Universidad de Alcalá
Escuela Politécnica Superior

Grado en Ingeniería de Computadores

Trabajo Fin de Grado

Ejecución de malware en entornos controlados

Autor: Adolfo Gómiz Barreda

Tutor/es: Manuel Sánchez Rubio

2022

Agradecimientos

Al fin, tras muchos años, estoy entregando este trabajo con el que puedo poner fin a una importante etapa de mi vida.

Quisiera comenzar dando las gracias a mi familia, que siempre me ha apoyado académica y personalmente. De no ser por su constancia e insistencia no habría llegado a ser la persona que soy hoy.

Quiero dar las gracias también a todos aquellos que han recorrido este camino conmigo, tanto dentro como fuera de la facultad. A mis compañeros de carrera y en especial a mi constante compañero de laboratorio y ahora compañero de fiestas por haber estado ahí desde el principio apoyándonos el uno en el otro para salir adelante hasta llegar a la graduación. Sin esas infinitas tardes y noches de estudio juntos ambos sabemos que quizás no hubiese sido más difícil, pero sin duda mucho más aburrido.

Y por último, quería agradecer especialmente a mi pareja, que ha estado a mi lado desde que entré en la carrera y que sigue a mi lado muchos años después. Gracias por no haberte rendido nunca y haber compartido esta carga poniendo todo de tu parte para conseguir que terminase esta etapa universitaria que tan larga se nos ha hecho a los dos. Sin tu ayuda no estaría aquí hoy.

Resumen

El análisis de malware ha sido y sigue siendo la principal herramienta de defensa que existe contra la propagación del código malicioso. Si bien gran parte de los análisis que se llevan a cabo hoy día son realizados de forma automática, sigue siendo necesaria la intervención de analistas profesionales que puedan analizar manualmente las nuevas muestras de malware que surgen cada día para identificar patrones en los mismos y mejorar las normas de análisis automático.

Es por esto que, a lo largo de este trabajo, estudiaremos las diferentes tecnologías, herramientas y técnicas para llevar a cabo estos análisis. Una vez estudiadas, escogeremos las herramientas para analizar una serie de muestras de malware obtenidas de la red y comentaremos las conclusiones obtenidas a partir de las mismas.

Palabras clave: malware, ransomware, gusano, dorkbot

Abstract

Malware analysis has been the main tool in the war against de propagation of malicious code until these days. Even if most of the analyses now are done by automatic tools, professional analysts are still needed to manually analyze all the new malware that is created everyday in order to identify logics and rules and to keep these tools updated.

Due to this in this project we will study the different technologies, tools and techniques required to make these analyses. Once we have studied them all we will choose the ones we are going to use in the study cases we have obtained from the web and will comment on the conclusions obtained.

Keywords: malware, ransomware, worm, dorkbot

Índice

1 · Introducción

En los últimos años los casos de ataques de malware han aumentado considerablemente. Ya en el año 2018 los costes asociados con únicamente ataques de ransomware se estimaron en torno a ocho mil millones de dólares, ya que, si bien es cierto que más del 90% de las víctimas no pagaban el rescate de sus archivos, la pérdida de estos suponía perder una enorme cantidad de horas de trabajo. [1]

En los últimos dos años ha tenido lugar un nuevo pico en los ciberataques. Debido al confinamiento, la necesidad de usar las plataformas virtuales para cualquier tipo de trámite aumentaron un 100% los casos de ransomware, siendo los sectores financieros y de cuidados infantiles los más afectados. [2]

Todo esto hace que hoy día sea más importante que nunca la capacidad de detectar el malware lo más pronto posible y evitar que llegue a ejecutar su código malicioso. El malware conocido es fácil de detectar, pero siempre habrá código binario desconocido que no se podrá prever.

El procedimiento para detectar el nuevo código malicioso suele consistir en el análisis de un experto que averigüe la naturaleza del código ejecutable. Si decide que se trata de código malicioso creará una firma para el mismo basada en las diferentes características que posea para que en el futuro los antivirus sean capaces de detectar este malware (o alguna variante similar). Este sistema, si bien es bastante fiable, tiene como principal problema el coste de tiempo que conlleva. Es inviable aplicar este estudio a todos los ejecutables que surgen cada día. Por ello se han creado herramientas de análisis automático que ejecutan las diferentes técnicas existentes para averiguar las intenciones del archivo. Tras el análisis de estas muestras se produce un informe que describe los diferentes efectos de la muestra, incluyendo si es benigna o maligna. De esta forma sólo se requiere el análisis de un experto para los casos de código nunca antes visto.

Las técnicas de análisis de código han evolucionado con el paso de los años, y en la actualidad son capaces de identificar no solo más formas de ataque, sino incluso las diferentes formas en que el malware intenta evitar ser reconocido.

Los objetivos que realizaré durante el transcurso del estudio serán:

- Investigación sobre las diferentes herramientas software disponibles para el análisis de malware.
- Estudio de los dos tipos de análisis de código que se aplican en el estudio del malware: análisis estático y análisis dinámico.
- Planteamiento de soluciones contra dicho malware de forma preventiva para evitar la infección y de forma correctiva tras haber sido infectados.
-

Estos objetivos se desarrollarán a lo largo de los diferentes capítulos de este proyecto de la siguiente manera:

En el capítulo dos, estado del arte, evaluaremos las diferentes herramientas que existen para el análisis de malware en entornos seguros. Al concluir el capítulo tendremos nuestro entorno de pruebas completamente montado con todas las herramientas necesarias explicadas.

En el capítulo tres estudiaremos los distintos acercamientos al análisis de malware. Para ello explicaremos en qué consiste el análisis estático, el análisis dinámico, las diferencias entre ambos y las ventajas y desventajas de uno con respecto al otro.

En el capítulo cuatro aplicaremos ambas técnicas con diferentes muestras de malware obtenidas de la red. Durante este capítulo pondremos en práctica tanto las herramientas software del capítulo dos como las técnicas de análisis del capítulo tres. Tras cada análisis expondremos, una vez estudiado su funcionamiento, las diferentes formas en que se puede evitar una infección de dicho malware o eliminarlo del sistema una vez ha sido infectado, siempre que las haya.

Por último, en el capítulo cinco expondré las conclusiones resultantes del proyecto, así como las diferentes mejoras que se podrían hacer del mismo.

2 · Estado del arte:

En este capítulo estudiaré las categorías de malware existentes así como las diferentes tecnologías software que se usan actualmente en el análisis de malware en entornos controlados.

Antes de proseguir es importante aclarar en qué consiste el malware y cómo se agrupan sus diferentes variantes.

2.1 · Categorías de malware

El término malware es un acrónimo de las palabras malicious software, haciendo referencia a scripts o ejecutables que realizan actividad maliciosa o perjudicial. El malware puede aparecer bajo diferentes formatos: scripts, ejecutables, firmware… Para el objetivo de este proyecto, cuando empleemos el término malware nos referiremos en concreto a ficheros de código binario con intenciones maliciosas.

El malware busca comprometer la confidencialidad, la integridad o la disponibilidad de un sistema, ya sea mediante la explotación de vulnerabilidades ya existentes o mediante la creación de nuevas.

Dentro del malware existen, además, varias categorías. No hay un único criterio a la hora de clasificarlas, así que vamos a explicar los dos más comunes.

Si clasificamos el malware por tipos nos encontramos con la división más común del mismo. Estos términos son los más conocidos y, si bien es cierto que el malware moderno suele pertenecer a varios de estos grupos, sigue siendo una clasificación cómoda para entender el alcance y funcionamiento de los mismos. A continuación procedo a explicarlos brevemente.

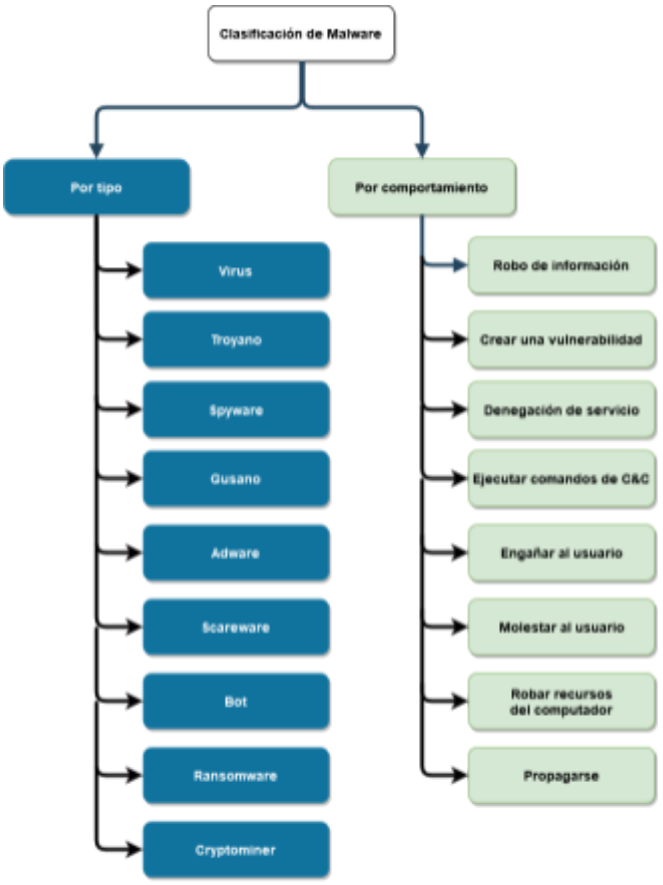


Figura 2.1 - 1 Clasificación de malware

- **Virus**: un virus se propaga introduciendo código malicioso en otros archivos, pudiendo llegar así a infectar otras máquinas.
- **Troyano**: se trata de un tipo de software malicioso que se oculta bajo la apariencia de software normal. Existe una gran cantidad de troyanos que desarrollan diferentes tareas.
- **Spyware**: registran la actividad del usuario sin su consentimiento e informan al atacante. La información que almacenan puede ser muy distinta: credenciales, actividad en la red, etc.
- **Gusano**: se trata de un programa que se autorreplica a través de la red. Los gusanos se multiplican rápidamente y pueden llegar a saturar la red, pero no suelen ser difíciles de eliminar una vez descubiertos.
- **Adware:** este malware muestra anuncios al usuario infectado. Estos anuncios pueden aparecer en diferentes páginas web o incluso dentro de otros programas.

- **Scareware**: este malware muestra una ventana o una notificación informando al usuario de que su ordenador ha sido infectado. A menudo están diseñados para ser similares a las notificaciones de un antivirus y buscan que compres un software que los elimina.
- **Bot:** se trata de un software malicioso que realiza distintas actividades sin el consentimiento del usuario de forma remota. Todos las máquinas infectadas reciben los comandos desde un servidor de control, y el conjunto de máquinas infectadas se denomina Botnet. Entre las diferentes actividades que pueden llevar a cabo se incluye expandir malware, realizar ataques de denegación de servicio o visitar ciertas webs.
- **Ransomware**: uno de los tipos de malware más extendidos. Este software encripta los archivos del usuario y exige un pago para obtener la clave de desencriptación. En general no suelen restringir el uso de la máquina infectada, pero hace inaccesibles todos los archivos del mismo. Con la aparición de las criptomonedas han pasado a ser aún más populares al permitir recibir los pagos y permanecer en el anonimato.
- **Cryptominers:** este malware es el más reciente. Aprovechan los recursos de la máquina infectada para minar criptodivisas. Las máquinas infectadas sufren el deterioro por el uso continuado y el atacante recibe los beneficios de las criptodivisas.

Aunque los términos arriba descritos son prácticos para definir los tipos de malware, a la hora de analizarlos es más cómodo referirse a los distintos malware por el comportamiento que tienen al ejecutarse. A continuación procedo a explicar los diferentes tipos.

- **Robo de información:** este es uno de los comportamientos más comunes, ya sea el robo de información bancaria o el robo de credenciales. Dentro de esta categoría entrarían los distintos softwares de robo de credenciales, keyloggers, spywares y sniffers.
- **Crear una vulnerabilidad:** otro comportamiento muy común, el malware suele buscar la forma de facilitar la infección a más malware. Para esto se pueden tratar de desinstalar los antivirus, abrir formas para acceder a la máquina infectada o cambiar ajustes de firewall. De esta forma aunque se elimine el malware, seguirá siendo útil al ayudar a infectar de nuevo la máquina por nuevo software.

- **Denegación de servicio**: este tipo de ataque busca bloquear el acceso o la disponibilidad de distintos servicios. Esto se puede hacer mediante la saturación de servidores con falsas peticiones (DDoS), bloqueando el acceso a los recursos de una máquina (por ejemplo, mediante ransomware) o dañando el propio hardware.
- **Ejecución de comandos de C&C:** como vimos con los bots, el malware puede incluir formas de ejecutar procesos de forma remota desde un servidor.
- **Engañar al usuario:** este comportamiento tiene lugar principalmente en el sector financiero mediante ataques de phishing o sustituyendo las webs reales por versiones ficticias que roban los datos bancarios de la víctima.
- **Molestar al usuario:** este comportamiento consiste en alterar la experiencia del usuario al usar la máquina infectada, ya sea mediante anuncios (adware) o modificando las configuraciones de la máquina (cambiando el navegador por defecto).
- **Robar recursos del computador:** ciertos tipos de malware, como el criptominer, buscan usar los recursos de la máquina infectada en beneficio propio. En general este malware no suele impedir el uso de la máquina infectada, pero sí que afecta a la experiencia reduciendo el porcentaje de recursos al que tiene acceso el usuario.
- **Propagarse:** el comportamiento más común en los virus y los gusanos. En muchos casos no hace falta nada más, ya que, como ocurrió con el gusano Morris, la simple propagación del gusano en la máquina infectada puede llegar a provocar un comportamiento de denegación de servicio.

Al igual que con la clasificación anterior, es muy común que una pieza de malware pertenezca a varias categorías a la vez. Cualquier malware que se dedique al robo de datos de la máquina se beneficia a su vez de una forma de propagarse, o de engañar al usuario.

2.2 · Herramientas de análisis

Para poder analizar las diferentes muestras de malware los investigadores recurren a una gran variedad de herramientas que facilitan el trabajo. En este punto vamos a cubrir las más utilizadas a día de hoy agrupadas en las siguientes categorías: entornos seguros, desensambladores y monitores.

2.2.1 · Entornos seguros

Para poder llevar a cabo un análisis de malware de forma segura lo primero que necesitamos es conseguir un entorno seguro donde poder ejecutar esta prueba de malware sin poner en peligro al resto de máquinas de nuestra red. Para conseguir esto vamos a necesitar un entorno virtualizado, y actualmente hay varias opciones:

- **Máquinas virtuales:** las máquinas virtuales han sido vitales para el desarrollo del análisis de malware. Ofrecen un entorno virtualizado casi idéntico al de una máquina real donde poder investigar cómo afecta el malware al entorno. Se trata de herramientas muy sencillas de configurar, pudiendo virtualizar distintos sistemas operativos creando entornos limpios para las pruebas en apenas minutos. Los diferentes programas que permiten la creación de máquinas virtuales ofrecen además funciones de gran utilidad para el análisis, como la posibilidad de hacer capturas del estado de una máquina antes de ejecutar el malware para restaurarla después de las pruebas. Por otro lado, aunque una máquina virtual es casi idéntica a una máquina real, hay formas de detectar las diferencias y algunos programas maliciosos usan este conocimiento para modificar su comportamiento para evitar ser analizados. Algunas de estas son diferencias entre el hardware emulado y el hardware real a muy bajo nivel o diferencias entre los tiempos de ejecución de determinados comandos que pueden hacer pensar al malware que se trata de un entorno virtualizado.

- **Entornos sandbox:** similares a las máquinas virtuales, estos entornos están específicamente diseñados para analizar software ya que permiten que un programa se ejecute con acceso a los recursos de un sistema pero manteniendo todas las modificaciones hechas por dicho programa aisladas del resto. De esta forma se puede estudiar cómo interactúa el malware en una máquina pero una vez termina su ejecución todos los cambios que ha llevado a cabo son descartados. Debido a la forma en que trabajan los entornos sandbox son excelentes como herramientas de carácter preventivo debido a que entre sus principales ventajas está el bajo consumo de recursos y se pueden usar únicamente para ejecutar determinados programas más susceptibles a infectar la máquina (generalmente navegadores web).

- **Contenedores:** los contenedores son una agrupación del código de una aplicación con las bibliotecas, los archivos de configuración y las dependencias necesarias para su ejecución independientemente del sistema sobre el que se ejecute. Para ello virtualiza el sistema operativo anfitrión y permite que la aplicación del contenedor interactúe con los recursos del mismo sin poder percibir el resto de procesos en ejecución. Dado que aunque hay un aislamiento con respecto al resto de procesos y programas del sistema se trata de una alternativa menos segura para el análisis de malware debido a que al no virtualizar los recursos ni el espacio de trabajo el nivel de aislamiento que proporcionan es menor que el de los entornos sandbox o las máquinas virtuales.

Por tanto, para este punto vamos a decidir entre una máquina virtual o un entorno sandbox. Los sandbox no requieren apenas recursos lo que supone una ventaja con respecto a las máquinas virtuales, que requieren una imagen del sistema operativo a virtualizar y se reservan durante su ejecución todos los recursos de la máquina que le hayamos asignado. Por otro lado, una máquina virtual ofrece un mayor control sobre el entorno virtualizado. Los entornos sandbox permiten la ejecución aislada de software en el mismo entorno de la máquina original. Esto nos permite estudiar el funcionamiento de dicho malware en nuestra propia máquina de forma segura, pero no es tan sencillo estudiar un punto intermedio de esa ejecución o mantener una máquina infectada durante largos periodos de tiempo para su estudio sin comprometer la máquina original. Este no es el caso con una máquina virtual que, aunque de forma imperfecta, virtualiza todos los recursos de la máquina permitiéndonos trabajar de forma independiente entre la máquina virtual y la real.

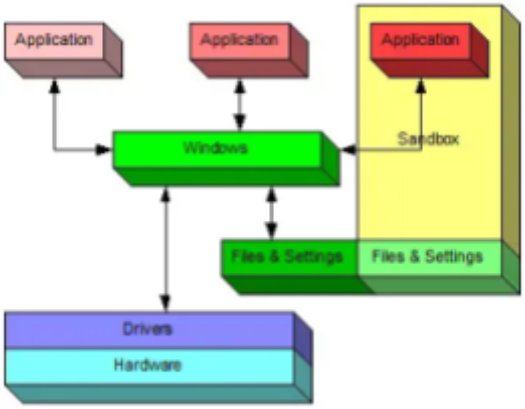


Figura 2.2.1 - 1 Esquema de un entorno sandbox

Figura 2.2.1 - 2 Esquema de una máquina virtual

Además, la configuración de un entorno sandbox es la configuración de la máquina subyacente por lo que para probar con diferentes sistemas o configuraciones necesitas aplicar dichos cambios en una máquina real. En cambio en una máquina virtual puedes emplear un sistema operativo diferente, con configuraciones distintas y todo ello monitorizado en una ventana dentro de tu propio ordenador. Es por esto que para este proyecto vamos a trabajar con una máquina virtual y el programa VirtualBox.

2.2.2 · Desensambladores y debuggers

Un desensamblador es un software que permite traducir el lenguaje máquina a lenguaje ensamblador. Se trata de una herramienta de ingeniería inversa que nos permite analizar de una forma legible para un humano, el código de una muestra de malware a partir de su ejecutable. Actualmente hay dos opciones que compiten por el puesto de herramienta más utilizada para el desensamblado. Las herramientas de debugging, por otro lado, son una parte indispensable para el análisis dinámico, ya que nos permiten ejecutar por partes el código del malware para analizar en tiempo real los cambios que efectúa en el sistema infectado.

- **Ghidra**: publicado en marzo del año 2019 por la Agencia de Seguridad Nacional de los Estados Unidos (NSA) se trata de una herramienta de código abierto en lenguaje Java. Entre sus principales características está la capacidad de trabajar con software compilado para diversas plataformas como Linux, Windows y MacOS. Además, al tratarse de un framework además de un desensamblador, incluye herramientas que no se ven en otros desensambladores, como la capacidad de producir código fuente además de código ensamblador. Como punto negativo, al tratarse de una herramienta de código abierto es importante estar al día de los avances de la comunidad para sacarle todo el partido.

- **IDA Pro:** se trata de un desensamblador multiplataforma que, al igual que Ghidra, traduce el código máquina a código ensamblador. Durante años fue el desensamblador de uso más extendido y posee una versión de pago y otra gratuita. Entre sus principales características está la capacidad de debuggear el código a través de mapas

de ejecución de las diferentes funciones lo que reduce el número de herramientas software que necesitaremos. Además, al haber sido el estándar de la industria hasta la aparición de Ghidra, cuenta con muchísima información online que puede ser de utilidad a la hora de empezar a trabajar con él.

- **WinDbg:** windows debugger es una herramienta del sistema operativo windows. Ha formado parte del conjunto de herramientas básico de los analistas de malware durante años, aunque hoy día hay varias alternativas para los recién iniciados. Posee una enorme cantidad de tutoriales en la red para comenzar a usarlo, aunque puede ser complicado de inicio.

- **x32/x64:** este debugger se trata de una herramienta de código abierto que ha acabado por convertirse en una de las herramientas más populares para debuggear en windows. Su principal ventaja es que, al contrario de su predecesor winDbg, es muy sencilla de utilizar y su interfaz es más fácil de entender.

Ambos son los mejores desensambladores que se pueden encontrar actualmente. La versión de pago de IDA es superior a Ghidra en temas de rendimiento e interfaz pero su precio es elevado (365 dólares al año). Ghidra a cambio ofrece la posibilidad de obtener un resultado de código en C (aunque si el malware ha aplicado alguna técnica de ofuscación el resultado puede seguir siendo ilegible) y es completamente gratuito. Para este proyecto he optado por usar la versión gratuita de IDA, que cuenta con muchos tutoriales en la red para comenzar a utilizarlo e incluye la posibilidad de debuggear el código con la misma herramienta, lo que nos permite reducir el número de herramientas a aprender.

2.2.3 · Monitores

Una de las herramientas básicas para analizar la ejecución de malware son los monitores. Existen multitud de programas que pueden cumplir esta función, así que aquí solo comentaré los más populares.

- **Process Hacker:** se trata de una herramienta que nos permite ver todos los procesos que se están ejecutando en una máquina. Se trata de una herramienta de pago que nos permite identificar los procesos aunque estos se intenten hacer pasar por procesos del

sistema cambiando su nombre ya que puede ordenarlos por hora de creación y registrar estos cambios. Entre sus principales características está la capacidad de observar las cadenas de texto recientes almacenadas en memoria. Durante la ejecución de malware estas cadenas pueden tratarse de información importante como direcciones IP o webs a las que el malware está intentando acceder.

- **Process Monitor:** una de las mejores herramientas de monitorización que es gratis para los administradores de sistemas Windows. Cuenta con una serie de filtros que nos permiten identificar los procesos generados por un ejecutable aunque estos hayan muerto, identificar las dependencias padre/hijo de estos procesos con otros procesos ejecutados en el sistema y recuperar información de los mismos.

- **Wireshark:** wireshark es el software estándar de la industria para el análisis de tráfico de la red. Este software gratuito nos permite inspeccionar paquetes de información en tiempo real, filtrarlos para eliminar el ruido y revisar sus contenidos. Imprescindible si se va a analizar cualquier tipo de malware que se base en el intercambio de información con el atacante.

3 · Análisis de malware:

El análisis de malware es el proceso por el que se estudia el funcionamiento de una pieza de malware. Esto incluye la obtención de información sobre qué tipo de malware es, el daño que es capaz de causar y, lo más importante, cómo eliminarlo.

Dentro de los análisis de malware existen varias categorías, dependiendo principalmente de si se ejecuta o no el código a estudiar. Es por esto que podemos diferenciar dos tipos diferentes de análisis: estáticos y dinámicos.

3.1 · Análisis estático

El análisis estático es una técnica que consiste en el análisis del código de un programa sin la ejecución del mismo. Este proceso nos permite conocer y entender el funcionamiento del programa sin comprometer la seguridad de la máquina. El análisis estático comprende desde los detalles más generales de la muestra de malware, como los metadatos (nombre del fichero, tipo o tamaño). Incluso el MD5 puede ser suficiente para averigüar información sobre el malware comprobando diversas bases de datos de análisis previos.

Cuando se necesita más información, se recurre a un análisis más avanzado. En este punto se requiere el código del malware, por lo que se recurre a herramientas externas como los desensambladores. Una vez obtenido el código de ensamblador podemos comenzar a estudiarlo. Un analista es capaz de aplicar diversas técnicas para reconocer patrones dentro de dicho código y así averigüar los detalles del mismo. Vamos a explicar a continuación algunas de estas técnicas:

- **Obtener la hash:** esta técnica es la más útil para encarar un análisis de malware. Nos va a permitir generar unas claves únicas para el archivo que vamos a analizar que podremos después contrastar con las bases de datos públicas de malware. Cada día se actualizan con miles de nuevos registros, por lo que es muy probable que la muestra que vamos a analizar ya haya sido analizada anteriormente.
  Obtener una clave hash es un proceso por el que, al aplicar un algoritmo al archivo, obtendremos una clave única del mismo. Existen varios algoritmos, aunque los más populares son MD5, SHA1, SHA256 y SHA512.

Algorithm	Output Bits	Broken
MD5	128	Yes
SHA1	160	Yes

Como se puede ver, prácticamente existen dos claves, MD5 y SHA, que posee diferentes versiones variando la longitud de las claves. Las diferencias entre ambas claves son que la clave MD5 es más rápida de generar, mientras que las claves SHA son mucho más complejas. Existen multitud de formas de obtener estas claves, pero la más sencilla es usando la consola Windows Powershell. Esta incluye un comando que automáticamente nos devuelve las claves hash de un archivo, y nos puede devolver cualquiera de las anteriormente citadas.

Una vez obtenida alguna de estas claves podemos acceder a cualquier base de datos pública de malware, como VirusTotal, y comprobar si ya ha sido analizado antes.

- **Revisar las cadenas de texto:** esta técnica es especialmente útil para analizar malware que requiere de direcciones o mensajes definidos. Esas cadenas de texto pueden ser todo lo que necesitemos para entender las intenciones del programa y saber si se trata o no de malware. Para esto existe una herramienta que forma parte de la Microsoft Sysinternals Suite llamada Strings. Esta herramienta recorrerá todo el contenido del archivo registrando todas las cadenas de texto que encuentre. Una vez las ha obtenido todas, mediante expresiones regulares podemos descartar todas las que no sean más que ruido y revisar el resto.

- **Revisar las librerías y funciones importadas:** además de las cadenas de texto, una fuente de información clave para entender las capacidades de una muestra de malware son las librerías y las funciones de las mismas que emplea. Para esto existen varias herramientas, como Dependency Walker.

- **Revisar las capacidades del malware:** este último punto no es exactamente una técnica, pero se puede obtener mucha información del malware que vamos a analizar aplicando previamente un análisis automático con herramientas como CAPA. Estas herramientas cuentan con una gran cantidad de "reglas" que les permiten identificar de forma automática patrones que denoten ciertos comportamientos o funciones consideradas maliciosas. Estas reglas son actualizadas continuamente por la comunidad o equipo detrás del proyecto y nos pueden servir para hacernos una idea de qué tipo de fichero vamos a investigar y de lo que es capaz.

Por otro lado, los autores de malware a veces emplean técnicas de ofuscación para que su malware sea más complicado de analizar. Se trata de técnicas cuya función es crear segmentos de datos difíciles de interpretar. Esto de por sí ya puede considerarse una pista de si se trata de código malicioso o no, ya que un programa no malicioso probablemente esté poco ofuscado y se podrán identificar varias cadenas de texto sin encriptar dentro del mismo. Algunas de estas técnicas son:

- **Inserción de código muerto:** esta técnica es muy sencilla, y consiste en añadir segmentos de código que no tienen ninguna función. El ejemplo más típico es la función NOP, que no hace nada y permite que la ejecución del programa siga con la siguiente instrucción. Este tipo de ofuscación se puede evitar si antes de analizar el código se eliminan todos los segmentos inútiles de forma automática.

- **XOR:** este tipo de ofuscación es sencillo. Se basa en aplicar la operación XOR para encriptar la información a nivel de bytes. La operación XOR tiene la particularidad de que, si uno de los factores es aleatorio, el producto es completamente impredecible. Esto se debe a que la tabla de la verdad de la función XOR devuelve verdadero cuando los factores son distintos, y falso cuando son idénticos.

XOR Truth Table		
Input		Output
0	0	0
0	1	1
1	0	1

- **Reemplazo de instrucciones:** esta técnica consiste en intercambiar las instrucciones sin alterar el funcionamiento del programa. Para ello basta con emplear una librería que incluya instrucciones equivalentes.

- **Transposición de código:** esta técnica reordena el código para complicar su legibilidad. Para ello existen dos formas: la forma más sencilla es reorganizando todas las instrucciones del programa añadiendo después instrucciones de salto obligatorias que permiten seguir el flujo de ejecución original, aunque se puede recuperar el código original siguiendo esa línea de ejecución y eliminando los saltos. La forma más compleja consiste en reorganizar únicamente aquellas instrucciones cuyo orden no afecta a la ejecución del software. Debido a lo complicado que puede ser identificar todas las instrucciones independientes este sistema es más difícil de emplear, pero también es más laborioso de corregir.

- **Integración en código:** esta técnica inserta el código de ejecución del malware dentro del código de otro programa. Identifica los segmentos independientes para introducir sus instrucciones en mitad de dicha ejecución.

- **Empaquetadores:** hay casos donde el programa completo se ofusca comprimido dentro de otro programa encargado de desempaquetarlo. De esta forma el código ejecutable que resulta tras la descompresión es una variante del código original.

Estas han sido solo unas pocas de las muchas técnicas que se utilizan hoy día para evitar que el malware sea identificado y analizado, y es importante conocerlas para poder superarlas durante los análisis.

3.2 · Análisis dinámico

El análisis dinámico de malware consiste en la ejecución del software malicioso en un entorno seguro para poder estudiarlo en funcionamiento sin poner en riesgo el resto de máquinas de la misma red. Se trata de un tipo de análisis mucho más rápido, que nos permite conocer el funcionamiento del programa malicioso sin el coste de tiempo que requiere un análisis estático.

Para preparar este entorno seguro lo más importante es evitar la conexión directa con la red. Aunque el programa se esté ejecutando en una máquina virtual, si el malware tiene acceso a la red se podrá propagar a través de la misma. Lo mismo ocurre con los recursos compartidos con la máquina anfitriona, ya que si hay carpetas compartidas podrán ser infectadas afectando a la misma.

Una vez tenemos un entorno seguro y hemos ejecutado el software podemos revisar la ejecución del mismo mediante herramientas de monitorización como las descritas en el capítulo 2.

Como podemos ver, el análisis dinámico es más sencillo de ejecutar que el análisis estático, pero sigue requiriendo una serie de condiciones para que la información obtenida sea lo más completa y acertada posible. Estos puntos serían:

- **Los datos no deben de verse comprometidos:** si queremos obtener información de cómo afecta un software malicioso a una máquina no debemos permitir que éste tome control de la misma a riesgo de que la información que observemos después se vea manipulada.
- **El software no debe saber que está siendo analizado:** como vimos anteriormente, una de las medidas de seguridad que pueden implementar los autores de malware es un sistema para identificar si están siendo ejecutados en un entorno virtualizado, y en

ese caso no ejecutar comandos maliciosos. Esto puede ser peligroso ya que podemos percibir como inofensivo un software que solo es más complicado de estudiar.

- **El entorno debe de ser el que el malware espera:** no sirve de nada ejecutar un malware que requiere de ciertos comandos de windows en una máquina virtual con el sistema operativo linux.

- **Se debe limitar y controlar la conexión a la red:** en determinados casos será necesario que el software se conecte a la red, ya sea para realizar alguna comprobación o para obtener instrucciones. En estos casos se debe recurrir a software que monitoriza dicha conexión o que la emula (como AapateDNS).

4 · Casos de análisis:

A continuación vamos a aplicar los conceptos vistos en los anteriores capítulos para llevar a cabo análisis sobre distintas piezas de malware, indicando durante el proceso la forma en que funcionan y cómo se puede evitar que infecten la máquina o formas de eliminarlos una vez ya ha sido infectada.

4.1 · Gusano Dorkbot

El malware Dorkbot se trata de un gusano que afecta a las máquinas Windows. Como indica su nombre, una vez infectados los ordenadores pasan a formar parte de una red de bots controlados por C&C.

Como veremos a continuación, su forma de propagación es a través de memorias usb que infecta. Una vez la máquina ha sido infectada puede ser controlada a través del protocolo IRC para llevar a cabo las siguientes acciones: robo de credenciales, ataques DDoS, bloqueo de direcciones IP y redirección de tráfico para ataques phishing, entre otros.

4.1.1 · Análisis estático

Para este análisis vamos a comenzar obteniendo la clave hash de la muestra de malware para comprobar si ya hay información sobre el mismo en alguna base de datos online.

Mediante el comando de powershell Get-FileHash obtenemos las siguientes claves MD5 y SHA.

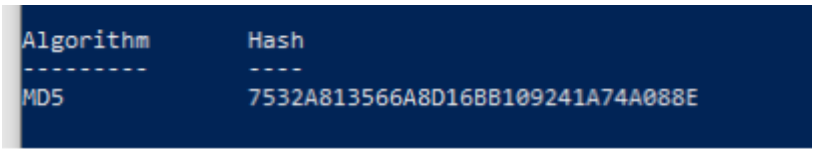


Figura 4.1.1 - 1 Clave MD5

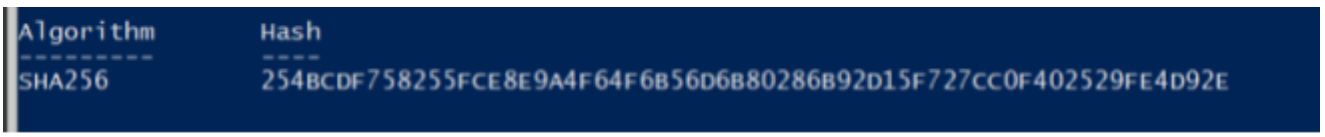


Figura 4.1.1 - 2 Clave SHA256

Cuando buscamos con esa clave en VirusTotal encontramos que ya ha sido analizado con anterioridad, por lo que para completar el análisis estático vamos a revisar su funcionamiento de cara a la ejecución del análisis dinámico.

El malware intentará crear los siguientes archivos:

- %APPDATA%\c731200
- %APPDATA%\update\explorer.exe
- %APPDATA%\microsoft\windows\nqsisv.exe
- <SYSTEM32>\tasks\windows debugger

Además creará multitud de archivos .lnk en el dispositivo extraíble y modificará las claves de registro. El listado de ficheros que va a modificar se encuentra principalmente en las rutas AppData, System32 y los dispositivos extraíbles.

Las claves de registro que va a modificar son principalmente de la ruta \SOFTWARE\Microsoft\Windows NT\CurrentVersion\ como se muestra en las imágenes a continuación.

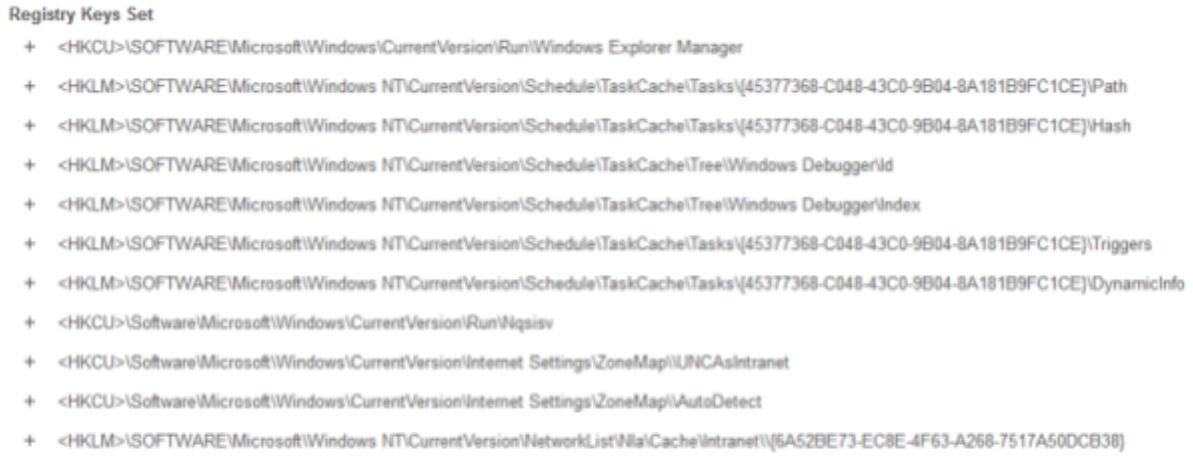
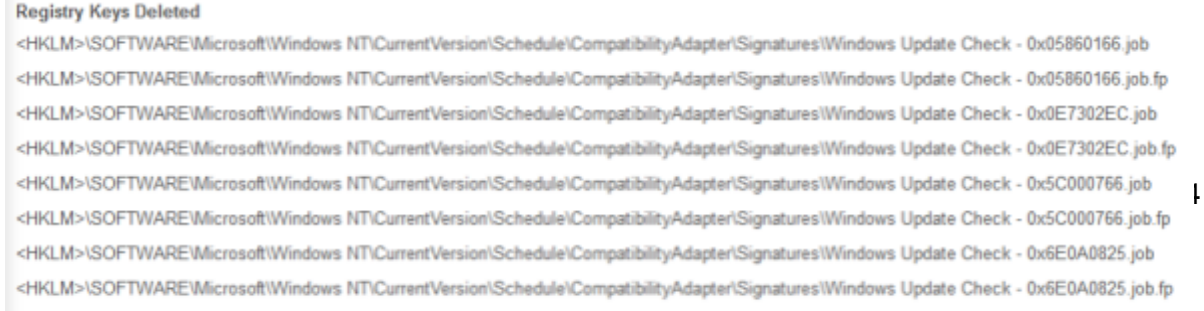


Figura 4.1.1 - 3 Claves de registro que va a modificar el malware



Por último, el malware puede intentar acceder a las siguientes direcciones de red.

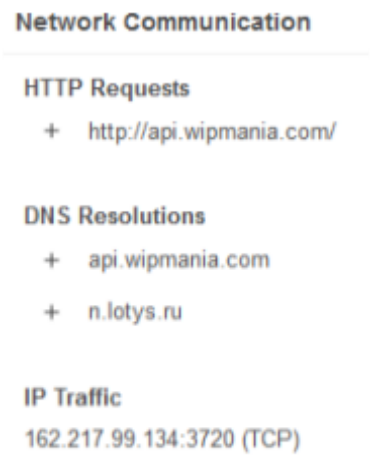


Figura 4.1.1 - 5 Direcciones de red a las que contacta el malware

Con toda esta información nos podemos hacer una idea del funcionamiento del malware. Al haber sido identificado gracias a la clave hash como un gusano de la variante Dorkbot, sabemos que su funcionamiento consiste en propagarse añadiendo máquinas a una botnet dejando una entrada trasera para la ejecución de comandos de forma remota. Esas serán las direcciones a las que intenta atacar el malware probablemente. Para propagarse tratará de infectar una unidad USB que tengamos conectada a la máquina virtual, y ya que todos los archivos que instalará en la misma según el análisis son archivos .lnk sabemos que va a dejar

archivos que son accesos directos, probablemente para ejecutar el malware al acceder a los mismos.

4.1.2 · Análisis dinámico

Tras ejecutar por primera vez el malware podemos ver que el ejecutable ha desaparecido. Como hemos confirmado con el análisis estático, al tratarse de un malware de la familia DorkBot va a tratar de infectar nuestras memorias USB. Esto lo confirmamos en cuanto insertamos un USB con un archivo txt y un par de .zips ya que el contenido del mismo automáticamente pasa a convertirse en accesos directos.

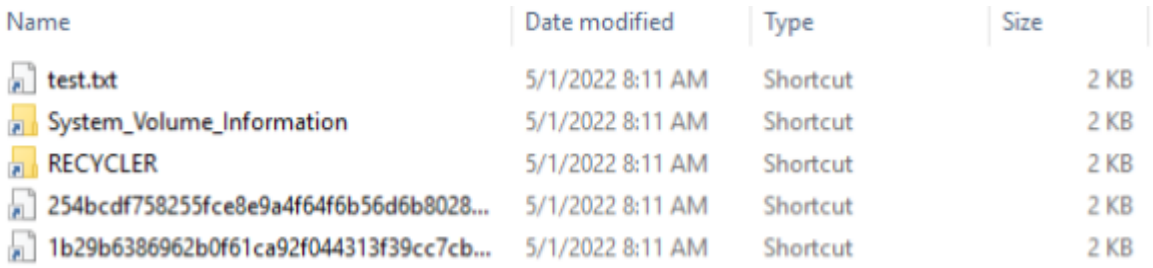


Figura 4.1.2 - 1 Contenido de la memoria USB tras la infección

Si cambiamos la vista para ver archivos ocultos podemos ver que todos nuestros archivos originales siguen en la memoria, por lo que este malware no nos hace perder los archivos.

En esta primera prueba estamos ejecutando el malware con el acceso a la red bloqueado, así que aunque sabemos que necesitará la conexión para ejecutar los comandos de forma remota no tiene un sistema para detectar si hay o no conexión para detener su ejecución y pasar desapercibido.

Si revisamos estos accesos directos vemos que cada uno de ellos incluye una serie de comandos para generar un fichero en la ruta %temp%\keivQ\HpfnMju.exe además de abrir el archivo correspondiente. Esto parece ser un intento de pasar desapercibido para infectar máquinas manteniendo el acceso a los documentos de la memoria USB.

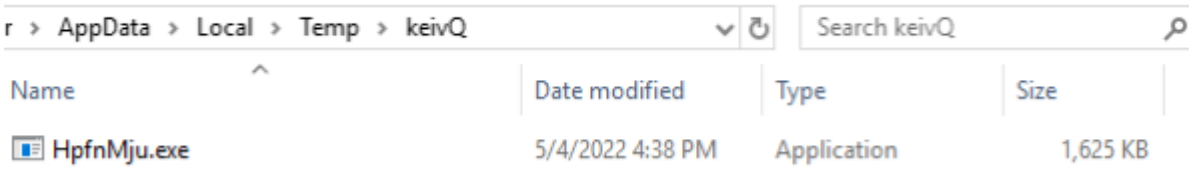


Figura 4.1.2 - 2 Fichero ejecutable creado en la ruta Temp

Si revisamos las carpetas AppData encontraremos varios ficheros generados con fecha de hoy, tal y como vimos en el resumen del análisis estático.

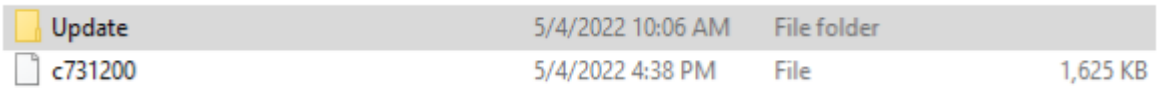


Figura 4.1.2 - 3 Ficheros creados tras la infección en la ruta AppData

Dentro de la carpeta encontraremos un archivo Explorer.exe. Ese ejecutable reemplazará a nuestro explorador de archivos y probablemente sea la puerta de acceso que permitirá al agresor ejecutar comandos de forma remota en la máquina infectada.

4.1.3 · Conclusiones del análisis

Este malware presenta dos caras claramente diferenciadas. Por un lado tenemos un malware que se propaga a través de los dispositivos USB, reemplazando los archivos originales por accesos directos que aprovechan para ejecutar código malicioso antes de redirigirnos al archivo original para pasar desapercibidos. Esta primera parte del malware no es excesivamente problemática, ya que se puede arreglar con un simple formateo de la memoria USB y ni siquiera tenemos que perder nuestros archivos originales, ya que siguen ocultos en la misma. Por otro lado, las máquinas infectadas son más complicadas de recuperar. No es recomendable una limpieza manual de la misma, ya que como hemos visto en el análisis dinámico genera multitud de archivos con la capacidad de volver a infectar la máquina. La mejor alternativa que tenemos es recuperar una imagen limpia de nuestro ordenador. Si tenemos configurado el sistema operativo para que haga capturas de su estado

de forma regular podemos restablecerla sin haber perdido nada de información. En caso de que no tengamos dicha configuración activa lo más recomendable es recurrir a software como Malwarebytes para escanear todas las trazas del malware y eliminarlas. Al tratarse de una variante muy conocida el software de limpieza es capaz de eliminarlo de forma fiable.

4.2 · Ransomware DearCry

El malware Ransomware es un programa malicioso que bloquea el acceso a tus propios archivos mediante encriptación para a continuación exigir un pago (generalmente mediante criptodivisas) a cambio de poder recuperarlos. El ransomware a menudo está diseñado para propagarse a través de la red de máquinas conectadas entre sí, por lo que si un ordenador de trabajo es víctima de uno de estos ataques puede llegar a paralizarse toda la empresa. Se trata de uno de los malware más comunes y genera anualmente pérdidas millonarias.

Uno de los casos más famosos de este tipo de malware es el de DearCry, que aprovechaba una serie de vulnerabilidades de Microsoft Exchange para ejecutar código de forma remota haciéndose pasar por administrador para así propagarse. Otro caso famoso es el de WannaCry, que aprovechando una vulnerabilidad de los sistemas Windows que ya había sido arreglada en un parche dos meses antes de los ataques (EternalBlue).

A continuación vamos a analizar un ejemplo de este tipo de malware: DearCry Ransomware. Para estudiar esta muestra de malware vamos a aplicar ambos análisis, estático y dinámico.

4.2.1 · Análisis dinámico

Si ejecutamos la muestra de malware en la máquina virtual podemos ver cómo todos nuestros ficheros parecen haber sido encriptados.

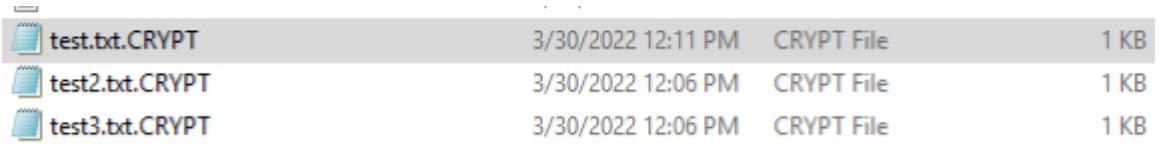


Figura 4.2.1 - 1 Ficheros encriptados

A continuación vamos a abrir uno de estos ficheros encriptados para ver su contenido. Al ver que nuestros archivos han cambiado a .CRYPT podemos confirmar que efectivamente se trata de una muestra de ransomware.

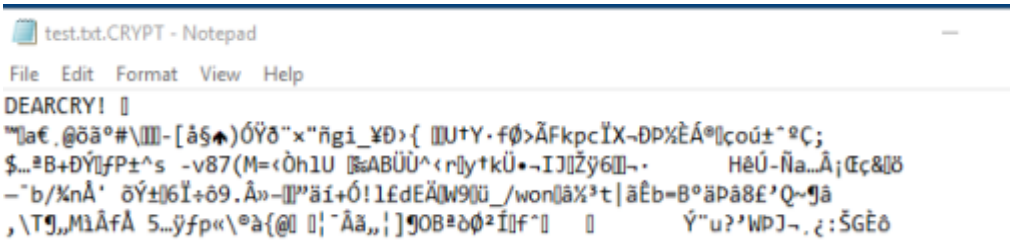


Figura 4.2.1 - 2 Contenido de un fichero cifrado

Como podemos comprobar, el contenido del fichero ha quedado completamente ilegible. Por otro lado, dentro del fichero podemos encontrar una firma en la primera línea. Probablemente sea una cadena de texto que use el malware para no encriptar varias veces el mismo archivo, pero por ahora ya sabemos que se trata de un ransomware de la familia DearCry.

Si revisamos el resto de archivos de la máquina virtual podemos comprobar que los ficheros .exe siguen siendo ejecutables y sólo parece haber encriptado los ficheros de texto e imágenes. El malware quizás tenga algún tipo de filtro de extensiones que encriptar.

Por último, en cada ruta en que se ha encriptado algún fichero podemos encontrar un archivo readme.txt con el siguiente mensaje.

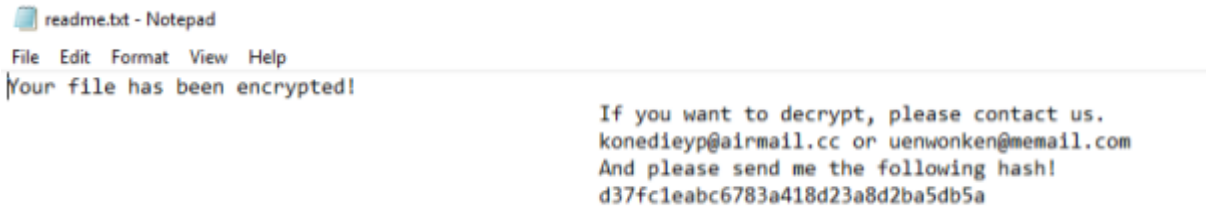


Figura 4.2.1 - 3 Fichero readme.txt con los dirección de contacto del atacante

Dado que el malware se ha ejecutado en un entorno sin acceso a la red y que en el mensaje se nos indica la hash con la que ha sido encriptado, podemos suponer que su sistema de encriptación funciona de forma completamente local sin necesidad de obtener una clave de cifrado de algún servidor. Por lo tanto toda la información que podamos necesitar sobre el mismo debería estar en el propio código del malware.

4.2.2 · Análisis estático

El primer punto que necesitamos aclarar es si el código del ejecutable está ofuscado o no. Para ello vamos a llevar a cabo un primer análisis de cadenas de texto con el comando Microsoft Sysinternals Suite Strings.

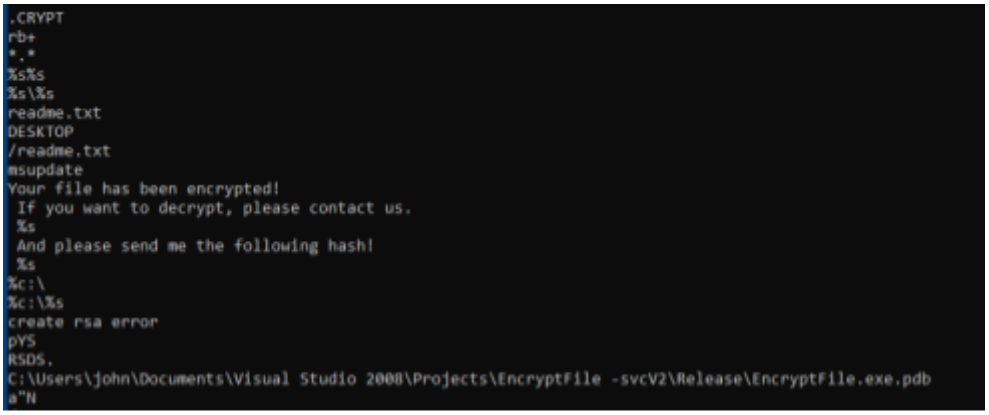


Figura 4.2.2 - 1 Cadenas de texto de la muestra sin ofuscar

Tras descartar mediante expresiones regulares las cadenas que no son más que ruido encontramos varios detalles interesantes. Podemos ver una nomenclatura de archivo .CRYPT, que como hemos visto será con la que renombre los archivos cifrados. Además encontramos el mensaje con el que rellenará los ficheros readme.txt tras el ataque y una referencia a un proyecto de visual studio 2008, concretamente a un .pdb. Con esta información podemos suponer que se trata de un proyecto escrito en C o C++.

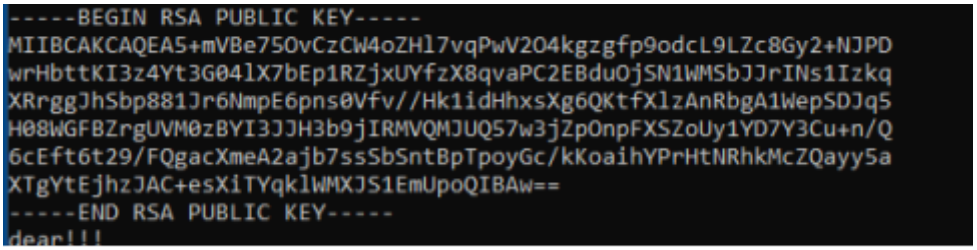
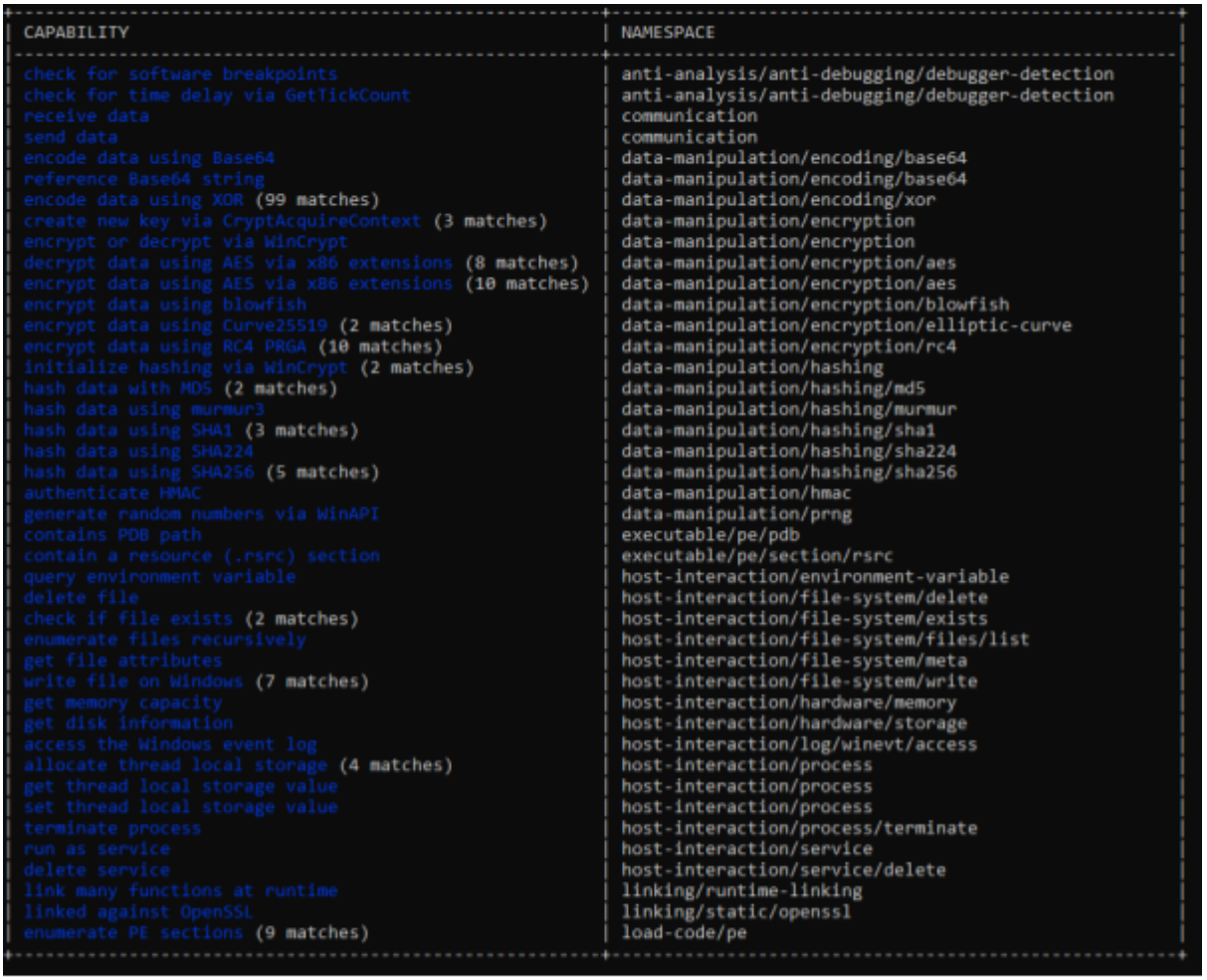


Figura 4.2.2 - 2 Cadena de texto que contiene la clave pública RSA

Además, como supusimos tras el análisis dinámico, la clave pública para realizar el cifrado se encuentra en el propio código como cadena de texto.

Una vez hemos revisado las cadenas de texto vamos a analizar sus capacidades con la herramienta CAPA.



CAPABILITY	NAMESPACE
check for software breakpoints	anti-analysis/anti-debugging/debugger-detection
check for time delay via GetTickCount	anti-analysis/anti-debugging/debugger-detection
receive data	communication
send data	communication
encode data using Base64	data-manipulation/encoding/base64
reference Base64 string	data-manipulation/encoding/base64
encode data using XOR (99 matches)	data-manipulation/encoding/xor
create new key via CryptAcquireContext (3 matches)	data-manipulation/encryption
encrypt or decrypt via WinCrypt	data-manipulation/encryption
decrypt data using AES via x86 extensions (8 matches)	data-manipulation/encryption/aes
encrypt data using AES via x86 extensions (10 matches)	data-manipulation/encryption/aes
encrypt data using blowfish	data-manipulation/encryption/blowfish
encrypt data using Curve25519 (2 matches)	data-manipulation/encryption/elliptic-curve
encrypt data using RC4 PRGA (10 matches)	data-manipulation/encryption/rc4
initialize hashing via WinCrypt (2 matches)	data-manipulation/hashing
hash data with MD5 (2 matches)	data-manipulation/hashing/md5
hash data using murmur3	data-manipulation/hashing/murmur
hash data using SHA1 (3 matches)	data-manipulation/hashing/sha1
hash data using SHA224	data-manipulation/hashing/sha224
hash data using SHA256 (5 matches)	data-manipulation/hashing/sha256
authenticate HMAC	data-manipulation/hmac
generate random numbers via WinAPI	data-manipulation/prng
contains PDB path	executable/pe/pdb
contain a resource (.rsrc) section	executable/pe/section/rsrc
query environment variable	host-interaction/environment-variable
delete file	host-interaction/file-system/delete
check if file exists (2 matches)	host-interaction/file-system/exists
enumerate files recursively	host-interaction/file-system/files/list
get file attributes	host-interaction/file-system/meta
write file on Windows (7 matches)	host-interaction/file-system/write
get memory capacity	host-interaction/hardware/memory
get disk information	host-interaction/hardware/storage
access the Windows event log	host-interaction/log/winevt/access
allocate thread local storage (4 matches)	host-interaction/process
get thread local storage value	host-interaction/process
set thread local storage value	host-interaction/process
terminate process	host-interaction/process/terminate
run as service	host-interaction/service
delete service	host-interaction/service/delete
link many functions at runtime	linking/runtime-linking
linked against OpenSSL	linking/static/openssl
enumerate PE sections (9 matches)	load-code/pe

Figura 4.2.2 - 3 Resultado del análisis de capacidades

Aquí vemos que hay un gran número de llamadas a funciones de codificación y encriptación, así como una ruta a un fichero pdb como el que pudimos identificar en el análisis de las cadenas de texto anteriores.

Tras el análisis dinámico y la revisión de las cadenas de texto y capacidades podemos hacernos una idea del funcionamiento de la muestra de malware.

A continuación vamos a proceder con el análisis de IDA Pro, tras el cual podemos ver que ha dividido el código en una enorme cantidad de funciones, pero no ha sido capaz de identificarlas. Para corregir esto, gracias a la información que hemos obtenido antes analizando las cadenas de texto, vamos a intentar que las reconozca aplicando una firma FLIRT. Revisando las cadenas de texto sin ofuscar dentro del binario hemos encontrado una

ruta de Visual Studio 2008, y tras el análisis con capa encontramos llamadas a una librería OpenSSL, por lo que para mejorar la legibilidad del análisis vamos a descargarnos la firma FLIRT correspondiente a OpenSSL para windows compilado en Visual Studio 2008. Si conseguimos identificar la mayoría de las llamadas nos va a ser más sencillo encontrar qué parte del código es la que ejecuta las funciones del malware.

El siguiente paso es encontrar la función principal de nuestro malware. Durante el análisis de cadenas de texto pudimos ver que el malware intenta acceder a un fichero .pdb, por lo que se debe de tratar de un programa escrito en C, C + + o Visual Basic.

Una vez hemos identificado el proceso de inicio (IDA lo encuentra automáticamente) podemos corroborar que se trata de un ejecutable gracias al siguiente bloque de código:

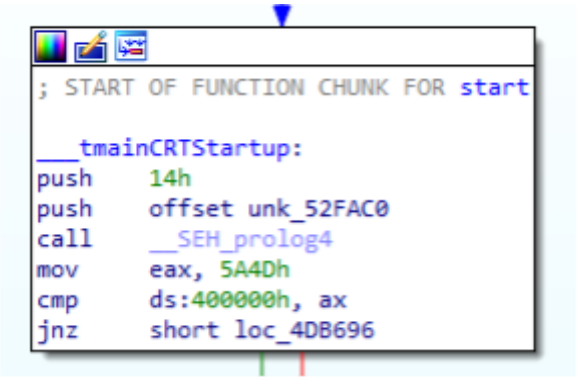


Figura 4.2.2 - 4 Función de inicio del programa

Aquí podemos reconocer la cabecera del programa gracias a la clásica comprobación MZ para evitar problemas de compatibilidad en caso de que se ejecute en DOS. A continuación buscaremos una llamada con 3 atributos, ya que probablemente será la llamada a la función principal. En este caso sólo encontramos una llamada con esas características prácticamente al final del bloque de inicio.

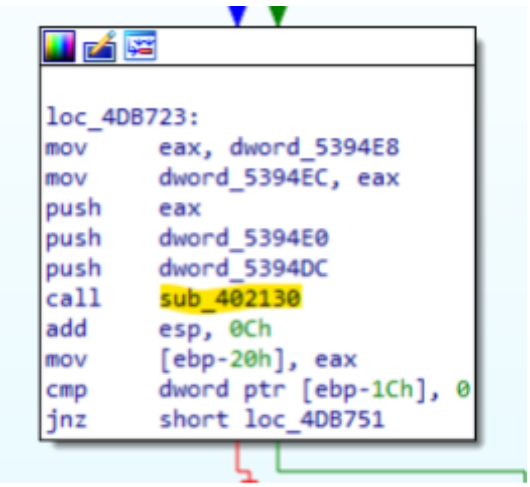
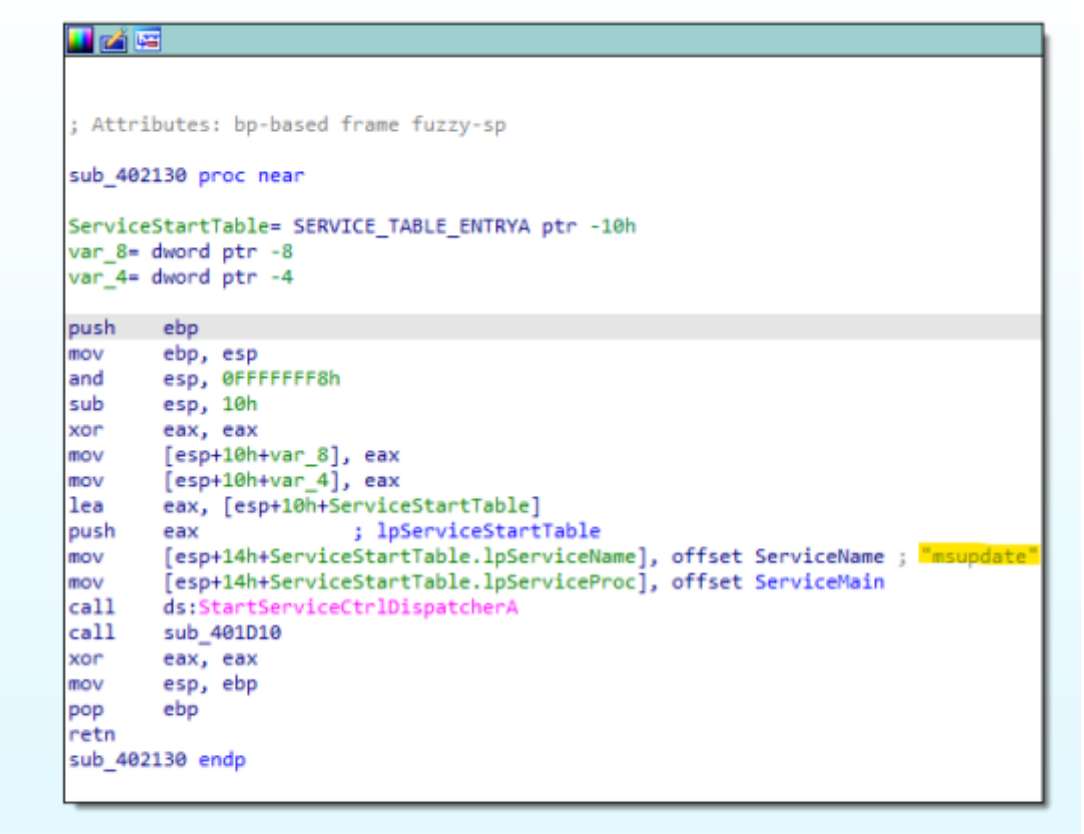


Figura 4.2.2 - 5 Segmento de llamada a la función main

Vamos a renombrar esa función como “main” para hacer el código un poco más legible. Una vez dentro vemos que realiza únicamente 2 llamadas: una a la función StartServiceCtrlDispatcherA con la cadena “msupdate” y el segmento de código de la función ServiceMain y otra a una función que no se ha podido identificar llamada sub_401D10.



```
; Attributes: bp-based frame fuzzy-sp

sub_402130 proc near

ServiceStartTable= SERVICE_TABLE_ENTRYA ptr -10h
var_8= dword ptr -8
var_4= dword ptr -4

push    ebp
mov     ebp, esp
and     esp, 0FFFFFFF8h
sub     esp, 10h
xor     eax, eax
mov     [esp+10h+var_8], eax
mov     [esp+10h+var_4], eax
lea     eax, [esp+10h+ServiceStartTable]
push    eax             ; lpServiceStartTable
mov     [esp+14h+ServiceStartTable.lpServiceName], offset ServiceName ; "msupdate"
mov     [esp+14h+ServiceStartTable.lpServiceProc], offset ServiceMain
call    ds:StartServiceCtrlDispatcherA
call    sub_401D10
xor     eax, eax
mov     esp, ebp
pop     ebp
retn
sub_402130 endp
```

Figura 4.2.2 - 6 Segmento de la función main

Tras investigar en internet sobre la función StartServiceCtrlDispatcherA averiguamos que se trata de una función de windows encargada de lanzar procesos. En este caso se va a crear un proceso de nombre “msupdate” que se encargará de ejecutar el código de la función ServiceMain.

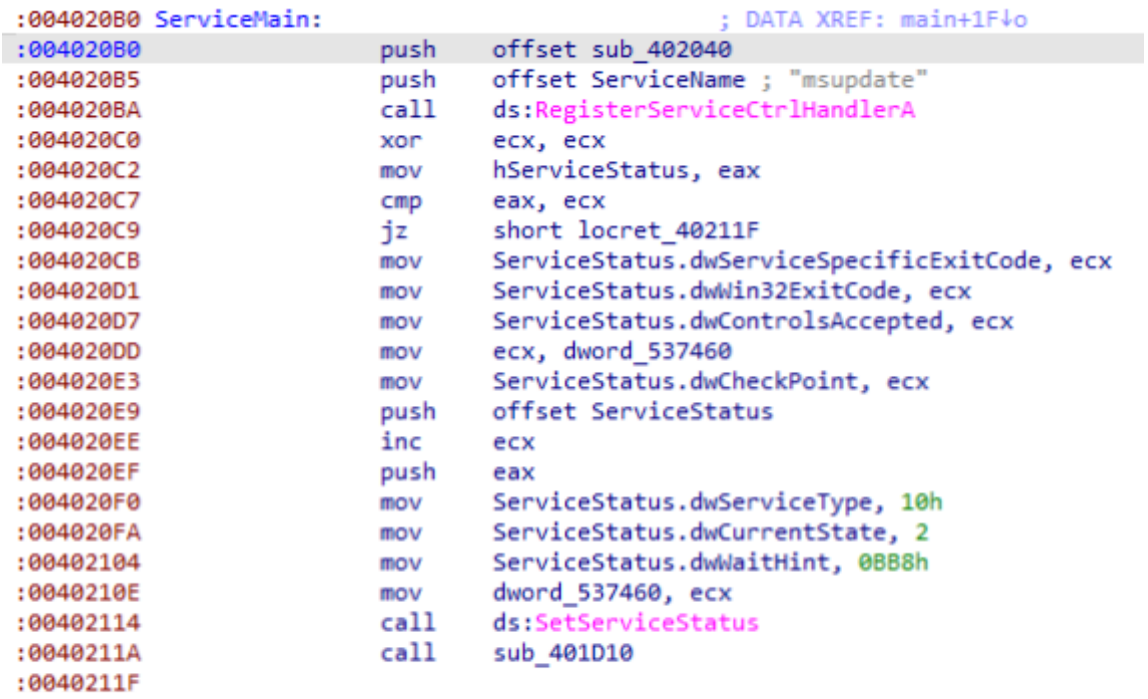


Figura 4.2.2 - 7 Contenido de la función Service Main

Dentro de esta función vemos que se hacen todas las configuraciones para arrancar el proceso para, a continuación, llamar a la función sub_401D10. Esta es la misma función que va a lanzarse desde la función principal, por lo que probablemente sea la función donde se desarrolla el código del malware.

Para averiguarlo vamos a revisar una a una las diferentes llamadas que se hacen dentro de esta función. La primera sería la referencia al segmento sub_402F00, pero este no parece de interés para este análisis por lo breve y sencillo que es. Las siguientes llamadas que tienen lugar son a una función de la librería OpenSSL y a la función calloc, por lo que tampoco son de interés para el análisis. Llegamos así al segmento sub_401000.

```
; Attributes: bp-based frame fuzzy-sp

sub_401D10 proc near

var_11C= dword ptr -11Ch
var_118= dword ptr -118h
lpMem= dword ptr -114h
var_110= dword ptr -110h
Buffer= byte ptr -100h
var_107= byte ptr -107h
var_4= dword ptr -4

push    ebp
mov     ebp, esp
and     esp, 0FFFFFFF8h
sub     esp, 11Ch
mov     eax, ___security_cookie
xor     eax, esp
mov     [esp+11Ch+var_4], eax
mov     eax, hServiceStatus
push    ebx
push    esi
push    edi
xor     ebx, ebx
push    offset ServiceStatus ; lpServiceStatus
push    eax             ; hServiceStatus
mov     ServiceStatus.dwCurrentState, 4
mov     ServiceStatus.dwWin32ExitCode, ebx
mov     ServiceStatus.dwWaitHint, ebx
mov     ServiceStatus.dwControlsAccepted, 1
mov     ServiceStatus.dwCheckPoint, ebx
call    ds:SetServiceStatus
call    sub_402F00
push    eax
call    unknown_libname_41 ; OpenSSL 1.1.0 Library x86 (MSVC 10, /MT)
add     esp, 4
push    1               ; Size
push    100011h         ; Count
mov     [esp+130h+var_110], eax
call    _calloc
mov     esi, eax
push    1               ; Size
push    100011h         ; Count
mov     [esp+138h+var_118], esi
call    _calloc
mov     edi, eax
mov     [esp+138h+lpMem], edi
call    sub_401000
mov     ecx, dword_73A9A8
push    ecx
push    offset aKonedieypAirma ; "konedieyp@airmail.cc or uenwonken@memai"...
push    offset aYourFileHasBee ; "Your file has been encrypted!\n\t\t\t\t"...
push    offset Buffer   ; Buffer
call    _sprintf
add     esp, 20h
cmp     esi, ebx
jz      loc_401FDE
```

Figura 4.2.2 - 8 Segmento que probablemente ejecute las llamadas de encriptado

Dentro de este fragmento sí que encontramos información interesante. Como vimos en el análisis de cadenas de texto al inicio la clave RSA para la generación de la hash de cifrado está escrita en el propio código. Dado que este segmento comienza copiando esa clave y para luego modificarla, podemos asumir que su función es la de generar la clave de encriptación a partir de esa clave pública RSA.

Tras esta llamada vemos cómo se prepara el mensaje para indicarnos que nuestros archivos han sido encriptados, así como una dirección de correo electrónico. Si seguimos con

el análisis encontraremos varias llamadas a funciones de C + + encargadas de obtener una lista con todos los discos del ordenador infectado.

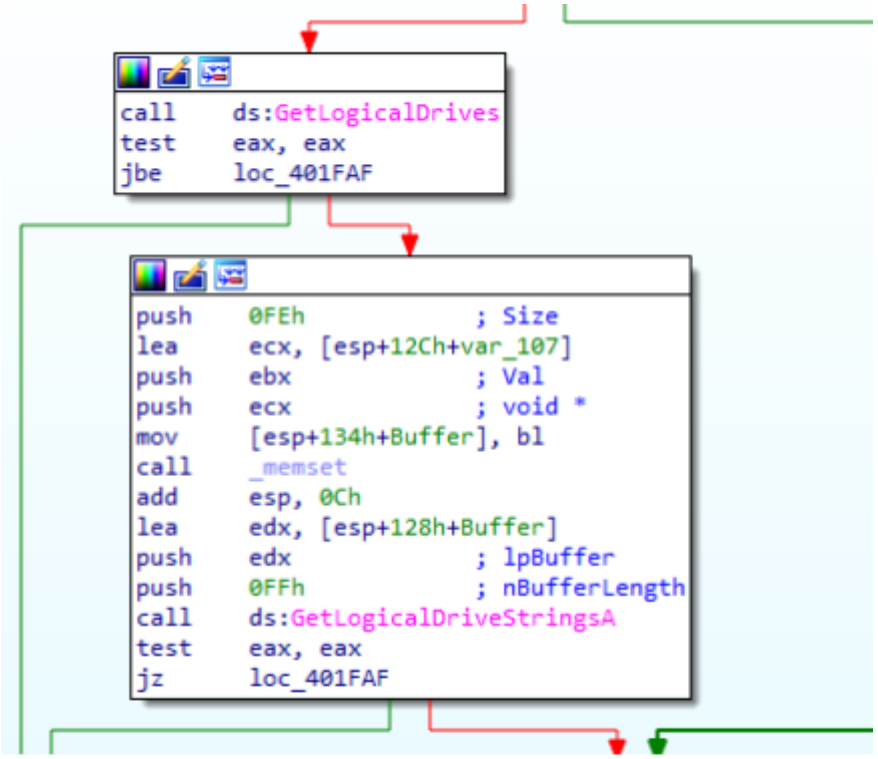


Figura 4.2.2 - 9 Segmento de código encargado de obtener la lista de discos.

Conociendo el funcionamiento de esta muestra de ransomware, podemos suponer que ahora analizará esos discos recorriendo todos los directorios encriptando carpeta a carpeta.

Para confirmar nuestra suposición, si revisamos las llamadas que hace tras obtener los distintos discos encontramos las funciones FindFirstFileA y FindNextFileA, por lo que debe cifrar archivo a archivo

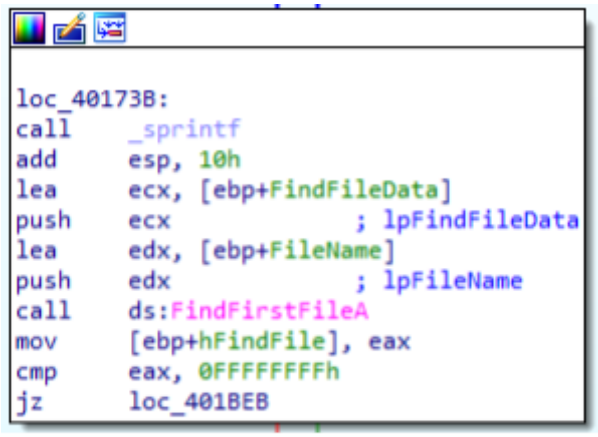


Figura 4.2.2 - 10 Segmento que obtiene el primer archivo de una ruta

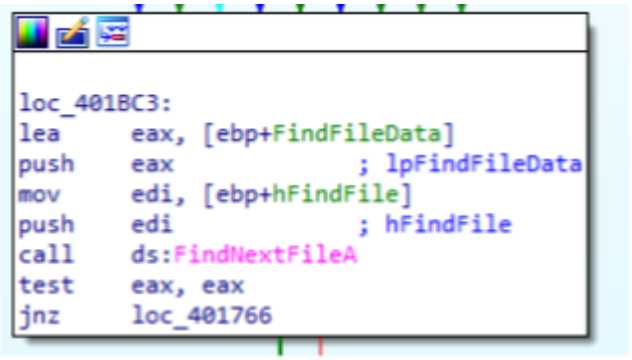


Figura 4.2.2 - 11 Segmento que obtiene el siguiente archivo de una ruta

4.2.3 · Conclusiones del análisis

Dado que una vez encriptados tus archivos no hay forma de desencriptarlos salvo cumpliendo con las condiciones que te imponga el propio malware y ni siquiera eso supone una garantía (en los casos de ataques WannaCry varias víctimas pagaron para recuperar sus archivos y aun así las claves de desencriptación que recibieron no funcionaron, probablemente debido a que los atacantes no sabían con certeza las claves de la encriptación para esa máquina concreta), podemos concluir que la mejor forma de tratar con este malware es la preparación previa. Los puntos más importantes a cubrir son:

- **Mantener el sistema operativo y tu antivirus actualizado**: gracias a los continuos avances analizando y documentando estos ejemplos de malware mantener tus sistemas de seguridad actualizados es la mejor manera de protegerte. Casos como el ransomware DearCry son en su mayoría automáticamente detectados tanto por el propio sistema operativo (mediante windows defender) como por los antivirus que tengamos instalados.
- **Preparar copias de seguridad de los archivos más importantes:** para evitar perder nuestros datos aún en el caso de que tengamos todas las medidas de seguridad actualizadas lo mejor es realizar una copia de seguridad frecuente con los archivos más importantes. De esta forma podemos limpiar el equipo infectado sin perder nada.

5 · Conclusiones:

La intención inicial de este proyecto era la de obtener muestras de código malicioso para ejecutarlas en un entorno seguro y averigüar su funcionamiento. Tras investigar las diferentes técnicas de análisis que existen la idea se mantuvo, pero decidí hacer un mayor hincapié en los sistemas de análisis de las muestras dentro de estos entornos.

La primera conclusión que pude extraer a lo largo del proyecto es que la labor de la comunidad es esencial a la hora de combatir la propagación del malware. La cantidad de recursos que existen sin ánimo de lucro para comparar resultados o buscar información es monumental. A lo largo de este trabajo he encontrado multitud de repositorios, blogs, foros y canales repletos de vídeos en los que discutir acerca de los distintos análisis que se realizan sobre una misma muestra y compartir los resultados. Dentro de este grupo incluyo también los repositorios de malware de acceso gratuito que he podido encontrar, desde Malware Bazaar que ofrece multitud de muestras de malware para su estudio hasta otros que requieren de unas tramitaciones de acceso con las que por desgracia no he podido contar como Contagio o VirusSign.

La idea llevada a cabo con las dos muestras analizadas es la de mostrar distintas facetas del análisis de malware. Por un lado tenemos un ejemplo del que hemos localizado análisis previos mediante la generación de una clave hash pudiendo proceder directamente con el análisis dinámico, mientras que en el segundo caso hemos realizado el análisis estático usando algunas de las herramientas más populares del momento como Capa, Strings e IDA Pro.

La segunda conclusión que he podido extraer del proyecto es que las tecnologías de análisis siguen en plena evolución a día de hoy. Hasta marzo del año 2019 la herramienta predilecta era IDA Pro. Ese mes la NSA lanzó su propio programa desensamblador conocido como Ghidra, que presentaba múltiples mejoras con respecto al software anterior. Esto ha motivado que desde Hex-Rays sacasen múltiples añadidos para IDA que han mejorado la calidad de vida de los analistas. A esto se le suman las múltiples herramientas, gratuitas y de pago, que surgen continuamente para facilitar las diferentes etapas del análisis para acabar de formar un conjunto de herramientas que permiten que, independientemente del nivel técnico o la

experiencia que poseas, puedas adentrarte en el mundo del análisis de malware de una forma rápida y sencilla.

Por último, la conclusión principal que saco tras terminar este proyecto es que independientemente de la experiencia que tengas o los años que lleves trabajando en este entorno, todos los días surgen nuevas lógicas para que el malware supere los análisis existentes y la presión de adelantarse a estas innovaciones es constante. Si los entornos seguros son las máquinas virtuales, el malware encontrará la forma de identificarlas y ocultar su comportamiento. Si el código que se está analizando requiere de una conexión a internet para funcionar buscará la forma de pasar desapercibido en segundo plano hasta que las condiciones para su propagación sean las ideales. La tarea de los analistas sigue siendo increíblemente complicada a día de hoy, por lo que todas las herramientas que se conozcan y todas las técnicas que se empleen siempre serán un elemento vital para mantener los sistemas de seguridad actualizados ante las nuevas amenazas.

En cuanto a futuras mejoras sobre este proyecto, hemos analizado muestras de malware cuyo funcionamiento no depende de la conexión a la red. Una posible mejora sobre el mismo podría ser la de analizar los intercambios de paquetes que lleva a cabo una máquina infectada con un servidor desde el que recibe instrucciones, o monitorizar los intentos de propagación que pueda llevar a cabo el malware atacando a direcciones IP aleatorias como es el caso del gusano Brambul.

Bibliografía:

Artículos web:

E. Gonzalez. (2022, 6 de junio). “Los costes del ransomware superarán los 265.000 en 2031” [Online]. Disponible: https://bitlifemedia.com/2022/06/ransomware-costes/

R. Wittek. (2022, 7 de junio). “Jump in cyber attacks during Covid-19 confinement” [Online] Disponible: https://www.swissinfo.ch/eng/jump-in-cyber-attacks-during-covid-19-confinement/45818794

A. Jeyashankar. (2022, 2 de febrero). “Most Common Malware Obfuscation Techniques” [Online]. Disponible: https://www.socinvestigation.com/most-common-malware-obfuscation-techniques/

J. Channel. (2016, 31 de marzo). “Nowhere to Hide: Three methods of XOR obfuscation” [Online]. Disponible: https://blog.malwarebytes.com/threat-analysis/2013/05/nowhere-to-hide-three-methods-of-xor-obfuscation/

K. Johnson. (2021, agosto) “Top static malware analysis techniques for beginners” [Online]. Disponible: https://www.techtarget.com/searchsecurity/feature/Top-static-malware-analysis-techniques-for-beginners

R. Bellairs. (2020, 10 de febrero). “What Is Static Analysis? Static Code Analysis Overview” [Online]. Disponible: https://www.perforce.com/blog/sca/what-static-analysis

Ring Zero Labs (2017, 9 de septiembre) “Worms Caught In Brambuls” [Online]. Disponible: http://www.ringzerolabs.com/2017/08/worms-caught-in-brambuls.html

P. Tavares (2021, 30 de junio). “DearCry ransomware: How it works and how to prevent it” [Online]. Disponible:

https://resources.infosecinstitute.com/topic/dearcry-ransomware-how-it-works-and-how-to-prevent-it/

Lifars, LLC (2021, 8 de abril). “DearCry Ransomware Malware Analysis and Reverse Engineering” [Online]. Disponible: https://www.lifars.com/knowledge-center/dearcry-ransomware-malware-analysis-and-reverse-engineering/

M. Lechtik (2018, 4 de febrero). “DorkBot: An Investigation” [Online]. Disponible: https://research.checkpoint.com/2018/dorkbot-an-investigation/

McAfee Labs (2017, 9 de marzo). “Analyzing a Fresh Variant of the Dorkbot Botnet” [Online]. Disponible: https://www.mcafee.com/blogs/other-blogs/mcafee-labs/analyzing-fresh-variant-dorkbot-botnet/

S. Ninja (2015, 19 de abril). “Static malware analysis” [Online]. Disponible: https://resources.infosecinstitute.com/topic/malware-analysis-basics-static-analysis/

N. Zerof (2029, marzo). “Ghidra vs. IDA Pro. Strengths and weaknesses of NSA’s free reverse engineering toolkit” [Online]. Disponible: https://hackmag.com/security/nsa-ghidra/

Publicaciones:

Ori Or-Meir, Nir Nissim, Yuval Elovici, Lior Rokach. (2019, 13 de septiembre). “Dynamic Malware Analysis in the Modern Era—A State of the Art Survey” [Online] Disponible: https://dl.acm.org/doi/10.1145/3329786

Amir Afianian, Salman Niksefat, Babak Sadeghiyan, David Baptiste (2018, junio) “Malware Dynamic Analysis Evasion Techniques: A Survey” [Online]. Disponible: https://arxiv.org/pdf/1811.01190.pdf

Maxat Akbanov, Vassilios Vassilakis (2019, abril). “WannaCry Ransomware: Analysis of Infection, Persistence, Recovery Prevention and Propagation Mechanisms” [Online].

Disponible: https://www.researchgate.net/publication/332088162_WannaCry_Ransomware_Analysis_of_Infection_Persistence_Recovery_Prevention_and_Propagation_Mechanisms

Pablo Ramos (2022, enero). “Dorkbot: conquistando Latinoamérica” [Online]. Disponible: https://www.welivesecurity.com/wp-content/uploads/2014/01/dorkbot_conquistando_latinoamerica.pdf

Vídeos de referencia:

D. Mandal (2020, 5 de agosto). “IDA Pro reverse engineering tutorial for beginners” [Online]. Disponible: https://www.youtube.com/playlist?list=PLKwUZp9HwWoDDBPvoapdbJ1rdofowT67z

Bases de datos:

Roman Hüssy. “Malware Bazaar” [Online]. Disponible: https://bazaar.abuse.ch/

Hispasec Sistemas. “VirusTotal” [Online]. Disponible: https://www.virustotal.com/gui/home/upload

Universidad de Alcalá
Escuela Politécnica Superior

Universidad de Alcalá

ESCUELA POLITECNICA SUPERIOR