Static Malware Analysis

By Jack McGrath

Malware analysis is the process of examining the behavior and purpose of a malicious file using various methods to assess its functionality and potential risks. This analysis can provide insights for future detection and prevention, making it a key component in cybersecurity and incident response.

Typically malware analysis is split up into three categories:

1. Static Analysis
   This involves examining the code or structure of a malicious file without executing it. This is done by inspecting file names, hashes, strings such as IP addresses, domains, and file header data. Tools like disassemblers and network analyzers can be used to observe the malware without actually running it in order to collect information on how the malware works.

2. Dynamic Analysis
   Dynamic analysis involves executing the code in a safe and controlled environment called a sandbox. This allows for closer inspection on its behaviour, network activity and it's interactions with the system.

3. Hybrid Analysis
   This involves a combination of the previous two methods, static and dynamic analysis. Leveraging the strengths of both methods, this can provide a more comprehensive understanding of the malware.

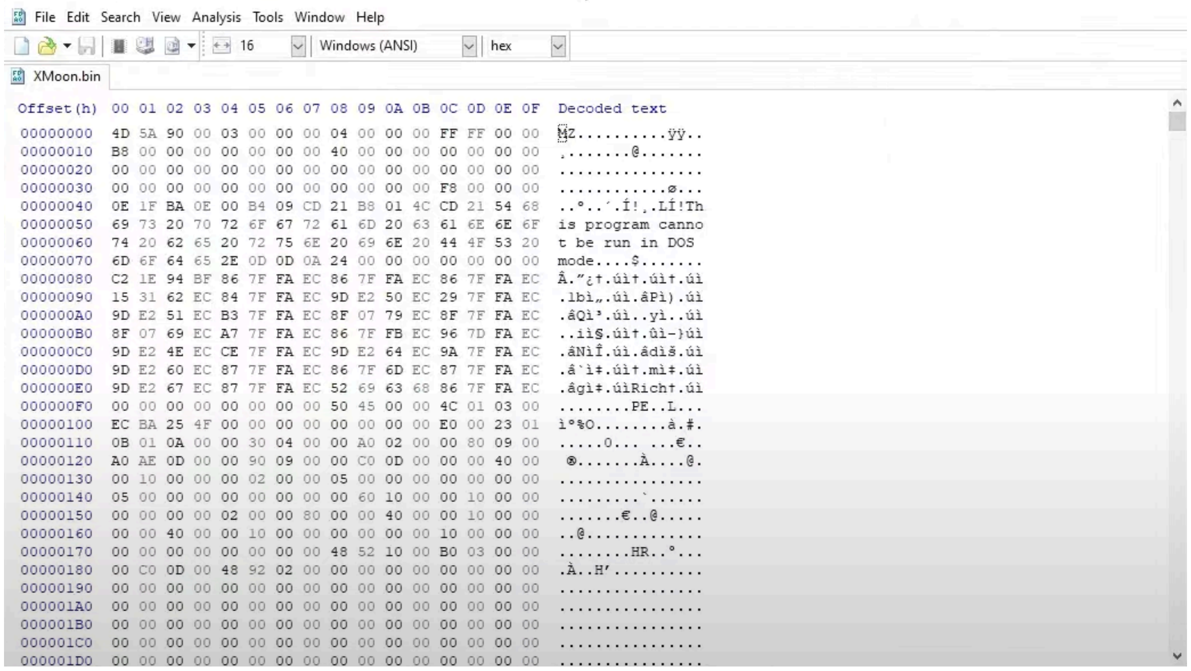
This document will focus on the static analysis of a malware commonly known as "XMoon.exe".

Static Analysis Methodology
The first step I took in conducting this analysis was the use of a virtual machine (VM). A virtual machine is a software-based emulation of a separate physical computer, which allows it to run an operating system and applications as though they were on a distinct machine. This approach helps to isolate the malware and prevents it from affecting my host computer. I then utilized FLARE VM to create a secure analysis environment, as it provides all the necessary tools for effective malware analysis.

When I extracted the XMoon file, it was saved as a .bin file. Changing the extension to .bin ensures that the file cannot be double-clicked or accidentally executed, adding an extra layer of safety during analysis.

The next step in the process was utilising a tool called HxD.

HxD is a freeware hex, disk and memory-editor developed in 2003.
It can open files larger than 4 GiB and open and edit the raw contents of disk drives, as well as display and edit the memory used by running processes. Among other features, it can calculate various checksums, compare files, or shred files. [1] - wikipedia
When XMoon.bin is loaded into HxD the following data appears:



```
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 F8 00 00 00  ............ø...
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  C2 1E 94 BF 86 7F FA EC 86 7F FA EC 86 7F FA EC  Â.”¿†.úì†.úì†.úì
00000090  15 31 62 EC 84 7F FA EC 9D E2 50 EC 29 7F FA EC  .1bì„.úì.âPì).úì
000000A0  9D E2 51 EC B3 7F FA EC 8F 07 79 EC 8F 7F FA EC  .âQì³.úì..yì..úì
000000B0  8F 07 69 EC A7 7F FA EC 86 7F FB EC 96 7D FA EC  ..iì§.úì†.ûì–}úì
000000C0  9D E2 4E EC CE 7F FA EC 9D E2 64 EC 9A 7F FA EC  .âNìÎ.úì.âdìš.úì
000000D0  9D E2 60 EC 87 7F FA EC 86 7F 6D EC 87 7F FA EC  .â`ì‡.úì†.mì‡.úì
000000E0  9D E2 67 EC 87 7F FA EC 52 69 63 68 86 7F FA EC  .âgì‡.úìRich†.úì
000000F0  00 00 00 00 00 00 00 00 50 45 00 00 4C 01 03 00  ........PE..L...
00000100  EC BA 25 4F 00 00 00 00 00 00 00 00 E0 00 23 01  ìº%O........à.#.
00000110  0B 01 0A 00 00 30 04 00 00 A0 02 00 00 80 09 00  .....0... ...€..
00000120  A0 AE 0D 00 00 90 09 00 00 C0 0D 00 00 00 40 00   ®.......À....@.
00000130  00 10 00 00 00 02 00 00 05 00 00 00 00 00 00 00  ................
00000140  05 00 00 00 00 00 00 00 00 60 10 00 00 10 00 00  .........`......
00000150  00 00 00 00 02 00 00 80 00 00 40 00 00 10 00 00  .......€..@.....
00000160  00 00 40 00 00 10 00 00 00 00 00 00 10 00 00 00  ..@.............
00000170  00 00 00 00 00 00 00 00 48 52 10 00 B0 03 00 00  ........HR..°...
00000180  00 C0 0D 00 48 92 02 00 00 00 00 00 00 00 00 00  .À..H’..........
00000190  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000001A0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000001B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000001C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000001D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

This is the raw contents of the file in hexadecimal format, the numbers and letters are representative of the byte values of the files data. One noticeable thing when reviewing the data is that the first combination of numbers and letters is 4D 5A, and in the decoded text column the letters MZ. This can be identified as a file signature, specifically for a .exe file. Even though we have changed the original extension from .exe to .bin, it is still an executable.

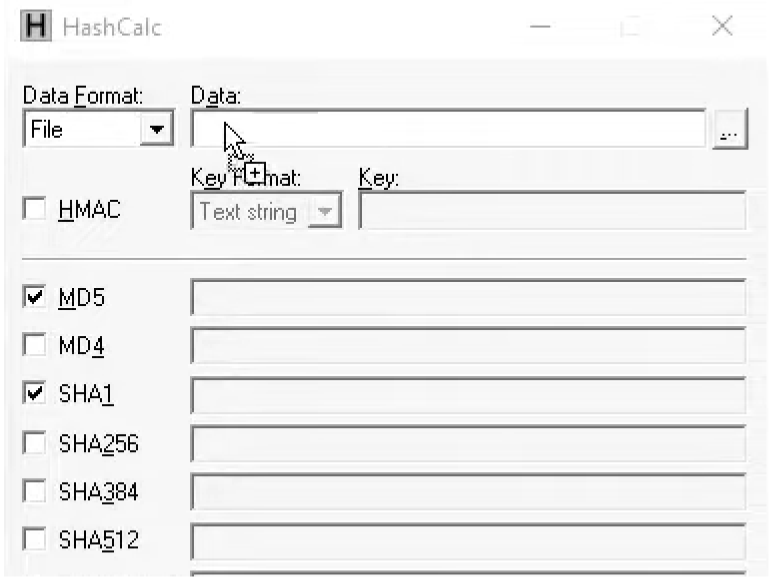
This shows how malware can hide in plain sight. Even if the file is changed to be an image, such as a .jpg, it will look like an image file but it will still show the executable file signature in the hexadecimal format.

Another indicator of a file being an executable is the message “This program cannot be run in DOS mode” in the decoded text column. This is because the file has an MZ header (common in EXE files) as part of DOS compatibility. It serves as a recognizable marker that the file is an executable and was designed to be run in a Windows environment rather than DOS.

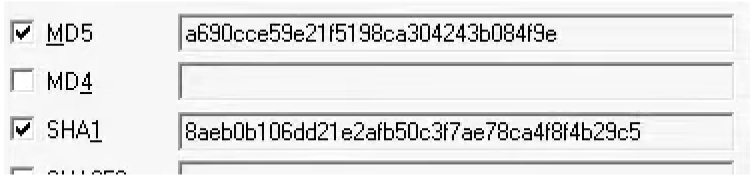
File Fingerprinting:

Fingerprinting in the context of malware is the process of uniquely identifying the malware based on distinct characteristics. A common method of file fingerprinting is hashing. Hashing involves creating a hash, which is a fixed-length value generated from the file's content. This means even a small change in the file, such as adding one character to a paragraph, will result in a completely different hash. This makes it quite easy to track specific malware samples. However this technique is mainly useful for static analysis, as if the file is modified, the hash changes.

The next step in the analysis was generating a hash for this file. This was done using Hashcalc, which allows a user to take a file and generate a hash for it using various hashing algorithms.



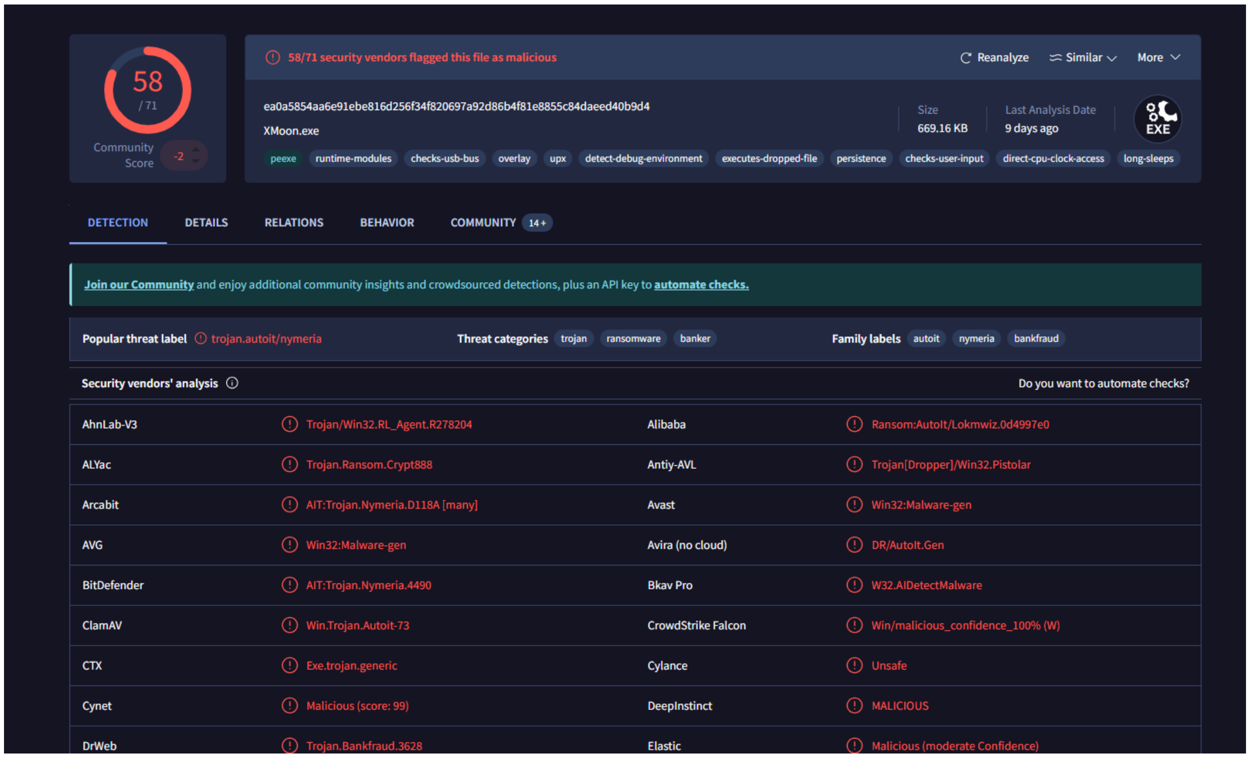
When the file is loaded into Hashcalc, I received the following output:



```
MD5   a690cce59e21f5198ca304243b084f9e
MD4
SHA1  8aeb0b106dd21e2afb50c3f7ae78ca4f8f4b29c5
```

Taking the generated hash, I then searched for it on VirusTotal to find out more about it.

VirusTotal is an online service that analyzes files, URLs, and other content for potential malware and viruses. It aggregates results from a variety of antivirus engines and tools to help users quickly determine whether a file or website is malicious.



58/71 security vendors flagged this file as malicious

ea0a5854aa6e91ebe816d256f34f820697a92d86b4f81e8855c84daeed40b9d4
XMoon.exe

Size 669.16 KB
Last Analysis Date 9 days ago

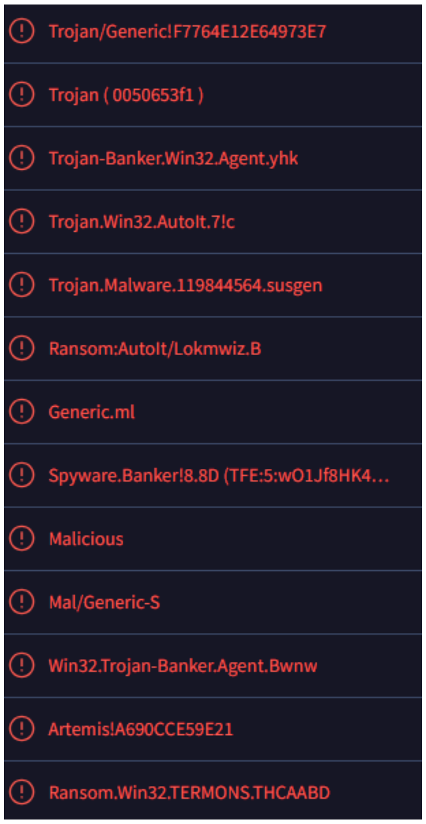
Join our Community and enjoy additional community insights and crowdsourced detections, plus an API key to automate checks.

Popular threat label trojan.autoit/nymeria
Threat categories trojan ransomware banker
Family labels autoit nymeria bankfraud

Security vendors' analysis

Do you want to automate checks?

AhnLab-V3	Trojan/Win32.RL_Agent.R278204	Alibaba	Ransom:AutoIt/Lokmwiz.0d4997e0
ALYac	Trojan.Ransom.Crypt888	Antiy-AVL	Trojan[Dropper]/Win32.Pistolar
Arcabit	AIT:Trojan.Nymeria.D118A [many]	Avast	Win32:Malware-gen
AVG	Win32:Malware-gen	Avira (no cloud)	DR/AutoIt.Gen
BitDefender	AIT:Trojan.Nymeria.4490	Bkav Pro	W32.AIDetectMalware
ClamAV	Win.Trojan.Autoit-73	CrowdStrike Falcon	Win/malicious_confidence_100% (W)
CTX	Exe.trojan.generic	Cylance	Unsafe
Cynet	Malicious (score: 99)	DeepInstinct	MALICIOUS
DrWeb	Trojan.Bankfraud.3628	Elastic	Malicious (moderate Confidence)

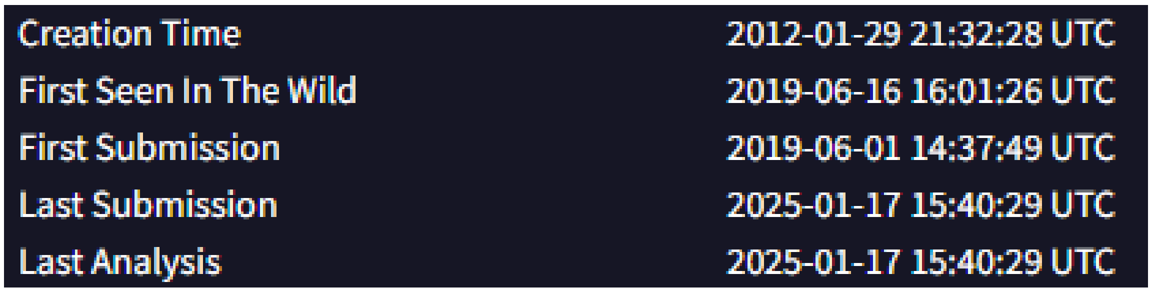


Trojan/Generic!F7764E12E64973E7
Trojan (0050653f1)
Trojan-Banker.Win32.Agent.yhk
Trojan.Win32.AutoIt.7!c
Trojan.Malware.119844564.susgen
Ransom:AutoIt/Lokmwiz.B
Generic.ml
Spyware.Banker!8.8D (TFE:5:wO1Jf8HK4...
Malicious
Mal/Generic-S
Win32.Trojan-Banker.Agent.Bwnw
Artemis!A690CCE59E21
Ransom.Win32.TERMONS.THCAABD

The malware has been identified as a trojan banker, which is a type of banking Trojan malware designed specifically to steal sensitive financial information, such as bank account details, login credentials, and credit card information.

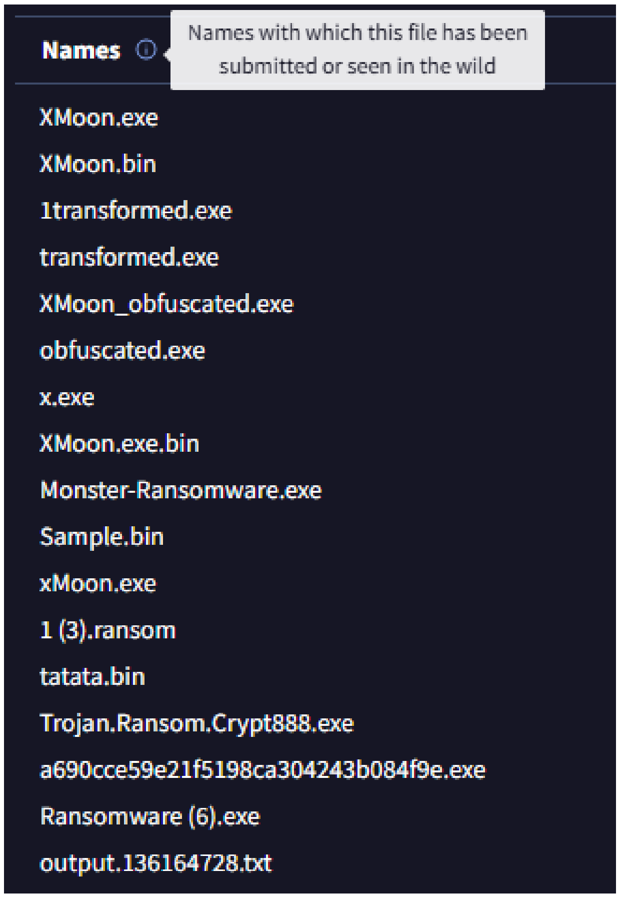
Banking Trojans are highly sophisticated pieces of software and are designed to operate covertly and remain undetected for long periods of time. This often means they don't exhibit obvious symptoms and are very difficult to get rid of.

Further examination provides information such as when it was first seen in the wild, other names it has been using and its behaviour. It was created on January 29th, 2012 and wasn't seen in the wild until 2019.



Creation Time	2012-01-29 21:32:28 UTC
First Seen In The Wild	2019-06-16 16:01:26 UTC
First Submission	2019-06-01 14:37:49 UTC
Last Submission	2025-01-17 15:40:29 UTC
Last Analysis	2025-01-17 15:40:29 UTC

The names it has been detected under can be seen below, it is evident that the file extension has been changed on some of them to .txt and .bin. This is to remain undetected by the user, as most often an executable file raises suspicions.

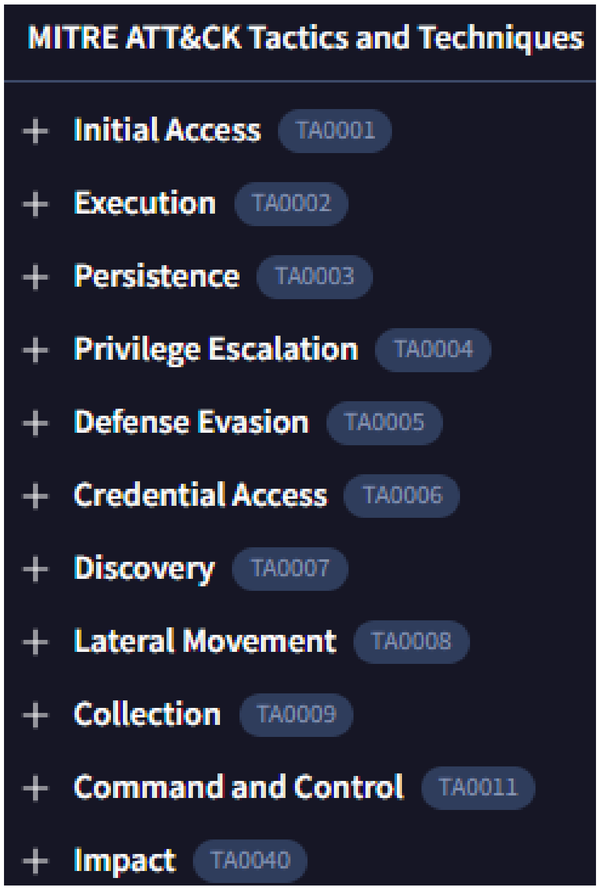


Names

Names with which this file has been submitted or seen in the wild

```
XMoon.exe
XMoon.bin
1transformed.exe
transformed.exe
XMoon_obfuscated.exe
obfuscated.exe
x.exe
XMoon.exe.bin
Monster-Ransomware.exe
Sample.bin
xMoon.exe
1 (3).ransom
tatata.bin
Trojan.Ransom.Crypt888.exe
a690cce59e21f5198ca304243b084f9e.exe
Ransomware (6).exe
output.136164728.txt
```

During the investigation of its behavior, the malware utilized the following MITRE ATT&CK tactics and techniques.



Initial access: Replication through removable media - *Checks for available system drives (often done to infect USB drives)*

Execution: Command & Scripting Interpreter - usage of command-line interfaces or scripting environments to execute commands, scripts, or binaries on a compromised system. - *Utilises Windows Update Standalone Installation*

Persistence: Boot/Logon Initilization Scripts, Registry run keys / Startup folder - Adding an entry to the "run keys" in the Registry or startup folder will cause the program referenced to be executed when a user logs in. These programs will be executed under the context of the user and will have the account's associated permissions level.

These are just some examples to show how advanced the malware is.

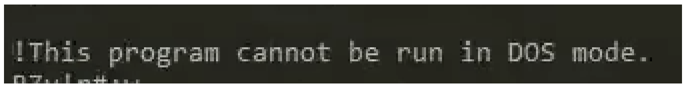
String Investigation

Using cmder I was able to inspect the contents of the file without executing it, ensuring that it remained isolated from my system. To analyze the file, I employed the Strings command. This searches binary files for readable text.

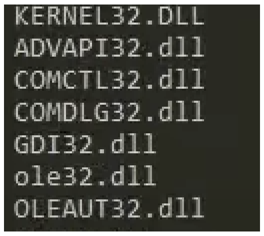
strings -n 8 XMoon.bin

This searched the file for anything containing 8 characters and higher. This approach allowed me to filter through the file more effectively, focusing on significant strings, which could be indicative of important data, like file paths, URLs, IP addresses, or even the names of functions and libraries used by the malware.

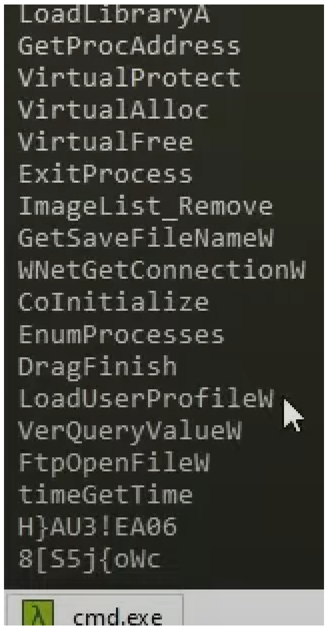
One string that I noticed in the output was an identifier of an executable file:


```
!This program cannot be run in DOS mode.
```

Other strings that were returned were some .DLL files and Windows API



```
KERNEL32.DLL
ADVAPI32.dll
COMCTL32.dll
COMDLG32.dll
GDI32.dll
ole32.dll
OLEAUT32.dll
```



```
LoadLibraryA
GetProcAddress
VirtualProtect
VirtualAlloc
VirtualFree
ExitProcess
ImageList_Remove
GetSaveFileNameW
WNetGetConnectionW
CoInitialize
EnumProcesses
DragFinish
LoadUserProfileW
VerQueryValueW
FtpOpenFileW
timeGetTime
H}AU3!EA06
8[S5j{oWc
```

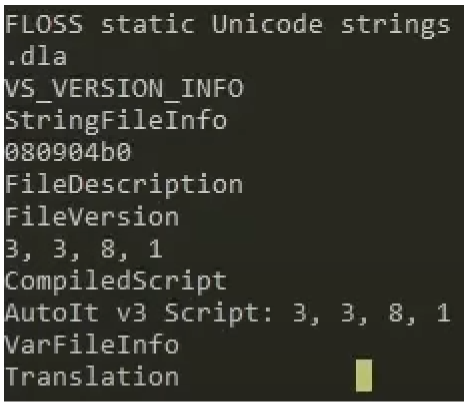
Encoded Strings:
Sometimes files can have encoded strings. Encoded strings are strings that represent data in a different from than it’s original. The encoding process converts the data (which could be text, images, or other types of information) into a set of printable characters or another acceptable format, making it easier to handle in different systems or applications. An example of a type of encoding would be Base64. Base64 works by converting binary data into a set of 64 different printable characters.

There are tools specifically to find encoded strings such as Xorsearch and Floss.

Xorsearch is a tool that examines the file's contents, looking for contents encoded using the XOR-based algorithm.

The FLARE Obfuscated String Solver (FLOSS, formerly FireEye Labs Obfuscated String Solver) uses advanced static analysis techniques to automatically extract and deobfuscate all strings from malware binaries.

Firstly I used “Xorsearch XMoon.exe http” to find and return any matches that have http. After some other attempts with keywords such as “Create” and “This”, it was evident they didn’t come back with anything beneficial. To keep searching I thought to use Floss to also search for encoded strings. Although Floss initially returned some results, it ultimately indicated that zero encoded strings were found in the file.



```
FLOSS static Unicode strings
.dla
VS_VERSION_INFO
StringFileInfo
080904b0
FileDescription
FileVersion
3, 3, 8, 1
CompiledScript
AutoIt v3 Script: 3, 3, 8, 1
VarFileInfo
Translation
```

Packing

Packing in malware refers to the technique used by attackers to compress or encrypt their malicious code to make it harder for security tools to detect, analyze, or reverse-engineer. The process involves wrapping the malware's code in a "packer" (a special program or algorithm) that modifies the file's structure, often by compressing or encrypting it, before delivering it to the victim’s system.

Once the packed malware is executed, the packer decompresses or decrypts the code in memory, allowing the malicious payload to run. Since the packed version of the malware looks different from the original (and typically contains little recognizable code or readable data), it can evade detection by antivirus programs or other security measures that rely on signatures or heuristics.

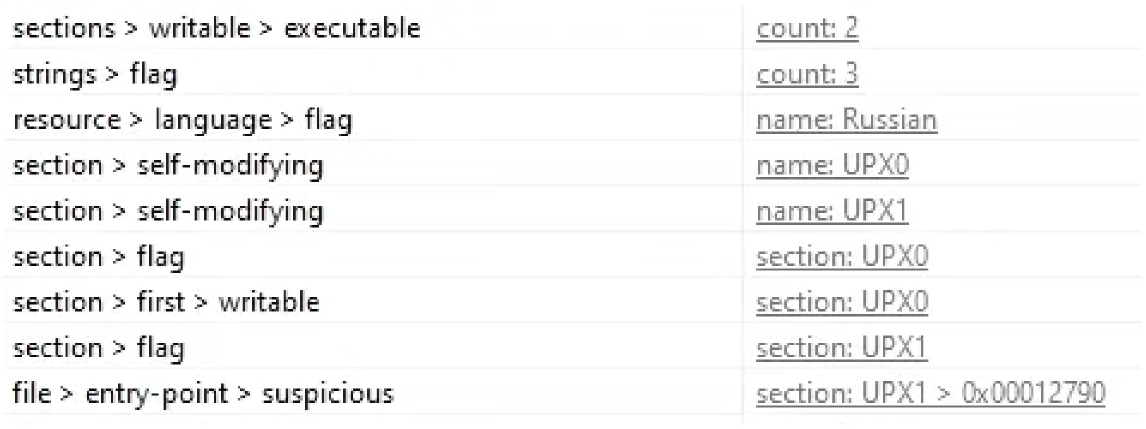
In order to see if malware is “packed” a tool called “Exeinfo” can be utilised.

When the XMoon file is loaded into Exeinfo it returns with what “packer” the malware is using. XMoon is using UPX. Beneath this information is instructions on how to decode it using the command “upx -d -o UnpackedMalware.exe XMoon.exe”.

This must be done in the directory where the malware is located. When successful an unpacked version of the malware is now created under the name “UnpackedMalware”. The noticeable difference between the unpacked and packed versions is that the unpacked version is 10 kb larger than the other. Lastly, to gain more information a tool called PEStudio can be used.

PEStudio

PEStudio is a software tool used for analyzing Windows executable files (EXE, DLLs, etc.) for potential security threats, malware analysis, or reverse engineering. One of the benefits of PEStudio is that it automatically submits the file to VirusTotal. When the unpacked file is loaded into PEStudio it returns excessive amounts of information such as threat indicators, strings and more. The information returned identified that the malware could possibly come from Russia.



sections > writable > executable	count: 2
strings > flag	count: 3
resource > language > flag	name: Russian
section > self-modifying	name: UPX0
section > self-modifying	name: UPX1
section > flag	section: UPX0
section > first > writable	section: UPX0
section > flag	section: UPX1
file > entry-point > suspicious	section: UPX1 > 0x00012790

Summary

Malware analysis is a critical process in cybersecurity, helping to understand the behavior and risks associated with malicious files. Through the use of various tools malicious files can be examined and broken down to reveal their functionality without ever executing them, this is especially important in preventing damage to the system or network. By examining elements such as file signatures, hashes, and embedded strings, analysts can uncover key information about the malware's purpose and methods. Static analysis, in particular, plays a vital role in identifying malicious code early, enabling quicker responses and improving overall cybersecurity defenses.