uc3m | Universidad Carlos III de Madrid

Máster Universitario en Ciberseguridad
2017/2018

Trabajo Fin de Máster

OpenRansim

Íñigo Serrano Salgado

Tutor
Juan Tapiador

Lugar y fecha de presentación prevista

Palabras clave: Ransomware, simulation, security, malware, encryption.
Resumen: En este documento, se va a presentar una herramienta de código abierto que sirve para comprobar la protección de un sistema informático frente a ataques de Ransomware.

Índice

Índice de figuras

Resumen

Hoy en día, uno de los ataques más comunes y eficaces que existen en el mundo de Internet, es el llamado Ransomware. Este tipo de ataques empezó a surgir en 1989 con AIDS Trojan. El Ransomware, se basa en secuestrar los datos de un sistema informático y pedir un rescate por estos. Técnicamente no requiere grandes conocimientos de seguridad para desarrollar un ejecutable, aunque, es cierto, que muchos emplean exploits para la fase de propagación. Lo cual lo convierte en uno de los ataques más efectivos y que suele obtener sus objetivos.

Es, por este motivo, que se ha decidido llevar a cabo este Trabajo de Fin de Máster, con el objetivo de aportar facilidades, para poder así evitar una infección en nuestros equipos y la consecuente remuneración para poder acceder de nuevo a nuestros archivos.

1. Introducción

Como en muchas facetas de la vida real, la posibilidad de ganar dinero de forma fácil y a costa de los demás, ha llegado ya al mundo Online, como no podía ser de otro modo.

Con el paso de los años, han ido apareciendo muchos ciberdelincuentes o, incluso, mafias que están actuando a través de Internet. El secuestro de información, el ciberespionaje, el tráfico de cualquier dato sensible o la venta de piezas de malware, son cada vez más habituales en los tiempos que corren. Debido a esta serie de factores, es de vital importancia que exista, a disposición del usuario una ayuda como la presentada en este ensayo.

Figura 1: Imagen que aparece en un ordenador infectado

De todas las opciones que existen, los ataques de Ransomware, quizás sean los más rentables en términos económicos. Su único objetivo es secuestrar los datos de un ordenador para poder pedir un rescate por ellos. El uso de la criptografía y de las nuevas formas de pago, que garantizan el anonimato, son fundamentales para realizar un buen ataque.

El primer Ransomware que apareció, fue AIDS Trojan en 1989. Este malware escondía y cifraba los nombres de todos los ficheros que se encontrasen en el disco

C. Esto hacia que el ordenador se quedase inservible, teniendo que pagar 189$ a una oficina de correos en Panamá.

En 2011 apareció el primer Trojan, que se aprovechaba de la moda de los pagos anónimos como medio de pago del rescate de los datos. Actualmente, el uso de medios de pago anónimos es algo muy común, aunque no siempre es fácil para el atacante recuperar el dinero.

Según las últimas tendencias de Ransomware, el uso de criptografía simétrica es la forma más empleada para hacer que los datos no puedan recuperarse, a menos que se tenga la contraseña. Pero esto no siempre ha sido así, hubo un tiempo en el que la idea principal no era cifrar los datos de un ordenador, sino bloquearlo y hacer que su uso fuera limitado. Estos son los llamados "Lockers", que se valen de exploits y ciertas vulnerabilidades del sistema operativo anulando cualquier interacción con la máquina afectada.

Históricamente, este tipo de ataques ha estado siempre dirigido a ordenadores personales, por lo que los ordenadores con un sistema operativo de la familia Microsoft, eran más propensos a este tipo de ataque. Con la llegada de los Smartphone y la capacidad que tienen para ser ordenadores personales portátiles, en 2014 salió un Ransomware centrado en el sistema operativo Android.

El desarrollo del proyecto en cuestión, se centrará en la presentación de una nueva herramienta creada para facilitar la protección de sistemas informáticos frente a ataques de Ransomware. La idea en la que se basará este trabajo, surgió al descubrir algo similar en el mercado. Una aplicación llamada Ransim. Ésta, ejecuta una serie de escenarios, los más comunes, intentando asemejarse a un programa de Ransomware real.

El objetivo, es llevar a cabo los procedimientos necesarios para que, dicha herramienta, sea capaz de realizar la misma función que la previamente descubierta, aunque con una diferencia, y es que esta nueva herramienta presentaría un código abierto con posibilidad de ejecutar escenarios ad hoc. Por este motivo, el nombre elegido, ha sido OpenRansim. OpenRansim podrá encontrarse en la siguiente URL: Github. Al tratarse de una aplicación de código abierto, cualquier persona con un cierto interés en el tema, podrá colaborar en su desarrollo, lo cual, será clave y podrá favorecer al correcto funcionamiento de la misma.

1.1. Motivación del proyecto

Parece ser, que nos encontramos en un nuevo mundo, el mundo de la digitalización. En este nuevo mundo, uno de los elementos más valiosos, ha pasado a ser la disposición de datos y su control. Por este motivo, el Ransomware, se ha convertido, en la opción con más posibilidades de hacer negocio para los ciberdelincuentes.

A comienzos del año 2017, apenas se conocía la existencia de este tipo de ataques, hasta que saltó a la fama WannaCry. En Mayo del mismo año, se lanzó un

ataque masivo que afectó a nivel mundial, incluyendo grandes empresas españolas. El 27 de Junio de 2017 salió a la luz otro Ransomware llamado NotPetya. Este último también se propagó a escala mundial afectando a Ucrania, Estados Unidos e Italia, principalmente.

Debido al notable aumento de este tipo de ataques, la existencia de una herramienta que pueda medir, de forma fiable, como de protegido se encuentra un ordenador frente a este tipo de amenazas, pasa a convertirse en un recurso vital para compañías de todo el mundo. OpenRansim, ha sido desarrollado, para que cualquier persona, sea capaz de cubrir sus necesidades de protección ante posibles ataques de Ransomware.

1.2. Contribuciones

Como todo proyecto final, es necesario que presente ciertas aportaciones, pudiendo ser mejoras de algo ya existente o totalmente nuevas en el sector. Para este proyecto se ha puesto el foco en presentar nuevas soluciones que ayuden, en la medida de lo posible, a que el mundo Online y de los ordenadores tenga más seguridad. Estas son las nuevas características que presenta OpenRansim:

- Simulador de Ransomware - Como aportación principal se encuentra OpenRansim, un simulador de Ransomware que verifica de forma rápida posibles infecciones sin afectar los datos del equipo.
- Modularidad - Este proyecto se ha desarrollado en módulos, haciendo que sea muy fácil de implementar nuevos snippets para los escenarios.
- Código abierto - El desarrollo de OpenRansim se ha hecho bajo la Licencia de código abierto GPL v3.0, permitiendo su uso a cualquier usuario, en cualquier ámbito y de forma gratuita.
- Fácil extensión - Gracias a la modularización de las operaciones más habituales, es muy sencillo reutilizar el código y crear un nuevo escenario sin necesidad de añadir nuevas funcionalidades.
- Optimización - Al estar implementado en Golang, permite un desarrollo muy rápido a la vez que un gran rendimiento, haciendo que se pueda ejecutar cualquier escenario en segundos, empleando los parámetros por defecto.

En la Sección 2 se introduce OpenRansim, cuales son sus funcionalidades y características. Posteriormente en la Sección 3 se describe cuales son las pruebas que se han realizado y los resultados obtenidos. En la Sección 4 se mencionan proyectos relacionados y las diferencias que tienen con éste. Finalmente, en la Sección 5 se concluye el trabajo y en los apéndices A y B se proporciona una descripción de la planificación y presupuesto del proyecto, respectivamente.

2. OpenRansim

2.1. Introducción

OpenRansim es una herramienta de línea de comando, desarrollada en Golang bajo la licencia GPL 3.0 de uso libre. La puesta en marcha, se ha llevado a cabo, mediante la utilización de 10 escenarios básicos, aunque su diseño, basado en módulos, permite la creación de escenarios nuevos, haciendo que sea personalizable.

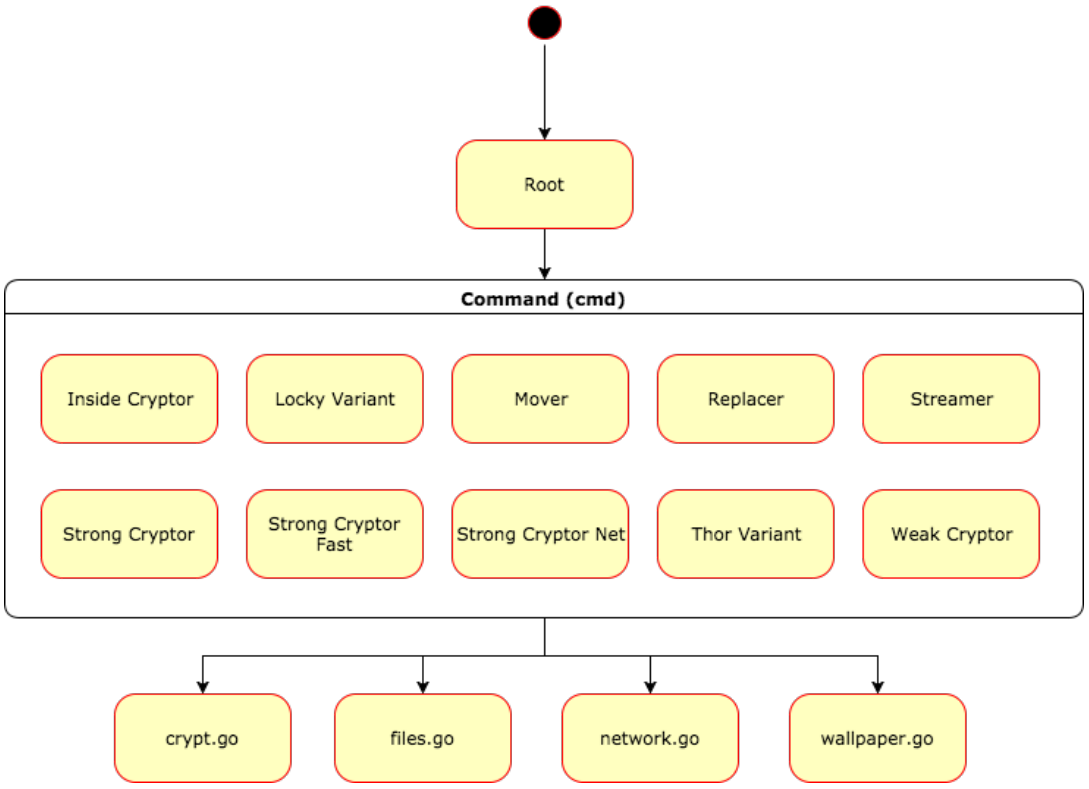

Figura 2: Clases y módulos de OpenRansim

En la Figura 2 se muestra un diagrama de como está diseñado OpenRansim. En este programa se pueden apreciar tres bloques distintos: Root, Command y los ficheros externos de Golang.

- Root - Este paquete representa el Framework empleado para crear CLI en Golang. Este es el punto de entrada de la herramienta y quien va a llamar a los diferentes comandos.
- Command - En este conjunto se encuentran los escenarios implementados. Estos, solo contienen el flujo de ejecución, pues la lógica se encuentra en el último grupo de clases.
- Ficheros externos - Estos archivos ejecutable de Golang son los que contienen los métodos que, por ejemplo, crean directorios, listan ficheros, generan claves o hacen llamadas HTTP.

2.2. Pre task & Post task

Para comprobar que un escenario se ha ejecutado correctamente, y poder así dictaminar si existe una posibilidad de infección o no, se han añadido al desarrollo de la herramienta, las llamadas: pre-task y post-task. Antes de empezar a ejecutar un escenario, la herramienta crea una carpeta especifica donde se van a realizar las pruebas definidas por cada escenario. De esta forma se ejecuta la prueba bajo un entorno controlado sin que afecte a ningún archivo real.

Una vez ejecutado un escenario, es necesario comprobar si éste se ha completado correctamente o si, por el contrario, el sistema operativo ha interrumpido la ejecución, detectando y bloqueando, de esta manera, la posible acción maliciosa.

2.3. Escenarios

Para esta primera versión de OpenRansim se han llevado a cabo una serie de escenarios, que intentan simular las ejecuciones de varios tipos de Ransomware. Para ello, se han estudiado cuales son las formas de funcionar de las familias de Ransomware más representativas. Estos son los 10 escenarios explicados obviando las fases de pre-task & post-task:

1. InsideCryptor – Cifra los datos y sobrescribe los archivos originales.
   a) Se va leyendo archivo por archivo.
   b) Sobreescribe el mismo ficheros con los datos cifrados.
2. LockyVariant – Simula una de las incontables variables del Ransomware Locky.
   a) Se va leyendo archivo por archivo.
   b) Crea un nuevo fichero con el mismo nombre y la extensión '.locky'
   c) Escribe los datos cifrados del ficheros original en el nuevo fichero.
   d) Elimina el fichero original.
3. Mover – Cifra los datos en una carpeta diferente a la original y elimina los ficheros originales.
   a) Se va leyendo archivo por archivo.
   b) Crea un nuevo fichero con el mismo nombre pero en una carpeta distinta.
   c) Escribe los datos cifrados del ficheros original en el nuevo fichero.
   d) Elimina el fichero original.
   e) Elimina la carpeta original.
4. Replacer – Reemplaza el contenido de los archivos.
   a) Escribe cadenas aleatorias en los ficheros. (No realiza ninguna operación criptológica).

5. Streamer – Cifra todos los datos y los agrupa en un único fichero.
    a) Se va leyendo archivo por archivo.
    b) Cifra los datos de los ficheros y guarda el resultado en memoria.
    c) Cuando ha leido todos los ficheros crea uno en el que vuelca el contenido cifrado de todos los ficheros.
    d) Elimina todos los ficheros originales.
6. StrongCryptor – Cifra los datos y borra los ficheros originales de forma segura.
    a) Se va leyendo archivo por archivo.
    b) Crea un nuevo fichero con el mismo nombre y la extensión '.copy'
    c) Escribe los datos cifrados del fichero original en su copia.
    d) Realiza 30 iteraciones escribiendo información cifrada aleatoria en el archivo original.
    e) Borra los archivos originales.
7. StrongCryptorFast – Cifra los datos y borra los archivos originales.
    a) Se va leyendo archivo por archivo.
    b) Crea un nuevo fichero con el mismo nombre y la extensión '.copy'
    c) Escribe los datos cifrados del fichero original en su copia.
    d) Borra los ficheros originales.
8. StrongCrytptorNet – Cifra los datos, borra los ficheros originales y simula una conexión HTTP.
    a) Se va leyendo archivo por archivo.
    b) Crea un nuevo fichero con el mismo nombre y la extensión '.copy'
    c) Escribe los datos cifrados del fichero original en su copia.
    d) Elimina los archivos originales.
    e) Realiza peticiones HTTP a un servidor mock con parte de la información cifrada.
9. ThorVariant – Simula una de las incontables variables del Ransomware Thor.
    a) Se va leyendo archivo por archivo.
    b) Crea un nuevo fichero con el mismo nombre y la extensión '.thor'
    c) Escribe los datos cifrados del fichero original en su copia.
    d) Borra el archivo original.
    e) Cambia el wallpaper del ordenador.
10. WeakCryptor – Utiliza un cifrado débil para cifrar los datos y elimina los archivos originales.

a) Se va leyendo archivo por archivo.

b) Crea un nuevo fichero con el mismo nombre y la extensión '.copy'

c) Escribe los datos cifrados del fichero original en su copia.

d) Borra los archivos originales.

2.4. Implementación

Para el desarrollo de OpenRansim se ha utilizado el lenguaje Golang. Éste, desarrollado por Google, tiene las ventajas de C en tiempo de ejecución y las de Python, en velocidad de desarrollo. Para poder crear un CLI (Command Line Interface) se ha usado el Framework Cobra. Gracias a Cobra es muy fácil e intuitivo crear CLIs totalmente personalizables con comandos y flags de cualquier tipo. Como se observó en la Figura 2 la arquitectura de OpenRansim se divide en tres bloques. Los dos más importantes son el paquete cmd y el conjunto de ficheros externos.

El bloque cmd contiene todos los comandos que se pueden ejecutar. Estos comandos siguen una forma de implementarse definida por Cobra. Es por eso, que cada comando tiene una fase preliminar y otra posterior a la lógica propia del comando. Estos solo definen una serie de llamadas a funciones con diferentes parámetros de entrada para que se pueda crear un nuevo escenarios. Estas funciones se encuentran implementadas en el grupo de ficheros externos.

Estos, que actúan como librerías, contienen funciones muy genéricas y totalmente reutilizables que permiten crear flujos y escenarios muy variados. Dentro de estos ficheros encontramos los siguientes:

- crypt - Dentro de esta libreria se pueden encontrar un conjunto de operaciones relacionadas con criptografía, es por eso que se usa una librería especial para generar números aleatorios. En este fichero tenemos las siguientes funciones:
  - Cifrar - Esta función recibe como argumentos un texto en plano, una clave y un algoritmo de cifrado, a día de hoy solo se usan DES y AES con diferentes longitudes de claves. Devuelve un texto cifrado.
  - Descifrar - Esta función tiene la misma firma que la anterior pero devolviendo texto en plano.
  - Generar clave RSA - Esta función genera un par de claves RSA de las cuales se extrae la clave que se usará en cada escenario, si procede, para cifrar los archivos.
  - Generar números aleatorios - Ya el propio nombre de la función explica su lógica. Se puede usar para rellenar ficheros, para poner nombres aleatorios o para la función anterior.
- files - En esta clase se encuentran todas las funciones relacionadas con el tratamiento de ficheros.

  - Crear directorio - Crea una carpeta en la ruta que se le pase por argumento.
  - Comprobar que existe un directorio - Comprueba la existencia de una carpeta en la ruta que se le pase por argumento.
  - Crear ficheros - Crea N ficheros en la carpeta seleccionada siempre y cuando se tengan permisos.
  - Crear cadenas aleatorias - Genera N caracteres aleatorios que luego se concatena.
  - Obtener información de ficheros - Obtiene toda la información relacionada con los archivos de una carpeta.
  - Escribir en fichero - Escribe texto en un fichero.
  - Leer de fichero - Devuelve todo el contenido de un fichero o un error.
  - Eliminar fichero - Borra de forma permanente un archivo.
  - Eliminar directorio - Elimina una carpeta y todos los ficheros que tuviera dicha carpeta.

- network - Este fichero solo contiene la función de realizar peticiones HTTP.
- wallpaper - Pasándole como parámetro una URL, cambia el fondo de pantalla por la imagen que se encuentra en esa URL.

2.5. BYOS, Build Your Own Scenario

Debido a su gran modularidad, OpenRansim está diseñado para que pueda ser fácilmente extendido y/o mejorado. Cuando se habla de crear un nuevo escenario, es necesario saber un par de características de la herramienta que hacen que sea muy sencillo.

Como ya se ha comentado anteriormente, un escenario se basa en ejecutar las funciones que ya existen con los argumentos adecuados. Por lo que para construir un nuevo flujo, es posible que eligiendo las funciones adecuadas y el orden deseado se pueda crear un nuevo test. En caso de que las funciones ya existentes no sean suficientes, es posible crear unas nuevas en los ficheros ya existentes o nuevos ficheros. Gracias al diseño de OpenRansim, cualquier código que se añada a los ficheros externos o librerías van a estar disponibles para ser usadas en cualquier comando, sin necesidad de realizar nuevas importaciones ni precargas.

3. Validación experimental

En esta sección se van a tratar varios puntos relacionados con el tiempo de ejecución y con los resultados obtenidos de las ejecuciones de los diferentes escenarios.

En cuanto a tiempos de ejecución, OpenRansim, al estar desarrollado en Go, es un código muy optimizado y que consume muy poca memoria y CPU comparado con otros lenguajes. Si bien es cierto que obtener un tiempo medio de ejecución de la herramienta aporta mucho valor, la diferencia de escenarios a ejecutar podrían falsear estos tiempos. Es por eso que se ha considerado más oportuno medir los tiempos por cada escenario. Estos son los resultados:

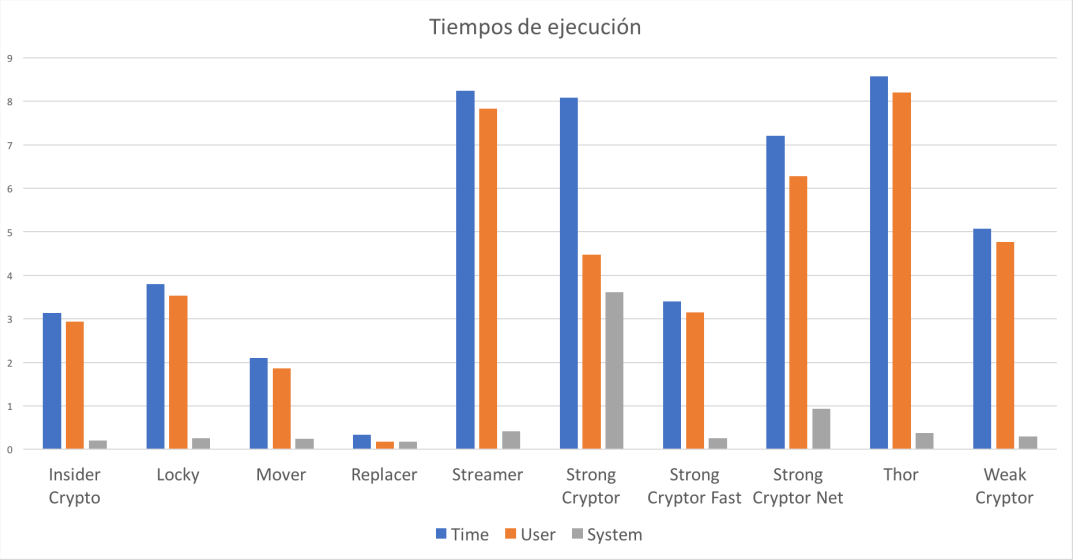

Figura 3: Tiempos de ejecución por escenarios

Según se muestra en la Figura 3 los tiempos no superan los 9 segundos y solo un escenario tarda menos de un segundo. Las condiciones, que son comunes para todos los escenarios son las mismas, en este caso es el número de ficheros que maneja cada flujo, 500.

Se puede observar como los escenarios que más operaciones criptográficas realizan son los que más tiempo requieren. Aunque cabe destacar, el escenario Streamer, uno de los que más tarda, pues no contiene gran cantidad de instrucciones de cifrado. Debido a que requiere mucha memoria para guardar todos los datos cifrados es el escenario que más memoria consume.

Por otro lado, Replacer es el caso que menos tarda. Esto se debe a que no necesita generar ninguna clave ni cifrar la información contenida en los diferentes ficheros.

La otra pata de esta sección es la de interpretar los resultados después de lanzar cada uno de los escenarios. Algo muy interesante habría sido ejecutar todos los escenarios en diferentes sistemas operativos y con distintas medidas de protección,

pero se ha considerado que ese análisis podría ser otro proyecto de fin de Máster. Es por eso que solo se van a mostrar los resultados obtenidos en un Mac personal sin ningún tipo de protección anti-Ransomware, a excepción de las que vienen por defecto en Mac OS X.

Insider Crypto	No
Locky	Si
Mover	No
Replacer	No
Streamer	Si
Strong Cryptor	Si
Strong Cryptor Fast	No
Strong Cryptor Net	No
Thor	No
Weak Cryptor	No

Tabla 1: Resultados de los escenarios

Según se aprecia en la Tabla 1, solo tres de los diez flujos son detectados y es el sistema operativo quien corta la ejecución. Comparándolo con la Figura 3 son estos tres mismos casos los que más tiempo tardan en ejecutarse, más operaciones criptográficas realizan o más memoria consume. Esto indica que Mac OS X es capaz de detectar e interrumpir ciertos procesos que debe de considerar como sospechosos.

4. Trabajos relacionados

5. Conclusión

No siempre se devuelven los archivos después de pagar

A. Planificación del proyecto

B. Presupuesto