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 ademas, 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 este 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.

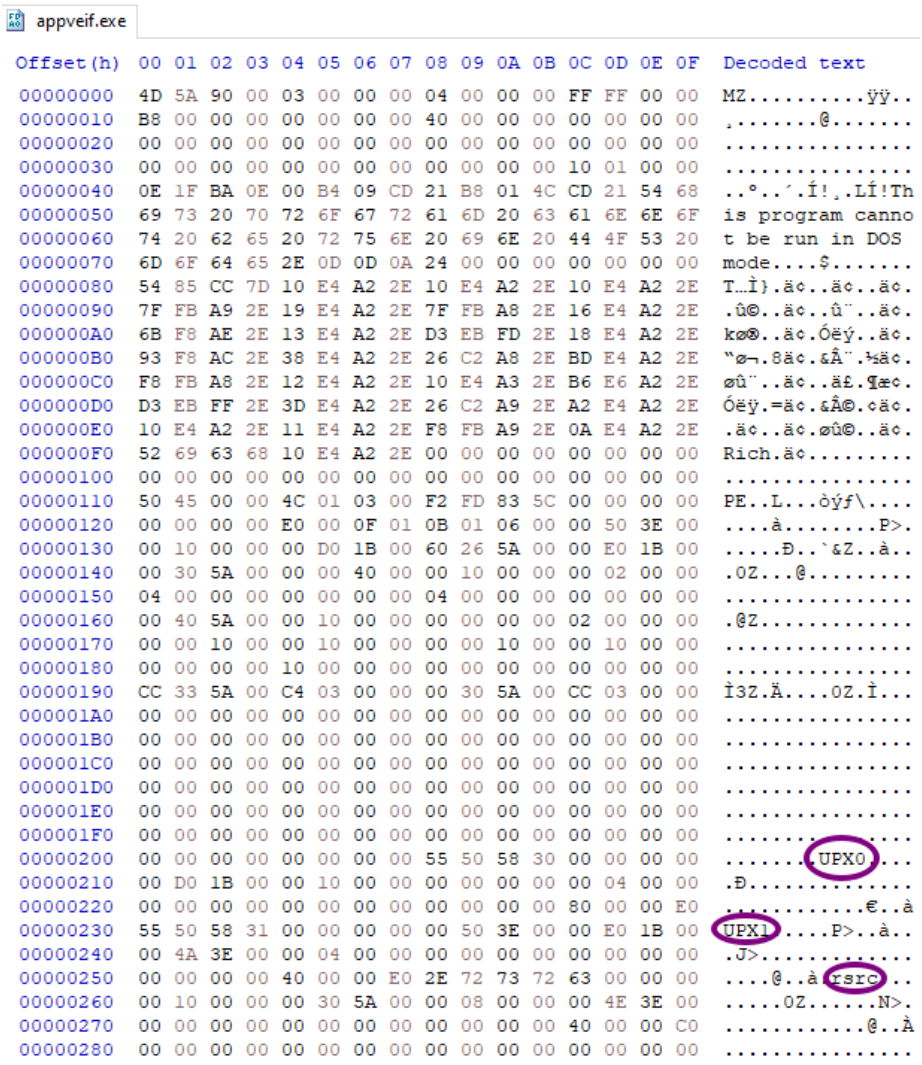


Figure 1: HxD prueba de UPX

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

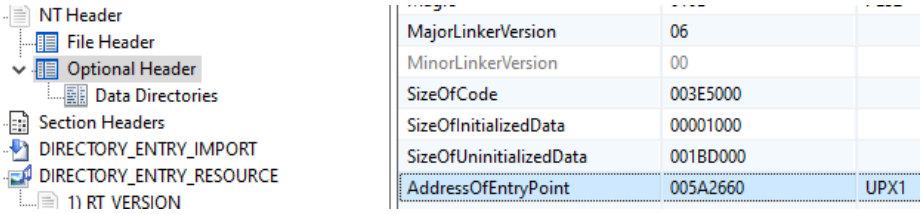
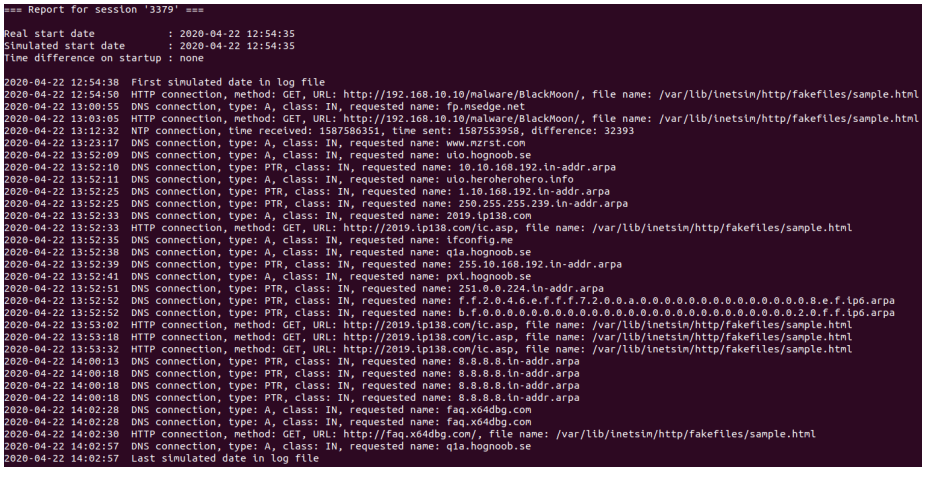


Figure 3: PPEE Address of Entry Point



```
=== 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/malware/BlackMoon/, file name: /var/lib/inetsim/http/fakefiles/sample.html
2020-04-22 13:00:55  DNS connection, type: A, class: IN, requested name: fp.msedge.net
2020-04-22 13:03:05  HTTP connection, method: GET, URL: http://192.168.10.10/malware/BlackMoon/, file name: /var/lib/inetsim/http/fakefiles/sample.html
2020-04-22 13:12:32  NTP connection, time received: 1587586351, time sent: 1587553958, difference: 32393
2020-04-22 13:23:17  DNS connection, type: A, class: IN, requested name: www.mzrst.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: A, class: IN, requested name: uio.heroherohero.info
2020-04-22 13:52:25  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/ic.asp, file name: /var/lib/inetsim/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: q1a.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 13:52:41  DNS connection, type: A, 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.7.2.0.0.a.0.0.0.0.0.0.0.0.0.0.0.0.0.0.8.e.f.ip6.arpa
2020-04-22 13:52:52  DNS connection, type: PTR, class: IN, requested name: b.f.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.2.0.f.f.ip6.arpa
2020-04-22 13:53:02  HTTP connection, method: GET, URL: http://2019.ip138.com/ic.asp, file name: /var/lib/inetsim/http/fakefiles/sample.html
2020-04-22 13:53:18  HTTP connection, method: GET, URL: http://2019.ip138.com/ic.asp, file name: /var/lib/inetsim/http/fakefiles/sample.html
2020-04-22 13:53:32  HTTP connection, method: GET, URL: http://2019.ip138.com/ic.asp, file name: /var/lib/inetsim/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/inetsim/http/fakefiles/sample.html
2020-04-22 14:02:57  DNS connection, type: A, class: IN, requested name: q1a.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, esta 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. Ademas, 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 esta packed, podemos ejecutar la muestra con nuestro debugger hasta que se llame al proceso `VirtualAlloc` del API

c41cbad1ee87b9156c389962608cf

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`.

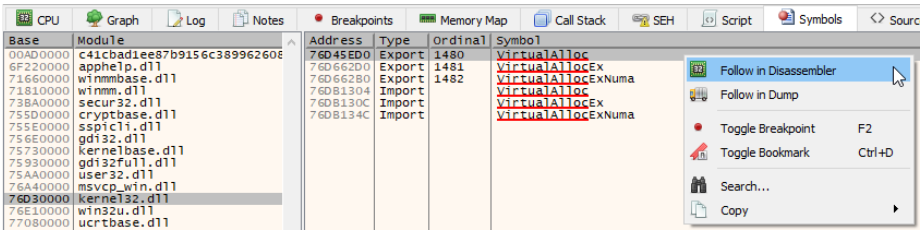


Figure 6: VirtualAlloc Breakpoint

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`.

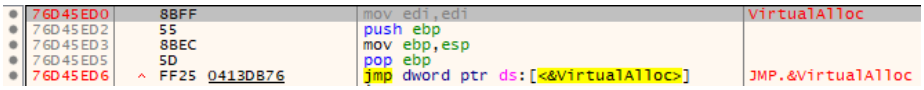


Figure 7: VirtualAlloc KernelBase

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.

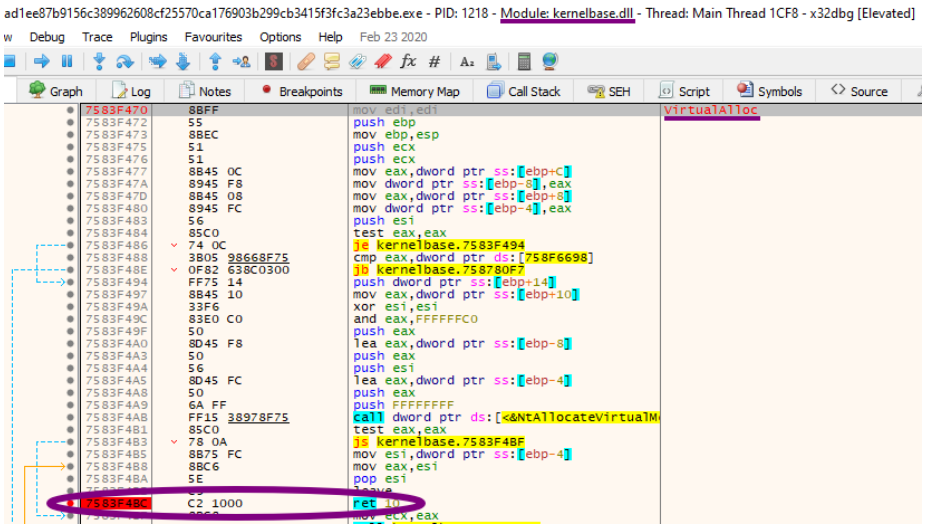


Figure 8: VirtualAlloc return de KernelBase

Type	Address	Module/Label/Exception	State	Disassembly	Hits	Summary
Software						
	00AD16F0	<c41cbad1ee87b9156c389	One-time	xchg ebp,eax	0	entry breakpoint
	7583F4BC	kernelbase.dll	Enabled	ret 10	0	

Figure 9: Breakpoints

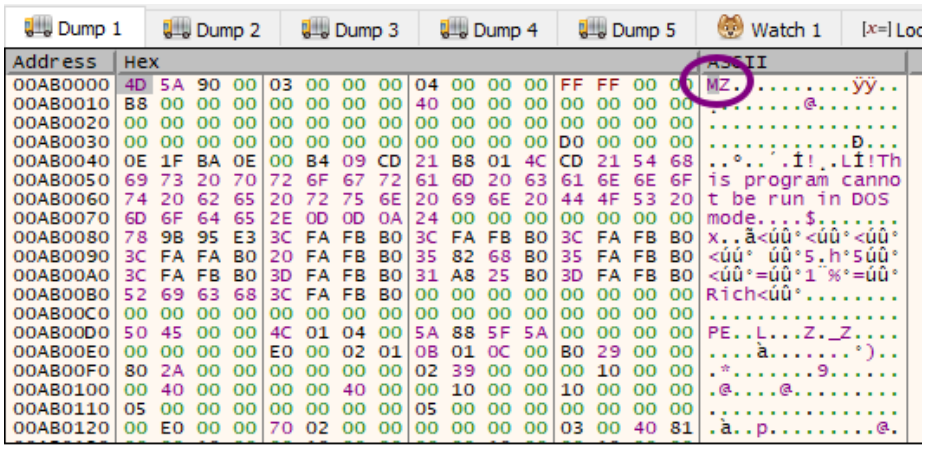


Figure 10: PE generado

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:

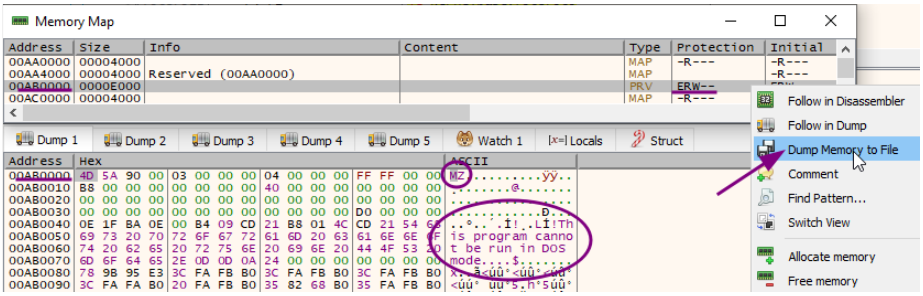


Figure 11: PE encontrado en memoria y volcando