
Análisis de malware. WannaCry

Seguridad en Sistemas Operativos

Darío Abad Tarifa - darioabta@correo.ugr.es

Álvaro García Jaén - alvarogj@correo.ugr.es

Índice

Introducción	3
Antecedentes	4
Funcionamiento del WannaCry	4
Análisis técnico de una muestra	5
Impacto	11
Atribución del ataque	11
Conclusión	12
Bibliografía	13

Introducción

Un ransomware 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 ransomware WannaCry ha sido descrito como sin precedentes 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 ransomware, 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. Llamamos *kill-switch* al método que tiene un malware de parar su propagación. En el análisis qué hacía el WannaCry para saber si debía propagarse o no.

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 ransomware. 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

El término ransomware se volvió más y más conocido gracias al famoso WannaCry. Sin embargo, no debemos olvidar que el WannaCry es uno de tantos ransomware. En este punto veremos otros muchos que hubo y como fueron evolucionando hasta llegar a ser uno de los malware más temidos por muchas empresas.

Comenzamos nuestro viaje en 1989, cuando el doctor Joseph Popp, investigador del SIDA, difundió un troyano haciéndolo pasar por una aplicación que ayudaba a determinar el riesgo de una persona a contraer dicha enfermedad, por lo cual esta claro que los más perjudicados por este ataque sería el sistema sanitario. Este malware permanecía en el ordenador de la víctima de manera pasiva hasta que se activada y cifraba el nombre archivos de la máquina. Tenía una manera muy curiosa de activarse: contaba el número de veces que el sistema se había iniciado y, cuando el contador llevaba a 90, ya no había vuelta atrás. El autor intentaba cubrirse las espaldas afirmando que, cuando un usuario instalaba este programa, se comprometía a pagar la licencia del mismo y, si no lo hacía, tendría consecuencias como que «su ordenador dejará de funcionar como lo hace normalmente». A un nivel más técnico, la realidad es que se trataba de un malware bastante rudimentario. En ningún momento se cifraba el contenido de los archivos, sino que simplemente se cifraba su nombre. Esto hacía que su recuperación fuera muchísimo más asequible. Aún así, para la época, fue un gran ataque en un entorno que estaba poco preparado. Este ransomware recibió el nombre de *AIDS Trojan* (troyano SIDA).

A raíz de este primer ataque, se desarrollaron otros ya más avanzados como el CryptoLocker. El primer ataque del que se tuvo constancia fue el 5 de septiembre de 2013 y hasta junio de 2014 no pudo mitigarse completamente. Afectaba, igual que el caso anterior, a sistemas Windows (DOS). Es lógico, puesto que la mayoría de usuarios finales utilizan Windows como sistema operativo, que son los objetivos últimos de estos ataques. Se propagó mediante correo electrónico y también mediante máquinas que previamente pertenecían a una botnet. Cifraba archivos tanto locales como también aquellos a los que podía llegar a través de la red donde estaba conectado el usuario mediante criptografía asimétrica (RSA). Esto hacía que, aunque se obtenera la clave con la que el atacante había cifrado la información de la víctima con técnicas como la ingeniería inversa o se conociera el *kill-switch* para parar el ransomware, no se pudieran recuperar los datos cifrados puesto que se necesitaba la clave privada que se mantenía en los servidores del atacante.

Funcionamiento del WannaCry

A continuación, estudiaremos como funciona el WannaCry. Para ello, iremos directamente a la práctica para ver, a nivel de código, qué hacía y cómo.

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 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 entre sí.

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.iuqerfsodp9ifjaposdfjhgosurijfaewrgwea.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[.]iuqerfsodp9ifjaposdfjhgosurijfaewrgwea[.]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 renombrada 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 «mssecsvc2.0» 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 aparece 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_icaccls_/_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 }

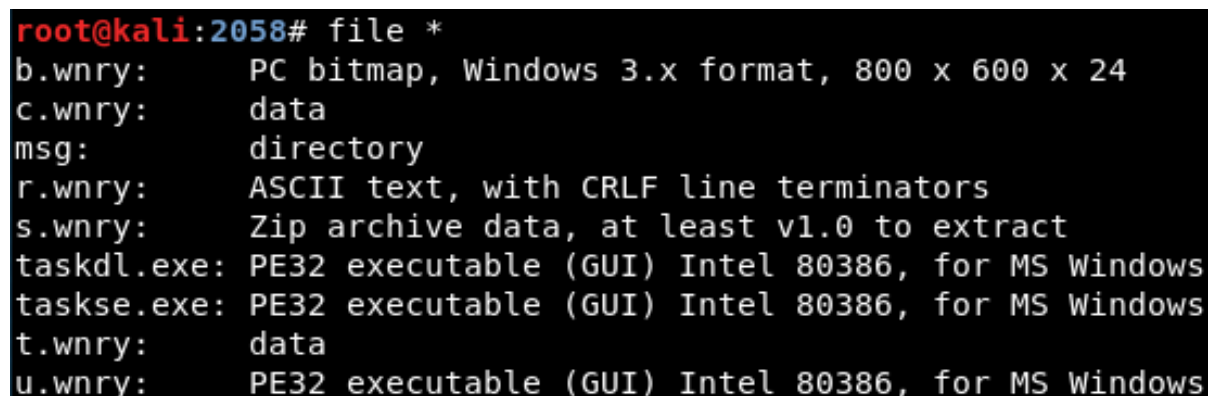
```

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 maliciosos 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 binarios 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.

En resumen, 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 *mssecv2.0*, que intentaría enumerar sesiones RDP activas en busca de nuevos equipos.

Impacto

Como ya hemos mencionado, el WannaCry tuvo un impacto sin precedentes. Se dice que alrededor 300.000 ordenadores fueron infectados en 150 países. Russia, Ucrania, India y Taiwan fueron los países más afectados.

Al igual que en el primer ransomware conocido, el troyano SIDA, el sistema sanitario fue una de las partes que salió peor parada, en este caso los hospitales de el Servicio Nacional de Salud (National Health Service) de Inglaterra y Escocia. Lo preocupante de todo esto es que no se estaba explotando un 0-day (Zero-day, vulnerabilidad que el desarrollador ha tenido cero días para parchear), sino que *EternalBlue* ya había sido reportado. Cabe destacar que los 300.000 ordenadores infectados ocurrió solo en 4 días gracias a que el *kill-switch* fue descubierto de manera prematura, pero podemos intuir que había muchas más máquinas desactualizadas (ya no solo que no tenían el último parche de seguridad, sino que muchos utilizaban Windows XP que no recibía actualizaciones desde 2014).

No debemos olvidar que quien sea que lanzó el ataque, podría haber cambiado fácilmente el *kill-switch* y lanzar una segunda oleada de ransomware cambiando el dominio que debía comprobar para continuar propagándose, pero por motivos que desconocemos prefirió dejarlo ahí.

Se estima que las pérdidas económicas de este ransomware rondan los 4 billones de dolares americanos.

Atribución del ataque

Tras analizar el ransomware, las pistas apuntan a que el ataque fue realizado por Corea del Norte. Se basan en los siguientes puntos:

- La máquina del atacante tenía archivos de *Hangul* instaladas. Esto es el alfabeto coreano.
- Dichos archivos, además contenían metadatos. Estos metadatos establecen que el ransomware fue creado en un ordenador que tenía como zona horaria UTC+09:00, la zona horaria de Corea.
- El WannaCry tiene similitudes en cuanto a código con otros malware usado en ciber ataques relacionados con Corea del Norte.

Teniendo esto en cuenta, además de más información que probablemente sea clasificada, Estados Unidos atribuyó públicamente el ataque a Corea del Norte. Esta obviamente negó su implicación en el mismo.

Es muy difícil saber a ciencia cierta quien realizó el ataque debido al uso de criptodivisas y a como se propagaba el ataque por las propias máquinas infectadas. Además, no debe descartarse que todos los puntos citados anteriormente se hayan dejado a propósito para culpar a Corea del Norte del ataque por diferentes cuestiones políticas.

Conclusión

Los ataques ransomware llevados a cabo por WannaCry tuvieron una gran repercusión mediática, quizás desmedida, y generaron una gran alarma social, afectando a grandes compañías que pudieron parecer intocables hasta ese momento.

El caos informático, magnificado por los medios de comunicación y las redes sociales, quizás tuvo algo positivo, esto es que puso a debate el nivel de seguridad de los sistemas que nos rodean a diario y de los que depende nuestro ritmo de vida.

En cuanto a qué podemos hacer para protegernos como empresa o usuario, está claro: nunca subestimar el ámbito de la ciberseguridad, mantener nuestros equipos actualizados y tener un plan de respuesta para actuar en caso de una catastrofe similar. Estar al día en cuanto a las noticias de ciberseguridad también es importante, de esta manera podemos saber qué tipo de amenazas hay actualmente y prepararnos al respecto.

WannaCry no fue un malware realmente sofisticado y, sin embargo, el mundo no estaba preparado. Nos hace plantearnos qué pasaría si se lanzara uno mucho más potente, con el objetivo no solo de recoger dinero sino de únicamente hacer daño. Si es cierto, como mencionamos en la atribución del ataque, que fue todo un plan ideado por Corea del Norte, es preocupante el poder que pueden llegar a tener los gobiernos. Llegará el punto en el que la fuerza militar de una nación no se mida por las armas nucleares que tiene o armas de destrucción masiva, sino por malware capaz de dejar a un adversario totalmente desconectado del mundo. A día de hoy es más importante la información que los territorios de un país, y la inversión de estos cada vez es mayor. Con casi total seguridad, la próxima guerra mundial se producirá en la red (si es que no se está llevando ya a cabo).

Bibliografía

- [WannaCry Ransomware Attack. Wikipedia](#)
- [AIDS \(Trojan horse\)](#)
- [CryptoLocker](#)
- [First known ransomware attack in 1989 also targeted healthcare](#)
- [Impact of WannaCry: Major disruption as organisations go back to work](#)
- [WannaCry Malware Profile. Fireeye](#)
- [WannaCry Ransomware: Analysis of Infection, Persistence, Recovery Prevention and Propagation Mechanisms.](#)