

Ofuscación de malware

Descripción de las herramientas necesarias para el análisis de malware

PEstudio

X64dbg

Scylla

Prueba 1

Análisis estático de la prueba de malware descargada

Podemos ver en la siguiente captura de pantalla, pruebas de que el malware a analizar ha sido empaquetado por una herramienta bastante conocida llamada UPX

Otro indicativo de que el binario ha sido comprimido es que la diferencia entre el tamaño virtual y el tamaño real es enorme, tanto, que el tamaño real de la parte comprimida es 0.

Vemos además, como en las cabeceras opcionales nos dice que el **Address of Entry Point** es en UPX0

Todas estas indicaciones nos hacen concluir que se ha comprimido con UPX la sección UPX0, que, una vez ejecutado el malware, se iniciara la rutina de descompresión y se cederá el control a esta sección, contenedora del malware, ya descomprimido.

Análisis dinámico del malware

Levantando la aplicación de Wireshark e INetSim para analizar el comportamiento del malware a nivel de red, vemos que hace una serie de peticiones al servidor DNS para averiguar las direcciones IP de lo que aparentemente, parece que sean los servidores C2.

Podemos ver que algunas de las direcciones que pide son:

- pxi.hognoob.se
- qia.hognoob.se
- uio.hognoob.se
- uio.herohero.info

Al estar usando el INetSim, estas direcciones no se resuelven y el virus no es capaz de contactar con sus servidores C2.

En esta sección, se pretende encontrar alguna de las características estudiadas en el temario del curso.

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00000000	4D	5A	90	00	03	00	00	00	04	00	00	00	FF	FF	00	00	MZ.....ÿÿ..
00000010	B8	00	00	00	00	00	00	00	40	00	00	00	00	00	00	00	,.....@.....
00000020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000030	00	00	00	00	00	00	00	00	00	00	00	00	10	01	00	00
00000040	0E	1F	BA	0E	00	B4	09	CD	21	B8	01	4C	CD	21	54	68	..°...!í!..Lí!Th
00000050	69	73	20	70	72	6F	67	72	61	6D	20	63	61	6E	6E	6F	is program canno
00000060	74	20	62	65	20	72	75	6E	20	69	6E	20	44	4F	53	20	t be run in DOS
00000070	6D	6F	64	65	2E	0D	0D	0A	24	00	00	00	00	00	00	00	mode....\$......
00000080	54	85	CC	7D	10	E4	A2	2E	10	E4	A2	2E	10	E4	A2	2E	T...ì}.äc..äc..äc.
00000090	7F	FB	A9	2E	19	E4	A2	2E	7F	FB	A8	2E	16	E4	A2	2E	.û@..äc..û".äc.
000000A0	6B	F8	AE	2E	13	E4	A2	2E	D3	EB	FD	2E	18	E4	A2	2E	kø@..äc..Ôëÿ..äc.
000000B0	93	F8	AC	2E	38	E4	A2	2E	26	C2	A8	2E	BD	E4	A2	2E	"ø-.8äc.&Ã".äc.
000000C0	F8	FB	A8	2E	12	E4	A2	2E	10	E4	A3	2E	B6	E6	A2	2E	øû".äc..äc..äc.
000000D0	D3	EB	FF	2E	3D	E4	A2	2E	26	C2	A9	2E	A2	E4	A2	2E	Ôëÿ.=äc.&Ã@.cäc.
000000E0	10	E4	A2	2E	11	E4	A2	2E	F8	FB	A9	2E	0A	E4	A2	2E	.äc..äc..øû@..äc.
000000F0	52	69	63	68	10	E4	A2	2E	00	00	00	00	00	00	00	00	Rich.äc.....
00000100	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000110	50	45	00	00	4C	01	03	00	F2	FD	83	5C	00	00	00	00	PE..L...òÿf\....
00000120	00	00	00	00	E0	00	0F	01	0B	01	06	00	00	50	3E	00à.....P>.
00000130	00	10	00	00	00	D0	1B	00	60	26	5A	00	00	E0	1B	00Đ...`&Z..à..
00000140	00	30	5A	00	00	00	40	00	00	10	00	00	00	02	00	00	.0Z...@.....
00000150	04	00	00	00	00	00	00	00	04	00	00	00	00	00	00	00
00000160	00	40	5A	00	00	10	00	00	00	00	00	00	02	00	00	00	.@Z.....
00000170	00	00	10	00	00	10	00	00	00	10	00	00	10	00	00	00
00000180	00	00	00	00	10	00	00	00	00	00	00	00	00	00	00	00
00000190	CC	33	5A	00	C4	03	00	00	30	5A	00	CC	03	00	00	00	î3Z..Ã....0Z..î...
000001A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000001B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000001C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000001D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000001E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000001F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000200	00	00	00	00	00	00	00	55	50	58	30	00	00	00	00	00UPX0.....
00000210	00	D0	1B	00	00	10	00	00	00	00	00	00	04	00	00	00	.Đ.....
00000220	00	00	00	00	00	00	00	00	00	00	00	80	00	00	E0	00€...à
00000230	55	50	58	31	00	00	00	00	50	3E	00	00	E0	1B	00	00	UPX1.....P>..à..
00000240	00	4A	3E	00	00	04	00	00	00	00	00	00	00	00	00	00	.J>.....
00000250	00	00	00	00	40	00	00	E0	2E	72	73	72	63	00	00	00@...à.rsrc..
00000260	00	10	00	00	00	30	5A	00	00	08	00	00	00	4E	3E	000Z.....N>.
00000270	00	00	00	00	00	00	00	00	00	00	00	40	00	00	C0	00@...Ã
00000280	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Figure 1: HxD prueba de UPX

Name	VirtualAd...	VirtualSize	RawAddre...	RawSize
UPX0	00001000	001BD000	00000400	00000000
UPX1	001BE000	003E5000	00000400	003E4A00
.rsrc	005A3000	00001000	003E4E00	00000800

Figure 2: PPEE Tamaño virtual vs raw

NTHeader	MajorLinkerVersion	06	
File Header	MinorLinkerVersion	00	
Optional Header	SizeOfCode	003E5000	
Data Directories	SizeOfInitializedData	00001000	
Section Headers	SizeOfUninitializedData	001BD000	
DIRECTORY_ENTRY_IMPORT	AddressOfEntryPoint	005A2660	UPX1
DIRECTORY_ENTRY_RESOURCE			
1) RT_VERSION			

Figure 3: PPEE Address of Entry Point

```

# net Report for session '3379' ==#
Real start date      : 2020-04-22 12:54:35
Simulated start date : 2020-04-22 12:54:35
Time difference on startup : none

2020-04-22 12:54:38 First simulated date in log file
2020-04-22 12:54:50 HTTP connection, method: GET, URL: http://192.168.10.10/waIwara/blackMoon/, file name: /var/lib/iptables/http/fakefiles/sample.html
2020-04-22 12:54:55 DNS connection, type: A, class: IN, requested name: fp.made.net
2020-04-22 13:03:05 HTTP connection, method: GET, URL: http://192.168.10.10/waIwara/blackMoon/, file name: /var/lib/iptables/http/fakefiles/sample.html
2020-04-22 13:12:32 NET connection, time received: 1587586353, time sent: 1587553958, difference: 322397
2020-04-22 13:23:17 DNS connection, type: A, class: IN, requested name: www.nrst.com
2020-04-22 13:52:09 DNS connection, type: A, class: IN, requested name: uio.hognoob.se
2020-04-22 13:52:10 DNS connection, type: PTR, class: IN, requested name: 10.10.168.192.in-addr.arpa
2020-04-22 13:52:11 DNS connection, type: PTR, class: IN, requested name: uio.hognoob.se
2020-04-22 13:52:11 DNS connection, type: PTR, class: IN, requested name: uio.hognoob.se
2020-04-22 13:52:11 DNS connection, type: PTR, class: IN, requested name: 1.10.168.192.in-addr.arpa
2020-04-22 13:52:25 DNS connection, type: PTR, class: IN, requested name: 250.255.255.239.in-addr.arpa
2020-04-22 13:52:33 DNS connection, type: A, class: IN, requested name: 2019.ip138.com
2020-04-22 13:52:33 HTTP connection, method: GET, URL: http://2019.ip138.com/it/csp, file name: /var/lib/iptables/http/fakefiles/sample.html
2020-04-22 13:52:35 DNS connection, type: A, class: IN, requested name: ifconfig.me
2020-04-22 13:52:38 DNS connection, type: A, class: IN, requested name: qia.hognoob.se
2020-04-22 13:52:39 DNS connection, type: PTR, class: IN, requested name: 255.10.168.192.in-addr.arpa
2020-04-22 13:52:52 DNS connection, type: PTR, class: IN, requested name: pxl.hognoob.se
2020-04-22 13:52:51 DNS connection, type: PTR, class: IN, requested name: 251.0.0.224.in-addr.arpa
2020-04-22 13:52:52 DNS connection, type: PTR, class: IN, requested name: f.f.2.0.4.6.e.f.f.f.f.7.2.0.0.a.0.0.0.0.0.0.0.0.0.0.0.0.e.f.ip6.arpa
2020-04-22 13:52:52 DNS connection, type: PTR, class: IN, requested name: b.f.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.2.0.f.f.ip6.arpa
2020-04-22 13:53:02 HTTP connection, method: GET, URL: http://2019.ip138.com/it/csp, file name: /var/lib/iptables/http/fakefiles/sample.html
2020-04-22 13:53:18 HTTP connection, method: GET, URL: http://2019.ip138.com/it/csp, file name: /var/lib/iptables/http/fakefiles/sample.html
2020-04-22 13:53:32 HTTP connection, method: GET, URL: http://2019.ip138.com/it/csp, file name: /var/lib/iptables/http/fakefiles/sample.html
2020-04-22 14:00:13 DNS connection, type: PTR, class: IN, requested name: 8.8.8.8.in-addr.arpa
2020-04-22 14:00:18 DNS connection, type: PTR, class: IN, requested name: 8.8.8.8.in-addr.arpa
2020-04-22 14:00:18 DNS connection, type: PTR, class: IN, requested name: 8.8.8.8.in-addr.arpa
2020-04-22 14:00:18 DNS connection, type: PTR, class: IN, requested name: 8.8.8.8.in-addr.arpa
2020-04-22 14:02:28 DNS connection, type: A, class: IN, requested name: faq.x64dbg.com
2020-04-22 14:02:28 DNS connection, type: A, class: IN, requested name: faq.x64dbg.com
2020-04-22 14:02:30 HTTP connection, method: GET, URL: http://faq.x64dbg.com/, file name: /var/lib/iptables/http/fakefiles/sample.html
2020-04-22 14:02:57 DNS connection, type: A, class: IN, requested name: qia.hognoob.se
2020-04-22 14:02:57 Last simulated date in log file

```

Figure 4: INetSim reporte filtrado

- Cifrado Cesar
- Base64
- XOR
- identificación de cryptosignatures

Análisis desde el punto de vista del empaquetado de ejecutables

En esta sección se pretende desempaquetar el malware seleccionado para posteriormente hacer un volcado de memoria virtual a disco y averiguar las cabeceras. De esta forma, podemos analizar el malware como se ha creado desde el inicio, sin los inconvenientes introducidos por los packers.

Desempaquetado manual

Se ha decidido usar X64dbg para hacer el desempaquetado manual. Se inicia el procedimiento, con los programas de análisis dinámico iniciados, para monitoreos el proceso.

El procedimiento es el siguiente:

1. Se descomprime el virus a analizar
2. Se abre ProcessHacker, para poder tener un rastro del proceso
3. Se abre Noriben, para que nos de un resumen de los procesos que hemos mirado con process monitor
4. Se inicia x64dbg con privilegios de administrador
 1. En este momento, está el virus cargado en memoria
 2. Tenemos que llegar al momento en el que se acaba de descomprimir el malware y se le cede el control
5. Seguimos ejecutando el malware hasta que los avisa de que estamos cediendo el control a una dirección desconocida
6. La siguiente instrucción es el OEP del malware.

Desempaquetado automático

Prueba 2

Análisis Estático

Con el análisis de las cabeceras, podemos ver que es un ejecutable, porque tiene la cabecera mágica. Además, observando el tamaño del RAW Size de la sección .text y el tamaño del virtual size, vemos que existe una diferencia importante.

Análisis Dinámico

Al ver que es una muestra que está packed, podemos ejecutar la muestra con nuestro debugger hasta que se llame al proceso `VirtualAlloc` del API

Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocat
Byte[8]	Dword	Dword	Dword	Dword	Dword	Dword	Word
.text	00002C10	00001000	00003000	00001000	00000000	00000000	0000
.rdata	00015FE8	00004000	00016000	00004000	00000000	00000000	0000
.data	000011B8	0001A000	00001000	0001A000	00000000	00000000	0000
.rsrc	00007170	0001C000	00008000	0001B000	00000000	00000000	0000
.reloc	000000E0	00024000	00001000	00023000	00000000	00000000	0000

Figure 5: Cabeceras

Kernel32.dll.

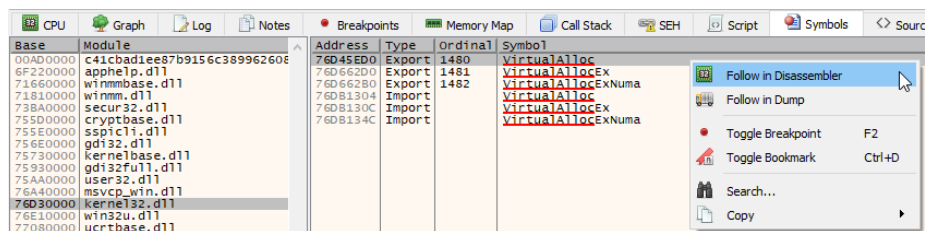


Figure 6: VirtualAlloc Breakpoint

Hacemos esto porque sabemos que el virus es autocontenido y que no crea un nuevo proceso desde si mismo. Esto quiere decir, que en algún momento, se debe reservar memoria para el virus. Esta memoria se reserva con **VirtualAlloc**, pero si nos damos cuenta y entramos en dicha función, vemos que se llama a **KernelBase**.

76D45ED0	8BFF	mov edi,edi	VirtualAlloc
76D45ED2	55	push ebp	
76D45ED3	8BEC	mov ebp,esp	
76D45ED5	5D	pop ebp	
76D45ED6	FF25 0413DB76	jmp dword ptr ds:[&VirtualAlloc]	JMP.&VirtualAlloc

Figure 7: VirtualAlloc KernelBase

Para obtener el valor que devuelve la función, debemos poner el breakpoint en el return del código de **KernelBase**.

En este momento, tenemos los siguientes Breakpoints:

Según la documentación de Microsoft, podemos saber que este método devuelve el puntero a memoria en el registro **EAX**, entonces, basta con seguir el registro en el Dump para obtener el programa **Unpacked**. Tras dos iteraciones, se ha generado un archivo PE en una sección de memoria.

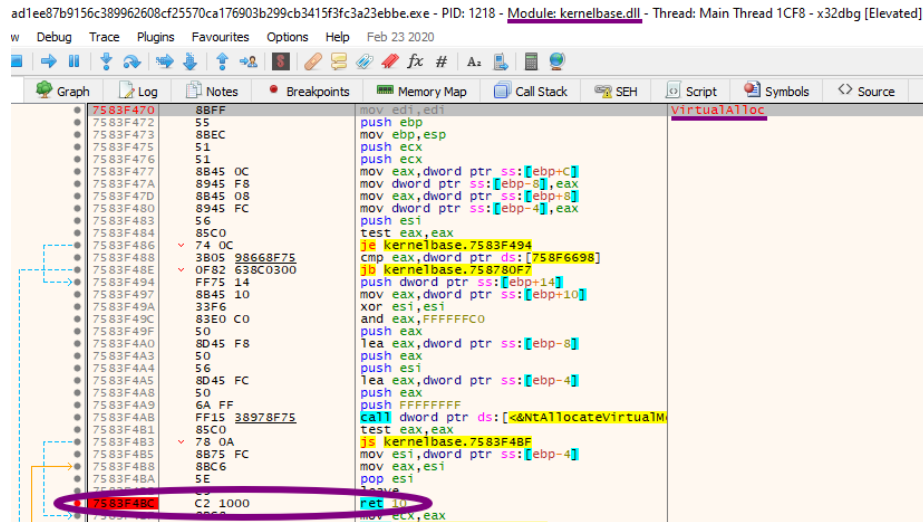


Figure 8: VirtualAlloc return de KernelBase

Type	Address	Module/Label/Exception	State	Disassembly	Hits	Summary
Software	00AD16F0	<c41cbad1ee87b9156c389962608cf25570ca176903b299cb3415f3fc3a23ebbe.exe	One-time	xchg ebp,eax	0	entry breakpoint
	7583F48C	kernelbase.dll	Enabled	ret 10	0	

Figure 9: Breakpoints

Address	Hex	ASCII
00AB0000	4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00	MZ.....yy..
00AB0010	B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00@.....
00AB0020	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00D.....
00AB0030	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00AB0040	0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68	...!.Li!Th
00AB0050	69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F	is program canno
00AB0060	74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20	t be run in DOS
00AB0070	6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00	mode....\$....
00AB0080	78 98 95 E3 3C FA FB 80 3C FA FB 80 3C FA FB 80	x..â<ûû*ûû*ûû*
00AB0090	3C FA FA 80 20 FA FB 80 35 82 68 80 35 FA FB 80	<ûû* ûû*5.h*5ûû*
00AB00A0	3C FA FB 80 3D FA FB 80 31 A8 25 80 3D FA FB 80	<ûû*=ûû*1'%*=ûû*
00AB00B0	52 69 63 68 3C FA FB 80 00 00 00 00 00 00 00 00	Rich<ûû*.....
00AB00C0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00AB00D0	50 45 00 00 4C 01 04 00 5A 88 5F 5A 00 00 00 00	PE..L...Z..Z....
00AB00E0	00 00 00 00 E0 00 02 01 08 01 0C 00 80 29 00 00â.....°)..
00AB00F0	80 2A 00 00 00 00 00 00 02 39 00 00 00 10 00 00	.*.....9.....
00AB0100	00 40 00 00 00 00 40 00 00 10 00 00 10 00 00 00	..@.....e.....
00AB0110	05 00 00 00 00 00 00 00 05 00 00 00 00 00 00 00
00AB0120	00 E0 00 00 70 02 00 00 00 00 00 00 03 00 40 81	..a..p.....e..

Figure 10: PE generado

En este momento, sabemos que se ha guardado el archivo unpacked en memoria. Para llegar a el, tenemos que observar las direcciones de memoria que tienen el bit de ejecutable habilitado. En este sample en concreto, tenemos dos. Siguiendo la primera en el dump, podemos ver que esta en esta dirección nuestro PE. Para extraer el sample unpacked, tenemos que volcar el contenido de la memoria en disco, como en la siguiente imagen:

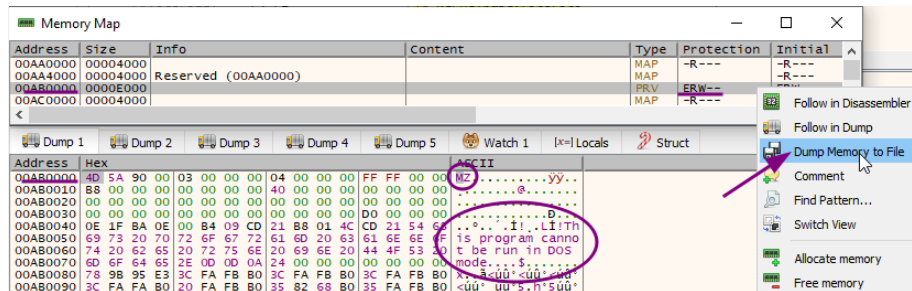


Figure 11: PE encontrado en memoria y volcando