

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.	2
2. Integración de SSH con Google Authenticator.	5
CONCLUSION	8

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.

```
Enter code from app (-1 to skip): 287791
Code confirmed
Your emergency scratch codes are:
  87241637
  77570755
  87269485
  79381650
  27202693

Do you want me to update your "/root/.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
root@CIBER-esermar-LXCUbuntu22:~/Practica# █
```

- **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.

```
Ubuntu 22.04.3 LTS CIBER-esermar-LXCUbuntu22 tty1
CIBER-esermar-LXCUbuntu22 login: root
Verification code:
Password:
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 6.5.11-8-pve x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/pro
Last login: Thu Feb  8 13:07:35 UTC 2024 on tty1
root@CIBER-esermar-LXCUbuntu22:~#
```

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

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

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

```
# Authentication:

#LoginGraceTime 2m
PermitRootLogin yes
#PermitRootLogin prohibit-password
#StrictModes yes
MaxAuthTries 3
#MaxSessions 10
```

Ahora vamos a modificar la norma de PAM para SSH.

nano etc/pam.d/sshd

```
# 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_permit.so
```

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

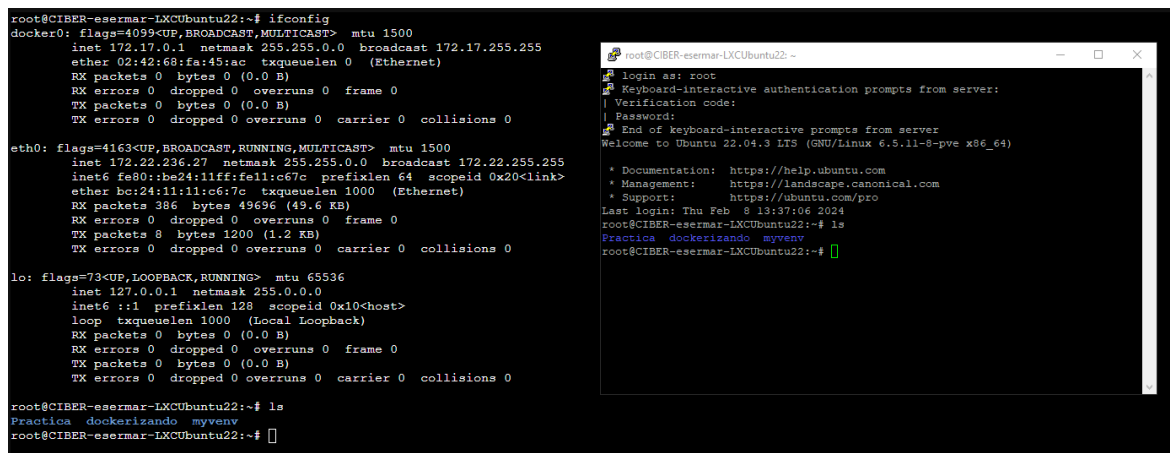
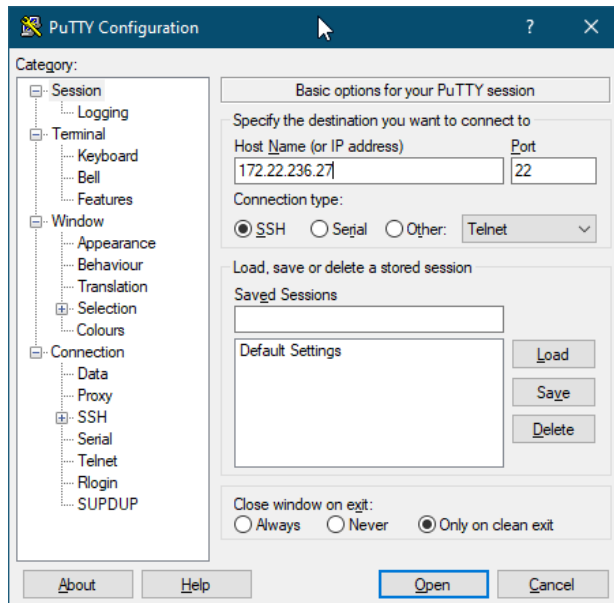
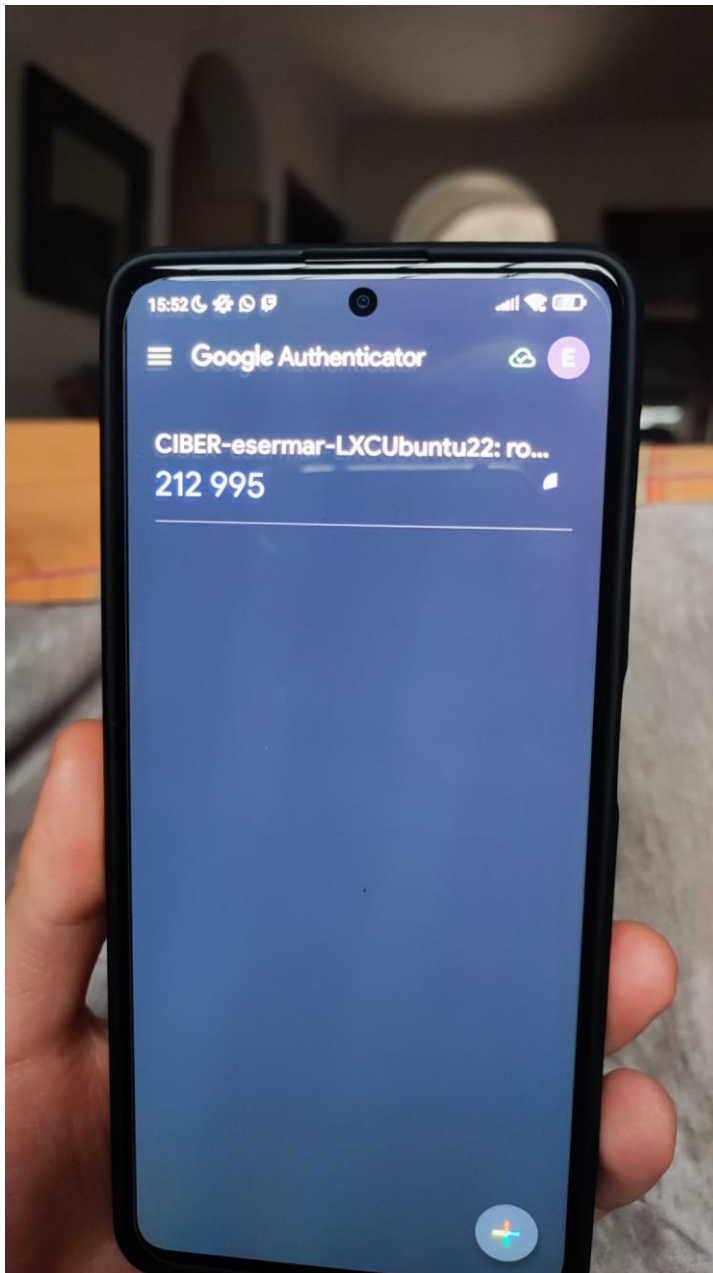


Foto de mi móvil con Google Authenticator.



CONCLUSION

Hacer esta práctica me ha demostrado la efectividad y la simplicidad de implementar el sistema de autenticación 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 autenticación sino que también por ser un código único generado en tiempo real y que va cambiando cada 30 segundos.