First we give the bin file executable rights
`chmod +x impossible_password.bin`

then open the file using radare2
use `aaa` to analyze the whole binary file

```
r2 impossible_password.bin
WARN: Relocs has not been applied. Please use `-e bin.relocs.apply=true`
or `-e bin.cache=true` next time
[0x004006a0]> aaa
INFO: Analyze all flags starting with sym. and entry0 (aa)
INFO: Analyze imports (af@@@i)
INFO: Analyze entrypoint (af@ entry0)
INFO: Analyze symbols (af@@@s)
INFO: Analyze all functions arguments/locals (afva@@@F)
INFO: Analyze function calls (aac)
INFO: Analyze len bytes of instructions for references (aar)
INFO: Finding and parsing C++ vtables (avrr)
INFO: Analyzing methods (af @@ method.*)
INFO: Recovering local variables (afva@@@F)
INFO: Type matching analysis for all functions (aaft)
INFO: Propagate noreturn information (aanr)
INFO: Use -AA or aaaa to perform additional experimental analysis
[0x004006a0]>
```

We scan for functions and list them with `afl` and `s` to select a function

```
[0x004006a0]> afl
0x004005f0    1      6 sym.imp.putchar
0x00400600    1      6 sym.imp.printf
0x00400610    1      6 sym.imp.__libc_start_main
0x00400620    1      6 sym.imp.srand
0x00400630    1      6 sym.imp.strcmp
0x00400650    1      6 sym.imp.time
0x00400660    1      6 sym.imp.malloc
0x00400670    1      6 sym.imp.__isoc99_scanf
0x00400680    1      6 sym.imp.exit
0x00400690    1      6 sym.imp.rand
0x004006a0    1     41 entry0
0x0040085d    5    283 main
0x0040078d    7    208 fcn.0040078d
0x00400978    5     96 fcn.00400978
0x00400760    8     99 entry.init0
0x00400740    3     28 entry.fini0
0x004006d0    4     41 fcn.004006d0
0x00400640    1      6 loc.imp.__gmon_start__
```

```
0x004005c0     3      26 fcn.004005c0
[0x004006a0]> s main
```

```
283: int main (int argc, char **argv);
`- args(rdi, rsi) vars(25:sp[0x10..0x58])
        0x0040085d      55              push rbp
        0x0040085e      4889e5          mov rbp, rsp
        0x00400861      4883ec50        sub rsp, 0x50
        0x00400865      897dbc          mov dword [var_44h], edi     ; argc
        0x00400868      488975b8        mov qword [var_50h], rsi     ; argv
        0x0040086c      488d745f8870..  mov qword [s2], str.SuperSeKretKey     ; 0x400a70 ; "SuperSeKretKey"
        0x00400874      c645c041        mov byte [var_40h], 0x41     ; 'A' ; 65
        0x00400878      c645c15d        mov byte [var_3fh], 0x5d     ; ']' ; 93
        0x0040087c      c645c24b        mov byte [var_3eh], 0x4b     ; 'K' ; 75
        0x00400880      c645c372        mov byte [var_3dh], 0x72     ; 'r' ; 114
        0x00400884      c645c43d        mov byte [var_3ch], 0x3d     ; '=' ; 61
        0x00400888      c645c539        mov byte [var_3bh], 0x39     ; '9' ; 57
        0x0040088c      c645c66b        mov byte [var_3ah], 0x6b     ; 'k' ; 107
        0x00400890      c645c730        mov byte [var_39h], 0x30     ; '0' ; 48
        0x00400894      c645c83d        mov byte [var_38h], 0x3d     ; '=' ; 61
        0x00400898      c645c930        mov byte [var_37h], 0x30     ; '0' ; 48
        0x0040089c      c645ca6f        mov byte [var_36h], 0x6f     ; 'o' ; 111
        0x004008a0      c645cb30        mov byte [var_35h], 0x30     ; '0' ; 48
        0x004008a4      c645cc3b        mov byte [var_34h], 0x3b     ; ';' ; 59
        0x004008a8      c645cd6b        mov byte [var_33h], 0x6b     ; 'k' ; 107
        0x004008ac      c645ce31        mov byte [var_32h], 0x31     ; '1' ; 49
        0x004008b0      c645cf3f        mov byte [var_31h], 0x3f     ; '?' ; 63
        0x004008b4      c645d06b        mov byte [var_30h], 0x6b     ; 'k' ; 107
        0x004008b8      c645d138        mov byte [var_2fh], 0x38     ; '8' ; 56
        0x004008bc      c645d231        mov byte [var_2eh], 0x31     ; '1' ; 49
        0x004008c0      c645d374        mov byte [var_2dh], 0x74     ; 't' ; 116
        0x004008c4      bf7f0a4000      mov edi, 0x400a7f            ; '\x7f\n@' ; "* " ; const char *format
```

From what we see there is a string that stands out and might be the key from `str.SuperSecretKey` which is `SuperSeKretKey` and when we test it out from running the program it accepts

```
./impossible_password.bin
* SuperSeKretKey
[SuperSeKretKey]
**
```

But the flag isn't showing so there's got to be something else.

```
        0x0040095b      4889d6          mov rsi, rdx                 ; const char *s2
        0x0040095e      4889c7          mov rdi, rax                 ; const char *s1
        0x00400961      e8cafcffff      call sym.imp.strcmp          ;[5] ; int strcmp(const char *s1, const char
        0x00400966      85c0            test eax, eax
    ┌─< 0x00400968      75bc            jne 0x400976
    │   0x0040096a      488d45c0        lea rax, [var_40h]
    │   0x0040096e      4889c7          mov rdi, rax                 ; int64_t arg1
    │   0x00400971      e802000000      call fcn.00400978            ;[6]
    │   ; CODE XREF from main @ 0x400968(x)
    └─> 0x00400976      c9              leave
        0x00400977      c3              ret
        ; CALL XREF from main @ 0x400971(x)
96: fcn.00400978 (int64_t arg1);
`- args(rdi) vars(4:sp[0x10..0x20])
```

further analysis show's that after test operand for eax it jumps to the leave function instead of running the call function that might print the flag so we need to change the program at address `0x00400968` to prevent it from jumping. We do so by replacing the jump instruction with a nop.

use oo+ to switch to write mode then use wx to make it write two intel nops then wa to assign the nop operation which means no operation so the program will just continue without skipping.

```
[0x004006a0]> oo+
[0x004006a0]> wx 9090
[0x004006a0]> wa nop
INFO: Written 1 byte(s) (nop) = wx 90 @ 0x004006a0
[0x004006a0]>
```

```
0x00400940    b800000000    mov eax, 0
0x00400945    e826fdffff    call sym.imp.__isoc99_scanf ;[4]
0x0040094a    bf14000000    mov edi, 0x14               ; 20
0x0040094f    e839feffff    call 0x40078d               ;[5]
0x00400954    4889c2        mov rdx, rax
0x00400957    488d45e0      lea rax, [rbp - 0x20]
0x0040095b    4889d6        mov rsi, rdx
0x0040095e    4889c7        mov rdi, rax
0x00400961    e8cafcffff    call sym.imp.strcmp         ;[1]
0x00400966    85c0          test eax, eax
0x00400968    90            nop
0x00400969    90            nop
0x0040096a    488d45c0      lea rax, [rbp - 0x40]
0x0040096e    4889c7        mov rdi, rax
0x00400971    e882000000    call 0x400978               ;[6]
0x00400976    c9            leave
0x00400977    c3            ret
0x00400978    55            push rbp
```

If we check the same address we see that it changed to nop instead of jne which means it overwrote the instruction.
Now when we enter the key and enter a random value it prints the flag

```
Downloads]$ ./impossible_password.bin
* SuperSeKretKey
[SuperSeKretKey]
** whoismod
HTB{40b949f92b86b18}
```