CALIFORNIA STATE UNIVERSITY

FULLERTON

CPSC 458 - Malware Analysis

Project 2 - Fall 2024

Toddeh Alexander

Rocco Catalasan

Phu Lam

Wayne Muse

Sokheng Teang

Malware Sample

A colleague has been working on a PowerShell script to compute the percentage of free RAM on a Windows machine. They figured out that they could use the following command to find the raw numbers they needed, the planned to finish the script when they returned from lunch.

```
PS C:\> Get-CIMInstance Win32_OperatingSystem |

Select FreePhysicalMemory, TotalVisibleMemorySize


FreePhysicalMemory TotalVisibleMemorySize

------------------ ----------------------

          27575792               41620212
```

While on their lunch break, they used a personal device to browse the “Technology” section of a well-known image-based bulletin board, and found a command-line utility that seems to do the job. Since the site they visited is a little… */g/narly*… you are suspicious.

A file named project2.7z is available in Canvas. This file is encrypted with password `malware`, and is suspected to contain malware that runs on Windows 10 and 11.

Documenting your results

When you have completed your detailed analysis, write a high-level executive summary identifying the purpose of the malware in terms of the tactics and techniques described by the MITRE ATT&CK framework.

At the beginning of your report, include the following:

- The names of each member of the team who participated in the project
- The course number, section, and semester
- The project number

For each step of your analysis:

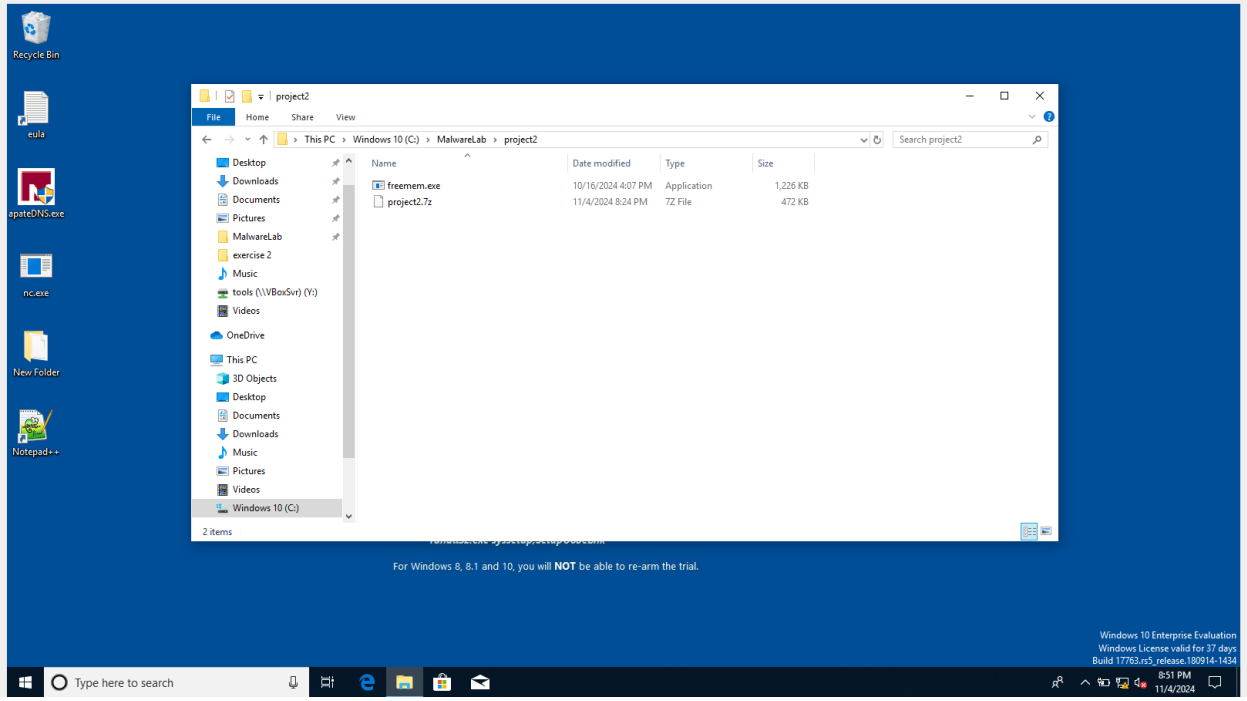
- Identify what is to be done.
- Document the tools to be used.
- Describe your results, including definitions, diagrams, screenshots, or code as appropriate.
- Comment on the results, including references you consulted, variations you considered, or issues you encountered.

As a rule of thumb, the level of detail in your report should resemble the *Detailed Analysis* sections of the *Solutions to Labs* in the textbook.

General Analysis

Determine and document the following:

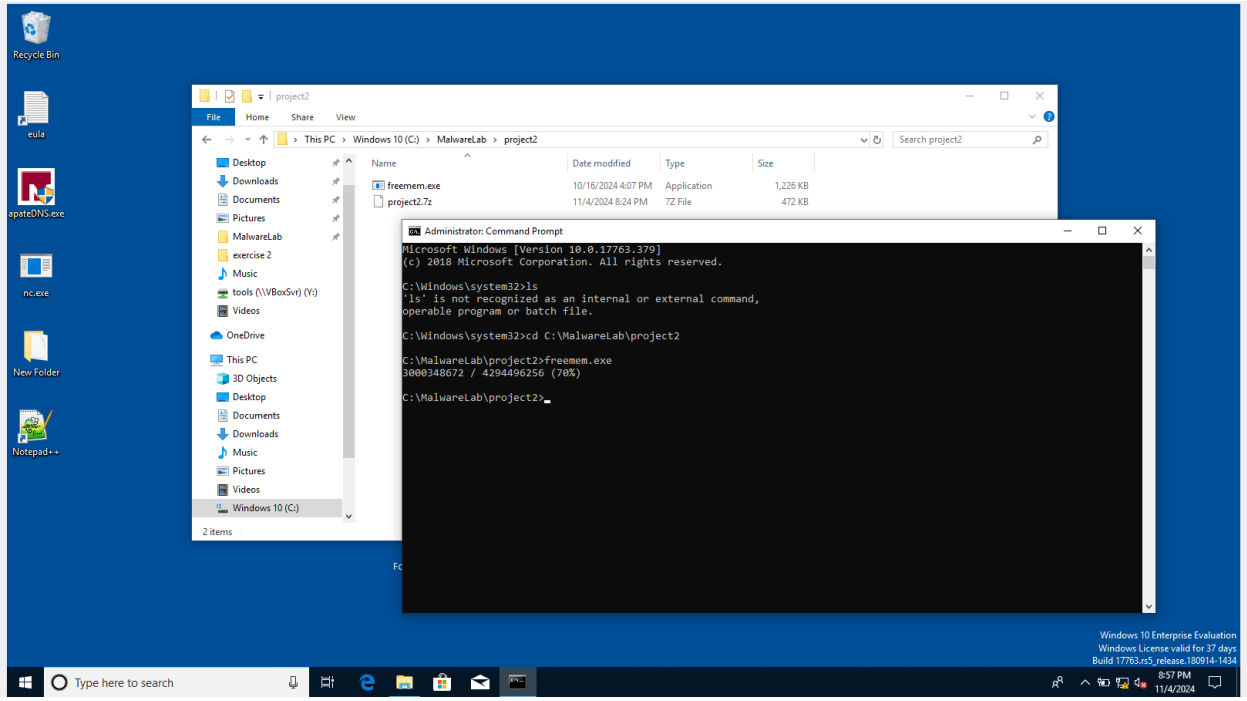
1. The surface-level functionality of the program (i.e. how it appears to work).



When double-clicking freemem.exe, nothing appears to happen:

- No windows open.
- No error messages are displayed.
- The program does not show any visible signs of execution.

2. Whether the program performs any additional actions. Describe these actions in detail, including how they are triggered.



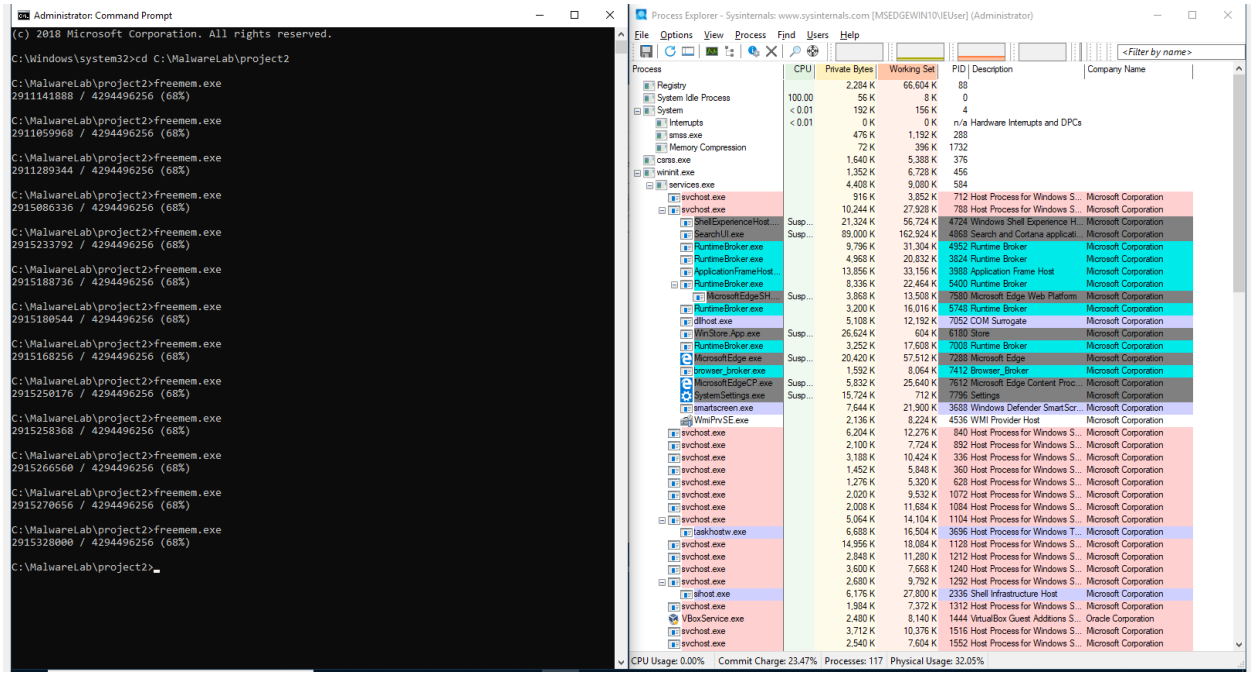
Run the malware through command prompt as administrator, we got the following message.

This output suggests that the program performs a memory usage check, which aligns with its intended functionality.

The values displayed likely represent:

- 3000348672: Free physical memory in bytes.
- 4294496256: Total visible memory size in bytes.
- (70%): The percentage of free memory available on the system.

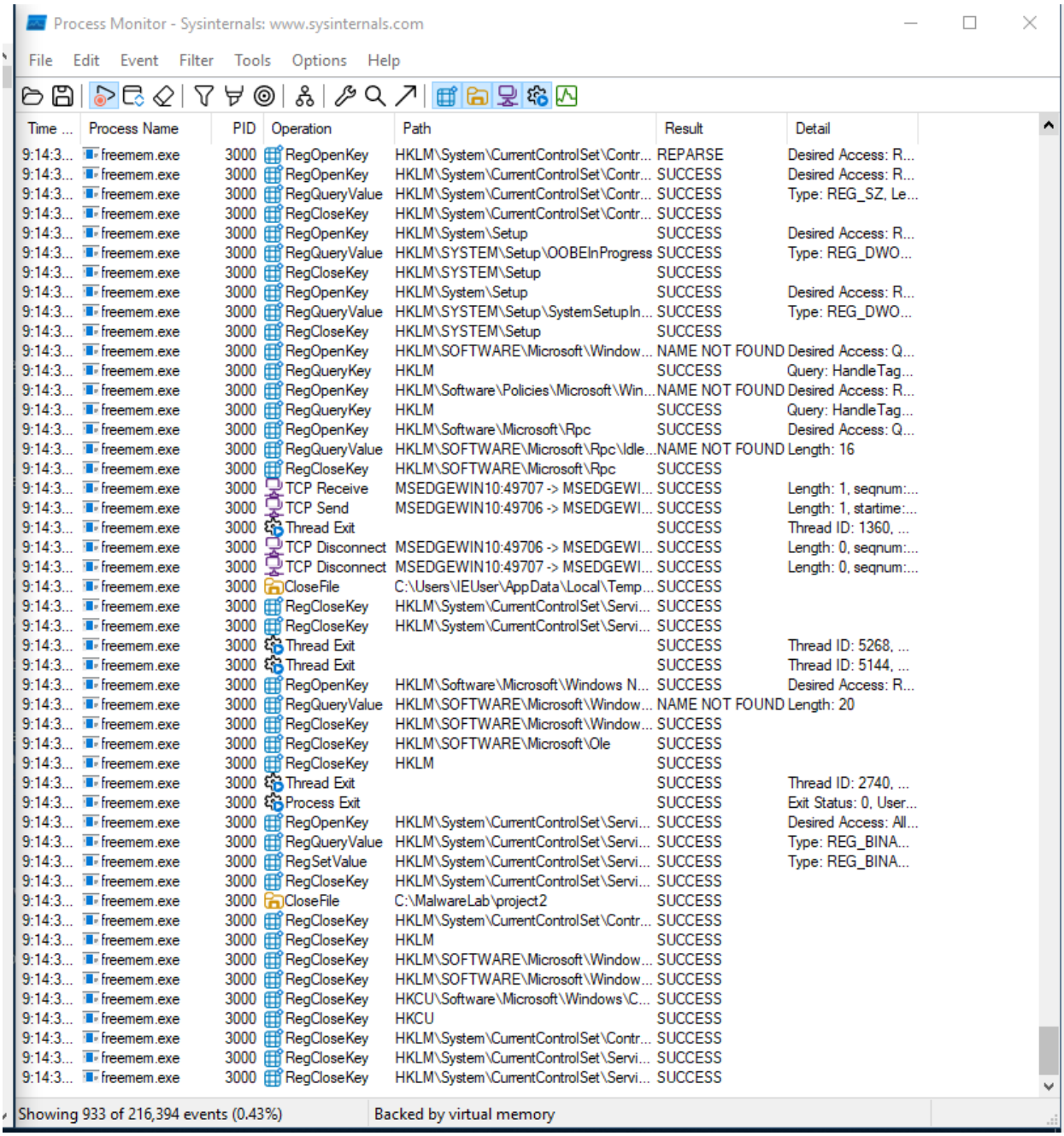
3. Host- and network-based indicators of compromise that can be used to determine whether the malware is present.

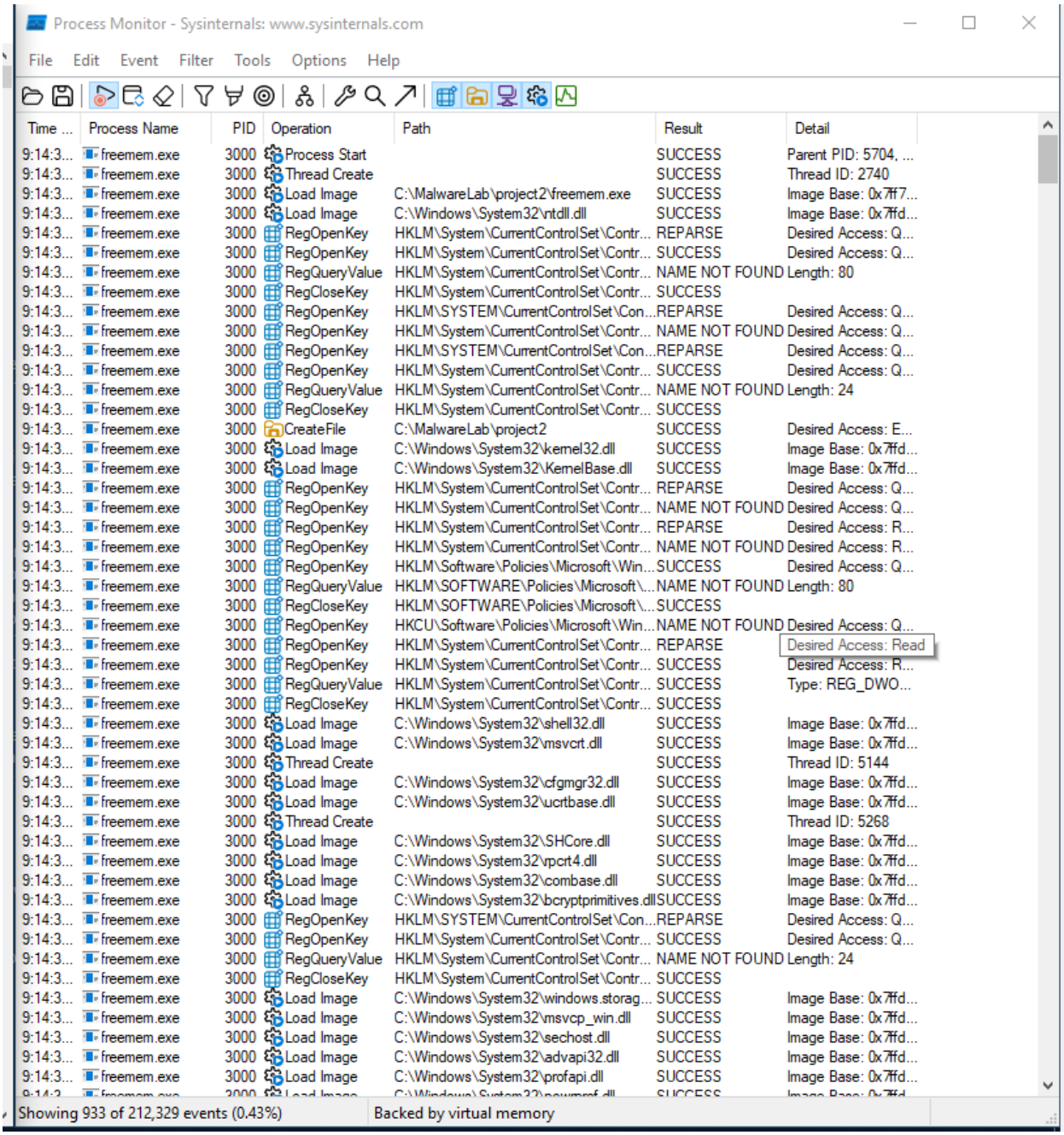


Process explorer:

- After running the malware multiple times from command prompt as administrator, I observed some processes with red color appear, but disappear back very fast in like 1 or 2 second. I cannot get a screenshot of the process name.

Process monitor


Process Monitor - Sysinternals: www.sysinternals.com
File Edit Event Filter Tools Options Help
Time ... Process Name PID Operation Path Result Detail
9:14:3... freemem.exe 3000 RegOpenKey HKLM\System\CurrentControlSet\Contr... REPARSE Desired Access: R...
9:14:3... freemem.exe 3000 RegOpenKey HKLM\System\CurrentControlSet\Contr... SUCCESS Desired Access: R...
9:14:3... freemem.exe 3000 RegQueryValue HKLM\System\CurrentControlSet\Contr... SUCCESS Type: REG_SZ, Le...
9:14:3... freemem.exe 3000 RegCloseKey HKLM\System\CurrentControlSet\Contr... SUCCESS
9:14:3... freemem.exe 3000 RegOpenKey HKLM\System\Setup SUCCESS Desired Access: R...
9:14:3... freemem.exe 3000 RegQueryValue HKLM\SYSTEM\Setup\OOBEInProgress SUCCESS Type: REG_DWO...
9:14:3... freemem.exe 3000 RegCloseKey HKLM\SYSTEM\Setup SUCCESS
9:14:3... freemem.exe 3000 RegOpenKey HKLM\System\Setup SUCCESS Desired Access: R...
9:14:3... freemem.exe 3000 RegQueryValue HKLM\SYSTEM\Setup\SystemSetupIn... SUCCESS Type: REG_DWO...
9:14:3... freemem.exe 3000 RegCloseKey HKLM\SYSTEM\Setup SUCCESS
9:14:3... freemem.exe 3000 RegOpenKey HKLM\SOFTWARE\Microsoft\Window... NAME NOT FOUND Desired Access: Q...
9:14:3... freemem.exe 3000 RegQueryKey HKLM SUCCESS Query: HandleTag...
9:14:3... freemem.exe 3000 RegOpenKey HKLM\Software\Policies\Microsoft\Win... NAME NOT FOUND Desired Access: R...
9:14:3... freemem.exe 3000 RegQueryKey HKLM SUCCESS Query: HandleTag...
9:14:3... freemem.exe 3000 RegOpenKey HKLM\Software\Microsoft\Rpc SUCCESS Desired Access: Q...
9:14:3... freemem.exe 3000 RegQueryValue HKLM\SOFTWARE\Microsoft\Rpc\Idle... NAME NOT FOUND Length: 16
9:14:3... freemem.exe 3000 RegCloseKey HKLM\SOFTWARE\Microsoft\Rpc SUCCESS
9:14:3... freemem.exe 3000 TCP Receive MSEDGEWIN10:49707 -> MSEDGEWI... SUCCESS Length: 1, seqnum:...
9:14:3... freemem.exe 3000 TCP Send MSEDGEWIN10:49706 -> MSEDGEWI... SUCCESS Length: 1, startime:...
9:14:3... freemem.exe 3000 Thread Exit SUCCESS Thread ID: 1360, ...
9:14:3... freemem.exe 3000 TCP Disconnect MSEDGEWIN10:49706 -> MSEDGEWI... SUCCESS Length: 0, seqnum:...
9:14:3... freemem.exe 3000 TCP Disconnect MSEDGEWIN10:49707 -> MSEDGEWI... SUCCESS Length: 0, seqnum:...
9:14:3... freemem.exe 3000 CloseFile C:\Users\IEUser\AppData\Local\Temp... SUCCESS
9:14:3... freemem.exe 3000 RegCloseKey HKLM\System\CurrentControlSet\Servi... SUCCESS
9:14:3... freemem.exe 3000 RegCloseKey HKLM\System\CurrentControlSet\Servi... SUCCESS
9:14:3... freemem.exe 3000 Thread Exit SUCCESS Thread ID: 5268, ...
9:14:3... freemem.exe 3000 Thread Exit SUCCESS Thread ID: 5144, ...
9:14:3... freemem.exe 3000 RegOpenKey HKLM\Software\Microsoft\Windows N... SUCCESS Desired Access: R...
9:14:3... freemem.exe 3000 RegQueryValue HKLM\SOFTWARE\Microsoft\Window... NAME NOT FOUND Length: 20
9:14:3... freemem.exe 3000 RegCloseKey HKLM\SOFTWARE\Microsoft\Window... SUCCESS
9:14:3... freemem.exe 3000 RegCloseKey HKLM\SOFTWARE\Microsoft\Ole SUCCESS
9:14:3... freemem.exe 3000 RegCloseKey HKLM SUCCESS
9:14:3... freemem.exe 3000 Thread Exit SUCCESS Thread ID: 2740, ...
9:14:3... freemem.exe 3000 Process Exit SUCCESS Exit Status: 0, User...
9:14:3... freemem.exe 3000 RegOpenKey HKLM\System\CurrentControlSet\Servi... SUCCESS Desired Access: All...
9:14:3... freemem.exe 3000 RegQueryValue HKLM\System\CurrentControlSet\Servi... SUCCESS Type: REG_BINA...
9:14:3... freemem.exe 3000 RegSetValue HKLM\System\CurrentControlSet\Servi... SUCCESS Type: REG_BINA...
9:14:3... freemem.exe 3000 RegCloseKey HKLM\System\CurrentControlSet\Servi... SUCCESS
9:14:3... freemem.exe 3000 CloseFile C:\MalwareLab\project2 SUCCESS
9:14:3... freemem.exe 3000 RegCloseKey HKLM\System\CurrentControlSet\Contr... SUCCESS
9:14:3... freemem.exe 3000 RegCloseKey HKLM SUCCESS
9:14:3... freemem.exe 3000 RegCloseKey HKLM\SOFTWARE\Microsoft\Window... SUCCESS
9:14:3... freemem.exe 3000 RegCloseKey HKLM\SOFTWARE\Microsoft\Window... SUCCESS
9:14:3... freemem.exe 3000 RegCloseKey HKCU\Software\Microsoft\Windows\C... SUCCESS
9:14:3... freemem.exe 3000 RegCloseKey HKCU SUCCESS
9:14:3... freemem.exe 3000 RegCloseKey HKLM\System\CurrentControlSet\Contr... SUCCESS
9:14:3... freemem.exe 3000 RegCloseKey HKLM\System\CurrentControlSet\Servi... SUCCESS
9:14:3... freemem.exe 3000 RegCloseKey HKLM\System\CurrentControlSet\Servi... SUCCESS
Showing 933 of 216,394 events (0.43%)
Backed by virtual memory



Filter the name freemem.exe, we got the following

Registry Access:

- freemem.exe performs numerous Registry operations, primarily RegOpenKey and RegQueryValue.
- It's accessing keys related to system configuration and control settings (HKLM\System\CurrentControlSet\Control).

File Operations:

- Creates files, including in directories like C:\MalwareLab\project2.
- CreateFile operations in the user's AppData directory, specifically C:\Users\[User]\AppData\Local\Temp. It utilizes the Temp directory to store temporary or staging files.

Network Activity:

- There are TCP Send and TCP Receive events, indicating that malware is attempting to establish network connections.

4. If a host has been compromised, how to undo the damage.

   Close the VM:

   - Shut down the virtual machine to stop all processes, network connections, and any hidden activities initiated by the malware.

   Restore from Snapshot:

   - Use the snapshot to revert the VM to a clean, unmodified state.
   - Restoring from a snapshot effectively removes all files, registry changes, network configurations, and other modifications that malware may have introduced.

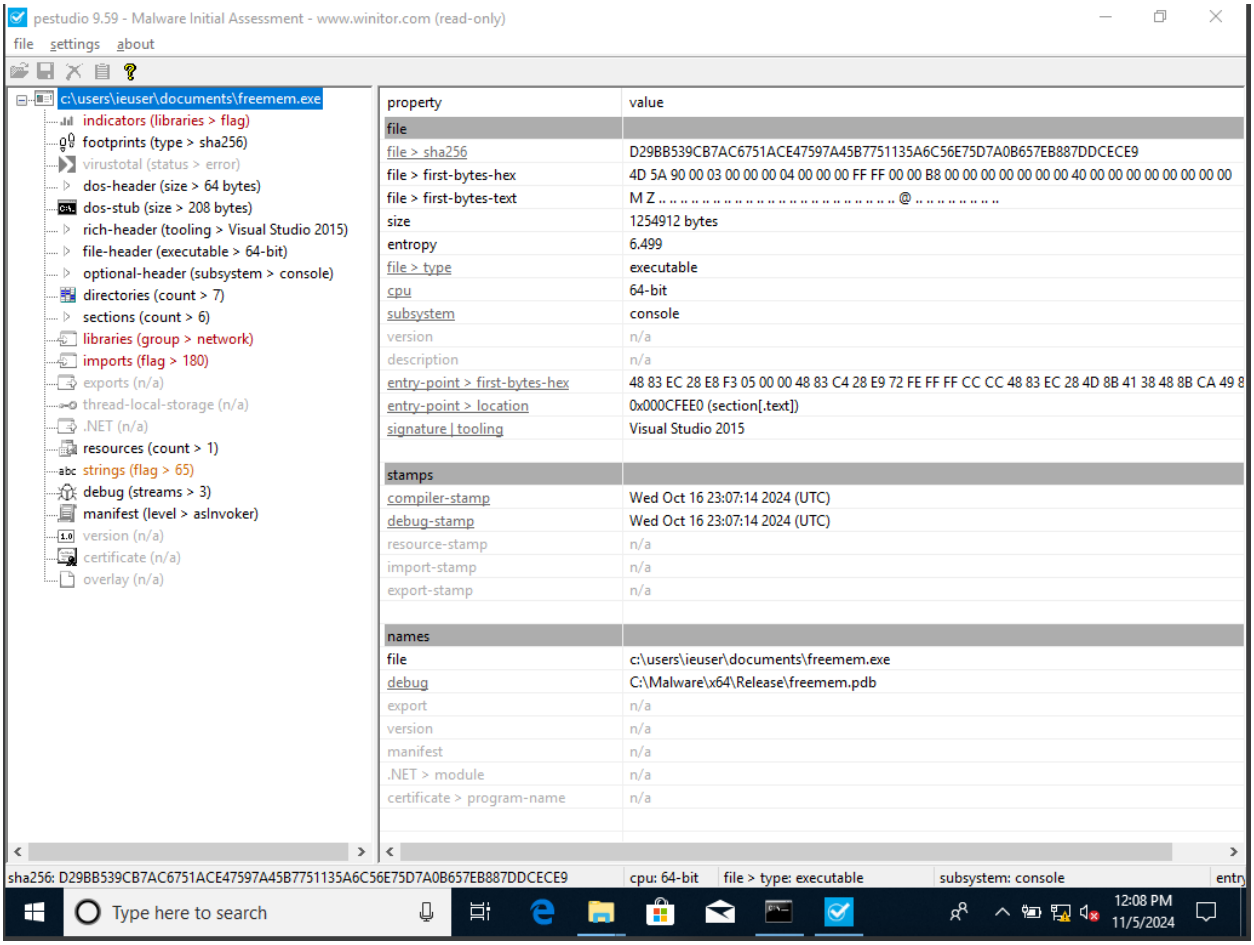
   Verify the Restoration:

   - After restoring the snapshot, reboot the VM and we verify that malware and any temporary files, registry entries, or network configurations associated with the malware are no longer present.

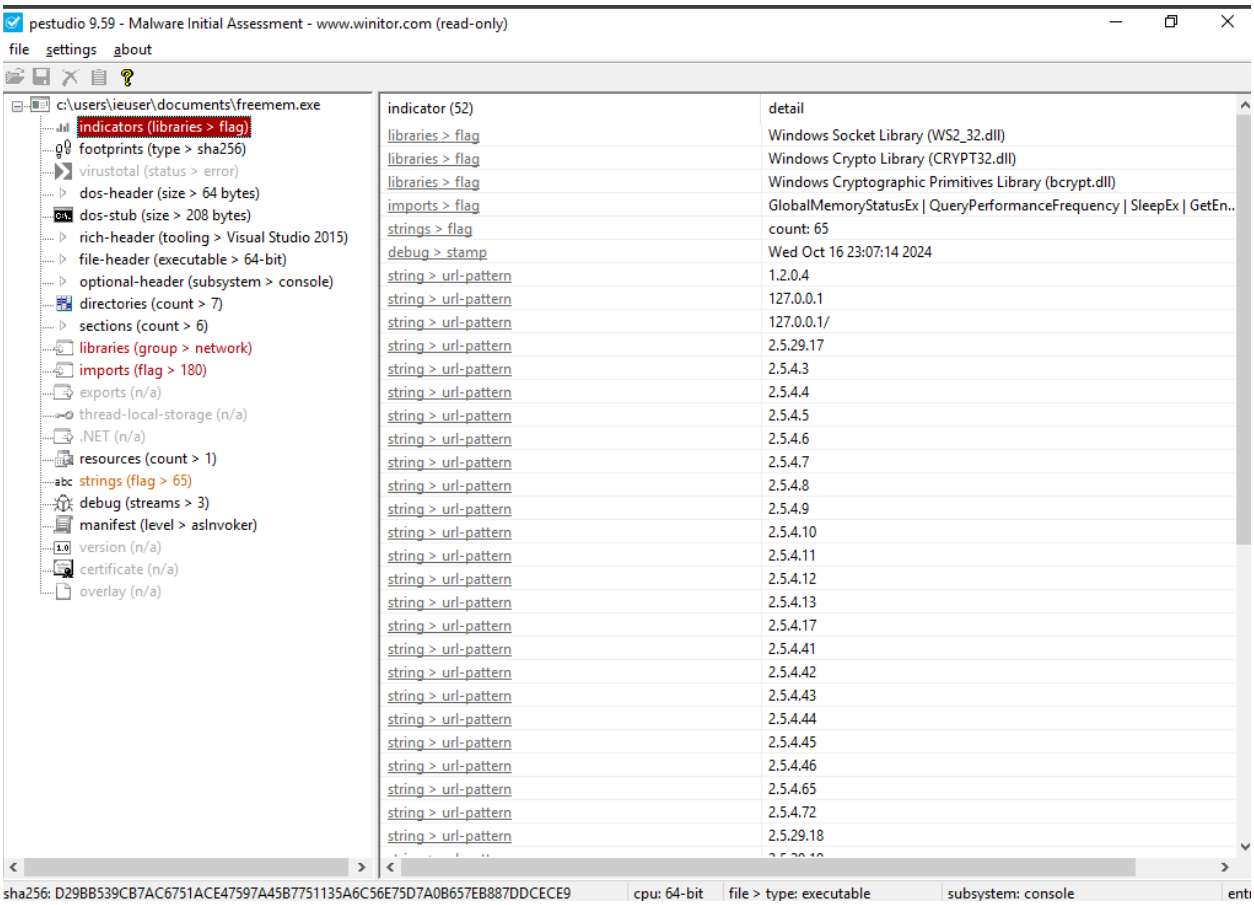
Basic Static Analysis

PeStudio

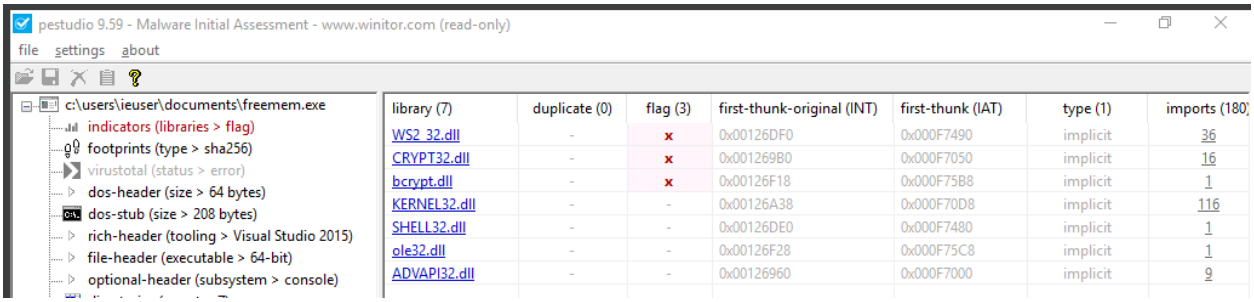
We loaded the freemem file into PEStudio to do a basic static analysis. After installing the PEStudio:

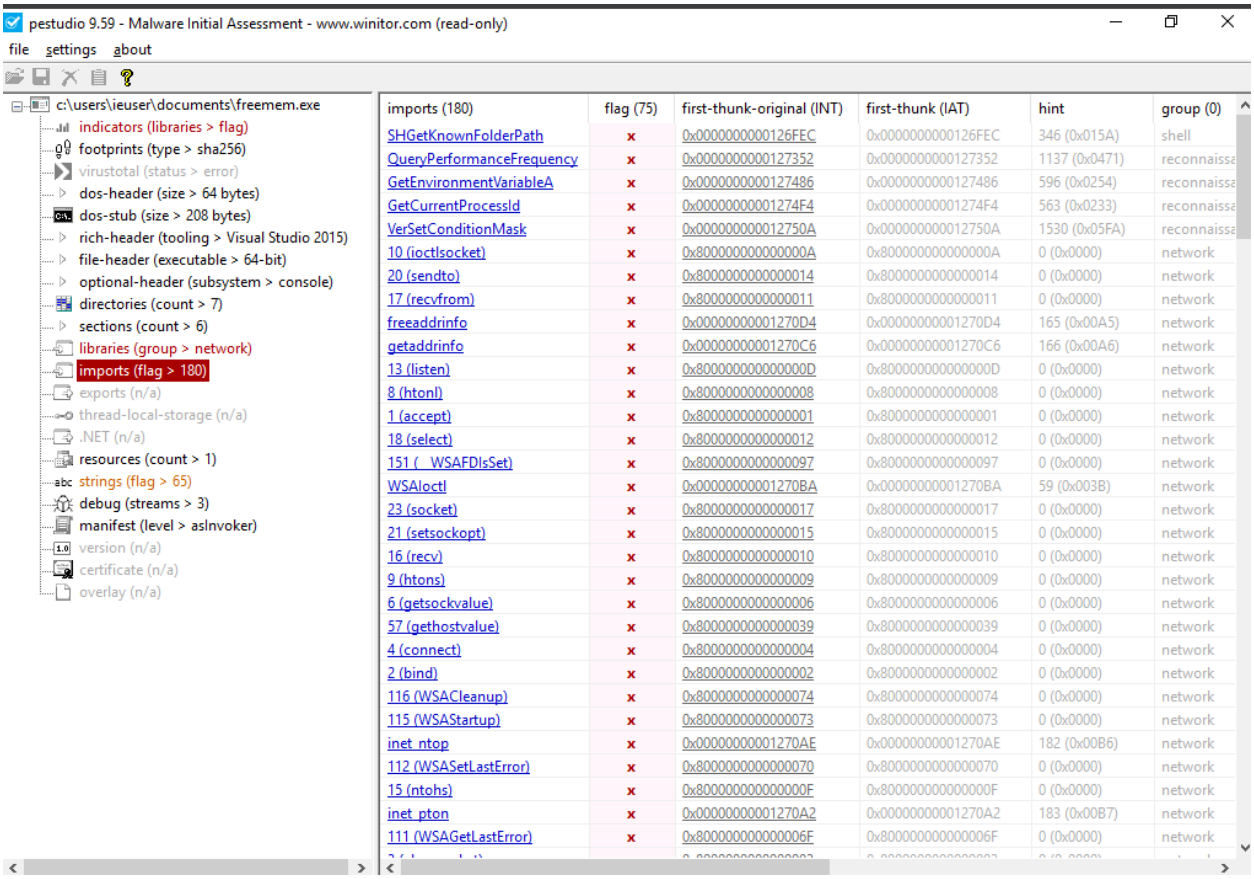


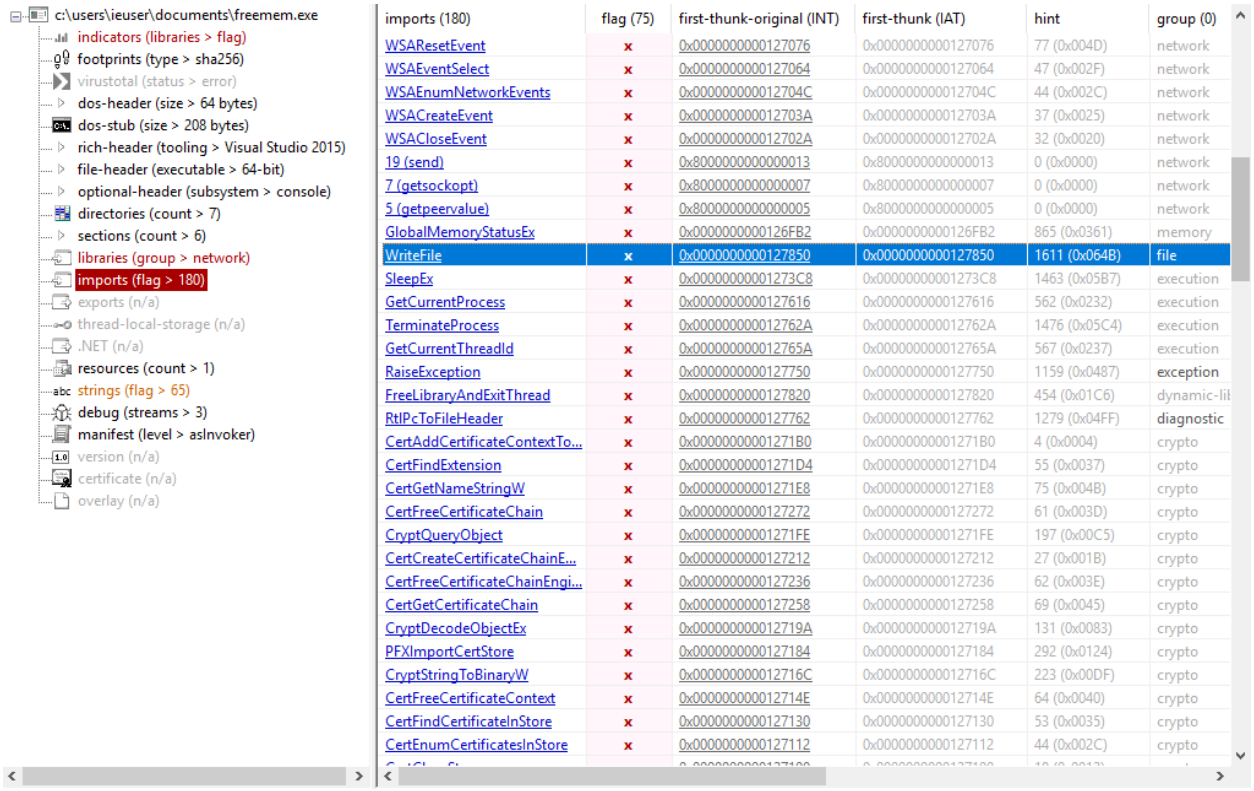
Within the indicators tab we see various flags that direct towards the library section.

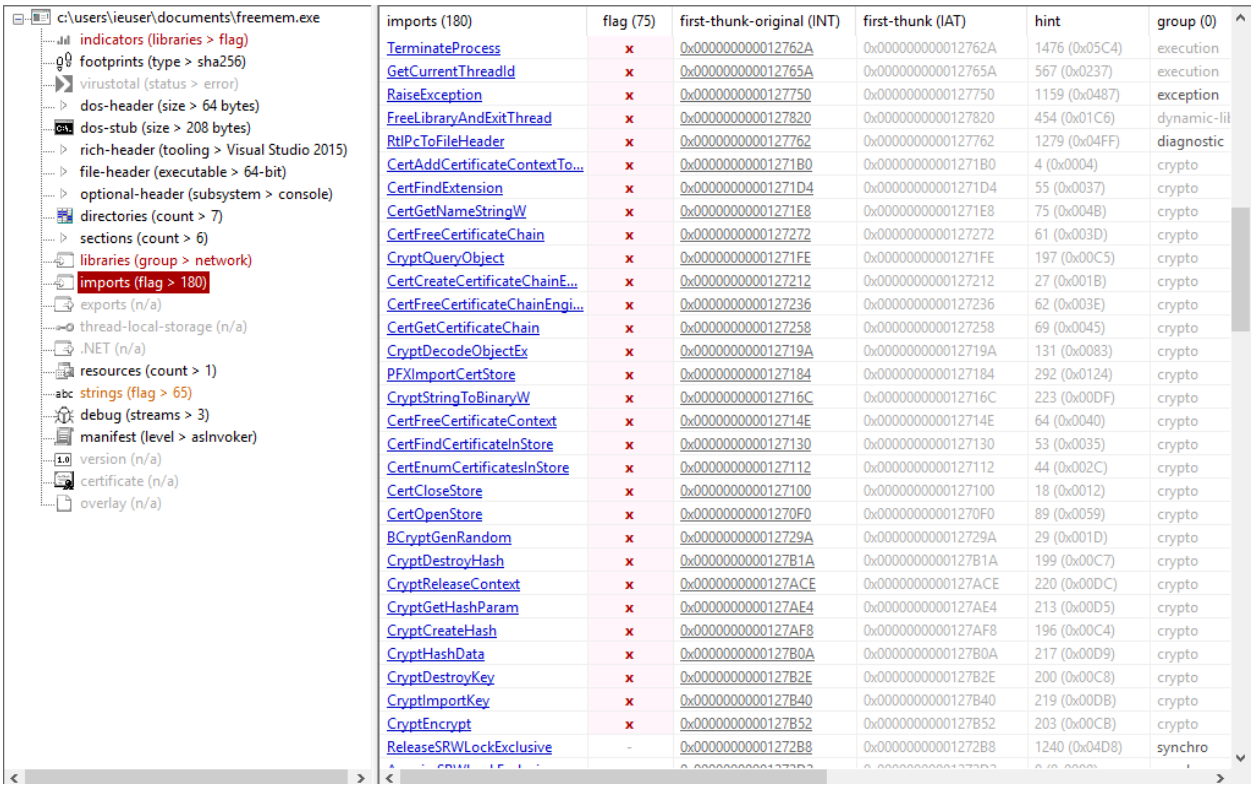


Of the libraries getting used, three are flagged: WS2_32.dll, CRYPT32.dll bcrypt.dll. This indicates that this malware has both network functionality and tries to access Certificate and Cryptographic Messaging functions.




pestudio 9.59 - Malware Initial Assessment - www.winitor.com (read-only)
file settings about
c:\users\ieuser\documents\freemem.exe
indicators (libraries > flag)
footprints (type > sha256)
virustotal (status > error)
dos-header (size > 64 bytes)
dos-stub (size > 208 bytes)
rich-header (tooling > Visual Studio 2015)
file-header (executable > 64-bit)
optional-header (subsystem > console)
directories (count > 7)
sections (count > 6)
libraries (group > network)
imports (flag > 180)
exports (n/a)
thread-local-storage (n/a)
.NET (n/a)
resources (count > 1)
strings (flag > 65)
debug (streams > 3)
manifest (level > asInvoker)
version (n/a)
certificate (n/a)
overlay (n/a)
imports (180) flag (75) first-thunk-original (INT) first-thunk (IAT) hint group (0)
SHGetKnownFolderPath x 0x0000000000126FEC 0x0000000000126FEC 346 (0x015A) shell
QueryPerformanceFrequency x 0x0000000000127352 0x0000000000127352 1137 (0x0471) reconnaissa
GetEnvironmentVariableA x 0x0000000000127486 0x0000000000127486 596 (0x0254) reconnaissa
GetCurrentProcessId x 0x00000000001274F4 0x00000000001274F4 563 (0x0233) reconnaissa
VerSetConditionMask x 0x000000000012750A 0x000000000012750A 1530 (0x05FA) reconnaissa
10 (ioctlsocket) x 0x800000000000000A 0x800000000000000A 0 (0x0000) network
20 (sendto) x 0x8000000000000014 0x8000000000000014 0 (0x0000) network
17 (recvfrom) x 0x8000000000000011 0x8000000000000011 0 (0x0000) network
freeaddrinfo x 0x00000000001270D4 0x00000000001270D4 165 (0x00A5) network
getaddrinfo x 0x00000000001270C6 0x00000000001270C6 166 (0x00A6) network
13 (listen) x 0x800000000000000D 0x800000000000000D 0 (0x0000) network
8 (htonl) x 0x8000000000000008 0x8000000000000008 0 (0x0000) network
1 (accept) x 0x8000000000000001 0x8000000000000001 0 (0x0000) network
18 (select) x 0x8000000000000012 0x8000000000000012 0 (0x0000) network
151 (__WSAFDIsSet) x 0x8000000000000097 0x8000000000000097 0 (0x0000) network
WSAIoctl x 0x00000000001270BA 0x00000000001270BA 59 (0x003B) network
23 (socket) x 0x8000000000000017 0x8000000000000017 0 (0x0000) network
21 (setsockopt) x 0x8000000000000015 0x8000000000000015 0 (0x0000) network
16 (recv) x 0x8000000000000010 0x8000000000000010 0 (0x0000) network
9 (htons) x 0x8000000000000009 0x8000000000000009 0 (0x0000) network
6 (getsockvalue) x 0x8000000000000006 0x8000000000000006 0 (0x0000) network
57 (gethostvalue) x 0x8000000000000039 0x8000000000000039 0 (0x0000) network
4 (connect) x 0x8000000000000004 0x8000000000000004 0 (0x0000) network
2 (bind) x 0x8000000000000002 0x8000000000000002 0 (0x0000) network
116 (WSACleanup) x 0x8000000000000074 0x8000000000000074 0 (0x0000) network
115 (WSAStartup) x 0x8000000000000073 0x8000000000000073 0 (0x0000) network
inet_ntop x 0x00000000001270AE 0x00000000001270AE 182 (0x00B6) network
112 (WSASetLastError) x 0x8000000000000070 0x8000000000000070 0 (0x0000) network
15 (ntohs) x 0x800000000000000F 0x800000000000000F 0 (0x0000) network
inet_pton x 0x00000000001270A2 0x00000000001270A2 183 (0x00B7) network
111 (WSAGetLastError) x 0x800000000000006F 0x800000000000006F 0 (0x0000) network


c:\users\ieuser\documents\freemem.exe
indicators (libraries > flag)
footprints (type > sha256)
virustotal (status > error)
dos-header (size > 64 bytes)
dos-stub (size > 208 bytes)
rich-header (tooling > Visual Studio 2015)
file-header (executable > 64-bit)
optional-header (subsystem > console)
directories (count > 7)
sections (count > 6)
libraries (group > network)
imports (flag > 180)
exports (n/a)
thread-local-storage (n/a)
.NET (n/a)
resources (count > 1)
strings (flag > 65)
debug (streams > 3)
manifest (level > asInvoker)
version (n/a)
certificate (n/a)
overlay (n/a)
imports (180) flag (75) first-thunk-original (INT) first-thunk (IAT) hint group (0)
WSAResetEvent x 0x0000000000127076 0x0000000000127076 77 (0x004D) network
WSAEventSelect x 0x0000000000127064 0x0000000000127064 47 (0x002F) network
WSAEnumNetworkEvents x 0x000000000012704C 0x000000000012704C 44 (0x002C) network
WSACreateEvent x 0x000000000012703A 0x000000000012703A 37 (0x0025) network
WSACloseEvent x 0x000000000012702A 0x000000000012702A 32 (0x0020) network
19 (send) x 0x8000000000000013 0x8000000000000013 0 (0x0000) network
7 (getsockopt) x 0x8000000000000007 0x8000000000000007 0 (0x0000) network
5 (getpeervalue) x 0x8000000000000005 0x8000000000000005 0 (0x0000) network
GlobalMemoryStatusEx x 0x000000000126FB2 0x000000000126FB2 865 (0x0361) memory
WriteFile x 0x0000000000127850 0x0000000000127850 1611 (0x064B) file
SleepEx x 0x00000000001273C8 0x00000000001273C8 1463 (0x05B7) execution
GetCurrentProcess x 0x0000000000127616 0x0000000000127616 562 (0x0232) execution
TerminateProcess x 0x000000000012762A 0x000000000012762A 1476 (0x05C4) execution
GetCurrentThreadId x 0x000000000012765A 0x000000000012765A 567 (0x0237) execution
RaiseException x 0x0000000000127750 0x0000000000127750 1159 (0x0487) exception
FreeLibraryAndExitThread x 0x0000000000127820 0x0000000000127820 454 (0x01C6) dynamic-lib
RtlPcToFileHeader x 0x0000000000127762 0x0000000000127762 1279 (0x04FF) diagnostic
CertAddCertificateContextTo... x 0x00000000001271B0 0x00000000001271B0 4 (0x0004) crypto
CertFindExtension x 0x00000000001271D4 0x00000000001271D4 55 (0x0037) crypto
CertGetNameStringW x 0x00000000001271E8 0x00000000001271E8 75 (0x004B) crypto
CertFreeCertificateChain x 0x0000000000127272 0x0000000000127272 61 (0x003D) crypto
CryptQueryObject x 0x00000000001271FE 0x00000000001271FE 197 (0x00C5) crypto
CertCreateCertificateChainE... x 0x0000000000127212 0x0000000000127212 27 (0x001B) crypto
CertFreeCertificateChainEngi... x 0x0000000000127236 0x0000000000127236 62 (0x003E) crypto
CertGetCertificateChain x 0x0000000000127258 0x0000000000127258 69 (0x0045) crypto
CryptDecodeObjectEx x 0x000000000012719A 0x000000000012719A 131 (0x0083) crypto
PFXImportCertStore x 0x0000000000127184 0x0000000000127184 292 (0x0124) crypto
CryptStringToBinaryW x 0x000000000012716C 0x000000000012716C 223 (0x00DF) crypto
CertFreeCertificateContext x 0x000000000012714E 0x000000000012714E 64 (0x0040) crypto
CertFindCertificateInStore x 0x0000000000127130 0x0000000000127130 53 (0x0035) crypto
CertEnumCertificatesInStore x 0x0000000000127112 0x0000000000127112 44 (0x002C) crypto



Remnux

Step 1: File Identification

- Command: file /media/sf_malwarevm/freemem.exe.malz
```
remnux@remnux:~$ file /media/sf_malwarevm/freemem.exe.malz
/media/sf_malwarevm/freemem.exe.malz: PE32+ executable (console) x86-64, for MS Windows
```
- Output: /media/sf_malwarevm/freemem.exe.malz: PE32+ executable (console) x86-64, for MS Windows
    - We have a PE32+ executable file that is designed for 64-bit Windows systems and as a portable executable format that shows that it may contain malware from running from a console application or a command line or terminal on Windows.

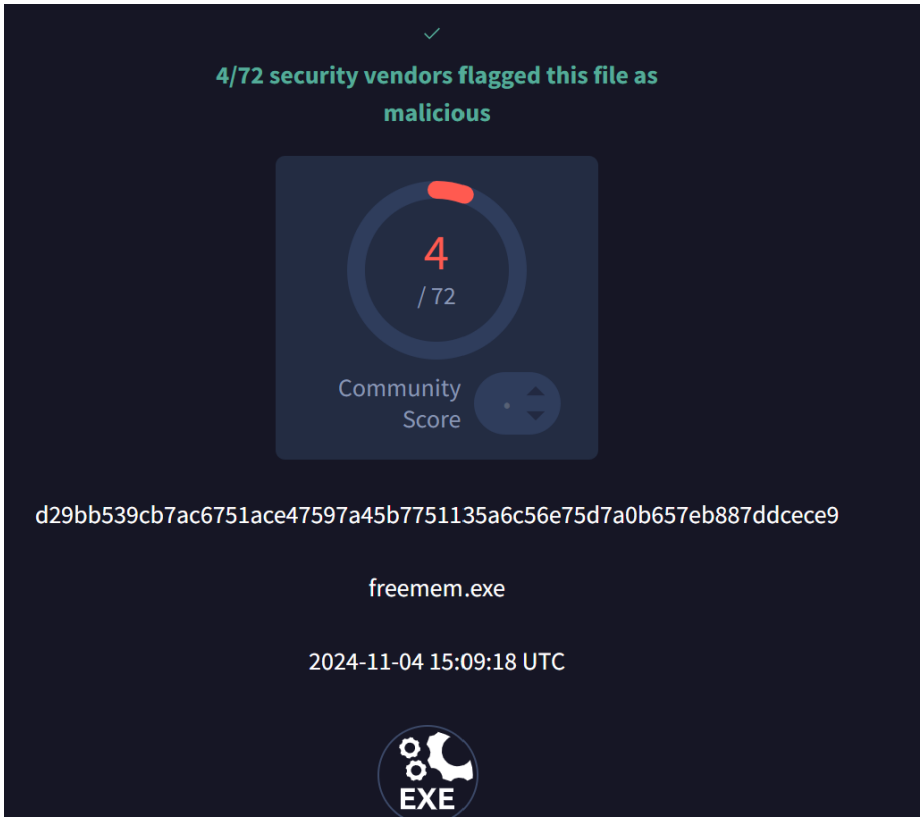
Step 2: Hash File

- Commands:

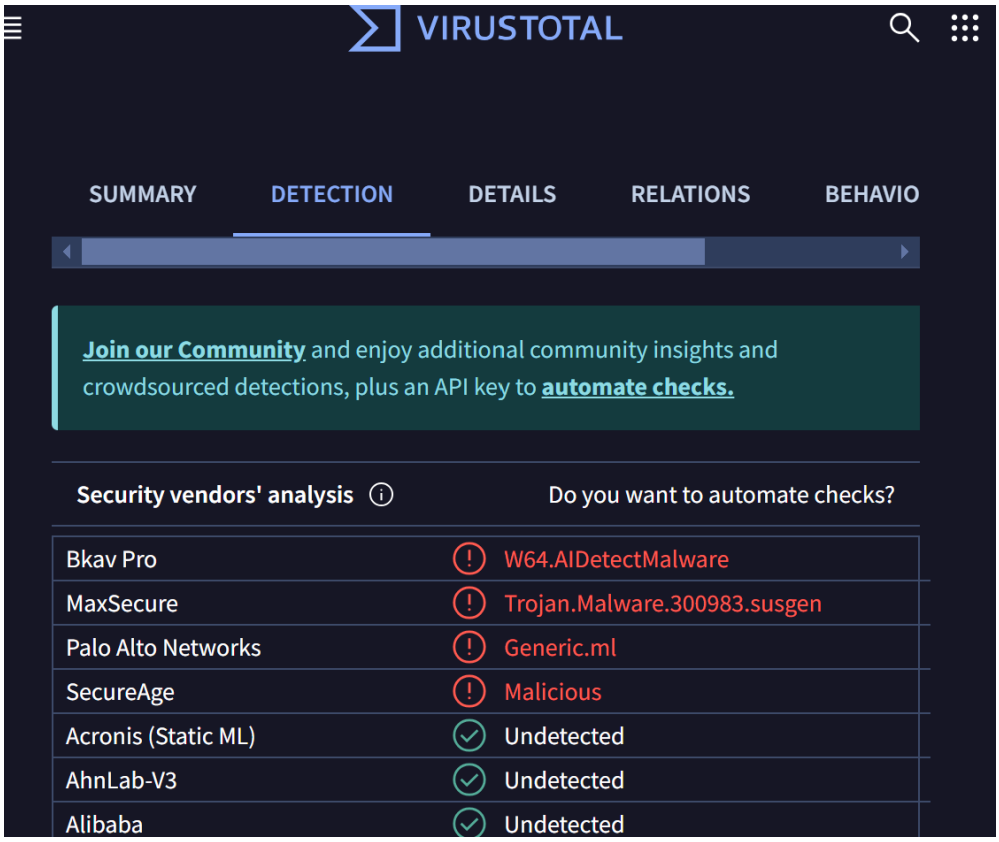
- md5sum /media/sf_malwarevm/freemem.exe.malz
- sha1sum /media/sf_malwarevm/freemem.exe.malz
- sha256sum /media/sf_malwarevm/freemem.exe.malz

```
remnux@remnux:~$ md5sum /media/sf_malwarevm/freemem.exe.malz
b1ec32bf5f6e8f935c932a1059d43829  /media/sf_malwarevm/freemem.exe.malz
remnux@remnux:~$ sha1sum /media/sf_malwarevm/freemem.exe.malz
44fcb6a30cb80b5fe36854bf345d0f8c17efce64  /media/sf_malwarevm/freemem.exe.malz
remnux@remnux:~$ sha256sum /media/sf_malwarevm/freemem.exe.malz
d29bb539cb7ac6751ace47597a45b7751135a6c56e75d7a0b657eb887ddcece9  /media/sf_malwarevm/freemem.exe.malz
remnux@remnux:~$
```

-
- Output:
  - MD5: b1ec32bf5f6e8f935c932a1059d43829
  - SHA-1: 44fcb6a30cb80b5fe36854bf345d0f8c17efce64
  - SHA-256: d29bb539cb7ac6751ace47597a45b7751135a6c56e75d7a0b657Eb887ddcece9
- Analysis:
  - These hashes can be used to match malware from public databases and used for cross-referencing to see if these hashes remain unchanged from running malicious files.
  - VirusTotal Output



-
  - Virtus Total detects and recognizes malicious file from SHA-256 hash



-
  - These vendors are malware detection vendors that detected the files as malicious. So Bkav Pro detected the file W64.AIDetectMalware, MaxSecure flagged Trojan.Malware.300933.susgen, etc..

Basic properties

MD5	b1ec32bf5f6e8f935c932a1059d43829
SHA-1	44fcb6a30cb80b5fe36854bf345d0f8c17efce64
SHA-256	d29bb539cb7ac6751ace47597a45b7751135a6c56e75d7a0b6…
Vhash	016066655d1565555092z13z747z17z1az247z
Authentihas…	d4de90b46121ebf609679b6b2cff60d0f5748a886da95359fd1…
Imphash	a7dfb13698e449be8a1c19644b751237
Rich PE head…	1ac6ebec054f58dba6059539ef7581aa
SSDEEP	24576:CFQhGHkc/v//pNpZA1xN13/bSRpeE9c6nx7BmQD/vAI…
TLSH	T12C458D83F3A540EDD0BBC178C556831BEBB2745513209B…
File type	Win32 EXE executable windows win32 pe peexe
Magic	PE32+ executable (console) x86-64, for MS Windows
TrID	Win64 Executable (generic) (48.7%) | Win16 NE executable …
DetectItEasy…	PE64 | Compiler: Microsoft Visual C/C++ (19.36.33813) [LTC…
Magika	PEBIN
File size	1.20 MB (1254912 bytes)

History

Creation Tim…	2024-10-16 23:07:14 UTC
First Submiss…	2024-10-30 09:04:58 UTC
Last Submiss…	2024-11-04 21:45:39 UTC
Last Analysis…	2024-11-04 15:09:18 UTC

-
  - We find the same hash outputs from Remnux commands and the file type as well.

Imports

- \+ KERNEL32.dll
- \+ SHELL32.dll
- \+ ole32.dll
- \+ WS2_32.dll
- \+ CRYPT32.dll
- \+ bcrypt.dll
- \+ ADVAPI32.dll

-

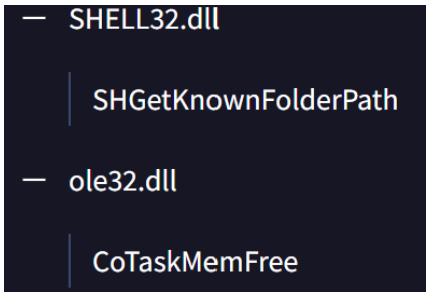
- Fully Listed Import from KERNEL32.dll
    - AcquireSRWLockExclusive
    - CloseHandle
    - CompareStringW
    - CreateDirectoryW
    - CreateFileW
    - CreateThread
    - DeleteCriticalSection
    - DeleteFileW
    - EncodePointer
    - EnterCriticalSection
    - ExitProcess
    - ExitThread
    - FileTimeToSystemTime
    - FindClose
    - FindFirstFileExW
    - FindNextFileW
    - FlsAlloc
    - FlsFree
    - FlsGetValue
    - FlsSetValue
    - FlushFileBuffers
    - FormatMessageW
    - FreeEnvironmentStringsW
    - FreeLibrary
    - FreeLibraryAndExitThread
    - GetACP
    - GetCommandLineA
    - GetCommandLineW
    - GetConsoleMode
    - GetConsoleOutputCP
    - GetCPInfo
    - GetCurrentDirectoryW
    - GetCurrentProcess

- GetCurrentProcessId
- GetCurrentThreadId
- GetDateFormatW
- GetDriveTypeW
- GetEnvironmentStringsW
- GetEnvironmentVariableA
- GetFileAttributesW
- GetFileInformationByHandle
- GetFileSizeEx
- GetFileType
- GetFullPathNameW
- GetLastError
- GetModuleFileNameW
- GetModuleHandleA
- GetModuleHandleExW
- GetModuleHandleW
- GetOEMCP
- GetProcAddress
- GetProcessHeap
- GetStartupInfoW
- GetStdHandle
- GetStringTypeW
- GetSystemDirectoryW
- GetSystemTimeAsFileTime
- GetTempFileNameW
- GetTempPathW
- GetTickCount
- GetTimeFormatW
- GetTimeZoneInformation
- GlobalMemoryStatusEx
- HeapAlloc
- HeapFree
- HeapReAlloc
- HeapSize
- InitializeCriticalSectionAndSpinCount
- InitializeCriticalSectionEx
- InitializeSListHead
- IsDebuggerPresent
- IsProcessorFeaturePresent

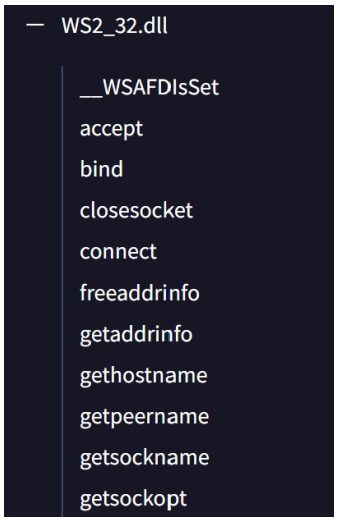
- IsValidCodePage
- LCMapStringW
- LeaveCriticalSection
- LoadLibraryExW
- LoadLibraryW
- MoveFileExW
- MultiByteToWideChar
- PeekNamedPipe
- QueryPerformanceCounter
- QueryPerformanceFrequency
- RaiseException
- ReadConsoleW
- ReadFile
- ReleaseSRWLockExclusive
- RtlCaptureContext
- RtlLookupFunctionEntry
- RtlPcToFileHeader
- RtlUnwind
- RtlUnwindEx
- RtlVirtualUnwind
- SetEndOfFile
- SetEnvironmentVariableW
- SetFilePointerEx
- SetLastError
- SetStdHandle
- SetUnhandledExceptionFilter
- Sleep
- SleepEx
- SystemTimeToTzSpecificLocalTime
- TerminateProcess
- TlsAlloc
- TlsFree
- TlsGetValue
- TlsSetValue
- UnhandledExceptionFilter
- VerifyVersionInfoW
- VerSetConditionMask
- WaitForMultipleObjects
- WaitForSingleObject

- WaitForSingleObjectEx
- WakeAllConditionVariable
- WideCharToMultiByte
- WriteConsoleW
- WriteFile

    - Findings:
        - Everything highlighted are notably important API calls from the malware function that may interact with file manipulation, system interactions, process management, and error handling which are critical for the intention of the malware and its tactics.

- 

    - Findings:
        - These are just file recollection pointers and a memory allocator to ensure memory is released to the system.

- 

    - Fully Listed Imports from WS2_32.dll

- __WSAFDIsSet
- accept
- bind
- closesocket

- connect
- freeaddrinfo
- getaddrinfo
- gethostname
- getpeername
- getsockname
- getsockopt
- htonl
- htons
- inet_ntop
- inet_pton
- ioctlsocket
- listen
- ntohs
- recv
- recvfrom
- select
- send
- sendto
- setsockopt
- socket
- WSACleanup
- WSACloseEvent
- WSACreateEvent
- WSAEnumNetworkEvents
- WSAEventSelect
- WSAGetLastError
- WSAIoctl
- WSAResetEvent
- WSASetLastError
- WSAStartup
- WSAWaitForMultipleEvents

    - Findings:
        - **socket**, **connect**, **send**, **recv**, **accept**, **bind**, and **listen**: These functions are used for establishing and managing network connections, and sockets create network endpoints. Connect initiates a connection to a remote server, send and recv handle data

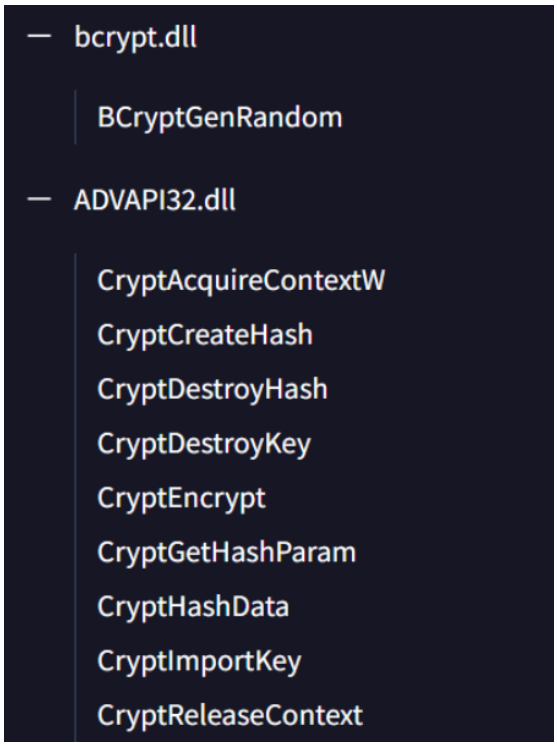
transmission and reception, and accept, bind, and listen are used for server-like operations, which may indicate that the malware is waiting to receive commands or act as a server.
    - **getaddrinfo** and **freeaddrinfo**: These resolve domain names into IP addresses, which may indicate the malware's intention to communicate with command-and-control (C2) servers or other network endpoints by name rather than IP.

-
    - Findings:
        - **CertAddCertificateContextToStore**: This function adds a certificate to a specified store. Malware might use this to install certificates in the system to bypass security measures, establish trust, or facilitate man-in-the-middle attacks.
        - **CertCloseStore** and **CertOpenStore**: These functions manage certificate stores which is the location of where the certificates are, and malware

may open, manipulate, or close stores to handle certificates needed for secure communications or to bypass security.

- **CertCreateCertificateChainEngine**, **CertGetCertificateChain**, and **CertFreeCertificateChainEngine**: These are used to validate certificate chains and ensure the authenticity of certificates. Malware may use this chain validation to establish secure connections while appearing legitimate.
- **CertEnumCertificatesInStore** and **CertFindCertificateInStore**: These functions search for certificates within a store. Malware may find specific ones for use in cryptographic operations or to find security software certificates.
- **CertFindExtension**: This function locates specific extensions within a certificate. Malware might use this to gather information from certificates or customize their use based on specific extension data.
- **CertFreeCertificateChain** and **CertFreeCertificateContext**: These functions release memory for certificate chains and certificate contexts. This is part of standard memory management when handling certificates.
- **CertGetNameStringW**: This function retrieves a name string from a certificate. Malware may use this to extract and manipulate identifying information from certificates for reconnaissance or as part of impersonation tactics.
- **CryptDecodeObjectEx** and **CryptQueryObject**: These functions decode encoded data structures and query certificate data. They are useful for parsing certificate and encryption data so the malware might alter encrypted communications.
- **CryptStringToBinaryW**: Converts a string to a binary data format. Malware might use this to decode or handle encoded data strings.
- **PFXImportCertStore**: Imports a certificate store from a PFX file (Personal Information Exchange format). This could allow malware to install a set of

certificates. Used for establishing encrypted channels, impersonating trusted sources, or bypassing security mechanisms.

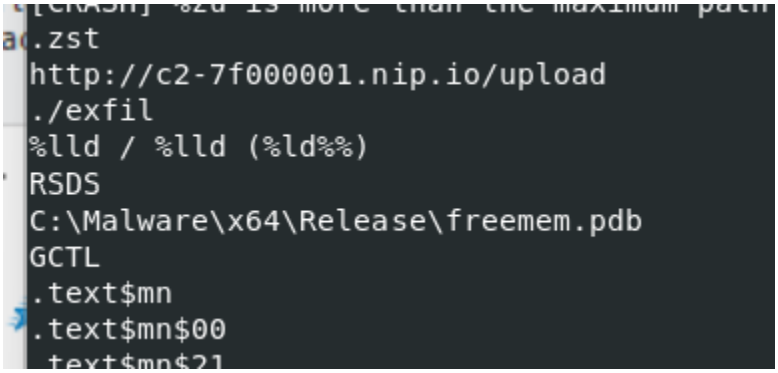
- 

    - **BCryptGenRandom**: This function generates cryptographically secure random numbers. Malware can create unique identifiers, encryption keys, or obfuscate data to make detection more difficult.
    - **CryptCreateHash**: Creates a hash object, which can then be used to hash data. Might be used to obfuscate or encrypt data in a way that prevents straightforward analysis.
    - **CryptDestroyHash** and **CryptDestroyKey**: These functions clean up and free resources for hash and key objects. Malware may use these to remove traces of its cryptographic operations and minimize its footprint after operations are completed.
    - **CryptEncrypt**: Encrypts data using a cryptographic key. If malware imports this function it may be encrypting files, sensitive data, or communication channels to avoid detection.
    - **CryptGetHashParam**: Retrieves parameters of a hash object, such as the hash value itself. Malware might use this to get the result of hashed data, which could then be used

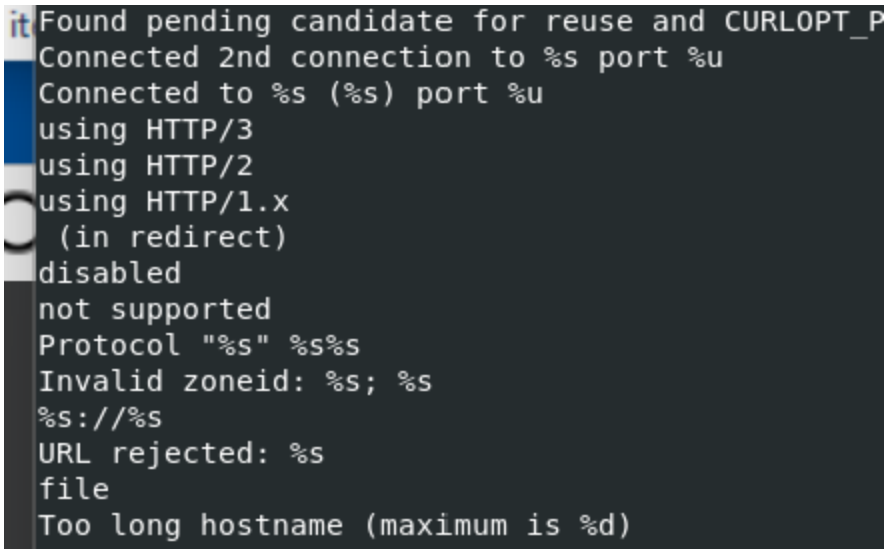
for data integrity checks, password hashing, or digital signatures within its own operations.

- **CryptHashData**: Adds data to a hash object, which will later be hashed. Malware may use this to create hash values for data chunks, which can then be used to verify data integrity or check for tampering.
- **CryptImportKey**: Imports a cryptographic key, enabling it to be used for encryption, decryption, or hashing. Malware could use this to import keys for decrypting payloads, setting up secure communication, or managing cryptographic routines.
- **CryptReleaseContext**: Releases the handle acquired by **CryptAcquireContextW**. This is part of standard cryptographic resource management, often used by malware to clean up after cryptographic operations.

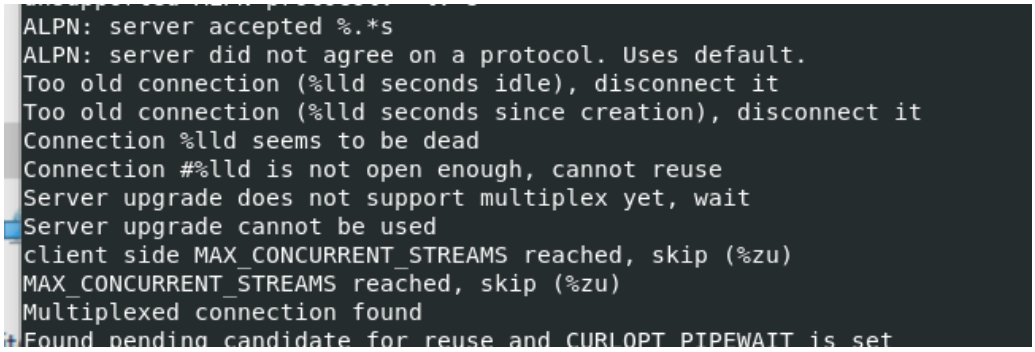
Step 3: Strings Analysis

- Command Used: strings /media/sf_malwarevm/freemem.exe.malz > extracted_strings.txt
- Output File: extracted_strings.txt
- Findings:
  - 

    - **http://c2-7f000001.nip.io/upload**: This string resembles a URL pointing to a Command and Control (C2) server, with an endpoint for uploading data (/upload). This is important because it indicates potential exfiltration activity. The nip.io domain is commonly used for local testing and dynamic DNS, which could allow the malware to exfiltrate data or receive commands from a remote server dynamically. This string is critical as it may reveal the malware's communication methods and infrastructure.



-
    - **Using HTTP/1/2/3**: These indicate that the malware is capable of using different HTTP protocol versions for its communication. This flexibility in protocol support is notable, as it could allow the malware to adapt to various network environments and leverage features specific to each version.



-
    - Patterns such as closing connection #%lld and Too old connection (%lld seconds idle), disconnect it: These entries suggest that the malware is implementing mechanisms to manage its connections actively. By closing idle connections, it can minimize its footprint and avoid detection by network monitoring systems.

Building on these insights, further examination of the extracted strings reveals additional indicators of the malware's functionality and intent. By focusing on network functions, encryption APIs, HTTP strings, and error-handling routines, we can infer that this malware may be designed for remote command and control, secure communication, and evasion of detection methods. The analysis below expands on these capabilities, highlighting how each of these elements contributes to its potential for covert operations

1. **System- and Process-Related API Calls**:

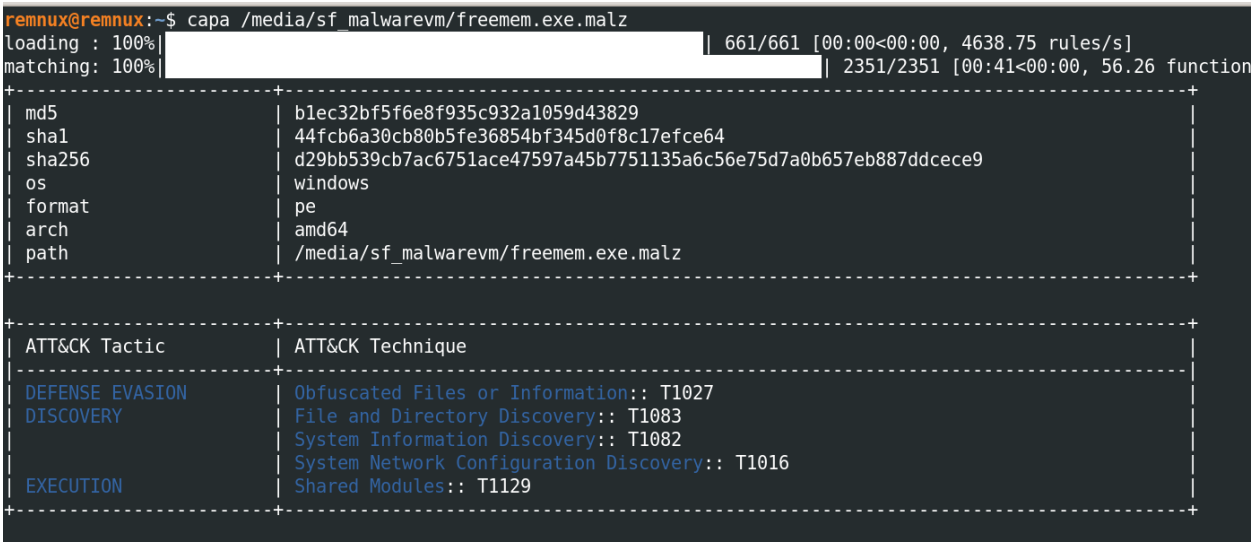
    - GetCurrentProcessId, GetCurrentProcessorNumber, GetCurrentThreadId, GetModuleHandleW, LoadLibraryExW, CreateThread, CreateFileW, OpenProcess, ReadFile, WriteFile, DeleteFileW, FlushFileBuffers, HeapAlloc, HeapFree, and TerminateProcess.
    - These are common in malware for obtaining system information, loading additional libraries, managing files, and creating threads for multitasking and process manipulation.
2. **Cryptographic API Calls**:
    - CryptAcquireContextW, CryptCreateHash, CryptDestroyHash, CryptHashData, CryptImportKey, CryptQueryObject, and BCryptGenRandom.
    - These functions are often used for encryption and decryption, hashing, or generating random numbers, which may hint at data protection methods for storing information securely or encrypting communication with command and control servers.
3. **Network and Socket Functions**:
    - getaddrinfo, gethostname, connect, send, recvfrom, setsockopt, closesocket, and inet_ntop.
    - These are typical in malware that requires network functionality, especially if the malware communicates with external servers to exfiltrate data or receive commands.
4. **HTTP-Related Terms and Headers**:
    - User-Agent, HTTP/1.1, Content-Type, Host, Proxy-Connection, and WWW-Authenticate.
    - These could indicate HTTP-based communication, often associated with web request forging, proxy tunneling, or interacting with web-based command and control servers.
5. **DNS and IP Address Management**:
    - Terms like localhost, .onion, dns_cache, 127.0.0.1, and Resolve address.
    - These could indicate the malware's capability to manipulate or query DNS settings, and .onion suggests potential access to hidden services on the Tor network, possibly for stealthy C2 communication.
6. **Certificate and TLS-Related API Calls**:
    - CertAddCertificateContextToStore, CertCloseStore, CertCreateCertificateChainEngine, CertFindCertificateInStore, and CertOpenStore.
    - These functions suggest the malware may manage certificates, perhaps to establish secure connections, perform man-in-the-middle attacks, or handle SSL/TLS sessions.
7. **Registry and System Information Functions**:

    - GetSystemTimeAsFileTime, GetTickCount64, RtlGetVersion, GetFileType, GetEnvironmentStringsW.
    - These may be used to gather system information, interact with the Windows Registry, or fetch environmental variables.
8. **HSTS, HTTP/2, and Alt-Svc Headers**:
    - Strings like Strict-Transport-Security, alt-svc, and HSTS.
    - HSTS (HTTP Strict Transport Security) and alt-svc (Alternate Services) headers could suggest that the malware may be aware of HTTPS requirements and might attempt to bypass certain security controls in HTTP connections.
9. **Suspicious Placeholder or Obfuscated Strings**:
    - Strings like @A_A^A]A_^][or HA_A^A]A_^[] may represent placeholder or obfuscated patterns, often used in malware to avoid detection by simple signature-based antivirus tools.
10. **Chunked Data Transfer, Proxy Usage, and HTTP Errors**:
    - Strings such as Chunky upload, proxy, and Transfer closed with %lld bytes remaining to read.
    - These indicate that the malware may be capable of uploading data in chunks, use proxies to disguise its network traffic, and handle network transmission errors gracefully, which are features found in advanced malware designed for resilience and stealth.
11. **Error and Warning Messages**:
    - WARNING: failed to open cookie file, Failed to resolve, Re-using existing connection, and Got HTTP failure 417 while waiting for a 100.
    - Error strings may reveal resilience features or logging for debugging, showing the malware can attempt retries or recover from common network errors.
12. **".onion" and Localhost References**:
    - .onion, localhost, and 127.0.0.1.
    - The presence of .onion addresses implies the malware may use the Tor network for anonymous communication, which is typical for stealthy command and control (C2) channels.

Overall, the network functions, encryption APIs, HTTP strings, and error handling indicate that this malware is likely designed for remote control, data exfiltration, and evasion of network-based detection methods.

Step 4: CAPA Analysis from Remnux Tools

- Command: capa /media/sf_malwarevm/freemem.exe.malz
- Output:
-



  - Findings:
    1. **Obfuscated Files or Information (T1027)**
       - The malware employs obfuscation techniques to evade detection by security tools, making it harder to identify its true functionality.
    2. **File and Directory Discovery (T1083)**
       - This technique indicates the malware searches for files and directories on the host system, potentially to identify valuable data or system configurations.
    3. **System Information Discovery (T1082)**
       - The malware gathers information about the host system's details, such as OS version and hardware specifications, which can aid in tailoring its operations to the specific environment.
    4. **System Network Configuration Discovery (T1016)**
       - By probing the network configuration, the malware gains insight into the system's network settings, which can be used

for lateral movement or establishing secure communication channels.

5. **Shared Modules (T1129)**
    - Utilizing shared libraries and modules may allow the malware to execute code within legitimate processes, further enhancing its stealth and avoiding detection by blending in with regular system activity.

MBC Objective	MBC Behavior
ANTI-BEHAVIORAL ANALYSIS	Debugger Detection::Software Breakpoints [B0001.025]
COMMAND AND CONTROL	C2 Communication::Receive Data [B0030.002]
	C2 Communication::Send Data [B0030.001]
COMMUNICATION	DNS Communication::Resolve [C0011.001]
	HTTP Communication::Read Header [C0002.014]
	Interprocess Communication::Read Pipe [C0003.003]
	Socket Communication::Connect Socket [C0001.004]
	Socket Communication::Create TCP Socket [C0001.011]
	Socket Communication::Get Socket Status [C0001.012]
	Socket Communication::Initialize Winsock Library [C0001.009]
	Socket Communication::Receive Data [C0001.006]
	Socket Communication::Send Data [C0001.007]
	Socket Communication::Set Socket Config [C0001.001]
	Socket Communication::Start TCP Server [C0001.005]
	Socket Communication::TCP Client [C0001.008]
CRYPTOGRAPHY	Cryptographic Hash:: [C0029]
	Encrypt Data::RC4 [C0027.009]
	Encryption Key:: [C0028]
	Generate Pseudo-random Sequence::RC4 PRGA [C0021.004]
	Generate Pseudo-random Sequence::Use API [C0021.003]
	Hashed Message Authentication Code:: [C0061]
DATA	Check String:: [C0019]
	Checksum::Adler [C0032.005]
	Checksum::Luhn [C0032.002]
	Compression Library:: [C0060]
	Encode Data::Base64 [C0026.001]
	Encode Data::XOR [C0026.002]
DEFENSE EVASION	Obfuscated Files or Information::Encoding-Standard Algorithm [E1027.m02]
DISCOVERY	Code Discovery::Enumerate PE Sections [B0046.001]
FILE SYSTEM	Create Directory:: [C0046]
	Delete File:: [C0047]
	Get File Attributes:: [C0049]
	Move File:: [C0063]
	Read File:: [C0051]
	Writes File:: [C0052]
OPERATING SYSTEM	Environment Variable::Set Variable [C0034.001]
PROCESS	Create Thread:: [C0038]
	Terminate Process:: [C0018]

-
    - Findings:
        - **Anti-Behavioral Analysis:** The malware incorporates techniques like debugger detection (Software Breakpoints [B0001.025]) to identify and evade analysis in sandboxed or debugging environments.
        - **Command and Control (C2):** C2 communication techniques, such as data sending and receiving ([B0030.001, B0030.002]), suggest the malware can establish remote communication with an attacker's server to receive instructions or exfiltrate data.

- **Communication:** The sample uses various socket communication methods (e.g., Create TCP Socket [C0001.011], Connect Socket [C0001.004]) and DNS and HTTP communications ([C0011.001, C0002.014]) for data transfer and control, indicating it has multiple network channels to maintain persistent connections.
- **Cryptography:** The malware includes cryptographic capabilities, such as RC4 encryption (Encrypt Data :: RC4 [C0027.009]) and cryptographic hashing (Cryptographic Hash [C0029]), to secure data in transit or hide malicious content.
- **Data Manipulation:** Encoding techniques, including Base64 ([C0026.001]) and XOR ([C0026.002]), enable the malware to obscure its data, possibly to evade detection or secure exfiltrated information.
- **Defense Evasion and Discovery:** Techniques for evasion include obfuscation (Encoding-Standard Algorithm [E1027.m02]) and discovery of code sections (Enumerate PE Sections [B0046.001]), which help it blend in with normal operations and evade security tools.
- **File System and Process Management:** The malware performs various file system operations, such as creating, deleting, moving, and writing files ([C0046, C0047, C0063]), which are typical in data handling or persistence. Process management functions like Create Thread ([C0038]) and Terminate Process ([C0018]) further indicate it can manipulate system processes to control execution flow.

CAPABILITY	NAMESPACE
check for software breakpoints (2 matches)	anti-analysis/anti-debugging/debugger-detection
receive data (4 matches)	communication
send data (11 matches)	communication
check HTTP status code (3 matches)	communication/http/client
read pipe	communication/named-pipe/read
get socket information (5 matches)	communication/socket
get socket status	communication/socket
initialize Winsock library	communication/socket
set socket configuration (4 matches)	communication/socket
act as TCP client	communication/tcp/client
start TCP server	communication/tcp/serve
compute adler32 checksum	data-manipulation/checksum/adler32
validate payment card number using luhn algorithm	data-manipulation/checksum/luhn
encode data using Base64	data-manipulation/encoding/base64
reference Base64 string	data-manipulation/encoding/base64
encode data using XOR (118 matches)	data-manipulation/encoding/xor
create new key via CryptAcquireContext	data-manipulation/encryption
encrypt data using RC4 PRGA	data-manipulation/encryption/rc4
hash data via WinCrypt (2 matches)	data-manipulation/hashing
initialize hashing via WinCrypt (2 matches)	data-manipulation/hashing
hash data using MD4	data-manipulation/hashing/md4
authenticate HMAC	data-manipulation/hmac
generate random numbers via WinAPI (2 matches)	data-manipulation/prng
contains PDB path	executable/pe/pdb
contain a resource (.rsrc) section	executable/pe/section/rsrc
query environment variable (3 matches)	host-interaction/environment-variable
set environment variable (3 matches)	host-interaction/environment-variable
get common file path (3 matches)	host-interaction/file-system
create directory	host-interaction/file-system/create
delete file	host-interaction/file-system/delete
enumerate files via kernel32 functions	host-interaction/file-system/files/list
get file attributes	host-interaction/file-system/meta
get file size (2 matches)	host-interaction/file-system/meta
move file	host-interaction/file-system/move
read file on Windows (10 matches)	host-interaction/file-system/read
write file on Windows (2 matches)	host-interaction/file-system/write
get memory capacity	host-interaction/hardware/memory
get disk information (2 matches)	host-interaction/hardware/storage
get local IPv4 addresses (5 matches)	host-interaction/network/address
resolve DNS	host-interaction/network/dns/resolve
get hostname	host-interaction/os/hostname
check OS version	host-interaction/os/version
terminate process	host-interaction/process/terminate
create thread	host-interaction/thread/create
link function at runtime on Windows (5 matches)	linking/runtime-linking
linked against libcurl	linking/static/libcurl
linked against ZLIB	linking/static/zlib
enumerate PE sections	load-code/pe
parse PE header (4 matches)	load-code/pe

-
    - Findings:
        - The malware exhibits various capabilities indicative of data exfiltration, remote control, and anti-analysis techniques. It communicates via TCP, HTTP, and DNS, and acts as both a client and server. This suggests potential command-and-control (C2) interactions and we also find that it manipulates files (create, delete, move, read, write) which reveals its intent to exfiltrate data or maintain persistence. The malware employs encryption (RC4), hashing (MD4, HMAC), and encoding (Base64, XOR) for obfuscation and secure communication. It also utilizes anti-debugging techniques to avoid detection and queries environment variables to adjust its behavior. Additionally, the malware interacts with system resources (memory, disk, processes) and leverages

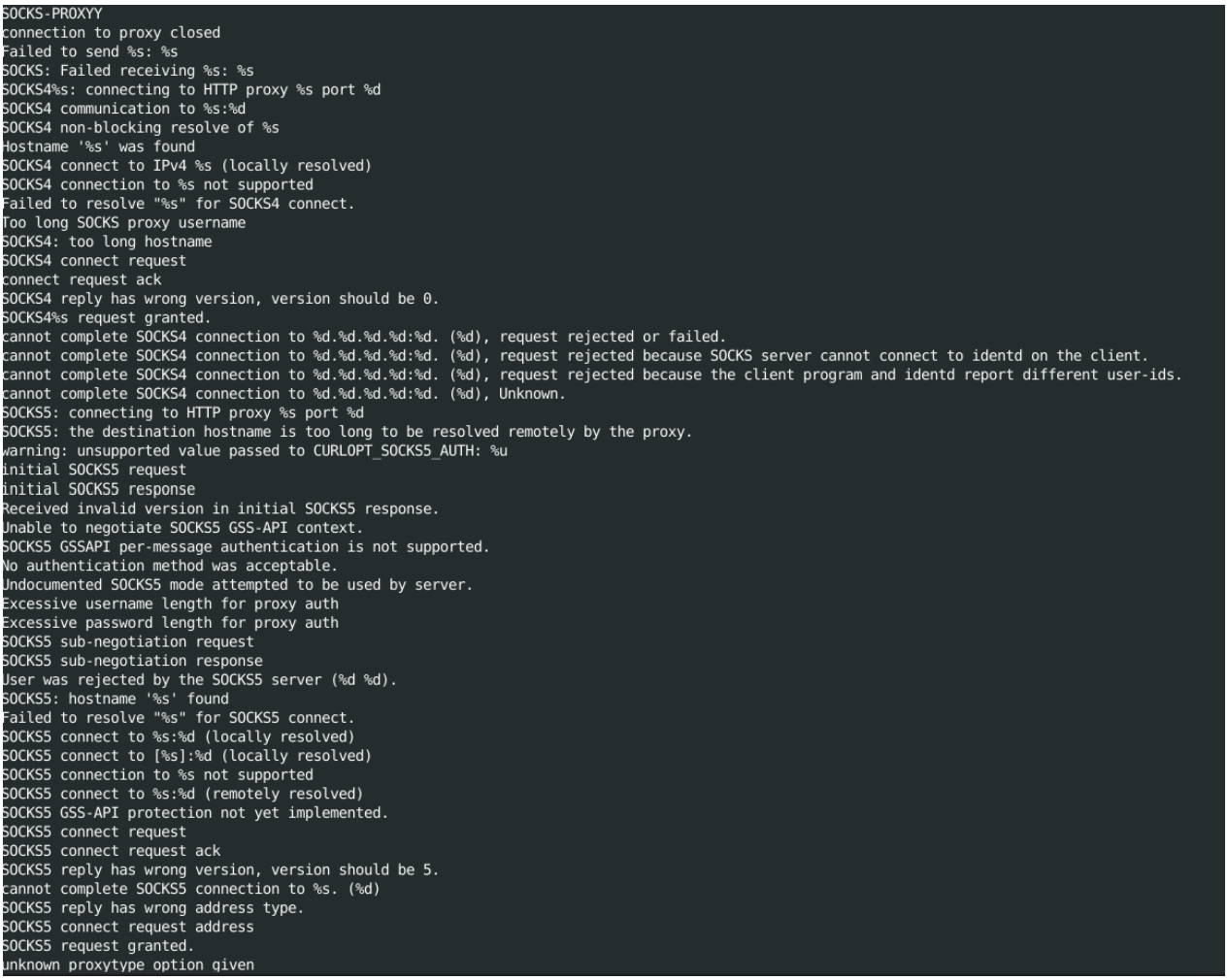
libraries like libcurl for HTTP communication and Zlib for compression. These behaviors point to a sophisticated, stealthy malware aiming for remote control, data theft, and evasion of detection

Step 5: FLOSS

```
CONNECT: fwd auth header '%s'
Ignoring Content-Length in CONNECT %03d response
Ignoring Transfer-Encoding in CONNECT %03d response
CONNECT responded chunked
HTTP/1.
Proxy CONNECT connection closed
Proxy CONNECT aborted
chunk reading DONE
CONNECT response too large
Ignore %lld bytes of response-body
Ignore chunked response-body
CONNECT: no content-length or chunked
Proxy CONNECT aborted due to timeout
CONNECT start
CONNECT send
CONNECT receive
CONNECT response
CONNECT need to close+open
Connect me again please
CONNECT tunnel failed, response %d
CONNECT tunnel established, response %d
HAPROXY
PROXY UNKNOWN
TCP6
TCP4
PROXY %s %s %s %i %i
HTTPS-CONNECT
connect+handshake %s: %dms, 1st data: %dms
```

-
    - Findings:
        - **SOCKS-PROXY**: Indicates that the malware is capable of routing traffic through a SOCKS proxy to improve its ability to obscure its true origin.
        - **HAPROXY**: A popular load balancer and proxy service. May indicate the malware is using advanced networking.
        - **CONNECT phase completed**: Indicates that the initial connection establishment through the proxy was completed.

- **Failed sending CONNECT to proxy**: Suggests that the malware attempted to establish a connection via a proxy but encountered an error.
- **CONNECT tunnel established/failed**: Further indicates the success or failure of proxy tunneling attempts, which can imply the malware's reliance on proxy services for communication, potentially for evasion tactics.

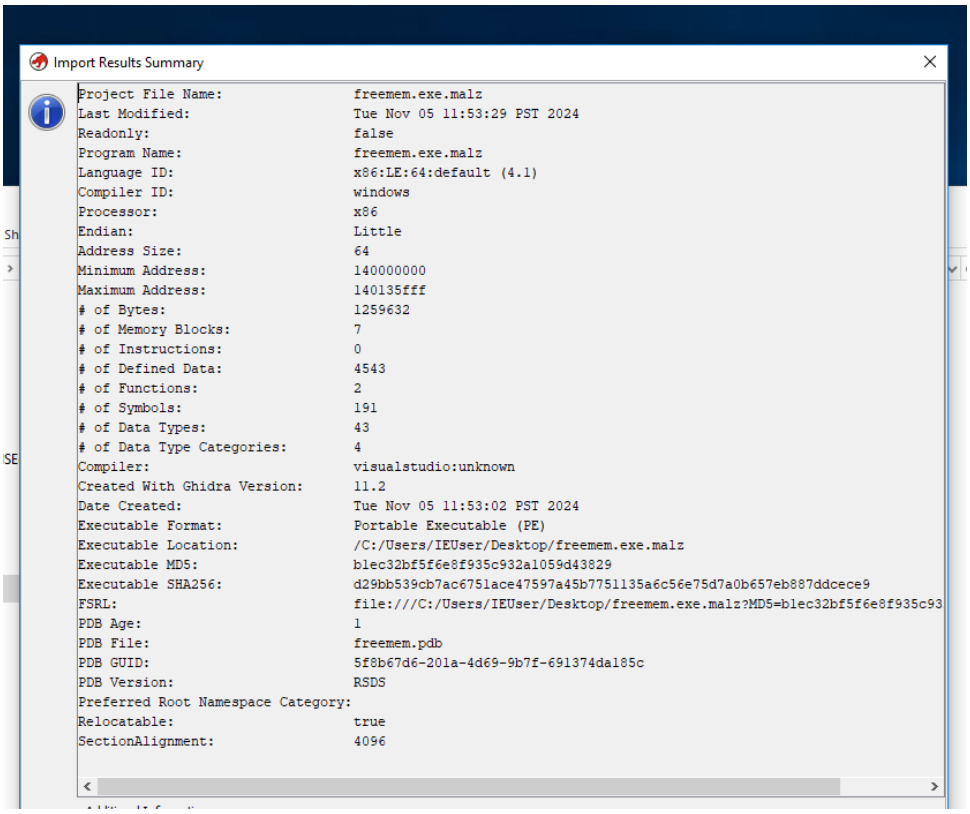


- Findings:
  - **SOCKS5** is a protocol that allows clients to connect to servers through a proxy server. It can handle various types of traffic, including TCP and UDP.

```
        /:#?!@{}[]\$'"^`*<>=;,+&()%
/../
/..?
127.0.0.1/
/\:*?"<>|
ftp.
dict.
ldap.
ldap
imap.
smtp.
pop3.
file://%s%s%s
%.*s%%25%s]
xn--
%s://
%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s
macdef
machine
login
password
HOME
USERPROFILE
%s%s.netrc
%s%s_netrc
hds-collect
header_collect pushed(type=%x, len=%zu) -> %d
HTTP-PROXY
%s%s%s:%d
connect
installing subfilter for HTTP/1.1
CONNECT tunnel: HTTP/1.%d negotiated
CONNECT tunnel: unsupported ALPN(%d) negotiated
H1-PROXY
%s cannot be done over CONNECT
allocate connect buffer
new tunnel state 'init'
new tunnel state 'connect'
new tunnel state 'receive'
new tunnel state 'response'
new tunnel state 'established'
CONNECT phase completed
new tunnel state 'failed'
Establish HTTP proxy tunnel to %s
Failed sending CONNECT to proxy
CONNECT: fwd auth header '%s'
Ignoring Content-Length in CONNECT %03d response
```

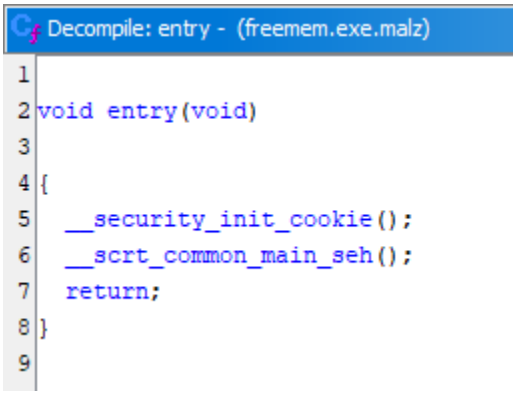
- Findings:
    - **127.0.0.1/**: The use of the localhost IP address could suggest attempts to connect to local services or exploit local network vulnerabilities.
    - **FTP, LDAP, IMAP, SMTP, POP3**: The presence of these protocols indicates that the malware might attempt to use various communication methods for data exfiltration or c2 communications.
    - **file://%s%s%s**: Indicates potential for accessing local files or URLs, which could be part of a file manipulation or exfiltration strategy and we also see the %s pattern from data obfuscation.
    - **new tunnel state 'init' / 'connect' / 'receive' / 'response'**: These state transitions suggest that the malware is designed to establish and manage multiple connections through proxies.

Advanced Static Analysis (Reverse-Engineer) w/ Ghidra



1. Surface-Level Functionality

- The malware is designed to handle both network communication and memory management. Its apparent behavior includes network monitoring, managing socket connections, and error handling. At the entry point (entry), the program initializes security cookies with __security_init_cookie, which are used to protect against certain classes of exploits, followed by __scrt_common_main_seh, the main function handling structured exception handling (SEH).



- The primary network operations include socket management, monitoring events, and transferring data through network sockets using functions like **WSAEventSelect** and **WSAEnumNetworkEvents**. These are typical of a network utility, which might mislead analysts into perceiving the program as a benign tool for network diagnostics or resource monitoring.

2. Additional Malicious Actions and Triggers

Despite appearing as a network management utility, the malware performs several hidden, malicious functions:

- **Data Exfiltration and Command-and-Control (C2) Communication**:
  - The program manages connections with external servers, conditionally sending or receiving data in chunks. The presence of specific error messages **(e.g., "Connection timed out after %lld milliseconds")** suggests it actively monitors network communication and handles partial data transmission, likely enabling it to upload or download sensitive data without causing disruption.

Decompile: FUN_1400070f0 - (freemem.exe.malz)

```
58      lVar2 = FUN_140029160(local_68[0]);
59      if ((*(int *)(lVar2 + 0x60) == 1) && (lVar6 = FUN_1401e220(lVar2,&local_48,'\0'), lVar6 < 0))
60      {
61        uVar7 = *(ulonglong *)(lVar2 + 0xb80);
62        uVar4 = *(ulonglong *)(lVar2 + 0xb88);
63        if (*(int *)(lVar2 + 0x60) == 4) {
64          local_78 = local_48;
65          uStack_70 = uStack_40;
66          local_58 = uVar7;
67          uStack_50 = uVar4;
68          uVar7 = FUN_140027050((longlong *)&local_78,(longlong *)&local_58);
69          FUN_1401b100(lVar2,"Resolving timed out after %lld milliseconds",uVar7,param_4);
70        }
71        else {
72          local_58 = local_48;
73          uStack_50 = uStack_40;
74          local_78 = uVar7;
75          uStack_70 = uVar4;
76          if (*(int *)(lVar2 + 0x60) == 5) {
77            uVar7 = FUN_140027050((longlong *)&local_58,(longlong *)&local_78);
78            FUN_1401b100(lVar2,"Connection timed out after %lld milliseconds",uVar7,param_4);
79          }
80          else {
81            lVar6 = *(longlong *)(lVar2 + 0x100);
82            uVar7 = FUN_140027050((longlong *)&local_58,(longlong *)&local_78);
83            param_4 = *(IMAGE_DOS_HEADER **)(lVar2 + 0x110);
84            if (lVar6 == -1) {
85              FUN_1401b100(lVar2,
86                           "Operation timed out after %lld milliseconds with %lld bytes received",
87                           uVar7,param_4);
88            }
89            else {
90              local_88 = lVar6;
91              FUN_1401b100(lVar2,
92                           "Operation timed out after %lld milliseconds with %lld out of %lld bytes
                              received"
93                           ,uVar7,param_4);
94            }
95          }
96        }
97        if (*(longlong *)(lVar2 + 0x18) != 0) {
98          if (9 < *(int *)(lVar2 + 0x60)) {
99            FUN_1401dfe0(*(longlong *)(lVar2 + 0x18),2);
100         }
101         uVar7 = CONCAT71((int7)(uVar7 >> 8),1);
102         FUN_140007880(lVar2,0x1c,uVar7);
103       }
104       FUN_1401b1f0(lVar2,"PENDING handle timeout",uVar7,param_4);
105       FUN_140007800((longlong)param_1,lVar2);
106     }
```

- **C2 Communication Trigger**: Malicious actions such as data exfiltration and receiving commands from a remote C2 are triggered based on network conditions, ensuring they activate only in connected environments or when specific socket events occur.

- **Memory Manipulation and Persistence**:
    - The program extensively modifies **IMAGE_DOS_HEADER** fields like **e_lfarlc, e_ovno, and e_res_4_** to manage its presence in memory. These adjustments serve as a form of stealth by erasing traces of its operations from memory, which reduces its chances of detection.
    - **Memory Persistence Trigger**: The malware conditionally clears and resets memory states, particularly after network operations or if a failure occurs. Functions like **FUN_140009760** and **FUN_14001a500** manage memory cleanup, ensuring that allocated resources are freed and reducing its detectable footprint.

Decompile: FUN_140009760 - (freemem.exe.malz)

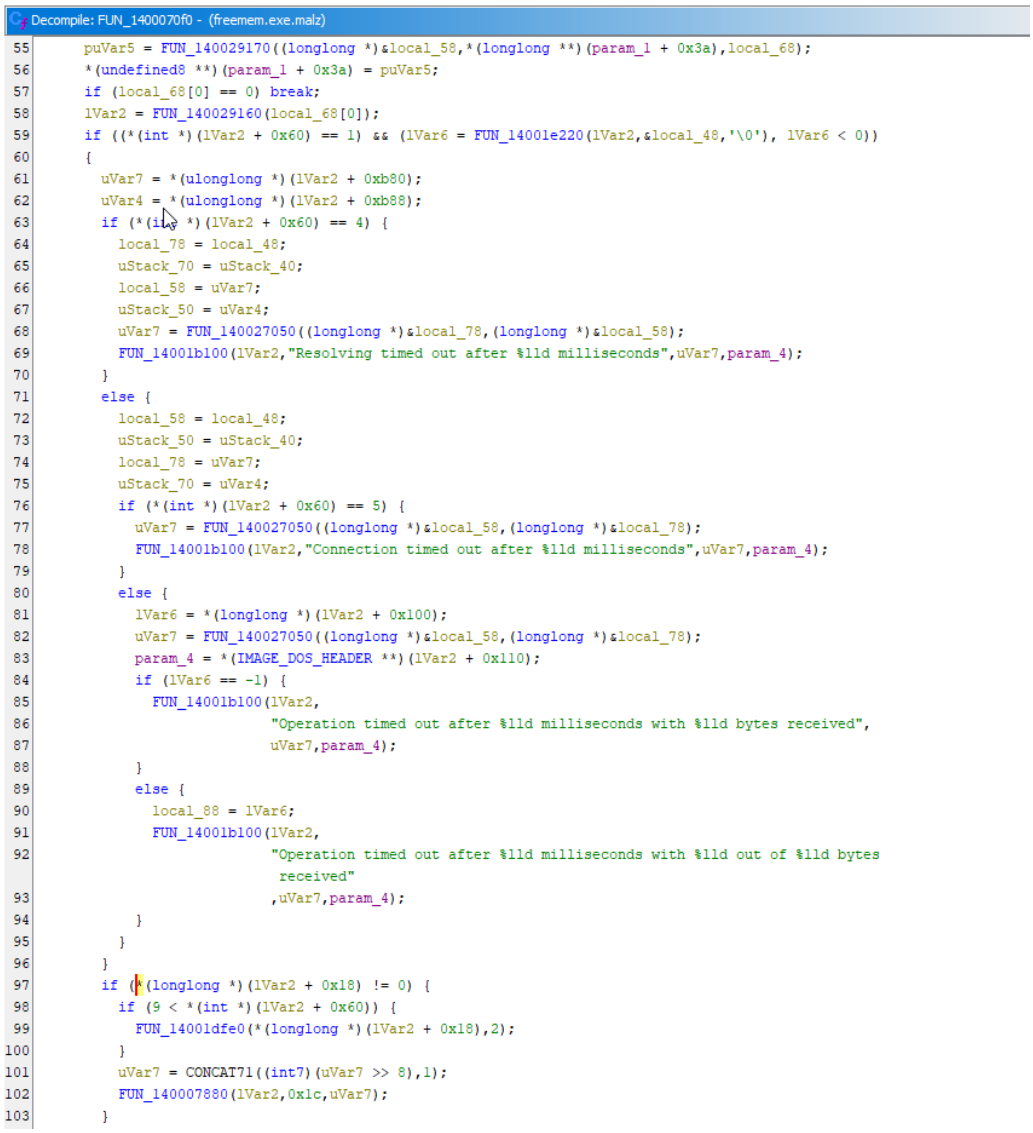
```
1
2 void FUN_140009760(longlong param_1)
3
4 {
5   (*(code *)PTR_FUN_140128008)(*(undefined8 *)(param_1 + 0xf0));
6   *(undefined8 *)(param_1 + 0xf0) = 0;
7   *(undefined8 *)(param_1 + 0xf8) = 0;
8   *(undefined *)(param_1 + 0x25d) = 0;
9   (*(code *)PTR_FUN_140128008)(*(undefined8 *)(param_1 + 0x100));
10  *(undefined8 *)(param_1 + 0x100) = 0;
11  *(undefined8 *)(param_1 + 0x108) = 0;
12  *(undefined *)(param_1 + 0x25e) = 0;
13  return;
14 }
15
```

Decompile: FUN_14001a500 - (freemem.exe.malz)

```
1
2 void FUN_14001a500(longlong param_1,undefined8 param_2)
3
4 {
5   longlong *plVar1;
6   longlong lVar2;
7   undefined8 *puVar3;
8
9   puVar3 = (undefined8 *)(param_1 + 0x268);
10  lVar2 = 2;
11  do {
12    for (plVar1 = (longlong *)*puVar3; plVar1 != (longlong *)0x0; plVar1 = (longlong *)plVar1[1]) {
13      if (*(code **)(*plVar1 + 0x58) != FUN_14000c690) {
14        (**(code **)(*plVar1 + 0x58))(plVar1,param_2,2,0,0);
15      }
16    }
17    puVar3 = puVar3 + 1;
18    lVar2 = lVar2 + -1;
19  } while (lVar2 != 0);
20  return;
21 }
22
```

- **Adaptive Error Handling and Stealth**
    - The malware demonstrates adaptive error handling by monitoring network and memory conditions, ensuring continuous and stealthy operation even under unexpected states. Here's how it achieves this:

1. Timeout and Network Error Handling:
    - Functions like **LAB_1400adcbb** and **FUN_140009760** include error logging for network events. For instance, when a network timeout occurs, messages like "Connection timed out after %lld milliseconds" are triggered. These messages accompany checks that monitor socket events and network responses. If an operation fails, the malware logs the error and attempts to reset or retry, avoiding noticeable system errors.



```
Decompile: FUN_1400070f0 - (freemem.exe.malz)
55      puVar5 = FUN_140029170((longlong *)&local_58,*(longlong **)(param_1 + 0x3a),local_68);
56      *(undefined8 **)(param_1 + 0x3a) = puVar5;
57      if (local_68[0] == 0) break;
58      lVar2 = FUN_140029160(local_68[0]);
59      if ((*(int *)(lVar2 + 0x60) == 1) && (lVar6 = FUN_14001e220(lVar2,&local_48,'\0'), lVar6 < 0))
60      {
61        uVar7 = *(ulonglong *)(lVar2 + 0xb80);
62        uVar4 = *(ulonglong *)(lVar2 + 0xb88);
63        if (*(int *)(lVar2 + 0x60) == 4) {
64          local_78 = local_48;
65          uStack_70 = uStack_40;
66          local_58 = uVar7;
67          uStack_50 = uVar4;
68          uVar7 = FUN_140027050((longlong *)&local_78,(longlong *)&local_58);
69          FUN_14001b100(lVar2,"Resolving timed out after %lld milliseconds",uVar7,param_4);
70        }
71        else {
72          local_58 = local_48;
73          uStack_50 = uStack_40;
74          local_78 = uVar7;
75          uStack_70 = uVar4;
76          if (*(int *)(lVar2 + 0x60) == 5) {
77            uVar7 = FUN_140027050((longlong *)&local_58,(longlong *)&local_78);
78            FUN_14001b100(lVar2,"Connection timed out after %lld milliseconds",uVar7,param_4);
79          }
80          else {
81            lVar6 = *(longlong *)(lVar2 + 0x100);
82            uVar7 = FUN_140027050((longlong *)&local_58,(longlong *)&local_78);
83            param_4 = *(IMAGE_DOS_HEADER **)(lVar2 + 0x110);
84            if (lVar6 == -1) {
85              FUN_14001b100(lVar2,
86                            "Operation timed out after %lld milliseconds with %lld bytes received",
87                            uVar7,param_4);
88            }
89            else {
90              local_88 = lVar6;
91              FUN_14001b100(lVar2,
92                            "Operation timed out after %lld milliseconds with %lld out of %lld bytes
                               received"
93                            ,uVar7,param_4);
94            }
95          }
96        }
97        if (*(longlong *)(lVar2 + 0x18) != 0) {
98          if (9 < *(int *)(lVar2 + 0x60)) {
99            FUN_14001dfe0(*(longlong *)(lVar2 + 0x18),2);
100         }
101         uVar7 = CONCAT71((int7)(uVar7 >> 8),1);
102         FUN_140007880(lVar2,0x1c,uVar7);
103       }
```

2. Socket Event Management:
    - The malware uses WSAEventSelect and WSAEnumNetworkEvents in functions like **FUN_140009760** to manage socket events. When a socket operation fails, it resets associated memory fields (param_1 + 0xf0, param_1 + 0xf8) and clears flags, effectively handling errors without drawing attention. This behavior allows the malware to continue operating without leaving lingering socket events or detectable memory anomalies.

3. Memory Cleanup and State Reset:
    - By iterating through memory pointers and invoking specific cleanup routines, **FUN_14001a500** ensures that previously allocated resources are freed or reset, likely to avoid lingering references in memory. This cleanup

mechanism helps the malware remain undetectable by clearing memory references that might otherwise persist.

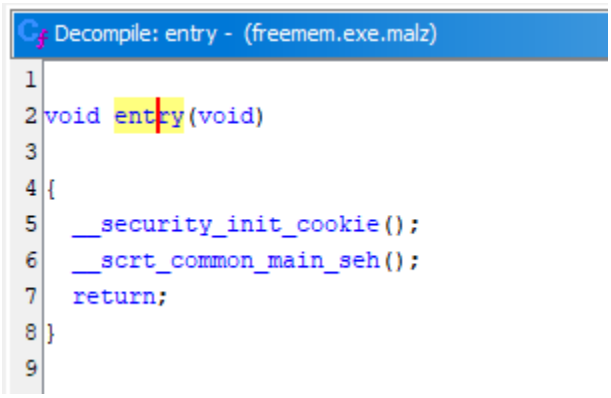
Decompile: FUN_14001a500 - (freemem.exe.malz)

```
1
2 void FUN_14001a500(longlong param_1,undefined8 param_2)
3
4 {
5   longlong *plVar1;
6   longlong lVar2;
7   undefined8 *puVar3;
8
9   puVar3 = (undefined8 *)(param_1 + 0x268);
10  lVar2 = 2;
11  do {
12    for (plVar1 = (longlong *)*puVar3; plVar1 != (longlong *)0x0; plVar1 = (longlong *)plVar1[1]) {
13      if (*(code **)(*plVar1 + 0x58) != FUN_14000c690) {
14        (**(code **)(*plVar1 + 0x58))(plVar1,param_2,2,0,0);
15      }
16    }
17    puVar3 = puVar3 + 1;
18    lVar2 = lVar2 + -1;
19  } while (lVar2 != 0);
20  return;
21 }
22
```

In the broader context, **FUN_14001a500** contributes to the malware's stealth by handling and resetting memory pointers based on specific conditions. This approach prevents the accumulation of memory or resource states that could expose its presence, ensuring that it adapts by clearing or resetting references in memory. Thus, it supports adaptive behavior by ensuring memory consistency and minimizing detection risks, even if direct error logging is not part of its specific function.

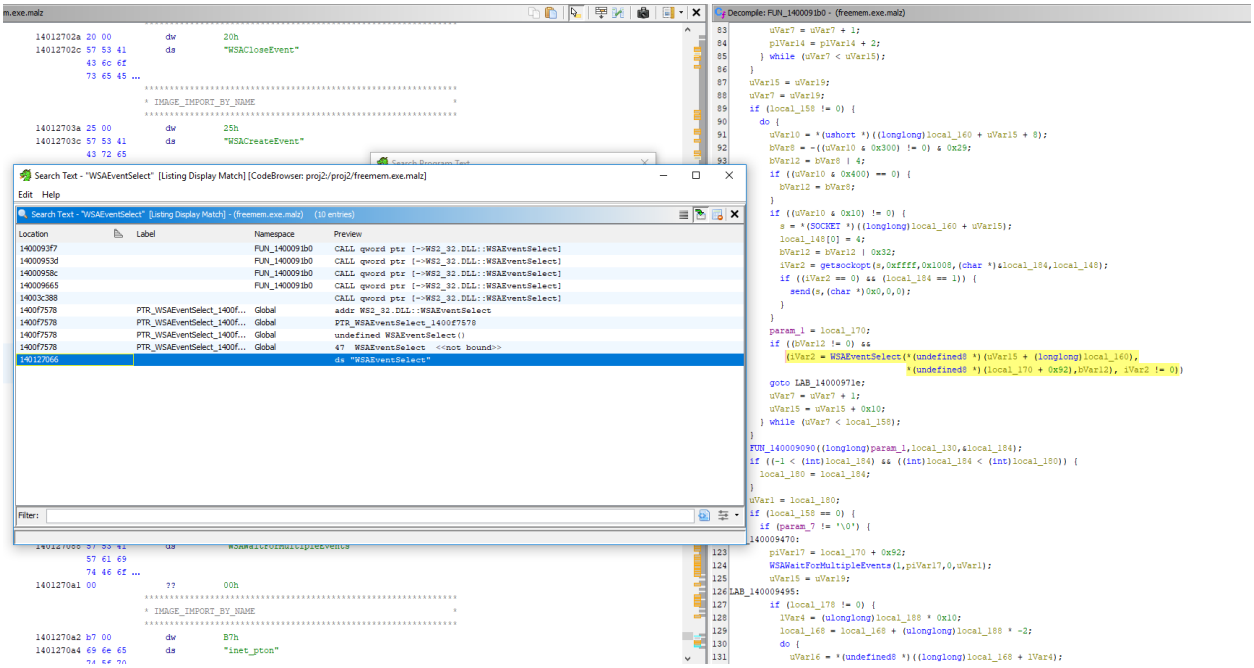
- **Entry and Security Initializations**:
  - **entry**: At the entry point, the function initializes security measures and triggers the main exception handling function (__scrt_common_main_seh).
  - **__security_init_cookie**: This function, commonly seen in Visual Studio binaries, generates a randomized security cookie to protect the program against buffer overflows and other memory-based attacks. This cookie uses system time and unique identifiers such as thread and process IDs, effectively creating a dynamic value that varies between sessions.
  - **__scrt_common_main_seh**: Handles the program's structured exception handling (SEH) and manages startup locks, resource allocation, and critical sections within the main runtime environment. This function checks

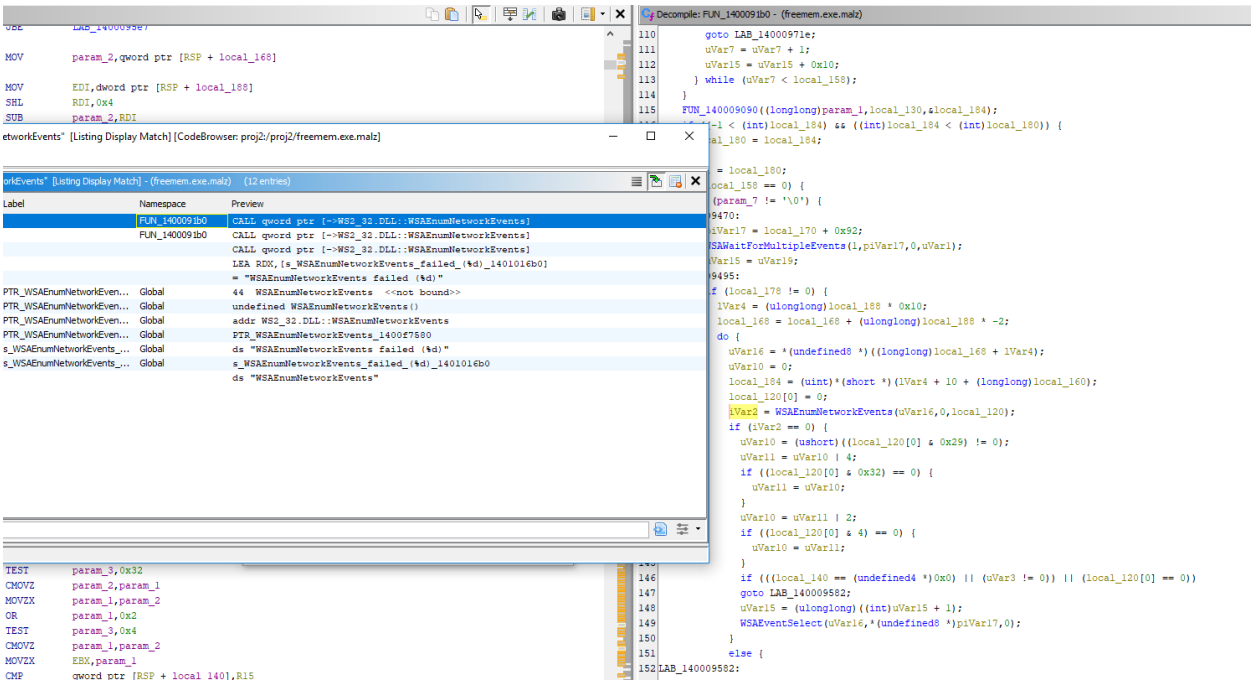
various startup conditions, initializing resources dynamically and ensuring graceful handling of exceptions. If a startup condition fails, it triggers a fallback function (FUN_1400d05ec), allowing the malware to continue running in a reduced state.



3. Indicators of Compromise (IOCs)

- **Host-Based IOCs**:
  - **Unusual Socket Event Monitoring**: Use of **WSAEventSelect** and **WSAEnumNetworkEvents** to handle socket states. These functions, if observed in monitoring tools, indicate abnormal behavior, especially when continuously resetting events or logging connection states.



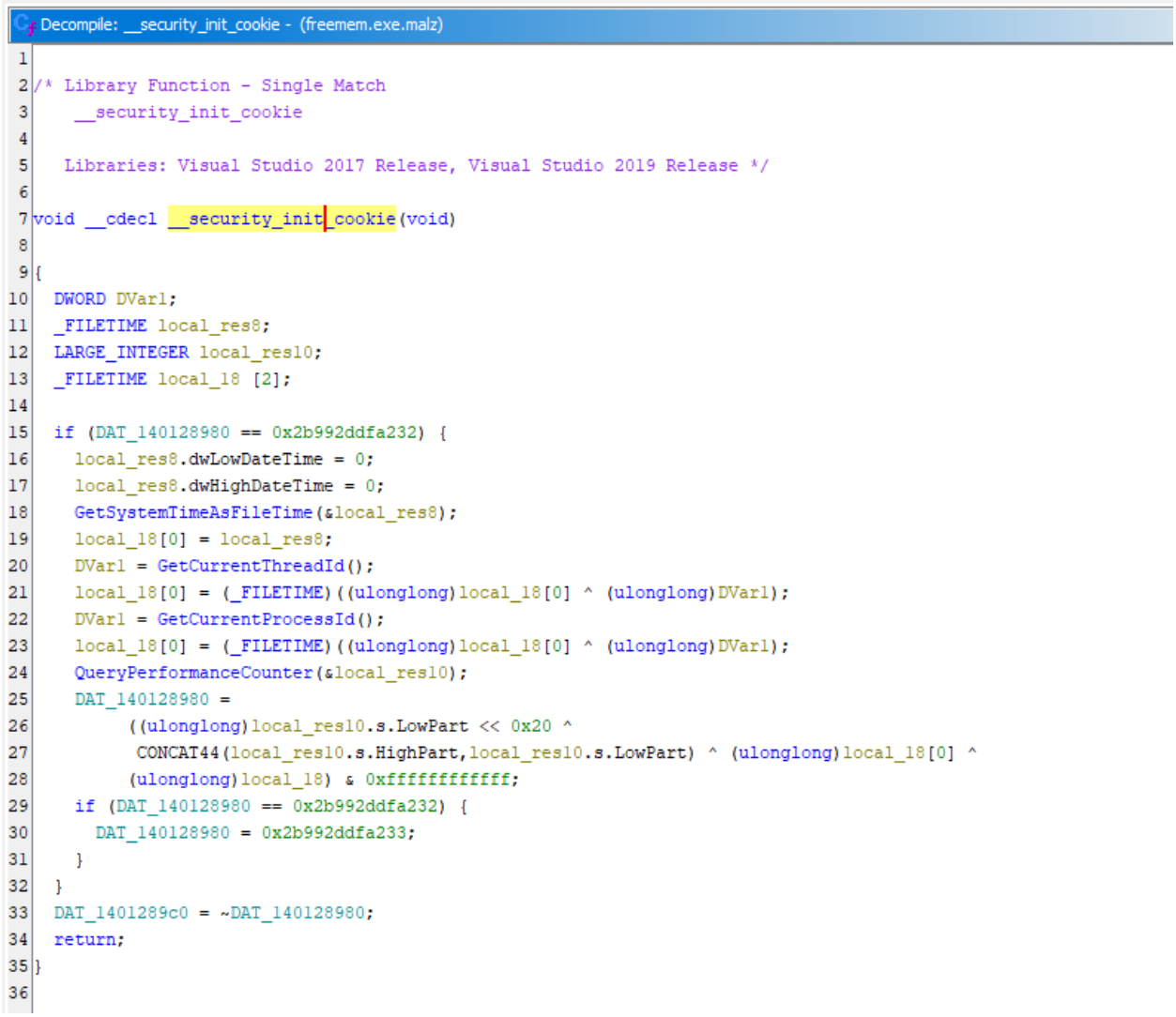


- **Memory Manipulation in IMAGE_DOS_HEADER**: Frequent adjustments to fields such as **e_lfarlc**, **e_ovno**, and **e_res_4_** are indicators of this malware. These adjustments suggest that memory monitoring or analysis tools would detect recurrent resetting or clearing of these fields

```
        assume DF = 0x0  (Default)
                        IMAGE_DOS_HEADER_140000000                XREF[1]:     140000140(*)
140000000 4d 5a 90          IMAGE_DO...
          00 03 00
          00 00 04 ...
  140000000 4d 5a           char[2]   "MZ"              e_magic                       XREF[1]:     140000140(*)
    140000000 [0]             'M', 'Z'
  140000002 90 00           dw        90h               e_cblp        Bytes of last page
  140000004 03 00           dw        3h                e_cp          Pages in file
  140000006 00 00           dw        0h                e_crlc        Relocations
  140000008 04 00           dw        4h                e_cparhdr     Size of header in ...
  14000000a 00 00           dw        0h                e_minalloc    Minimum extra para...
  14000000c ff ff           dw        FFFFh             e_maxalloc    Maximum extra para...
  14000000e 00 00           dw        0h                e_ss          Initial (relative)...
  140000010 b8 00           dw        B8h               e_sp          Initial SP value
  140000012 00 00           dw        0h                e_csum        Checksum
  140000014 00 00           dw        0h                e_ip          Initial IP value
  140000016 00 00           dw        0h                e_cs          Initial (relative)...
  140000018 40 00           dw        40h               e_lfarlc      File address of re...
  14000001a 00 00           dw        0h                e_ovno        Overlay number
  14000001c 00 00 00 00 00  dw[4]                       e_res[4]      Reserved words
            00 00 00
  140000024 00 00           dw        0h                e_oemid       OEM identifier (fo...
  140000026 00 00           dw        0h                e_oeminfo     OEM information; e...
  140000028 00 00 00 00 00  dw[10]                      e_res2[10]    Reserved words
            00 00 00 00 00
            00 00 00 00 00...
  14000003c 10 01 00 00     ddw       110h              e_lfanew      File address of ne...
  140000040 0e 1f ba 0e 00  db[64]                      e_program     Actual DOS program
            b4 09 cd 21 b8
            01 4c cd 21 54...
```

Breakdown of __security_init_cookie

1. **System Time Initialization**:
    - The function begins by zeroing out **local_res8** and then calling **GetSystemTimeAsFileTime(&local_res8)** to retrieve the current system time. This **FILETIME** structure, stored in **local_res8**, serves as the initial base value for the security cookie.
2. **Thread and Process ID XOR Operations**:
    - Next, the current thread ID is retrieved with **GetCurrentThreadId()** and **XORed** with **local_res8**, adding an element of randomness based on the active thread.
    - The current process ID, obtained via **GetCurrentProcessId()**, is then **XORed** with **local_res8** again, incorporating process-specific information. These two XOR operations ensure that each process and thread generates a unique base value.
3. **Performance Counter Addition**:
    - The function calls **QueryPerformanceCounter(&local_res10)** to obtain a high-resolution timestamp, which is stored in **local_res10**. This counter value is then shifted and combined with the previous **XORed** values of **local_res8** and **local_18[0]**, contributing further randomness based on the system's performance state.
4. **Final Cookie Calculation**:
    - The final cookie value, **DAT_140128980**, is computed by combining **local_res10** and **local_18** through **XOR** and shift operations, ensuring a unique, unpredictable result. It masks the result with **0xffffffffffff** to limit the size of the final value.
    - A check ensures that if **DAT_140128980** matches the hardcoded value **0x2b992ddfa232**, it's incremented slightly to avoid any potential repetition.
5. **Inversion for Additional Security**:
    - Lastly, the function stores the bitwise negation of **DAT_140128980** in **DAT_1401289c0**, adding an additional layer of security by creating a complementary value for later use.



- **Network-Based IOCs**:

Function Analysis: FUN_140016830

FUN_140016830 appears to manage complex network interactions, possibly related to establishing or reusing network connections. Here are the main actions and details that are relevant for identifying its behaviors as potential Network-Based Indicators of Compromise:

1. **Network Activity and Timeout Management**:
    - The function manages network connections and includes a variety of conditional steps that involve allocating and freeing resources, reassigning connection pointers, and performing error-handling in the event of network timeouts or connection failures. Specific messages, such as "Connection timed out after %lld milliseconds" and "Resolving timed out after %lld

milliseconds," could appear in logs if network traffic monitoring is enabled. These timeout messages indicate that the malware encounters and responds to network issues, potentially trying to connect to external servers like a Command-and-Control (C2) server.

```
300   FUN_14001b1f0(param_1,"Re-using existing connection with %s %s",pcVar14,pcVar16);
301   lVar9 = lVar11;
```

```
330         lVar11 = *(longlong *)(param_1 + 0xa18);
331         pcVar17 = "Allowing DoH to override max connection limit";
332         if (lVar11 < 0) {
333           pcVar17 = "No connections available in cache";
334         }
335         FUN_14001b1f0(param_1,pcVar17,pcVar14,pcVar16);
336         if (lVar11 < 0) goto LAB_14001717d;
337       }
338       uVar12 = FUN_140013df0(param_1,lVar9);
339       if ((int)uVar12 != 0) goto LAB_140017la1;
340       FUN_140005b00(param_1,lVar9);
341       uVar12 = FUN_1400105c0(param_1,lVar9);
342       if ((int)uVar12 != 0) goto LAB_140017la1;
343       if (((*(byte *)(param_1 + 0xd14) & 8) != 0) && (*(char *)(param_1 + 0xd1c) != (char)uVar12))
344       {
345         FUN_14001b1f0(param_1,"NTLM picked AND auth done set, clear picked",pcVar14,pcVar16);
346         *(undefined4 *)(param_1 + 0xd14) = 0;
347         *(char *)(param_1 + 0xd1c) = (char)uVar12;
348       }
349       if (((*(byte *)(param_1 + 0xd24) & 8) != 0) && (*(char *)(param_1 + 0xd2c) != '\0')) {
350         FUN_14001b1f0(param_1,"NTLM-proxy picked AND auth done set, clear picked",pcVar14,pcVar16)
351         ;
352         *(undefined4 *)(param_1 + 0xd24) = 0;
353         *(undefined *)(param_1 + 0xd2c) = 0;
354       }
355       goto LAB_1400170ec;
356     }
357     FUN_14001b1f0(param_1,"No more connections allowed to host",pcVar14,pcVar16);
358   }
359 LAB_14001717d:
360   FUN_14001b1f0(param_1,"No connections available.",pcVar14,pcVar16);
361   FUN_1400156e0(param_1,lVar9);
362   *param_2 = 0;
363 }
```

- This retry and error-logging behavior serves as an indicator of stealthy, persistent connection attempts. Repeated connection errors, partial data transmission, or socket resets may show up as patterns in network monitoring logs.

2. **Data Transmission Anomalies**:
   - FUN_140016830 handles various stages of data transmission, with fallback mechanisms for incomplete or interrupted transfers. These anomalies, such as partial transmissions and retries, could be detectable with network analysis tools, especially if connections repeatedly fail or are directed toward suspicious IPs or untrusted domains.
3. **Anonymous and Default Credentials (ftp@example.com and anonymous)**:
   - The function assigns default credentials, "ftp@example.com" and "anonymous", to the pcVar17 and pcVar14 variables, respectively. These

values are used conditionally; they may represent fallback credentials for anonymous FTP access or a similar default configuration.

```
77  pcVar14 = "anonymous";
78  pcVar17 = "ftp@example.com";
79  if ((*(byte *)(*(longlong *)(lVar
```

```
296    pcVar14 = "host";
297    if (*(char *)(lVar11 + 0x3
298      pcVar14 = "proxy";
299    }
```

- Depending on specific flags, these credentials are either retained or cleared, with alternative connection parameters applied if certain conditions are met. This dynamic credential management could indicate attempts to mask the origin or identity of the connection, as well as evade detection by using non-specific, low-privilege credentials.

4. **Connection Reuse and Authentication Management**:
   - The function reuses existing connections under certain conditions, checking states and managing credentials. Strings like "Re-using existing connection with %s %s" and "No connections available" confirm that it's managing connection limits and re-allocating resources, which could help it establish persistence over network connections.

```
300    FUN_14001b1f0(param_1,"Re-using existing connection with %s %s",pcVar14,pcVar16);
301    lVar9 = lVar11;
```

```
  pcVar17 = "No connections available in cache";
}
```

Network-Based Indicators of Compromise (IOCs)

Based on the behavior observed in FUN_140016830, here are some specific IOCs:

- **Connection Patterns with Timeouts and Retries**: The program logs and handles timeouts and connection errors. If repeated timeout messages or connection errors with external IPs are detected, this may indicate attempts to reach a C2 server.
- **Anomalous Data Transfer Activity**: Incomplete or partial transfers, along with repeated retries and socket resets, could show up in network logs as unusual patterns, suggesting malware presence.

- **Use of Default FTP Credentials**: The presence of "ftp@example.com" and "anonymous" as credentials in network traffic, especially if unusual for the host environment, can serve as a red flag for suspicious activity.

These indicators, combined with the logging messages and network management behaviors in FUN_140016830, highlight its role in stealthy, persistent communication, potentially aimed at maintaining unauthorized access or data exfiltration.

Data Transmission Anomalies: Patterns of data transfer with partial transmission logs, retries, or socket event resets suggest the presence of this malware. Network monitoring tools may detect repeated connection attempts with untrusted domains or IPs, especially when connections are incomplete or interrupted.

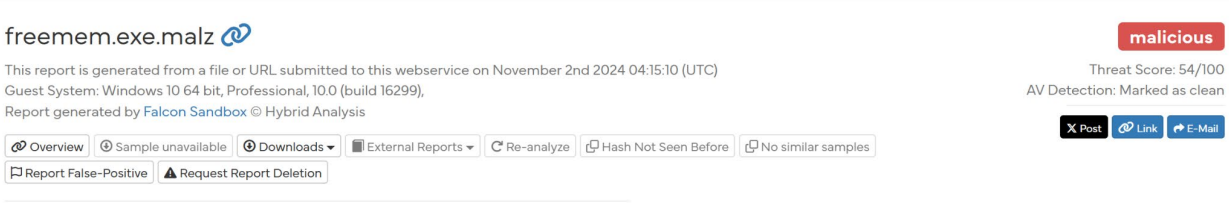
4. Remediation Steps for a Compromised Host

- **Network Connection Analysis and Blocking**: Investigate unusual or unknown IP addresses that the malware repeatedly contacts. Blocking suspicious IPs and monitoring network traffic for specific error messages can help in identifying and preventing future connections to potential C2 servers.
- **Memory and Process Scanning**: Using advanced memory inspection tools, scan for processes that manipulate IMAGE_DOS_HEADER fields or exhibit unusual memory allocation patterns. This malware's reliance on repeated memory resets and pointer adjustments can serve as an identifiable trait.
- **Process Termination and Safe Mode Scan**: If malware processes are identified, terminate them using a tool like Process Explorer. Reboot the system in Safe Mode and perform a comprehensive antivirus scan to detect and remove any remaining components.

Hybrid Analysis

Executive Summary MITRE Attacks & Techniques

-



- We used another tool for malware static analysis and this hash has a match on the website and outputs to be a malicious file.
    - The tool: https://www.hybrid-analysis.com/
    - The analysis: https://www.hybrid-analysis.com/sample/d29bb539cb7ac6751ace47597a45b7751135a6c56e75d7a0b657eb887ddcece9
- MITRE Techniques and Tactics Used

ATT&CK ID	Name	Tactics	Description	Malicious Indicators	Suspicious Indicators	Informative Indicators
T1588.004	Digital Certificates	• Resource Development	Adversaries may buy and/or steal SSL/TLS certificates that can be used during targeting. Learn more		• Possibly contains SSL certificate	

-
    - Everything is labeled and displayed for us and shows us the capabilities and techniques listed from MITRE attacks of what the malicious file can do.

Execution

ATT&CK ID	Name	Tactics	Description	Malicious Indicators	Suspicious Indicators	Informative Indicators
T1559	Inter-Process Communication	• Execution	Adversaries may abuse inter-process communication (IPC) mechanisms for local code or command execution. Learn more			• Contains ability to create/connect to a named pipe (API string)
T1106	Native API	• Execution	Adversaries may interact with the native OS application programming interface (API) to execute behaviors. Learn more		• Imports suspicious APIs	• Contains ability to create/open files (API string) • Contains ability to execute Windows APIs • Contains ability to set/get the last-error code for a calling thread (API string) • Contains ability to retrieve the fully qualified path of module (API string) • Contains ability to retrieve/modify process thread (API string) • Contains ability to load modules (API string) • Contains ability to retrieve address of exported function from a DLL (API string) • Calls an API typically used to retrieve function addresses • Imports GetCommandLine API
T1059.003	Windows Command Shell	• Execution	Adversaries may abuse the Windows command shell for execution. Learn more			• Contains ability to retrieve the command-line string for the current process (API string)
T1129	Shared Modules	• Execution	Adversaries may execute malicious payloads via loading shared modules. Learn more			• Loads modules at runtime • Calls an API typically used to load libraries • Loads the RPC (Remote Procedure Call) module DLL

-
    - Imports malicious APIs that can be used to jumpstart the malware.

Persistence

ATT&CK ID	Name	Tactics	Description	Malicious Indicators	Suspicious Indicators	Informative Indicators
T1543.003	Windows Service	• Persistence • Privilege Escalation	Adversaries may create or modify Windows services to repeatedly execute malicious payloads as part of persistence. Learn more			• Contains ability to access device drivers • Creates or modifies windows services
T1543	Create or Modify System Process	• Persistence • Privilege Escalation	Adversaries may create or modify system-level processes to repeatedly execute malicious payloads as part of persistence. Learn more			• Contains ability to retrieve the contents of the STARTUPINFO structure (API string)

-
    - Frequently calls windows services to create, modify, delete, and retrieve content.

Privilege Escalation

ATT&CK ID	Name	Tactics	Description	Malicious Indicators	Suspicious Indicators	Informative Indicators
T1055.001	Dynamic-link Library Injection	• Defense Evasion • Privilege Escalation	Adversaries may inject dynamic-link libraries (DLLs) into processes in order to evade process-based defenses as well as possibly elevate privileges. Learn more			• Contains ability to load/free library (API string)
T1543.003	Windows Service	• Persistence • Privilege Escalation	Adversaries may create or modify Windows services to repeatedly execute malicious payloads as part of persistence. Learn more			• Contains ability to access device drivers • Creates or modifies windows services
T1055.003	Thread Execution Hijacking	• Defense Evasion • Privilege Escalation	Adversaries may inject malicious code into hijacked processes in order to evade process-based defenses as well as possibly elevate privileges. Learn more			• Creates a thread in a self process
T1055	Process Injection	• Defense Evasion • Privilege Escalation	Adversaries may inject code into processes in order to evade process-based defenses as well as possibly elevate privileges. Learn more			• Contains ability to inject code into another process (API string)
T1543	Create or Modify System Process	• Persistence • Privilege Escalation	Adversaries may create or modify system-level processes to repeatedly execute malicious payloads as part of persistence. Learn more			• Contains ability to retrieve the contents of the STARTUPINFO structure (API string)
T1055.011	Extra Window Memory Injection	• Defense Evasion • Privilege Escalation	Adversaries may inject malicious code into process via Extra Window Memory (EWM) in order to evade process-based defenses as well as possibly elevate privileges. Learn more		• 1 confidential indicators	

-
    - Injects into DLL and has the capability to promote to administrator to evade detection and also hijack, inject, create, and modify system-level processes.

Defense Evasion

ATT&CK ID	Name	Tactics	Description	Malicious Indicators	Suspicious Indicators	Informative Indicators
T1027	Obfuscated Files or Information	• Defense Evasion	Adversaries may attempt to make an executable or file difficult to discover or analyze by encrypting, encoding, or otherwise obfuscating its contents on the system or in transit. Learn more		• 3 confidential indicators	• Contains ability to perform AES/RC4 encryption (API string) • Contains ability to encrypt/encode message (API string) • Contains CRYPTO related strings • Contains ability to use Cryptographic classes • Loads the Bcrypt module DLL • Matched Compiler/Packer signature (DIE) • Shows ability to obfuscate file or information
T1070.006	Timestomp	• Defense Evasion	Adversaries may modify file time attributes to hide new or changes to existing files. Learn more			• Contains ability to retrieve file time (API string)
T1055.001	Dynamic-link Library Injection	• Defense Evasion • Privilege Escalation	Adversaries may inject dynamic-link libraries (DLLs) into processes in order to evade process-based defenses as well as possibly elevate privileges. Learn more			• Contains ability to load/free library (API string)
T1622	Debugger Evasion	• Defense Evasion • Discovery	Adversaries may employ various means to detect and avoid debuggers. Learn more			• Contains ability to check debugger is running (API string) • Contains ability to register a top-level exception handler (API string)
T1055.003	Thread Execution Hijacking	• Defense Evasion • Privilege Escalation	Adversaries may inject malicious code into hijacked processes in order to evade process-based defenses as well as possibly elevate privileges. Learn more			• Creates a thread in a self process
T1027.009	Embedded Payloads	• Defense Evasion	Adversaries may embed payloads within other files to conceal malicious content from defenses. Learn more		• section contains high entropy	

-
    - Uses Bcrypt module certificates to encrypt/decrypt itself and its contents to obfuscate and to hide itself within the system.

T1497.003	Time Based Evasion	• Defense Evasion • Discovery	Adversaries may employ various time-based methods to detect and avoid virtualization and analysis environments. Learn more			• Contains ability to delay execution by waiting for signal/timeout (API string) • Contains ability to retrieve the time elapsed since the system was started (API string)
T1055	Process Injection	• Defense Evasion • Privilege Escalation	Adversaries may inject code into processes in order to evade process-based defenses as well as possibly elevate privileges. Learn more			• Contains ability to inject code into another process (API string)
T1070.004	File Deletion	• Defense Evasion	Adversaries may delete files left behind by the actions of their intrusion activity. Learn more			• Contains ability to delete files/directories (API string)
T1497.002	User Activity Based Checks	• Defense Evasion • Discovery	Adversaries may employ various user activity checks to detect and avoid virtualization and analysis environments. Learn more			• Able to identify virtual environment by using user activity (API string)
T1055.011	Extra Window Memory Injection	• Defense Evasion • Privilege Escalation	Adversaries may inject malicious code into process via Extra Window Memory (EWM) in order to evade process-based defenses as well as possibly elevate privileges. Learn more		• 1 confidential indicators	
T1140	Deobfuscate/Decode Files or Information	• Defense Evasion	Adversaries may use Obfuscated Files or Information to hide artifacts of an intrusion from analysis. Learn more			• Contains ability to perform decryption (API string) • Shows ability to deobfuscate/decode files or information
T1027.005	Indicator Removal from Tools	• Defense Evasion	Adversaries may remove indicators from tools if they believe their malicious tool was detected, quarantined, or otherwise curtailed. Learn more			• Contains XOR operation loops [Stream disassembly]
T1480	Execution Guardrails	• Defense Evasion	Adversaries may use execution guardrails to constrain execution or actions based on adversary supplied and environment specific conditions that are expected to be present on the target. Learn more			• Shows ability to use execution guardrails

-
    - Malware is able to loop operations and keep injecting itself to create, delete, modify, and obfuscate files.

Credential Access

ATT&CK ID	Name	Tactics	Description	Malicious Indicators	Suspicious Indicators	Informative Indicators
T1555	Credentials from Password Stores	• Credential Access	Adversaries may search for common password storage locations to obtain user credentials. Learn more			• Contains string like password/secret

Discovery

ATT&CK ID	Name	Tactics	Description	Malicious Indicators	Suspicious Indicators	Informative Indicators
T1083	File and Directory Discovery	• Discovery	Adversaries may enumerate files and directories or may search in specific locations of a host or network share for certain information within a file system. Learn more		• Reads configuration files (.ini files)	• Contains ability to retrieve file and directory information (API string) • Contains ability to read files (API string) • Tries to access non-existent files (executable) • Reads files • Calls an API's typically used for searching a directory for a files • Contains ability to enumerate files on disk (API string) • Touches files
T1622	Debugger Evasion	• Defense Evasion • Discovery	Adversaries may employ various means to detect and avoid debuggers. Learn more			• Contains ability to check debugger is running (API string) • Contains ability to register a top-level exception handler (API string)

-
    - Ability to retrieve directory paths and also access, read, and write to files.

T1082	System Information Discovery	• Discovery	An adversary may attempt to get detailed information about the operating system and hardware, including version, patches, hotfixes, service packs, and architecture. Learn more		• Found system commands related strings	• Contains ability to retrieve the OS information (API string) • Contains ability to read software policies • Contains ability to determine disk drive type (API string) • Contains ability to query volume/memory size (API string) • Contains ability to retrieve the host's architecture (API string) • Contains ability to retrieve a module handle (API string) • Contains ability to retrieve the host name of a computer (API string) • Calls an API typically used to get system version information • Calls an API typically used to get product type • Reads the active computer name • Reads information about supported languages
T1057	Process Discovery	• Discovery	Adversaries may attempt to get information about running processes on a system. Learn more		• Queries process information	• Contains ability to enumerate process and/or its information (API string) • Contains ability to retrieve/open a process (API string) • Calls an API typically used for taking snapshot of the specified processes • Queries basic information of the specified process

-
    - Can query information and enumerate processes to host's architecture

T1012	Query Registry	• Discovery	Adversaries may interact with the Windows Registry to gather information about the system, configuration, and installed software. Learn more		• 1 confidential indicators	• Opens registry keys • Queries registry keys
T1497.003	Time Based Evasion	• Defense Evasion • Discovery	Adversaries may employ various time-based methods to detect and avoid virtualization and analysis environments. Learn more			• Contains ability to delay execution by waiting for signal/timeout (API string) • Contains ability to retrieve the time elapsed since the system was started (API string)
T1124	System Time Discovery	• Discovery	An adversary may gather the system time and/or time zone settings from a local or remote system. Learn more			• Contains ability to retrieve machine time (API string) • Contains ability to retrieve machine timezone (API string)
T1497.002	User Activity Based Checks	• Defense Evasion • Discovery	Adversaries may employ various user activity checks to detect and avoid virtualization and analysis environments. Learn more			• Able to identify virtual environment by using user activity (API string)
T1016	System Network Configuration Discovery	• Discovery	Adversaries may look for details about the network configuration and settings, such as IP and/or MAC addresses, of systems they access or through information discovery of remote systems. Learn more			• Contains ability to retrieve the local name for a socket (API string)
T1007	System Service Discovery	• Discovery	Adversaries may try to gather information about registered local system services. Learn more			• Queries services related registry keys

-
    - Has the ability to retrieve and open registry keys for system time based evasion techniques.

Command and Control

ATT&CK ID	Name	Tactics	Description	Malicious Indicators	Suspicious Indicators	Informative Indicators
T1095	Non-Application Layer Protocol	• Command and Control	Adversaries may use an OSI non-application layer protocol for communication between host and C2 server or among infected hosts within a network. Learn more			• Contains ability to communicate with C2 using Sockets (API string)
T1071	Application Layer Protocol	• Command and Control	Adversaries may communicate using OSI application layer protocols to avoid detection/network filtering by blending in with existing traffic. Learn more			• Found potential URL in binary/memory • Contains ability to communicate with network (API string) • Found potential IP address in binary/memory • Found potential URLs in memory dumps
T1105	Ingress Tool Transfer	• Command and Control	Adversaries may transfer tools or other files from an external system into a compromised environment. Learn more			• Contains ability to transfer data using curl • Contains ability to write files (API string) • Dropped files
T1573	Encrypted Channel	• Command and Control	Adversaries may employ an encryption algorithm to conceal command and control traffic rather than relying on any inherent protections provided by a communication protocol. Learn more		• 1 confidential indicators	• Possibly tries to communicate over SSL connection (HTTPS) • Making HTTPS connections using secure TLS/SSL version
T1071.003	Mail Protocols	• Command and Control	Adversaries may communicate using application layer protocols associated with electronic mail delivery to avoid detection/network filtering by blending in with existing traffic. Learn more			• Contains reference to Mail transfer protocol • Found mail related domain names
T1571	Non-Standard Port	• Command and Control	Adversaries may communicate using a protocol and port pairing that are typically not associated. Learn more			• Contains ability to open a port and listen for incoming connection (API string)

-

- Malware can open/close ports and make connections over HTTPS as well as mail protocols to communicate within the network.

T1071.003	Mail Protocols	• Command and Control	Adversaries may communicate using application layer protocols associated with electronic mail delivery to avoid detection/network filtering by blending in with existing traffic. Learn more			• Contains reference to Mail transfer protocol • Found mail related domain names
T1571	Non-Standard Port	• Command and Control	Adversaries may communicate using a protocol and port pairing that are typically not associated. Learn more			• Contains ability to open a port and listen for incoming connection (API string)
T1071.001	Web Protocols	• Command and Control	Adversaries may communicate using application layer protocols associated with web traffic to avoid detection/network filtering by blending in with existing traffic. Learn more			• Found user-agent related strings • Found string related to HTTP headers
T1132	Data Encoding	• Command and Control	Adversaries may encode data to make the content of command and control traffic more difficult to detect. Learn more			• Contains ability to perform Base64 encoding/decoding
T1090.003	Multi-hop Proxy	• Command and Control	Adversaries may chain together multiple proxies to disguise the source of malicious traffic. Learn more			• Contains Tor Onion URL
T1573.001	Symmetric Cryptography	• Command and Control	Adversaries may employ a known symmetric encryption algorithm to conceal command and control traffic rather than relying on any inherent protections provided by a communication protocol. Learn more			• Shows ability to use encryption for command and control traffic
T1573.002	Asymmetric Cryptography	• Command and Control	Adversaries may employ a known asymmetric encryption algorithm to conceal command and control traffic rather than relying on any inherent protections provided by a communication protocol. Learn more			• Shows ability to use asymmetric encryption algorithm

-

Exfiltration

ATT&CK ID	Name	Tactics	Description	Malicious Indicators	Suspicious Indicators	Informative Indicators
T1029	Scheduled Transfer	• Exfiltration	Adversaries may schedule data exfiltration to be performed only at certain times of day or at certain intervals. Learn more			• Contains ability to perform scheduled transfer (API string)

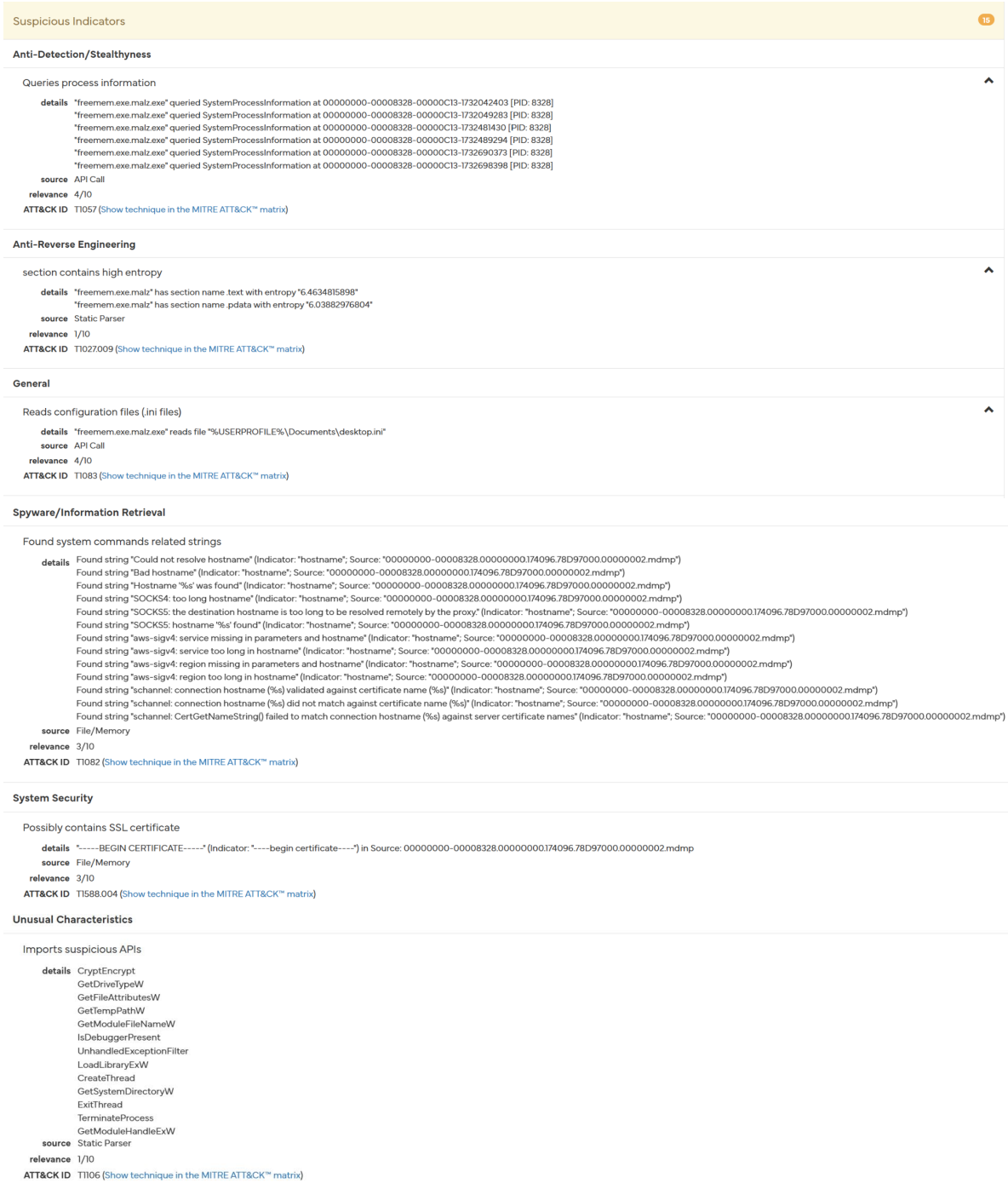
Impact

ATT&CK ID	Name	Tactics	Description	Malicious Indicators	Suspicious Indicators	Informative Indicators
T1486	Data Encrypted for Impact	• Impact	Adversaries may encrypt data on target systems or on large numbers of systems in a network to interrupt availability to system and network resources. Learn more		• 1 confidential indicators	• Contains reference to cryptographic keys
T1489	Service Stop	• Impact	Adversaries may stop or disable services on a system to render those services unavailable to legitimate users. Learn more			• Contains ability to terminate a process (API string)
T1529	System Shutdown/Reboot	• Impact	Adversaries may shutdown/reboot systems to interrupt access to, or aid in the destruction of, those systems. Learn more			• Contains ability to shutdown system

-

- Malware can stop and disable services on a system to interrupt availability of system and network resources to forcefully execute API imports and calls.

- Suspicious Indicators



- Informative

Anti-Detection/Stealthyness

- Contains ability to delay execution by waiting for signal/timeout (API string)
- Contains ability to inject code into another process (API string)
- Contains ability to load/free library (API string)
- Contains ability to perform Base64 encoding/decoding
- Contains ability to use Cryptographic classes

Anti-Reverse Engineering

- Contains ability to check debugger is running (API string)
- Contains ability to register a top-level exception handler (API string)

Cryptographic Related

- Contains XOR operation loops [Stream disassembly]
- Contains ability to encrypt/encode message (API string)
- Contains ability to perform AES/RC4 encryption (API string)
- Contains ability to perform decryption (API string)
- Contains reference to cryptographic keys
- Shows ability to deobfuscate/decode files or information
- Shows ability to obfuscate file or information
- Shows ability to use asymmetric encryption algorithm
- Shows ability to use encryption for command and control traffic

Environment Awareness

- Able to identify virtual environment by using user activity (API string)
- Calls an API typically used to get product type
- Calls an API typically used to get system version information
- Contains ability to perform scheduled transfer (API string)
- Contains ability to query volume/memory size (API string)
- Contains ability to read software policies
- Contains ability to retrieve file time (API string)
- Contains ability to retrieve machine time (API string)
- Contains ability to retrieve machine timezone (API string)
- Contains ability to retrieve the OS information (API string)
- Contains ability to retrieve the contents of the STARTUPINFO structure (API string)
- Reads the active computer name

General

- Attempts to read Outlook-related files (file access)
- Calls an API typically used to load libraries
- Calls an API typically used to retrieve function addresses
- Contains ability to create directories (API string)
- Contains ability to create/open files (API string)
- Contains ability to execute Windows APIs
- Contains ability to move file or directory (API string)
- Contains ability to retrieve the command-line string for the current process (API string)
- Contains ability to retrieve/modify process thread (API string)
- Contains ability to retrieve/open a process (API string)
- Contains ability to set/get the last-error code for a calling thread (API string)
- File contains dynamic base/NX flags
- Found user-agent related strings
- Loads modules at runtime
- Loads the Bcrypt module DLL
- Loads the RPC (Remote Procedure Call) module DLL
- Matched Compiler/Packer signature (DIE)
- PE file contains Debug data directory
- PE file contains executable sections
- PE file contains writable sections
- PE file entrypoint instructions
- PE file has a high image base
- Reads information about supported languages
- Writes files in a temp directory

Installation/Persistence

- Contains ability to load modules (API string)
- Creates a thread in a self process
- Dropped files
- Opens registry keys
- Queries basic information of the specified process
- Queries registry keys
- Reads files
- Shows ability to use execution guardrails
- Touches files
- Tries to access non-existent files (executable)
- Writes files

Network Related

- Contains Tor Onion URL
- Contains ability to communicate with C2 using Sockets (API string)
- Contains ability to communicate with network (API string)
- Contains ability to create/connect to a named pipe (API string)
- Contains ability to open a port and listen for incoming connection (API string)
- Contains ability to transfer data using curl
- Contains reference to Mail transfer protocol
- Found mail related domain names
- Found potential IP address in binary/memory
- Found potential URL in binary/memory
- Found potential URLs in memory dumps
- Found string related to HTTP headers
- JA3 SSL client fingerprint
- Making HTTPS connections using secure TLS/SSL version
- Possibly tries to communicate over SSL connection (HTTPS)
- Shows ability to send encrypted data to the internet

Spyware/Information Retrieval

- Calls an API typically used for taking snapshot of the specified processes
- Calls an API's typically used for searching a directory for a files
- Contains CRYPTO related strings
- Contains ability to determine disk drive type (API string)
- Contains ability to enumerate files on disk (API string)
- Contains ability to enumerate process and/or its information (API string)
- Contains ability to read files (API string)
- Contains ability to retrieve a module handle (API string)
- Contains ability to retrieve address of exported function from a DLL (API string)
- Contains ability to retrieve file and directory information (API string)
- Contains ability to retrieve the fully qualified path of module (API string)
- Contains ability to retrieve the host name of a computer (API string)
- Contains ability to retrieve the host's architecture (API string)
- Contains ability to retrieve the local name for a socket (API string)
- Contains ability to retrieve the time elapsed since the system was started (API string)
- Contains reference to Outlook data files path
- Contains string like password/secret
- Imports GetCommandLine API

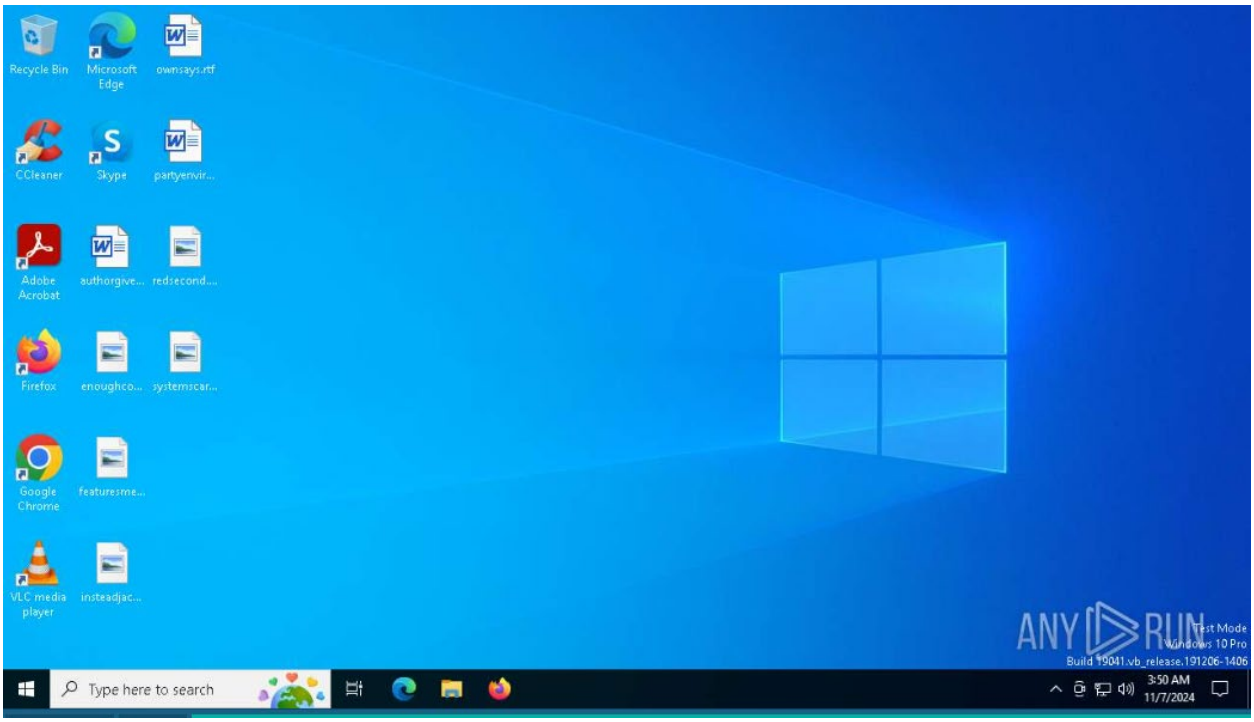
System Security

- Contains ability to access device drivers
- Contains ability to delete files/directories (API string)
- Contains ability to shutdown system
- Contains ability to terminate a process (API string)
- Contains ability to write files (API string)
- Creates or modifies windows services
- Queries services related registry keys

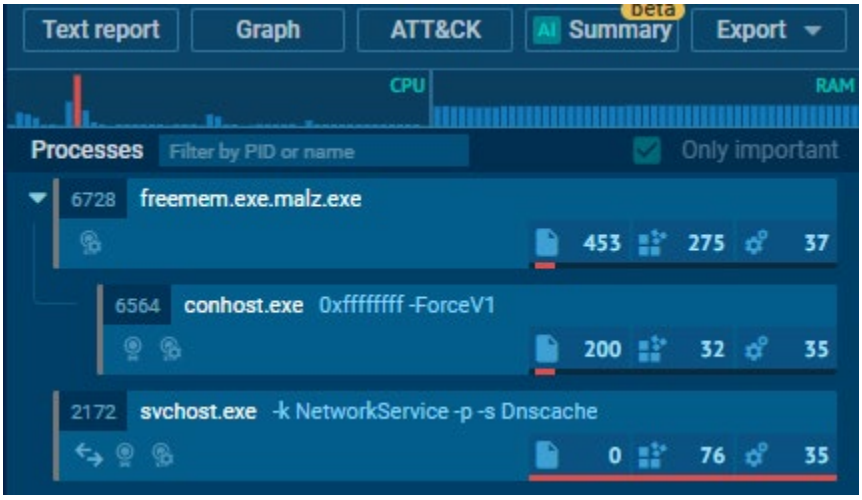
Dynamic Analysis

For a more in-depth approach to Dynamic Analysis, we used ANY.RUN to simulate running and monitoring the malware. To simulate a more hospitable host environment to the malware, this tool was configured to run in 64-bit Windows 10 OS. The full analysis can be reached here: https://app.any.run/tasks/7e66da12-eed3-4deb-83ba-1c9f3ab4b29a

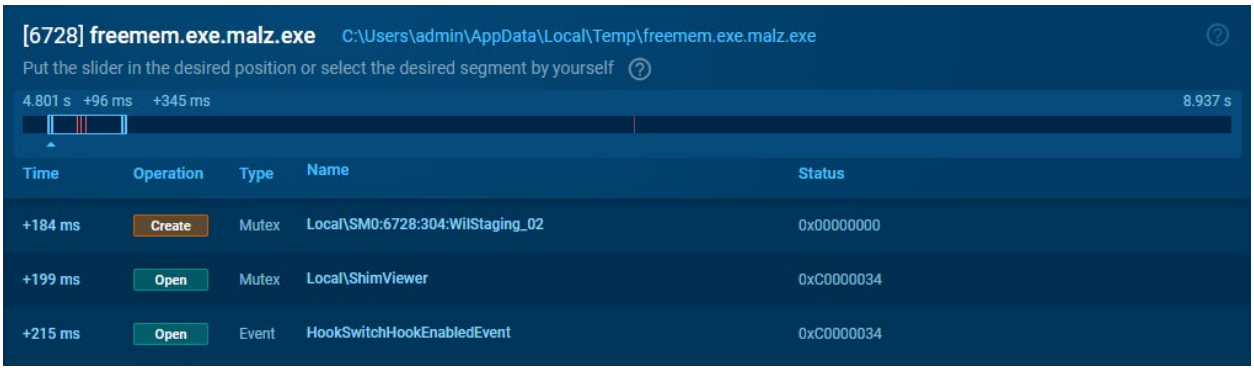
On a surface level, the malware appears to be inert; it does not visually open any programs, much apropos to the screenshots in the section under General Analysis. Case in point, it appears that the malware is making a concerted effort to make itself appear as innocuous as possible. In the screenshot below, we can see there are no visible changes or opened programs in the background. This maintains throughout the length of the malware's execution as there are no changes, pop-ups, or applications being launched in the meantime.



Throughout its execution, the malware only creates three active processes which spike the CPU once before returning to normalcy, though it appears that the RAM is consistent throughout. However, we cannot fully rely on the RAM levels alone as analyzing the decompiled code in Ghidra shows us that the malware may be privy to manipulating memory as a means of obfuscation. Unsurprisingly, only the first process, titled “freemem.exe.malz.exe,” is the only one to set off flags associated with malware. The program makes a couple of system calls, namely to read what the computer’s name is. Under ATT&CK classifications of threat actions, this falls under the Query Registry (T1012) and System Information Discovery (T1082) subclasses of the greater Discovery category.

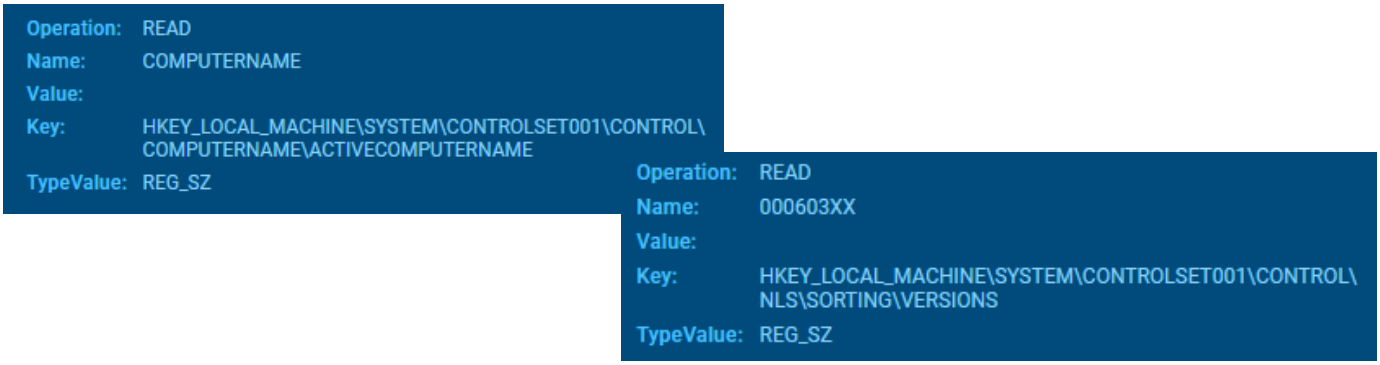


Additionally, it appears that the program modified a temporary file found in the “C:\Users\admin\AppData\Local\Temp\” directory called “exfB0F9.tmp.zst”. While legitimate programs use temporary files all the time, there is no telling what it does within this file. This operation was also referenced and documented in the General Analysis section under File Operations as well. It appears the creation of the file occurs roughly around the same time as the spike in CPU. The given area in the “freemem.exe.malz.exe” timeline highlights this fact.



These operations were found pointing to the following keys:

- HKEY_LOCAL_MACHINE\SYSTEM\CONTROLSET001\CONTROL\NLS\SORTING\VERSIONS
- HKEY_LOCAL_MACHINE\SYSTEM\CONTROLSET001\CONTROL\COMPUTERNAME\ACTIVECOMPUTERNAME



The “NLS” part of the former key refers to the National Language Support functions which contain supported languages, regional settings, and other sorting conventions. It is possible that reading this key may be a workaround for finding which version of the operating system the host computer may be running on given that the sorting version numbers are different depending on which iteration of Windows you may be running. One can find the computer’s name in the latter key. The operations conducted to these keys were specifically ‘read’ instructions, meaning that nothing was altered or destroyed. The fact that the malware is looking to read these particular elements of the computer can be indicative that it could be collecting information to profile the environment where it is running.

Timeshift	Status	Rep	Domain	IP
BEFORE	Responded		settings-win.data.microsoft.com	40.127.240.158
BEFORE	Responded		crl.microsoft.com	2.16.164.49 2.16.164.9
BEFORE	Responded		www.microsoft.com	95.101.149.131
BEFORE	Responded		google.com	142.250.186.174
2529 ms	Responded		c2-7f000001.nip.io	127.0.0.1
8635 ms	Responded		www.bing.com	2.23.209.130 2.23.209.179 2.23.209.182 2.23.209.187 2.23.209.149 2.23.209.140
8637 ms	Responded		ocsp.digicert.com	192.229.221.95
9037 ms	Responded		login.live.com	20.190.159.75 40.126.31.73 40.126.31.71 20.190.159.2 40.126.31.69 20.190.159.0 20.190.159.6 20.190.159.2
9037 ms	Responded		th.bing.com	2.23.209.140 2.23.209.130 2.23.209.179 2.23.209.182 2.23.209.187 2.23.209.149
11641 ms	Responded		go.microsoft.com	184.28.89.16
28052 ms	Responded		slscr.update.microsoft.com	4.245.163.56
28052 ms	Responded		www.microsoft.com	184.30.21.17
29052 ms	Responded		fe3cr.delivery.mp.microsoft.com	13.85.23.206
29053 ms	Responded		fe3cr.delivery.mp.microsoft.com	13.85.23.206
29053 ms	Responded		slscr.update.microsoft.com	4.245.163.56
30155 ms	Responded		arc.msn.com	20.223.36.55
30155 ms	Responded		fd.api.iris.microsoft.com	20.223.36.55
34158 ms	Responded		settings-win.data.microsoft.com	52.137.106.2
59777 ms	Responded		nexusrules.officeapps.live.com	52.111.236.2

HTTP Requests 7 | Connections 36 | DNS Requests 19 | Threats 1 | Filter by PID, domain, name or ip | PCAP

Timeshift	Protocol	Rep	PID	Process name	CN	IP	Port	Domain	ASN	Traffic
BEFORE	UDP		4	System		192.168.100.255	137	–	–	↑ 1020 b ↓ –
BEFORE	TCP		6944	svchost.exe		40.127.240.158	443	settings-win.data.microso...	MICROSOFT-CORP-MSN-AS-...	No Data
BEFORE	TCP		–	–		40.127.240.158	443	settings-win.data.microso...	MICROSOFT-CORP-MSN-AS-...	↑ 1 Kb ↓ 4 Kb
BEFORE	TCP		5488	MoUsoCoreWorker.exe		2.16.164.49	80	crl.microsoft.com	Akamai International B.V.	↑ 216 b ↓ 1 Kb
BEFORE	TCP		5488	MoUsoCoreWorker.exe		95.101.149.131	80	www.microsoft.com	Akamai International B.V.	↑ 209 b ↓ 1 Kb
BEFORE	TCP		–	–		40.127.240.158	443	settings-win.data.microso...	MICROSOFT-CORP-MSN-AS-...	↑ 888 b ↓ 4 Kb
BEFORE	TCP		–	–		40.127.240.158	443	settings-win.data.microso...	MICROSOFT-CORP-MSN-AS-...	↑ 860 b ↓ 4 Kb
BEFORE	TCP		–	–		40.127.240.158	443	settings-win.data.microso...	MICROSOFT-CORP-MSN-AS-...	↑ 1 Kb ↓ 18 Kb
BEFORE	UDP		4020	svchost.exe		239.255.255.250	1900	–	–	↑ 411 b ↓ –
1433 ms	TCP		816	svchost.exe		40.127.240.158	443	settings-win.data.microso...	MICROSOFT-CORP-MSN-AS-...	↑ 2 Kb ↓ 8 Kb
2542 ms	UDP		4	System		192.168.100.255	138	–	–	↑ 2 Kb ↓ –
8649 ms	TCP		4360	SearchApp.exe		2.23.209.130	443	th.bing.com	Akamai International B.V.	↑ 341 Kb ↓ 91 Kb
8653 ms	TCP		4360	SearchApp.exe		2.23.209.130	443	th.bing.com	Akamai International B.V.	↑ 448 b ↓ 4 Kb
8658 ms	TCP		4360	SearchApp.exe		192.229.221.95	80	ocsp.digicert.com	EDGECAST	↑ 234 b ↓ 578 b
8661 ms	TCP		4360	SearchApp.exe		2.23.209.130	443	th.bing.com	Akamai International B.V.	↑ 448 b ↓ 4 Kb
8663 ms	TCP		4360	SearchApp.exe		2.23.209.130	443	th.bing.com	Akamai International B.V.	↑ 448 b ↓ 4 Kb
9044 ms	TCP		624	svchost.exe		20.190.159.75	443	login.live.com	MICROSOFT-CORP-MSN-AS-...	↑ 38 Kb ↓ 11 Kb
9049 ms	TCP		624	svchost.exe		192.229.221.95	80	ocsp.digicert.com	EDGECAST	↑ 236 b ↓ 737 b
9054 ms	TCP		4360	SearchApp.exe		2.23.209.140	443	th.bing.com	Akamai International B.V.	↑ 13 Kb ↓ 642 Kb
9142 ms	TCP		624	svchost.exe		20.190.159.75	443	login.live.com	MICROSOFT-CORP-MSN-AS-...	↑ 65 Kb ↓ 12 Kb
11647 ms	TCP		780	svchost.exe		184.28.89.167	443	go.microsoft.com	AKAMAI-AS	↑ 2 Kb ↓ 7 Kb
11649 ms	TCP		780	svchost.exe		184.28.89.167	443	go.microsoft.com	AKAMAI-AS	↑ 2 Kb ↓ 7 Kb
11651 ms	TCP		780	svchost.exe		184.28.89.167	443	go.microsoft.com	AKAMAI-AS	↑ 2 Kb ↓ 7 Kb
11653 ms	TCP		780	svchost.exe		184.28.89.167	443	go.microsoft.com	AKAMAI-AS	↑ 2 Kb ↓ 7 Kb
11655 ms	TCP		780	svchost.exe		184.28.89.167	443	go.microsoft.com	AKAMAI-AS	↑ 3 Kb ↓ 7 Kb
11658 ms	TCP		780	svchost.exe		184.28.89.167	443	go.microsoft.com	AKAMAI-AS	↑ 2 Kb ↓ 7 Kb
11660 ms	TCP		780	svchost.exe		184.28.89.167	443	go.microsoft.com	AKAMAI-AS	↑ 3 Kb ↓ 7 Kb
28059 ms	TCP		6296	SIHClient.exe		4.245.163.56	443	slscr.update.microsoft.com	MICROSOFT-CORP-MSN-AS-...	↑ 752 b ↓ 3 Kb
28062 ms	TCP		6296	SIHClient.exe		184.30.21.171	80	www.microsoft.com	AKAMAI-AS	↑ 484 b ↓ 2 Kb
29059 ms	TCP		6296	SIHClient.exe		13.85.23.206	443	fe3cr.delivery.mp.microsof...	MICROSOFT-CORP-MSN-AS-...	↑ 602 b ↓ 3 Kb
29063 ms	TCP		6296	SIHClient.exe		4.245.163.56	443	slscr.update.microsoft.com	MICROSOFT-CORP-MSN-AS-...	↑ 584 b ↓ 3 Kb
29065 ms	TCP		6296	SIHClient.exe		4.245.163.56	443	slscr.update.microsoft.com	MICROSOFT-CORP-MSN-AS-...	↑ 750 b ↓ 3 Kb
30163 ms	TCP		5604	backgroundTaskHost...		20.223.36.55	443	arc.msn.com	MICROSOFT-CORP-MSN-AS-...	↑ 2 Kb ↓ 7 Kb
30168 ms	TCP		5604	backgroundTaskHost...		192.229.221.95	80	ocsp.digicert.com	EDGECAST	↑ 240 b ↓ 736 b
30172 ms	TCP		5604	backgroundTaskHost...		20.223.36.55	443	arc.msn.com	MICROSOFT-CORP-MSN-AS-...	↑ 1 Kb ↓ 7 Kb
34163 ms	TCP		6944	svchost.exe		52.137.106.217	443	settings-win.data.microso...	MICROSOFT-CORP-MSN-AS-...	↑ 2 Kb ↓ 4 Kb

Through its execution, 7 HTTP requests, 36 connections, and 19 DNS requests were made. The right-most figure is a list of said connections while the left-most provides the DNS requests. Most of these connections and requests are made out to seemingly harmless sites and domains such as Bing or other verified Microsoft sites - something that is normal and completely expected with Windows computers. The autonomous system numbers (ASN) column in the connections lists verify these connections as originating from Microsoft (specifically Microsoft Azure) and their partner company Akamai International B.V. These are just part and parcel with running any Windows system. However, there is only one such query that seems out of place from the rest of these.

Only one of these queries appears to be flagged as potentially malicious, that being “c2-7f000001.nip.io.” It is the fifth DNS request listed and occurs at between the 2529 and 2705 ms marks. Given that system calls, operations, and other queries are kept to a minimum throughout the malware’s execution, it can be inferred that something was uploaded or communicated rather than downloaded. In the case that something was downloaded, there would usually be extra activity in the CPU or RAM afterwards to set up or execute anything remotely downloaded. To the extent of the analysis through this tool, there is no concrete method of verifying legitimate RAM and CPU activity after the initial spike.

This same domain was also found during Static Analysis in Remnux, where in particular it can be found that the URL in question is specifically an upload link - something that corroborates the aforementioned inference. The contents of the upload are unknown, but it can be inferred that it has something to do with whatever details or specifications that the initial read of the host name and languages/locale that the malware had ascertained.

It is entirely possible that, as previously mentioned in the Remnux section of Basic Static Analysis, while the previous requests were for harmless websites, these queries could be to either pad out network traffic or for probing the network to verify if a valid connection to a test/example site can be made.

Appendix (Participation)

Toddeh Alexander: Ghidra Analysis, Static Analysis

Rocco Catalasan: Dynamic Analysis

Phu Lam: Static Analysis w/ Remnux, Format Documentation and Template

Wayne Muse: Static Analysis, Created project discord

Sokheng Teang: General Analysis (using Process explorer, Process Monitor, and run command prompt as administrator)