Author: L.Radhakrishnan
Reviewed By: KMV.Subramanyam

Pluto Obfuscator

Pluto obfuscator:-

- *Pluto is an **open-source** code obfuscator based on the **LLVM** compiler infrastructure. It is designed to protect software from being **reverse engineered** or **cracked** by making the code harder to understand and analyze. Obfuscation works by transforming the code into a functionally equivalent but more complex form, making it difficult for attackers to extract the original logic or functionality.*

Purpose:-

- *The purpose of this guide is to demonstrate how to obfuscate C code using **Pluto Obfuscator** on Ubuntu 20.04, targeting the **x86_64** architecture*

Pluto Obfuscator Installation and Usage Guide

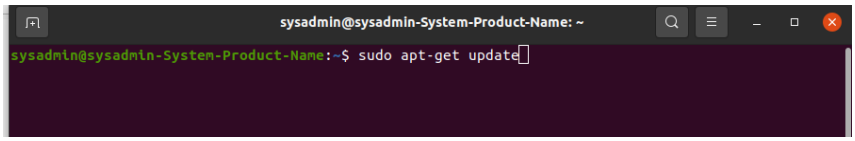
Prerequisites:-

- Ubuntu 20.04.3 LTS
- CMake 3.16.3
- Ninja 1.10.0
- Git
- LLVM 12.0.1 and clang 12.0.1 (optional)

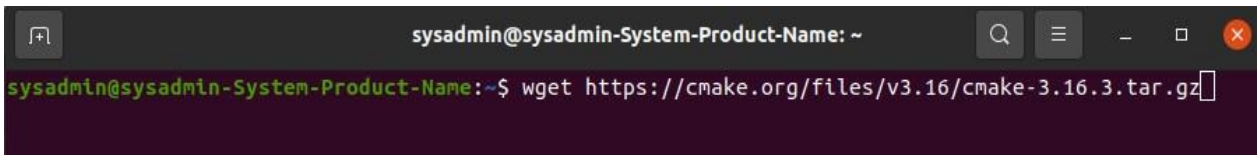
I. Installing CMake 3.16.3 on Ubuntu 20.04
what is cmake?

- CMake is an open-source cross-platform **build system** that manages the build process in a **compiler-independent manner**. It generates build files such as **Makefiles** or project files for various **Integrated Development Environments** (IDEs) based on a simple script format.With CMake, developers can define their build process in a **clear** and **platform-independent way**, making it easier to maintain and port projects across **different platforms** and **development environments**.

Step 1: Update Package Lists



```
sudo apt-get update
```



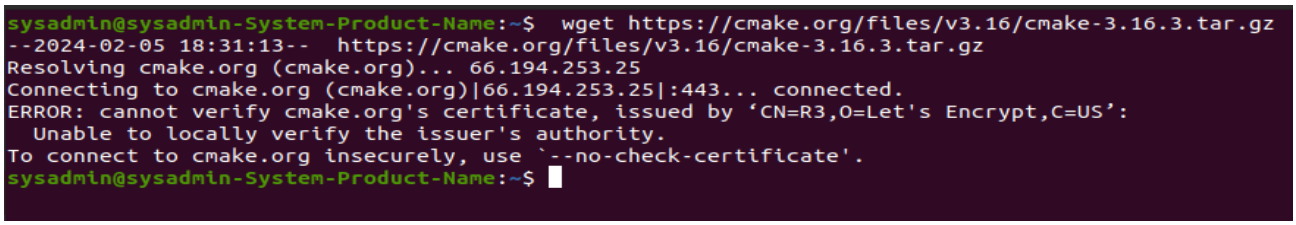
Step 2: Download CMake 3.16.3 Source Code

```
wget https://cmake.org/files/v3.16/cmake-3.16.3.tar.gz
```

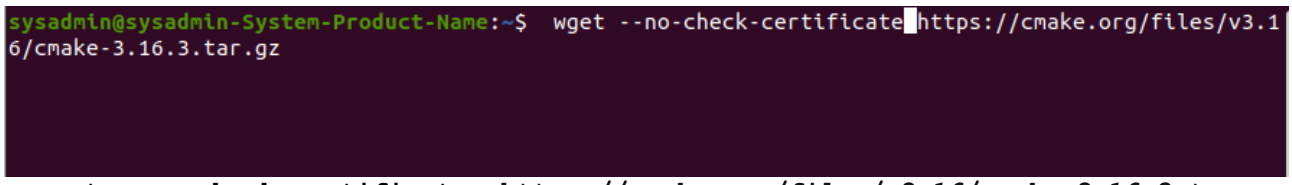
Note: When you get an error like this below, just take the following steps.

ERROR: cannot verify github.com's certificate, issued by 'CN=DigiCert TLS Hybrid ECC SHA384 2020 CA1,O=DigiCert Inc,C=US':

Unable to locally verify the issuer's authority.



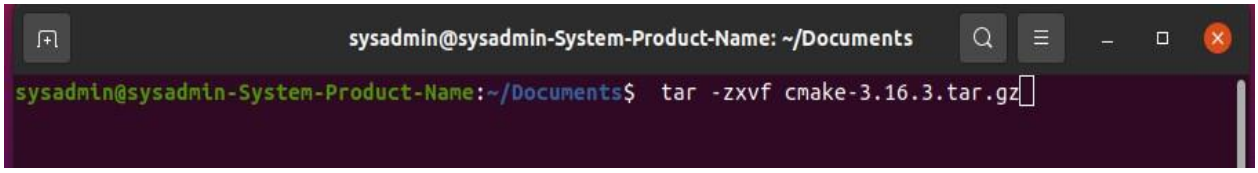
-->To resolve the issue just run the below command.




```
wget --no-check-certificate https://cmake.org/files/v3.16/cmake-3.16.3.tar.gz
```

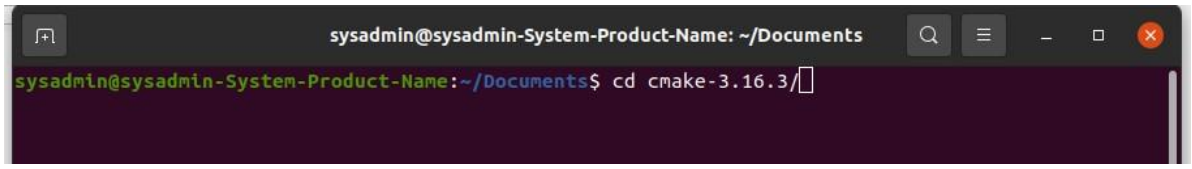
--no-check-certificate -> if you encounter SSL certificate verification issues while using wget, you can bypass certificate checking by using the --no-check-certificate option

Step 3: Extract the Archive



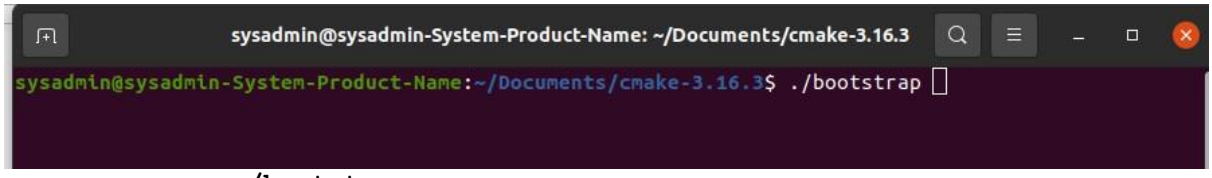

```
tar -zxvf cmake-3.16.3.tar.gz
```

Step 4: Navigate to the CMake Directory



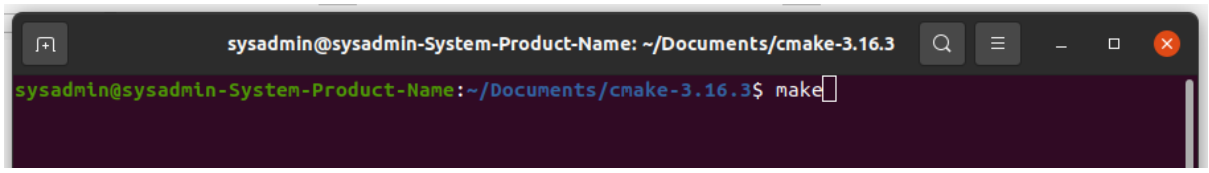

```
cd cmake-3.16.3
```

Step 5: Run the Bootstrap Script



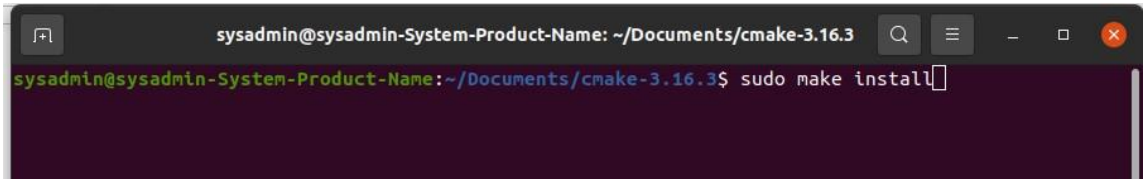

```
./bootstrap
```

Step 6: Build Cmake



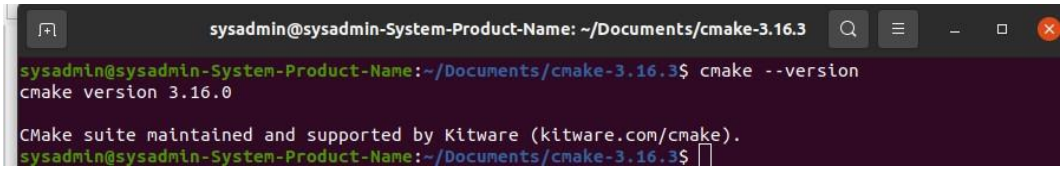

```
make
```

Step 7: Install Cmake




```
sudo make install
```

Step 8: Verify the Installation




```
cmake --version
```

Author: L.Radhakrishnan
Reviewed By: KMV.Subramanyam

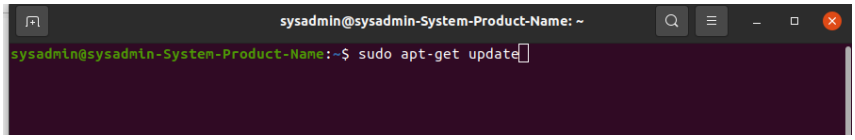
This should display the version number of CMake, confirming the successful installation.

II. Installing Ninja 1.10.0 on Ubuntu 20.04

what is Ninja ?

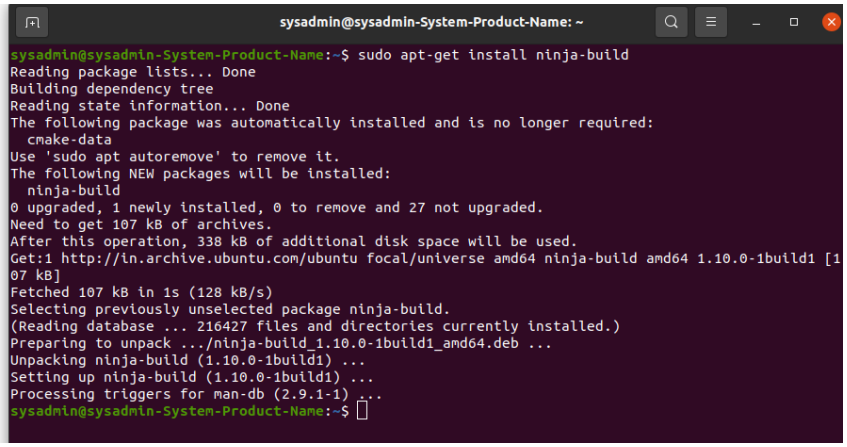
- Ninja is a fast and efficient **build system** designed for **speed** and **scalability.** It is commonly used as a replacement for other build tools like **Make** due to its superior performance. Ninja utilizes a simple build file format and parallelizes builds to take full advantage of modern hardware, making it particularly well-suited for **large projects** with complex dependencies.

Step 1: Update Package Lists



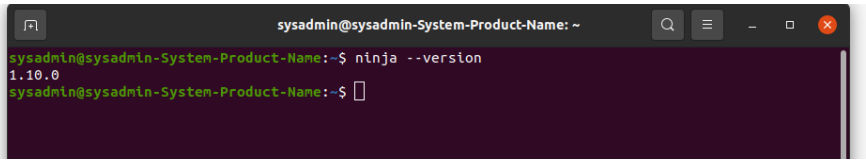

```
sudo apt-get update
```

Step 2: Install the Ninja Build



sudo apt install ninja-build

Step 3: Verify the Installation



```
ninja --version
```

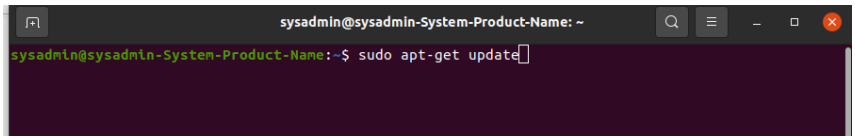
*This should display the version number of Ninja , confirming the successful installation.

III. Installing Git on Ubuntu 20.04

what is git?

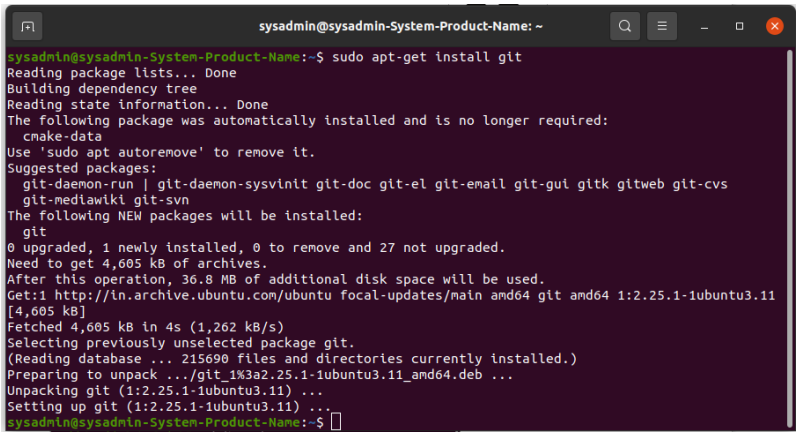
- Git is a **version control system** designed to track changes in files and coordinate work among multiple people. It allows developers to efficiently collaborate on projects by **managing revisions**, **branching**, **a**nd **merging** of code. With Git, users can track the history of changes, revert to previous versions, and work concurrently on different parts of a project. It has become a fundamental tool in software development, enabling teams to effectively manage and maintain codebases of any size.

Step 1: Update Package Lists



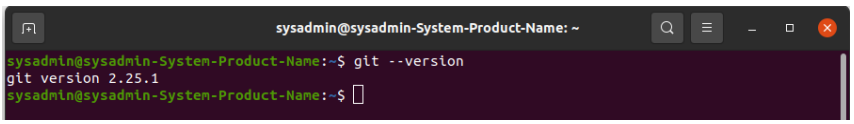

```
sudo apt-get update
```

Step 2: Install the Git




```
sudo apt-get install git
```

Step 3: Verify the Installation



git --version

IV. LLVM and Clang

what is LLVM ?

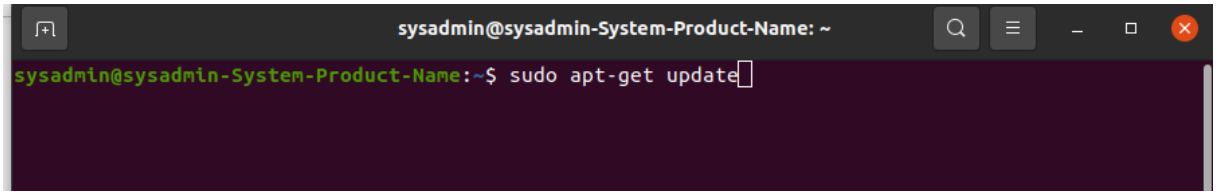
- LLVM is a versatile compiler infrastructure project initially developed at the University of Illinois. It offers a suite of tools and libraries for building compilers, code analyzers, and other programming tools. Known for its modular architecture and intermediate representation (IR), LLVM facilitates efficient code generation and optimization across different programming languages and platforms

what is clang?

- Clang is a high-performance compiler front end for C, C++, and Objective-C languages, part of the LLVM project. It's known for its speed, adherence to language standards, and advanced features like enhanced diagnostics and integration with development tools.

V. Download the source code from github

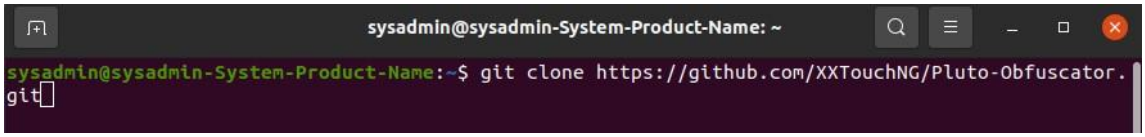
Step1: Update Package Lists



sudo apt-get update

Step1: Clone a Source Code from github:

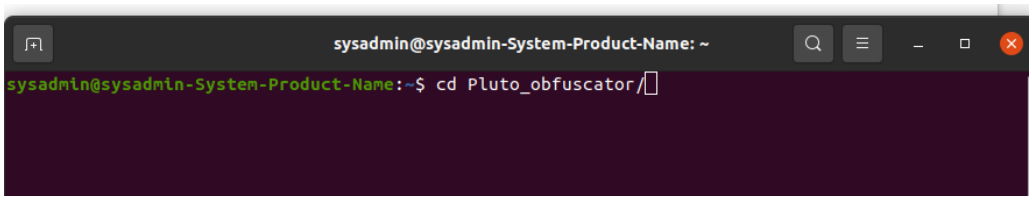
Clone the Pluto Obfuscator repository from GitHub by using below command in terminal:



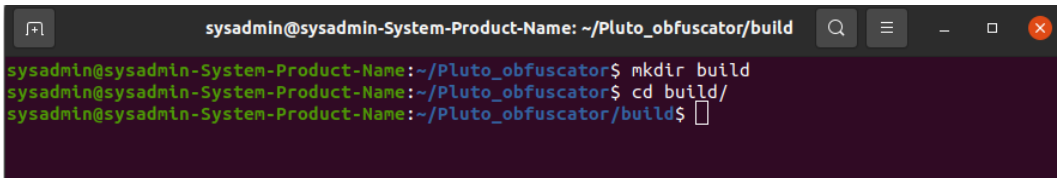
git clone https://github.com/XXTouchNG/Pluto-Obfuscator.git

Step2: Build Directory:

Create a build directory within the Pluto Obfuscator source directory:



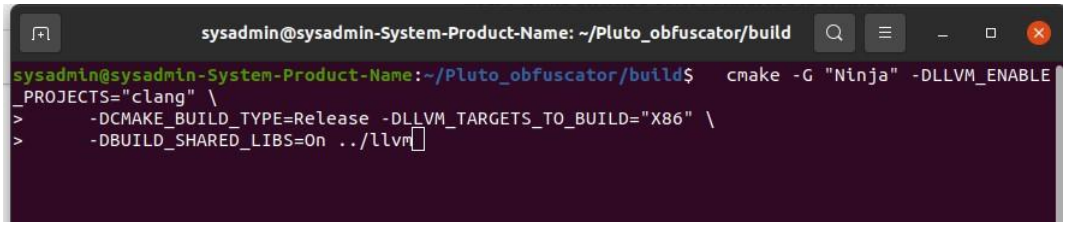
cd Pluto_obfuscator



mkdir build
cd build

Step3: Generate Build System:

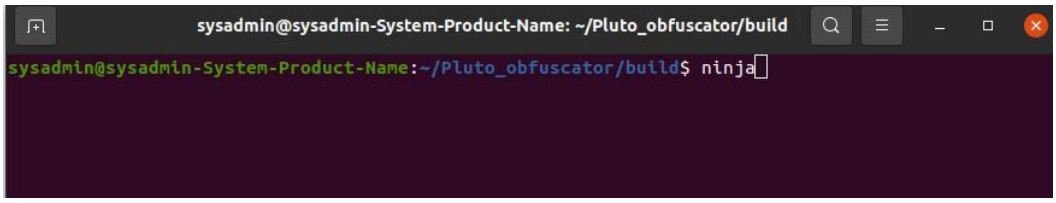
Configure the build system with Cmake:



cmake -G "Ninja" -DLLVM_ENABLE_PROJECTS="clang" \
-DCMAKE_BUILD_TYPE=Release -DLLVM_TARGETS_TO_BUILD="X86" \
-DBUILD_SHARED_LIBS=On ../llvm

Step4: Build Pluto Obfuscator:

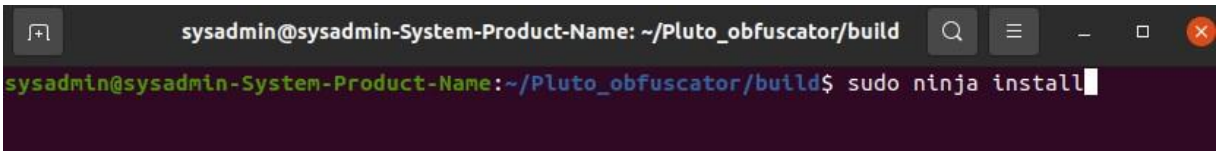
Build the source code using Ninja:



ninja

Step5: Install Pluto Obfuscator:

Install the Pluto Obfuscator binaries as root:

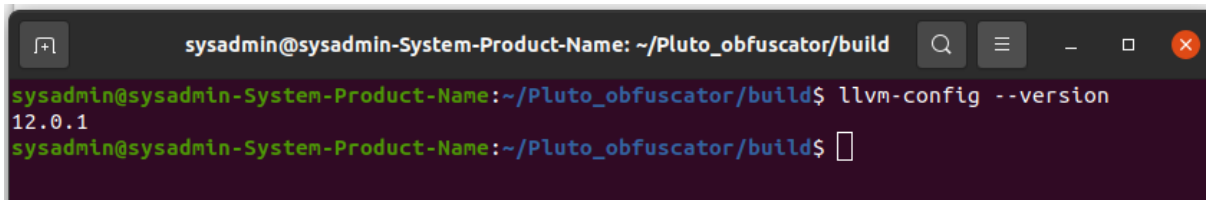


sudo ninja install

VI. Verify:

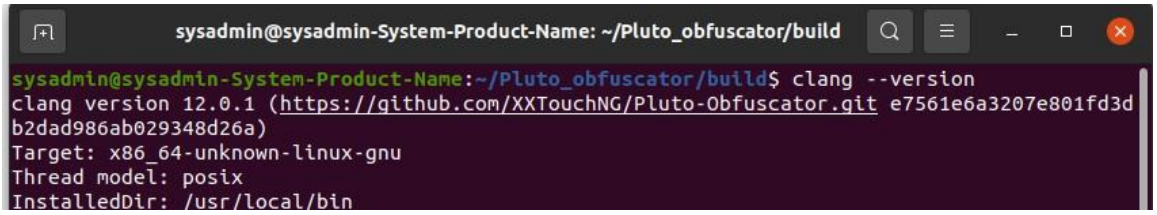
Verify installation by checking the version of llvm-config and clang:

i) verify the llvm installation



llvm -config --version

ii) verify the clang installation



clang --version

*The output will tell you if the tools are installed and their versions.

Author: L.Radhakrishnan
Reviewed By: KMV.Subramanyam

Features:

- Control Flow Flattening
- Bogus Control Flow
- Instruction Substitution
- Random Control Flow
- Variable Substitution
- String Encryption
- Globals Encryption
- MBA Obfuscation

Obfuscation Flags:

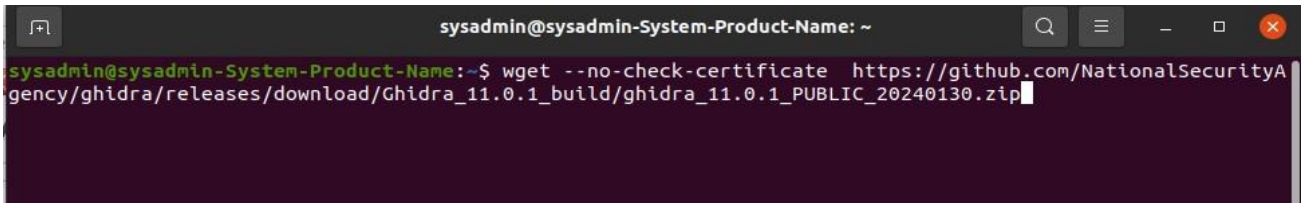
- Flattening: `-O2 -mllvm -fla`
- BogusControlFlow: `-O2 -mllvm -bcf`
- Substitution: `-O2 -mllvm -sub`
- GlobalsEncryption: `-O2 -mllvm -gle`
- MBAObfuscation: `-O2 -mllvm -mba -mllvm -mba-prob=100`
- FullProtection (**HIGHLY RECOMMENDED**): `-s -mllvm -mba -mllvm -mba-prob=50 -mllvm -fla -mllvm -gle`

Ghidra

- **Ghidra** is an open-source software **reverse engineering** suite developed by the **NSA**(National Security Agency). Released in 2019, it provides tools for analyzing **compiled code**, including **disassembly**, **decompilation**, and **debugging**. With a user-friendly interface, scripting support, and collaboration features, Ghidra is widely used in **cybersecurity** for tasks such as **malware analysis** and **vulnerability research.**

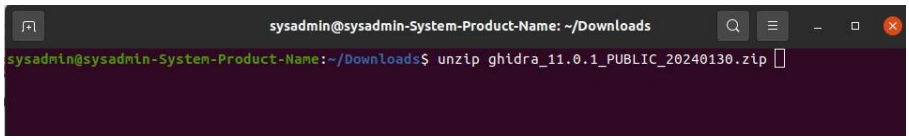
Install Ghidra:-

Step:1 Download Ghidra




```
wget --no-check-certificate
https://github.com/NationalSecurityAgency/ghidra/releases/download/Ghidra_11.0.1
_build/ghidra_11.0.1_PUBLIC_20240130.zip
```

Step:2 Extract the ZIP File:



- Open a terminal.
- Navigate to the directory containing the downloaded ZIP file.
- Run `unzip ghidra_11.0.1_PUBLIC_20240130.zip` to extract the contents.

Step:3 Navigate to Ghidra Directory:

- Move into the Ghidra directory using `cd ghidra_*`.

Step:4 Run Ghidra

- Launch Ghidra with **`./ghidraRun`** in the terminal.

Test the obfuscation

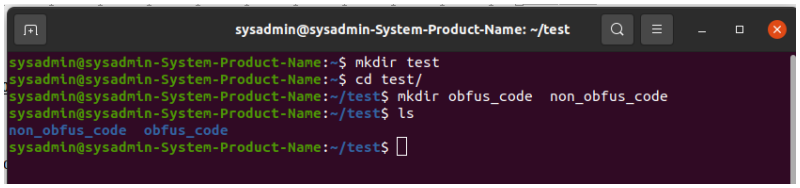
1. Create a C file:

Write a simple C program (e.g., test.c) with a main function printing "Hello, World!".

Author: L.Radhakrishnan
Reviewed By: KMV.Subramanyam

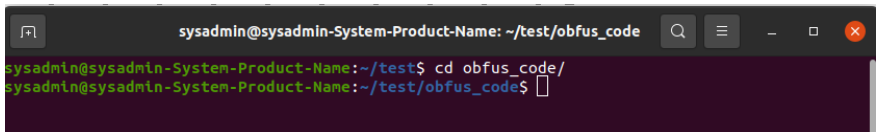
Open terminal and run the following command

Step:1 create a directory and two subdirectory

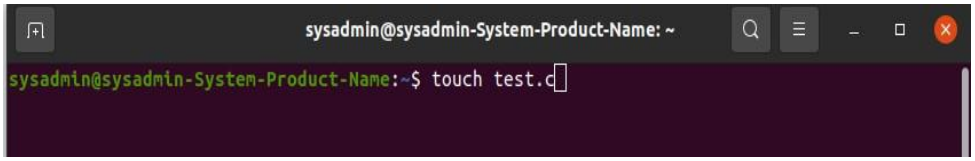


mkdir test
cd test
mkdir obfus_code nonobfus_code

Step:2 Create a c file in obfus_code directory

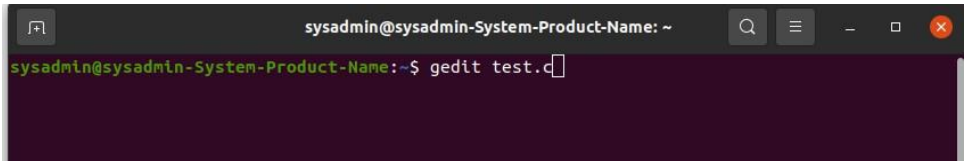


cd obfus_code



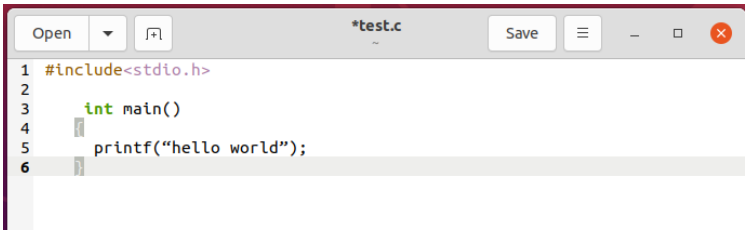
touch test.c

Step:2 Open a test.c file:-



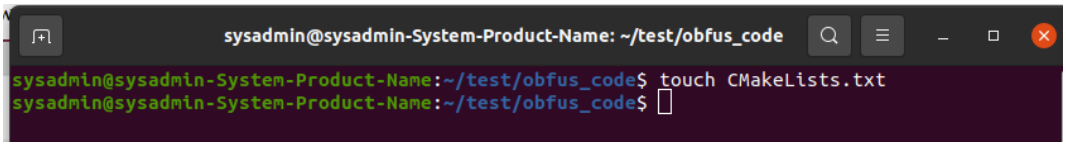
gedit test.c

Step:3 paste the below code and save it.

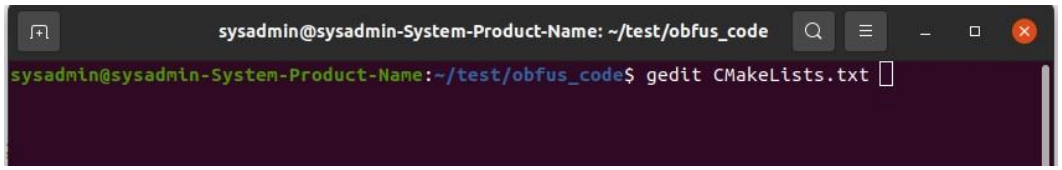


```
#include<stdio.h>

 int main()
 {
 printf("hello world");
 }
```

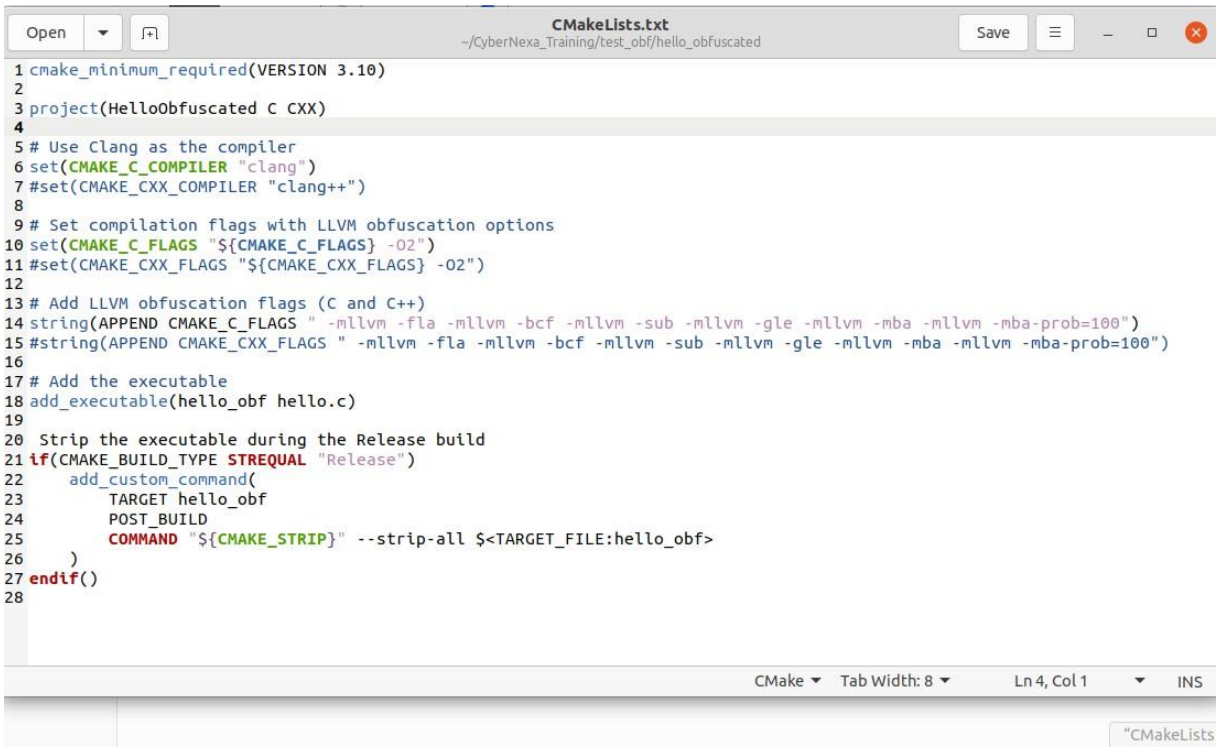
Step:3 Create a Cmake file



touch CmakeLists.txt

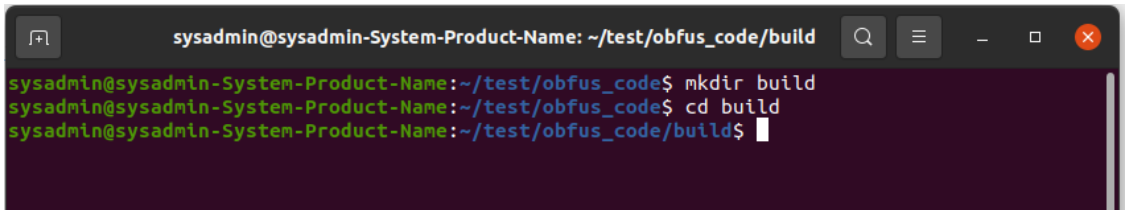


gedit CmakeLists.txt



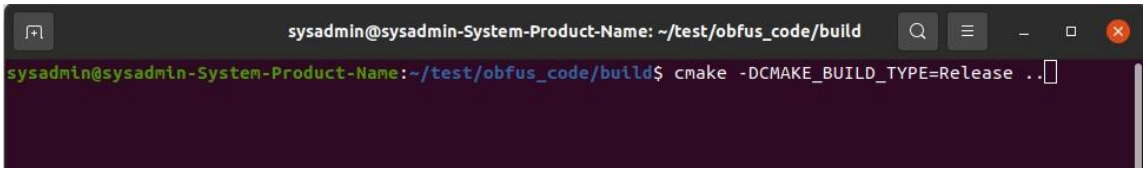
```
cmake_minimum_required(VERSION 3.10)
project(HelloObfuscated C CXX)

# Use Clang as the compiler
set(CMAKE_C_COMPILER "clang")
# Set compilation flags with LLVM obfuscation options
set(CMAKE_C_FLAGS "${CMAKE_C_FLAGS} -O2")

# Add LLVM obfuscation flags (C and C++)
string(APPEND CMAKE_C_FLAGS " -mllvm -fla -mllvm -bcf -mllvm -sub -mllvm -gle -mllvm -mba -mllvm -mba-prob=100")

# Add the executable
add_executable(hello_obf hello.c)

 Strip the executable during the Release build
if(CMAKE_BUILD_TYPE STREQUAL "Release")
    add_custom_command(
        TARGET hello_obf
        POST_BUILD
        COMMAND "${CMAKE_STRIP}" --strip-all $<TARGET_FILE:hello_obf>
    )
endif()
```

Step:4 Create a Build Directory



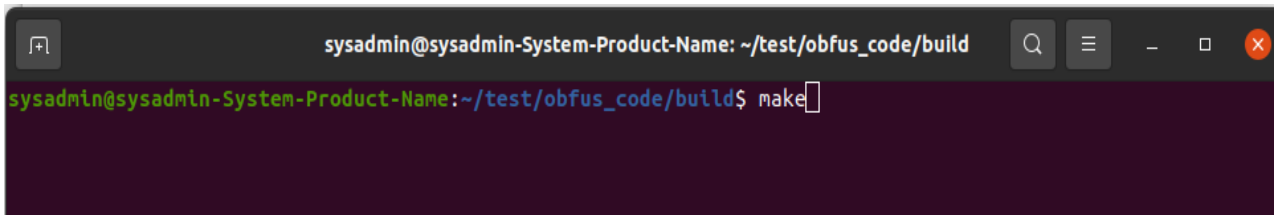
mkdir build
cd build

Step:5 compile the code



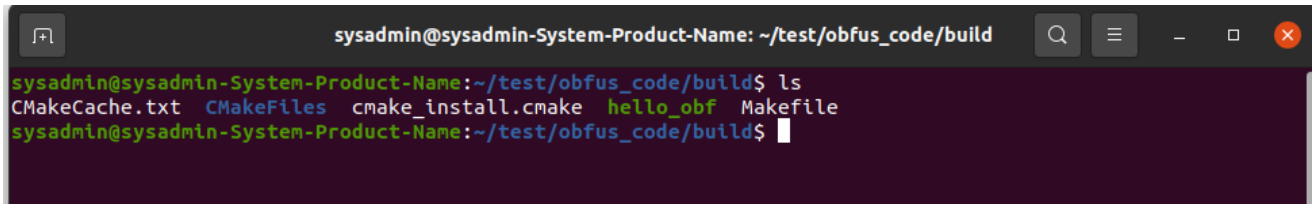

```
cmake -DCMAKE_BUILD_TYPE=Release ..
```

Step:6 Build



make

--> this will create a binary file hello_obf



Step:7 Verify Obfuscation:

Check if the test_obfs file was created after compilation.
Use various methods to verify if the code is obfuscated:

Strings: Use the strings command to see if readable strings are present in the compiled binary.

Ghidra: Tools like Ghidra can help analyze the assembly code for obfuscation indicators.

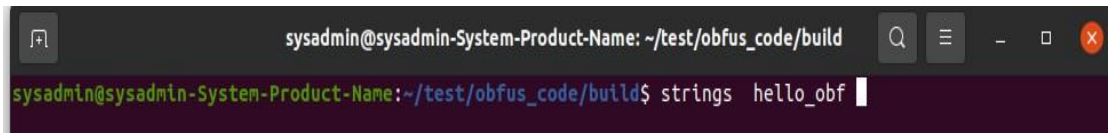
<u>Strings command</u>

- The **`strings`** command in Linux is used to **extract human-readable** text strings from **binary files**. These strings could be anything from printable characters, such as error messages or variable names, to more **sensitive information** like hardcoded **passwords** or **cryptographic keys.**

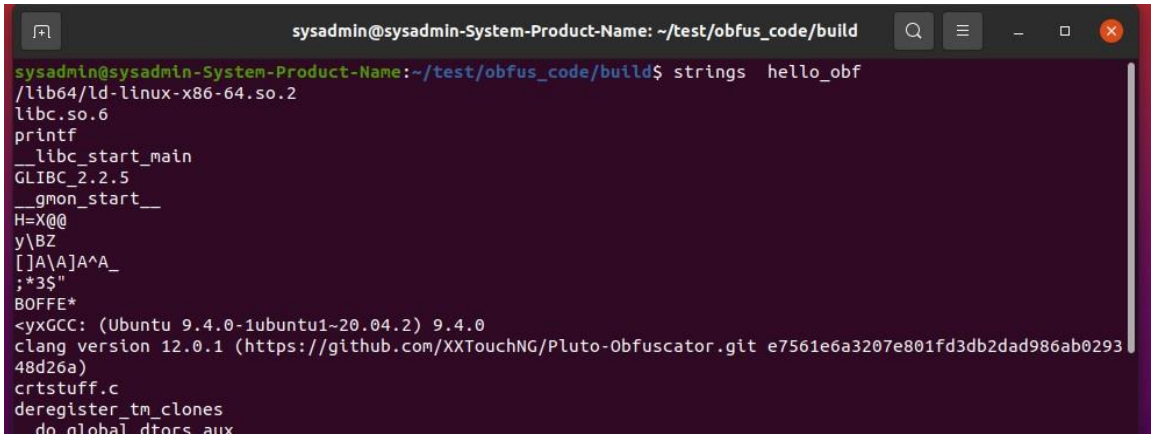
Here's how the `strings` command is typically used:

strings [options] filename

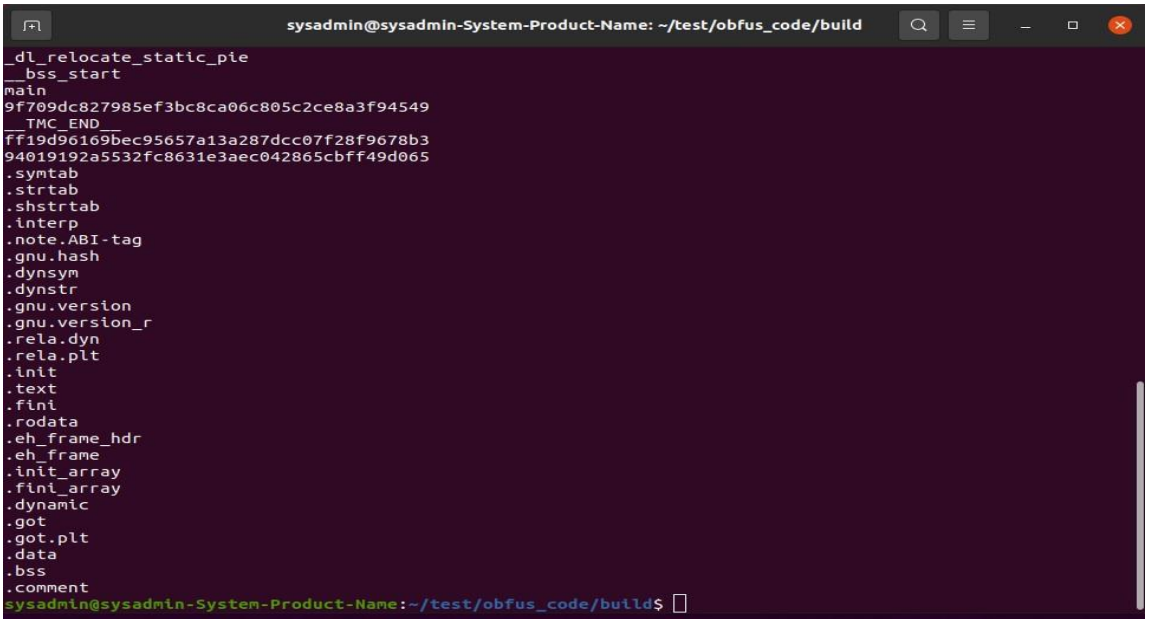
Let's we check with our hello_obf binary file



strings hello_obf



*The continuation of the terminal image is below.



***We can't see any **"hello world"** strings right,So we can ensure its obfuscated.
If you test this with non obfuscated binary, you can see the strings like "hello world".

Ghidra Tool:-

Step:1 Open the Ghidra tool

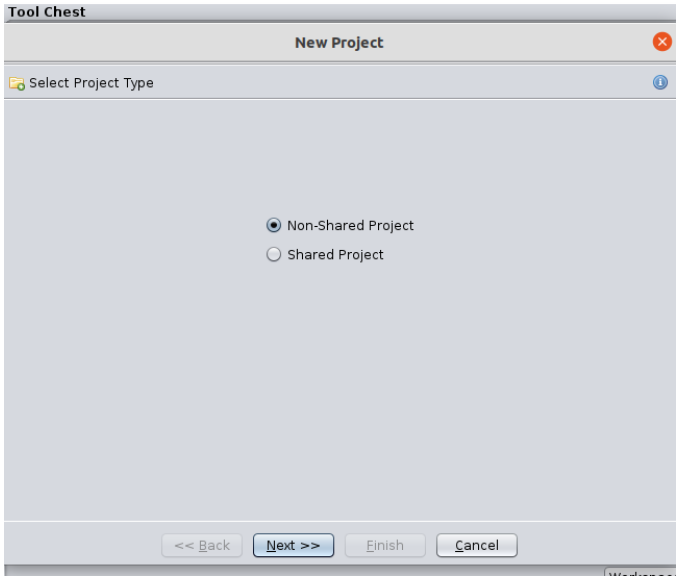
->Navigate to the Ghidra directory.

->Run the following command

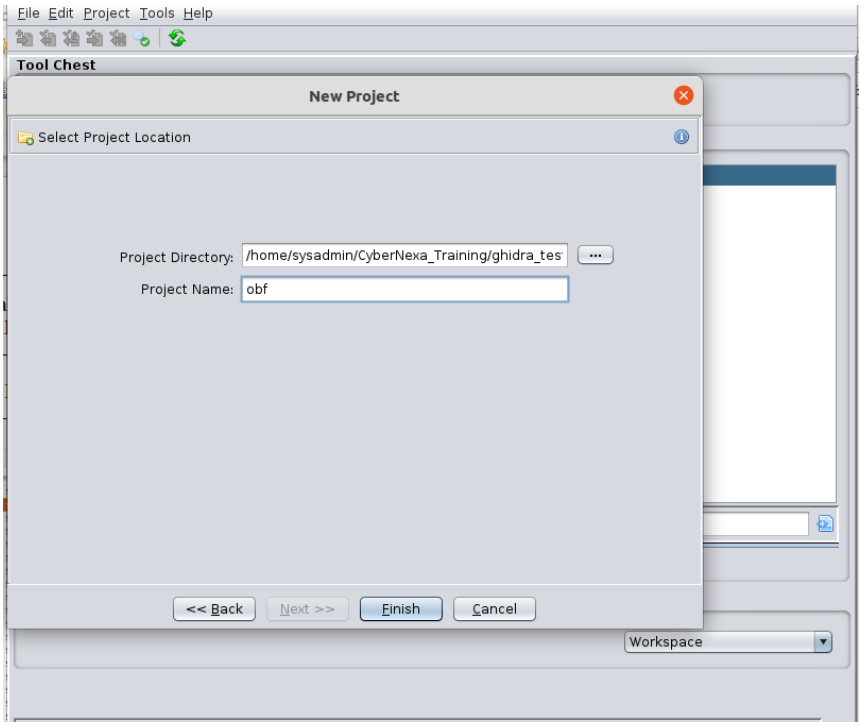
```
./ghidraRun
```

Step:2 Create a New Project

i) select project type

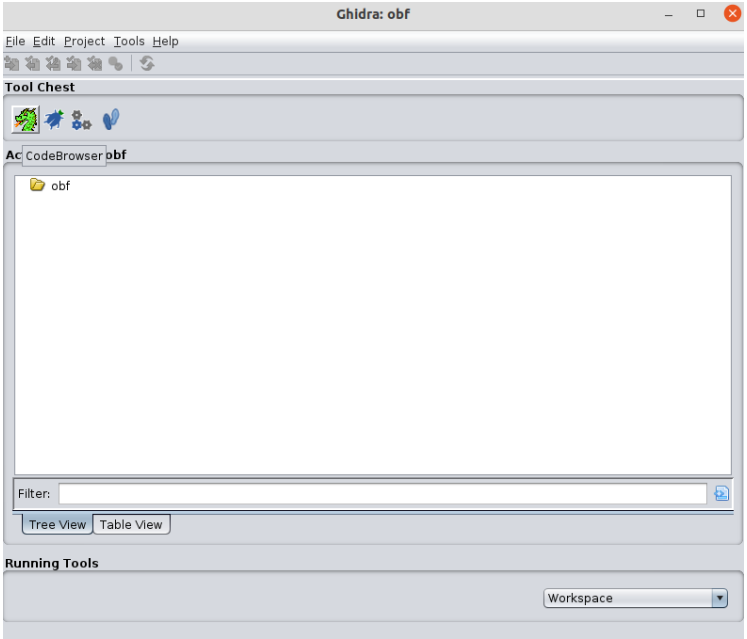


ii) Project name



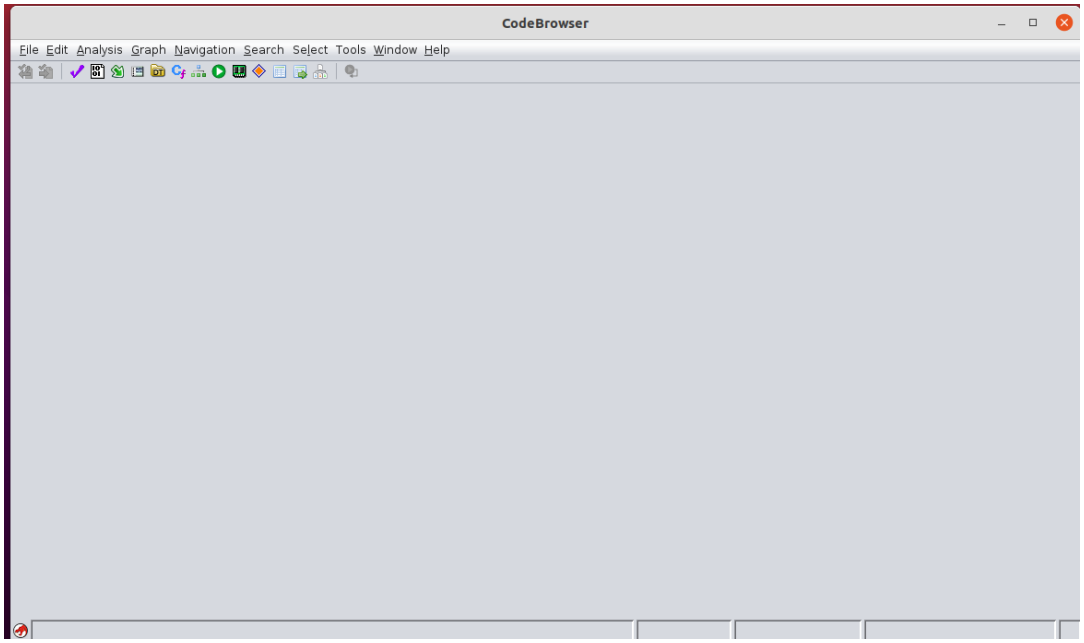
--> give a name for your project

iii) open code browser

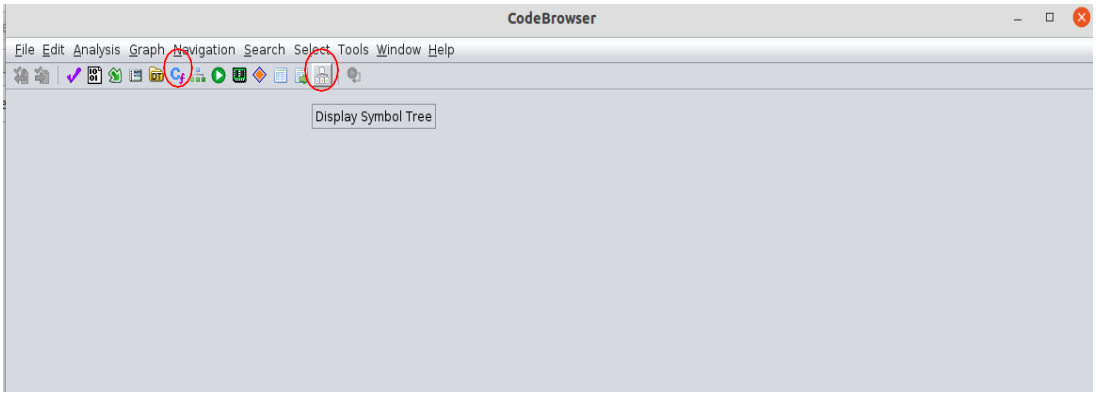


---> click the code browser button (dragon symbol).
few seconds after new window will be open.

Code browser windows

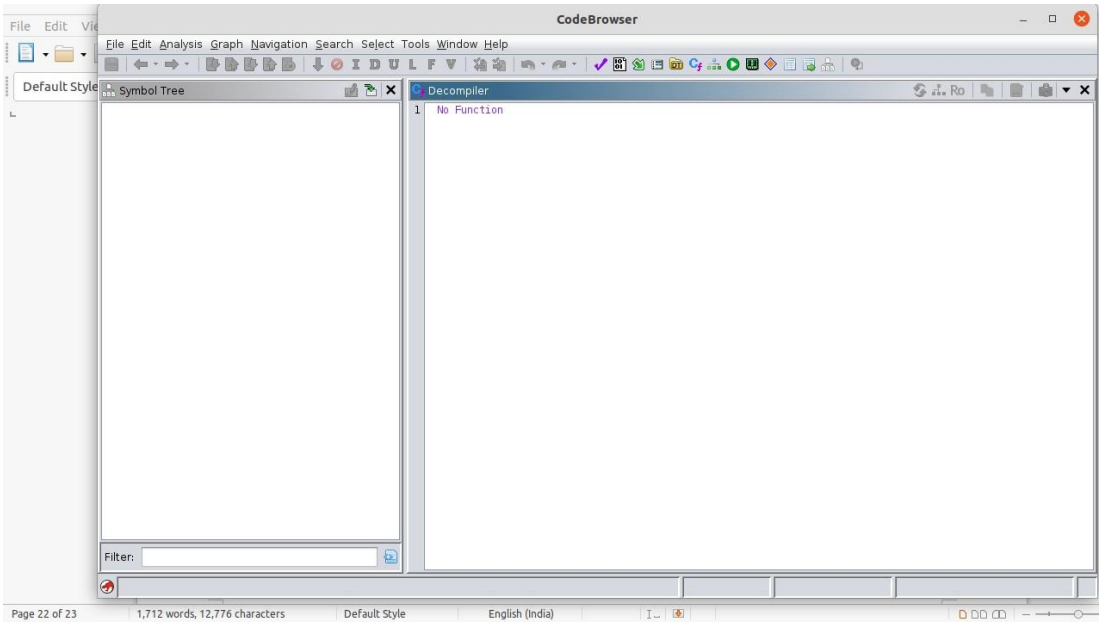


Author: L.Radhakrishnan
Reviewed By: KMV.Subramanyam



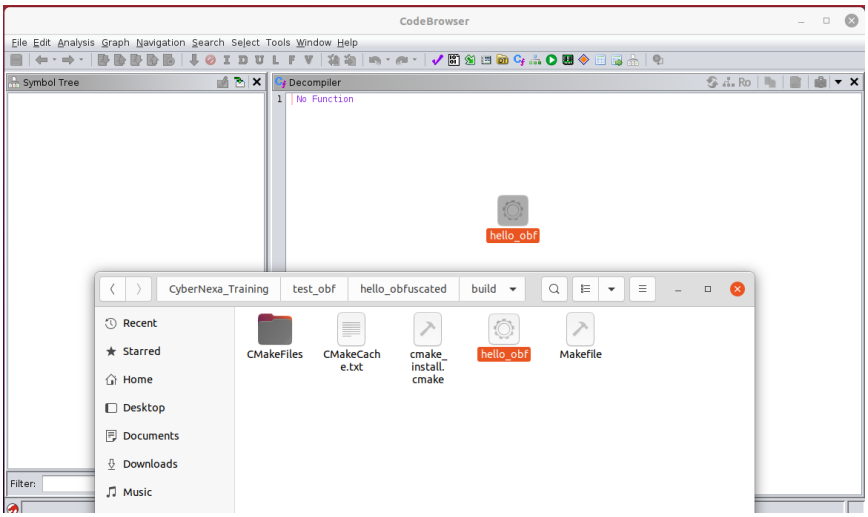
--->click the display symbol tree button
--->click the Display decompiler button

*** After click the above buttons you see the window like below**

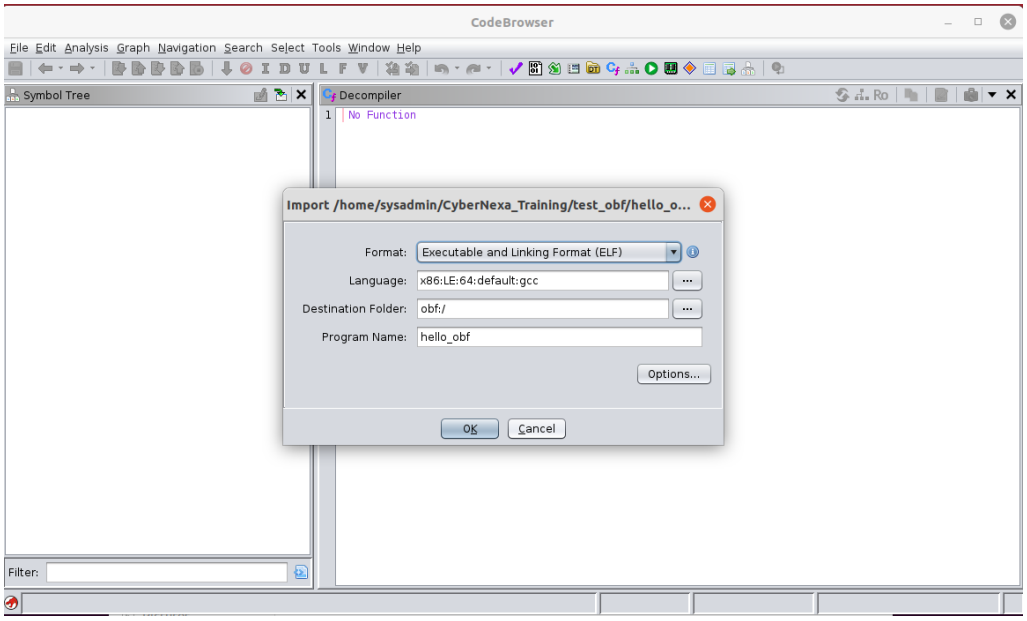


Author: L.Radhakrishnan
Reviewed By: KMV.Subramanyam

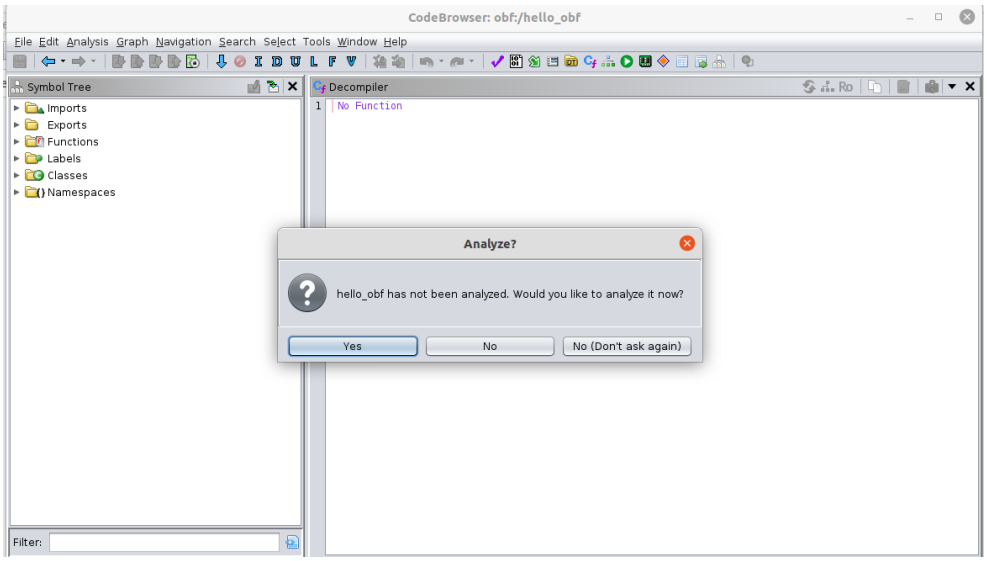
*** drag and drop your binary from your files**



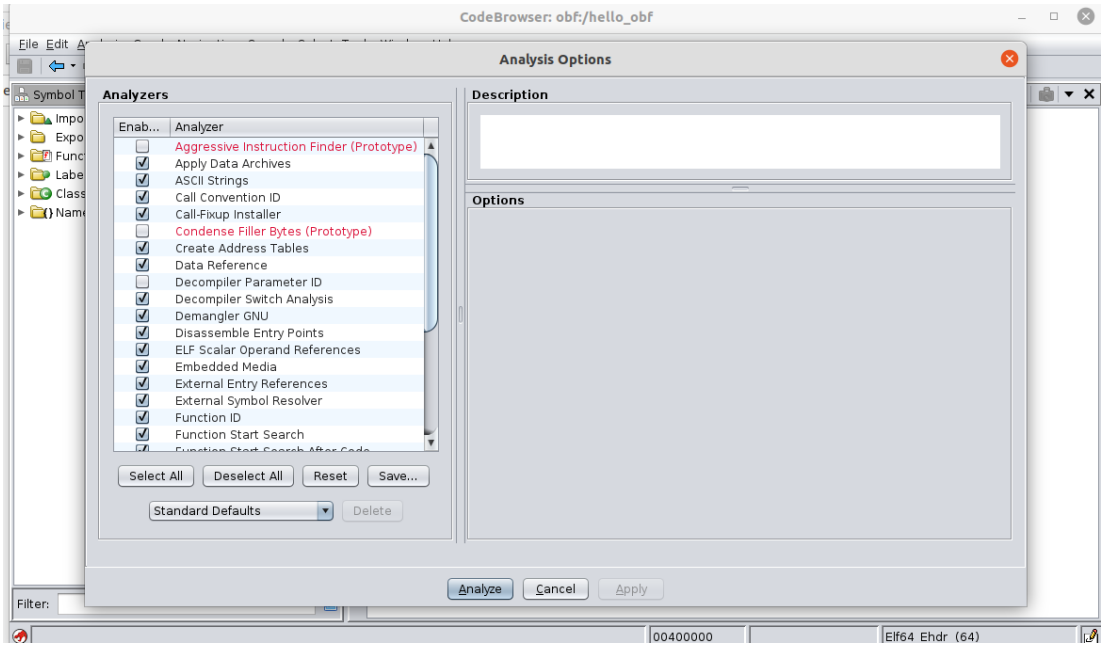
--> After drag and drop your binary file its shows like below



-->click **ok** button



-->click **Yes** button.

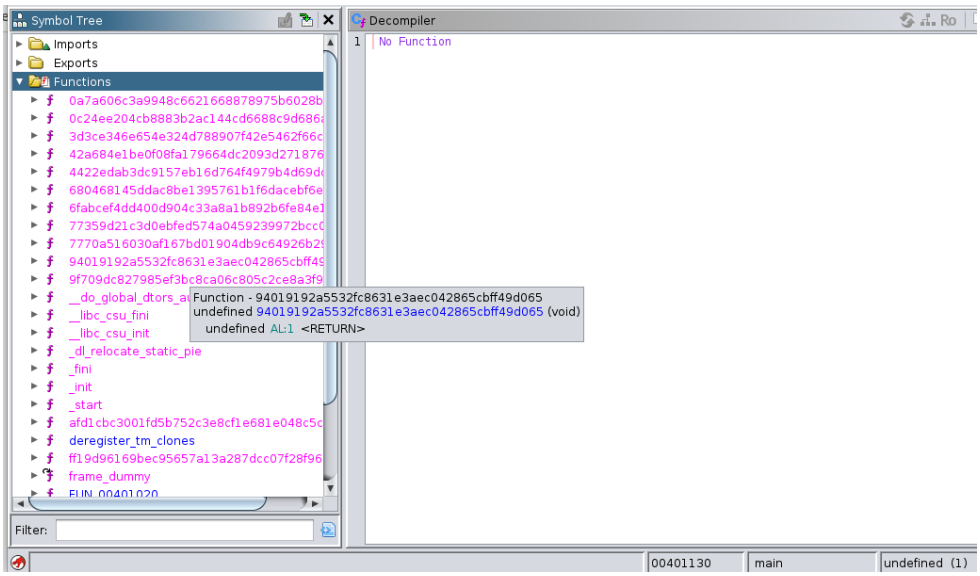


-->click **Analyze** button

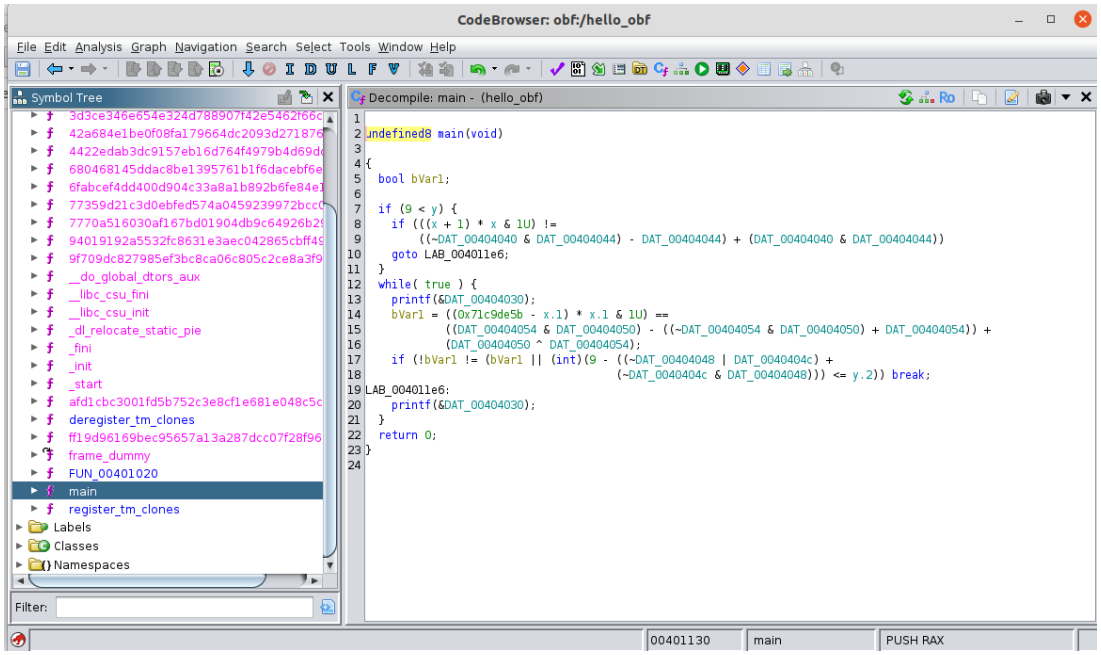
-->after analyze click a **ok** button

Test the code is obfuscate or not

-->we just write a hello world code ,so start with main function



--> code browser click a function button,it will show the functions list



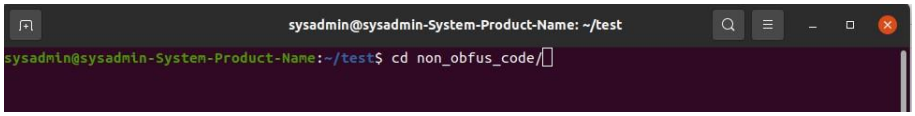
--> we write main function only, so click the main function ,it will show the code.

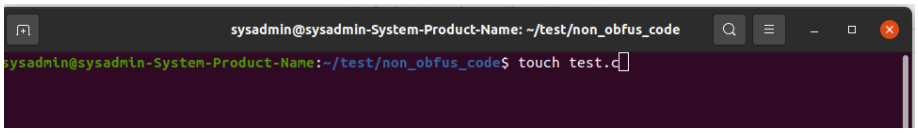
Author: L.Radhakrishnan
Reviewed By: KMV.Subramanyam

-->its not show any strings and original code.

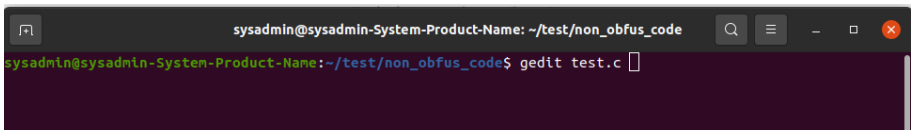
Let's test the non obfuscated code

Step:1 create a c file

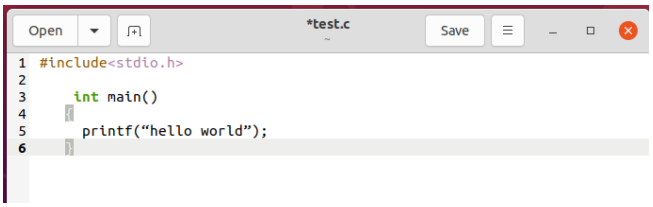




Step:2 Open the file

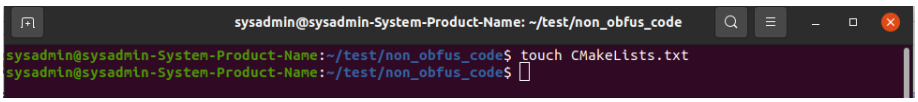


--> paste the below code and save it.

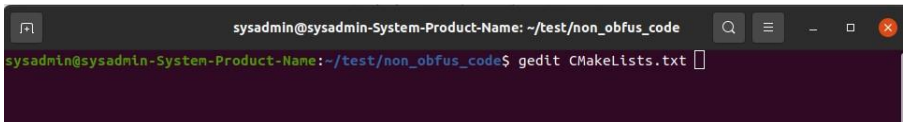



```
#include<stdio.h>
int main()
{
 printf("hello world");
}
```

Step:3 Create a Cmakefile



Step:4 Open the File and write a code



cmake_minimum_required(VERSION 3.10)

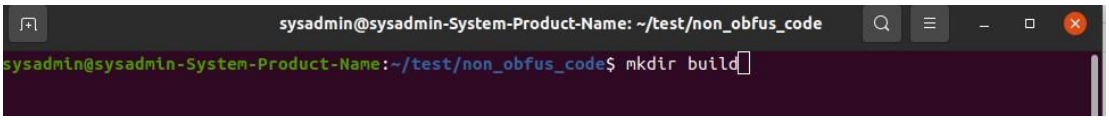
project(HelloObfuscated C CXX)

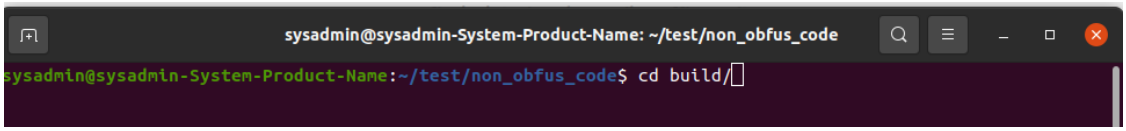
# Use Clang as the compiler
set(CMAKE_C_COMPILER "clang")
#set(CMAKE_CXX_COMPILER "clang++")

# Set compilation flags with LLVM obfuscation options
set(CMAKE_C_FLAGS "${CMAKE_C_FLAGS} -O2")
#set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -O2")

# Add the executable
add_executable(hello hello.c)

Step:5 Create a Build directory

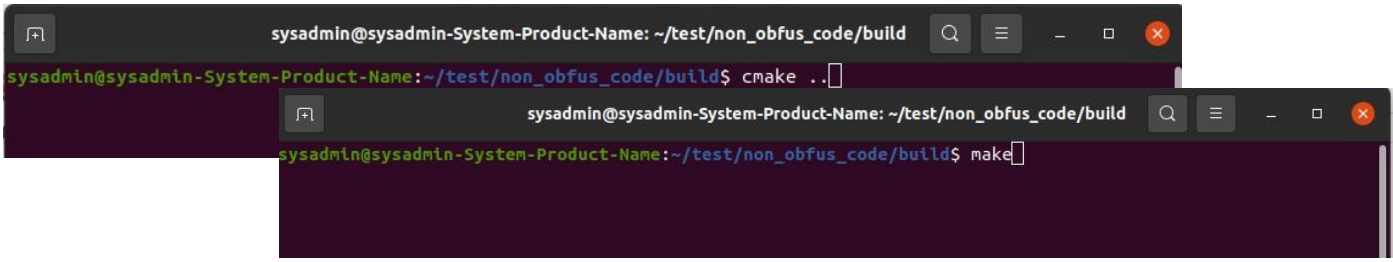




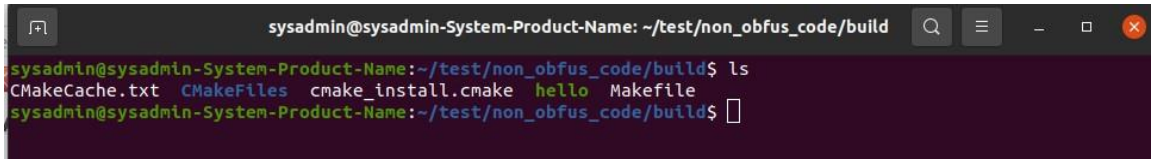
mkdir build
cd build

Author: L.Radhakrishnan
Reviewed By: KMV.Subramanyam

Step:6 Compile the code



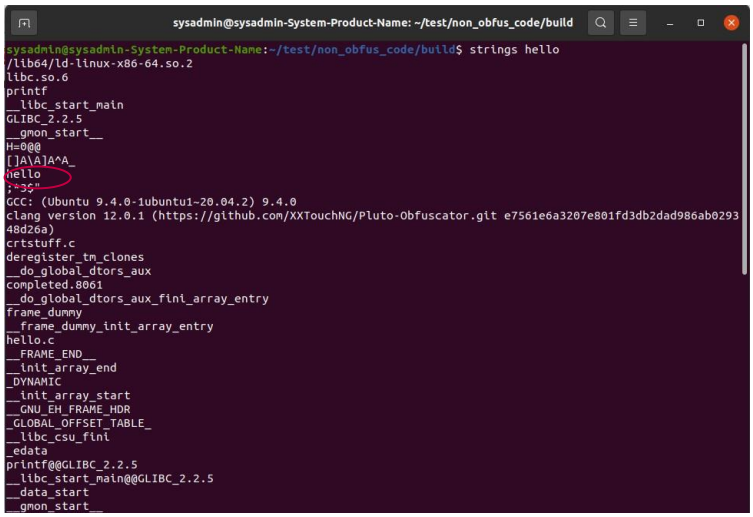
--> this will create a binary file hello



Step:7 Verify Obfuscation

Strings command:

strings hello

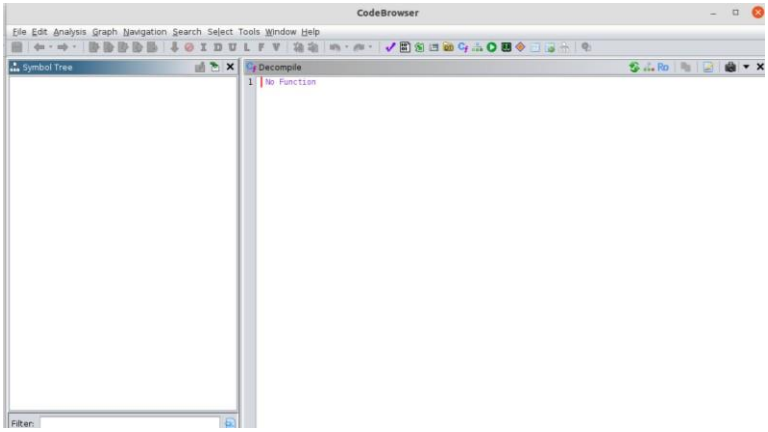


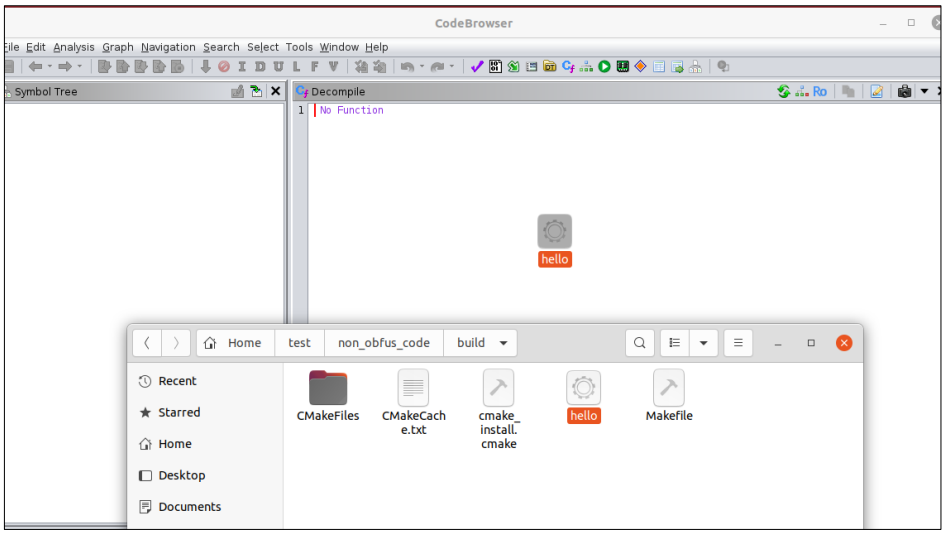
*we can see the hello string, so its not obfuscated.

Ghidra Tool

---> you know how do use the ghidra tool by previous guide.

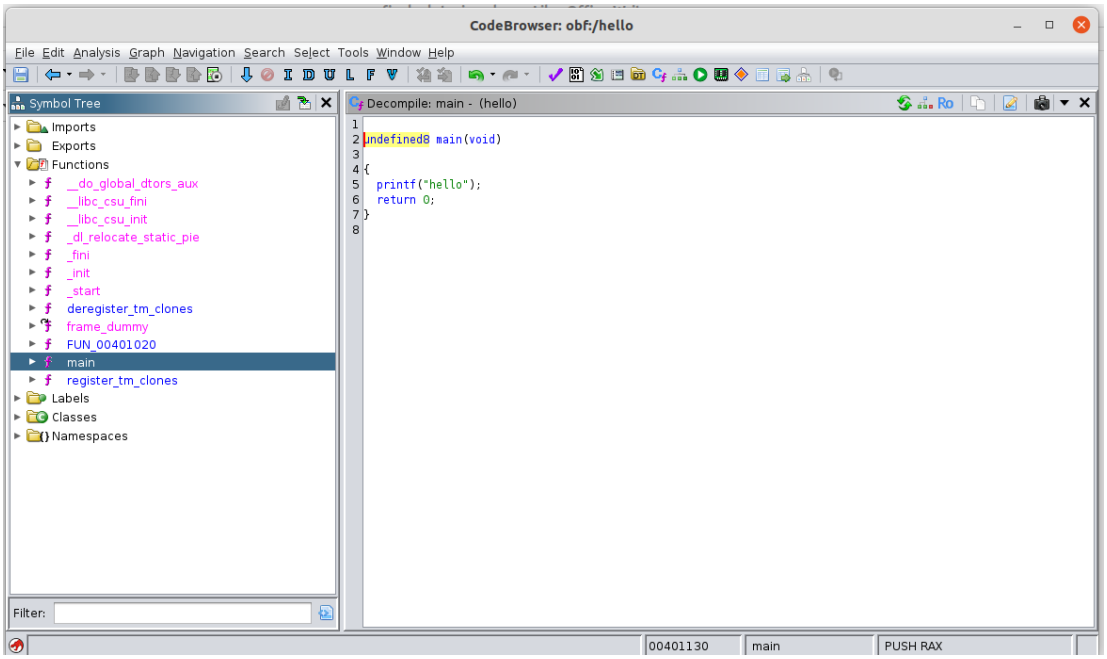
Let's check with ghidra





Author: L.Radhakrishnan
Reviewed By: KMV.Subramanyam

-->drag and drop your non obfuscated binary file



-->we can see the original code,so it's not obfuscated.