Static & Dynamic Malware Lab
Antivirus Engines
Flare VM: ApacheMOD1.exe
Date: 08/01/2024
Examiner Name: Jonathan Brendese

Table of Figures

Figure 1: Logging into Engage……6
Figure 2: 'Lab4_CYB643_sample.zip'……6
Figure 3: 'Infected'……..7
Figure 4: Hash File……...7
Figure 5: File Signatur..……...8
Figure 6: ApacheMod1.exe. properties……...8
Figure 7: PEStudio……...9
Figure 8: imports (flag>156)……9
Figure 9: WriteFileFunction...……...10
Figure 10: LookUpPriviges……...……11
Figure 11: .text/Entropy/VirtualAddress……...12
Figure 12: File Type PE32+ ……..13
Figure 13: ApacheMOD1.exe: applications…….....13
Figure 14: Floss……14
Figure 15: Decrypting Files……...15
Figure 16: Paths to systems of user directories……..15
Figure 17: Terminate Thread……16
Figure 18: WritePrivateProfileStringA……..16
Figure 19: pestuido strings…….....17
Figure 20: Analysis of Strings……...……18
Figure 21: Detect It Easy ApacheMOD1.exe……19
Figure 22: PE Detective ApacheMOD1.exe……..19
Figure 23: Wireshark……20
Figure 24: Transmission Control Protocol……20
Figure 25: Notepad.exe……..21
Figure 26: Procmon File Data Creation #1……22
Figure 27: Procmon File Data Creation #2……22
Figure 28: IXP001 Unfindable File……...23
Figure 29: RegCreateKey Sample……24
Figure 30: RegCreateKey Sample/DataStore……25
Figure 31:RegCreateKey Sample/OOBE Launch User……25
Figure 32:RegCreateKey Sample\RunOnce……..26
Figure 33: Regshot Compare #1……27
Figure 34: Regshot Compare #2……27
Figure 34A: API Monitor……..27
Figure 35: ApacheMOD1.exe……....29
Figure 36: Kernel32.DLL……..30
Figure 37: ProcWatch……31
Figure 38: Fakenet #1……31
Figure 39: Fakenet #2……32
Figure 40: UPX Packing……33
Figure 41: Unzipped UPX……33
Figure 42: NewApache.exe……34
Figure 43: Packed Hash File NewApache……34

Figure 44: Packed File Signature……...35
Figure 45: Packed NewApache.exe properties……..36
Figure 46: pestudio comparison…….....37
Figure 47: Import Comparison……...38
Figure 48: Pestudio UPX……...39
Figure 49: Packed File Type……..40
Figure 50: Packed MIME……...41
Figure 51: FLOSS Comparisons……42
Figure 52: Detect It East ApacheMOD1.exe……43
Figure 53: Detect It Easy NewApache.exe……44
Figure 54: Detect It Easy NewApache.exe Entropy……..44
Figure 55: Themida Download……..45
Figure 56: Unpacked Themida……..45
Figure 57: Themida x64……46
Figure 58: Themida……47
Figure 59: Themida Packed ApahceMOD1_protected.EXE……48
Figure 60: Themida Packed ApahceMOD1_protected.EXE File Signature……48
Figure 61: Themida Packed ApahceMOD1_protected.EXE Compatibility……49
Figure 62: Pestudio Themida……50
Figure 63: (PE32+ executable (GUI)) x86-64, for MS Windows, 10 sections……51
Figure 64: ApacheMOD1.exe: application/vnd.microsoft.portable- executable; charset=binary.52
Figure 65: Floss ApacheMOD_Protected.EXE……53
Figure 66: VirusTotal ApacheMOD1.EXE……...54
Figure 67: Virus Total NewApache.exe……55
Figure 68: Virus Total ApacheMOD_Protected.exe)……56
Figure 69: Comparison of detected and undetected files……...57

Introduction

This lab will follow the procedures included within a dynamic and static analysis of malware, specifically ApacheMOD1.EXE. A dynamic analysis is a method utilized to evaluate the behavior of malware within a controlled setting. Executing a dynamic assessment allows the user to study the malware's intent, functionality, and potential impact on infected systems within a safe environment. In other words, a dynamic assessment observes malware in real-time, whereas a static analysis is confined to just studying the code structure (Jones, 2021).

Then, the lab will include the procedures that are involved when packing a file. Packing a file refers to compressing a file or group of files into a smaller, more manageable size, typically for storage purposes. However, threat actors have a different use for packers. Nevertheless, packing entails using a file compression algorithm to reduce the amount of real estate a file takes up on a storage medium. Packing a file can include compression, encryption, and or universal, which entails both compression and encryption (MITREATT&CK, n.d.).

UPX, Ultimate Packer for eXecutables, is the first packer utilized to pack ApacheMOD1.EXE; UPX is an open-source executable packer. According to github.io (n.d.), UPX is an advanced executable file compressor that will reduce the size of programs and DLLs by 50 to 70%. UPX will reduce disk space, network load, download, and overall obfuscation. UPX supports various executable formats, such as Windows programs and DLLs, macOS applications, and Linux executables (Yadav, 2021).

Themida, developed by Oreans Technologies, will also be used to pack ApacheMOD1.EXE and compared to UPX's abilities. According to Oreans.com (n.d.), Themida is a software protection program that prevents reverse engineering and the "cracking" of software applications. Themida uses advanced techniques such as code virtualization, obfuscation, and anti-debugging to protect software from suspicious use and tampering. Many features Themida has, such as anti-debugger techniques, anti-API scanners, different encryption algorithms and keys in each protected application, virtual machine emulation, advanced API-Wrapping techniques, and many more, making Themida a superior product.

Lastly, Flare VM will be used to pack and conduct static analyses. Flare VM is a Windows-based virtual machine developed by Fire Eye that is utilized for malware analysis, reverse engineering, and incident response tasks. Flare VM Is installed preconfigured with a comprehensive set of tools, frameworks, and scripts that can be utilized to evaluate binaries. Lastly, Flare VM is constantly updated and maintained by Fire Eye's experts for effective malware analysis (Mandiant, n.d.).

Objective

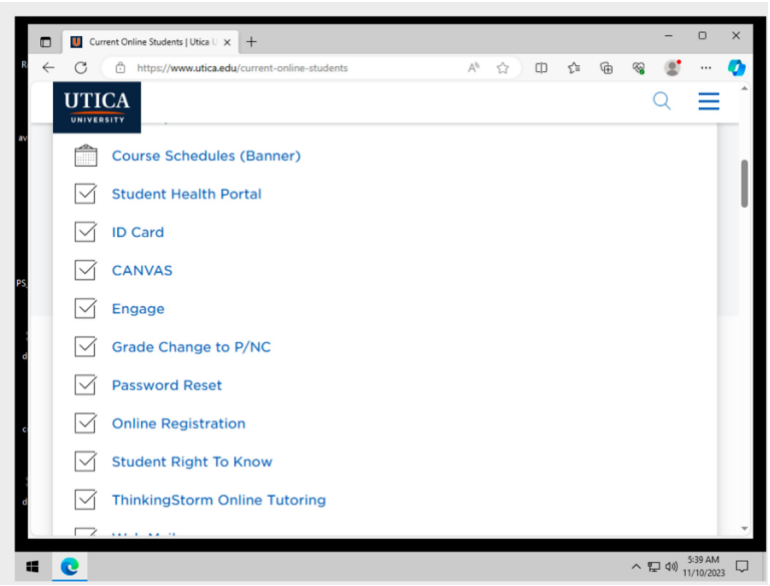
The user will execute a dynamic analysis on a binary using Flare VM to observe the ApacheMOD1.EXE's full potential. The user will then pack ApacheMOD1.EXE using both UPX and Themida packers. Static analyses will be conducted on files NewApace.exe and ApacheMOD1_Protected.EXE and compared to one another.

In addition, the user will use VirusTotal to compare and contrast packing effectiveness between UPX and Themida. Information such as data from various antivirus products and online scan engines will be investigated throughout the assessment process.

Lab Work

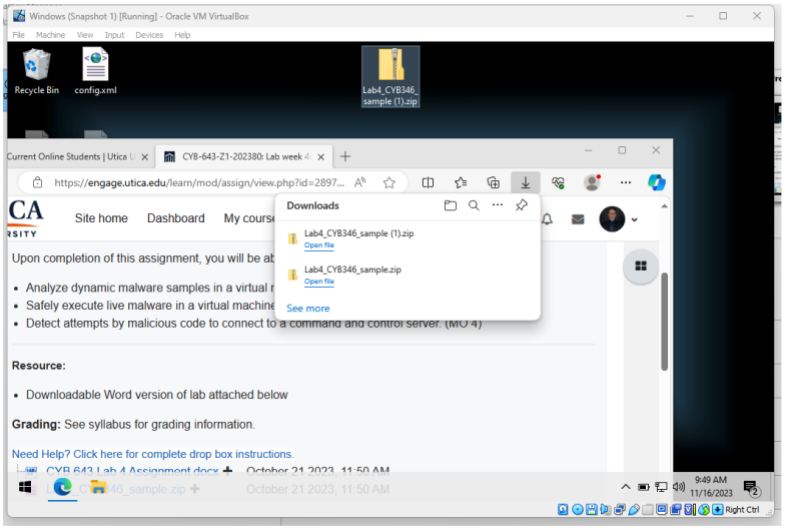
The user downloaded the 'Lab4_CYB643_sample.zip' from the Engage course shell to the Flare VM desktop (see Figure 1).

Figure 1: Lab4_CYB643_sample.zip



Then, the user downloaded 'Lab4_CYB643_sample.zip' from the Engage course shell to the Flare VM desktop (see Figure 2).

Figure 2: 'Lab4_CYB643_sample.zip'



The user right-clicked on the zip file and unpacked it into a file on the desktop called ‘Malware Analysis.’ In order to unpack the zip file, the user had to enter the password ‘infected.’ Then, the file was populated into the folder as ‘ApacheMOD1.exe’ (see Figure 3).

Figure 3: ‘Infected’

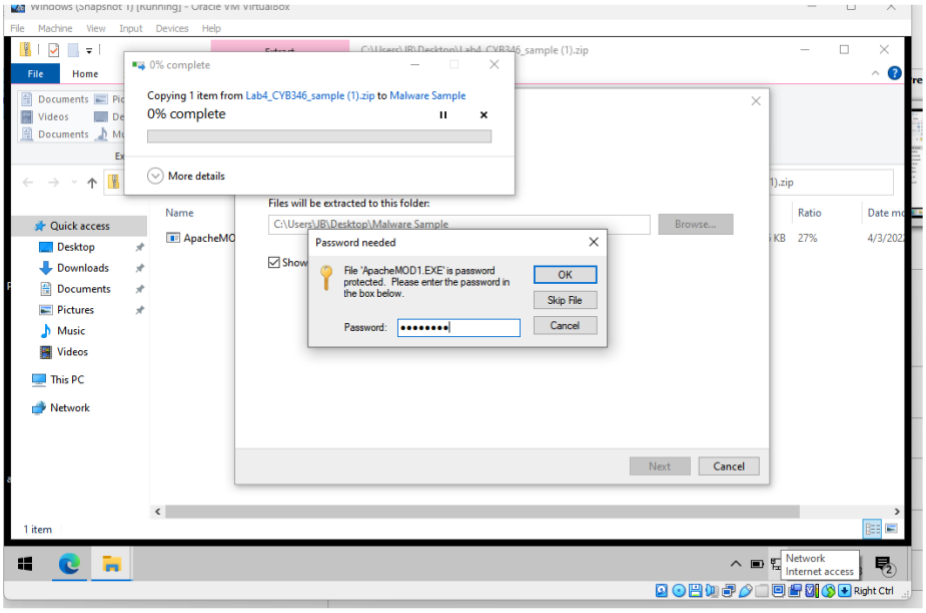
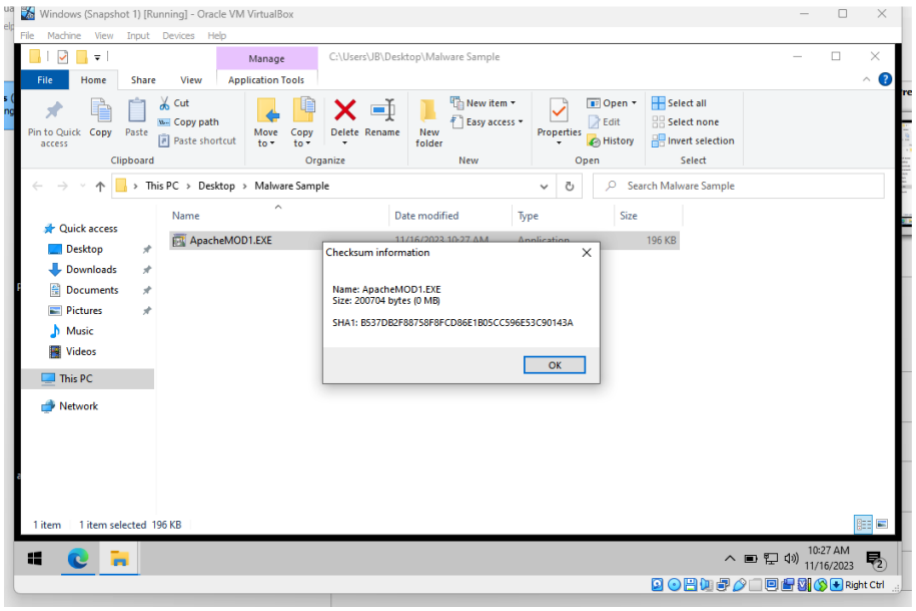


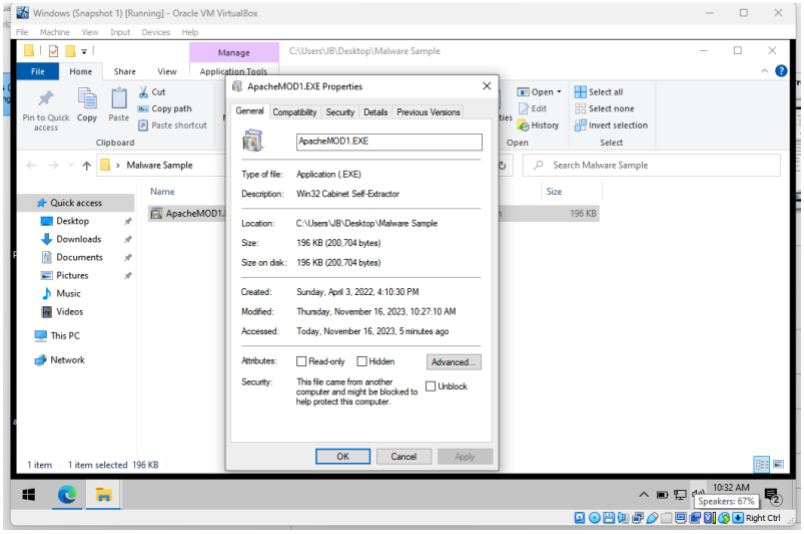
Figure 4: Hash File



,

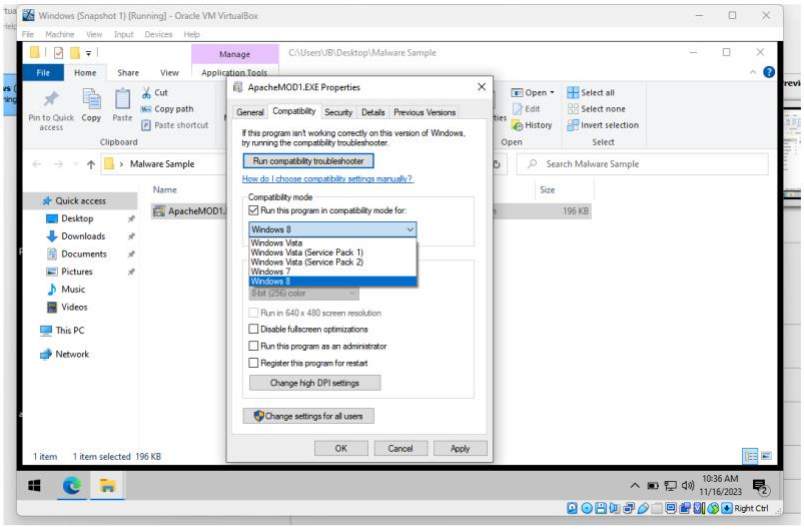
The user then right-clicked on the file again to inspect for Metadata. The user validated that there was no digital signature, suggesting a security risk. A digital signature validates who created the file and if the file has been altered or corrupted (Assaad, 2021) (see Figure 5).

Figure 5: File Signature



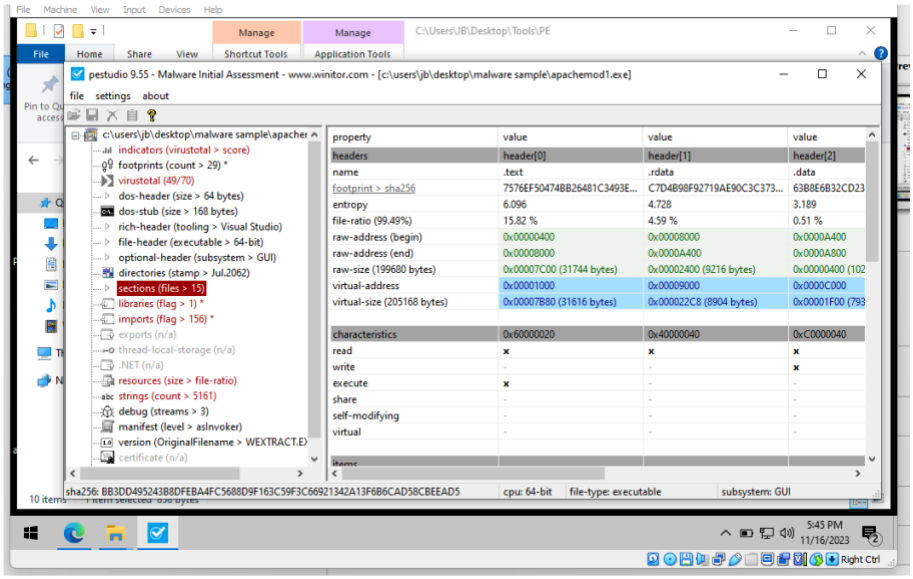
The user right-clicked on the file, clicked on ‘compatibility,’ checked ‘run this program in compatibility mode for,’ and then clicked the down arrow to identify the various Windows OS (see Figure 6); the choices evidenced a 64-bit architect type.

Figure 6: ApacheMOD1.exe properties



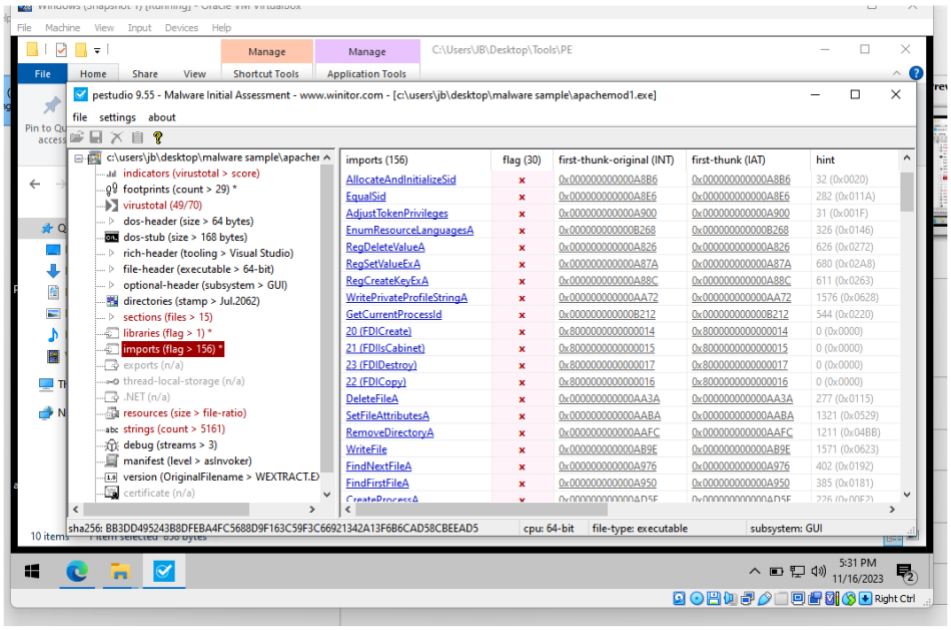
Then, the user executed ‘pestudio 9.55’ from the tools included within the Flare VM suite. The user dragged the ApacheMOD1.exe file over for the program to evaluate its contents (see Figure 7: pestudio).

Figure 7: pestudio



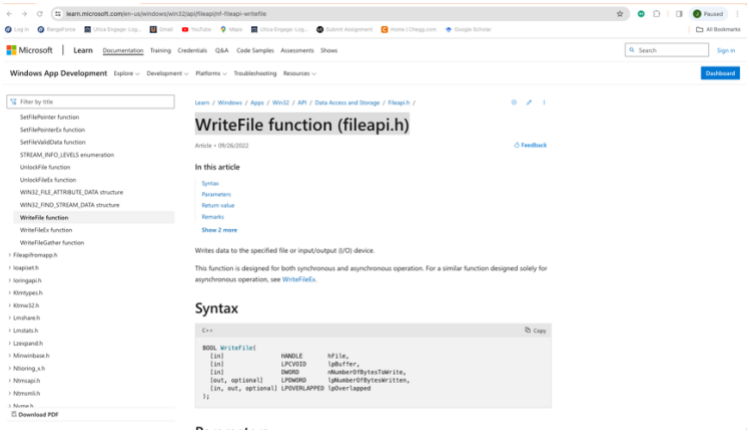
The user utilized the program and allocated 30 files flagged by the program (see Figure 8).

Figure 8: imports (flag > 156



The user investigated several of the 32 files to evaluate for malicious conduct. For example, ‘WriteFile’ function (fileapi.h)’ can be utilized to create or modify files, overwrite critical system files, save stolen data from the user such as personal information, credentials, keylogging data, etc, and propagate by creating copies of itself or additional malware components to the users' network (Alvinashcraft, n.d.) (see Figure 9).

Figure 9: ‘WriteFile function (fileapi.h)’



Also, the user made note of ‘LookupPrivilegeValueA’ as a potential tool used for malicious purposes, as this can be utilized for escalating privileges. In other words, malware or other security compromising software can use this function as part of an exploit. Using ‘LookupPrivilegeValueA’ will allow threat actors to find the LUID for a privilege it wants to add to its token before trying to adjust the tokens privileges (Alvinashcraft, n.d.) (see Figure 10).

Figure 10: ‘LookupPrivilegeValueA’

Learn / Windows / Apps / Win32 / API / Security and Identity / Winbase.h /

LookupPrivilegeValueA function (winbase.h)

Article • 02/08/2023 Feedback

In this article

Syntax
Parameters
Return value
Remarks
Show 2 more

The **LookupPrivilegeValue** function retrieves the locally unique identifier (LUID) used on a specified system to locally represent the specified privilege name.

Syntax

C++ Copy

```
BOOL LookupPrivilegeValueA(
  [in, optional] LPCSTR lpSystemName,
  [in]           LPCSTR lpName,
  [out]          PLUID  lpLuid
);
```

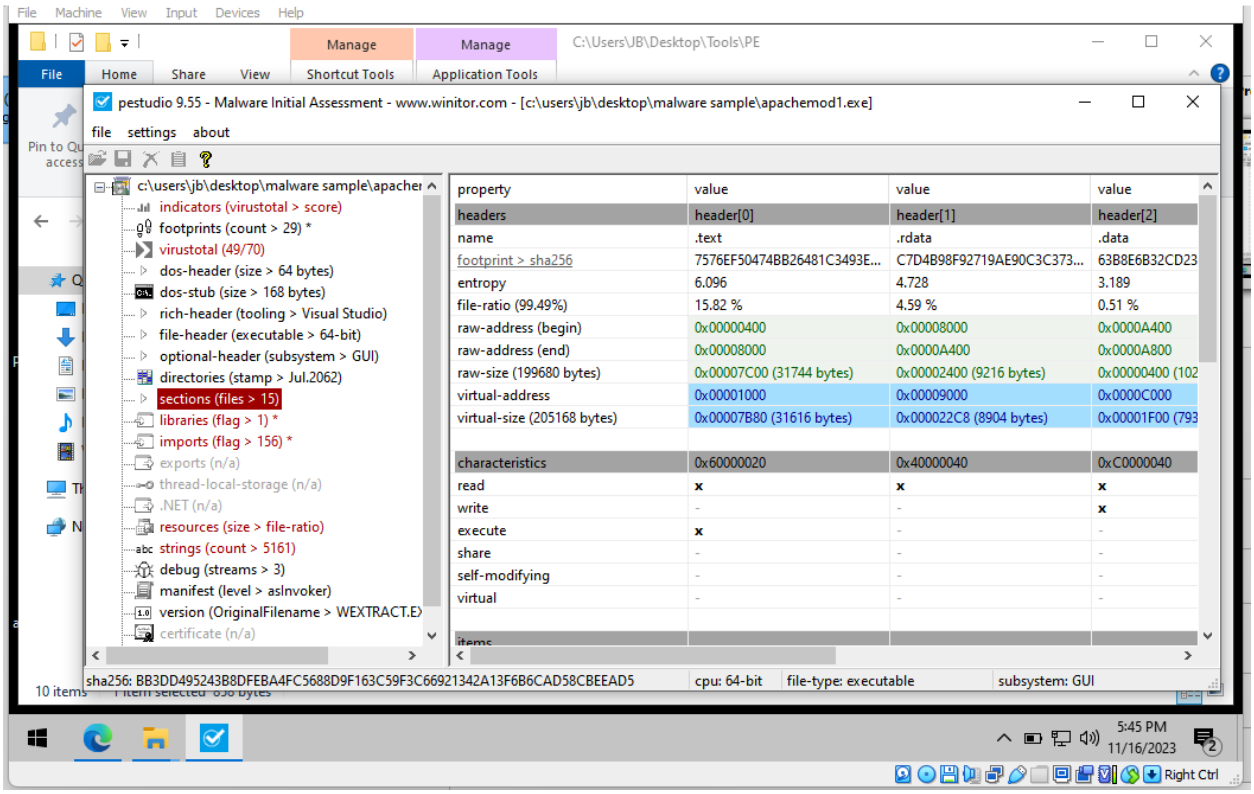
Parameters

`[in, optional] lpSystemName`

A pointer to a null-terminated string that specifies the name of the system on which the privilege name is retrieved. If a null string is specified, the function attempts to find the privilege name on the local system.

As noted during the previous lab, *Pestudio 9.55* was heavily utilized throughout the course of the static analysis to obtain information. The user used it to identify various sections and entropy; the higher the entropy, the more likely randomness and disorder data sets. However, this yields encryption and the use of packers. Standard "native" executables have an entropy of 6.8 or less. Entropy that is closer to 8 or greater indicates obfuscation or compression (see Figures 11). Please note that the virtual address for the .text value is 0x00001000, and the entropy is 6.2.

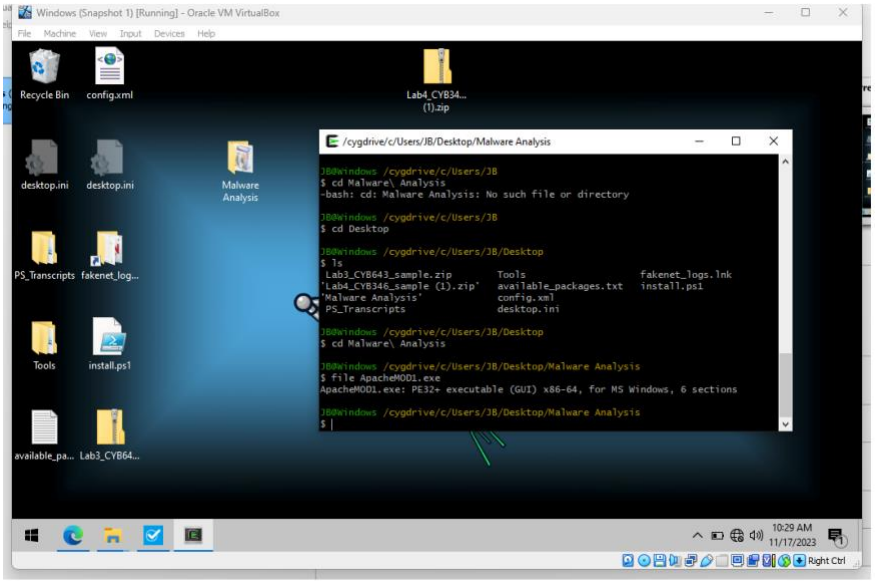
Figure 11: .text/Entropy/Virtual Address



Several sections are included in the sections portion of pestudio 9.55, such as .text, .rdata, .data, .pdata, .rsrc, and .reloc. The virtual addresses are assigned to each value (e.g., .text 0x00001000; .rdata – 0x00009000; .pdata – 0x0000E000; .data – 0x0000C000; and /reloc – 0x00035000.

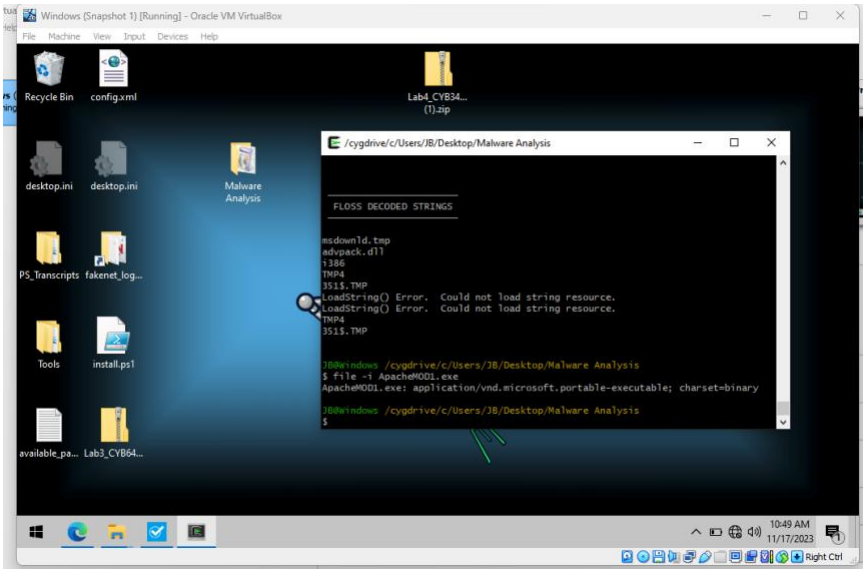
The user then opened the start menu/Cygwin/Cygwin64 Terminal/cd C:\Users/cd JB\/cd Desktop/\cd Malware\ Analysis/file apache.exe (see Figure 12). This identifies the file type (PE32+ executable (GUI)), executable file format in both 32 and 64-bit versions, and that six sections make up the structure of the executable file (Karl-Bridge-Microsoft, n.d.).

Figure 12: File Type PE32+ executable



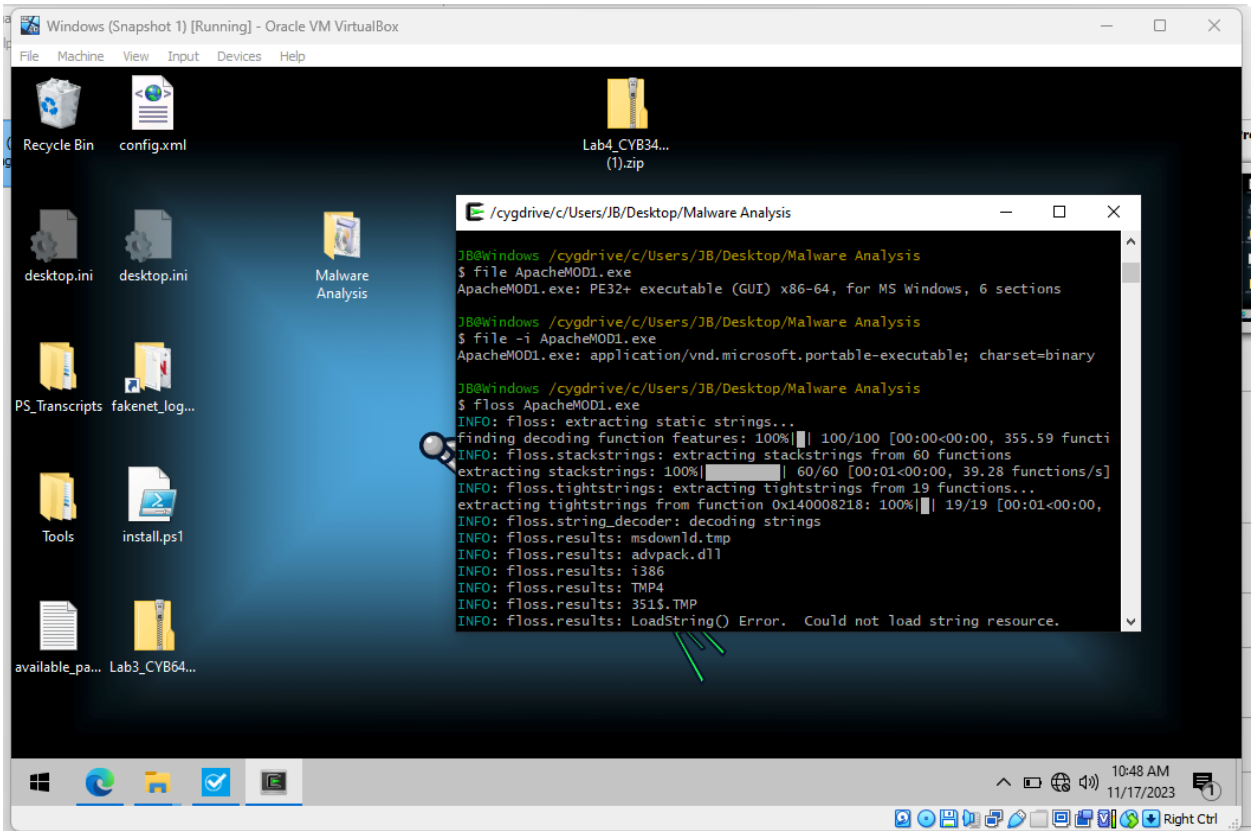
The user then executed 'file -i ApacheMOD1.exe to obtain the MIME Type, which is 'apache.exe: application/vnd.microsoft.portable-executable; charset=binary' (see Figure 13).

Figure 13: ApacheMOD1.exe: application/vnd.microsoft.portable- executable; charset=binary



The user opened the file within the folders icon (C:\Users\JB\Desktop\Malware Analysis\ApacheMOD1.exe) and entered ‘cmd,’ which opened Windows Command Prompt. The user entered ‘floss apache.exe’ to utilize the “floss” (Free/Libre and Open Source Software) program, which compiled strings of source code within an executable file (see Figure 14).

Figure 14: Floss



The next couple of figures will highlight some suggestive malware activities, such as decrypting files (see Figure 15), paths to systems or user directories (see Figure 16), terminate thread (see Figure 17), and writeprivateprofilestringA (see Figure 18).

Figure 15: Decrypting Files

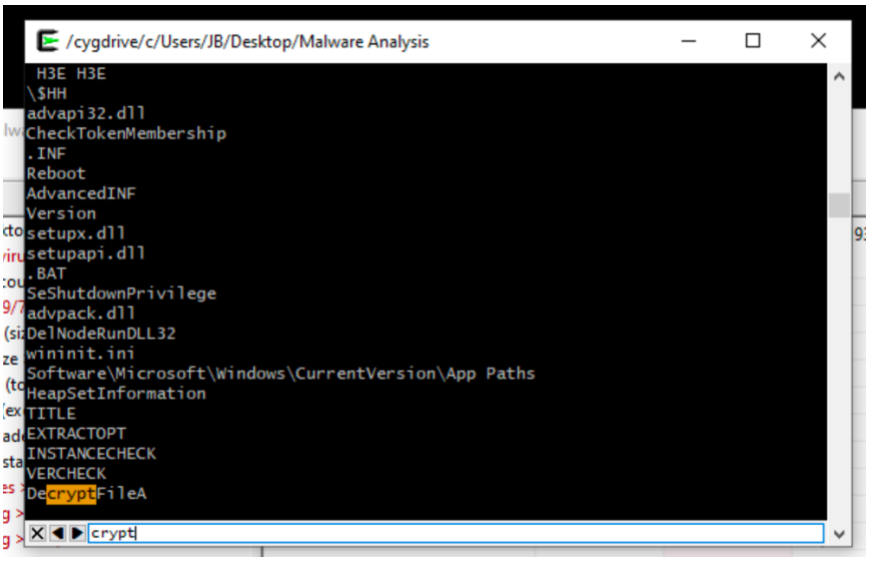


Figure 16: paths to systems or user directories

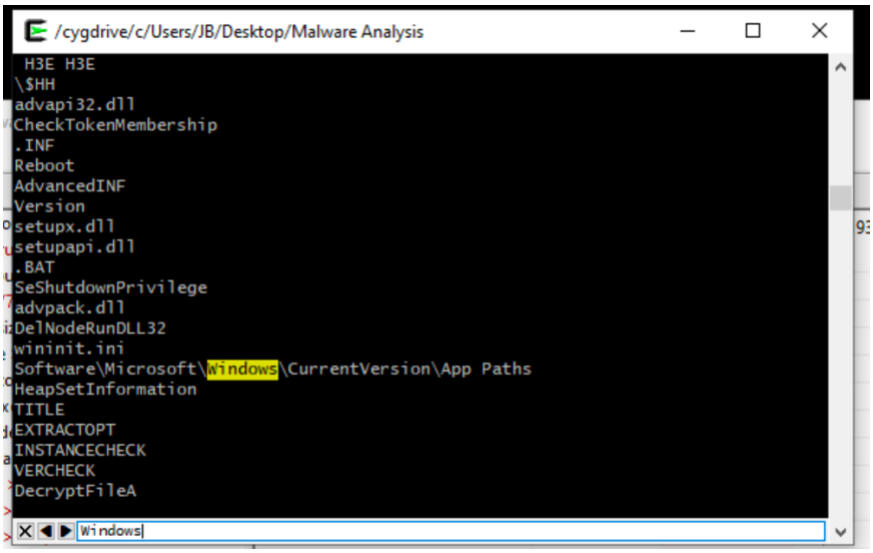


Figure 17: TerminateThread

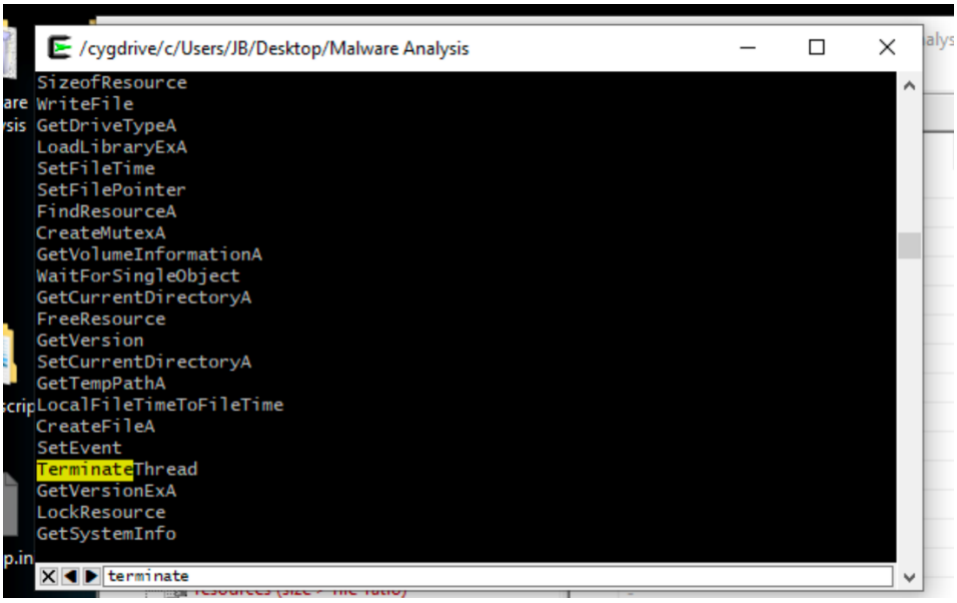
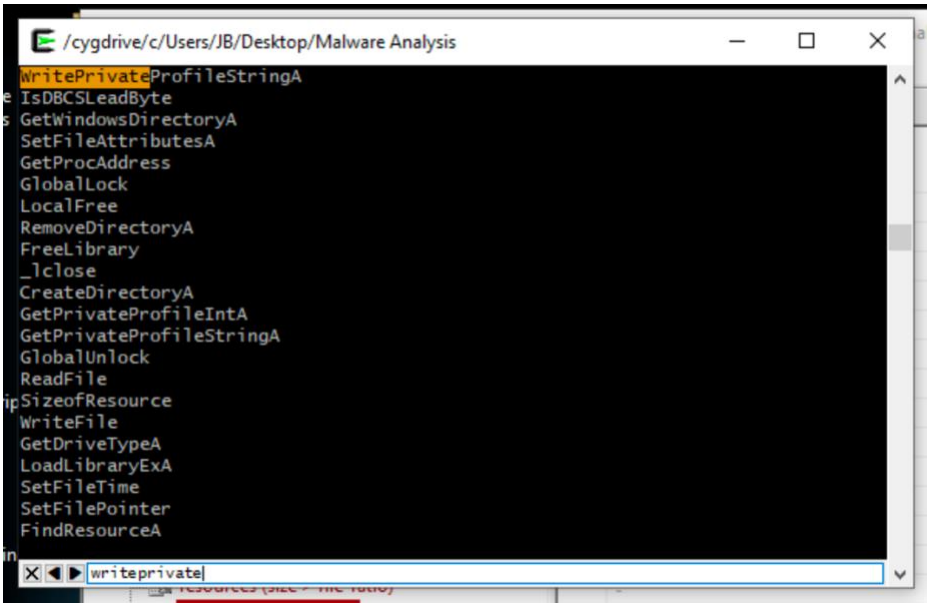
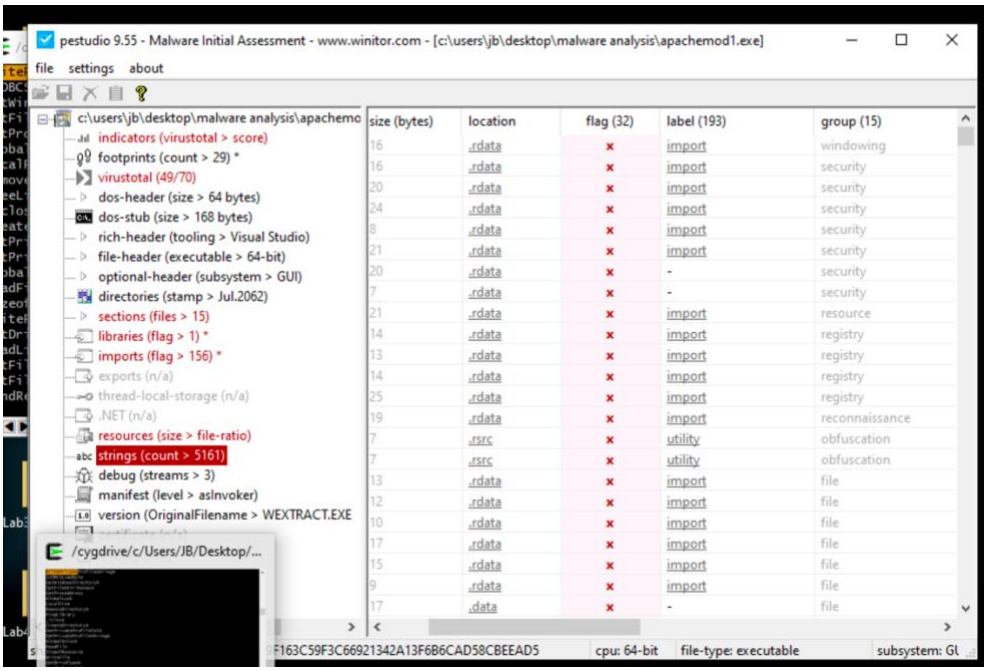


Figure 18: WritePrivateProfileStringA



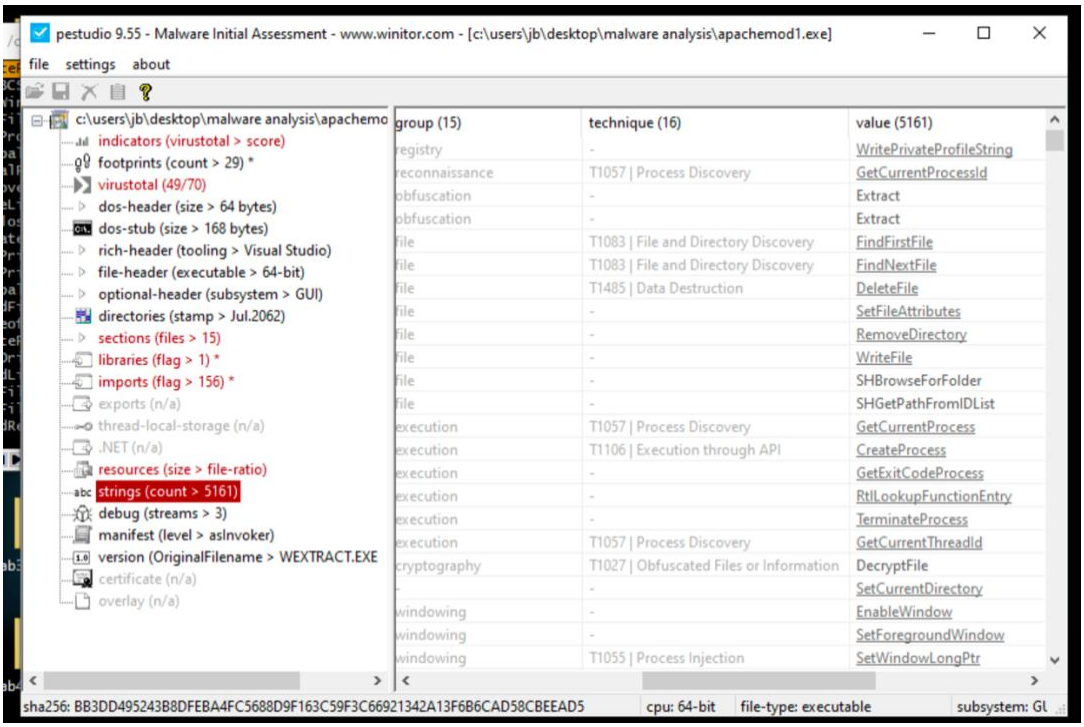
The user also utilized the strings function within the pestudio program in order to assist with identifying possible malicious strings (see Figure 19).

Figure 19: *pestudio* strings



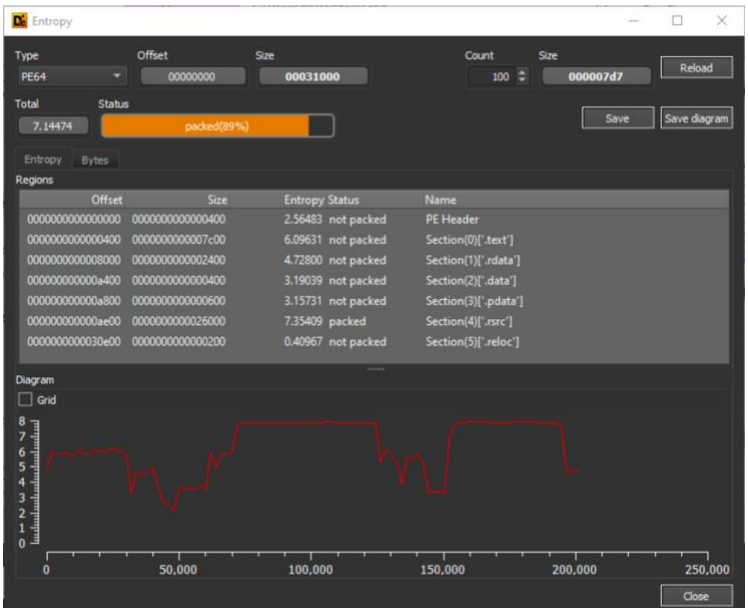
The user analyzed some of the 30 strings that populated within the source code that could be utilized maliciously by a threat actor. For example, T1134 (Access Token Manipulation/.rdata) can be utilized to escalate privileges. Another possible malicious string is (T1106 Create Process), which allows the threat actor to create a process that could be used for malicious code. Technique T1027, also known as "Obfuscated Files of Information," can assist with keeping shellcode or malicious files hidden. Other suspect files include T1485 Data Destruction, T1112 Modify Registry, T1057 Process Recovery, and T1083 File and Directory Discover (Alvinashcraft, n.d.) (see Figure 20).

Figure 20: Analysis of Strings for ApacheMOD1.exe



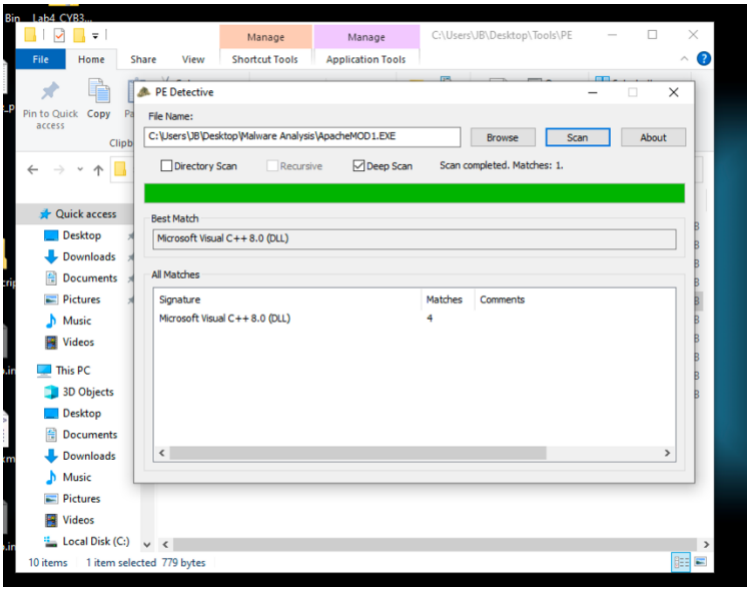
The user executed “Detect It Easy v3.07 (DIE)” to assess if the binary was packed and what was used to pack it. According to the results, the binary is packed by Microsoft Visual C/C++(2017 v.15.8)(see Figures 21 & 22).

Figure 21: Detect It Easy ApacheMOD1.exe



The user also executed PE Detective as a forensic tool to identify malware in file headers, section analysis, import/export table analysis, string extraction, resource exploration, and disassembly. PE Detective yielded the same results and DIE.

Figure 22: PE Detective ApacheMOD1.exe



The user opened Wireshark, started capturing traffic, and opened ApacheMOD1.exe and ran simultaneously. The user observed the machine attempting to communicate to an unknown destination IP address (192.168.100.100). The user exploded the transmission control protocol to observe additional pieces of information such as the destination port, source port, et cetera (see Figure 23).

Figure 23: Wireshark

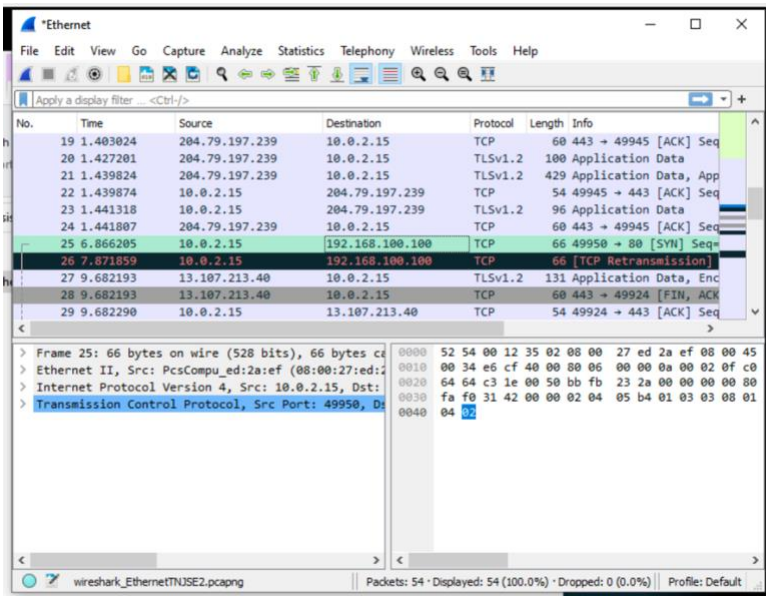
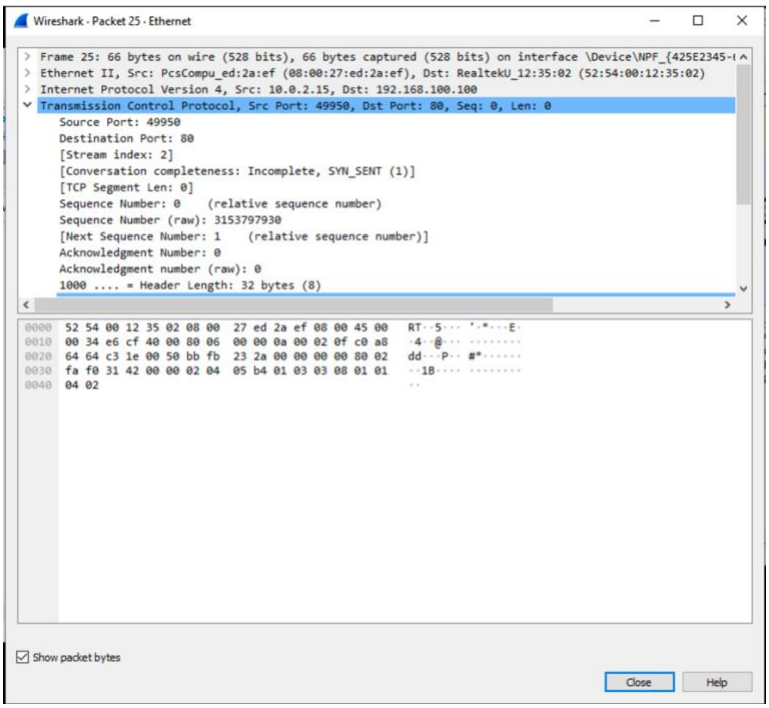
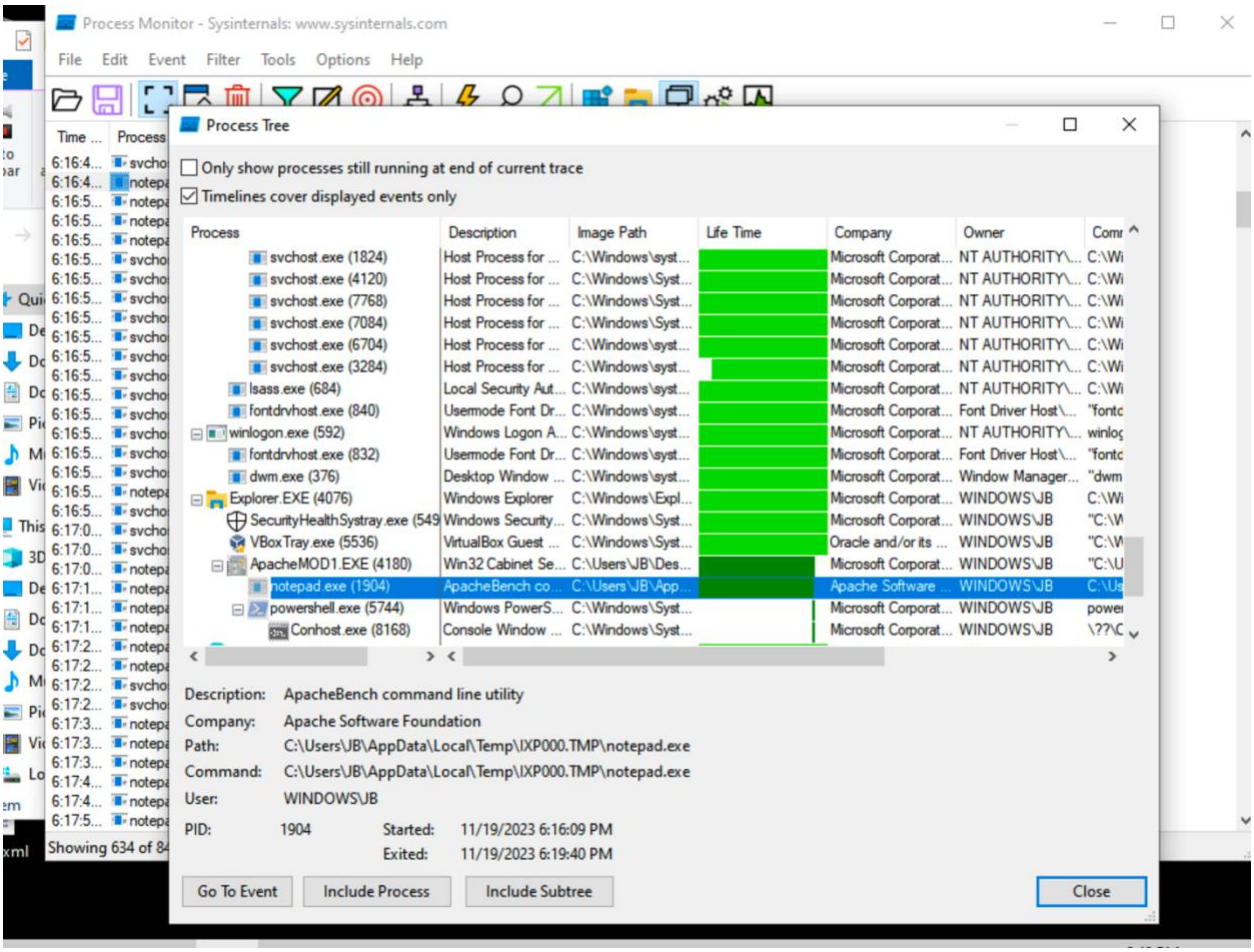


Figure 24: Transmission Control Protocol



The user opened Procmon (i.e., Process Monitor) and ceased all other tasks running in the background in order to obtain a clean sample. Then, the user opened ApacheMOD1.exe and ran the sample in order to obtain information from the sample. The user unchecked all of the other applications other than 'network activity.' When opening the process tree, the user observed that ApacheMOD1.exe branched into 'notepad.exe' (see Figure 25).

Figure 25: Notepad.exe



The user utilizes Procmon again in order to monitor file system activities. Therefore, the user then chooses the ‘file system creator’ in order to identify if ‘ApacheMOD1.exe’ was indeed creating any files were created. The user was able to identify that ‘ApacheMOD1.exe’ was creating files, which corresponded with the Floss and Strings data in the static analysis (see Figures 26 & 27).

Figure 26: Procmon File Data Creation #1

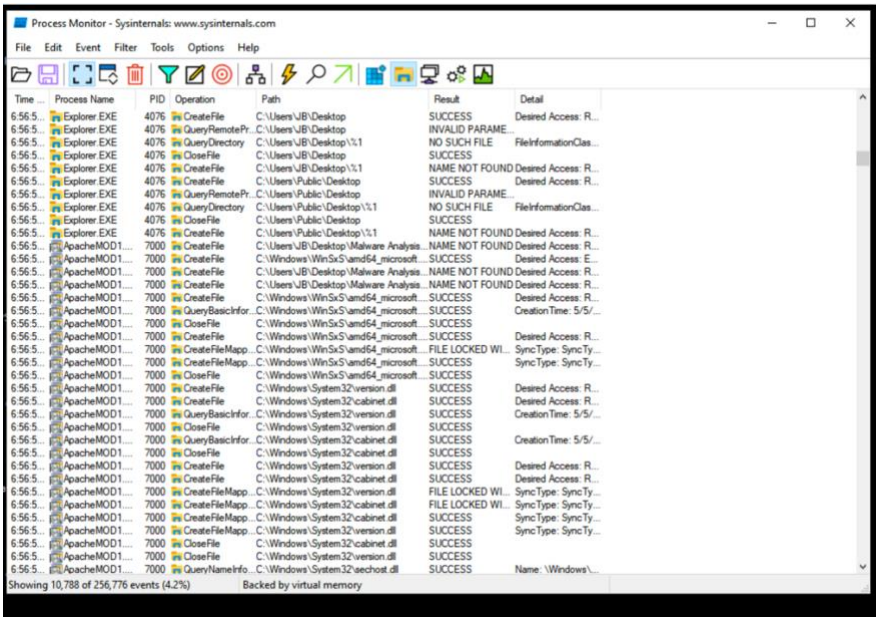
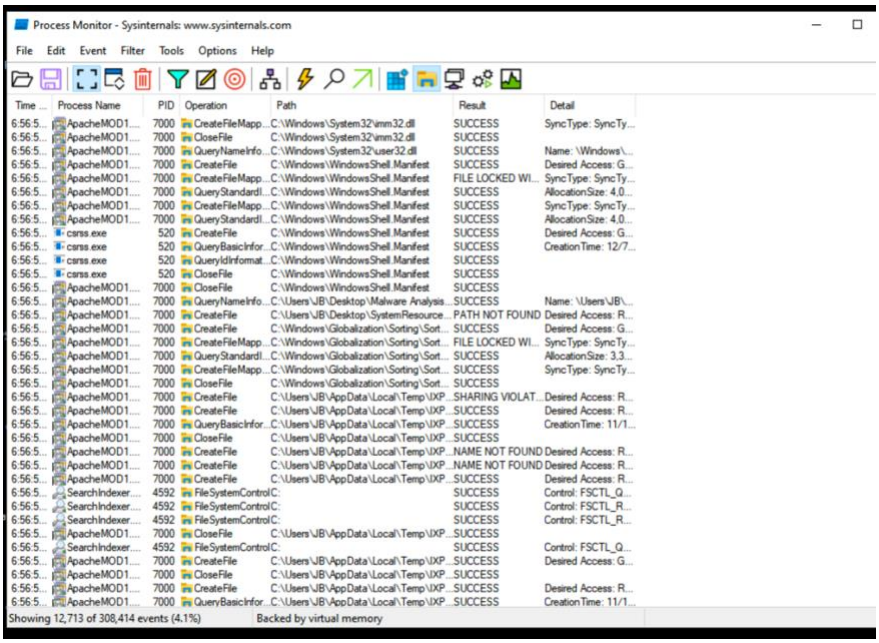
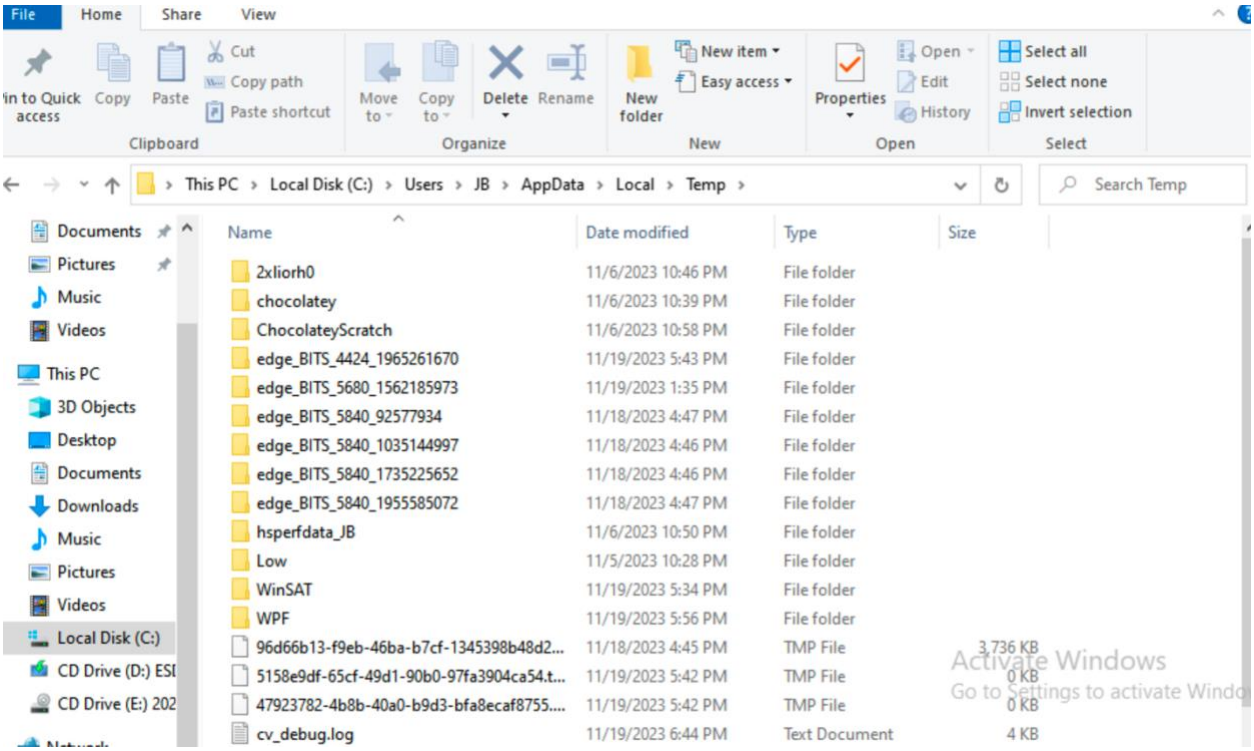


Figure 27: Procmon File Data Creation #2



Please note the user was unable to find notepad.exe. The user executed local disk (C:)\Users\JB\AppData\Local\Temp; however, IXP001.TMP was unavailable (see Figure 28).

Figure 28: IXP001 Unfindable File



The user continued to use Procmon; however, this time, the user utilized Registry Key Modifications. The Modifications. The user would look to identify any files that would have “run once” or “run services.” Other “RegCreateKey” operations that would be of interest would include, but not limited to, shell extensions and handlers, software registration keys, policies, system, et cetera (see Figures 29 - 32).

Figure 29: RegCreateKey Sample

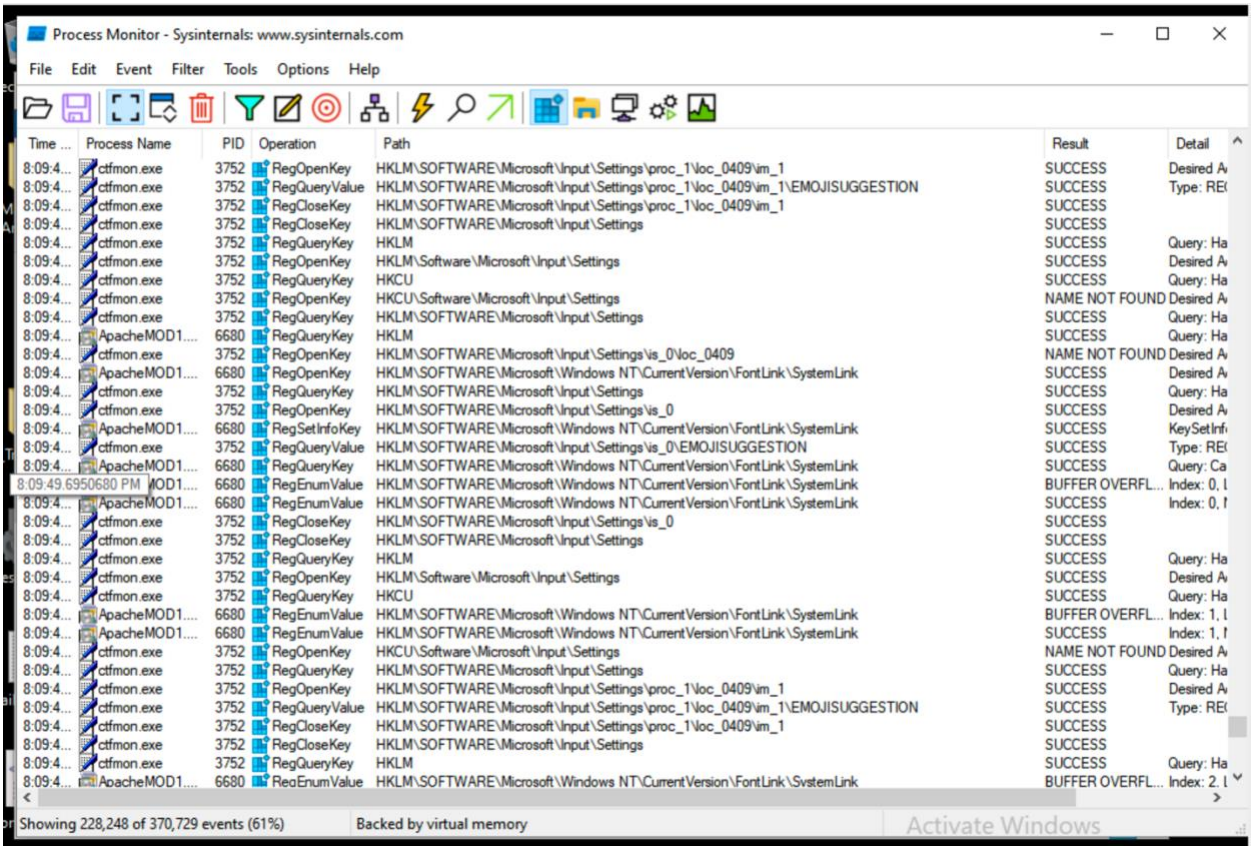


Figure 30: RegCreateKey Sample/DataStore

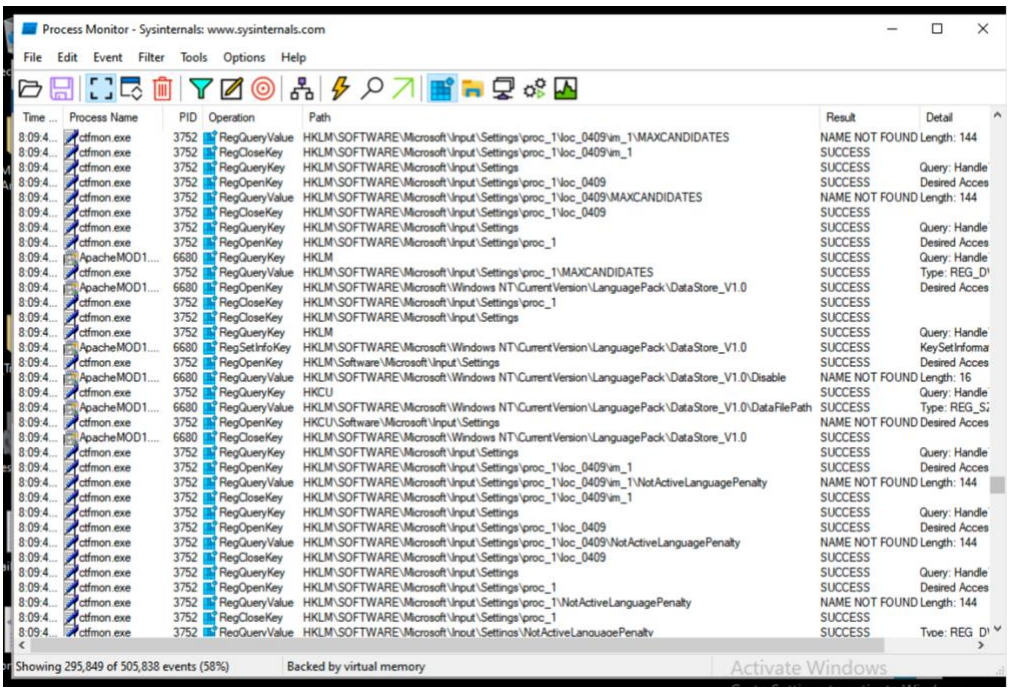


Figure 31:RegCreateKey Sample/OOBE Launch User

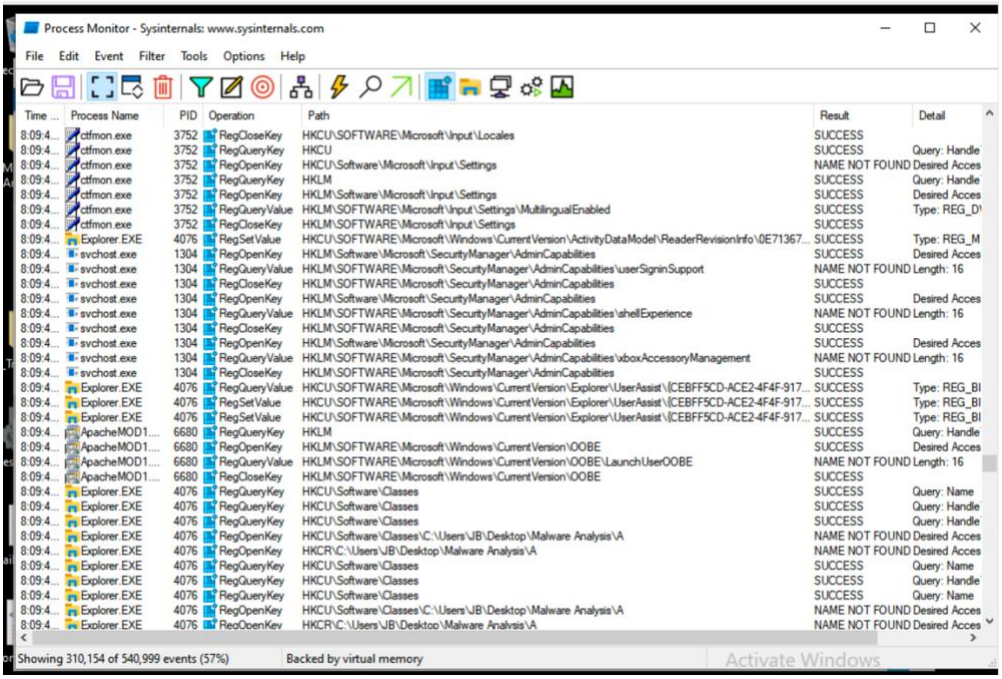
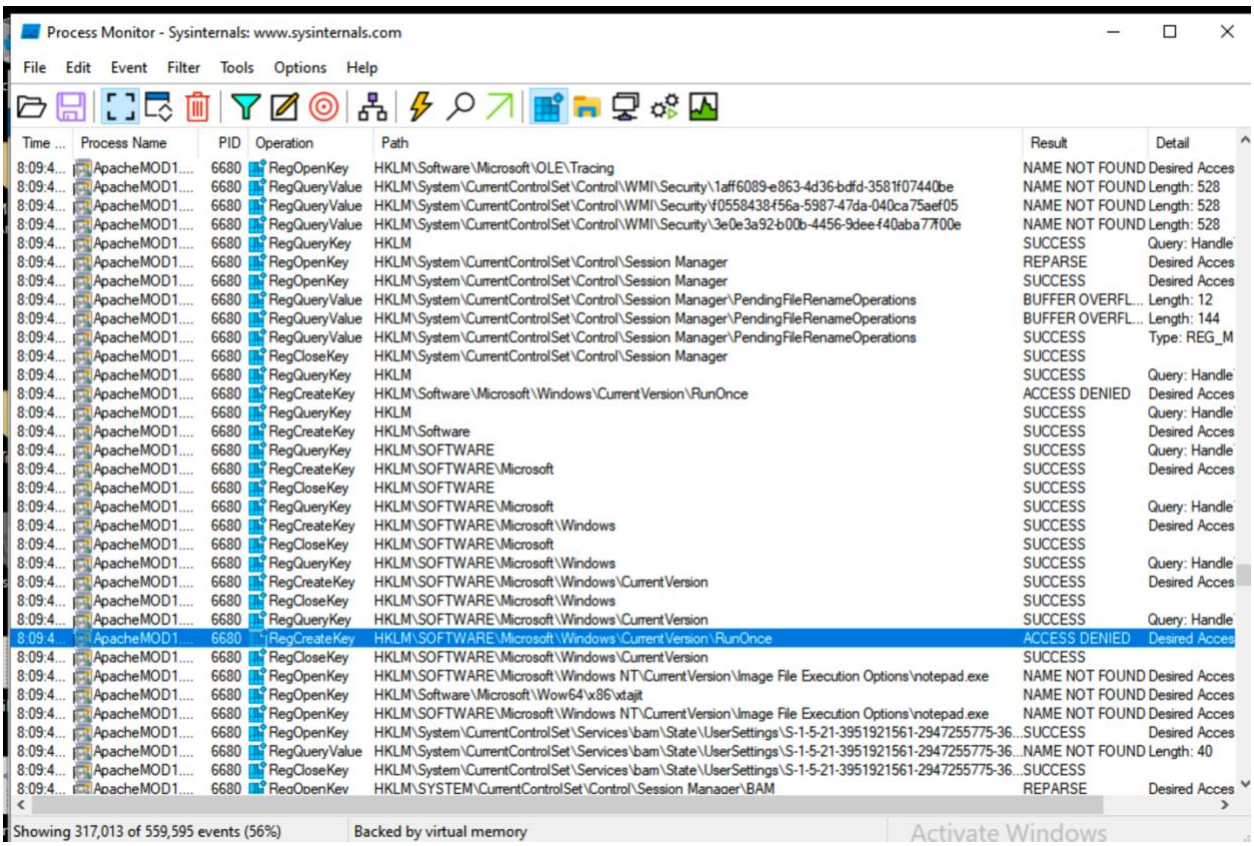


Figure 32: RegCreateKey Sample\RunOnce



Time ...	Process Name	PID	Operation	Path	Result	Detail
8:09:4...	ApacheMOD1....	6680	RegOpenKey	HKLM\Software\Microsoft\OLE\Tracing	NAME NOT FOUND	Desired Acces
8:09:4...	ApacheMOD1....	6680	RegQueryValue	HKLM\System\CurrentControlSet\Control\WMI\Security\1aff6089-e863-4d36-bdfd-3581f07440be	NAME NOT FOUND	Length: 528
8:09:4...	ApacheMOD1....	6680	RegQueryValue	HKLM\System\CurrentControlSet\Control\WMI\Security\f0558438-f56a-5987-47da-040ca75aef05	NAME NOT FOUND	Length: 528
8:09:4...	ApacheMOD1....	6680	RegQueryValue	HKLM\System\CurrentControlSet\Control\WMI\Security\3e0e3a92-b00b-4456-9dee-f40aba77f00e	NAME NOT FOUND	Length: 528
8:09:4...	ApacheMOD1....	6680	RegQueryKey	HKLM	SUCCESS	Query: Handle
8:09:4...	ApacheMOD1....	6680	RegOpenKey	HKLM\System\CurrentControlSet\Control\Session Manager	REPARSE	Desired Acces
8:09:4...	ApacheMOD1....	6680	RegOpenKey	HKLM\System\CurrentControlSet\Control\Session Manager	SUCCESS	Desired Acces
8:09:4...	ApacheMOD1....	6680	RegQueryValue	HKLM\System\CurrentControlSet\Control\Session Manager\PendingFileRenameOperations	BUFFER OVERFL...	Length: 12
8:09:4...	ApacheMOD1....	6680	RegQueryValue	HKLM\System\CurrentControlSet\Control\Session Manager\PendingFileRenameOperations	BUFFER OVERFL...	Length: 144
8:09:4...	ApacheMOD1....	6680	RegQueryValue	HKLM\System\CurrentControlSet\Control\Session Manager\PendingFileRenameOperations	SUCCESS	Type: REG_M
8:09:4...	ApacheMOD1....	6680	RegCloseKey	HKLM\System\CurrentControlSet\Control\Session Manager	SUCCESS	
8:09:4...	ApacheMOD1....	6680	RegQueryKey	HKLM	SUCCESS	Query: Handle
8:09:4...	ApacheMOD1....	6680	RegCreateKey	HKLM\Software\Microsoft\Windows\CurrentVersion\RunOnce	ACCESS DENIED	Desired Acces
8:09:4...	ApacheMOD1....	6680	RegQueryKey	HKLM	SUCCESS	Query: Handle
8:09:4...	ApacheMOD1....	6680	RegCreateKey	HKLM\Software	SUCCESS	Desired Acces
8:09:4...	ApacheMOD1....	6680	RegQueryKey	HKLM\SOFTWARE	SUCCESS	Query: Handle
8:09:4...	ApacheMOD1....	6680	RegCreateKey	HKLM\SOFTWARE\Microsoft	SUCCESS	Desired Acces
8:09:4...	ApacheMOD1....	6680	RegCloseKey	HKLM\SOFTWARE	SUCCESS	
8:09:4...	ApacheMOD1....	6680	RegQueryKey	HKLM\SOFTWARE\Microsoft	SUCCESS	Query: Handle
8:09:4...	ApacheMOD1....	6680	RegCreateKey	HKLM\SOFTWARE\Microsoft\Windows	SUCCESS	Desired Acces
8:09:4...	ApacheMOD1....	6680	RegCloseKey	HKLM\SOFTWARE\Microsoft	SUCCESS	
8:09:4...	ApacheMOD1....	6680	RegQueryKey	HKLM\SOFTWARE\Microsoft\Windows	SUCCESS	Query: Handle
8:09:4...	ApacheMOD1....	6680	RegCreateKey	HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion	SUCCESS	Desired Acces
8:09:4...	ApacheMOD1....	6680	RegCloseKey	HKLM\SOFTWARE\Microsoft\Windows	SUCCESS	
8:09:4...	ApacheMOD1....	6680	RegQueryKey	HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion	SUCCESS	Query: Handle
8:09:4...	ApacheMOD1....	6680	RegCreateKey	HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce	ACCESS DENIED	Desired Acces
8:09:4...	ApacheMOD1....	6680	RegCloseKey	HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion	SUCCESS	
8:09:4...	ApacheMOD1....	6680	RegOpenKey	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\notepad.exe	NAME NOT FOUND	Desired Acces
8:09:4...	ApacheMOD1....	6680	RegOpenKey	HKLM\Software\Microsoft\Wow64\x86\xtajit	NAME NOT FOUND	Desired Acces
8:09:4...	ApacheMOD1....	6680	RegOpenKey	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\notepad.exe	NAME NOT FOUND	Desired Acces
8:09:4...	ApacheMOD1....	6680	RegOpenKey	HKLM\System\CurrentControlSet\Services\bam\State\UserSettings\S-1-5-21-3951921561-2947255775-36...	SUCCESS	Desired Acces
8:09:4...	ApacheMOD1....	6680	RegQueryValue	HKLM\System\CurrentControlSet\Services\bam\State\UserSettings\S-1-5-21-3951921561-2947255775-36...	NAME NOT FOUND	Length: 40
8:09:4...	ApacheMOD1....	6680	RegCloseKey	HKLM\System\CurrentControlSet\Services\bam\State\UserSettings\S-1-5-21-3951921561-2947255775-36...	SUCCESS	
8:09:4...	ApacheMOD1....	6680	RegOpenKey	HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\BAM	REPARSE	Desired Acces

The user then executed the *Regshot program*, choose "1st shot" and ran the program. Once the file was done running, the user then saved the data as a file on the desktop as "1st" and then re-ran the program and saved it as "2nd." Then the user was able to compare the two snapshots in order to assess the differences between the two files (see Figures 33 & 34).

Figure 33: Regshot Compare #1

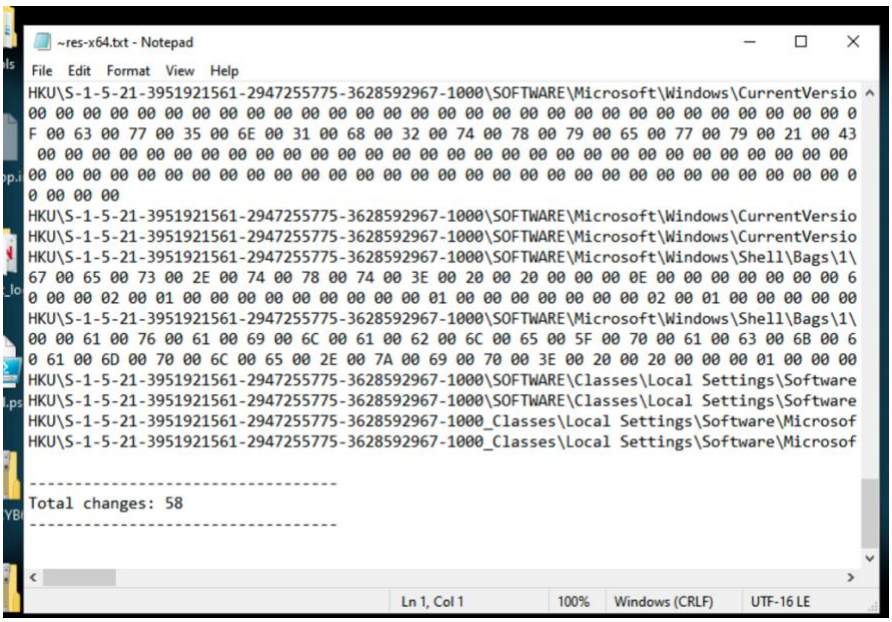
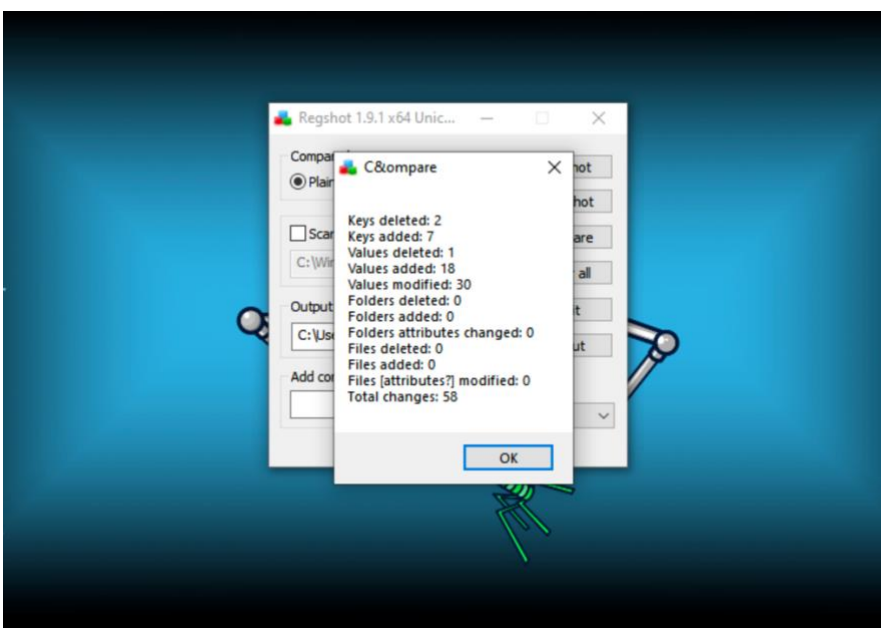
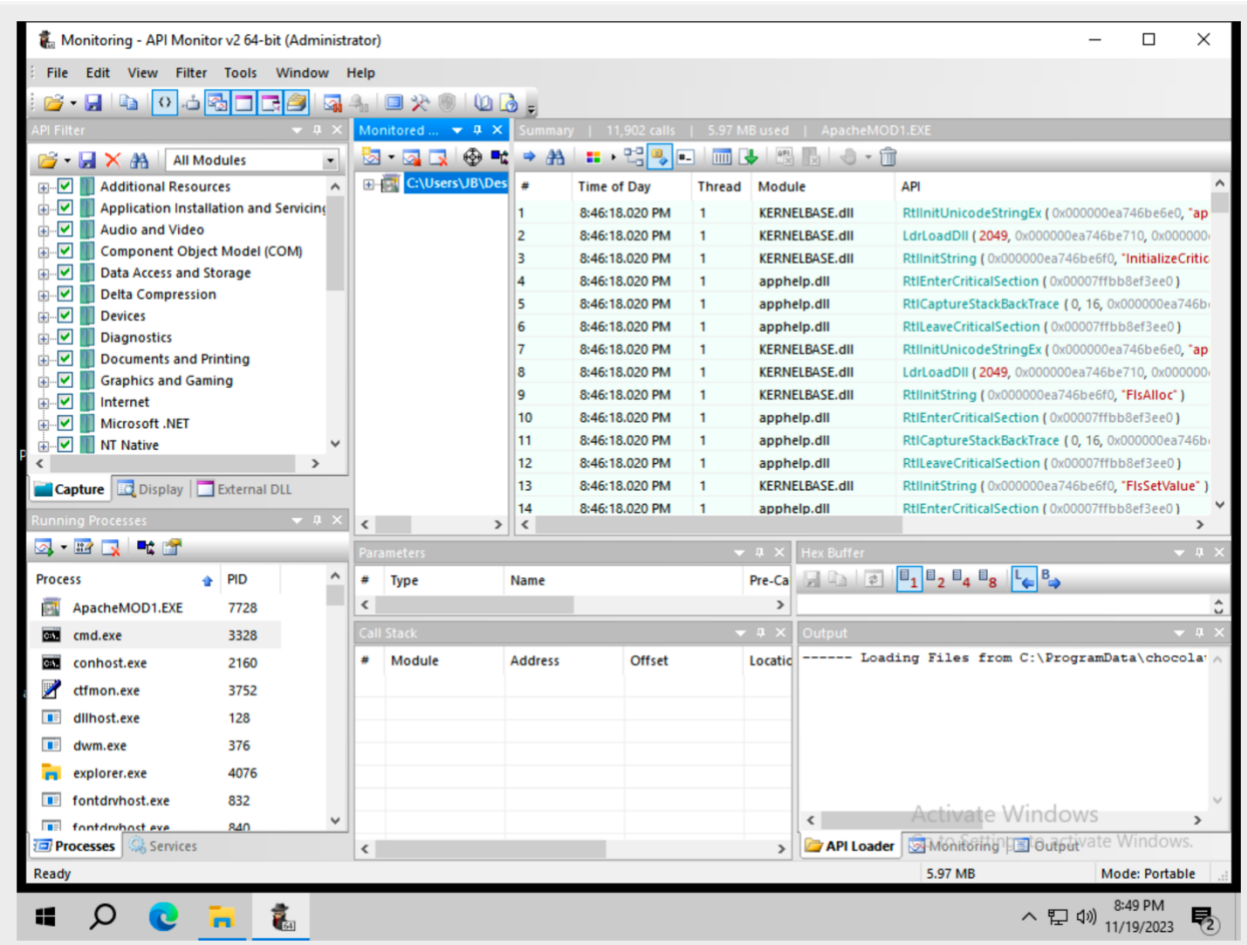


Figure 34: Regshot Compare #2



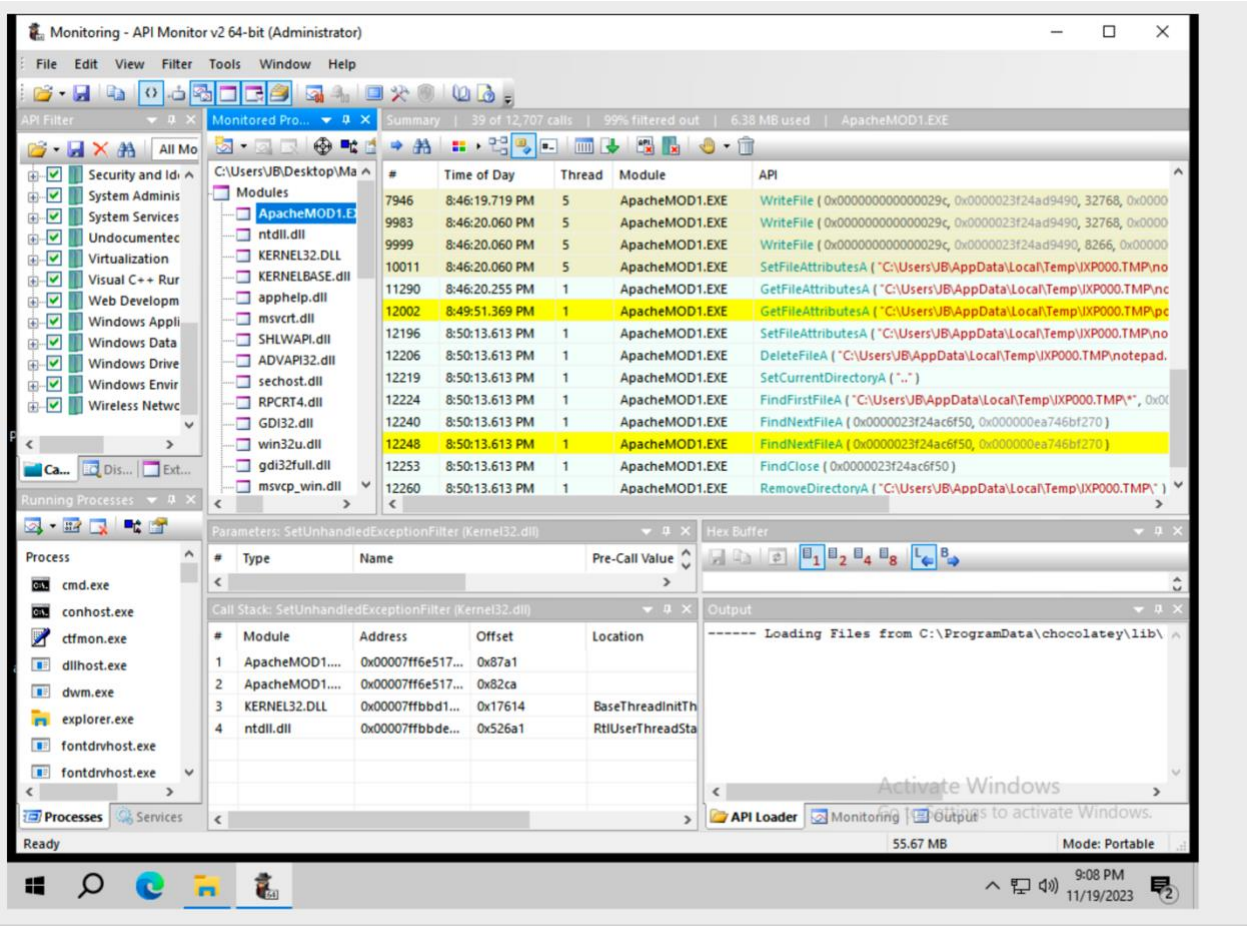
The user wanted to assess Loaded Libraries at Runtime and API calls; therefore, the user executed the API Monitor App. The user opened the search toolbar\'API'\Apimonitor\Run as Admin\MonitorNewProcess\Desktop\ApacheMOD1.exe\Open\OK (see Figure 34A).

Figure 34A: API Monitor



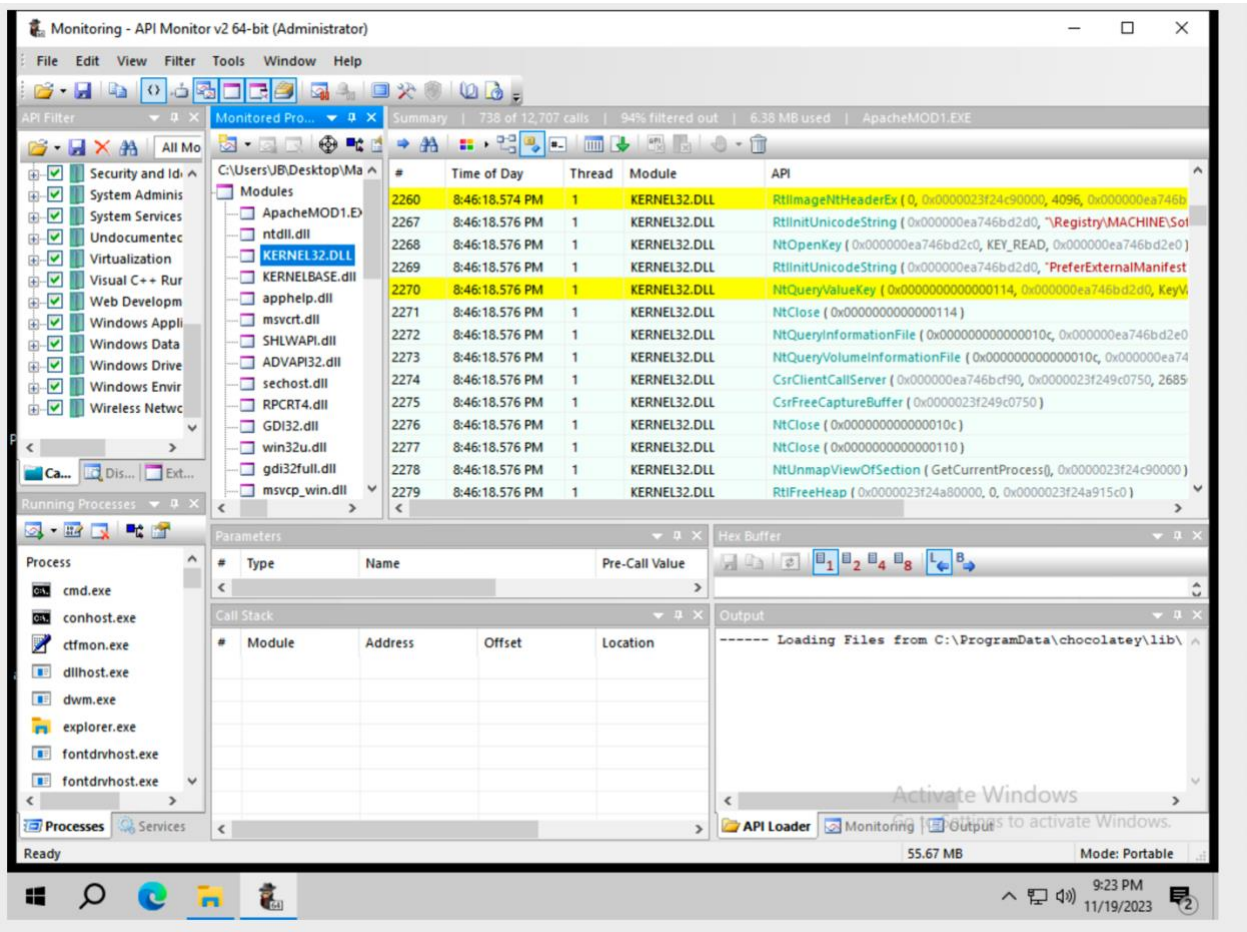
The user made some notable mentions when viewing the API calls. For example, the user came across (again) ‘RemoveDirectoryA’ and ‘GetFileAttributes’ when screening the ApacheMOD1.exe module (see Figure 35).

Figure 35: ApacheMOD1.exe



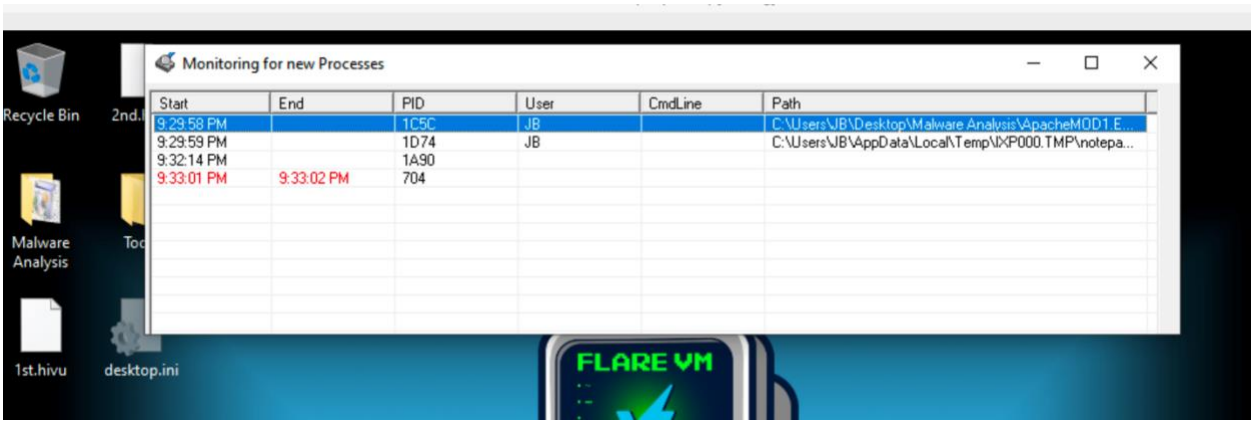
The user made note of Kernel32.DLL; this can be very useful as it is a critical system file and the Windows operating system that provides applications with access to essential functions such as memory management, input/ output operations, process and thread creation, and synchronization of services (Technopedia, n.d.). There is a multitude of reasons for a threat actor to find interest in Kernel32. For instance, process manipulation, memory management, file operations, DLL injection, system information, and networking Can all be misused by malicious software for in a nefarious purposes (Reddy, 2019) (see Figure 36).

Figure 36: Kernel32.DLL



The user then executed *ProcWatch* as it allowed the user to monitor the processes on the system. Therefore, the user executed Procwatch\open\desktop\MalwareAnalysis\ApacheMOD1.exe\run and then captured the process The user noted this succession of processes within the first two columns of data (see Figure 37).

Figure 37: ProcWatch



The user ran Fakenet while executing ApacheMOD1.exe in order to capture data. Fakenet is a tool that is often used in an analysis of malware. The user observed 'notepad.exe' being utilized and communicating with IP address 192.168.100.100 (see Figures 38 & 39).

Figure 38: Fakenet #1

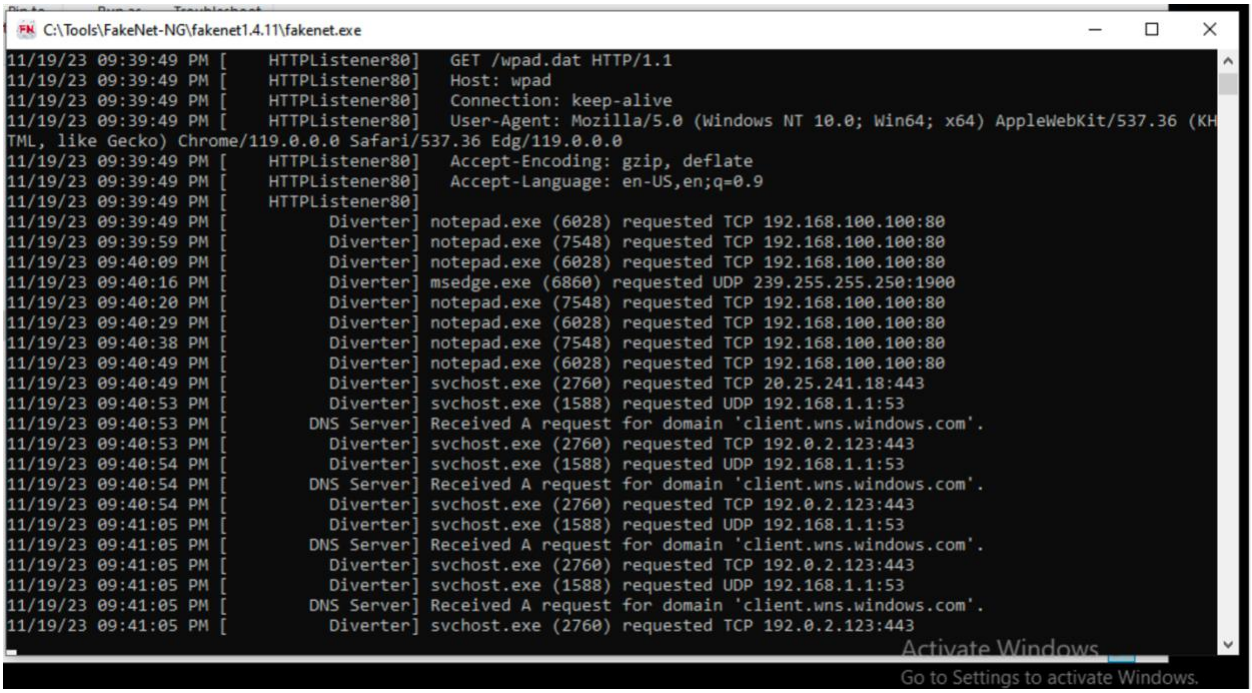
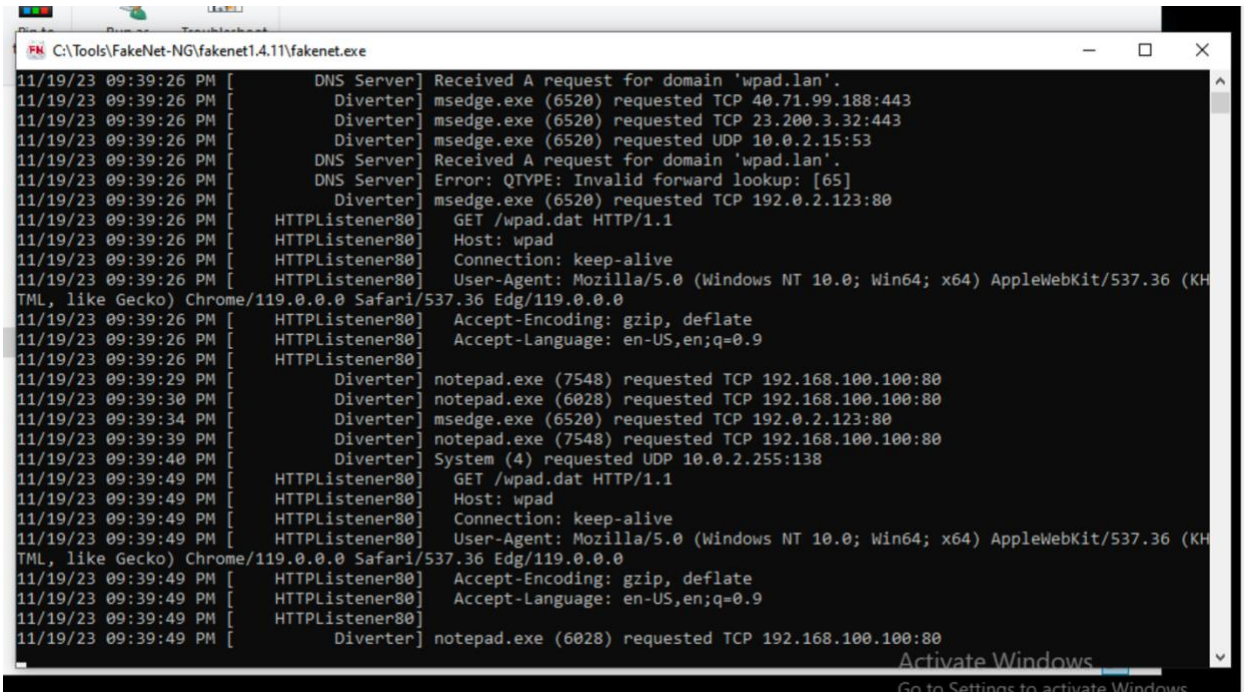


Figure 39: Fakenet #2

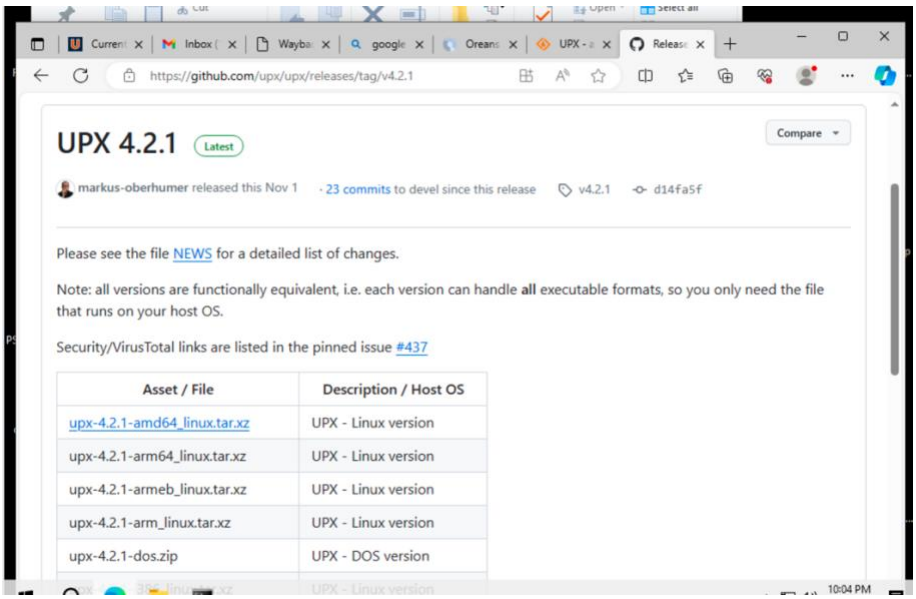


```
C:\Tools\FakeNet-NG\fakenet1.4.11\fakenet.exe
11/19/23 09:39:26 PM [        DNS Server] Received A request for domain 'wpad.lan'.
11/19/23 09:39:26 PM [          Diverter] msedge.exe (6520) requested TCP 40.71.99.188:443
11/19/23 09:39:26 PM [          Diverter] msedge.exe (6520) requested TCP 23.200.3.32:443
11/19/23 09:39:26 PM [          Diverter] msedge.exe (6520) requested UDP 10.0.2.15:53
11/19/23 09:39:26 PM [        DNS Server] Received A request for domain 'wpad.lan'.
11/19/23 09:39:26 PM [        DNS Server] Error: QTYPE: Invalid forward lookup: [65]
11/19/23 09:39:26 PM [          Diverter] msedge.exe (6520) requested TCP 192.0.2.123:80
11/19/23 09:39:26 PM [    HTTPListener80]   GET /wpad.dat HTTP/1.1
11/19/23 09:39:26 PM [    HTTPListener80]   Host: wpad
11/19/23 09:39:26 PM [    HTTPListener80]   Connection: keep-alive
11/19/23 09:39:26 PM [    HTTPListener80]   User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KH
TML, like Gecko) Chrome/119.0.0.0 Safari/537.36 Edg/119.0.0.0
11/19/23 09:39:26 PM [    HTTPListener80]   Accept-Encoding: gzip, deflate
11/19/23 09:39:26 PM [    HTTPListener80]   Accept-Language: en-US,en;q=0.9
11/19/23 09:39:26 PM [    HTTPListener80]
11/19/23 09:39:29 PM [          Diverter] notepad.exe (7548) requested TCP 192.168.100.100:80
11/19/23 09:39:30 PM [          Diverter] notepad.exe (6028) requested TCP 192.168.100.100:80
11/19/23 09:39:34 PM [          Diverter] msedge.exe (6520) requested TCP 192.0.2.123:80
11/19/23 09:39:39 PM [          Diverter] notepad.exe (7548) requested TCP 192.168.100.100:80
11/19/23 09:39:40 PM [          Diverter] System (4) requested UDP 10.0.2.255:138
11/19/23 09:39:49 PM [    HTTPListener80]   GET /wpad.dat HTTP/1.1
11/19/23 09:39:49 PM [    HTTPListener80]   Host: wpad
11/19/23 09:39:49 PM [    HTTPListener80]   Connection: keep-alive
11/19/23 09:39:49 PM [    HTTPListener80]   User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KH
TML, like Gecko) Chrome/119.0.0.0 Safari/537.36 Edg/119.0.0.0
11/19/23 09:39:49 PM [    HTTPListener80]   Accept-Encoding: gzip, deflate
11/19/23 09:39:49 PM [    HTTPListener80]   Accept-Language: en-US,en;q=0.9
11/19/23 09:39:49 PM [    HTTPListener80]
11/19/23 09:39:49 PM [          Diverter] notepad.exe (6028) requested TCP 192.168.100.100:80
```

Packing & Static Analysis

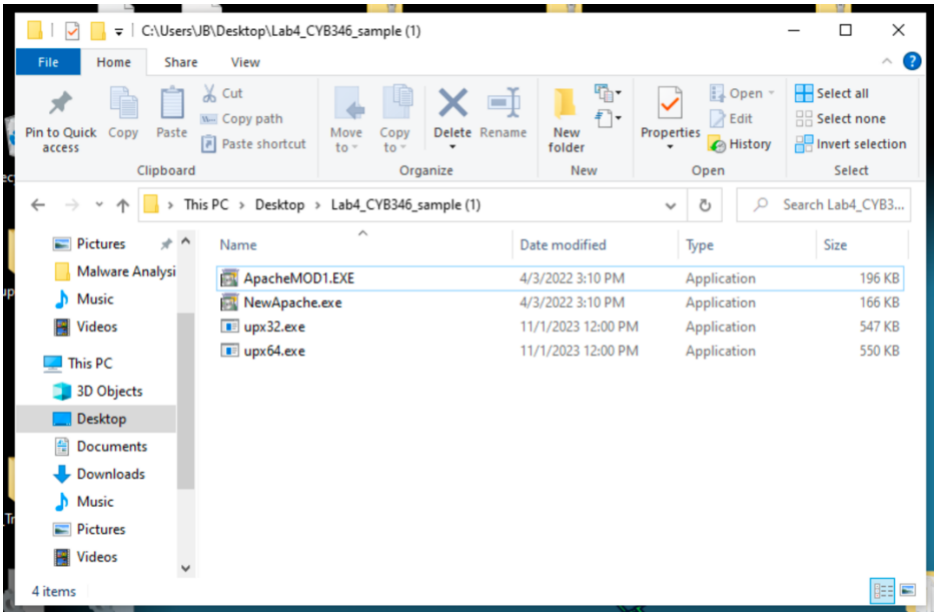
The user downloaded the UPX packer using github.com and made sure that the correct version was downloaded (see Figure 40).

Figure 40: UPX Packing



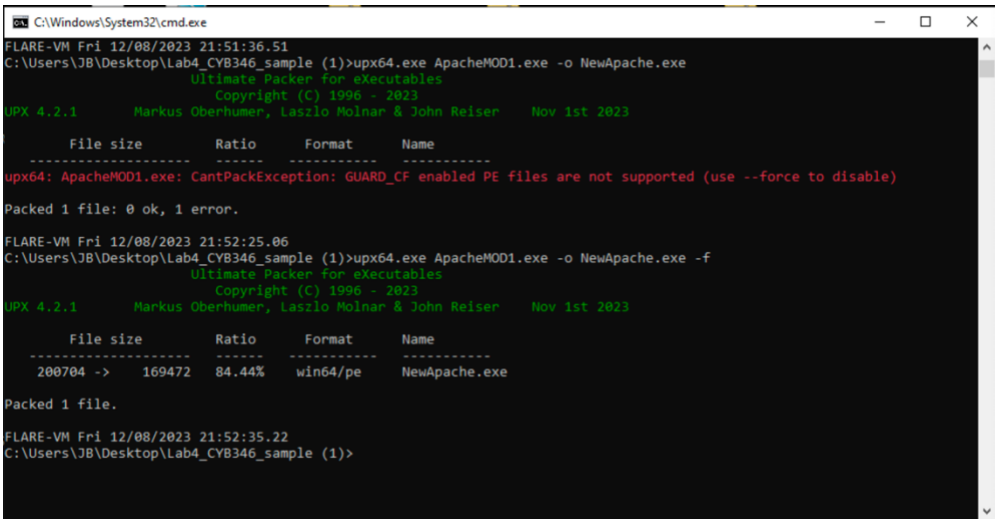
The user then unzipped the file using 7-Zip and added the executable file to the Lab4_CYB346_sample(1) folder (see Figure 41).

Figure 41: Unzipped UPX



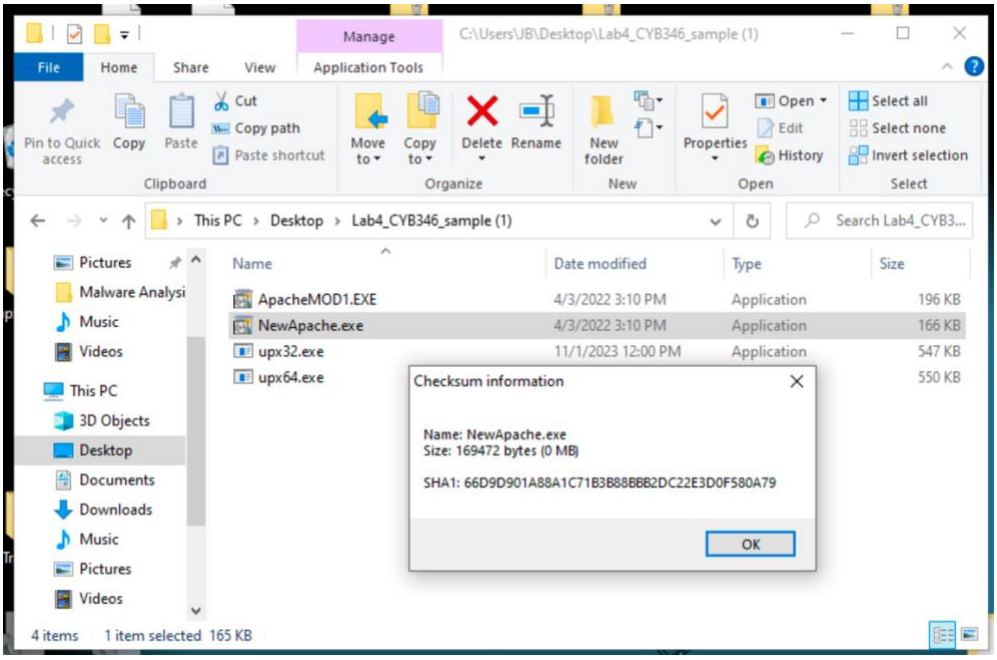
The user opened a command prompt within the Lab4_CYB346_sample(1) file and executed ‘upx64.exe ApacheMOD1.exe -o **NewApache.exe** -f’ to pack the file (see Figure 42).

Figure 42: NewApache.exe



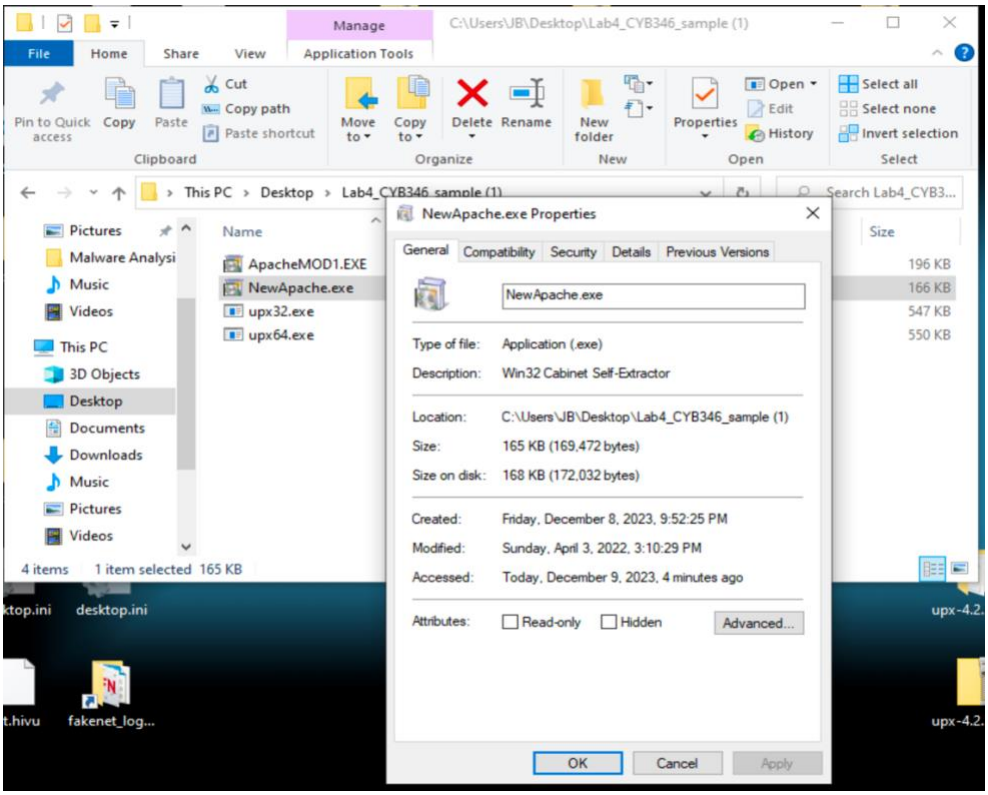
In order to allocate the Hash Values, the user right-clicked on the file →CRC SHA→SHA-1, which displayed the Hash File (see Figure 43). Note that the file size is smaller in comparison to ApacheMOD1.EXE

Figure 43: Packed Hash File NewApache



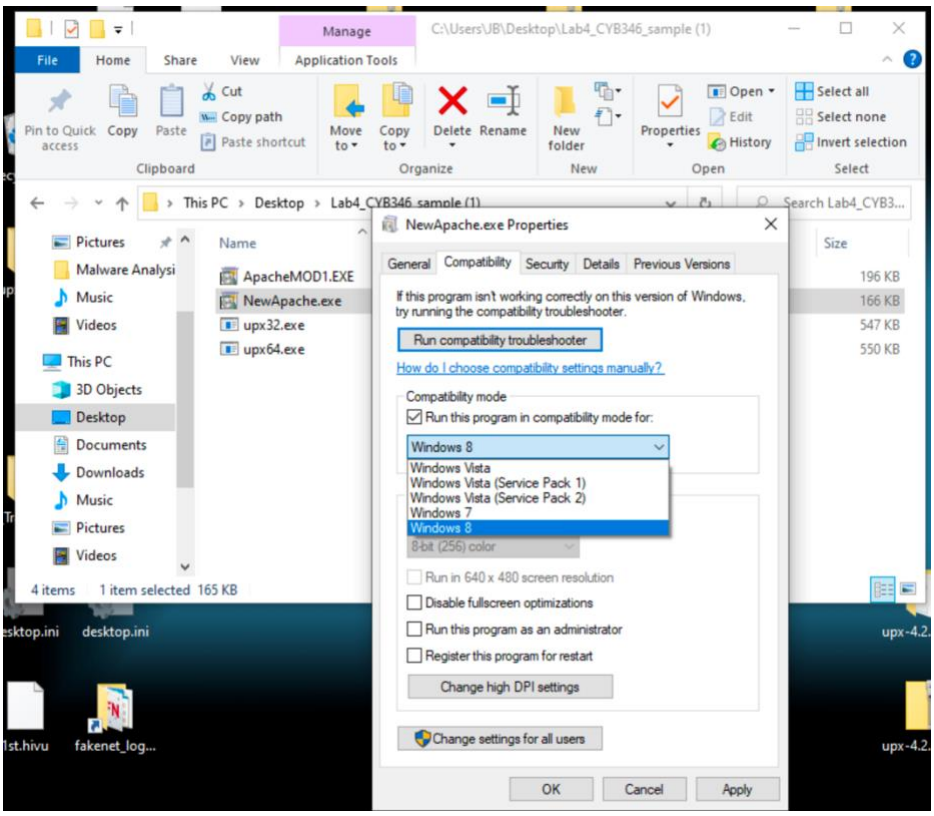
The user then right-clicked on the file again to inspect for Metadata. The user validated that there was no digital signature, suggesting a security risk. A digital signature validates who created the file and if the file has been altered or corrupted (Assaad, 2021) (see Figure 44).

Figure 44: Packed File Signature



The user right-clicked on the file, clicked on ‘compatibility,’ checked ‘run this program in compatibility mode for,’ and then clicked the down arrow to identify the various Windows OS (see Figure 45); the choices evidenced a 64-bit architect type. Please note that this is the same as the unpacked file.

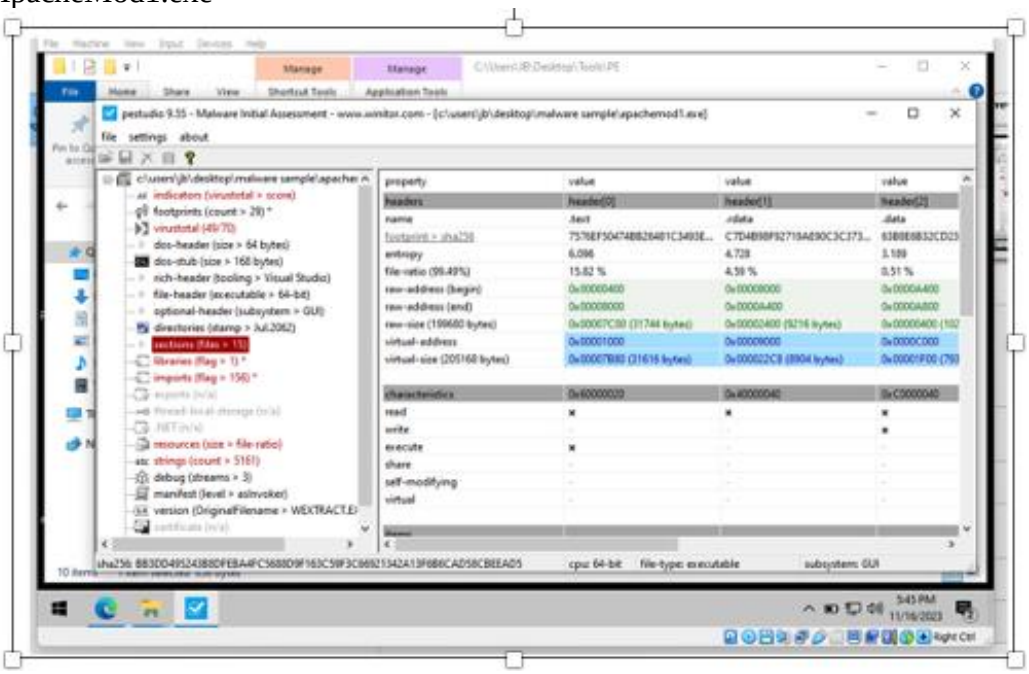
Figure 45: Packed NewApache.exe properties



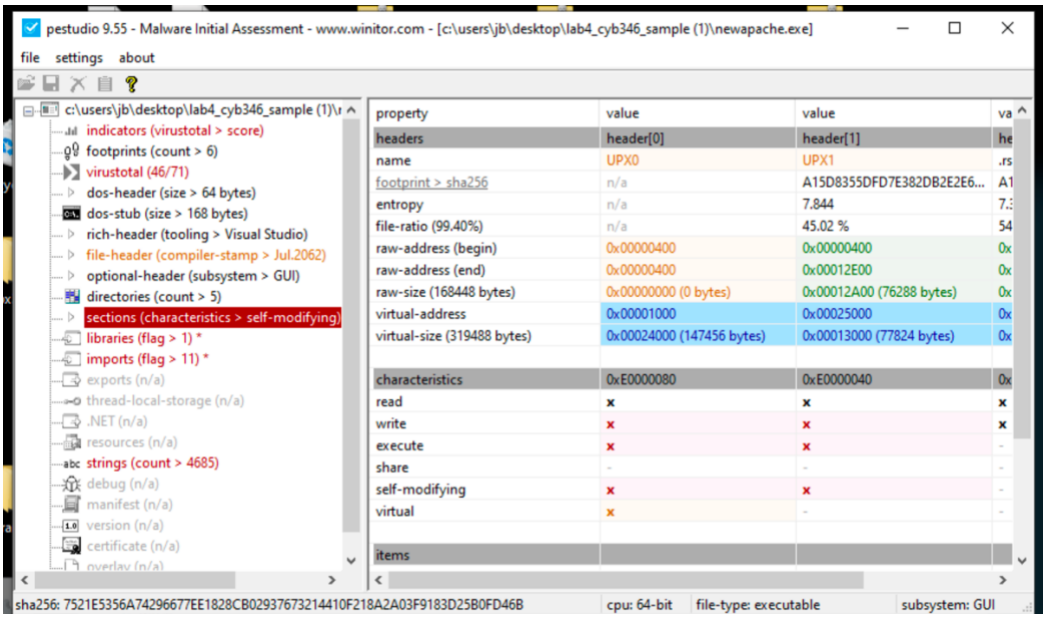
Then, the user executed ‘pestudio 9.55’ from the tools included within the Flare VM suite. The user opened the ApacheMOD1.exe file for the program to evaluate its contents. Please note that “UPX0 and UPX1” take the place of “.text and .rdata” in the original file (see Figure 46).

Figure 46: pestudio comparison

ApacheMod1.exe



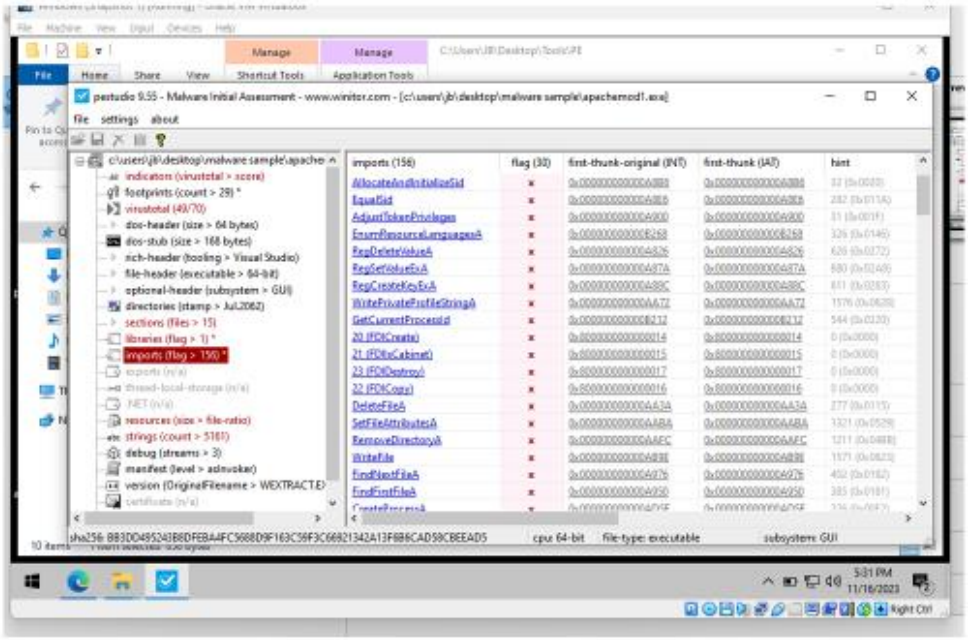
NewApache.exe



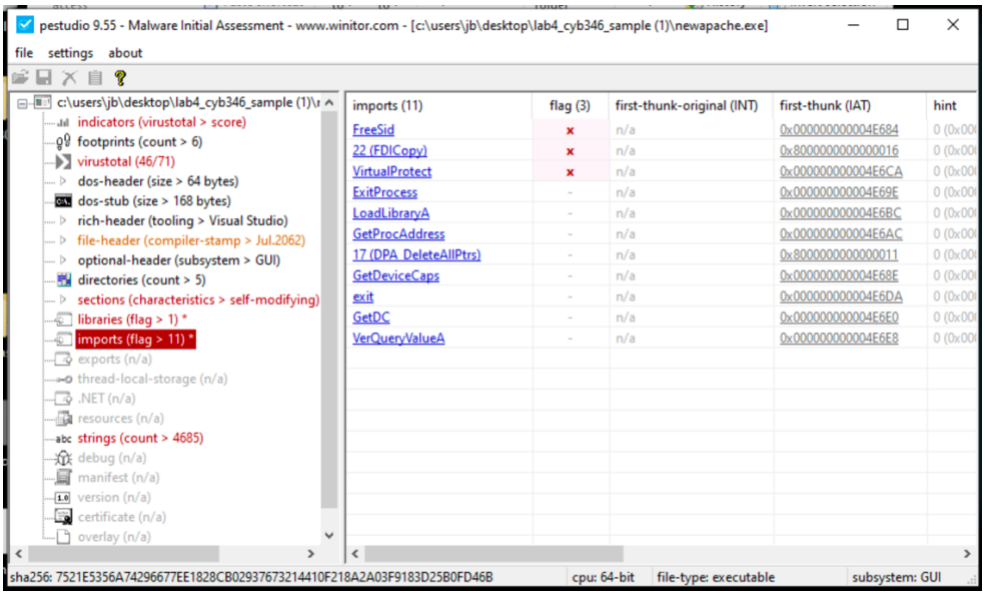
Then, the user utilized the program and allocated only 3 files flagged by the program, which is essential to note when comparing to the original file. Please note that several pieces of information were missing, such as first-thunk-original (INT), first-thunk (IAT), group, technique, and hint (see Figure 47).

Figure 47: Import Comparison

ApacheMOD1.exe



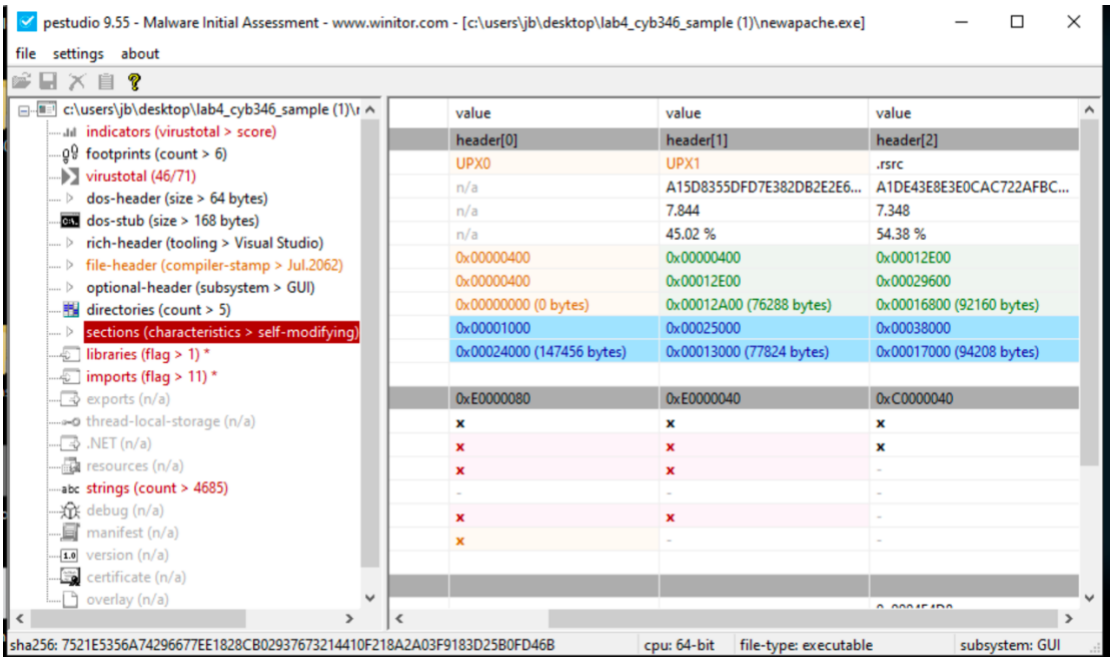
NewApache.exe



The user made note of the three imports flagged (i.e., FreeSid, 22 (FDICopy), and VirtualProtect). According to Alvinashcraft (n.d.), this file can be utilized for privilege escalation, access manipulation, audit evasion, account mimicking, and persistence. Import 22 (FDICopy) allows attackers to obtain information using data exfiltration, file manipulation, persistence, delivery of payload, and bypassing security measures. VirtualProtect can allow threat actors to inject code, bypass security measures, unhooking API calls, packing and unpacking code, dynamically modifying code, and utilize reflective DLL injection.

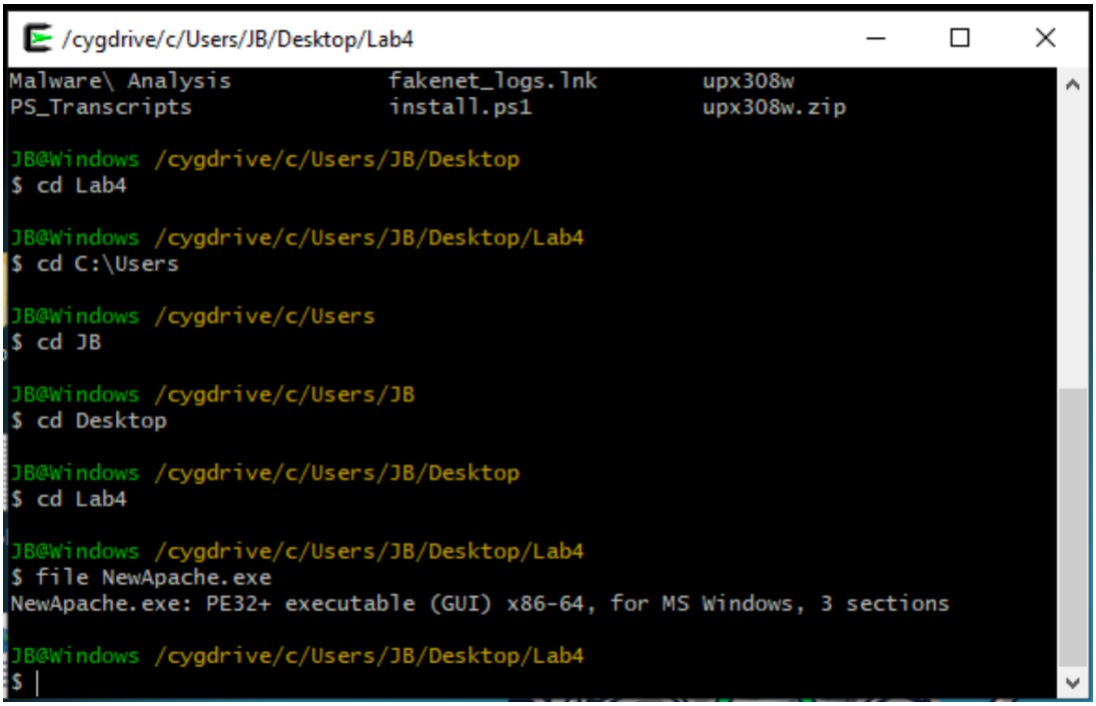
As noted during the previous lab, *Pestudio 9.55* was heavily utilized throughout the static analysis to obtain information. The user used it to identify various sections and entropy; the higher the entropy, the more likely randomness and disorder data sets. However, the packed sample only provided entropy data for UPX1 and .rsrc, both close to the value of 8 secondary to being packed (see Figure 48).

Figure 48: Pestudio UPX



The user then opened the start menu/Cygwin/Cygwin64 Terminal/cd C:\Users/cd JB\/cd Desktop/\cd Lab4/file NewApache.exe (see Figure 49). This identifies the file type (PE32+ executable (GUI)) x86-64, for MS Windows, 3 sections, which makes up the structure of the executable file (Karl-Bridge-Microsoft, n.d.). Please note that this is distinctly different from the unpacked file (i.e., (PE32+ executable (GUI)), executable file format in both 32 and 64-bit versions)).

Figure 49: Packed File Type



The user then executed ‘file -i NewApache.exe to obtain the MIME Type, ‘apache.exe: application/vnd.microsoft.portable-executable; charset=binary’ (see Figure 50).

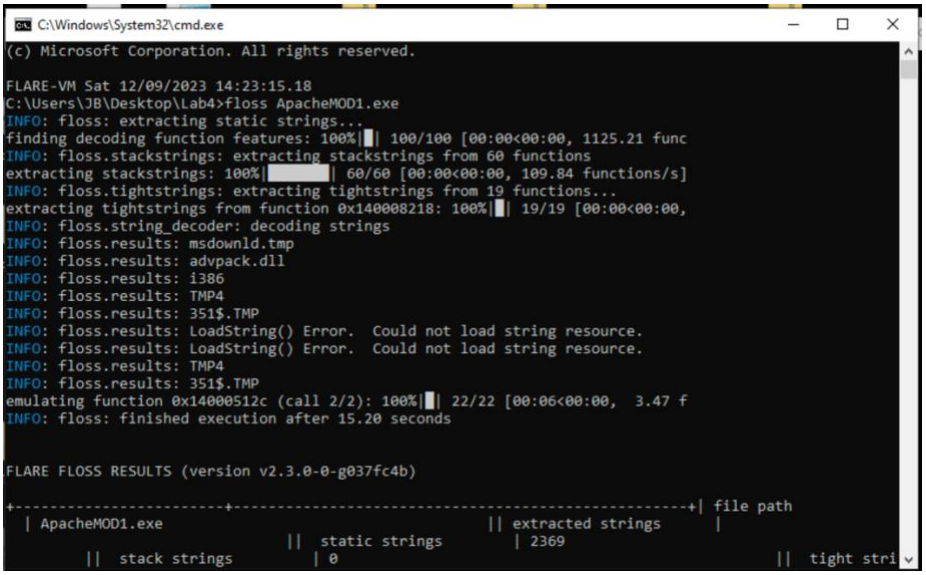
Figure 50: Packed MIME

```
/cygdrive/c/Users/JB/Desktop/Lab4
$ cd Lab4

JB@Windows /cygdrive/c/Users/JB/Desktop/Lab4
$ cd C:\Users

JB@Windows /cygdrive/c/Users
$ cd JB

JB@Windows /cygdrive/c/Users/JB
$ cd Desktop

JB@Windows /cygdrive/c/Users/JB/Desktop
$ cd Lab4

JB@Windows /cygdrive/c/Users/JB/Desktop/Lab4
$ file NewApache.exe
NewApache.exe: PE32+ executable (GUI) x86-64, for MS Windows, 3 sections

JB@Windows /cygdrive/c/Users/JB/Desktop/Lab4
$ file -i NewApache.exe
NewApache.exe: application/vnd.microsoft.portable-executable; charset=binary

JB@Windows /cygdrive/c/Users/JB/Desktop/Lab4
$ |
```

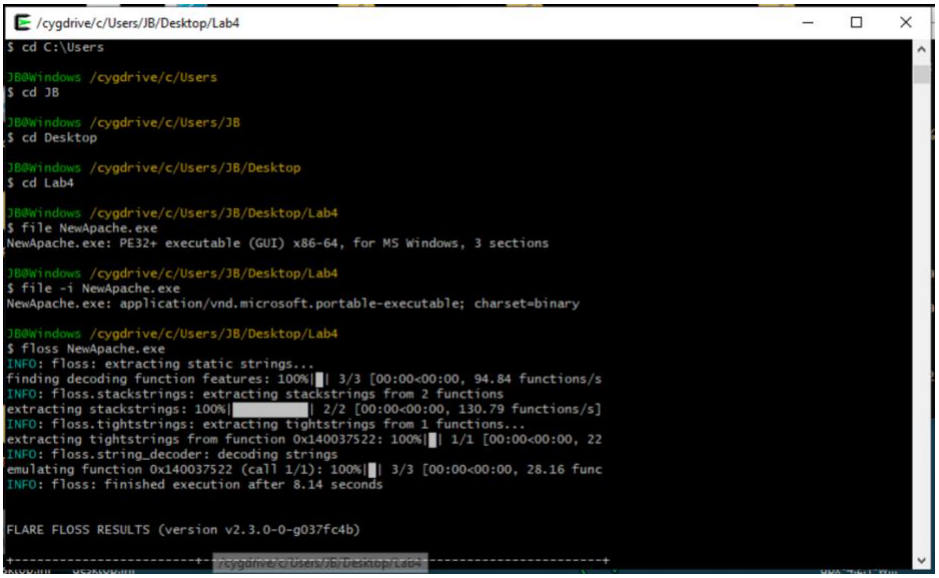
The user opened the file within the folders icon (C:\Users\JB\Desktop\Malware Analysis\ApacheMOD1.exe) and entered ‘cmd,’ which opened Windows Command Prompt. The user entered ‘floss apache.exe’ to utilize the “floss” (Free/Libre and Open Source Software) program, which compiled strings of source code within an executable file (see Figure 51). Please note many differences when comparing the packed and the unpacked files. For example, the packed file ran faster and contained less information (e.g., Floss Decoded Strings), as suspected, because it is compressed.

Figure 51: FLOSS Comparisons

NewApache.exe



ApacheMOD1.exe



The user executed “Detect It Easy v3.07 (DIE)” to assess if the binary was packed and what was used to pack it. According to the results, the binary is packed by Microsoft Visual C/C++(2017 v.15.8). ApacheMOD1.exe only has one section packed (i.e., .rsrc). Meanwhile, NewApache.exe displays packing by UPX (see Figures 52-54).

Figure 52: Detect It East ApacheMOD1.exe

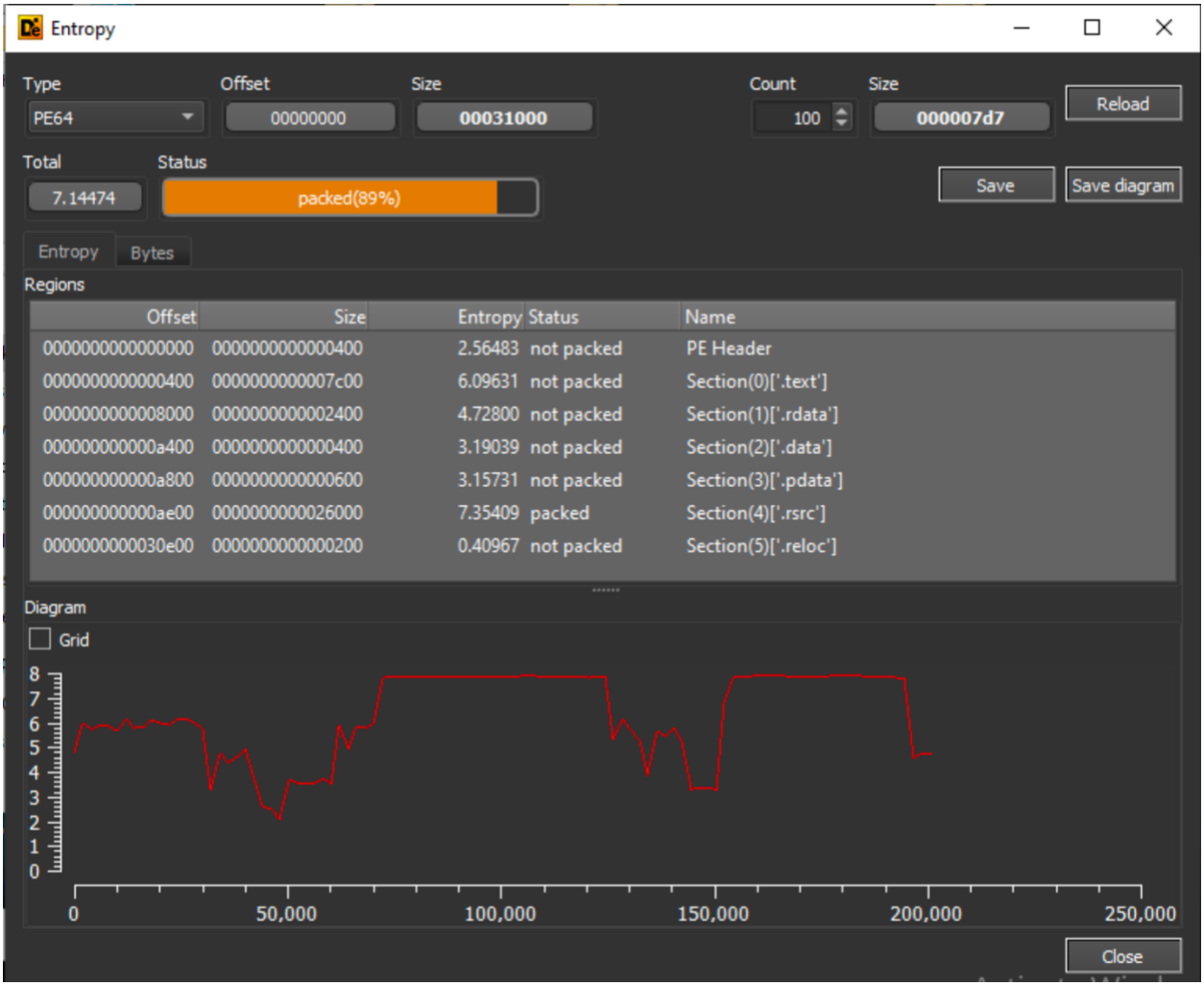


Figure 53: Detect It Easy NewApache.exe

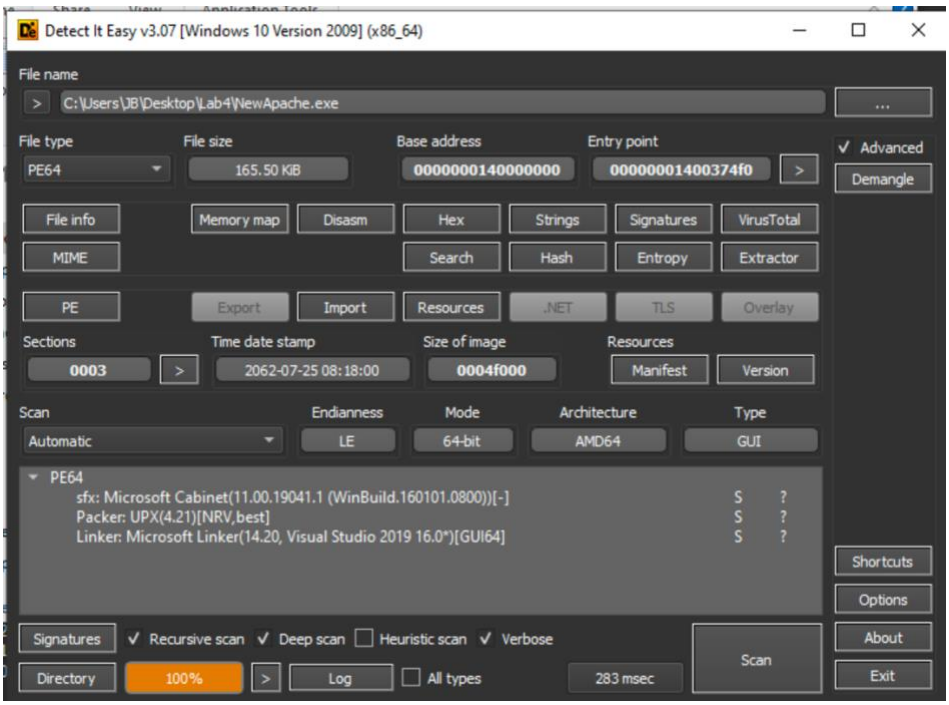
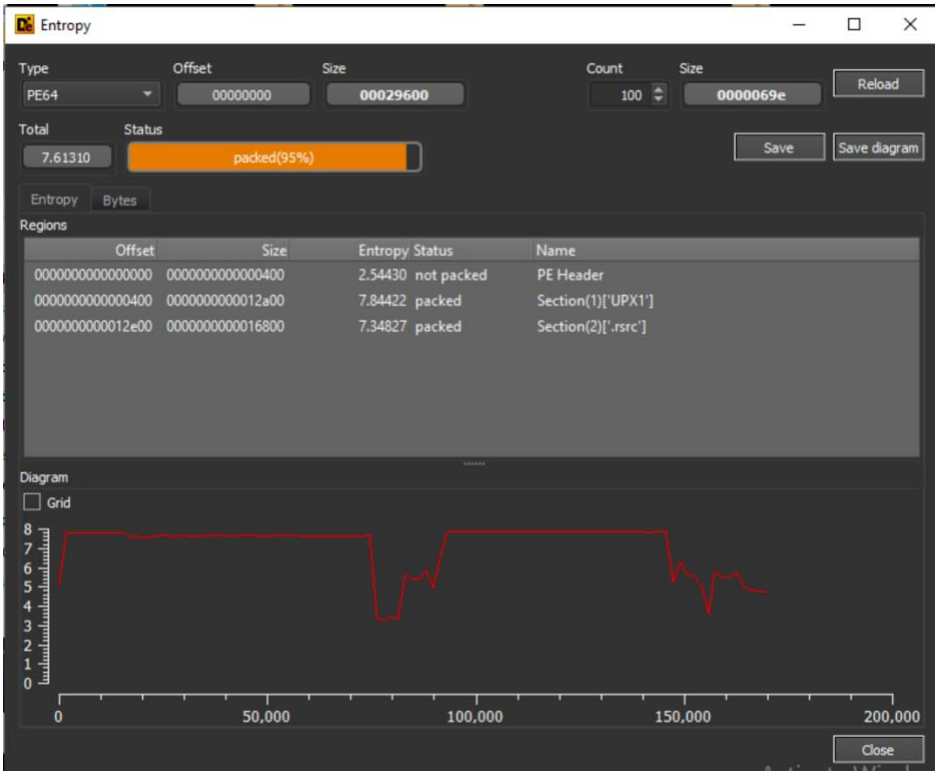


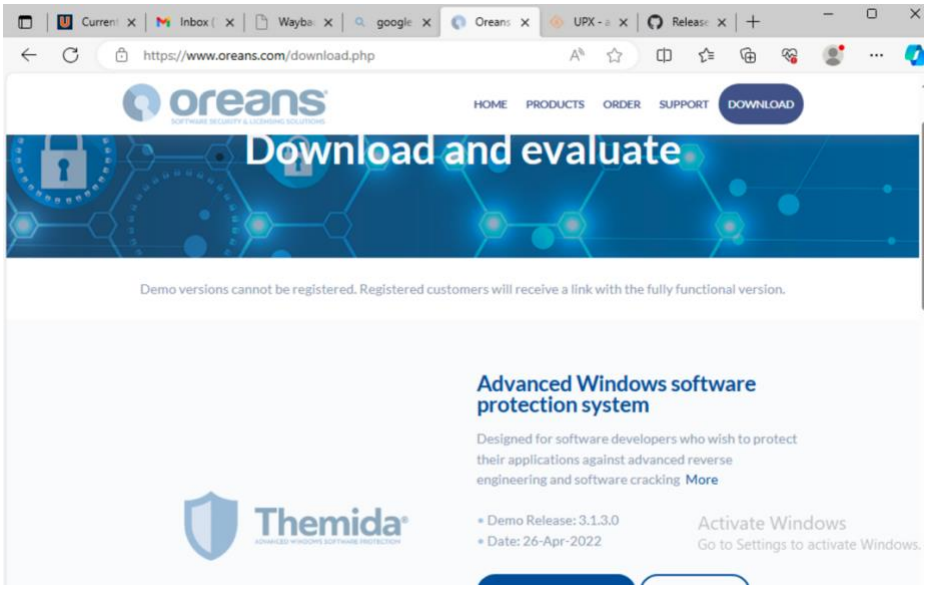
Figure 54: Detect It Easy NewApache.exe Entropy



Themida

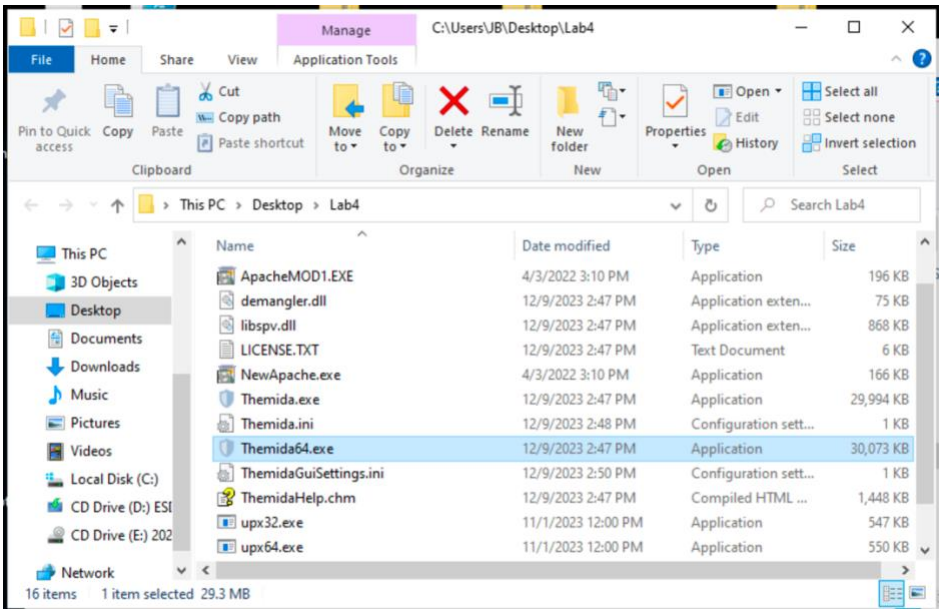
The user first went to the Oreans website and downloaded a demo version of the Themida program (see Figure 55).

Figure 55: Themida Download



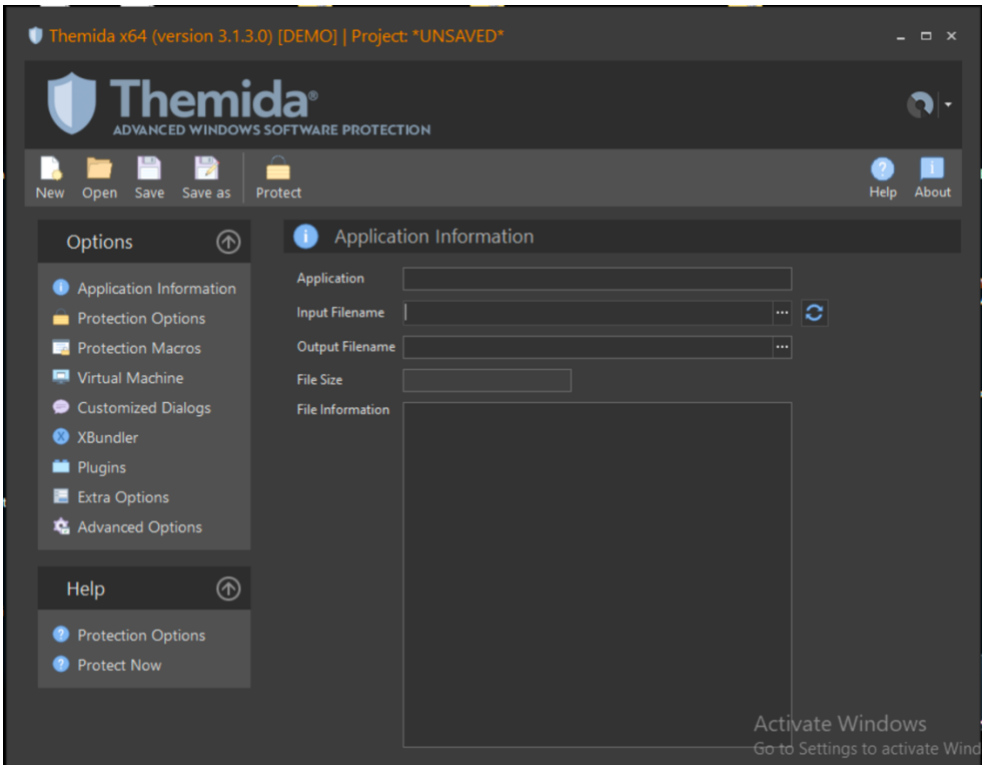
The user extracted the themida-packed file into Lab4 using Zip7 (see Figure 56).

Figure 56: Unpacked Themida



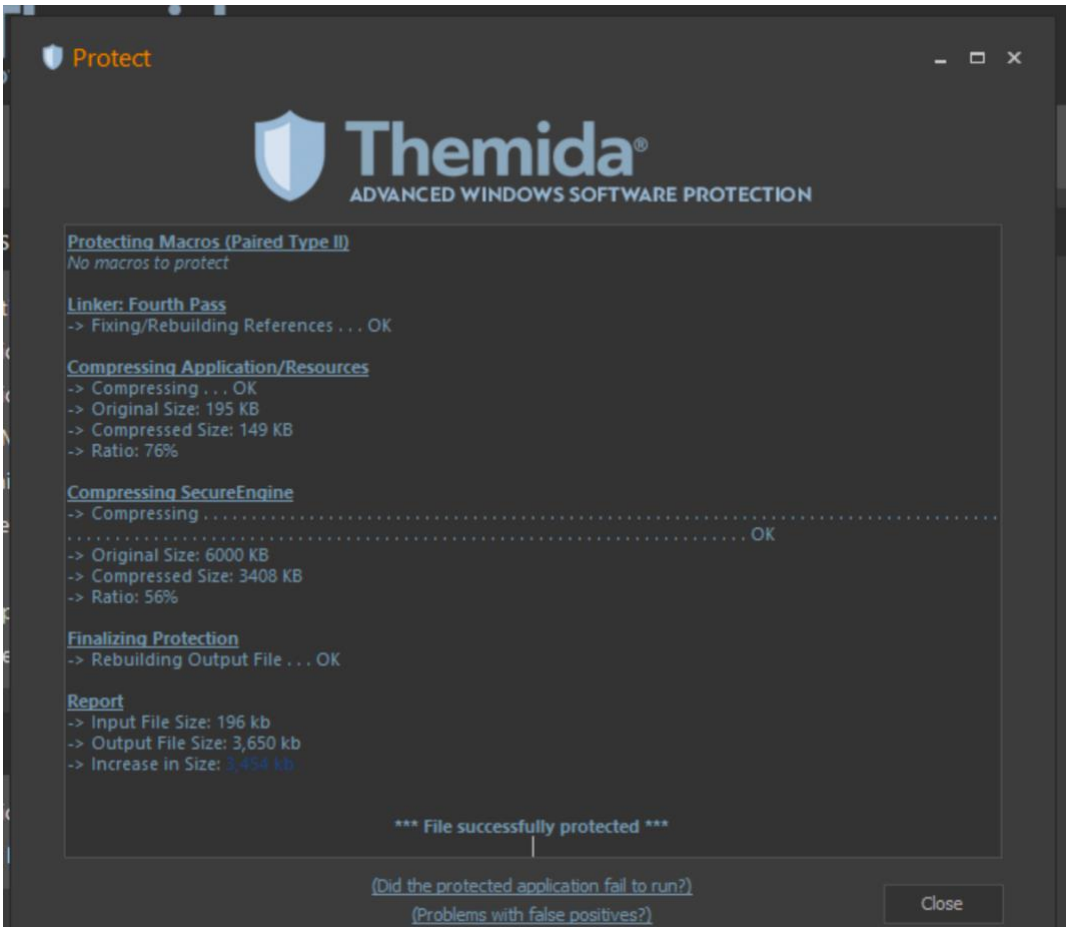
The user ran Themida64.exe as this would comply with the ApacheMOD1.exe file (see Figure 57).

Figure 57: Themida x64



The user ran ApacheMOD1.exe to pack and encrypt the file to ApacheMOD1_protected.EXE. Please note that the demo version has some limitations. Nevertheless, Themida displays a distinct difference in file sizes (see Figure 58).

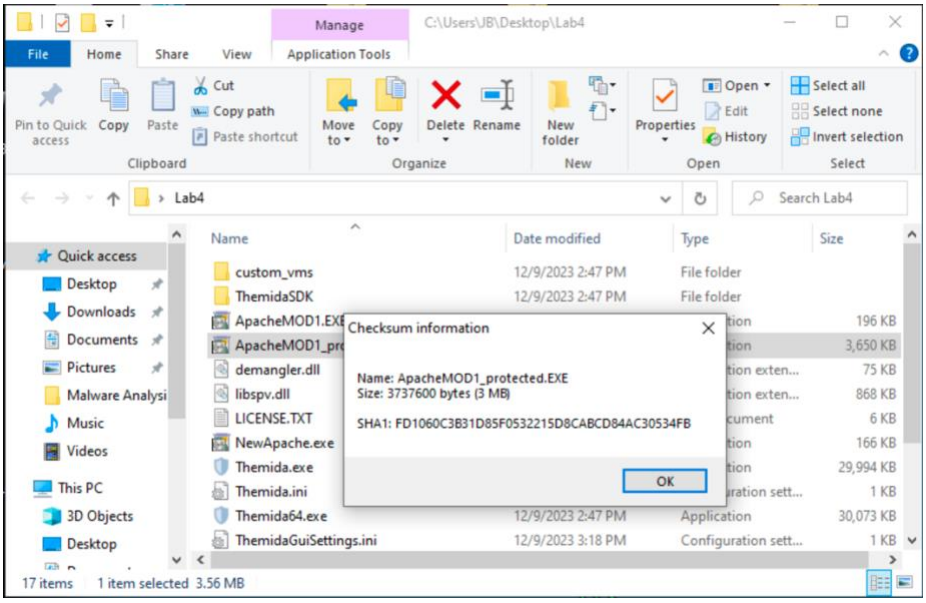
Figure 58: Themida



Static Analysis

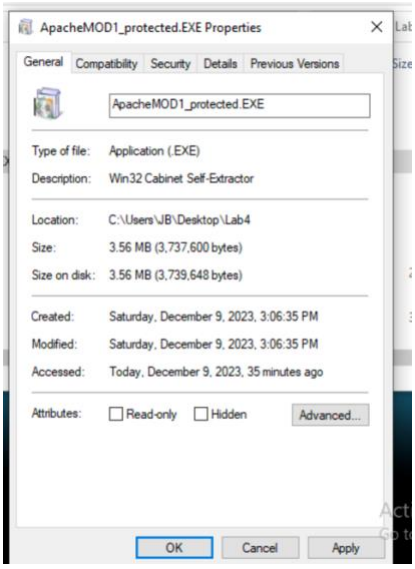
In order to allocate the Hash Values, the user right-clicked on the file →CRC SHA→SHA-1, which displayed the Hash File (see Figure 59).

Figure 59: Themida Packed ApahceMOD1_protected.EXE



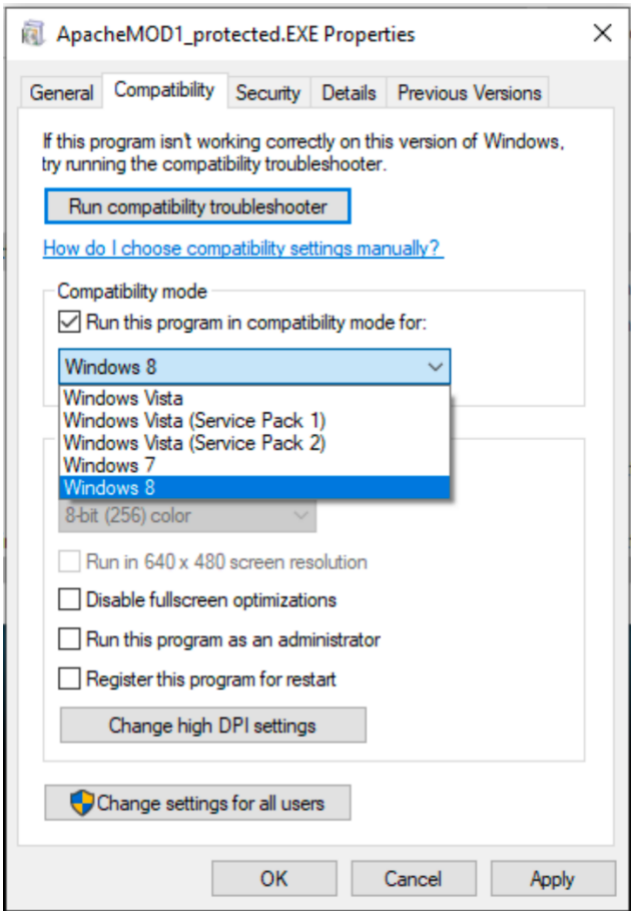
The user then right-clicked on the file again to inspect for Metadata. The user validated that there was no digital signature, suggesting a security risk. A digital signature validates who created the file and if the file has been altered or corrupted (Assaad, 2021) (see Figure 60).

Figure 60: Themida Packed ApahceMOD1_protected.EXE File Signature



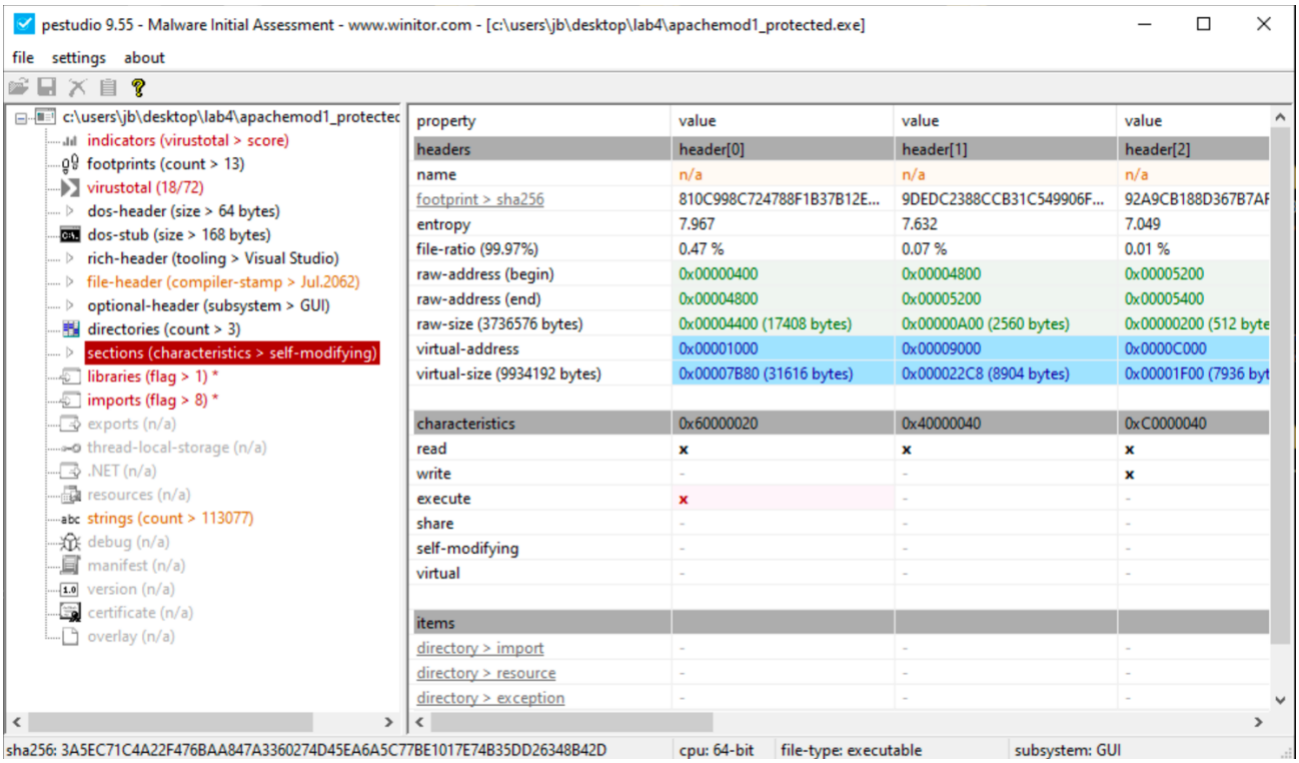
The user right-clicked on the file, clicked on ‘compatibility,’ checked ‘run this program in compatibility mode for,’ and then clicked the down arrow to identify the various Windows OS (see Figure 61); the choices evidenced a 64-bit architect type.

Figure 61: Themida Packed ApahceMOD1_protected.EXE Compatibility



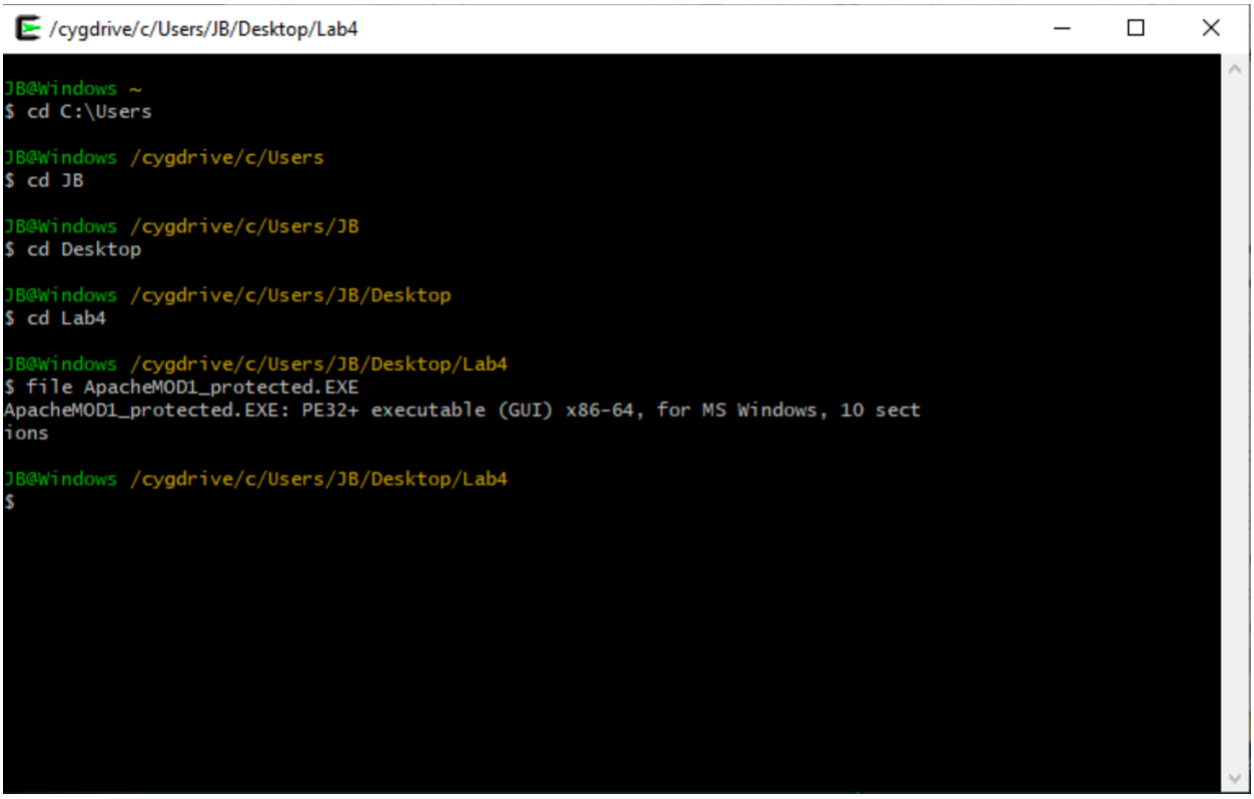
Then, the user executed ‘pestudio 9.55’ from the tools included within the Flare VM suite. The user dragged the ApacheMOD_Protected.exe file over for the program to evaluate its contents. After taking several minutes to populate data, the user noted several differences. First, many of the names were replaced with n/a, the majority of entropy values were close to 8, only 8 imports were reported, and “Cabinet.dll” was the only Windows systems file flagged (see Figure 62).

Figure 62: Pestudio Themida



The user then opened the start menu/Cygwin/Cygwin64 Terminal/cd C:\Users/cd JB\/cd Desktop/\cd Malware\ Analysis/file apache.exe (see Figure 63). This identifies the file type (PE32+ executable (GUI)) x86-64; for MS Windows, 10 sections make up the structure of the executable file (Karl-Bridge-Microsoft, n.d.).

Figure 63: (PE32+ executable (GUI)) x86-64, for MS Windows, 10 sections

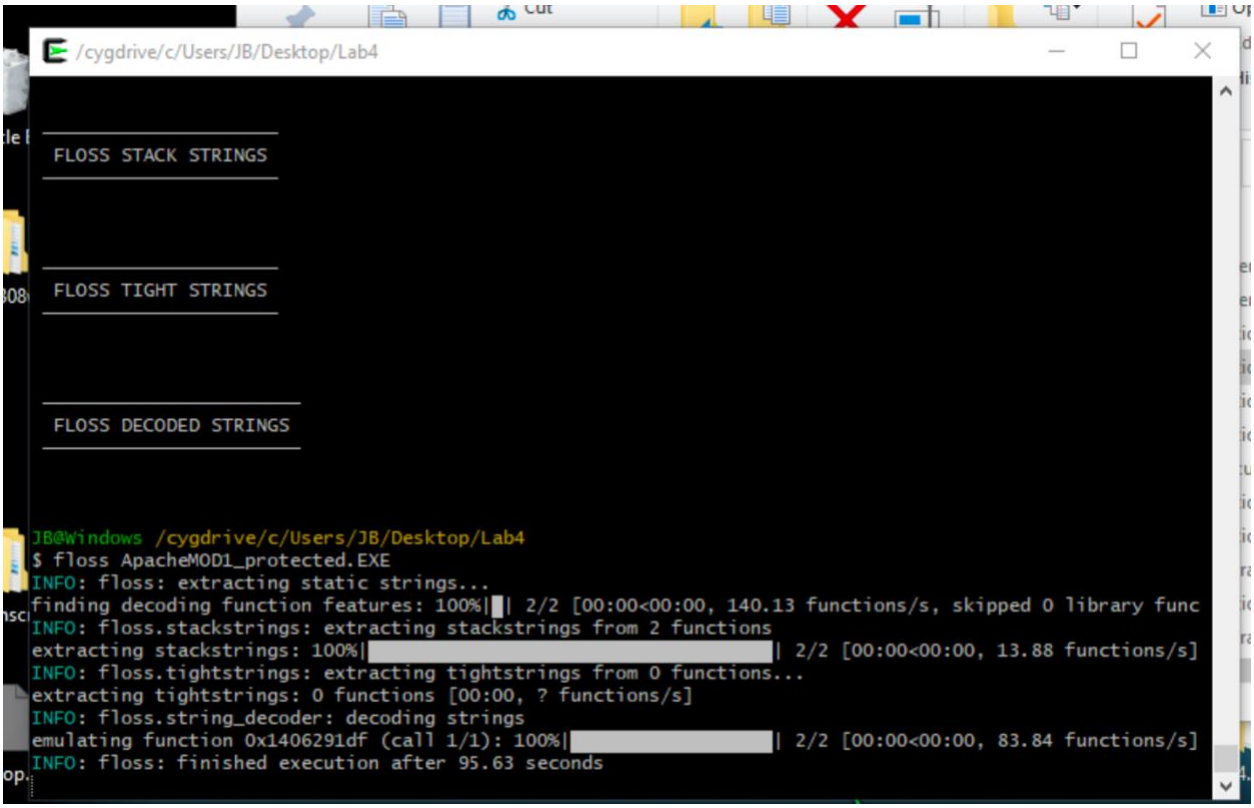


The user then executed ‘file -i ApacheMOD1_protected.EXE’ to obtain the MIME Type, which is ‘apache.exe: application/vnd.microsoft.portable-executable; charset=binary’ (see Figure 64).

Figure 64: ApacheMOD1.exe: application/vnd.microsoft.portable- executable; charset=binary

The user opened the file within the folders icon (C:\Users\JB\Desktop\Malware Analysis\ApacheMOD1_Protected) and entered ‘cmd,’ which opened Windows Command Prompt. The user entered ‘floss apache.exe’ to utilize the “floss” (Free/Libre and Open Source Software) program, which took much longer when compared to the unpacked file and packed file by UPX (i.e., 95.63 seconds). Also, there was no information regarding Floss Stack Strings, Tight Strings, and or Decoded Strings) (see Figure 65).

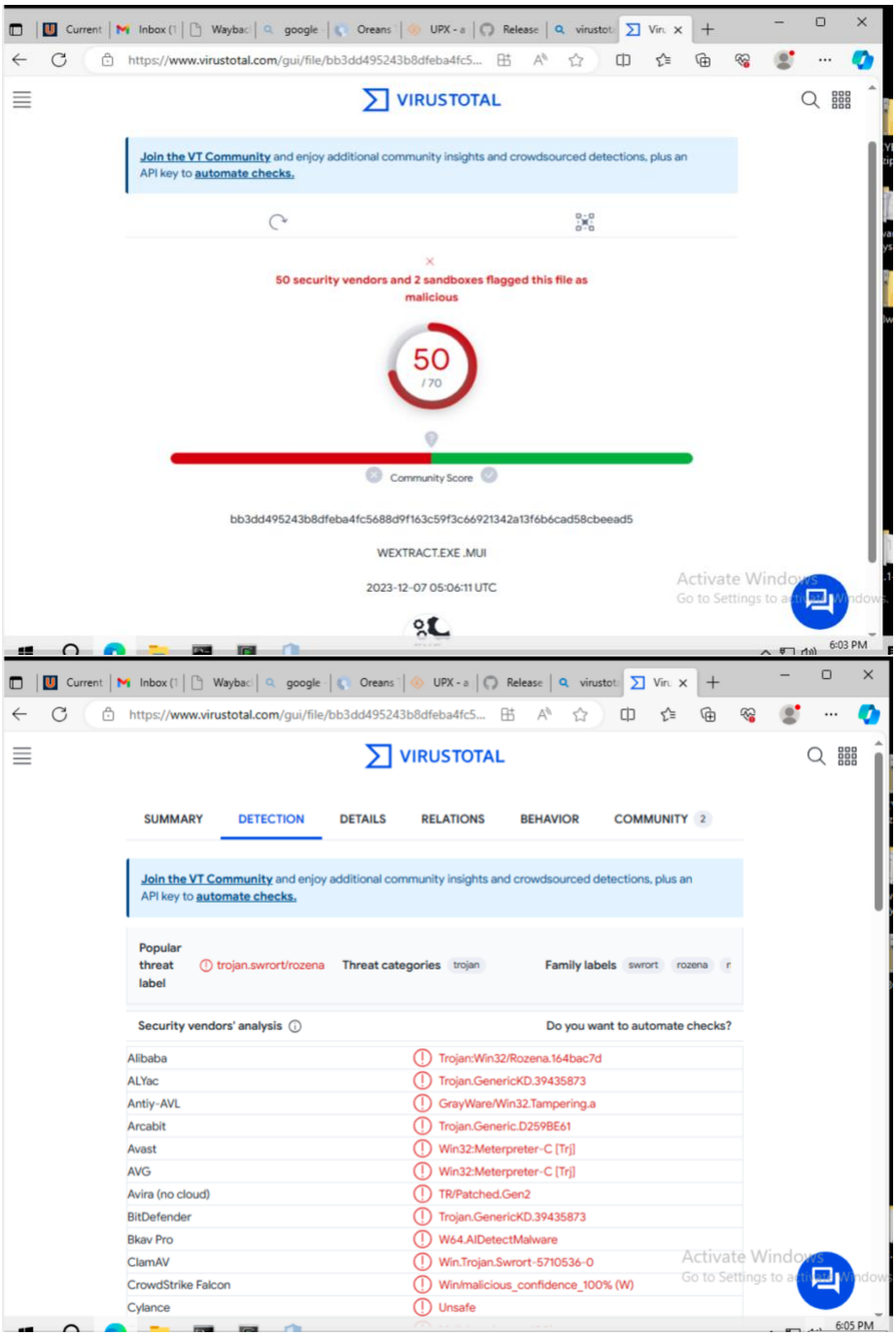
Figure 65: Floss ApacheMOD_Protected.EXE



Virus Total

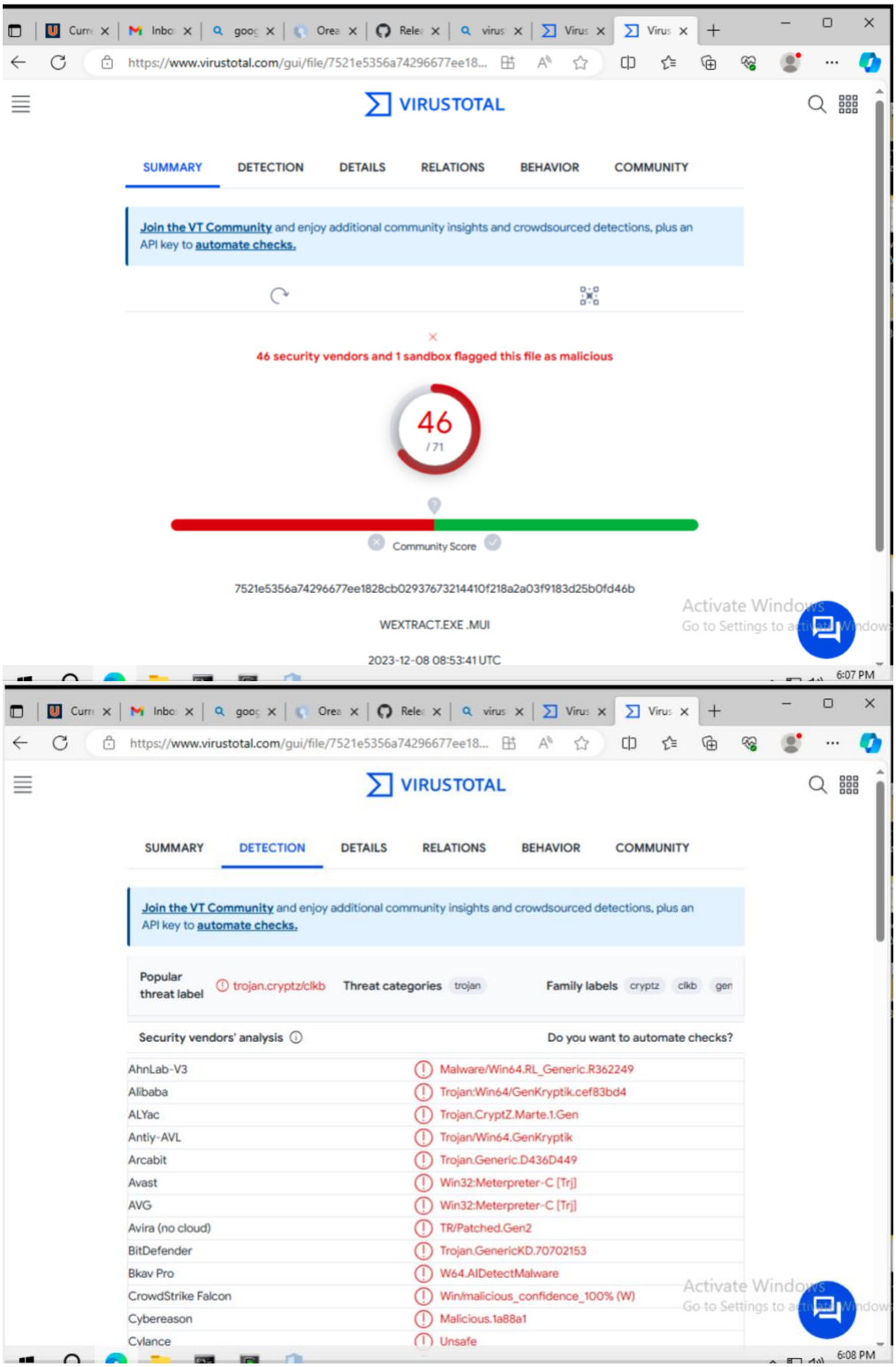
The user opened virustotal.com and entered the original malicious file (i.e., ApacheMOD1.exe), which populated the results (see Figure 66).

Figure 66: VirusTotal ApacheMOD1.EXE



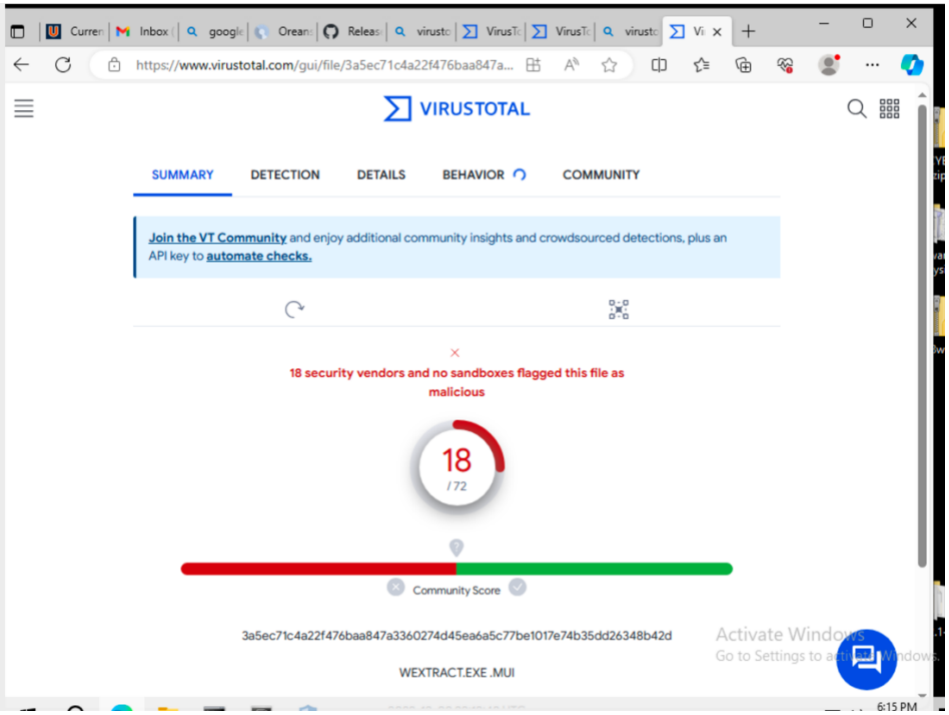
The user then entered the packed file by UPX (i.e., NewApache.exe) (see Figure 67).

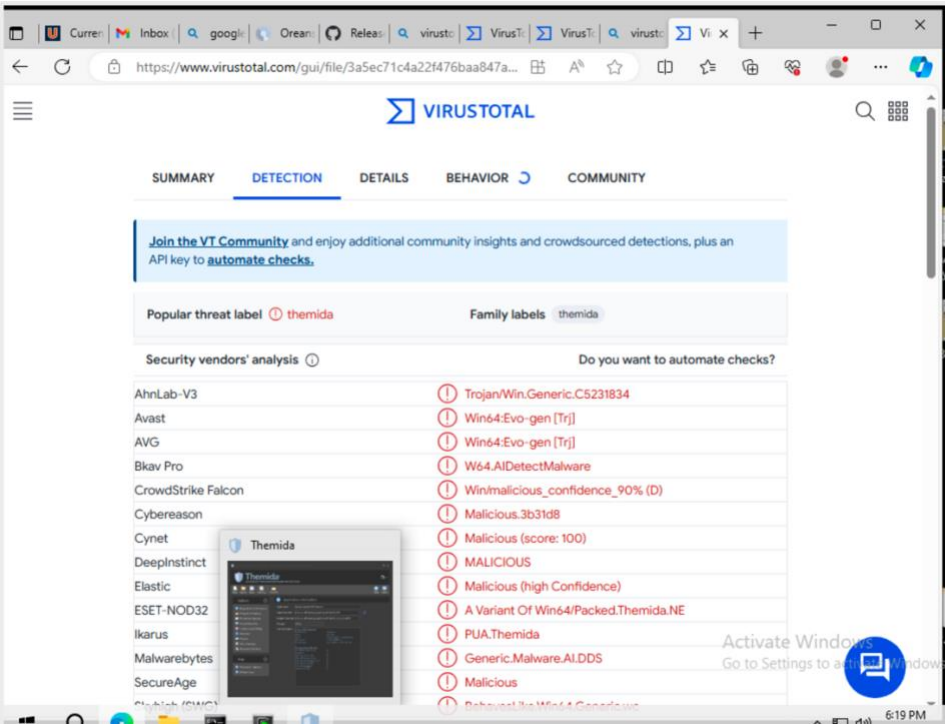
Figure 67: Virus Total NewApache.exe



The user then entered the packed file by Themida (i.e., ApacheMOD_Protected.exe). VirusTotal needed more time to run the program. Furthermore, Themida obfuscated the file much better than UPX (see Figure 68).

Figure 68: Virus Total ApacheMOD_Protected.exe)





The user compared the “undetected” files between UPX and Themida, and there were some notable mentions. For example, Kaspersky, McAfee, and Microsoft were flagged when UPX packed the file but undetected when packed by Themida (see Figure 69).

Figure 69: Comparison of detected and undetected files.

UPX

VIRUSTOTAL

CMC	Undetected
DrWeb	Undetected
Jiangmin	Undetected
K7AntiVirus	Undetected
K7GW	Undetected
NANO-Antivirus	Undetected
Palo Alto Networks	Undetected
QuickHeal	Undetected
SentinelOne (Static ML)	Undetected
SUPERAntiSpyware	Undetected
TACHYON	Undetected
TEHTRIS	Undetected
Trapmine	Undetected
TrendMicro	Undetected
VBA32	Undetected
VirIT	Undetected
ViRobot	Undetected
Webroot	Undetected
Xcitium	Undetected
Zillya	Undetected
Zoner	Undetected

Themida

Acronis (Static ML)	Undetected
Alibaba	Undetected
ALYac	Undetected
Antiy-AVL	Undetected
Arcabit	Undetected
Avira (no cloud)	Undetected
Baidu	Undetected
BitDefender	Undetected
BitDefenderTheta	Undetected
ClamAV	Undetected
CMC	Undetected
Cylance	Undetected
DrWeb	Undetected
Emsisoft	Undetected
eScan	Undetected
F-Secure	Undetected
Fortinet	Undetected
GData	Undetected
Google	Undetected
Gridinsoft (no cloud)	Undetected
Jiangmin	Undetected
K7AntiVirus	Undetected
K7GW	Undetected
Kaspersky	Undetected

Conclusion

The user met all of the lab's objectives by utilizing the tools within Flare VM to analyze ApacheMOD1.exe. The dynamic and static analysis results provided several indicators suggesting that the binary was malicious. These indicators include suspicious file names and hash values, the identification of signature information and imports and exports, And detailed information about the binary's architecture, sections, file type, and MIME type. Furthermore, packet analysis yielded the filename 'notepad.exe' communicating with IP address 192.168.100.100. Malicious behavior was evidenced when the user observed the data obtained from the string analysis.

Furthermore, the user also successfully packed ApacheMOD1.EXE with UPX and Themida, as evidenced in the former.

Based on the information provided, Themida was the superior packer as it obfuscated more information than UPX. For example, the original file size was 195 KB; UPX reduced the size to 166 KB, and Themida reduced the size to 149 KB. Pestudio displayed several differences between packed files, such as many names being replaced with "n/a," most entropy values being close to 8, only 8 imports being reported, and "Cabinet.dll" being the only Windows systems file flagged. Floss took substantially longer to run (i.e., 95.63 seconds compared to 15.22 seconds); less information was populated when running the file packed by Themida. Lastly, VirusTotal displayed Themida as the superior packer, as evidenced by an overall score of 18/72. Big-name companies such as Kaspersky, McAfee, and Microsoft did not detect files packed by Themida; however, they did identify packed files by UPX.

GitHub (n.d.) reported several variables to consider when packing a file such as using compression algorithms, reducing duplication (i.e., deduplication), archiving information, file segmentation, binary delta compression, file system choice, using custom compression schemes, organizing files in a hierachial fashion, use efficient backup strategies, archive using solid compression, and ultimate uncessary data.

In sum, UPX and Themida packed ApacheMOD1.EXE, but Themida was efficient and effective in obfuscating data.

References

Alvinashcraft. (n.d.-a). *Adjusttokenprivileges function (securitybaseapi.h) - win32 apps*. Win32 apps | Microsoft Learn. https://learn.microsoft.com/en-us/windows/win32/api/securitybaseapi/nf-securitybaseapi-adjusttokenprivileges

Alvinashcraft. (n.d.). *FreeSid function (securitybaseapi.h) - win32 apps*. Win32 apps | Microsoft Learn. https://learn.microsoft.com/en-us/windows/win32/api/securitybaseapi/nf-securitybaseapi-freesid

Alvinashcraft. (n.d.-b). *Deletefilea function (fileapi.h) - win32 apps*. Win32 apps | Microsoft Learn. https://learn.microsoft.com/en-us/windows/win32/api/fileapi/nf-fileapi-deletefilea

Assaad, A. E. (2021, December 3). *What is a digital signature & why is it important?*. Smallpdf. https://smallpdf.com/blog/what-is-a-digital-signature-and-why-is-it-important

Chaudhary, A. (2022, December 7). *What is static code analysis? its limitations and benefits*. Hatica. https://www.hatica.io/blog/static-code-analysis/

Chkadmin. (2021, July 15). *What is static code analysis?*. Check Point Software. https://www.checkpoint.com/cyber-hub/cloud-security/what-is-static-code-analysis/

Karl-Bridge-Microsoft. (n.d.). *PE format - win32 apps*. Win32 apps | Microsoft Learn. https://learn.microsoft.com/en-us/windows/win32/debug/pe-format

Mandiant. (n.d.). *Flare VM: The Windows Malware Analysis Distribution You've always needed!*https://www.mandiant.com/resources/blog/flare-vm-the-windows-malware

Obfuscated files or information: Software packing. Obfuscated Files or Information: Software Packing, Sub-technique T1027.002 - Enterprise | MITRE ATT&CK®. (n.d.). https://attack.mitre.org/techniques/T1027/002/

Packing-Box. (n.d.). *Packing-box/awesome-executable-packing: A curated list of awesome resources related to executable packing*. GitHub. https://github.com/packing-box/awesome-executable-packing

Reddy, K. K. (2019, December 28). *Basic static analysis of malware and common DLL*. Medium. https://medium.com/mrx-007/basic-static-analysis-of-malware-and-common-dll-ef9455d49968

Reji, A. R. (2023, October 30). *How to analyze windows executable files using PEStudio?*. The Sec Master. https://thesecmaster.com/how-to-analyze-windows-executable-files-using-pestudio/

Themida Overview. Oreans Technologies : Software Security Defined. (n.d.). https://www.oreans.com/Themida.php

UPX. (n.d.). https://upx.github.io/

VirusTotal, what is it and what is it good for?. RSS. (n.d.). https://cylab.be/blog/253/virustotal-what-is-it-and-what-is-it-good-for

What is kernel32.dll? - definition from Techopedia. (n.d.). https://www.techopedia.com/definition/3379/kernel32dll