Análisis de malware. WannaCry

Álvaro García Jaén

Darío Abad Tarifa

Índice

Introducción

Un ramsonware es un tipo de malware que secuestra la información de la víctima y amenaza con publicarla o bloquear su acceso de forma permanente mediante su cifrado a menos que su rescate sea pagado. Dicho pago se hace normalmente a través de criptodivisas como Bitcoin, lo que dificulta el rastreo de los delincuentes.

El ataque sucedido el 12 de mayo de 2017 que tuvo como protagonista al ramsonware WannaCry ha sido descrito como sin precendentes y afectó a más de 230.000 computadoras de más de 150 países, incluyendo compañías importantes como Telefónica en España, FedEx e incluso departamentos del servicio nacional de salud de Gran Bretaña.

Es común encontrar descripciones que definen este malware como un criptogusano, esto es debido al comportamiento que refleja y lo que, probablemente, propició el gran alcance del ataque a nivel mundial. Esta característica es algo poco común respecto a otros ramsonware, ya que lo normal es que los ataques de este tipo se lleven a cabo mediante troyanos que llegan a los equipos de las víctimas a través de campañas de phishing. Sin embargo WannaCry se expandía de forma automáticamente por los equipos de la red tras la infección de un equipo.

Dicha expansión se llevaba a cabo mediante el uso del exploit *EternalBlue*, que fue revelado por el grupo *The Shadow Brokers* el 14 de abril (1 mes antes del ataque). Dicho exploit se aprovecha de la vulnerabilidad *MS17-010* del protocolo *Server Message Block*. El detalle importante es que Microsoft liberó un parche para esta vulnerabilidad el 14 de marzo (2 meses antes del ataque), sin embargo, es probable que debido al poco margen de tiempo y la falta de concienciación dicha actualización no fuera aplicada en la mayoría de equipos. Una actualización que además no cubría versiones de Windows XP.

Aunque es un ataque de gran gravedad, las consecuencias podrían haber sido peores si no se hubiera descubierto el *kill-switch* que detuvo la propagación o si hubiera sido un ataque dirigido a infraestructuras críticas, como centrales nucleares o eléctricas.

En este documento se tratará el contexto que acompañaba al suceso, se documentará el funcionamiento del malware, se hablará del impacto que tuvo y la perspectiva actual que se tiene sobre los ataques ramsonware. Además se comentarán las diferentes teorías sobre la, poco clara, autoría del ataque y se realizará un análisis técnico de una muestra del malware mediante técnicas de ingeniería inversa.

Antecedentes

Funcionamiento

Impacto

Atribución del ataque

Análisis técnico de una muestra

Para el desarrollo de esta parte, se ha realizado un **análisis estático** de una muestra del malware que se encuentra en el repositorio the Github *TheZoo*. Las principales herramientas usadas son:

- **Ghidra**: Framework de ingeniería inversa desarrollado por la NSA.
- **wrestool**: Herramienta para la extracción de recursos en ejecutables Windows perteneciente al paquete *icoutils*.

El análisis se ha llevado a cabo en un una máquina virtual aislada con sistema operativo *Kali Linux 2019.3*. Aunque es un malware que no funciona en Linux, se considera una buena práctica realizar análisis de malware en equipos aislados y que no tengan un uso compartido. Esto es para evitar infectar un sistema que pueda tener información que nos resulte importante y evitar así contagios en otros equipos de la red local.

El procedimiento que se ha llevado a cabo en el análisis consiste resumidamente en identificar los puntos de entrada del ejecutable, a partir del `main`, inferir el funcionamiento de las funciones usadas por orden y una a una de forma recursiva. Tras llevar a cabo el renombrado de las variables existentes en la función y haber comentado los bloques con el fin de aclarar el flujo de ejecución, se ha dado un nombre característico a cada función que permita conocer la tarea que realiza de forma clara.

Se ha determinado como objetivos de este análisis dos sencillos puntos que permitan ilustrar de forma esquemática y empírica el funcionamiento del malware:

- Encontrar el famoso *kill-switch*, relevante sobretodo para la expansión del malware.
- Realizar el desempaquetado del ejecutable con el fin de extraer los recursos que contiene para poder diferenciar los módulos que componen el malware y como interactúan ente si.

Dicho esto, comenzamos el análisis cargando el *primer* ejecutable en Ghidra. Se trata de un PE32, es decir, un ejecutable para sistemas de 32 bits. Conociendo el funcionamiento de *Portable Executable* debemos identificar la llamada a la función `main` situada al final de la función `entry`. Una vez nos hemos desplazado al `main`, comenzamos el renombrado de variables y funciones para clarificar el funcionamiento de esta primera parte.

```
1
2  int WinMain(HINSTANCE hInstance,HINSTANCE hPrevInstance,PWSTR pCmdLine,int nCmdShow)
3
4  {
5    HINTERNET hInternet;
6    HINTERNET HINTERNET_return;
7    int i;
8    char *dom_killswitch;
9    char *dom_killswitch_copy;
10   char dom_buffer [14];
11
12   i = 14;
13                     /* Dominio: "http://www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com" */
14   dom_killswitch = s_http://www.iuqerfsodp9ifjaposdfj_004313d0;
15   dom_killswitch_copy = dom_buffer;
16   while (i != 0
17                     /* strcpy(dom_killswitch_copy,dom_killswitch,14); */) {
18     i = i + -1;
19     *(undefined4 *)dom_killswitch_copy = *(undefined4 *)dom_killswitch;
20     dom_killswitch = dom_killswitch + 4;
21     dom_killswitch_copy = dom_killswitch_copy + 4;
22   }
23   *dom_killswitch_copy = *dom_killswitch;
24   InternetOpenA((LPCSTR)0x0,1,(LPCSTR)0x0,(LPCSTR)0x0,0);
25   HINTERNET_return = InternetOpenUrlA(hInternet,dom_buffer,(LPCSTR)0x0,0,0x84000000,0);
26                     /* Si la conexión falla */
27   if (HINTERNET_return == (HINTERNET)0x0) {
28     InternetCloseHandle(hInternet);
29     InternetCloseHandle(0);
30     CRYPTOR_real_entry();
31     return 0;
32   }
33                     /* Si no falla, el programa termina */
34   InternetCloseHandle(hInternet);
35   InternetCloseHandle(HINTERNET_return);
36   return 0;
37 }
38
```

Figura 1: Función main depurada

Sin habernos demorado más de 10 minutos, hemos encontrado el *kill-switch* encontrado por Marcus Hutchins, con el que paró el ataque en su día y que, según él, fue de forma accidental.

Como vemos en la figura, tenemos una variable con la URL al dominio www[.]iuqerfsodp9ifjapos dfjhgosurijfaewrwergwea[.]com que es copiada a un buffer e inmediatamente después se inicia

una conexión y se hace una petición a dicha URL. Después, se comprueba si ha encontrado el dominio o no. Si no lo encuentra, es decir, la conexión falla, el malware continúa la ejecución a través de la función que ha sido llamada `CRYPTOR_REAL_ENTRY()`. Si consigue la conexión, el programa termina su ejecución.

Continuamos el análisis en la función definida como el punto de entrada real del malware. Una vez se ha depurado la función, se puede ver que se comprueba el número de argumentos y que si éste es uno, se continua por una función nombrada por el analista como `cocinar_wannacry()`.

```
1
2  void CRYPTOR_real_entry(void)
3
4  {
5    int *argc;
6    SC_HANDLE hSCManager;
7    SC_HANDLE hSCObject;
8    SERVICE_TABLE_ENTRYA SStack16;
9    undefined4 uStack8;
10   undefined4 uStack4;
11
12   GetModuleFileNameA((HMODULE)0x0,(LPSTR)&path_ejecutable,0x104);
13   argc = (int *)__p___argc();
14   if (*argc < 2) {
15     cocinar_wannacry();
16     return;
17   }
18   hSCManager = OpenSCManagerA((LPCSTR)0x0,(LPCSTR)0x0,0xf003f);
19   if (hSCManager != (SC_HANDLE)0x0) {
20     hSCObject = OpenServiceA(hSCManager,s_mssecsvc2.0_004312fc,0xf01ff);
21     if (hSCObject != (SC_HANDLE)0x0) {
22       FUN_00407fa0(hSCObject,0x3c);
23       CloseServiceHandle(hSCObject);
24     }
25     CloseServiceHandle(hSCManager);
26   }
27   SStack16.lpServiceName = s_mssecsvc2.0_004312fc;
28   SStack16.lpServiceProc = (LPSERVICE_MAIN_FUNCTIONA)&LAB_00408000;
29   uStack8 = 0;
30   uStack4 = 0;
31   StartServiceCtrlDispatcherA(&SStack16);
32   return;
33 }
34
```

Figura 2: Punto de entrada real

Es una función bastante simple que contiene únicamente la llamada a otras dos funciones, cuyo análisis completo no se va a incluir en este documento. En definitiva, la primera función, crea un servicio de nombre «mssecvc2.0» que será usado en etapas posteriores por la componente gusano del malware. La segunda función, extrae un recurso (1831) del ejecutable, nombrado *tasksche.exe*, y lo ejecuta.

```
1
2  undefined4 cocinar_wannacry(void)
3
4  {
5                      /* Como resultado de esta funcion, se creara el servicio: mssecsvc2.0 y se
6                         extraera el recurso que funciona de crypter en el archivo tasksched.exe */
7    crear_servicio_wannacry();
8    recurso_1831_a_tasksche.exe();
9    return 0;
10 }
```

Figura 3: *Cocinar* WannaCry

Para extraer dicho recurso nosotros mismos, se hace uso de la herramienta *wrestool* como se ilustra en la figura. Como se puede comprobar, se trata de otro ejecutable que vamos a cargar en Ghidra para continuar el análisis.

```
root@kali:wannacry# wrestool wannacry
--type='R' --name=1831 --language=1033 [offset=0x3100a4 size=3514368]
--type=16 --name=1 --language=1033 [type=version offset=0x66a0a4 size=944]
root@kali:wannacry# wrestool --name=1831 -x -R wannacry > 1831_CRYPTER.bin
root@kali:wannacry# file 1831_CRYPTER.bin
1831_CRYPTER.bin: PE32 executable (GUI) Intel 80386, for MS Windows
root@kali:wannacry#
```

Figura 4: Extracción del recurso 1831

Cargado el ejecutable, repetimos el procedimiento renombrando variables y funciones de la función `main` obteniendo como resultado lo que aparace en la figura.

```
29  GetModuleFileNameA((HMODULE)0x0,&filename,520);
30  randomstr_generator((char *)&lpMultiByteStr_0040f8ac);
31  argc = (int *)__p___argc();
32                      /* Si ejecuta con un argumento */
33  if (*argc == 2) {
34    stri_/i = s_/i_0040f538;
35    argv = (char ***)__p___argv();
36                      /* strcmp(argumento, "/i") */
37    arg_cmp = strcmp((*argv)[1],stri_/i);
38    if ((arg_cmp == 0) && (uVar1 = crea_directorio_oculto((wchar_t *)0x0), uVar1 != 0)) {
39      CopyFileA(&filename,s_tasksche.exe_0040f4d8,0);
40      DVar2 = GetFileAttributesA(s_tasksche.exe_0040f4d8);
41      if ((DVar2 != 0xffffffff) && (arg_cmp = crear_arrancar_servicio_tasksche(), arg_cmp != 0)) {
42        return 0;
43      }
44    }
45  }
46  stri_/i = strrchr(&filename,0x5c);
47  if (stri_/i != (char *)0x0) {
48    stri_/i = strrchr(&filename,0x5c);
49    *stri_/i = '\0';
50  }
51  SetCurrentDirectoryA(&filename);
52                      /* PERSISTENCIA */
53  crear_entradas_registro(1);
54                      /* Extrae recursos necesarios (2058) para la ejecucion de la GUI y el cifrado de
55                         archivos. Zip con contraseña "WNcry@2ol7" */
56  extraer_recursos((HMODULE)0x0,password_recursos);
57  btc_addr();
58  FUN_00401064(s_attrib_+h_._0040f520,0,(LPDWORD)0x0);
59  FUN_00401064(s_icacls_._/grant_Everyone:F_/T_/C_0040f4fc,0,(LPDWORD)0x0);
60  arg_cmp = FUN_0040170a();
61  if (arg_cmp != 0) {
62    FUN_004012fd();
63    arg_cmp = FUN_00401437((int)local_6e8,(LPCSTR)0x0,0,0);
64    if (arg_cmp != 0) {
65      local_8 = 0;
66      psVar3 = (short *)FUN_004014a6((int)local_6e8,s_t.wnry_0040f4f4,&local_8);
67      if (((psVar3 != (short *)0x0) &&
68          (argc = (int *)FUN_004021bd(psVar3,local_8), argc != (int *)0x0)) &&
69         (pcVar4 = (code *)FUN_00402924(argc,s_TaskStart_0040f4e8), pcVar4 != (code *)0x0)) {
70        (*pcVar4)(0,0);
71      }
72    }
73    FUN_0040137a();
74  }
```

Figura 5: Función main 1813

De forma resumida, este ejecutable, primero, va a crear una carpeta oculta generando el nombre a partir del hostname y va crear/arrancar el servicio que se ha comentado en la etapa anterior para asegurar la persistencia. Después, va a crear entradas en el registro que apunten a los ejecutables malicioso para asegurar la persistencia también y va a extraer una segunda ronda de recursos. Estos recursos están comprimidos en un zip con la contraseña «WNcry@2ol7» y contiene imágenes, textos, traducciones y claves para el cifrado de archivos, además de otros ejecutables. Estos ejecutables serán ejecutados desde hebras distintas y tienen que ver con la

GUI que se presenta a la víctima, utilidades de limpieza de archivos y la expansión del malware mediante EternalBlue.

Por último, se van a extraer dichos recursos para completar el despezado del malware y poder señalar cada módulo o parte y cómo interactúan entre sí.

```
root@kali:2058# file *
b.wnry:     PC bitmap, Windows 3.x format, 800 x 600 x 24
c.wnry:     data
msg:        directory
r.wnry:     ASCII text, with CRLF line terminators
s.wnry:     Zip archive data, at least v1.0 to extract
taskdl.exe: PE32 executable (GUI) Intel 80386, for MS Windows
taskse.exe: PE32 executable (GUI) Intel 80386, for MS Windows
t.wnry:     data
u.wnry:     PE32 executable (GUI) Intel 80386, for MS Windows
```

Figura 6: Extracción recurso 2058

Los archivos son los siguientes:

- *b.wnry*: Imagen que el malware coloca como fondo de pantalla en la que da indicaciones a la víctima.
- *c.wnry*: Contiene información sobre dominios TOR con los que se comunica el malware.
- *msg/*: Contiene las traducciones del texto que se presenta a la víctima con las instrucciones para recuperar la información.
- *r.wnry*: Más texto informativo para la víctima.
- *s.wnry*: Zip que contiene archivos relacionados con TOR y la comunicación con el C&C.
- *taskdl.exe*: Utilidad de limpieza que el malware utiliza.
- *taskse.exe*: Ejecutable maliciosos usado en la etapa de expansión que explora la red en busca de equipos vulnerables.
- *t.wnry*: Una DLL cifrada que es la que, en realidad, desempeña el cifrado de los archivos del sistema.
- *u.wnry*: Ejecutable que muestra la GUI a la víctima con la cuenta atrás y las instrucciones para el pago del rescate

Por tanto, el malware ejecutaría dos etapas previas al cifrado de archivos en las que realiza acciones que aseguran la persistencia del malware en el sistema y despliega los recursos empaquetados en lugares específicos que después son ejecutados para cifrar los archivos (*t.wnry*), informar a la víctima del secuestro (*u.wnry*) y expandirse a otros equipos mediante el exploit

EternalBlue (*taskse.exe*). Este último ejecutable se comunicaría con el servicio creado en una etapa anterior, llamado *mssecvc2.0*, que intentaría enumerar sesiones RDP activas en busca de nuevos equipos.

Conclusión

Bibliografia

- WannaCry Ransomware Attack. Wikipedia
- WannaCry Malware Profile. Fireeye
- WannaCry Ransomware: Analysis of Infection, Persistence, Recovery Prevention and Propagation Mechanisms.