Multi-Factor Authentication

OLUG Presentation
January 7, 2025

By Aaron Grothe

Introduction

Passwords alone aren't enough any more. You should be using two factor authentication nowadays.

This talk will be about enabling mfa using Google's multifactor authenticator software in Linux at both the local login level and via ssh.

Links are at the end of the talk

Slides are posted at my website
https://www.grothe.us/presentations

Introduction (Continued)

If you have questions/comments please feel free to ask them anytime. You don't have to hold them until the end of the talk.

If there are other resources similar to these that you think might be useful to people please let the group know.

Hopefully this will be an interactive and productive session.

Disclaimer

You can have real issues and can lock yourself out of your system by setting this up.

Tips

- Have an SSH session already established, then open a new session so you can undo changes
- Make backups of all the pam.d config files
- Small incremental changes
- Accept you're going to have issues with this
- Know how to use console if you're doing it remotely

Types of Authentication

- Something You Know: This typically refers to passwords, PINs, or secret questions and answers.
- Something You Have: This involves possessing a physical or virtual object, such as a security token, smart card, mobile phone, or authentication app.
- Something You Are: This utilizes biometric characteristics, such as fingerprints, facial recognition, voice recognition, iris scans, or retinal scans.

Types of Authentication

Something You Know: This typically refers to passwords, PINs, or secret questions and answers.

Pros: Relatively easy to implement.
Cons: Susceptible to phishing, keylogging, and social engineering attacks.

Types of Authentication

Something You Have: This involves possessing a physical or virtual object, such as a security token, smart card, mobile phone, or authentication app.

Pros: Adds an extra layer of security beyond passwords.
Cons: Can be lost, stolen, or compromised if not properly secured.

Types of Authentication

Something You Are: This utilizes biometric characteristics, such as fingerprints, facial recognition, voice recognition, iris scans, or retinal scans.

Pros: Highly unique and difficult to replicate.
Cons: Can be susceptible to spoofing attacks in some cases. Also changing it can be very difficult/impossible.

Google Multifactor Authentication

Google's libpam-google-authenticator is available for most distros

To install on Ubuntu debian

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

To install on Fedora

```
# sudo dnf install google-authenticator

# sudo dnf install qrencode-libs
```

Google Multifactor Authentication

To install on RHEL/Cloud Linux/Alma, Etc.

Need to install EPEL repos first

```
# sudo dnf install epel-release
```

Then same as Fedora

```
# sudo dnf install google-authenticator

# sudo dnf install qrencode-libs
```

Setting up Google-Authenticator

```
$ google-authenticator
```

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

-> This controls whether the token is time based. Your system clock and phone clock needs to be relatively in sync. You probably want this.

Setting up Google-Authenticator

Your new secret key is: IIC4GBXT3BZA64YCIIV3AZKNZI

Generates QR code and secret key for setting up authenticator on your phone, device.

QR code is easier

Setting up Google-Authenticator

```
Enter code from app (-1 to skip): 281151
```

-> Enter the code from your authenticator app. This syncs the two devices and confirms they are able to synchronize

Setting up Google-Authenticator

```
Code confirmed
Your emergency scratch codes are:
  82938582
  31849142
  78882876
  97387681
  42872527
```

-> Code has been confirmed. Devices are in Sync
-> emergency scratch codes are one time use codes in case you lose your authenticator or have issues. Each is a one time use. Save these somewhere.

Setting up Google-Authenticator

Do you want me to update your "/home/grothe/.google_authenticator" file? (y/n)

-> Save this setup? If you don't nothing is saved on the ubuntu box.
-> You want to save the settings probably

Setting up Google-Authenticator

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)

-> Pretty much as described. One thing to keep in mind if you set this and you have it setup for sudo as well. You will have to wait 30 seconds after logging in before you can sudo.

Setting up Google-Authenticator

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)

Setting up Google-Authenticator

-> explains it pretty well. Answer y and it lets you have a time skew of 4 minutes instead of only 90 seconds. You probably want “y” for this.

Setting up Google-Authenticator

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)

-> Rate limiting is good. You probably want this by, so “y”

Contents of .google-authenticator file

```
grothe@ubuntumfa:~$ cat .google_authenticator
IIC4GBXT3BZA64YCIIV3AZKNZI
" RATE_LIMIT 3 30
" WINDOW_SIZE 17
" DISALLOW_REUSE
" TOTP_AUTH
82938582
31849142
78882876
97387681
42872527
```

Experiment #1 - Local Login/Sudo/Ssh

We'll add the additional factor to the gdm3 login and sudo

```
# cd /etc/pam.d
# cp common-auth common-auth.sav
```

Append to end of file

```
auth required pam_google_authenticator.so
```

Reboot system

```
# reboot
```

Experiment #1 - Local Login/Sudo/Ssh

Task #1: Login to system, does it ask for Verification code?

Success?

Task #2: Sudo, does it ask for Verification code?

Success?

Task #3: SSH to localhost?

Failure

Experiment #1 - Local Login/Sudo/Ssh

Why does SSH to localhost fail?

We need to tell the ssh daemon that it needs to ask for the verification code.

```
# cd /etc/ssh
# cp sshd_config sshd_config.sav
# vi sshd_config
```

Experiment #1 - Local Login/Sudo/Ssh

What do we need to change

Look for the line

KbdInteractiveAuthentication no

Change this to yes. On older systems this might be called

ChallengeResponseAuthentication no

In that case change it to yes

Experiment #1 - Local Login/Sudo/Ssh

Time to restart the sshd daemon

```
# systemctl restart ssh
```

Task #3b: SSH to localhost?

Success?

Experiment #1 - Results

So. Is it working? Yes

Experiment #2 - User without google-auth

Time to create a new user without google-authentication setup

```
# adduser aaron
```

Step through options

Setup in groups the same as grothe user

```
# vi /etc/group
```

Experiment #2 - User without google-auth

```
% ssh aaron@localhost
```

Fails. It asking for the verification code.

Three options

1. Modify the /etc/pam.d/common-auth to allow nullok
2. Setup with a key so doesn't need google-auth
3. Run google-authenticator for user

Experiment #2 - User without google-auth

We'll go with Option #1, and add nullok to common-auth

```
# cd /etc/pam.d
# cp common-auth common-auth.sav.sav
# vi common-auth
```

Experiment #2 - User without google-auth

Change auth required line from

```
auth required pam_google_authenticator.so
```

To

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

Restart sshd

```
# systemctl restart ssh
```

Experiment #2 - User without google-auth

Attempt login

```
% ssh aaron@localhost
```

Only asks for password, success

Experiment #3 - Only SSH

We'll reset common-auth

```
# cd /etc/pam.d
# cp common-auth.sav common-auth

# cp sshd sshd.sav
```

Experiment #3 - Only SSH

```
# vi sshd
```

Add following line to the end

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

Restart ssh

```
# systemctl restart ssh
# reboot
```

Experiment #3 - Only SSH

Login - no verification required

We'll go ahead and ssh to ourselves

```
# ssh grothe@localhost
```

It asks for the verification code.

We can also add this line to the sudo and it will force it for sudo

Experiment #4 - SSH Keys

Generate key for grothe@localhost account

```
% ssh-keygen
```

Install ssh-key on localhost

```
% ssh-copy-id grothe@localhost
```

Asks for password and verification code

Experiment #4 - SSH Keys

Time to login

```
% ssh grothe@localhost
```

No verification code request. Why?

By default - keys in ssh take precedence over passwords

This can be changed, but isn't perfect.

Experiment #4 - SSH Keys

How to change this?

We'll go all out. Set it to require the password, the public key and the verification code

```
# vi /etc/ssh/sshd_config
```

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

Experiment #4 - SSH Keys

Restart the sshd daemon

```
# systemctl restart sshd
```

Login with verbose so we can see all the information

```
# ssh -v grothe@localhost
```

Login with public key, password, and verification code

Summary

So we've done 4 classes of experiments here

1. Setup for Local login, Sudo
2. User without Google Auth (allowing)
3. SSH only, no local login, no Sudo
4. SSH with public key, password, and verification code

Gives you a bit of an overview of some of the options

Caveats - Display Manager

Display Manager issues

Doesn't work with a lot of Display managers if you're using it for login.

Works with gdm3

Doesn't work with sddm, lightdm, etc.

You mileage may vary

Caveats - Configuration

How to setup for multiple users?

You can create a basic profile and put it into /etc/skel. That way when you create users that will be copied by default

If you do that it will have the same scratch codes, which is a potential security issue. You can use your provisioning code to generate different emergency scratch codes

Five Things I Wish I Knew

- Need the right display manager (gdm3), spent quite a bit of time trying to get this working with kubuntu/ssdm, kubuntu/gdm3 works fine
- Turning on verbose mode -v to -vvvv with ssh are very useful to find out what is going on with your connections
- You can't login via ssh and then jump to sudo right away. You need to wait for the next code. Don't know why I was trying this
- Keep your old ssh session open until you make sure the new session is working
- Nullok and an account that doesn't have mfa setup can save your bacon as well

Would I use it?

This started out as a research project for work about better locking down our workstations.

It isn't in our current plans. I do see it possibly being implemented in the next 6-9 months. Compared to some other things being considered like usbguard, static kernel configs and similar options.

Think the risk/reward for this is pretty good.

May flip the NEbraskaCERT webserver to it later this year.

Thank You, Slides and Q&A

Thanks for listening.

Slides are at my website
https://www.grothe.us/presentations

Any Questions?

Links

Linode - 2FA guide

https://www.linode.com/docs/guides/secure-ssh-access-with-2fa/

Ubuntu 2FA guide (bit older)

https://ubuntu.com/tutorials/configure-ssh-2fa#1-overview

Links

Digital Ocean guide

https://www.digitalocean.com/community/tutorials/how-to-set-up-multi-factor-authentication-for-ssh-on-ubuntu-18-04