ورود دو مرحله ای در لینوکس

روی ۳ سناریو (use case) تمرکز خواهیم کرد:

A. Password authentication over SSH
B. Local console log-on
C. Local console log-on and sudo access

Requirements:

- Google authenticator (or any other similar 2FA) app on your smartphone

تمرین ما بر روی یک سیستم با مشخصات زیر انجام می پذیرد:

- Ubuntu 22.04.4 LTS (64-Bit) \ VMware workstation

مرحله اول) نصب پکیج مورد نیاز:

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

مرحله دوم) ساخت کد 2FA برای کاربر فعلی

توجه: هر یوزر برای فعالسازی احراز هویت دومرحله ای، باید مراحل زیر را در اکانت خودش جداگانه اعمال کند.

برای شروع با دستور زیر برنامه ستاپ گوگل را فراخوانی می کنیم:

یا می توانید از command option های این ستاپ استفاده کنید. بکارگیری ستاپ بدون آپشن ها موجب اجرای interactive setup خواهد شد یعنی گزینه های حیاتی مرحله به مرحله با یوزر چک خواهند شد.

```
Sudo google-authenticator
```

```
Do you want authentication tokens to be time-based (Y)
```

به این معنی که کلیدهای تولید شده دارای اعتبار محدود از نظر زمانی می باشند. در نتیجه از بروز و معتبر بودن تایم سیستم اطمینان حاصل کنید. فاصله دار بودن زمان سیستم با دستگاه تولید کننده کد های 2FA(مثلا گوشی همراه شما) موجب رد کدهای تولید شده در سمت سیستم خواهد شد.

پس از مرحله فوق، برنامه به شما یک QR Code نمایش خواهد داد که با اپلیکیشن قابل نصب روی گوشی همراه(به طور مثال Google Authenticator) آن را اسکن و ایمپورت می کنیم. همزمان برنامه برای نهایی سازی فرآیند ستاپ کد، از ما کد تولید شده روی اپلیکیشن گوشی همراه را درخواست می کند. در صورت وارد نمودن صحیح کد، ستاپ پنج کد دسترسی اضطراری (Scratch Code) را نمایش می دهد که در جای امنی ذخیره می کنیم و سپس به مرحله بعد می رویم.

```
Do you want me to update your "/root/.google_authenticator" file? (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)
```

```
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? (More secure: N | More comfort: 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)
```

در اینجا کار ستاپ اولیه 2FA به اتمام رسیده. اما برای پیاده کردن امن اون روی اکانت جاری، نیاز به طی کردن چندین گام دیگر داریم.
همانطور که قبلا گفته شد، این راهکار در سه use-case مختلف قابل اجراست:

A. Password authentication over SSH

ابتدا باید تغییراتی در فایل pam مربوط به ssh ایجاد کنیم:

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

پس از باز شدن فایل بالا، به انتهای فایل رفته و خط زیر را اضافه کرده و ذخیره می کنیم:

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

گزینه nullok به کاربرانی که هنوز کد 2FA تولید نکرده‌اند اجازه ورود می‌دهد، در حالی که اگر کاربر مرحله ۲ را انجام داده باشد، کدها مورد نیاز هستند. این گزینه در زمانی که عملیات پیاده سازی همچنان در دست انجام است مفید خواهد بود. بعد از اینکه همه کاربران کدها را تولید کردند، می‌توانید گزینه nullok را حذف کنید تا 2FA برای همه اجباری باشد.

ویرایش SSH Daemon file

```
sudo nano /etc/ssh/sshd_config
```

عبارت ChallengeResponseAuthentication را یافته و مقدار آنرا به yes تغییر دهید.

در صورتی که عبارت فوق یافت نشد، به دنبال KbdInteractiveAuthentication باشید.

در انتها با دستور زیر سرویس SSH را ریستارت می کنیم و در لاگین بعدی می توانیم از 2FA بهره بگیریم:

```
sudo systemctl restart ssh
```

B. Local console log-on

برای بهره گیری از 2FA در لاگین های لوکال (کنسولی) به سراغ ویرایش فایل زیر رفته:

```
Sudo nano /etc/pam.d/common-session
```

و در انتهای فایل فوق عبارت زیر را اضافه می کنیم:

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

C. Local console log-on and sudo access

در این سناریو، ما به سراغ تغییری که در مرحله قبل ایجاد کردیم رفته و آنرا کامنت می کنیم؛ چرا که با انجام مراحل این بخش، علاوه بر ورود دومرحله ای در لوکال کنسول که در قبل آنرا پیاده سازی کنیم، قابلیت 2FA برای sudo access نیز فعال خواهد گشت. (یعنی دستوراتی که با کمک sudo در یوزرهای غیر root اجرا می شوند)

فایل زیر را به منظور ویرایش باز می کنیم:

```
Sudo nano /etc/pam.d/common-auth
```

و همانند مراحل پیشین، دستور زیر را در انتهای فایل اضافه می کنیم:

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

نکته امنیتی

پروفایل 2FA هر یوزر در مسیر زیر قرار دارد که حاوی scratch code های اضطراری او نیز می باشد:

```
/home/user/.google_authenticator
```

نسبت به حذف کدهای دسترسی اضطراری اقدام کنید چرا که امکان مشاهده آنها برای کاربر لاگین کرده با دسترسی کافی وجود دارد.

غیرفعالسازی آپشن 2FA

غیرفعالسازی در سطح یک یوزر

اقدام به حذف فایل

```
.google_authenticator
```

از مسیر دایرکتوری home آن کاربر با دستور زیر:

```
sudo rm /home/user/.google_authenticator
```

غیرفعالسازی برای تمامی کاربران

تغییراتی که در دوفایل sshd_config و common-auth یا common-session داده شد، به حالت اول برگردانده شوند.