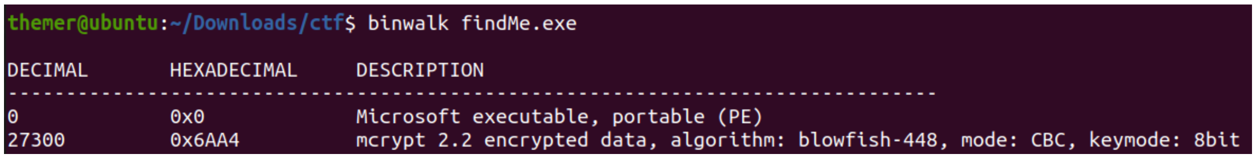
Category: Rev

Challenge: Find Me

Writeup:

When I run binwalk, I see the following information:



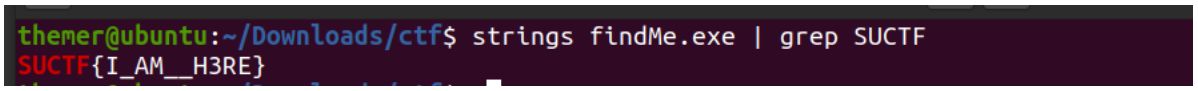
```
themer@ubuntu:~/Downloads/ctf$ binwalk findMe.exe

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             Microsoft executable, portable (PE)
27300         0x6AA4          mcrypt 2.2 encrypted data, algorithm: blowfish-448, mode: CBC, keymode: 8bit
```

It is an unpacked windows executable. Since it is not packed, there is no need to unpacking process before reversing.

There are multiple solutions exist.

Solution 1:

Just write the following and you will get the flag

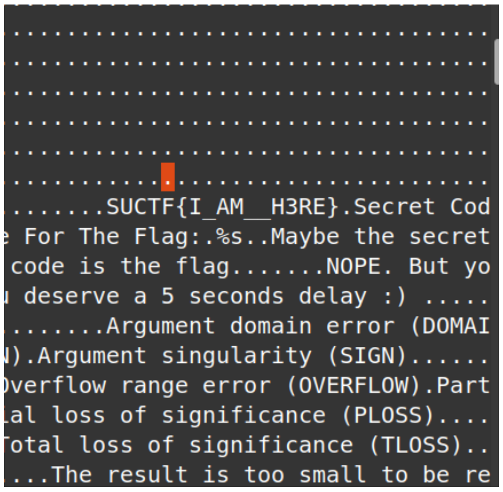


```
themer@ubuntu:~/Downloads/ctf$ strings findMe.exe | grep SUCTF
SUCTF{I_AM__H3RE}
```

Solution 2:

You can open it with a hex editor.

At around 0x25F8 address you can see the flag:



```
.................................
.................................
.................................
.................................
.................................
.................................
.......SUCTF{I_AM__H3RE}.Secret Cod
e For The Flag:.%s..Maybe the secret
 code is the flag.......NOPE. But yo
u deserve a 5 seconds delay :) .....
.......Argument domain error (DOMAI
N).Argument singularity (SIGN)......
Overflow range error (OVERFLOW).Part
ial loss of significance (PLOSS)....
Total loss of significance (TLOSS)..
...The result is too small to be re
```

Solution 3:

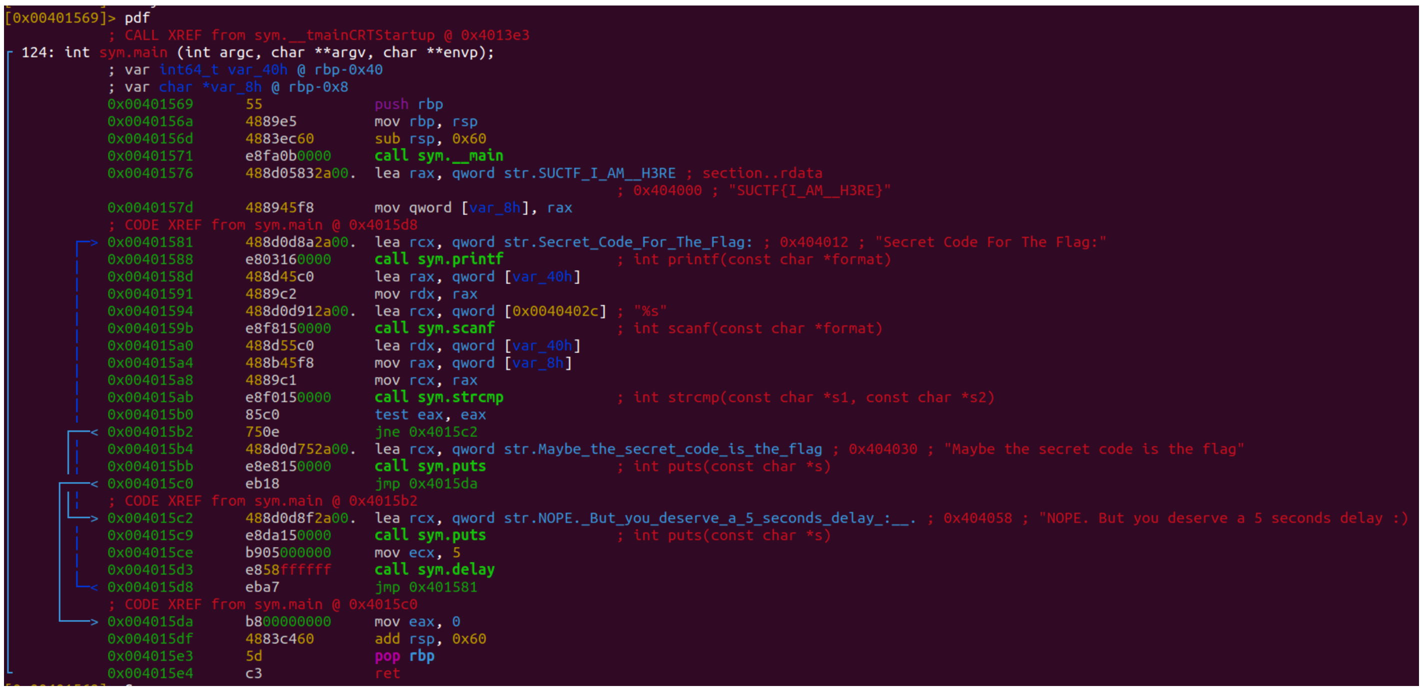
You can disassemble it. I will use radare2 now. But ofc. you can use any other disassembler.

run following:

$ radare2 findme.exe

- aaa
- afl
- s sym.main
- pdf

and you will see following:



```
[0x00401569]> pdf
            ; CALL XREF from sym.__tmainCRTStartup @ 0x4013e3
┌ 124: int sym.main (int argc, char **argv, char **envp);
│           ; var int64_t var_40h @ rbp-0x40
│           ; var char *var_8h @ rbp-0x8
│           0x00401569      55             push rbp
│           0x0040156a      4889e5         mov rbp, rsp
│           0x0040156d      4883ec60       sub rsp, 0x60
│           0x00401571      e8fa0b0000     call sym.__main
│           0x00401576      488d05832a00.  lea rax, qword str.SUCTF_I_AM__H3RE ; section..rdata
│                                                                  ; 0x404000 ; "SUCTF{I_AM__H3RE}"
│           0x0040157d      488945f8       mov qword [var_8h], rax
│           ; CODE XREF from sym.main @ 0x4015d8
│       ┌─> 0x00401581      488d0d8a2a00.  lea rcx, qword str.Secret_Code_For_The_Flag: ; 0x404012 ; "Secret Code For The Flag:"
│       ╎   0x00401588      e803160000     call sym.printf              ; int printf(const char *format)
│       ╎   0x0040158d      488d45c0       lea rax, qword [var_40h]
│       ╎   0x00401591      4889c2         mov rdx, rax
│       ╎   0x00401594      488d0d912a00.  lea rcx, qword [0x0040402c] ; "%s"
│       ╎   0x0040159b      e8f8150000     call sym.scanf               ; int scanf(const char *format)
│       ╎   0x004015a0      488d55c0       lea rdx, qword [var_40h]
│       ╎   0x004015a4      488b45f8       mov rax, qword [var_8h]
│       ╎   0x004015a8      4889c1         mov rcx, rax
│       ╎   0x004015ab      e8f0150000     call sym.strcmp              ; int strcmp(const char *s1, const char *s2)
│       ╎   0x004015b0      85c0           test eax, eax
│      ┌──< 0x004015b2      750e           jne 0x4015c2
│      │╎   0x004015b4      488d0d752a00.  lea rcx, qword str.Maybe_the_secret_code_is_the_flag ; 0x404030 ; "Maybe the secret code is the flag"
│      │╎   0x004015bb      e8e8150000     call sym.puts                ; int puts(const char *s)
│     ┌───< 0x004015c0      eb18           jmp 0x4015da
│     ││╎   ; CODE XREF from sym.main @ 0x4015b2
│     │└──> 0x004015c2      488d0d8f2a00.  lea rcx, qword str.NOPE._But_you_deserve_a_5_seconds_delay_:__. ; 0x404058 ; "NOPE. But you deserve a 5 seconds delay :)
│     │ ╎   0x004015c9      e8da150000     call sym.puts                ; int puts(const char *s)
│     │ ╎   0x004015ce      b905000000     mov ecx, 5
│     │ ╎   0x004015d3      e858ffffff     call sym.delay
│     │ └─< 0x004015d8      eba7           jmp 0x401581
│     │     ; CODE XREF from sym.main @ 0x4015c0
│     └───> 0x004015da      b800000000     mov eax, 0
│           0x004015df      4883c460       add rsp, 0x60
│           0x004015e3      5d             pop rbp
└           0x004015e4      c3             ret
```

flag is there.