Malware Analysis

An investigation of Ryuk Ransomware

Stephen Broadbridge

CMP320: Advanced Ethical Hacking

2023/24

Note that Information contained in this document is for educational purposes.

.

Abstract

The presence of malware presents a major risk to computer systems and continues to escalate in complexity and scale. These strains of malicious software are not only more sophisticated an ever but are also used in various domains including but not limited to financial fraud, critical infrastructure sabotage and espionage. Ransomware attacks have become more common and “are now considered the most prevalent cyber security threat affecting businesses today” (Hassan, 2019). Due to this fact, countermeasures must be developed to mitigate the impact of these cyber-attacks. This includes the effective use of malware analysis techniques to analyse the multiple strains and types of malicious software so that they can be successfully countered.

This project is aimed at performing a malware analysis investigation on a malware sample to identify it and understand its inner workings.

The methodology used throughout the investigation was based on Lab Exercises and techniques taken from *Malware Analysis Techniques* (Barker, 2021). This includes using multiple techniques which include both dynamic and static to identify the malware sample's functionality and characteristics when interacting with an infected host machine.

The entire investigation was performed on a Windows 10 Virtual Machine with a Flare VM provided with all the necessary tools for analysing and reverse engineering the malware sample. All the results from the project were then analysed and documented within this report.

The investigation determined that the examined malware sample was a variant of the Ryuk ransomware. The analysis highlighted that the ransomware runs millions of operations simultaneously which enabled it to swiftly encrypt files on an infected host machine. Signs were also noted at its potential ability to spread across networked systems. Furthermore, the ransomware also disguised its malicious executables within PDF icons to trick users into activating its malicious payload.

The investigation showcased the impact that the Ryuk ransomware attack could have on any infected network which emphasises the critical importance of countermeasures.

.

Contents

.

.

1 INTRODUCTION

1.1 BACKGROUND

The integration of technology into almost every aspect of modern society has brought many benefits in terms of connectivity and near-instant access to information. However, with these benefits comes risk in the form of an increased attack surface for malware-based attacks.

In healthcare, the reliance on electronic health records and other digital tools to increase efficiency also exposes patient data to potential cyber threats. Healthcare is not the only industry or sector at risk. According to Statistica the number of malware-based attacks in 2023 were over 6 billion.

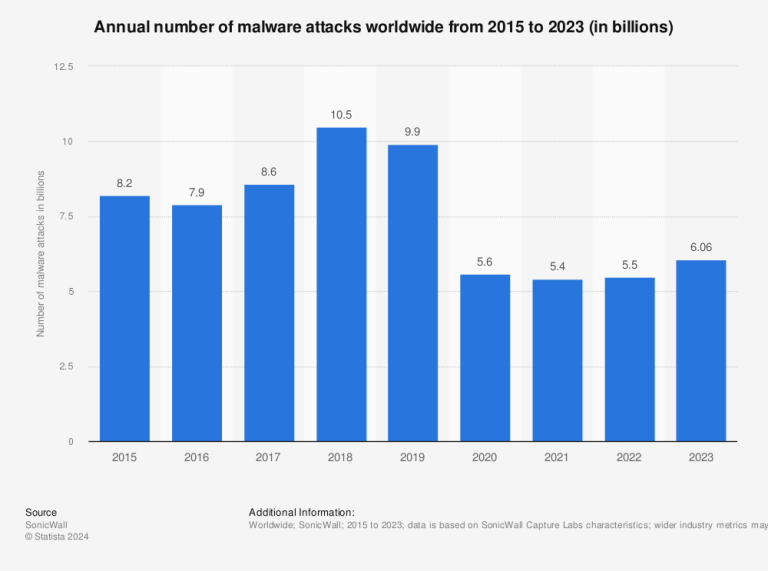

Figure 1-1 Number Malware Attacks Worldwide

To combat the threat of malware, organisations and individuals must implement a range of countermeasures. These include deploying malware detection systems and regularly updating and patching systems to prevent known vulnerabilities from being exploited. In addition to technical countermeasures, cyber security awareness and education for staff and individuals should be implemented as often “human beings are often the weak point that cybercriminals can exploit” (McNeal, 2023).

Understanding the need for countermeasures begins with an analysis of the malware itself. Malware analysis is typically broken into two distinct categories – static, dynamic and hybrid. Static analysis is looking for malicious intent by looking at the code structure without executing it. This method is particularly useful for initially identifying specific variations of malware but might not be sufficient to detect more sophisticated strains. Dynamic analysis consists of analysing malware running in a controlled and safe environment without the risk of harming live systems. Both methods provide useful insights into malware capabilities and impact.

1.2 AIM

The primary objective of this project is to perform a thorough analysis of a selected malware sample to identify its characteristics, capabilities, and potential impact on an infected host machine. The specific objectives of this analysis include:

- Utilise both static and dynamic analysis of the malware sample.
- Understand the behaviour and evasion techniques used by the malware.
- Document the methodology and results so findings from the investigation can be replicated.

2 PROCEDURE

2.1 OVERVIEW OF PROCEDURE

For this investigation, the malware sample chosen was number 3 from the 9 provided in the Flare VM within compressed folders.

2.1.1 Methodology

The methodology used throughout the investigation combines techniques from Lab Exercises for CMP320 Advanced Ethical Hacking module, *Malware Analysis Techniques* (Barker, 2021) and self-learning. This combination covers both Static and Dynamic analysis of the malware sample.

The steps of static analysis include identification techniques using hashing and antivirus tools, determining if the sample is obfuscated and obtaining further information about its functionality from PE headers, libraries that have been imported, functions and strings.

The methodology also looks at the steps for dynamic analysis which involves running the malware and analysing its behaviour.

2.1.2 Tools Used

- **HashMyFiles** – Calculates file hash values.
- **VirusTotal** – Analyses files and URL's for malware using multiple engines.
- **101Editior** – Edits text and binary data in large files
- **Strings.exe** – Extracts readable strings from binary files.
- **PiED** – Identifies obfuscation techniques in portable executable files.
- **Pestudio** – Assesses Windows applications for malware without executing them.
- **IDA** – Disassembles binary programs for reverse engineering.
- **FakeNet** – Simulates network traffic to analyse network communications.
- **Process Monitor** – Monitors in real-time file system, registry and process activity.
- **Process Explorer** – Displays detailed information about which files and directories open processes have.
- **OllyDbg** – A 32bit assembler level analysing debugger for Microsoft applications.

2.2 STATIC ANALYSIS

2.2.1 Signature Identification

The investigation must begin by generating a hash of the malware sample to check if it has been identified previously by other sources. Once the sample was extracted from the secure .zip folder a MD5 hash is created using the HashMyFiles tool. This is shown in Figure 2-1.

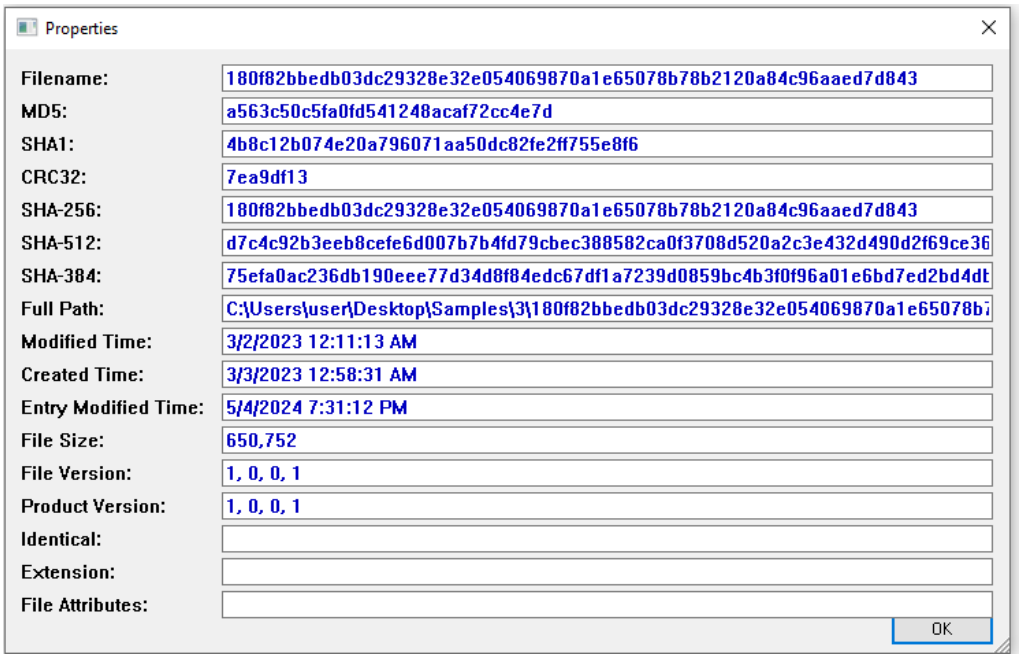

Figure 2-1 HashMyFiles

The hash that was obtained *(a563c50c5fa0fd541248acaf72cc4e7d)* was then immediately uploaded to VirusTotal. This scan VirusTotal scan revealed that this malware sample has been previously identified as the Ryuk ransomware. This information being revealed at this stage is a critical step in analysing the malwares behaviour. This means that the following stages of the investigation can be adjusted to gain a more in-depth understanding. The output from VirusTotal can be seen in Figure 2-2

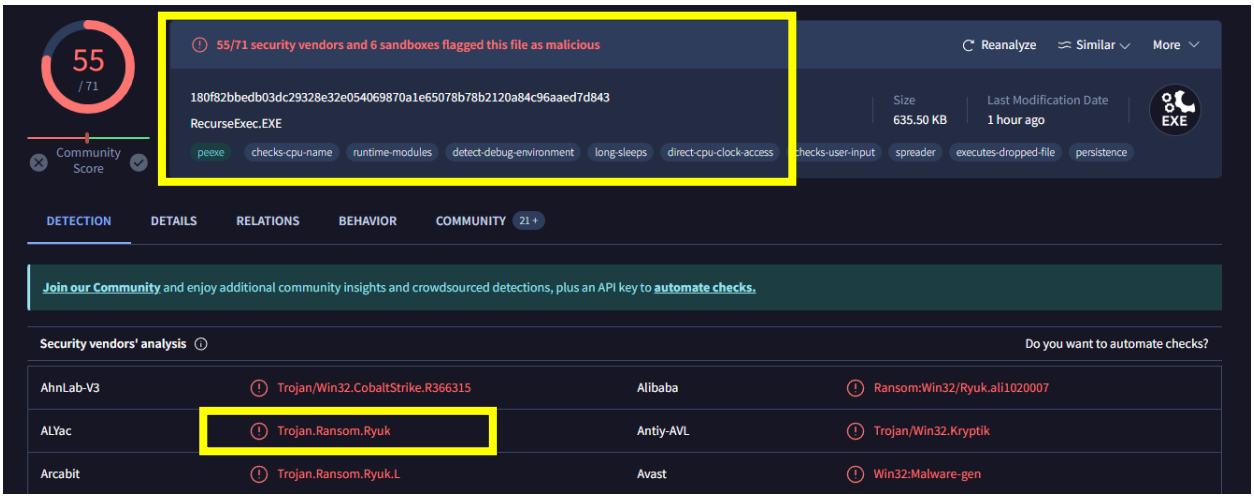

Figure 2-2 - VirusTotal Results

2.2.2 String Analysis

The strings of the executable can hold valuable information about what the file may do upon execution. Using the strings tool within Sysinternals, strings are extracted from the sample1 file using the following command.

```
strings.exe -n 5 <malicious file> > output_strings.txt
```

The command entered above will search for ANSI and Unicode strings in binary images and output the results into a text document.

The output_strings.txt document reveals several strings have been returned. This includes some Windows application programming interface modules.

Continuing to analyse the output_strings.txt file. It displays those strings such as 'KERNAL32.dll', 'USER32.dll and 'WS2_32.dll' indicate the malicious software may be utilizing Windows libraries.

Further analysis of the output_strings.txt file reveals the use of functions such as 'CryptDecrypt' which indicates data encryption for ransomware purposes. Figure 2-3 shows a fragment of the output from the output_strings.txt

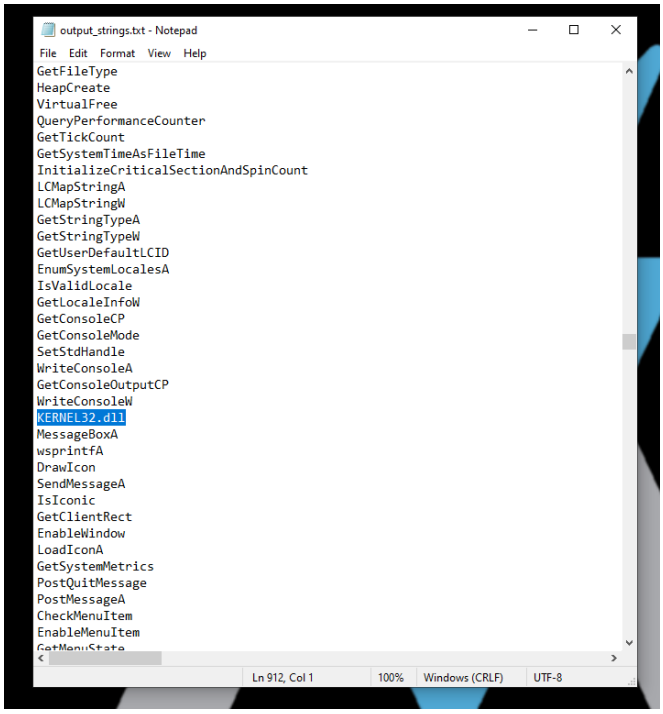

Figure 2-3 Strings Output

2.2.3 File Identification

The next step is of the investigation was to open the malware sample in 010Editior which will reveal the file signature This can be seen in the Figure 2-4. In this case, the file signature begins with MZ which indicates that file is using the DOS MZ executable file format typically used by Windows executables such as .exe files.

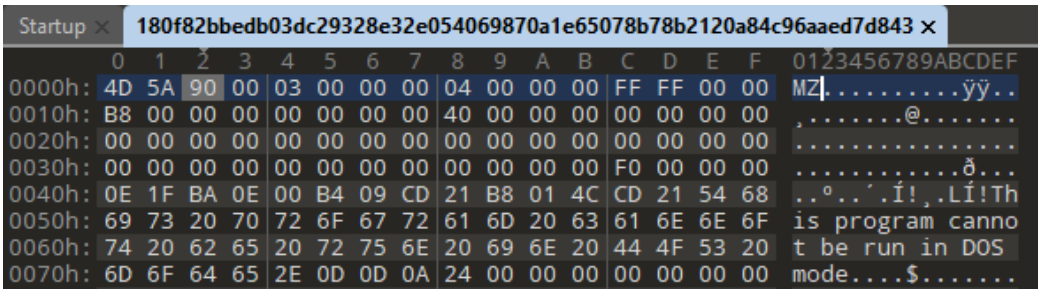

Figure 2-4 010Editor Analysis

Using this information the sample was then renamed to end with '.exe'.

2.2.4 Detecting Packers

No progress further into the investigation, it is important to identify if the malware is showing any indications of being packed. Utilising the PEiD tool the following output was produced:

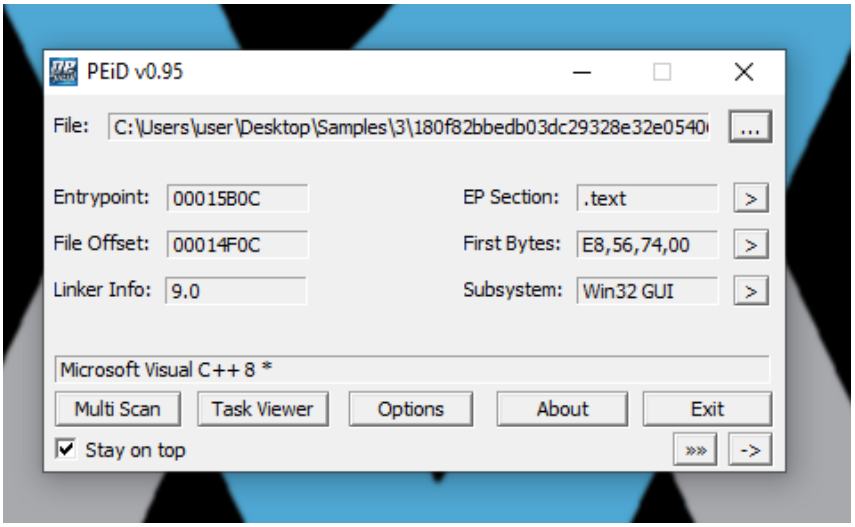

Figure 2-5 PiED Results

The output revealed that this malware sample is not packed, and the investigation can continue.

2.2.5 Libraries and Functions

Utilising Pestudio all libraries and functions were examined. Figure 2-6 shows a total of 10 libraries that this executable file imports.

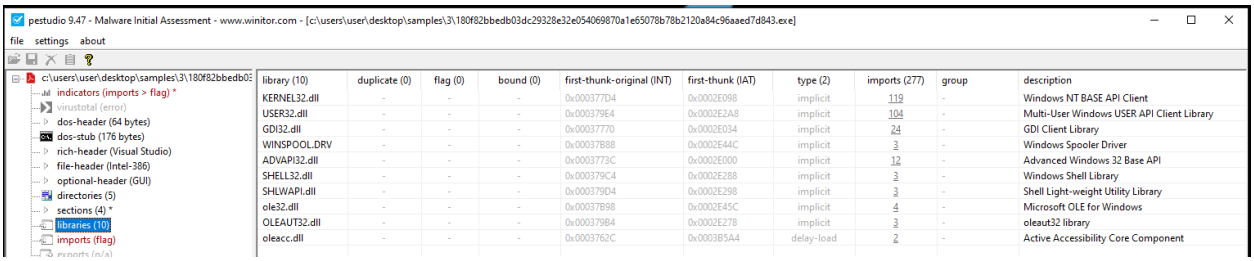

Figure 2-6 PEstudio Imported Libraries

In addition to these imported libraries, Pestudio revealed that 277 functions (see Appendix A – Imported Function Names) were identified. A snippet of the Pestudio output can be seen in the Figure 2-7.

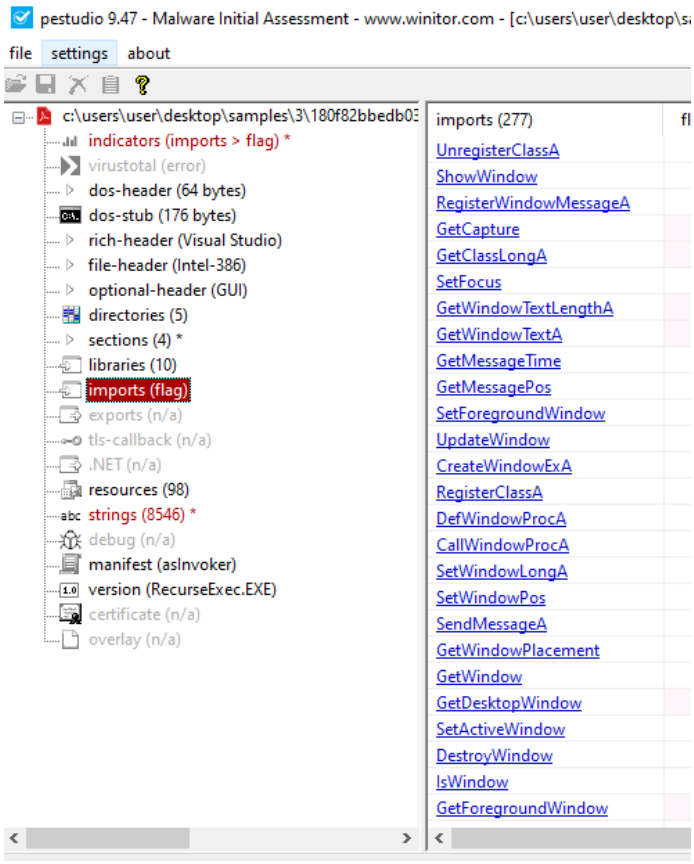

Figure 2-7 PEstudio Imported Functions

2.2.6 Code Analysis

The static code analysis was primarily done utilising the IDA tool provided. First, the starting/entry point was identified which can be seen in the Figure 2-8. Following this, the flow diagram created by the IDA disassembler was followed. The functions called were observed for their contents which provided an insight into the inner workings and capabilities of the ransomware.

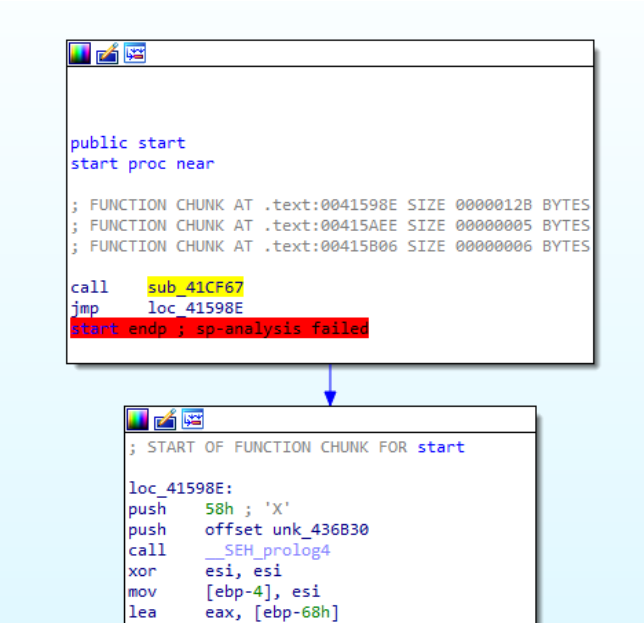

Figure 2-8 IDA Start Point

Utilising the search feature within the IDA disassembler tool, DLL functions and keywords. This allowed for quick and easy identification of the locations where these functions are called which in turn gave a much deeper understanding of how they interact within the code. The full breakdown and findings of this stage are documented in the Results section of this report.

2.3 DYNAMIC ANALYSIS

The dynamic analysis stage of the investigation involved running the malware sample provided in a safe environment. Before testing could begin, the testing environment was established using FakeNet for spoofing a network connect and opening Process Monitor(procmon.exe) and Process Explorer(procexp.exe), tools within Sysinternals to analyse the malware in real time.

2.3.1 Process Explorer

During the dynamic analysis phase, the behvior of the ransomware was monitored using Process Explored. It was observed that 'icals.exe' was utilising a considerable amount of CPU usage. The icals.exe executable was utilising the following command: 'icacls C:*" /grant Everyone:F /T /C /Q'. This command is designed to grand full access rights to all users silently affecting all files and directores from the root down. This can be seen in Figure 2-9.

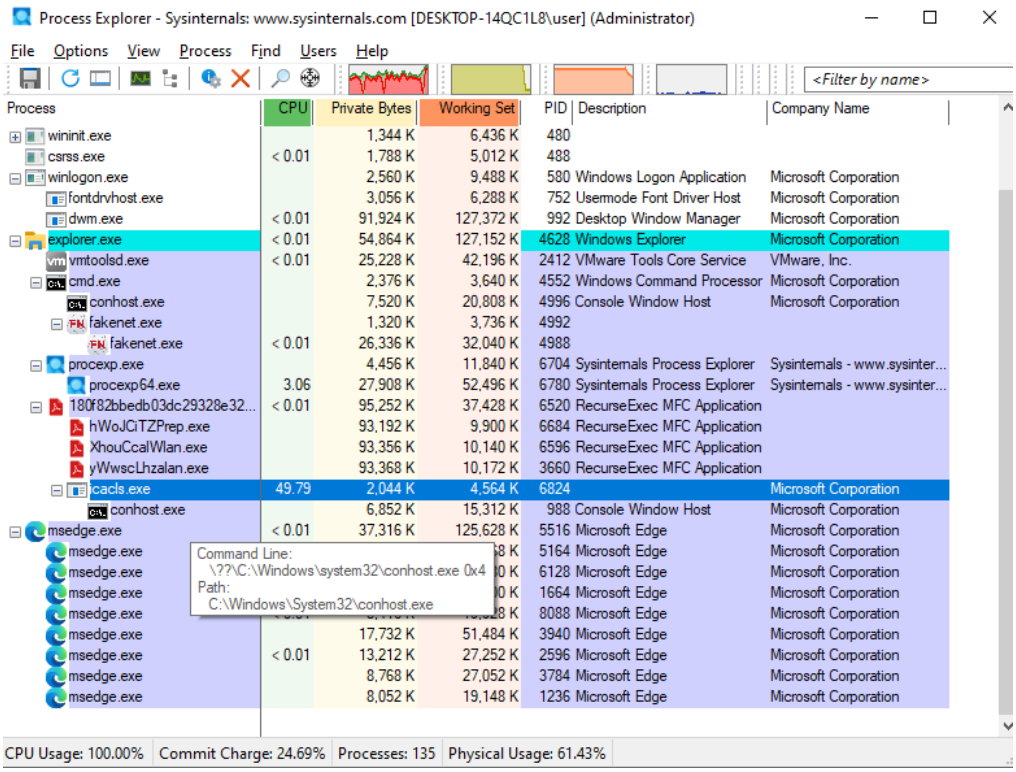

Figure 2-9 Process Explorer

It should be noted that within this phase of the investigation the malware rapidly escalated CPU usage to 100% during the encryption process which indicates a high computational demand. This is typical behaviour of ransomware as it will attempt to lock down the system and user files as quickly as possible to evade detection.

2.3.2 Process Monitor

The analysis of the malware sample through the Process Monitor tool revealed activity with millions of operations executed rapidly after the malware sample was executed which led to the Process Monitor tool crashing due to memory issues. These operations predominantly involved accessing specific directories to enumerate information and encrypt files which highlights the malware's approach to compromising user and system data.

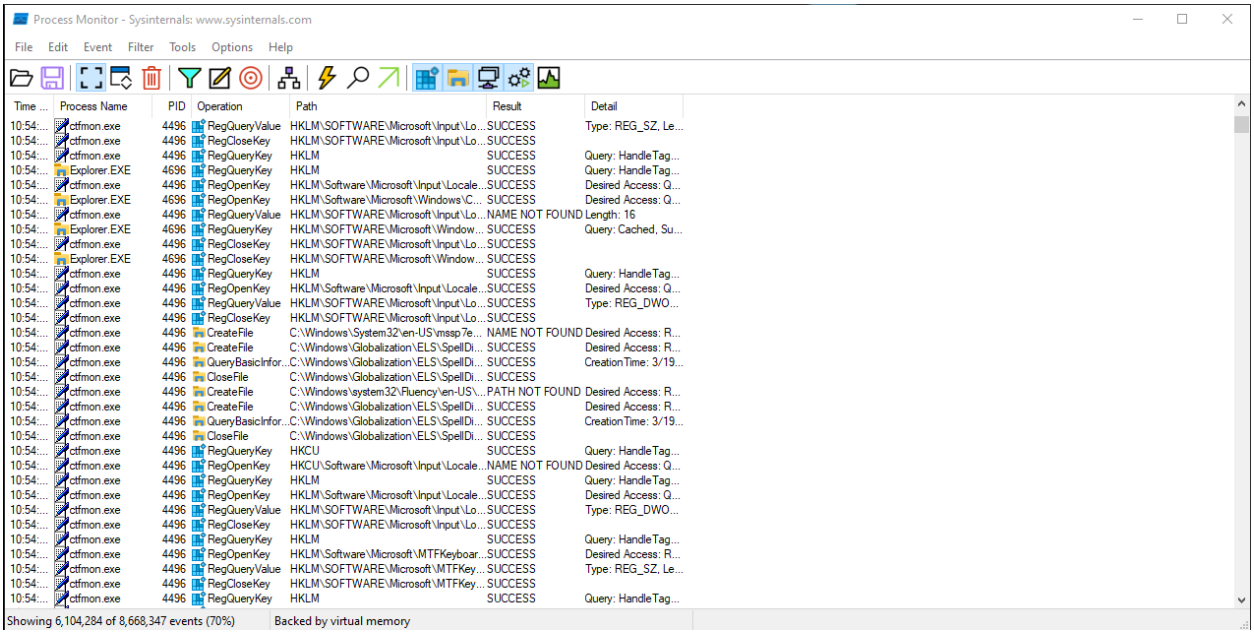

Figure 2-10 Process Monitor

The malware's repeated modifications to system settings and files suggest aggressive attempts to establish persistence and evade any detection countermeasures in place. Key activities observed was the manipulation of registry settings under 'HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer' which is typically associated with startup processes and user interface configurations.

2.3.3 Network Connection Monitoring

The final stage of the dynamic analysis consisted of analysing the network capabilities of the malware sample. Figure 2-1 was extracted from the FakeNet console and highlights an instance where the malware was executed from the entry directory and then made requests to multicast the IP address '244.0.0.22' and '244.0.0.2451'. This suggests potential network discovery or service identification aimed at local network components.

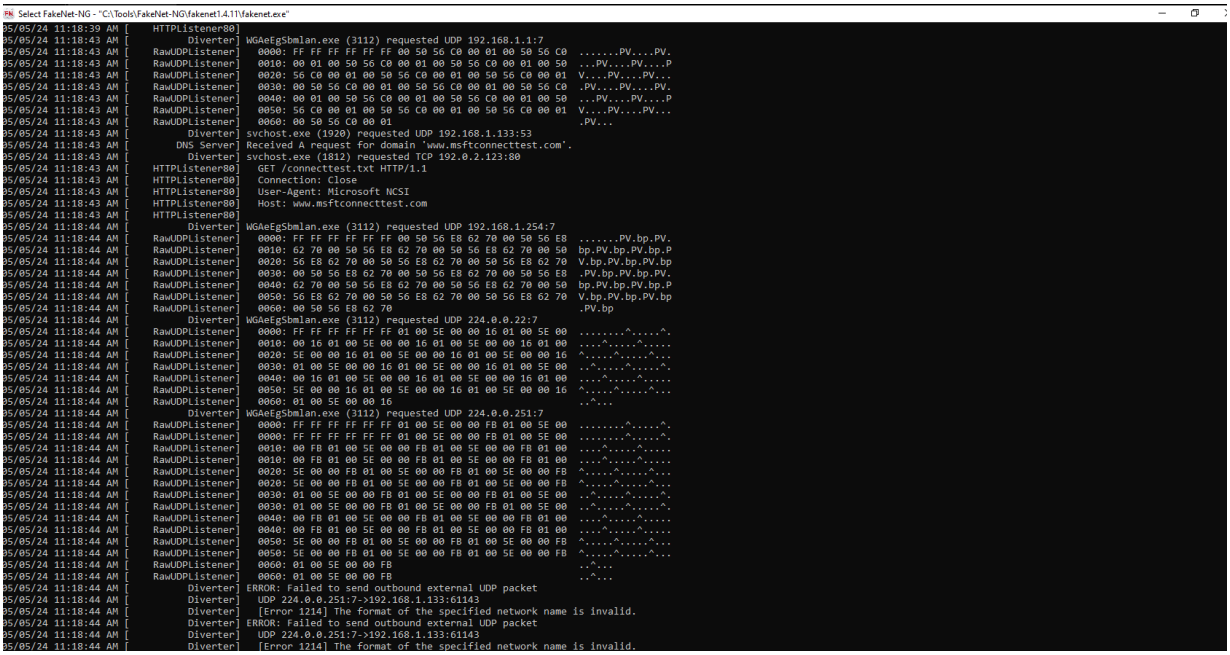

Figure 2-11 FakeNet Logs

2.3.4 Code Analysis

The debugger used for the dynamic code analysis was Ollydbg. Within the tools logs data option, it observed calls to 775 known and 1072 to guessed functions, 365 loops and 30 switches in total. This is shown in the figure below.

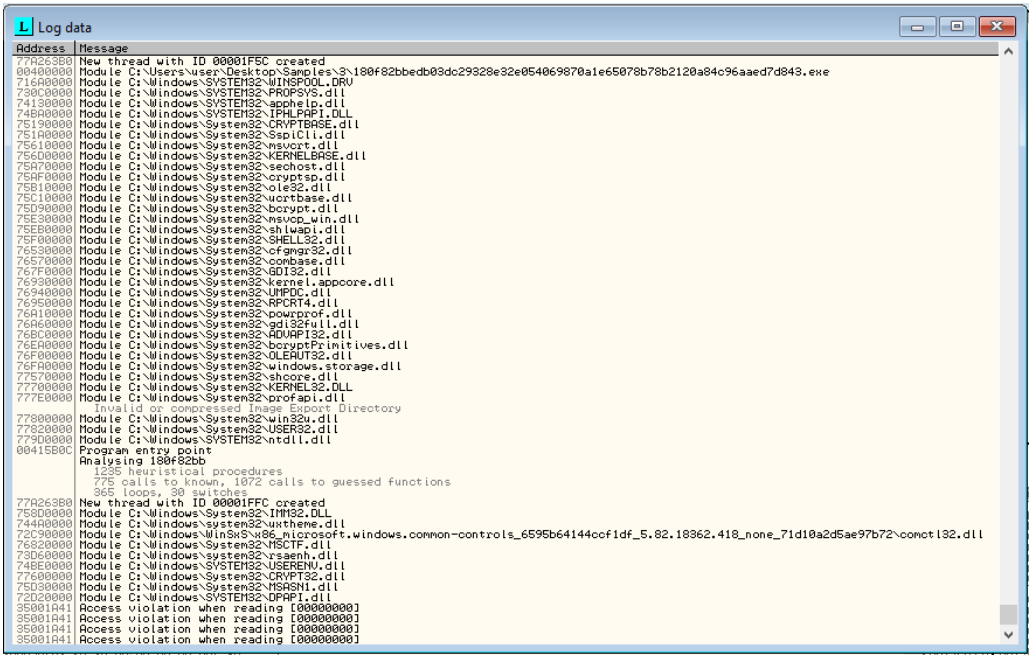

Figure 2-12 OllyDbg Log Data

Figure 2-12 also shows an Access Violation when reading error during the debugging process. This likely indicates the possibility of the malware sample contains anti-debugging techniques.

3 RESULTS

3.1 STATIC ANALYSIS

The results of the static analysis investigation revealed that the malware sample chosen was Ryuk ransomware.

As previously discussed, the file is a portable executable. When utilising the PiED tool it revealed that the malware sample was not packed and it was compiled using Microsoft Visual C++.

An interesting discovery was made when the sample was identified to be an executable file and then renamed to have '.exe' and the end. When this change was implemented, the icon used by the file changed to an Adobe icon, likely an attempt to trick users into executing the malware.

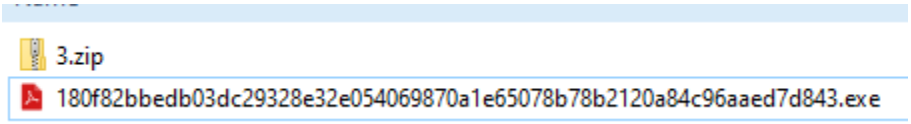

Figure 3-1 New Adobe Icon

Open further examination of the PE file within PEstudio, the compiler-stamp contained the information 'Thu Feb 11 02:51:01 2021 | UTC' and the file name 'RecurseExec.exe'. This matches with the information provided by VirusTotal to confirm the malware sample is Ryuk Ransomware.

Within Pestudio ten imported libraries were discovered. One library of which stood out was the the driver file 'WINSPOOL.DRV'. This contains information to interact with the print spooler. The imported functions (see Appendix A – Imported Function Names) revealed functions such as 'OpenPrinterA' and 'ClosePrinter'. This is likely to print off ransom notes from the printers like the Egregor variation of ransomware. "The Egregor ransomware uses a novel approach to get a victim's attention after an attack - shoot ransom notes from all available printers" (Abrams, 2020).

Another crucial imported library in the malware sample is ADVAPI32.dll, which provides API services related to Windows registry, encryption, and security. This DLL includes functions like 'CryptImportKey' and 'CryptEncrypt'. CryptEncrypt is used to encrypt victim data using imported keys, converting files into unreadable, encrypted formats.

3.2 DYNAMIC ANALYSIS

When the malware sample was initially launched, three new executables were added to the current directory. The names of these files changed when running the malware on multiple occasions indicating that these are likely randomly generated.

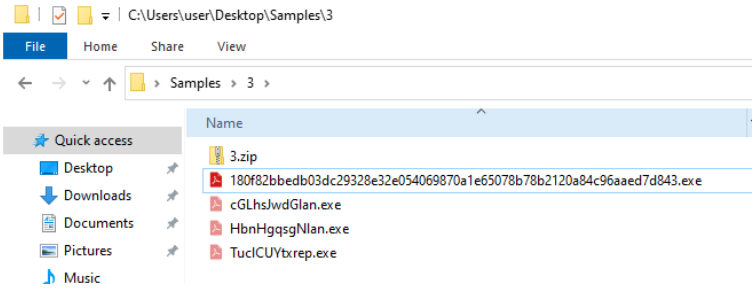

Figure 3-2 Three new excutables appear

In addition to these three new files was a HTML file named 'RyukReadMe.html' was created which could be found in the 'My Documents' directory. Clicking this brought up the default web browser and a page which is displayed in the Figure3-3.

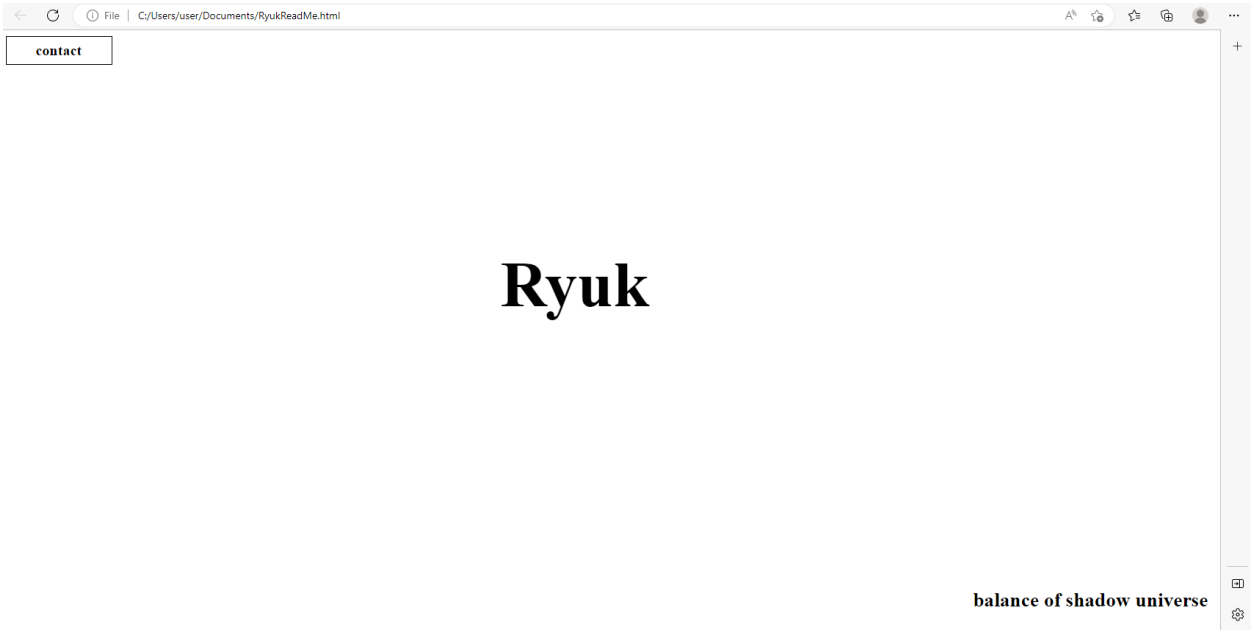

Figure 3-3 Ryuk Landing page

Upon clicking the contact button located to the top right of the screen, a new alert appears providing instructions to the user to access an onion site and to fill out a form with the given password. These details were not changed when running the malware on multiple occasions. The alert can be seen in Figure 3-4.

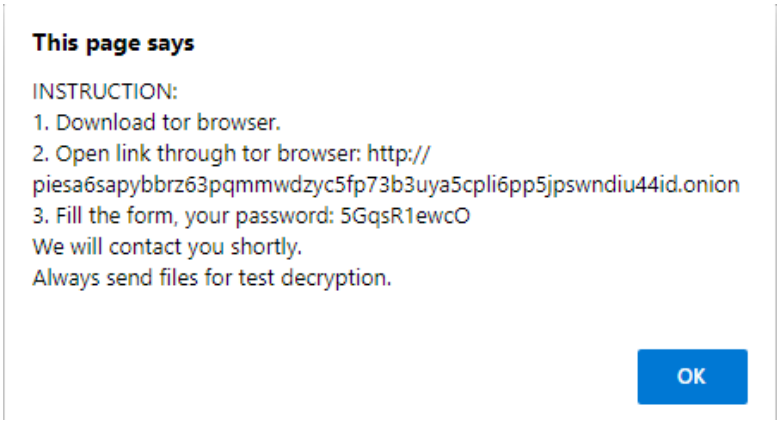

Figure 3-4 - Ryuk Alert Instructions

When analysing the processes of the malware sample the most significant finding was the excessive use of the icals.exe, an executable utilised for managing file and directory access control permissions. The specific command used was 'icacls C:*" /grant Everyone:F /T /C /Q' which modified security settings by granting full access rights to all users for every file and directory on the C drive, including subdirectories. The command utilised the '/C' option to continue on error and the '/Q' option to suppress any success messages. This is to any evade detection countermeasures.

In the final stage of the dynamic analysis, the network capabilities of the malware were examined to understand its interaction with networked environments. When the malware sample was launched from the entry directory it initiated requests to multicast IP addresses '244.0.0.22' and '244.0.0.2451' which were captured in the FakeNet Console. These indicate that the malware has capabilities of network discovery to propagate onto other machines within the network. This behavior matches the characteristic of Ryuk Ransomware according to SentinalOne "Ryuk is one of the first ransomware families to have the ability to identify and encrypt network drives and resources" (SentinalOne, 2024).

3.3 CODE ANALYSIS

3.3.1 Static Analysis

Analysing all function calls from the initial starting position reveals how the malware begins. Initially, the malware enumerates the current system time, process, and thread ID's, which may be used to generate unique identifiers or schedule other malicious attacks. This is shown in Figure 3-5. This information gathering step is critical as it helps the malware adapt its execution strategy.

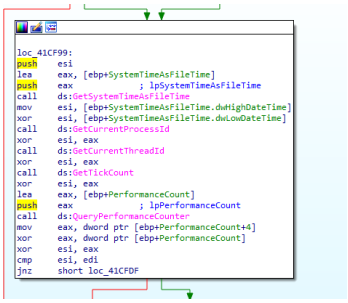

Figure 3-5 - Intial Enumartion Function

Continuing the exploration in IDA, the function 'loc_4015CC' is encountered which marks a shift in the malware behaviour from passive observation to active file manipulation. Here the malware utilised 'FindFirstFileA' to search for files within a current directory. This is the beginning of a larger scan across directories, potentially identifying targets for encryption. This is shown in Figure 3-6.

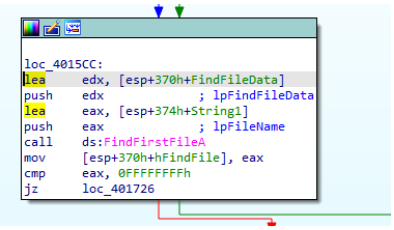

Figure 3-6 Directory Mapping Function

Following this stage the malware attempts to establish a foothold within the infected host by calling function 'loc_401B4B' which utilises cryptographic functions. It imports a new encryption key using 'CryptImportKey' for encryption tasks.

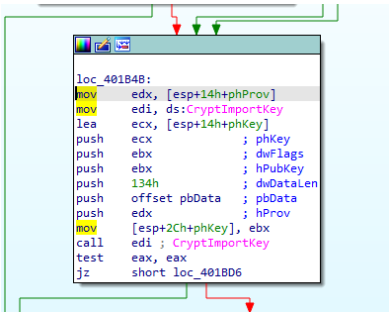

Figure 3-7 Cryptographic Functions

3.3.2 Dynamic Analysis

The dynamic analysis part of the investigation revealed that all the DLLs being used by the malware sample and other DLLs for function utility are being imported before the malware entry point is reached. This is shown in the Figure 3-8.

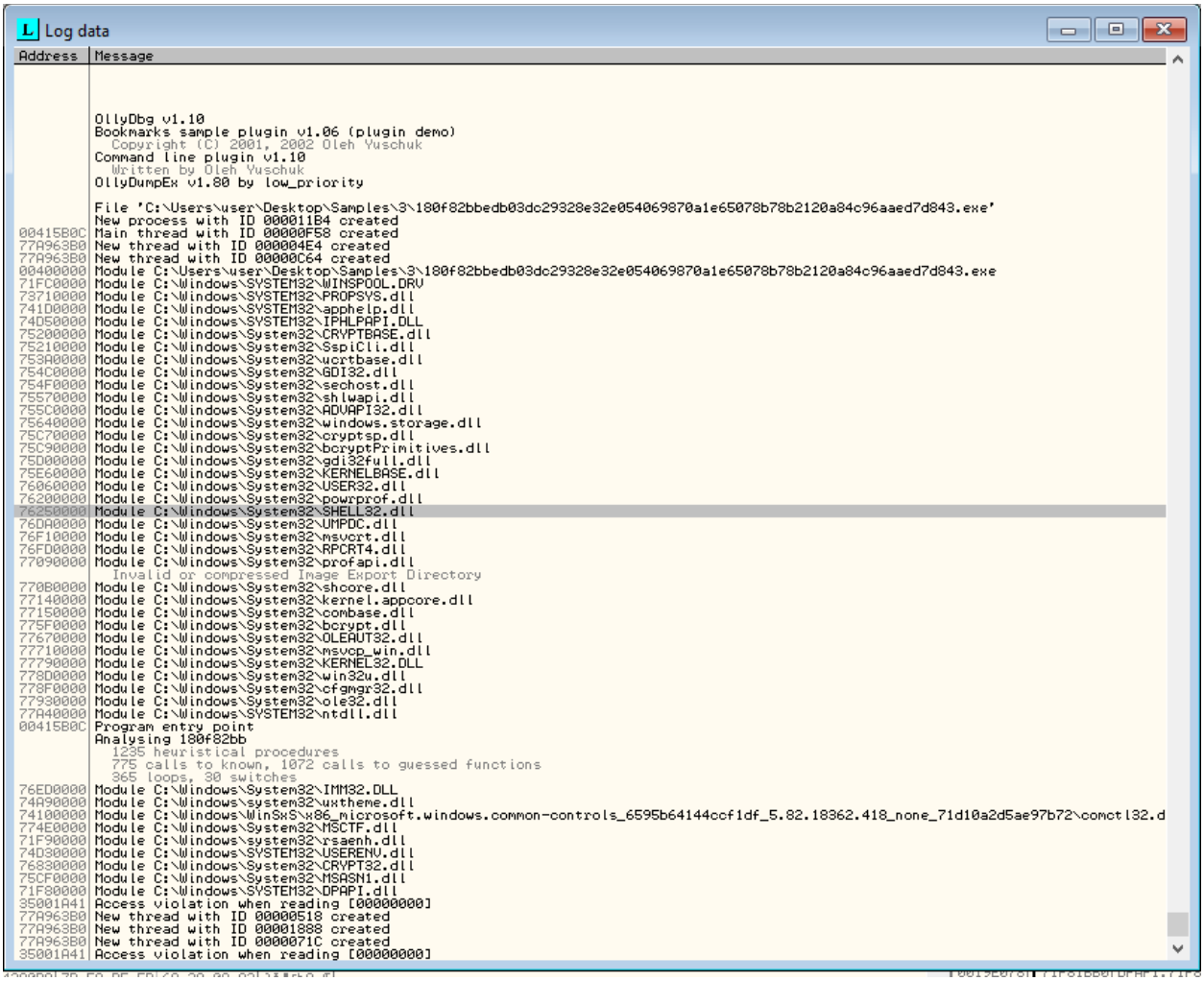

Figure 3-8 Log Data

During the debugging sessions an Access Violation error was continuously triggered. This error is significant as it suggests the presence of anti-debugging measures within the malware sample code. Access Violation at strategic points such as the one displayed in the Log Data, right at the beginning of the malware activation indicate a deliberate technique to hide its processes.

This behaviour is consistent with modern malware strains that employ different mechanisms to avoid detection, analysis, and mitigation.

4 DISCUSSION

4.1 GENERAL DISCUSSION

The tested sample was identified to a strain of the Ryuk ransomware. The Ryuk ransomware begin circulating in August 2018. The name originates from a character from the popular anime manga “Death Note” called Ryuk who allowed targeted killings.

Ryuk Ransomware primarily “targets high-profile organizations where they can find critical information that cripples the victim’s operations” (SentinalOne, 2024). Unlike other ransomware strains that cast a wide net and aim to infect as many machines as possible, Ryuk takes a dynamic approach to distribution where tactics are specifically crafted for the target victim. These tactics are typically broken down into three categories:

- Accessing an unprotected RDP port.
- Phishing campaigns to gain remote access.
- Deploying malicious email attachments and downloads to gain access to the network.

The identifying features of a system that has been infected with Ryuk Ransomware is that the encrypted files end with a ‘.RYK’ extension, like the ones found in this analysis.

This project has given an understanding on how the specifics of the Ryuk Ransomware works. Through the investigative process it was discovered that the Ransomware enumerates the system, uses icals.exe to elevate permissions then encrypts system data. To decrease the runtime of the encryption process and avoid detection, Ryuk runs millions of operations simultaneously.

Effective strategies to mitigate the risks of Ryuk ransomware and other ransomware variants are critical for maintaining cyber security for organisations and individuals. The implementation of anti-malware solutions and the consistent application of security updates are fundamental defences. It’s also important to monitor network activities to detect any animalities that could indicate a ransomware attack. Often Ransomware will linger within a network before deploying its payload. Regular backups of essential data ideally stored off-network are vital as they enable recovery from ransomware attacks and help mitigate the damage caused.

4.2 FUTURE WORK

The findings from this investigation have laid a solid foundation for a more in-depth analysis of the Ryuk ransomware. Moving forward an area of focus would be the detailed examination of the disassembled code to uncover deeper operational insights and potential vulnerabilities that could be exploited for defensive purposes. One aspect that would warrant further attention is the identification and analysis of any kill switches within the malware code.

Evidence of printing and network capabilities was also detected within the Ryuk ransomware. Further investigations are warranted to understand the extent of this malware's impact on networked environments. An effective method to achieve this would be to deploy an additional Windows host machine within a controlled network. This would allow an assessment of how the malware behaves when it has access to multiple connected systems.

REFERENCES

For URLs, Blogs:

Ojha, A. (2020) Static Malware Analysis: Identifying malware, Medium. Available at: https://theankitojha.medium.com/static-malware-analysis-identifying-malware-c8c7060799 (Accessed: 18 April 2024).

McNeal, A. (2023) More than half of ransomware infections are caused by phishing, Graphus. Available at: https://www.graphus.ai/blog/more-than-half-of-ransomware-infections-are-caused-by-phishing/ (Accessed: 20 April 2024).

Abrams, L. (2020) Egregor ransomware print bombs printers with ransom notes, BleepingComputer. Available at: https://www.bleepingcomputer.com/news/security/egregor-ransomware-print-bombs-printers-with-ransom-notes/#:~:text=The%20Egregor%20ransomware%20uses%20a,notes%20from%20all%20available%20printers. (Accessed: 28 April 2024).

What is ryuk ransomware?: A detailed breakdown (2024) SentinelOne. Available at: https://www.sentinelone.com/cybersecurity-101/ryuk-ransomware/?utm_source=google-paid&utm_medium=paid-search&utm_campaign=uki-bau-dsa&utm_term=&campaign_id=20072475253&ad_id=657024483078&gad_source=1&gclid=CjwKCAjw3NyxBhBmEiwAyofDYcAKhs6JNCmIxdWzfm5NvwxoTiek1rfF_2ezw0OnL6NqACarqghWBhoCyrYQAvD_BwE (Accessed: 03 May 2024).

(No date) Idapro cheatsheet. Available at: https://hex-rays.com/products/ida/support/freefiles/IDA_Pro_Shortcuts.pdf (Accessed: 25 April 2024).

(No date a) CLOUDOSCOPE: Detecting anti-forensic malware using Public Cloud Environments. Available at: https://dl.acm.org/doi/fullHtml/10.1145/3590777.3590793#:~:text=Malware%20uses%20anti%2Ddebugging%20and,to%20detect%20and%20evade%20debuggers. (Accessed: 22 April 2024).

For Books:

Barker, D. (2021) Malware analysis techniques tricks for the triage of adversarial software. Birmingham: Packt Publishing, Limited.

Sikorski, M. and Honig, A. (2012) Practical malware analysis: The hands-on guide to dissecting malicious software. San Francisco (California, EEUU): No Starch Press.

APPENDICES

Note that Appendices should be referenced in the main body of the text.

APPENDIX A

4.2.1 Imported Function Names

UnregisterClassA
ShowWindow
RegisterWindowMessageA
GetCapture
GetClassLongA
SetFocus
GetWindowTextLengthA
GetWindowTextA
GetMessageTime
GetMessagePos
SetForegroundWindow
UpdateWindow
CreateWindowExA
RegisterClassA
DefWindowProcA
CallWindowProcA
SetWindowLongA
SetWindowPos
SendMessageA
GetWindowPlacement
GetWindow
GetDesktopWindow
SetActiveWindow
DestroyWindow
IsWindow
GetForegroundWindow
EnableWindow
GetFocus
PeekMessageA
GetWindowLongA
GetLastActivePopup
IsWindowEnabled
GetMessageA
TranslateMessage
DispatchMessageA
GetActiveWindow
IsWindowVisible
CreateStdAccessibleObject
InitializeCriticalSectionAndSpinC
ount
InterlockedIncrement
DeleteCriticalSection
InitializeCriticalSection
EnterCriticalSection
LeaveCriticalSection
InterlockedDecrement
InterlockedExchange
WaitForSingleObject
WinHelpA
FreeResource
EnumResourceLanguagesA
FindResourceA
LoadResource
LockResource
SizeofResource
LoadCursorA
LoadIconA
WritePrivateProfileStringA
RegSetValueExA
RegCreateKeyExA
RegQueryValueA
RegOpenKeyA
RegEnumKeyA
RegDeleteKeyA
RegOpenKeyExA
RegQueryValueExA
RegCloseKey
GetStartupInfoA
IsDebuggerPresent
QueryPerformanceCounter
GetTickCount
GetUserDefaultLCID
EnumSystemLocalesA
GetVersionExA
GetCurrentProcessId
GetSystemMetrics
HeapAlloc
HeapFree
HeapReAlloc
HeapSize
HeapCreate
VirtualFree
GetStringTypeA
GetStringTypeW
LocalReAlloc
GlobalReAlloc
LocalAlloc
GlobalFlags
LocalFree
GlobalUnlock
GlobalFree
GlobalLock
GlobalAlloc
VirtualAlloc
SHGetMalloc
CoTaskMemFree
GetKeyState
UnhookWindowsHookEx
SetWindowsHookExA
CallNextHookEx
GetSystemTimeAsFileTime
FlushFileBuffers
CreateFileA
SetFilePointer
FindFirstFileA
FindNextFileA
FindClose
ReadFile
WriteFile
GetFileType
SHBrowseForFolderA
SHGetPathFromIDListA
PathFindFileNameA
PathRemoveFileSpecW
PathFindExtensionA
GetCommandLineA
Sleep
ExitProcess
FreeEnvironmentStringsA
GetEnvironmentStrings
FreeEnvironmentStringsW
GetEnvironmentStringsW
TlsFree
TlsSetValue
TlsAlloc
TlsGetValue
GetCurrentThread
GetCurrentThreadId
GetCurrentProcess

CreateProcessA
TerminateProcess
PostQuitMessage
PostMessageA
GetWindowThreadProcessId
RaiseException
UnhandledExceptionFilter
SetUnhandledExceptionFilter
GetModuleHandleW
GetModuleFileNameW
GetModuleFileNameA
LoadLibraryA
FreeLibrary
GetModuleHandleA
GetProcAddress
GetCursorPos
GlobalGetAtomNameA
GlobalFindAtomA
GlobalAddAtomA
GlobalDeleteAtom
CreatePipe
CryptAcquireContextW
CryptImportKey
CryptEncrypt
RtlUnwind
GetACP
IsValidCodePage
GetStdHandle
SetHandleCount
LCMapStringA
LCMapStringW
IsValidLocale
GetLocaleInfoW
GetConsoleCP
GetConsoleMode
SetStdHandle
WriteConsoleA
GetConsoleOutputCP
WriteConsoleW
SetErrorMode
GetOEMCP
GetCPInfo
GlobalHandle
lstrcmpW
FormatMessageA
MultiByteToWideChar
MulDiv
SetLastError
ConvertDefaultLocale
GetLocaleInfoA
WideCharToMultiByte
CompareStringA
lstrcpynA
lstrcatA
lstrcmpA
lstrcpyA
DuplicateHandle
GetLastError
lstrlenA
CloseHandle
DestroyMenu
GetSysColorBrush
WindowFromPoint
EndPaint
BeginPaint
ReleaseDC
GetDC
ClientToScreen
GrayStringA
DrawTextExA
DrawTextA
TabbedTextOutA
SetWindowTextA
IsDialogMessageA
SendDlgItemMessageA
GetClassNameA
SetPropA
GetPropA
RemovePropA
GetTopWindow
MapWindowPoints
SetMenu
GetClassInfoExA
GetClassInfoA
GetSysColor
AdjustWindowRectEx
ScreenToClient
CopyRect
PtInRect
GetDlgCtrlID
GetMenu
MessageBoxA
wsprintfA
DrawIcon
SystemParametersInfoA
GetWindowRect
GetMenuItemID
GetMenuItemCount
GetSubMenu
CreateDialogIndirectParamA
IsIconic
GetClientRect
CheckMenuItem
EnableMenuItem
GetMenuState
ModifyMenuA
GetParent
LoadBitmapA
GetMenuCheckMarkDimensions
SetMenuItemBitmaps
ValidateRect
GetDlgItem
GetNextDlgTabItem
EndDialog
SetCursor
DeleteDC
GetStockObject
ScaleWindowExtEx
SetWindowExtEx
RectVisible
ScaleViewportExtEx
SetViewportExtEx
OffsetViewportOrgEx
SetViewportOrgEx
SelectObject
Escape
ExtTextOutA
CreateBitmap
PtVisible
DeleteObject
SetMapMode
RestoreDC
SaveDC
GetObjectA
SetBkColor
SetTextColor
GetClipBox
GetDeviceCaps
TextOutA
DocumentPropertiesA
OpenPrinterA
ClosePrinter
CoCreateInstance
CoUninitialize
CoInitializeEx
9 (VariantClear)
12 (VariantChangeType)
8 (BSTR_UserUnmarshal)
LresultFromObject

4.2.2 Output_strings.txt

Note: Invalid strings have been removed from this output.

!This program cannot be run in DOS mode.
RichI
.text
`.rdata
@.data
.rsrc
CloseHandle
ReadFile error!
Close handle error!
Duplicate handle error 2!
Duplicate handle error!
Stdout pipe creation error!
Finished operating in the current directory "%s".
Will I continue into the (sub)directory(s) ?
Sub directory operation query
Finished operating in the current directory "%s".
Will I continue into the subdirectory(s) ?
bad allocation
Win32 Executables(*.exe)|*.exe||
RedirectedStream
You have not filled up essential entries.
These values are essential for me to function properly.
Try again.
Invalid field entries
Task Completed.
Recursive Execution completed sucessfully
ios_base::eofbit set
ios_base::failbit set
ios_base::badbit set
Here you can specify the (sub)directory under which to start recursing.
Enter the "current directory" environment to pass to the program. It's recommended to enter "<>" to indicate the current directory being recursed.
Leave it blank to operate in root directory too
A value of -1 implies Infinite wait state
Enter "<>" (without the quotes) where you want me to substitute the current directory into ..
Fuck Def
NTDLL.dll
CWinApp
Settings
PreviewPages
DeactivateActCtx
ActivateActCtx
ReleaseActCtx
CreateActCtxA
KERNEL32
Software\Microsoft\Windows\CurrentVersion\Policies\Explorer

NoRun
NoDrives
RestrictRun
NoNetConnectDisconnect
NoRecentDocsHistory
NoClose
Software\Microsoft\Windows\CurrentVersion\Policies\Network
NoEntireNetwork
Software\Microsoft\Windows\CurrentVersion\Policies\Comdlg32
NoPlacesBar
NoBackButton
NoFileMru
ntdll.dll
GetSystemDefaultUILanguage
GetUserDefaultUILanguage
kernel32.dll
%s%s.dll
%s (%s:%d)
%s (%s:%d)
Exception thrown in destructor
f:\dd\vctools\vc7libs\ship\atlmfc\src\mfc\appcore.cpp
CCmdTarget
CWinThread
Software\Classes\
Software\
CDialog
COleException
DISPLAY
CInvalidArgException
CNotSupportedException
CMemoryException
CSimpleException
CException
AfxWnd90s
AfxControlBar90s
AfxMDIFrame90s
AfxFrameOrView90s
AfxOleControl90s
AfxOldWndProc423
EnumDisplayDevicesA
GetMonitorInfoA
EnumDisplayMonitors
MonitorFromPoint
MonitorFromRect
MonitorFromWindow
GetSystemMetrics
USER32
accParent
accChildCount

accChild
accName
accValue
accDescription
accRole
accState
accHelp
accHelpTopic
accKeyboardShortcut
accFocus
accSelection
accDefaultAction
accSelect
accLocation
accNavigate
accHitTest
accDoDefaultAction
InitCommonControls
InitCommonControlsEx
HtmlHelpA
hhctrl.ocx
F#32768
f:\dd\vctools\vc7libs\ship\atlmfc\include\afxwin2.inl
commctrl_DragListMsg
CreateActCtxW
comctl32.dll
comdlg32.dll
shell32.dll
CEdit
CGdiObject
CPaintDC
CUserException
CResourceException
f:\dd\vctools\vc7libs\ship\atlmfc\include\afxwin1.inl
CFileDialog
p4GetOpenFileNameA
GetSaveFileNameA
SHCreateItemFromParsingName
Shell32.dll
CToolTipCtrl
tooltips_class32
CObject
Delete
NoRemove
ForceRemove
software
combobox
f:\dd\vctools\vc7libs\ship\atlmfc\src\mfc\auxdata.cpp
System

CMenu
CMapPtrToPtr
CArchiveException
CCommonDialog
commdlg_SetRGBColor
commdlg_help
commdlg_ColorOK
commdlg_FileNameOK
commdlg_ShareViolation
commdlg_LBSelChangedNotify
CMapStringToPtr
NotifyWinEvent
user32.dll
%2\CLSID
%2\Insertable
%2\protocol\StdFileEditing\verb\0
&Edit
%2\protocol\StdFileEditing\server
CLSID\%1
CLSID\%1\ProgID
CLSID\%1\InprocHandler32
ole32.dll
CLSID\%1\LocalServer32
CLSID\%1\Verb\0
&Edit,0,2
CLSID\%1\Verb\1
&Open,0,2
CLSID\%1\Insertable
CLSID\%1\AuxUserType\2
CLSID\%1\AuxUserType\3
CLSID\%1\DefaultIcon
%3,%7
CLSID\%1\MiscStatus
CLSID\%1\InProcServer32
CLSID\%1\DocObject
%2\DocObject
CLSID\%1\Printable
CLSID\%1\DefaultExtension
%9, %8
CByteArray
CObArray
CPtrArray
Unknown exception
CorExitProcess
mscoree.dll
kernel32.dll
HeapQueryInformation
bad exception
EncodePointer

KERNEL32.DLL
DecodePointer
FlsFree
FlsSetValue
FlsGetValue
FlsAlloc
LC_TIME
LC_NUMERIC
LC_MONETARY
LC_CTYPE
LC_COLLATE
LC_ALL
 !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~
runtime error
TLOSS error
SING error
DOMAIN error
R6034
An application has made an attempt to load the C runtime library incorrectly.
Please contact the application's support team for more information.
R6033
- Attempt to use MSIL code from this assembly during native code initialization
This indicates a bug in your application. It is most likely the result of calling an MSIL-compiled (/clr) function from a native constructor or from DllMain.
R6032
- not enough space for locale information
R6031
- Attempt to initialize the CRT more than once.
This indicates a bug in your application.
R6030
- CRT not initialized
R6028
- unable to initialize heap
R6027
- not enough space for lowio initialization
R6026
- not enough space for stdio initialization
R6025
- pure virtual function call
R6024
- not enough space for _onexit/atexit table
R6019
- unable to open console device
R6018
- unexpected heap error
R6017
- unexpected multithread lock error
R6016
- not enough space for thread data

This application has requested the Runtime to terminate it in an unusual way.
Please contact the application's support team for more information.
R6009
- not enough space for environment
R6008
- not enough space for arguments
R6002
- floating point support not loaded
Microsoft Visual C++ Runtime Library
<program name unknown>
Runtime Error!
Program:
(null)
(null)
(8PX
700WP
`h````
xpxxxx
('8PW
700PP
`h`hhh
xppwpp
e+000
GAIsProcessorFeaturePresent
SunMonTueWedThuFriSat
JanFebMarAprMayJunJulAugSepOctNovDec
 Complete Object Locator'
 Class Hierarchy Descriptor'
 Base Class Array'
 Base Class Descriptor at (
 Type Descriptor'
`local static thread guard'
`managed vector copy constructor iterator'
`vector vbase copy constructor iterator'
`vector copy constructor iterator'
`dynamic atexit destructor for '
`dynamic initializer for '
`eh vector vbase copy constructor iterator'
`eh vector copy constructor iterator'
`managed vector destructor iterator'
`managed vector constructor iterator'
`placement delete[] closure'
`placement delete closure'
`omni callsig'
 delete[]
 new[]
`local vftable constructor closure'
`local vftable'
`RTTI

`udt returning'
`copy constructor closure'
`eh vector vbase constructor iterator'
`eh vector destructor iterator'
`eh vector constructor iterator'
`virtual displacement map'
`vector vbase constructor iterator'
`vector destructor iterator'
`vector constructor iterator'
`scalar deleting destructor'
`default constructor closure'
`vector deleting destructor'
`vbase destructor'
`string'
`local static guard'
`typeof'
`vcall'
`vbtable'
`vftable'
operator
 delete
__unaligned
__restrict
__ptr64
__clrcall
__fastcall
__thiscall
__stdcall
__pascal
__cdecl
__based(
       (((((H
       h((((H
                          H
 !"#$%&'()*+,-./0123456789:;<=>?@abcdefghijklmnopqrstuvwxyz[\]^_`abcdefghijklmnopqrstuvwxyz{|}~
 !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`ABCDEFGHIJKLMNOPQRSTUVWXYZ{|
}~
HH:mm:ss
dddd, MMMM dd, yyyy
MM/dd/yy
December
November
October
September
August
April
March
February
January

Saturday
Friday
Thursday
Wednesday
Tuesday
Monday
Sunday
united-states
united-kingdom
trinidad & tobago
south-korea
south-africa
south korea
south africa
slovak
puerto-rico
pr-china
pr china
new-zealand
hong-kong
holland
great britain
england
czech
china
britain
america
swiss
swedish-finland
spanish-venezuela
spanish-uruguay
spanish-puerto rico
spanish-peru
spanish-paraguay
spanish-panama
spanish-nicaragua
spanish-modern
spanish-mexican
spanish-honduras
spanish-guatemala
spanish-el salvador
spanish-ecuador
spanish-dominican republic
spanish-costa rica
spanish-colombia
spanish-chile
spanish-bolivia
spanish-argentina
portuguese-brazilian

norwegian-nynorsk
norwegian-bokmal
norwegian
italian-swiss
irish-english
german-swiss
german-luxembourg
german-lichtenstein
german-austrian
french-swiss
french-luxembourg
french-canadian
french-belgian
english-usa
english-us
english-uk
english-trinidad y tobago
english-south africa
english-nz
english-jamaica
english-ire
english-caribbean
english-can
english-belize
english-aus
english-american
dutch-belgian
chinese-traditional
chinese-singapore
chinese-simplified
chinese-hongkong
chinese
canadian
belgian
australian
american-english
american english
american
Norwegian-Nynorsk
GetProcessWindowStation
GetUserObjectInformationA
GetLastActivePopup
GetActiveWindow
MessageBoxA
USER32.DLL
1#QNAN
1#INF
1#IND
1#SNAN

CONOUT$
string too long
invalid string position
bad cast
=L9o<
OLEACC.dll
CreateStdAccessibleObject
LresultFromObject
CloseHandle
lstrlenA
WriteFile
GetLastError
ReadFile
TerminateProcess
WaitForSingleObject
CreateProcessA
DuplicateHandle
GetCurrentProcess
CreatePipe
lstrcpyA
FindClose
FindNextFileA
lstrcmpA
FindFirstFileA
lstrcatA
lstrcpynA
SetFilePointer
CreateFileA
VirtualAlloc
GetProcAddress
GetModuleHandleA
FreeLibrary
GlobalAlloc
GlobalLock
InterlockedExchange
SizeofResource
LockResource
LoadResource
FindResourceA
CompareStringA
WideCharToMultiByte
LoadLibraryA
GetLocaleInfoA
GetModuleFileNameA
EnumResourceLanguagesA
ConvertDefaultLocale
GetCurrentThreadId
GetCurrentThread
GlobalDeleteAtom

GlobalAddAtomA
SetLastError
GetCurrentProcessId
FreeResource
GlobalFree
GlobalUnlock
MulDiv
MultiByteToWideChar
LocalFree
FormatMessageA
GetVersionExA
lstrcmpW
GlobalFindAtomA
GlobalGetAtomNameA
GetModuleFileNameW
InterlockedDecrement
WritePrivateProfileStringA
GlobalFlags
LocalAlloc
LeaveCriticalSection
TlsGetValue
EnterCriticalSection
GlobalReAlloc
GlobalHandle
InitializeCriticalSection
TlsAlloc
TlsSetValue
LocalReAlloc
DeleteCriticalSection
TlsFree
InterlockedIncrement
FlushFileBuffers
GetModuleHandleW
GetCPInfo
GetOEMCP
SetErrorMode
RtlUnwind
RaiseException
GetCommandLineA
GetStartupInfoA
HeapAlloc
HeapFree
Sleep
ExitProcess
HeapReAlloc
HeapSize
UnhandledExceptionFilter
SetUnhandledExceptionFilter
IsDebuggerPresent

GetACP
IsValidCodePage
GetStdHandle
FreeEnvironmentStringsA
GetEnvironmentStrings
FreeEnvironmentStringsW
GetEnvironmentStringsW
SetHandleCount
GetFileType
HeapCreate
VirtualFree
QueryPerformanceCounter
GetTickCount
GetSystemTimeAsFileTime
InitializeCriticalSectionAndSpinCount
LCMapStringA
LCMapStringW
GetStringTypeA
GetStringTypeW
GetUserDefaultLCID
EnumSystemLocalesA
IsValidLocale
GetLocaleInfoW
GetConsoleCP
GetConsoleMode
SetStdHandle
WriteConsoleA
GetConsoleOutputCP
WriteConsoleW
KERNEL32.dll
MessageBoxA
wsprintfA
DrawIcon
SendMessageA
IsIconic
GetClientRect
EnableWindow
LoadIconA
GetSystemMetrics
PostQuitMessage
PostMessageA
CheckMenuItem
EnableMenuItem
GetMenuState
ModifyMenuA
GetParent
GetFocus
LoadBitmapA
GetMenuCheckMarkDimensions

SetMenuItemBitmaps
ValidateRect
GetCursorPos
PeekMessageA
GetKeyState
IsWindowVisible
GetActiveWindow
DispatchMessageA
TranslateMessage
GetMessageA
CallNextHookEx
SetWindowsHookExA
SetCursor
IsWindowEnabled
GetLastActivePopup
GetWindowLongA
GetWindowThreadProcessId
EndDialog
GetNextDlgTabItem
GetDlgItem
IsWindow
DestroyWindow
CreateDialogIndirectParamA
SetActiveWindow
GetDesktopWindow
GetSubMenu
GetMenuItemCount
GetMenuItemID
GetWindow
GetWindowRect
GetWindowPlacement
SystemParametersInfoA
SetWindowPos
SetWindowLongA
GetMenu
CallWindowProcA
DefWindowProcA
GetDlgCtrlID
PtInRect
CopyRect
ScreenToClient
AdjustWindowRectEx
GetSysColor
RegisterClassA
GetClassInfoA
GetClassInfoExA
CreateWindowExA
UpdateWindow
SetForegroundWindow

SetMenu
MapWindowPoints
GetMessagePos
GetMessageTime
UnhookWindowsHookEx
GetTopWindow
GetForegroundWindow
GetWindowTextA
GetWindowTextLengthA
SetFocus
RemovePropA
GetPropA
SetPropA
GetClassNameA
GetClassLongA
GetCapture
WinHelpA
SendDlgItemMessageA
RegisterWindowMessageA
IsDialogMessageA
SetWindowTextA
ShowWindow
TabbedTextOutA
DrawTextA
DrawTextExA
GrayStringA
ClientToScreen
GetDC
ReleaseDC
BeginPaint
EndPaint
WindowFromPoint
LoadCursorA
GetSysColorBrush
DestroyMenu
UnregisterClassA
USER32.dll
CreateBitmap
GetDeviceCaps
GetClipBox
SetTextColor
SetBkColor
GetObjectA
SaveDC
RestoreDC
SetMapMode
DeleteObject
PtVisible
RectVisible

TextOutA
ExtTextOutA
Escape
SelectObject
SetViewportOrgEx
OffsetViewportOrgEx
SetViewportExtEx
ScaleViewportExtEx
SetWindowExtEx
ScaleWindowExtEx
DeleteDC
GetStockObject
GDI32.dll
ClosePrinter
DocumentPropertiesA
OpenPrinterA
WINSPOOL.DRV
CryptEncrypt
CryptImportKey
CryptAcquireContextW
RegCloseKey
RegQueryValueExA
RegOpenKeyExA
RegDeleteKeyA
RegEnumKeyA
RegOpenKeyA
RegQueryValueA
RegCreateKeyExA
RegSetValueExA
ADVAPI32.dll
SHGetPathFromIDListA
SHBrowseForFolderA
SHGetMalloc
SHELL32.dll
PathFindExtensionA
PathFindFileNameA
PathRemoveFileSpecW
SHLWAPI.dll
CoTaskMemFree
CoUninitialize
CoCreateInstance
CoInitializeEx
ole32.dll
OLEAUT32.dll
.?AVCGetFileList@@
.?AVCRecurseExecApp@@
.?AVCWinApp@@
.?AVCWinThread@@
.?AVCCmdTarget@@

.?AVCObject@@
^2?SnrwkZSzpuPI
1.HKe
.?AVbad_alloc@std@@
.?AVexception@std@@
.?AVCEdit@@
.?AVCWnd@@
.?AVCRecurseExecDlg@@
.?AVCDialog@@
.?AVruntime_error@std@@
.?AVfailure@ios_base@std@@
.PAVCException@@
.?AVCCmdUI@@
.PAVCMemoryException@@
.?AVCOleException@@
.?AVCException@@
.PAVCOleException@@
.PAVCObject@@
.PAVCSimpleException@@
.PAVCNotSupportedException@@
.PAVCInvalidArgException@@
.?AVCSimpleException@@
.?AVCMemoryException@@
.?AVCNotSupportedException@@
.?AVCInvalidArgException@@
.?AVXAccessible@CWnd@@
.?AVXAccessibleServer@CWnd@@
.?AVCTestCmdUI@@
.?AV_AFX_HTMLHELP_STATE@@
.?AVCNoTrackObject@@
.PAVCUserException@@
.?AV?$IAccessibleProxyImpl@VCAccessibleProxy@ATL@@@ATL@@
.?AUIAccessible@@
.?AUIDispatch@@
.?AUIUnknown@@
.?AUIAccessibleProxy@@
.?AV?$CMFCComObject@VCAccessibleProxy@ATL@@@@
.?AVCAccessibleProxy@ATL@@
.?AV?$CComObjectRootEx@VCComSingleThreadModel@ATL@@@ATL@@
.?AVCComObjectRootBase@ATL@@
.?AUIOleWindow@@
.?AV_AFX_THREAD_STATE@@
.?AVAFX_MODULE_THREAD_STATE@@
.?AVAFX_MODULE_STATE@@
.?AVCDllIsolationWrapperBase@@
.?AVCComCtlWrapper@@
.?AVCCommDlgWrapper@@
.?AVCShellWrapper@@
.?AV_AFX_BASE_MODULE_STATE@@

.?AVCAfxStringMgr@@
.?AUIAtlStringMgr@ATL@@
.PAVCResourceException@@
.?AVCResourceException@@
.?AVCUserException@@
.?AVCGdiObject@@
.?AVCDC@@
.?AVCPaintDC@@
.?AVCCommonDialog@@
.?AVXFileDialogEvents@CFileDialog@@
.?AUIFileDialogEvents@@
.?AVXFileDialogControlEvents@CFileDialog@@
.?AUIFileDialogControlEvents@@
.?AVCFileDialog@@
.?AVCToolTipCtrl@@
.?AUCThreadData@@
.?AVCHandleMap@@
.?AVCMenu@@
.?AVCMapPtrToPtr@@
.PAVCArchiveException@@
.?AVCArchiveException@@
.?AVCMapStringToPtr@@
.?AVCObArray@@
Apartment
.?AVCByteArray@@
.?AV?$CArray@W4LoadArrayObjType@CArchive@@ABW412@@@
.?AVCPtrArray@@
.?AVtype_info@@
.?AVbad_cast@std@@
.?AVbad_exception@std@@

abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ

abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ
z?aUY
zc%C1
-64OS
.?AVlogic_error@std@@
.?AVlength_error@std@@
.?AVout_of_range@std@@
.?AVfacet@locale@std@@
.?AUctype_base@std@@
.?AVios_base@std@@
.?AV?$_Iosb@H@std@@
.?AV?$basic_ostream@DU?$char_traits@D@std@@@std@@
.?AV?$basic_ios@DU?$char_traits@D@std@@@std@@
.?AV?$ctype@D@std@@

.?AV?$basic_streambuf@DU?$char_traits@D@std@@@std@@
.?AV?$basic_filebuf@DU?$char_traits@D@std@@@std@@
.?AVcodecvt_base@std@@
.?AV?$codecvt@DDH@std@@
.?AV_Locimp@locale@std@@
Copyright (c) 1992-2004 by P.J. Plauger, licensed by Dinkumware, Ltd. ALL RIGHTS RESERVED.
A Kamal Shankar Quick Tool - Recursive Executer
MS Sans Serif
Cancel
Program options
Redirect STDIO
Check2
Confirm new entry
Check this if you want the program to ask for confirmation everytime it enters a new (sub)directory..
Execution Options
&Browse..
Select the program you want to execute
Select the starting directory
Browse ..
Executable arguments
Directory Wildcard
Timeout (ms)
Current Dir Value
MS Shell Dlg
&New
Cancel
&Help
MS Shell Dlg
Save As
All Files (*.*)
Untitled
an unnamed file
&Hide
No error message is available.#Attempted an unsupported operation.$A required resource was unavailable.
Out of memory.
An unknown error has occurred.!Encountered an improper argument.
Incorrect filename.
Failed to open document.
Failed to save document.
Save changes to %1? Failed to create empty document.
The file is too large to open.
Could not start print job.
Failed to launch help.
Internal application error.
Command failed.)Insufficient memory to perform operation.PSystem registry entries have been removed and the INI file (if any) was deleted.BNot all of the system registry entries (or INI file) were removed.FThis program requires the file %s, which was not found on this system.tThis program is linked to the missing export %s in the file %s. This machine may have an incompatible version of %s.
Enter an integer.

Enter a number.#Enter an integer between %1 and %2.!Enter a number between %1 and %2.!Enter no more than %1 characters.
Select a button.#Enter an integer between 0 and 255.
Enter a positive integer.
Enter a date and/or time.
Enter a currency.
Enter a GUID.
Enter a time.
Enter a date.
Unexpected file format.O%1
Cannot find this file.
Verify that the correct path and file name are given.
Destination disk drive is full.5Unable to read from %1, it is opened by someone else.AUnable to write to %1, it is read-only or opened by someone else.1Encountered an unexpected error while reading %1.1Encountered an unexpected error while writing %1.
%1: %2
Continue running script?
Dispatch exception: %1
#Unable to read write-only property.#Unable to write read-only property.
#Unable to load mail system support.
Mail system DLL is invalid.!Send Mail failed to send message.
No error occurred.-An unknown error occurred while accessing %1.
%1 was not found.
%1 contains an incorrect path.8Could not open %1 because there are too many open files.
Access to %1 was denied.0An incorrect file handle was associated with %1.8Could not remove %1 because it is the current directory.2Could not create %1 because the directory is full.
Seek failed on %14Encountered a hardware I/O error while accessing %1.3Encountered a sharing violation while accessing %1.3Encountered a locking violation while accessing %1.
Disk full while accessing %1.$Attempted to access %1 past its end.
No error occurred.-An unknown error occurred while accessing %1.%Attempted to write to the
reading %1.$Attempted to access %1 past its end.&Attempted to read from the writing %1.
%1 has a bad format."%1 contained an unexpected object. %1 contains an incorrect schema.
pixels
Uncheck
Check
Mixed
VS_VERSION_INFO
StringFileInfo
040904B0
CompanyName
FileDescription
RecurseExec MFC Application
FileVersion
1, 0, 0, 1
InternalName
RecurseExec
LegalCopyright
Copyright (C) 2004
LegalTrademarks

OriginalFilename
RecurseExec.EXE
ProductName
RecurseExec Application
ProductVersion
1, 0, 0, 1
VarFileInfo
Translation
<assembly xmlns="urn:schemas-microsoft-com:asm.v1" manifestVersion="1.0">
  <trustInfo xmlns="urn:schemas-microsoft-com:asm.v3">
    <security>
      <requestedPrivileges>
        <requestedExecutionLevel level="asInvoker" uiAccess="false"></requestedExecutionLevel>
      </requestedPrivileges>
    </security>
  </trustInfo>
</assembly>PAPADDINGXXPADDINGPADDINGXXPADDINGPADDINGXXPADDINGPADDINGXXPADDINGPADDINGXXP
ADDINGPADDINGXXPADDINGPADDINGXXPADDINGPADDINGX