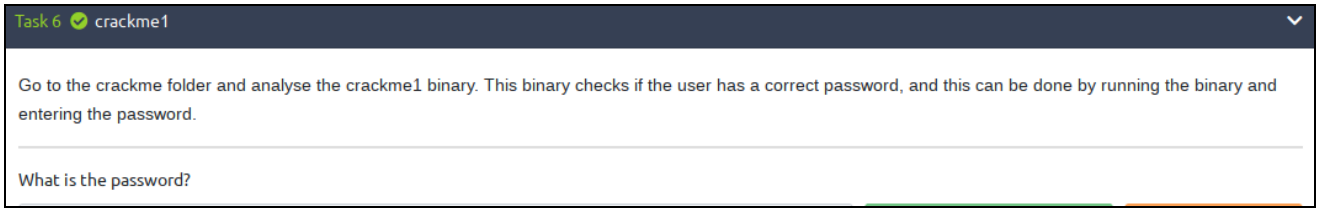
Report

Write Up TryHackMe Challenge Intro to x86-64

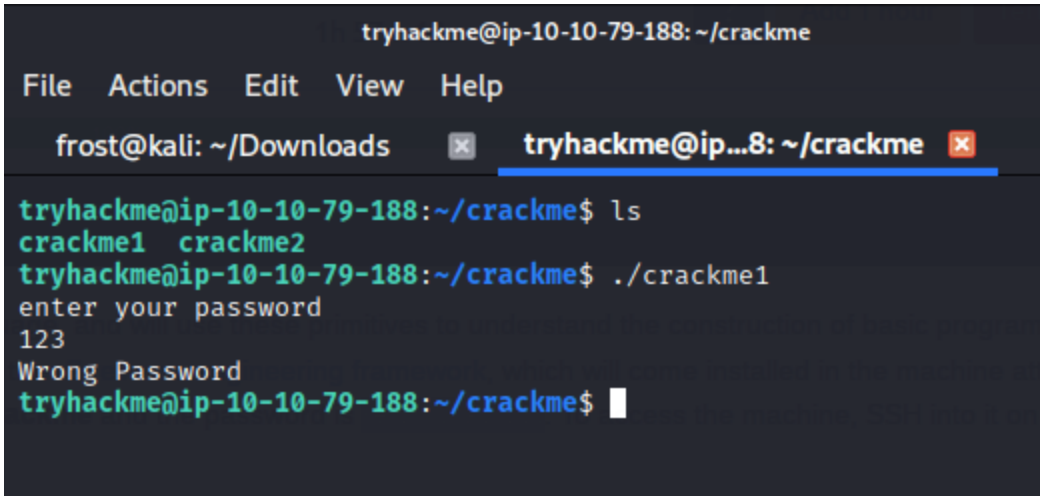
Akmal Ilmi

7th April, 2021

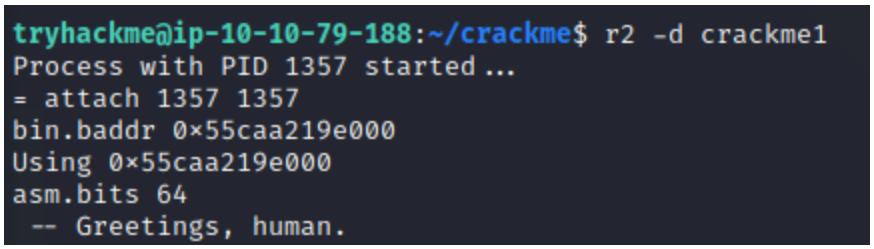
Crackme1



First Of All, i am connecting the machine in TryHackMe and open the exe crackme1 file



Our Task is to crack the code, we need to debug this exe file using radare2



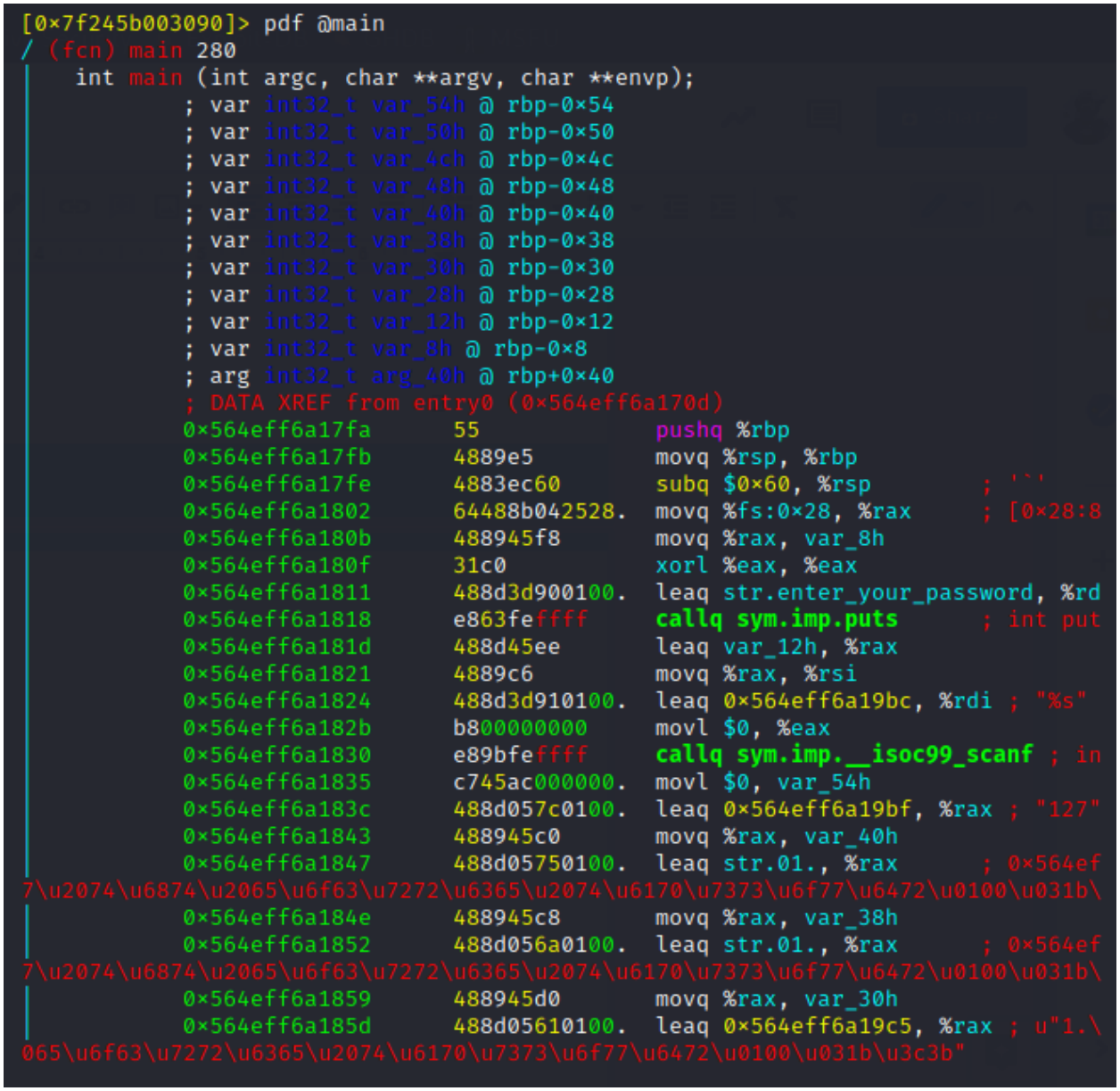
After that, check and analyze this exe file using command aaa that's stand for analyze all

```
[0×7f7ebeb09090]> aaa
[x] Analyze all flags starting with sym. and entry0 (aa)
[Warning: Invalid range. Use different search.in=? or anal.in=dbg.maps.x
Warning: Invalid range. Use different search.in=? or anal.in=dbg.maps.x
[x] Analyze function calls (aac)
[x] Analyze len bytes of instructions for references (aar)
[x] Check for objc references
[x] Check for vtables
[TOFIX: aaft can't run in debugger mode.ions (aaft)
[x] Type matching analysis for all functions (aaft)
[x] Use -AA or aaaa to perform additional experimental analysis.
```

We can also list function, using afl command, maybe we can see something useful

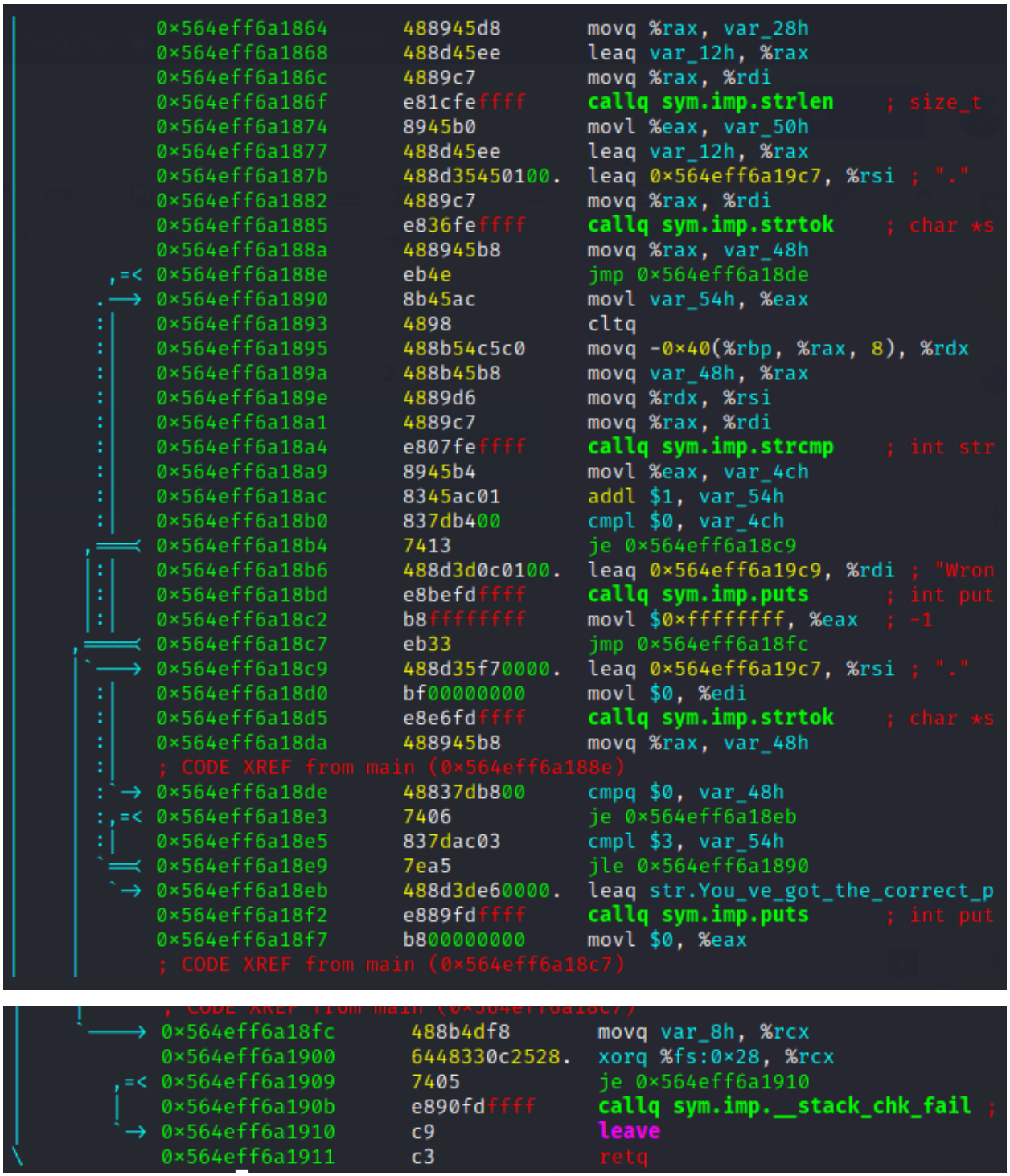
```
[0×7f1eaed77090]> afl
0×559f769a36f0    1 42           entry0
0×559f76ba3fe0    1 4124         reloc.__libc_start_main
0×559f769a3720    4 50   → 40    sym.deregister_tm_clones
0×559f769a3760    4 66   → 57    sym.register_tm_clones
0×559f769a37b0    5 58   → 51    entry.fini0
0×559f769a36e0    1 6            sym..plt.got
0×559f769a37f0    1 10           entry.init0
0×559f769a3990    1 2            sym.__libc_csu_fini
0×559f769a3994    1 9            sym._fini
0×559f769a3920    4 101          sym.__libc_csu_init
0×559f769a37fa   12 283          main
0×559f769a3650    3 23           sym._init
0×559f769a3680    1 6            sym.imp.puts
0×559f769a3690    1 6            sym.imp.fread
0×559f769a36a0    1 6            sym.imp.strlen
0×559f769a36b0    1 6            sym.imp.__stack_chk_fail
0×559f769a3000    2 25           map.home_tryhackme_crackme_crackme2.r_x
0×559f769a36c0    1 6            sym.imp.fopen
0×559f769a36d0    1 6            sym.imp.__isoc99_scanf
```

We got a main function, we can call it and debug



```
[0×7f245b003090]> pdf @main
/ (fcn) main 280
|   int main (int argc, char **argv, char **envp);
|           ; var int32_t var_54h @ rbp-0×54
|           ; var int32_t var_50h @ rbp-0×50
|           ; var int32_t var_4ch @ rbp-0×4c
|           ; var int32_t var_48h @ rbp-0×48
|           ; var int32_t var_40h @ rbp-0×40
|           ; var int32_t var_38h @ rbp-0×38
|           ; var int32_t var_30h @ rbp-0×30
|           ; var int32_t var_28h @ rbp-0×28
|           ; var int32_t var_12h @ rbp-0×12
|           ; var int32_t var_8h @ rbp-0×8
|           ; arg int32_t arg_40h @ rbp+0×40
|           ; DATA XREF from entry0 (0×564eff6a170d)
|           0×564eff6a17fa      55              pushq %rbp
|           0×564eff6a17fb      4889e5          movq %rsp, %rbp
|           0×564eff6a17fe      4883ec60        subq $0×60, %rsp        ; ``
|           0×564eff6a1802      64488b042528.   movq %fs:0×28, %rax     ; [0×28:8
|           0×564eff6a180b      488945f8        movq %rax, var_8h
|           0×564eff6a180f      31c0            xorl %eax, %eax
|           0×564eff6a1811      488d3d900100.   leaq str.enter_your_password, %rd
|           0×564eff6a1818      e863feffff      callq sym.imp.puts      ; int put
|           0×564eff6a181d      488d45ee        leaq var_12h, %rax
|           0×564eff6a1821      4889c6          movq %rax, %rsi
|           0×564eff6a1824      488d3d910100.   leaq 0×564eff6a19bc, %rdi ; "%s"
|           0×564eff6a182b      b800000000      movl $0, %eax
|           0×564eff6a1830      e89bfeffff      callq sym.imp.__isoc99_scanf ; in
|           0×564eff6a1835      c745ac000000.   movl $0, var_54h
|           0×564eff6a183c      488d057c0100.   leaq 0×564eff6a19bf, %rax ; "127"
|           0×564eff6a1843      488945c0        movq %rax, var_40h
|           0×564eff6a1847      488d05750100.   leaq str.01., %rax      ; 0×564ef
7\u2074\u6874\u2065\u6f63\u7272\u6365\u2074\u6170\u7373\u6f77\u6472\u0100\u031b\
|           0×564eff6a184e      488945c8        movq %rax, var_38h
|           0×564eff6a1852      488d056a0100.   leaq str.01., %rax      ; 0×564ef
7\u2074\u6874\u2065\u6f63\u7272\u6365\u2074\u6170\u7373\u6f77\u6472\u0100\u031b\
|           0×564eff6a1859      488945d0        movq %rax, var_30h
|           0×564eff6a185d      488d05610100.   leaq 0×564eff6a19c5, %rax ; u"1.\
065\u6f63\u7272\u6365\u2074\u6170\u7373\u6f77\u6472\u0100\u031b\u3c3b"
```

You can see there is a scanf for entering password, got “127” and “.” alongside **callq sym.imp.strtok ; char *strtok(char *s1, const char *s2).**


0x564eff6a1864 488945d8 movq %rax, var_28h
0x564eff6a1868 488d45ee leaq var_12h, %rax
0x564eff6a186c 4889c7 movq %rax, %rdi
0x564eff6a186f e81cfeffff callq sym.imp.strlen ; size_t
0x564eff6a1874 8945b0 movl %eax, var_50h
0x564eff6a1877 488d45ee leaq var_12h, %rax
0x564eff6a187b 488d35450100. leaq 0x564eff6a19c7, %rsi ; "."
0x564eff6a1882 4889c7 movq %rax, %rdi
0x564eff6a1885 e836feffff callq sym.imp.strtok ; char *s
0x564eff6a188a 488945b8 movq %rax, var_48h
0x564eff6a188e eb4e jmp 0x564eff6a18de
0x564eff6a1890 8b45ac movl var_54h, %eax
0x564eff6a1893 4898 cltq
0x564eff6a1895 488b54c5c0 movq -0x40(%rbp, %rax, 8), %rdx
0x564eff6a189a 488b45b8 movq var_48h, %rax
0x564eff6a189e 4889d6 movq %rdx, %rsi
0x564eff6a18a1 4889c7 movq %rax, %rdi
0x564eff6a18a4 e807feffff callq sym.imp.strcmp ; int str
0x564eff6a18a9 8945b4 movl %eax, var_4ch
0x564eff6a18ac 8345ac01 addl $1, var_54h
0x564eff6a18b0 837db400 cmpl $0, var_4ch
0x564eff6a18b4 7413 je 0x564eff6a18c9
0x564eff6a18b6 488d3d0c0100. leaq 0x564eff6a19c9, %rdi ; "Wron
0x564eff6a18bd e8befdffff callq sym.imp.puts ; int put
0x564eff6a18c2 b8ffffffff movl $0xffffffff, %eax ; -1
0x564eff6a18c7 eb33 jmp 0x564eff6a18fc
0x564eff6a18c9 488d35f70000. leaq 0x564eff6a19c7, %rsi ; "."
0x564eff6a18d0 bf00000000 movl $0, %edi
0x564eff6a18d5 e8e6fdffff callq sym.imp.strtok ; char *s
0x564eff6a18da 488945b8 movq %rax, var_48h
; CODE XREF from main (0x564eff6a188e)
0x564eff6a18de 48837db800 cmpq $0, var_48h
0x564eff6a18e3 7406 je 0x564eff6a18eb
0x564eff6a18e5 837dac03 cmpl $3, var_54h
0x564eff6a18e9 7ea5 jle 0x564eff6a1890
0x564eff6a18eb 488d3de60000. leaq str.You_ve_got_the_correct_p
0x564eff6a18f2 e889fdffff callq sym.imp.puts ; int put
0x564eff6a18f7 b800000000 movl $0, %eax
; CODE XREF from main (0x564eff6a18c7)
0x564eff6a18fc 488b4df8 movq var_8h, %rcx
0x564eff6a1900 6448330c2528. xorq %fs:0x28, %rcx
0x564eff6a1909 7405 je 0x564eff6a1910
0x564eff6a190b e890fdffff callq sym.imp.__stack_chk_fail ;
0x564eff6a1910 c9 leave
0x564eff6a1911 c3 retq

```
:|   0x55a305a0989a      488b45b8        movq var_48h, %rax
:|   0x55a305a0989e      4889d6          movq %rdx, %rsi
:|   0x55a305a098a1      4889c7          movq %rax, %rdi
:|   ;-- rip:
:|   0x55a305a098a4 b    e807feffff      callq sym.imp.strcmp    ; int strcmp(const char *s1, const char *s2)
:|   0x55a305a098a9      8945b4          movl %eax, var_4ch
:|   0x55a305a098ac      8345ac01        addl $1, var_54h
```

I'm gonna set a breakpoint in strcmp, why, because usually %rsi and %rdi is for input that will be compared from our input and the password in the program. We can see the binary from %rsi and %rdi :

```
[0x55a305a098a4]> px @rsi
- offset -       0 1  2 3  4 5  6 7  8 9  A B  C D  E F  0123456789ABCDEF
0x55a305a099bf  3132 3700 3000 3100 2e00 5772 6f6e 6720  127.0.1...Wrong
0x55a305a099cf  5061 7373 776f 7264 0059 6f75 2776 6520  Password.You've
0x55a305a099df  676f 7420 7468 6520 636f 7272 6563 7420  got the correct
0x55a305a099ef  7061 7373 776f 7264 0001 1b03 3b3c 0000  password. ... ;<..
0x55a305a099ff  0006 0000 0078 fcff ff88 0000 00e8 fcff  ..... x. .. . ... .. .
0x55a305a09a0f  ffb0 0000 00f8 fcff ff58 0000 0002 feff  .. ... .. ..X ... .. .
0x55a305a09a1f  ffc8 0000 0028 ffff ffe8 0000 0098 ffff  .. ... ( ... . ... . ..
0x55a305a09a2f  ff30 0100 0000 0000 0014 0000 0000 0000  .0..............
0x55a305a09a3f  0001 7a52 0001 7810 011b 0c07 0890 0107  ..zR..x.........
0x55a305a09a4f  1014 0000 001c 0000 0098 fcff ff2b 0000  .. ... . ... .. ..+..
0x55a305a09a5f  0000 0000 0000 0000 0014 0000 0000 0000  ................
0x55a305a09a6f  0001 7a52 0001 7810 011b 0c07 0890 0100  ..zR..x.........
0x55a305a09a7f  0024 0000 001c 0000 00e8 fbff ff70 0000  .$... . ... .. ..p..
0x55a305a09a8f  0000 0e10 460e 184a 0f0b 7708 8000 3f1a  .. ..F..J..w.. .?.
0x55a305a09a9f  3b2a 3324 2200 0000 0014 0000 0044 0000  ;*3$"..... ... D..
0x55a305a09aaf  0030 fcff ff08 0000 0000 0000 0000 0000  .0. .. ..........
```

```
[0x55a305a098a4]> px @rdi
- offset -       0 1  2 3  4 5  6 7  8 9  A B  C D  E F  0123456789ABCDEF
0x7ffc101c230e  6164 6164 6100 fc7f 0000 0073 bb40 bb3e  adada.. ... s.@.>
0x7ffc101c231e  be75 2099 a005 a355 0000 976b 7ad2 f17f  .u ....U..kz..
0x7ffc101c232e  0000 0100 0000 0000 0000 0824 1c10 fc7f  ..........$... .
0x7ffc101c233e  0000 0080 0000 0100 0000 fa97 a005 a355  ... .. . ... .....U
0x7ffc101c234e  0000 0000 0000 0000 0000 0ae8 6b5c b1af  ..........k\..
0x7ffc101c235e  c07f f096 a005 a355 0000 0024 1c10 fc7f  .......U...$... .
0x7ffc101c236e  0000 0000 0000 0000 0000 0000 0000 0000  ................
0x7ffc101c237e  0000 0ae8 4b28 c884 7e2b 0ae8 b5b8 0400  .. ..K( ..~+......
0x7ffc101c238e  652b 0000 0000 fc7f 0000 0000 0000 0000  e+..............
0x7ffc101c239e  0000 0000 0000 0000 0000 3367 b8d2 f17f  ..........3g... .
0x7ffc101c23ae  0000 38c6 b6d2 f17f 0000 3db7 0800 0000  ..8..... ..=.. ...
0x7ffc101c23be  0000 0000 0000 0000 0000 0000 0000 0000  ................
0x7ffc101c23ce  0000 0000 0000 0000 0000 f096 a005 a355  ..............U
0x7ffc101c23de  0000 0024 1c10 fc7f 0000 1a97 a005 a355  ... $... .. .....U
0x7ffc101c23ee  0000 f823 1c10 fc7f 0000 1c00 0000 0000  .. .#... .. ......
0x7ffc101c23fe  0000 0100 0000 0000 0000 7437 1c10 fc7f  .. ........t7... .
```

We can see %rsi is compared with our input in %rdi with adada, the conclude is the password 127.0.0.1

Crackme2

Task 7 crackme2

Analyse the crackme2 binary and try find the correct password, as with the previous question.

Well this is the program sam as cracme1, but maybe a little harder and a bit tricky, still the same, we need to input the correct password, its time to debugging!

```
tryhackme@ip-10-10-142-2:~/crackme$ ./crackme2
Please enter password
adada
Wrong Password
```

Using Radare for the debugging

```
tryhackme@ip-10-10-142-2:~/crackme$ r2 -d crackme2
Process with PID 1308 started...
= attach 1308 1308
bin.baddr 0x55e9f3609000
Using 0x55e9f3609000
asm.bits 64
 -- Documentation is for weak people.
```

First all we need to analyze all of it again using aaa command

```
[0×7fd797f43090]> aaa
[x] Analyze all flags starting with sym. and entry0 (aa)
[Warning: Invalid range. Use different search.in=? or anal.in=d
bg.maps.x
Warning: Invalid range. Use different search.in=? or anal.in=db
g.maps.x
[x] Analyze function calls (aac)
[x] Analyze len bytes of instructions for references (aar)
[x] Check for objc references
[x] Check for vtables
[TOFIX: aaft can't run in debugger mode.ions (aaft)
[x] Type matching analysis for all functions (aaft)
[ ] Use -AA or aaaa to perform additional experimental analysis
[x] Use -AA or aaaa to perform additional experimental analysis
.
```

Using command afl to list all the function in the exe file

```
[0×7fd797f43090]> afl
0×562ad04586f0    1 42           entry0
0×562ad0658fe0    1 4124         reloc.__libc_start_main
0×562ad0458720    4 50   → 40    sym.deregister_tm_clones
0×562ad0458760    4 66   → 57    sym.register_tm_clones
0×562ad04587b0    5 58   → 51    entry.fini0
0×562ad04586e0    1 6            sym..plt.got
0×562ad04587f0    1 10           entry.init0
0×562ad0458990    1 2            sym.__libc_csu_fini
0×562ad0458994    1 9            sym._fini
0×562ad0458920    4 101          sym.__libc_csu_init
0×562ad04587fa   12 283          main
0×562ad0458650    3 23           sym._init
0×562ad0458680    1 6            sym.imp.puts
0×562ad0458690    1 6            sym.imp.fread
0×562ad04586a0    1 6            sym.imp.strlen
0×562ad04586b0    1 6            sym.imp.__stack_chk_fail
0×562ad0458000    2 25           map.home_tryhackme_crackme_cra
ckme2.r_x
0×562ad04586c0    1 6            sym.imp.fopen
0×562ad04586d0    1 6            sym.imp.__isoc99_scanf
```

Lets check the main program, got something interesting, there is some file in /home/tryhackme/install-files/secret.txt, lets open it first!

```
[0x7f1eaed77090]> pdf @main
/ (fcn) main 283
|   int main (int argc, char **argv, char **envp);
|           ; var int32_t var_44h @ rbp-0x44
|           ; var int32_t var_40h @ rbp-0x40
|           ; var int32_t var_3ch @ rbp-0x3c
|           ; var int32_t var_38h @ rbp-0x38
|           ; var int32_t var_2eh @ rbp-0x2e
|           ; var int32_t var_23h @ rbp-0x23
|           ; var int32_t var_18h @ rbp-0x18
|           ; DATA XREF from entry0 (0x559f769a370d)
|           0x559f769a37fa      55              pushq %rbp
|           0x559f769a37fb      4889e5          movq %rsp, %rbp
|           0x559f769a37fe      53              pushq %rbx
|           0x559f769a37ff      4883ec48        subq $0x48, %rsp        ; 'H'
|           0x559f769a3803      64488b042528.   movq %fs:0x28, %rax     ; [0x28:
8]=-1 ; '(' ; 40
|           0x559f769a380c      488945e8        movq %rax, var_18h
|           0x559f769a3810      31c0            xorl %eax, %eax
|           0x559f769a3812      488d358f0100.   leaq 0x559f769a39a8, %rsi ; "r"
|           0x559f769a3819      488d3d900100.   leaq str.home_tryhackme_install_
files_secret.txt, %rdi ; 0x559f769a39b0 ; "/home/tryhackme/install-files/secret
.txt"
|           0x559f769a3820      e89bfeffff      callq sym.imp.fopen     ; file*f
open(const char *filename, const char *mode)
|           0x559f769a3825      488945c8        movq %rax, var_38h
|           0x559f769a3829      488b55c8        movq var_38h, %rdx
|           0x559f769a382d      488d45d2        leaq var_2eh, %rax
|           0x559f769a3831      4889d1          movq %rdx, %rcx
|           0x559f769a3834      ba0b000000      movl $0xb, %edx         ; 11
|           0x559f769a3839      be01000000      movl $1, %esi
|           0x559f769a383e      4889c7          movq %rax, %rdi
|           0x559f769a3841      e84afeffff      callq sym.imp.fread     ; size_t
 fread(void *ptr, size_t size, size_t nmemb, FILE *stream)
|           0x559f769a3846      8945c4          movl %eax, var_3ch
|           0x559f769a3849      837dc400        cmpl $0, var_3ch
|       ,=< 0x559f769a384d      7916            jns 0x559f769a3865
|       |   0x559f769a384f      488d3d830100.   leaq str.Error_Reading_File, %rd
i ; 0x559f769a39d9 ; "Error Reading File"
```

The program read the secret.txt file

```
[0x7fd797f43090]> cat /home/tryhackme/install-files/secret.txt
vs3curepwd
```

Lets try it!

```
tryhackme@ip-10-10-142-2:~/crackme$ ./crackme2
Please enter password
vs3curepwd
Wrong Password
```

Still wrong, we need to see what the program does.

```
|       |   0x559f769a3856      e825feffff     callq sym.imp.puts      ; int pu
ts(const char *s)
|       |   0x559f769a385b      b8ffffffff     movl $0xffffffff, %eax  ; -1
|      ,==< 0x559f769a3860      e995000000     jmp 0x559f769a38fa
|      |`-> 0x559f769a3865      488d3d800100.  leaq str.Please_enter_password,
%rdi ; 0x559f769a39ec ; "Please enter password"
|       |   0x559f769a386c      e80ffeffff     callq sym.imp.puts      ; int pu
ts(const char *s)
|       |   0x559f769a3871      488d45dd       leaq var_23h, %rax
|       |   0x559f769a3875      4889c6         movq %rax, %rsi
|       |   0x559f769a3878      488d3d830100.  leaq str.11s, %rdi      ; 0x559f
769a3a02 ; "%11s"
|       |   0x559f769a387f      b800000000     movl $0, %eax
|       |   0x559f769a3884      e847feffff     callq sym.imp.__isoc99_scanf ; i
nt scanf(const char *format)
|       |   0x559f769a3889      c745bc090000.  movl $9, var_44h
|       |   0x559f769a3890      c745c0000000.  movl $0, var_40h
|       |,=< 0x559f769a3897      eb33           jmp 0x559f769a38cc
|      .--> 0x559f769a3899      8b45bc         movl var_44h, %eax
|      :|   0x559f769a389c      4898           cltq
|      :|   0x559f769a389e      0fb65405d2     movzbl -0x2e(%rbp, %rax), %edx
|      :|   0x559f769a38a3      8b45c0         movl var_40h, %eax
|      :|   0x559f769a38a6      4898           cltq
|      :|   0x559f769a38a8      0fb64405dd     movzbl -0x23(%rbp, %rax), %eax
|      :|   0x559f769a38ad      38c2           cmpb %al, %dl
|     ,===< 0x559f769a38af      7413           je 0x559f769a38c4
|     |:||   0x559f769a38b1      488d3d4f0100.  leaq str.Wrong_Password, %rdi ;
0x559f769a3a07 ; "Wrong Password"
|     |:||   0x559f769a38b8      e8c3fdffff     callq sym.imp.puts      ; int pu
ts(const char *s)
|     |:||   0x559f769a38bd      b8ffffffff     movl $0xffffffff, %eax  ; -1
|    ,====< 0x559f769a38c2      eb36           jmp 0x559f769a38fa
|    |`---> 0x559f769a38c4      836dbc01       subl $1, var_44h
|    | :||   0x559f769a38c8      8345c001       addl $1, var_40h
|    | :||   ; CODE XREF from main (0x559f769a3897)
|    | :|`-> 0x559f769a38cc      837dbc00       cmpl $0, var_44h
|    | :|,=< 0x559f769a38d0      7e17           jle 0x559f769a38e9
|    | :||   0x559f769a38d2      8b45c0         movl var_40h, %eax
|    | :||   0x559f769a38d5      4863d8         movslq %eax, %rbx
```

```
|    | :||   0x559f769a38d8      488d45dd       leaq var_23h, %rax
|    | :||   0x559f769a38dc      4889c7         movq %rax, %rdi
|    | :||   0x559f769a38df      e8bcfdffff     callq sym.imp.strlen    ; size_t
 strlen(const char *s)
|    | :||   0x559f769a38e4      4839c3         cmpq %rax, %rbx
|    | `===< 0x559f769a38e7      72b0           jb 0x559f769a3899
|    |  |`-> 0x559f769a38e9      488d3d260100.  leaq str.Correct_Password, %rdi
; 0x559f769a3a16 ; "Correct Password"
|    |  |    0x559f769a38f0      e88bfdffff     callq sym.imp.puts      ; int pu
ts(const char *s)
|    |  |    0x559f769a38f5      b800000000     movl $0, %eax
|    |  |    ; CODE XREFS from main (0x559f769a3860, 0x559f769a38c2)
|    `--`--> 0x559f769a38fa      488b4de8       movq var_18h, %rcx
|            0x559f769a38fe      6448330c2528.  xorq %fs:0x28, %rcx
|         ,=< 0x559f769a3907      7405           je 0x559f769a390e
|         |   0x559f769a3909      e8a2fdffff     callq sym.imp.__stack_chk_fail ;
 void __stack_chk_fail(void)
|         `-> 0x559f769a390e      4883c448       addq $0x48, %rsp        ; 'H'
|             0x559f769a3912      5b             popq %rbx
|             0x559f769a3913      5d             popq %rbp
\             0x559f769a3914      c3             retq
```

Let See rsi and rdi in this, we got %11s, that means the input is 10 characters long.

```
|       |    0x562ad0458875      4889c6          movq %rax, %rsi
|       |    0x562ad0458878      488d3d830100.   leaq str.11s, %rd
i       ; 0x562ad0458a02 ; "%11s"
```

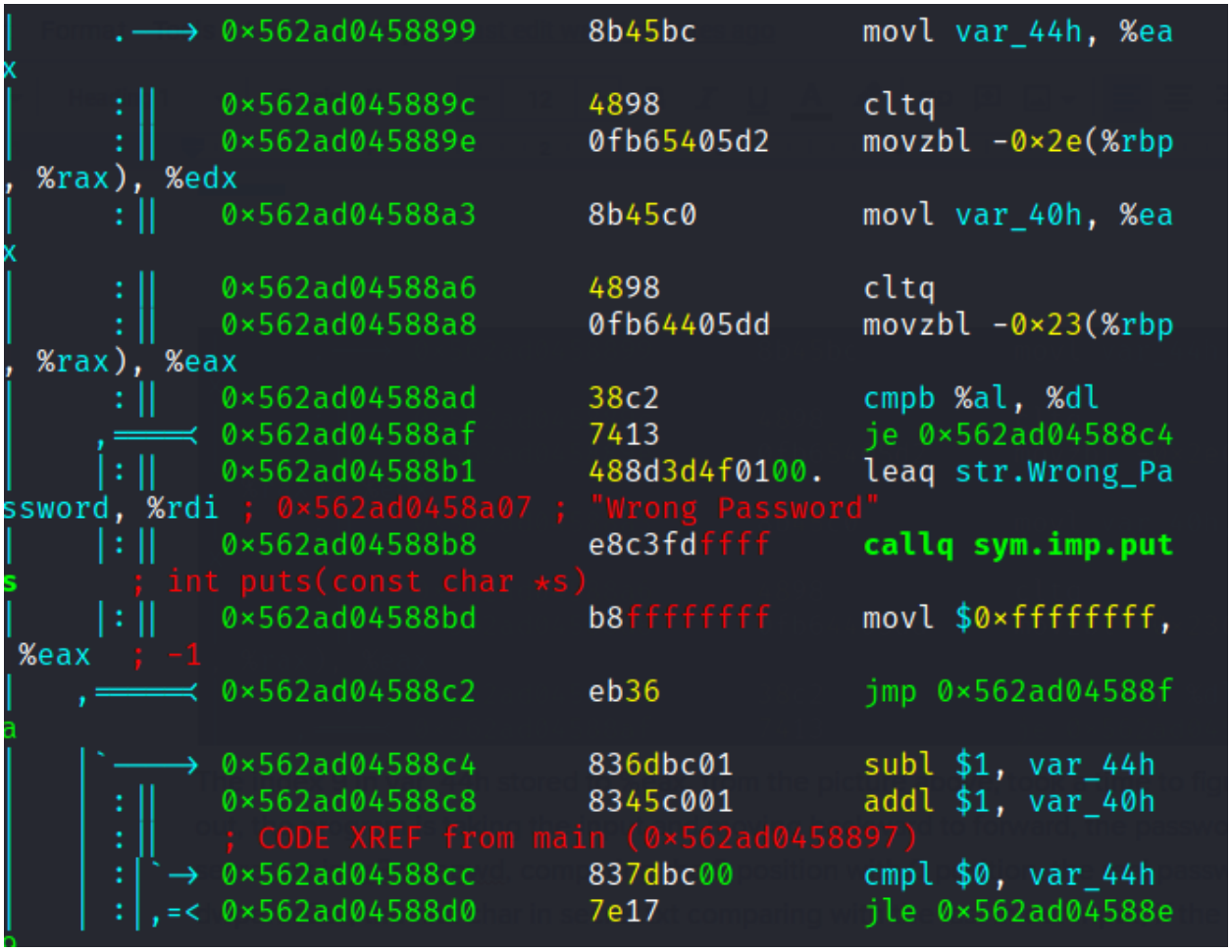
Got this too, and lets jump to 0x562ad04588c

```
|       |    0x562ad0458884      e847feffff      callq sym.imp.__i
soc99_scanf ; int scanf(const char *format)
|       |    0x562ad0458889      c745bc090000.   movl $9, var_44h
|       |    0x562ad0458890      c745c0000000.   movl $0, var_40h
|       |,=< 0x562ad0458897      eb33            jmp 0x562ad04588c
```

Var_44h is compared with 0, you can see 9 is not less than zero, this is false, and then we are not jumping to jle, instead we are going down and do stuff for strlen for length

```
|    | :|`-> 0x562ad04588cc      837dbc00        cmpl $0, var_44h
|a ar| :|,=< 0x562ad04588d0      7e17            jle 0x562ad04588e
9
|    | :||   0x562ad04588d2      8b45c0          movl var_40h, %ea
x
|    | :||   0x562ad04588d5      4863d8          movslq %eax, %rbx
|    | :||   0x562ad04588d8      488d45dd        leaq var_23h, %ra
x
|    | :||   0x562ad04588dc      4889c7          movq %rax, %rdi
|    | :||   0x562ad04588df      e8bcfdffff      callq sym.imp.str
len     ; size_t strlen(const char *s)
|    | :||   0x562ad04588e4      4839c3          cmpq %rax, %rbx
|    | `===< 0x562ad04588e7      72b0            jb 0x562ad0458899
```

After comparing strlen, jumping back to jb 0x562ad0458899



The index 9 in var_44h stored to %eax, from the picture above, took a time to figure it out, the program is taking the input and moving backward to forward, the password in secret.txt is vs3curepwd, comparing the nine positions with zero positions, the real password is dwperuc3sv, the first char in secret.txt compares with the last in our input, if the last input, not v, than the password wrong, and then this is looping until 10 character, so basically the password in secret.txt are the reverse.

```
tryhackme@ip-10-10-142-2:~/crackme$ ./crackme2
Please enter password
dwperuc3sv
Correct Password
```