

COS20030

**Malware Analysis**

*Lab 1*

**Abnormal behaviour, PCAP Analysis, Introduction to Ghidra**

# Purpose

This is the first lab in the Malware Analysis subject. Before we start learning how to analyse malware, we have to first learn what is the normal behaviour of the Windows OS and what is not.

In the first part of this lab, we will learn about normal characteristics and behaviour of some of the main processes of Windows Operating System. We will also learn about some examples of abnormal behaviour of these processes which could be signs of a malware infection on a machine.

In the second part of this lab, we will analyse network PCAP(Packet Capture) files to find malicious network activities.

We will end this lab by starting to use Ghidra, a free open-source disassembler tool. A disassembler is one of the most important tools in a malware analyst’s toolset. In later labs we will learn how Ghidra can be used in malware analysis.

# Outcome

* Learn to detect abnormal behaviour of OS essential processes
* Learn to use WireShark and NetMiner for network analysis
* Learn to use Ghidra to disassemble binary files

# Abnormal system behaviour

Malwares often try to pretend to be a part of the operating system in order not to raise any suspicion. To detect this sort of behaviour, it’s important to learn what is the normal behaviour of the system processes and what is the abnormal behaviour.

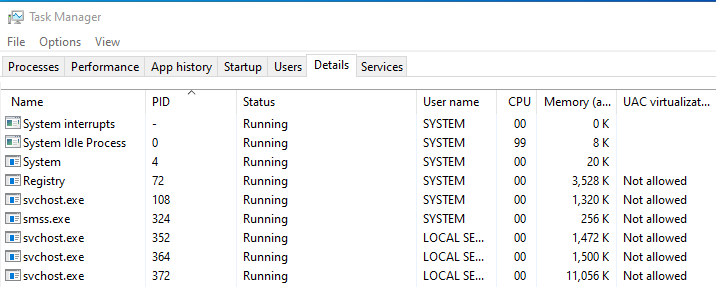
The following article summarises the normal and abnormal behaviour of some of the essential processes of the Windows operating system.

[https://0xcybery.github.io/blog/Core-Processes-In-Windows-System##smss.exe-(Session-Manager-Subsystem)](https://0xcybery.github.io/blog/Core-Processes-In-Windows-System)

Read the article and based on the knowledge you gained, try to find the abnormal behaviour shown in each screenshot below.

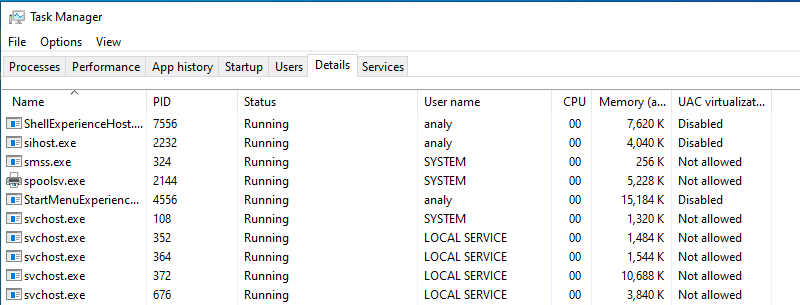
##### Exercise 1

|  |  |
| --- | --- |
| Question | Answer |
| List all the abnormalities seen in the screenshot | System pid not 4, multiple svchost.exe, one of them has high memory usage |



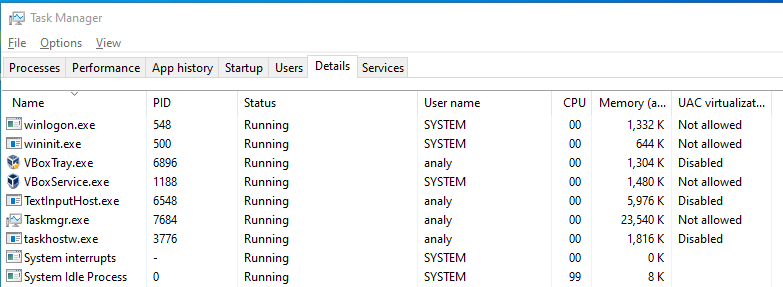
##### Exercise 2

|  |  |
| --- | --- |
| Question | Answer |
| List all the abnormalities seen in the screenshot | Multiple instances of smss.exe, unusual PIDs for system processes, high memory usage by StartMenuExperienceHost, UAC virtualization is disabled |



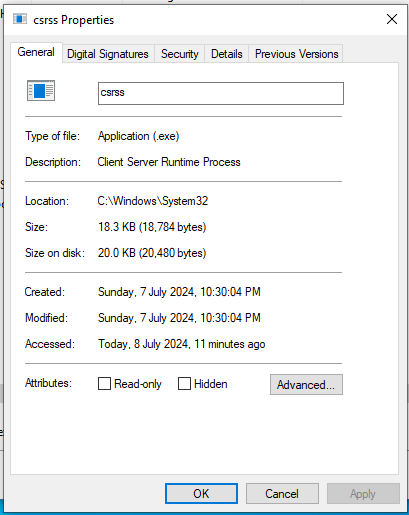
##### Exercise 3

|  |  |
| --- | --- |
| Question | Answer |
| List all the abnormalities seen in the screenshot | Multiple instances of virtual machine (only one should be running), unusual memory usage by Taskmgr.exe (23k), high cpu usage by System Idle Process, system interrupts shows 0k memory. |



##### Exercise 4

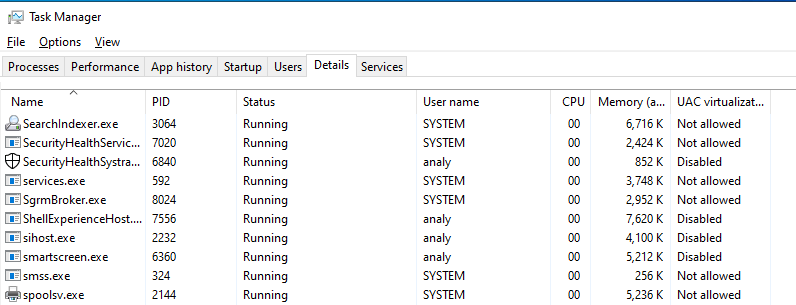
|  |  |
| --- | --- |
| Question | Answer |
| List all the abnormalities seen in the screenshot |  |



##### 

##### Exercise 5

|  |  |
| --- | --- |
| Question | Answer |
| List all the abnormalities seen in the screenshot |  |



# PCAP Analysis

##### Sample 1

In this exercise we want to analyse a packet capture (PCAP) file which contains the network traffic captured from a system infected with a malicious downloader program. The malicious program connects to a remote server, downloads another malware and executes this malware. The downloaded malware is a sample of the Lokibot malware. You can read about this malware here:

<https://www.cisa.gov/news-events/cybersecurity-advisories/aa20-266a>

In the PCAP file there is evidence of connection to the remote server and downloading the LokiBot executable file. We are also aware that LokiBot was also executed on the infected machine and attempted to communicate with its command and control server.

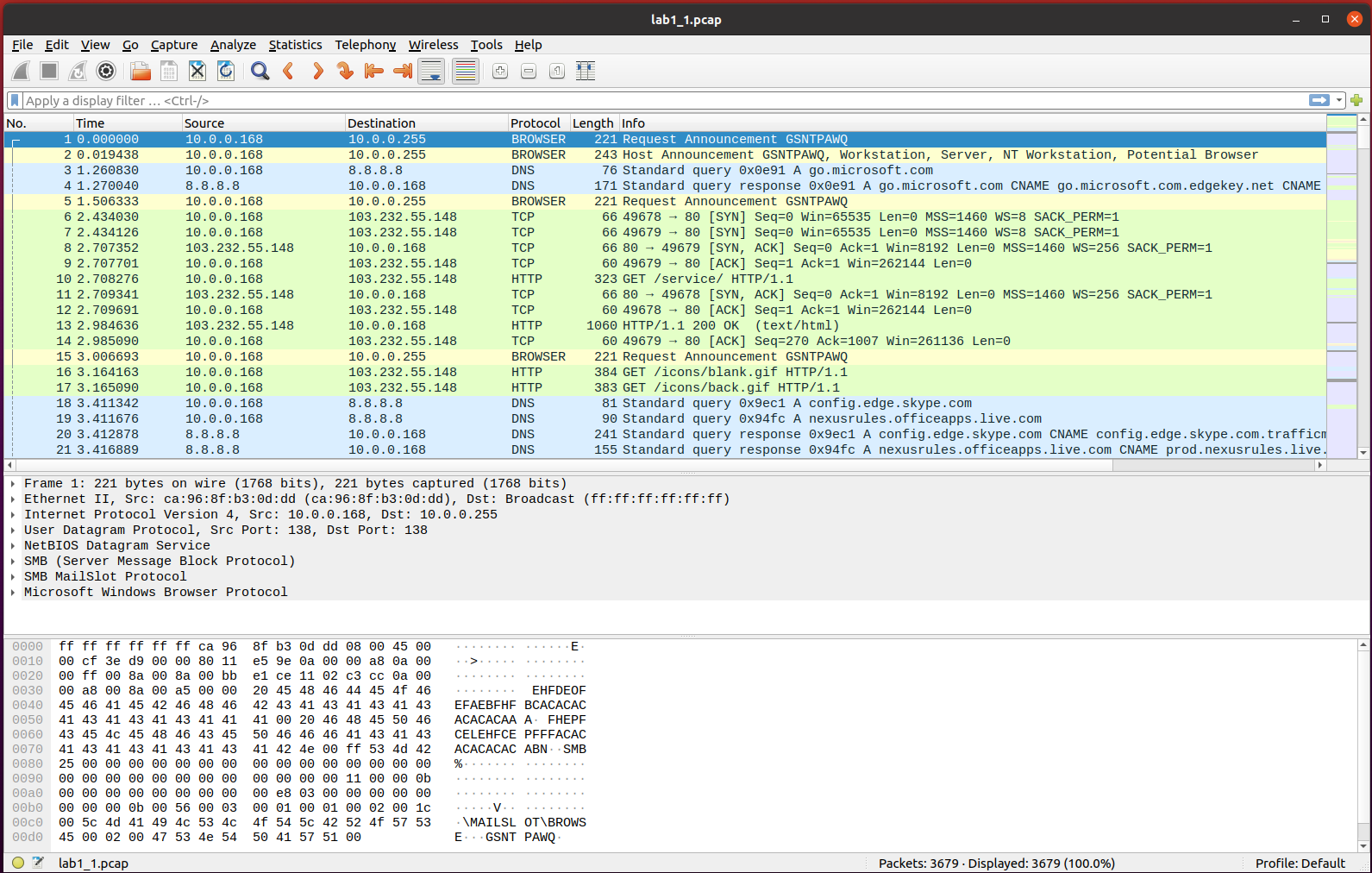
Our goal is to find the address of the download server, extract the downloaded LokiBot and find the address of the server that LokiBot communicates with.

###### WireShark

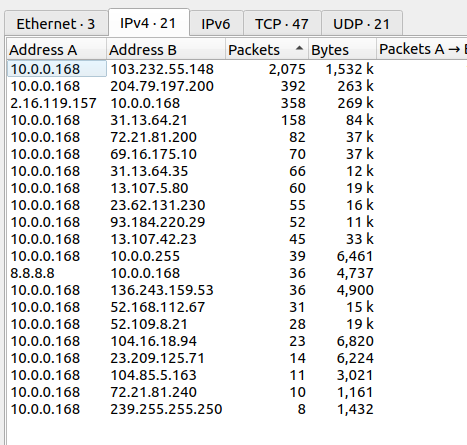
First tool to use is WireShark. Wireshark is a tool capable of capturing network traffic on a machine as well as analysing a PCAP file created previously on the same machine or on a different machine.

Open Wireshark via its desktop shortcut and navigate to the Lab1\_1.pcap file in the samples directory.

Once the PCAP file is loaded, all the sent and received packets are shown. You can use different features of this tool to find valuable information about the network traffic. You can also create different filters to see only specific packets.



Let’s first find the IP address with which the highest number of packets were communicated. This can be the download server's IP address, but we need to confirm this hypothesis. Go to Statistics-> Conversations and sort the list by number of packets transmitted by clicking on the Packets column



From this list we can clearly see the IP address of the infected machine is 10.0.0.168 and we also found the IP address we were looking for.

Now, let’s look at the network traffic between the source IP and this IP address. To only see the communication between these two IP addresses we need to create a filter. You can do that by right-clicking on the line which shows these two IP addresses and selecting

Apply As Filter -> Selected -> A A black line with arrows

Description automatically generated B

Now close this window and return to the main window. You can add more items to the filter created for you. In this case, let’s add “http” to the current filter, like below.



The packets are now filtered to only HTTP packets transmitted between the infected machine and the suspected IP address.

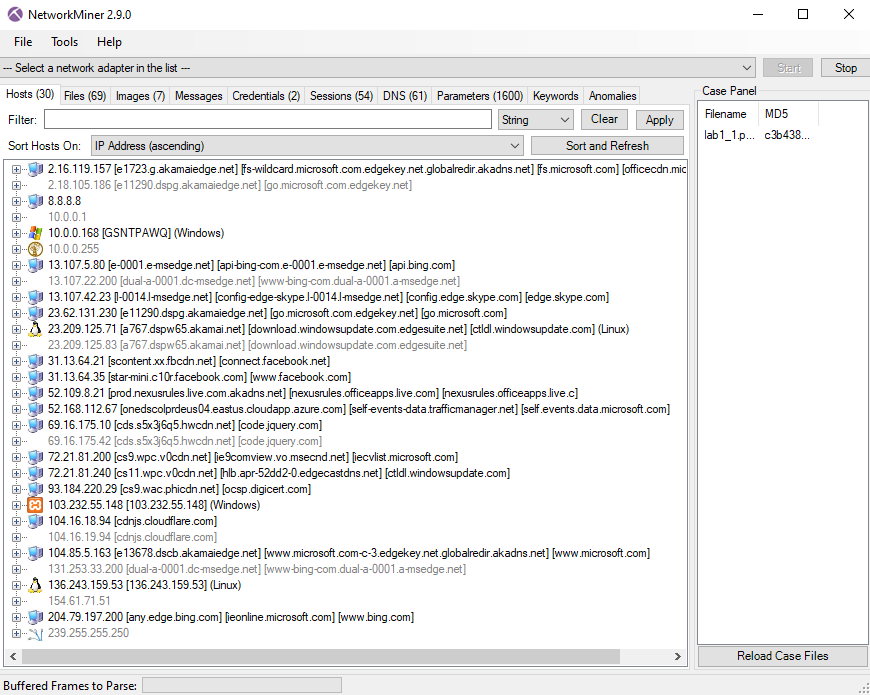
Now, just scroll down and look through the “Info” column to find a reference to a GET packet requesting a file with “.exe” extension. This is the LokiBot sample downloaded from the remote server.

Right click on the line showing the GET request and choose Follow -> TCP Stream to see the full TCP communication in this session.



In the TCP stream, we can see the executable file being downloaded from the server. “MZ” string at the beginning of the packet is the signature for the Windows Portable Executable(PE) file. We will learn more about PE files in later weeks.

###### NetworkMiner

Next step is to extract this file for further investigation. In order to do that, we use a different tool called NetworkMiner. Open this tool via its desktop shortcut and open the PCAP file into the tool.

NetworkMiner is a very useful tool for network analysis and it can extract important details from PCAP files. It also automatically extracts all the files being sent or received in the network traffic.

In the first tab which is the “Hosts” tab, you can find various details about all the hosts found in the PCAP file listed under their IP addresses.

Find the IP address where the LokiBot sample is downloaded from and fill in the table below.

|  |  |
| --- | --- |
| Question | Answer |
| IP Address |  |
| MAC address |  |
| The type of OS |  |
| Number of incoming sessions |  |

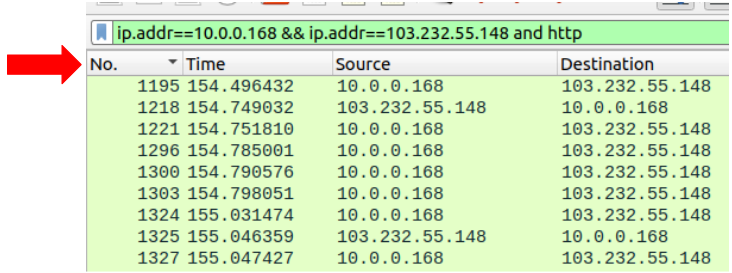
From the “Files” tab, find the line which corresponds to the LokiBot sample, right-click on the line and choose “Open Folder” to find the extracted file. Make a copy of the file somewhere that you can access easily. This is a malicious file so change the extension of the file to “.exe\_” instead of “.exe” to make sure that it doesn’t accidentally get executed. Now change the filename to lab1\_3.exe\_. You will use this file to practise using Ghidra in the last part of this lab.

###### LokiBot communication with its command-and-control server

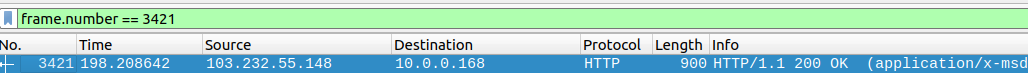
Once the LokiBot sample is downloaded, it starts communicating with a new domain which is its command and control server. Our next goal is to find the command and control domain.

In order to do that, first we need to filter out all the communication packets prior to the download of the LokiBot sample.

In order to do that, let's find the last packet of communication before the LokiBot sample is downloaded. Keep the previous filter and click on the first column which is the “No.” (Frame Number) to sort the packets based on their frame number.



Now scroll all the way down to find the last packet. Right-click on the frame number and create a filter by choosing Apply As a Filter -> Selected.



Now, just edit the filter to show the frame numbers greater than the value and also add “http” to the filter.

A screen shot of a computer

Description automatically generated

If you have done everything correctly, you can see the command and control server’s IP address.

Right-click on the first TCP packet and choose Follow->TCP Stream to see the whole communication in this session.

Fill in the table below to summarise your findings.

|  |  |
| --- | --- |
| Question | Answer |
| What is the Command and Control’s IP address? |  |
| What is the URI used for POST requests |  |
| There is a specific string ending in “.ru” sent to the server. What is the full string? |  |

##### Sample 2

Use the lab1\_2.pcap file for this exercise. In this exercise, we have a PCAP file that is captured from a machine infected with multi-stage malware.

The PCAP file starts with the packets generated by the first-stage malware which connects to a domain and downloads two files. The first file is downloaded as a base64-encoded stream and the second one is a password-protected zip file.

The second stage of the attacks is decoded from the base64-encoded stream. This stage then generates more network traffic. The second stage malware is very deceptive malware and is responsible for downloading the final payload of the attack. It uses a number of tricks to imitate the normal network traffic of the system. For example, it adds fake “host” and “referrer” fields to the HTTP packet headers to mimic normal network traffic and it uses “.js” as the extension of the file it downloads from its server, even though the file is a sophisticated standalone shellcode and not a JavaScript file.

The goal of this exercise is to find information about the malware files and the domains used in this attack.

Go through the following steps and fill the table with relevant information.

1- Use Wireshark to find the two most contacted IP addresses in the PCAP file

2- In Wireshark create a filter on “dns” and find the domains resolved to the IP addresses found in the previous step.

3- Follow the TCP stream from the second domains and find the fake “Host” and “Referrer” values

4- Use NetworkMiner to find the files downloaded from these two domains.

|  |  |
| --- | --- |
| Question | Answer |
| Domain from the first stage of the attack |  |
| The IP address hosted the first domain |  |
| Domain from the second stage of the attack |  |
| The IP address hosted the second domain |  |
| The URIs were the files downloaded from the first domain |  |
| The URIs were the files downloaded from the first domain |  |
| Fake “Host” value |  |
| Fake “Referrer” value |  |

# 

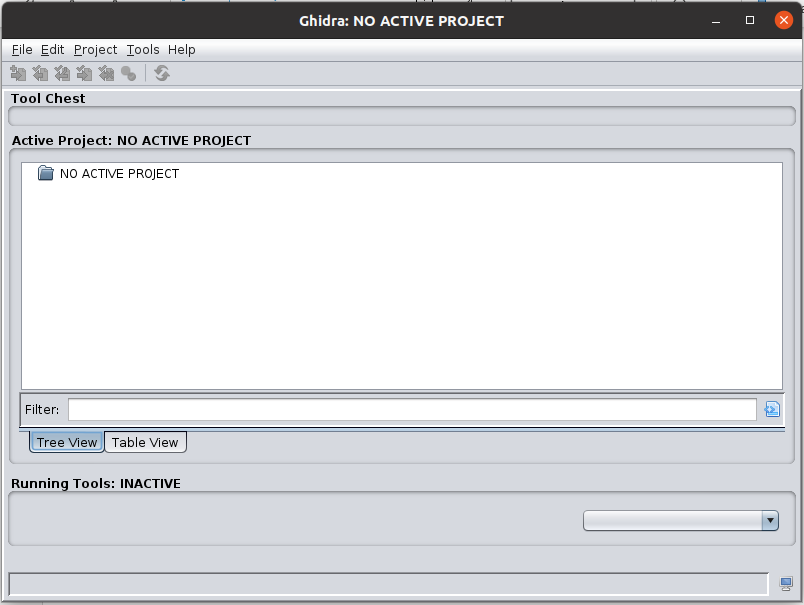
# Use Ghidra for static analysis

Ghidra is a free and open-source reverse engineering tool developed by the National Security Agency of the United States. Although the tool is relatively new compared to other tools with similar capabilities, it has been widely used by security experts since its release in 2019.

In this part of the lab, we are going to use Ghidra to have a quick look at the disassembled and decompiled code of a compiled program without getting into too much detail.

Execute Ghidra via its desktop shortcut. Everything in Ghidra happens in a project. Therefore, we need to create a new project for our analysis and then import the sample into this project.

The first time you open Ghidra, you will see a dialogue that shows there is no active project.

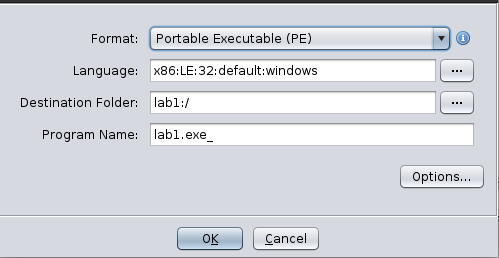


Go to File-> New Project and choose Non-shared Project. Then you need to choose a name for your project. Write lab1 in the Project Name field. Now, you will see the project is created.



From File -> Import file, import the lab1\_3.exe\_ file to this project.

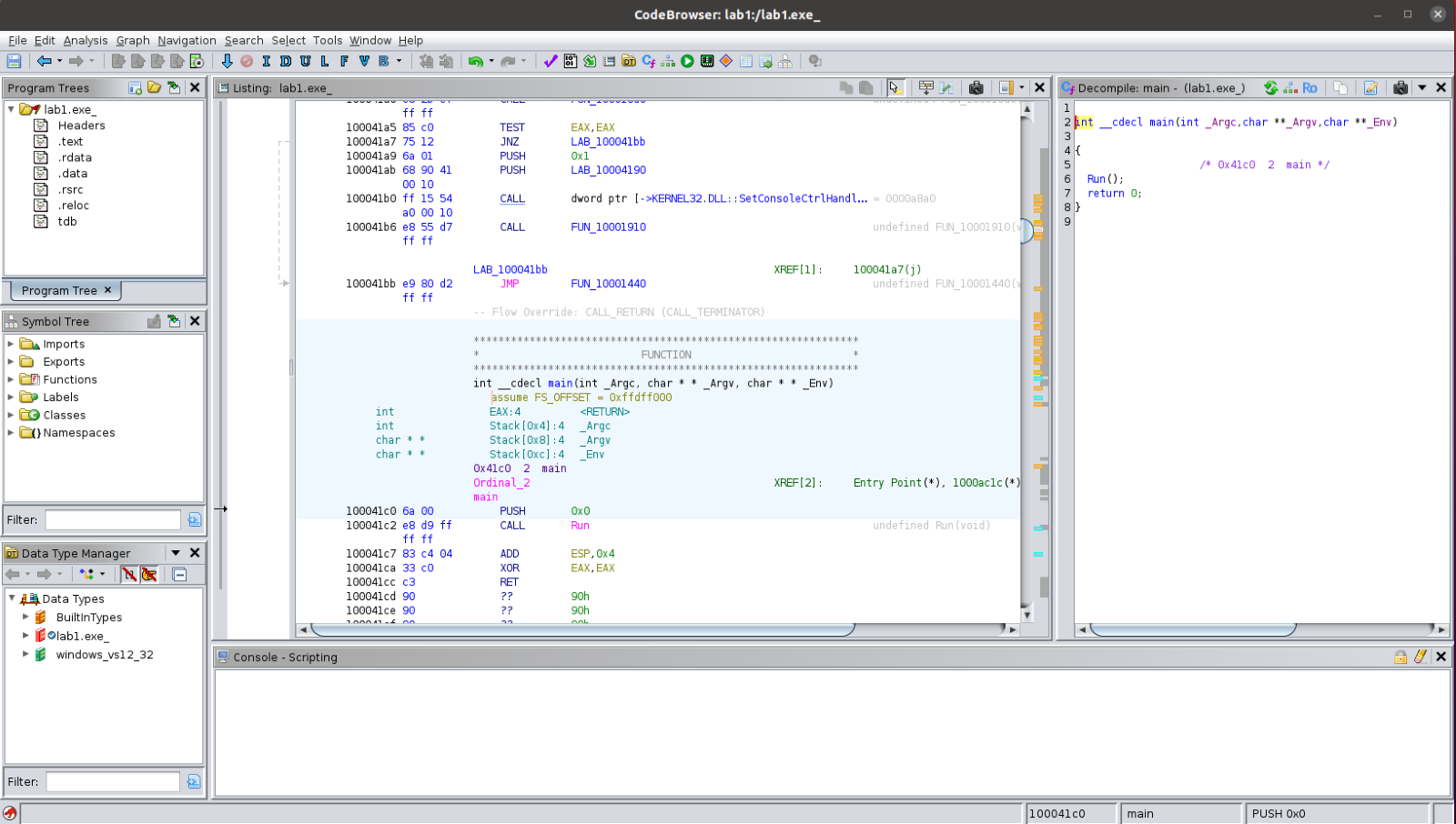
In the next window, you can see that Ghidra automatically recognises the file format and the platform the file is compiled for.



Click “OK” on the next two steps and allow the file to be imported. Finally, double-click on the imported file to start the analysis.

When asked if you want to analyse the file, click “Yes” and then “Analyze”.

Once the initial analysis by Ghidra is finished and the file is ready for our investigation, you will see the following layout.



In Ghidra you can see the disassembly and the decompiled code in the two main windows next to each other. As the two windows are synchronised, you can navigate through the code in each of the windows. To move to a new function, you can simply double-click on the name of the function in the code and to go back to the previous function, you can use Alt + left.

To better see the functions of the executable file, go to Window -> Functions.

Familiarise yourself with the layout of the tool. We will use this tool for analysis in the coming weeks.

*End of Lab*