CONFIGURACIÓN DE 2FA EN LINUX

ACT0403 - BASTIONADO DE REDES Y SISTEMAS

ERIC SERRANO MARÍN

Contenido

1. Prepara una PoC (prueba de concepto) en la que muestres a los de Sistemas la simplicidad de desplegar un sistema de 2FA basado en google. Dicho sistema se basa en la librería PAM de Linux.

Primero vamos a instalar el paquete PAM de Google-Authenticator.

sudo apt install libpam-google-authenticator

```
root@CIBER-esermar-LXCUbuntu22:~/Practica# sudo apt install libpam-google-authenticator
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 3 not upgraded.
Need to get 69.7 kB of archives.
After this operation, 205 kB of additional disk space will be used.
Do you want to continue? [Y/n]
```

Después pasaremos a configurar Google PAM en Ubuntu.

nano etc/pam.d/common-auth y añadiremos **auth required pam_google_authenticator.so**.

```
  GNU nano 6.2                          /etc/pam.d/common-auth *
#
# /etc/pam.d/common-auth - authentication settings common to all services
#
# This file is included from other service-specific PAM config files,
# and should contain a list of the authentication modules that define
# the central authentication scheme for use on the system
# (e.g., /etc/shadow, LDAP, Kerberos, etc.).  The default is to use the
# traditional Unix authentication mechanisms.
#
# As of pam 1.0.1-6, this file is managed by pam-auth-update by default.
# To take advantage of this, it is recommended that you configure any
# local modules either before or after the default block, and use
# pam-auth-update to manage selection of other modules.  See
# pam-auth-update(8) for details.

auth required pam_google_authenticator.so

# here are the per-package modules (the "Primary" block)
auth    [success=1 default=ignore]      pam_unix.so nullok
# here's the fallback if no module succeeds
auth    requisite                       pam_deny.so
# prime the stack with a positive return value if there isn't one already;
# this avoids us returning an error just because nothing sets a success code
# since the modules above will each just jump around
auth    required                        pam_permit.so
# and here are more per-package modules (the "Additional" block)
auth    optional                        pam_cap.so
# end of pam-auth-update config
```

Guardamos y cerramos.

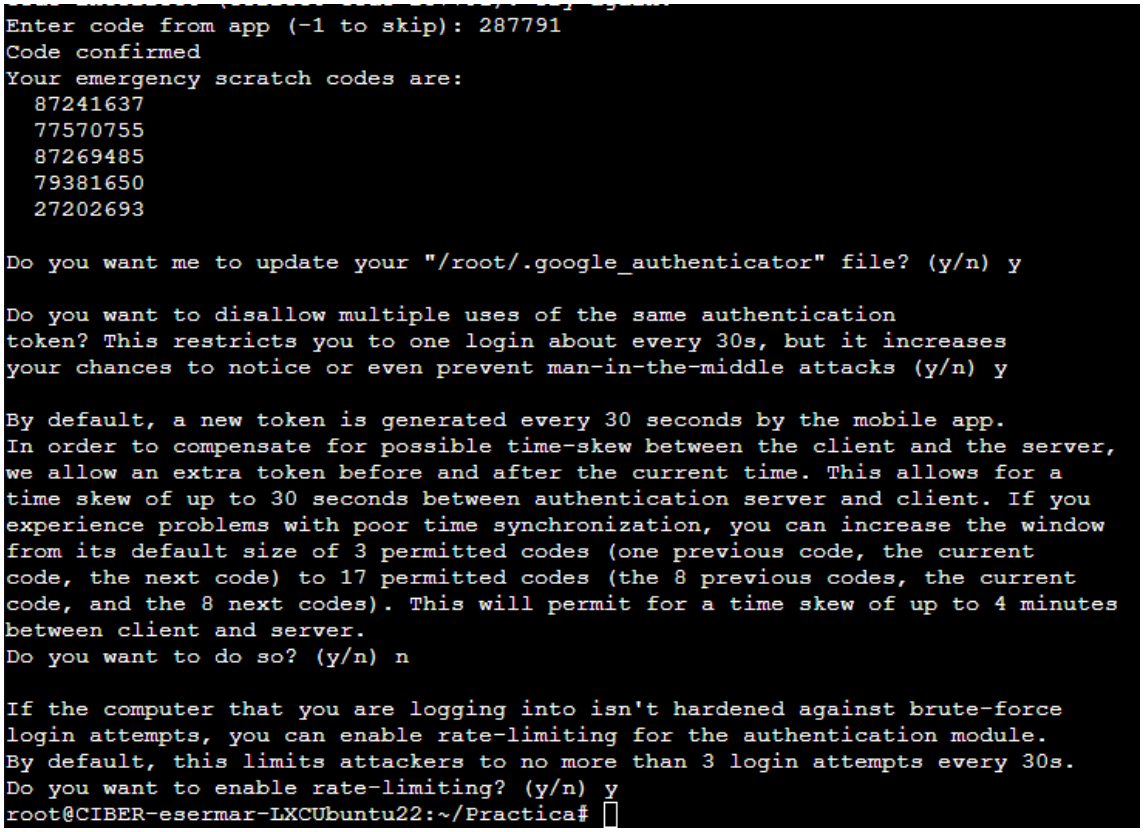
Ahora inicializaremos PAM.

google-authenticator

Este código QR lo vamos a escanear con nuestra aplicación previamente instalada “Google Authenticator”. También existe otra opción mediante código, que sería insertando la secret key que nos da abajo del QR, yo personalmente lo he hecho escaneando.

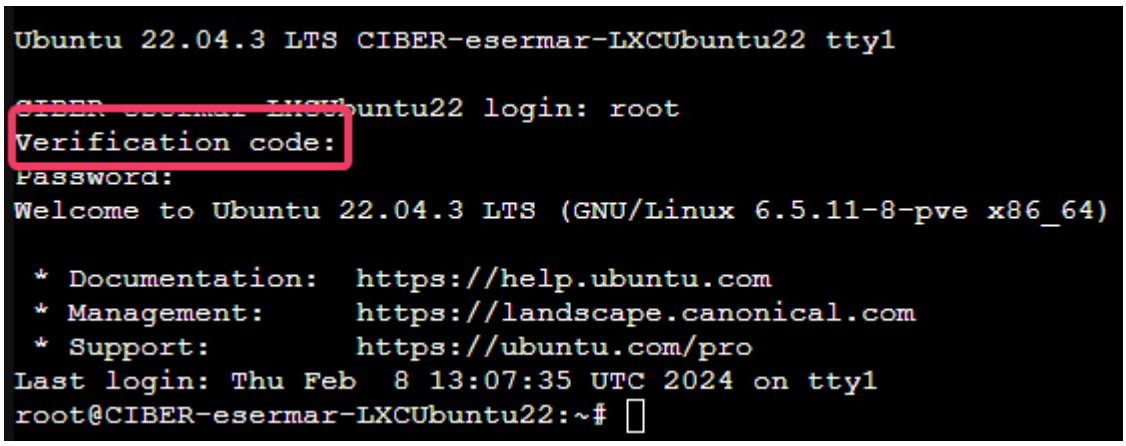
En cuanto escaneemos o metamos la secret key nos aparecerá en la aplicación una contraseña de 6 dígitos que irá cambiando cada 30 segundos.

El último mensaje nos dice “Enter code from app”, pues eso haremos, meteremos el código de 6 dígitos que nos aparece.



- **Primer mensaje**: le diremos “y” que es para hacer un update al archivo en nuestra carpeta.
- **Segundo mensaje**: le diremos también “y”, este es para restringir el inicio de sesión a solo un registro cada 30 segundos, así evitaremos ataques de man-in-the-middle.
- **Tercer mensaje**: “n”, para no permitir extender la duración del tiempo para ingresar el código de verificación. Es más seguro decirle que no, a menos que tengamos problemas de sincronización de tiempo entre servidor y los dispositivos de los usuarios. Decir no a esta opción garantizará una mayor seguridad en el proceso de autenticación.
- **Cuarto mensaje**: “y”, este será para que solo haga 3 intentos de login.

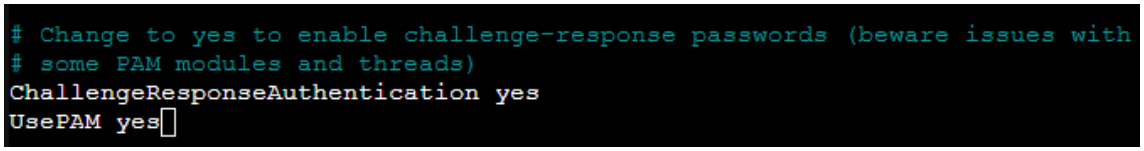
Ahora si hacemos un reboot y intentamos entrar a nuestro Ubuntu nos pedirá el código de verificación.



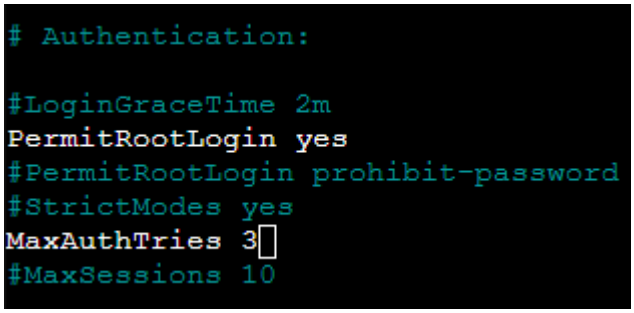
2. Integración de SSH con Google Authenticator.

Para habilitar SSH para usuarios vamos a añadir al archivo sshd_config lo siguiente:

nano /etc/ssh/sshd_config

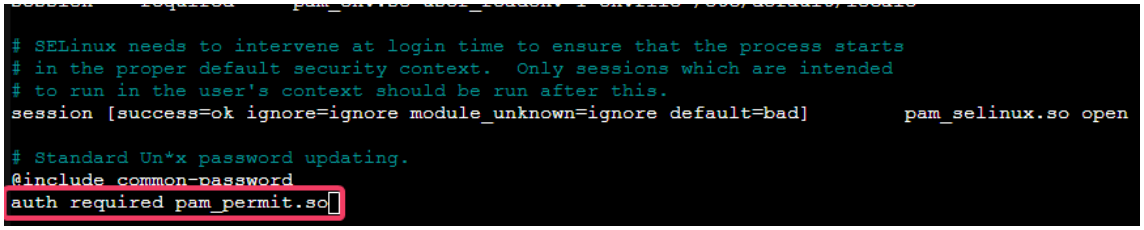


Ahora vamos a poner para que solo se puedan hacer 3 intentos.

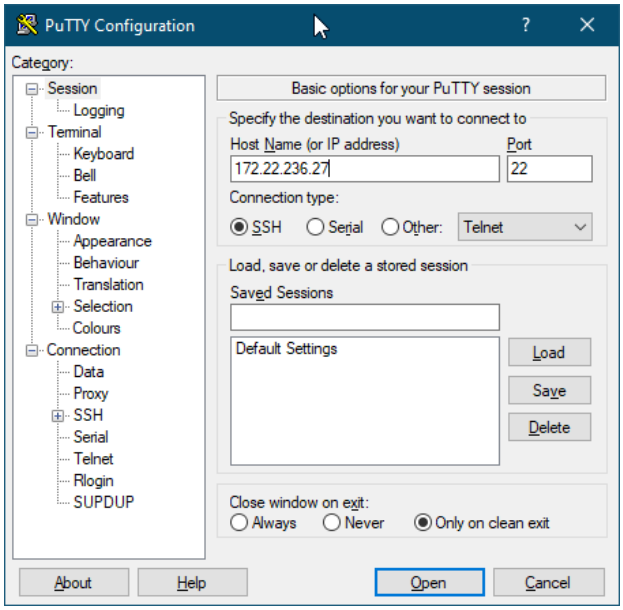


Ahora vamos a modificar la norma de PAM para SSH.

nano etc/pam.d/sshd



Y aquí tenemos como conectándonos mediante SSH con Putty nos pide la verificación y nos deja entrar.



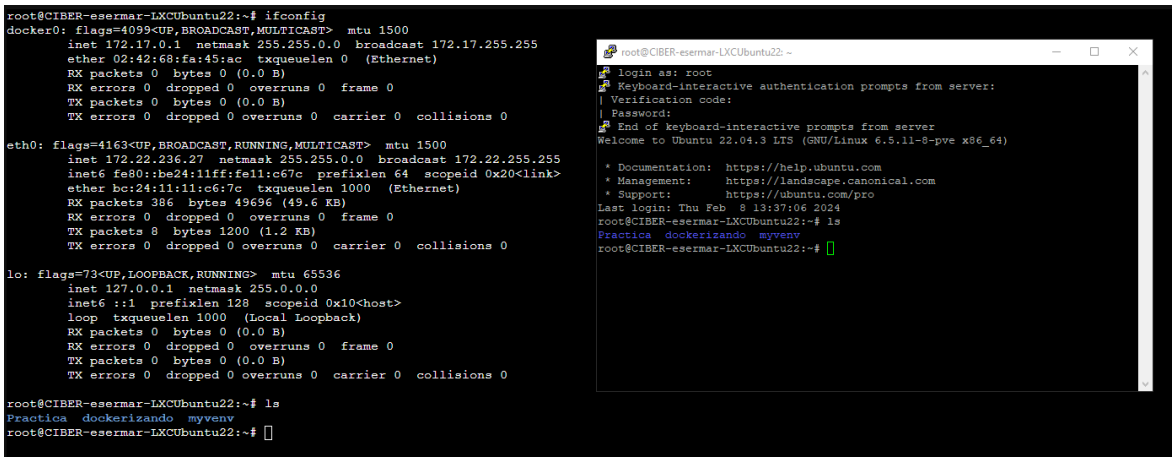
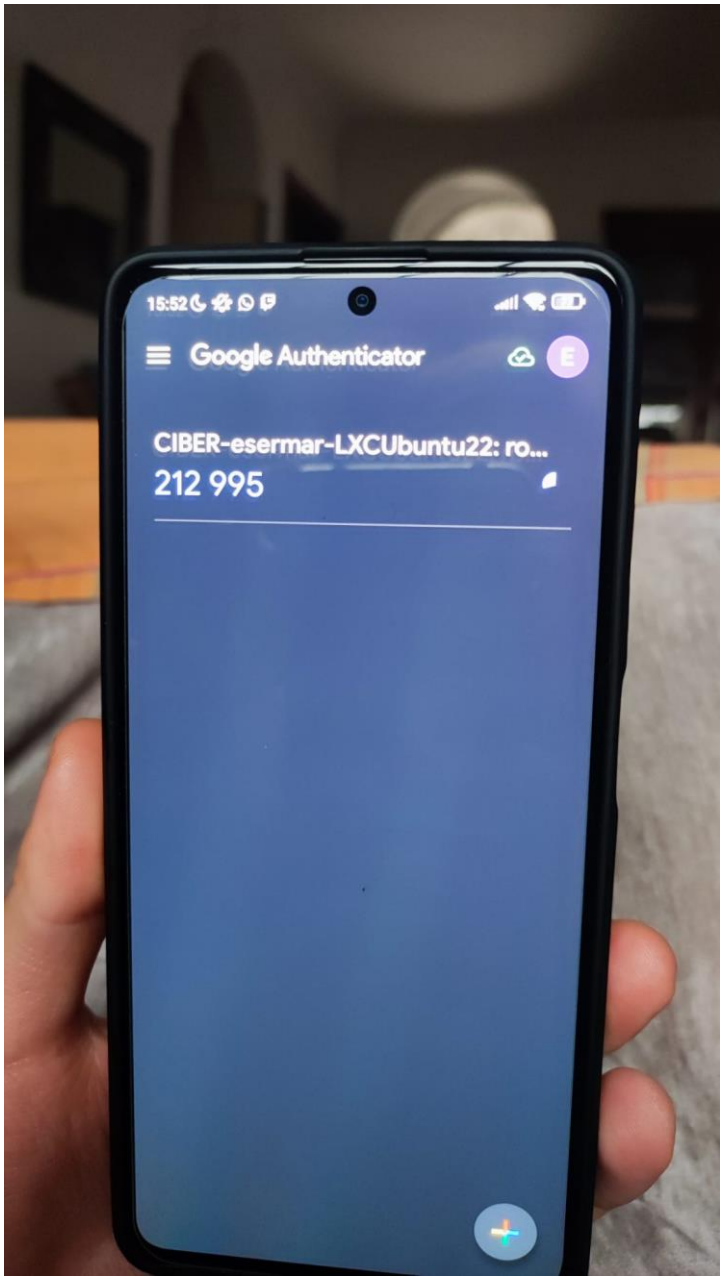


Foto de mi móvil con Google Authentificator.



CONCLUSION

Hacer esta práctica me ha demostrado la efectividad y la simplicidad de implementar el sistema de autentificación de doble factor utilizando Google Authenticator en un entorno Linux. Además he podido aprender como esta solución puede fortalecer significativamente la seguridad de nuestros sistemas, no sólo al requerir doble autentificación sino que también por ser un código único generado en tiempo real y que va cambiando cada 30 segundos.