A black and red sign with white text

Description automatically generated

Swinburne University of Technology

**COS20030 Malware Analysis**

Assignment 1

Marco Giacoppo (104071453)

**Section 1: Q1.exe**

1. UPX was used.

- size difference between size and virtual size on UPX0

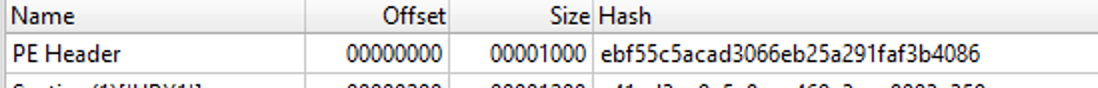
- few imported DLL files

- high entropy on PE Header and UPX1.

**A screenshot of a computer

Description automatically generated**

1. A) ebf55c5acad3066eb25a291faf3b4086



B) 7.39257 (inside entropy in DIE)

A screenshot of a computer

Description automatically generated

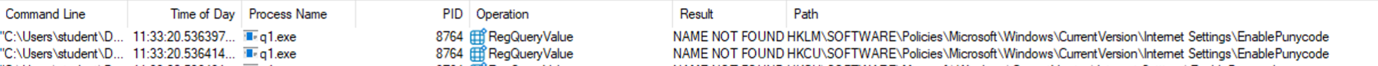
C) 2001 – 11 -21 04:17:53

1. 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.
2. A)

A screenshot of a computer

Description automatically generated

B) "In Process Monitor, I observed that Q1.exe accessed the registry path **HKLM\Software\WOW6432Node\Microsoft\Windows\CurrentVersion\Internet Settings\EnablePunyCode**

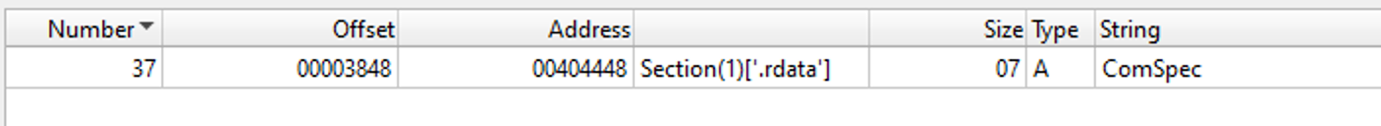
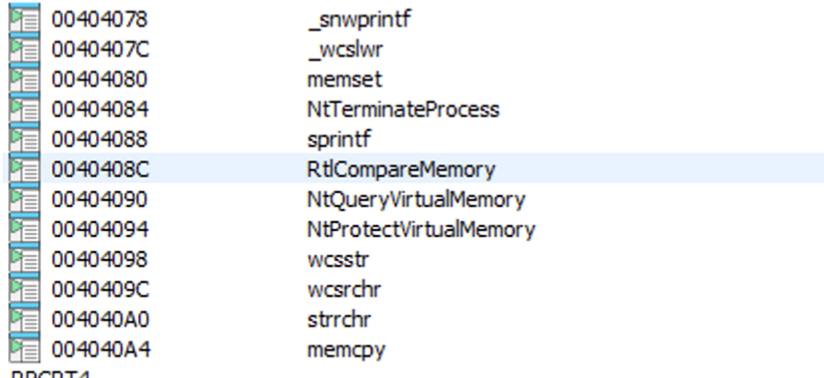
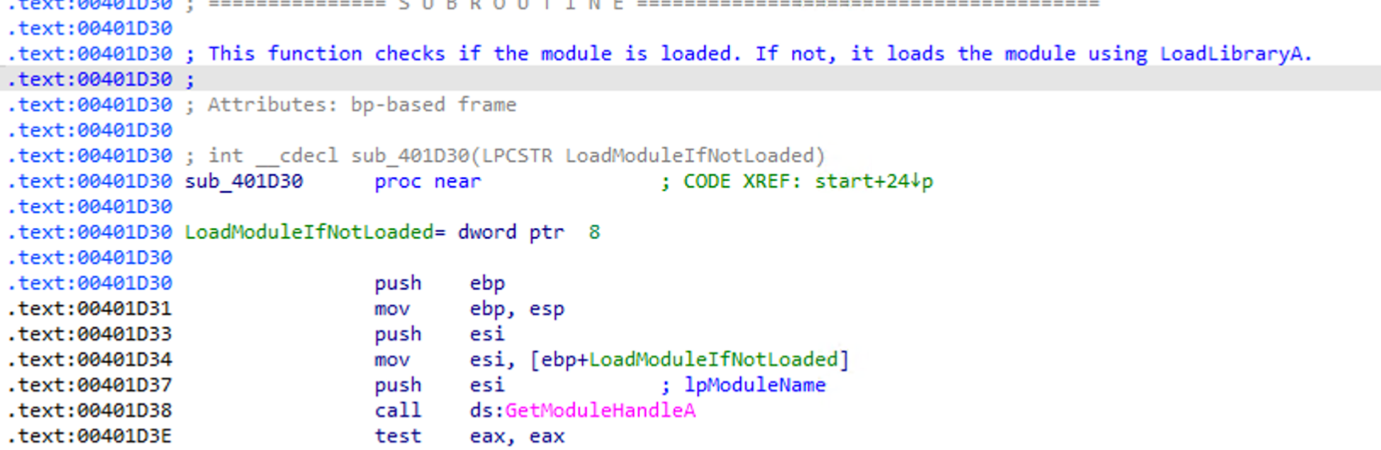


* + It’s scanning through web applications probably from Microsoft Edge.

1. 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**. This path was observed in the Detail column of Process Monitor when the registry key was being created.
2. [Unpacking Q1 File](https://youtu.be/KF4NvT1ZIqw)
3. The malware uses the URLDownloadToFile API to download files from the specified URL

**Section 2: Q2.exe**

1. 4 files ( KERNEL32.dll, ntdll.dll, RPCRT4.dll, and SHELL32.dll )A screenshot of a computer

   Description automatically generated
2. The address is ‘00404448’
3. The function FUN\_00401d30 takes 1 argument which is “ntdll.dll”
4. NtTerminateProcess, NtQueryVirutalMemory, and NtProtectVirtualMemory are not documented. 
5. The parameter `param\_1` was renamed to `LoadModuleIfNotLoaded` to reflect its purpose in the function. The function uses `param\_1` to check if a specific module is already loaded into memory via `GetModuleHandleA`. If the module is not loaded, the function then uses `param\_1` to load the module via `LoadLibraryA`. 
6. 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).
7. The function at 0x402090 is responsible for preparing a command-line string that can be used to delete the current executable file. Given these operations, the function has been renamed to SelfDeleteExecutable.A screenshot of a computer program

   Description automatically generated

Reflection on the Assignment:

This assignment was quite tricky, it forces us to be very careful in reverse engineering and binary analysis using few tools. One of the main challenges was interpreting the assembly code and mapping it to higher-level logic without any prior context or variable names. It required a careful examination of API calls to deduce the purpose of the function and its parameters. Finding the param\_1 was also quite challenging since the IDA renamed it automatically to ipmodulename, which I found out after cross checking with Ghidra.

A key highlight was analyzing system-level interactions, such as identyfing how functions like IsWOw64Process and ShellExecuteA were used. The process of renaming and commenting on the self-deletion function provided insight into how programs or malware might clean up after execution. Overall, I’m pretty happy with how everything turned out.