Implementing MFA on a Linux Server

Written by Tyler Weiss

Assignment

In this assignment , you will implement Multi-Factor Authentication (MFA) on a Linux server using CLI commands. Capture the setup process with screenshots, starting from initialization to final verification. Then, in a brief narrative, define MFA, discuss its setup steps, and highlight its importance in cybersecurity.

Please compile your screenshots and written explanation into a clear and concise document. This document should serve as both a technical guide and a conceptual overview of MFA's importance in cybersecurity.

Reference:
https://www.linuxbabe.com/ubuntu/two-factor-authentication-ssh-key-ubuntu

Exercise 1

Multi-Factor Authentication (MFA) is a multi-layered approach to authenticating access by using two or more methods of verification; something you know, something you have, something you are, and something you do. MFA can deny access by unauthorized persons even if they posses the proper credentials. Phishing, spear phishing, keyloggers, credential stuffing, brute force, reverse brute force, and man-in-the-middle (MITM) attacks can be prevented using MFA. There are two types of One Time Passwords (OTP); TOTP and HMAC-based One-Time Passwords (HOTP). In this exercise we will use a password that you remember, something you know, and a TOTP with a cell phone, something you have.

In order to setup MFA on a Linux server we must first install software that will that will create a secure connection between the server and the cell phone app. This software application is called 'Google Authenticator'. Next an application must be installed on

phone to provide association with the server and the TOTPs. Once installed and run, Google Authenticator will generate a secret key that can be scan or input into the app to create a relationship between the app and the server. After this relationship is established the ssh daemon and PAM configuration files on the server must be modified to allow the use of MFA using Google Authenticator. Finally the ssh daemon needs to be restarted for changes to the settings to take effect. If all steps have been done properly you will be prompted to input a TOTP, provided by the phone app, during the ssh login process and you will be granted access.

The first step is to install 'libpam-google-authenticator' which will be used to support 2FA on the the Linux server. Login to the server and run the following command.

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

```
rec0nrat@demoserver1:~$ sudo apt install libpam-google-authenticator -y
[sudo] password for rec0nrat:
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
```

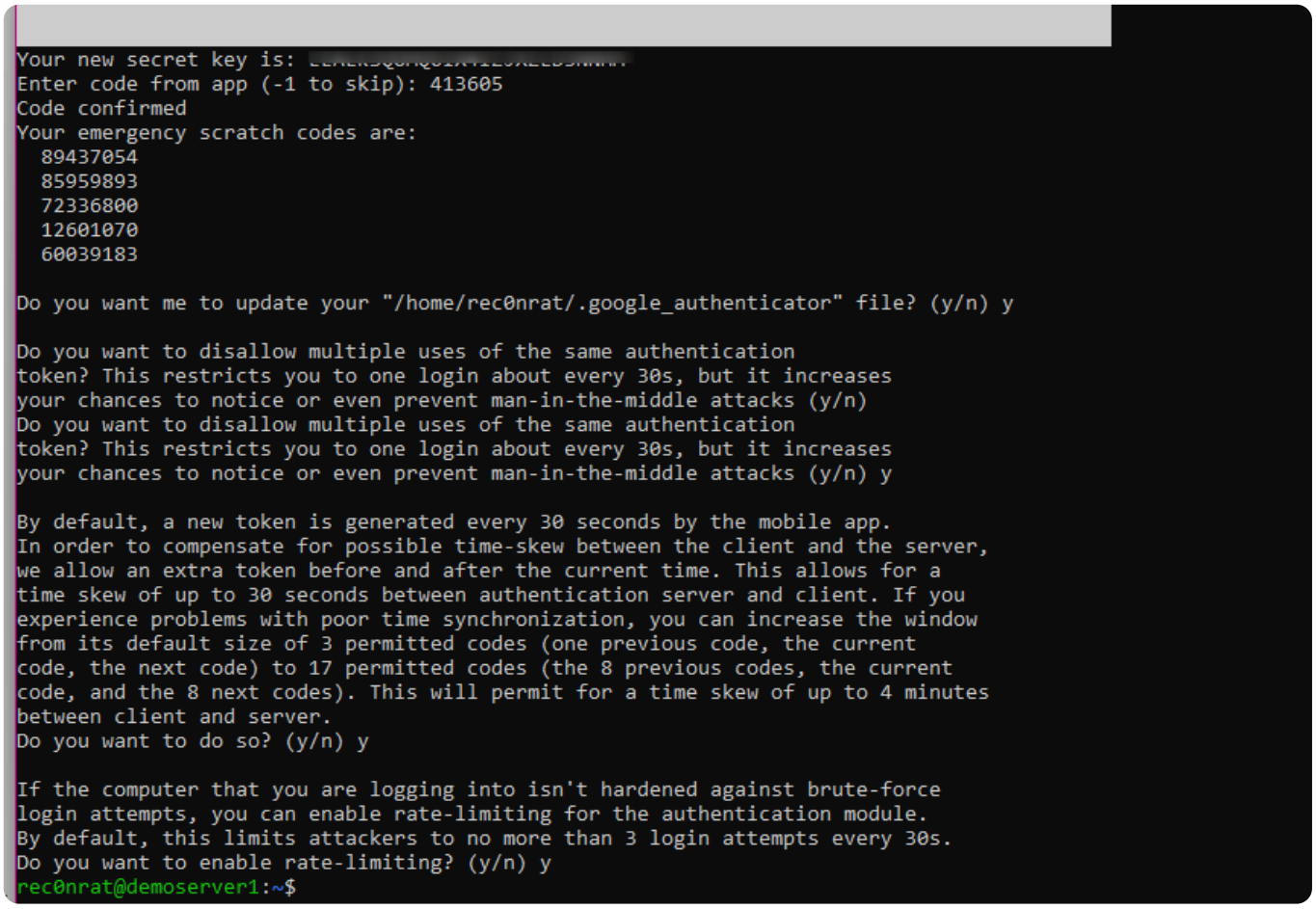
Before continuing the Google Authenticator mobile app or FreeOTP, an open-source Time-Based One Time Password (TOTP) mobile app developed by Red Hat. When the QR code is presented scan it into the app and type the one-time password in the prompt on the server. Keep in mind the the password will only be present for 30 seconds and must be used during that time. Run the app that was just install to create to create a new secret key. When asked if you want to use time-based authentication token type 'y'.

```
rec0nrat@demoserver1:~$ google-authenticator

Do you want authentication tokens to be time-based (y/n) y
Warning: pasting the following URL into your browser exposes the OTP secret to Google:
  https://www.google.com/chart?chs=200x200&chld=M|0&cht=qr&chl=otpauth://totp/rec0nrat@demoserver1%3Fsecret%3DEEALKSQ
OMQ6IX4IZJXZLD3NNHM%26issuer%3Ddemoserver1
```

Once the password has been input emergency scratch codes will be presented and you questions about the configuration of the app. In this example we accept all suggestions. The QR code represents the secret key. Save the secret key and emergency scratch code

somewhere safe because you may need them to access the server in case something goes wrong.



To allow authentication using using a password login via ssh we need to edit the '/etc/ssh/sshd_config' file.

```
sudo vim /etc/ssh/sshd_config
```

If you are using Ubuntu 22.04 set the following options:

```
UsePAM yes
KbdInteractiveAuthentication yes
```

Otherwise set these options below:

```
UsePAM yes
ChallengeResponseAuthentication yes
```

If you want to add public key authentication also add this line to the file:

```
AuthenticationMethods publickey,keyboard-interactive
```

The Pluggable Authentication Module provides an easy way to plug different authentication methods into Linux. To enable the authenticator app with SSH, PAM and Challenge-Response authentication must be enabled. To allow the root user to login using 2FA the 'PermitRootLogin' must be set to 'yes'. Save and close the file.

```
# 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
```

Now edit the PAM rule file for SSH daemon by adding the following lines are added to the '/etc/pam.d/sshd' file. This authorizes the use of the authenticator app during login. Then save and close the rule file.

```
@include common-auth

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

```
rec0nrat@demoserver1:~$ sudo vim /etc/pam.d/sshd
[sudo] password for rec0nrat:
```

```
# 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
```

Finally restart the ssh daemon allowing the changes to take effect.

```
sudo systemctl restart ssh
```

```
rec0nrat@demoserver1:~$ sudo systemctl restart ssh
rec0nrat@demoserver1:~$
```

Logout and lg back in. you should be prompted to use the authenticator app and input the password before access to the server is granted.