



Victima	Printif
Protección	UPX, Serial
Herramientas	Olly, Filemon, [PE Explorer]
Objetivo	Parchear limite de 30 días

Introducción

Hola a todos,

Bueno, antes de seguir con WL hacemos un alto en el camino y atendemos una petición de mi amigo Ralba.

En este caso es un programa creado con un compilador nuevo para mí y que nos puede traer un poco de cabeza en futuras víctimas.

El objetivo que me pidió Ralba era quitar el límite de 30 días, no sé como de complejo es analizar el serial ya que me centré solo en esto.

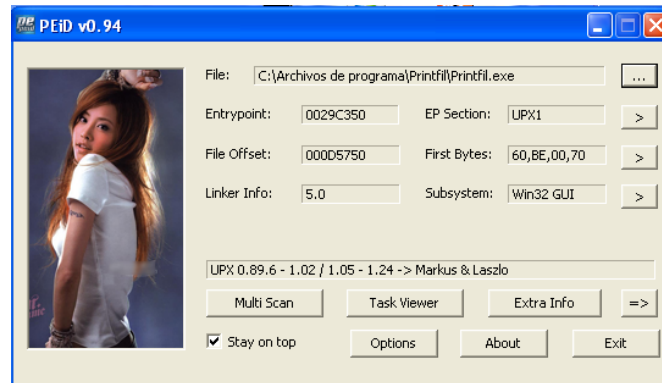
Recordar que todo lo que se muestra aquí es meramente educativo, el que vaya a utilizar el software que lo compre.

Dicho esto empecemos.

Analizando al enemigo

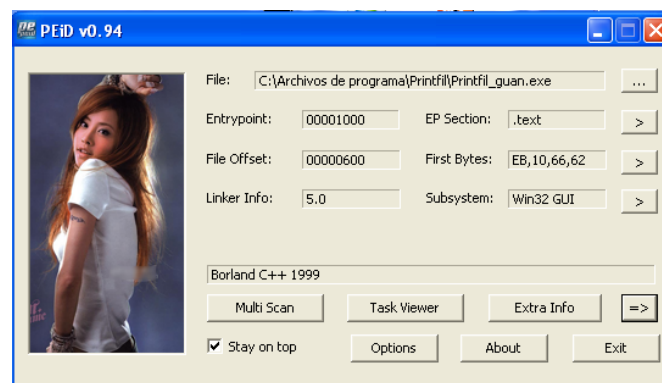
Aunque no creo que en este caso sea relevante decir que tengo el Olly protegido con el phanom.

Una vez dicho esto, si analizamos el ejecutable con el PEiD vemos que está comprimido con UPX



Como andamos un poco con fiaca y tenemos el PE Explorer a mano simplemente lo analizamos y guardamos por ejemplo con Printfil_guan.exe

Si lo analizamos ahora vemos



Si ejecutamos el programa nos aparece un splash screen indicando que tenemos 30 días



Como sabemos que es por tiempo pues pongamos los bps en las típicas APIs (GetLocalTime y GetSystemTime), veamos si para y en donde.

0012FC50	004A68D4	CALL to GetLocalTime from PrintFil.004A68CF
0012FC54	0012FC58	pLocaltime = 0012FC58
0012FC58	0012FC64	

La primera vez nos para y es llamado desde el programa veamos donde estamos.

PUSH EBP	
MOV EBP,ESP	
ADD ESP,-10	
LEA EAX,[LOCAL.4]	
PUSH EAX	
CALL <JMP.&KERNEL32.GetLocalTime>	pLocaltime
MOVZX EDX,WORD PTR SS:[EBP-10]	GetLocalTime
MOV ECX,[ARG.1]	
MOV DWORD PTR DS:[ECX],EDX	
MOVZX EAX,WORD PTR SS:[EBP-E]	
MOV EDX,[ARG.2]	
MOV DWORD PTR DS:[EDX],EAX	
MOVZX ECX,WORD PTR SS:[EBP-A]	
MOV EAX,[ARG.3]	
MOV DWORD PTR DS:[EAX],ECX	
MOV ESP,EBP	
POP EBP	
RETN	

Bueno aquí toma los datos y los guarda en una estructura en memoria. Salimos de esta rutina y estamos:

PUSH EBP	
MOV EBP,ESP	
ADD ESP,-0C	
LEA EAX,[LOCAL.3]	
PUSH EAX	Arg3
LEA EDX,[LOCAL.2]	
PUSH EDX	Arg2
LEA ECX,[LOCAL.1]	
PUSH ECX	Arg1
CALL Printfil.004A68C5	Printfil.004A68C5
ADD ESP,0C	
PUSH [LOCAL.3]	Arg3
PUSH [LOCAL.2]	Arg2
PUSH [LOCAL.1]	Arg1
CALL Printfil._hb_retd	_hb_retd
ADD ESP,0C	
MOV ESP,EBP	
POP EBP	
RETN	

Pasa los datos guardados y llama a la función _hb_retd. Como cosa que llama la atención es que parece que este programa tiene los símbolos de parte del programa y por eso vemos los nombres de ciertas funciones.

```

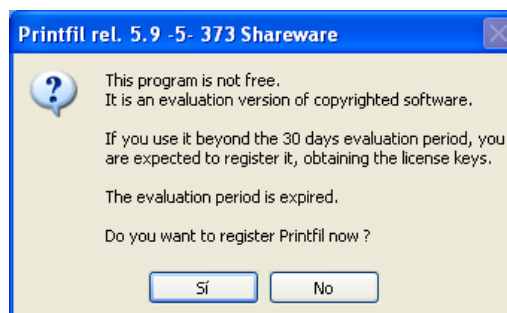
- TEST BYTE PTR DS:[EBX+5].1
-~ JE SHORT Printfil.0046DD59
- PUSH DWORD PTR DS:[EAX+4]
- PUSH DWORD PTR DS:[EAX]
- CALL Printfil._hb_vmExecute
- ADD ESP,8
-~ JMP SHORT Printfil.0046DDC6
> CALL EAX
-~ JMP SHORT Printfil.0046DDC6
> PUSH -1
- PUSH DWORD PTR DS:[EBX]
- PUSH 0
- PUSH 3E9
- PUSH 0C
- CALL Printfil.00459961
-~
Arg5 = FFFFFFFF
Arg4 =
Arg3 = 00000000
Arg2 = 000003E9
Arg1 = 0000000C
Printfil.00459961

```

Llegados a este punto ya me hizo plantearme las cosas de otra forma. Ese nombre de función de `_hb_vmExecute` y por lo que vi traceando me indica que estamos dentro de una máquina virtual.

Intentar seguir los datos tomados de la API `GetLocalTime` también es complicado, más cuando esta API es llamada desde varias partes ante de arrancar el programa y se hace complicado saber que llamada es la que realmente realiza la comparación.

Para intentar reducir el tema adelante el año del reloj, con lo que el programa ya no arranca apareciendo el siguiente cartel

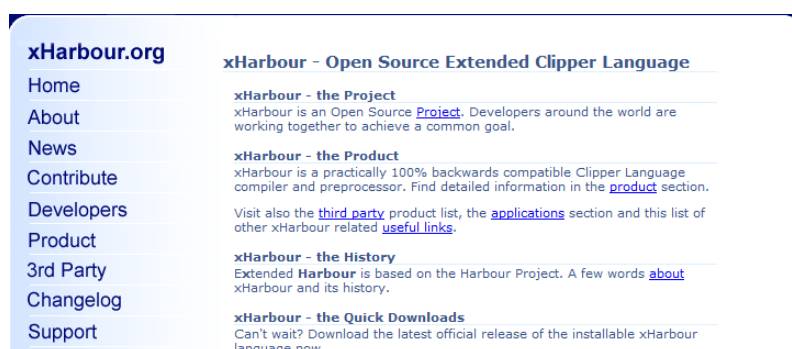


El resultado fue parecido, se llama como 4 o 5 veces a la API `GetLocalTime` antes de sacar la Nag.

Mirando por las referencias a cadenas podemos ver algo como esto:

```
005F2988 ASCII "xHarbour build %d.%d.%d Intl. (%s) (Rev. %d)"
```

Esto me hizo buscar un poco en la red y encontré esto <http://www.xharbour.org/>

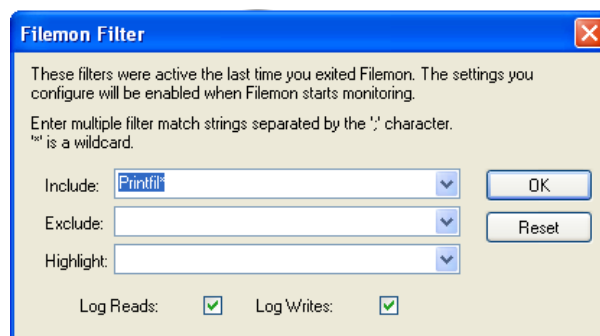


Al parecer el programa está hecho en xHarbour, y me imagino que esta Virtual Machine pues pertenece al núcleo del compilador xHarbour. Este tipo de compiladores nos complican un poco la vida a la hora de tracear el código.

Volviendo al programa, tras adelantar el reloj y reponerlo nos encontramos con que el programa no vuelve funcionar, solo nos muestra la nag, pero de alguna forma debe de saber que ya caduco y es justo aquí donde encontramos la debilidad del programa.

Los programas actuales para llevar la cuenta suelen usar el registro, pero un estudio con el ntregmon no vi nada, por lo que si no usa el registro debe de usar un fichero, más alternativas no quedan.

Abrimos el filemon y lo configuramos para que lo haga log de nuestro progama



Arrancamos el programa y cuando sale la nag paramos el log y vemos que tenemos.

1779	23:47:23	Printfil.exe:304	FASTIO_QUERY_STAN...	C:\WINDOWS\161491591.dll
1780	23:47:23	Printfil.exe:304	IRP_MJ_READ	C:\WINDOWS\161491591.dll
1781	23:47:23	Printfil.exe:304	FASTIO_READ	C:\WINDOWS\161491591.dll
1782	23:47:23	Printfil.exe:304	IRP_MJ_CLEANUP	C:\WINDOWS\161491591.dll
1783	23:47:23	Printfil.exe:304	IRP_MJ_CREATE	C:\WINDOWS\161491592.dll
1784	23:47:23	Printfil.exe:304	FASTIO_QUERY_STAN...	C:\WINDOWS\161491592.dll

Tenemos un par de dll un poco sospechosas en c:\Windows, vamos a ver que son

Al parecer no son dlls, sino ficheros binarios a los que se le ha dado esa extensión para despistar.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
00000000	0D	0A	30	33	2F	30	36	2F	32	30	31	31	0D	0A	20	20	. .03/06/2011..
00000010	20	20	20	20	20	33	37	33	0D	0A	30	33	2F	30	36	2F	373..03/06/
00000020	32	30	31	31	0D	0A	20	20	20	20	20	20	20	33	37	33	2011.. 373
00000030	1A																.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16
00000000	74	00	73	00	85	00	8B	00	69	00	65	00	62	00	63	00	68	00	62	00	1A		

¿Qué pasa si las borramos? Pues que el programa vuelve a funcionar como si estuviéramos en el día 1 jejeje.

Creando el injerto

En este caso, lo más fácil puede ser crear un loader que borre las 2 dlls antes de ejecutar el programa, por lo que así saltamos la comprobación del tiempo y nunca caduca.

Hablando de esto con Ralba nos dimos cuenta de que las dlls tenían unos valores distinto en mi máquina que en la suya, así que para hacerlo un poco más genérico lo que vamos a hacer es crear un injerto que borre las dlls antes de que las vaya a abrir, el resultado es el mismo, pero como el programa es el que genera el nombre de las dlls pues así ya funcionaría para todas las máquinas.

Así que volvemos a poner nuestro programa en Olly y ponemos un BP en CreateFileA, vamos dando a Run hasta que veamos que quiere abrir alguna de las 2 dlls.

Y llegamos aquí

0012E68C	0043036C	CALL to CreateFileA from Printfil.00430367
0012E690	00C3A308	FileName = "C:\WINDOWS\161491591.dll"
0012E694	80000000	Access = GENERIC_READ
0012E698	00000003	ShareMode = FILE_SHARE_READ FILE_SHARE_WRITE
0012E69C	00000000	pSecurity = NULL
0012E6A0	00000003	Mode = OPEN_EXISTING
0012E6A4	00000000	Attributes = NORMAL
0012E6A8	00000000	hTemplateFile = NULL

Vemos que se llama desde el programa salimos de la API y estamos aquí

PUSH EBX	FileName
CALL Printfil.004C3030	CreateFileA
MOV ESI,EAX	
PUSH 0	Arg2 = 00000000
CMP ESI,-1	
SETNE AL	
AND EAX,1	
PUSH EAX	Arg1
CALL Printfil.004318CD	Printfil.004318CD
ADD ESP,8	
CMP [LOCAL.1],0	
JE SHORT Printfil.0043038F	
PUSH EBX	
CALL Printfil.0045CE3B	
POP ECX	
MOV EAX,ESI	
POP ESI	
POP EBX	
MOV ESP,EBP	
POP EBP	
RETN	

Salimos de esta función y llegamos aquí

0043AFE8	~ 0F84 A1000000	JE Printfil.0043B091	
0043AFF0	68 C0000000	PUSH 0C0	
0043AFF5	53	PUSH EBX	Arg1
0043AFF6	E8 4FA40300	CALL Printfil.0047544A	Printfil.0047544A
0043AFFB	59	POP ECX	
0043AFFC	50	PUSH EAX	Arg1
0043AFFD	E8 1953FFFF	CALL Printfil.0043031B	Printfil.0043031B
0043B002	83C4 00	ADD ESP,8	
0043B005	8B08	MOV EBX,EAX	
0043B007	83FB FF	CMP EBX,-1	
0043B00A	74 7B	JE SHORT Printfil.0043B087	
0043B00C	6A 02	PUSH 2	Arg3 = 00000002
0043B00E	6A 00	PUSH 0	Arg2 = 00000000
0043B010	53	PUSH EBX	Arg1
0043B011	E8 0B5DFFFF	CALL Printfil.00430D21	Printfil.00430D21

Aquí es donde se comprueba si el Handle es bueno o hubo error, vamos a considera buena esta zona para el injerto y como creo esto es de la virtual machine, vamos a poner un HE en la dirección 43AFFD (el PUSH EAX) y vamos a comprobar si en esta zona solo se abre nuestra dll falsa o pasan por aquí la apertura de todos los ficheros que use el programa.

Reiniciamos el Olly y tenemos esto:

1º parada

PUSH EAX	Arg1 = 00BF7484 ASCII "C:\WINDOWS\OVP413.dll"
CALL Printfil.0043031B	Printfil.0043031B

2º parada

PUSH EAX	Arg1 = 00C3A308 ASCII "C:\WINDOWS\161491591.dll"
CALL Printfil.0043031B	Printfil.0043031B

3º parada

PUSH EAX	Arg1 = 00C3A2D4 ASCII "C:\WINDOWS\161491592.dll"
CALL Printfil.0043031B	Printfil.0043031B

Y ya no hay más paradas salvo que mandemos a imprimir algo, por lo que sabemos que esta zona es la que se usa para abrir los ficheros, sea el que sea.

En este caso podemos hacer un injerto un poco inteligente usando un contador, de forma que la primera vez que pase por aquí no haga nada, la 2º y 3º vez borren el fichero apuntado por el registro EAX antes de mandarlo a abrir y después de borrar en el fichero la tercera vez pues simplemente se anula el injerto el solito.

Buscamos una zona de ceros y el injerto queda como sigue.

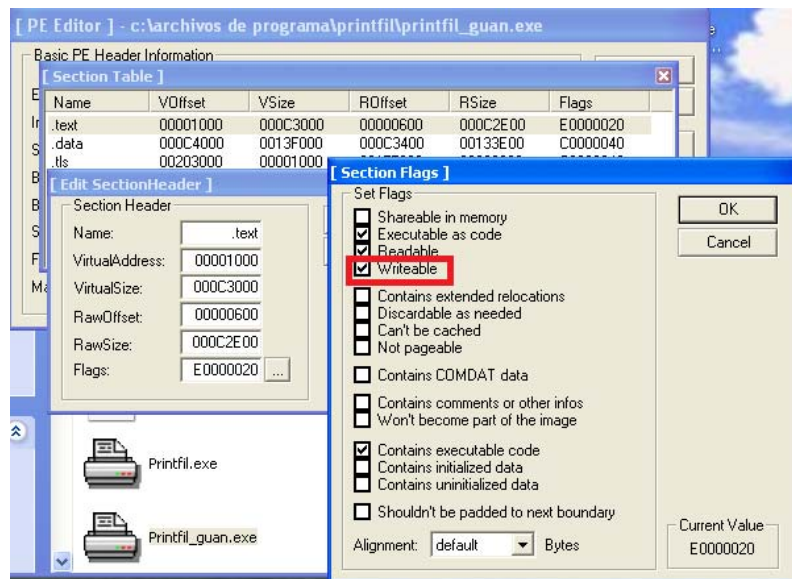
004C3C30	> 60	PUSHAD	
004C3C31	50	PUSH EAX	
004C3C32	FE05 2A3C4C00	INC BYTE PTR DS:[4C3C2A]	Direccion de nuestro contador
004C3C38	8A1D 2A3C4C00	MOV BL,BYTE PTR DS:[4C3C2A]	
004C3C3E	58	POP EAX	
004C3C3F	80FB 01	CMPL BL,1	
004C3C42	74 22	JE SHORT Printfil.004C3C66	1º vez salimos
004C3C44	50	PUSH EAX	FileName
004C3C45	E8 FEF3FFFF	CALL <JMP.&KERNEL32.DeleteFileA>	borramos el fichero
004C3C4A	8A1D 2A3C4C00	MOV BL,BYTE PTR DS:[4C3C2A]	
004C3C50	80FB 03	CMPL BL,3	
004C3C53	75 11	JNZ SHORT Printfil.004C3C66	
004C3C55	08 FCAF4300	MOV EAX,Printfil.0043AFFC	si es la 3º vez reponemos los valores originales
004C3C5A	66:C700 50E8	MOV WORD PTR DS:[EAX],0E850	
004C3C5F	C740 02 1953FFFF	MOV DWORD PTR DS:[EAX+2],FFFF5319	
004C3C66	> 61	POPAD	
004C3C67	50	PUSH EAX	Fin del injerto
004C3C68	E8 AEC6F6FF	CALL Printfil.hb_fsOpen	
004C3C6D	E9 9079F7FF	JMP Printfil.0043B002	

El cambio de flujo a nuestro injerto lo tenemos aquí

0043AFFB	59	POP ECX
0043AFFC	E9 2F8C0000	JMP Printfil.004C3C3C
0043B001	90	NOP
0043B002	83C4 08	ADD ESP,8
0043B005	8BD8	MOV EBX,EAX
0043B007	83FB FF	CMPL EBX,-1
0043B00A	74 7B	JE SHORT Printfil.0043B087

Para que todo esto funcione queda un detalle, y es que tenemos que poner la región de código como escritura, sino no podremos incrementar nuestro contador que lo puse en la misma sección de código.

Así que con LordPE nos vamos a secciones y cambiamos las características



Salvamos los cambios, arrancamos y todo va de maravilla.
Adelantamos el reloj lo atrasamos y todo sigue bien ☺

Agradecimientos

Especialmente a Ralba que con este mini reto me ha hecho pasar unas horas divertidas, a todo el grupo que forman CLS y especialmente a ti por haber leído este tutorial

Hasta la próxima.

