**Swinburne University of Technology**  **School of Science, Computing and Engineering Technologies**

# COS20030 Malware Analysis

## Assignment 1, Semester 2, 2024

**Important Information**

|  |  |
| --- | --- |
| **Due Date** | **23:59 on 8th September Sunday- Mid-semester Break**  (Late submission penalty: 5% of total available marks per day) |
| **Submission Method** | **Canvas** |
| Contribution to Final  Assessment | 25% |

## Purpose of the assignment

This practical assignment is divided into two sections. In each section, you will analyse a different malware file, answer a set of questions, and provide supporting evidence. This assignment offers you an additional opportunity to apply and enhance the malware analysis skills you've developed in Labs 1 through 5.

## Submission

1. A Word document with the assignment cover, including your full name and student ID. This document should contain all your answers to the questions and screenshots of the list of commands that Q1.exe can handle at the end. Additionally, you need to write a couple of paragraphs at the end reflecting on this assignment, discussing the challenges and highlights.
2. Record a short video of yourself demonstrating the process of unpacking Q1

(maximum length: five minutes). Upload the video to YouTube and post the YouTube link as a comment on Canvas.

1. The unpacked file for Q1. This file must be zipped and password-protected (use "infected" as the password).
2. Upload the .idb file from IDA related to Q2, containing the renamed function and variable names, as well as comments made in the code. Be sure to include your student ID and name in addition to the other comments

## Specified Requirements

### Section 1: Q1.exe (55 points)

In this section, you have a sample of a passive backdoor malware. A passive backdoor does not reach out to a command-and-control server; instead, it waits for a connection to be made to it. More specifically, a passive backdoor starts listening on a port for incoming connections. When a connection is made to it, it starts communicating and receiving commands to perform different actions on the infected machine.

Our goals in this exercise are:

1. Perform basic static analysis to find out more about the malware file:
   1. Confirm that it’s packed and recognise the packer.
   2. Find the MD5 hash and Entropy of the packed file’s PE header
   3. Find when the file was compiled
2. Perform dynamic analysis to:
   1. Find out which port number the backdoor listens on
   2. Use the Netcat (ncat) tool to emulate the server contacting and communicating the backdoor
   3. Try some of the features of the backdoor by sending specific commands to the backdoor and observing its malicious behavior
3. The final goal is to unpack the backdoor sample using a debugger and a plugin to dump the unpacked code and reconstruct its headers.

The file to analyse is **Md5: 8f7e26469d4a00136c0b76e6249ce684**

#### *Questions*

1. Use q1.exe for this exercise. List three indicators that q1.exe is packed. What packer was used to pack q1.exe? (5 points)  **UPX used, big size difference between size and virtual size, few imported dll files, high entropy.**
2. Answer the following questions:
3. What is the MD5 hash of the packed file’s PE Header? (2 points) **8f7e26469d4a00136c0b76e6249ce684**
4. What is the entropy value of the packed file’s PE Header? How did you find this? (3 points) **6.60263 (inside entropy in DIE)**
5. On what date was this program compiled? (2 points) **2001 – 11 -21 04:17:53**
6. Execute the program and find out what port q1.exe listens on. How did you find this information? (5 points) The `Q1.exe` malware is listening on port 5277. I identified this by running the `netstat -an | find “LISTEN”` command in the command prompt, which displayed all the open ports on the system. Among the listed ports, 5277 was identified as the one `Q1.exe` was using for listening to incoming connections. I confirmed this by matching the PID associated with this port to the PID of `Q1.exe` using Process Explorer.
7. While the program is running, open a CMD window then type the following command and use the port number you found in the previous question. The ncat tool will connect to the port number specified and start communicating on this port. This should result in ncat talking to the q1.exe process.

**ncat localhost <port number>**

* 1. Once you have run the ncat command, type “?” to list the different commands that the q1.exe can execute. Provide a screenshot of the list of commands. (3 points)

A screen shot of a computer

Description automatically generated

* 1. Once you have run the ncat command, type “?” to list the different commands that the q1.exe can execute. Provide a screenshot of the list of commands. (3 points)

Use the "URL [http://www.maldomain.com"](http://www.maldomain.com/) command to attempt to download a file with q1.exe. This operation will fail due to not being able to contact the real website, however, the executable scans the registry for the status of multiple keys relating to network connectivity. One of those keys relates to [PunyCode.](https://www.jamf.com/blog/punycode-attacks/) What application does q1.exe check to see if PunyCode is enabled? How do you know? (7 points)

1. **Path Observed:**
   * "In Process Monitor, I observed that Q1.exe accessed the registry path HKLM\SOFTWARE\Microsoft\AppModel\Lookaside\user\SOFTWARE\Classes\PROTOCOLs\Handler\URL Http after the URL http://www.maldomain.com command was issued."
2. **Possible Interpretation:**
   * "This path appears to be related to how the system handles and processes HTTP URLs. Given that PunyCode is involved in the encoding of international domain names, Q1.exe might be checking whether the system or certain applications can properly handle such URLs."
3. **Conclusion:**
   * "This suggests that the malware is verifying the URL handling capabilities of the system, which could include PunyCode handling."
4. Through ncat, send the “i” command to make q1.exe install itself. What is the registry key that the malware uses for persistence? What is the full path of the file that is made persistent? How did you find this information? (10 points)

During the analysis, Q1.exe attempted to create a persistence mechanism by writing to the registry key HKLM\Software\WOW6432Node\Microsoft\Windows\CurrentVersion\Run\bndshell. However, this operation resulted in an 'Access Denied' error, indicating that the malware was blocked from modifying this key. This suggests that Q1.exe may have been prevented from establishing persistence on the system, possibly due to insufficient permissions or the intervention of security software."

### ****Conclusion:****

1. **Significance:** This finding is crucial as it indicates that the malware may not have been successful in its attempt to maintain persistence, reducing its impact on the system.
2. **Security Effectiveness:** It also highlights the potential effectiveness of your system’s security measures in preventing unauthorized modifications to critical system areas.
3. Unpack the file, record a video of yourself doing it, and submit both the unpacked file and the video. (3 points for the file + 12 points for the video)
4. Find out what API is used by this malware to download files from the given URL.  
   (3 points)

### Section 2: Q2.exe (40 points)

In this section, you have a sample of a different malware. We want to learn about some of the characteristics of this file and understand some of the features of this malware by examining its code in a disassembler.

Our goals in this exercise are:

1. Use the IDA Freeware tool to disassemble the file and find out about the libraries imported by this malware
2. Find where a specific string is used in the code
3. Analyze and understand some of the functions in the malware by examining the decompiled code (Pseudocode), renaming function names and variables to meaningful names and adding comments in the code to explain what the function does.
4. Analyze and understand a part of the disassembled code (Assembly code) of the program by reading the assembly instructions between two given addresses in the code and adding comments in the disassembled code to explain what the code does.

File to analyse: **Md5: 60b12ad2f23f2e4a594daa17cb8f517c**

#### *Questions*

1. How many DLL files does q2.exe import functions from? (3 points) **4**
2. What’s the address of the string “ComSpec”? (3 points) **00404448**
3. How many arguments are passed to FUN\_00401d30? (3 points) The function **FUN\_00401d30 takes 3 arguments**
4. What undocumented functions from NTDLL.DLL are used in this program? (6 points)

Besides the documented functions from NTDLL.DLL, Q2.exe also imports several standard C runtime functions such as \_snwprintf, memset, sprintf, and others. These functions are commonly used for string manipulation, memory operations, and formatting tasks, and they are all documented in the standard C runtime libraries.

1. Rename param\_1 in FUN\_00401d30 to something more descriptive. Justify the variable name you chose and save your work in IDA (7 points)

**Acknowledgement: "In the analysis of FUN\_00401d30, IDA automatically named the first parameter as lpModuleName. This name was generated because the parameter is passed to GetModuleHandleA and LoadLibraryA, both of which typically take a module name as an argument."**

**Justification: "The name lpModuleName is appropriate because it clearly describes the parameter's role in the function—it is a pointer to a string that specifies the name of the module to be loaded or retrieved. This aligns with standard Windows API naming conventions, where lpModuleName is commonly used in functions dealing with module handles."**

1. Describe what the code between 0x0040239d and 0x004023b9 does and add comments in IDA. (8 points)

**The code between 0x0040239d and 0x004023b9 performs several checks and operations to determine whether the current process is running under WoW64 (a 32-bit process on a 64-bit system). It calls IsWow64Process to set a flag indicating the environment, retrieves the process and module handles, and conditionally jumps based on the results. The section culminates in either proceeding with further module-related operations or skipping them based on the environment and module status.**

1. Briefly describe what the function located at 402090 does. Suggest a descriptive name for that function. Change the name of the function, add comments in the code and **save your work in IDA** (10 points)

**The function at 0x402090 is responsible for preparing a command-line string that can be used to delete the current executable file. It first retrieves the full path of the executable using GetModuleFileNameA, converts this path to its short form using GetShortPathNameA, and then constructs a command string using lstrcpyA to prepare a deletion command. Given these operations, the function has been renamed to PrepareDeleteCommand."**

#### Reflection (5 points)

Please reflect on your experience completing this assignment and describe the challenges and highlights you encountered.

#### Assignment Requirements Checklist

|  |  |
| --- | --- |
| **Requirements** |  |
| Screenshots of the list of commands that the Q1.exe can handle. Answer to all questions written in a Word document  Record a short video of yourself going through the process of unpacking Q1 (maximum five minutes) and upload it to YouTube |  |
| The unpacked file for Q1. This file must be zipped and password protected (use “infected” as password)  Save an .idb file from IDA, related to Q2 that contains the renamed function and variable names as well as the comments made in  the code (include your student ID and name in addition to the other comments) |  |

#### Deductions

|  |  |
| --- | --- |
| **Requirement** |  |
| The YouTube video link does not open | -10 |
| No progress check | -10 |

**Warning: All submissions are automatically checked for similarities to other submissions through Turnitin.**