Trojan Backdoor Sample: Static and Dynamic Analysis - a Hands-On Project

Shane Irons

CAP5137 Software Reverse Engineering Fall 22 – FSU

Dr. Xiuwen Liu

12/9/2022

I. Introduction

This document serves as a report showcasing my hands-on analysis of a malicious bin file. I am using a generic trojan malicious sample found at [5]; There is no information given about this file other than that it is a real-world sample and is a "Generic Trojan". I will be using a Windows 7 VM configured in VMWare Workstation Pro with static and dynamic malware analysis tools used during class and the Malware Analysis Workshop. This project has shown me that it can be difficult to determine what a program does through static analysis alone and that for a bigger picture, analysts should implement techniques in both static and dynamic analysis to safely determine what a sample program is capable of.

II. Report: including results and screenshot demonstration

My configuration includes a Windows 7 VM in VMWare Workstation Pro without VMWare tools installed. The tools installation for VMWare was not working, so I continued the project without them. I had the network disabled so the malicious file would not have the chance to communicate over the network (Figures 1 and 2). Figure 2 shows the Windows 7 VM with the analysis tools that are installed as well as the malware itself in a zip folder. This VM was the basic installation without any updates or upgrades. These tools include procmon and procexp, regshot and regfsnotify, 7zip, and IDA Free. This configuration is almost the same as the one used in the workshop assignment. These tools were downloaded and installed following the workshop assignment instructions.

With configuration out of the way, the first step I take to analyzing the malware is to boot IDA and load the sample into it. I use IDA for my static analysis. In IDA view, MZKERNEL32.DLL can be seen near functions for LoadLibraryA and GetProcAddress (figure 7). These two functions target KERNEL32. Kernel32 is the 32-bit dynamic link library for Windows OS that on boot, is loaded into a protected memory [6]. This dll performs background processes like memory management, interrupts, and I/O operations [6]. This is a red flag as MZKERNEL32.DLL is said in the bin file to have read write execute access and is in a vulnerable part of the machine that is executed on boot. From static analysis, it seems that MZKERNEL32.DLL could possibly be attempting to replace or alter Kernel32.dll.This immediately catches my attention before beginning the dynamic analysis (figures 3 and 4).

Figures 5-9 document the results and outcome of executing this sample. Procmon did not pick up a lot, however, it does show a read on kernel32.dll, which was present in the IDA disassembly and static analysis explanation. From this, I know the sample is working as intended. Figure 6 shows registry keys being added, mostly by procmon and other tools I am using for analysis. The interesting part of this was the keys in HKLM\SOFTWARE\WINDOWS and HKLM\SOFTWARE\CLASSES. These locations are known to host potentially malicious drivers for spyware [1]. A lot of this can be legitimate system commands, however, with the knowledge that this is a Trojan using MZKERNEL32.dll as a potential backdoor, I feel that possibly legitimate files are being corrupted. Furthermore, the regshot file showed that something was messing around with registry keys in HKLM\SOFTWARE\WINDOWS and \CLASSES. I also would like to note that this program does not stop on its own. This can be spyware possibly, however, I could not totally conclude this. I am not convinced this is spyware though because of the fact the sample does not call or create any services or direct the functions to any suspicious external site/process. Regardless, I am certain that the malicious sample creates a backdoor in

MZKERNEL32.DLL [2] [3] [4]. This was concluded through both static and dynamic analysis, as well as research into MZKERNEL32 and HKLM\SOFTWARE common trends regarding malware [1] [2] [3] [4].

After this conclusion, I reset the VM to a clean snapshot to run the sample a second time. I had the same configurations as before and executed the steps in the same order. This is noteworthy because there are some differences in my dynamic analysis from the second execution. The most glaring changes that occurred during the second run of the malware is the fact that registry keys were altered then removed (figure 10). This is different because the first execution has regshot show keys being added whereas the second execution shows keys being removed. These two executions of the sample were conducted in the same clean environments, so this was odd. Knowing that Kernel32.dll is a kernel module kept in protected memory leads me to believe that MZKERNEL32.DLL has potentially overwritten it and maintained a backdoor even after reloading a snapshot [6]. It is known that the Kernel32.dll carries out functions with I/O operations, so it is possible that the backdoor attaches itself to I/O streams (alongside being in protected memory) for persistence and propagation. I believe infected core files persisted and alerted the sample that a backdoor had already been established. With this information, I now believe this file to establish a backdoor on a system that when accessed again, spies on or propagates through persistent channels against the target machine. I am still unsure about a decisive conclusion as to the total behavior of this sample since it does not seem to have any other services or functions that would do this in IDA. The sample could be using obfuscation techniques that make static analysis more challenging. My machine has experienced no noticeable slowdown and no alarming processes have been caught running afterward. Whether the spyware and propagation claims are true or not, I am certain at the very least this is a Trojan Horse sample that creates a backdoor.

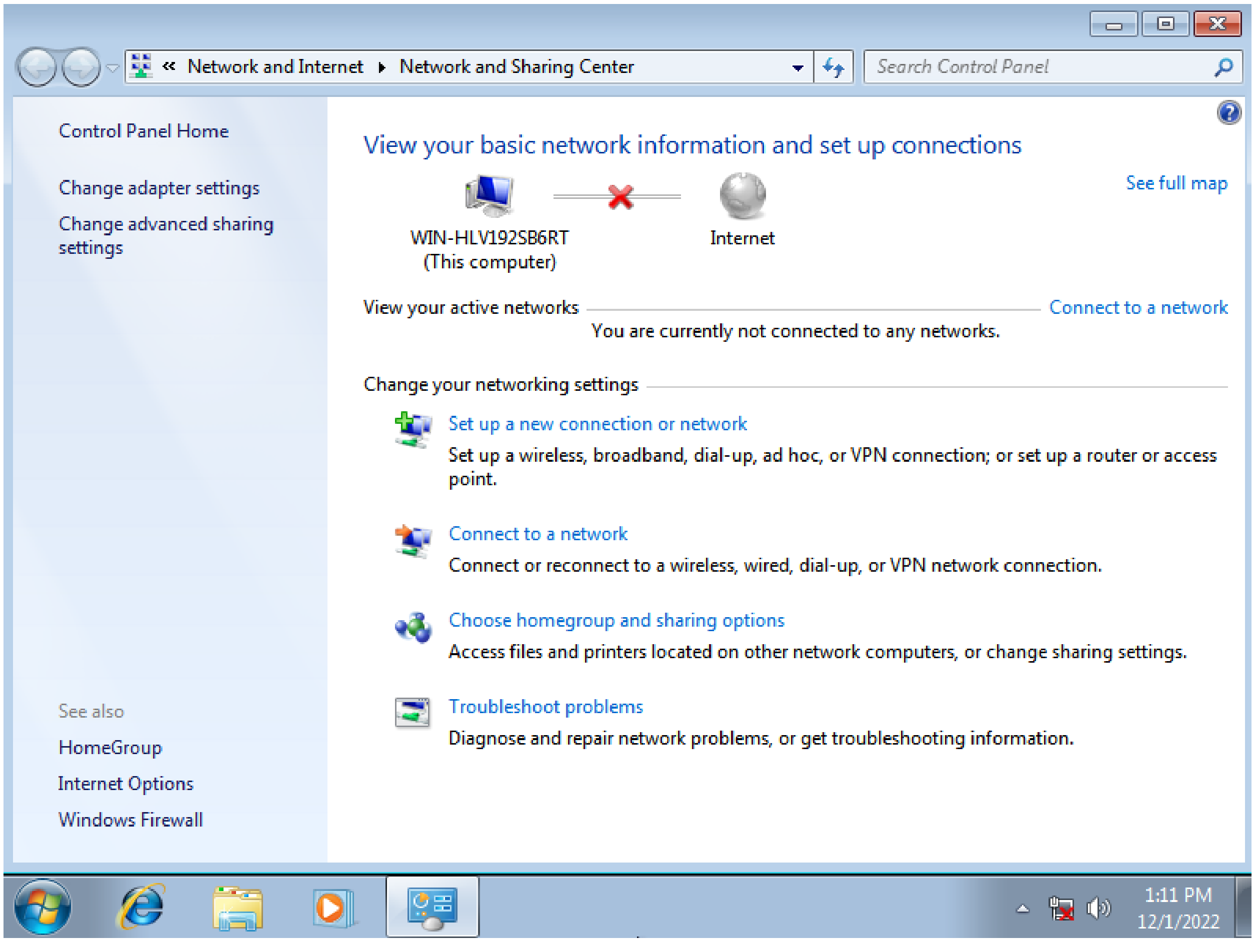


Figure 1: Network Configuration

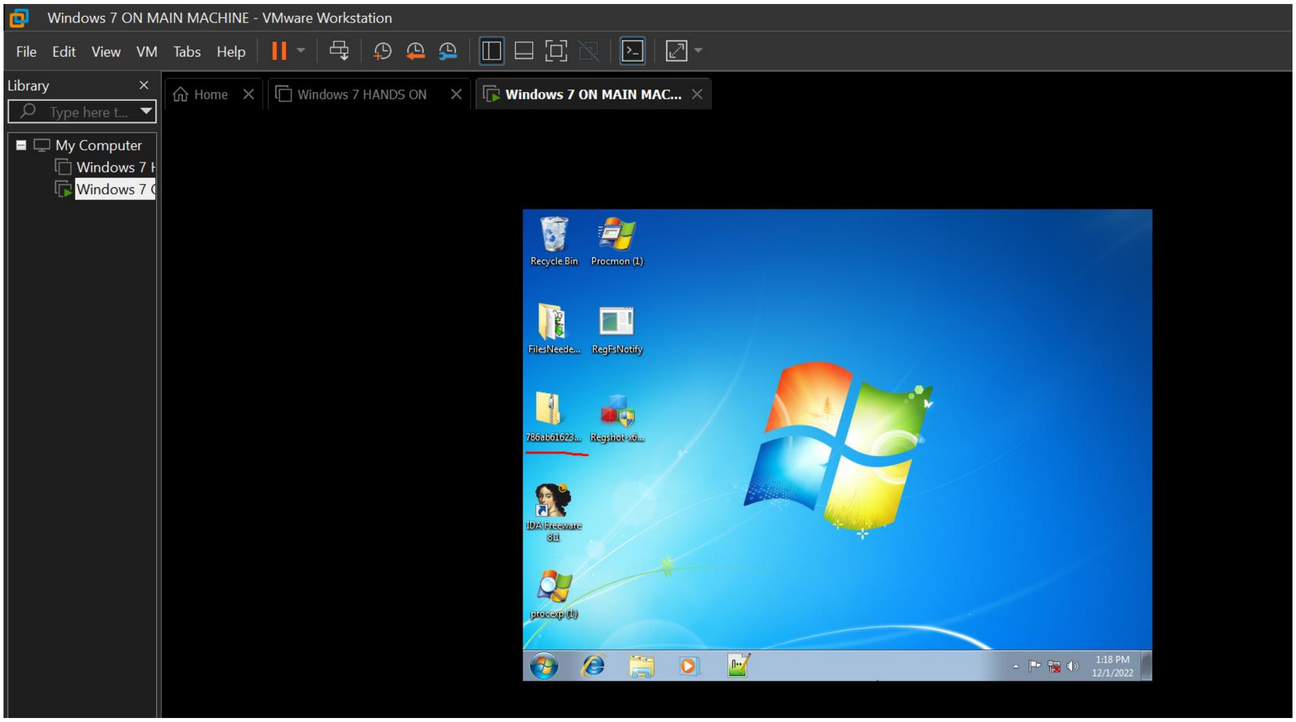


Figure 2: Tools and Malware

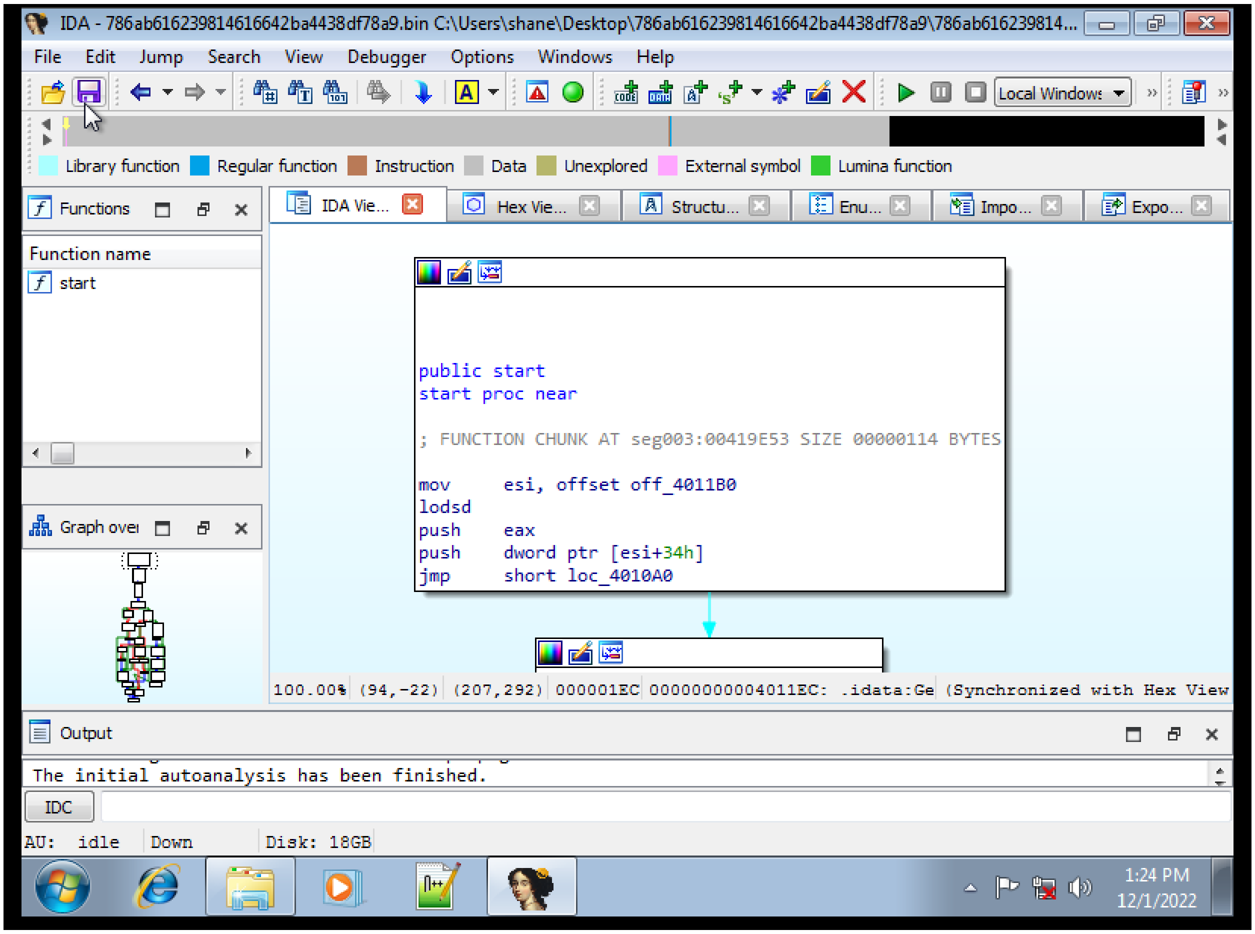


Figure 3: Initial IDA screen

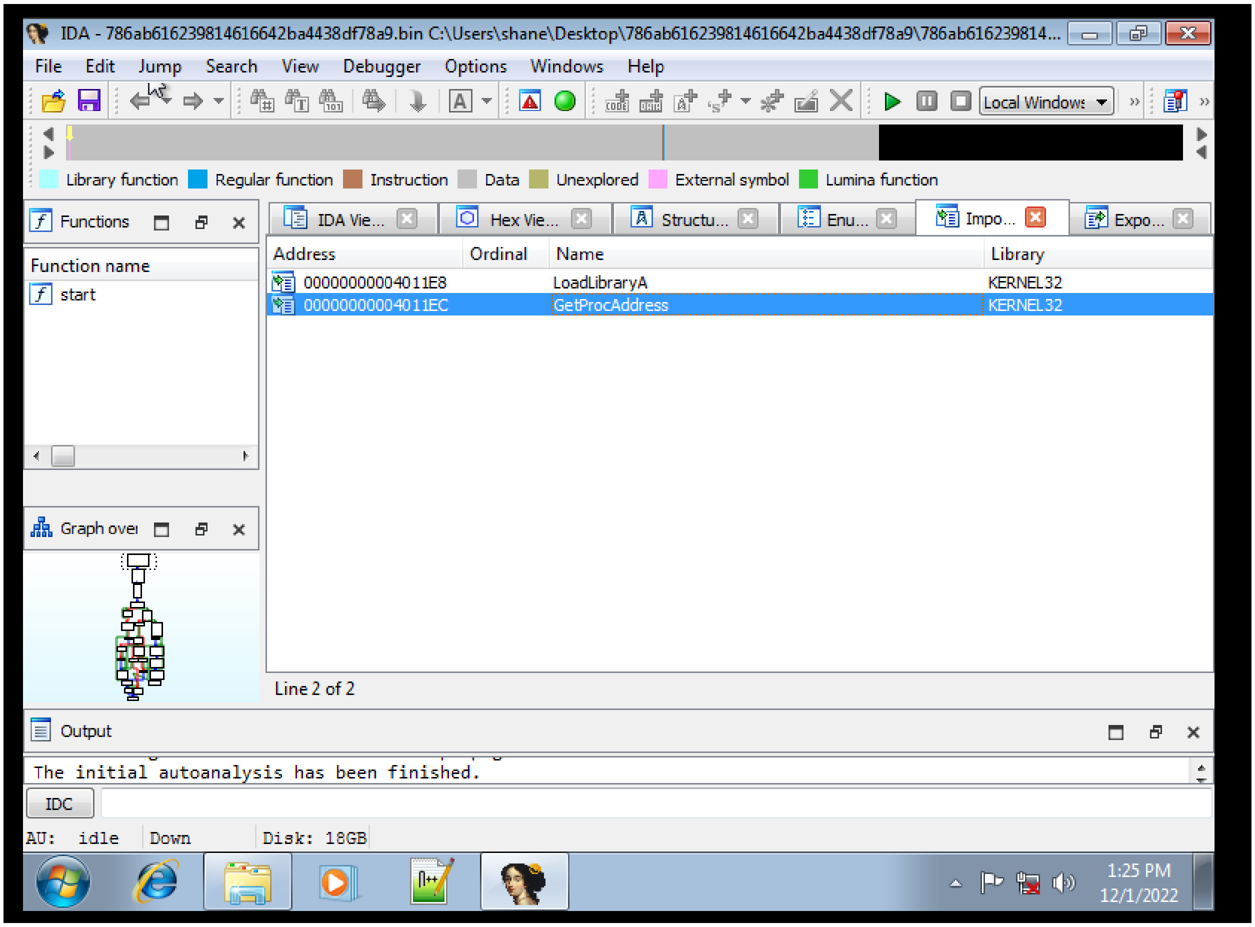


Figure 4: Important notification from IDA

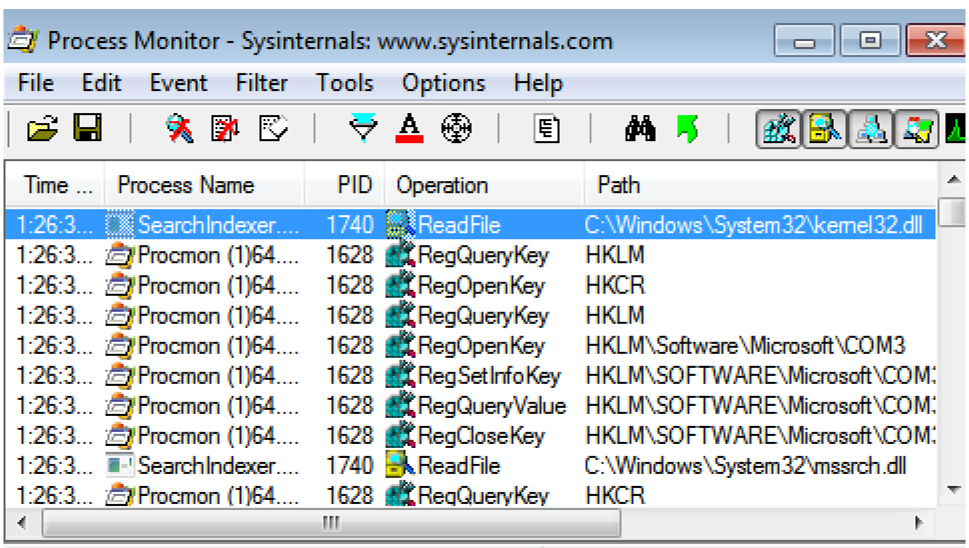
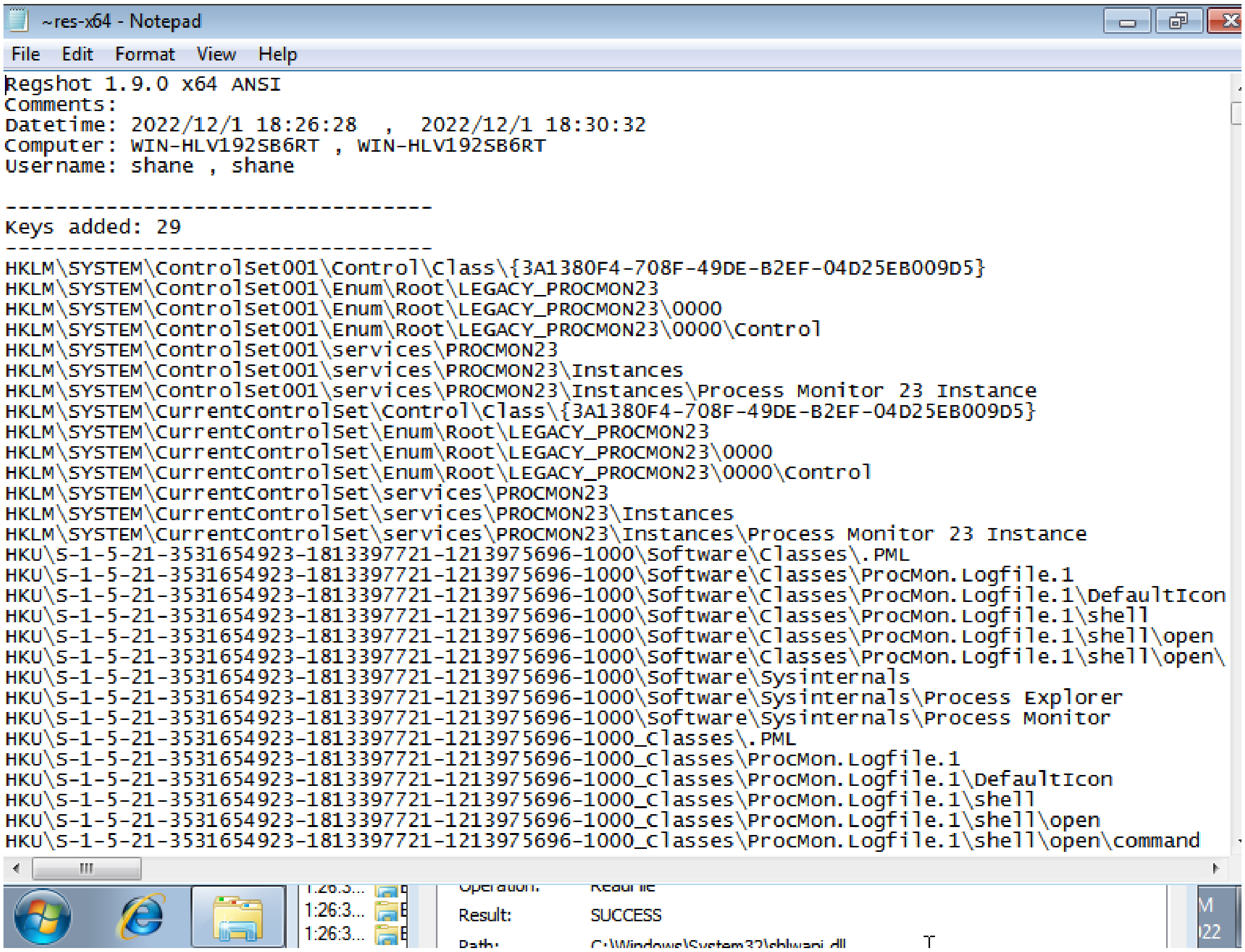


Figure 5: Procmon First Execution




```
Regshot 1.9.0 x64 ANSI
Comments:
Datetime: 2022/12/1 18:26:28  ,  2022/12/1 18:30:32
Computer: WIN-HLV192SB6RT , WIN-HLV192SB6RT
Username: shane , shane

----------------------------------
Keys added: 29
----------------------------------
HKLM\SYSTEM\ControlSet001\Control\Class\{3A1380F4-708F-49DE-B2EF-04D25EB009D5}
HKLM\SYSTEM\ControlSet001\Enum\Root\LEGACY_PROCMON23
HKLM\SYSTEM\ControlSet001\Enum\Root\LEGACY_PROCMON23\0000
HKLM\SYSTEM\ControlSet001\Enum\Root\LEGACY_PROCMON23\0000\Control
HKLM\SYSTEM\ControlSet001\services\PROCMON23
HKLM\SYSTEM\ControlSet001\services\PROCMON23\Instances
HKLM\SYSTEM\ControlSet001\services\PROCMON23\Instances\Process Monitor 23 Instance
HKLM\SYSTEM\CurrentControlSet\Control\Class\{3A1380F4-708F-49DE-B2EF-04D25EB009D5}
HKLM\SYSTEM\CurrentControlSet\Enum\Root\LEGACY_PROCMON23
HKLM\SYSTEM\CurrentControlSet\Enum\Root\LEGACY_PROCMON23\0000
HKLM\SYSTEM\CurrentControlSet\Enum\Root\LEGACY_PROCMON23\0000\Control
HKLM\SYSTEM\CurrentControlSet\services\PROCMON23
HKLM\SYSTEM\CurrentControlSet\services\PROCMON23\Instances
HKLM\SYSTEM\CurrentControlSet\services\PROCMON23\Instances\Process Monitor 23 Instance
HKU\S-1-5-21-3531654923-1813397721-1213975696-1000\Software\Classes\.PML
HKU\S-1-5-21-3531654923-1813397721-1213975696-1000\Software\Classes\ProcMon.Logfile.1
HKU\S-1-5-21-3531654923-1813397721-1213975696-1000\Software\Classes\ProcMon.Logfile.1\DefaultIcon
HKU\S-1-5-21-3531654923-1813397721-1213975696-1000\Software\Classes\ProcMon.Logfile.1\shell
HKU\S-1-5-21-3531654923-1813397721-1213975696-1000\Software\Classes\ProcMon.Logfile.1\shell\open
HKU\S-1-5-21-3531654923-1813397721-1213975696-1000\Software\Classes\ProcMon.Logfile.1\shell\open\
HKU\S-1-5-21-3531654923-1813397721-1213975696-1000\Software\Sysinternals
HKU\S-1-5-21-3531654923-1813397721-1213975696-1000\Software\Sysinternals\Process Explorer
HKU\S-1-5-21-3531654923-1813397721-1213975696-1000\Software\Sysinternals\Process Monitor
HKU\S-1-5-21-3531654923-1813397721-1213975696-1000_Classes\.PML
HKU\S-1-5-21-3531654923-1813397721-1213975696-1000_Classes\ProcMon.Logfile.1
HKU\S-1-5-21-3531654923-1813397721-1213975696-1000_Classes\ProcMon.Logfile.1\DefaultIcon
HKU\S-1-5-21-3531654923-1813397721-1213975696-1000_Classes\ProcMon.Logfile.1\shell
HKU\S-1-5-21-3531654923-1813397721-1213975696-1000_Classes\ProcMon.Logfile.1\shell\open
HKU\S-1-5-21-3531654923-1813397721-1213975696-1000_Classes\ProcMon.Logfile.1\shell\open\command
```


Figure 6: Regshot first execution

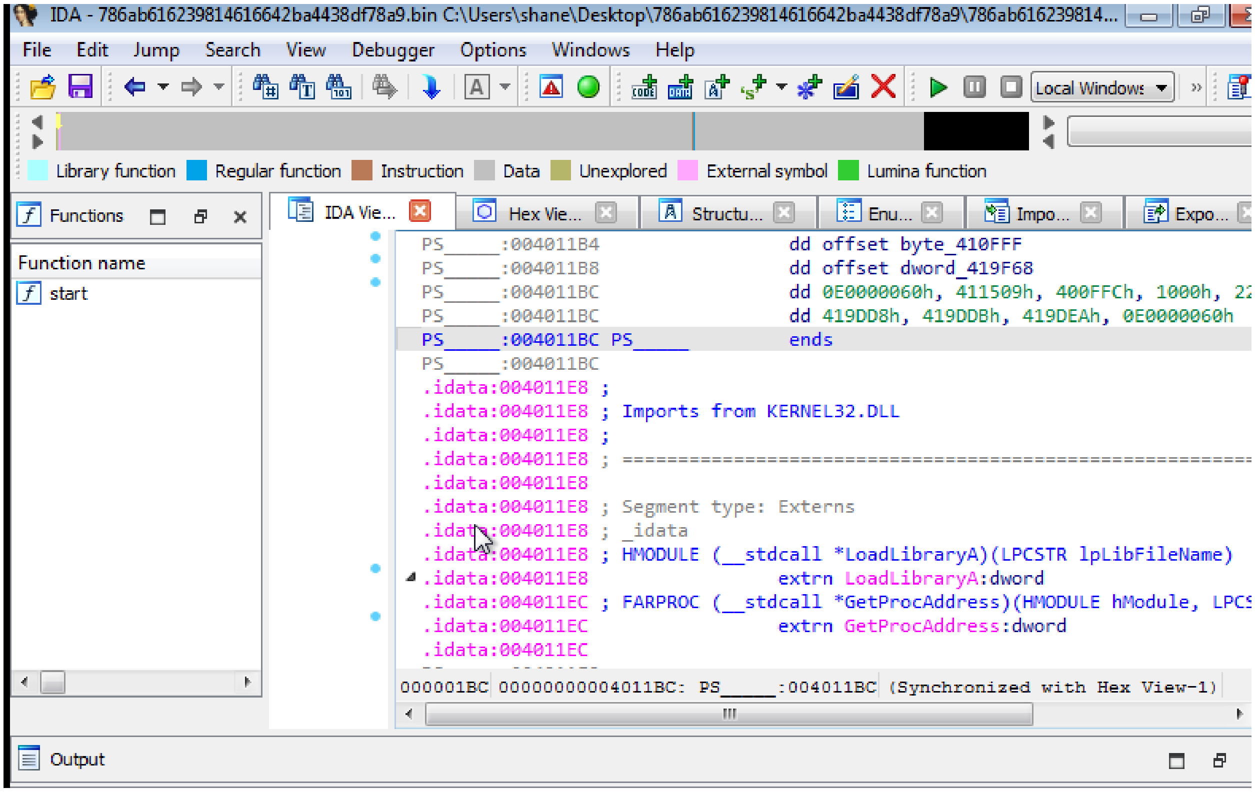


Figure 7: IDA after first execution

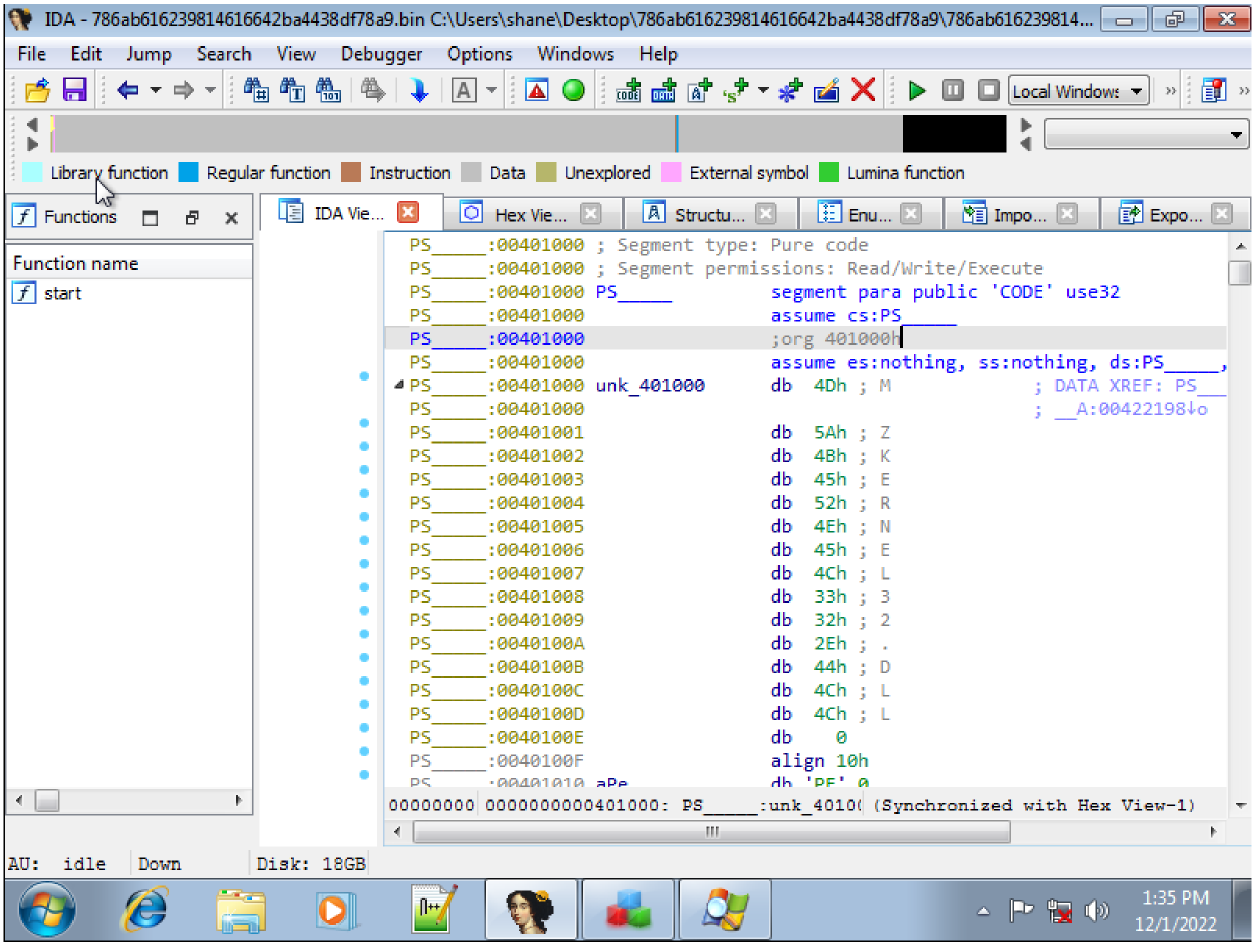


Figure 8: MZKERNEL32.dll showing up for the first time

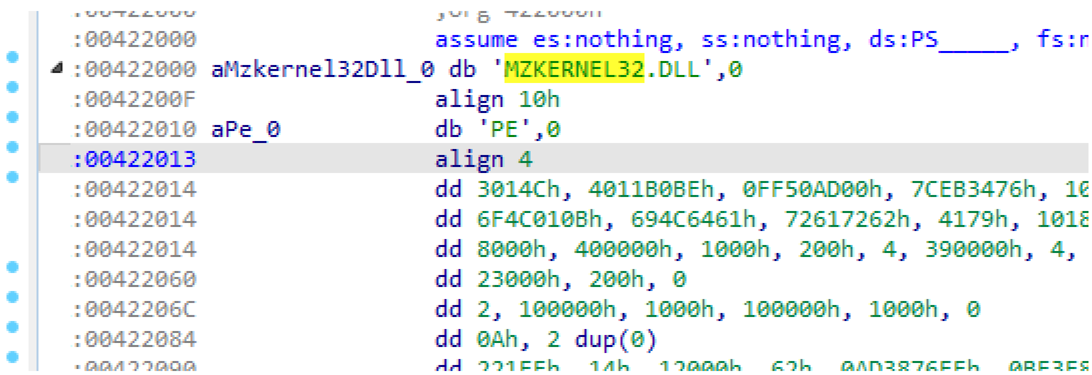


Figure 9: MZKERNEL32.dll being persistent

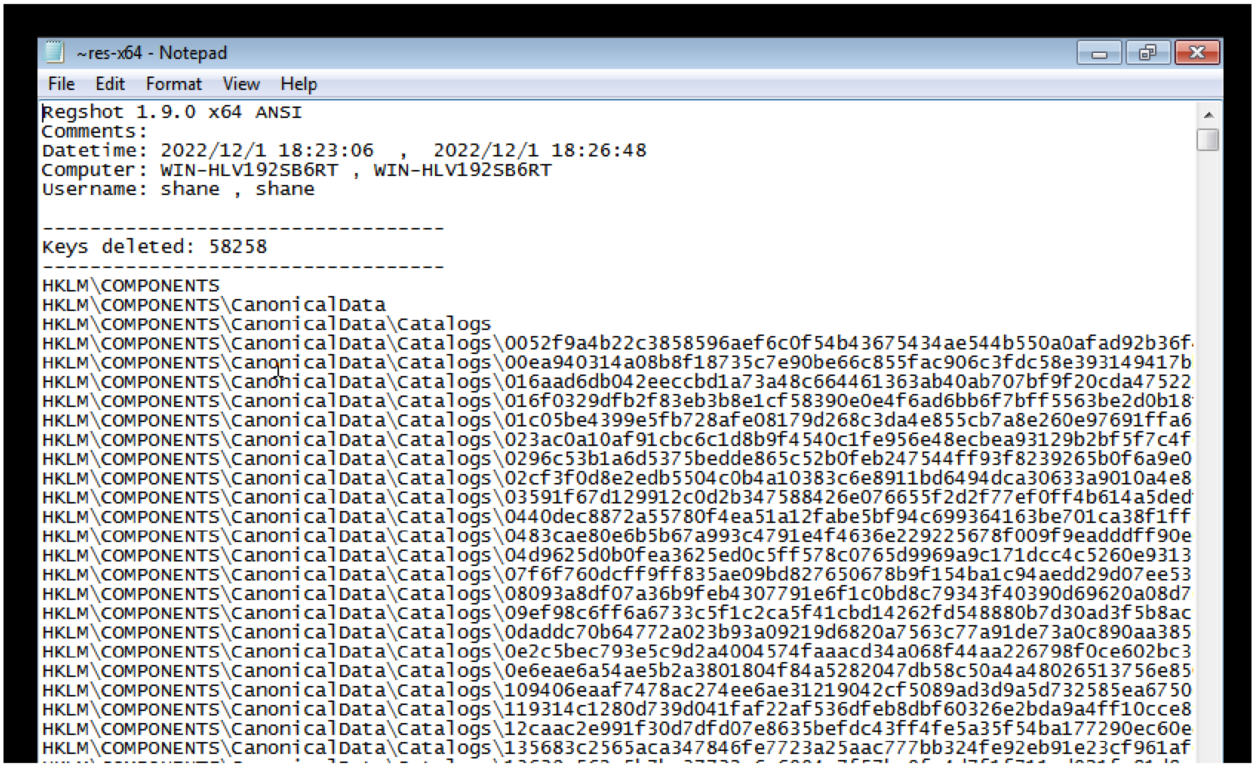


Figure 10: Regshot results from second execution

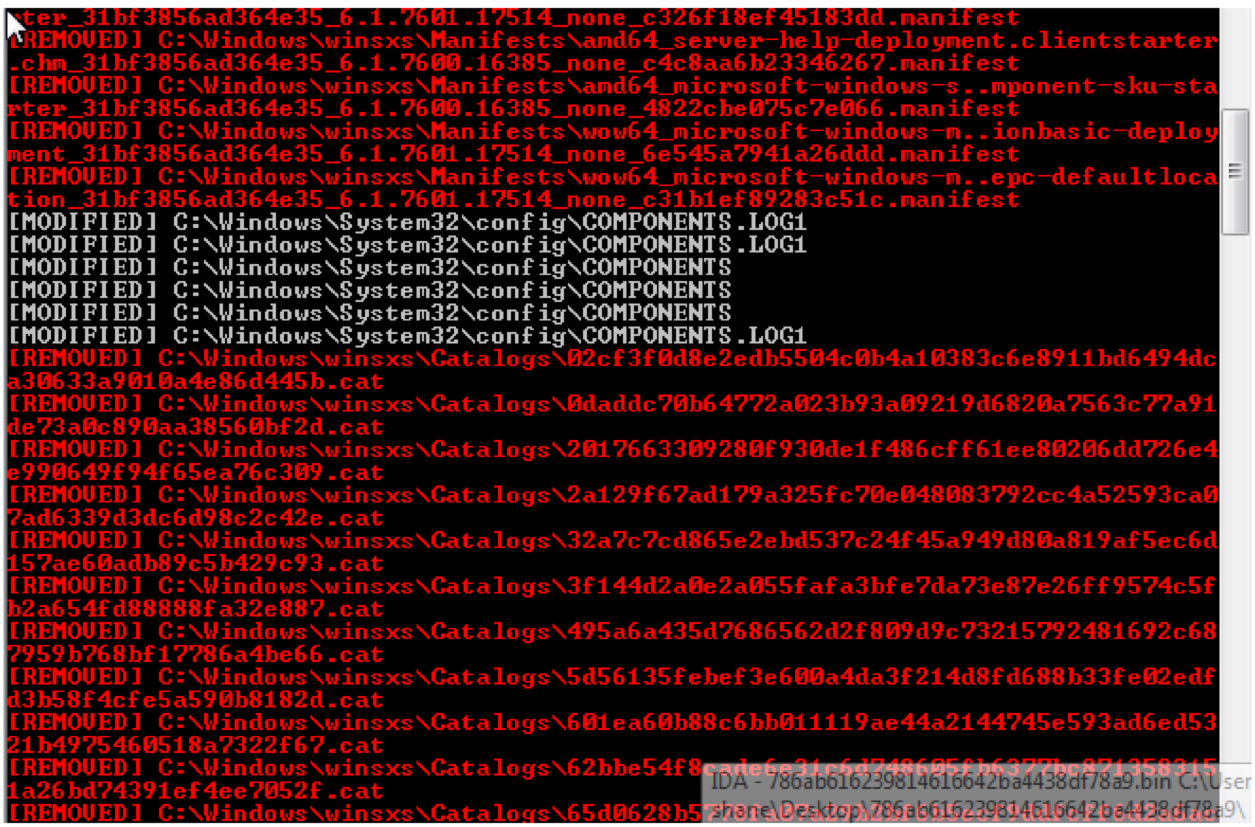


Figure 11: regfsnotify results from second execution

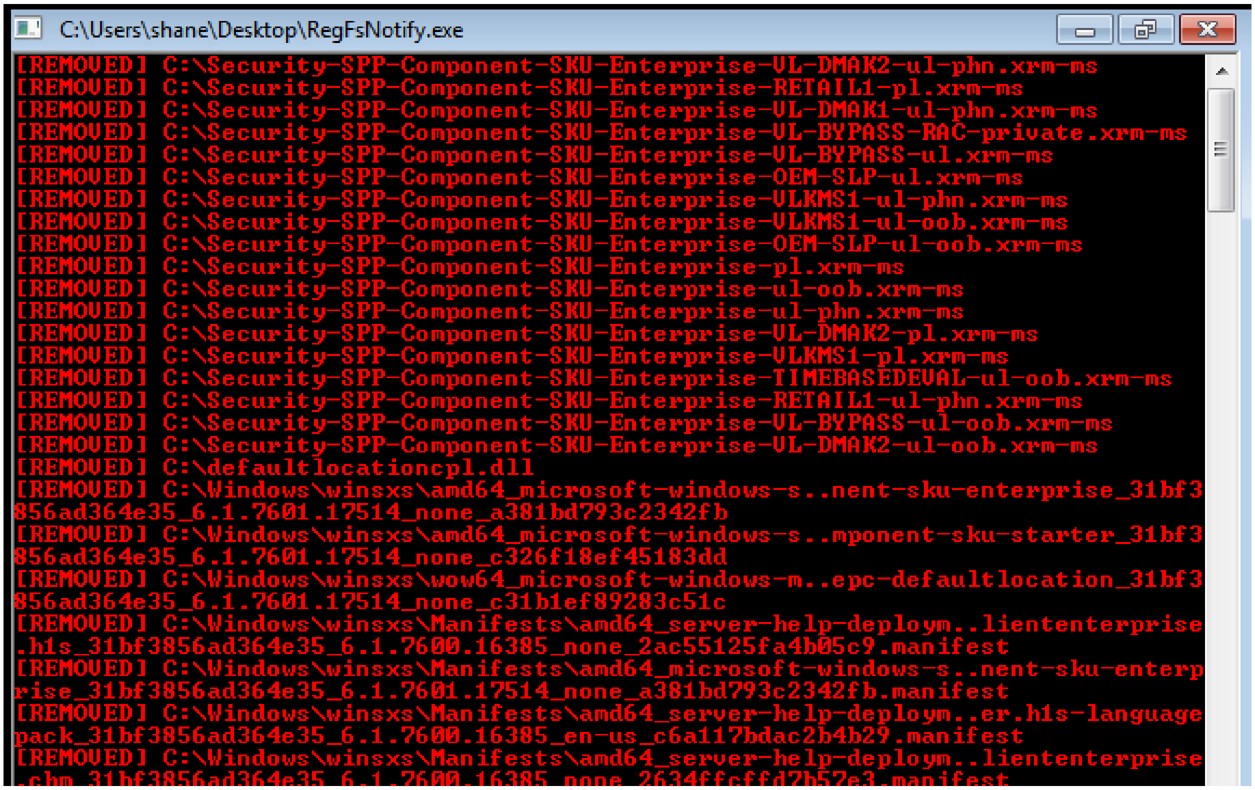


Figure 12: regfsnotify results from second execution cont.

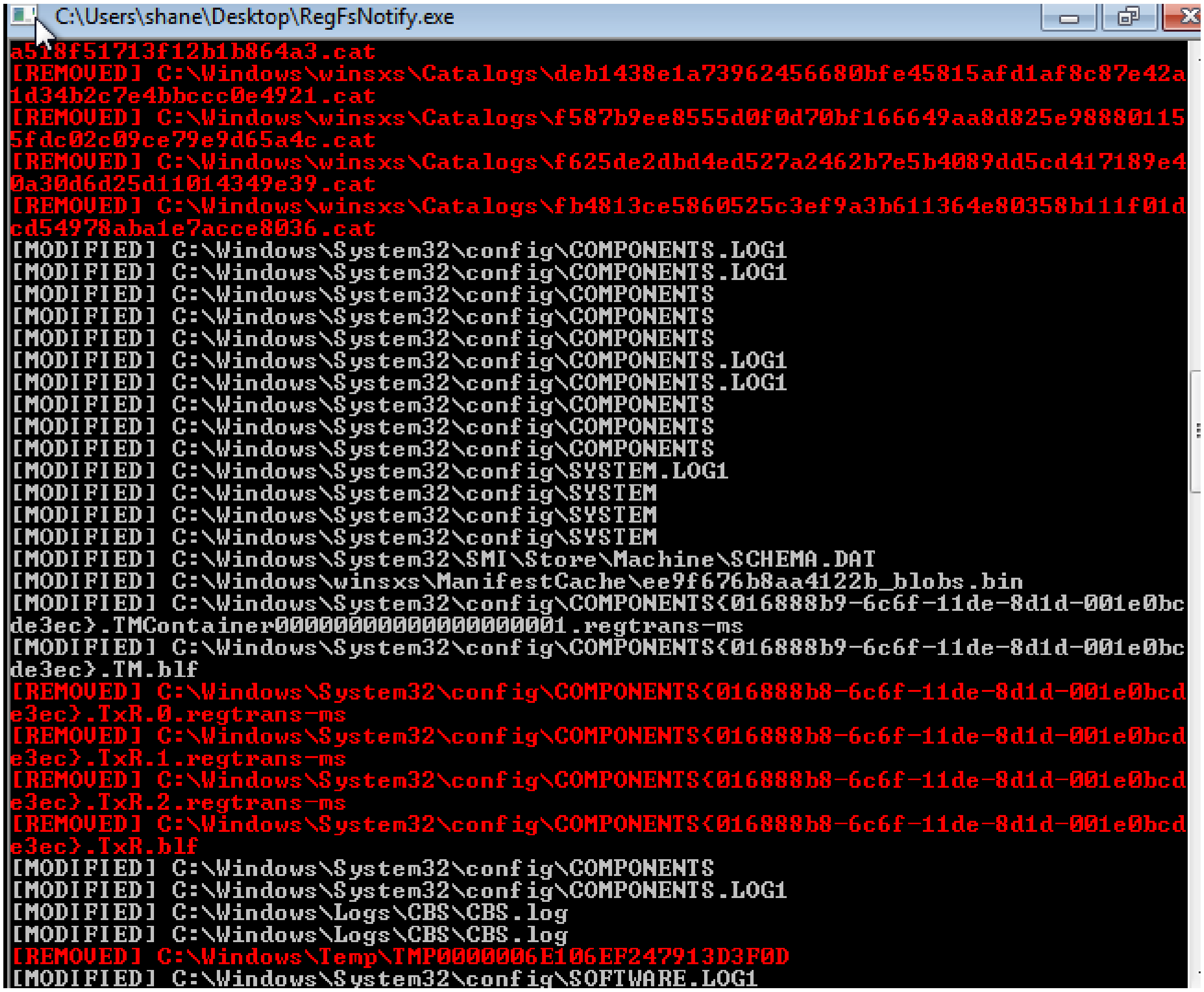


Figure 13: regfsnotify results from second execution final

III. Efforts

The total time for this solo project between configuration of the machine and tools, static and dynamic analysis, research regarding results from analysis, and development of this document was between 6-10 hours of work. There is a large gap between these numbers because configuring the VM and setting up the tools involved a lot of waiting for updates and installations as well as configuring my host machine to support another VM. The Windows 7 VM and the tools are the same ones from the Malware Analysis Workshop, so I followed along the guide once again when setting them up. The bulk of my effort was spent hands-on in dynamic analysis, going through logs and monitoring tools while the sample was running as well as after. There were tons of lines I read through to find the information needed to conclude this is a backdoor Trojan. It was supported by my static analysis findings as well as additional sources [1] [2] [3] [4].

IV. Conclusion

This report serves to document and explain the static and dynamic analysis of a malicious sample [5]. Through analysis techniques and online research [1] [2] [3] [4], it has been concluded that the sample is a Trojan backdoor bin file that targets KERNEL32.DLL. It is ultimately undecided whether this file performs further malicious actions such as installation of spyware or propagation. A series of figures is used to showcase the progress and results of configuration and analysis. The steps and thought process during analysis and the results are discussed in section II. Amounts of effort are quantified and explained in section III. For future analysis of samples, obfuscation techniques will be researched and better identified, as well as improvement in proficiency with IDA disassembly reading and understanding. Furthermore, a wider range of analysis tools can be learned and integrated for dynamic analysis to find otherwise hidden or unknown attributes of potentially malicious files.

V. References

1. Mattz (2018). Is HKLM\SOFTWARE\MICROSOFT… a threat/PUP? MalwareBytes. https://forums.malwarebytes.com/topic/220671-is-hklmsoftwaremicrosoft-a-threatpup/
2. Grazfather (2016). Practical Malware Labs. Github. https://github.com/Grazfather/PracticalMalwareLabs/blob/master/chapter18/readme.md
3. Sokolov, D. (2018). Mzkernel32.dll – Dangerous. Greatis. https://www.greatis.com/appdata/d/m/mzkernel32.dll.htm
4. Adaware (2021). Trojan.GenericKD.3278476_e090554364. https://support.adaware.com/hc/en-us/articles/4405521856916-Trojan-GenericKD-3278476-e090554364
5. Fabrimagic72 (2020). Malware-samples. https://github.com/fabrimagic72/malware-samples
6. Techopedia (2022). Kernel32.dll. https://www.techopedia.com/definition/3379/kernel32dll#:~:text=operations%20and%20interrupts.-,Kernel32.,other%20system%20or%20user%20processes.