LAB :: SSH 2-Factor Authentication

- `#` super user command.
- `$` normal user command.
- Username `apnic` and password `training`.

VM Details

```
[group01.apnictraining.net] [192.168.30.1]
[group02.apnictraining.net] [192.168.30.2]
......
[group10.apnictraining.net] [192.168.30.10]
[group11.apnictraining.net] [192.168.30.11]
......
[group20.apnictraining.net] [192.168.30.20]
[group21.apnictraining.net] [192.168.30.21]
......
[group30.apnictraining.net] [192.168.30.30]
```

NOTE: We will enable two factor authentication for SSH in our Ubuntu server using an OATH-TOTP (open-authentication time-based one-time password) app in addition to the SSH password. We will us Google Authenticator as the PAM (pluggable authentication module).

Step 1: Install Google Authenticator (or similar apps) on your phone

- Search for `google-autheticator` in Google Play or Apple Store.

Step 2: Install PAM and create an Authentication Key

1. Login to your server and update the repo

   ```
   sudo apt update && sudo apt upgrade
   ```

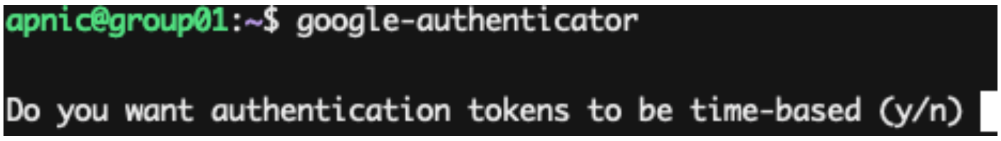
2. Install Google's PAM first:

   ```
   sudo apt install libpam-google-authenticator
   ```

3. Run the `google-authenticator` command and say `yes` when prompted (`y`)to create a

secret key for that user.

```
$ google-authenticator
```

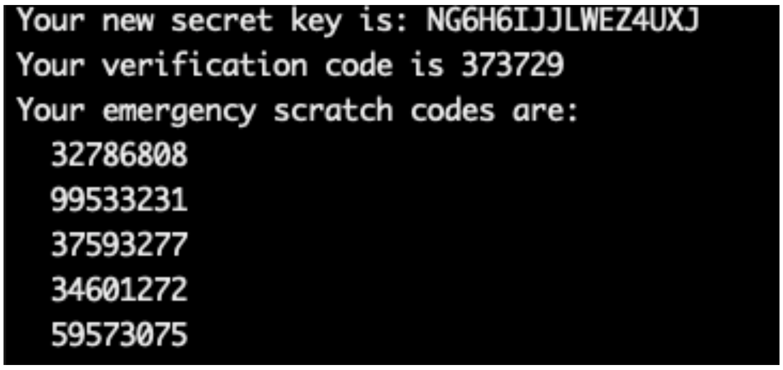


```
apnic@group01:~$ google-authenticator

Do you want authentication tokens to be time-based (y/n)
```

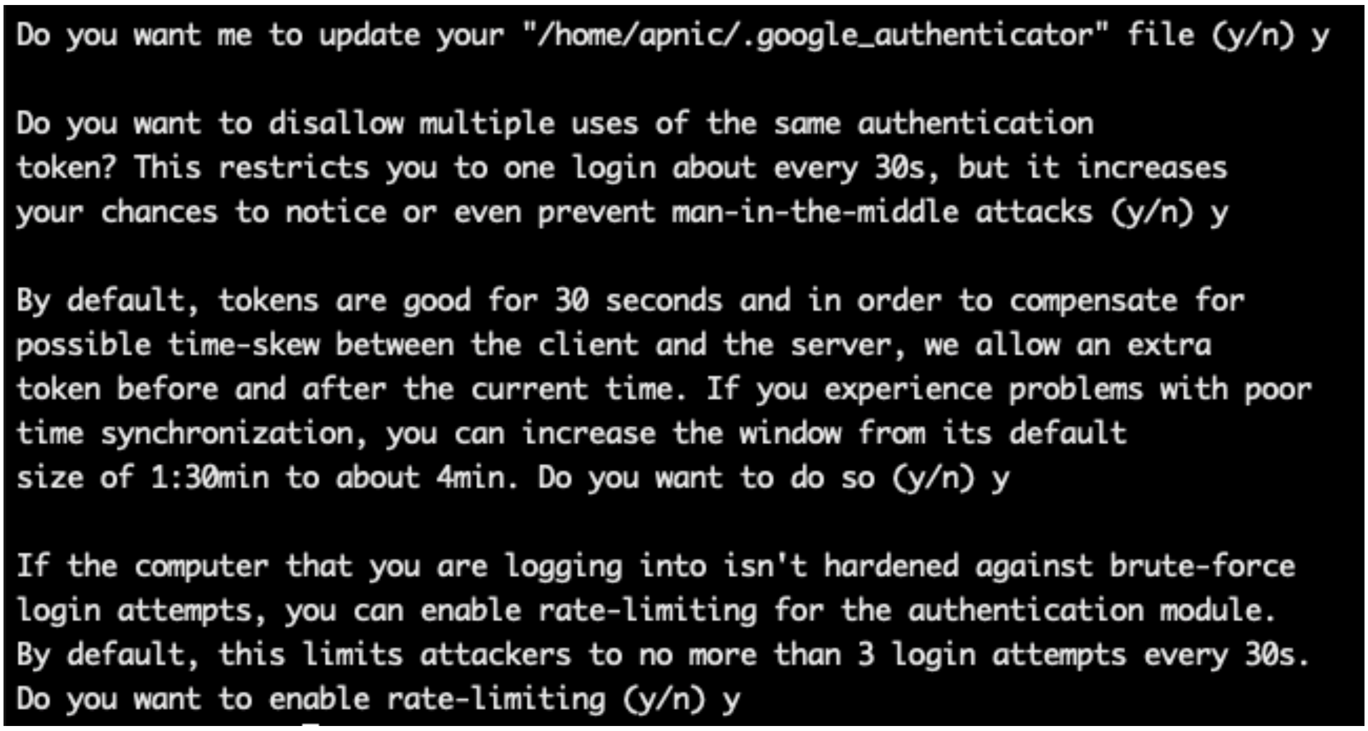
4. You will be presented a QR code ouput like below. Scan the QR code with the authenticator app on your phone:

5. Save the backup codes (secret key and other verfication codes) below your QR code somewhere, in case you have problems scanning the code, or you lose your phone with the Authenticator app (allows you to regain access).
    - **NOTE:** *You can manually add them to Authenticator app on your phone.*



```
Your new secret key is: NG6H6IJJLWEZ4UXJ
Your verification code is 373729
Your emergency scratch codes are:
  32786808
  99533231
  37593277
  34601272
  59573075
```

6. You will be prompted several questions; unless you have a good reason to, the defaults presented are sane. Please read them for your understanding and enter `y` for them.



```
Do you want me to update your "/home/apnic/.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, tokens are good for 30 seconds and 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. If you experience problems with poor
time synchronization, you can increase the window from its default
size of 1:30min to about 4min. 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
```

Step 3: Enable Google's PAM for SSH

1. Edit `/etc/pam.d/sshd` using your favourite text editor, that describes which PAM (pluggable authentication module) modules take care of SSH authentication:

```
sudo vi /etc/pam.d/sshd
```

2. Add the following line at the end of the file (which forces this authentication module to be `required` instead of `optional`):

```
auth required pam_google_authenticator.so
```

`:wq` to save and quit.

NOTE: *Where you might have more than one user accessing a server, you can use the* `nullok` *keyword as shown below. This allows users without a OATH-TOTP token to still log in using their SSH key. Once all users have an OATH-TOTP token, you can remove* `nullok` *from this line to make 2FA mandatory.*

```
auth required pam_google_authenticator.so nullok
```

3. Now we will make changes to the SSH configuration file to support 2-Factor Authentication:

```
sudo vi /etc/ssh/sshd_config
```

4. Search for `ChallengeResponseAuthentication` and replace `no` with `yes`

```
ChallengeResponseAuthentication yes
```

`:wq` Save and quit.

5. Now we need to restart the SSH service to reload the configuration file.

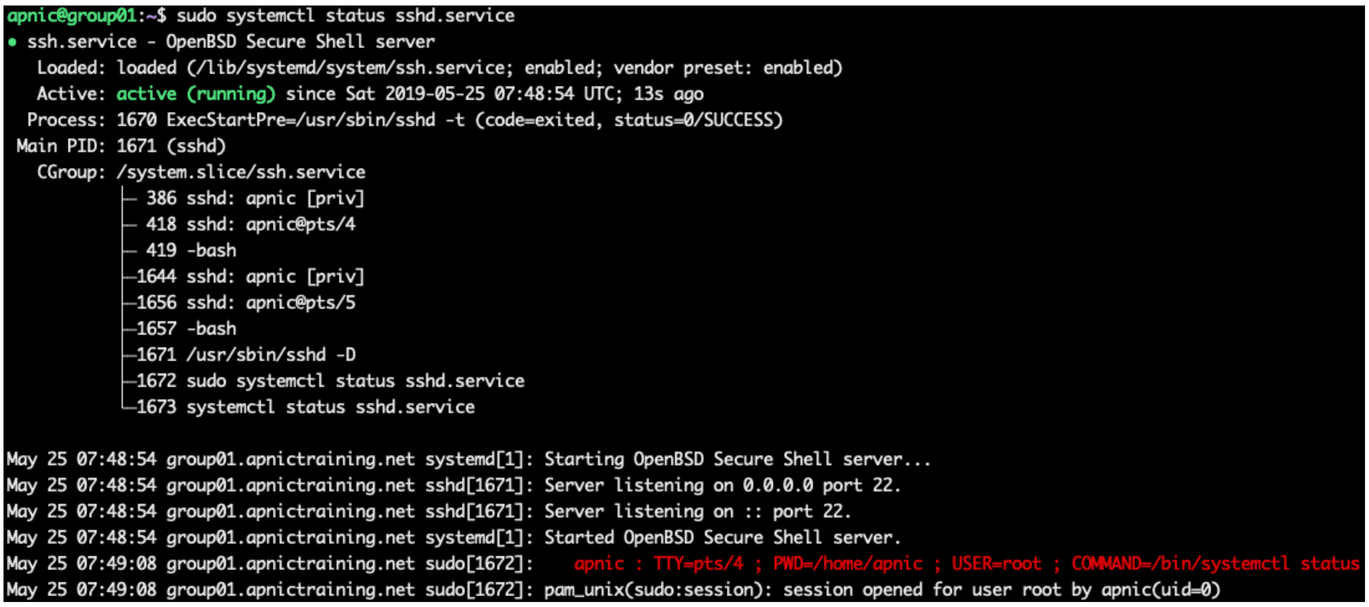
```
sudo systemctl restart sshd.service
```

NOTE: if the above command does not work, use:

```
sudo service ssh restart
```

6. Check the status of the SSH service:

```
sudo systemctl status sshd.service
```



```
apnic@group01:~$ sudo systemctl status sshd.service
● ssh.service - OpenBSD Secure Shell server
   Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enabled)
   Active: active (running) since Sat 2019-05-25 07:48:54 UTC; 13s ago
  Process: 1670 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS)
 Main PID: 1671 (sshd)
   CGroup: /system.slice/ssh.service
           ├─ 386 sshd: apnic [priv]
           ├─ 418 sshd: apnic@pts/4
           ├─ 419 -bash
           ├─1644 sshd: apnic [priv]
           ├─1656 sshd: apnic@pts/5
           ├─1657 -bash
           ├─1671 /usr/sbin/sshd -D
           ├─1672 sudo systemctl status sshd.service
           └─1673 systemctl status sshd.service

May 25 07:48:54 group01.apnictraining.net systemd[1]: Starting OpenBSD Secure Shell server...
May 25 07:48:54 group01.apnictraining.net sshd[1671]: Server listening on 0.0.0.0 port 22.
May 25 07:48:54 group01.apnictraining.net sshd[1671]: Server listening on :: port 22.
May 25 07:48:54 group01.apnictraining.net systemd[1]: Started OpenBSD Secure Shell server.
May 25 07:49:08 group01.apnictraining.net sudo[1672]:    apnic : TTY=pts/4 ; PWD=/home/apnic ; USER=root ; COMMAND=/bin/systemctl status
May 25 07:49:08 group01.apnictraining.net sudo[1672]: pam_unix(sudo:session): session opened for user root by apnic(uid=0)
```

7. ***NOTE:*** *Since we are making all these changes over SSH, make sure you do not exit/close your initial*

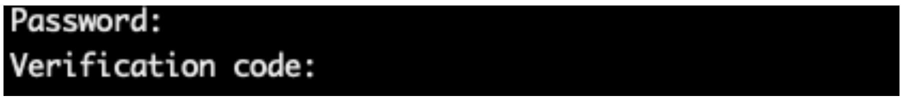
SSH connection. Instead, open a second SSH session to do testing for the next step.

Step 4: Login to the server with 2FA

1. Try to ssh to the server from a ***new terminal*** using the verbose option (`-v`)

```
ssh apnic@192.168.30.X -v
```

2. You should be prompted for a password first and then a verification code (Authenticator app on your phone)



```
Password:
Verification code:
```

```
***END OF EXERCISE***
```