Malware Analysis Exam

Analysis of sample_09_rmc

Student:

PATIL Kunal

MSc Artificial Intelligence Systems

Teacher:

KHALDI Adel

Table of Contents

Binary Data Manipulation

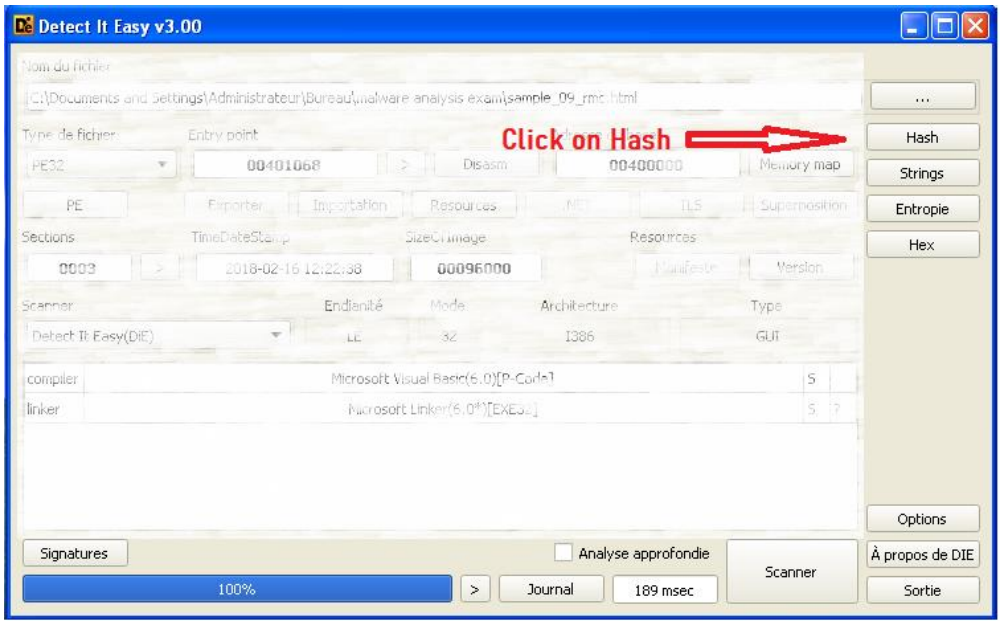
File hash

Approach 1

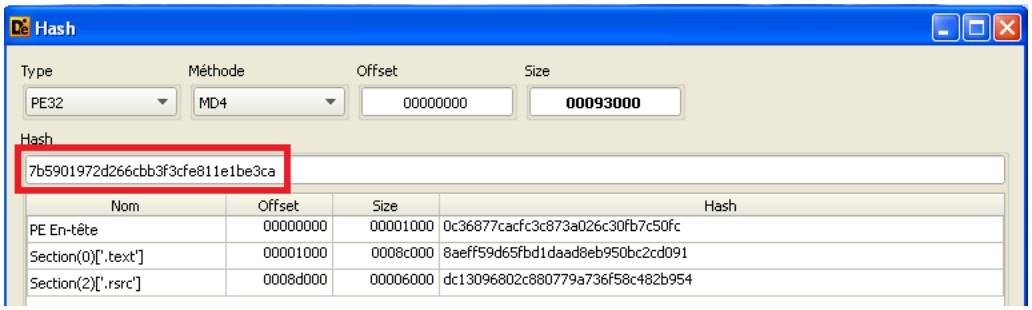
Using Tool: Detect it Easy (DiE)

Procedure:

1. Open Detect it Easy.
2. Browse/Drag and drop file
3. Click on Hash to check Hash value



4. Note the value of Hash as highlighted below.

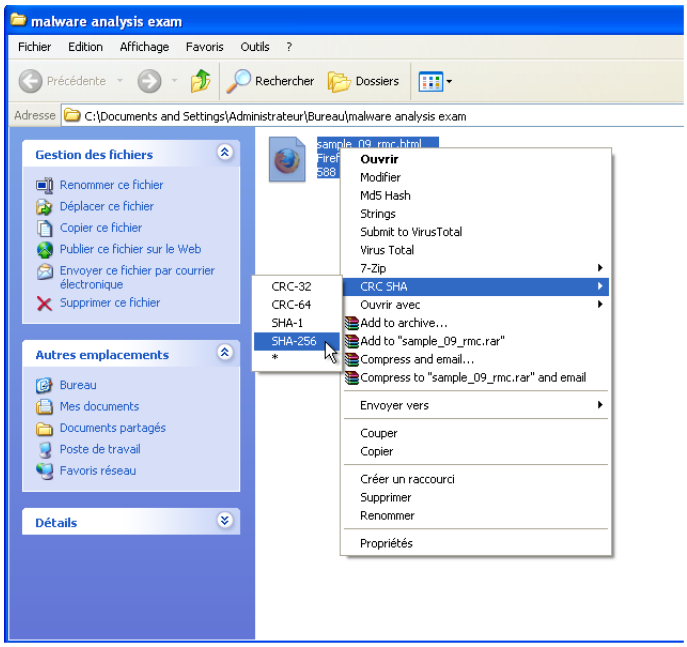


Approach 2

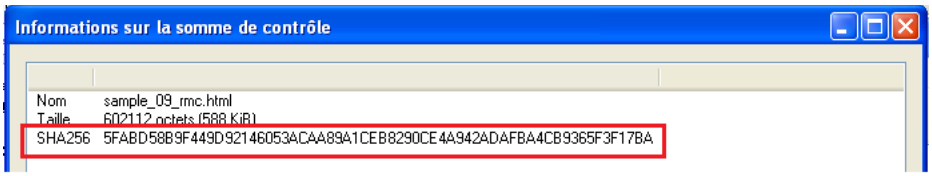
Using Right Click on file to check SHA256 hash

Procedure:

1. Right Click on the file.
2. Select CRC SHA → SHA-256



3. Check SHA-256 Hash Value



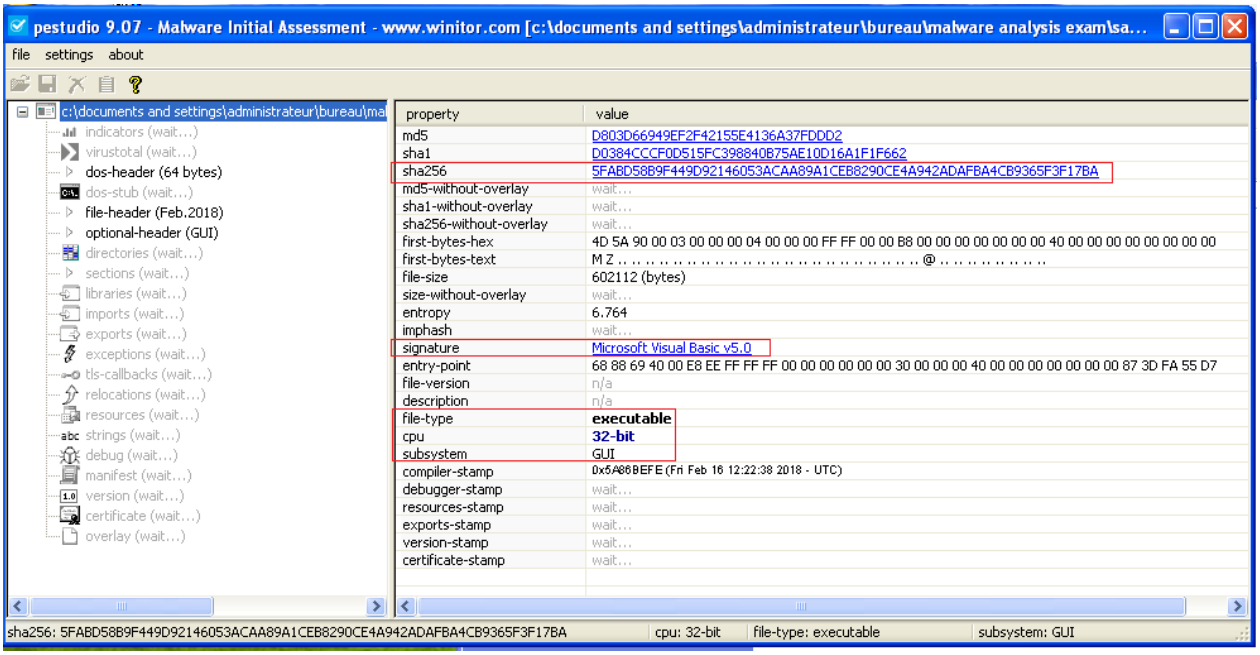
To Check whether it is known/detected sample

Approach

Using Tool: PEStudio

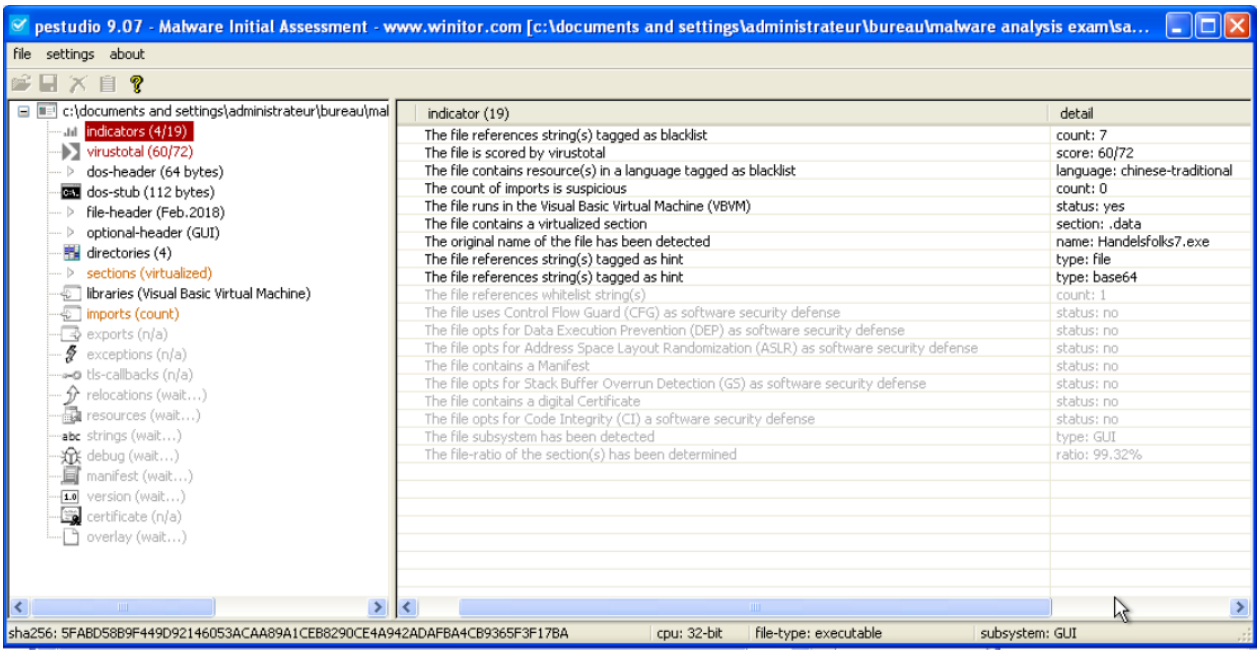
Procedure:

1. Open file using PEStudio
2. Check details as shown in below screenshot



The malware file is executable, 32 bit and GUI application, compiled using Visual Basic studio.

3. Check Indicators in PEStudio

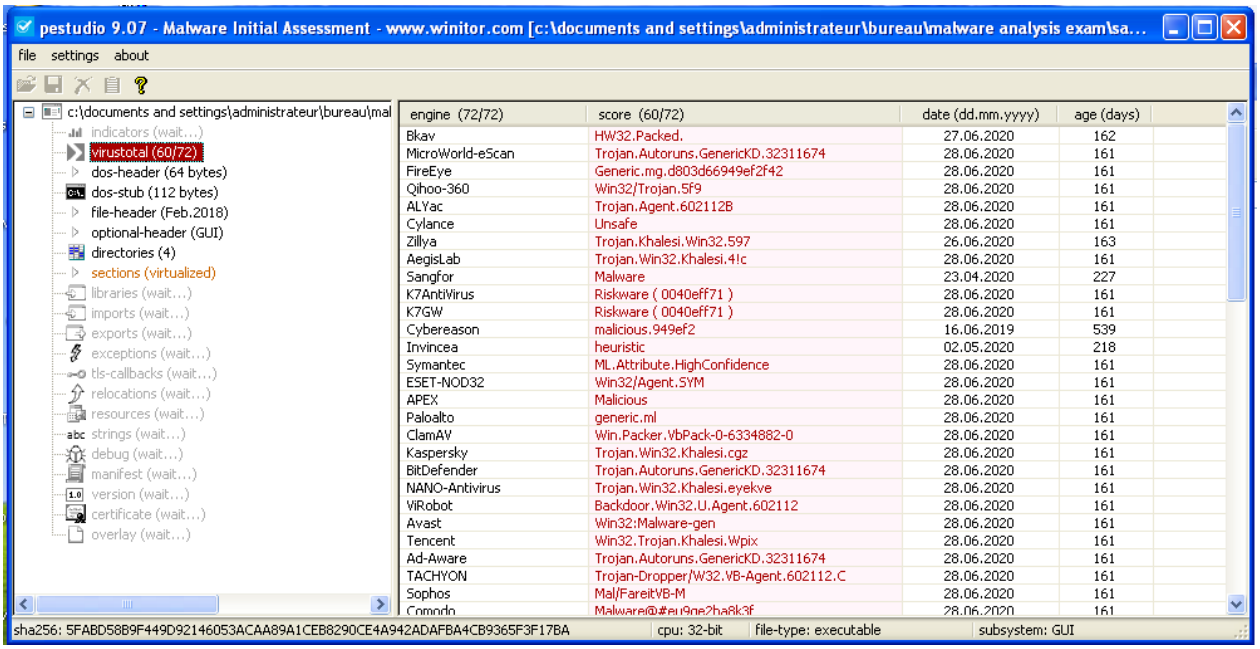


Score by virusTotal 60/72

Language contain: Chinese-traditional

Original name of file: handelfolks7.exe

4. Check virusTotal in detail



The file indicating W32/Khalesi which is classified as a trojan. A trojan is a type of malware that performs activities without the user's knowledge. These activities commonly include establishing remote access connections, capturing keyboard input, collecting system information, downloading/uploading files, dropping other malware into the infected system, performing denial-of-service (DoS) attacks, and running/terminating processes.

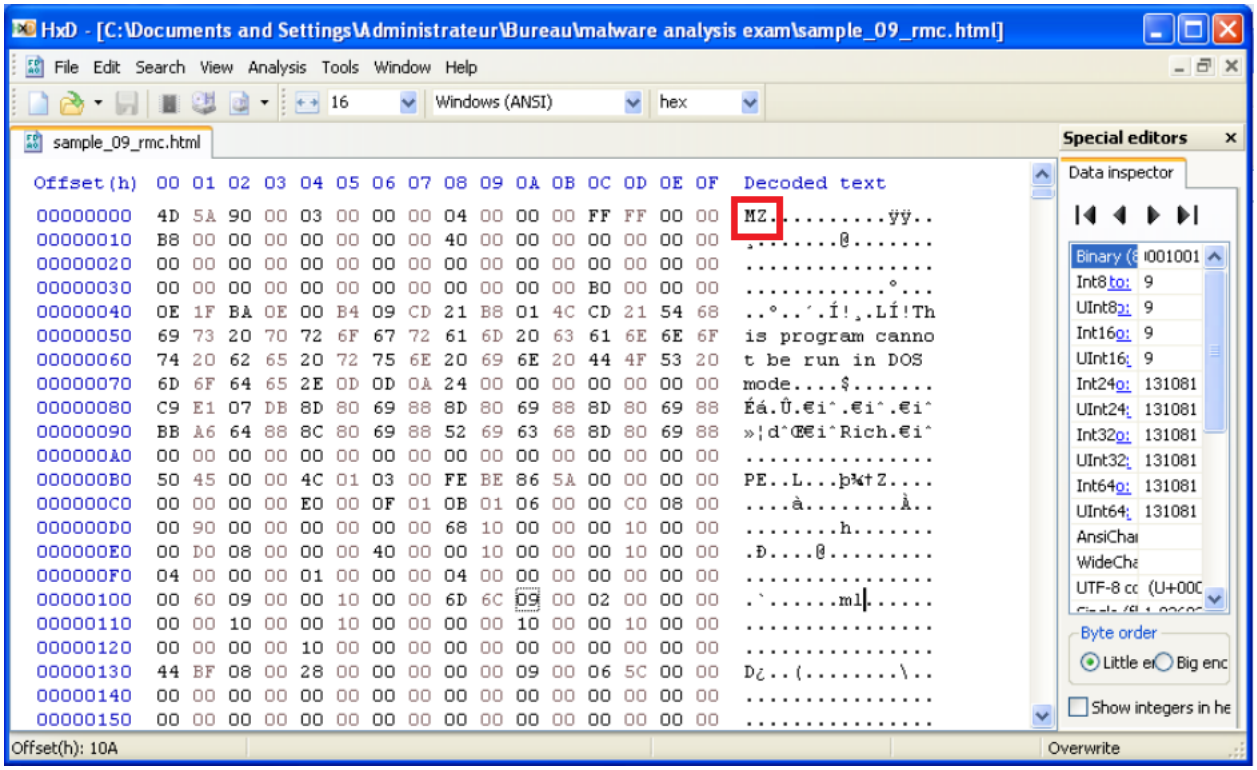
File type of malware

Approach 1

Using Tool: HxD

Procedure:

1. Open file using HxD
2. Check the first bytes and check for the magic number



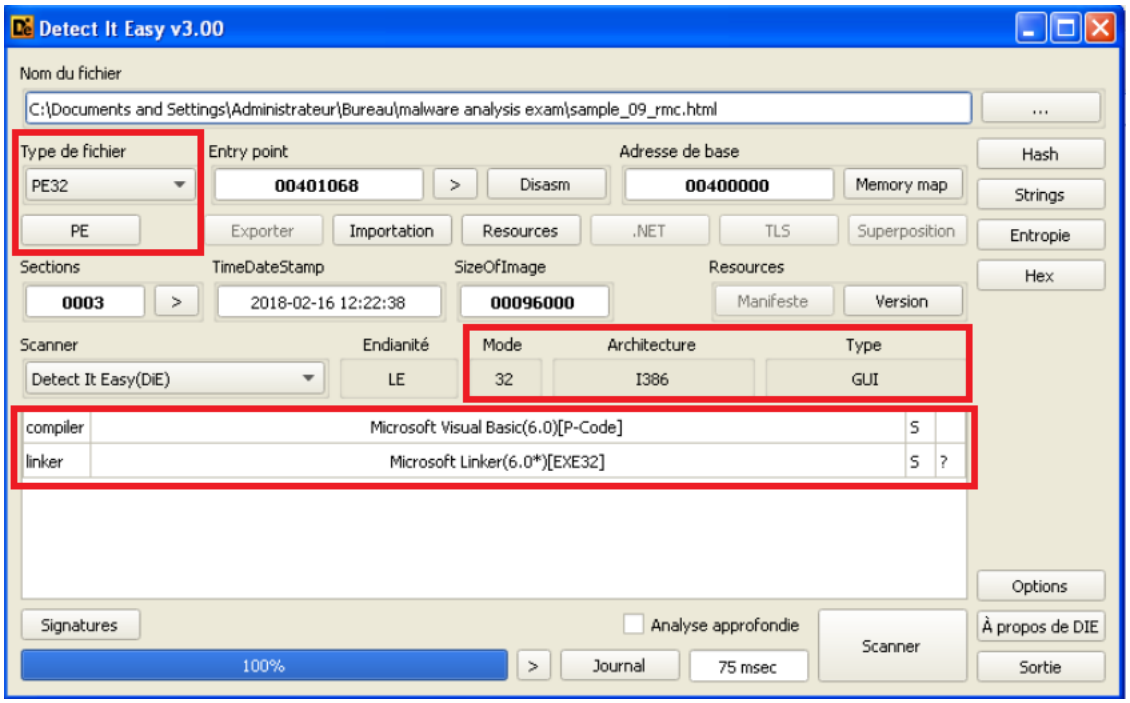
The magic byte is "MZ" means the program cannot be run in DOS mode, it is Portable Executable (PE). (.exe file)

Approach 2

Using Tool: Detect it Easy

Procedure:

1. Open file using Detect it Easy
2. Observe type of file, mode architecture, type and compiler



The file is Portable executable 32 bit and GUI, compiler is Microsoft Visual Basic.

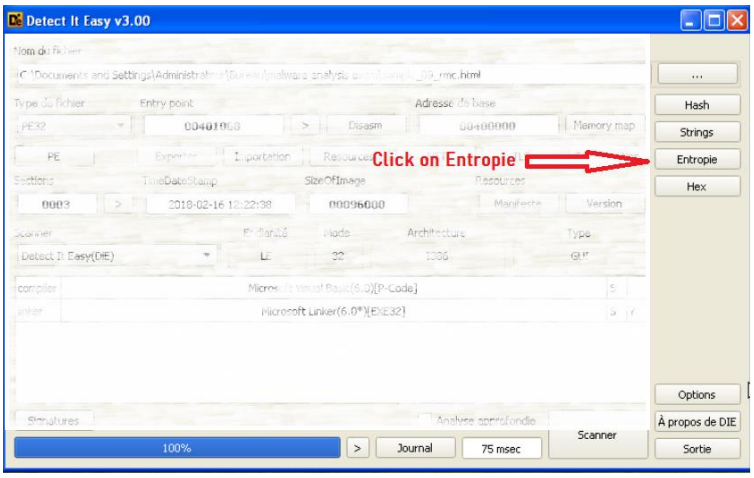
Entropy distribution across the file content

Approach

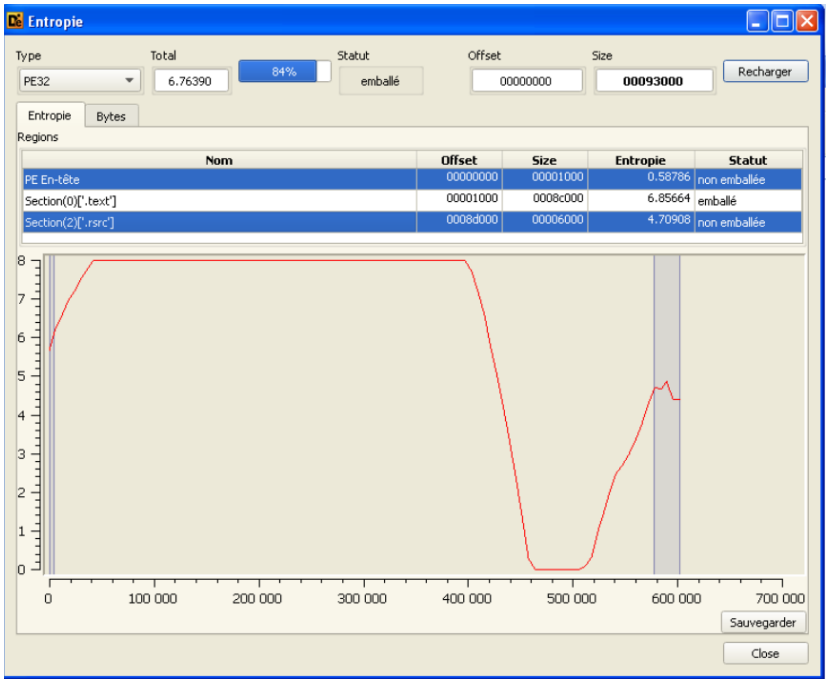
Using Tool: Detect it Easy

Procedure:

1. Open Detect it Easy.
2. Browse/Drag and drop file
3. Click on Entropy to check Entropy and packing per section



4. Obsereve the entropy graph and check packing per section.



We can observe, the text section has high entropy almost 8 at starting and then its dropping at the end to 0, So starting part of the section could contain code/text and which is highly packed.

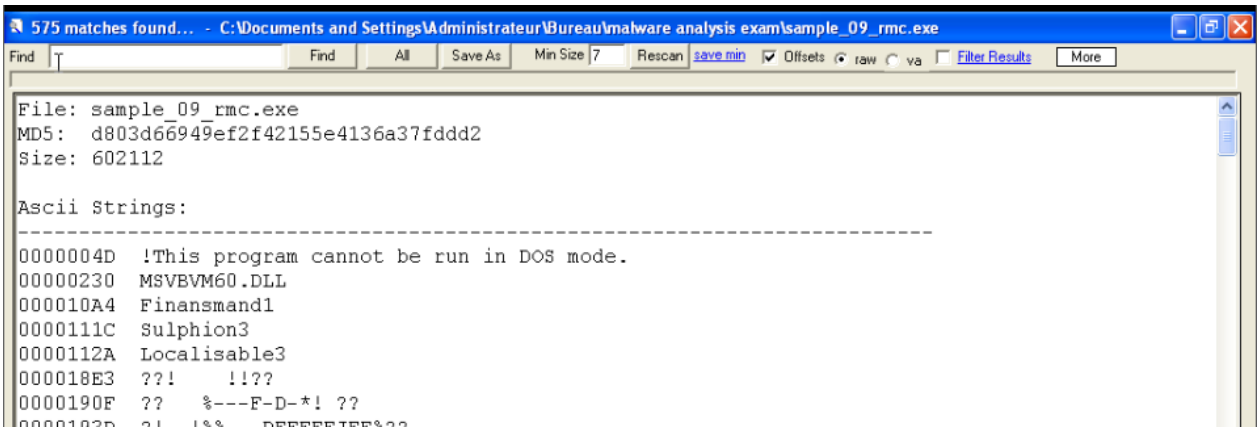
Interesting strings

Approach

With right click on file → strings

Procedure:

1. Right click on file and click on strings
2. Observe the strings and see if any strings are interesting providing insights.



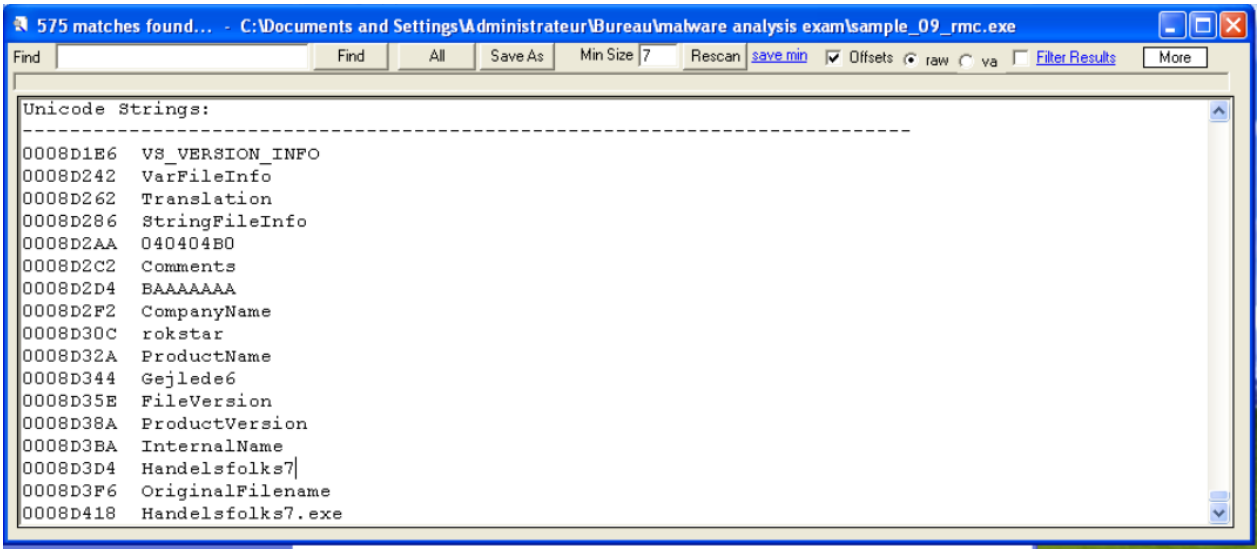
Below strings seems interesting:

```
00007F38  C:\Program Files (x86)\Microsoft Visual Studio\VB98\VB6.OLB
000080E4  IPHlpApi
000080F4  GetAdaptersInfo
0000813C  kernel32
0000814C  GetPath
0000818C  OpenMutexA
00008198  Krills
000081DC  SetWindowPos
00008224  WaitMessage
00008268  VirtualProtect
000082B0  user32
000082BC  EnumThreadWindows
0000836C  VBA6.DLL

0006DFEA  KERNEL32
0006E2A7  HeapAlloc
0006E432  WriteProfileStringW

0008BF94  MSVBVM60.DLL
0008BFA1  MethCallEngine
0008BFB6  EVENT_SINK_AddRef
0008BFCA  DllFunctionCall
0008BFDC  EVENT_SINK_Release
0008BFF2  EVENT_SINK_QueryInterface
0008C00E  __vbaExceptHandler
```

String	Description
OpenMutexA	The handle returned by **CreateMutex** has the **MUTEX_ALL_ACCESS** access right; it can be used in any function that requires a handle to a mutex object, provided that the caller has been granted access.
GetAdaptersInfo	Retrieves adapter information for the local computer.
WriteProfileStringW	Copies a string into the specified section of the Win.ini file. If Win.ini uses Unicode characters, the function writes Unicode characters to the file. Otherwise, the function writes ANSI characters.
EnumThreadWindows	Enumerates all nonchild windows associated with a thread by passing the handle to each window, in turn, to an application-defined callback function
SetWindowPos	Changes the size, position, and Z order of a child, pop-up, or top-level window. These windows are ordered according to their appearance on the screen. The topmost window receives the highest rank and is the first window in the Z order.
VirtualProtect	Changes the protection on a region of committed pages in the virtual address space of the calling process.

Source: https://docs.microsoft.com/en-us/

3. Check Unicode strings



It gives the information about the file, company, version, product.

PE file format analysis

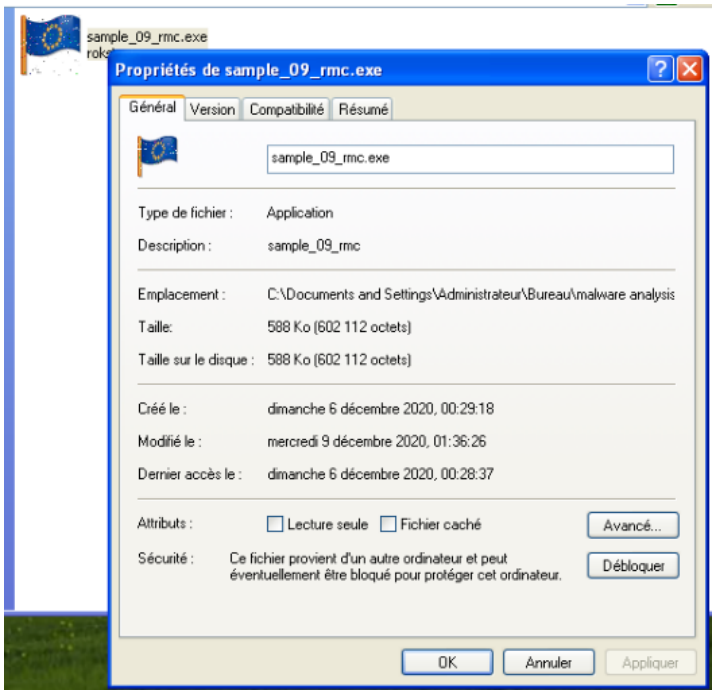
To check if executable is signed

Approach

Check signature in properties of file

Procedure:

1. Right click on file and select properties
2. Navigate through tabs and see if signature tab is present



For this file no signature tab found, so this executable is not signed.

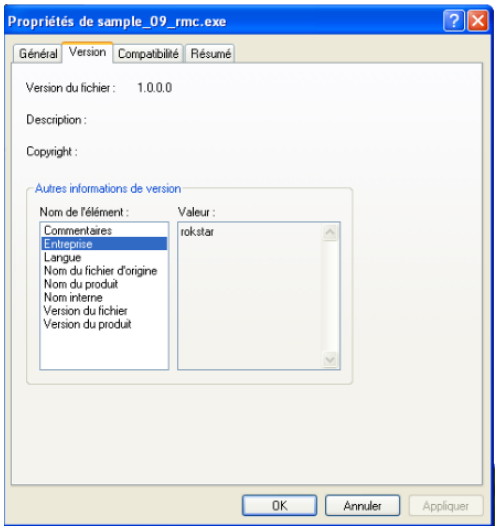
Check the metadata of the executable

Approach

Right click on file

Procedure:

1. Right click on file and select properties
2. Navigate through tabs and look for the version tab.



Below details are found for the executable:

- InternalName: Handelsfolks7
- Language: Chinese (Taiwan)
- FileVersion: 1.00
- CompanyName: rokstar
- Comments: BAAAAAAA
- ProductName: Gejlede6
- ProductVersion: 1.00
- OriginalFilename: Handelsfolks7.exe

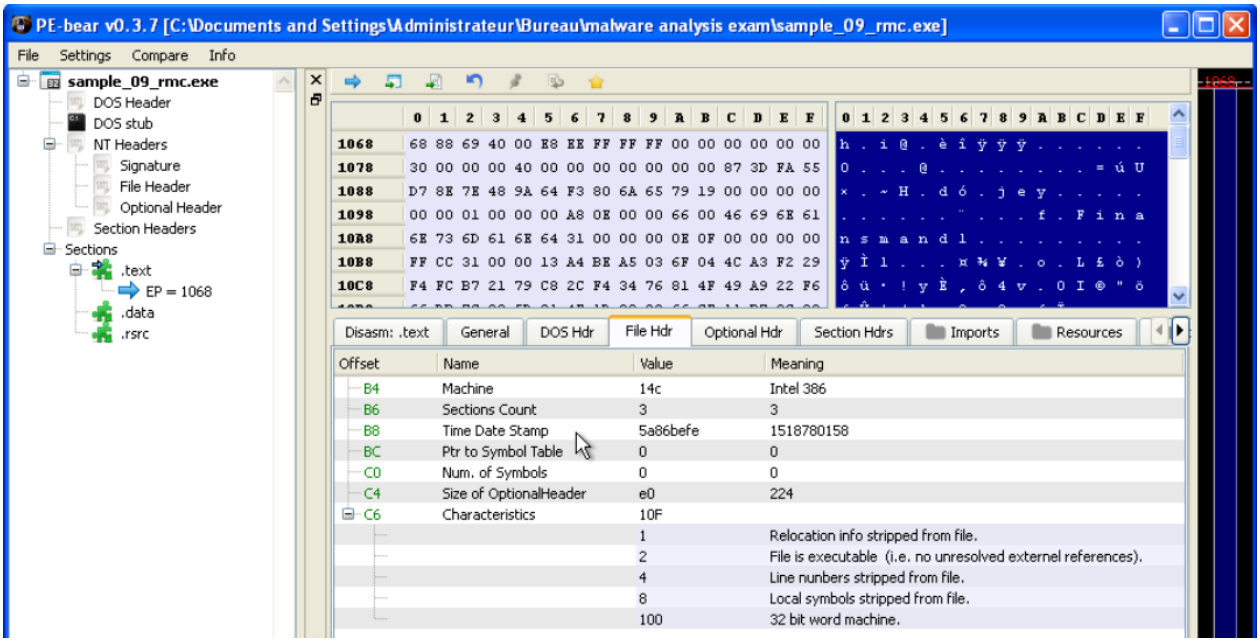
Timestamp of compilation and architecture

Approach

Using Tool: PE Bear

Procedure:

1. Open file using PE bear
2. Observe the file header and look for the values



Below details are found for the executable:

Compilation time: 1518780158 (Friday, February 16, 2018 11:22:38 AM)

Architecture: Windows 32 bits

Description: PE32 exec, Intel 386

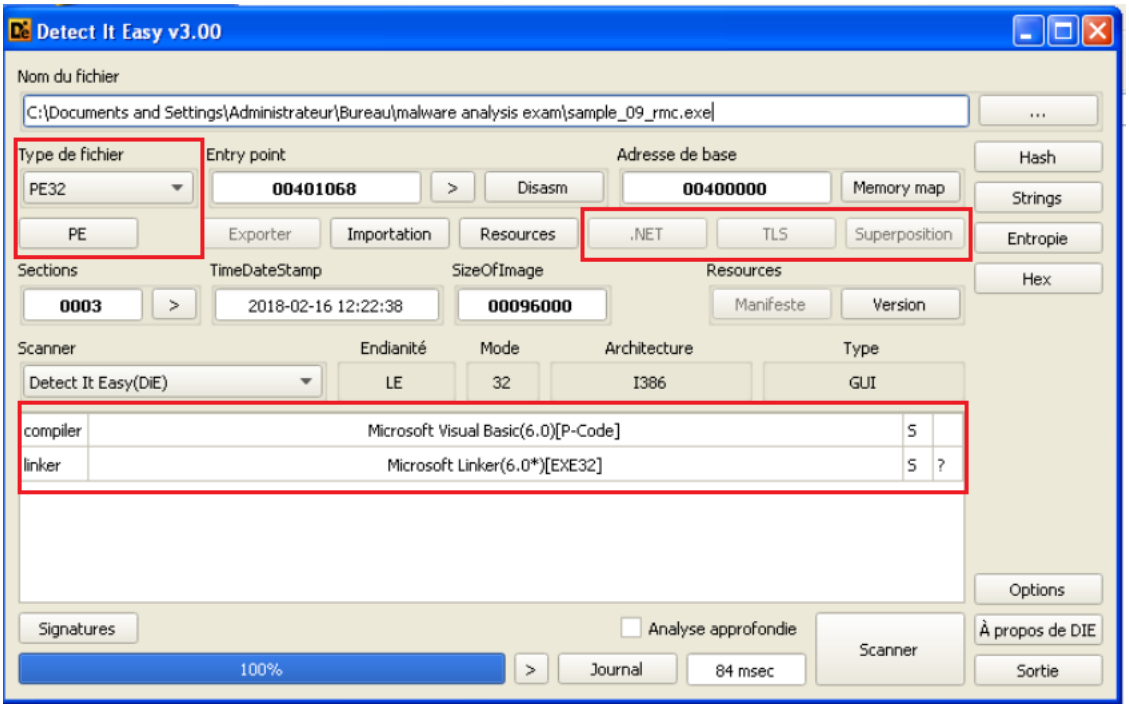
To check whether the file is executable or DLL

Approach

Using Tool: Detect it Easy

Procedure:

1. Open file using Detect it Easy
2. Check for the below highlighted details



The file is executable. Not a .NET file.

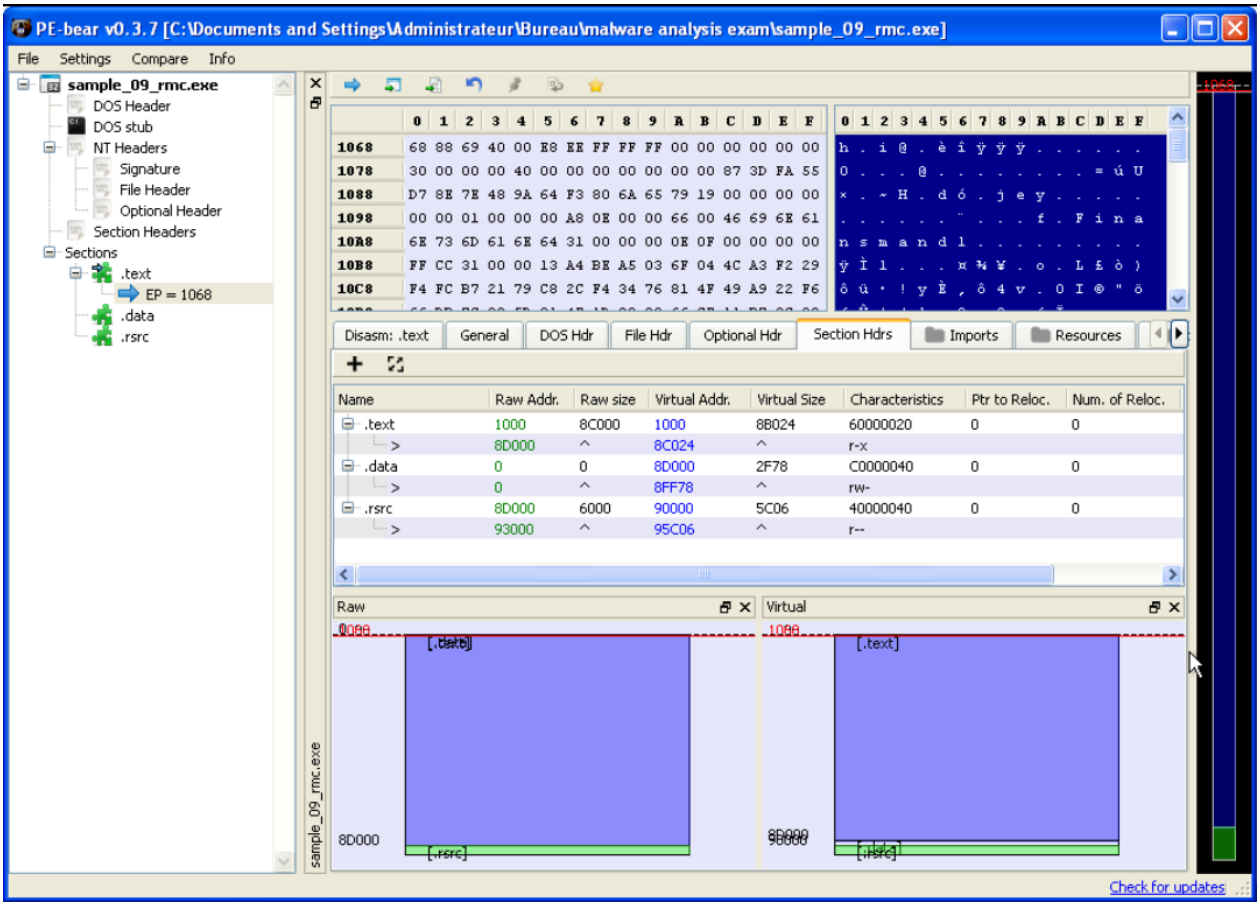
Sections and Entry point

Approach

Using Tool: PE Bear

Procedure:

1. Open file using PE bear
2. Navigate to the section Headers tab



The section header contains three components, text, data and resource.

Section name	Virtual size	Raw size	Permission
Text	8B024	8C000	Read, Exec
Data	2F78	0	Read, Write
Resource	5C06	6000	Read

Here data section seems suspicious as its raw size is 0 and virtual size showing some bits. Also text section size reduced by some so it is possible that data section is embedded with text section and data section have read and write permission so its probably writing some registries.

The entry point is located at 1068 which is in text section.

Imported DLL and APIs

Approach

Using Tool: PE Bear

Procedure:

1. Open file using PE bear
2. Navigate to the imports tab

Offset	Name	Func. Count	Bound?	OriginalFirstThunk	TimeDateStamp	Forwarder	NameRV
8BF44	MSVBVM60.DLL	9	TRUE	8BF6C	FFFFFFFF	FFFFFFFF	8BF94

Details

Call via	Name	Ordinal	Original Thunk	Thunk	Forwarder	Hint
1000	MethCallEngine	-	8BFA2	72A43B68	-	0
1004	-	253	80000253	72A1D132	-	-
1008	EVENT_SINK_Ad...	-	8BFB4	72A09B74	-	0
100C	DllFunctionCall	-	8BFC8	7294A0FD	-	0
1010	EVENT_SINK_Rel...	-	8BFDA	72A09B87	-	0
1014	EVENT_SINK_Qu...	-	8BFF0	72A09A85	-	0
1018	__vbaExceptHan...	-	8C00C	72A247DF	-	0
101C	-	284	80000284	72A1DE99	-	-
1020	-	64	80000064	729435A4	-	-

MethCallEngine API: possibly because it is compiled as Visual Basic p-code

Dynamic Analysis

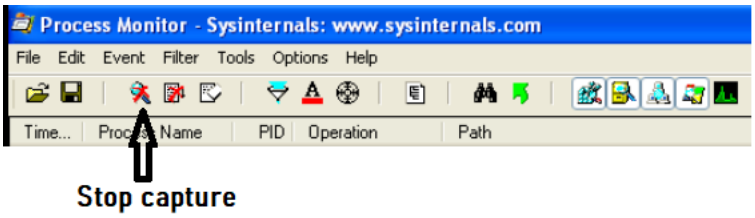
Malware's process name

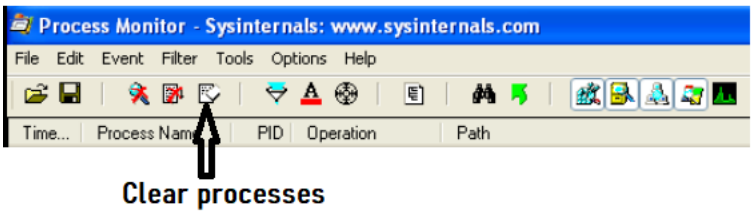
Approach

Process Hacker and Process Monitor

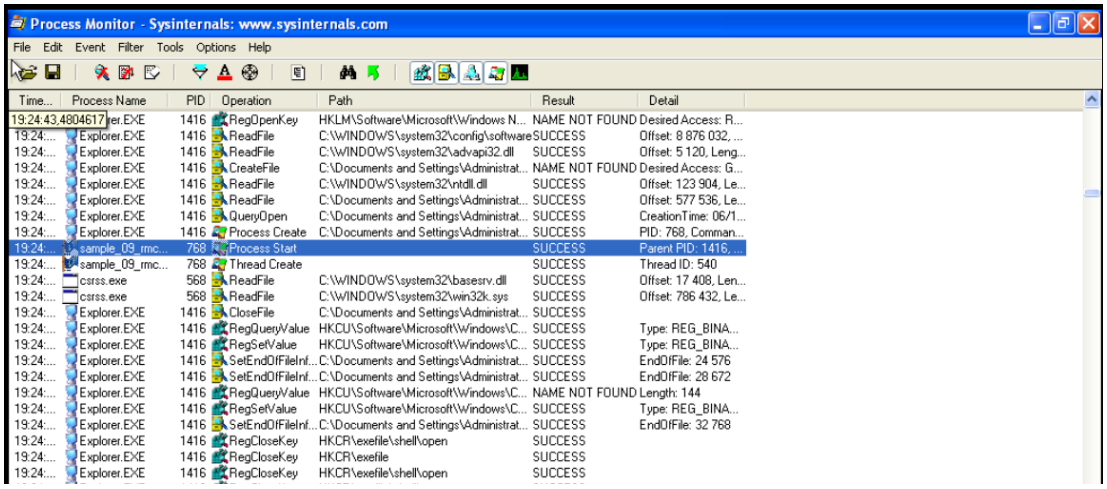
Procedure:

1. Open Process Hacker and proc mon in parallel.
2. Stop the capture and clear the processes

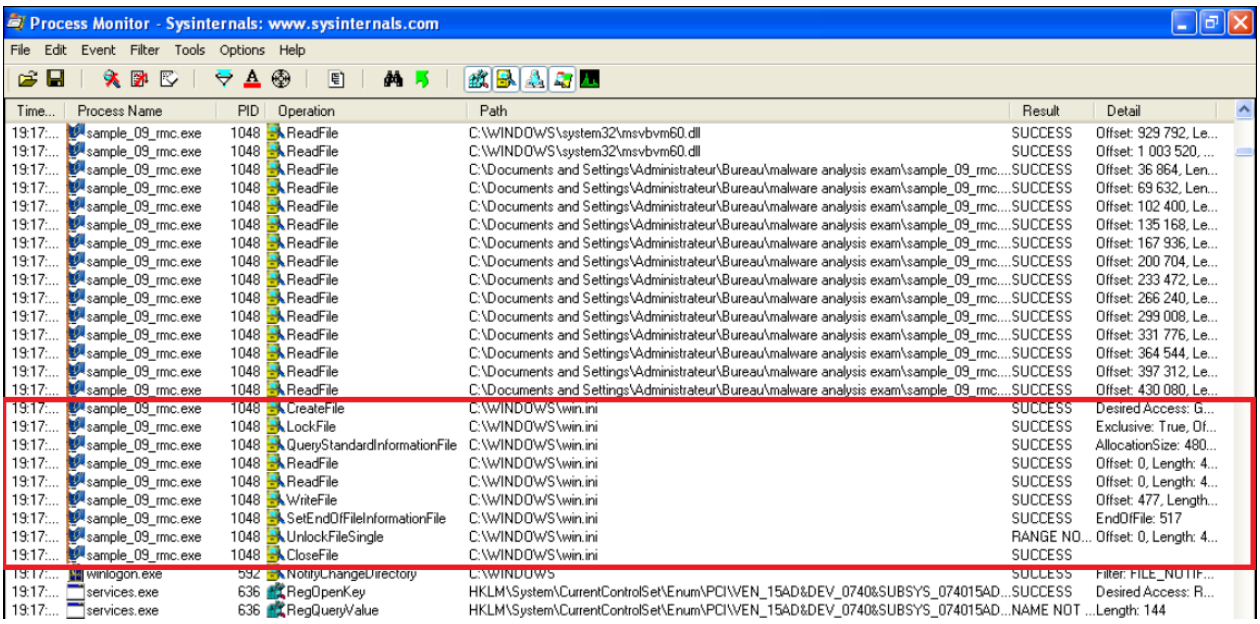




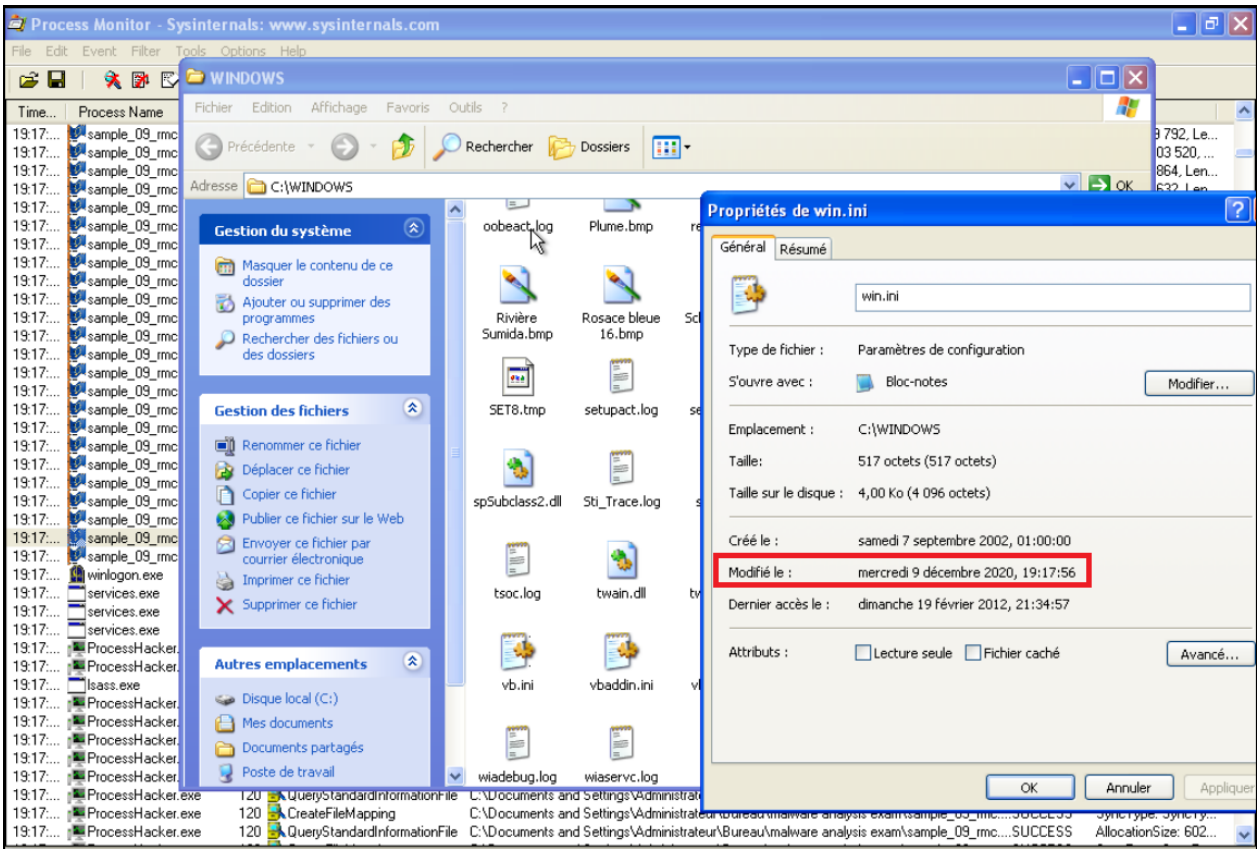
3. Launch malware file and observe process hacker.
4. In parallel Stop the capture again on proc mon and search for the corresponding process



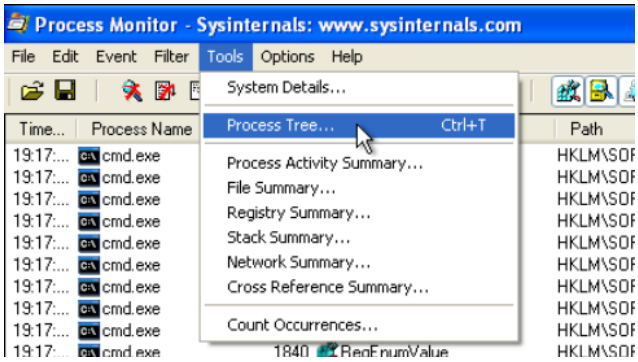
Located the process of the malware file where operation “Process start” with PID 768. Further explored the processes that creates the thread.



Here we can see, malware tries and succeed to modify the win.ini file which is responsible for storing the boot settings. If we see the timestamp of the file on the location, we can see its get modified.

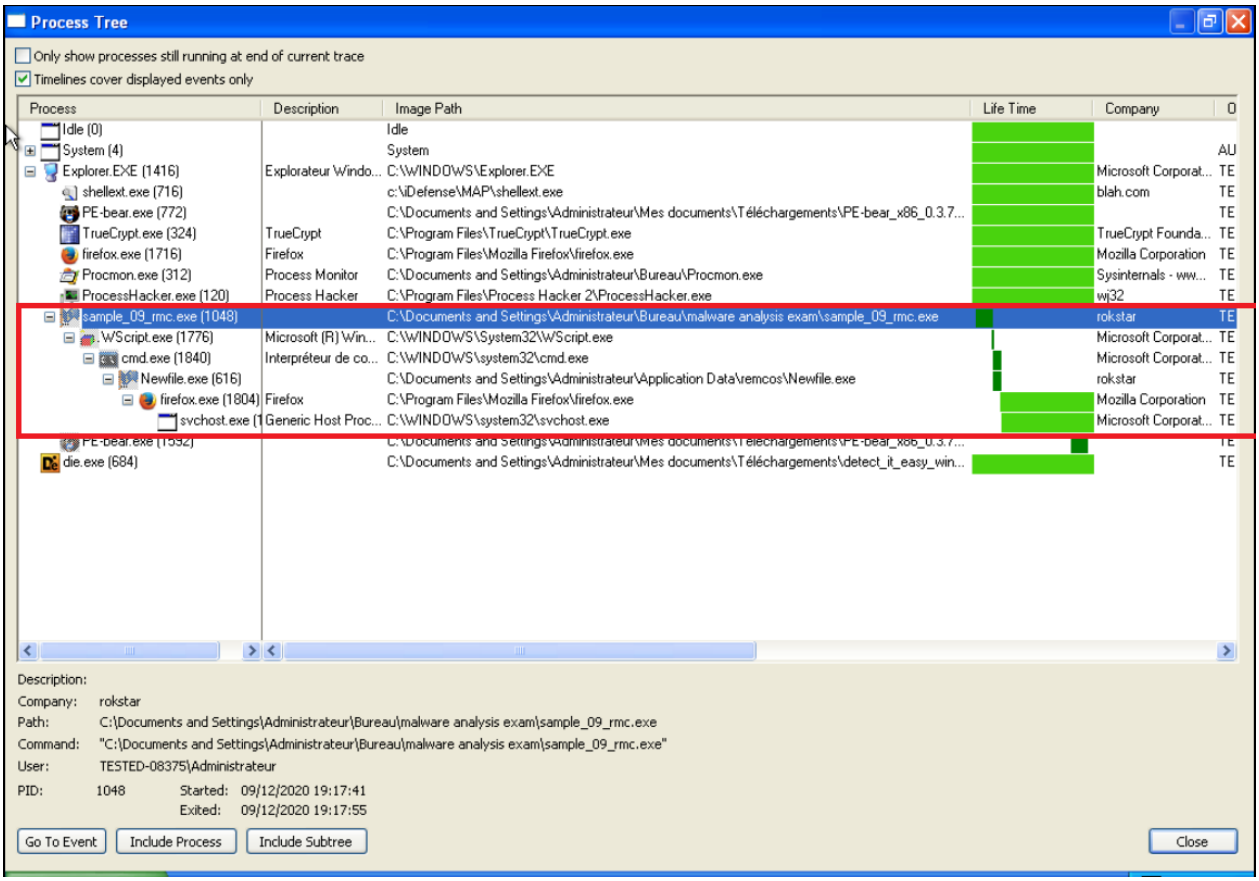


Further checking the process tree in proc mon, and observe the events due to the malware execution.

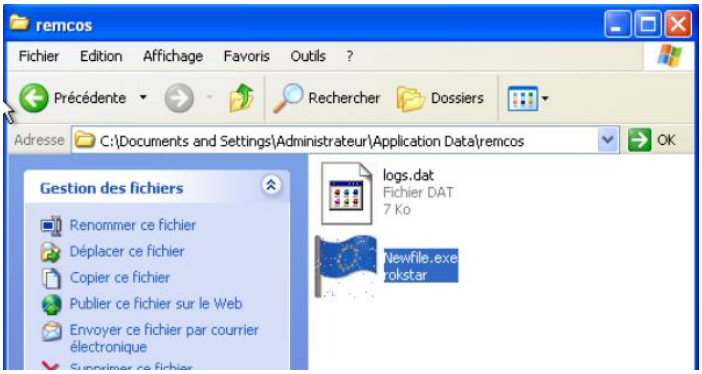


Observing process tree will give the processes that are invoked after execution of malware. The malware tries to take access to WScript and perform operations as mentioned in below screenshot.

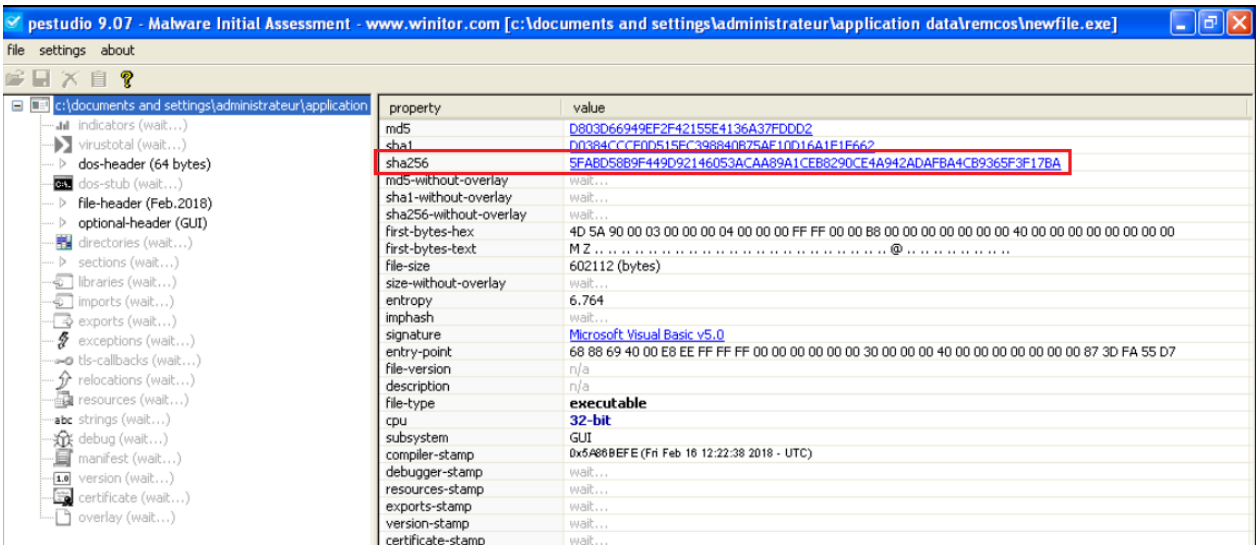
After that it executes the cmd and creates a duplicate file named “newfile.exe”



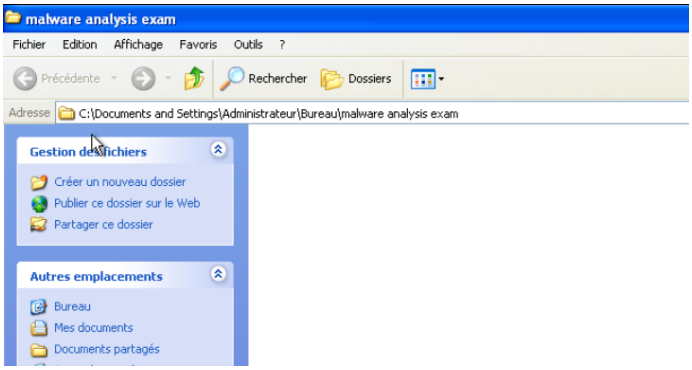
Search for the newfile.exe which is located at location



Check the hash SHA256 whether this is the same file or not.

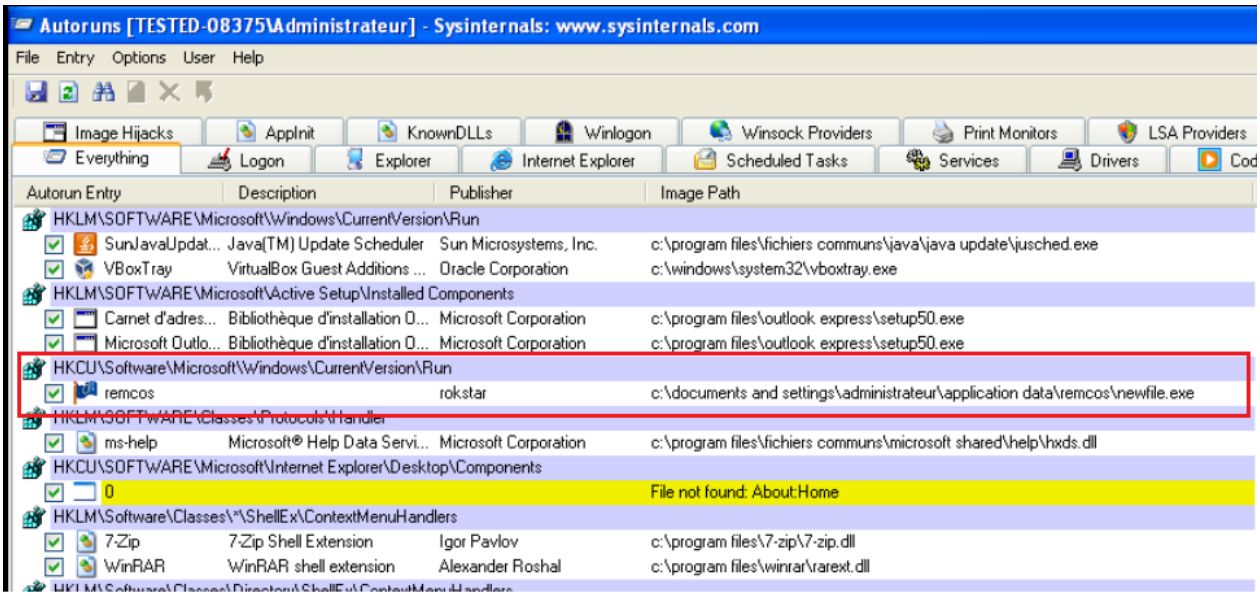


Yes, this is the same file as when the malware executed, the original file delete itself and make a duplicate in another location.



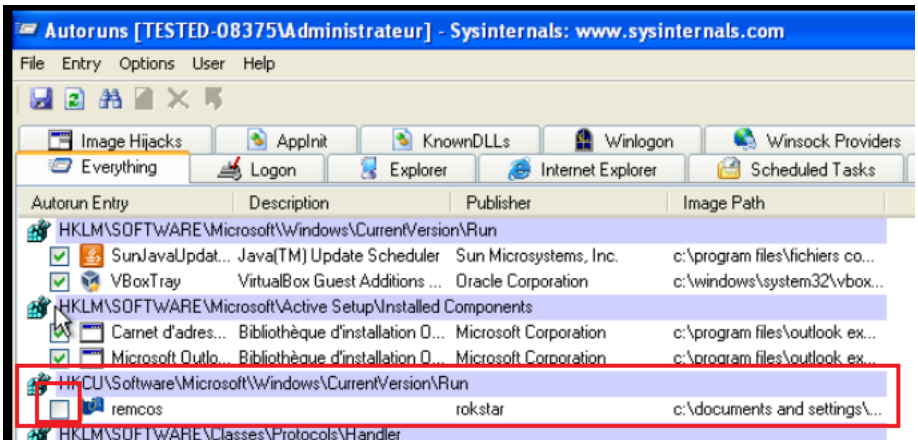
Original Location

Now, Check the run or autoruns, means whether malware is persistent or not.

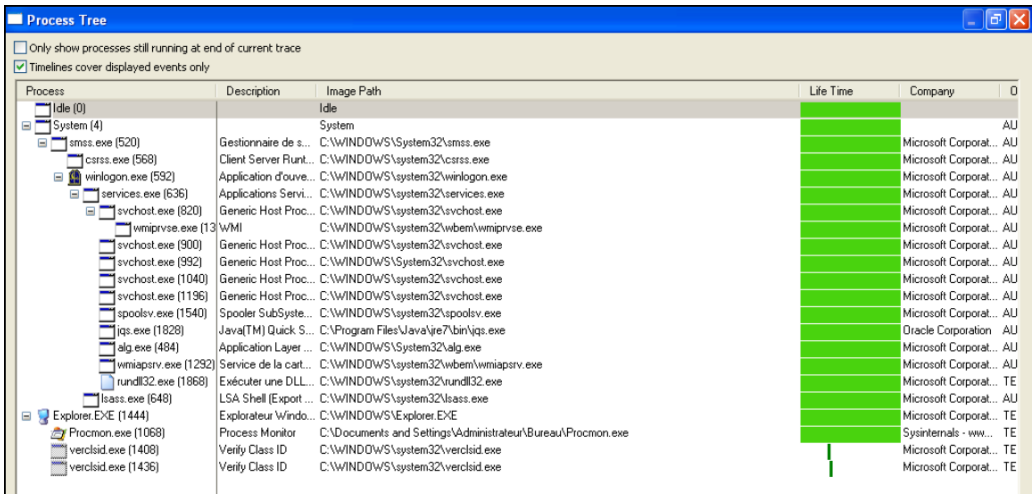


The malware is copied to run, where it will be running even after reboot

Deselect the checkbox for the entry and reboot the system.



After rebooting, check the proc mon and process tree.



The process is not running again after rebooting.

Further, Remove the newfile.exe and log.dat files from the detected location.

Scan from virusTotal

https://www.virustotal.com/gui/file/5fabd58b9f449d92146053acaa89a1ceb8290ce4a942adafba4cb9365f3f17ba/behavior

- This executable modifies contents in below files:
  - C:\WINDOWS\win.ini
  - C:\Documents and Settings\Administrator\Application Data\remcos\Newfile.exe
  - C:\Documents and Settings\Administrator\Application Data\remcos\logs.dat

- Processes created by the executable
  - C:\WINDOWS\system32\wscript.exe
  - C:\WINDOWS\system32\cmd.exe
  - C:\Program Files\Internet Explorer\IEXPLORE.EXE
  - C:\WINDOWS\system32\svchost.exe

- Processes terminated by the executable
  - C:\Documents and Settings\Administrator\Local Settings\Temp\EB93A6\996E.exe
  - C:\WINDOWS\system32\wscript.exe
  - C:\Documents and Settings\Administrator\Application Data\remcos\Newfile.exe
  - C:\WINDOWS\system32\cmd.exe

- Processes injected by executable
  - C:\Program Files\Internet Explorer\IEXPLORE.EXE
  - C:\WINDOWS\system32\svchost.exe