



Universidad Nacional Autónoma de México

Facultad de Ciencias

CRIPTOGRAFÍA Y SEGURIDAD

Práctica 6: We Will Ransom You

FECHA DE ENTREGA: 26/10/2023

Equipo:

Criptonianos

Acosta Arzate Rubén - 317205776

Bernal Marquez Erick - 317042522

Deloya Andrade Ana Valeria - 317277582

Marco Antonio Rivera Silva - 318183583



1. Introducción

Esta práctica es acerca de lo que se conoce en criptografía en seguridad como *malware* o *software malicioso*, el cual es utilizado con el fin de dañar, infectar o recopilar información de dispositivos sin contar con el consentimiento de la víctima. Siendo en la mayoría de las veces realizado con el objetivo de chantajear a la víctima para así obtener ganancias financieras.

Existen varios tipos de malwares, cada uno con una funcionalidad distinta pero todos comparten el objetivo de comprometer la seguridad y la información de los dispositivos víctimas de éstos. Durante la realización de la práctica aprenderemos acerca del desarrollo de dos tipos de malware, con ello conoceremos cómo es que funcionan, así como también sus usos, siendo éstos *ransomware* y *spyware*.

El objetivo **no** es incentivar el uso de malwares, sino saber que existen y cómo funcionan, todo esto bajo la ética universitaria.

Por mi raza hablará el espíritu.



2. Desarrollo

Utilizamos máquinas virtuales para no comprometer nuestros dispositivos e información personal durante el desarrollo de los *malwares* solicitados para la práctica. Para realizar las pruebas del *spyware* utilizamos una máquina virtual con sistema Debian 11 y para las pruebas del *ransomware* utilizamos una máquina virtual con sistema Windows 10. Cabe destacar que utilizamos *chatGPT* para el desarrollo de los *malware*, sin embargo debemos saber cómo y qué escribir para que arroje una salida que sea de utilidad ya que escribir cosas como “ayúdame a hacer un ransomware” no funciona pues nos dice cosas como “No puedo ayudarte a hacer programar maliciosos ya que es ilegal”. Además cabe mencionar que en ocasiones pide demostrar al usuario que se trata de una persona real (captcha) o bien limitar el uso de mensajes durante periodos de tiempo.

A continuación explicamos cómo se llevó a cabo el desarrollo de ambos malwares:

2.1. Ransomware

Para la creación de este malware decidimos usar el lenguaje de programación python, debido a que tiene una serie de bibliotecas instaladas las cuales nos permitieron crear el ransomware con mayor facilidad.

Debido a que se especifica que sea un archivo *.exe* usamos la biblioteca *pyinstaller* para generar un archivo *.exe* que contenga todo lo necesario para la ejecución.

Aclaramos que para la ejecución del ransomware se tomó en cuenta que Windows defender estuviera apagado, además de que se ejecuta con permisos de administrador, el *script* puede tardar varios minutos en ejecutarse ya que compila bibliotecas desde caché. Por último, en el mismo directorio donde esté el *.exe* nos va a dejar la llave AES cifrada.

Además, para lograr realizar el script le pedimos ayuda a *chatGPT*, el cual nos proporcionó ejemplos acerca de cómo encontrar el directorio *system32*.



El código del ransomware no fue exitoso al primer intento, pues todavía debíamos hacer mejoras tales como guardar la clave AES, eliminar bien los archivos, mostrar la imagen del rescate, etc.

Algunas complicaciones eran más fáciles de resolver que otras, tuvimos que leer varias preguntas en *stackoverflow* para darnos una idea de cómo hacer lo que queríamos, sobre todo para las llaves AES y encriptación RSA. Por lo cual se puede decir que de cierta manera debíamos tener conocimientos en programación para entender mejor lo que estábamos haciendo. *[Las páginas consultadas se encuentran en las referencias.]*

Sin embargo, cabe destacar dos principales complicaciones, la primera es que para hacer pruebas debíamos de ejecutar una sola vez ya que había cosas que por cada vez que ejecutábamos se sobrescribían, tal como la llave RSA, pues generaba una llave diferente cada vez que ejecutábamos el *script*, teníamos que volver a crear archivos “importantes” y repetir el proceso de encriptación.

Otro gran reto fue convertir el *script* en un ejecutable *.exe*, ya que algunas bibliotecas no funcionaban en Windows, debían instalarse o pasar parámetros un tanto raros. De nuevo, tuvimos que modificar el código para su correcto funcionamiento consultando distintas fuentes.

El código ya arreglado es el siguiente:

Por razones de seguridad no pondremos el código del ransomware en este apartado, sin embargo cabe aclarar que el desarrollo del mismo se logró con éxito

A continuación se muestran capturas de cómo funciona el ejecutable `.exe`

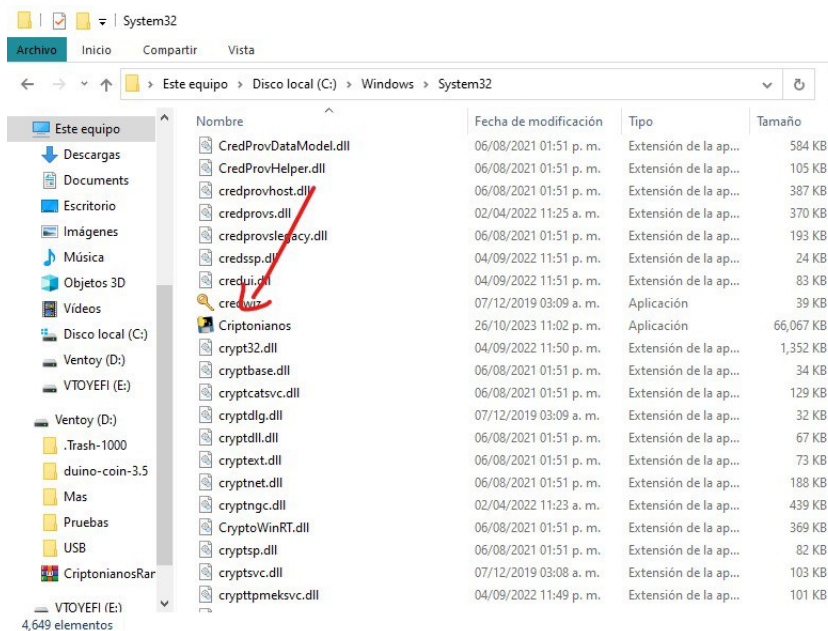


Figura 1: Ransomware

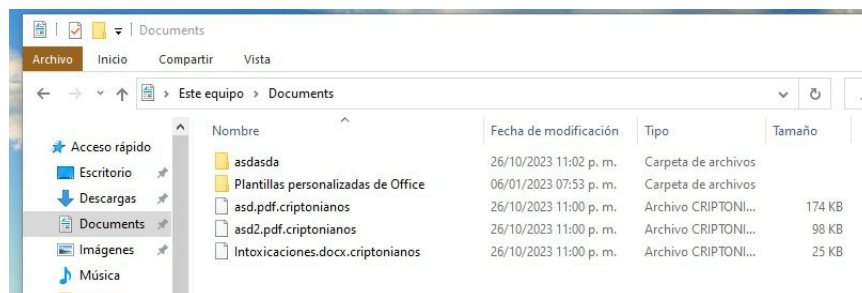


Figura 2: Archivos cifrados



Figura 3: Se le pide rescate a la víctima por sus archivos

2.2. Spyware

Para la ejecución del *spyware* suponemos que el usuario pertenece al grupo de *sudoers*, esto para que pueda instalar cualquier programa y acceder a cualquier comando que requiera uso de la contraseña de superusuario. Además de se le darán permisos de ejecución al archivo con `chmod +x spy.sh`. Para ejecutar el archivo basta con poner `./spy.sh` en la terminal.

Para lograr que *chatGPT* nos diera el código del *spyware* en *bash* debíamos escribir la solicitud de manera ingeniosa. Lo primero fue darle un contexto muy diferente a lo que realmente estabamos haciendo, pues empezamos escribiendo que teníamos una empresa donde todos eran programadores y debíamos monitorear las actividades de cada persona por lo cual queremos saber información del kernel, de la arquitectura, del sistema operativo, etc. Además, al tratarse de supuestas bitácoras, queríamos guardar la información en un archivo de texto *.log*. La información se guarda en una carpeta llamada *System monitoring*

Por razones de seguridad no pondremos el código del spyware en este apartado, sin embargo cabe aclarar que el desarrollo del mismo se logró con éxito

El resultado fue casi exitoso, el detalle era que no podíamos acceder a la carpeta */etc/shadow*, pues esta contiene información sensible e importante tal como las contraseñas de el usuario, por lo que al momento de ejecutar el script mostraba un mensaje diciendo “acceso no permitido”.

La solución parecía obvia, cambiar los permisos de la carpeta y ejecutar el archivo como superusuario *sudo*, sin embargo esto no funcionó. Por más intentos que hiciéramos probando con distintas formas de extraer información de la carpeta, simplemente no logramos obtener la información.

En su lugar optamos por obtener más detalles acerca del dispositivo de la víctima tal como especificaciones del hardware, discos, configuración de red, etc. Además, obtener contraseñas es algo que podemos realizar mediante ataques por diccionario.

También tuvimos problemas al momento de enviar la información al atacante, intentamos con *scp* y *netcat* pero pedían contraseñas o claves SSH para enviar la información o el programa se quedaba “colgado”, aun con la ip pública para conectar con la computadora del atacante :(También podíamos “harcodear” nuestro usuario y contraseña al ponerlo en el mismo *spyware* pero sería dar mucha información acerca de nosotros, el atacante.

Otra opción era tener algún servidor con algún dominio *midominio.com*, el inconveniente era que daba más problemas que soluciones puesto que debíamos tener un dominio (aunque en realidad basta con la ip), establecer un servidor, hacer configuraciones del lado del servidor (nuestra computadora como atacantes) y hacer configuraciones del lado del cliente (la computadora de la víctima).

Fuera de estos problemas, el resto de la información recopilada a través del *malware* fue exitoso. Por motivos de seguridad las muestras de los *malwares* se enviarán por otro medio.

```

ruben@ruben-IdeaCentre-AIO-3-22ADA05:~/Documentos/CriptonianosSpyware$ \. archivo.sh
AcostaArzateRuben.odt      Guia_de_trabajo_git-Github.pdf      Redes
'Analisis de algoritmos'    helado.odg                          redtiger.txt
'Archivo HTML.html'        ingenieria_de_Software              Respuestas
'comando docker.txt'      'Inteligencia artificial'           Sistemas_Operativos
Compiladores               Lenguajes                           Tecnologias
Complejidad                'machine Learning'                 'Time_bug_BurstDebugInformation_DoNotShip'
computoDistribuido         Main.class                          'Time_bug_Data'
Concurrente                Main.java                          'Time_bug_x86_64'
Cripto                     modelado                           'token git.odt'
CriptonianosSpyware        Moviles                           token_git.txt
'Diseño de interfaces'     practica01-intro-to-c-connect4-el-refugio-201  UnityPlayer.so
'Estructuras Avanzadas'    practicasArqui                     Videojuegos
FBD                        practicasICC
Front-end                  pruebas.c

Hola, te estamos jakiando
Registro completado en /home/ruben/system_monitoring/system_info_2023-10-26.log
ruben@ruben-IdeaCentre-AIO-3-22ADA05:~/Documentos$

```

Figura 4: Ejecutando el spyware

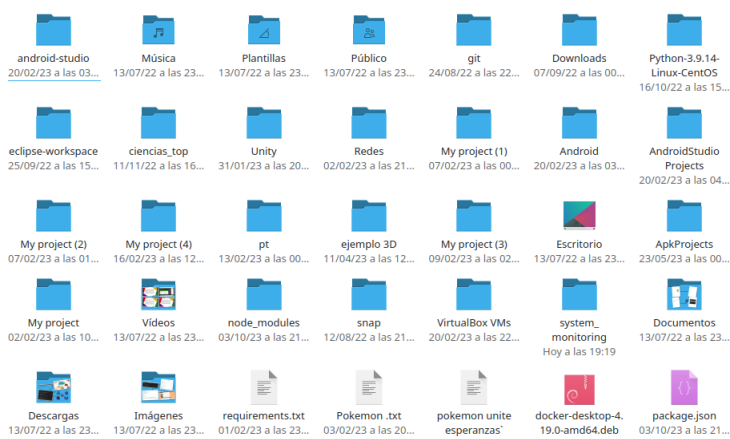
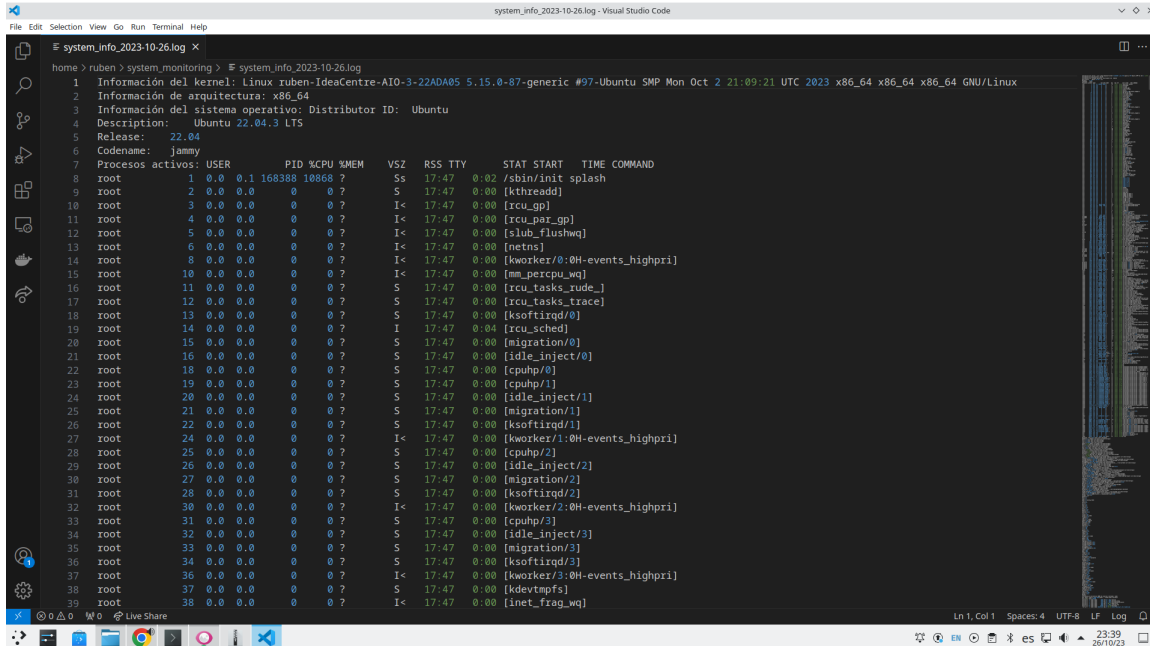


Figura 5: Carpeta donde se implantó la información robada



```

system_info_2023-10-26.log - Visual Studio Code
File Edit Selection View Go Run Terminal Help
# system_info_2023-10-26.log X
home > ruben > system_monitoring > # system_info_2023-10-26.log
1 Información del kernel: Linux ruben-IdeaCentre-AIO-3-22ADA05 5.15.0-87-generic #97-Ubuntu SMP Mon Oct 2 21:09:21 UTC 2023 x86_64 x86_64 x86_64 GNU/Linux
2 Información de arquitectura: x86_64
3 Información del sistema operativo: Distributor ID: Ubuntu
4 Descripción: Ubuntu 22.04.3 LTS
5 Release: 22.04
6 Codename: jammy
7 Procesos activos: USER PID %CPU %MEM VSZ RSS TTY STAT START TIME COMMAND
8 root 1 0.0 0.1 168388 10868 ? Ss 17:47 0:02 /sbin/init splash
9 root 2 0.0 0.0 0 0 ? S 17:47 0:00 [kthreadd]
10 root 3 0.0 0.0 0 0 ? I< 17:47 0:00 [rcu_gp]
11 root 4 0.0 0.0 0 0 ? I< 17:47 0:00 [rcu_per_gp]
12 root 5 0.0 0.0 0 0 ? I< 17:47 0:00 [slub_flushwq]
13 root 6 0.0 0.0 0 0 ? I< 17:47 0:00 [netns]
14 root 8 0.0 0.0 0 0 ? I< 17:47 0:00 [kworker/0:0H-events_highpri]
15 root 10 0.0 0.0 0 0 ? I< 17:47 0:00 [mm_percpu_wq]
16 root 11 0.0 0.0 0 0 ? S 17:47 0:00 [rcu_tasks_xude_]
17 root 12 0.0 0.0 0 0 ? S 17:47 0:00 [rcu_tasks_trace]
18 root 13 0.0 0.0 0 0 ? S 17:47 0:00 [ksoftirqd/0]
19 root 14 0.0 0.0 0 0 ? I 17:47 0:04 [rcu_sched]
20 root 15 0.0 0.0 0 0 ? S 17:47 0:00 [migration/0]
21 root 16 0.0 0.0 0 0 ? S 17:47 0:00 [idle_inject/0]
22 root 18 0.0 0.0 0 0 ? S 17:47 0:00 [cpuhp/0]
23 root 19 0.0 0.0 0 0 ? S 17:47 0:00 [cpuhp/1]
24 root 20 0.0 0.0 0 0 ? S 17:47 0:00 [idle_inject/1]
25 root 21 0.0 0.0 0 0 ? S 17:47 0:00 [migration/1]
26 root 22 0.0 0.0 0 0 ? S 17:47 0:00 [ksoftirqd/1]
27 root 24 0.0 0.0 0 0 ? I< 17:47 0:00 [kworker/1:0H-events_highpri]
28 root 25 0.0 0.0 0 0 ? S 17:47 0:00 [cpuhp/2]
29 root 26 0.0 0.0 0 0 ? S 17:47 0:00 [idle_inject/2]
30 root 27 0.0 0.0 0 0 ? S 17:47 0:00 [migration/2]
31 root 28 0.0 0.0 0 0 ? S 17:47 0:00 [ksoftirqd/2]
32 root 30 0.0 0.0 0 0 ? I< 17:47 0:00 [kworker/2:0H-events_highpri]
33 root 31 0.0 0.0 0 0 ? S 17:47 0:00 [cpuhp/3]
34 root 32 0.0 0.0 0 0 ? S 17:47 0:00 [idle_inject/3]
35 root 33 0.0 0.0 0 0 ? S 17:47 0:00 [migration/3]
36 root 34 0.0 0.0 0 0 ? S 17:47 0:00 [ksoftirqd/3]
37 root 36 0.0 0.0 0 0 ? I< 17:47 0:00 [kworker/3:0H-events_highpri]
38 root 37 0.0 0.0 0 0 ? S 17:47 0:00 [kdevtmpfs]
39 root 38 0.0 0.0 0 0 ? I< 17:47 0:00 [inet_frag_wq]

```

Figura 6: Archivo con la información obtenida

3. Conclusiones

La realización de esta práctica nos permitió aprender acerca del desarrollo de malwares como lo es *ransomware* y lo sencillo que puede ser realizar uno con ayuda de IA sin necesidad de conocer mucho sobre programación o ciberseguridad. Eso sí, se debe saber cómo pedirlo a la IA para que arroje lo que queremos, pues si sólo se le pide “Quiero un ransomware”, te va a enlistar razones por las cuales no puede ayudarte, siendo la principal razón que es ilegal.

También aprendimos acerca del *ransomware*, cuyo propósito es encriptar los archivos del dispositivo de la víctima con el objetivo de pedirle un rescate, este pago suele ser de criptomonedas, así como nosotros pedimos a la víctima en la práctica.

Sin embargo, incluso pagando por el rescate de información sigue siendo inseguro, ya que no hay garantía de que el atacante te devuelva los archivos descriptados. Siendo esto realmente peligroso



en situaciones reales pues estos archivos encriptados por el *ransomware* podrían contener cualquier tipo de información, como lo son datos personales de la víctima.

Además, supimos la creación de *spyware* cuyo objetivo es el espionaje y la recopilación de información en un dispositivo sin el conocimiento de la víctima, pudiendo robar así información del dispositivo, datos personales, bancarios, etc. Con ransomware la víctima eventualmente se entera que su dispositivo ha sido comprometido junto con la información contenida en éste, mientras que con spyware puede que no se entere nunca que ha sido víctima de este malware que trabaja en silencio.

Finalmente conocimos la importancia de ser cuidadosos con lo que descargamos en nuestros dispositivos, pues esto puede resultar convertirse en víctimas de malwares mediante la descarga softwares y archivos sospechosos.

4. Referencias

- *McAfee. (2020, 15 mayo). ¿Qué es malware? McAfee.* <https://www.mcafee.com/es-mx/antivirus/malware.html>
- *RSA Encrypt / Decrypt - Examples - Practical Cryptography for Developers. (s.f.).* <https://cryptobook.nakov.com/asymmetric-key-ciphers/rsa-encrypt-decrypt-examples>
- *RSA - PyCryptoDome 3.19.0 Documentation. (s.f.).* https://pycryptodome.readthedocs.io/en/latest/src/public_key/rsa.html
- *Py2Exe and Matplotlib: Plot won't appear. (s.f.). Stack Overflow.* <https://stackoverflow.com/questions/8301694/py2exe-and-matplotlib-plot-wont-appear>
- *AES Encryption And Decryption in Python: Implementation, Modes And Key Management. (s.f.). onboardbase.* <https://onboardbase.com/blog/aes-encryption-decryption/>



- *Pyinstaller and -Onefile: How to include an image in the EXE file. (s.f.). Stack Overflow.*
<https://stackoverflow.com/questions/31836104/pyinstaller-and-onefile-how-to-include-an-image-in-the-exe-file>