Nem Negash

CMSC 426

10/10/2021

**Malware Analysis Lab**

Text

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Diagram

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

A picture containing graphical user interface

Description automatically generated

Graphical user interface, text, application, chat or text message

Description automatically generated

**Conclusion and Lab Report:**

Your company’s systems have been hacked and your CIO is not pleased. You have had a chance to work through the intrusion scenario using various tools to piece together what happened. Write a short (at least 1 page in addition to screen shots) report on the nature of this malware and how it operated on your system. Use screen shots of the tools you used and explain to your CIO the process of what happened during this breach. What are some strategies that could be used to mitigate this threat in the future?

Additional: Which tools did you find to be the most useful? For disassembly, did you find *IDA* or *Ghidra* to be more helpful?

We first made sure the executable file was in fact a malware. The first part of analysis done was looking at other antivirus tools outside of our internal UTM that consider the file to be somewhat malicious. From there we examined the strings embedded in the executable. This reveals filenames, hostnames, software APIs, and URLs. This also shows us that the file is using cryptic strings which means it has been packed and encrypted to hide from antivirus tools. Using Exeinfo PE we can see that it was packed using a common packer called UPX. PeStudio is another tool that was used to calculate hash hats that can be used to get information through public and private sources. This shows us that the program cannot be run in DOS mode. Using four main tools, Process Hacker, Process Monitor, WireShark, and ProcDOT we proceeded to do further investigation. Process Monitor showed us the malware runs a file called “system\_clean.exe” and Wireshark showed the infected system issues DNS queries to resolve google.com to ensure an internet connection. We used FakeNet and INetSim to simulate a fake connection for the malware to continue working. From here using ProcDOT, it showed us that the “WinHost32.exe” process wrote a new file to disc. Running strings on this output we see it is a PE and contains information such as a potential URL to the malware “mother ship” and several API calls. These disassembled the PE and produced the assembly code that makes up the file. Also, we noticed the program calls WriteProcessMemory and Terminate process which no distinction of what is being terminated. The use of VirtualAllocEx together with WriteProcessMemory is a standard pattern used with code injection malware. We found IDA to be most helpful since I have decent experience with assembly 80x86 NASM. Looking at assembly code helps me understand what exactly what the malware is doing at a very low level to the registers on the host system.