**Windows 8 Malware Lab**

**Disclaimer:**

*To fully understand what is happening with the debugging process, it is recommended that you using Microsofts resources to view the passed arguments to each function covered. This will give you a deeper understanding of what you need to look for in your future analysis and could potentially help lead straight to the meat of the analysis.*

**Intial Setup**

**Python**

* To interact with the malware with the provided script file, you will need to install python3 on the your host machines.
  + <https://www.python.org/downloads/>

**Virtual Box**

* You will first need to import the ova file into Virtualbox. To do this simply open Virtualbox and select File > Import Appliance. From this menu dialog simply select the folder to open the downloaded ova file.

**Login Information**

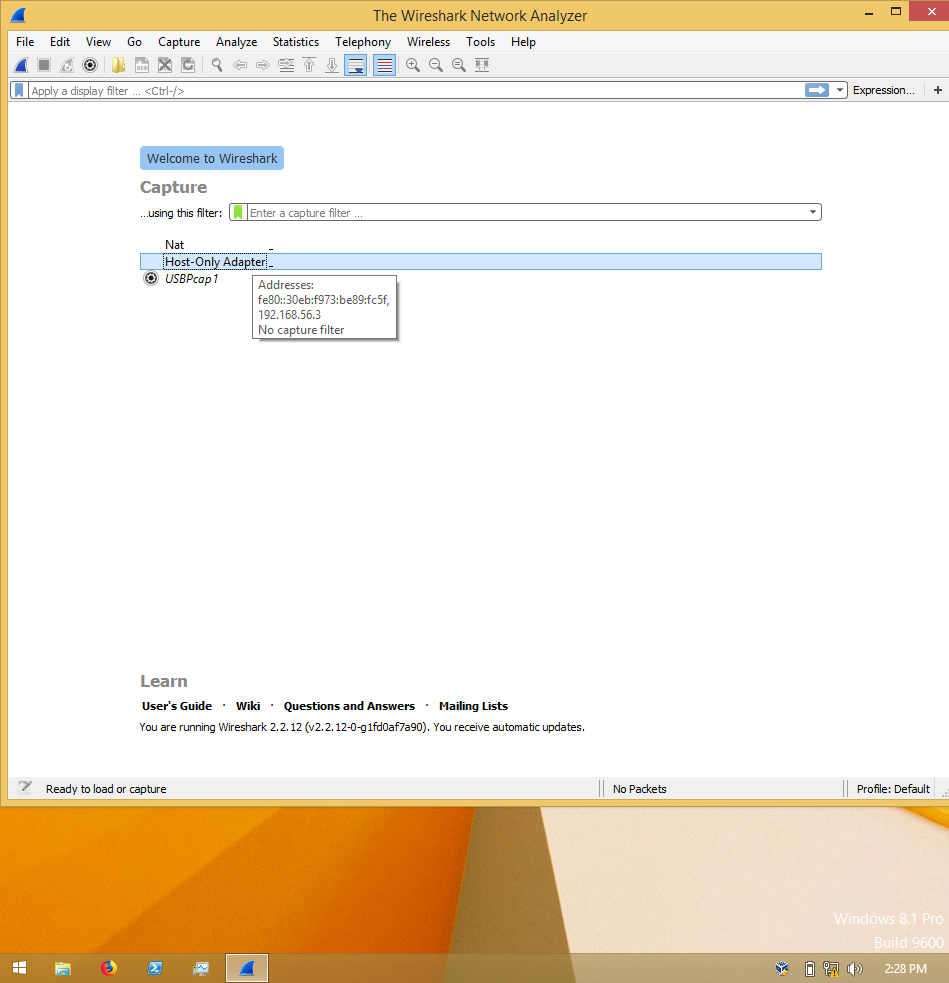
Password: malware

**Getting Started**

**Background**: You have been handed an image of a workstation or server and are told that the local adminstrator account has been been flagged for logging in a non-work hours. The IT team noticed this, and issued a password reset but the problem is persistant. They suspect that the machine may been infected with Malware, but they are not sure. It is your job to find out.

**Investigations steps:**

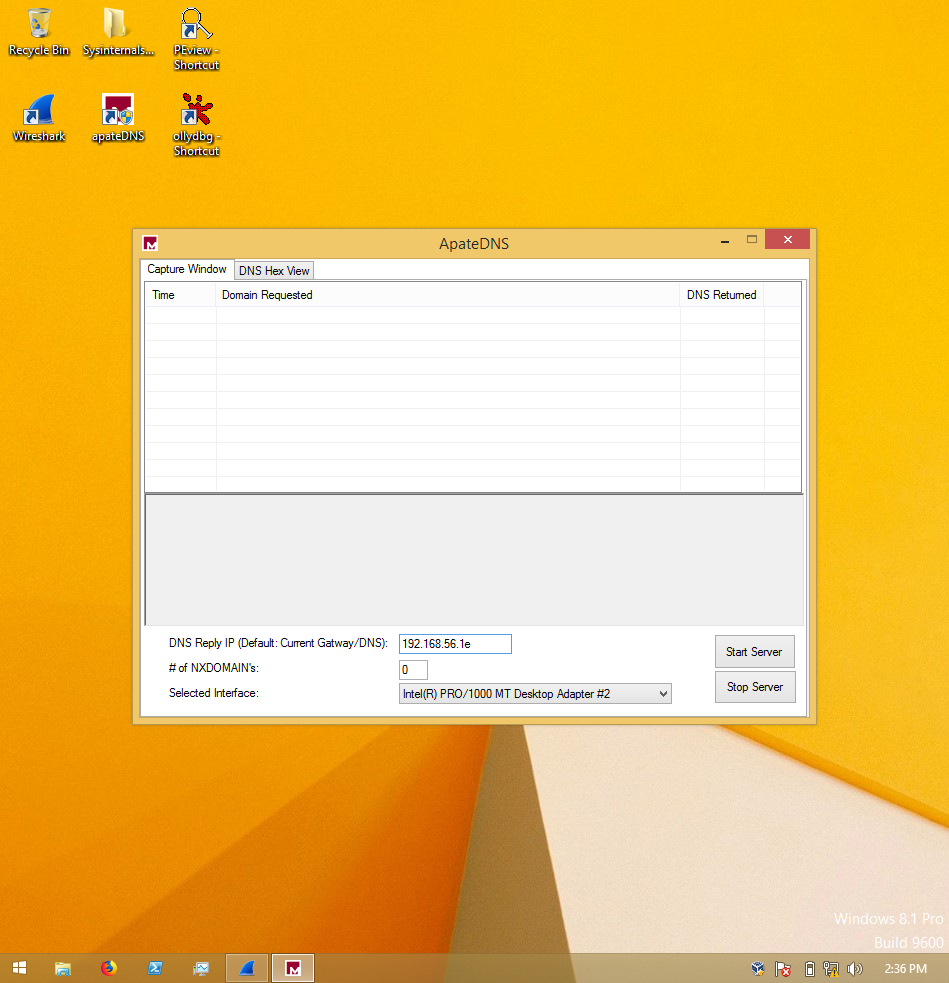
1. Open **Wireshark**
   1. We initially open wireshark to try to capture traffic going through our host adapter by selecting the following:

**Note**: The flow of traffic may be non-existant because we are not connected to the internet and some services cannot resolve the IP address for the services they are querying. In the future you may find it useful to have a network simulator on your host machine to ensure that you can interact with all the network services on the machine.

1. Open **ApateDNS**

To ensure that we are capture all of the traffic from the network processes on the machine, we will utilize a tool called apateDNS that allows us to process DNS requests

* 1. To activate apateDNS we are going to need to enter the IP of our Host Adapter into the field DNS Reply IP and select Start Server.



While wireshark continues to capture traffic we can examine some other options to see if we can find any programming interacting with our machine in a way that it should not be.

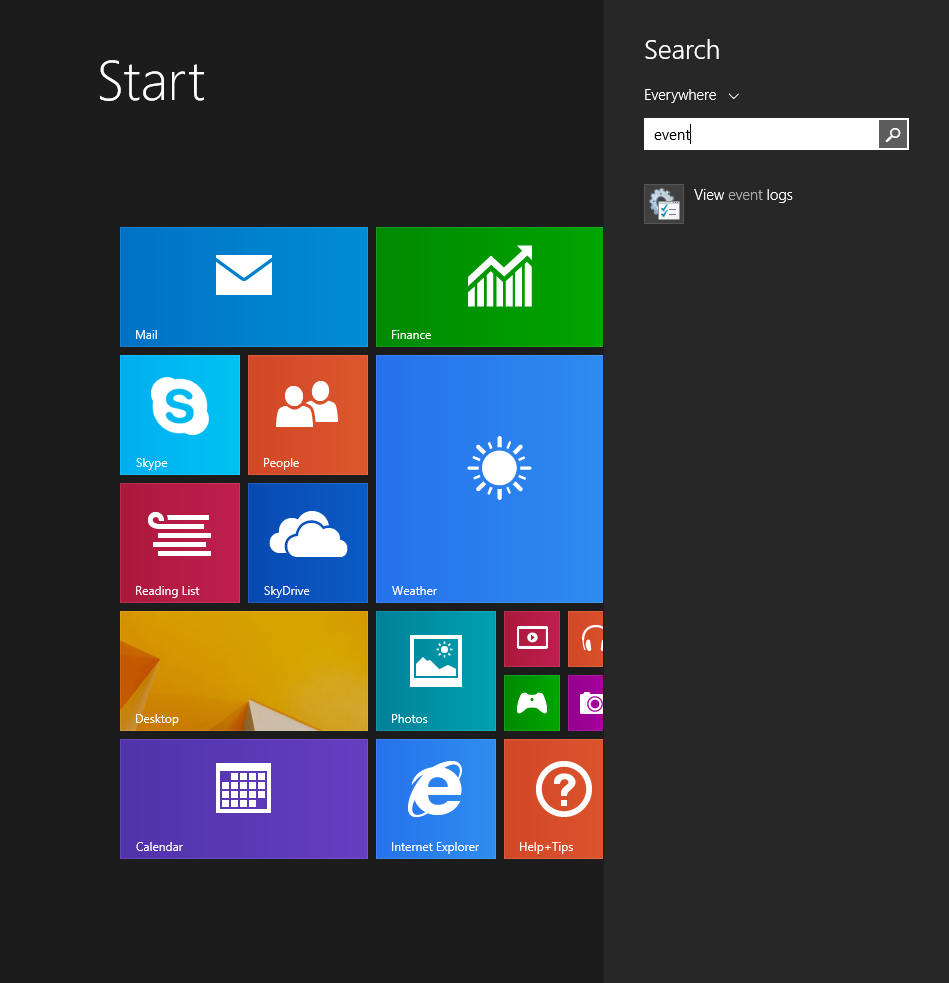
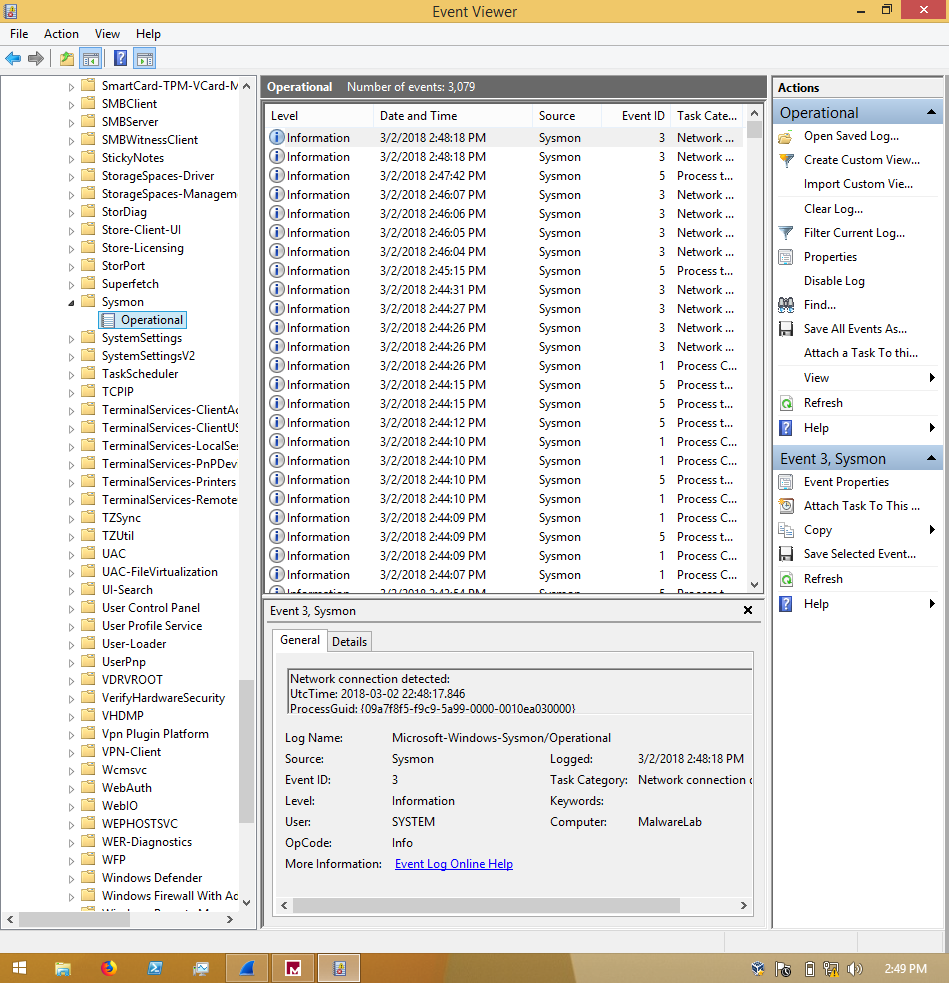
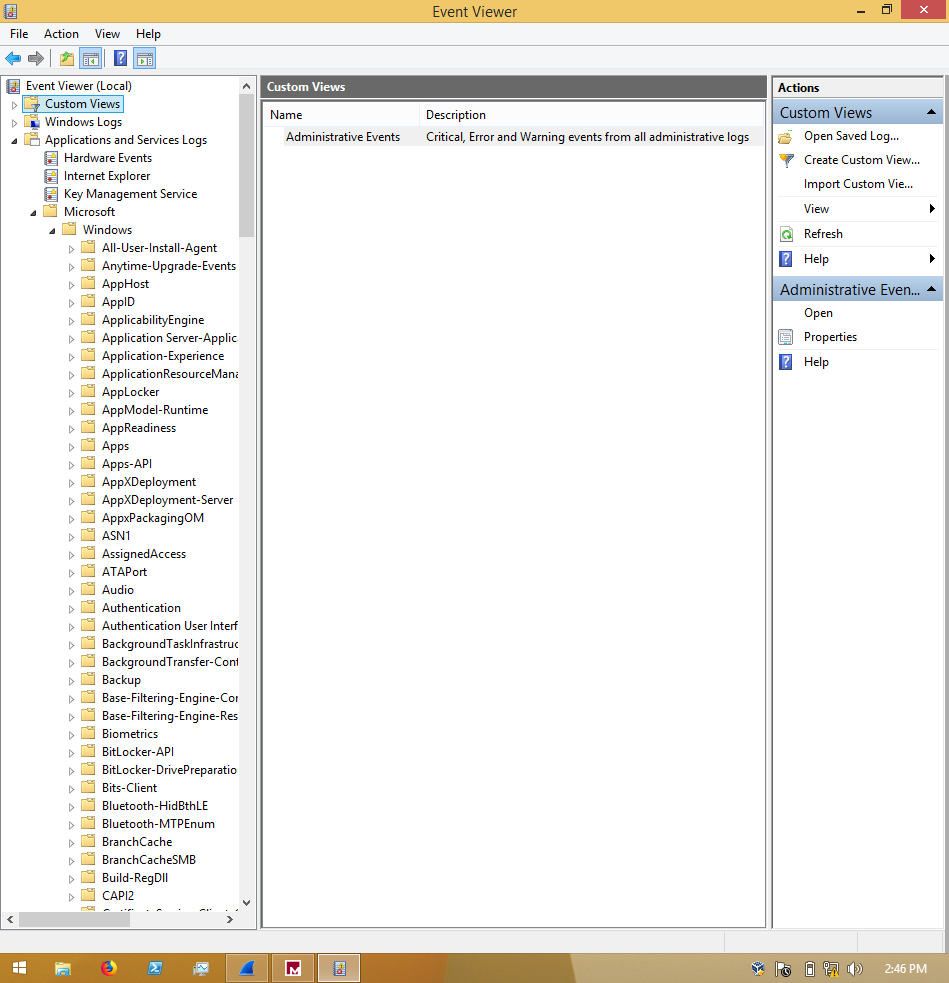
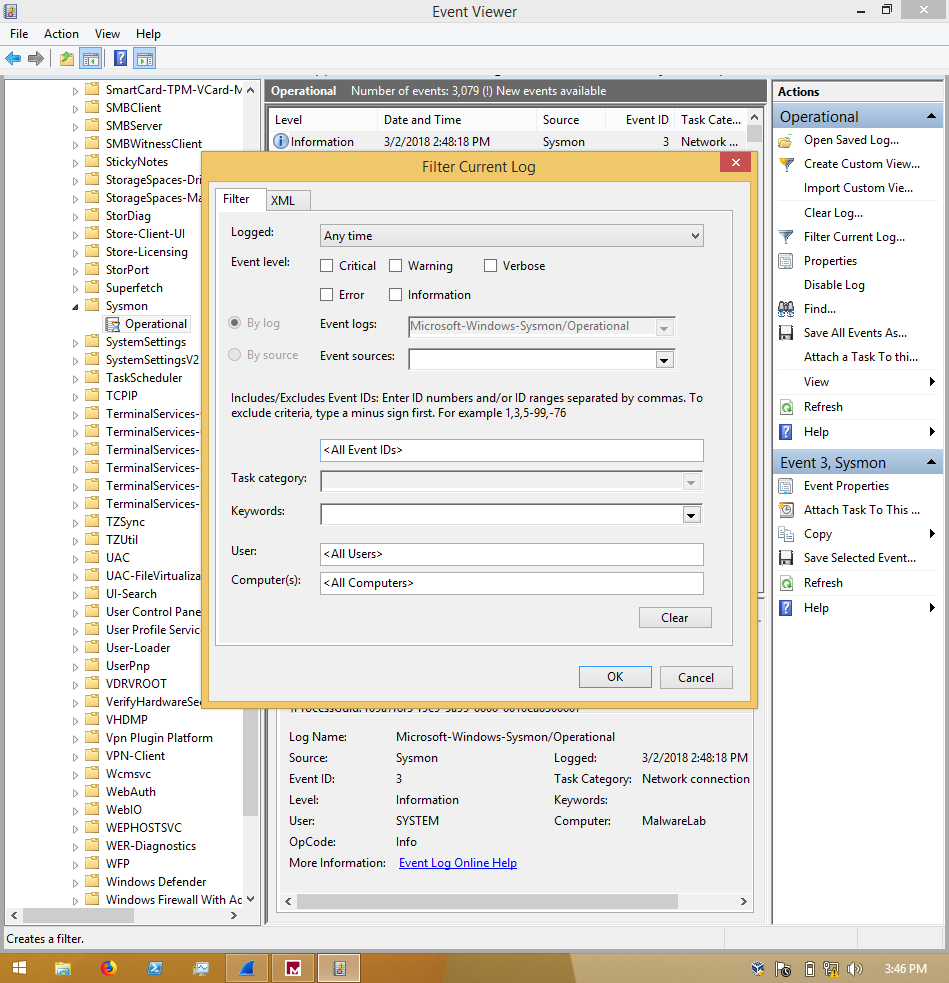
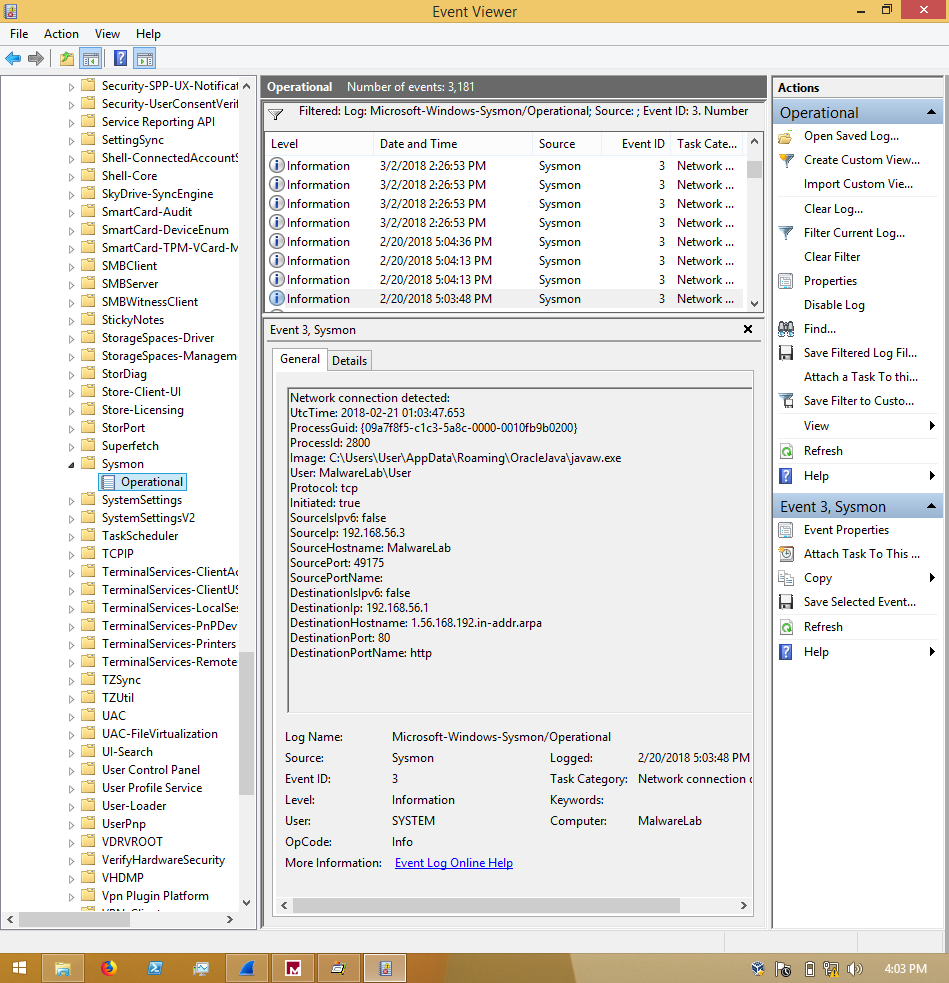
1. Utlities within the **Sysinternals Suite**

**Note**: Using this suite of tools provided by Microsoft, we will mainly focus on using procmon, procexe, and sysmon.

* + 1. Sysmon

**Note**: The install process is not covered, but to install sysmon on your personal machine run the following in your command prompt. You can attach configuration files when installing sysmon but it is not needed to use it’s base logging feature.

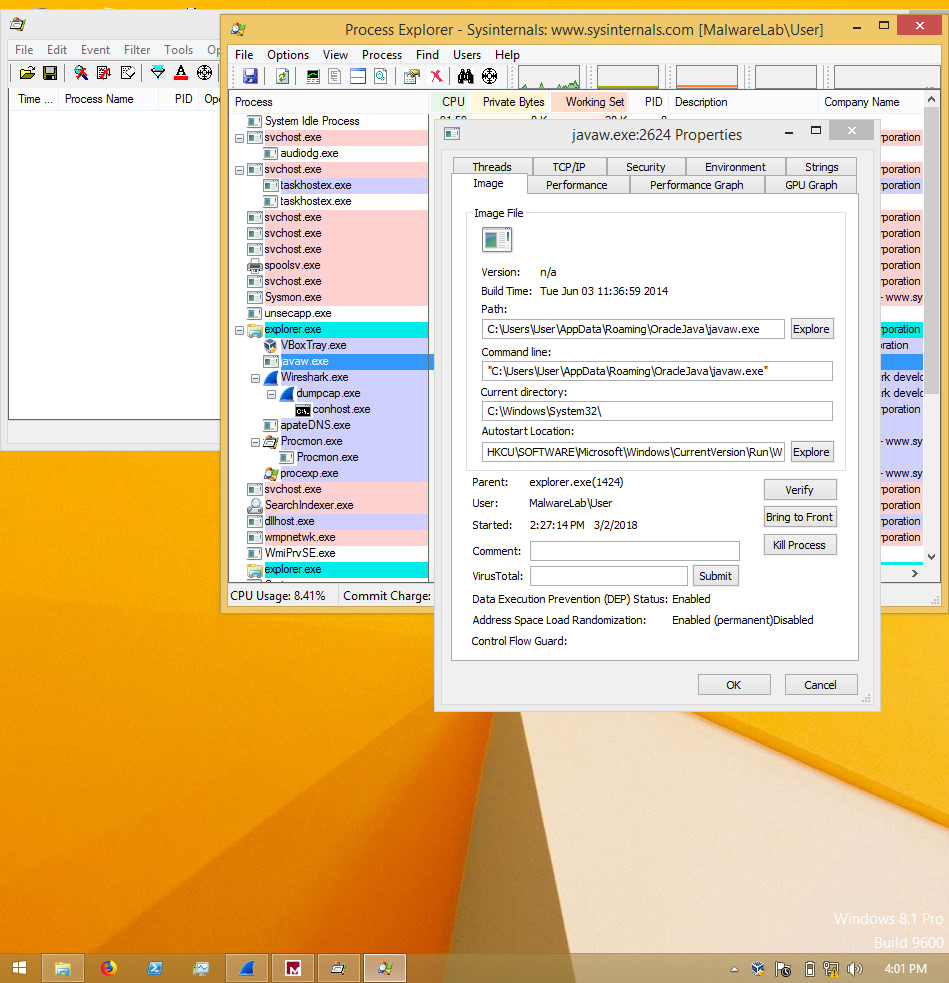
sysmon.exe -i -n

* + - 1. Open Event Viewer
         1. You can search event to find event viewer within Windows 8
         2. To navigate to the sysmon logs, you will follow the path and select the operational file: Applications and Service Logs > Microsoft > Windows > Sysmon > Operational
         3. Within Event Viewer we want to filter the sysmon events to see if any application is making any network requests. We can view this by selecting Filter Current Log.
         4. To filter for networking traffic, we input a 3 into the <All Event Ids> field.
         5. After filtering for network traffic, we can search for the process that is using the network that we did not start.

**Note**: Your process ID may be different

1. Procexp (Process Explorer)
   1. Within Process Explorer we can right click the executable to view its properties. From this menu we find that the executable is running from:

C:\Users\User\AppData\Roaming\OracleJava\javaw.exe

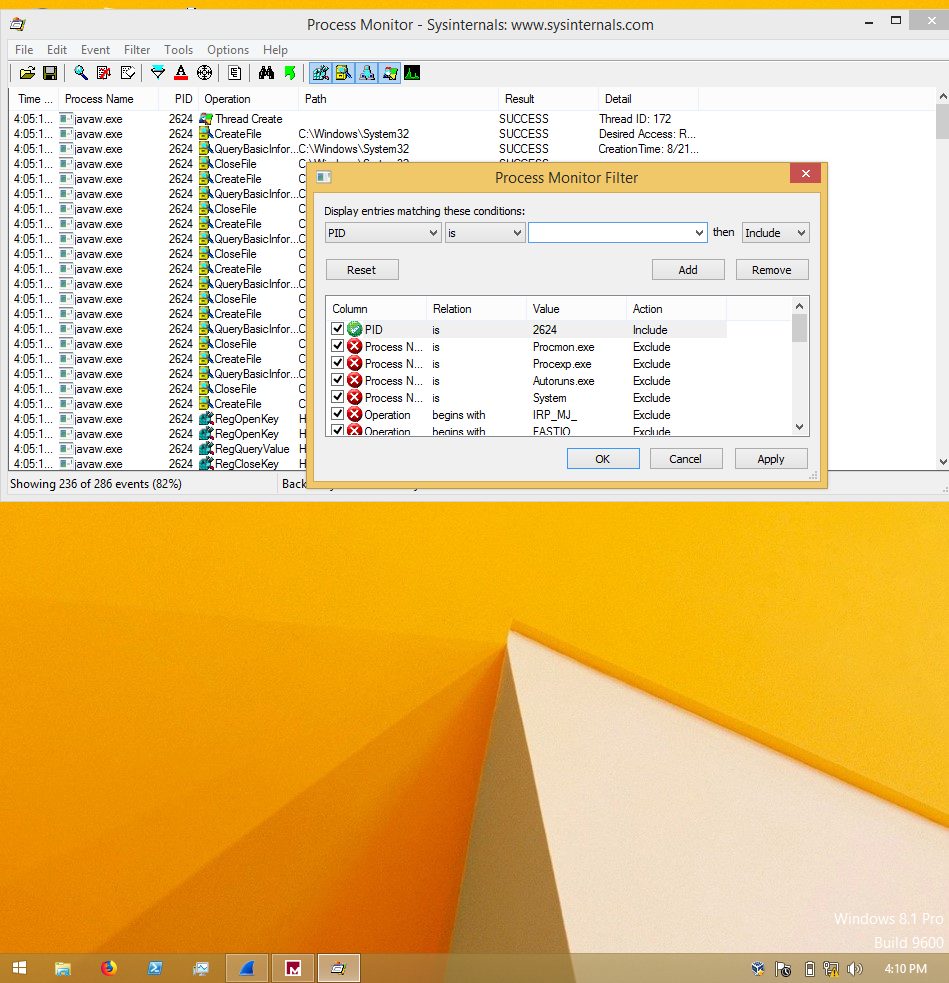
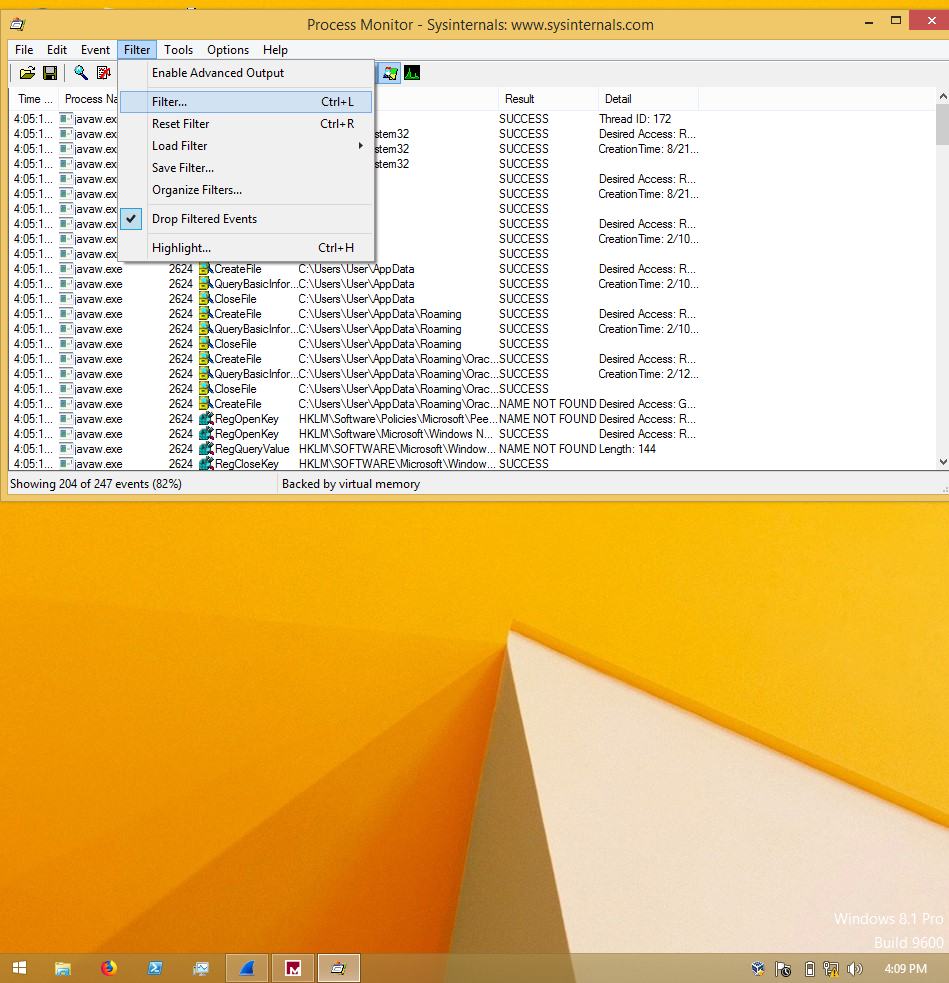
We can select Explore to go into the directory

* 1. Within the directory we find a file named Log

1. Procmon (Process Monitor)

You can ues the information obtained from Process Explore to leverage Process Monitor to find useful information.

* 1. Filter Process Monitor by the PID of the javaw to view the operations performed by the executable

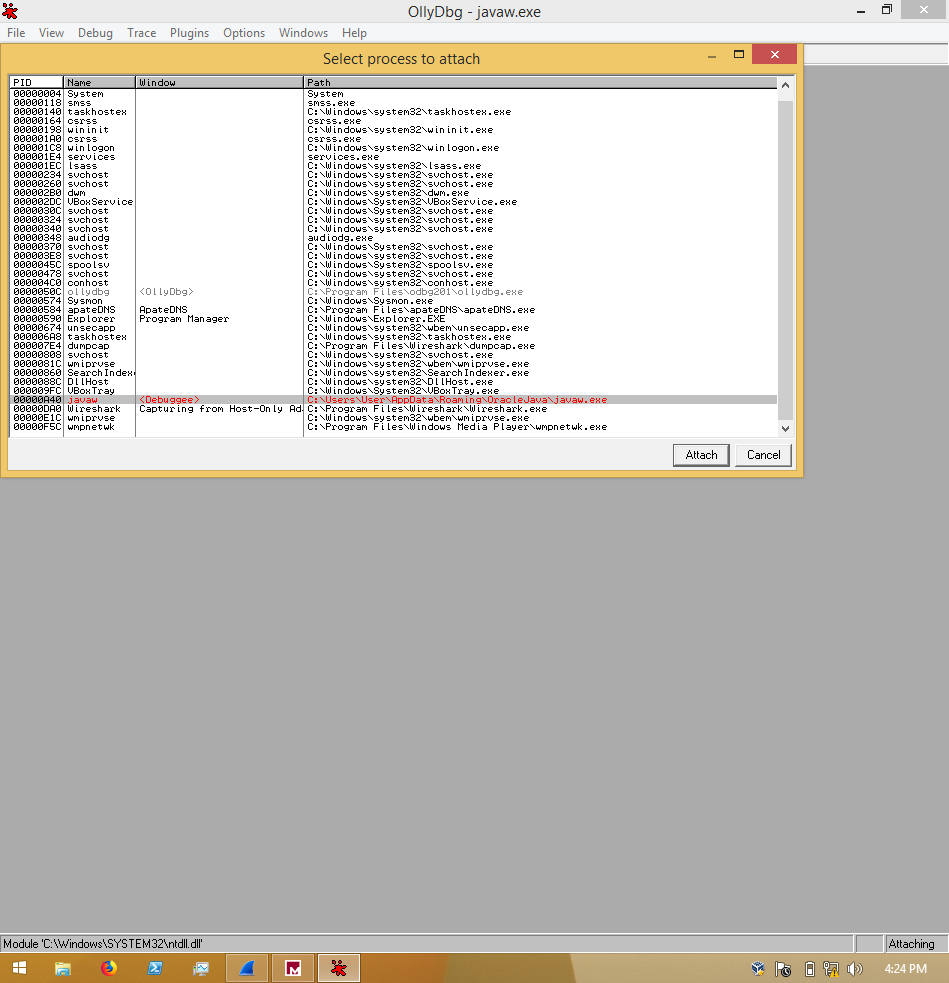
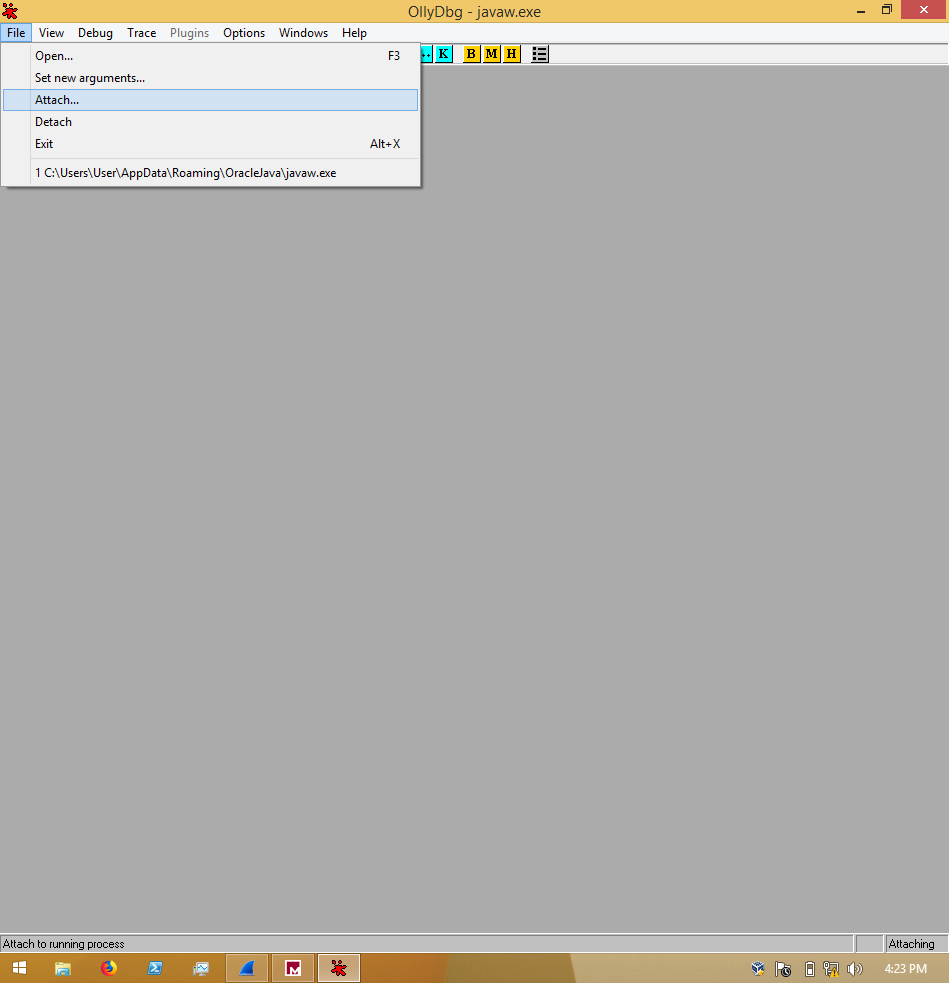
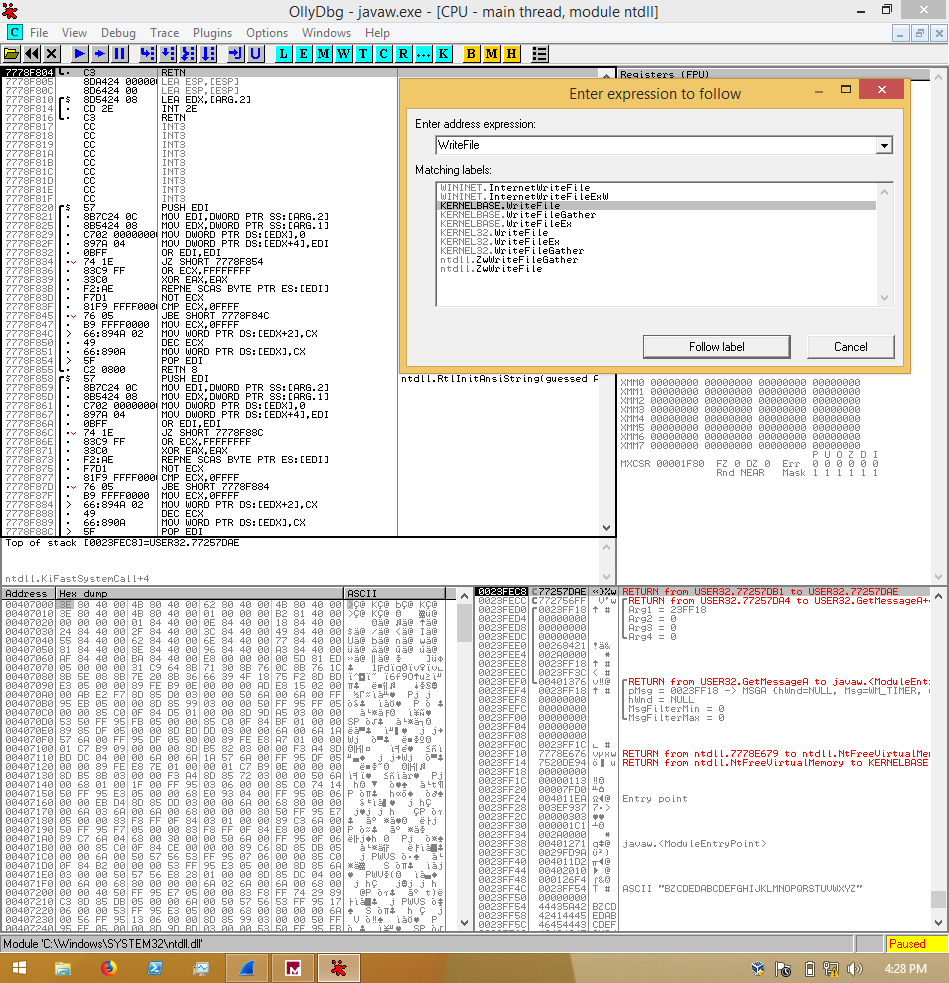
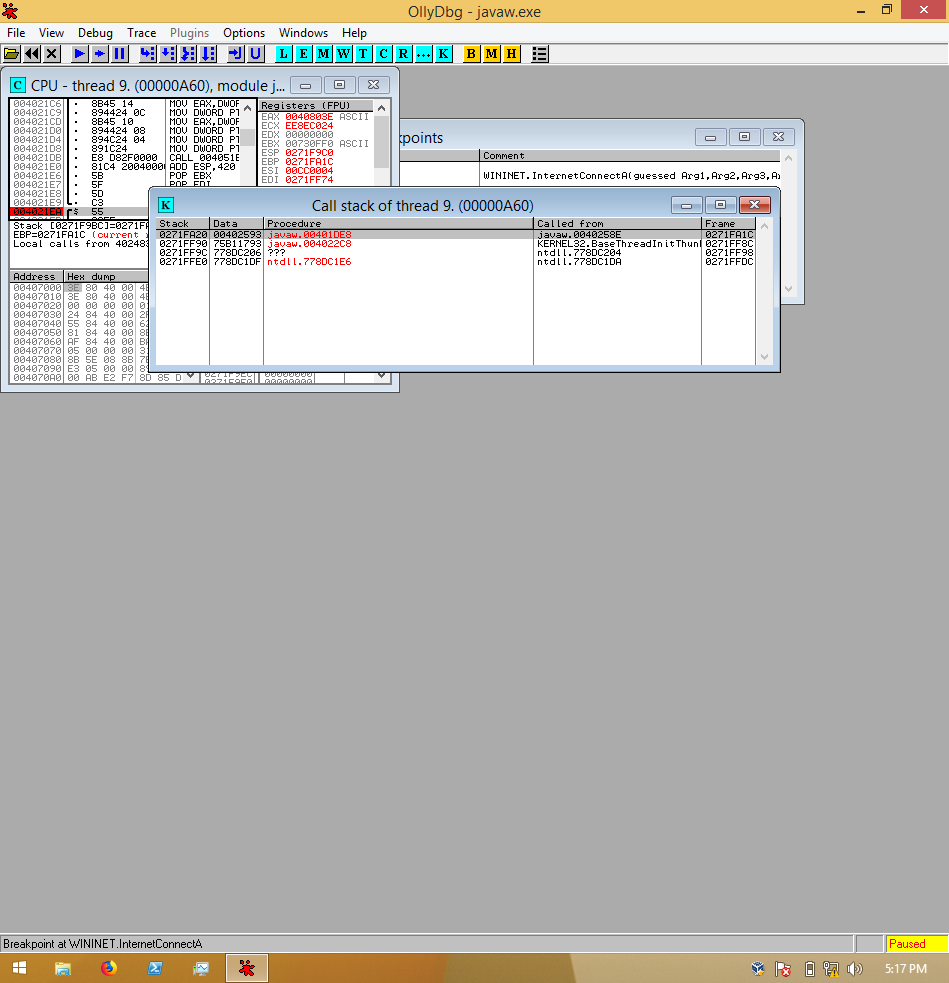
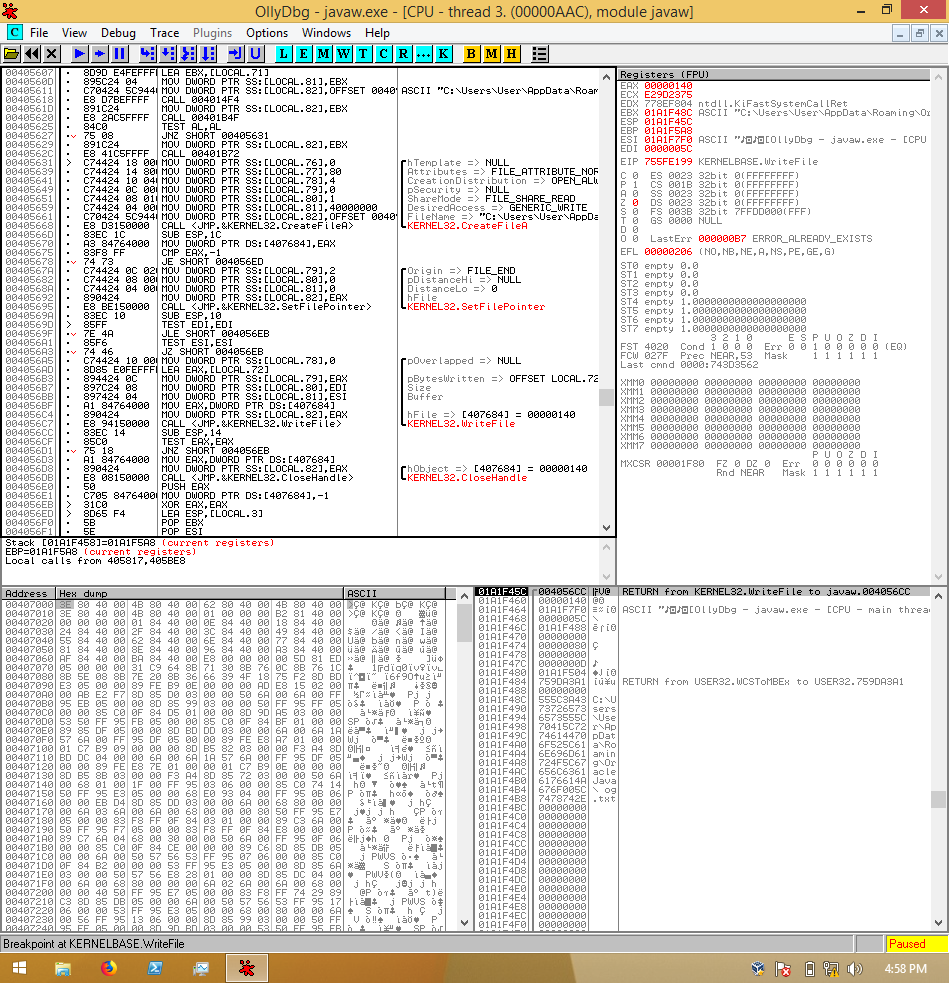


* 1. Within Process Explorer we see that the javaw executable is writing to a file.

**Note**: If Process Explorer is already filtering a PID simply disable it or remove it to see results. You may also want to select the box Drop Filtered Events to save disk space.

1. Open OllyDBG

You will need to attach the executable to OllyDBG to futher understand what the executable is doing.

* 1. To attach the process select File > Attach then select the process which can be seen with its uniquely long path.
  2. Search for where the executable is writing to a file using Ctrl+G to open a prompt to jump to functions or address.
  3. Within this window search WriteFile looking for the kernel base function call within the executable.
  4. To test if the malware is calling this function after select it and set a breakpoint using F2. After you run the process in hopes it hits the breakpoint F9.
  5. You can confirm if the malware is calling the selected function if it hits your break point after you run the process. Step up the call stack to see where within the malware this function is being called. Open the call stack menu with Alt+K
  6. After opening the call stack window we can double click on the address within the called from column in the call stack menu to jump to that location.

**Note**: After double clicking the address we notice two things. The first is that the malware address space is prefixed with 004. The second thing we notice as we start analyzing the process is the malware calls muliple file managing functions ex. CreateFileA, SetFilePointer, etc.

* 1. Remove the break point that we set within the entry of the WriteFile function. Set a new break point at our current landing.

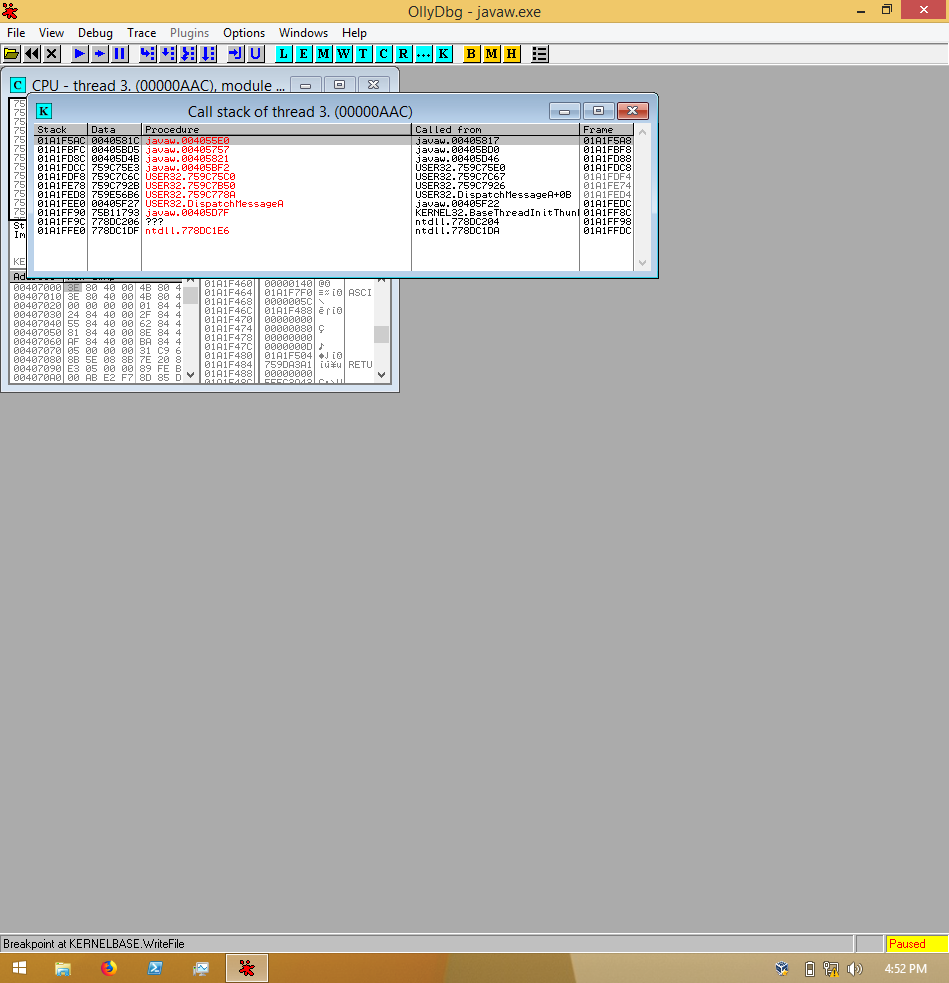
**Note**: To toggle the breakpoint menu you can press Alt+B

* 1. To understand how the process writes to the file we can analyze the following addresses within the malware:

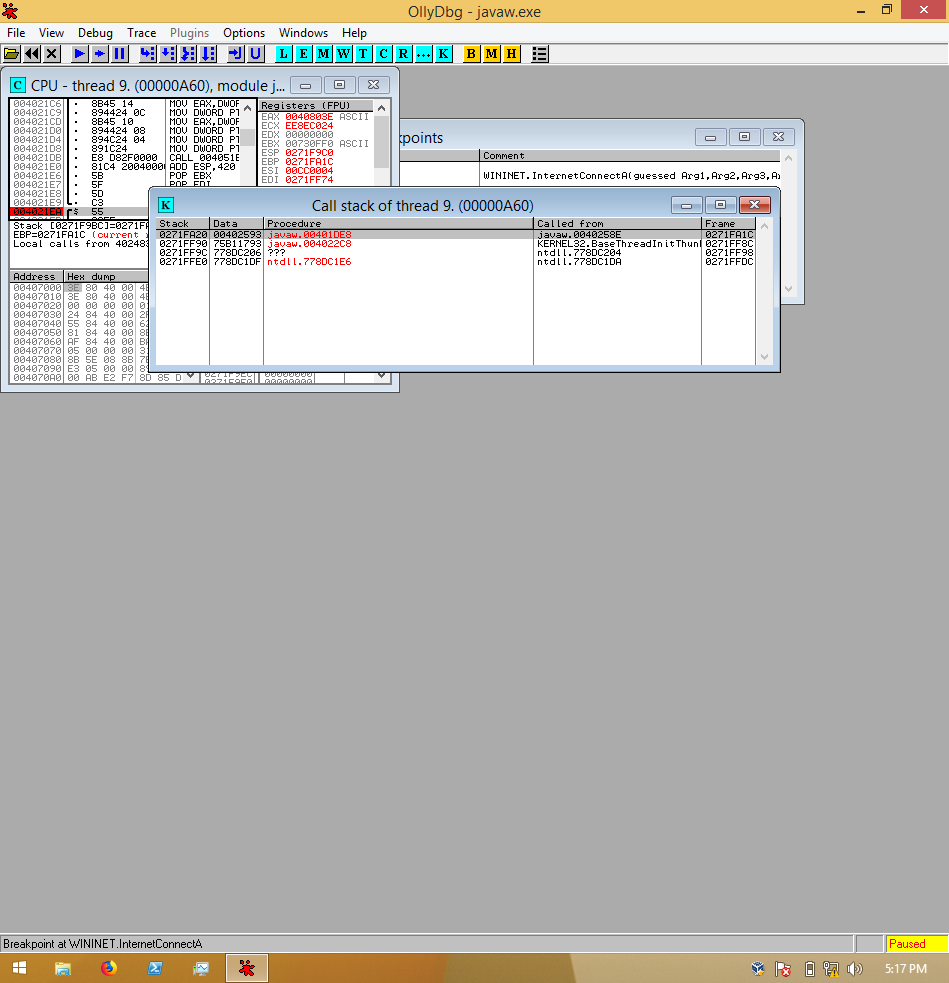
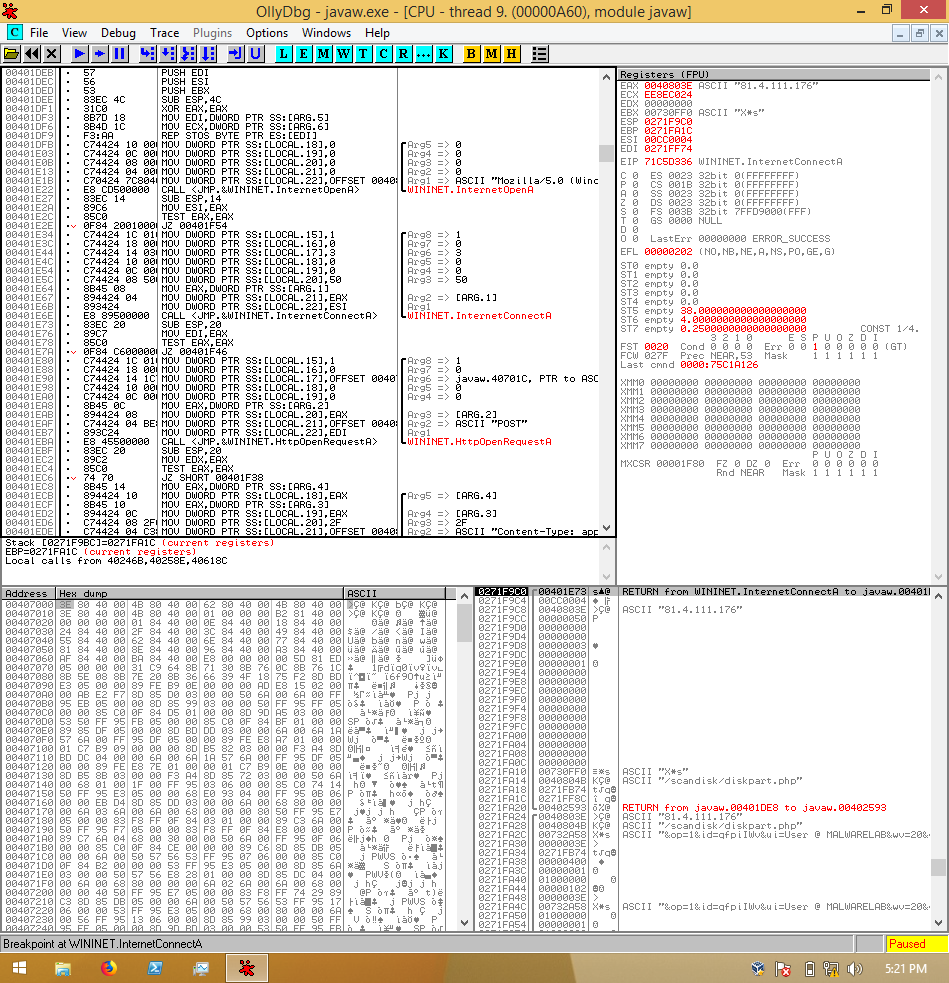
**Note**: I do not go into detail because this step mainly hightlights the mechanics of how the process writes to a file and it is not needed to get a basic understanding of the malware.

* + - Start of file writing thread javaw.004055E0
    - There is a compare to see if the file is created including a set function pointer at javaw.004055F9
    - * If there is not resulting file, then the process checks if a directory is created. If there is no directory created the process calls function javaw.00405631
      * After the directory is created the process continues to create the file at javaw.00405660 by calling KERNEL32.CreateFileA
      * The process then has another check which is at javaw.00405678 which checks if the file pointer is set. If it is not it calls KERNEL32.SetFilePointer at javaw.0040569
    - To understand how the process is able to caputure the keys and convert specifc keys ex. Caps lock to text, you can navigate up the call stack to the calling function which begins at javaw.004405821.
    - The KERNE:32.WriteFile function is called at javaw.004056C7

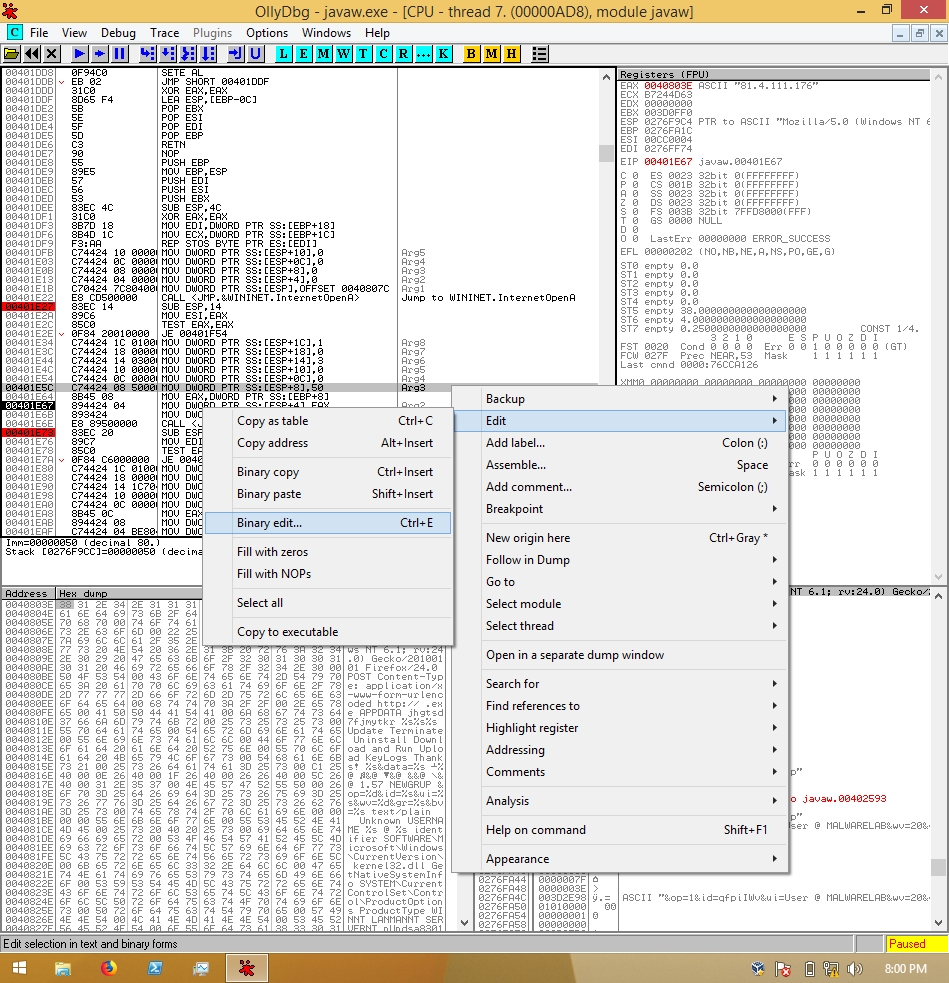
**Note**: With a new understanding of how the executable writes to a file, we can further analyze the attached process to see what it is attempting to do while opening a network connection.

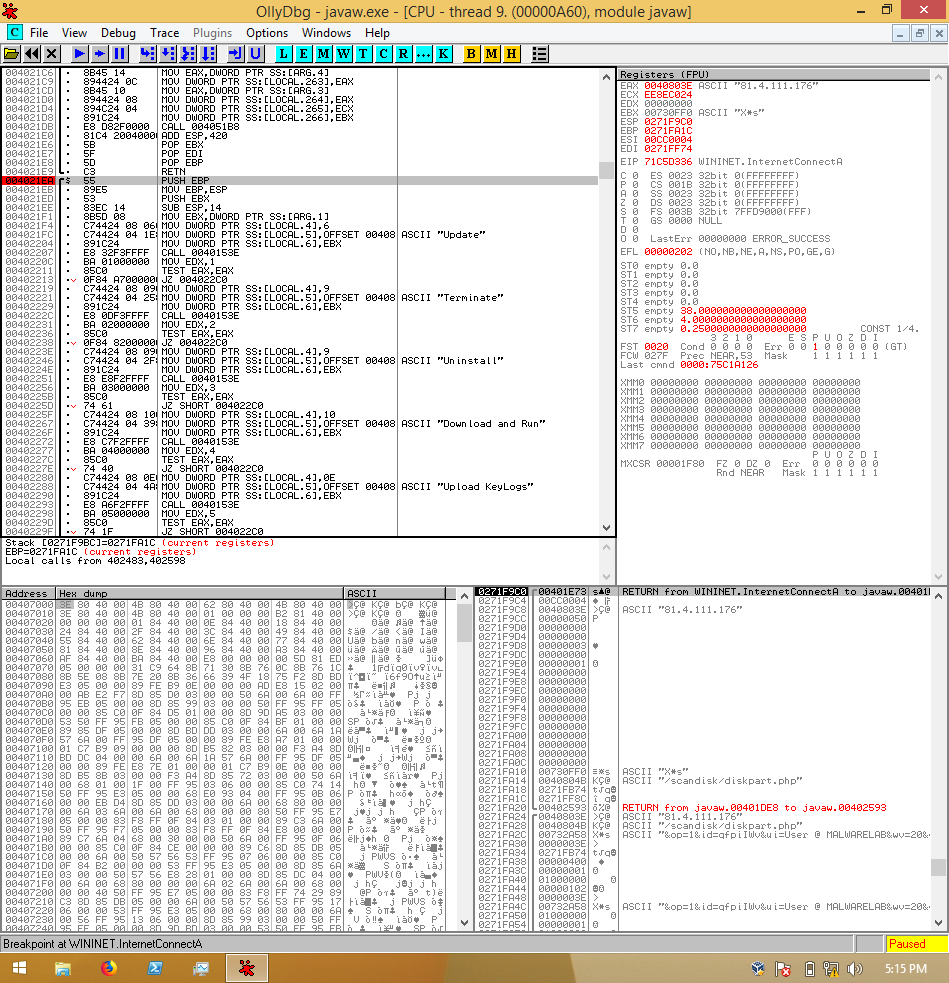
* 1. Using the same window we used to search for the WriteFile function (Ctrl+G), we can search for the InternetOpen function call. Similar to WriteFile, we want to check if the function we choose is called by the executable by setting a break point and running the process.

**Note**: If you have any break points still in the WriteFile section of the executable you are going to want to remove them.

* 1. After the break point is hit, we can navigate up the call stack to the area of code calling this function. In this case we see that two javaw functions are called: javaw.**00401DE8**, which is the calling function of InternetOpen in the Procedure column and javaw.**0040258E**
  2. Within the function located at javaw.**00401DE8,** we see that this function is actually handling the networking interaction.
  3. After InternetOpenA returns we can see that the executable moves the IP address “81.4.11.176” from ESI into EAX.

**Note**: The IP address is stored at 0x0040803E if we need to change it. To change the port number we are going to need to enter the decimal value of our wanted port before javaw.00401E5C is executed. Within that line we see the decimal value 50, for port 80, being moved onto the stack. If we change this value we will successfully move the port of interaction.



* 1. After javaw.**00401DE8** finishes it execution the execution within the calling function javaw.**00401DE8,** leads to another function call javaw.**004021EA**. Inside of this function we witness a string compare of the passed arguement from InternetReadFile.
  2. At this point, we can uses this strings to see if passing them to the malware insights any thing different from the malware. To do this we are going to set a breakpoint after the call to java.**004021EA.**

**Note:** If you had to restart the machine at any point in your analysis make sure that you start apateDNS for this to work.

