Static malware analysis report

Introduction

This report presents the results of a static malware analysis performed on a suspicious executable file.
The main goal of this exercise is to learn and understand how to perform a static analysis. Identifying how a file is structured, what tools can reveal about it, and how indicators of malicious behavior can be discovered without executing the file.

I am particularly interested in system logging, data analysis, and understanding how malware operates at a technical level.
By exploring the internal structure of this sample, I aim to strengthen my understanding of how executable files are built, how they interact with Windows, and what signs can reveal whether a program is malicious or harmless.

Objectives

- Perform basic static analysis on a given malware sample.
- Use and get familiar with tools such as HashMyFiles, CFF Explorer, Exeinfo PE, PEStudio etc.
- Understand malware naming schemes and identify the malware type, infected platform, family name, and group name.
- Be able to detect whether a malware sample has a valid code signing certificate.

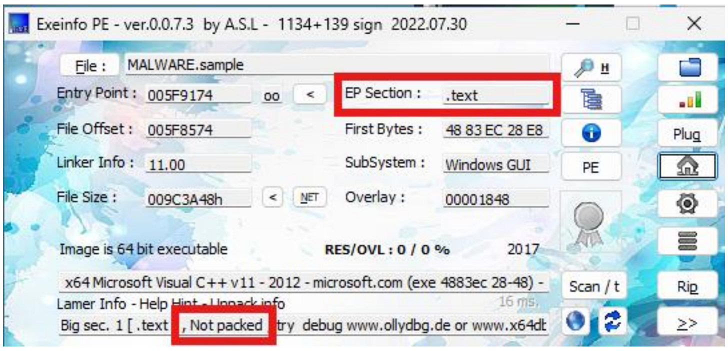
Tools used

- HashMyFiles
- CFF Explorer
- HxD
- Exeinfo PE
- PEStudio
- FlareVM Strings utility

#1 - Packed or unpacked analysis

In this analysis, I used Exeinfo PE to determine whether the malware sample is packed or unpacked. According to the results shown in the picture below, Exeinfo PE identifies the file as a 64-bit executable compiled with Microsoft Visual C++ v11 (2012). The tool also indicated that the sample is not packed.

Additionally, the Entry Point and the EP Section(.text) confirm that the executable code is located in the standard code section, which is typical for unpacked files.



Exeinfo PE - ver.0.0.7.3 by A.S.L - 1134+139 sign 2022.07.30

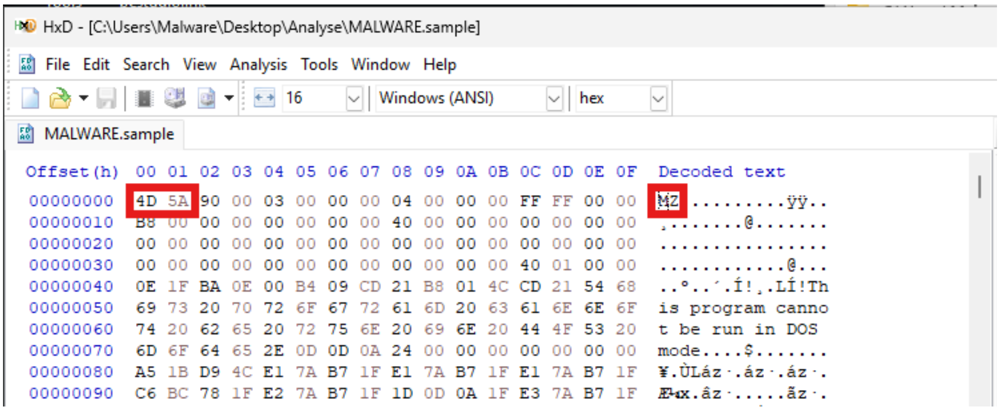
File : MALWARE.sample
Entry Point : 005F9174
EP Section : .text
File Offset : 005F8574
First Bytes : 48 83 EC 28 E8
Linker Info : 11.00
SubSystem : Windows GUI
File Size : 009C3A48h
Overlay : 00001848
Image is 64 bit executable
RES/OVL : 0 / 0 %
2017
x64 Microsoft Visual C++ v11 - 2012 - microsoft.com (exe 4883ec 28-48) -
Lamer Info - Help Hint - Unpack info
16 ms.
Big sec. 1 [.text , Not packed try debug www.ollydbg.de or www.x64dt

#2 – File format identification

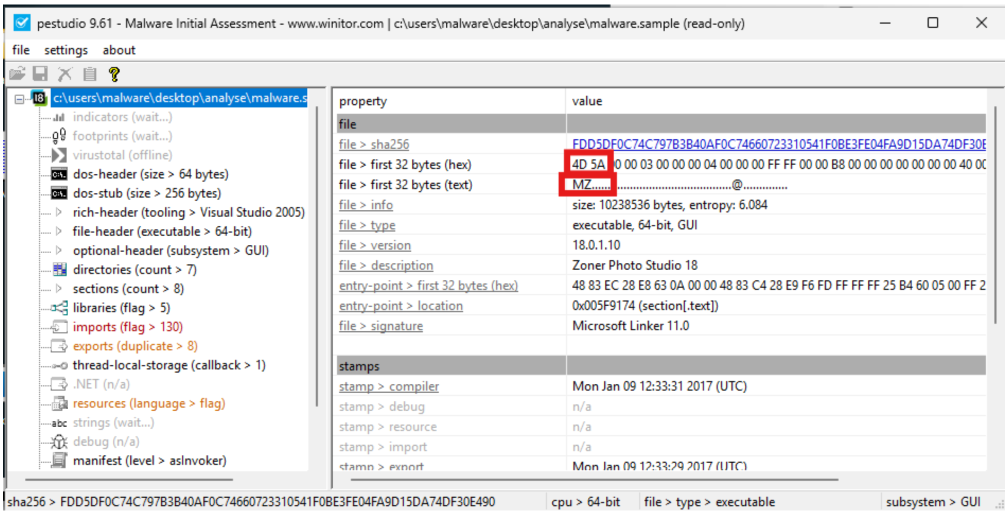
To determine the type of file and confirm whether it is a valid Windows executable, the sample was first examined using HxD(hex editor). As shown in the screenshots below, the first two bytes of the file are 4D 5A, which corresponds to the ASCII characters "MZ". This indicates that the file follows the Portable Executable (PE) format used by Windows executables and DLL files.

Further verification was performed using PEStudio, which automatically parses the PE headers. The tool confirmed that the file is a 64-bit executable with a Windows GUI subsystem. The linker version 11.00 suggests that it was compiled using Microsoft Visual Studio 2012. Additionally, PEStudio shows an entropy value of 6.084, which is within the normal range for unpacked executables.

Based on these results, we can conclude that the sample is a valid Windows Portable Executable (PE) file and is not corrupted or obfuscated at the header level.


HxD - [C:\Users\Malware\Desktop\Analyse\MALWARE.sample]
File Edit Search View Analysis Tools Window Help
16
Windows (ANSI)
hex
MALWARE.sample

```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text
00000000  4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00  MZ..........ÿÿ..
00000010  B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00  ¸.......@.......
00000020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000030  00 00 00 00 00 00 00 00 00 00 00 00 40 01 00 00  ............@...
00000040  0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68  ..º..´.Í!¸.LÍ!Th
00000050  69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F  is program canno
00000060  74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20  t be run in DOS 
00000070  6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00  mode....$.......
00000080  A5 1B D9 4C E1 7A B7 1F E1 7A B7 1F E1 7A B7 1F  ¥.ÙLáz·.áz·.áz·.
00000090  C6 BC 78 1F E2 7A B7 1F 1D 0D 0A 1F E3 7A B7 1F  Æ¼x.âz·.....ãz·.
```


pestudio 9.61 - Malware Initial Assessment - www.winitor.com | c:\users\malware\desktop\analyse\malware.sample (read-only)
file settings about

- c:\users\malware\desktop\analyse\malware.s
  - indicators (wait...)
  - footprints (wait...)
  - virustotal (offline)
  - dos-header (size > 64 bytes)
  - dos-stub (size > 256 bytes)
  - rich-header (tooling > Visual Studio 2005)
  - file-header (executable > 64-bit)
  - optional-header (subsystem > GUI)
  - directories (count > 7)
  - sections (count > 8)
  - libraries (flag > 5)
  - imports (flag > 130)
  - exports (duplicate > 8)
  - thread-local-storage (callback > 1)
  - .NET (n/a)
  - resources (language > flag)
  - strings (wait...)
  - debug (n/a)
  - manifest (level > asInvoker)

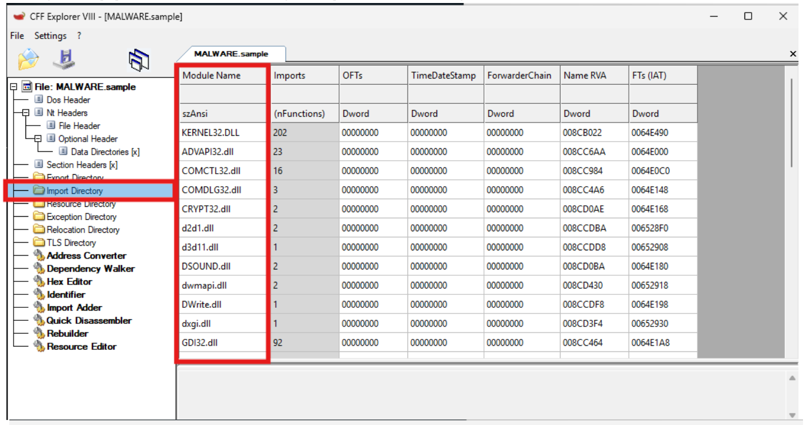
property	value
file	
file > sha256	FDD5DF0C74C797B3B40AF0C74660723310541F0BE3FE04FA9D15DA74DF30E
file > first 32 bytes (hex)	4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 B8 00 00 00 00 00 00 00 40 0C
file > first 32 bytes (text)	MZ...................................@.............
file > info	size: 10238536 bytes, entropy: 6.084
file > type	executable, 64-bit, GUI
file > version	18.0.1.10
file > description	Zoner Photo Studio 18
entry-point > first 32 bytes (hex)	48 83 EC 28 E8 63 0A 00 00 48 83 C4 28 E9 F6 FD FF FF FF 25 B4 60 05 00 FF 2
entry-point > location	0x005F9174 (section[.text])
file > signature	Microsoft Linker 11.0
stamps	
stamp > compiler	Mon Jan 09 12:33:31 2017 (UTC)
stamp > debug	n/a
stamp > resource	n/a
stamp > import	n/a
stamp > export	Mon Jan 09 12:33:29 2017 (UTC)

sha256 > FDD5DF0C74C797B3B40AF0C74660723310541F0BE3FE04FA9D15DA74DF30E490 | cpu > 64-bit | file > type > executable | subsystem > GUI

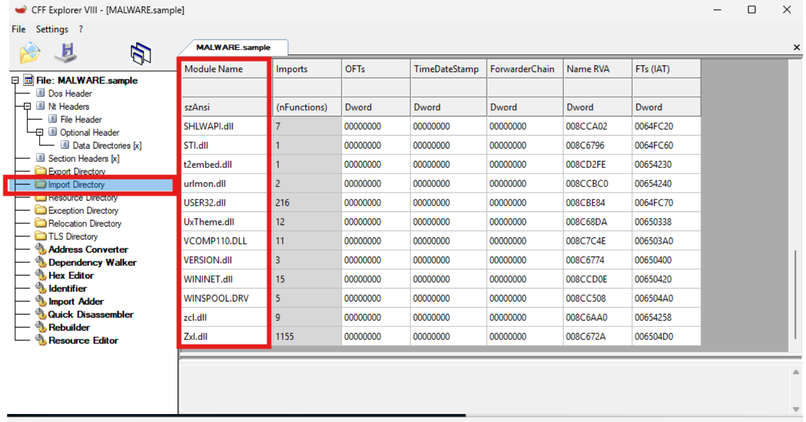
#3 – Identifying libraries and packages for file execution

The imported libraries required for the executable to run were examined using CFF Explorer. The import table shows that the sample depends on standard Windows system libraries such as kernel32.dll, user32.dll, advapi32.dll, and wininet.dll, which are typical for legitimate applications. These libraries handle basic functions like file operations, registry access, and network communication.

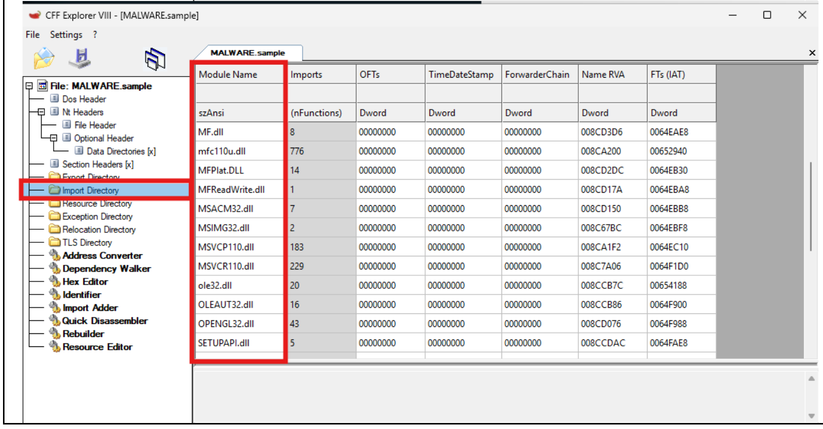
No third-party or missing libraries were detected, indicating that the sample can execute normally on a standard Windows system.
However, the presence of networking-related APIs (e.g., InternetOpen, HttpSendRequest) may suggest that the program is capable of external communication.



Module Name	Imports	OFTs	TimeDateStamp	ForwarderChain	Name RVA	FTs (IAT)
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword	Dword
KERNEL32.DLL	202	00000000	00000000	00000000	008CB022	0064E490
ADVAPI32.dll	23	00000000	00000000	00000000	008CC6AA	0064E000
COMCTL32.dll	16	00000000	00000000	00000000	008CC984	0064E0C0
COMDLG32.dll	3	00000000	00000000	00000000	008CC4A6	0064E148
CRYPT32.dll	2	00000000	00000000	00000000	008CD0AE	0064E168
d2d1.dll	2	00000000	00000000	00000000	008CCDBA	006528F0
d3d11.dll	1	00000000	00000000	00000000	008CCDD8	00652908
DSOUND.dll	2	00000000	00000000	00000000	008CD0BA	0064E180
dwmapi.dll	2	00000000	00000000	00000000	008CD430	00652918
DWrite.dll	1	00000000	00000000	00000000	008CCDF8	0064E198
dxgi.dll	1	00000000	00000000	00000000	008CD3F4	00652930
GDI32.dll	92	00000000	00000000	00000000	008CC464	0064E1A8



Module Name	Imports	OFTs	TimeDateStamp	ForwarderChain	Name RVA	FTs (IAT)
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword	Dword
SHLWAPI.dll	7	00000000	00000000	00000000	008CCA02	0064FC20
STI.dll	1	00000000	00000000	00000000	008C6796	0064FC60
t2embed.dll	1	00000000	00000000	00000000	008CD2FE	00654230
urlmon.dll	2	00000000	00000000	00000000	008CCBC0	00654240
USER32.dll	216	00000000	00000000	00000000	008CBE84	0064FC70
UxTheme.dll	12	00000000	00000000	00000000	008C68DA	00650338
VCOMP110.DLL	11	00000000	00000000	00000000	008C7C4E	006503A0
VERSION.dll	3	00000000	00000000	00000000	008C6774	00650400
WININET.dll	15	00000000	00000000	00000000	008CCD0E	00650420
WINSPOOL.DRV	5	00000000	00000000	00000000	008CC508	006504A0
zcl.dll	9	00000000	00000000	00000000	008C6AA0	00654258
Zxl.dll	1155	00000000	00000000	00000000	008C672A	006504D0



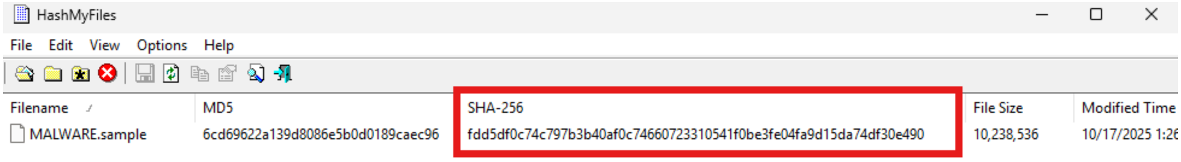
Module Name	Imports	OFTs	TimeDateStamp	ForwarderChain	Name RVA	FTs (IAT)
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword	Dword
MF.dll	8	00000000	00000000	00000000	008CD3D6	0064EAE8
mfc110u.dll	776	00000000	00000000	00000000	008CA200	00652940
MFPlat.DLL	14	00000000	00000000	00000000	008CD2DC	0064EB30
MFReadWrite.dll	1	00000000	00000000	00000000	008CD17A	0064EBA8
MSACM32.dll	7	00000000	00000000	00000000	008CD150	0064EBB8
MSIMG32.dll	2	00000000	00000000	00000000	008C67BC	0064EBF8
MSVCP110.dll	183	00000000	00000000	00000000	008CA1F2	0064EC10
MSVCR110.dll	229	00000000	00000000	00000000	008C7A06	0064F1D0
ole32.dll	20	00000000	00000000	00000000	008CCB7C	00654188
OLEAUT32.dll	16	00000000	00000000	00000000	008CCB86	0064F900
OPENGL32.dll	43	00000000	00000000	00000000	008CD076	0064F988
SETUPAPI.dll	5	00000000	00000000	00000000	008CCDAC	0064FAE8

#4 – Calculating the hash of the file

To calculate the hash of the file, I used the tool HashMyFiles. To check when the file was last analysed, I used VirusTotal.

The calculated SHA-256 hash was:
FDD5D0FC74C797B3B40AFOC74660723310541F0BE3FE04F9D15DA74DF30E4F90.

When submitted to VirusTotal, 52 out of 70 antivirus engines flagged the file as malicious, primarily identifying it as a Trojan.Win32.Starter variant.



Filename	MD5	SHA-256	File Size	Modified Time
MALWARE.sample	6cd69622a139d8086e5b0d0189caec96	fdd5df0c74c797b3b40af0c74660723310541f0be3fe04fa9d15da74df30e490	10,238,536	10/17/2025 1:2

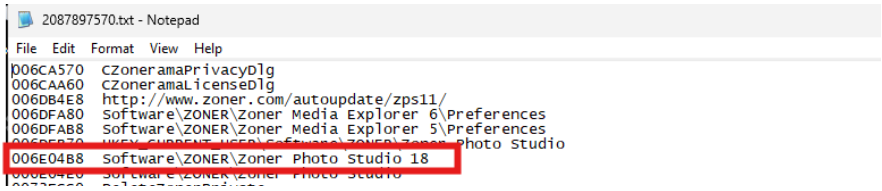
#5 – Identifying suspicious strings

During the static analysis phase, I extracted and examined readable strings from the malware sample to identify possible indicators of malicious activity, using the “Strings” utility in Flare VM.

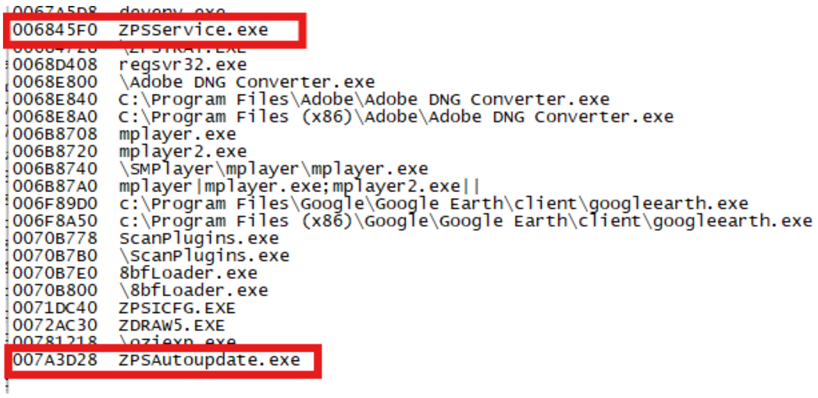
The main goal was to look for signs of network communication, persistence mechanisms, process injection, or anti-analysis techniques. Specifically, I searched for URLs, IP addresses, registry keys, Windows API functions (e.g., CreateRemoteThread, VirtualAlloc, WriteProcessMemory), and references to legitimate Windows utilities such as regsvr32.exe or rundll32.exe which are commonly abused by malware.

From the extracted strings, several entries referred to Zoner Photo Studio (e.g., Software\ZONER\Zoner Photo Studio 18, ZPSAutoupdate.exe, ZPSService.exe), which suggests that the binary is related to that legitimate application.
Some strings such as regsvr32.exe and vmx_fb.dll were noted as potentially suspicious, as they can be used in malicious contexts to register or load harmful DLLs. However, no clear evidence of obfuscation, command-and-control (C2) domains, or injection-related API calls was found in the visible strings.

Based on these observations, the sample appears to contain both legitimate software components and potentially exploitable functionality. Further dynamic analysis would be required to determine if regsvr32.exe or any DLLs are being used maliciously during execution.



```
006CA570  CZoneramaPrivacyDlg
006CAA60  CZoneramaLicenseDlg
006DB4E8  http://www.zoner.com/autoupdate/zps11/
006DFA80  Software\ZONER\Zoner Media Explorer 6\Preferences
006DFAB8  Software\ZONER\Zoner Media Explorer 5\Preferences
006DFB70  HKEY_CURRENT_USER\Software\ZONER\Zoner Photo Studio
006E04B8  Software\ZONER\Zoner Photo Studio 18
006E04E0  Software\ZONER\Zoner Photo Studio
0072F6C0  DeleteZonerPrivate
```

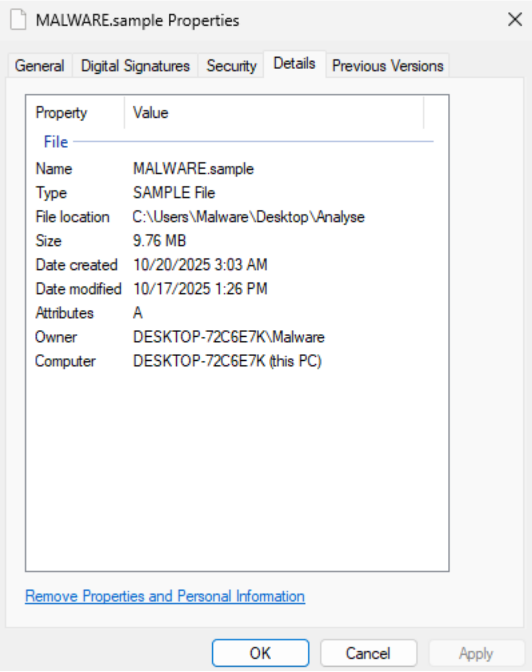


```
0067A5D8  devenv.exe
006845F0  ZPSService.exe
00684720  \ZPSTRAY.EXE
0068D408  regsvr32.exe
0068E800  \Adobe DNG Converter.exe
0068E840  C:\Program Files\Adobe\Adobe DNG Converter.exe
0068E8A0  C:\Program Files (x86)\Adobe\Adobe DNG Converter.exe
006B8708  mplayer.exe
006B8720  mplayer2.exe
006B8740  \SMPlayer\mplayer\mplayer.exe
006B87A0  mplayer|mplayer.exe;mplayer2.exe||
006F89D0  C:\Program Files\Google\Google Earth\client\googleearth.exe
006F8A50  C:\Program Files (x86)\Google\Google Earth\client\googleearth.exe
0070B778  ScanPlugins.exe
0070B7B0  \ScanPlugins.exe
0070B7E0  8bfLoader.exe
0070B800  \8bfLoader.exe
0071DC40  ZPSICFG.EXE
0072AC30  ZDRAW5.EXE
00781218  \oziexp.exe
007A3D28  ZPSAutoupdate.exe
```

#6 – Identifying code signing certificate

When examining the file properties, no code signing certificate was found under the Digital Signatures tab. Additionally, the Details tab does not show any publisher or product information, indicating that the file is unsigned and lacks verified authorship.

Unsigned executables are common in malware, as threat actors often avoid using legitimate code signing certificates to prevent attribution and detection. This increases the likelihood that the file is malicious or untrusted.



Property	Value
File	
Name	MALWARE.sample
Type	SAMPLE File
File location	C:\Users\Malware\Desktop\Analyse
Size	9.76 MB
Date created	10/20/2025 3:03 AM
Date modified	10/17/2025 1:26 PM
Attributes	A
Owner	DESKTOP-72C6E7K\Malware
Computer	DESKTOP-72C6E7K (this PC)

Conclusion

Through this static malware analysis, the sample was identified as a 64-bit Windows Portable Executable (PE) compiled with Microsoft Visual C++ 2012.
The file was confirmed to be unpacked, contained a normal PE header structure, and relied primarily on standard Windows system libraries.
String analysis revealed several references to Zoner Photo Studio, indicating that the sample may originate from or impersonate a legitimate application.
However, suspicious entries such as regsvr32.exe and vmx_fb.dll suggest potential misuse for malicious activity.

According to VirusTotal detections, the sample belongs to the *Starter* malware family, targeting the Windows platform.

No code signing certificate or publisher information was found, increasing the likelihood that the file is untrusted or malicious.

While the static analysis provided strong indicators of the file’s purpose and origin, dynamic analysis would be required to confirm its runtime behavior, persistence mechanisms, and possible network communication with remote servers.