ITSC 303

Malware Analysis

Final Project - Weekly Report 5

D’Artagnan Boocock & Coleton Sanheim

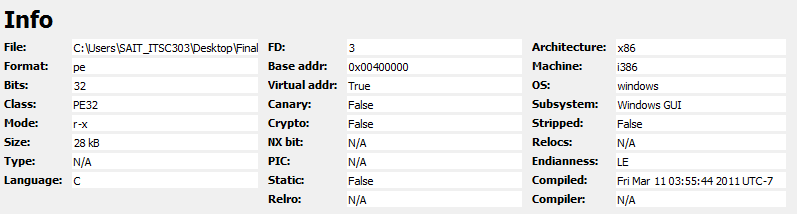
09/04/2022

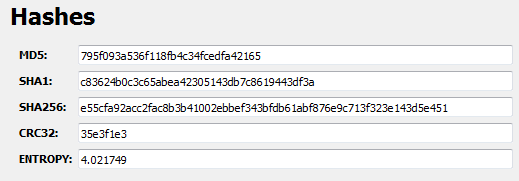
Sample 1:

Note that for both the samples, we disassembled them using a program called Cutter, we found that Cutter was much better equipped to handle this analysis than Ghidra.

Basic info:

This is a 28kB 32 bit binary

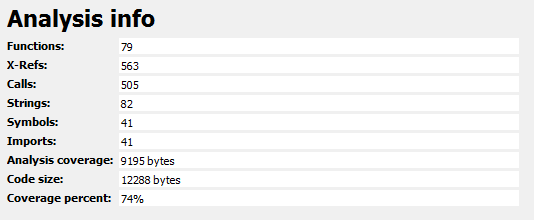




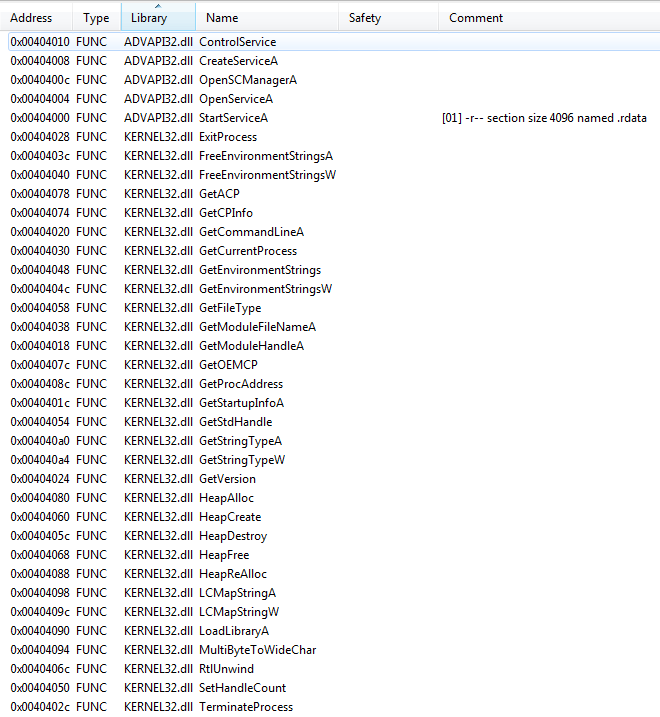
The thing imports the advapi32.dll library to get access to ControlService, CreateServiceA, OpenSCManagerA, OpenServiceA and StartServiceA



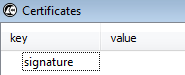
The binary has 79 functions consisting of an entry0 function as well as the main function



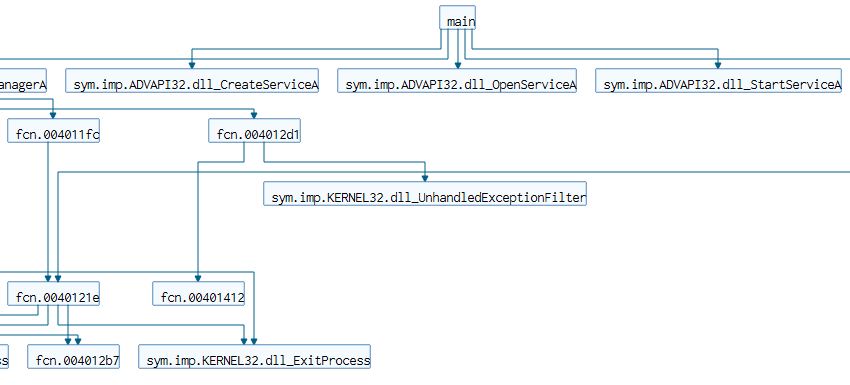
The binary loads ADVAPI32.dll and KERNEL32.dll to import the following functions:



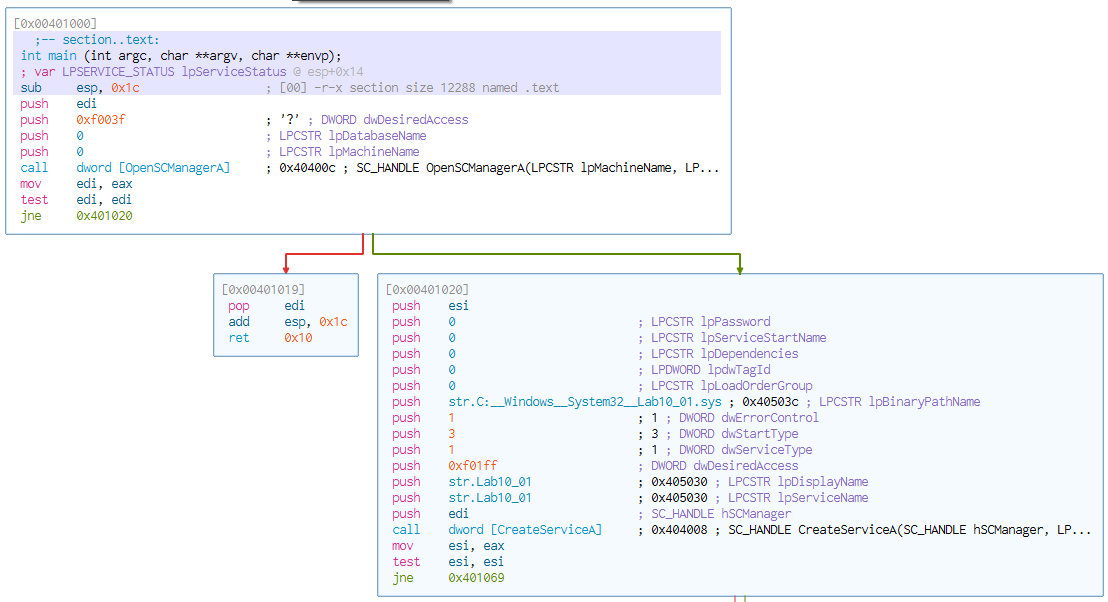
The binary is not signed

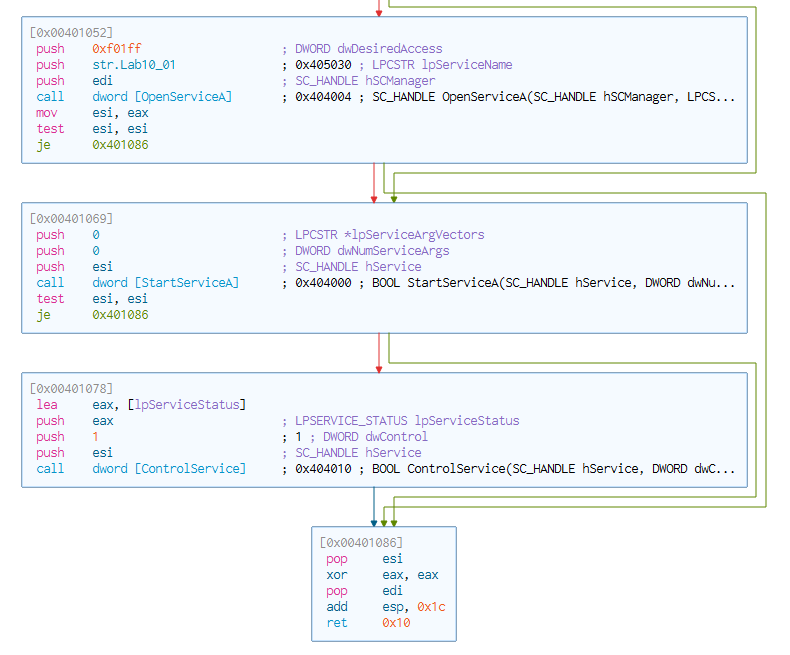


The global call table maps the relations between all 79 functions



Break down of main function:

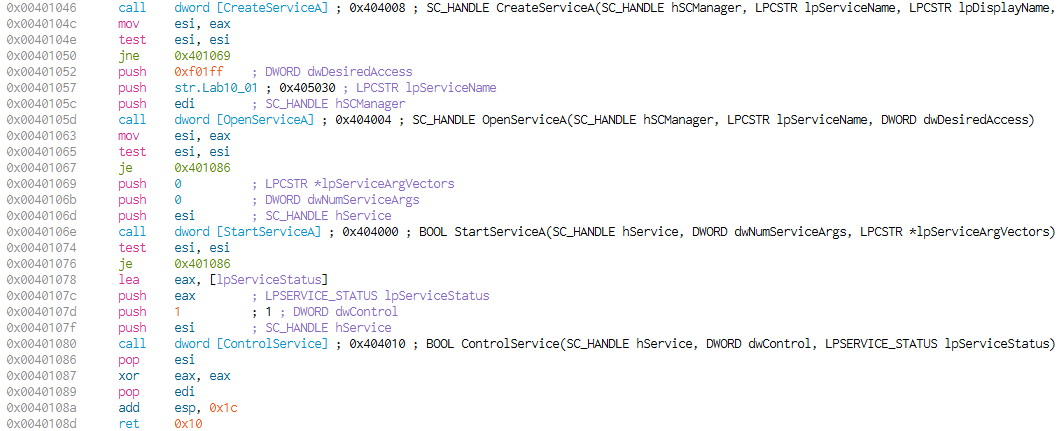


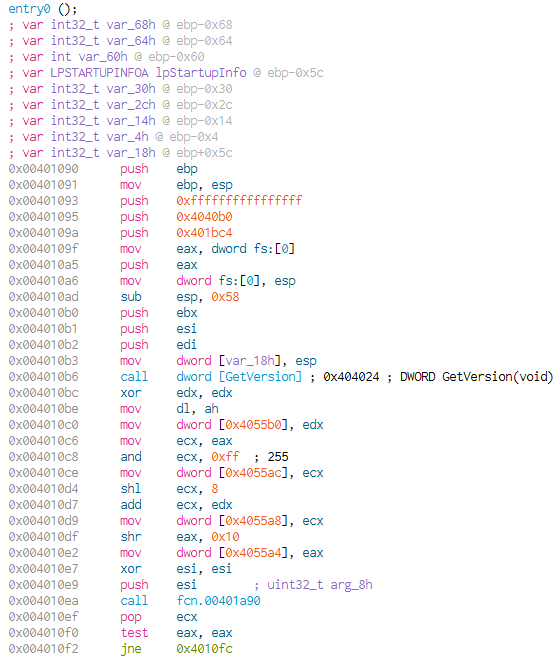


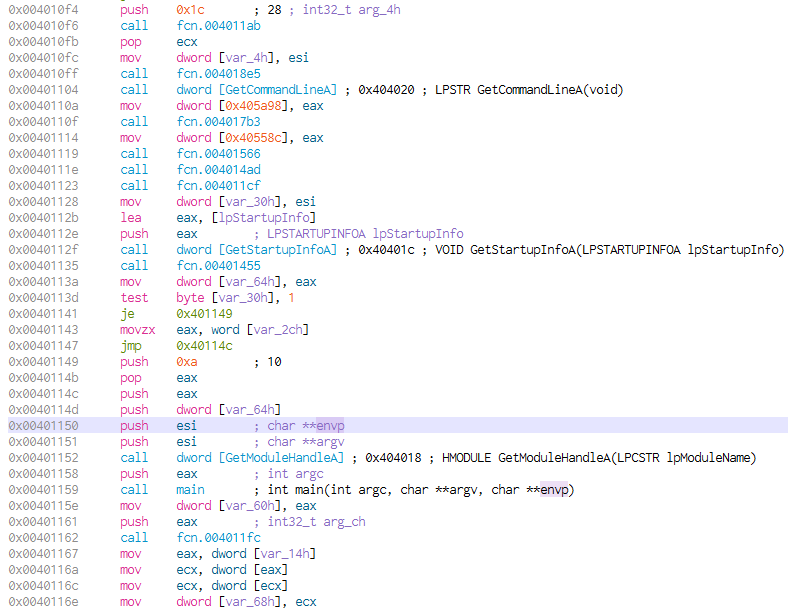
Assembly code from the binary:

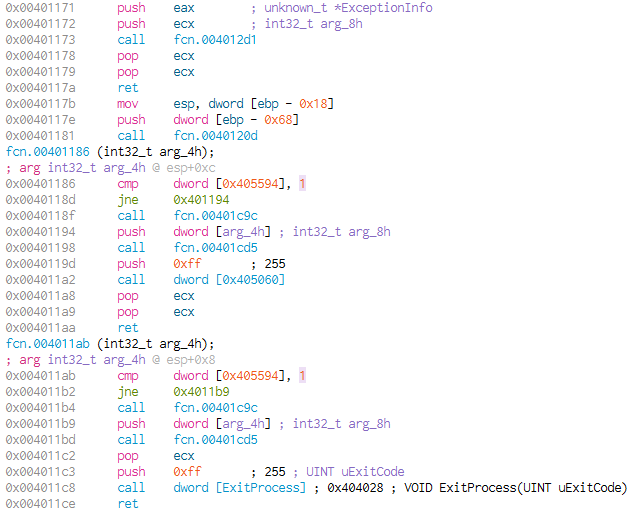
This is the main code function

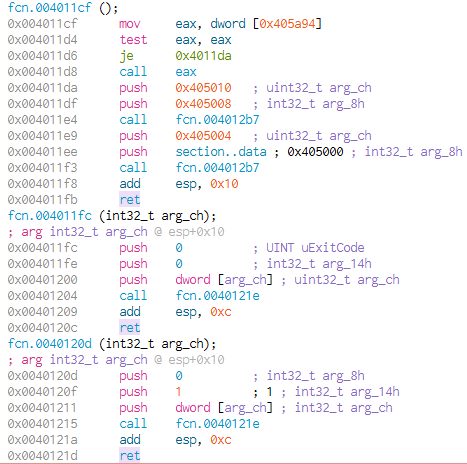


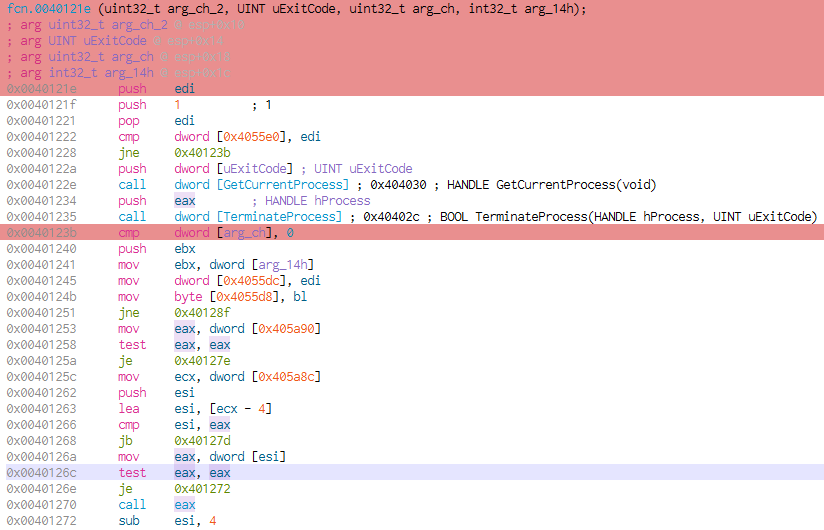


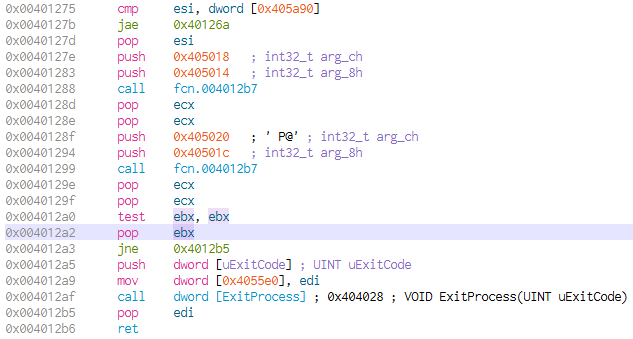








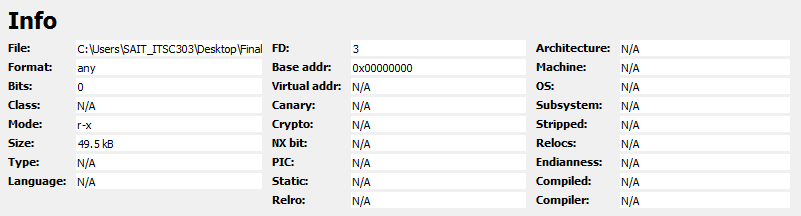


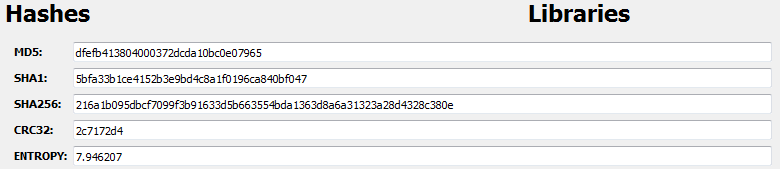


There is a ridiculous amount of code to dissect here, presumably from an attempt to obfuscate the malicious code. Dynamic analysis is required to gain more insight into the actions of the malicious code, and to separate it from the benign code.

Sample 2:

The debugger does not recognize the file as an executable. This is because it's a pdf file.



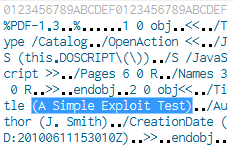


This malware contains only a single function within the payload.

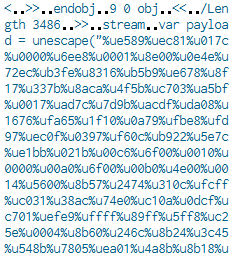


Hex dump analysis

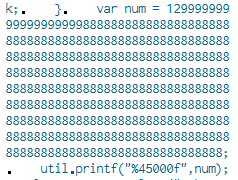
The beginning of the hexdump reveals the PDF magic number as well as the author name, script type and the title “A Simple Exploit Test”



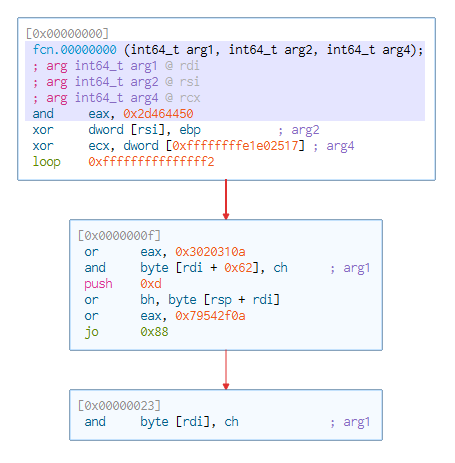
The 802 byte payload is provided in plain-text within the hexdump of the pdf file.



An interesting variable stands out. The variable is initialized as a value far exceeding Javascript’s MAX\_SAFE\_INTEGER of 9x10^15 (perhaps to trigger some sort of buffer overflow?)



Function graph

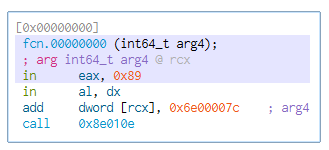


Sample 2 payload

The payload is 802 bytes long with 22 cross references



The payload consists of a single function used to perform the exploit



To determine what the shellcode does, further dynamic analysis is required.

Conclusion

Further static analysis will not yield anymore information beyond what we have already gathered. To determine the purpose and effects of these samples we need to begin dynamic analysis.