Course: CY5210 Information System Forensics
Instructor: Elton Booker
Assignment: Lab Assignment 6B
Student Name:

Scope: In this assignment, students will learn the basics of malware triage, static analysis, and how to analyze memory using the standalone Volatility application.

MALWARE TRIAGE

Step 1: Download the assignment files from Canvas or the noted OneDrive directory. Although students will review general program attributes, and review some adware, they will not perform dynamic malware analysis or reverse engineering of any malware, since this is outside the scope of this course.

This section will focus on identifying whether or not a file is malicious, how to identify basic characteristics automatically, or how to identify a program's capabilities before spending time performing static malware analysis. This same process can serve as a checklist for each sample in this assignment and is one of many processes used by analysts in the field.

Students will complete steps 2 and 3 below on an adware sample to familiarize themselves with malicious program attributes without infecting any systems. Some programs have functions and operations that are similar to truly malicious applications. This sample is the malware sample pulled from the Lab 2 image.

Step 2: Students will run an application through several automated tools to become familiar with some capabilities to quickly triage a program to determine if additional analysis is necessary. This is more of a triage process and is not malware analysis.

Answer the following questions related to the triage of the "Cat_And_Dog_Screensaver.exe" file exported from the Lab 2 image. This file is in the assignment .ZIP file on Dropbox (link posted on Blackboard).

1. DO NOT EXECUTE THE FILE
2. Upload the file to VirusTotal (https://www.virustotal.com/#/home/upload)

Name: Cat_And_Dog_Screensaver.exe

a. How many engines detected the file?

26 security vendors and no sandboxes flagged this file as malicious

b. What name did McAfee give to this file?

McAfee gave this file listed as "Artemis!95DC5CCEF3CB" or the GW-Edition gave it "Artemis"

c. What did Microsoft name this file?

Microsoft named this file "Trojan:Win32/Zpevdo.B"

d. What type of malware might this .exe be?

The type of malware the .exe may be is a "Trojan" as identified by Alibaba, K7AntiVirus, K7GW, Lionic, Microsoft, NANO-Antivirus, Symantec, VBA32, Yandex, and Zillya

3. Look at the Details tab
   a. What is the MD5 hash?

MD5:
95dc5ccef3cba81028da04b28dcbe4da

   b. What is the file type?

FileType: Win32 EXE

   c. Name 2 specific "Registry Keys" that may be set and used for IOCs or detection methods across your enterprise for this suspicious screensaver.

4. I would have students download and install the cuckoo sandbox (https://cuckoosandbox.org/about), but I have removed this step in the sake of time. Please move forward to the static analysis (Step #3).

Step 3: Students will perform static malware analysis on a binary to identify capabilities, attributes and characteristics of the sample.

1. Download the .ZIP file named "Assignment 10 Tools."

2. Unzip and install the following tools with the defaults:
   a. ExplorerSuite
   b. BinText
   c. PEiD
   d. Strings (command-line tool and does not require install)

3. **Strings**
    a. In the Strings directory, run the following command: ***strings64.exe*** to view the command line options. This tool is very basic, but powerfull.
    b. Run the command ***strings64.exe \<full path>\filename more*** and you'll quickly see how many strings can be pulled from a file. Typically you would use grep to search for particular files or pipe this command through the more command. This will allow you to page through the identified strings. Press the spacebar to page through and review some of the processes, handles and system calls.
    c. There are also switches to change the default length of strings, and using grep (or findstr on Windows) is useful if you're looking for a specific function or system call. Run this command: ***strings64.exe \<full path>\filename | findstr Open.*** What are the number and name of the results?

$.\strings64.exe '..\..\Lab Assignment Evidence Files\Cat_And_Dog_Screensaver.exe' | findstr Open

OpenClipboard
RegOpenKeyExA
OpenProcessToken

4. **BinText**
    a. Import the screensaver file into BinText 3.0.3 by using the **Browse** function and click **Go** to begin scanning the file for strings. You'll notice the same results are collected as with the command-line utility.
    b. Under the "**Search**" tab, you'll see a header and some sections that are required for certain types of files (.text, .rdata, .rsrc, etc).
    c. Click the "**Filter**" tab and review the various options for specific settings if you wanted to change the characters, length, or set specific parameters for your searches. Often you may want to reduce the default 5 character length for strings depending on an IOC.
    d. In **Stage 3: Essentials** area, check the "**MUST contain these**" checkbox and enter **Open**. Now return to the Search tab and select **Go** once again. Do you see additional or different functions/system calls than when using strings? This is another example of how important using more than one tool can be during a forensic or incident response case.

```
00000000004D  00000040004D   0  !This program cannot be run in DOS mode.
000000007126  000000407F26   0  OpenClipboard
000000007282  000000408082   0  SHFileOperationA
000000007396  000000408196   0  RegOpenKeyExA
0000000078CC  0000004092CC   0  OpenProcessToken
000000021030  00000045B630   0  <?xml version="1.0" encoding="UTF-8" standalone="yes"?><assembly xmlns="urn:schemas-microsoft-com:asm.v1" manifestVersion="1.0"><assemblyIdentity version="1.0.0.0" processorArchitecture="X86" name="Nullsoft.NSIS.exehead" type="win32"/><description>Nullsoft Install System v2.46</description><dependency><dependentAssembly><assemblyIdentity type="win32" name="Microsoft.Windows.Common-Controls" version="6.0.0.0" processorArchitecture="X86" publicKeyToken="6595b64144ccf1df" language="*" /></dependentAssembly></dependency><trustInfo xmlns="urn:schemas-microsoft-com:asm.v3"><security><requestedPrivileges><requestedExecutionLevel level="requireAdministrator" uiAccess="false"/></requestedPrivileges></security></trustInfo><compatibility xmlns="urn:schemas-microsoft-com:compatibility.v1"><application><supportedOS Id="{35138b9a-5d96-4fbd-8e2d-a2440225f93a}"/><supportedOS Id="{e2011457-1546-43c5-a5fe-008deee3d3f0}"/></application></compatibility></assembly>
00000000004D  00000040004D   0  !This program cannot be run in DOS mode.
000000007126  000000407F26   0  OpenClipboard
000000007282  000000408082   0  SHFileOperationA
```

000000007396 000000408196 0 RegOpenKeyExA
0000000078CC 0000004092CC 0 OpenProcessToken
000000021030 00000045B630 0 <?xml version="1.0" encoding="UTF-8" standalone="yes"?><assembly xmlns="urn:schemas-microsoft-com:asm.v1" manifestVersion="1.0"><assemblyIdentity version="1.0.0.0" processorArchitecture="X86" name="Nullsoft.NSIS.exehead" type="win32"/><description>Nullsoft Install System v2.46</description><dependency><dependentAssembly><assemblyIdentity type="win32" name="Microsoft.Windows.Common-Controls" version="6.0.0.0" processorArchitecture="X86" publicKeyToken="6595b64144ccf1df" language="*" /></dependentAssembly></dependency><trustInfo xmlns="urn:schemas-microsoft-com:asm.v3"><security><requestedPrivileges><requestedExecutionLevel level="requireAdministrator" uiAccess="false"/></requestedPrivileges></security></trustInfo><compatibility xmlns="urn:schemas-microsoft-com:compatibility.v1"><application><supportedOS Id="{35138b9a-5d96-4fbd-8e2d-a2440225f93a}"/><supportedOS Id="{e2011457-1546-43c5-a5fe-008deee3d3f0}"/></application></compatibility></assembly>

5. **CFF Explorer** (https://ntcore.com/?page_id=388) is a suite of PE editor tools.
    a. Open the Cat_And_Dog_Screensaver.exe in CFF Explorer
    b. Click the File: icon at the top left of the application to see dependencies.
        i. What is the Company Name that made the application?

            ScreenSaverGift.com

        ii. When was the file created?

            Saturday 31 March 2018, 10:42:15

        iii. What is the file version?

            1.0.0.0

    c. While in CFF Explorer, click the "Import Directory" folder on the left side of the screen.
        i. What DLLs are imported when this application is executed?

            KERNEL.32.dll (59 functions)
            USER32.dll (62 functions)
            GDI32.dll (8 functions)
            SHELL32.dll (6 functions)
            ADVAPI32.dll (9 functions)
            COMCTL32.dll (4 functions)
            Ole32.dll (4 functions)
            VERSION.dll (3 functions)

        ii. Review some of the module names and some of their functions.

            Ex. Functions under KERNEL.32.dll: CompareFileTime, SearchPathA, GetShortPathNameA, etc.

6. Open **PEiD** to review PE information
    a. Import the Cat_And_Dog_Screensaver.exe file.
    b. Review the Multi Scan, Task Viewer, Options, and About buttons.
    c. Review the three > buttons in the middle of the console.
    d. Click the >>> >>> button and then the three "-" icons. Is this program packed?

Entropy: 6.29 (Not Packed)
EP Check: Not Packed
Fast Check: Not Packed

Students can also perform dynamic malware analysis and reverse engineering of malware, but these two skills are outside the scope of this introductory course. Additional analysis may identify IOCs with greater fidelity or even determine the initial detections were in fact false positives.

Deliverable: Answer the following questions based on the commands used above.

1. Using the findstr command on Windows, how many strings containing "Open" can be identified in the Cat_And_Dog_Screensaver.exe? What are the names?

   Using the fintstr command, the following 3 strings containing "Open" can be found: OpenClipboard, RegOpenKeyExA, and OpenProcessToken.

2. When searching the file Cat_And_Dog_Screensaver.exe for strings containing the word "Open" with BinText, were there any additional functions or system calls were identified? If so, what were they?

   Yes, in addition to the 3 strings from Part 1 the following string was found: SHFileOperationA

3. Open the file in CFF Explorer. What is the Company Name that made the application? When was the file created? What is the file version?

   Company Name: ScreenSaverGift.com
   FileCreated: Saturday 31 March 2018, 10:42:15
   FileVersion: 1.0.0.0

4. While in CFF Explorer, identify the 8 imported modules. Look up one of the KERNEL32.DLL function names and explain its function.

   One of the Function Names, CreateProcessA, seems to be creating a new process within the application. This can be dangerous since the application can create a new process which can be potentionally hidden from the user of the application.

5. Using PEiD, can you determine if the application is packed or not?

   Within the PEiD tool, the .exe Extra Information states that the application is not packed with the following:
   Entropy: 6.29 (Not Packed)

EP Check: Not Packed
Fast Check: Not Packed

MEMORY ANALYSIS (Using Volatility and Mandiant Redline)

Step 1: Download the standalone volatility application from the site below. There are applications for macOS, Linux and Windows; however, the instructor has **only** tested this new 64-bit version on Windows 10. Unzip the directory and run the application from this directory. Also install Mandiant Redline from Blackboard, if you have not done so for Assignment 8, or if you've uninstalled the application.

Volatility Standalone Application: https://www.volatilityfoundation.org/26

Step 2: Students will review a memory capture to review the basic volatility plugins to identify suspicious processes, incident response and forensic data.

Make sure you unzip the folder and use this as your working directory or copy the contents to a folder named, "Assignment 10." You must run all commands from this directory and moving your memory evidence files to this same directory might make running commands easier. As with any command-line utility, the proper syntax will be crucial. Unzip the .ZIP file titled "memory-images" as well. You could run these commands on the ~17GB memdump.mem file from Lab 4 as well, but these commands take a significant amount of time.

1. Run the command ***volatility_2.6_win64_standalone.exe –h*** to review the tool's options and supported plugin commands.

Run the commands and answer the questions below; you may use screenshots, if desired. Note, once you run the first command, it's easier to press the up arrow and replace the plugin name each time to reduce the amount of typing and

2. Run the command ***volatility_2.6_win64_standalone.exe –f \<Full Directory Path>\xp-laptop-2005-07-04-1430.img imageinfo.*** The ***imageinfo*** plugin is used to identify potential profiles and is the only plugin where specifying the OS profile is not required***.*** Note the profile and image local date and time to answer the questions below.

   Suggested Profiles: WinXPSP2x86, WinXPSP3x86 (Instantiated with WinXPSP2x86)
   Image Data and Time: 2005-07-04 18:30:32 UTC

3. Run the command ***volatility_2.6_win64_standalone.exe –f \<Full Directory Path>\xp-laptop-2005-07-04-1430.img --profile=WinXPSP2x86 pslist.*** The plist plugin lists all the running processes, including their respective Process ID (PID), Parent Process ID (PPID), number of threads, handles, and the process start and exit times (when applicable). Answer the questions below about the processes.

4. Run the command ***volatility_2.6_win64_standalone.exe –f \<Full Directory Path>\xp-laptop-2005-07-04-1430.img --profile=WinXPSP2x86 psscan.*** In the course lecture, we discussed the difference between listing and scanning and the fact that scanning is a

brute force method for identifying processes that the OS may no longer be aware of. This will often identify processes that have exited, or are in the process of exiting, and those that are hidden. You'll also notice that scanning does not place the processes in chronological order, which the plist plugin does since it's walking the list in memory.

5. Run the command ***volatility_2.6_win64_standalone.exe –f \<Full Directory Path>\xp-laptop-2005-07-04-1430.img --profile=WinXPSP2x86 pstree.*** This plugin will identify a tree list view of processes and their children. This allows for an easier view of processes and the ability to identify rogue processes.

6. Run the command ***volatility_2.6_win64_standalone.exe –f \<Full Directory Path>\xp-laptop-2005-07-04-1430.img --profile=WinXPSP2x86 psxview.*** This plugin shows a summary view of several plugins and allows you to easily compare the pslist and psscan results to identify processes that could easily be identified during the scan, but not the listing. This may indicate exited processes, or even hidden processes that should be further investigated.

7. Run the command ***volatility_2.6_win64_standalone.exe –f \<Full Directory Path>\xp-laptop-2005-07-04-1430.img --profile=WinXPSP2x86 connections.*** This plugin prints a list of active, open TCP connections (XP/2003 only).

8. Run the command ***volatility_2.6_win64_standalone.exe –f \<Full Directory Path>\xp-laptop-2005-07-04-1430.img --profile=WinXPSP2x86 connscan.*** This plugin scans memory for TCP connections , including those closed and unlinked (XP/2003 only). What external IPs and ports is the PluckUpdater.ex process connecting to?

   PluckUpdater.ex PID: 368
   Local: 192.168.2.7:1145 Remote: 170.224.8.51:80
   Local: 192.168.2.7:1130 Remote: 216.239.115.140:80
   Local 192.168.2.7:1144 Remote: 170.224.8.51:80

9. Run the command ***volatility_2.6_win64_standalone.exe –f \<Full Directory Path>\xp-laptop-2005-07-04-1430.img --profile=WinXPSP2x86 sockets.*** This plugin prints a list of active, available sockets (any protocol) (XP/2003 only).

10. Run the command ***volatility_2.6_win64_standalone.exe –f \<Full Directory Path>\xp-laptop-2005-07-04-1430.img --profile=WinXPSP2x86 sockscan.*** This plugin scans memory for sockets, including those closed or unlinked (any protocol) (XP/2003 only).

11. Run the command ***volatility_2.6_win64_standalone.exe –f \<Full Directory Path>\xp-laptop-2005-07-04-1430.img --profile=WinXPSP2x86 ldrmodules.*** This plugin identifies unlinked DLLs and image binaries. If the output is too long, you can redirect the standard output to a text file by adding "> ldrmodule.txt at the end of the command. Most processes will run from \Windows\System32\, but not all of them. This is one way to narrow down the malicious files. Name three files by their full paths that are not running from this directory that may need to be investigated.

    \dd\UnicodeRelease\getopt.dll

    \WINDOWS\WinSxS\x86_Microsoft.Windows.Common-Controls_6595b64144ccf1df_6.0.2600.2180_x-ww_a84f1ff9\comctl32.dll

    \Program Files\Common Files\Logitech\Scrolling\LGMSGHK.DLL

12. Run the command ***volatility_2.6_win64_standalone.exe –f \<Full Directory Path>\xp-laptop-2005-07-04-1430.img --profile=WinXPSP2x86 malfind > malfind.txt.*** This plugin scans process memory sections looking for indications of code injection. Identified sections can be extracted for further analysis.

Deliverable: Answer the following questions based on the commands used above.

1. What operating system was this memory image collected from, according to the "Profile?"

    WinXPSP2x86, WinXPSP3x86 (Instantiated with WinXPSP2x86)

2. What was the local date/time when this memory was collected?

    Image Data and Time: 2005-07-04 18:30:32 UTC

3. NUM 3
    a. When was the dd.exe process started?

        2005-07-04 18:30:32

    b. What was the process ID?

        PID: 3300

c. What parent process initiated dd.exe? (hint: use the plist plugin)

PPID: 3256

4. What processes could you identify that had previously closed and were not in the plist output? (hint: use the psscan plugin)

By using the pscview plugin (duplicates)

PluckUpdater.ex PID: 368
mqtgsvc.exe PID: 712
tcpsvcs.exe PID: 1548
explorer.exe PID: 2392

5. What is the PID of the PluckUpdater.ex process? What external IPs and ports is this process connected to, or has connected recently?

PluckUpdater.exe PID: 368
Local: 192.168.2.7:1145 Remote: 170.224.8.51:80
Local: 192.168.2.7:1130 Remote: 216.239.115.140:80
Local 192.168.2.7:1144 Remote: 170.224.8.51:80

6. Name three programs, listed by their full path, that might warrant investigation (hint: ldrmodules plugin).

\dd\UnicodeRelease\getopt.dll

\WINDOWS\WinSxS\x86_Microsoft.Windows.Common-Controls_6595b64144ccf1df_6.0.2600.2180_x-ww_a84f1ff9\comctl32.dll

\Program Files\Common Files\Logitech\Scrolling\LGMSGHK.DLL

Step 3: Students will use Mandiant Redline on the same memory image as above to see the differences and automation offered by a GUI tool. Often students begin conducting memory analysis with Redline and once they become more familiar with memory structures, will use a more advanced tool such as volatility. As we've seen above, using the command-line tool can be slower, take a greater amount of skill, and a keen eye.

1. Start Mandiant Redline.

2. Under the **Analyze Data** section, select "**From a Saved Memory File**," to collect data from a saved memory file.

3. In the **Configuration** box, select **Browse** to navigate to the same xp-laptop-2005-07-04-1430.img memory file. Select **Next**.

4. Select "**Edit you script**" under the **Review Script Configuration** section.

5. There are fewer options when reviewing a memory sample than when creating an IOC collector that was done during Assignment 8. Select all available options and click **OK**. The Redline tool will then begin analysis.

6. Select the "**Host**" tab under the **Analysis Data** section. Select the top-level "**Processes**."
    a. What version of the JRE was running on the system?

        Path: C:\Program Files\Java\jre\1.5.0_02\bin\jusched.exe
        Version: 1.5.0_02

    b. What dd command was used to collect memory?

        D:\dd\UnicodeRelease\ dd.exe if=\\.\PhysicalMemory of=c:\xp-2005-07-04-1430.img conv=noerror

7. Click the "**Hierarchical Processes**" section. Notice that this display shows a tree-like structure like the volatility pstree plugin.
    a. What is the parent process and PPID of the DirectCD.exe process?

        PPID: 2392
        PID: 2456

8. Click the "Timeline" section and review the order of processes and their start times when the memory sample was collected. This allows you to understand the order of events on a system when looking at memory.
    a. What antivirus software was running on the system?

        DefWatch.exe, Rtvscan.exe

Deliverable: Answer the following questions related to the Mandiant Redline output.

1. What version of the Java Runtime Environment (JRE) was running?

Path: C:\Program Files\Java\jre\1.5.0_02\bin\jusched.exe
Version: 1.5.0_02

2. What was the dd command used to collect this memory sample?

   D:\dd\UnicodeRelease\ dd.exe if=\\.\PhysicalMemory of=c:\xp-2005-07-04-1430.img conv=noerror

3. What is the parent process and PPID of the DirectCD.exe process?

   PPID: 2392
   PID: 2456

4. What antivirus solution was running on the system?

   DefWatch.exe, Rtvscan.exe

Step 4: Students will review a memory sample of a system infected with the Zeus Trojan, or Zbot, to review a real-world example of a system infection without infecting their systems. This sample was collected with Mandiant Redline.

Zeus/Zbot provides a great opportunity to see code injection in progress. This persistent malware was designed to steal credentials for FTP, POP3, online credentials and credit card data. Zeus is usually successful at being persistent, but because it needs to run, it also must identified at some point.

Some characteristics (IOCs) are standard, but there are many variants, one of which does the following:

- Copies itself to %system32%\sdra64.exe
- Injects code into winlogon.exe OR explorer.exe
  - Additionally injects code into every process except csrss and smss
- Auto-start path: HKLM\Software\Microsoft\Windows NT\winlogon\userinit
- Creates local.ds & user.ds in %system32%\lowsec\
- Retrieves files from a command and control server
- Mutant: _AVIRA_
- Hooks over 50 system APIs

Complete the following activity:

1. Load the zeus.mans file into Redline by selecting the "**Open Previous Analysis**" option under **Analyze Data**. Navigate to the .mans file and select **Open**.

2. When prompted to upgrade the session, select "**Yes**."

3. You'll see three tabs under Analysis Data: Host (this specific evidence data), IOC Reports (if you've opted to create an IOC collector), and Not Collected (which shows those options that were not selected during this specific collection.

4. Select the Processes > Handles > Show Mutant Handles option. Search for the mutant "_AVIRA_" and identify the full mutant handle name, the associated process that may be injected and the PID.
    a. What process might be injected?

        Handle Name: _AVIRA_2109
        Process Injected: winlogon.exe
        PID: 632

        Handle Name: _AVIRA_2108
        Process Injected: cvchost.exe
        PID: 856

5. Click "Strings" and search for "AVIRA." What different mutants can you find for the Zeus bot? Name three injected processes? Now students can begin to see the importance of indicators of compromise (IOCs).

    Handle Names: _AVIRA_2110, 2101, 2108, 2109, 21099

    Process Name: wuauclt.exe
    PID: 1732

    Process Name: svchost.exe
    PID: 1028

    Process Name: TPAutoConnSvc.exe
    PID 1968

6. Select **Memory Sections > Injected Memory Sections**." Once loaded, do you see winlogon.exe as possibly injected? What is the process ID? What is the protection type that identifies this process as potentially injected?

Yes

PID: 632,

Protection Type: EXECUTE_WRITECOPY ImageMap Inherit

Deliverable: Answer the following questions:

1. Were you able to find a mutant handle in the memory image? If so, what is the mutant handle name, associated process, and PID?

   Handle Name: _AVIRA_2109
   Process Injected: winlogon.exe
   PID: 632

   Handle Name: _AVIRA_2108
   Process Injected: cvchost.exe
   PID: 856

2. What process might be injected?

   winlogon.exe and cvchost.exe

3. Name three potentially injected processed using the strings function?

   Handle Names: _AVIRA_2110, 2101, 2108, 2109, 21099

   Process Name: wuauclt.exe
   PID: 1732

   Process Name: svchost.exe
   PID: 1028

   Process Name: TPAutoConnSvc.exe
   PID 1968

4. Does winlogon.exe appear to be injected? What is the protection type?

   Yes

   PID: 632,

Protection Type: EXECUTE_WRITECOPY ImageMap Inherit

5. Lookup winlogon.exe. What is the function of this process?

   Functions: services.exe, svchost.exe and TPAutoConnSvc.exe

***Please submit the final assignment as a single .PDF and any applicable reports as a .ZIP file.**
**Screenshots may also be added to this document when appropriate.