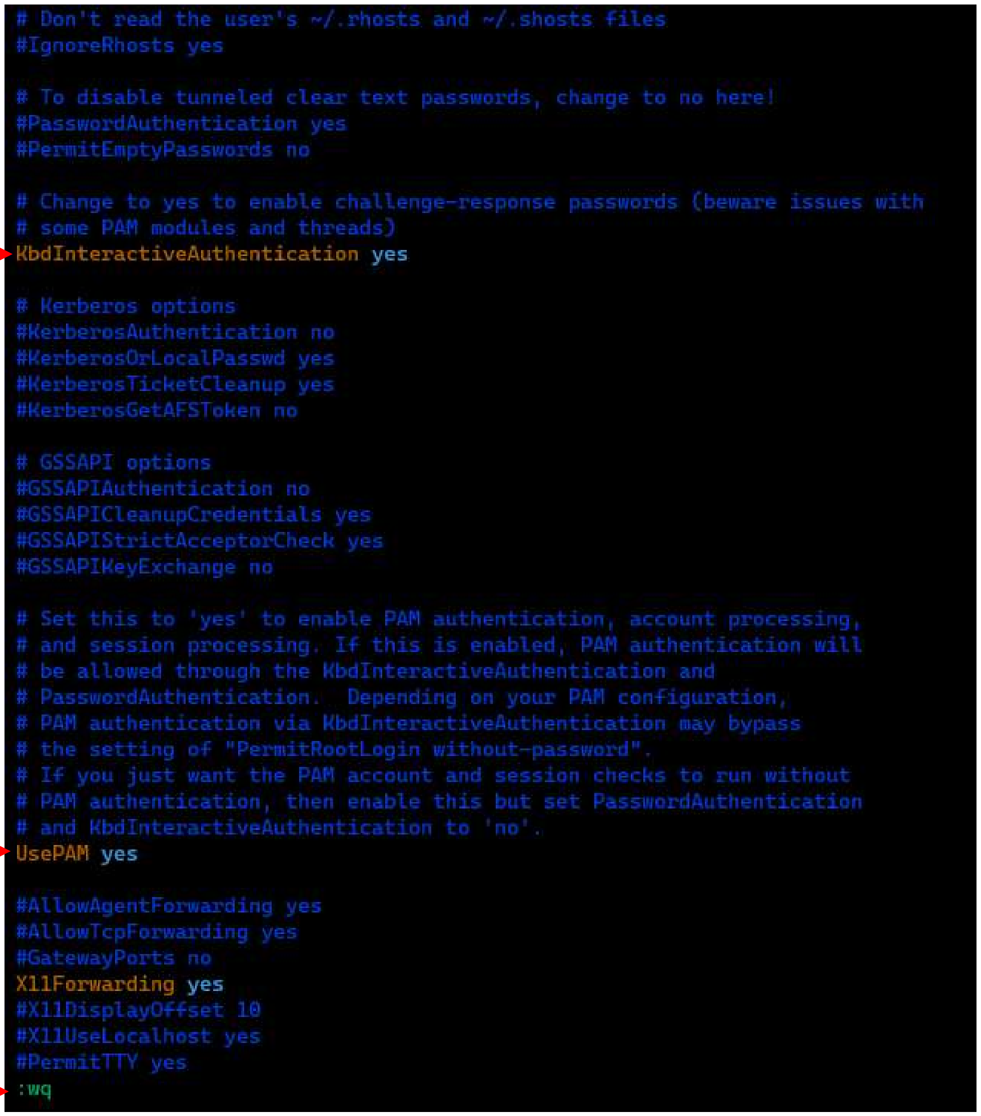
*We are now in the editor and going to make a couple changes. KbdInteractiveAuthentication **yes***

Kbd is for keyboard and what this is does is tells the system we are going to request more information beyond the password

*Next we want to make sure UsePAM is set to **yes***

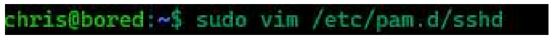
PAM or Pluggable Authentication Module is what allows us to bring in additional modes of authentication

Don’t forget “i” to enter INSERT mode and then “esc” back to normal mode. We then enter command mode using “:” and then input wq to write (save) changes and quit.



```
# Don't read the user's ~/.rhosts and ~/.shosts files
#IgnoreRhosts yes

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

# Change to yes to enable challenge-response passwords (beware issues with
# some PAM modules and threads)
KbdInteractiveAuthentication yes

# Kerberos options
#KerberosAuthentication no
#KerberosOrLocalPasswd yes
#KerberosTicketCleanup yes
#KerberosGetAFSToken no

# GSSAPI options
#GSSAPIAuthentication no
#GSSAPICleanupCredentials yes
#GSSAPIStrictAcceptorCheck yes
#GSSAPIKeyExchange no

# 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

#AllowAgentForwarding yes
#AllowTcpForwarding yes
#GatewayPorts no
X11Forwarding yes
#X11DisplayOffset 10
#X11UseLocalhost yes
#PermitTTY yes
:wq
```

Next we need to modify the PAM rule file for the SSH Daemon.

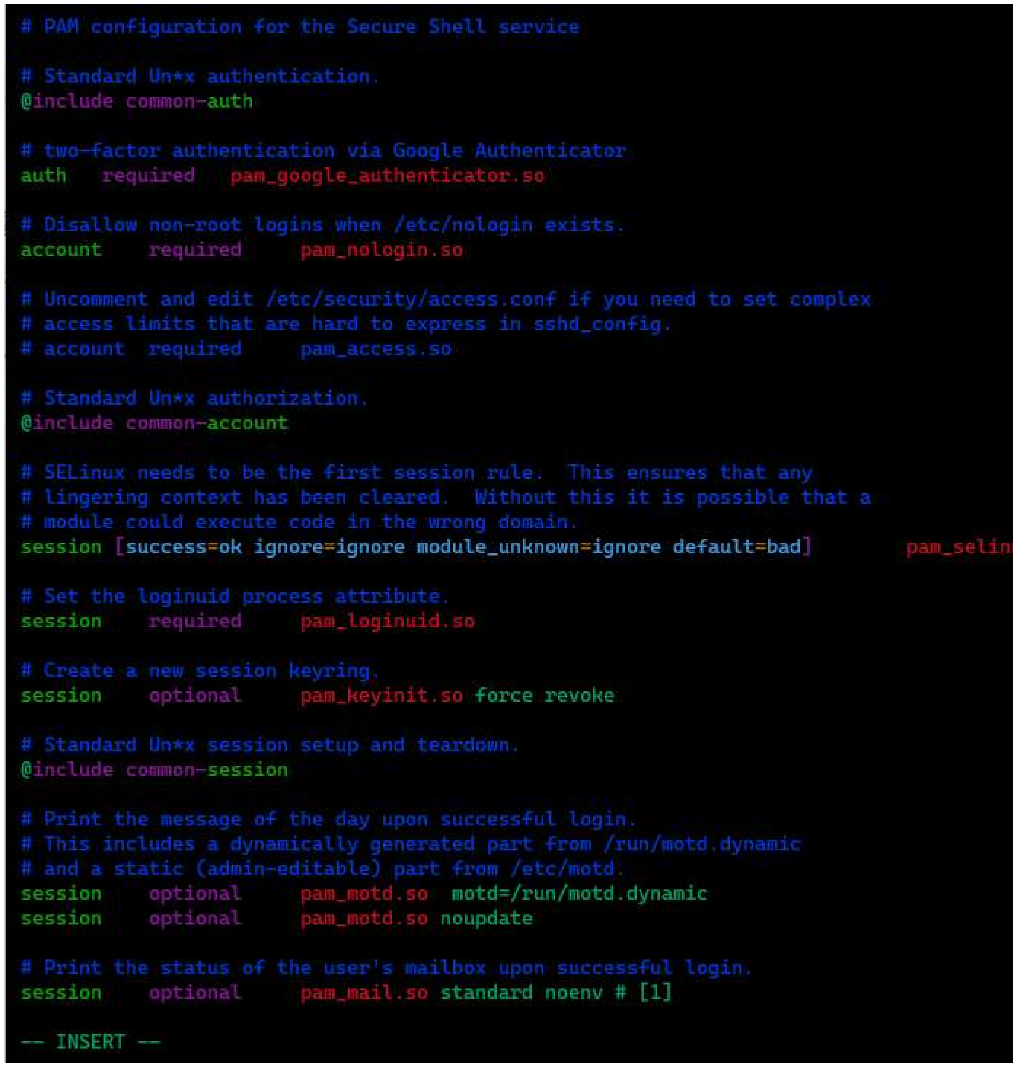


```
chris@bored:~$ sudo vim /etc/pam.d/sshd
```

Input: sudo vim /etc/pam.d/sshd

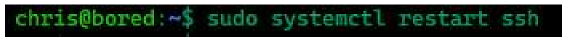
Then we need to add the following lines:
# two-factor authentication via Google Authenticator
auth   required   pam_google_authenticator.so

Don’t forget “i” to enter INSERT mode and then “esc” back to normal mode. We then enter command mode using “:” and then input wq to write (save) changes and quit.



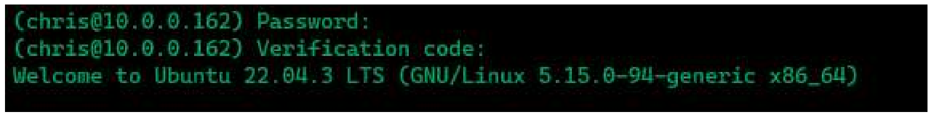
```
# PAM configuration for the Secure Shell service

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

# two-factor authentication via Google Authenticator
auth    required    pam_google_authenticator.so

# 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_selin

# 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
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]

-- INSERT --
```

Now we just about there. We need to restart the ssh daemon so we input
sudo systemctl restart ssh



```
chris@bored:~$ sudo systemctl restart ssh
```

Then to verify it worked logout and back in and you should now be asked for your password followed by a verification code



```
(chris@10.0.0.162) Password:
(chris@10.0.0.162) Verification code:
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.0-94-generic x86_64)
```