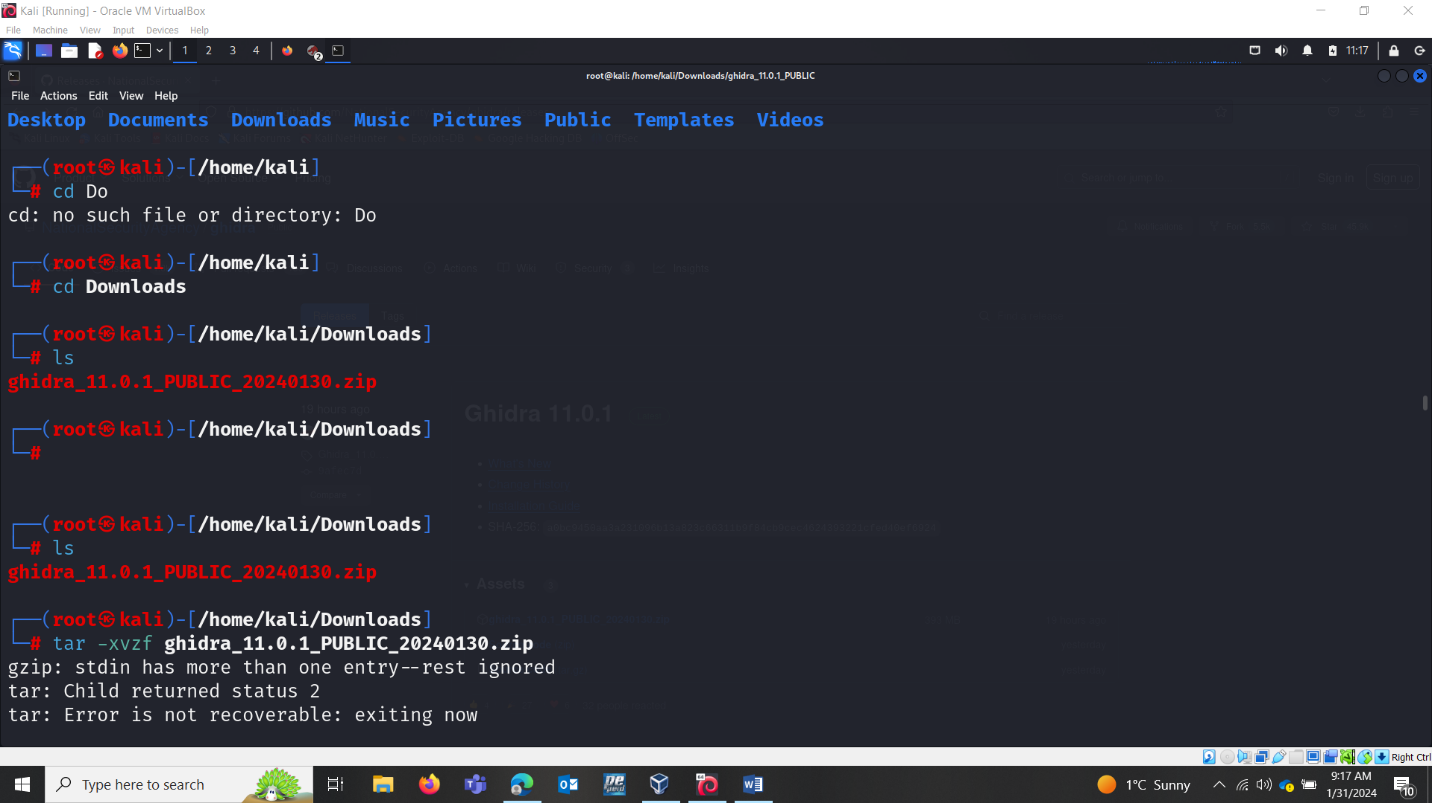
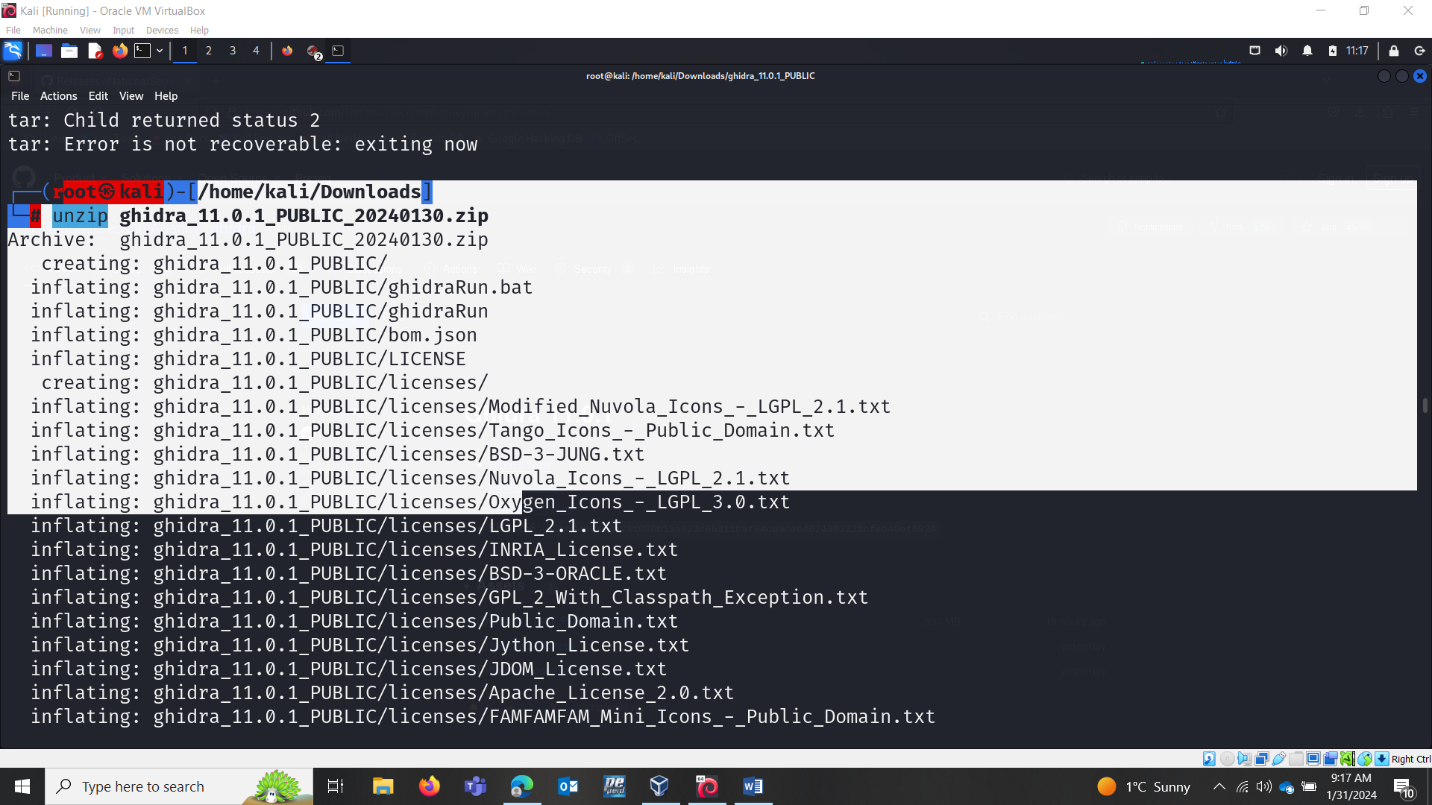
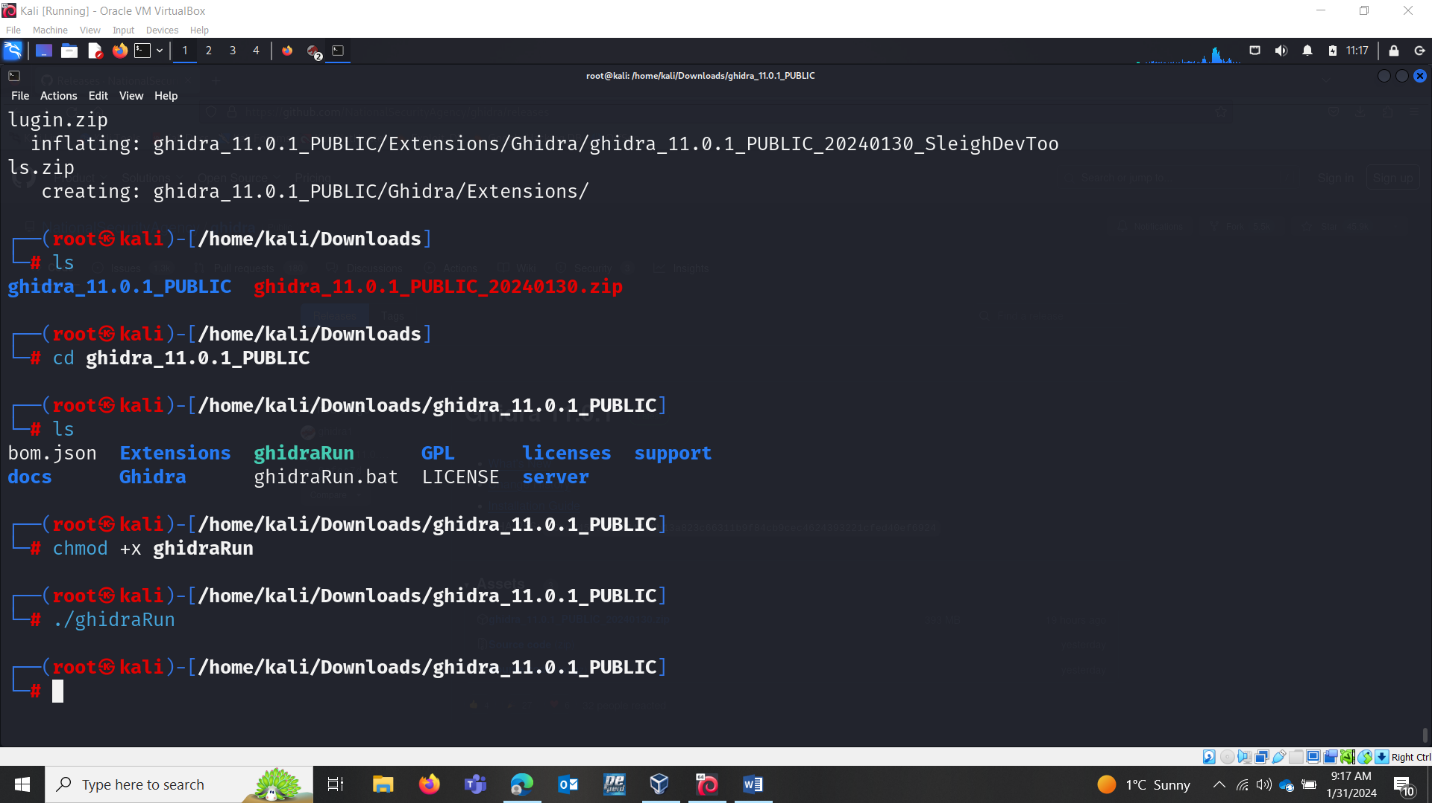
**GHIDRA**

* Download the package here: [Releases · NationalSecurityAgency/ghidra (github.com)](https://github.com/NationalSecurityAgency/ghidra/releases)
* Unzip it:

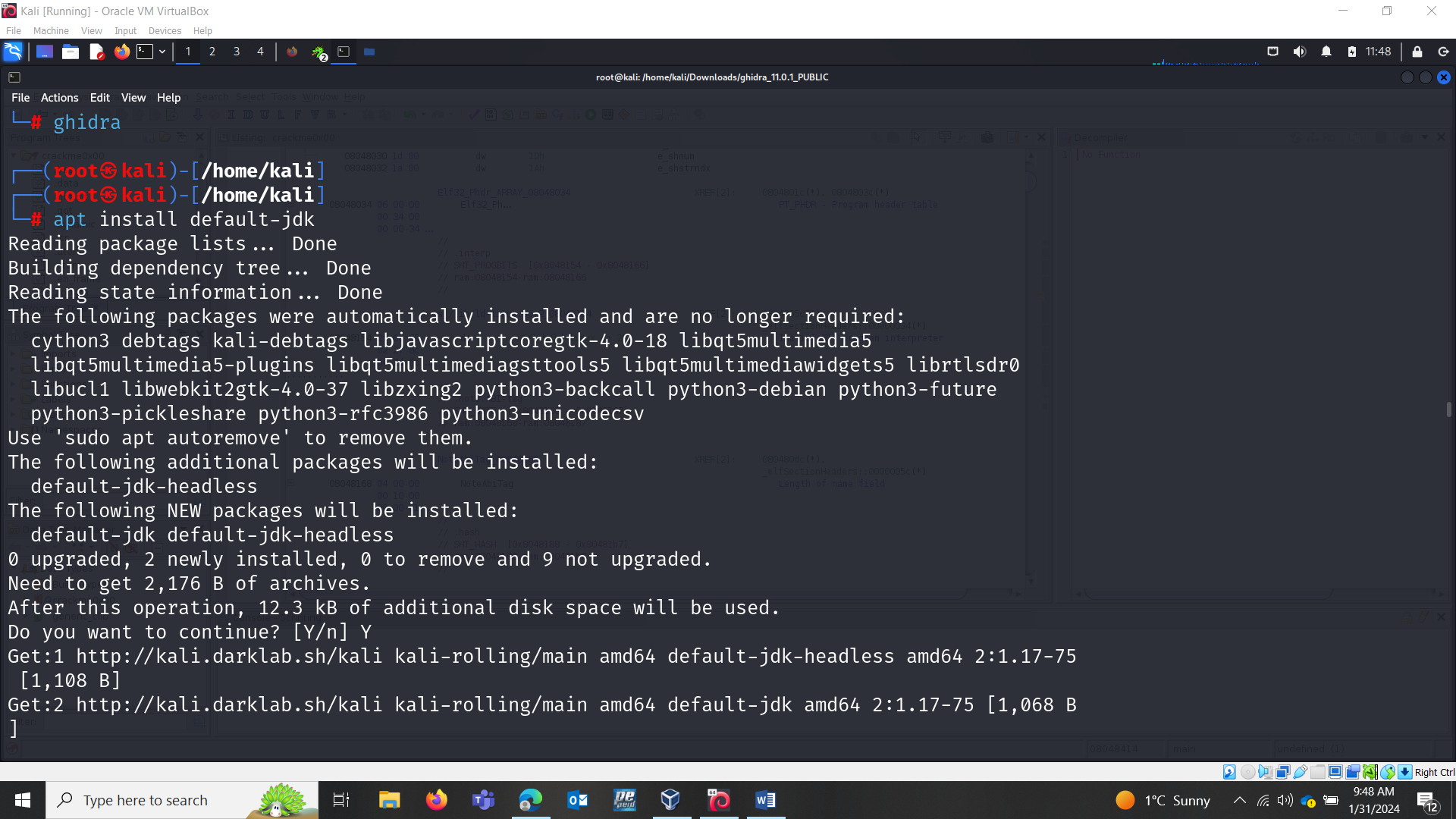




* Permissions;



DEPENDENCY: jdk



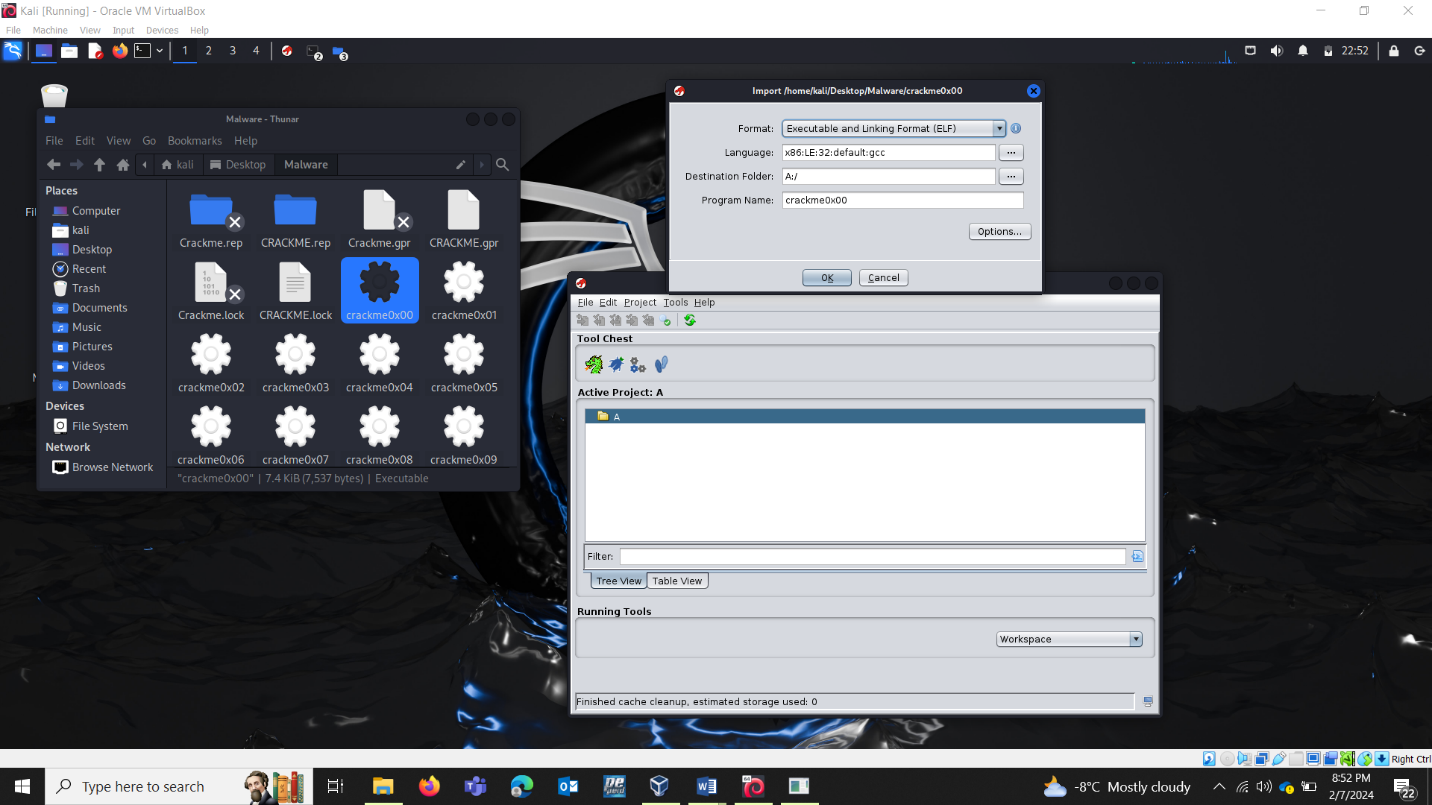
**GHIDRA OVERVIEW:**



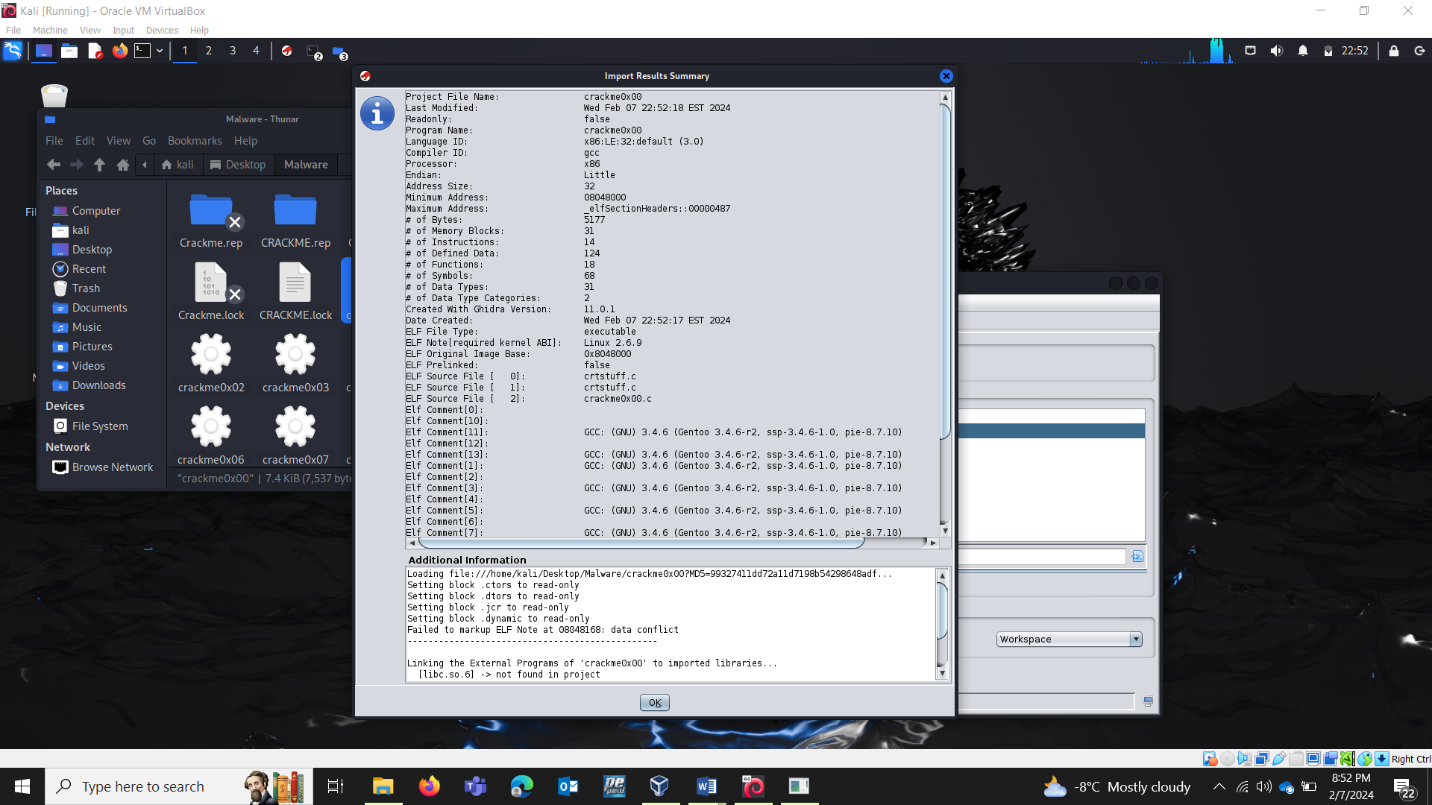
You can open a new project or existing project.

FIRST ANALYSIS:

* Create a new project.
* Drag and drop the file which you want to analyse in it.



* Double click on the program which you want. It will give you a brief summary.



You can select analyzers.

BASIC FUNCTIONS:

* Binary windows
* Graph
* Decompiler
* Script manager
* If you press F1, it will give you help page.
* Program tree will give you starting points of assembly code/binary.
* If you modularize your executable by dominance, main function is easy to find.

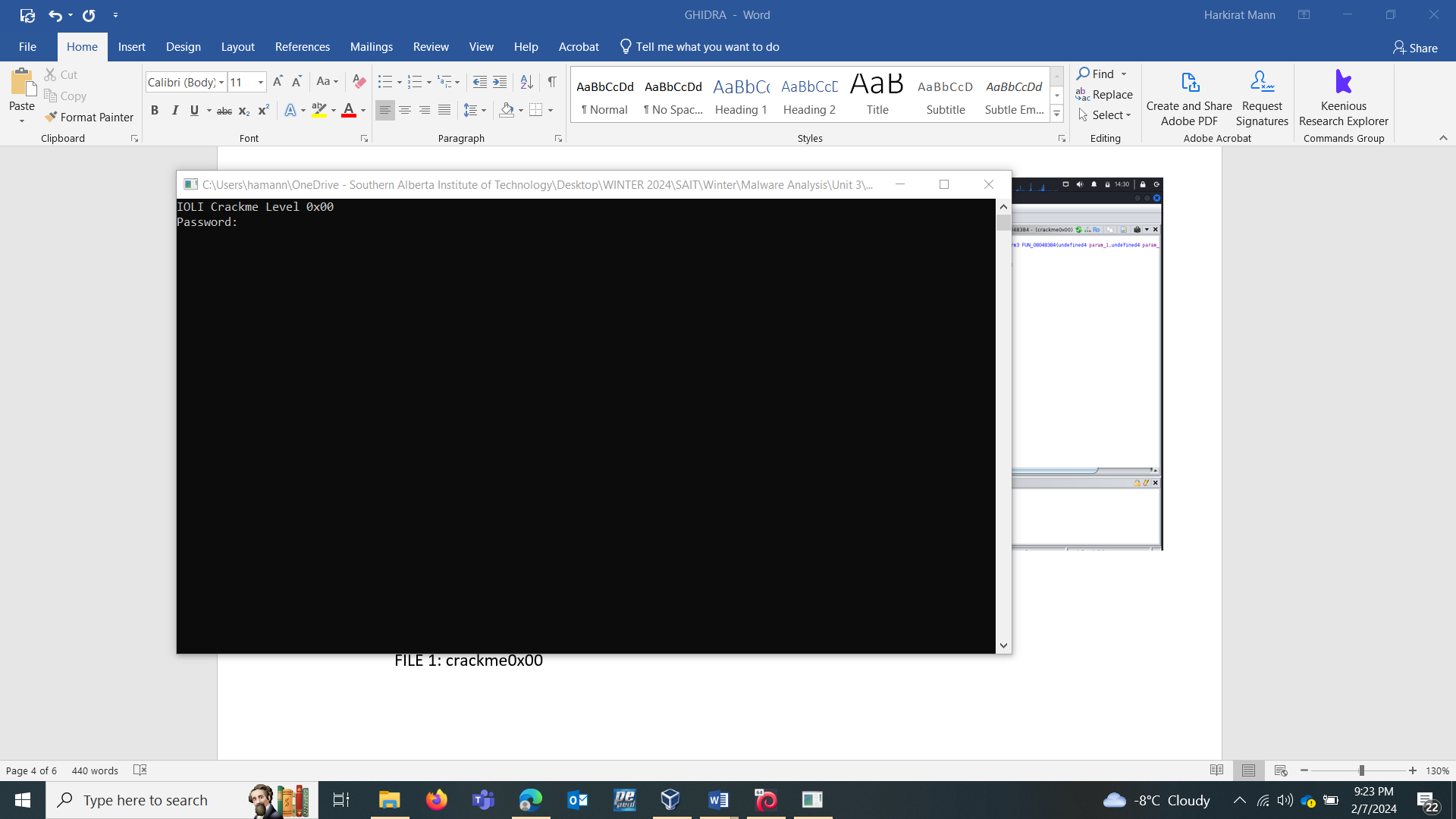
WORKSPACE OF GHIDRA:



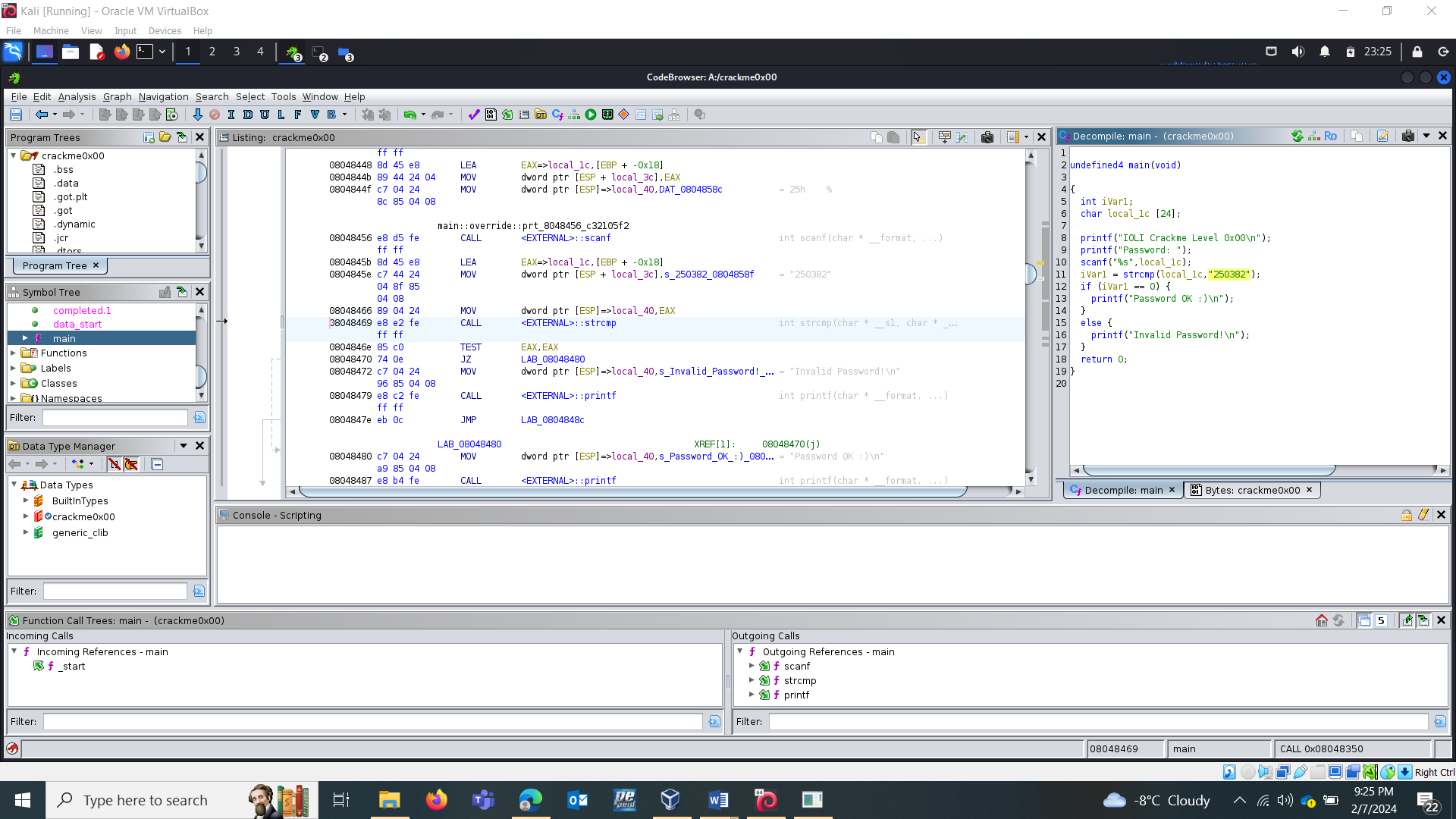
ANALYZING FILES:

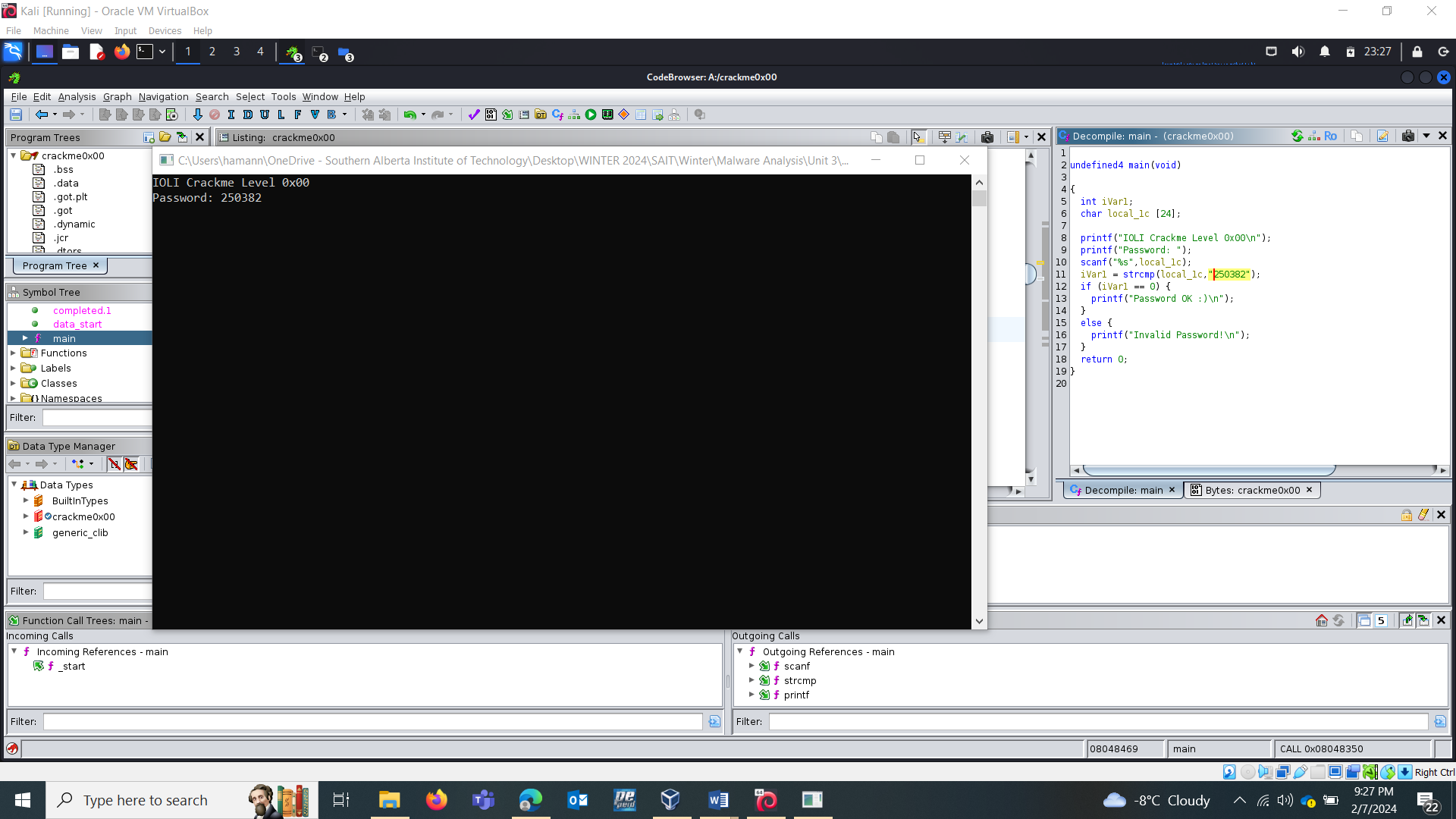
FILE 1: crackme0x00

It asks for a password.



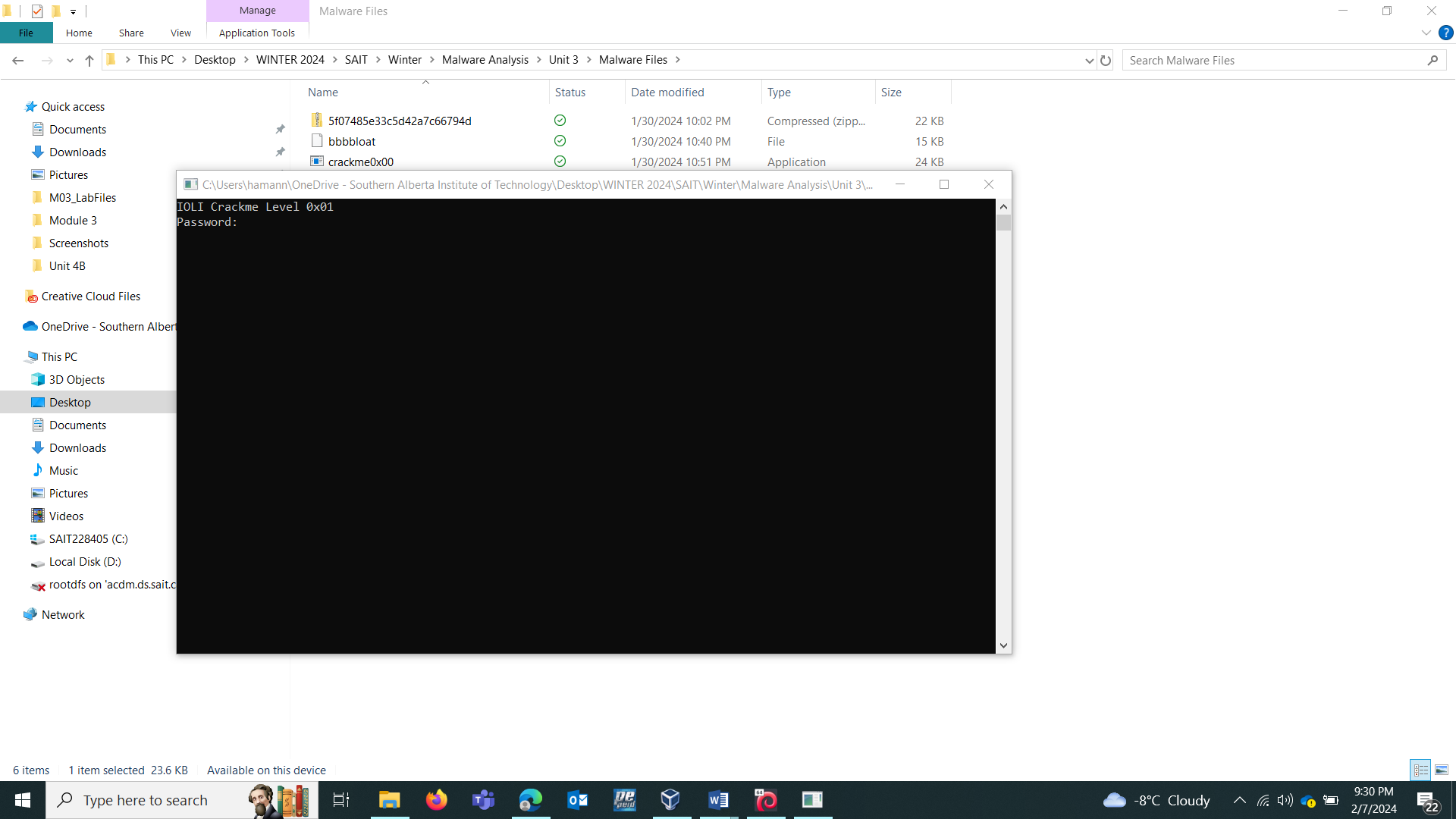
ANALYSIS:

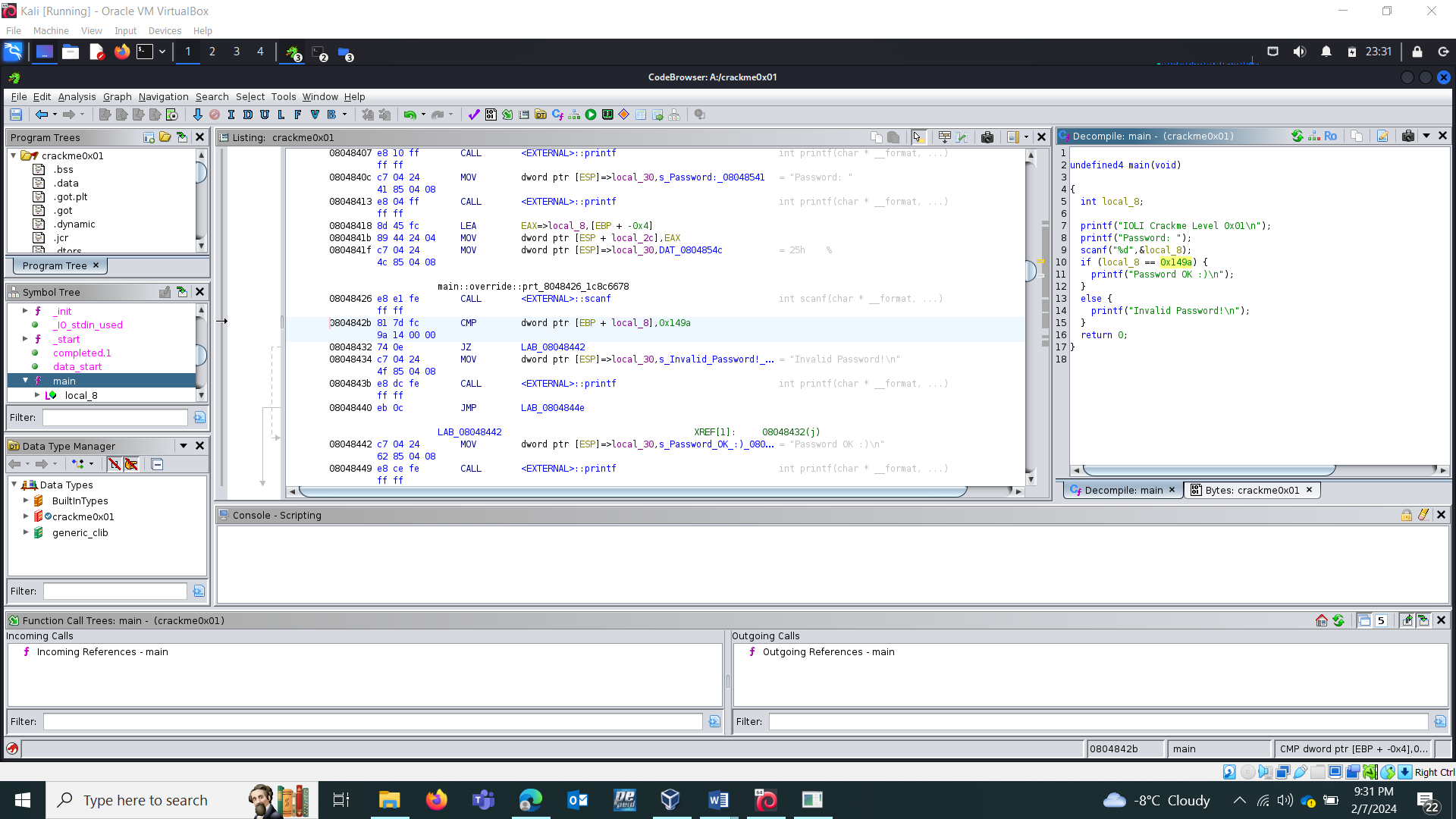




ANS : 250382

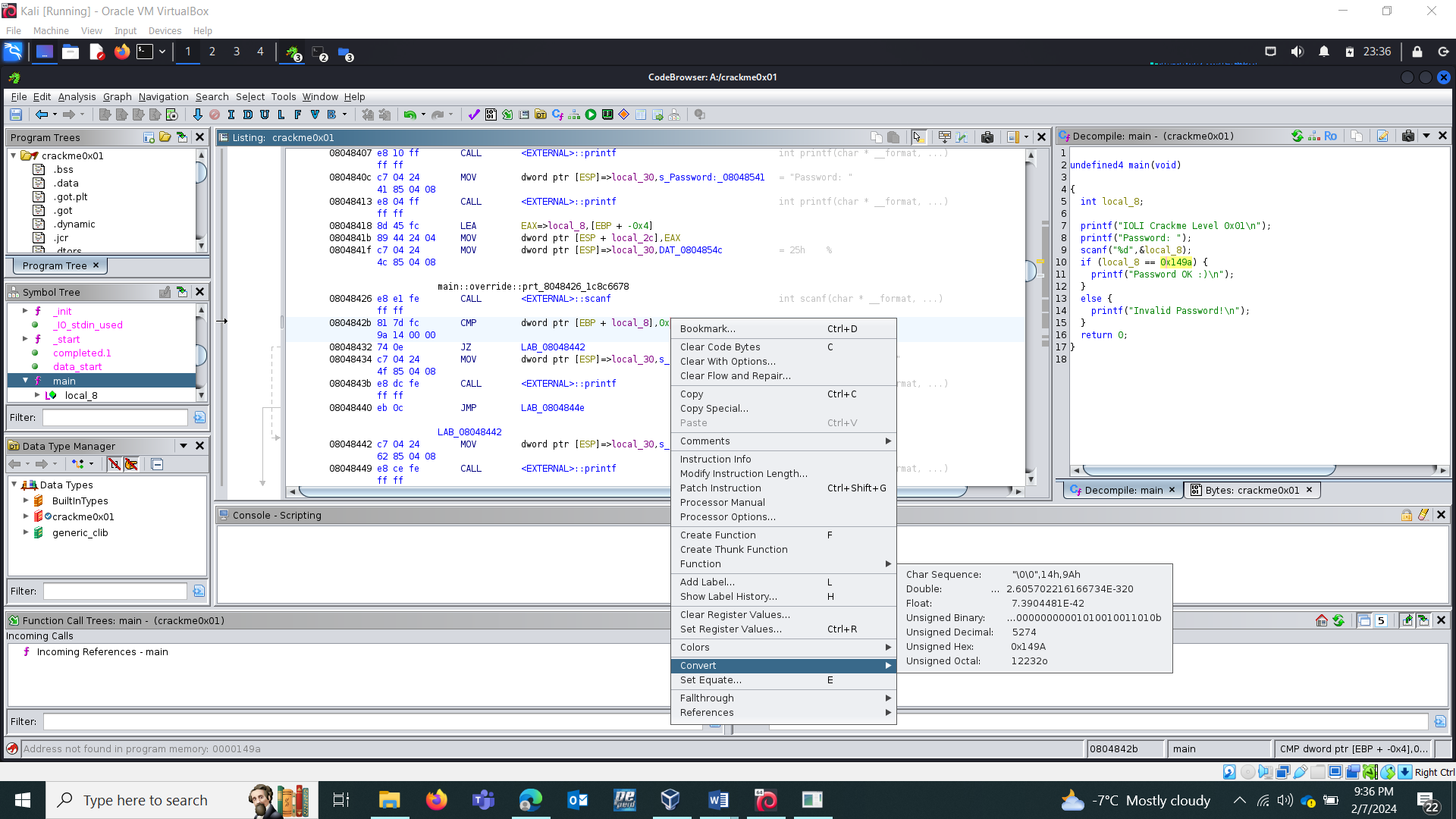
FILE 2: crackme0x01





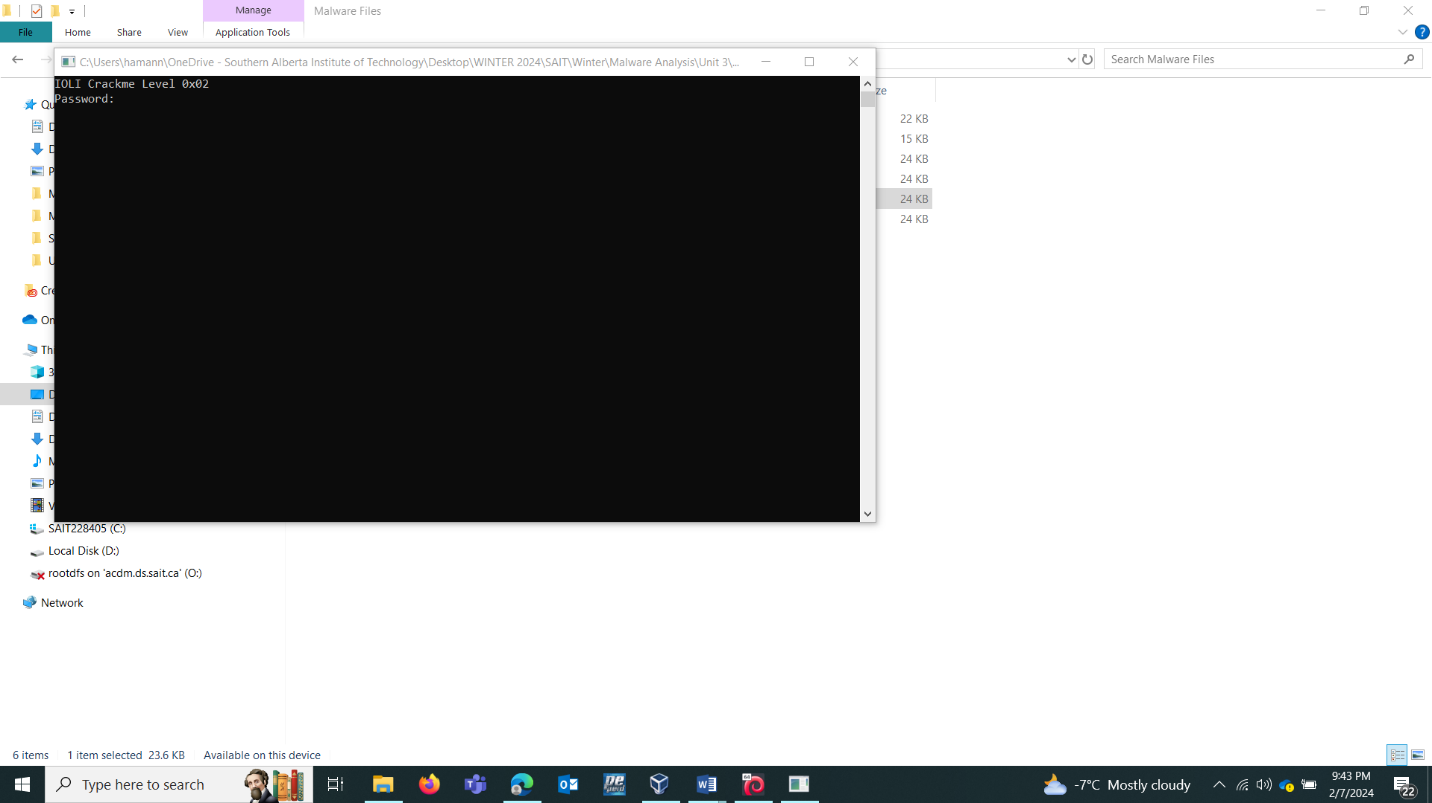
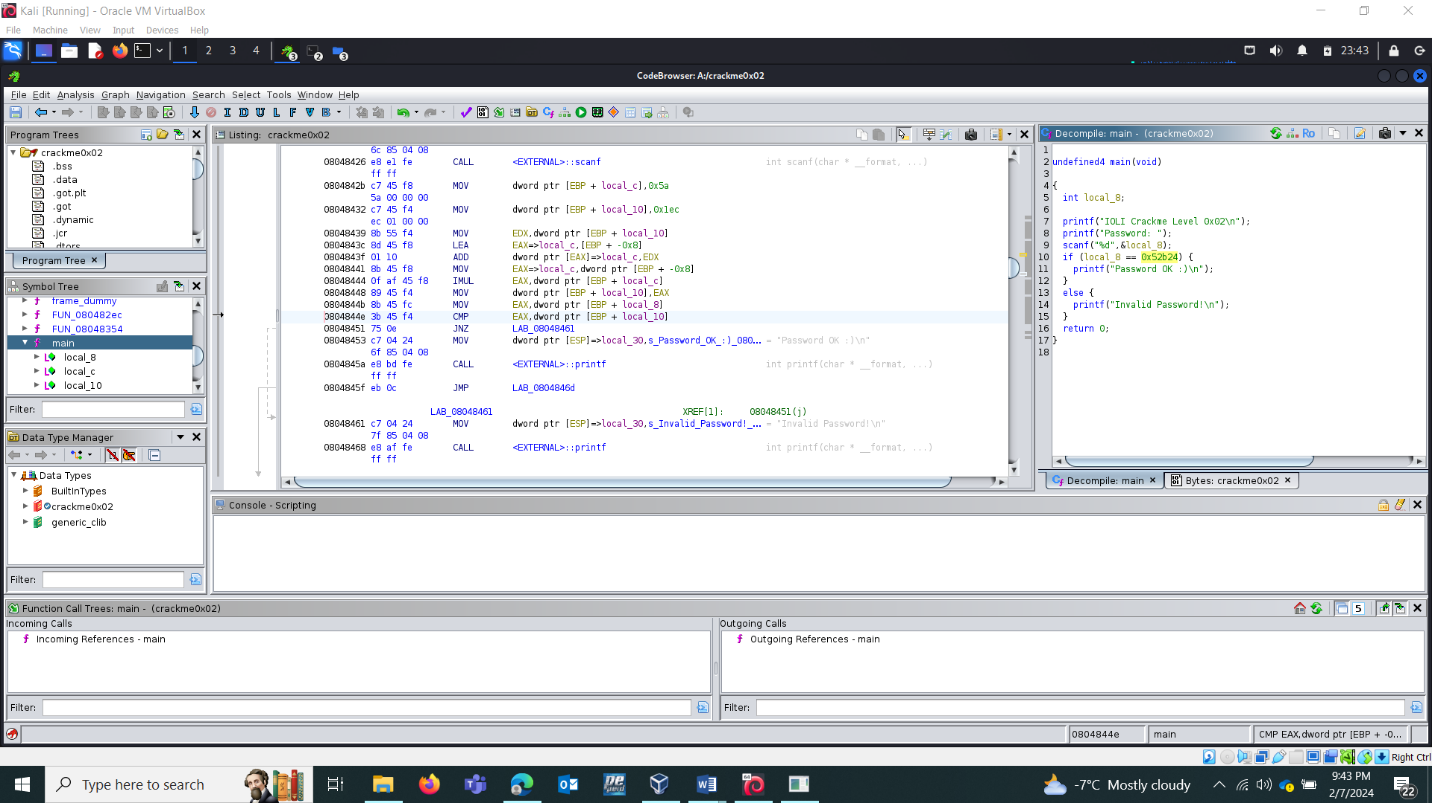
Try passing 0x149a

Try converting hex to decimal.



Find the answer!

ANALYSIS 3: crackme0x02



HINT: Hover over!

# **Ghidra Additional Setup**

## **This is important to complete the lab with the least number of headaches.**

When performing malware analysis, it is sometimes required but often challenging to return the code to the original state prior to compilation. That is unless you have the debug files included along with the executable. In Windows, Visual Studio IDE can create a .PDB file that can be distributed with the executable. In Linux the equivalent is embedded into the executable. These files contain some important information referred to as symbols.

### **What are symbols?**

Fundamentally they are primitive identifiers associated with the functions, arguments, constants and other components that were used in the original source code. If you would like a bit more details see the following links:

<https://en.wikipedia.org/wiki/Symbol_(programming)>

<https://en.wikipedia.org/wiki/Symbol_table>

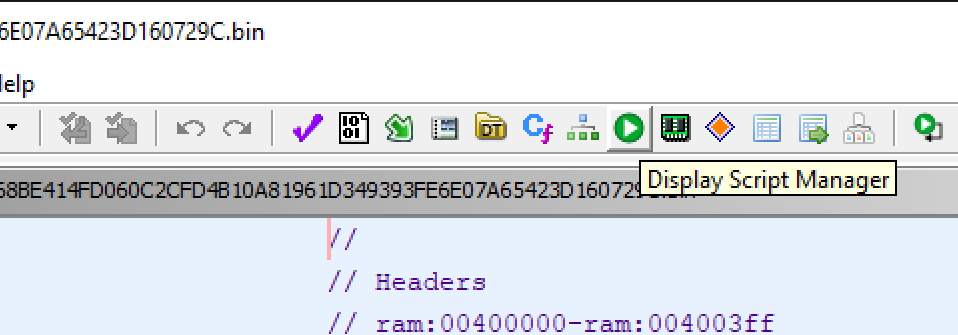
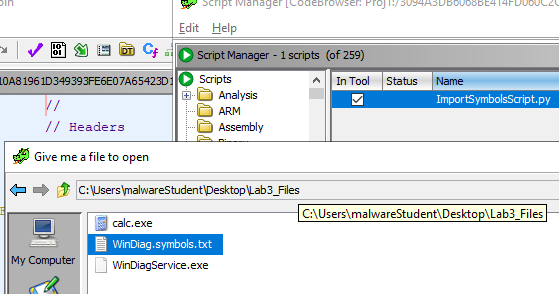
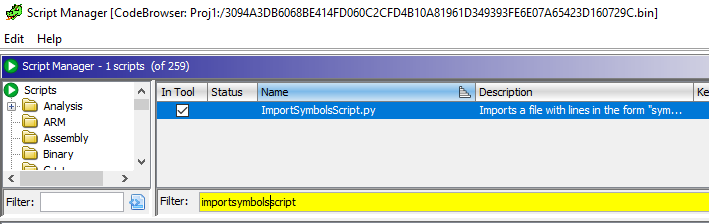
### **Why are symbols important?**

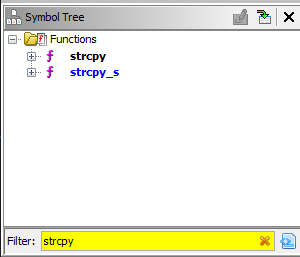
Malware authors strip the symbols from the final executable, before deployment. Without them the process of decompilation is difficult and time consuming. This is true despite using powerful and capable tools like Ghidra and IDA Pro – to a lesser extent.

### **How to resolve some symbols for commonly known functions?**

Ghidra doesn’t always resolve all symbols in a PE file correctly. It appears that the appropriate DLL don’t always load correctly and therefore the API/functions don’t get recognized properly or at all. In these cases, you will need to give Ghidra a hand to resolve the names of Library APIs and functions.

If you don’t do the following you will likely spend/waste time evaluating functions that are not part of the malware author’s code.

1. After you have renamed the WinDiagService.exe file as requested in the lab; open Ghidra with the ***<sha256hash>.bin*** file.
2. Open the sample in the Listing Window
3. Click on the Script Manager
4. In the filter field shown below, type: **ImportSymbolsScript.py**
5. When asked select the **WinDiag\_symbols.txt** file
6. Click the button “**Go baby go!”**
7. To confirm this work search for strcpy as shown below in the Symbols Tree



1. Continue analyzing your code.