Assignment 2

Justin Kennelly

University Of Arizona

CYBV 454 MALWARE THREATS & ANALYSIS

Professor Galde

February 14, 2022

LAB 3-1

* Lab03-01.exe : MD5 Checksum Value: d537acb8f56a1ce206bc35cf8ff959c0

LAB 3-2

* Lab03-02.dll : MD5 Checksum Value: 84882c9d43e23d63b82004fae74ebb61

LAB 3-3

* Lab03-03.exe : MD5 Checksum Value: e2bf42217a67e46433da8b6f4507219e

LAB 3-4

* Lab03-04.exe : MD5 Checksum Value: b94af4a4d4af6eac81fc135abda1c40c

# LAB 3-1

# Files Examined: Lab03-01.exe

## LAB 3-1 Question 1

What are this malware’s imports and strings?

Table

Description automatically generated

*Figure 1: PEview data returned from opening Lab03-01.exe*

*Graphical user interface, text, application, email

Description automatically generated*

*Figure 2: PEiD data returned from opening Lab03-01.exe*

Text

Description automatically generated

*Figure 3: Output from running the Strings command on Lab03-01.exe*

Text

Description automatically generated

*Figure 4: Continued output from running the Strings command on Lab03-01.exe*

I started analyzing Lab03-01.exe by first using static analysis techniques. I opened the file in PEview to see what imports it uses. Upon a closer look I found that Lab03-01.exe only appears to use the kernel32.dll library with the import ExitProcess (Figure 1). With the small number of visible imports shown from PEview I was led to believe that this file may be packed. I then opened the file in PEiD to back up my theory. After doing so, PEiD showed me that Lab03-01.exe is packed using PEncrypt 3.1 Final -> junkcode (Figure 2). Continuing my static analysis, I ran the strings command on Lab03-01.exe to see if I could glean any more information. A lot of strings were returned which is interesting because the file appeared to be packed initially. I noticed that in addition to the imports I already found there was also a string CONNECT%s%i HTTP/1.0 (Figure 3). This shows me that this file may start a two-way connection with a requested resource, by opening a tunnel. More interesting strings I found include more potential imports: advapi32 (used for API services), ntdll (common dll), user32 (common dll) (Figure 4). I could also see that there were strings relating to registry keys, a URL: www.practicalmalwareanalysis.com, and VideoDriver,WinVMX32-,vmx32to64.exe (Figure 4). At this point I have gained as much information from static analysis as I could so it was time to move onto dynamic analysis and running this file.

*Continued on next page*

Table

Description automatically generated

*Figure 5: Event properties returned from ProcMon targeting Lab03-01.exe*

I began my dynamic analysis by opening ProcMon, I stopped capturing, cleared the table and ran the file Lab03-01.exe. Once the file was running, I filtered my table for Process Name>is>Lab03-01.exe so I got only results that are relevant to this file. I ran the program for a few minutes and then went in to ProcessExplorer to suspended it. At this point I had gained all of the data required for some more analysis. I scrolled to the end of the data in the ProcMon window and right clicked to view the properties. I switched to the tab “Properties” I could see that the file used a lot more imports than previously known (Figure 5). The imports that stand out to me are: hnetcfg.dll, mswsock.dll, wshtcpip.dll, ws2help.dll, ws2\_32.dll, rpcrt4.dll all of these tell me that this file uses network capabilities in its process. The rest of the imports appear to be standard Microsoft dll files for running programs.

## LAB 3-1 Question 2

What are the malware’s host-based indicators?

Text

Description automatically generated

*Figure 6: Data returned from Process Monitor filtered on Lab03-01.exe*

*Continued on next page*

*Graphical user interface, text, application

Description automatically generated*

*Figure 7: Event properties for RegSetValue operation*

By running the file and analyzing it in Process Monitor I noticed some interesting operations. In Figure 6, we see that a file is created called vmx32to64.exe this also matches the string we saw during static analysis of the file in Figure 4. After it is created, we can also see that the file is written to the system. Also seen in Figure 6, we see an entry for a RegSetValue relating to the path HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\VideoDriver. To inspect further I double clicked the operation found in ProcMon and looked at the properties. With the event properties open I can see that vmx32to64.exe was written to the registry (Figure 7). Lab03-01.exe can be identified using vmx32to64.exe as a host-based indicator because In a compromised machine this data can be found added to the registry path HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\VideoDriver and set to run, I’m assuming during startup of the machine.

## LAB 3-1 Question 3

Are there any useful network-based signatures for this malware? If so, what are they?



*Figure 8: Operations found in ProcMon while running Lab03-01.exe*

Text

Description automatically generated

*Figure 9: Data retrieved while running FakeNet*

*Text

Description automatically generated*

*Figure 10: Data retrieved from NetCat listening on port 443*

*Graphical user interface, application

Description automatically generated*

*Figure 11: Packet view of Network Traffic*

While running Lab03-01.exe I was able to notice network calls being made while viewing data in Process Monitor. I saw that there was TCP traffic occurring over port 443 and data was being sent 256 bytes in size (Figure 8). Additionally, while running FakeNet, I recognized a DNS query under the name www.practicalmalwareanalysis.com (Figure 9) which also matches the initial string found in Figure 4. Wireshark packets from FakeNet also verified that the TCP connection occurred in Figure 11. Lastly, I reboot the analysis machine. I could see in Process manager that vmx32to64.exe was a running process at startup. I opened the cmd prompt and ran NetCat to listen over port 443. After some time, I could see some captured data appear, it was unreadable but it shows that network traffic was occurring. Network-based indicators with this malware are that TCP packets of data can be found being sent over port 443 after connecting to the URL www.practicalmalwareanalysis.com.

# LAB 3-2

# Files Examined: Lab03-02.dll

## LAB 3-2 Question 1

## How can you get this malware to install itself?

Table

Description automatically generated

*Figure 12: Data returned from opening Lab03-02.dll in PEview*

*Table

Description automatically generated*

*Figure 13: Data returned from opening Lab03-02.dll in PEview*

*Table

Description automatically generated*

*Figure 14: Export Directory data within PEview for Lab03-02.dll*

*Text

Description automatically generated*

*Figure 15: Output from Strings command on Lad03-02.dll*

Text

Description automatically generated

*Figure 16: Output from Strings command on Lad03-02.dll*

Like all malware analysis I started with a static look at Lab03-02.dll. I began by looking at the imports through PEview. There were a number of Imports and Exports shown which leads me to believe this file is not packed. There were interesting imports I found such as (Figure 12 & 13): ADVAPI32.dll (access provided to Windows components like Service Manager and Registry), KERNEL32.dll (Core functionality to access and manipulation of files, memory, and hardware), WININET.dll (High-level networking functions implementing protocols like FTP, HTTP, NTP), WS2\_32.dll (Potentially performs networking tasks or connects directly to a network). A closer look at the import functions helps us get a clearer idea of the functionality of the program. The functions that stand out are: RegOpenKeyExA (Opens a handle to the registry key for reading and writing, could be used for persistence, and contains a lot of information of the OS and app settings), OpenSCManager (Opens a service control manager handle), CreateService (Could create a boot time service, in malware it is used for persistence, stealth, or loading kernel modules), RegSetValueExA (Could be used to manipulate the registry files and create a new entry), GetTempPath (Returns temporary file path, should check to see if files are read or written to the temp path), GetProcAddress (Returns memory address of DLL functions loaded to memory, could be used in identifying import functions), GetModuleFilename ( Gets filename of the module that is loaded into the memory address, malware uses this function to modify or copy files in the running process). In Wininet.dll we see many functions relating to HTTP requests, HttpOpen/Send requests tells me this file could reach out to a URL and send/ receive data. Taking a look at the Exports seen in Figure 14, There are Install/Uninstall export functions and a ServiceMain which tells me a file could be installed and then ran as a service. Continuing with my static analysis I took a look at the files strings. A lot of strings were returned, in Figure 15, we see networking strings like: GET and HTTP/1.1 which could hint at the file making a get request to a URL and part of a potential header file. We see strings that relate to commands like: quit, exit, getFile, cmd.exe /c, parameters, type, start. We also see many registry file paths strings that could be useful in our dynamic analysis. Figure 16, shows the strings related to the Export functions I noticed in PEview, from what I have learned DLL files are not run the same way as EXE’s therefore seeing these strings leads me to believe they could be used as arguments to install the DLL file. This leads us into the Dynamic portion of the analysis of Lab03-02.dll.

Graphical user interface

Description automatically generated

*Figure 17: Screen capture of Dynamic Analysis environment*

*Text

Description automatically generated*

*Figure 18: Installation of the file Lab03-02.dll*

*Text, letter

Description automatically generated*

*Figure 19: Data returned from comparing Shot 1 & 2 after installing Lab03-02.dll*

To start I set up my dynamic analysis environment by opening Process Monitor, Process Explorer, Start FakeNet, and take an initial registry snapshot using RegShot (Figure 17). After the registry shot has been taken I open a CMD prompt and install the malware by running rundll32.exe Lab03-02.dll, installA (The potential argument I discovered in PEview Exports & Strings command) (Figure 18). Having Process Explorer up and running allows me to see when the command starts and finishes as a process, additionally there were no errors returned from the command so at this point I am confident that it was installed. To backup this idea I can take a second RegShot to see if there were any changes made to the registry after installing the DLL. After comparing the two shots I can clearly see that 27 changes have been made to the registry after installing Lab03-02.dll (Figure 19).

## LAB 3-2 Question 2

## How would you get this malware to run after installation?



*Figure 20: Closer look at returned RegShot data from Shot 1 & 2*

*Text

Description automatically generated*

*Figure 21: Running the created service IPRIP*

I see that a key was added with the path: HKLM\SYSTEM\CurrentControlSet\Services\IPRIP, I remember also seeing the string IPRIP in my static analysis while looking at strings in Figure 15. Under values added in Figure 19, we see the path %SystemRoot%\System32\svchost.exe -k netsvcs as the ImagePath (The same string found in Figure 15). This tells me that the file has been set to run and managed as a svchost.exe as the service IPRIP. Looking back at the imports identified in my static analysis, OpenSCManager and CreateService import functions could have made this possible and further back my claim. I then start the service IPRIP and receive confirmation messages stating that the INA+ service has started successfully (Figure 21).

## LAB 3-2 Question 3

How can you find the process under which this malware is running?

A picture containing table

Description automatically generated

*Figure 22: Process Explorer data looking at svchost.exe*

At this point I understand that the malware has been set to run as a service in the svchost.exe path so with Process Explorer still running I am able to look through the svchost.exe processes until I find identifying attributes of the malware file. I used the view tab to show DLL information in the lower portion of the window and selected each svchost.exe process until I found something. I the process with Lab03-02.dll loaded and hovered over it to see the IPRIP service name within its data (Figure 22).

## LAB 3-2 Question 4

## Which filters could you set in order to use procmon to glean information?

Table

Description automatically generated

*Figure 23: Service host information retrieve from Process Explorer*

Graphical user interface

Description automatically generated with low confidence

*Figure 24: Process Monitor data filtered on our service host*

Initially I thought I could simply apply a filter in ProcMon for process name is svchost.exe, but quickly discovered that this became convoluted with the other processes named svchost.exe (duh..). I went back to Process Explorer and found the unique PID specific to the service I am analyzing (Figure 23). I set the filter on ProcMon to PID is 864 and started capturing data on just the IPRIP service (Figure 24).

## LAB 3-2 Question 5

## What are the malware’s host-based indicators?

Referring back to Figure 19, we now understand that when Lab03-02.dll is installed it creates a new registry key for the service IPRIP at the path: HKLM\SYSTEM\ControlSet001\Services\IPRIP. A value is added to run a service in the svchost.exe processes with the display name: Intranet Network Awareness (INA+). The created registry keys with paths that appear like HKLM\SYSTEM\ControlSet001\Services\IPRIP\ or HKLM\SYSTEM\CurrentControlSet\Services\IPRIP and all subfolders can be used as a host-based indicator on infected machines.

## LAB 3-2 Question 6

Are there any useful network-based signatures for this malware?

Text

Description automatically generated

*Figure 25: Data returned from FakeNet on port 80*

Text

Description automatically generated

*Figure 26: HTTP packet captured while running FakeNet opened in WireShark for analysis*

By using FakeNet while running the malware service we are able to capture the network traffic. In Figure 25, I noticed that the service resolves the domain name for www.practicalmalwareanalysis.com. Additionally, I can see that a connection is established at port 80 and a GET request is made for serve.html, which is another string recognized from the static analysis strings command (Figure 25). I looked at the packet data captured in WireShark and followed the HTTP and TCP stream. I found the same data from FakeNet in the HTTP packet with a length of 156 bytes from source: 127.0.0.2 to destination: 127.0.0.1 several times. This request made over port 80 to the host www.practicalmalwareanalysis.com can be used as a network-based identifier.

# LAB 3-3

# Files Examined: Lab03-03.exe

**Initial Static Analysis:**

**Table

Description automatically generated**

*Figure 27: Data returned from opening Lab03-03.exe in PEview*

*Table

Description automatically generated*

*Figure 28: Resource data returned from opening Lab03-03.exe in PEview*

*Text

Description automatically generated*

*Figure 29: Output from Strings command on Lad03-03.exe*

*A picture containing table

Description automatically generated*

*Figure 30: Output from Strings command on Lad03-03.exe*

## LAB 3-3 Question 1

## What do you notice when monitoring this malware with Process Explorer?

Table

Description automatically generated

*Figure 31: Captured data from Process Monitor filtered on Lab03-03.exe*

*Graphical user interface

Description automatically generated with medium confidence*

*Figure 32: Data from Process Explorer*

After completing my initial static analysis I set up the environment for dynamic analysis. Once all of my tools were setup and running (and taking an initial registry snapshot) I ran the file Lab03-03.exe by double clicking on it. Immediately, FakeNet shut down which I found quite odd. I did manage to capture data in ProcMon, interesting operations I noticed was a FileSystemControl operation directed to the path C:\Labs\BinaryCollection\Chapter\_3L which is where the Lab03-03.exe file is held. Next, a file was created called svchost.exe at the path C:\WINDOWS\system32\svchost.exe. Moving down the list and highlighted in teal in Figure 31, we see that a process was created at the same path: C:\WINDOWS\system32\svchost.exe and then Lab03-03.exe is closed. Based off what I learned from my static analysis analyzing Lab03-03.exe Imports in PEview (Figure 27) I understood that KERNEL32.dll was the only import, this import has the ability to modify access files, memory, and hardware. Among the functions imported are: VirtualAllocEX (Memory-allocation that can allocate memory in remote processes, Malware can use this for process injection), Write/ReadProcessMemory (Ability to write and read data to/from memory, potential process injection). There are also many file manipulation imports as well as thread and resource manipulation imports. With the information gleaned from my static analysis backed by live dynamic analysis I am confident that the Lab03-03.exe file executed only to allocate and run the svchost.exe process and then close. Since a process was created I referred to my Process Explorer window to see if I could identify it. I knew I was looking for a process called svchost.exe from information I had already gained. Sure enough, it was fairly easy to identify which svchost.exe to look for because in Process Explorer there was a svchost.exe process running, but not in its typical location which would be within the services.exe tree (Figure 32).

## LAB 3-3 Question 2

## Can you identify any live memory modifications?

Graphical user interface, application

Description automatically generated

*Figure 33: Compared Image and Memory strings for svchost.exe service in Proc Exp*

After identifying the created service I was able to look at its’ properties in Process Explorer. To view memory modifications I compared the strings from the image and the strings from the memory. I noticed a lot of differences in the memory strings like import functions for mapping getting, and setting strings (Figure 33),. I also saw string blocks that were keyboard commands such as: SHIFT, ENTER, BACKSPACE,TAB, among other things I had already written on the keyboard (Figure 33). Another interesting difference was a log file in the memory strings called practicalmalwareanalysis.log (Figure 33).

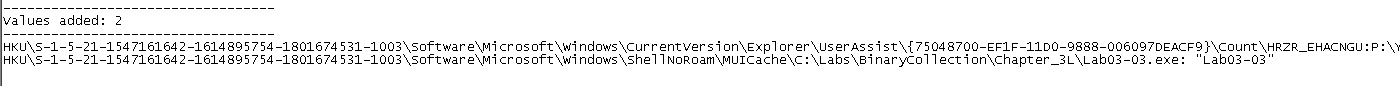
## LAB 3-3 Question 3

## What are the malware’s host-based indicators?

Graphical user interface, application, Word

Description automatically generated

*Figure 34: Folder where Lab03-03.exe is originally stored with a newly created log file*

**

*Figure 35: RegShot comparison of shot before malware execution and after*

I still had the C:\Labs\BinaryCollection\Chapter\_3L folder open while performing my dynamic analysis. After I ran Lab03-03.exe some time had passed and I noticed practicalmalwareanalysis.log appear in the same folder (Figure 34). This log file can be used to as a host-based indicator of intrusion. Additionally, I took a second RegShot after the execution of Lab03-03.exe. This is where I saw several modifications being made to the registry at paths: HKU\S-1-5-21-1547161642-1614895754-1801674531-1003\Software\Microsoft\Windows\CurrentVersion\Explorer\UserAssist\

HKU\S-1-5-21-1547161642-1614895754-1801674531-1003\Software\Microsoft\Windows\ShellNoRoam\MUICache\C:\Labs\BinaryCollection\Chapter\_3L\Lab03-03.exe: "Lab03-03"

HKLM\SYSTEM\ControlSet001\Services\WinSock2\Parameters\Protocol\_Catalog9\Catalog\_Entries\000000000001\PackedCatalogItem

HKLM\SYSTEM\CurrentControlSet\Services\WinSock2\Parameters\Protocol\_Catalog9\Next\_Catalog\_Entry\_ID

These registry modifications could also be used to identify host-based intrusion (Figure 35).

## LAB 3-3 Question 4

What is the purpose of this program?

Graphical user interface, text, application, email

Description automatically generated

*Figure 36: log file created by svchost.exe*

With all of the information gleaned from the static and dynamic analysis performed we discovered that Labo3-03.exe executes, the creates a rogue process called svchost.exe that will log keystrokes and save them to a log file called practicalmalwareanalysis.log. Opening the log file we can clearly see the keystrokes that have been recorded (Figure 36). This malware is a keylogger.

# LAB 3-4

# Files Examined: Lab03-04.exe

**Initial Static Analysis:**

Table

Description automatically generated

*Figure 37: Import data returned from opening Lab03-04.exe in PEview*

Table

Description automatically generated

*Figure 38: Import data returned from opening Lab03-04.exe in PEview*

Table

Description automatically generated

*Figure 39: Import data returned from opening Lab03-04.exe in PEview*

*Text

Description automatically generated*

*Figure 40: Strings data returned from Lab03-04.exe*

*Text, timeline

Description automatically generated with medium confidence*

*Figure 41: Strings data returned from Lab03-04.exe*

*Text

Description automatically generated*

*Figure 42: Strings data returned from Lab03-04.exe*

## LAB 3-4 Question 1

What happens when you run this file?

Graphical user interface, text, application, email

Description automatically generated

*Figure 43: Captured data from Process Monitor filtered on Lab03-04.exe*

*Graphical user interface, application

Description automatically generated*

*Figure 44: Event Properties shown for Process Create Operation Lab03-04.exe*

Static analysis of Lab03-04.exe shows that Lab03-04.exe Import functions have the ability to manipulate services and registries using the ADVAPI32.dll (Figure 37). We can also see that Network functionality is being imported using WS2\_32.dll, but the functions are being called by ordinal (Figure 38). Finally, KERNEL32.dll module has many imports that range in functionality from creating processes, file manipulation, directory navigation, library loading, command line usage, string mapping, and heap manipulation (Figure 39). Moving onto strings, the command returned a lot of strings for Lab03-04.exe. We see heap error strings and returns, extensions: .bat, .com. .cmd, Networking strings like DOMAIN error and GET. Additionally we see commands like Download, Upload, Sleep, potential command arguments like -cc, -re, -in, a URL pointing to http://www.practicalmalwareanalysis.com and a file path to %SYSTEMROOT%\system32 (Figure 42). After analyzing this data I set up my dynamic analysis environment. I took an initial RegShot and ran Lab03-04.exe. I immediately saw a command prompt appear and after a few seconds it closed and Lab03-04.exe was removed from its folder at C:\Labs\BinaryCollection\Chapter\_3L. I observed the processes in ProcMon and discovered a Process Create operation (Figure 43). By double clicking on the process operation I was able to view its properties and could identify that the process created was a command line operation that did in fact delete Lab03-04.exe from the system at the same folder path (Figure 44).

## LAB 3-4 Question 2

## What is causing the roadblock in dynamic analysis?

Timeline

Description automatically generated

*Figure 45: Data found in Process Explorer*

*Text

Description automatically generated*

*Figure 46: Data in FakeNet While executing Lab03-04.exe*

*Text

Description automatically generated with medium confidence*

*Figure 47: Comparison of Shot 1 & 2 using RegShot before and after execution*

The roadblock for analysis of this file is caused by a lack of knowledge and clues to operation. I checked Process Explorer to see if any processes had been created and did not find any running processes that weren’t already running before execution (Figure 45). I had FakeNet running during execution and did not notice anything other that a DNS query at address: 47.197.139.10.in-addr.arpa (Figure 46). I do not think this query is related to this program at this time. Lastly, I looked at the registry changes obtained from RegShot before and after execution and could not find any clues to assist in my analysis at this time (Figure 47).

## LAB 3-4 Question 3

Are there other ways to run this program?

With all of the completed static and dynamic analysis completed I did not find another way to run this program at this time. The strings found in Figure 42 lead me to believe that further execution of another program will need to be executed, but I did not find a way to accomplish this at this time.