



TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE TLAXIACO

PRACTICA 1.

SEGURIDAD Y VIRTUALIZACION

CARRERA:

INGENIERIA EN SISTEMAS COMPUTACIONALES

GRUPO: 7US

PRESENTA:

RUFINO MENDOZA VAZQUEZ - 21620198

ANA MICHEL LEÓN LEÓN - 21620112

ROSA SALAZAR DOROTEO - 18620216

FERNANDA RUIZ HERAS - 21520151

DOCENTE

ING. EDWARD OSORIO SALINA

Tlaxiaco, Oax., 30 de Agosto del 2024.



"Educación, ciencia y tecnología, progreso día con día"®

INDICE

INTRODUCCION	3
PRACTICA 1 - CONTRASEÑAS Y CERTIFICADOS	3
Ilustración 1: Librería de import re	3
Ilustración 2: Función para devolver cadena	3
Ilustración 3: Verificación de la contraseña	4
Ilustración 4: Solicitar contraseña	4
Ilustración 5: Ingresar contraseña	4
Ilustración 6: Contraseña insegura	5
Ilustración 7: Contraseña segura	5
Ilustración 8: Librerías del segundo programa	5
Ilustración 9: Generación de contraseña segura	5
Ilustración 10: Longitud de 8 a 10 caracteres	6
Ilustración 11: Verificar que la contraseña cumpla con lo definido	6
Ilustración 12: Se define si genera una contraseña segura	6
Ilustración 13: Ejecución del código donde nos genera contraseñas aleatorias	7
Ilustración 14: Ejecución de código ssh-keygen -t ed25519 -C tu_email@ejemplo.com	7
Ilustración 15: Código cat ~/.ssh/id_ed25519.pub	8
Ilustración 16: Setting y SSH and GPG Keys	8
Ilustración 17: Nombre de la clave y código	8
Ilustración 18: autenticación de GitHub	9
Ilustración 19: Link de un proyecto con SSH	9
Ilustración 20: Clonar el proyecto	9
Ilustración 21: OpenSSI	10
Ilustración 22: Descargar archivo	10
Ilustración 23: Version de openss	10
Ilustración 24: Clave	10
CONCLUSION	11

INTRODUCCION

En esta guía, se aborda cómo validar contraseñas seguras utilizando un programa en Python y gestionar repositorios en GitHub con claves SSH. El programa en Python verifica si una contraseña cumple con criterios de seguridad específicos, como longitud mínima, variedad de caracteres (mayúsculas, minúsculas, números y símbolos), ausencia de espacios y repetición de caracteres consecutivos. También se muestra cómo generar contraseñas seguras automáticamente. Además, se explica el proceso para configurar una clave SSH: generar las claves, añadirlas a GitHub y clonar un repositorio usando la URL SSH. Estos pasos garantizan una gestión segura y eficiente tanto de contraseñas como de proyectos en GitHub.

PRACTICA 1 - CONTRASEÑAS Y CERTIFICADOS

1. Crea un programa en Python que permita al usuario ingresar una contraseña y que valide si la contraseña es segura o no. Una contraseña segura debe cumplir con los siguientes criterios basados en las recomendaciones de Google:

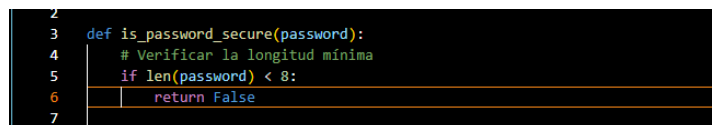
- Primero para comenzar tenemos que tener instalado lo que en Python o algún otro lenguaje de programación. Posteriormente empezamos con lo que es la importación del módulo `re` el cual se utiliza para trabajar con expresiones regulares.



```
Ejercicio1.py x Ejercicio2.py 1
Ejercicio1.py > is_password_secure
1 import re
2
```

Ilustración 1: Librería de import re

- La función `is_password_secure(password)` esto su función es recibir una cadena y devolver un valor booleano falso o verdadero que nos indica que la contraseña cumpla con los criterios de seguridad definido y también nos da un mínimo de caracteres.



```
2
3 def is_password_secure(password):
4     # Verificar la longitud mínima
5     if len(password) < 8:
6         return False
7
```

Ilustración 2: Función para devolver cadena

- En la siguiente se muestra la verificación que la contraseña contenga al menos una letra en mayúscula, al menos un dígito, un carácter especial y si no es así se retorna con falso.

```

7
8 # Verificar la presencia de al menos una letra mayúscula
9 if not re.search(r'[A-Z]', password):
10 | return False
11
12 # Verificar la presencia de al menos una letra minúscula
13 if not re.search(r'[a-z]', password):
14 | return False
15
16 # Verificar la presencia de al menos un número
17 if not re.search(r'\d', password):
18 | return False
19
20 # Verificar la presencia de al menos un carácter especial
21 if not re.search(r'[!@#$%^&*()\-_+=\[\]{};:~'\",.</?<`]', password):
22 | return False
23
24 # Verificar la ausencia de espacios en blanco
25 if ' ' in password:
26 | return False
27
28 # Verificar que no haya más de 2 caracteres iguales consecutivos
29 for i in range(len(password) - 2):
30 | if password[i] == password[i + 1] == password[i + 2]:
31 | return False
32
33

```

Ilustración 3: Verificación de la contraseña

- En caso de que la contraseña cumpla con todos los criterios se retorna verdadero, la función de main() gestiona la interacción con el usuario, la siguiente línea de código nos pide ingresar una contraseña y la línea de comando is_password_secure su función es realizar la validación de la contraseña ingresada dependiendo si está bien o mal se imprimirá un mensaje indicando si es seguro o no.
- Y por último se utiliza el bloque de código de la función main() que se ejecutara cuando el script se ejecuta directamente, no cuando se importe como un módulo en otro script.

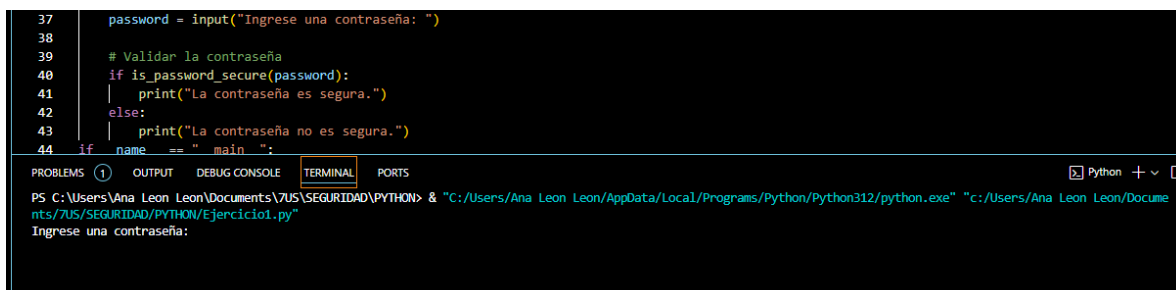
```

32
33 | return True
34
35 def main():
36 | # Solicitar la contraseña al usuario
37 | password = input("Ingrese una contraseña: ")
38
39 | # Validar la contraseña
40 | if is_password_secure(password):
41 | | print("La contraseña es segura.")
42 | else:
43 | | print("La contraseña no es segura.")
44 | if __name__ == "__main__":
45 | | main()
46

```

Ilustración 4: Solicitar contraseña

- Para finalizar ejecutaremos el código y nos pedirá que ingresemos una contraseña.



```

37 | password = input("Ingrese una contraseña: ")
38
39 | # Validar la contraseña
40 | if is_password_secure(password):
41 | | print("La contraseña es segura.")
42 | else:
43 | | print("La contraseña no es segura.")
44 | if __name__ == "__main__":
45 | | main()

```

PROBLEMS ① OUTPUT DEBUG CONSOLE **TERMINAL** PORTS

PS C:\Users\Ana Leon Leon\Documents\7US\SEGURIDAD\PYTHON> & "C:/Users/Ana Leon Leon/AppData/Local/Programs/Python/Python312/python.exe" "c:/Users/Ana Leon Leon/Docu-
nts/7US/SEGURIDAD/PYTHON/Ejercicio1.py"
Ingrese una contraseña:

Ilustración 5: Ingresar contraseña

- Aquí escribimos una contraseña y como no cumple con los requisitos especificados nos mandara el mensaje de que la contraseña no es segura.

```
PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS Python + v
PS C:\Users\Ana Leon Leon\Documents\7US\SEGURIDAD\PYTHON> "C:/Users/Ana Leon Leon/AppData/Local/Programs/Python/Python312/python.exe" "c:/Users/Ana Leon Leon/Docume
nts/7US/SEGURIDAD/PYTHON/Ejercicio1.py"
Ingrese una contraseña: AnaLeon
La contraseña no es segura.
```

Ilustración 6: Contraseña insegura

- Caso contrario aquí ya que al agregar una contraseña con los atributos especificados que son mayúsculas, minúsculas, números y caracteres especiales y ser más de 8 caracteres nos lo marca como contraseña segura.

```
PS C:\Users\Ana Leon Leon\Documents\7US\SEGURIDAD\PYTHON> "C:/Users/Ana Leon Leon/AppData/Local/Programs/Python/Python312/python.exe" "c:/Users/Ana Leon Leon/Docume
nts/7US/SEGURIDAD/PYTHON/Ejercicio1.py"
Ingrese una contraseña: AnaLeon280503#.
La contraseña es segura.
PS C:\Users\Ana Leon Leon\Documents\7US\SEGURIDAD\PYTHON>
```

Ilustración 7: Contraseña segura

2. Crea un programa que me recomiende una contraseña segura. La contraseña debe cumplir con los criterios de la instrucción anterior.
- Se importan las librerías que son el random, string y os el cual random se utiliza para generar números aleatorios y seleccionar características aleatorias. El string nos proporciona constantes para caracteres ASCII como letras, mayúsculas, minúsculas y dígitos y él os es el que interactúa con el sistema de archivos en particular para verificar la existencia de archivos.

```
Ejercicio2.py > generar_contraseña_segura
1 import random
2 import string
3 import os
4
```

Ilustración 8: Librerías del segundo programa

- Se genera una contraseña segura con al menos una letra mayúscula, una minúscula, un número y un carácter especial, que tenga una longitud de 10 caracteres con caracteres aleatorios.

```
4
5 def generar_contraseña_segura():
6     mayusculas = string.ascii_uppercase
7     minusculas = string.ascii_lowercase
8     numeros = string.digits
9     especiales = "!@#%&*()-_+[]{}|\\;:'\">/?~ "
10
11     contraseña = [
12         random.choice(mayusculas),
13         random.choice(minusculas),
14         random.choice(numeros),
15         random.choice(especiales)
16     ]
17
18     while len(contraseña) < 10:
19         contraseña.append(random.choice(mayusculas + minusculas + numeros + especiales))
20
21     random.shuffle(contraseña)
22
23     contraseña = ''.join(contraseña)
24
25     return contraseña
26
```

Ilustración 9: Generación de contraseña segura

- Se genera una contraseña insegura con una longitud de 8 a 10, utilizando una combinación de caracteres.

```

def generar_contraseña_insegura():
    longitud = random.randint(8, 10)
    caracteres = string.ascii_letters + string.digits
    contraseña = ''.join(random.choice(caracteres) for _ in range(longitud))

    if random.choice([True, False]):
        | contraseña += random.choice("!@#%&*()-_+[]{}|\\;:'\".,<.>/?~")

    return contraseña

```

Ilustración 10: Longitud de 8 a 10 caracteres

- Se verifica si la contraseña es segura con los criterios de tener al menos 10 caracteres, tener mayúsculas, minúsculas, números y caracteres especiales, no tener espacio en blancos y no tener más de dos caracteres iguales consecutivos.

```

36
37 def es_segura(contraseña):
38     if len(contraseña) < 10:
39         return False
40
41     tiene_mayuscula = any(c.isupper() for c in contraseña)
42     tiene_minuscula = any(c.islower() for c in contraseña)
43     tiene_numero = any(c.isdigit() for c in contraseña)
44     tiene_especial = any(c in "!@#%&*()-_+[]{}|\\;:'\".,<.>/?~" for c in contraseña)
45
46     for i in range(len(contraseña) - 2):
47         if contraseña[i] == contraseña[i+1] == contraseña[i+2]:
48             return False
49
50     if ' ' in contraseña:
51         return False
52
53     if tiene_mayuscula and tiene_minuscula and tiene_numero and tiene_especial:
54         return True
55
56     return False
57
58 # Verificar si el archivo existe para leer el estado
59 estado_archivo = "estado_alternar.txt"
60 if os.path.exists(estado_archivo):
61     with open(estado_archivo, "r") as file:
62         alternar = file.read().strip() == "True"
63 else:
64     alternar = True # Por defecto, empezamos con una contraseña segura
65

```

Ilustración 11: Verificar que la contraseña cumpla con lo definido

- Lee el estado falso o verdadero desde el archivo para decidir si se debe generar una contraseña segura o insegura alternando si la próxima contraseña es segura o insegura.

```

65
66 # Generar contraseña según el estado actual
67 if alternar:
68     contraseña = generar_contraseña_segura()
69 else:
70     contraseña = generar_contraseña_insegura()
71
72 # Guardar el estado alternado para la próxima ejecución
73 with open(estado_archivo, "w") as file:
74     file.write(str(not alternar))
75
76 print(f"Contraseña generada: {contraseña}")
77 if es_segura(contraseña):
78     print("La contraseña es segura.")
79 else:
80     print("La contraseña no es segura.")
81

```

Ilustración 12: Se define si genera una contraseña segura

- Para la comprobación nos muestra que nos arroja una contraseña segura y una contraseña insegura dado por si mismo.

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\Ana Leon Leon\Documents\7US\SEGURIDAD\PYTHON> & "C:/Users/Ana Leon Leon/AppData/Local/Programs/Python/Python312/python.exe" "c:/Users/Ana Leon Leon/Docume
nts/7US/SEGURIDAD/PYTHON/Ejercicio2.py"
Contraseña generada: 4Zu7t|
La contraseña no es segura.
PS C:\Users\Ana Leon Leon\Documents\7US\SEGURIDAD\PYTHON> & "C:/Users/Ana Leon Leon/AppData/Local/Programs/Python/Python312/python.exe" "c:/Users/Ana Leon Leon/Docume
nts/7US/SEGURIDAD/PYTHON/Ejercicio2.py"
Contraseña generada: ^ipwB8U(!
La contraseña es segura.
PS C:\Users\Ana Leon Leon\Documents\7US\SEGURIDAD\PYTHON>
```

Ilustración 13: Ejecución del código donde nos genera contraseñas aleatorias

3. Crea un certificado SSH, clave pública y clave privada, añade el certificado SSH a tu cuenta de GitHub y realiza un git clone de un repositorio nuevo utilizando la ruta SSH del repositorio.
- Primero necesitamos generar una clave SSH en el cual se utiliza el siguiente comando en la terminal ya sea de git Bash, la terminal de Windows o power Shell ejecutando el siguiente código `ssh-keygen -t ed25519 -C tu_email@ejemplo.com` aquí en este código solo cambiaremos el correo por el que tenemos agregado el GitHub.
- Posteriormente nos pedirá una ubicación para guardar la clave en el cual presionaremos enter para aceptar la ubicación predeterminada, en donde también nos pedirá una frase de paso que será opcional para escribirla o no.

```
C:\Users\DELL>ssh-keygen -t rsa -b 4096 -C "your_email@example.com"
Generating public/private rsa key pair.
Enter file in which to save the key (C:\Users\DELL\.ssh\id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in C:\Users\DELL\.ssh\id_rsa
Your public key has been saved in C:\Users\DELL\.ssh\id_rsa.pub
The key fingerprint is:
SHA256:tmcg2vV29jrnDjYHnq+moaUmW0tVKNNKqMH5o3PC7dhY yO
The key's randomart image is:
+----[RSA 4096]-----+
|
|  o S . .
| . oE* . = . o .
| . = . o+ . * = = . +
| +o . . + o = . * . =o+ .
| o+o+= . .oo oB+ . .
+----[SHA256]-----+

C:\Users\DELL>notepad C:\Users\DELL\.ssh\id_rsa.pub

C:\Users\DELL>
```

```
ssh-rsa
AAAAB3NzaC1yc2EAAA
Oml2KLeNZS5xbGKd+GL
ufmgK1D09hv4MGFZg
2GeAW4wp0zw3xDzCZ
9caspzC0eM+1oTJ87CBWAZS2DL869rq1
+vH6yKKJ+X06ChRLnM98/h/5t/KHoELjeNYpR/jybMj0Ck41cWa5W2MgYmr9TW8EpFaUJjeC4K
SaLiP2hZR9/yiMhwzgifpJiAKnAWgGG38nPTDxaVLW0jcugnAz1ppfVxQHfPzYsEugXs0PyF4/
godTAE8hS8VwFFqMHVviB0xNUREvZU0/kynhWGRhpq7G6qf1cuYI7VUyMKATbIEUsnJ6Wgp8Qd
KEAZjUGYahn/aL8c5gm1GSP30CTTnxsNQ1Lz74U74W11s+H4cE+wugkXKjsrXtsxPw6aurrmGN
BZYXjUrnXj4bNivvygUiq7UWxYUuhokrocenQPzt5sg7p6vJXcYZLSUVEHfBAnMpg2sDwVSpooD
nTeMbTGJALjRnjRi07SI2bpKEgc3JAQ== your_email@example.com
```

Ilustración 14: Ejecución de código `ssh-keygen -t ed25519 -C tu_email@ejemplo.com`

- El siguiente paso será agregar el siguiente código `cat ~/.ssh/id_ed25519.pub` en cual nos dará la clave SSH para agregarlo a la cuenta de github

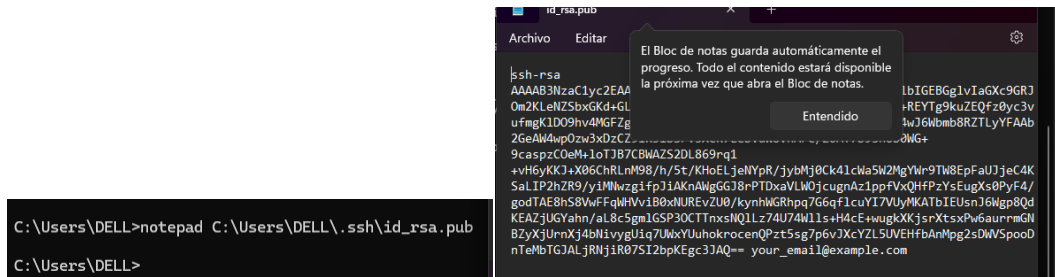


Ilustración 15: Código cat ~/.ssh/id_ed25519.pub

- Ya realizado el paso anterior vamos a GitHub y entramos a nuestras respectivas cuentas, seleccionamos la opción de setting y de ahí a SSH and GPG Keys.

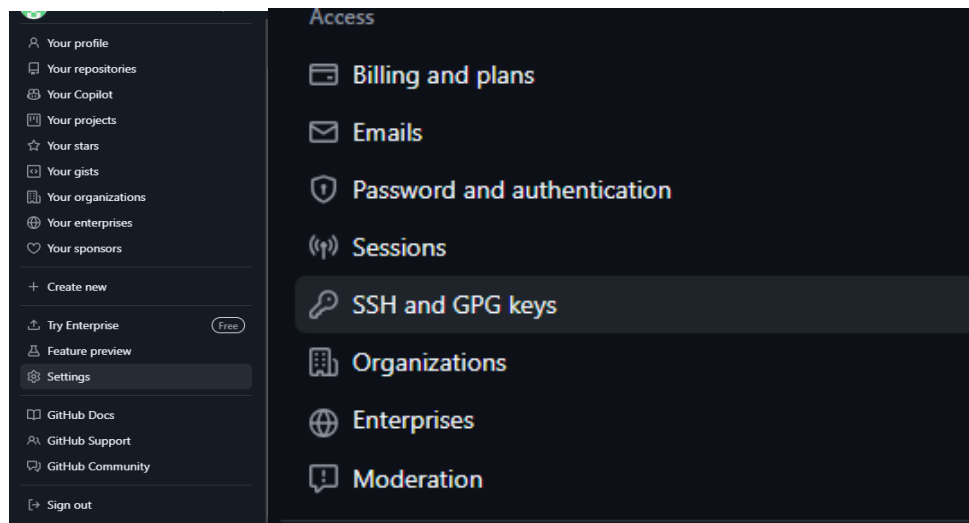


Ilustración 16: Setting y SSH and GPG Keys

- A continuación, seleccionamos la opción de New SSH Key en donde le daremos un nombre y pegaremos el código dado anteriormente.

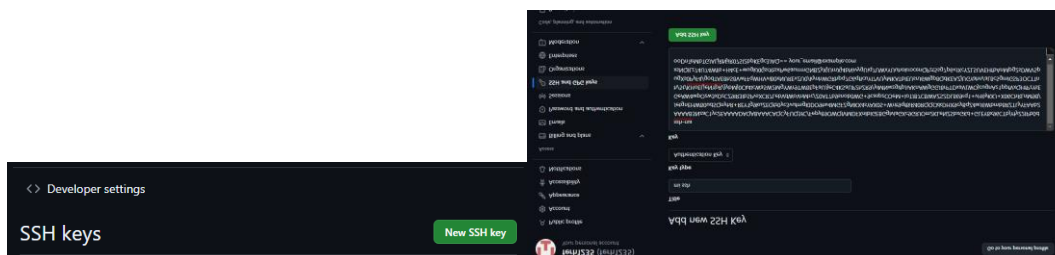


Ilustración 17: Nombre de la clave y código

- Este código que se muestra a continuación `ssh -T git@github.com` ejecutara que fue exitosa la autenticación con GitHub y que está listo para trabajar con repositorios usando SSH.


```
DELL@DESKTOP-MP0793M MINGW64 ~ (master)
$ ssh -T git@github.com
Hi ferh1235! You've successfully authenticated, but GitHub does not provide shell access.

DELL@DESKTOP-MP0793M MINGW64 ~ (master)
$ |
```

Ilustración 18: autenticación de GitHub

- En la siguiente ilustración nos muestra una URL SSH del repositorio desde GitHub en donde escogeremos un proyecto que queremos clonar.

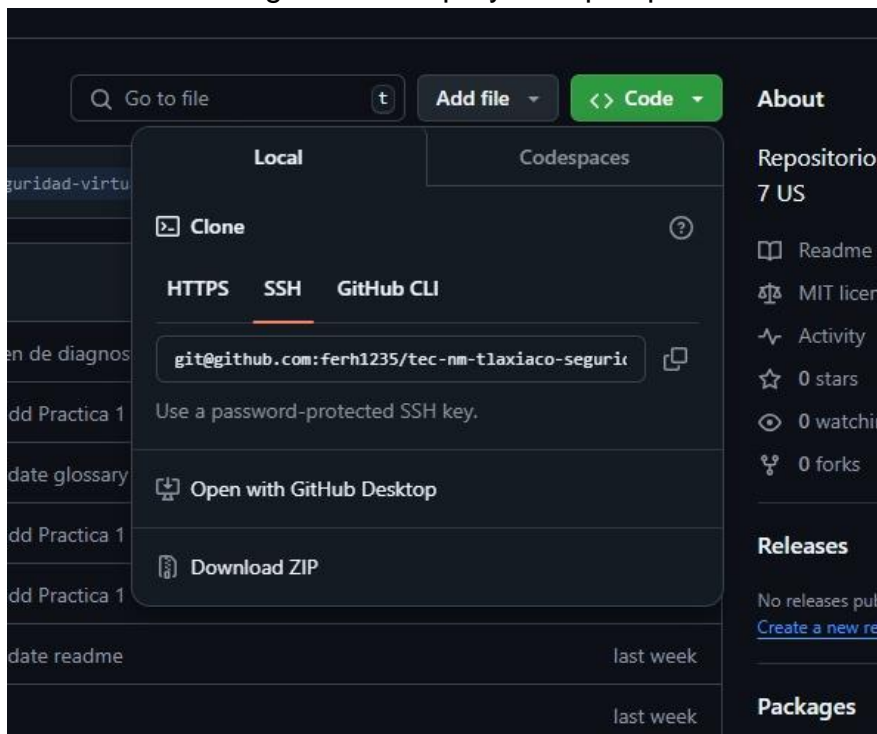


Ilustración 19: Link de un proyecto con SSH

- Aquí se muestra ya como se terminó el proceso de clonación del proyecto y guardado el proyecto ya en una carpeta especificada y se tendrá una copia del repositorio y así estaríamos concluyendo con la práctica 3.

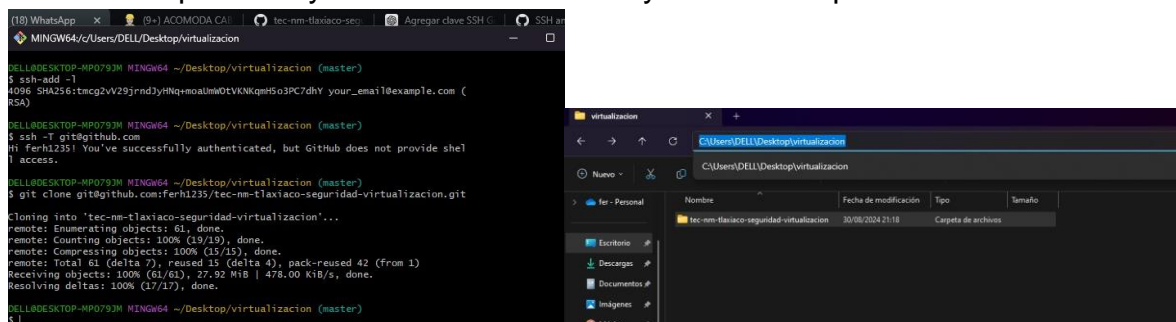


Ilustración 20: Clonar el proyecto

4. Crea un certificado SSL autofirmado con una validez de 365 días y añádelo a un servidor web local. Realiza una petición GET al servidor web local utilizando curl y muestra el certificado SSL.
- Crear el Certificado SSL en Windows
 - Descarga e instala OpenSSL desde el sitio web de OpenSSL. Elige la versión adecuada para tu sistema (Win32/Win64). OpenSSL no es una aplicación gráfica, sino una herramienta de línea de comandos

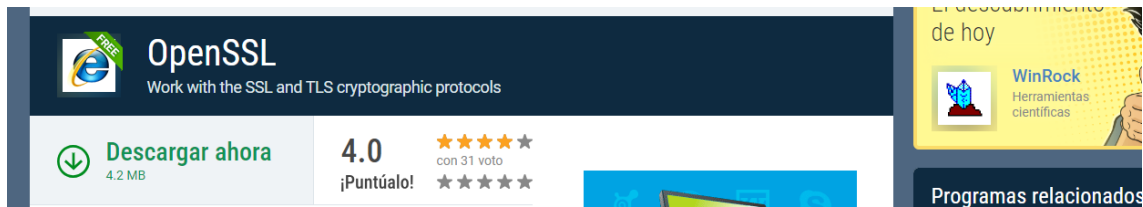


Ilustración 21: OpenSSL

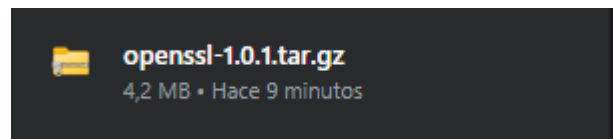


Ilustración 22: Descargar archivo

- OpenSSL 1.1.1s instalado en el sistema.

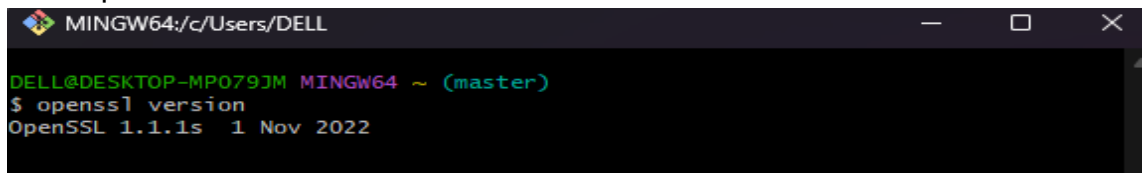


Ilustración 23: Version de openss

- Generar una Clave Privada con el siguiente comando:
- openssl genpkey -algorithm RSA -out server.key -aes256



Ilustración 24: Clave

CONCLUSION

En conclusión, en esta práctica, hemos realizado puntos importantes como es la practica uno donde se abordó la seguridad, con la creación de un programa en Python que permite a los usuarios validar la seguridad de sus contraseñas. Esto es importante, ya que una contraseña robusta es la primera línea de defensa contra accesos no autorizados. Además, implementamos un generador de contraseñas seguras, lo cual no solo facilita la creación de contraseñas complejas, sino que también garantiza que cumplan con las mejores prácticas recomendadas. El siguiente paso fue la generación y configuración de un par de claves SSH, una habilidad esencial para trabajar con repositorios Git de manera segura, especialmente en plataformas como GitHub. Este proceso asegura que las conexiones entre el desarrollador y el repositorio sean autenticadas, minimizando riesgos de seguridad. Aunque debo recalcar que no completamos el punto número cuatro ya que se nos dificultó al momento de realizarla ya que estaba relacionada con la creación de un certificado SSL y su implementación en un servidor web local, abordamos su importancia. Un certificado SSL permite establecer conexiones seguras entre clientes y servidores, protegiendo la integridad y confidencialidad de los datos.