

## Skills Learned

- **Static Analysis** (Dissecting binaries without execution)
- **Dynamic Analysis** (Executing malware in a controlled sandbox)
- **Reverse Engineering** (Using tools like IDA Pro, Ghidra, and x64dbg)
- **Network Traffic Analysis** (Using Wireshark and INetSim)
- **Detection Rule Writing** (YARA rules for identifying malware)
- **Process Monitoring & Memory Analysis** (Procmon, Volatility)
- **Virtualization & Network Isolation** (Ensuring malware stays contained)

## Tools Used

- **Windows 10 (FlareVM)**: Fully equipped with malware analysis tools.
- **Remnux**: Linux distro for reverse engineering and network analysis.
- **VirtualBox**: Isolated virtual environment for executing malware safely.
- **INetSim**: Simulates internet services to monitor malware behavior.
- **Wireshark**: Capturing and analyzing network traffic.
- **ProcMon & Process Hacker**: Monitoring system behavior.

# Static Malware Analysis

**Static analysis** refers to the process of examining a file, typically malware or a software program, without actually running it.

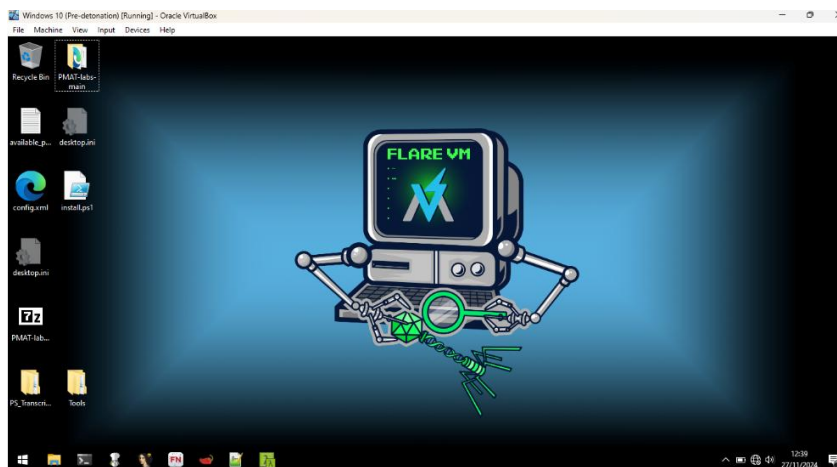
## Virtual Environment Setup

- Installed Windows 10 (FlareVM) in VirtualBox with a Host-Only Adapter for complete isolation.
- Installed Remnux in VirtualBox, also using a Host-Only Adapter.
- Ensured the lab is disconnected from the internet to prevent infections from

## 2 Configuring INetSim for Network Simulation

- Installed and configured INetSim on Remnux.

- Set FlareVM to route all traffic through Remnux, allowing malware to "believe" it has internet access.
- Captured DNS, HTTP, and other network interactions with Wireshark.



## Identify the file type

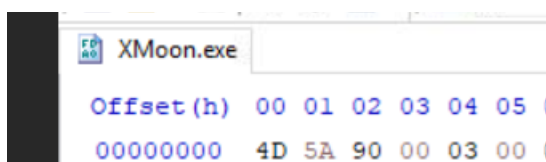
File Type : exe,png,dll

\*HxD

000DACC8	89 50 4E 47 0D 0A 1A 0A 00 00 00 0D 49 48 44 52	<div>^</div> <div>PNG IHDR</div> <div>\r</div> <div>f pHYs IDATx</div> <div>xTU E L</div> <div>B/ XP ew- X I</div> <div>fRfR C !</div> <div>*k] ]</div> <div>.</div> <div>{g { n</div> <div>9 i))</div>
000DACD8	00 00 01 00 00 00 01 00 08 06 00 00 00 5C 72 A8	
000DACE8	66 00 00 00 09 70 48 59 73 00 00 0B 13 00 00 0B	
000DACF8	13 01 00 9A 9C 18 00 00 20 00 49 44 41 54 78 9C	
000DAD08	ED 9D 07 78 54 55 DA C7 E3 F7 ED EE B7 45 CD 4C	
000DAD18	42 2F 82 82 58 50 BA AE 9D 65 77 2D A8 58 D0 49	
000DAD28	66 52 66 52 E6 DE 84 D0 43 17 21 F4 12 82 A2 08	
000DAD38	8A 8A 02 02 2A 6B 5D 15 5D 04 CB AA 8B 8A 0A 02	
000DAD48	2E 88 BD 81 8A 8A A0 80 20 E4 7C E7 BD C9 68 08	
000DAD58	B9 E7 DC 99 7B 67 EE 99 E4 FF 7B 9E FF C3 E3 6E	
000DAD68	E6 D6 F3 FE EF 39 EF 69 29 29 00 00 00 00 00 00	
000DAD78	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
000DAD88	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
000DAD98	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	

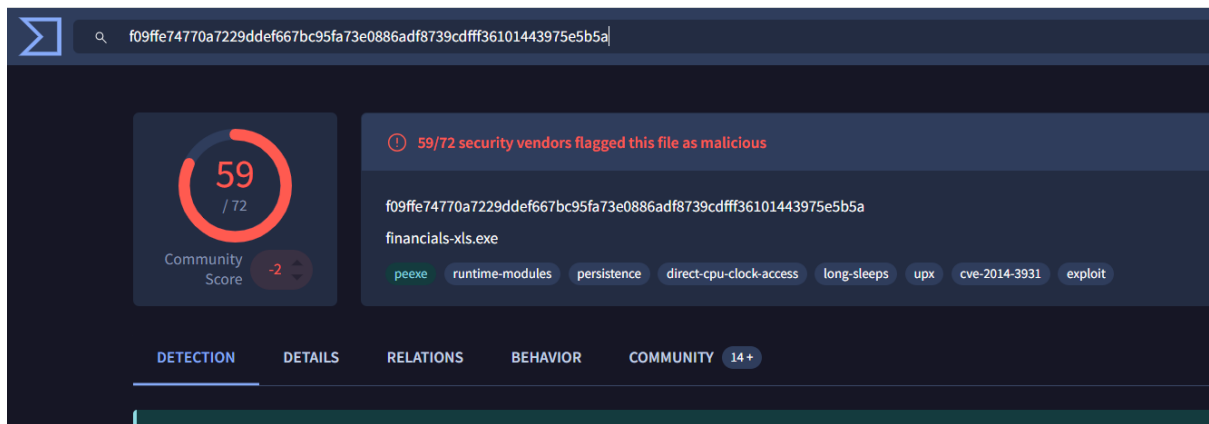
signatures for exes **malware XMoon.exe**

Keywords: MZ, 4D 5A, This program cannot be run in DOS mode

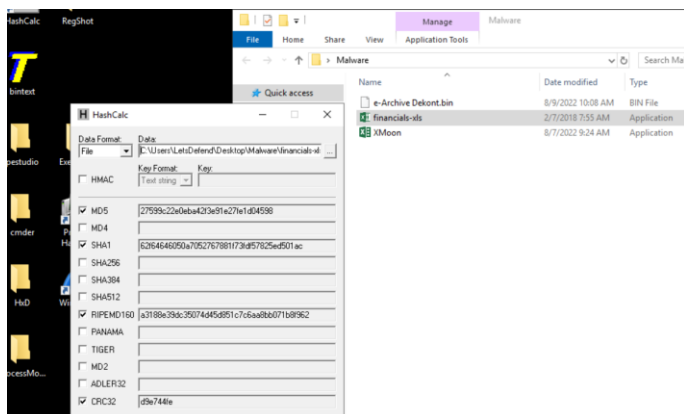


## fingerprinting the malware

\* Grab the hash and dump it in virustotal



\* Hashcal, Hashmyfiles



STRINGS

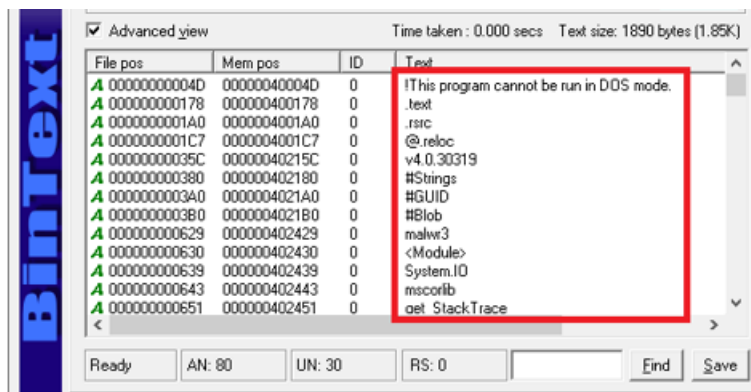
\* IP addresses

Base64 or any encoding techniques

File pos	Mem pos	ID	Text
00000000134C	00000040454C	0	</requestedPrivileges>
00000000136A	00000040456A	0	</security>
00000000137B	00000040457B	0	</trustInfo>
00000000138B	00000040458B	0	</assembly>
00000000093B	00000040273B	0	Connection
00000000095B	00000040275B	0	192.168.45.153
000000000979	000000402779	0	Connected
00000000098D	00000040278D	0	Enter the string to be transmitted :
0000000009D9	0000004027D9	0	Transmitting.....
0000000009FD	0000004027FD	0	Error.....
000000000E96	000000404096	0	VS_VERSION_INFO
000000000EF2	0000004040F2	0	VarFileInfo
000000000F12	000000404112	0	Translation

Command

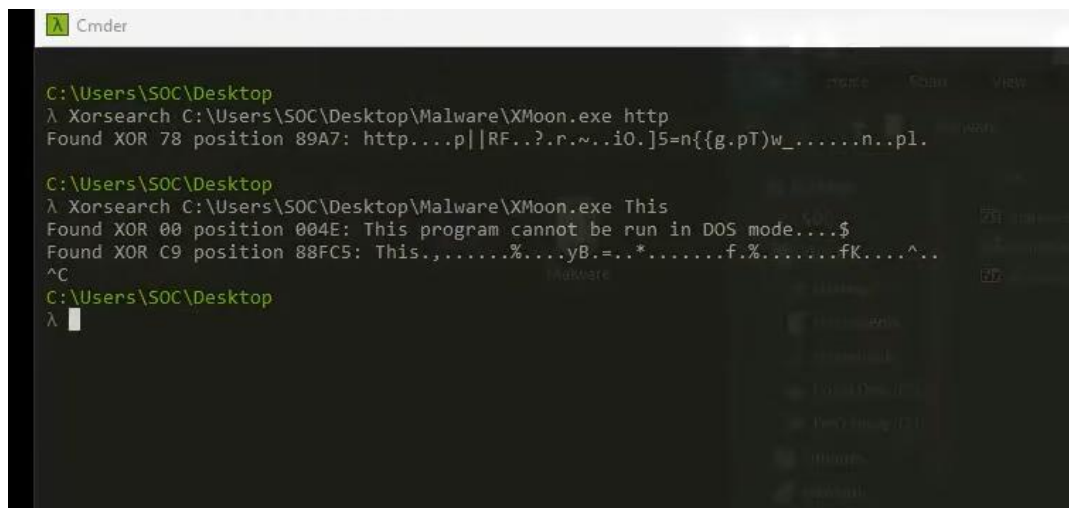
strings



## Decrypting Encoded strings

Xorsearch

Commands: **Xorsearch XMoon.exe http**



## Automatic Detection

Packing

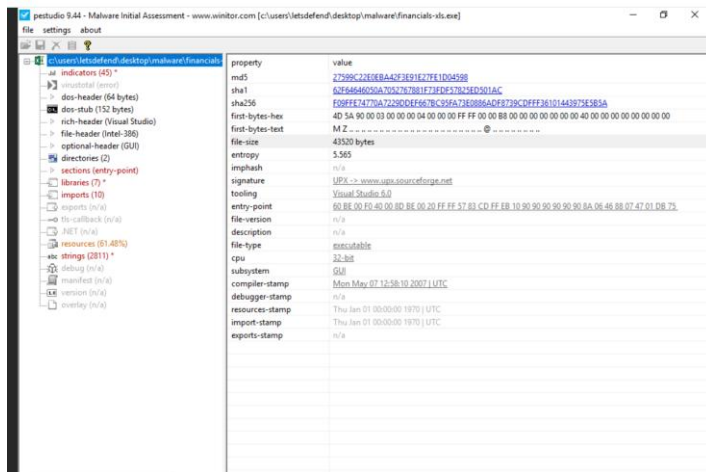
- **Packing:** The process of transforming an executable file into a different format using a special algorithm.
- **Unpacking:** Reverting the packed file to its original form by reversing the packing procedure.

(Using Packer Detection Tools)

These tools use **signatures** to identify common packers. Examples include:

✓ **DIE (Detect It Easy)** – Identifies packers based on file signatures.

- Eg: PEStudio**



Packed executables often have **unusual or extra sections**.

- ### Tools for PE Analysis:

PE-bear

PEView

## PE-bear

