Lab4
Issued: June 25th, 2021
Due: July 1st, 2021 11:59PM

Name: Kamalesh Ram Chandran Govindaraj

Overview: This lab will have you run a piece of malware multiple times to use IDA's debugger. The lab's focus will include understanding what the malware does to the machine itself as well as the origin of its network traffic. The malware we will examine is MyDoom. This is an interesting malware worm. It has been in existence since 2004 and is *still* found in the wild. Quite remarkable for a piece of malware to still be viable after 16 years and still very close to its original form. I have posted this article from 2019 that details the malware on Piazza. Give it a read.

MyDoom_Still_Active_in_2019.pdf

For this lab you will need the Windows Flare VM and the Kali VM connected together in host only mode with no connection to the Internet. The malware itself is found here on Piazza:

Hw4.zip

The zip files password is 'infected' Copy and unzip it on the Flare VMs desktop.

Question 1
The original EXE was packed with UPX. I did you a favor and unpacked it. What is the command that I needed to run to unpack this file given what packed it?

Score: / 2 pts
UPX -d -o filename_unpacked.exe filename.exe

Question 2
Open the unpacked malware file in IDA Educational - remember to run IDA as administrator. Get familiar with the file itself - as always start with the Imports and Exports and also look at the Strings (you will need to open this subview) to understand some of the capabilities. Look at the Strings view and let's focus on the DLL's listed in there. Many are also listed in the imports section but if you scan the list you'll find at least one listed there that is not. Look at the Import view and focus on the function `LoadLibraryA`. Look at the Microsoft documentation to become familiar with this function. Then, do the following:

1. Find the three DLLs that are loaded by the malware at runtime. The `LoadLibraryA` should lead you to them. Note - two of the DLLs are in the comments (thanks IDA!) and the other you will need to figure out - an ASCII table might help. You do not need to run the malware to determine this information. Also, provide a brief description of what

each DLL does - for this you may want to limit your Google search with "site:microsoft.com".

Score: / 6 pts

DLL Name	Description
dnsapi.dll	It contains the Windows functions which Windows and applications call to perform Domain Name Service Client tasks. It resides in "C:\Windows\System32"
iphlpapi.dll	It is a module containing the functions used by the Windows IP Helper API.
wininet.dll	It is a module that contains Internet-related functions used by Windows applications.

2. Let's run the malware. Be sure that you have Wireshark running in Ubuntu (started as Administrator!) to collect the network traffic. Be sure to open **View -> ToolBars -> DebuggerCommands** so it is easily accessible in IDA. Click the run button. IDA will come up with a window that says "Do you want to do this?" If you are sure you are in Host-only mode and connected to Kail - then yes, do it. IDA might be slightly unresponsive or have delayed response time as the malware starts - be patient! Let the malware run for about two minutes and then turn it off. Go to the Kali machine and turn off Wireshark. Then, answer the following:

   Set a filter on WireShark to filter by TCP so you only see the TCP traffic. Note, the filter only works if you type lowercase **tcp**. You will note that there are TCP connect requests coming from the Windows Flare machine. Collect the distinct IP numbers in the Destination column when the source IP address is the Windows Flare VM. Provide a list of the collected IP addresses below:

   Score: / 6 pts

   10.15.1.201
   158.185.142.164
   68.72.122.180
   172.16.1.232
   15.237.16.21
   68.88.130.90
   68.88.239.126
   16.155.50.163
   172.27.172.146
   156.23.129.237
   10.112.9.66
   1.1.0.71
   10.248.217.120
   16.83.192.27

Set a filter on WireShark to filter by DNS so you only see the DNS traffic. Again, the filter needs to be lowercase **dns**. Based on the information you read about MyDoom in the PDF, give a paragraph summary of the types of URLs trying to be resolved and what MyDoom is likely doing beneath.

Score: / 6 pts

These are the few URLs trying to be resolved when the malware is run yahoo.com, mail.ru, alumni.caltech.edu, 126.com, gzip.org, collabora.com, alice.it, …. Mydoom worm consists of an email attachment that searches for address book files and sends multiple copies of itself as email attachments.

Question 3

Question 2 shows the malware is establishing a DNS lookup. Let's figure out where it is doing this. Based on what you know about all the DLLs this malware uses, list out the DLL you believe is responsible for the DNS lookup. You only need to give the name of the DLL.

Score: / 2 pts

dnsapi.dll

With that identified, list the name of the function that is doing the DNS lookup or resolution. Here, I want two items, the name of the function and the memory address location where the function name is pushed on the stack. For memory address location I mean the address like .text0080000:

Score: / 6 pts

Name of function	sub_80399E
Memory address where pushed on stack	.text:008039A2

If you want, you may place a breakpoint on the suspected DNS lookup call and run the malware application. Verify that when you hit the DNS resolution call and step past it, Wireshark shows the URL that it requested be looked up.

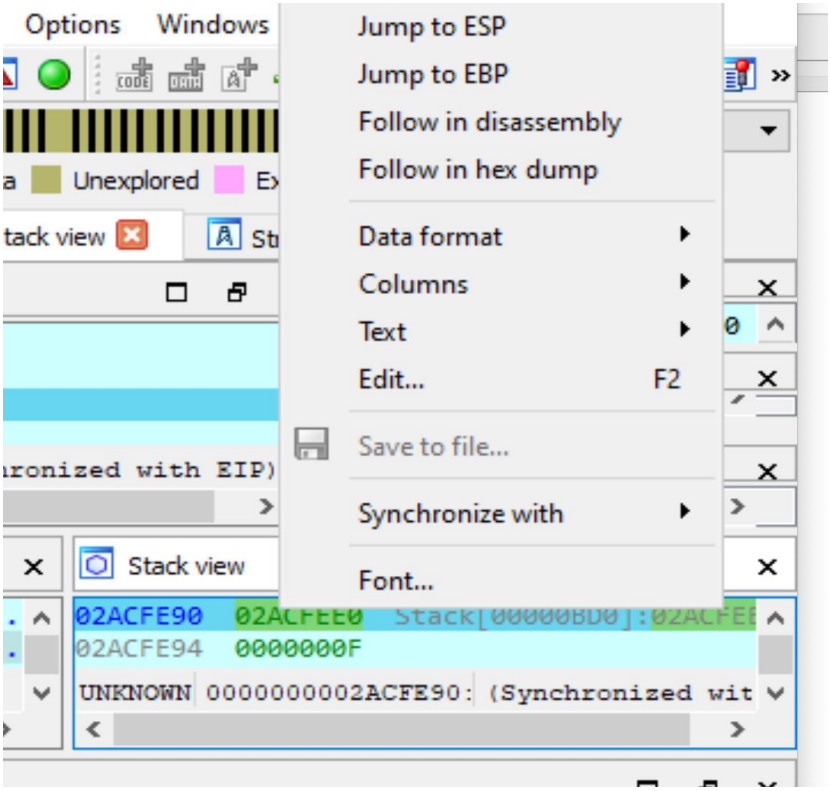
Last question here. URLs are provided for DNS lookup to the function you identified in the prior question. What is the memory address of the *parameter* passed to this function that holds the URL? That is, if the URL to be resolved is "something.mail.com", what is the memory address of that parameter when it is pushed on the stack?

Score: / 4 pts

Memory address of URL parameter passed to resolution function	.text:0008039BF

Question 4

The malware makes multiple calls to the imported function `CreateFileA`. Place a breakpoint at location 00802A94 (that literal address) for one of these calls. Run the malware and answer the following questions. Note, you may need to stop the malware, reset the breakpoint to different locations before the memory address above, and re-run the malware to answer all these questions. Also, you can set a breakpoint and identify the *value* of items pushed on the stack. I did not go over this in class (and will on Tuesday!) but it's relatively easy. After an item you are interested in is pushed on the stack it will be highlighted in the **Stack View**. Right click on the value in the middle column of highlighted, just written, stack item and click **Follow in hex dump**. The image below shows you what I mean. It will take you to the hex and, if it's an ASCII value written, you may simply read the value.



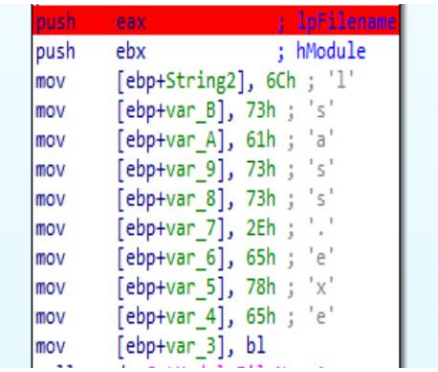
Score: / 4 pts

What file is the malware attempting to create? Please include the full path.

The malware is creating a file lsass.exe, the path of the file is C:\Windows\lsass.exe

Score: / 4 pts
The name of the file is not in the list of string resources. Where did the malware get the name of the file from?

The filename is stored as characters in variables and moved one by one. The the function GetModuleFileName is called



Score: / 4 pts
Where does the malware get the file from that it creates at the location? Give at least a sentence description that shows its use of the Windows API to get the path.

GetTempPathA function (fileapi.h) - Retrieves the path of the directory designated for temporary files.
GetWindowsDirectoryA function (sysinfoapi.h) - Retrieves the path of the Windows directory.