

Lab 09 - Memory, CPU and Network Monitoring (Windows)

01 - Objectives

- How to **find the root cause if the system runs out of memory** with TaskManager, Windows Performance Recorder, Windows Performance Analyzer, Visual Studio, VMMap.
- **Monitoring the CPU usage** with TaskManager, Windows Performance Recorder, Windows Performance Analyzer.
- **Check the amount of network traffic generated by processes** with TaskManager, Windows Performance Recorder, Microsoft Network Monitoring, Wireshark.

02 - Memory, CPU and Network Monitoring (Windows)

Introduction

- The Windows operating system comes with plenty of built-in tools to analyze resource usage. The most prominent one is probably the Windows Task Manager, as it highlights resource usage of individual processes, and gives admins and users options to kill any misbehaving ones.
- The Performance Monitor and Resource Monitor are two additional tools that admins and experienced Windows users may use to analyze performance or resources related issues on Windows PCs.

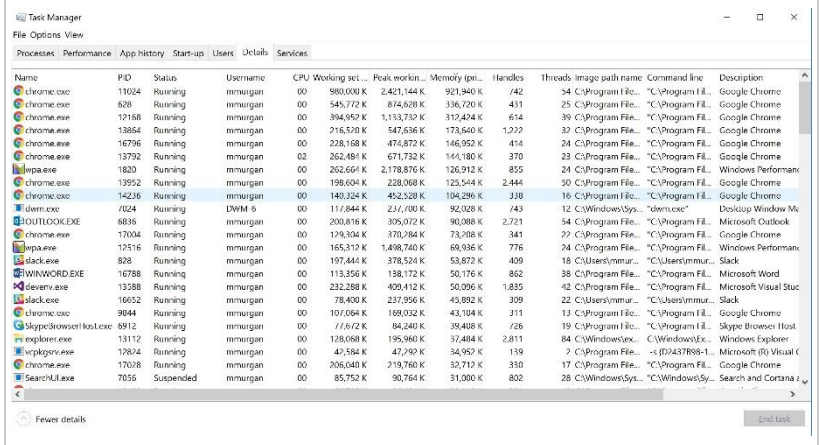
03 - Tutorials/Exercises

The password for log2.zip and build.zip is: *parola*

Exercise 01. [30p] RAM Monitoring

Task Manager

The memory usage by processes is dynamic, thus it is possible to have memory allocation spikes. **Task Manager** shows the amount of memory allocated to a process. To see this information, check the **Peak Working Set** value in the Details tab. It can be noticed that sometimes **Peak Working Set** can be significantly larger than **Working Set**.



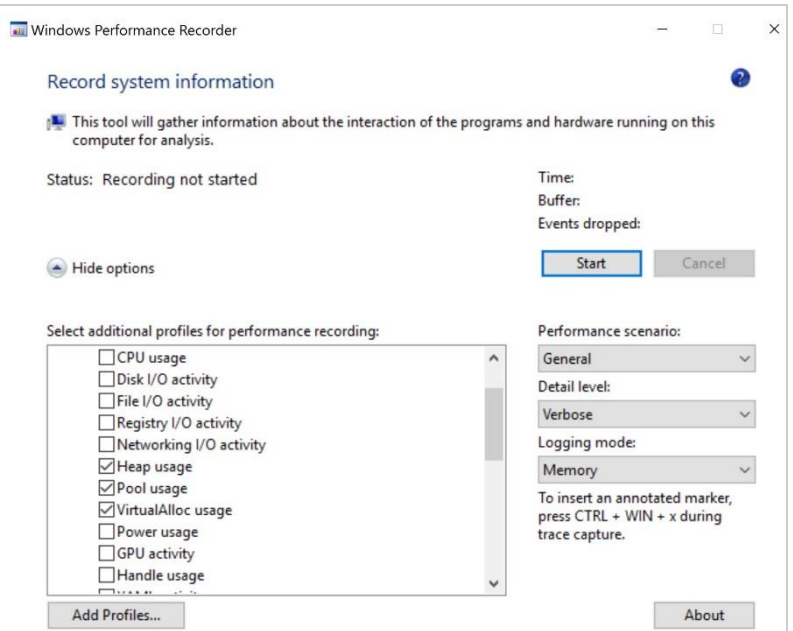
Name	PID	Status	Username	CPU	Working set...	Peak workin...	Memory (pri...	Handles	Threads	Image path name	Command line	Description
chrome.exe	11024	Running	mmurgan	00	980,000 K	2,421,144 K	921,940 K	/42	24	C:\Program F...	"C:\Program F...	Google Chrome
chrome.exe	528	Running	mmurgan	00	545,772 K	874,628 K	336,720 K	431	25	C:\Program F...	"C:\Program F...	Google Chrome
chrome.exe	12168	Running	mmurgan	00	394,952 K	1,133,732 K	312,424 K	614	39	C:\Program F...	"C:\Program F...	Google Chrome
chrome.exe	13884	Running	mmurgan	00	216,520 K	547,636 K	173,840 K	1,222	32	C:\Program F...	"C:\Program F...	Google Chrome
chrome.exe	16796	Running	mmurgan	00	238,168 K	476,832 K	146,952 K	414	24	C:\Program F...	"C:\Program F...	Google Chrome
chrome.exe	13792	Running	mmurgan	02	262,484 K	671,732 K	141,180 K	370	23	C:\Program F...	"C:\Program F...	Google Chrome
lropa.exe	1820	Running	mmurgan	00	262,664 K	2,178,876 K	126,912 K	855	24	C:\Program F...	"C:\Program F...	Windows Perform...
chrome.exe	13952	Running	mmurgan	00	198,604 K	228,068 K	123,544 K	2,444	50	C:\Program F...	"C:\Program F...	Google Chrome
chrome.exe	14236	Running	mmurgan	00	140,324 K	402,528 K	104,296 K	338	16	C:\Program F...	"C:\Program F...	Google Chrome
dem.exe	7024	Running	DNM 9	00	117,844 K	231,000 K	92,028 K	743	12	C:\Windows\Sys...	"C:\Windows\Sys...	Desktop Window M...
OUTLOOK.EXE	8836	Running	mmurgan	00	200,816 K	305,072 K	90,088 K	2,721	54	C:\Program F...	"C:\Program F...	Microsoft Outlook
chrome.exe	17004	Running	mmurgan	00	129,304 K	370,284 K	73,268 K	341	22	C:\Program F...	"C:\Program F...	Google Chrome
lropa.exe	12516	Running	mmurgan	00	165,312 K	1,498,740 K	69,936 K	776	24	C:\Program F...	"C:\Program F...	Windows Perform...
slack.exe	828	Running	mmurgan	00	197,444 K	378,524 K	53,872 K	409	18	C:\Users\mmur...	"C:\Users\mmur...	Slack
WINWORD.EXE	16780	Running	mmurgan	00	113,356 K	158,712 K	50,176 K	852	38	C:\Program F...	"C:\Program F...	Microsoft Word
sevens.exe	13358	Running	mmurgan	00	232,288 K	409,412 K	50,096 K	1,835	42	C:\Program F...	"C:\Program F...	Microsoft Visual Stuc
slack.exe	16652	Running	mmurgan	00	78,400 K	237,956 K	43,892 K	309	22	C:\Users\mmur...	"C:\Users\mmur...	Slack
chrome.exe	9844	Running	mmurgan	00	107,064 K	188,032 K	43,104 K	311	13	C:\Program F...	"C:\Program F...	Google Chrome
skypebrowserhost.exe	8912	Running	mmurgan	00	11,672 K	84,240 K	39,408 K	/26	19	C:\Program F...	"C:\Program F...	Skype Browser Host
explorer.exe	13112	Running	mmurgan	00	128,088 K	195,960 K	37,484 K	2,811	84	C:\Windows\ex...	"C:\Windows\ex...	Windows Explorer
explorer.exe	12824	Running	mmurgan	00	42,584 K	47,992 K	34,952 K	139	2	C:\Program F...	"C:\Program F...	Microsoft (G) Visual C
chrome.exe	17028	Running	mmurgan	00	206,040 K	219,760 K	32,712 K	330	17	C:\Program F...	"C:\Program F...	Google Chrome
SearchUI.exe	7056	Suspended	mmurgan	00	85,752 K	90,764 K	31,000 K	802	28	C:\Windows\Sys...	"C:\Windows\Sys...	Search and Cortana f...

[15p] Task A - Identify the problem

- What can you do if the system runs out of memory, but you do not know what causes this?

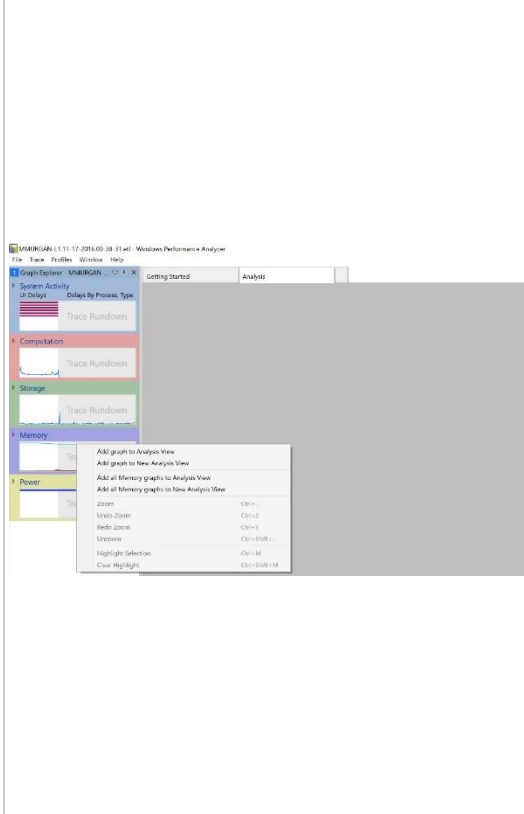
Windows Performance Recorder

Imagine a situation when a system encounters issues if some conditions are met, and these conditions can be reproduced. To find out the source of this problem, start **Windows Performance Recorder** configured as shown below.

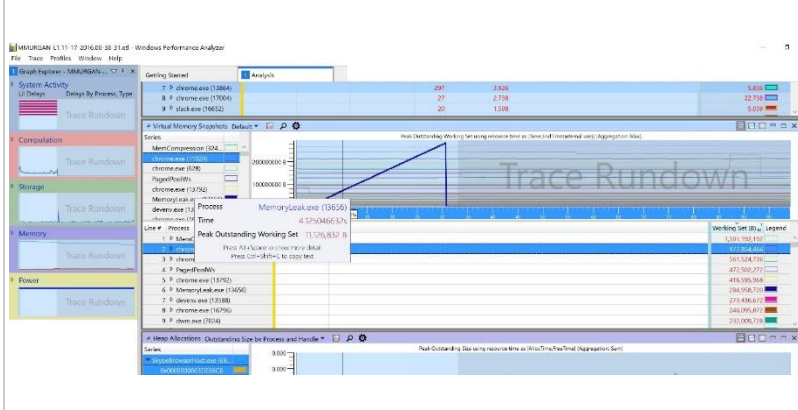


Windows Performance Analyzer

Run in parallel a program that allocates 1MB of memory every 100 milliseconds for a while and then stops. After the program stops running, save the capture and open it in **Windows Performance Analyzer**. You should get the following.



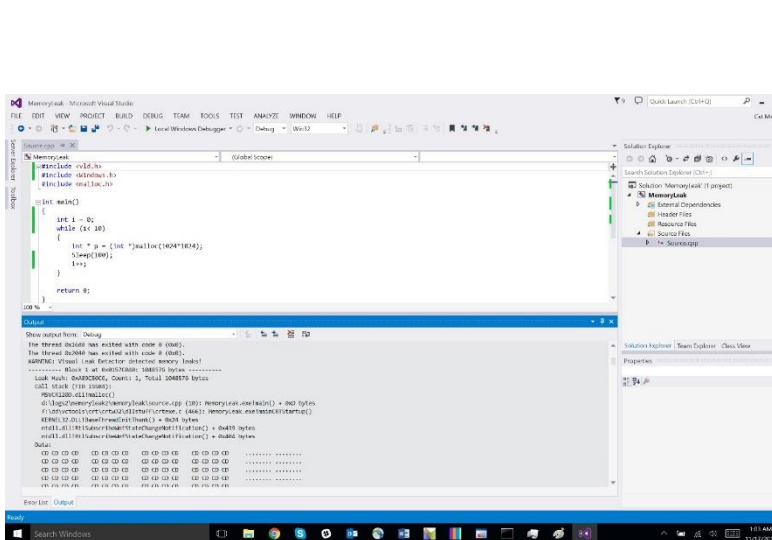
Click **Add all Memory graphs to Analysis View** and scroll down to the **Virtual Memory Snapshot** graph which



shows the memory usage for all processes.

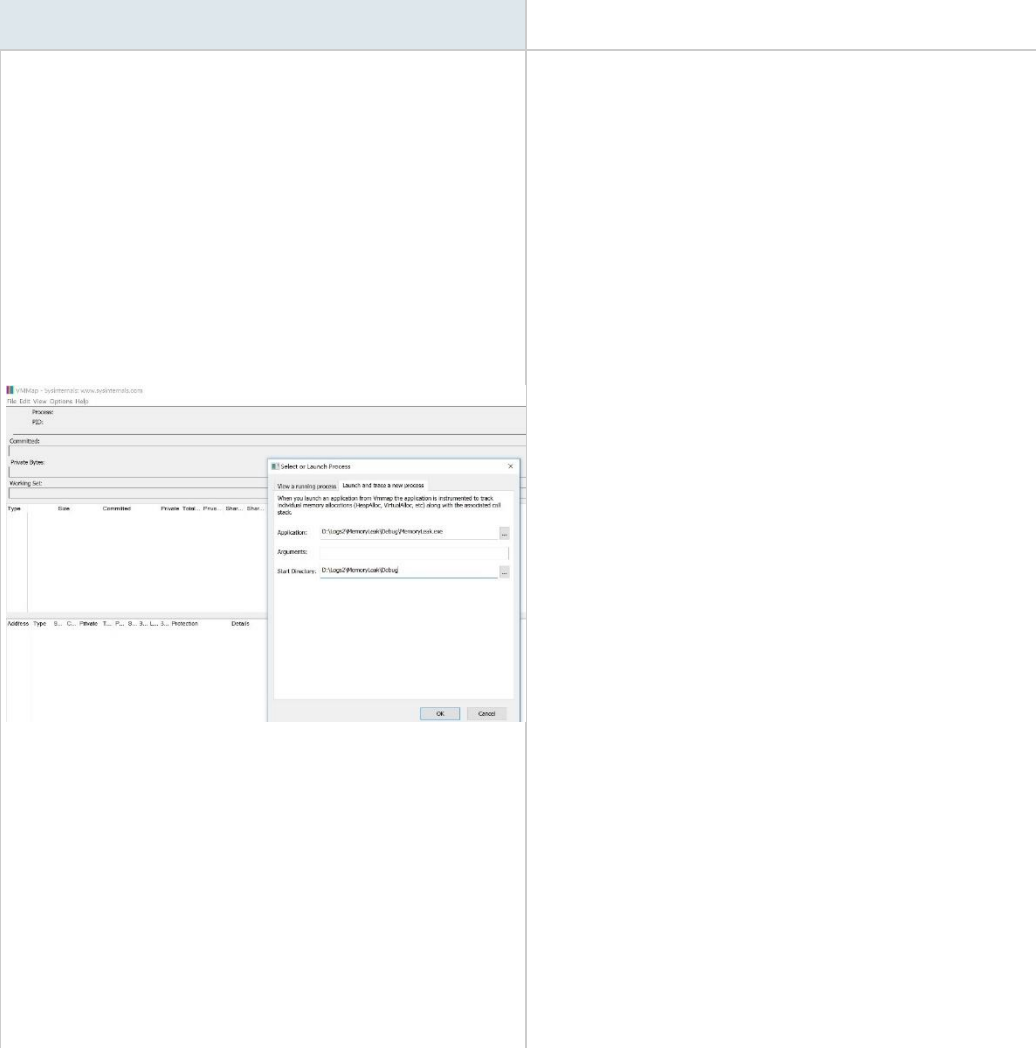
Visual Studio

After installing, it requires including the `vld.h` file. When writing the code, the following functions need to be overwritten: `malloc`, `free`, `new`, and `delete`. This allows each memory allocation and deallocation to be tracked. All the detected leakages (having an allocation that is not followed by a deallocation) will be saved in a log file that can be viewed after the program stops running. In the bottom part of the screenshot shown below, it can be noticed where the allocation took place and that it is not followed by a deallocation.



VMMMap

However, it might be the case that this is not a memory leak, and somewhere at the end of the program the memory gets unallocated. If this is the case, how can you determine which part of the program is responsible for generating the spike? This requires using another tool from SysInternals, [VMMMap](#). With this tool you can view a process's



memory allocations and usage. Use it to run the program that allocates 1 MB of memory every 100 milliseconds.

Prior to starting the tool, go to Options, Configure Symbols and set the paths to the program's, to the Microsoft Symbol Server, and to the program's source files. Start the tool, select **Launch and trace a new process**, select the process, select the directory where it will run, and let it run. You will see something similar to the screenshot below. To view the latest memory allocations, you need to double-click **Heap** in the upper-part of the screenshot, and hit F5 (refresh) from time to time. In the bottom

The screenshot displays a memory analysis tool interface. At the top, a table shows memory usage statistics:

Type	Size	Committed	Private	Total	Private	Shared	State
Total	11,984 K	70,208 K	98,644 K	102,760	97,420	2,000 K	1,252 K
Heap	7,872 K	7,560 K	308 K	1,844 K	204 K	1,640 K	1,320 K
Process	772 K	772 K	212 K	212 K	212 K	0 K	0 K
Stack	1,540 K	1,540 K	1,540 K	1,540 K	1,540 K	0 K	0 K
Image	10,944 K	86,988 K	86,988 K	86,988 K	86,988 K	0 K	0 K
Memory	6,400 K	292 K	292 K	84 K	84 K	0 K	0 K
Private	2,256 K	912 K	100 K	1,000 K	1,000 K	0 K	0 K
Private	300 K	300 K	300 K	300 K	300 K	0 K	0 K
Private	3,000 K	3,000 K	3,000 K	3,000 K	3,000 K	0 K	0 K
Private	2,000,504 K	3,800 K	3,800 K	3,800 K	3,800 K	0 K	0 K

Below the table is a process list with columns for Address, Type, Size, Private, Total, Private, Shared, and Process. The 'Heap' process is highlighted. A 'Call Stack' window is open, showing a list of memory addresses and their corresponding locations, such as '0x00401000' and '0x00401001'.

part of the screenshot, you can view the memory allocations. If you click one and press **Heap Allocations** you can see the stack where the allocation occurred. By pressing the “Source” button, you can view the actual code for the allocation.

[15p] Task B - Conclusions

- Discuss the output and call the assistant to show him/her your progress.

Exercise 02. [30p] CPU Monitoring

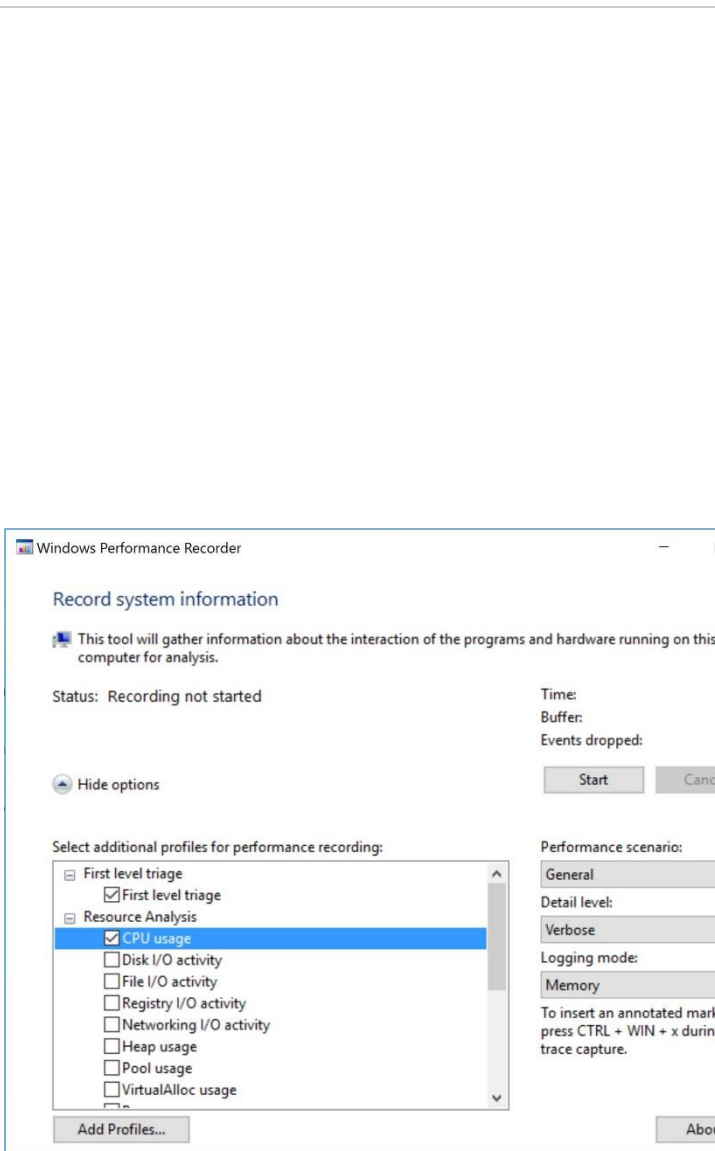
Task Manager

Monitoring the CPU usage presents similar issues to the ones encountered when monitoring the memory usage. Task Manager can help find out the current CPU usage for a process.

Name	PID	Status	Username	CPU	Memory (pri...)	Handles	Threads	Image path name	Command line	Description
System Idle Process	0	Running	SYSTEM	0	4 K		8			Percentage of time the processor is idle
audiohlg.exe	6220	Running	LOCAL S...V...	0	21,432 K	1,190	24	C:\Windows\Sys... C:\Windows\Sys...		Windows Audio Device Graph Isolation
taskmgr.exe	5104	Running	mmurgan	0	14,152 K	522	24	C:\Windows\Sys... C:\Windows\Sys...		Task Manager
rticff.exe	404	Running	mmurgan	0	26,856 K	212	11	C:\Program File... C:\Program File...		Adobe Reader
Skype.exe	6696	Running	mmurgan	0	71,104 K	1,734	67	C:\Program File... C:\Program File...		Skype
chrome.exe	2476	Running	mmurgan	0	145,580 K	386	23	C:\Program File... C:\Program File...		Google Chrome
System	4	Running	SYSTEM	0	28 K	1,594	182	C:\Windows\Sys...		NT Kernel & System
chrome.exe	7500	Running	mmurgan	0	110,500 K	394	22	C:\Program File... C:\Program File...		Google Chrome
chrome.exe	8816	Running	mmurgan	0	115,788 K	397	22	C:\Program File... C:\Program File...		Google Chrome
chrome.exe	3656	Running	mmurgan	0	194,232 K	508	36	C:\Program File... C:\Program File...		Google Chrome
System Interrupts	-	Running	SYSTEM	0	0 K	-	-			Deferred procedure calls and interrupt service routi
chrome.exe	8588	Running	mmurgan	0	145,160 K	2,358	46	C:\Program File... C:\Program File...		Google Chrome
chrome.exe	9104	Running	mmurgan	0	117,420 K	410	22	C:\Program File... C:\Program File...		Google Chrome
chrome.exe	9348	Running	mmurgan	0	128,608 K	321	18	C:\Program File... C:\Program File...		Google Chrome
chrome.exe	10084	Running	DWM 1	0	180,832 K	677	11	C:\Windows\Sys... chrome.exe		Desktop Window Manager
explorer.exe	4412	Running	mmurgan	0	29,376 K	2,022	79	C:\Windows\Sys... C:\Windows\Sys...		Windows Explorer
OUTLOOK.LXL	8100	Running	mmurgan	0	121,644 K	3,076	16	C:\Program File... C:\Program File...		Microsoft Outlook
chrome.exe	1236	Running	mmurgan	0	121,116 K	303	17	C:\Program File... C:\Program File...		Google Chrome
slack.exe	7356	Running	mmurgan	0	57,333 K	930	35	C:\Users\mmur... C:\Users\mmur...		Slack
SkypeBrowserHost.exe	10668	Running	mmurgan	0	51,272 K	619	24	C:\Program File... C:\Program File...		Skype Browser Host
slack.exe	8948	Running	mmurgan	0	119,216 K	425	23	C:\Users\mmur... C:\Users\mmur...		Slack
hSCPAH32v.exe	1121	Running	SYSTEM	0	2,318 K	146	4	C:\Program File... C:\Program File...		Stereo Vision Control Panel API Server
lync.exe	6772	Running	mmurgan	0	64,956 K	2,332	66	C:\Program File... C:\Program File...		Skype for Business

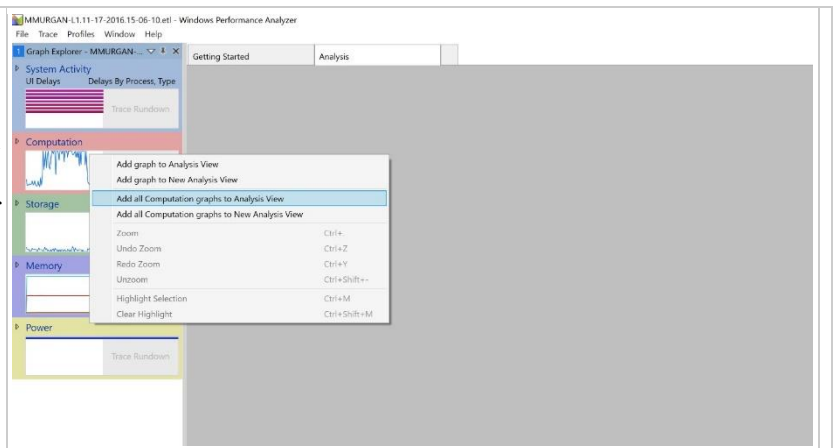
Windows Performance Recorder

Considering the same scenario as the one presented in the previous section: what if the CPU usage briefly spikes up and then goes back to normal, how can you catch this? Using Task Manager would mean having someone continuously watching what is happening to catch the moment when the spike occurs. Use Windows Performance Recorder, with the same settings as in the screenshot below.

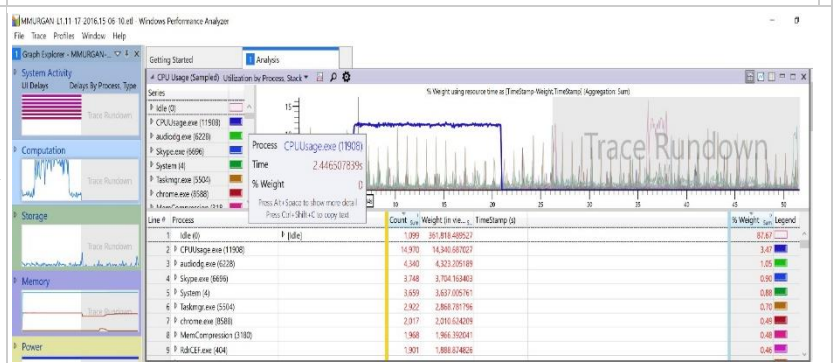


Windows Performance Analyzer

After clicking start, run CPUUsage.exe (part of the resources provided at the end of the tutorial), which generates CPU usage. Save and open the catch.



Right-click on to the Computation area and click “Add All Calculation graphs to Analysis View”.



The graph shows the impact of CPUUsage.exe, which inflicts a 12% CPU usage. Part of the problem is solved, it was determined who is generating the CPU usage. Now, to further debug the situation, as in the previous case of the memory, if this process was not written by you, check if it is useful, and if not, make sure to stop it. If it is useful, but it's not yours, you can try to find an update to fix the problem, or report the problem to the producer. If the program is written by you (this course - Performance Evaluation - targets the processes written by us), then it is important to determine what causes this problem. Unfortunately, unlike in the case of monitoring the memory usage, there is no tool that shows the stack with the problem, so you need to create one. Open EvenimenteProcMon, which has the purpose of integrating your messages with ProcessMonitor so they can be viewed as the process unfolds. It is necessary to understand any code, not perfectly, but at least to get the big picture of what is going on.

A ProcessMonitor class with 5 functions was created:

- **OpenProcMon** opens up a handle for the ProcessMonitor's message interface.
- **CloseProcMon** closes this handle.
- **ProcMonLog** writes the message that is passed as a parameter to the ProcessMonitor interface.
- **MyProcMon** is the class constructor. It is called when a MyProcMon object is declared.
- **~MyProcMon** is the class destructor. It is called to destroy the MyProcMon object.

The code below highlights that it was declared globally:

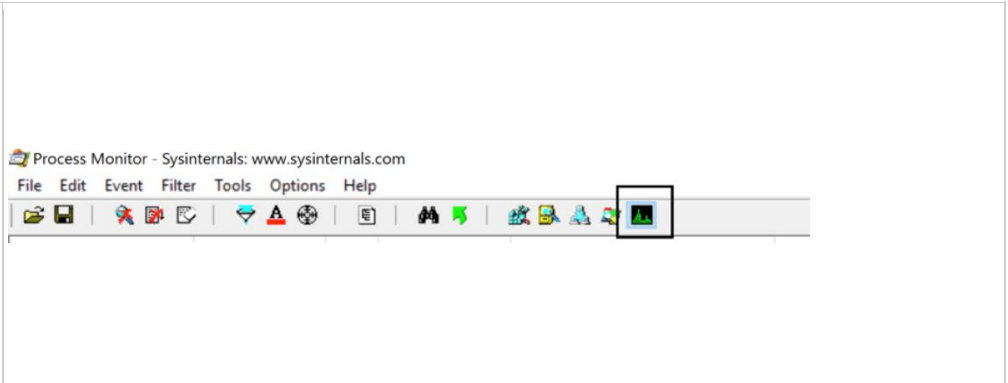
```
MyProcMon __procMon;
```

This means that at the start of the process, before executing the main function, when the global variables are initialized, our class instance will be constructed along with the implicit handle for the ProcessMonitor message interface. The handle is closed when the object is destroyed, after the program's execution ends.

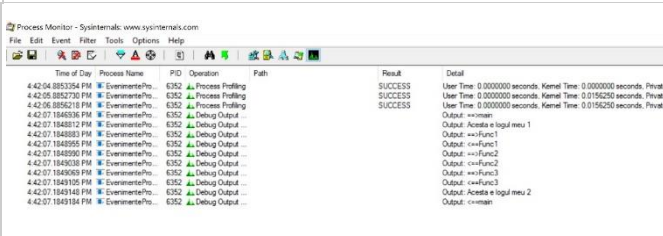
Another class was declared, ProcMonLogFunc, with the purpose of highlighting when entering and leaving a function. This led to defining the following macro, which declares a ProcMonLogFunc object and passes it the name of the current function as a parameter.

```
#define DBGTRACE_FN_() ProcMonLogFunc __my_log __ ( __ FUNCTIONW__ )
```

Start ProcessMonitor and change the filter to ProcessName contains EvenimenteProc Mon. Select the profiling button as shown below.



Running the program, generates a Process Monitor capture like this one.



Notice messages such as Output: ==> Func1 and Output: <== Func1, with the associated times for these events in the left-hand side of the screenshot, in the Time of Day column. The difference between the times (4:42:07.1848883 and 4:42:07.1848955) is 72, and since the times after the comma are expressed in hundreds of nanoseconds, this means that func1 took 7.2 microseconds.

As it is inefficient to calculate by hand the times for each function, save the output in csv format (File → Save and choose the “Comma-Separated Values” option). The generated file will look like this:

"4:42:07.1846936	PM", "EvenimenteProcMon.exe", "6352", "Debug Profiling", "", "", "Output: ==>main"	Output
"4:42:07.1848812	PM", "EvenimenteProcMon.exe", "6352", "Debug Profiling", "", "", "Output: Acesta e logul meu 1"	Output
"4:42:07.1848883	PM", "EvenimenteProcMon.exe", "6352", "Debug Profiling", "", "", "Output: ==>Func1"	Output
"4:42:07.1848955	PM", "EvenimenteProcMon.exe", "6352", "Debug Profiling", "", "", "Output: <==Func1"	Output
"4:42:07.1848990	PM", "EvenimenteProcMon.exe", "6352", "Debug Profiling", "", "", "Output: ==>Func2"	Output
"4:42:07.1849038	PM", "EvenimenteProcMon.exe", "6352", "Debug Profiling", "", "", "Output: <==Func2"	Output
"4:42:07.1849069	PM", "EvenimenteProcMon.exe", "6352", "Debug Profiling", "", "", "Output: ==>Func3"	Output

"4:42:07.1849105	PM", "EvenimenteProcMon.exe", "6352", "Debug Profiling", "", "", "Output: <==Func3"	Output
"4:42:07.1849148	PM", "EvenimenteProcMon.exe", "6352", "Debug Profiling", "", "", "Output: Acesta e logul meu 2"	Output
"4:42:07.1849184	PM", "EvenimenteProcMon.exe", "6352", "Debug Profiling", "", "", "Output: <==main"	Output

Making a parser in Python would make it easy to notice in which of the functions was spent the most time. If you only want to take into account the CPU usage, you need to have logging messages before and after every I/O operation, in order to not count in their time.

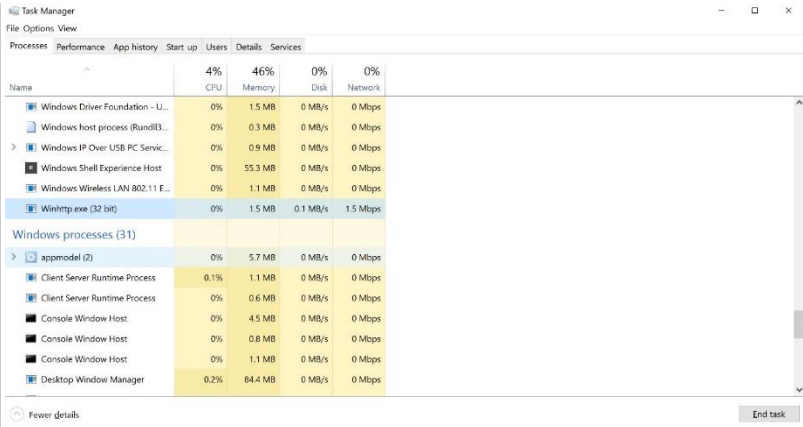
Integrate CPUUsage with ProcessMonitor and find out the total time spent in every function.

Exercise 03. [40p] Network Monitoring

Task A [20p] - Go through tutorial

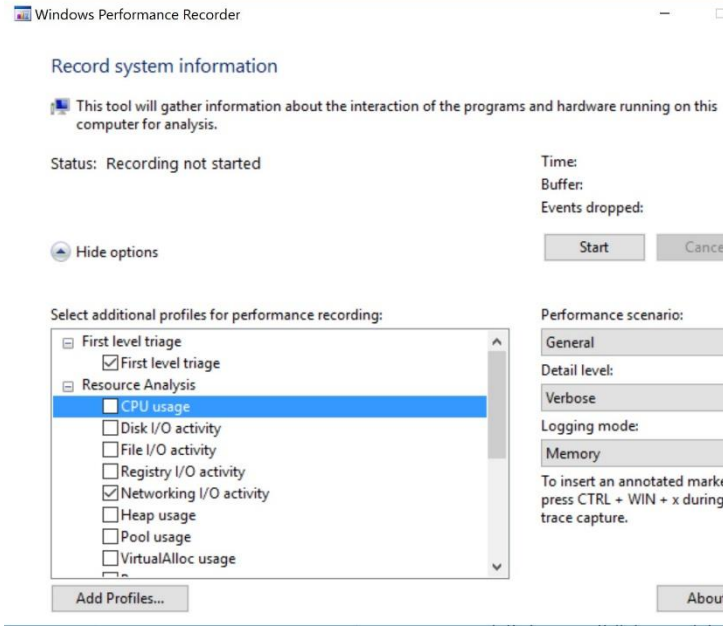
Task Manager

The amount of network traffic generated by a process can be seen using Task Manager.



Windows Performance Recorder

The resources for this tutorial include Winhttp.exe, a program that downloads putty.exe. The above screenshot displays its network activity. However, if the



process generating the network activity is unknown, you can use Windows Performance Recorder with the following settings. Save and open the capture to view it. The statistics offered by Windows Performance Analyzer are for the total use of the network, rather than per process statistics.

Microsoft Network Monitoring

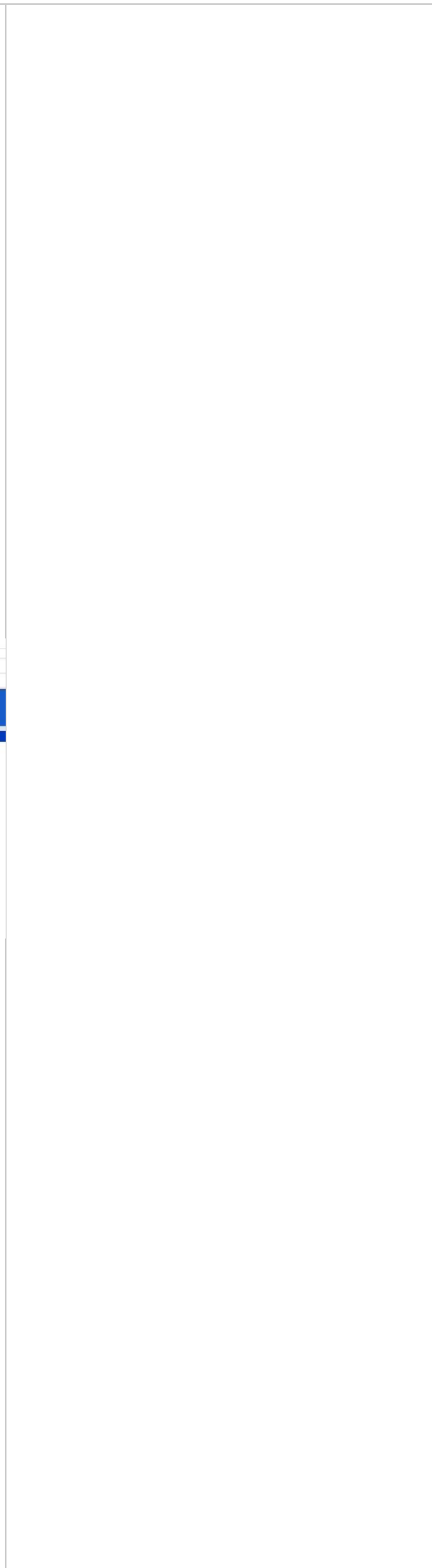
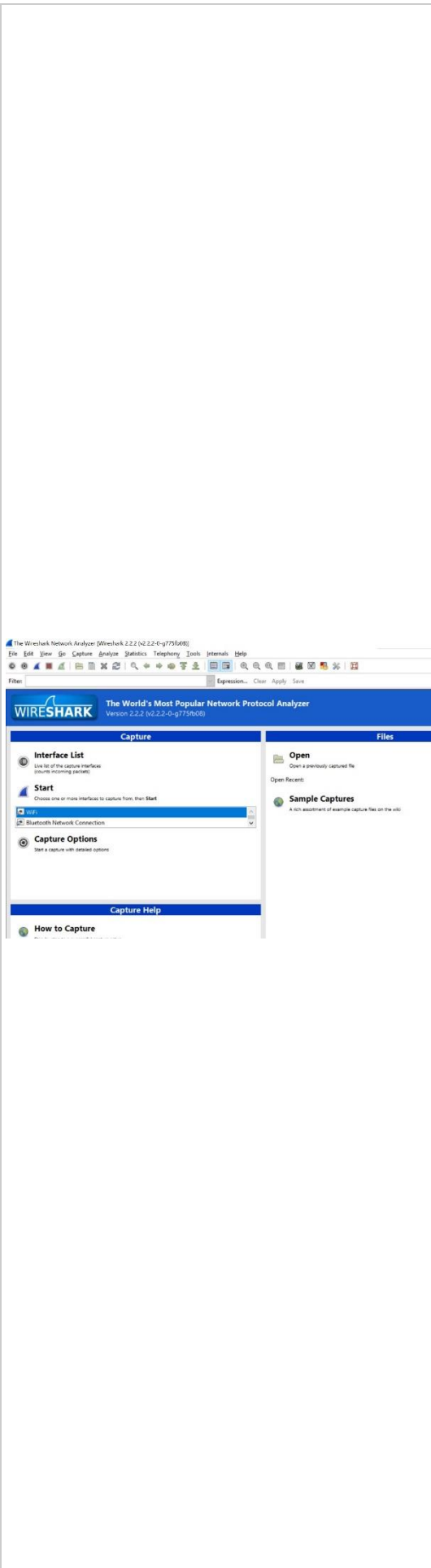
For this reason, we are calling upon another tool developed by Microsoft. Install it, start it using “Run as administrator”, and select the network interface through which the traffic is expected to pass (cable, wifi, ...). You should get a capture such as this one:

The screenshot displays the Microsoft Network Monitor 3.4 interface. The main window shows a list of network conversations. The selected conversation is expanded to show a list of frames. The frames are filtered to show only HTTP traffic. The frame list includes columns for Time, Size, Protocol, and Description. The description column shows the details of the HTTP request, including the method (GET), the URL (http://www.microsoft.com/), and the user agent (Mozilla/5.0).

Time	Size	Protocol	Description
7:50:20 PM 11/17/2010 0:1131233	1024	HTTP	HTTP GET http://www.microsoft.com/
7:50:20 PM 11/17/2010 0:1131234	1024	HTTP	HTTP GET http://www.microsoft.com/
7:50:20 PM 11/17/2010 0:1131235	1024	HTTP	HTTP GET http://www.microsoft.com/
7:50:20 PM 11/17/2010 0:1131236	1024	HTTP	HTTP GET http://www.microsoft.com/
7:50:20 PM 11/17/2010 0:1131237	1024	HTTP	HTTP GET http://www.microsoft.com/
7:50:20 PM 11/17/2010 0:1131238	1024	HTTP	HTTP GET http://www.microsoft.com/
7:50:20 PM 11/17/2010 0:1131239	1024	HTTP	HTTP GET http://www.microsoft.com/
7:50:20 PM 11/17/2010 0:1131240	1024	HTTP	HTTP GET http://www.microsoft.com/
7:50:20 PM 11/17/2010 0:1131241	1024	HTTP	HTTP GET http://www.microsoft.com/
7:50:20 PM 11/17/2010 0:1131242	1024	HTTP	HTTP GET http://www.microsoft.com/
7:50:20 PM 11/17/2010 0:1131243	1024	HTTP	HTTP GET http://www.microsoft.com/
7:50:20 PM 11/17/2010 0:1131244	1024	HTTP	HTTP GET http://www.microsoft.com/
7:50:20 PM 11/17/2010 0:1131245	1024	HTTP	HTTP GET http://www.microsoft.com/
7:50:20 PM 11/17/2010 0:1131246	1024	HTTP	HTTP GET http://www.microsoft.com/
7:50:20 PM 11/17/2010 0:1131247	1024	HTTP	HTTP GET http://www.microsoft.com/
7:50:20 PM 11/17/2010 0:1131248	1024	HTTP	HTTP GET http://www.microsoft.com/
7:50:20 PM 11/17/2010 0:1131249	1024	HTTP	HTTP GET http://www.microsoft.com/
7:50:20 PM 11/17/2010 0:1131250	1024	HTTP	HTTP GET http://www.microsoft.com/

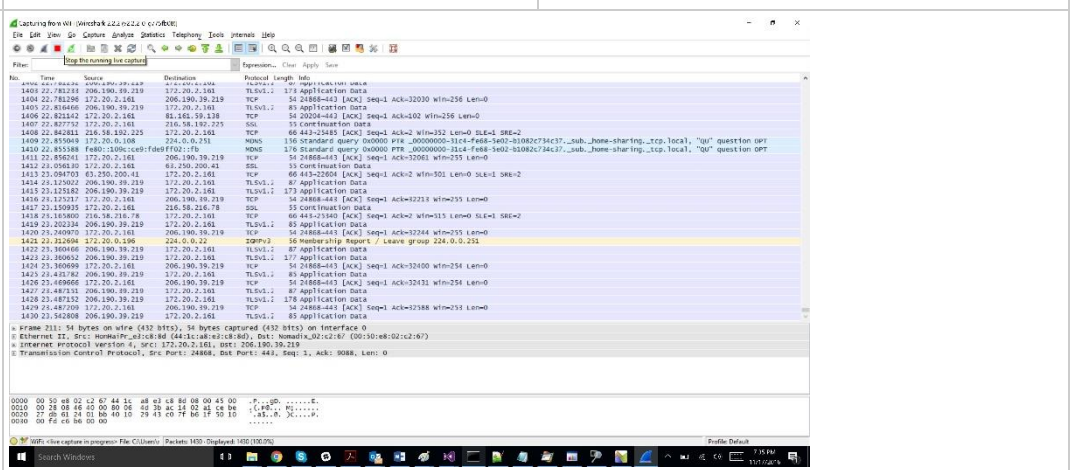
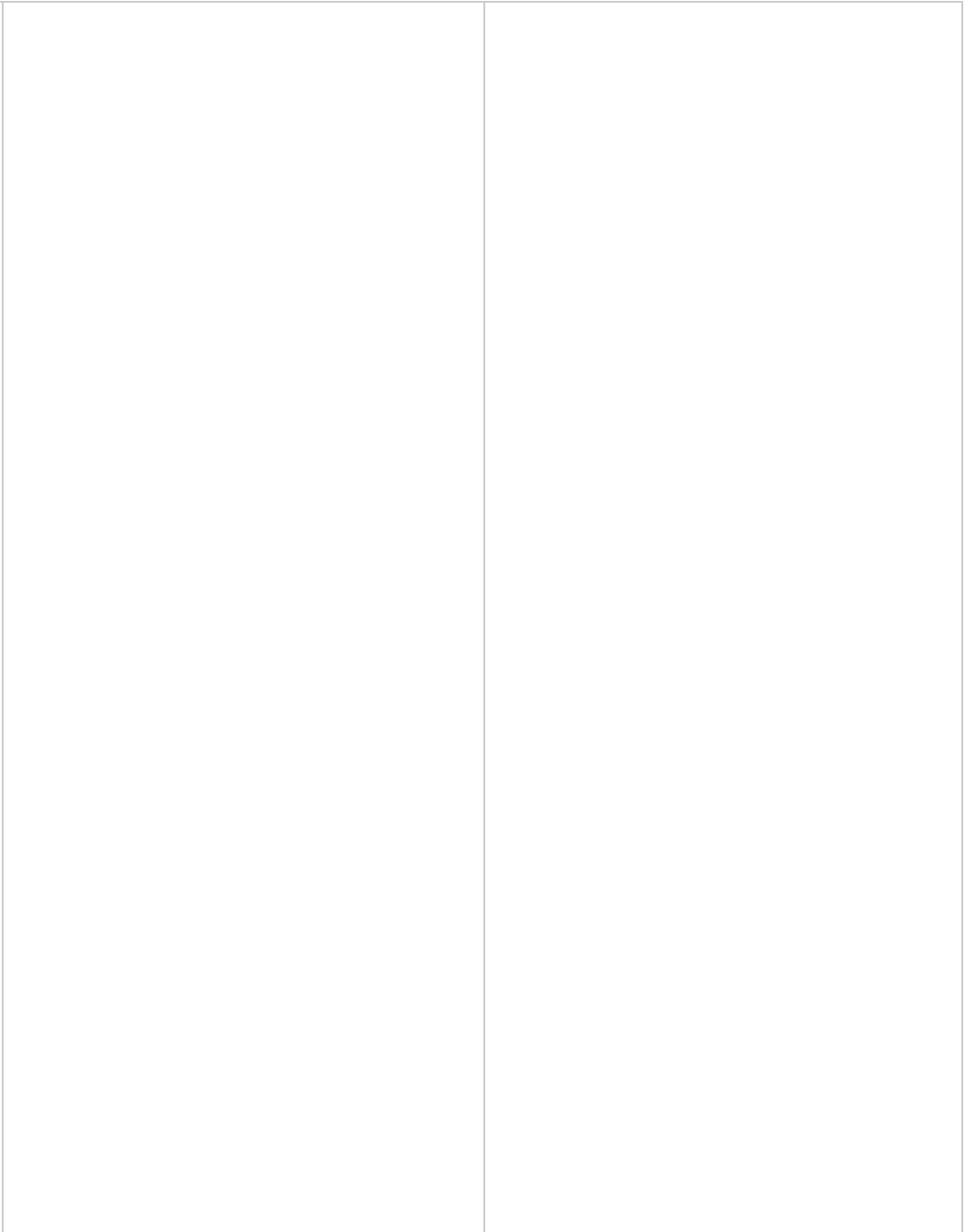
Wireshark

As in the case of the CPU, inspecting the events taking place on the network involves some amount of work for the analyst. However, this being a simple case, you can just expand the view on the traffic generated by Winhttp.exe, and notice the request for *putty.exe*. If it is not clear why some requests are there or why they last so long, you can integrate the application that you wish to investigate with ProcessMonitor. This way you can insert logging elements to find out what request are made and how long they take. The part with timing



the requests and traffic can be determined straight from Network Monitor by considering the times of the packets. For displaying all traffic on a http connection (it can also be https as long as you control the server, but this is not in the scope of this tutorial), you can use another tool, Wireshark. Install Wireshark (64bit!!!) accepting the default settings. Start it and select the interface that you want to listen to.

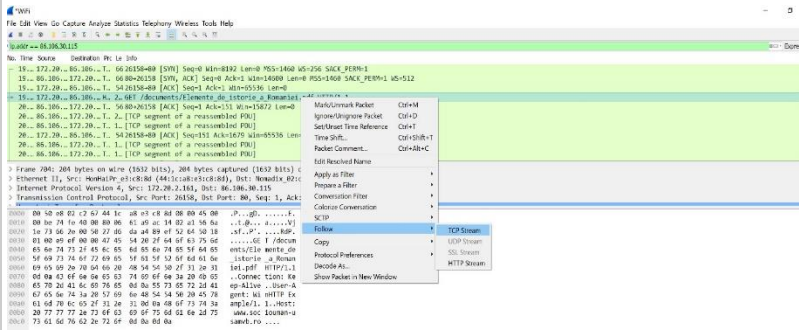
Click the Start button on and run Winhttp.exe. After Winhttp.exe stops, click the Stop button in Wireshark.



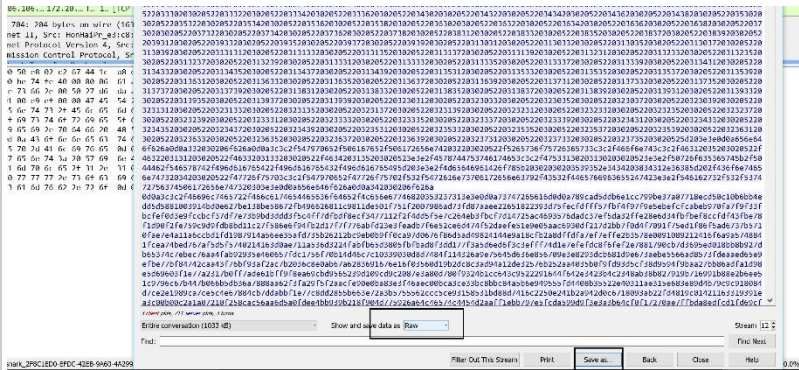
This way you have obtained a traffic capture while winhttp.exe was running. Viewing the code for winhttp.exe, it can be noticed that it makes a request to www.sociouman-usamvb.ro. Use the ping command to get the IP address for this url.


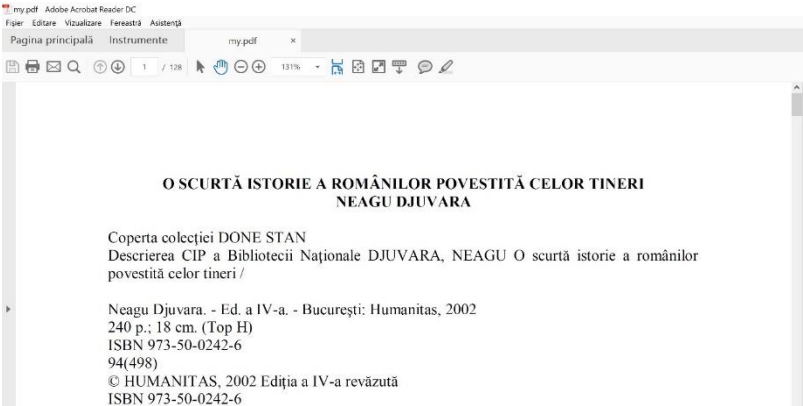
```
D:\Logs2\Winhttp\Debug>ping www.sociouman-usamvb.ro
Pinging sociouman-usamvb.ro [86.106.30.115] with 32 bytes of data:
Reply from 86.106.30.115: bytes=32 time=143ms TTL=48
Reply from 86.106.30.115: bytes=32 time=143ms TTL=48
Reply from 86.106.30.115: bytes=32 time=143ms TTL=48
Reply from 86.106.30.115: bytes=32 time=144ms TTL=48
```

Switching back to Wireshark, add a filter for ip.addr == 86.106.30.115 (make sure to use the IP address identified using ping command). Right click Get documents and choose Follow TCP Stream.



In the bottom part of the Wireshark window, at the "Show and save data as" option choose



<p>“Raw”. Save the capture (using the “Save as” button) as “my.pdf”.</p>	
<p>Use Notepad++ to open the my.pdf file and remove the headers as shown in the screenshot below.</p>	
<p>Save it, close Notepad++ and double-click on the newly saved file (my.pdf).</p>	

Task B [20p] - Conclusions

- Discuss the output and call the assistant to show him/her your progress.