TryHackMe Room Write-up

Title: Compiled

Description: Strings can help you so far.

Author: Igor Buszta

Description: *Download the task file and get started. The binary can also be found in the AttackBox inside the /root/Rooms/Compiled/ directory.*

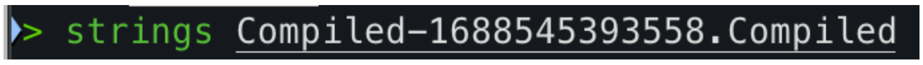
Note: The binary will not execute if using the AttackBox. However, you can still solve the challenge.

Objective: What is the password?

Tools: Ghidra, macOS, downloaded files (.Compiled).

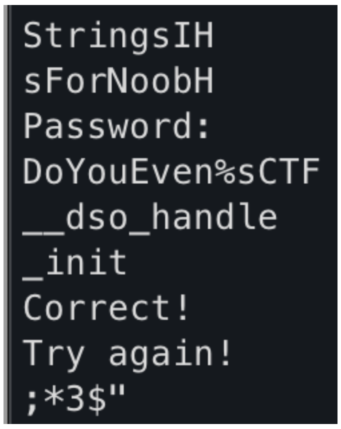
.Compiled file contains binary code which is understandable for machines. When we try to open it in Visual Code or text editor we see weird symbols, that's because the program translates these binary values to ASCII, where various values can output various symbols, signals etc.

Following our first instinct we want to see if the flag isn't in any strings hidden within the file. We use *strings* command to output any readable text which the file contains.



```
> strings Compiled-1688545393558.Compiled
```

Picture 1: Usage of strings command.



```
StringsIH
sForNoobH
Password:
DoYouEven%sCTF
__dso_handle
_init
Correct!
Try again!
;*3$"
```

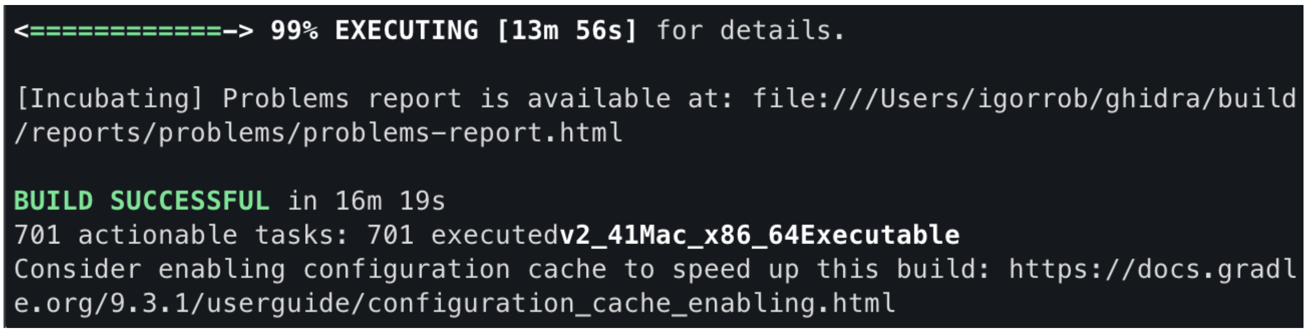
Picture 2: Selected output of strings command.

As noticed, these are just baits and taunts – not real flags. Our next step should be finding a program to reverse engineer compiled code into a readable instruction written in programming language. After research, we find Ghidra tool. I clone the repository from github, install gradle and tools with it.

In my case the gradle buildGhidra didn't work. I had to use command in main directory:

```
./gradlew buildGhidra
```

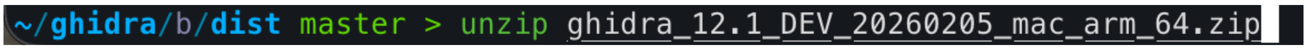
On my Macbook Air M1 it took 14 minutes to build Ghidra project.



```
<============-> 99% EXECUTING [13m 56s] for details.

[Incubating] Problems report is available at: file:///Users/igorrob/ghidra/build
/reports/problems/problems-report.html

BUILD SUCCESSFUL in 16m 19s
701 actionable tasks: 701 executedv2_41Mac_x86_64Executable
Consider enabling configuration cache to speed up this build: https://docs.gradl
e.org/9.3.1/userguide/configuration_cache_enabling.html
```

Picture 3: Finished output of ./gradlew buildGhidra.

After building, we need to unzip the file which was created in /build/dist/

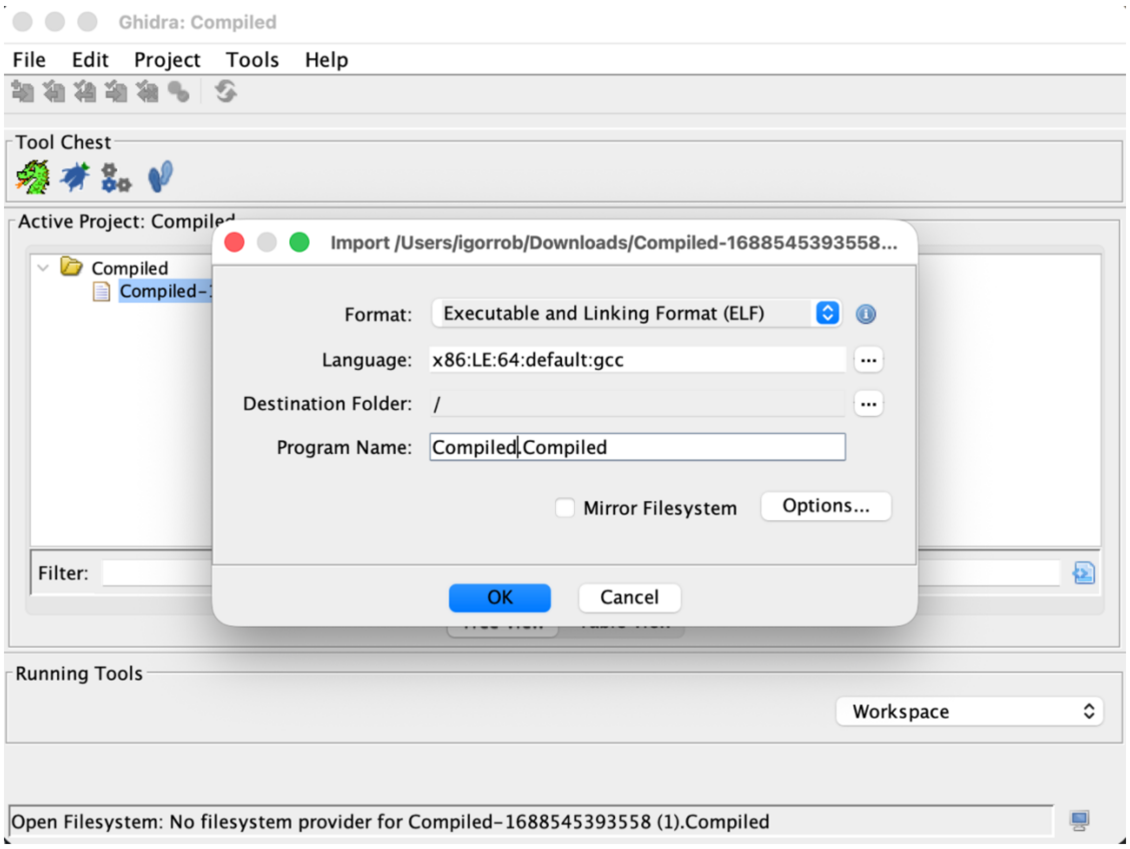


```
~/ghidra/b/dist master > unzip ghidra_12.1_DEV_20260205_mac_arm_64.zip
```

Picture 4: Unzipping ghidra file.

Change directory to ghidra_12.1_DEV file and enter ./ghidraRun command.

Let's import our .Compiled file into Ghidra.



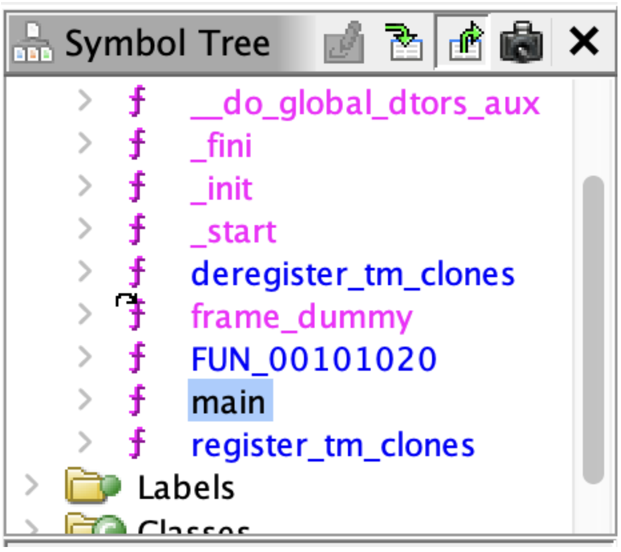
Import /Users/igorrob/Downloads/Compiled-1688545393558...

Format: Executable and Linking Format (ELF)
Language: x86:LE:64:default:gcc
Destination Folder: /
Program Name: Compiled.Compiled

Open Filesystem: No filesystem provider for Compiled-1688545393558 (1).Compiled

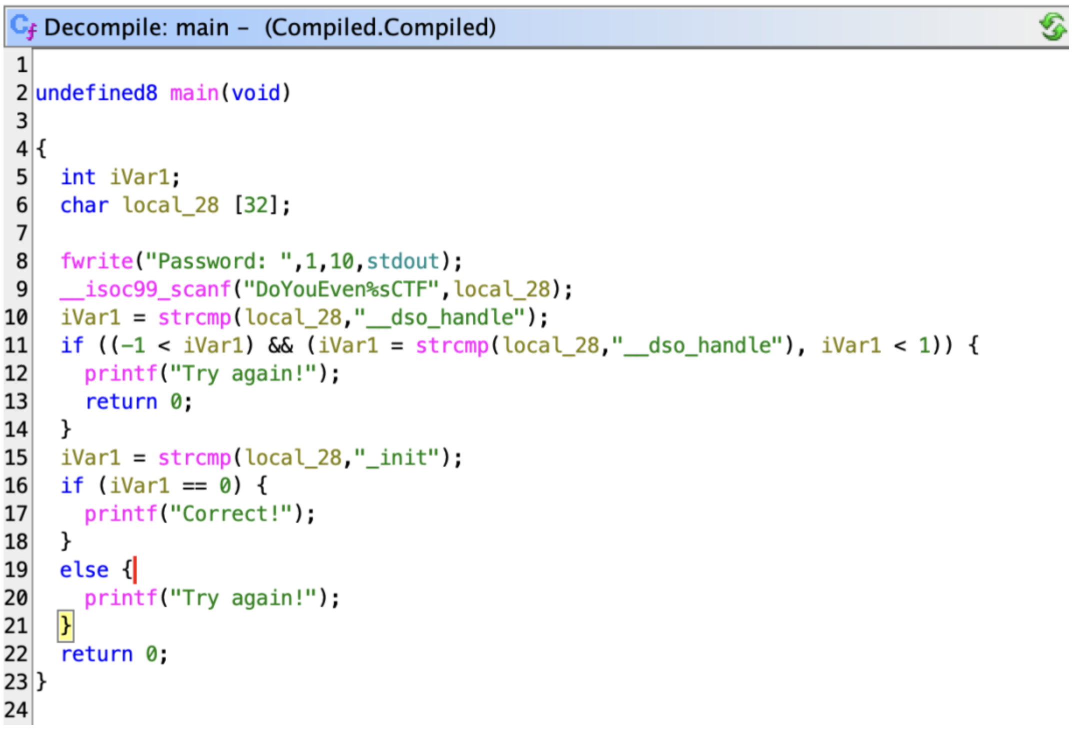
Picture 5: Importing .Compiled file into Ghidra.

The middle window shows the Assembler code, the right one shows the code in C. Let's turn our focus to main() function in this code – you can see every function in the little right window.



Picture 6: Functions in retrieved code.

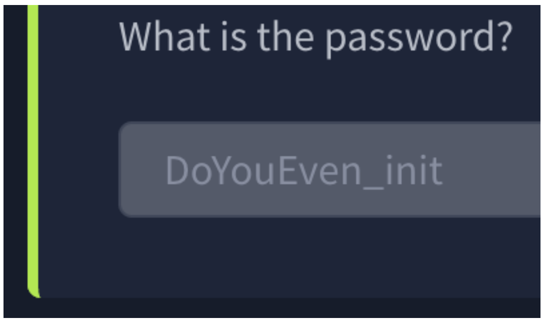
We are greet with the functionality of the code. We need to enter the password which is later compared with strcmp with specific string. The password is correct if we type in *DoYouEven[missing string]*CTF – looking at the anwser format in tryhackme room, we can omit the *CTF* part.



```

undefined8 main(void)

{
  int iVar1;
  char local_28 [32];

  fwrite("Password: ",1,10,stdout);
  __isoc99_scanf("DoYouEven%sCTF",local_28);
  iVar1 = strcmp(local_28,"__dso_handle");
  if ((-1 < iVar1) && (iVar1 = strcmp(local_28,"__dso_handle"), iVar1 < 1)) {
    printf("Try again!");
    return 0;
  }
  iVar1 = strcmp(local_28,"_init");
  if (iVar1 == 0) {
    printf("Correct!");
  }
  else {
    printf("Try again!");
  }
  return 0;
}

```

Picture 7: main function of retrieved code.

iVar1 compares local_28 (which is our guess) with „_init". If they're the same, the value is 0 (no differences" and the message printed is „Correct!" – therefore that's our password.



What is the password?

DoYouEven_init

Picture 8: The password we were looking for.