Behind The Scenes

This is a reverse engineering CTF. An executable is given which gdb is unable to disassemble.

file tool

```
┌──(moghees㉿kali)-[~/Desktop/CTFs/HTB/rev_behindthescenes]
└─$ file behindthescenes
behindthescenes: ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV), dynamically linked, in
terpreter /lib64/ld-linux-x86-64.so.2, BuildID[sha1]=e60ae4c886619b869178148afd12d0a5428bfe18, f
or GNU/Linux 3.2.0, not stripped
```

The file is not stripped which means,while making the executable '-g' flag was used and that information is still on file.

> You typically have to compile in debug mode (`-g` is the GCC command-line option) to *include* the debug symbols, it's not as if they're always there until stripped out. The default is to build in non-debug mode, without the symbols.

gdb

```
┌──(moghees㉿kali)-[~/Desktop/CTFs/HTB/rev_behindthescenes]
└─$ gdb ./behindthescenes
GNU gdb (Debian 13.2-1) 13.2
Copyright (C) 2023 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./behindthescenes...
(No debugging symbols found in ./behindthescenes)
(gdb) run
Starting program: /home/blackcat/Desktop/CTFs/HTB/rev_behindthescenes/behindthescenes
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".

Program received signal SIGILL, Illegal instruction.
0x000055555555552e6 in main ()
(gdb)
```

The same thing it is in C, which is defined by the operating system, not by the language.

It is an illegal instruction trap. It is actually defined by the hardware, and when Unix/Linux gets one of these traps, it reports it as SIGILL

The usual cause when using a high-level language is causing the processor to try to execute from a valid address that is not an instruction stream. This is most commonly caused by a buffer overrun of an array of pointers on the stack, which clobbers the return address and replaces it with a pointer to a data object. When the "return" instruction is executed, it sees the bad information on the stack as the return address, so it starts trying to execute at that location, which does not contain valid instructions.

While this is the most common cause, it is not the only cause. There are many ways of generating this condition. Passing a function pointer that is not a valid function pointer will also do it. And that's the second-most-common cause. You can also, by bad pointer usage, clobber the VTABLE (Virtual Method Table) in C++. I've seen all these. And this is still not the entire set.

```
0×000055555555552e6 in main ()
(gdb) next
Single stepping until exit from function main,
which has no line number information.
0×0000555555555229 in segill_sigaction ()
(gdb) step
Single stepping until exit from function segill_sigaction,
which has no line number information.
0×000055555555552e8 in main ()
(gdb) handle SIGILL nostop
Signal        Stop      Print   Pass to program Description
SIGILL        No        Yes     Yes             Illegal instruction
(gdb) run
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/blackcat/Desktop/CTFs/HTB/rev_behindthescenes/behindthescenes
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".

Program received signal SIGILL, Illegal instruction.

Program received signal SIGILL, Illegal instruction.
./challenge <password>

Program received signal SIGILL, Illegal instruction.
[Inferior 1 (process 8418) exited with code 01]
(gdb)
```

Research SIGILL.

strings tool

```
┌──(moghees㉿kali)-[~/Desktop/CTFs/HTB/rev_behindthescenes]
└─$ strings behindthescenes
/lib64/ld-linux-x86-64.so.2
libc.so.6
strncmp
puts
__stack_chk_fail
printf
strlen
sigemptyset
memset
sigaction
__cxa_finalize
__libc_start_main
GLIBC_2.4
GLIBC_2.2.5
_ITM_deregisterTMCloneTable
__gmon_start__
_ITM_registerTMCloneTable
u+UH
[]A\A]A^A_
./challenge <password>
> HTB{%s}
:*3$"
GCC: (Ubuntu 9.3.0-17ubuntu1~20.04) 9.3.0
crtstuff.c
deregister_tm_clones
__do_global_dtors_aux
completed.8060
__do_global_dtors_aux_fini_array_entry
frame_dummy
__frame_dummy_init_array_entry
main.c
__FRAME_END__
__init_array_end
_DYNAMIC
__init_array_start
__GNU_EH_FRAME_HDR
_GLOBAL_OFFSET_TABLE_
__libc_csu_fini
strncmp@@GLIBC_2.2.5
_ITM_deregisterTMCloneTable
puts@@GLIBC_2.2.5
sigaction@@GLIBC_2.2.5
_edata
strlen@@GLIBC_2.2.5
__stack_chk_fail@@GLIBC_2.4
printf@@GLIBC_2.2.5
memset@@GLIBC_2.2.5
__libc_start_main@@GLIBC_2.2.5
__data_start
segill_sigaction
sigemptyset@@GLIBC_2.2.5
__gmon_start__
__dso_handle
```

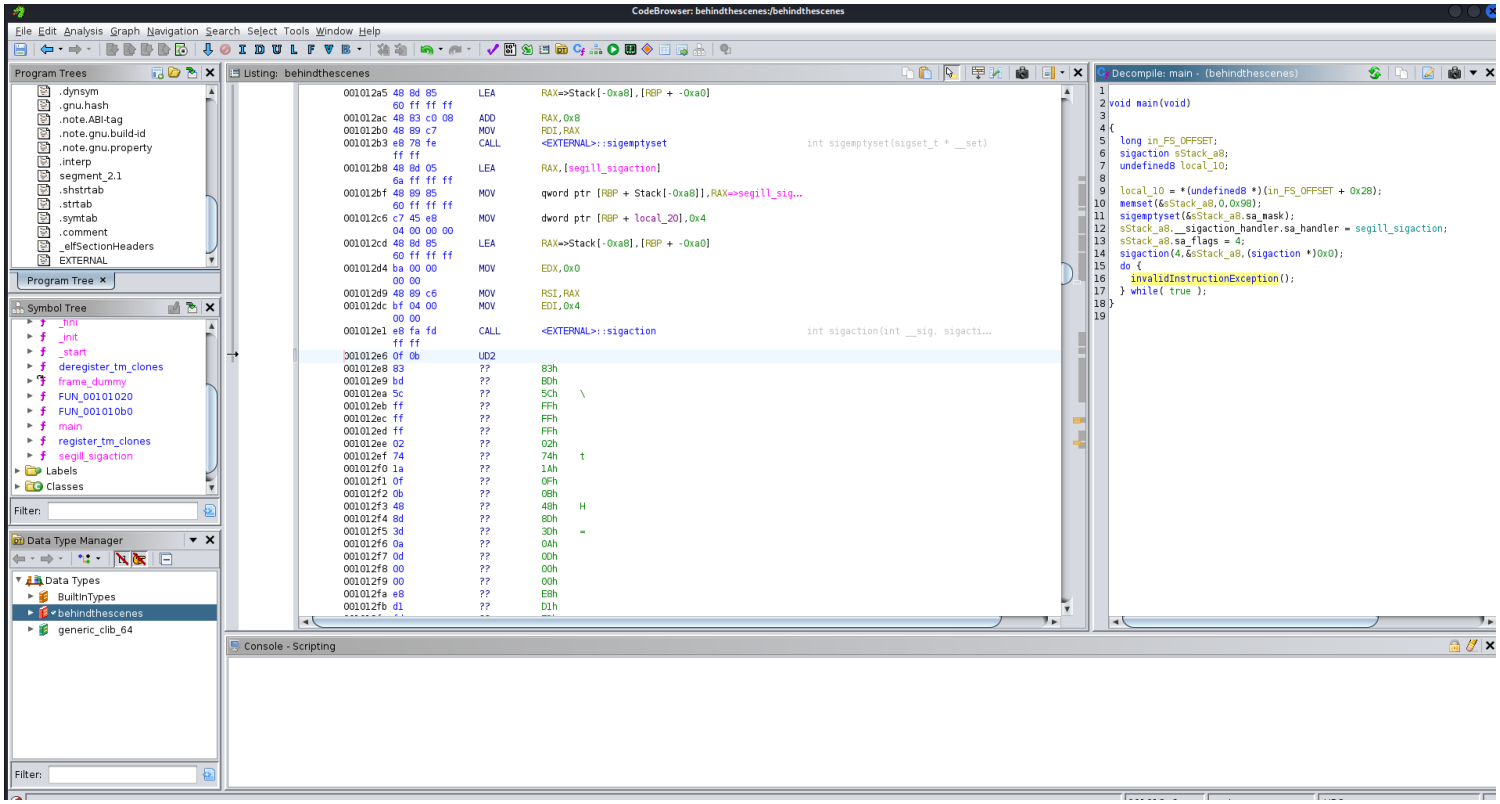
Here they are using strcmp and strlen. Which means it is checking for password length and comparing with original one.

ltrace

```
┌──(moghees㉿kali)-[~/Desktop/CTFs/HTB/rev_behindthescenes]
└─$ ltrace ./behindthescenes test123
--- SIGILL (Illegal instruction) ---
--- SIGILL (Illegal instruction) ---
--- SIGILL (Illegal instruction) ---
+++ exited (status 0) +++
```

Nothing special

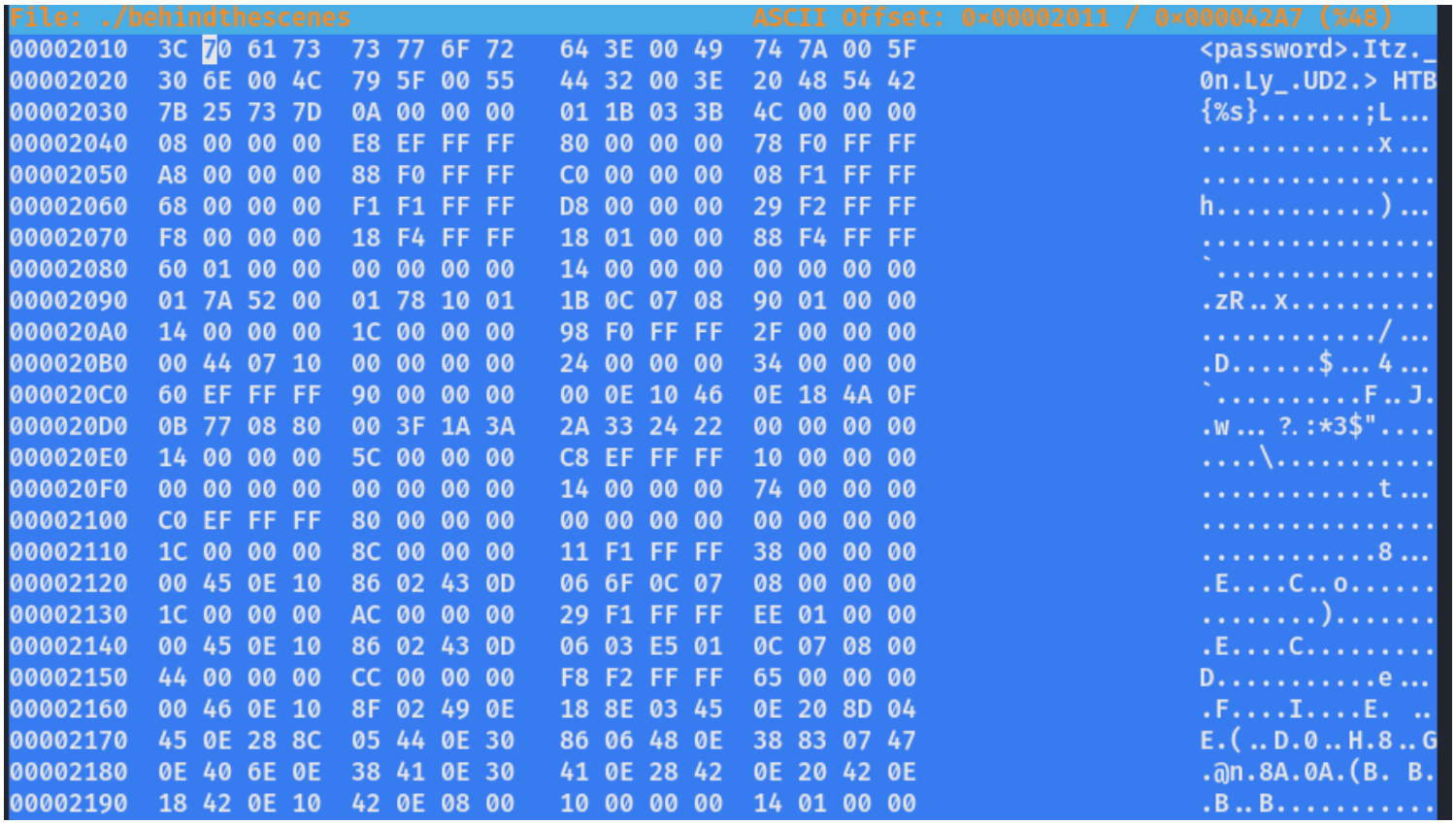
ghidra


CodeBrowser: behindthescenes:/behindthescenes
File Edit Analysis Graph Navigation Search Select Tools Window Help
Program Trees
.dynsym
.gnu.hash
.note.ABI-tag
.note.gnu.build-id
.note.gnu.property
.interp
segment_2.1
.shstrtab
.strtab
.symtab
.comment
_elfSectionHeaders
EXTERNAL
Program Tree
Symbol Tree
_fini
_init
_start
deregister_tm_clones
frame_dummy
FUN_00101020
FUN_001010b0
main
register_tm_clones
segill_sigaction
Labels
Classes
Filter:
Data Type Manager
Data Types
BuiltInTypes
behindthescenes
generic_clib_64
Filter:
Listing: behindthescenes
001012a5 48 8d 85 60 ff ff ff LEA RAX=>Stack[-0xa8],[RBP + -0xa0]
001012ac 48 83 c0 08 ADD RAX,0x8
001012b0 48 89 c7 MOV RDI,RAX
001012b3 e8 78 fe ff ff CALL <EXTERNAL>::sigemptyset int sigemptyset(sigset_t * __set)
001012b8 48 8d 05 6a ff ff ff LEA RAX,[segill_sigaction]
001012bf 48 89 85 60 ff ff ff MOV qword ptr [RBP + Stack[-0xa8]],RAX=>segill_sig...
001012c6 c7 45 e8 04 00 00 00 MOV dword ptr [RBP + local_20],0x4
001012cd 48 8d 85 60 ff ff ff LEA RAX=>Stack[-0xa8],[RBP + -0xa0]
001012d4 ba 00 00 00 00 MOV EDX,0x0
001012d9 48 89 c6 MOV RSI,RAX
001012dc bf 04 00 00 00 MOV EDI,0x4
001012e1 e8 fa fd ff ff CALL <EXTERNAL>::sigaction int sigaction(int __sig, sigacti...
001012e6 0f 0b UD2
001012e8 83 ?? 83h
001012e9 bd ?? BDh
001012ea 5c ?? 5Ch \
001012eb ff ?? FFh
001012ec ff ?? FFh
001012ed ff ?? FFh
001012ee 02 ?? 02h
001012ef 74 ?? 74h t
001012f0 1a ?? 1Ah
001012f1 0f ?? 0Fh
001012f2 0b ?? 0Bh
001012f3 48 ?? 48h H
001012f4 8d ?? 8Dh
001012f5 3d ?? 3Dh =
001012f6 0a ?? 0Ah
001012f7 0d ?? 0Dh
001012f8 00 ?? 00h
001012f9 00 ?? 00h
001012fa e8 ?? E8h
001012fb d1 ?? D1h
Decompile: main - (behindthescenes)
1 void main(void)
2 {
3 long in_FS_OFFSET;
4 sigaction sStack_a8;
5 undefined8 local_10;
6
7 local_10 = *(undefined8 *)(in_FS_OFFSET + 0x28);
8 memset(&sStack_a8,0,0x98);
9 sigemptyset(&sStack_a8.sa_mask);
10 sStack_a8.__sigaction_handler.sa_handler = segill_sigaction;
11 sStack_a8.sa_flags = 4;
12 sigaction(4,&sStack_a8,(sigaction *)0x0);
13 do {
14 invalidInstructionException();
15 } while(true);
16 }
17
18
19
Console - Scripting

```
00102000 01 00 02 00     undefined4  00020001h
00102004 2e 2f 63        ds          "./challenge <password>"
         68 61 6c
         6c 65 6e ...
0010201b 49              ??          49h    I
0010201c 74              ??          74h    t
0010201d 7a              ??          7Ah    z
0010201e 00              ??          00h
0010201f 5f              ??          5Fh    _
00102020 30              ??          30h    0
00102021 6e              ??          6Eh    n
00102022 00              ??          00h
00102023 4c              ??          4Ch    L
00102024 79              ??          79h    y
00102025 5f              ??          5Fh    _
00102026 00              ??          00h
00102027 55              ??          55h    U
00102028 44              ??          44h    D
00102029 32              ??          32h    2
0010202a 00              ??          00h
0010202b 3e              ??          3Eh    >
0010202c 20              ??          20h
0010202d 48              ??          48h    H
0010202e 54              ??          54h    T
0010202f 42              ??          42h    B
00102030 7b              ??          7Bh    {
00102031 25              ??          25h    %
00102032 73              ??          73h    s
00102033 7d              ??          7Dh    }
00102034 0a              ??          0Ah
00102035 00              ??          00h
.....
```

got the password but not complete.
Couldnt understand how its done in walkthrough so using 'HexEditor'

hexeditor



The Password is Itz_0nLy_UD2

```
┌──(moghees㉿kali)-[~/Desktop/CTFs/HTB/rev_behindthescenes]
└─$ ./behindthescenes Itz_0nLy_UD2
> HTB{Itz_0nLy_UD2}
```

Solved.
*** USED WALKTHROUGH***