|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name of malware | Basic Static | Basic Dynamic | Advanced Static | Advanced Dynamic |
| Ryuk Ransomware | ✔ (mabel) | ✔ (edward) | ✔ (edward) | Failed(mabel) |
| Dm.exe | ✔(edward) | Failed(mabel) | ✔(mabel) |  |
| Keylogger |  | ✔(mabel) |  |  |

## 3.2 Second Malware

0670d2681210a087440bf7fb1c0f541a5c5ea7a58eb5c36371965fdcb4b46cf5 (dm.exe)

### 3.2.1 Results from Basic Static Analysis

#### 3.2.1.1 Virus Total

We first placed the malware in the website, “VirusTotal” to gain a better understanding of what the malware does. From the results shown, we can see that many antivirus programs managed to detect the signature of this malware and many of them flagged it to be an adware. This gives us an idea of what the malware does which helps in our process of understanding what the malware will be trying to achieve and perform.

# Graphical user interface, application Description automatically generated

We also went to the “Details” subpage so as to see the hash values produced by the malware in the section “Basic Properties”. This can then be used to cross-refer to the result produced in section 3.2.1.2 to ensure the integrity of the malware.

Graphical user interface, text, application, email

Description automatically generated

We scrolled further down in the “Details” subpage and came across the section “History”. This section shows the creation of the malware which can be also used to cross-refer to the result of section 3.2.1.4.

Graphical user interface, text, application, email

Description automatically generated

#### 

#### 3.2.1.2 WinMD5

We hashed the malware using WinMD5 to ensure that the malware that is being used is the correct malware. This is used to check for the integrity of the application to ensure that the malware has not been modified when transferring to the virtual machine. This MD5 hash value can be compared to the results produced from the section in 3.2.1.1.

Table

Description automatically generated

Since the hash digests are an exact match, it means that the malware is accurate and has not been tampered with and thus further analysis can continue.

#### 

#### 3.2.1.3 PEID

After ensuring the malware is correct, we then place the malware sample in PEID to check to see if the malware is packed. As we can see, the output from at the bottom shows that the malware is packed by UPX. In order to continue the malware analysis, we have to unpack it.

Graphical user interface, application

Description automatically generated

PEID has its own plugins to unpack certain malware packers. Fortunately, PEID has an unpacker for UPX already installed and thus we can unpack the malware without going through tedious processes.

Graphical user interface, application

Description automatically generated

After unpacking the malware, we can see that unpacked.exe was added to the folder.

Graphical user interface, application

Description automatically generated

#### 3.2.1.4 PEview

Graphical user interface, text, application, email

Description automatically generated

We then added unpacked.exe into a tool called, “PEview” to help us identify some metadata related to the application. We can see that the date and time of creation was set to be “2031/05/20 Tue 20:45:19 UTC”. We suspect that there was an error when the image was created as the year of creation stated 2031. Thus, we compared the creation date of the malware shown in section 3.2.1.1 and it made much more sense if the malware was created in “2008-12-15”.

#### 3.2.1.5 Dependency Walker

The first library in Dependency Walker is ADVAPI32.DLL. This library is a system process that is needed in order for the computer to work properly. It is part of an advanced API services library that supports numerous APIs including many security and registry calls. It is responsible for things like the Windows registry, restarting and shutting down the system, starting/stopping and creating windows services and managing user accounts. Some suspicious functions that are imported and may be used by the malware include, “RegCreateKeyExA”, “RegDeleteKeyA”, “RegSetValueExA”. This means that the malware may be modifying the registry value in the system.

## Table Description automatically generated

The next library shown in the dependency walker is GDI32.DLL. This library contains functions for Windows Graphical Device Interface (GDI). It performs primitive drawing functions for output to video displays and printers. This allows Windows to create simple 2D objects and applications called the GDI functions to perform several actions such as low-level drawing and text output. Some suspicious imported functions are, “CreateCompatibleBitmap”, “DeleteObject”, “GetObjectA”. This means the malware may modify and create objects to be displayed. A bitmap of the system can also be created which suggests that data may be encoded or hidden into files.

Table

Description automatically generated with medium confidence

The third library shown is KERNEL32.DLL. It is a Windows kernel module and is a 32-bit dynamic link library that is used in Windows operating systems. It runs in the background and carries out important functions like memory management, and input and output operations. Some suspicious imports from this library are, “CopyFileA”, “CreateFileA”, “CreateMutexA”, and “DeleteFileA”. This suggests that the malware may make changes to the file system through deletion and addition of files.

Table

Description automatically generated

KERNEL32.DLL had imported a lot of functions and while scrolling through the imported functions, we noticed an important function that was imported. This function is identified to be “IsDebuggerPresent” which is a function that has the ability to determine if the process is being debugged by a user-mode debugger. If the process has been identified to be debugged by a user-mode debugger, it modifies the behavior of the process and whatever results achieved from the debugger will be inaccurate. This means that the malware prevents the usage of advanced dynamic analysis unless this function is removed.

Table

Description automatically generated

The fourth library is OLEAUT32.DLL which is a library that contains core Object Linking Embedding (OLE) functions. It facilitates communication of meaningful data between software applications and allows applications to handle files and information created by other applications. Some suspicious functions include, “SysAllocString” and “LoadTypeLib”. This may imply that the malware may modify strings in the system and could also be loading other harmful libraries into the system.

Table

Description automatically generated

The fifth library is SHELL32.DLL and it contains Windows Shell API functions. This library is used when web pages and files are opened. It provides many functions of the Windows Shell. Some suspicious functions identified are, “ShBrowseForFolder”, “SHGetSpecialFolderLocation”, and “ShellExecuteA”. This can mean the malware may search for certain files in the system and may execute certain shell scripts stored in these files.

Table

Description automatically generated

The sixth library is USER32.DLL which is a library for Windows management functions for message handling, timers, menus, and communications. It contains Windows API functions related to Windows user interface. Some suspicious functions include “CreateWindowExA”, “CallWindowProcA”, and “DispatchMessageA”. “CreateWindowExA” creates a pop up window and “CallWindowProcA” passes message information to a specified window. “DispatchMessageA” also dispatches messages to window procedures. From these functions, we can see that the malware may be creating pop up windows and are passing information to these windows to display to users.

Table

Description automatically generated

The seventh library is WININET.DLL and it contains Internet-related functions that are used by the applications in the system. Some suspicious functions that were imported include, “HttpOpenRequestA”, “InternetConnectA”, “InternetOpenA” and “InternetReadFile”. These functions may indicate that the malware connects to the internet to access a webpage in order to retrieve a file.

Table

Description automatically generated

The eighth library is WS2\_32.DLL and it contains the Windows Sockets API used by many Internet and network applications to handle network connections. Thus it is used to establish and handle network connections and is an important system process. The only function that was imported is, “gethostbyname”, which suggests that the malware gets the hostname of the system which can then be used for network connections.

Table

Description automatically generated

The ninth library is OLE32.DLL and it has similar functions to OLEAUT32.DLL. It contains many core OLE functions. This library is used to provide Component Object Model (COM) which is used for clients to supply basic functions for creating objects and for servers, COM is used as means of exposing objects. Some suspicious functions that are imputed include, “CoCreateInstance” and “CoTaskMemAlloc”. These functions may indicate that the malware creates new instances and allocates them into the memory to be executed.

Table

Description automatically generated

OLE32.DLL also imported many functions and we managed to identify a function that may indicate malicious activity. This function is “OleRun” which places an OLE compound document object into the running state. This may imply that the malware may have the ability to force the starting of certain objects or programs on the system which can then affect the system.

Table

Description automatically generated

Other than the main 9 libraries shown, we also went through the sub libraries and managed to discover some malicious functions that can help us determine what the malware may be trying to achieve. We managed to find four libraries that were found to be importing suspicious strings.

The first sub library that showed suspicious imported functions is “ADVAPI32.DLL” which is located under “SHLWAPI.DLL”, under “SETUPAPI.DLL”, under “MPRAPII.DLL”, under “IPHLPAPI.DLL”, under “DNSAPI.DLL”, under “NETAPI32.DLL”, under “SECUR32.DLL”, and lastly under “ADVAPI32.DLL”. It imported the functions, “GetCurrentHwProfileA”, “GetUserNameA” and “GetUserNameW”. “GetCurrentHwProfileA” retrieves information about the hardware profile for the computer. “GetUserNameA” and “GetUserNameW” retrieves the name of the user associated with the current thread. These functions may indicate that the malware may be engaging in spyware activities to retrieve information about the users.

Table

Description automatically generated

The second sub library that showed suspicious functions is “USER32.DLL” which is under “SHELL32.DLL”. This sub library imported a huge number of functions and some of the suspicious functions that were identified includes, “CreateMenu”, “CreatePopupMenu”, and “CreateWindowExW”. “CreateMenu” and “CreatePopupMenu” are empty menus at the start and can be filled with items to display. “CreateWindowExW” creates a pop up window. These menus can be assigned to a window and can be displayed by the pop up window. These functions may give the impression that the malware can create multiple tabs of pop ups that display information from the menus. The malware can then spam the users with these windows.

Table

Description automatically generated

The third sub library that showed a couple of suspicious imported functions is “URLMON.DLL” and it is under “SHDOCVW.DLL” which is under “SHELL32.DLL”. The suspicious imported functions include, “URLOpenBlockingStreamW”, “URLDownloadToFileW”, and “URLDownloadToCacheFileW”. “URLOpenBlockingStreamW” creates a blocking type stream object from a url and downloads data from the internet. “URLDownloadToFileW” downloads bits from the internet and saves them as a file. “URLDownloadToCacheFileW” downloads data to the internet cache and returns the file name of the cache location. From these functions, we can infer that the malware accesses the internet and downloads data from the internet and saves it to files on the system which the malware might use to attack the system.

Table

Description automatically generated

The fourth sub library that showed a couple of suspicious imported functions is “USER32.DLL” and it is under “MSHTML.DLL”, which is under “SHDOCVW.DLL”, which is under “SHELL32.DLL”. This library imported a malicious function called “GetAsyncKeyState” which determines whether a key was pressed after the function call. This means that the malware can detect the keys that were entered by the user which may imply that the malware may have functions similar to that of a keylogger.

Table

Description automatically generated

#### 3.2.1.6 Bintext

We then used Bintext to analyse the strings found in the malware to see if we can find any clues and suspicious strings that can give us an idea of what the malware is doing. There were many strings that we could not make out but there were some suspicious strings which may indicate what the malware is trying to accomplish. The first suspicious string identified was “<http://fe.trymedia.com/a.a>”. This can be a network based indicator as the malware can be trying to reach this website. From there, the malware could be retrieving malicious files or downloading contents onto the system.

Graphical user interface, text, application, Word

Description automatically generated

The next group of suspicious strings is “Software\Microsoft\Internet Explorer”, “Macrovision\_DM\_2.2146” and “Software\Trymedia Systems\Download Manager”. These strings are host-based indicators and can indicate that the malware could be trying to use internet explorer to go online to download an application called, “Macrovision\_DM\_2.2146” into the directory, “Software\Trymedia Systems\Download Manager”.

Graphical user interface, text, application, email

Description automatically generated

Furthermore, when “Macrovision\_DM” was googled, results showed that it was a malicious software. Thus, this string identified can be related to the page shown below and can be malicious.

Text

Description automatically generated with medium confidence

Another group of suspicious lines of strings is “DownloadFolder”, “Software\Microsoft\Windows\CurrentVersion\Run” and “C:\Downloads”. These strings are also host-based indicators and could imply that after installation of the malicious software, the malware might store the malicious software in the directory, “Software\Microsoft\Windows\CurrentVersion\Run” such that the malicious software is always executed when the system is logged in to, making it persistent.

Graphical user interface, text, application

Description automatically generated

“Dm.html” was the next suspicious string to be identified. This is a network-based indicator and suggests that the malware could be accessing this html page when it accesses the internet. This html page could be the holding location of the malicious software and once the malware manages to enter the page, the malicious software could have then be downloaded into the system.

Graphical user interface, text, application, Word, email

Description automatically generated

BinText also showed a huge chunk of code in html, javascript and css that could have been used by the malware. A suspicious string to take note of is the one highlighted in blue, which is “document.write(ml\_getString(ST\_DOWNLOADING));”. This is a script injected into the title of the webpage and this could be a way of hiding the download process of the malicious software. Although not much can be said about this string, it does confirm the fact that something is being downloaded into the system.

Graphical user interface, text, application, email

Description automatically generated

“<http://www.trygames.com>” could also be another network-based indicator as the malware could be trying to access this webpage. “Trygames” is a platform that hosts many gaming applications and users can play on. “Trygames.com” is not a website that exists anymore and the closest website that is similar to this url is “Trygames.online”. Thus, due to the similarity of the urls and the website not existing anymore, “<http://www.trygames.com>” could be a fake website created by the malware so as to trick the users into thinking that the application downloaded from this website is legitimate.

Graphical user interface, text, application, Word, email

Description automatically generated

“dm.exe” is another suspicious string that has been identified. This is a host based indicator as it is an executable. This executable is probably installed on the computer when the malware accesses the dm.html webpage or could be installed by Macrovision\_DM, which was a malicious software that was identified earlier.

Graphical user interface, text, application, Word, email

Description automatically generated

Another group of suspicious strings were identified in BinText. These strings are “TrymediaGetAdBrokerResponse” and “TrymediaCallAdBroker”. Although we cannot tell much from these strings, we can assume that the application appears to be retrieving advertisements and seems to be using these advertisements. As seen in Section 3.2.1.1, the malware was reported to be an adware. This may mean that when the malicious application is runned, the users could be spammed by users with advertisements.

Graphical user interface, text, application, Word, email

Description automatically generated

The last suspicious thing that was noticed in bintext was a huge number of variables and its corresponding output. These variables may be used by the malware. Through analysis of the output of the variables, we can see many messages suggesting that there were errors during the downloading process. Perhaps, from this we can see that the motive of the attackers is for the users to get error messages when downloading the malicious software.

Graphical user interface, text, application

Description automatically generated

### 3.2.4 Conclusion About Second Malware (dm.exe)

With information that was mainly obtained from basic static analysis and briefly attained from basic dynamic analysis, we can conclude that the second malware is an adware. The malware could be making window pop ups and displaying them to users. The malware used a library called “GDI32.DLL”. It contains functions for Windows Graphical Device Interface (GDI) and performs primitive drawing functions for output, such as low level drawing or text, to video displays and printers. This gives the impression that the malware will be displaying pop ups or messages on the users screen. The malware also uses suspicious functions such as “CreateWindowExA”, “CallWindowProcA”, and “DispatchMessageA” which suggests that the malware may be creating pop up windows and are passing information to these windows to display to users. There were also indications of the malware stealing information from users as functions such as “GetUserNameExA” and “GetAsyncKeyState” were imported. These functions are usually used in spywares to gain information about the user and thus this malware might be trying to achieve the same result. From the analysis of the strings in the malware, we can say that the malware tries to access the internet because network based indicators such as “<http://fe.trymedia.com/a.a>” and “dm.html” were identified. It was also shown that the malware actively made connections to the url, “d.trymedia.com” and a netcat listener also managed to catch a GET request being made by the malware for an application called, “TreasurePyramidSetup.exe”. We can also infer that the malware downloads resources such as “dm.exe” and also created a new file called, “Downloads” in the desktop of the system. We also assume that the malware aims to spam the users with advertisements as the strings “TrymediaGetAdBrokerResponse” and “TrymediaCallAdBroker” were found. With these indications, it leads us to strongly believe that the malware retrieves malicious software through the use of the internet and this malicious software will create pop ups that will be displayed on the user screens. This strongly implies that the malware functions like an adware as it has the capabilities to spam the user with many popups which could be filled with advertisements or error messages.

Based on further analysis with advanced static analysis, we were able to find out further about what the malware is able to achieve. Firstly, the malware is able to detect if there is any form of debugger that is being runned to debug the malware. Secondly, the malware is also able to draw and create a new window of a particular size that will constantly pop up whenever one window is closed. And lastly, it is also able to map a set of coordinate space that is relative to different windows. Thus, this gives the impression that the malware is creating an adware where if the victim tries to close one of the windows, another one will keep popping up inside the system.

# 6. Conclusion and Reflection

All in all, we as a team did analysis on three malware executables and one malware document. For the first malware, basic static analysis, basic dynamic analysis and advanced static analysis was done. For the second malware, basic static analysis and advanced static analysis was done. For the third malware, only basic dynamic analysis was done. There were issues that we encountered along the way which resulted in the splitting of the analysis in this manner.

For the first malware, the general conclusion that was made is that it is a ransomware that encrypts and prevents access to the system files and prevents the usage of normal operations. For the second malware, the general conclusion that was made is that the malware is an adware that accesses the internet to download malicious software. For the third malware, the general conclusion that was made is that the malware is a keylogger that discreetly records and logs every single key stroke that was entered into the system. For the malware document, the general conclusion that was made is that the document accesses malicious websites to download malicious programs and exploits the command prompt to execute malicious tasks. More information about the malware can be found in their individual segments.

## 6.1 Edward’s Reflection

I feel that the malware analysis which I did was pretty well done and detailed. I did the basic dynamic analysis and advanced static analysis of the first malware, and the basic static analysis of the second malware. I was really overwhelmed by the assignment at the start because understanding what the malware does is truly not an easy task. However, with time, I slowly managed to figure out and gained an idea of what the malware is trying to achieve. I managed to use the tools taught in school and applied the skills I have learnt through this assignment. It was a long and tiring process and I am glad that I was able to complete it. Through this assignment, it has taught me the value of perseverance because I had to keep trying to understand what the malware does and although there were difficult moments, I did not give up and continued. Although the assignment went fairly well, there will still room for improvements. I feel time management could have definitely been better. Doing this assignment was certainly rushed and the quality could have definitely been much better if I had started earlier. I feel that I could have also gained a better understanding of the malware if I had more time as well. Furthermore, if our team had noticed the discrepancies earlier, we would have time to change malware. However, we did the analysis quite late and as a result could not afford the time to conduct static analysis on the third malware and thus had to resort to leaving in the static analysis conducted on the second malware which resulted in the separation of analysis for the malwares. Due to the separation of analysis of the malwares, it was hard to come to a proper conclusion on what the malware does for the second and third malware and thus the accuracy of the conclusion was affected. Although we did face difficulties, we faced them together and I am grateful for working this assignment with Mabel and Clarence and am thankful for all the support they have given me throughout this assignment. They made this assignment much more fun and enjoyable. If given the opportunity to do this again with them, I definitely would.

7. Youtube Links

7.1 Introduction Video - Lab Setup

https://youtu.be/U6-TpgTG56E

7.2 Introduction Video - Description of Basic Static and Basic Dynamic Analysis Tools

https://youtu.be/6AgqU-wzphc

7.3 Introduction Video - Description of Advanced Static and Advanced Dynamic Analysis Tools

https://youtu.be/3bhFLmfctBA

7.4 Malware 1 - Basic Static Analysis

https://youtu.be/AGoctqoaBG0

7.5 Malware 1 - Basic Dynamic Analysis

https://youtu.be/hSuxMEn3xBc

7.6 Malware 1 - Advanced Static Analysis

https://youtu.be/uZ7cv-L8oVU

7.7 Malware 1 - Conclusion

https://youtu.be/IIUZkKx8KSc

7.8 Malware 2 - Basic Static Analysis

https://youtu.be/009-6yEvJpo

7.9 Malware 2 - Advanced Static Analysis

https://youtu.be/dJfqu9fShrw

7.10 Malware 2 - Conclusion

https://youtu.be/iTXypvKezOQ

7.11 Malware 3 - Basic Dynamic Analysis

https://youtu.be/TSHsnVeQbd8

7.12 Malware 3 - Conclusion

https://youtu.be/oe5urs\_HcRo

7.13 Document Analysis

https://youtu.be/DgghwUtblSg - Part 1

https://youtu.be/fZh9Hpd2irY - Part 2

https://youtu.be/fJYS\_-sSZmM - Part 3

7.14 Conclusion and Reflection - Edward

https://youtu.be/woqWyPMmjic

7.15 Reflection - Mabel

https://youtu.be/zAt5vG2rKHo

7.16 Reflection - Clarence

https://youtu.be/xSTKVES-0Wc