

COS20030 Assignment 2

Core Task Group Report

Group Members:

Name	Student ID
Clement Chang Cheng	102774912
Phang Xia Hui	102773508

1.0 Basic Static Analysis Results

The initial phase of the investigation involved Basic Static Analysis to gather preliminary intelligence on the suspect binary MA25A2_malware_sample.exe. This process focused on identifying the file's properties, cryptographic hashes, and potential indicators of malicious intent without executing the code.

1.1 VirusTotal Intelligence

12/71 security vendors flagged this file as malicious

bfccb3f7821a87872c9874e635f836bc573681c75c31728743e1bcf2444fbca1
MA25A2_malware_sample.exe

Size: 305.00 KB | Last Analysis Date: 2 days ago | EXE

Community Score: 12 / 71

DETECTION DETAILS RELATIONS BEHAVIOR COMMUNITY

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

Popular threat label: downloader. Threat categories: downloader, trojan

Security vendors' analysis: CrowdStrike Falcon (Win/malicious_confidence_60%), DeepInstinct (MALICIOUS), Kaspersky (Not-a-virus:Downloader.Win32.Snojan.hlys), McAfee Scanner (T1IBFCCB3F7821A), Sangfor Engine Zero (Suspicious.Win32.Save.a), SentinelOne (Static ML) (Static AI - Suspicious PE), VBA32 (Suspected Of TrojanDownloader.gen), AhnLab-V3 (Undetected), AliCloud (Undetected), Antiy-AVL (Undetected), Arctic Wolf (Undetected).

Security vendor	Analysis	Result
CrowdStrike Falcon	Win/malicious_confidence_60%	MALICIOUS
Elastic	Malicious (high Confidence)	Not-a-virus:Downloader.Win32.Snojan.hlys
MaxSecure	Trojan.Malware.324995110.susgen	T1IBFCCB3F7821A
Palo Alto Networks	Generic.ml	Suspicious.Win32.Save.a
SecureAge	Malicious	Static AI - Suspicious PE
Trapmine	Malicious.moderate.ml.score	Suspected Of TrojanDownloader.gen
Acronis (Static ML)	Undetected	Undetected
Alibaba	Undetected	Undetected
AIYac	Undetected	Undetected
Arcabit	Undetected	Undetected

 Sign in Sign up

[Join our Community](#) and enjoy additional community insights and crowdsourced detections, plus an API key to [automate checks](#).

Basic properties 

MD5	97b5eeb16eb4ff520988d3896d4c58f9
SHA-1	7b10ff2e90544b2a6628a4e67b1619ad915330ac
SHA-256	bfccb3f7821a87872c9874e635f836bc573681c75c31728743e1bcf2444fbca1
Vhash	035056656561d055038z5arz4b
Authentihash	8486b029b3c823b199fcf2d35a7766acd44fd67d18757a3f3cdab4432bdbc783c
Imphash	64f33de39ed816bce3a095d52cf0e9d3
Rich PE header hash	a5b1e23e856c598129515287cd3bc4bd
SSDEEP	6144:uXBygxu6sN+Tjy7XL4vA6edGmPy80R/gSiYRN62nbPM7CTCJpEensZ:EBygxu6PGXEvA6edGjd9gvSiYRC72JP
TLSH	T15B64B041F6C291F1DAE61A3006E5DBTA9A3D71144B10ECDBE3941F7AEF102C19A3279D
File type	Win32 EXE executable windows win32 pe pexe
Magic	PE32 executable (console) Intel 80386, for MS Windows
TrID	Win64 Executable (generic) (40.3%) Win16 NE executable (generic) (19.3%) Win32 Executable (generic) (17.2%) OS/2 Executable (generic) (...)
DetectItEasy	PE32 Compiler: EP:Microsoft Visual C/C++ (2017 v.15.5-6) [EXE32] Compiler: Microsoft Visual C/C++ (19.36.35209) [LTCG/C] Linker: Microsoft...
MagikA	PEBIN
File size	305.00 KB (312320 bytes)

History 

Creation Time	2025-10-27 14:03:06 UTC
---------------	-------------------------



Σ bfcbb3f7821a87872c9874e635f836bc573681c75c31728743e1bcf2444fbca1

History

Creation Time	2025-10-27 14:03:06 UTC
First Submission	2025-10-29 10:29:57 UTC
Last Submission	2025-11-07 07:59:36 UTC
Last Analysis	2025-11-05 08:04:03 UTC

Names

MA25A2_malware_sample.exe
sddof.exe

Portable Executable Info

Compiler Products

[...] Unmarked objects count=123
id: 0x103, version: 33140 count=11
id: 0x105, version: 33140 count=169
id: 0x104, version: 33140 count=19
id: 0xfd, version: 35207 count=3
id: 0x103, version: 35207 count=22

Σ bfcbb3f7821a87872c9874e635f836bc573681c75c31728743e1bcf2444fbca1

Compiler Products

[...] Unmarked objects count=123
id: 0x103, version: 33140 count=11
id: 0x105, version: 33140 count=169
id: 0x104, version: 33140 count=19
id: 0xfd, version: 35207 count=3
id: 0x103, version: 35207 count=22
id: 0x104, version: 35207 count=17
id: 0x105, version: 35207 count=38
id: 0x101, version: 33140 count=9
id: 0x104, version: 35209 count=87

Header

Target Machine	Intel 386 or later processors and compatible processors
Compilation Timestamp	2025-10-27 14:03:06 UTC
Entry Point	141433
Contained Sections	5

Σ bfcbb3f7821a87872c9874e635f836bc573681c75c31728743e1bcf2444fbca1

Sections

Name	Virtual Address	Virtual Size	Raw Size	Entropy	MDS	Chi2
.text	4096	250880	6.65	750c7311c25b9c08f9928e9e58c0bc7		1299202.12
.rdata	258048	52722	52736	6.11	a8346afc6231198ad98cb0c2dc3dd293	1191533
.data	311296	9356	6656	4.04	40bc01d6e14a84751363599ff2f0240	563478.5
.ftable	323584	128	512	0	bfc619eac0cdf3f68d496ea9344137e8b	130560
.rsrc	327680	480	512	4.7	ff3791ff3353a1cb142916deda2fa7fc	9406

Imports

- + KERNEL32.dll
- + ADVAPI32.dll
- + WININET.dll
- + bcrypt.dll

Contained Resources By Type

RT_MANIFEST	1
-------------	---

Contained Resources By Language

ENGLISH US	1
------------	---

Σ bfcbb3f7821a87872c9874e635f836bc573681c75c31728743e1bcf2444fbca1

Imports

- + KERNEL32.dll
- + ADVAPI32.dll
- + WININET.dll
- + bcrypt.dll

Contained Resources By Type

RT_MANIFEST	1
-------------	---

Contained Resources By Language

ENGLISH US	1
------------	---

Contained Resources

SHA-256	File Type	Type	Language	Entropy	Chi2
4bb79dcea0a901f7d9eac5aa05728ae92acb42e0cb22e5dd14134f4421a3d8df	XML	RT_MANIFEST	ENGLISH US	4.91	4031.47

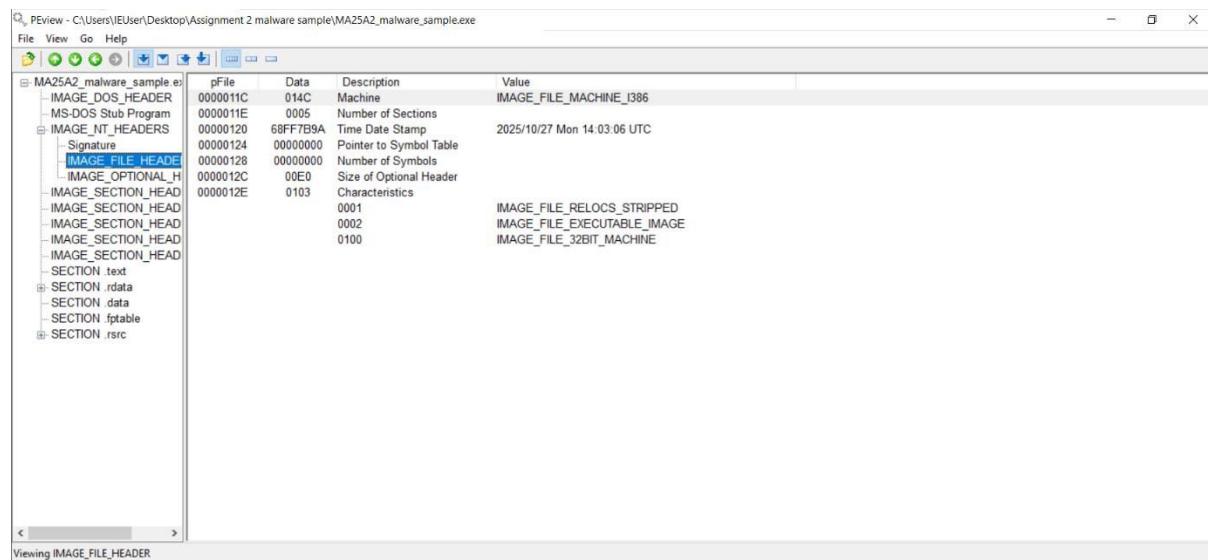
The sample was submitted to VirusTotal to check for known signatures and community reputation.

- **Detection Rate:** The file was flagged as malicious by **12 out of 71** security vendors.
- **Classification:** The threat is primarily categorized as a **Trojan and Downloader**. This suggests the malware's primary function is not necessarily to cause immediate damage itself, but to connect to a remote server and download additional malicious payloads.
- **File Hashes:**
 - **MD5:** 97b5eeb16eb4ff520988d3896d4c58f9
 - **SHA-256:**
bfccb3f7821a87872c9874e635f836bc573681c75c31728743e1bcf2444fbca1

Analysis: The detection ratio (12/71) is relatively low for a known malware sample. This, combined with the "Downloader" classification, often indicates that the file is a new variant or a "dropper" specifically designed to evade signature-based detection before fetching its main payload.

1.2 PEView and Timestamp Anomalies

We examined the Portable Executable (PE) header using PEView to inspect the file's metadata and compilation artifacts.



The screenshot shows the PEView interface with the file 'MA25A2_malware_sample.exe' open. The left pane displays the file structure tree, and the right pane shows detailed information for the 'IMAGE_FILE_HEADER'. The 'Time Date Stamp' field is highlighted with a blue border, showing the value '2025/10/27 Mon 14:03:06 UTC'. Other fields shown include Machine (014C), Number of Sections (0005), Pointer to Symbol Table (68F7B9A), Number of Symbols (00000000), Size of Optional Header (0E0), and Characteristics (0001, 0002, 0100).

pFile	Data	Description	Value
0000011C	014C	Machine	IMAGE_FILE_MACHINE_I386
0000011E	0005	Number of Sections	
00000120	68F7B9A	Time Date Stamp	2025/10/27 Mon 14:03:06 UTC
00000124	00000000	Pointer to Symbol Table	
00000128	00000000	Number of Symbols	
0000012C	00E0	Size of Optional Header	
0000012E	0103	Characteristics	
	0001		IMAGE_FILE_RELOCS_STRIPPED
	0002		IMAGE_FILE_EXECUTABLE_IMAGE
	0100		IMAGE_FILE_32BIT_MACHINE

Compilation Timestamp: 2025/10/27 Mon 14:03:06 UTC.

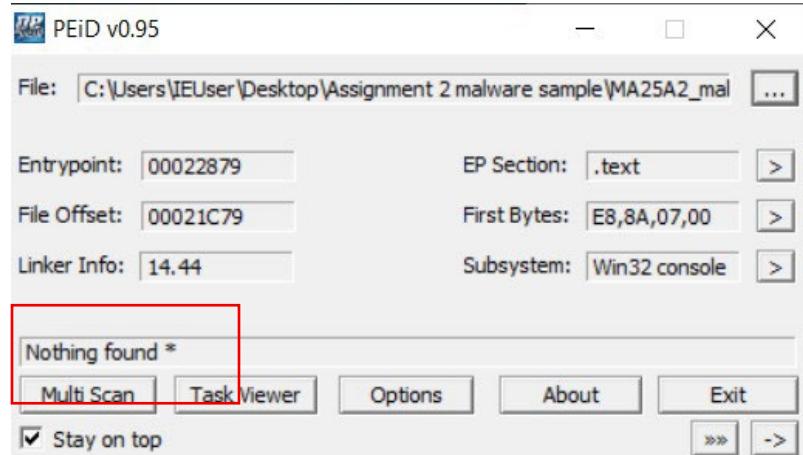
Observation: The analysis date (Nov 7, 2025) suggests this file was compiled recently relative to the current date, or potentially *in the future* depending on the exact analysis timeframe.

Analysis: Malware authors frequently manipulate the Time Date Stamp field in the PE header via "timestomping". By setting the compilation date to a future or seemingly legitimate past

date, attackers attempt to blend in with legitimate system files and evade forensic timelines or YARA rules that filter based on file age.

1.3 Packer Identification (PEiD)

To determine if the malware was obfuscated, we utilized PEiD.



PEiD Result: The tool returned "**Nothing found ***".

Analysis: The "Nothing found" result in PEiD is a strong indicator of a custom or unknown crypter. The malware code is compressed or encrypted to hide its true instructions from static analysis tools. This requires dynamic unpacking to reveal its true capabilities.

1.4 Dependency Analysis (Dependency Walker)

Dependency Walker was used to list the Dynamic Link Libraries (DLLs) and functions imported by the malware. The imports provided a high-level overview of the malware's capabilities, although many were likely hidden by the packer.

Module	Function	Hint	Ordinal	Pl
C:\Windows\SysWOW64\kernel32.dll	CreateDirectoryA	206 (0x00ce)	N/A	
C:\Windows\SysWOW64\kernel32.dll	MultiByteToWideChar	1055 (0x041e)	N/A	
C:\Windows\SysWOW64\kernel32.dll	createFileW	229 (0x00e5)	N/A	
C:\Windows\SysWOW64\kernel32.dll	DeleteFileW	308 (0x0134)	N/A	
C:\Windows\SysWOW64\kernel32.dll	FindFirstFileW	417 (0x01a1)	N/A	
C:\Windows\SysWOW64\kernel32.dll	GetFileAttributesW	616 (0x0268)	N/A	
C:\Windows\SysWOW64\kernel32.dll	GetFileAttributesExW	613 (0x0265)	N/A	
C:\Windows\SysWOW64\kernel32.dll	SetFileAttributesW	1362 (0x0552)	N/A	
C:\Windows\SysWOW64\kernel32.dll	MoveFileExW	1048 (0x0418)	N/A	
C:\Windows\SysWOW64\kernel32.dll	CreateProcessA	252 (0x00fc)	N/A	
C:\Windows\SysWOW64\kernel32.dll	GetLastError	646 (0x0286)	N/A	
C:\Windows\SysWOW64\kernel32.dll	GetTickCount	820 (0x0334)	N/A	
C:\Windows\SysWOW64\kernel32.dll	LocalFree	1022 (0x03fe)	N/A	
C:\Windows\SysWOW64\kernel32.dll	GetFileSizeEx	624 (0x0270)	N/A	
C:\Windows\SysWOW64\kernel32.dll	GetFileType	626 (0x0272)	N/A	
C:\Windows\SysWOW64\kernel32.dll	Getfiletime	625 (0x0271)	N/A	
C:\Windows\SysWOW64\kernel32.dll	ReadFile	1188 (0x04a4)	N/A	
C:\Windows\SysWOW64\kernel32.dll	SetFilePointerEx	1368 (0x0558)	N/A	
C:\Windows\SysWOW64\kernel32.dll	WriteConsoleW	1613 (0x064d)	N/A	
C:\Windows\SysWOW64\kernel32.dll	CloseHandle	156 (0x009c)	N/A	
C:\Windows\SysWOW64\kernel32.dll	GetTempPathA	800 (0x0320)	N/A	
C:\Windows\SysWOW64\kernel32.dll	WriteFile	1614 (0x064e)	N/A	
C:\Windows\SysWOW64\kernel32.dll	MoveFileA	1046 (0x0416)	N/A	
C:\Windows\SysWOW64\kernel32.dll	SetEndOfFile	1349 (0x0545)	N/A	
C:\Windows\SysWOW64\kernel32.dll	HeapSize	891 (0x037b)	N/A	
C:\Windows\SysWOW64\kernel32.dll	GetProcAddress	732 (0x02d1)	N/A	

Module	Function	Hint	Ordinal	Pl
C:\Windows\SysWOW64\kernel32.dll	HeapSize	891 (0x037b)	N/A	
C:\Windows\SysWOW64\kernel32.dll	GetProcessHeap	732 (0x02d1)	N/A	
C:\Windows\SysWOW64\kernel32.dll	GetStringTypeW	768 (0x0300)	N/A	
C:\Windows\SysWOW64\kernel32.dll	SetStdHandle	1411 (0x0583)	N/A	
C:\Windows\SysWOW64\kernel32.dll	SetEnvironmentVariableW	1353 (0x0549)	N/A	
C:\Windows\SysWOW64\kernel32.dll	FreeEnvironmentStringW	460 (0x01cc)	N/A	
C:\Windows\SysWOW64\kernel32.dll	GetEnvironmentStringsW	602 (0x025a)	N/A	
C:\Windows\SysWOW64\kernel32.dll	WideCharToMultiByte	1594 (0x063a)	N/A	
C:\Windows\SysWOW64\kernel32.dll	GetCPIInfo	483 (0x01e3)	N/A	
C:\Windows\SysWOW64\kernel32.dll	GetOEMCP	702 (0x02be)	N/A	
C:\Windows\SysWOW64\kernel32.dll	GetACP	468 (0x01d4)	N/A	
C:\Windows\SysWOW64\kernel32.dll	IsValidCodePage	954 (0x03ba)	N/A	
C:\Windows\SysWOW64\kernel32.dll	FindNextFileW	429 (0x01ad)	N/A	
C:\Windows\SysWOW64\kernel32.dll	FindFirstFileExW	412 (0x019c)	N/A	
C:\Windows\SysWOW64\kernel32.dll	FindClose	406 (0x0196)	N/A	
C:\Windows\SysWOW64\kernel32.dll	ReadConsoleW	1185 (0x04a1)	N/A	
C:\Windows\SysWOW64\kernel32.dll	UnhandledExceptionFilter	1513 (0x05e9)	N/A	
C:\Windows\SysWOW64\kernel32.dll	SetUnhandledExceptionFilter	1447 (0x05a7)	N/A	
C:\Windows\SysWOW64\kernel32.dll	GetCurrentProcess	570 (0x023a)	N/A	
C:\Windows\SysWOW64\kernel32.dll	TerminateProcess	1479 (0x05c7)	N/A	
C:\Windows\SysWOW64\kernel32.dll	IsProcessorFeaturePresent	948 (0x03b4)	N/A	
C:\Windows\SysWOW64\kernel32.dll	QueryPerformanceCounter	1149 (0x047d)	N/A	
C:\Windows\SysWOW64\kernel32.dll	GetCurrentProcessId	571 (0x023b)	N/A	
C:\Windows\SysWOW64\kernel32.dll	GetCurrentThreadId	575 (0x023f)	N/A	
C:\Windows\SysWOW64\kernel32.dll	GetSystemTimeAsFileTime	786 (0x0312)	N/A	
C:\Windows\SysWOW64\kernel32.dll	InitializeThreadLocalStorage	912 (0x0250)	N/A	

MA25A2_malware_sample

PI	Ordinal	Hint	Function	Module	De ^
		N/A	786 (0x0312) GetSystemTimeAsFileTime	C:\Windows\SysWOW64\kernel32.dll	Fa
		N/A	912 (0x0390) InitializeSListHead	C:\Windows\SysWOW64\kernel32.dll	Fa
		N/A	940 (0x03ac) IsDebuggerPresent	C:\Windows\SysWOW64\kernel32.dll	Fa
		N/A	761 (0x02f9) GetStartupInfoW	C:\Windows\SysWOW64\kernel32.dll	Fa
		N/A	670 (0x029e) GetModuleHandleW	C:\Windows\SysWOW64\kernel32.dll	Fa
		N/A	1288 (0x0508) RtlUnwind	C:\Windows\SysWOW64\kernel32.dll	Fa
		N/A	1384 (0x0568) SetLastError	C:\Windows\SysWOW64\kernel32.dll	Fa
		N/A	334 (0x0152) EnterCriticalSection	C:\Windows\SysWOW64\kernel32.dll	Fa
		N/A	1004 (0x03ec) LeaveCriticalSection	C:\Windows\SysWOW64\kernel32.dll	Fa
		N/A	301 (0x012d) DeleteCriticalSection	C:\Windows\SysWOW64\kernel32.dll	Fa
		N/A	908 (0x038c) InitializeCriticalSectionAndSpinC	C:\Windows\SysWOW64\kernel32.dll	Fa
		N/A	1497 (0x05d9) TlsAlloc	C:\Windows\SysWOW64\kernel32.dll	Fa
		N/A	1499 (0x05db) TlsGetValue	C:\Windows\SysWOW64\kernel32.dll	Fa
		N/A	1501 (0x05dd) TlsSetValue	C:\Windows\SysWOW64\kernel32.dll	Fa
		N/A	1498 (0x05da) TlsFree	C:\Windows\SysWOW64\kernel32.dll	Fa
		N/A	461 (0x01cd) FreeLibrary	C:\Windows\SysWOW64\kernel32.dll	Fa
		N/A	725 (0x02d5) GetProcAddress	C:\Windows\SysWOW64\kernel32.dll	Fa
		N/A	1010 (0x03f2) LoadLibraryExW	C:\Windows\SysWOW64\kernel32.dll	Fa
		N/A	334 (0x014e) EncodePointer	C:\Windows\SysWOW64\kernel32.dll	Fa
		N/A	1171 (0x0493) RaiseException	C:\Windows\SysWOW64\kernel32.dll	Fa
		N/A	383 (0x017e) ExitProcess	C:\Windows\SysWOW64\kernel32.dll	Fa
		N/A	669 (0x029d) GetModuleHandleExW	C:\Windows\SysWOW64\kernel32.dll	Fa
		N/A	763 (0x02f6) GetStdHandle	C:\Windows\SysWOW64\kernel32.dll	Fa
		N/A	666 (0x029a) GetModuleFileNameW	C:\Windows\SysWOW64\kernel32.dll	Fa
		N/A	504 (0x01f8) GetCommandLineA	C:\Windows\SysWOW64\kernel32.dll	Fa
		N/A	505 (0x01f9) GetCommandLineW	C:\Windows\SysWOW64\kernel32.dll	Fa
		N/A	882 (0x0372) HeapAlloc	C:\Windows\SysWOW64\kernel32.dll	Fa
		N/A	886 (0x0376) HeapFree	C:\Windows\SysWOW64\kernel32.dll	Fa
		N/A	443 (0x01bb) FlsAlloc	C:\Windows\SysWOW64\kernel32.dll	Fa
		N/A	445 (0x01bd) FlsGetValue	C:\Windows\SysWOW64\kernel32.dll	Fa
		N/A	447 (0x01bf) FlsSetValue	C:\Windows\SysWOW64\kernel32.dll	Fa
		N/A	444 (0x01bc) FlsFree	C:\Windows\SysWOW64\kernel32.dll	Fa
		N/A	909 (0x038d) InitializeCriticalSectionEx	C:\Windows\SysWOW64\kernel32.dll	Fa
		N/A	1544 (0x0608) VirtualProtect	C:\Windows\SysWOW64\kernel32.dll	Fa
		N/A	178 (0x00b2) CompareStringW	C:\Windows\SysWOW64\kernel32.dll	Fa
		N/A	992 (0x03e0) LCompareStringW	C:\Windows\SysWOW64\kernel32.dll	Fa
		N/A	546 (0x0222) GetConsoleOutputCP	C:\Windows\SysWOW64\kernel32.dll	Fa
		N/A	542 (0x021e) GetConsoleMode	C:\Windows\SysWOW64\kernel32.dll	Fa
		N/A	827 (0x033b) GetTimezoneInformation	C:\Windows\SysWOW64\kernel32.dll	Fa
		N/A	889 (0x0379) HeapReAlloc	C:\Windows\SysWOW64\kernel32.dll	Fa
		N/A	449 (0x01c1) FlushFileBuffers	C:\Windows\SysWOW64\kernel32.dll	Fa
		N/A	294 (0x0126) DecodePointer	C:\Windows\SysWOW64\kernel32.dll	Fa

Dependencies (WoW64)

File View Options Help

MA25A2_malware_sample

PI	Ordinal	Hint	Function	Module	De ^
		N/A	1010 (0x03f2) LoadLibraryExW	C:\Windows\SysWOW64\kernel32.dll	Fa
		N/A	334 (0x014e) EncodePointer	C:\Windows\SysWOW64\kernel32.dll	Fa
		N/A	1171 (0x0493) RaiseException	C:\Windows\SysWOW64\kernel32.dll	Fa
		N/A	383 (0x017e) ExitProcess	C:\Windows\SysWOW64\kernel32.dll	Fa
		N/A	669 (0x029d) GetModuleHandleExW	C:\Windows\SysWOW64\kernel32.dll	Fa
		N/A	763 (0x02f6) GetStdHandle	C:\Windows\SysWOW64\kernel32.dll	Fa
		N/A	666 (0x029a) GetModuleFileNameW	C:\Windows\SysWOW64\kernel32.dll	Fa
		N/A	504 (0x01f8) GetCommandLineA	C:\Windows\SysWOW64\kernel32.dll	Fa
		N/A	505 (0x01f9) GetCommandLineW	C:\Windows\SysWOW64\kernel32.dll	Fa
		N/A	882 (0x0372) HeapAlloc	C:\Windows\SysWOW64\kernel32.dll	Fa
		N/A	886 (0x0376) HeapFree	C:\Windows\SysWOW64\kernel32.dll	Fa
		N/A	443 (0x01bb) FlsAlloc	C:\Windows\SysWOW64\kernel32.dll	Fa
		N/A	445 (0x01bd) FlsGetValue	C:\Windows\SysWOW64\kernel32.dll	Fa
		N/A	447 (0x01bf) FlsSetValue	C:\Windows\SysWOW64\kernel32.dll	Fa
		N/A	444 (0x01bc) FlsFree	C:\Windows\SysWOW64\kernel32.dll	Fa
		N/A	909 (0x038d) InitializeCriticalSectionEx	C:\Windows\SysWOW64\kernel32.dll	Fa
		N/A	1544 (0x0608) VirtualProtect	C:\Windows\SysWOW64\kernel32.dll	Fa
		N/A	178 (0x00b2) CompareStringW	C:\Windows\SysWOW64\kernel32.dll	Fa
		N/A	992 (0x03e0) LCompareStringW	C:\Windows\SysWOW64\kernel32.dll	Fa
		N/A	546 (0x0222) GetConsoleOutputCP	C:\Windows\SysWOW64\kernel32.dll	Fa
		N/A	542 (0x021e) GetConsoleMode	C:\Windows\SysWOW64\kernel32.dll	Fa
		N/A	827 (0x033b) GetTimezoneInformation	C:\Windows\SysWOW64\kernel32.dll	Fa
		N/A	889 (0x0379) HeapReAlloc	C:\Windows\SysWOW64\kernel32.dll	Fa
		N/A	449 (0x01c1) FlushFileBuffers	C:\Windows\SysWOW64\kernel32.dll	Fa
		N/A	294 (0x0126) DecodePointer	C:\Windows\SysWOW64\kernel32.dll	Fa

Loading PE file "C:\Windows\SysWOW64\advapi32.dll" successful.

Library import: kernel32.dll

Functions import: CreateDirectoryA, MultiByteToWideChar, CreateFileW, DeleteFileW, FindFirstFileW, GetFileAttributesW, GetFileAttributesExW, SetFileAttributesW, MoveFileExW, CreateProcessA, GetLastError, GetTickCount, LocalFree, GetFileSizeEx, GetFileType, GetFileTime, ReadFile, SetFilePointerEx, WriteConsoleW, CloseHandle, GetTempPathA, WriteFile, MoveFileA, SetEndOfFile, HeapSize, GetProcessHeap, GetStringTypeW, SetStdHandle, SetEnvironmentVariableW, FreeEnvironmentStringsW, GetEnvironmentStringsW, WideCharToMultiByte, GetCPIInfo, GetOEMCP, GetACP, IsValidCodePage, FindNextFileW, FindFirstFileExW, FindClose, ReadConsoleW, UnhandledExceptionFilter, SetUnhandledExceptionFilter, GetCurrentProcess, TerminateProcess, IsProcessFeaturePresent, QueryPerformanceCounter, GetCurrentProcessId, GetCurrentThreadId, GetSystemTimeAsFileTime, InitializeSListHead, IsDebuggerPresent, GetStartupInfoW, GetModuleHandleW, RtlUnwind, SetLastError, EnterCriticalSection, LeaveCriticalSection, DeleteCriticalSection, InitializeCriticalSectionAndSpinC, TlsAlloc, TlsGetValue, TlsSetValue, TlsFree, FreeLibrary, GetProcAddress, LoadLibraryExW, EncodePointer, RaiseException, ExitProcess, GetModuleHandleExW, GetStdHandle,

GetModuleFileNameW, GetCommandLineA, GetCommandLineW, HeapAlloc, HeapFree, FlsAlloc, FlsGetValue, FlsSetValue, FlsFree, InitializeCriticalSectionEx, VirtualProtect, CompareStringW, LCMMapStringW, GetConsoleOutputCP, GetConsoleMode, GetTimeZoneInformation, HeapReAlloc, FlushFileBuffers, DecodePointer

PI	Ordinal	Hint	Function	Module	Delay
		N/A	671 (0x029E) RegSetValueExA	C:\Windows\SysWOW64\advapi32.dll	False
		N/A	642 (0x0282) RegOpenKeyExA	C:\Windows\SysWOW64\advapi32.dll	False
		N/A	594 (0x0252) RegCloseKey	C:\Windows\SysWOW64\advapi32.dll	False

Library imports: advapi32.dll

Functions import: RegSetValueExA, RegOpenKeyExA, RegCloseKey

PI	Ordinal	Hint	Function	Module	Delay
		N/A	199 (0x00c7) InternetOpenUrlA	C:\Windows\SysWOW64\WININET.dll	False
		N/A	149 (0x0095) InternetCloseHandle	C:\Windows\SysWOW64\WININET.dll	False
		N/A	206 (0x00ce) InternetReadFile	C:\Windows\SysWOW64\WININET.dll	False
		N/A	198 (0x00c6) InternetOpenA	C:\Windows\SysWOW64\WININET.dll	False

Library imports: WININET.dll

Functions import: InternetOpenUrlA, InternetCloseHandle, InternetReadFile, InternetOpenA

PI	Ordinal	Hint	Function	Module	Delay
		N/A	14 (0x000e) BCryptDestroyKey	C:\Windows\SysWOW64\bcrypt.dll	False
		N/A	18 (0x0012) BCryptEncrypt	C:\Windows\SysWOW64\bcrypt.dll	False
		N/A	31 (0x001f) BCryptGenerateSymmetricKey	C:\Windows\SysWOW64\bcrypt.dll	False
		N/A	35 (0x0023) BCryptHashData	C:\Windows\SysWOW64\bcrypt.dll	False
		N/A	53 (0x0035) BCryptSetProperty	C:\Windows\SysWOW64\bcrypt.dll	False
		N/A	33 (0x0021) BCryptGetProperty	C:\Windows\SysWOW64\bcrypt.dll	False
		N/A	39 (0x0027) BCryptOpenAlgorithmProvider	C:\Windows\SysWOW64\bcrypt.dll	False
		N/A	6 (0x0006) BCryptCreateHash	C:\Windows\SysWOW64\bcrypt.dll	False
		N/A	27 (0x001b) BCryptFinishHash	C:\Windows\SysWOW64\bcrypt.dll	False
		N/A	13 (0x000d) BCryptDestroyHash	C:\Windows\SysWOW64\bcrypt.dll	False
		N/A	12 (0x000c) BCryptDeriveKeyPBKDF2	C:\Windows\SysWOW64\bcrypt.dll	False
		N/A	29 (0x001d) BCryptGenRandom	C:\Windows\SysWOW64\bcrypt.dll	False
		N/A	2 (0x0002) BCryptCloseAlgorithmProvider	C:\Windows\SysWOW64\bcrypt.dll	False

Library imports: bcrypt.dll

Functions: BCryptDestroKey, BCryptEncrypt, BCryptGenerateSymmetricKey, BCryptHashData, BCryptSetProperty, BCryptGetProperty, BCryptOpenAlgorithmProvider, BCryptCreateHash, BCryptFinishHash, BCryptDestroyHash, BCryptDeriveKeyPBKDF2, BCryptGenRandom, BCryptCloseAlgorithmProvider

Key Imported DLLs:

- WININET.dll:** Imports InternetOpenUrlA and InternetReadFile. This confirms the malware has the capability to initiate network connections and download data from the internet, aligning with the "Downloader" classification.

- **ADVAPI32.dll:** Imports RegSetValueExA and RegOpenKeyExA. These functions are used to modify the Windows Registry, strongly suggesting the malware attempts to achieve **persistence** (surviving system reboots).
- **BCRYPT.dll:** Imports various encryption functions like BCryptDecrypt and BCryptEncrypt. This indicates the malware likely uses encryption, possibly to decrypt its downloaded payload or to secure its command-and-control (C2) communications.
- **KERNEL32.dll:** Imports standard file manipulation APIs (CreateFile, WriteFile), further supporting the hypothesis that it drops files onto the disk.

1.5 String Analysis (BinText)

We scanned the binary for readable strings to find hardcoded IP addresses, URLs, or messages.

File pos	Mem pos	ID	Text
0x00000049E56	0x00000448650	0	GetFileAttributesW
0x00000049E6C	0x0000044866C	0	GetFileAttributeExW
0x00000049E84	0x00000448684	0	SetFileAttributeW
0x00000049E9A	0x0000044869A	0	MoveFileExW
0x00000049EAB	0x000004486A8	0	VirtualAlloc
0x00000049E84	0x000004486B4	0	GetLastError
0x00000049EC4	0x000004486C4	0	GetTickCount
0x00000049E04	0x000004486D4	0	LocalFree
0x00000049EE0	0x000004486E0	0	GetFileSizeEx
0x00000049EF0	0x000004486F0	0	GetFileType
0x00000049F0C	0x000004486FC	0	GetFileTime
0x00000049F15	0x000004486F0	0	ReadFile
0x00000049F24	0x00000448724	0	KERNEL32.dll
0x00000049F3A	0x0000044873A	0	RegOpenKeyExA
0x00000049F4A	0x0000044874A	0	RegSetValueA
0x00000049F5C	0x0000044875C	0	RegCloseKey
0x00000049F68	0x00000448768	0	ADVAPI32.dll
0x00000049F78	0x00000448778	0	InternetOpenUrlA
0x00000049F8C	0x0000044878C	0	InternetOpenA
0x00000049F9C	0x0000044879C	0	InternetOpenHandle
0x00000049FB2	0x000004487B2	0	InternetReadFile
0x00000049FC4	0x000004487C4	0	WININET.dll
0x00000049FD2	0x000004487D2	0	BCryptOpenAlgorithmProvider
0x00000049FF0	0x000004487F0	0	BCryptGetProperty
0x0000004A004	0x00000448804	0	BCryptGetProperty
0x0000004A018	0x00000448818	0	BCryptCloseAlgorithmProvider
0x0000004A032	0x00000448832	0	BCryptGenerateSymmetricKey
0x0000004A056	0x00000448856	0	BCryptDecrypt
0x0000004A068	0x00000448868	0	BCryptDestroyKey
0x0000004A074	0x00000448874	0	RFNDFXReadEach

BinText revealed the all-caps fake DLL KERNEL32.dll at offset 0x4F52A which is a classic DLL search-order hijacking IOC that executes malicious code before the legitimate system kernel32.dll.

File pos	Mem pos	ID	Text
0x0000003E40	0x0000043E40	0	ObjectLength
0x0000003E45C	0x0000043E5C	0	HashDigestLength
0x0000003E528	0x0000043E528	0	Dapi-ms-win-core-fibers!1-1-1
0x00000043AAC	0x000004445AC	0	api-ms-win-core-synch!1-2-0
0x00000043E9	0x000004445E9	0	Kernel32
0x00000043AF0	0x000004445F0C	0	api-ms-
0x00000043B8	0x000004445F88	0	(null)
0x0000003F38	0x00000445738	0	mscoree.dll
0x000000442CD	0x00000445AC0	0	api-ms-core-datetime!1-1-1
0x00000044300	0x00000445B00	0	api-ms-win-core-fibers!1-1-2
0x0000004433C	0x00000445B3C	0	api-ms-win-core-file!1-1-0
0x00000044447	0x00000445C74	0	api-ms-win-core-file!1-2-2
0x0000004443C	0x00000445B8C	0	api-ms-win-core-file!2-1-4
0x0000004443E9	0x00000445E89	0	api-ms-win-core-localization!1-2-1
0x000000444430	0x00000445C30	0	api-ms-win-core-localization-absolute!1-2-0
0x000000444490	0x00000445C90	0	api-ms-win-core-processes!1-1-2
0x0000004444DC	0x00000445CDC	0	api-ms-win-core-string!1-1-0
0x00000044518	0x00000445D18	0	api-ms-win-core-synch!1-2-1
0x00000044539	0x00000445D95	0	api-ms-win-core-wint!1-1-0
0x00000044445	0x00000445E04	0	api-ms-win-core-timezone!1-1-0
0x0000004449D0	0x00000445D00	0	api-ms-win-core-windows!1-1-0
0x000000444620	0x00000445E20	0	api-ms-win-security-systemfunctions!1-1-0
0x000000444678	0x00000445E78	0	ext-ms-win-user32-dialogbox!1-1-0
0x0000004446C0	0x00000445E00	0	ext-ms-win-user32-windowsstation!1-1-0
0x000000444710	0x00000445F10	0	advapi32
0x000000444724	0x00000445F24	0	kernelbase
0x00000044473C	0x00000445F3C	0	ntdll
0x000000444740	0x00000445F40	0	api-ms-win-appmodel-runtime!1-1-2
0x000000444790	0x00000445F30	0	user32
0x0000004447A0	0x00000445FA0	0	svchost
0x0000004447C0	0x00000446200	0	System

These strings are automatically injected by the crypter when it replaces direct kernel32.dll imports with delayed Universal CRT redirects to hide real API calls until runtime.

Findings: The initial string scan revealed very little intelligible text. A search for keywords like "github" returned zero results in the packed file.

Analysis: The lack of readable strings is consistent with the file being packed. The packer has obfuscated the sensitive data.

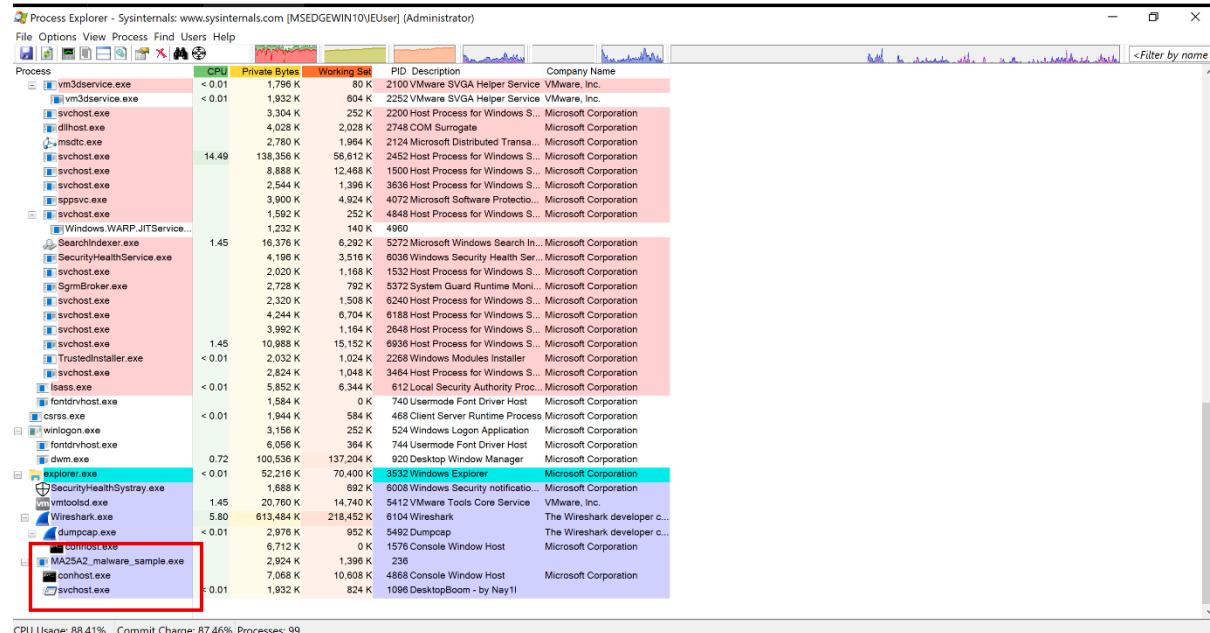
Fake DLL Indicator: A closer inspection revealed a string referencing a fake DLL path: C:\Windows\SysWOW64\kernel32.dll. This is a potential Indicator of Compromise (IOC) for **DLL Search Order Hijacking**, where the malware attempts to trick the operating system into loading a malicious DLL instead of the legitimate system library.

2.0 Basic Dynamic Analysis Results

During the basic dynamic analysis phase, the sample MA25A2_malware_sample.exe was executed within a controlled virtual environment. The analysis focused on observing the malware's interactions with the file system, registry, and network in real-time using Process Monitor, Process Explorer, and Wireshark. The specimen demonstrated behaviour consistent with evasion and reconnaissance.

2.1 Execution

Upon executing the sample, we first examined the process tree in **Process Explorer** to understand how the malware launched.



As observed in the screenshot above, MA25A2_malware_sample.exe spawned child processes, specifically `conhost.exe` and `svchost.exe`. Legitimate `svchost.exe` processes are typically children of `services.exe`. The fact that this malware launched `svchost.exe` as its own child process indicates process impersonation. This is a common evasion technique designed to hide malicious activity behind a benign-looking system process name.

2.2 Registry Alteration

Next, we analysed **Process Monitor** events to identify changes to the system configuration:

Time ...	Process Name	PID	Operation	Path	Result	Detail
11:26:5...	MA25A2_malw...	236	RegOpenKey	HKCU\Software\Microsoft\Windows\Cur...	SUCCESS	Desired Access: R...
11:26:5...	MA25A2_malw...	236	RegSetInfoKey	HKCU\Software\Microsoft\Windows\Cur...	SUCCESS	KeySetInformation...
11:26:5...	MA25A2_malw...	236	RegQueryKey	HKLML	SUCCESS	Query: HandleTag...
11:26:5...	MA25A2_malw...	236	RegQueryKey	HKLML	SUCCESS	Query: Name
11:26:5...	MA25A2_malw...	236	RegOpenKey	HKLM\Software\WOW6432Node\Policies\...	REPARSE	Desired Access: R...
11:26:5...	MA25A2_malw...	236	RegOpenKey	HKLM\Software\WOW6432Node\Policies\...	NAME NOT FOUND	Desired Access: R...
11:26:5...	MA25A2_malw...	236	RegQueryKey	HKCU	SUCCESS	Query: HandleTag...
11:26:5...	MA25A2_malw...	236	RegQueryKey	HKCU	SUCCESS	Query: Name
11:26:5...	MA25A2_malw...	236	RegOpenKey	HKCU\Software\Policies\Microsoft\Wind...	NAME NOT FOUND	Desired Access: R...
11:26:5...	MA25A2_malw...	236	RegQueryKey	HKLML	SUCCESS	Query: HandleTag...
11:26:5...	MA25A2_malw...	236	RegQueryKey	HKLML	SUCCESS	Query: Name
11:26:5...	MA25A2_malw...	236	RegOpenKey	HKLM\Software\WOW6432Node\Policies\...	REPARSE	Desired Access: R...
11:26:5...	MA25A2_malw...	236	RegOpenKey	HKLM\Software\WOW6432Node\Policies\...	NAME NOT FOUND	Desired Access: R...
11:26:5...	MA25A2_malw...	236	RegQueryKey	HKCU	SUCCESS	Query: HandleTag...
11:26:5...	MA25A2_malw...	236	RegQueryKey	HKCU	SUCCESS	Query: Name
11:26:5...	MA25A2_malw...	236	RegOpenKey	HKCU\Software\Microsoft\Windows\Cur...	SUCCESS	Desired Access: R...
11:26:5...	MA25A2_malw...	236	RegSetValue	HKCU\Software\Microsoft\Windows\Cur...	SUCCESS	KeySetInformation...
11:26:5...	MA25A2_malw...	236	RegQueryValue	HKCU\Software\Microsoft\Windows\Cur...	SUCCESS	Type: REG_DWO...
11:26:5...	MA25A2_malw...	236	RegQueryKey	HKCU	SUCCESS	Query: HandleTag...
11:26:5...	MA25A2_malw...	236	RegQueryKey	HKCU	SUCCESS	Query: Name
11:26:5...	MA25A2_malw...	236	RegOpenKey	HKCU\Software\Microsoft\Windows\Cur...	SUCCESS	Desired Access: R...
11:26:5...	MA25A2_malw...	236	RegSetValue	HKCU\Software\Microsoft\Windows\Cur...	SUCCESS	KeySetInformation...
11:26:5...	MA25A2_malw...	236	RegSetValue	HKCU\Software\Microsoft\Windows\Cur...	SUCCESS	Type: REG_DWO...
11:26:5...	MA25A2_malw...	236	RegSetValue	HKCU\Software\Microsoft\Windows\Cur...	SUCCESS	Type: REG_DWO...
11:26:5...	MA25A2_malw...	236	RegSetValue	HKCU\Software\Microsoft\Windows\Cur...	SUCCESS	Type: REG_DWO...
11:26:5...	MA25A2_malw...	236	RegSetValue	HKCU\Software\Microsoft\Windows\Cur...	SUCCESS	Type: REG_DWO...
11:26:5...	MA25A2_malw...	236	RegCloseKey	HKCU\Software\Microsoft\Windows\Cur...	SUCCESS	Type: REG_DWO...
11:26:5...	MA25A2_malw...	236	ReqCloseKey	HKCU\Software\Microsoft\Windows\Cur...	SUCCESS	

The capture above highlights multiple ***RegSetValue*** operations. The malware successfully wrote data to the registry keys under `HKCU\Software\Microsoft\Windows\CurrentVersion....`. Modifying these keys often indicates the malware is storing configuration settings or attempting to establish **persistence**, ensuring it remains active or re-launches after a system reboot.

2.3 File System Alteration

The malware demonstrated significant interaction with the file system. We tracked the sequence of file creation and data writing using **Process Monitor**.

Time o...	Process Name	PID	Operation	Path	Result	Detail
12:02:3...	MA25A2_malw...	4144	Process Start		SUCCESS	Parent PID: 3508, ...
12:02:3...	MA25A2_malw...	4144	Thread Create		SUCCESS	Thread ID: 4236
12:02:3...	MA25A2_malw...	4144	Load Image	C:\Users\IEUser\Desktop\DO NOT EXT...	SUCCESS	Image Base: 0x40...
12:02:3...	MA25A2_malw...	4144	Load Image	C:\Windows\System32\ntdll.dll	SUCCESS	Image Base: 0x7f...
12:02:3...	MA25A2_malw...	4144	Load Image	C:\Windows\SysWOW64\ntdll.dll	SUCCESS	Image Base: 0x77...
12:02:3...	MA25A2_malw...	4144	CreateFile	C:\Windows\Prefetch\MA25A2_MALWA...	NAME NOT FOUND	Desired Access: G...
12:02:3...	MA25A2_malw...	4144	RegOpenKey	HKEY\SYSTEM\CurrentControlSet\Contro...	REPARSE	Desired Access: Q...
12:02:3...	MA25A2_malw...	4144	RegOpenKey	HKEY\SYSTEM\CurrentControlSet\Contro...	SUCCESS	Desired Access: Q...
12:02:3...	MA25A2_malw...	4144	RegQueryValue	HKEY\SYSTEM\CurrentControlSet\Contro...	NAME NOT FOUND Length: 80	
12:02:3...	MA25A2_malw...	4144	RegCloseKey	HKEY\SYSTEM\CurrentControlSet\Contro...	SUCCESS	
12:02:3...	MA25A2_malw...	4144	RegOpenKey	HKEY\SYSTEM\CurrentControlSet\Contro..._REPARSE		Desired Access: Q...
12:02:3...	MA25A2_malw...	4144	RegOpenKey	HKEY\SYSTEM\CurrentControlSet\Contro..._NAME NOT FOUND		Desired Access: Q...
12:02:3...	MA25A2_malw...	4144	RegOpenKey	HKEY\SYSTEM\CurrentControlSet\Contro..._REPARSE		Desired Access: Q...
12:02:3...	MA25A2_malw...	4144	RegOpenKey	HKEY\SYSTEM\CurrentControlSet\Contro..._SUCCESS		Desired Access: Q...
12:02:3...	MA25A2_malw...	4144	RegQueryValue	HKEY\SYSTEM\CurrentControlSet\Contro..._NAME NOT FOUND Length: 24		
12:02:3...	MA25A2_malw...	4144	RegCloseKey	HKEY\SYSTEM\CurrentControlSet\Contro...	SUCCESS	
12:02:3...	MA25A2_malw...	4144	CreateFile	C:\Windows	SUCCESS	Desired Access: E...
12:02:3...	MA25A2_malw...	4144	Load Image	C:\Windows\System32\wow64.dll	SUCCESS	Image Base: 0x7f...
12:02:3...	MA25A2_malw...	4144	Load Image	C:\Windows\System32\wow64win.dll	SUCCESS	Image Base: 0x7f...
12:02:3...	MA25A2_malw...	4144	CreateFile	C:\Windows\System32\wow64log.dll	NAME NOT FOUND	Desired Access: R...
12:02:3...	MA25A2_malw...	4144	CreateFile	C:\Windows	SUCCESS	Desired Access: R...
12:02:3...	MA25A2_malw...	4144	QueryNameInfo...	C:\Windows	SUCCESS	Name: \Windows
12:02:3...	MA25A2_malw...	4144	CloseFile	C:\Windows	SUCCESS	
12:02:3...	MA25A2_malw...	4144	RegOpenKey	HKEY\Software\Microsoft\Wow64\i86	SUCCESS	Desired Access: R...
12:02:3...	MA25A2_malw...	4144	RegQueryValue	HKEY\SOFTWARE\Microsoft\Wow64\i86\...	NAME NOT FOUND Length: 520	
12:02:3...	MA25A2_malw...	4144	RegQueryValue	HKEY\SOFTWARE\Microsoft\Wow64\i86\...	SUCCESS	Type: REG_SZ, Le...
12:02:3...	MA25A2_malw...	4144	RegCloseKey	HKEY\SOFTWARE\Microsoft\Wow64\i86\...	SUCCESS	
12:02:3...	MA25A2_malw...	4144	Load Image	C:\Windows\System32\wow64cpu.dll	SUCCESS	Image Base: 0x77...
12:02:3...	MA25A2_malw...	4144	ReqOpenKey	HKEY\SYSTEM\CurrentControlSet\Contro..._REPARSE		Desired Access: Q...

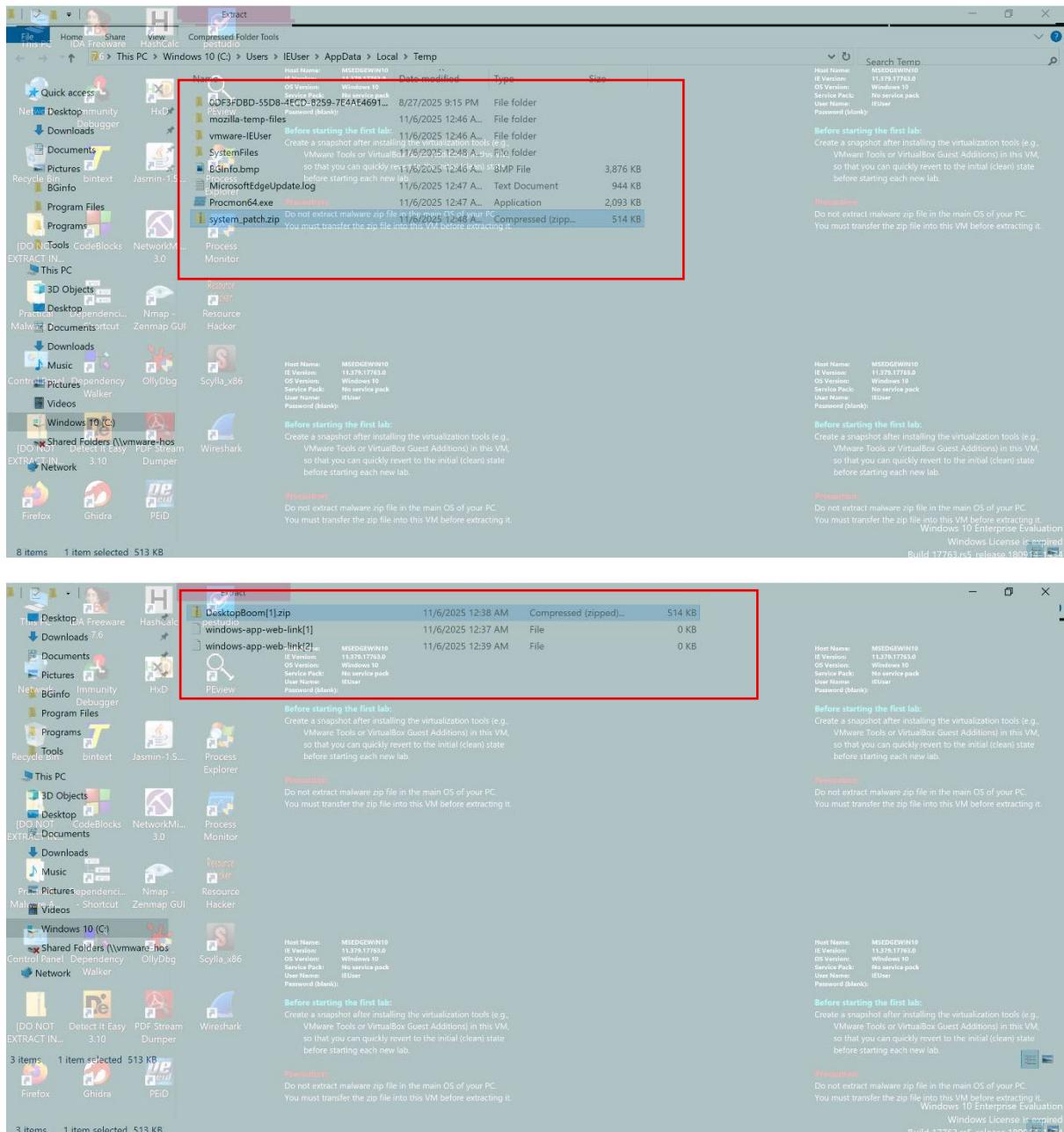
Initial **CreateFile** events as shown above revealed the malware obtaining handles to files in the **C:\Windows\Prefetch\MA25A2_MALWARE.....** directory. This was immediately followed by data ingestion:

Time o...	Process Name	PID	Operation	Path	Result	Detail
12:38:1...	MA25A2_malw...	4400	WriteFile	C:\Users\IEUser\AppData\Local\Microso...	SUCCESS	Offset: 116,736, L...
12:38:1...	MA25A2_malw...	4400	WriteFile	C:\Users\IEUser\AppData\Local\Microso...	SUCCESS	Offset: 117,760, L...
12:38:1...	MA25A2_malw...	4400	WriteFile	C:\Users\IEUser\AppData\Local\Microso...	SUCCESS	Offset: 118,784, L...
12:38:1...	MA25A2_malw...	4400	WriteFile	C:\Users\IEUser\AppData\Local\Temp\...	SUCCESS	Offset: 114,888, L...
12:38:1...	MA25A2_malw...	4400	WriteFile	C:\Users\IEUser\AppData\Local\Microso...	SUCCESS	Offset: 119,808, L...
12:38:1...	MA25A2_malw...	4400	WriteFile	C:\Users\IEUser\AppData\Local\Microso...	SUCCESS	Offset: 120,832, L...
12:38:1...	MA25A2_malw...	4400	WriteFile	C:\Users\IEUser\AppData\Local\Microso...	SUCCESS	Offset: 121,856, L...
12:38:1...	MA25A2_malw...	4400	WriteFile	C:\Users\IEUser\AppData\Local\Microso...	SUCCESS	Offset: 122,880, L...
12:38:1...	MA25A2_malw...	4400	WriteFile	C:\Users\IEUser\AppData\Local\Temp\...	SUCCESS	Offset: 118,784, L...
12:38:1...	MA25A2_malw...	4400	WriteFile	C:\Users\IEUser\AppData\Local\Microso...	SUCCESS	Offset: 123,904, L...
12:38:1...	MA25A2_malw...	4400	WriteFile	C:\Users\IEUser\AppData\Local\Microso...	SUCCESS	Offset: 124,928, L...
12:38:1...	MA25A2_malw...	4400	WriteFile	C:\Users\IEUser\AppData\Local\Microso...	SUCCESS	Offset: 125,952, L...
12:38:1...	MA25A2_malw...	4400	WriteFile	C:\Users\IEUser\AppData\Local\Microso...	SUCCESS	Offset: 126,976, L...
12:38:1...	MA25A2_malw...	4400	WriteFile	C:\Users\IEUser\AppData\Local\Temp\...	SUCCESS	Offset: 122,880, L...
12:38:1...	MA25A2_malw...	4400	WriteFile	C:\Users\IEUser\AppData\Local\Microso...	SUCCESS	Offset: 128,000, L...
12:38:1...	MA25A2_malw...	4400	WriteFile	C:\Users\IEUser\AppData\Local\Microso...	SUCCESS	Offset: 129,024, L...
12:38:1...	MA25A2_malw...	4400	WriteFile	C:\Users\IEUser\AppData\Local\Microso...	SUCCESS	Offset: 130,048, L...
12:38:1...	MA25A2_malw...	4400	WriteFile	C:\Users\IEUser\AppData\Local\Microso...	SUCCESS	Offset: 131,072, L...
12:38:1...	MA25A2_malw...	4400	WriteFile	C:\Users\IEUser\AppData\Local\Temp\...	SUCCESS	Offset: 126,976, L...
12:38:1...	MA25A2_malw...	4400	WriteFile	C:\Users\IEUser\AppData\Local\Microso...	SUCCESS	Offset: 132,096, L...
12:38:1...	MA25A2_malw...	4400	WriteFile	C:\Users\IEUser\AppData\Local\Microso...	SUCCESS	Offset: 133,120, L...
12:38:1...	MA25A2_malw...	4400	WriteFile	C:\Users\IEUser\AppData\Local\Microso...	SUCCESS	Offset: 134,144, L...
12:38:1...	MA25A2_malw...	4400	WriteFile	C:\Users\IEUser\AppData\Local\Microso...	SUCCESS	Offset: 135,168, L...
12:38:1...	MA25A2_malw...	4400	WriteFile	C:\Users\IEUser\AppData\Local\Temp\...	SUCCESS	Offset: 131,072, L...
12:38:1...	MA25A2_malw...	4400	WriteFile	C:\Users\IEUser\AppData\Local\Microso...	SUCCESS	Offset: 136,192, L...
12:38:1...	MA25A2_malw...	4400	WriteFile	C:\Users\IEUser\AppData\Local\Microso...	SUCCESS	Offset: 137,216, L...
12:38:1...	MA25A2_malw...	4400	WriteFile	C:\Users\IEUser\AppData\Local\Microso...	SUCCESS	Offset: 138,240, L...
12:38:1...	MA25A2_malw...	4400	WriteFile	C:\Users\IEUser\AppData\Local\Microso...	SUCCESS	Offset: 139,264, L...
12:38:1...	MA25A2_malw...	4400	WriteFile	C:\Users\IEUser\AppData\Local\Temp\...	SUCCESS	Offset: 135,168, L...

The **WriteFile** events confirmed that the malware was actively writing payloads to disk. Specifically, it targeted:

- C:\Users\IEUser\AppData\Local\Temp
- C:\Users\IEUser\AppData\Local\Microsoft\Windows\INetCache\IE\1BQLVC6W

To verify these findings, we navigated to the directories in the VM:



The screenshots above confirm the successful dropping of these artifacts such as the **DesktopBoom zip file** that would suggest desktop related activities due to its name. The use of the Temp and INetCache folders would suggest directories writable by standard users, allowing the malware to operate without needing administrative privileges. All the while the file explorer starts experiencing screen shaking and visibility issues.

2.4 Network Analysis and Reconnaissance

Finally, we investigated the network traffic generated by the malware.

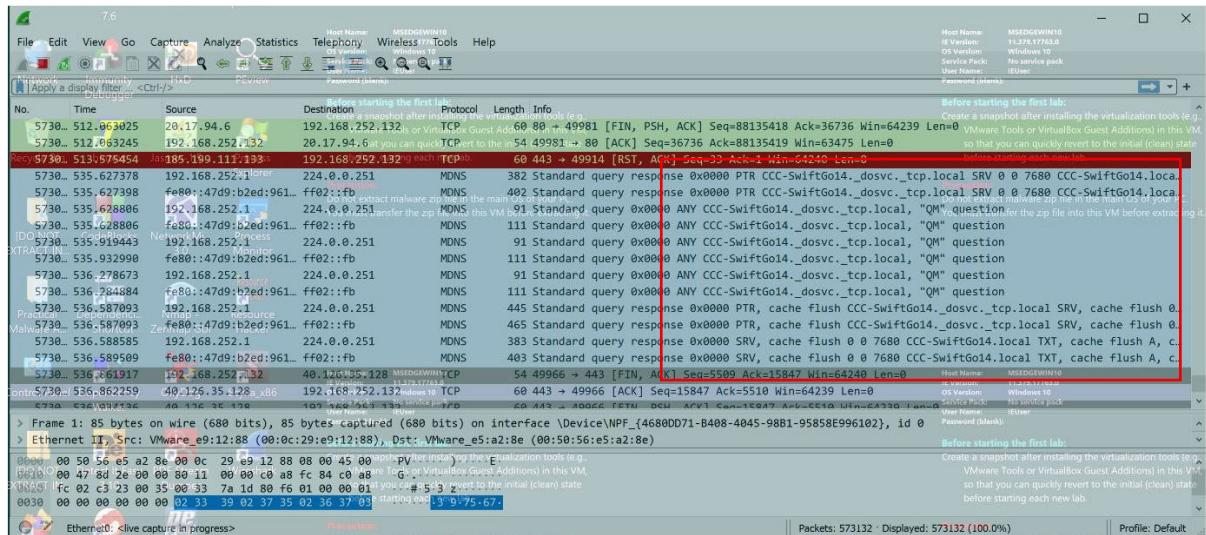
Time o...	Process Name	PID	Operation	Path	Result	Detail
12:02:3...	MA25A2_malw...	4144	TCP Connect	MSEDGEWIN10.localdomain:49763 -> ...	SUCCESS	Length: 0, mss: 14...
12:02:3...	MA25A2_malw...	4144	TCP Send	MSEDGEWIN10.localdomain:49763 -> ...	SUCCESS	Length: 171, starti...
12:02:3...	MA25A2_malw...	4144	TCP Receive	MSEDGEWIN10.localdomain:49763 -> ...	SUCCESS	Length: 1460, seq...
12:02:3...	MA25A2_malw...	4144	TCP Receive	MSEDGEWIN10.localdomain:49763 -> ...	SUCCESS	Length: 1460, seq...
12:02:3...	MA25A2_malw...	4144	TCP Connect	MSEDGEWIN10.localdomain:49764 -> ...	SUCCESS	Length: 412, seq...
12:02:3...	MA25A2_malw...	4144	TCP Send	MSEDGEWIN10.localdomain:49764 -> ...	SUCCESS	Length: 0, mss: 14...
12:02:3...	MA25A2_malw...	4144	TCP Receive	MSEDGEWIN10.localdomain:49764 -> ...	SUCCESS	Length: 232, starti...
12:02:3...	MA25A2_malw...	4144	TCP Connect	MSEDGEWIN10.localdomain:49765 -> ...	SUCCESS	Length: 2260, seq...
12:02:3...	MA25A2_malw...	4144	TCP Send	MSEDGEWIN10.localdomain:49765 -> ...	SUCCESS	Length: 0, mss: 14...
12:02:3...	MA25A2_malw...	4144	TCP Receive	MSEDGEWIN10.localdomain:49765 -> ...	SUCCESS	Length: 231, starti...
12:02:3...	MA25A2_malw...	4144	TCP Receive	MSEDGEWIN10.localdomain:49765 -> ...	SUCCESS	Length: 1798, seq...
12:02:3...	MA25A2_malw...	4144	TCP Connect	MSEDGEWIN10.localdomain:49766 -> ...	SUCCESS	Length: 0, mss: 14...
12:02:3...	MA25A2_malw...	4144	TCP Send	MSEDGEWIN10.localdomain:49766 -> ...	SUCCESS	Length: 229, starti...
12:02:3...	MA25A2_malw...	4144	TCP Receive	MSEDGEWIN10.localdomain:49766 -> ...	SUCCESS	Length: 1079, seq...
12:02:3...	MA25A2_malw...	4144	TCP Send	MSEDGEWIN10.localdomain:49763 -> ...	SUCCESS	Length: 93, startin...
12:02:3...	MA25A2_malw...	4144	TCP Receive	MSEDGEWIN10.localdomain:49763 -> ...	SUCCESS	Length: 51, sequin...
12:02:3...	MA25A2_malw...	4144	TCP Send	MSEDGEWIN10.localdomain:49763 -> ...	SUCCESS	Length: 190, starti...
12:02:3...	MA25A2_malw...	4144	TCP Receive	MSEDGEWIN10.localdomain:49763 -> ...	SUCCESS	Length: 1460, seq...
12:02:3...	MA25A2_malw...	4144	TCP Receive	MSEDGEWIN10.localdomain:49763 -> ...	SUCCESS	Length: 1460, seq...
12:02:3...	MA25A2_malw...	4144	TCP Connect	MSEDGEWIN10.localdomain:49767 -> ...	SUCCESS	Length: 0, mss: 14...
12:02:3...	MA25A2_malw...	4144	TCP Send	MSEDGEWIN10.localdomain:49767 -> ...	SUCCESS	Length: 186, starti...
12:02:3...	MA25A2_malw...	4144	TCP Receive	MSEDGEWIN10.localdomain:49767 -> ...	SUCCESS	Length: 1460, seq...
12:02:3...	MA25A2_malw...	4144	TCP Receive	MSEDGEWIN10.localdomain:49767 -> ...	SUCCESS	Length: 1460, seq...
12:02:3...	MA25A2_malw...	4144	TCP Connect	MSEDGEWIN10.localdomain:49767 -> ...	SUCCESS	Length: 1460, seq...
12:02:3...	MA25A2_malw...	4144	TCP Send	MSEDGEWIN10.localdomain:49767 -> ...	SUCCESS	Length: 1161, seq...
12:02:3...	MA25A2_malw...	4144	TCP Receive	MSEDGEWIN10.localdomain:49764 -> ...	SUCCESS	Length: 232, starti...
12:02:3...	MA25A2_malw...	4144	TCP Receive	MSEDGEWIN10.localdomain:49764 -> ...	SUCCESS	Length: 1798, seq...
12:02:3...	MA25A2_malw...	4144	TCP Receive	MSEDGEWIN10.localdomain:49764 -> ...	SUCCESS	Length: 470, seqn...

Process Monitor as shown above first alerted us to TCP activity, showing the process actively connecting to and receiving data from external endpoints. To get a clearer view of the destinations, we checked the TCP/IP tab in **Process Explorer**:

Protocol	Local Address	Remote Address	State
TCP	msedgewin10.localdo...	20.205.243.166https	CLOSE_WAIT
TCP	msedgewin10.localdo...	104.18.38.233:htp	ESTABLISHED
TCP	msedgewin10.localdo...	172.64.149.23:htp	ESTABLISHED
TCP	msedgewin10.localdo...	104.18.38.233:htp	ESTABLISHED
TCP	msedgewin10.localdo...	cdn-185-199-110-1...	ESTABLISHED

The properties window revealed established connections to external IP addresses, such as 104.18.38.233 and 20.205.243.166 which immediately disappears after. This suggests command-and-control (C2) communication or payload downloading.

To analyse the content of the traffic, we turned to **Wireshark**:



The packet capture uncovered unusual Multicast DNS or mDNS activity. The malware generated **multiple ANY requests** querying for **_dosv._tcp.local**. This is a potential indicator of network discovery. The malware is likely scanning the local subnet to identify other vulnerable hosts or specific services to propagate to, similarly to worms.

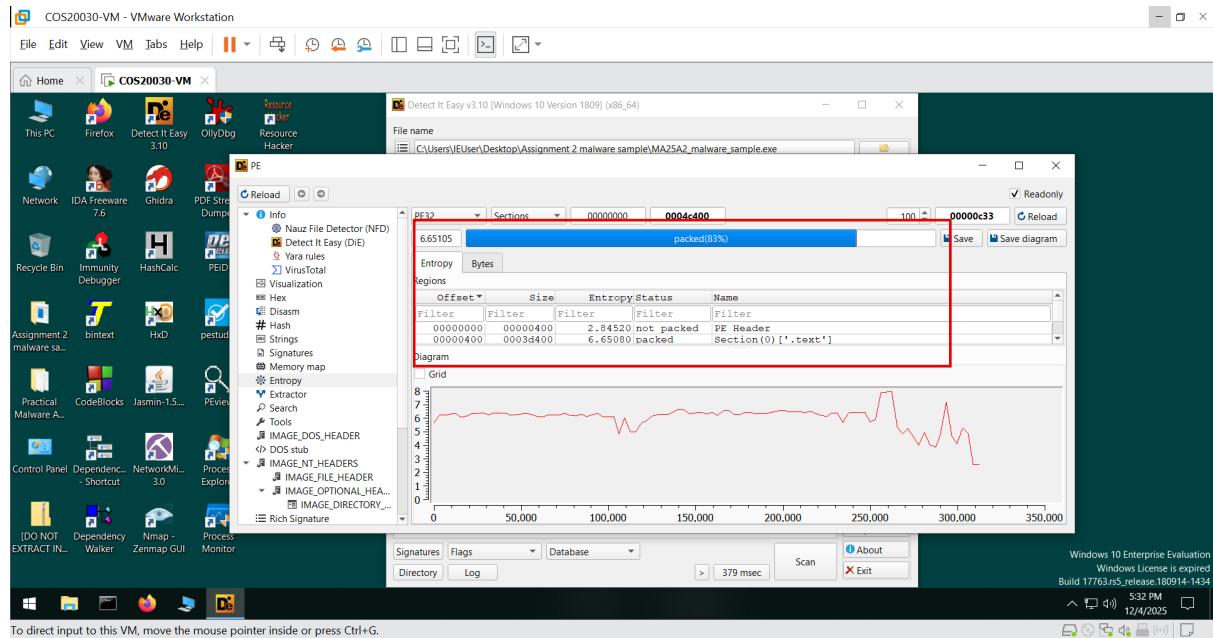
This Basic dynamic analysis confirms that **MA25A2_malware_sample.exe** is a sophisticated threat. It utilizes process impersonation to blend in, **drops persistence artifacts** in user-writable directories, and **performs active network reconnaissance** to potentially spreads itself across the network.

3.0 Advanced Static Analysis Results

Following the basic analysis, the team performed advanced static analysis to dissect the malware's internal code structure and capabilities without execution. This phase utilized **Detect It Easy (DIE)** for packer identification, **IDA Pro** for disassembly, and **Capa** for automated capability detection.

3.1 Packer Identification and Entropy Analysis

The malware sample was first analysed using **Detect It Easy (DIE)** to examine its entropy levels and identify potential packing or obfuscation.



Observation: The analysis revealed a high entropy value of **6.65** for the .text section whilst indicating the packed percentage of **83%**. The entropy graph consistently hovered in the 6-7 range, which is characteristic of compressed or encrypted data.

Analysis: A high entropy value is a strong indicator that the malware author has employed packing techniques to compress or encrypt the executable's code. This is a primary evasion tactic designed to conceal the malware's true functionality from antivirus scanners and hinder reverse engineering efforts. This finding necessitated the dynamic unpacking process detailed in the "Advanced Dynamic Analysis" section.

3.2 Disassembly and Code Analysis (IDA Pro)

The sample was loaded into **IDA Pro** to inspect its assembly code and internal logic. This deep dive revealed critical insights into the malware's entry point, obfuscation methods, and persistence mechanisms.

3.2.1 Entry Point Identification

Upon loading the binary, the program's Entry Point (EP) was identified at address 0x00422879. This is the memory address where the operating system transfers control to start the malware's execution.

IDA - MA25A2_malware_sample.exe C:\Users\IEUser\Desktop\Assignment 2 malware sample\MA25A2_malware_sample.exe

File Edit View VM Tabs Help

Library function Regular function Instruction Data Unexplored External symbol Lumina function

Functions Hex View-1 Structures Enums Imports Exports

Function name

sub_401000 ; .text:00422875 ; ====== S U B R O U T I N E ======

sub_401006 ; .text:00422879 ;

sub_401029 ; .text:00422879 ;

sub_401048 ; .text:00422879 ; public start

sub_401076 ; .text:00422879 ; start proc near

sub_401078 ; .text:00422879

sub_401092 ; .text:00422879 ; FUNCTION CHUNK AT .text:004226f4 SIZE 0000012A BYTES

sub_401169 ; .text:00422879 ; FUNCTION CHUNK AT .text:00422853 SIZE 00000026 BYTES

sub_401358 ; .text:00422879 call sub_422808

sub_401365 ; .text:00422879 jmp loc_4228f4

sub_401458 ; .text:0042287E ; Start [Synchronized with Hex View-i]

sub_401534 ; .text:0042287E

sub_401630 ; .text:00422881 align 10h

sub_401690 ; .text:00422880 ; [0000002B BYTES: COLLAPSED FUNCTION _alloca_probe. PRESS CTRL+NUMPAD+0 TO EXPAND]

sub_401E00 ; .text:004228C8 align 10h

sub_401F40 ; .text:004228D0 ; START OF FUNCTION CHUNK FOR sub_422980

sub_402130 ; .text:004228D0

sub_403160 ; .text:004228D0 loc_4228D0: ; CODE XREF: sub_422980+2A+j

00021CT9 0000000000422879: start (Synchronized with Hex View-i)

Output

Propagating type information...

Function argument information has been propagated

The initial autoanalysis has been finished.

ADC

AQ: idle Down Disk: 12GB

555 PM
12/4/2025

To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

Observation: The disassembly window clearly displays the start function at 0x00422879. The code begins with a call to a subroutine sub_423008 followed by a jump instruction, which is typical of standard Windows executable initialization.

3.2.2 Obfuscation Indicators (Stack Analysis Error)

Immediately following the entry point instructions, IDA Pro flagged a critical analysis warning.

The screenshot shows the IDA Pro interface for the file `MA25A2_malware_sample.exe`. The assembly view displays the following code snippet:

```
.text:00422879  
.text:00422879  
.text:00422879  
[...]  
.text:00422879 public start  
.text:00422879 start proc near  
.text:00422879  
.text:00422879 ; FUNCTION CHUNK AT .text:004226F4 SIZE 00000012A BYTES  
.text:00422879 ; FUNCTION CHUNK AT .text:00422853 SIZE 000000026 BYTES  

```

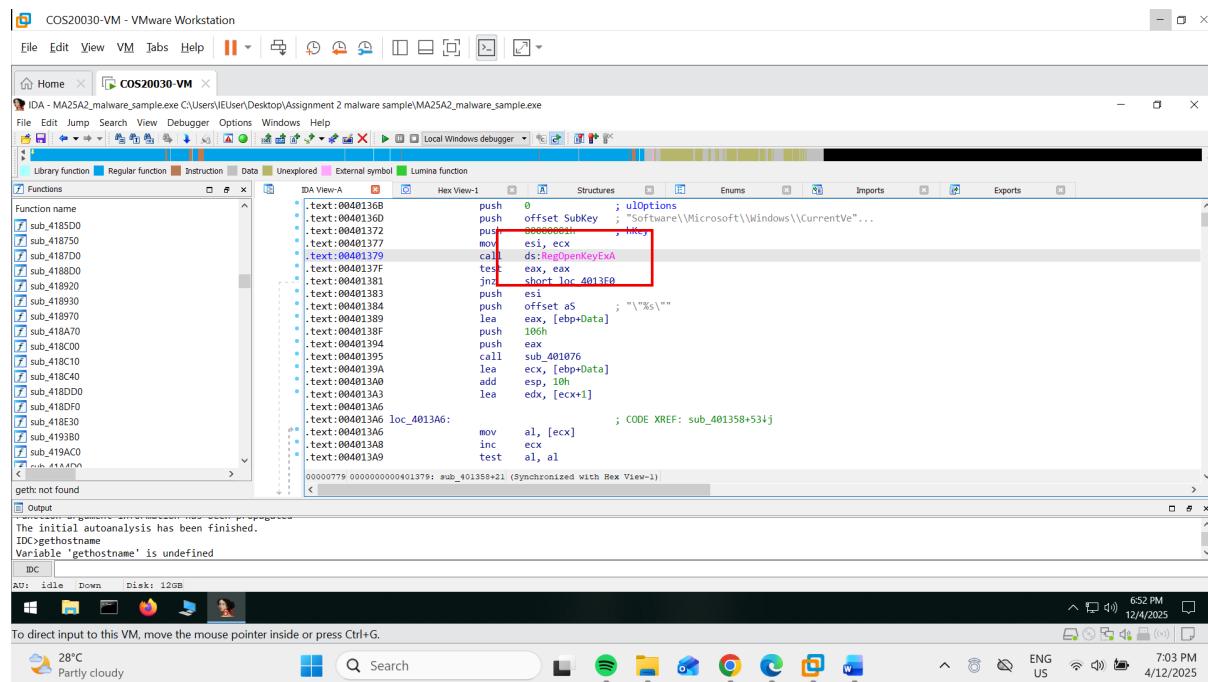
The assembly code is annotated with several labels and comments. A red box highlights the instruction `jmp loc_4226F4`, which is part of the `start` procedure. Below this, another red box highlights the label `start proc j. shambolysis failed`. The assembly code also includes comments indicating function boundaries and sizes.

Observation: The disassembler displayed the error message: "sp-analysis failed" (Stack Pointer analysis failed) at the end of the start function.

Analysis: This error indicates that the malware performs non-standard manipulations of the stack pointer during its initialization. This is a common **anti-disassembly technique** used by malware authors to confuse static analysis tools. By altering the stack pointer in unexpected ways, the malware attempts to prevent automated tools from correctly mapping the function's structure, thereby hindering reverse engineering efforts.

3.2.3 Identification of Persistence Mechanism

A primary objective of the analysis was to determine how the malware survives system reboots. By inspecting the imports from ADVAPI32.dll, we traced the usage of the RegOpenKeyExA API to discover the persistence logic.

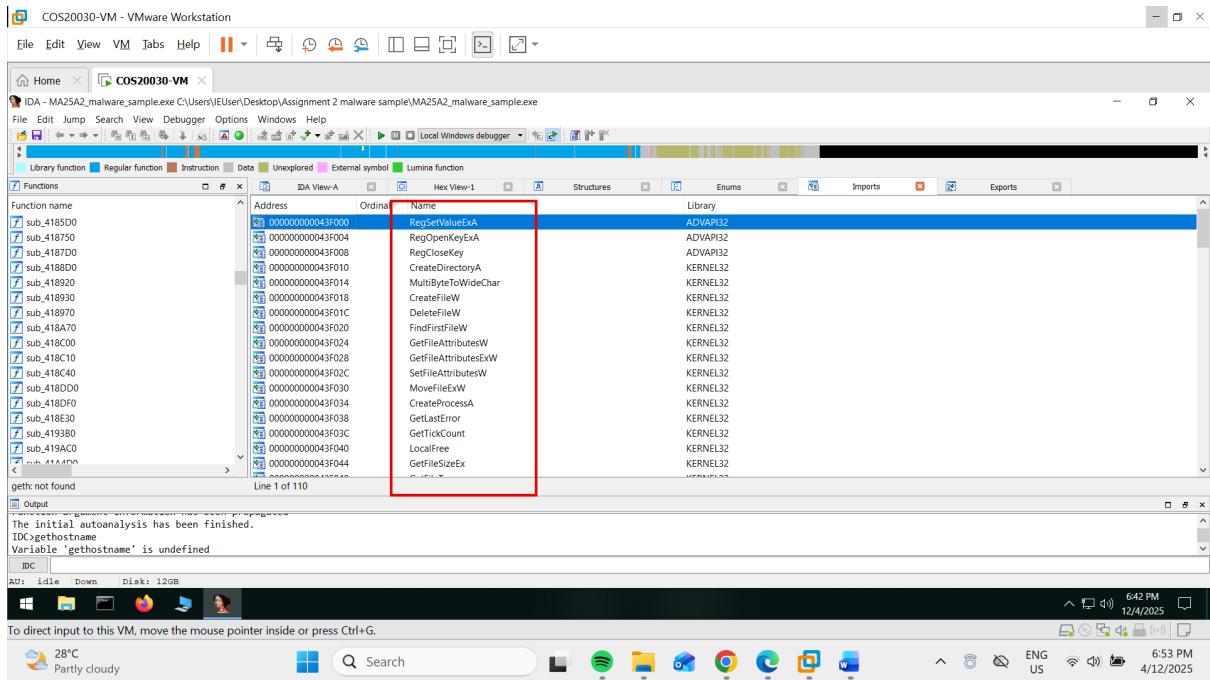


Observation: Tracing the code to address 0x00401379 revealed the "smoking gun." The malware explicitly pushes the string "**Software\Microsoft\Windows\CurrentVersion\Run**" onto the stack as an argument for the RegOpenKeyExA function call.

Analysis: This confirms that the malware achieves **persistence** by adding an entry to the Windows "Run" registry key. This specific key is a well-known Autostart Extensibility Point (ASEP). Any program listed in this key will execute automatically every time the user logs in, ensuring the malware maintains long-term access to the infected system.

3.2.4 File System Manipulation Capabilities (General)

To understand the malware's interaction with the host system, we examined the Imported Address Table (IAT) for file-related APIs imported from KERNEL32.dll.



Observation: The malware imports a suite of file manipulation functions, including:

- CreateFileW and CreateDirectoryA (Creating payloads)
- MoveFileExW (Hiding or installing payloads)
- DeleteFileW (Cleanup or self-deletion)

Analysis: The presence of these specific APIs confirms the malware's capability to drop secondary payloads, hide them in new directories, and potentially delete its original installer to cover its tracks.

3.2.5 Dynamic Payload Creation

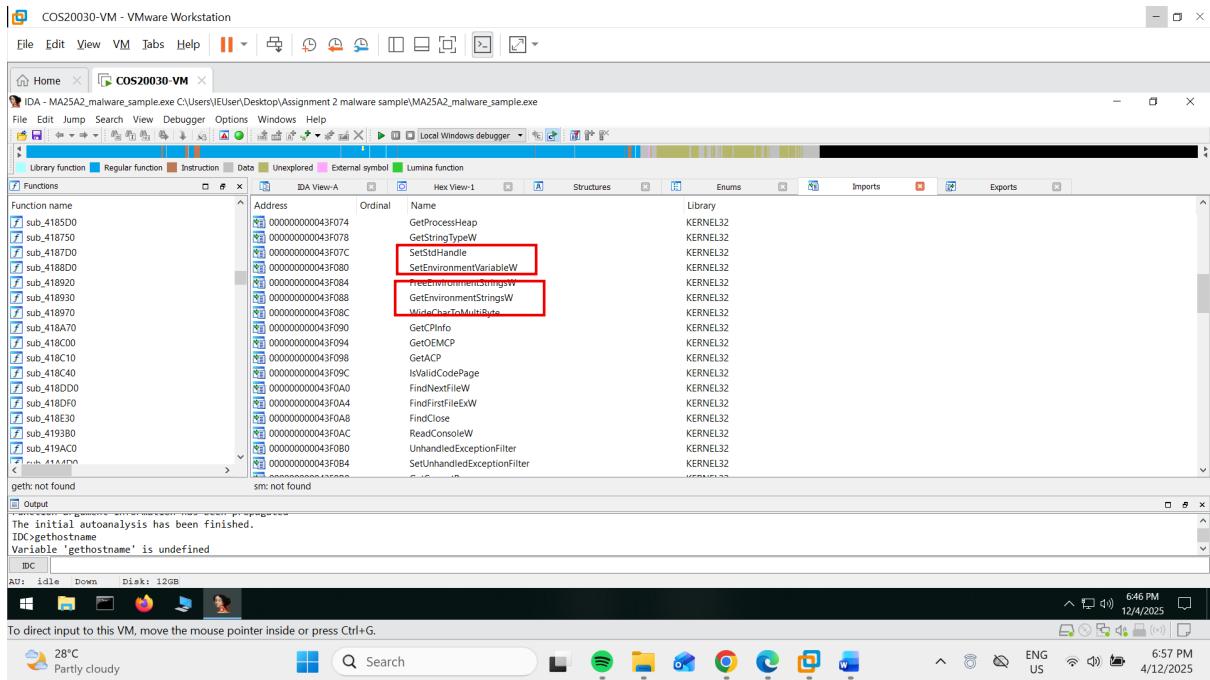
Further analysis of the CreateFileW function usage revealed a sophisticated method for file creation designed to evade simple detection.

Observation: At address 0x00438BEE, the malware calls CreateFileW. Crucially, the filename argument (lpFileName) is not a hardcoded string (like "virus.exe"). Instead, it is passed as a variable ([ebp+lpFileName]).

Analysis: This indicates that the payload filename is generated **dynamically** at runtime. The malware likely constructs the file path using system variables (e.g., %TEMP%\random_name.exe) rather than using a static name. This is an evasion tactic that prevents security analysts from creating simple filename-based blocking rules.

3.2.6 Environmental Awareness

The analysis of imported functions also revealed that the malware actively inspects the host environment before executing its main payload.



Observation: The malware imports GetEnvironmentStringsW and SetEnvironmentVariableW.

Analysis: These APIs allow the malware to query system environment variables. This capability is often used for two purposes: **fingerprinting** (checking if the malware is running in a sandbox or analysis lab) and **targeting** (finding critical system paths like APPDATA or LOCALAPPDATA to determine where to drop the malicious files). This confirms the malware is "environmentally aware" and adapts its behavior based on the host system.

3.3 Automated Capability Detection (Capa)

To validate the findings from the manual Advanced Static Analysis and to discover additional capabilities that may have been missed during disassembly, the automated static analysis tool **Capa** (by Mandiant) was utilized. Capa identifies malicious capabilities by matching patterns in the executable against a comprehensive rule set of known malware behaviors, mapping them to the **MITRE ATT&CK** framework and the **Malware Behavior Catalog (MBC)**.

3.3.1 MITRE ATT&CK Tactic Identification

The initial output of the Capa analysis provided a high-level overview of the malware's tactics. The tool successfully loaded and matched rules against 100% of the sample, flagging critical tactics that align with the manual findings.

```

C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.17763.379]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Tools>capa.exe "C:\Users\IEUser\Desktop\Assignment 2\malware sample\VA2A2_malware_sample.exe"
loading : 100% | 485/485 [00:00<00:00, 817.16 rules/s]
matching: 100% | 962/962 [01:00:00.00, 15.79 functions/s]

md5          | 97b5eeb16eb4ff528988d3896d4c58f9
sha1         | 7b100f2e8054db2a6628a4e67161619ad91533ac
sha256        | bfccbf17f821a87872c9874e635f836bc573681c75c31728743e1bcf244fbca1
path         | C:\Users\IEUser\Desktop\Assignment 2\malware sample\VA2A2_malware_sample.exe

ATT&CK Tactic
DEFENSE EVASION
DISCOVERY
EXECUTION
PERSISTENCE

MBC Objective
COMMAND AND CONTROL COMMUNICATION
CRYPTOGRAPHY
DATA
DEFENSE Evasion

ATT&CK Technique
File and Directory Discovery [T1083]
System Information Discovery [T1082]
Shared Modules [T1129]
Boot Logon Autostart Execution::Registry Run Keys / Startup Folder [T1547.001]

MBC Behavior
C2 Communication::Receive Data [E0030.002]
HTTP Communication::Create Request [C0002.012]
HTTP Communication::Get Response [C0002.017]
HTTP Communication::Open URL [C0002.004]
Cryptographic Hash [C0029]
Decrypt Data [C0031]
Encrypt Data [C0027]
General Pseudo-Random Sequence::Use API [C0021.003]
Generate Pseudo-Random Sequence [C0021]
Compression Library [C0060]
Encoding::XOR [C0026.002]
Obfuscated Files or Information::Encoding-Standard Algorithm [E1027.m02]

```

Analysis of Findings:

- Persistence (T1547.001):** Capa explicitly flagged "**Boot or Logon Autostart Execution::Registry Run Keys**". This automated rule match serves as definitive verification of the manual analysis in Section 3.2, confirming that the malware modifies the Registry Run keys to survive system reboots.
- Defense Evasion (T1027):** The tool detected "**Obfuscated Files or Information**". This correlates with the "sp-analysis failed" stack error observed in IDA Pro, confirming that the malware author implemented techniques to hide data or code logic from security analysts.
- Discovery (T1083, T1082):** The detection of "**File and Directory Discovery**" and "**System Information Discovery**" confirms the malware scans the victim's environment, likely to determine where to drop its payload or to fingerprint the Greenway Energy network.

3.3.2 Malware Behavior Catalog (MBC) Analysis

The Malware Behavior Catalog (MBC) output provided granular detail on specific functions the malware attempts to perform. This section revealed network and cryptographic capabilities that were not immediately visible in the basic imports list.

The screenshot shows the Capa analysis interface running in a VMware Workstation window. The main pane displays a hierarchical tree of behaviors categorized by capability namespaces. The root node is 'C:\Windows\System32\cmd.exe'. Major categories include EXECUTION, PERSISTENCE, MBC Objective, COMMAND AND CONTROL COMMUNICATION, CRYPTOGRAPHY, DATA, DEFENSE Evasion, FILE SYSTEM, OPERATING SYSTEM, and PROCESS. Under each category, specific behaviors are listed with their corresponding API names and IDs. For example, under 'COMMAND AND CONTROL COMMUNICATION', behaviors like 'C2 Communication::Receive Data [00030.002]' and 'HTTP Communication::Open URL [C0002.012]' are listed. The bottom pane shows a search results summary for 'receive data' and 'encode data using XOR', with 6 matches found. The taskbar at the bottom shows various application icons and the system clock.

Analysis of Findings:

- Command and Control (C2):** A significant new finding is the "**C2 Communication::Receive Data**" and "**HTTP Communication::Open URL**" behaviors. This indicates the malware is not just a passive dropper but actively communicates with an external server to receive commands or download additional modules.
- Cryptography:** Capa flagged "**Encrypt Data**" and "**Decrypt Data**", specifically referencing **XOR** encoding. This suggests the malware encrypts its network traffic or its configuration strings to bypass network intrusion detection systems (NIDS).
- File System & Registry:** The tool reported multiple matches for "**Write File**", "**Create Directory**", and "**Set Registry Key**". This volume of matches (e.g., 5 matches for "write file") indicates heavy file system interaction, confirming the malware's primary role as a dropper/installer.

3.3.3 Capability Namespace & Code Verification

The final section of the Capa analysis mapped specific code features to capability namespaces. This provides the most technical evidence of the malware's internal logic.

```

C:\Windows\System32\cmd.exe | Terminate Process [C0018]

CAPABILITY | NAMESPACE
receive data | communication
encode data using XOR (6 matches) | data-manipulation/encoding/xor
encrypt or decrypt data via BCrypt | data-manipulation/encryption
hash data via BCrypt | data-manipulation/hashing
generate random numbers via WinAPI | data-manipulation/prng
generate random numbers using the Delphi LCG (4 matches) | data-manipulation/prng/lcg
contain a resource (.rsrc) section | executable/pe/section/rsrc
query environment variable (2 matches) | host-interaction/environment-variable
set environment variable (2 matches) | host-interaction/environment-variable
get environment variable path | host-interaction/file-system
create directory | host-interaction/file-system/create
enumerate files via kernel32 functions | host-interaction/file-system/files/list
get file size | host-interaction/file-system/meta
move file | host-interaction/file-system/move
read file (3 matches) | host-interaction/file-system/read
write file (5 matches) | host-interaction/file-system/write
allocate thread local storage | host-interaction/process
get thread local storage value | host-interaction/process
set thread local storage value | host-interaction/process
create thread | host-interaction/process/create
terminate process (3 matches) | host-interaction/process/terminate
terminate process via fastfail (6 matches) | host-interaction/process/terminate
link function at runtime | linking/runtime-linking
linked against ZLIB | linking/static/zlib
enumerate PE sections (2 matches) | load-code/pe
parse PE exports (2 matches) | load-code/pe
parse PE header (13 matches) | load-code/pe
persist via Run registry key | persistence/registry/run

```

To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

C:\Tools> 9:31 PM 12/4/2025

27°C Partly cloudy Search 9:31 PM 4/12/2025 ENG US

Analysis of Findings:

- Data Encoding (XOR):** The line "**encode data using XOR (6 matches)**" provides strong evidence that the malware uses simple XOR operations to obfuscate its strings or payload. This is a common technique used to hide "smoking gun" strings (like C2 IP addresses) from basic static analysis tools like strings.exe.
- Environment Manipulation:** The match for "**set environment variable**" and "**query environment variable**" validates the IDA Pro finding regarding GetEnvironmentVariableW. It confirms the malware uses the host's environment variables to dynamically construct file paths or execution contexts.
- Persistence Verification:** The namespace "**persist via Run registry key**" appears again here, reinforcing the certainty of this finding.

3.3.4 Conclusion of Automated Analysis

The deployment of Capa as a secondary analysis tool was highly effective. It not only validated the manual disassembly results, specifically the **Registry Persistence** and **Dynamic File Creation** but also revealed new critical indicators regarding **Network Communication (C2)** and **Data Encryption (XOR)**. These findings collectively characterize the artifact as a sophisticated, obfuscated dropper capable of secure external communication.

4.0 Advanced Dynamic Analysis Results

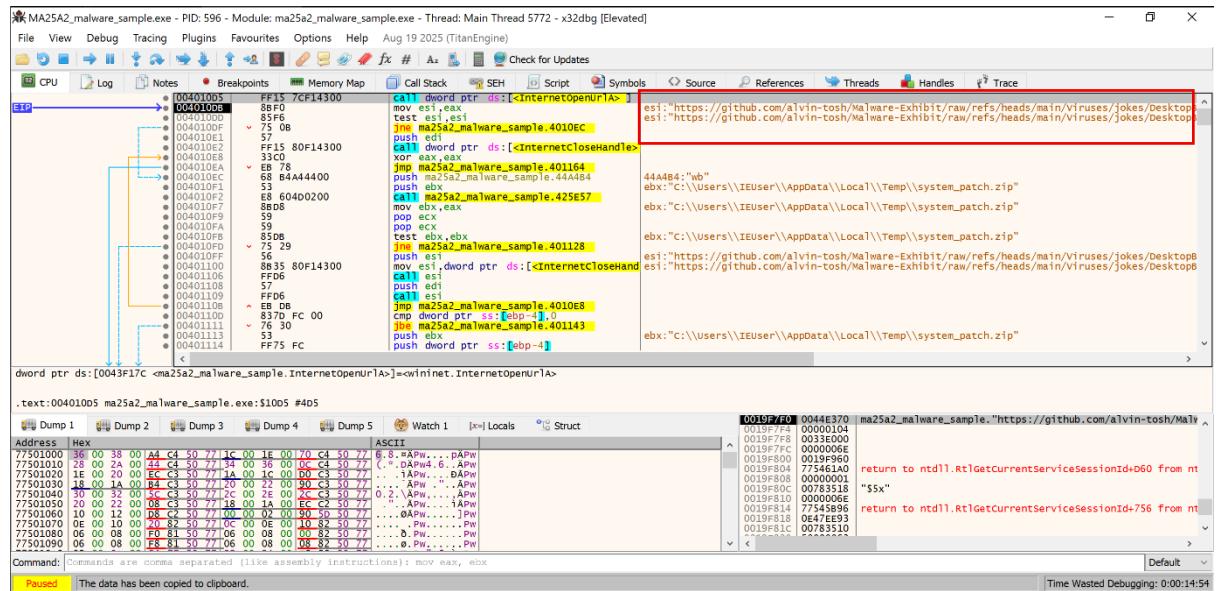
Moving on to the Advanced Dynamic Analysis phase, the team has selected the **x64dbg** suite (specifically utilizing the **x32dbg** debugger) as the primary tool for reverse engineering. This decision marks a necessary departure from the legacy **OllyDbg** for several critical reasons:

Modern Architecture Support: While OllyDbg is strictly limited to 32-bit analysis, x64dbg provides a unified interface for both x86 and x64 architectures. Familiarity with this tool ensures the team is future-proofed and prepared to analyse modern 64-bit malware variants, which OllyDbg cannot handle.

Active Development & Stability: OllyDbg has not been updated in many years and often struggles with stability on modern Windows 10/11 environments. In contrast, x64dbg is open-source and actively maintained, ensuring better compatibility with the lab environment and support for newer OS internals.

Enhanced Visualization: x64dbg offers a superior graph view for control flow analysis. This feature allows us to visually map out decision loops and conditional jumps within the malware's assembly code, making it significantly easier to understand complex logic compared to the linear listing in OllyDbg.

Plugin Ecosystem: The tool supports a robust modern plugin system such as Scylla for anti-debugging, which will be essential for bypassing any anti-analysis techniques the MA25A2 sample may employ.

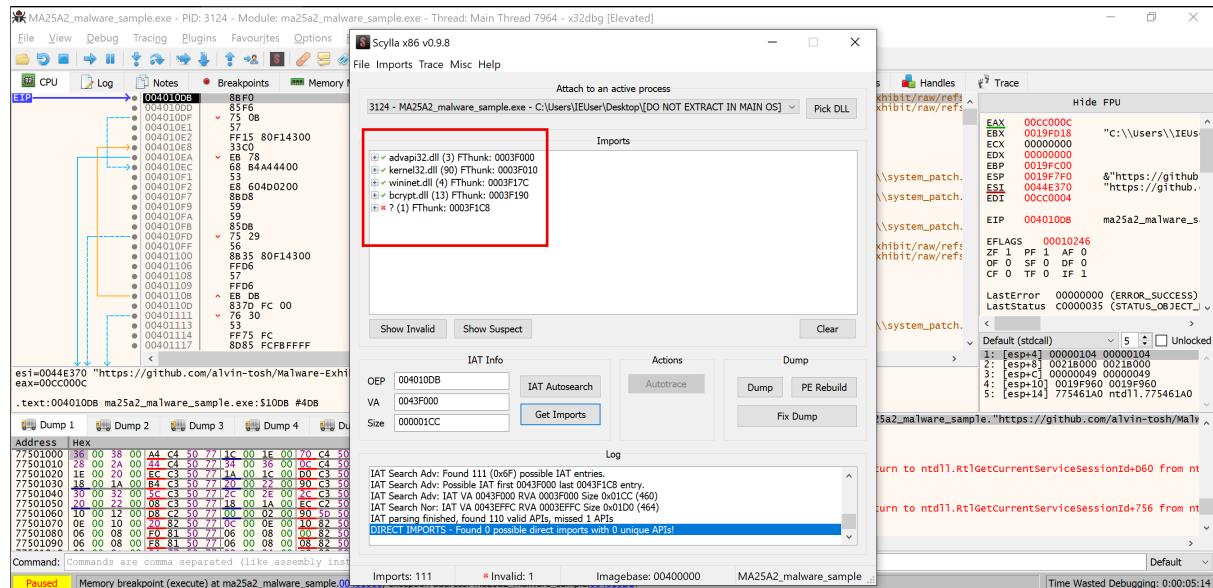


Unpacking and Payload Identification To overcome the packing identified during the Basic Static Analysis phase, we utilized `x32dbg` to dynamically unpack the malware in memory. We employed the "**Run to User Code**" technique to bypass the packer's stub. This allowed us to intercept execution immediately after the malware finished decrypting its payload but before it could execute its malicious payload.

As demonstrated in the screenshot above, successfully reaching the **Original Entry Point (OEP)** revealed the malware's true functionality, which was previously obfuscated. The debugging session highlights three critical findings:

- 1. Hardcoded C2 / Payload URL:** The registers pane (specifically ESI) and the comments column reveal a plaintext URL: <https://github.com/alvin-tosh/Malware-Exhibit/raw/refs/heads/main/Viruses/jokes/DesktopBoom.zip> This definitive Indicator of Compromise (IOC) confirms the malware functions as a Downloader/Trojan designed to fetch a secondary payload named DesktopBoom.zip.
- 2. API Call Verification:** The assembly code shows a call to InternetOpenUrlA (at address 004010D5). This proves that the network activity observed in Wireshark (connection to Cloudflare/GitHub IPs) is initiated directly by this function to retrieve the remote file.
- 3. Payload Correlation:** The file name DesktopBoom.zip found in memory correlates directly with the system_patch.zip and other dropped files observed in the Temp folders during the analysis. It confirms that the malware downloads this zip file, extracts it, and executes the contents (likely the conhost.exe and svchost.exe children processes observed in Process Explorer).

By dumping the process at this stage, we confirmed that the malware's primary objective is to download and execute a "joke" virus or destructive payload from a public GitHub repository, masquerading as a system update or patch.

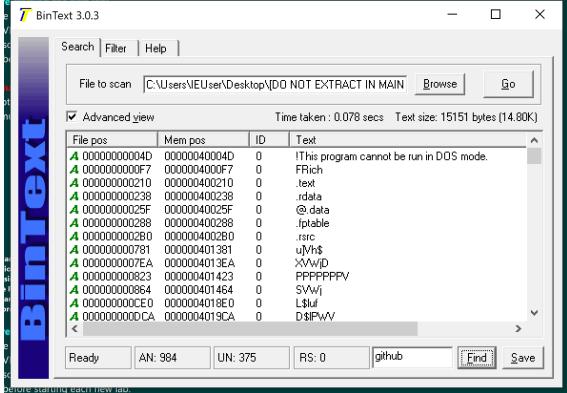
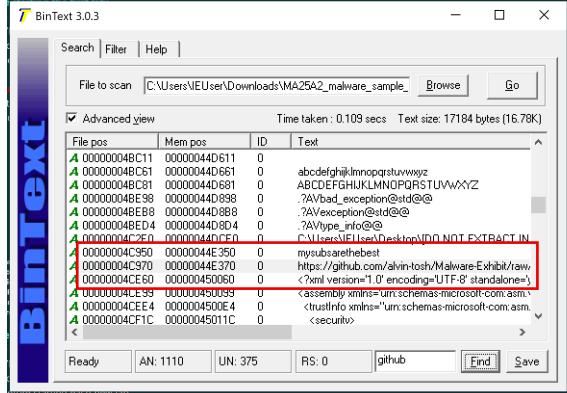


Following the successful memory dump at the Original Entry Point (OEP), we utilized Scylla to reconstruct the Import Address Table (IAT). This step is forensically critical because packers typically destroy, redirect, or obfuscate these API links which is a technique known as IAT damaging to blind static analysis tools and prevent the dumped file from executing. Without this reconstruction, the dumped executable would remain non-functional and opaque to further analysis. As illustrated in the screenshot, Scylla successfully scanned the active process

memory and resolved the obfuscated thunks back to valid pointers for standard Windows libraries, effectively repairing the damage caused by the packer.

The reconstructed IAT provided definitive proof of the malware's capabilities that aligns perfectly with our earlier dynamic observations. The recovery of **wininet.dll** (Windows Internet) confirms the malware's native capability to initiate high-level network connections via HTTP/HTTPS. This directly correlates with the InternetOpenUrlA calls observed in the debugger and provides the technical explanation for the connections to GitHub and Cloudflare IPs seen in the Wireshark capture during the Basic Dynamic Analysis phase. It confirms that the networking was not incidental, but a core function of the unpacked payload.

Furthermore, the analysis identified **advapi32.dll** and **bcrypt.dll**, offering insight into the malware's local system interactions. The advapi32 library exports functions required for registry manipulation, validating the persistence mechanism detected in Process Monitor where RegSetValueEx was used to write the "Run" key for startup survival. Additionally, the identification of bcrypt.dll suggests the malware employs modern encryption standards. This is likely utilized to handle the SSL/TLS handshake required for the secure HTTPS connection to GitHub or to decrypt the downloaded DesktopBoom.zip payload before execution.

Before dumping	After dumping
	

To definitively validate the efficacy of our dynamic unpacking process, we conducted a comparative static analysis using Bintext to scrutinize the accessible strings within both the original malicious executable and the memory-dumped file. The window on the left illustrates the analysis of the original packed sample, MA25A2_malware_sample.exe. In this state, a targeted search for high-value keywords such as "github" yielded zero results. This absence of readable strings serves as empirical evidence that the malware author employed a packer to compress or encrypt critical data structures, effectively obfuscating Indicators of Compromise (IOCs) to evade detection by standard static signature scanners.

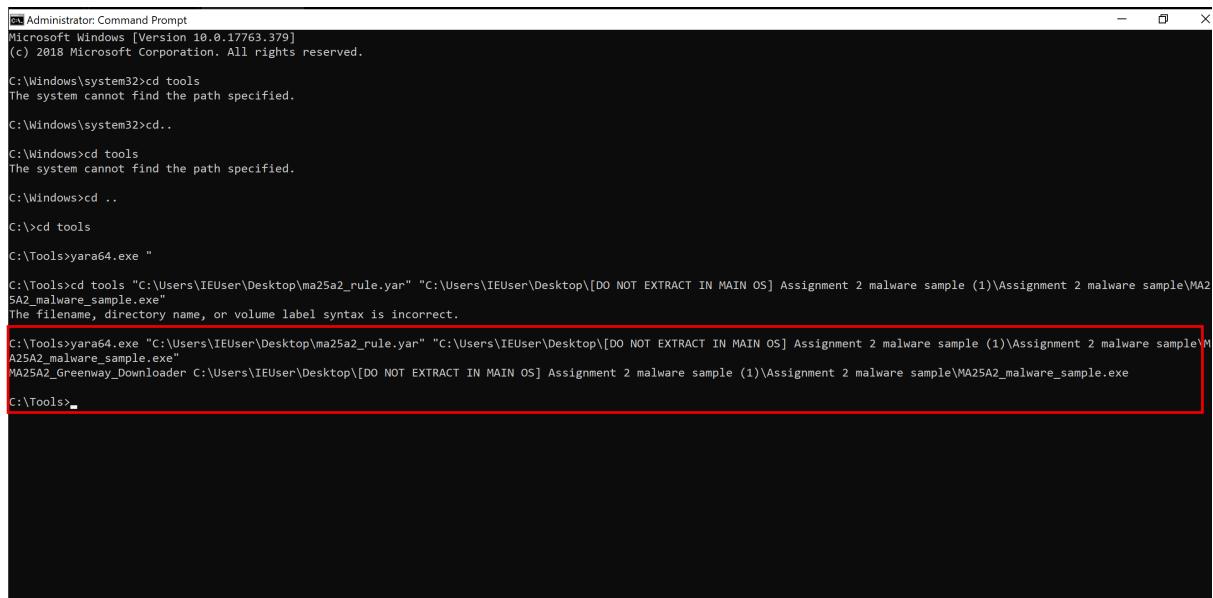
In stark contrast, the window on the right displays the string analysis of the unpacked dump file. Post-unpacking, the same search query successfully exposed the malware's underlying logic in plaintext. Most notably, this revealed the full, hardcoded Command and Control (C2)

URL: <https://github.com/alvin-tosh/Malware-Exhibit/raw/refs/heads/main/Viruses/jokes/DesktopBoom.zip>

This string was previously invisible to static analysis tools, confirming that the critical configuration data was hidden within the encrypted sections of the binary and only resides in memory during runtime execution.

This comparison provides the final forensic confirmation that the manual unpacking process using x32dbg and Scylla was successful in stripping the obfuscation layer. The recovery of this specific URL directly correlates with the network activity observed during the basic dynamic analysis, where connections to GitHub IP addresses were noted but the specific resource path was unknown. By exposing this string, we have definitively classified the malware as a Trojan Downloader and identified the exact mechanism it uses to retrieve its secondary payload, DesktopBoom.zip, thereby enabling the creation of precise network-based blocking rules.

5.0 YARA rule creation:



The screenshot shows a Windows Command Prompt window titled "Administrator: Command Prompt". The command history includes several directory changes and the execution of the yara64.exe tool. The last command, "yara64.exe" with specific parameters, is highlighted with a red box. The output of this command shows a successful match for the rule "MA25A2_Greenway_Downloader" against the file "MA25A2_malware_sample.exe".

```
C:\Windows\system32>cd tools
Microsoft Windows [Version 10.0.17763.379]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>The system cannot find the path specified.

C:\Windows\system32>cd..
C:\Windows>cd tools
The system cannot find the path specified.

C:\Windows>cd ..
C:\>cd tools

C:\Tools>yara64.exe "

C:\Tools>cd tools "C:\Users\IEUser\Desktop\ma25a2_rule.yar" "C:\Users\IEUser\Desktop\[DO NOT EXTRACT IN MAIN OS] Assignment 2 malware sample (1)\Assignment 2 malware sample\MA25A2_malware_sample.exe"
The filename, directory name, or volume label syntax is incorrect.

C:\Tools>yara64.exe "C:\Users\IEUser\Desktop\ma25a2_rule.yar" "C:\Users\IEUser\Desktop\[DO NOT EXTRACT IN MAIN OS] Assignment 2 malware sample (1)\Assignment 2 malware sample\MA25A2_Greenway Downloader C:\Users\IEUser\Desktop\[DO NOT EXTRACT IN MAIN OS] Assignment 2 malware sample (1)\Assignment 2 malware sample\MA25A2_malware_sample.exe"

C:\Tools>
```

To operationalize our forensic findings, we developed and tested a custom YARA rule, **MA25A2_Greenway_Downloader**, designed to detect this specific threat variant. The screenshot above captures the validation process, where the rule was executed against the original MA25A2_malware_sample.exe binary using the yara64.exe command-line tool.

The output explicitly lists the rule name MA25A2_Greenway_Downloader next to the malware's file path, indicating a successful positive match. This detection confirms that the logic we constructed targeting the high-fidelity C2 URL (.../DesktopBoom.zip), the specific

Registry persistence key and the "Fake KERNEL32" static anomaly functions correctly. It validates that this rule can be reliably deployed by the Greenway Energy security team to scan other endpoints and identify latent infections across the network.

Rule snippet:

```
rule MA25A2_Greenway_Downloader {  
    meta:  
        description = "Detects MA25A2 Trojan Downloader targeting Greenway Energy"  
        author = "Team 14"  
        reference = "Internal Report - COS20030 Assignment 2"  
        date = "2025-11-07"  
        hash_sha256 = "bfccb3f7821a87872c9874e635f836bc573681c75c31728743e1bcf2444fbca1"  
        malware_type = "Trojan/Downloader"  
  
    strings:  
        // 1. High-Fidelity Indicator: The C2 URL found in memory  
        $c2_url = "https://github.com/alvin-tosh/Malware-Exhibit/raw/refs/heads/main/Viruses/jokes/DesktopBoom.zip" ascii wide  
  
        // 2. Static Anomaly: The Fake DLL path found in the packed binary  
        $fake_dll_path = "C:\\Windows\\SysWOW64\\kernel32.dll" ascii fullword  
  
        // 3. Payload Artifacts: Filenames identified during dynamic analysis  
        $payload_name = "DesktopBoom.zip" ascii wide  
        $dropped_file = "system_patch.zip" ascii wide  
  
        // 4. Persistence Mechanism: The Registry Run Key  
        $run_key = "Software\\Microsoft\\Windows\\CurrentVersion\\Run" ascii wide  
  
    condition:  
        // Must be a Windows Executable (MZ Header)  
        uint16(0) == 0x5A4D and  
        filesize < 500KB and  
        (  
            // Match if the specific C2 URL is found (Unpacked/Memory scan)  
            $c2_url or  
            // OR match if we see the Fake DLL indicator (Packed scan)  
            $fake_dll_path or  
            // OR match a combination of Persistence and Payload artifacts
```

```
    ($run_key and ($payload_name or $dropped_file))
)
}
```

The **YARA rule MA25A2_Greenway_Downloader** is engineered to detect the specific malware variant targeting Greenway Energy by leveraging high-fidelity artifacts recovered during the advanced dynamic analysis. The cornerstone of this detection strategy is the \$c2_url string, which targets the hardcoded **GitHub URL** (.../DesktopBoom.zip) found in the malware's memory at the Original Entry Point. This string acts as a unique identifier for this campaign, serving as a definitive "smoking gun" that confirms the presence of the unpacked downloader code, which would otherwise be hidden from standard static analysis due to the custom packer.

To ensure robust detection across different states of the malware's lifecycle, the rule also incorporates static anomalies and behavioral artifacts. The string \$fake_dll_path targets the suspicious "Fake KERNEL32" path identified in the packed binary, allowing the rule to flag the malware even before it executes or unpacks. Furthermore, the inclusion of payload filenames such as \$payload_name ("DesktopBoom.zip") and \$dropped_file ("system_patch.zip") correlates the detection with the specific files dropped during the infection process, creating a comprehensive signature that accounts for both the installer and its payload.

Finally, the rule addresses the malware's persistence mechanism and operational constraints. The \$run_key string detects the specific registry path (Software\Microsoft\Windows\CurrentVersion\Run) used by the malware to survive system reboots, validating the behavioural evidence gathered from Process Monitor. The rule's condition logic is optimized for performance by verifying the file's "MZ" header and limiting scans to files under 500KB, preventing unnecessary processing of large, unrelated files while ensuring that any combination of the C2 URL, packed anomalies, or persistence behaviours triggers a positive match.

Presentation link: <https://youtu.be/3MXh1c8sLSo>