HT2 SDS JJHH

March 9, 2023

1 Hoja de trabajo Análisis de Malware

José Javier Hurtarte Hernández 19707

[ ]:
```
import pefile
from IPython.display import Image
```

[ ]:
```
FILE1 = 'sample_qwrty_dk2'
FILE2 = 'sample_vg655_25th.exe'

pe_exec1 = pefile.PE('./MALWR/'+FILE1)
pe_exec2 = pefile.PE('./MALWR/'+FILE2)
```

1.0.1 Examinar PE Header, DLL y API Calls

[ ]:
```
print('sample_qwrty_dk2')
for entry in pe_exec1.DIRECTORY_ENTRY_IMPORT:
    print('\nLlamada a DLL:', entry.dll)
    print('Llamadas a Funciones de:', entry.dll.decode())
    for fun in entry.imports:
        print(fun.name)
```

```
sample_qwrty_dk2

Llamada a DLL: b'KERNEL32.DLL'
Llamadas a Funciones de: KERNEL32.DLL
b'LoadLibraryA'
b'ExitProcess'
b'GetProcAddress'
b'VirtualProtect'

Llamada a DLL: b'MSVCRT.dll'
Llamadas a Funciones de: MSVCRT.dll
b'atol'

Llamada a DLL: b'SHELL32.dll'
Llamadas a Funciones de: SHELL32.dll
```

```
b'SHChangeNotify'

Llamada a DLL: b'USER32.dll'
Llamadas a Funciones de: USER32.dll
b'LoadStringA'

Llamada a DLL: b'WS2_32.dll'
Llamadas a Funciones de: WS2_32.dll
b'closesocket'
```

[ ]:
```
print('sample_vg655_25th.exe')
for entry in pe_exec2.DIRECTORY_ENTRY_IMPORT:
    print('\nLlamada a DLL:', entry.dll)
    print('Llamadas a Funciones de:', entry.dll.decode())
    for fun in entry.imports:
        print(fun.name.decode())
```

```
sample_vg655_25th.exe

Llamada a DLL: b'KERNEL32.dll'
Llamadas a Funciones de: KERNEL32.dll
GetFileAttributesW
GetFileSizeEx
CreateFileA
InitializeCriticalSection
DeleteCriticalSection
ReadFile
GetFileSize
WriteFile
LeaveCriticalSection
EnterCriticalSection
SetFileAttributesW
SetCurrentDirectoryW
CreateDirectoryW
GetTempPathW
GetWindowsDirectoryW
GetFileAttributesA
SizeofResource
LockResource
LoadResource
MultiByteToWideChar
Sleep
OpenMutexA
GetFullPathNameA
CopyFileA
GetModuleFileNameA
VirtualAlloc
VirtualFree
```

```
FreeLibrary
HeapAlloc
GetProcessHeap
GetModuleHandleA
SetLastError
VirtualProtect
IsBadReadPtr
HeapFree
SystemTimeToFileTime
LocalFileTimeToFileTime
CreateDirectoryA
GetStartupInfoA
SetFilePointer
SetFileTime
GetComputerNameW
GetCurrentDirectoryA
SetCurrentDirectoryA
GlobalAlloc
LoadLibraryA
GetProcAddress
GlobalFree
CreateProcessA
CloseHandle
WaitForSingleObject
TerminateProcess
GetExitCodeProcess
FindResourceA

Llamada a DLL: b'USER32.dll'
Llamadas a Funciones de: USER32.dll
wsprintfA

Llamada a DLL: b'ADVAPI32.dll'
Llamadas a Funciones de: ADVAPI32.dll
CreateServiceA
OpenServiceA
StartServiceA
CloseServiceHandle
CryptReleaseContext
RegCreateKeyW
RegSetValueExA
RegQueryValueExA
RegCloseKey
OpenSCManagerA

Llamada a DLL: b'MSVCRT.dll'
Llamadas a Funciones de: MSVCRT.dll
realloc
```

```
fclose
fwrite
fread
fopen
sprintf
rand
srand
strcpy
memset
strlen
wcscat
wcslen
__CxxFrameHandler
??3@YAXPAX@Z
memcmp
_except_handler3
_local_unwind2
wcsrchr
swprintf
??2@YAPAXI@Z
memcpy
strcmp
strrchr
__p___argv
__p___argc
_stricmp
free
malloc
??0exception@@QAE@ABV0@@Z
??1exception@@UAE@XZ
??0exception@@QAE@ABQBD@Z
_CxxThrowException
calloc
strcat
_mbsstr
??1type_info@@UAE@XZ
_exit
_XcptFilter
exit
_acmdln
__getmainargs
_initterm
__setusermatherr
_adjust_fdiv
__p__commode
__p__fmode
__set_app_type
_controlfp
```

Principalmente podemos observar que en el archivo .exe tenemos 4 dll y 114 llamadas a api, lo cual desde un inicio nos dice que hay muchas llamadas lo cual nos indica que tiene una alta interacción con el sistema y principalmente archivos. Además el archivo sample_qwrty_dk2 realiza pocas llamadas a API sin embargo podemos ver que usa WS2_32.dll la cual sirve para funciones de red y APIS y también podemos ver que puede usar llamadas para cargar código malicioso.

1.0.2 Obtenga la información de las secciones del PE Header.

[ ]:
```
print('Secciones ', FILE1)
for section in pe_exec1.sections:
    print(section.Name, hex(section.VirtualAddress), hex(section.
 ↪Misc_VirtualSize), section.SizeOfRawData)
```

```
Secciones  sample_qwrty_dk2
b'UPX0\x00\x00\x00\x00' 0x1000 0x5000 0
b'UPX1\x00\x00\x00\x00' 0x6000 0x1000 4096
b'.rsrc\x00\x00\x00' 0x7000 0x1000 512
```

[ ]:
```
print('Secciones ', FILE2)
for section in pe_exec2.sections:
    print(section.Name, hex(section.VirtualAddress), hex(section.
 ↪Misc_VirtualSize), section.SizeOfRawData)
```

```
Secciones  sample_vg655_25th.exe
b'.text\x00\x00\x00' 0x1000 0x69b0 28672
b'.rdata\x00\x00' 0x8000 0x5f70 24576
b'.data\x00\x00\x00' 0xe000 0x1958 8192
b'.rsrc\x00\x00\x00' 0x10000 0x349fa0 3448832
```

Que tengan UPX singifica que están comprimidos mediante el compresor de ejecutables UPX y cada una de estas secciones contiene datos comprimidos del ejecutable, sin embargo esta suele ser una tecnica que se suele utilizar para ocultar código malicioso, debido a esto se requiere descomprimirlo para poder analizar si realiza llamadas sospechosas.

[ ]:
```
!upx -d ./MALWR/sample_qwrty_dk2_COPY
```

```
                       Ultimate Packer for eXecutables
                          Copyright (C) 1996 - 2020
UPX 3.96        Markus Oberhumer, Laszlo Molnar & John Reiser   Jan 23rd 2020

        File size         Ratio      Format      Name
   --------------------   ------   -----------   -----------
      8192 <-      5632   68.75%    win32/pe     sample_qwrty_dk2_COPY

Unpacked 1 file.
```

[ ]:
```
print('sample_qwrty_dk2 DESCOMPRIMIDO:')
pe_exec3 = pefile.PE('./MALWR/sample_qwrty_dk2_COPY')
for entry in pe_exec3.DIRECTORY_ENTRY_IMPORT:
```

```
    print('\nLlamada a DLL:', entry.dll)
    print('Llamadas a Funciones de:', entry.dll.decode())
    for fun in entry.imports:
        print(fun.name.decode())
```

```
sample_qwrty_dk2 DESCOMPRIMIDO:

Llamada a DLL: b'KERNEL32.DLL'
Llamadas a Funciones de: KERNEL32.DLL
CloseHandle
WaitForSingleObject
CreateEventA
ExitThread
Sleep
GetComputerNameA
CreatePipe
DisconnectNamedPipe
TerminateProcess
WaitForMultipleObjects
TerminateThread
CreateThread
CreateProcessA
DuplicateHandle
GetCurrentProcess
ReadFile
PeekNamedPipe
SetEvent
WriteFile
SetProcessPriorityBoost
SetThreadPriority
GetCurrentThread
SetPriorityClass
lstrcatA
lstrcpyA
GetEnvironmentVariableA
GetShortPathNameA
GetModuleFileNameA
GetStartupInfoA
GetModuleHandleA

Llamada a DLL: b'MSVCRT.dll'
Llamadas a Funciones de: MSVCRT.dll
_controlfp
_beginthread
_strnicmp
sprintf
atol
strchr
```

```
free
malloc
_exit
_XcptFilter
exit
_acmdln
__getmainargs
_initterm
__setusermatherr
_adjust_fdiv
__p__commode
__p__fmode
__set_app_type
_except_handler3
_itoa

Llamada a DLL: b'SHELL32.dll'
Llamadas a Funciones de: SHELL32.dll
ShellExecuteExA
SHChangeNotify

Llamada a DLL: b'USER32.dll'
Llamadas a Funciones de: USER32.dll
LoadStringA

Llamada a DLL: b'WS2_32.dll'
Llamadas a Funciones de: WS2_32.dll
htons
connect
socket
WSAStartup
send
inet_addr
recv
closesocket
```

1.0.3 En que categoría sospechosas pueden clasificarse estos ejemplos en base a algunas de las llamadas a las APIs que realizan

Basado en llamadas y clasificación según el patrón del paper:

sample_vg655_25th.exe

- GetFileAttributesW: Behaviour 3 -> Get File Information
- GetFileSizeEx: Behaviour 3 -> Get File Information
- ReadFile: Behaviour 5 -> Read/Write Files
- GetFileSize: Behaviour 3 -> Get File Information
- WriteFile: Behaviour 5 -> Read/Write Files

- SetFileAttributesW: Behaviour 6 -> Change File Attributes
- GetFileAttributesA: Behaviour 3 -> Get File Information
- CopyFileA: Behaviour 2 -> Copy/Delete Files
- CloseHandle: Behaviour 2 -> Copy/Delete Files

sample_qwrty_dk2 Descomprimido

- CloseHandle: Behaviour 2 -> Copy/Delete Files
- WriteFile: Behaviour 5 -> Read/Write Files
- GetShortPathNameA: Behaviour 3 -> Get File Information

1.0.4 Para el archivo "sample_vg655_25th.exe" obtenga el HASH en base al algoritmo SHA256.

[ ]:
```
!sha256sum ./MALWR/sample_vg655_25th.exe
```

```
ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e41aa
./MALWR/sample_vg655_25th.exe
```

1.0.5 Para el archivo "sample_vg655_25th.exe", ¿cuál es el propósito de la DLL ADVAPI32.dll?

Esta lo que hace es que utiliza apis para las llamadas principalmete a funciones de edición de registro y configuraciones de seguridad. Además posee funciones criptográficas para encripción y decripción de datos y controles de acceso

Fuente: https://learn.microsoft.com/en-us/previous-versions/windows/desktop/legacy/ee391633(v=vs.85)

1.0.6 Para el archivo "sample_vg655_25th.exe", ¿cuál es el propósito de la API CryptReleaseContext?

Lo que hace esta unción es liberar el identificador de un proveedor de servicios criptográficos. Por lo que cuando una aplicación llama a esta función luego de haber utilizado un proveedor de servicios criptográficos, el identificador de este ya no es válido y el servicio criptográfico utilizado ya no se puede utilizar más. Sin embargo algo importante es que este no destruye los pares de claves o contenedores de claves.

Fuente: https://learn.microsoft.com/en-us/windows/win32/api/wincrypt/nf-wincrypt-cryptreleasecontext

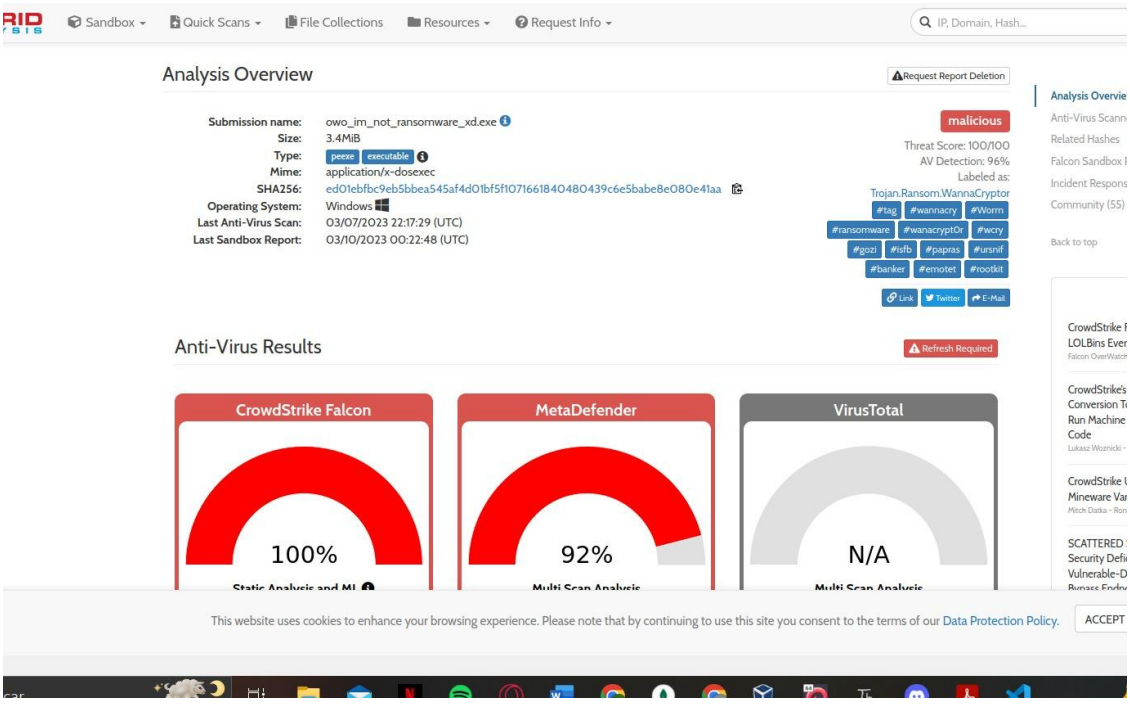
1.0.7 Con la información recopilada hasta el momento, indique para el archivo "sample_vg655_25th.exe" si es sospechoso o no, y cual podría ser su propósito.

Basado primero en las caracteristicas de las llamadas a funciones vemos que hace bastantes manipulaciones a archivos, tal como: obtener información de archivos, leer o escribir en ellos, copiarlos o eliminarlos y cambiar sus atributos. Además observando el propósito de ADVAPI32 y CryptReleaseContext nosotros podemos ver que utiliza funciones para modificar registros, ver permisos y configuraciones de seguridad, además de utilización de funciones criptográficas. Añadido todo esto podemos decir solo con este comportamiento que se trata de una aplicación que trata con archivos, los modifica o elimina y hace uso de criptografía, dandonos indicios de que puede ser un Malware de tipo Ransomware, por lo que sí es sospechoso.

1.0.8 Utilice la plataforma de análisis dinámico https://www.hybrid-analysis.com y cargue el archivo "sample_vg655_25th.exe". ¿Se corresponde el HASH de la plataforma con el generado? ¿Cuál es el nombre del malware encontrado? ¿Cuál es el propósito de este malware?

[ ]:
```
Image(filename='overview.jpg')
```

[ ]:



Sí corresponde el Hash generado, el nombre del malware es owo_im_not_ransomware_xd.exe, sin embargo tiene un nombre más reconocido el cual es wannacry o de manera más formal Trojan.Ransom.WannaCryptor y el propósito de este malware es encriptar archivos y pedir una recompensa en bitcoin para poder recuperarlos, siendo esta recompensa no en todos los casos una garantia de que se recuperarán los datos.

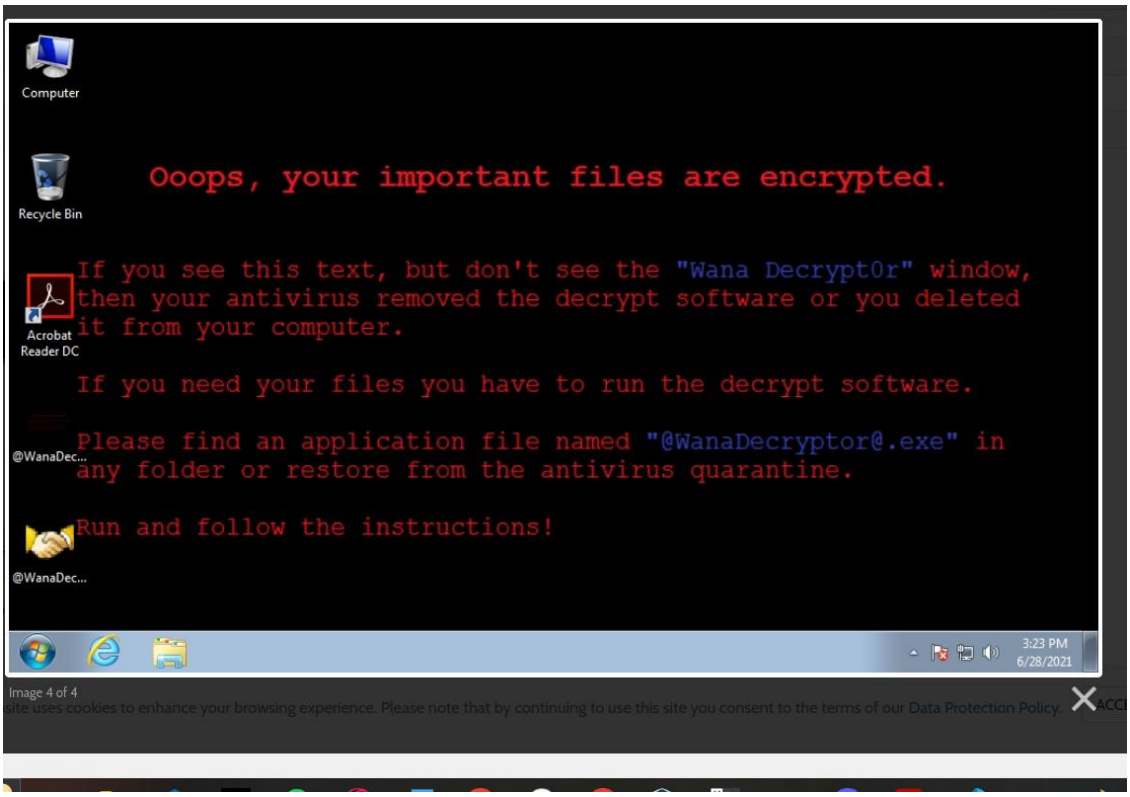
1.0.9 Muestre las capturas de pantalla sobre los mensajes que este malware presenta a usuario. ¿Se corresponden las sospechas con el análisis realizado en el punto 7?

[ ]:
```
Image(filename='1.jpg')
```

[ ]:

[ ]: `Image(filename='2.jpg')`

[ ]:



Podemos ver que si coincide ya que este ejecutable realizó operaciones con archivos, y junto con esto utilizó servicios criptográficos para poder encriptarlos y hacer que no estuvieran disponibles.