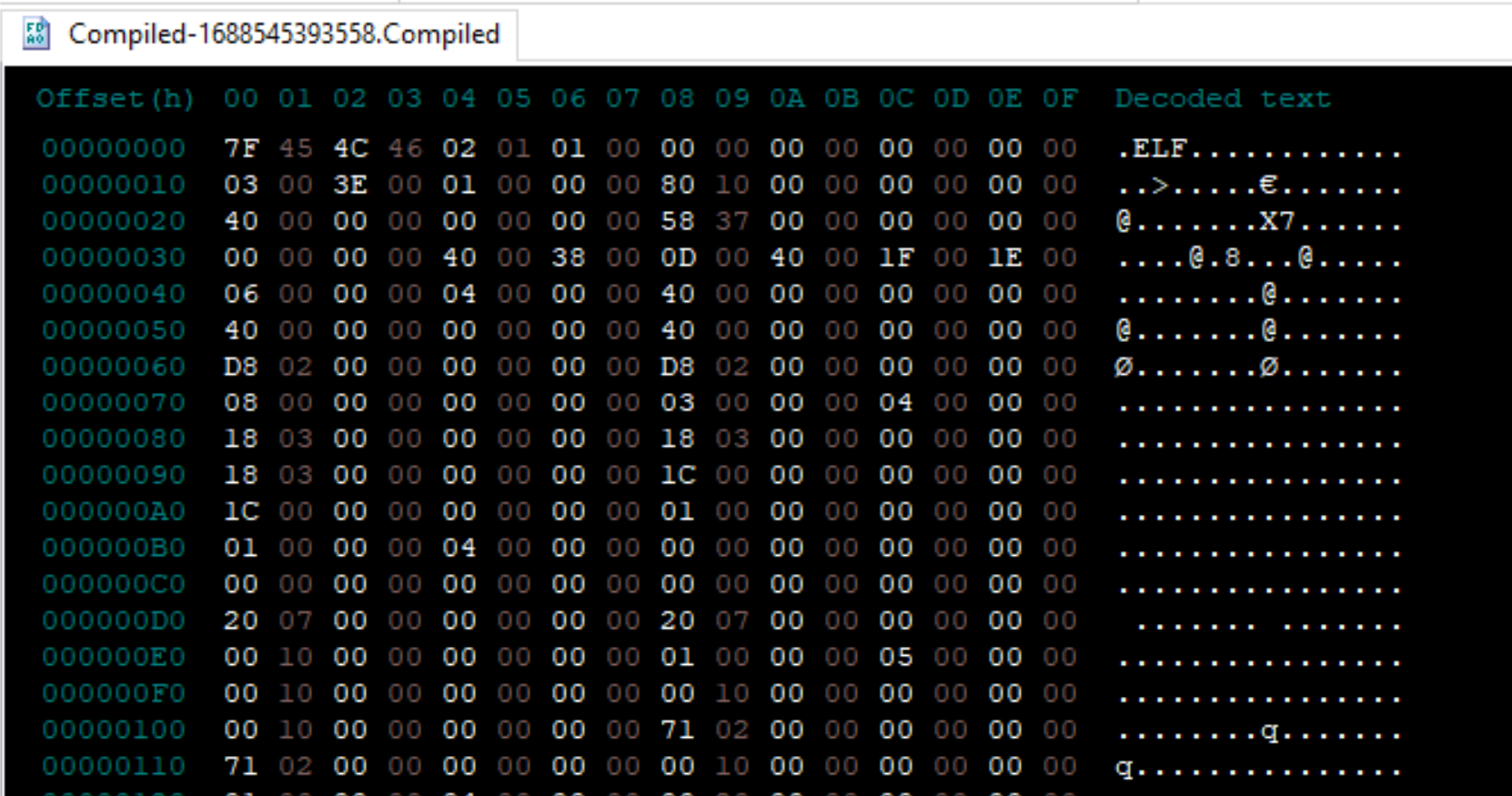
**Compiled\_Solution\_Documentation**

**Written by: Yuval Quina**

1. I began this challenge by downloading the “Compiled-1688545393558.Compiled” file and open it using HxD:

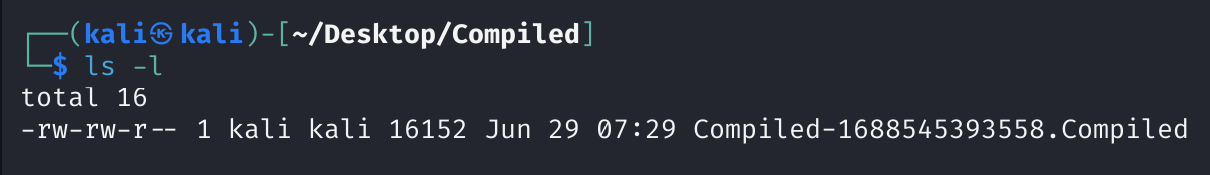


After analyzing it a bit I found that this is an executable Unix-based file.

1. So, I tried to run it from the terminal:



I saw that I got no permission to edit this file, so I checked its permission:



As I found, I can only read it in my current state.

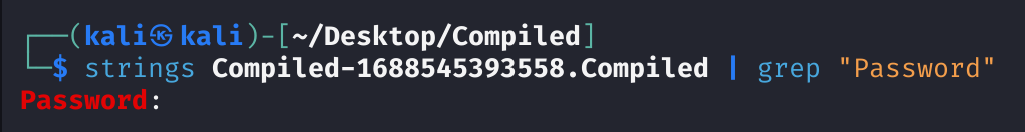
1. This made me try and open this file content:

תמונה שמכילה טקסט, גופן, צילום מסך, גרפיקה

תוכן בינה מלאכותית גנרטיבית עשוי להיות שגוי.

The content was unreadable.

1. Therefore, I tried to use the “strings” Linux tool that can help me find strings inside a file. I read the CTF description again and realized that I need to find a password, so I check for the string “Password”:



1. I found that there is a parameter that probably stores the password, so I check the location of the string “Password” (which line it is in) and found the next string that appears after that string by printing all the readable content of the file after from the specified line:



So, I found that the “Password” string is from line 21:

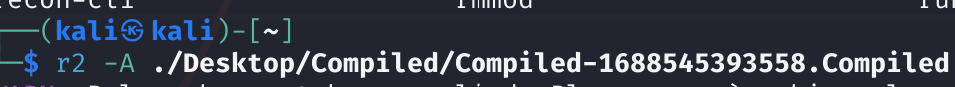


And I thought mistakenly that I found that password.

I tried to enter the password and finish with this CTF, but TryHackMe said that this is wrong password.

I realized that this is probably because of the ‘%’ that appeared in the password which is probably a pattern to the password. The pattern means that should be something else instead of the “%sCTF”.

1. Now, I opened the file using “r2” Linux tool:



I found all the functions:

תמונה שמכילה טקסט, צילום מסך, גופן, תפריט

תוכן בינה מלאכותית גנרטיבית עשוי להיות שגוי.

Now I chose the “main” function:

תמונה שמכילה טקסט, גופן, צילום מסך, גרפיקה

תוכן בינה מלאכותית גנרטיבית עשוי להיות שגוי.

And I saw it’s assembly code:

; DATA XREF from entry0 @ 0x1094(r)

┌ 253: int main (int argc, char \*\*argv, char \*\*envp);

│ ; var char \*s1 @ rbp-0x20

│ ; var int64\_t var\_30h @ rbp-0x30

│ ; var int64\_t var\_38h @ rbp-0x38

│ ; var int64\_t var\_40h @ rbp-0x40

│ 0x00001169 55 push rbp

│ 0x0000116a 4889e5 mov rbp, rsp

│ 0x0000116d 4883ec40 sub rsp, 0x40

│ 0x00001171 48b8537472.. movabs rax, 0x4973676e69727453 ; 'StringsI'

│ 0x0000117b 48ba73466f.. movabs rdx, 0x626f6f4e726f4673 ; 'sForNoob'

│ 0x00001185 488945c0 mov qword [var\_40h], rax

│ 0x00001189 488955c8 mov qword [var\_38h], rdx

│ 0x0000118d 66c745d07300 mov word [var\_30h], 0x73 ; 's'

│ 0x00001193 488b05962e.. mov rax, qword [obj.stdout] ; obj.\_\_TMC\_END\_\_

│ ; [0x4030:8]=0

│ 0x0000119a 4889c1 mov rcx, rax ; FILE \*stream

│ 0x0000119d ba0a000000 mov edx, 0xa ; size\_t nitems

│ 0x000011a2 be01000000 mov esi, 1 ; size\_t size

│ 0x000011a7 488d05560e.. lea rax, str.Password: ; 0x2004 ; "Password: "

│ 0x000011ae 4889c7 mov rdi, rax ; const void \*ptr

│ 0x000011b1 e8aafeffff call sym.imp.fwrite ; size\_t fwrite(const void \*ptr, size\_t size, size\_t nitems, FILE \*stream)

│ 0x000011b6 488d45e0 lea rax, [s1]

│ 0x000011ba 4889c6 mov rsi, rax

│ 0x000011bd 488d054b0e.. lea rax, str.DoYouEven\_sCTF ; 0x200f ; "DoYouEven%sCTF"

│ 0x000011c4 4889c7 mov rdi, rax ; const char \*format

│ 0x000011c7 b800000000 mov eax, 0

│ 0x000011cc e87ffeffff call sym.imp.\_\_isoc99\_scanf ; int scanf(const char \*format)

│ 0x000011d1 488d45e0 lea rax, [s1]

│ 0x000011d5 488d15420e.. lea rdx, str.\_\_dso\_handle ; 0x201e ; "\_\_dso\_handle"

│ 0x000011dc 4889d6 mov rsi, rdx ; const char \*s2

│ 0x000011df 4889c7 mov rdi, rax ; const char \*s1

│ 0x000011e2 e859feffff call sym.imp.strcmp ; int strcmp(const char \*s1, const char \*s2)

│ 0x000011e7 85c0 test eax, eax

│ ┌─< 0x000011e9 781a js 0x1205

│ │ 0x000011eb 488d45e0 lea rax, [s1]

│ │ 0x000011ef 488d15280e.. lea rdx, str.\_\_dso\_handle ; 0x201e ; "\_\_dso\_handle"

│ │ 0x000011f6 4889d6 mov rsi, rdx ; const char \*s2

│ │ 0x000011f9 4889c7 mov rdi, rax ; const char \*s1

│ │ 0x000011fc e83ffeffff call sym.imp.strcmp ; int strcmp(const char \*s1, const char \*s2)

│ │ 0x00001201 85c0 test eax, eax

│ ┌──< 0x00001203 7e46 jle 0x124b

│ ││ ; CODE XREF from main @ 0x11e9(x)

│ │└─> 0x00001205 488d45e0 lea rax, [s1]

│ │ 0x00001209 488d151b0e.. lea rdx, str.\_init ; 0x202b ; "\_init"

│ │ 0x00001210 4889d6 mov rsi, rdx ; const char \*s2

│ │ 0x00001213 4889c7 mov rdi, rax ; const char \*s1

│ │ 0x00001216 e825feffff call sym.imp.strcmp ; int strcmp(const char \*s1, const char \*s2)

│ │ 0x0000121b 85c0 test eax, eax

│ │┌─< 0x0000121d 7516 jne 0x1235

│ ││ 0x0000121f 488d050b0e.. lea rax, str.Correct\_ ; 0x2031 ; "Correct!"

│ ││ 0x00001226 4889c7 mov rdi, rax ; const char \*format

│ ││ 0x00001229 b800000000 mov eax, 0

│ ││ 0x0000122e e8fdfdffff call sym.imp.printf ; int printf(const char \*format)

│ ┌───< 0x00001233 eb2a jmp 0x125f

│ │││ ; CODE XREF from main @ 0x121d(x)

│ ││└─> 0x00001235 488d05fe0d.. lea rax, str.Try\_again\_ ; 0x203a ; "Try again!"

│ ││ 0x0000123c 4889c7 mov rdi, rax ; const char \*format

│ ││ 0x0000123f b800000000 mov eax, 0

│ ││ 0x00001244 e8e7fdffff call sym.imp.printf ; int printf(const char \*format)

│ ││┌─< 0x00001249 eb14 jmp 0x125f

│ │││ ; CODE XREF from main @ 0x1203(x)

│ │└──> 0x0000124b 488d05e80d.. lea rax, str.Try\_again\_ ; 0x203a ; "Try again!"

│ │ │ 0x00001252 4889c7 mov rdi, rax ; const char \*format

│ │ │ 0x00001255 b800000000 mov eax, 0

│ │ │ 0x0000125a e8d1fdffff call sym.imp.printf ; int printf(const char \*format)

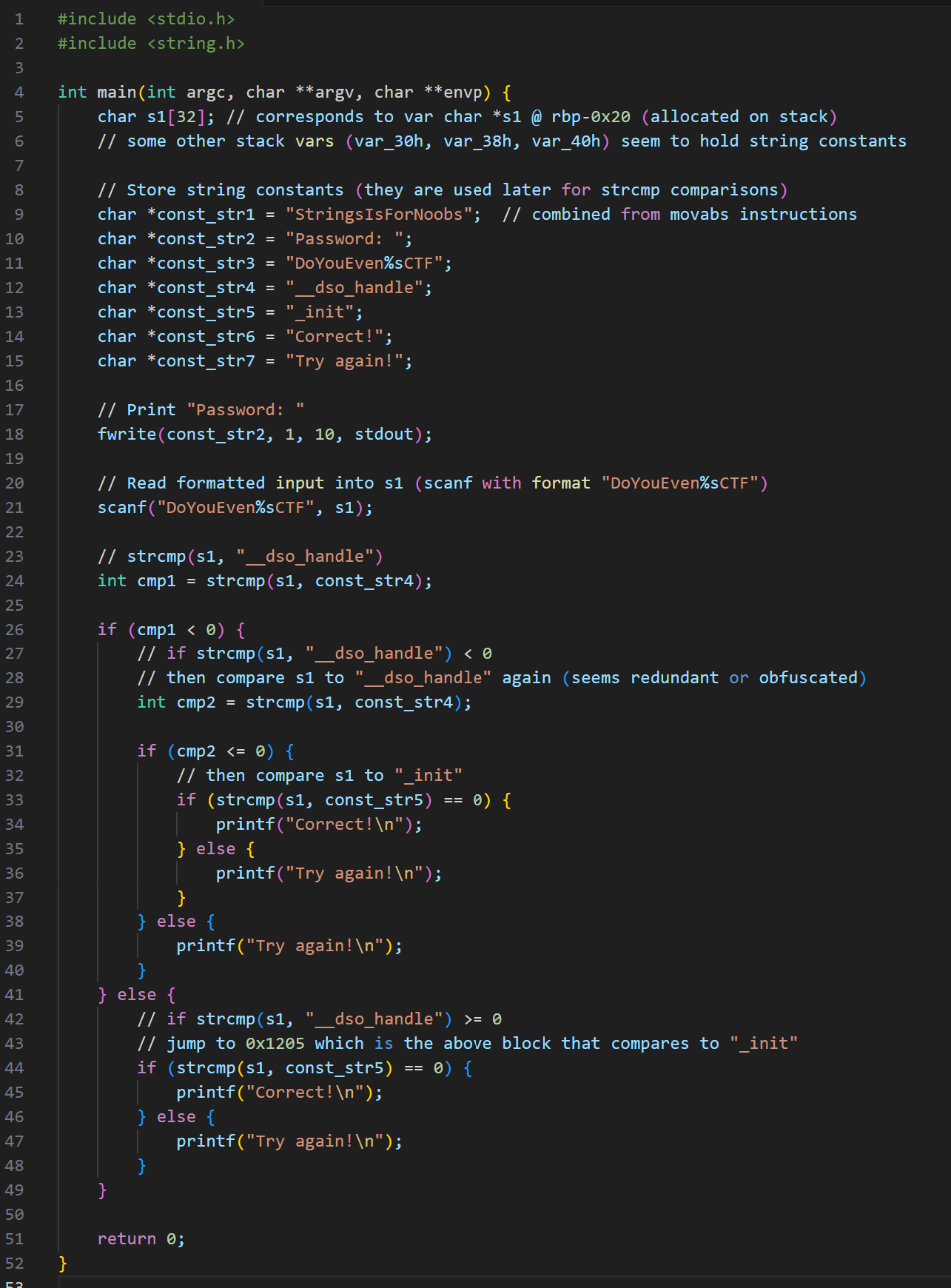
│ │ │ ; CODE XREFS from main @ 0x1233(x), 0x1249(x)

│ └─└─> 0x0000125f b800000000 mov eax, 0

│ 0x00001264 c9 leave

└ 0x00001265 c3 ret

After converting the code base to C:



Now, I found analyzing the code, I found that I need to replace “%sCTF” to “\_init” according to line 44.

And this is the correct password: “DoYouEven\_init”