

Reverse Engineering (crackme1)

Step by Step Procedure to Crack Software

Step 1:

```
(luchifer@luchifer)-[~/ctf/reverse]
$ ./crackme1
Argument 1 Missing.

(luchifer@luchifer)-[~/ctf/reverse]
$ ./crackme1 aass
Wrong Password.

(luchifer@luchifer)-[~/ctf/reverse]
$
```

So, the software takes one argument. If we give the argument (initially, we don't know the correct argument, so we give a random one).

I will be using radare2. You can pick anything you want.

Step 2:

To use radare2 in Linux we use:

```
r2 -d ./filename
```

The we use -> aaa to analyze function name.

```
(luchifer@luchifer)-[~/ctf/reverse]
$ r2 -d ./crackme1
[0x7f0c05984360]> aaa
[x] Analyze all flags starting with sym. and entry0 (aa)
[x] Analyze function calls (aac)
[x] Analyze len bytes of instructions for references (aar)
[x] Finding and parsing C++ vtables (avrr)
[x] Skipping type matching analysis in debugger mode (aaft)
[x] Propagate noreturn information (aanr)
[x] Use -AA or aaaa to perform additional experimental analysis.
```

To know about function in software we use: afl

afl means -> **Analyze Function List**

Then we use : db main

DB means -> **Debugging Mode**

DC means->**Debugging continue**

```
(luchifer@luchifer)-[~/ctf/reverse]
$ r2 -d ./crackme1
[0x7f6b378c2360]> aaa
[x] Analyze all flags starting with sym. and entry0 (aa)
[x] Analyze function calls (aac)
[x] Analyze len bytes of instructions for references (aar)
[x] Finding and parsing C++ vtables (avrr)
[x] Skipping type matching analysis in debugger mode (aaft)
[x] Propagate noreturn information (aanr)
[x] Use -AA or aaaa to perform additional experimental analysis.
[0x7f6b378c2360]> afl
0x004010d0 1 47 entry0
0x00401110 4 33 → 31 sym.deregister_tm_clones
0x00401140 4 49 sym.register_tm_clones
0x00401180 3 33 → 32 sym.__do_global_dtors_aux
0x004011b0 1 6 entry.init0
0x00401330 1 5 sym.__libc_csu_fini
0x00401338 1 13 sym._fini
0x004012c0 4 101 sym.__libc_csu_init
0x00401100 1 5 sym._dl_relocate_static_pie
0x004011b6 10 257 main
0x00401000 3 27 sym._init
0x00401080 1 11 sym.imp.puts
0x00401090 1 11 sym.imp.strlen
0x004010a0 1 11 sym.imp.__stack_chk_fail
0x004010b0 1 11 sym.imp.strcmp
0x004010c0 1 11 sym.imp.exit
[0x7f6b378c2360]> db main
[0x7f6b378c2360]> dc
hit breakpoint at: 0x4011b6
[0x004011b6]> VV
```

we will be using visual graph mode so we use VV for that.

Step 3:

to use command in visual mode type [shift + ;]

```
0x4011b6 [0a]
; DATA XREF from entry0 @ 0x4010f1
257: int main (int argc, char **argv);
; var int64_t var_40h @ rbp-0x40
; var int64_t var_34h @ rbp-0x34
; var int64_t var_30h @ rbp-0x30
; var int64_t var_28h @ rbp-0x28
; var int64_t var_20h @ rbp-0x20
; var int64_t var_1ch @ rbp-0x1c
; var int64_t var_1ah @ rbp-0x1a
; var int64_t var_18h @ rbp-0x18
; arg int argc @ rdi
; arg char **argv @ rsi
endbr64
```

```

push rbp
mov rbp, rsp
push rbx
sub rsp, 0x38
; argc
mov dword [var_34h], edi
; argv
mov qword [var_40h], rsi
mov rax, qword fs:[0x28]
mov qword [var_18h], rax
xor eax, eax
; 'S3uper_S'
movabs rax, 0x535f726570753353
; '3cr3t_Pa'
movabs rdx, 0x61505f7433726333
mov qword [var_30h], rax
mov qword [var_28h], rdx
; '33w0'
mov dword [var_20h], 0x30773333
; 'rd'
mov word [var_1ch], 0x6472
mov byte [var_1ah], 0
cmp dword [var_34h], 1
jg 0x401222

```

we compare var_34h with 1

to know whats the value of '@rbp-0x34' we can use px command

```

:> px@rbp-0x34
- offset -      0 1  2 3  4 5  6 7  8 9  A B  C D  E F  0123456789ABCDEF
0x7ffec17adf0c  0100 0000 3333 7373 7373 7373 0000 0000 00a9 9b88  t_Pa33w0rd... ...
0x7ffec17adf1c  745f 5061 3333 7730 7264 0000 00a9 9b88  t_Pa33w0rd... ...
0x7ffec17adf2c  9131 7be2 0000 0000 0000 0000 58e0 7ac1  .1{.....X.z.
0x7ffec17adf3c  fe7f 0000 0100 0000 0000 0000 cab6 dbc7  .....
0x7ffec17adf4c  4d7f 0000 40e0 7ac1 fe7f 0000 b611 4000  M...@.z... ..@.
0x7ffec17adf5c  0000 0000 4000 4000 0100 0000 58e0 7ac1  ....@.@.... X.z.
0x7ffec17adf6c  fe7f 0000 58e0 7ac1 fe7f 0000 0a98 0123  ....X.z... ..#
0x7ffec17adf7c  94b2 e07e 0000 0000 0000 0000 68e0 7ac1  ...~.....h.z.
0x7ffec17adf8c  fe7f 0000 0000 0000 0000 0000 0020 fcc7  .....
0x7ffec17adf9c  4d7f 0000 0a98 a39d 6130 1d81 0a98 074e  M.....a0.....N
0x7ffec17adfbc  233d 7b80 0000 0000 0000 0000 0000 0000  #{.....
0x7ffec17adfcc  0000 0000 0000 0000 0000 0000 0000 0000  .....
0x7ffec17adfd0  0000 0000 58e0 7ac1 fe7f 0000 00a9 9b88  ....X.z... ..
0x7ffec17adfdc  9131 7be2 0e00 0000 0000 0000 85b7 dbc7  .1{.. .....
0x7ffec17adfec  4d7f 0000 b611 4000 0000 0000 0000 0000  M... ..@.....
0x7ffec17adffc  4d7f 0000 0000 0000 0000 0000 0000 0000  M.....
:>

```

so the value is 1 . so we comparing 1 & (1 meaning the arguments). so, we have to give one argument if we so not give it will print argument missing.

```

(luchiifer@luchiifer)-[~/ctf/reverse]
$ ./crackme1
Argument 1 Missing.

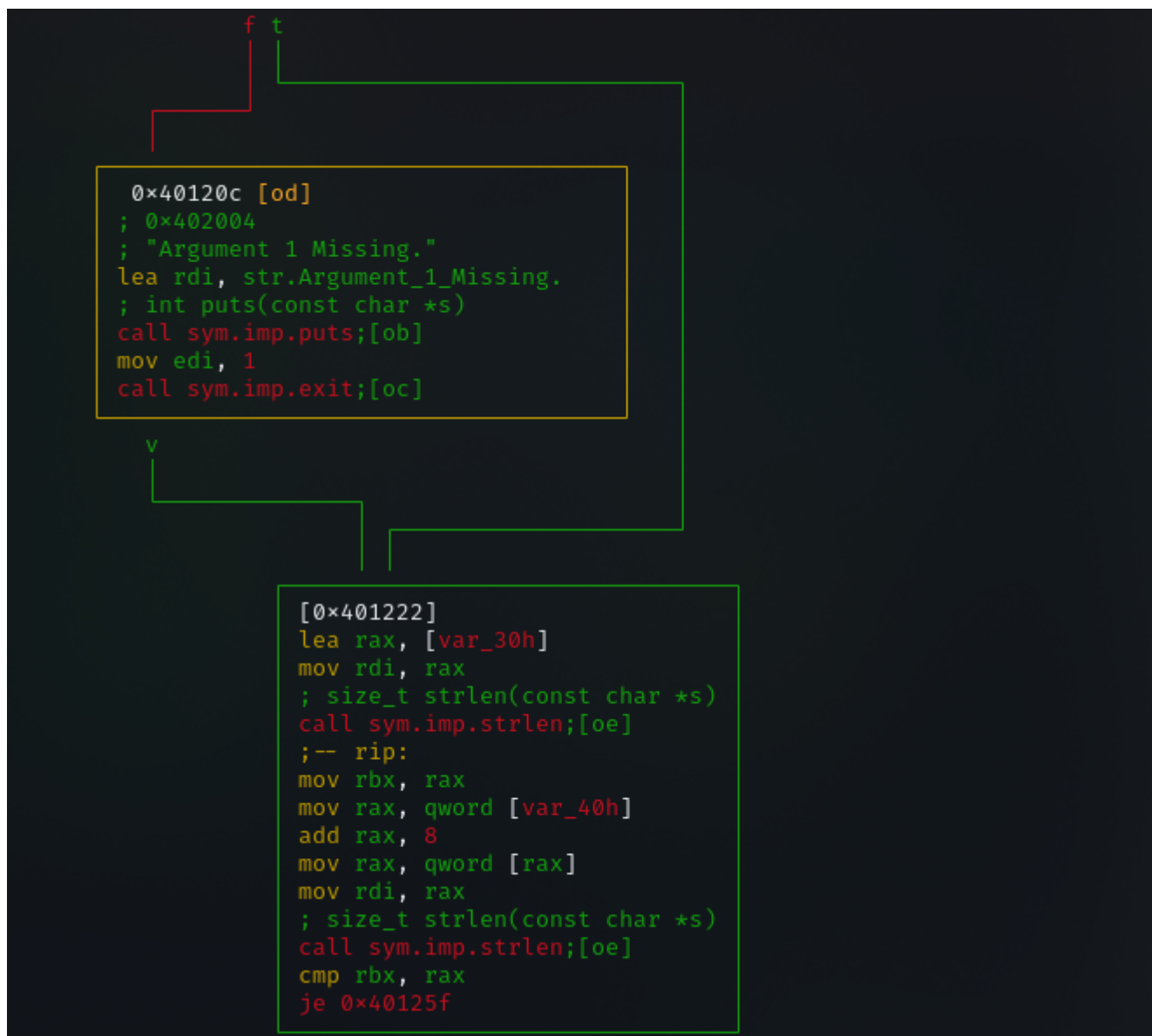
(luchiifer@luchiifer)-[~/ctf/reverse]
$ ./crackme1 aass
Wrong Password.

(luchiifer@luchiifer)-[~/ctf/reverse]
$

```

Step 4:

now lets rerun the program with arguments 9 lenght.



here cmp is comparing password length and our input string length.well how to i know it?

to know about register value value we type -dr

to know about more information we type -drr(hex to decimel value)

```

:> dr
rax = 0x00000010
rbx = 0x00000016
rcx = 0x7f1d6192fb00
rdx = 0x00000000
r8 = 0x00000400
r9 = 0x00000410

```

```

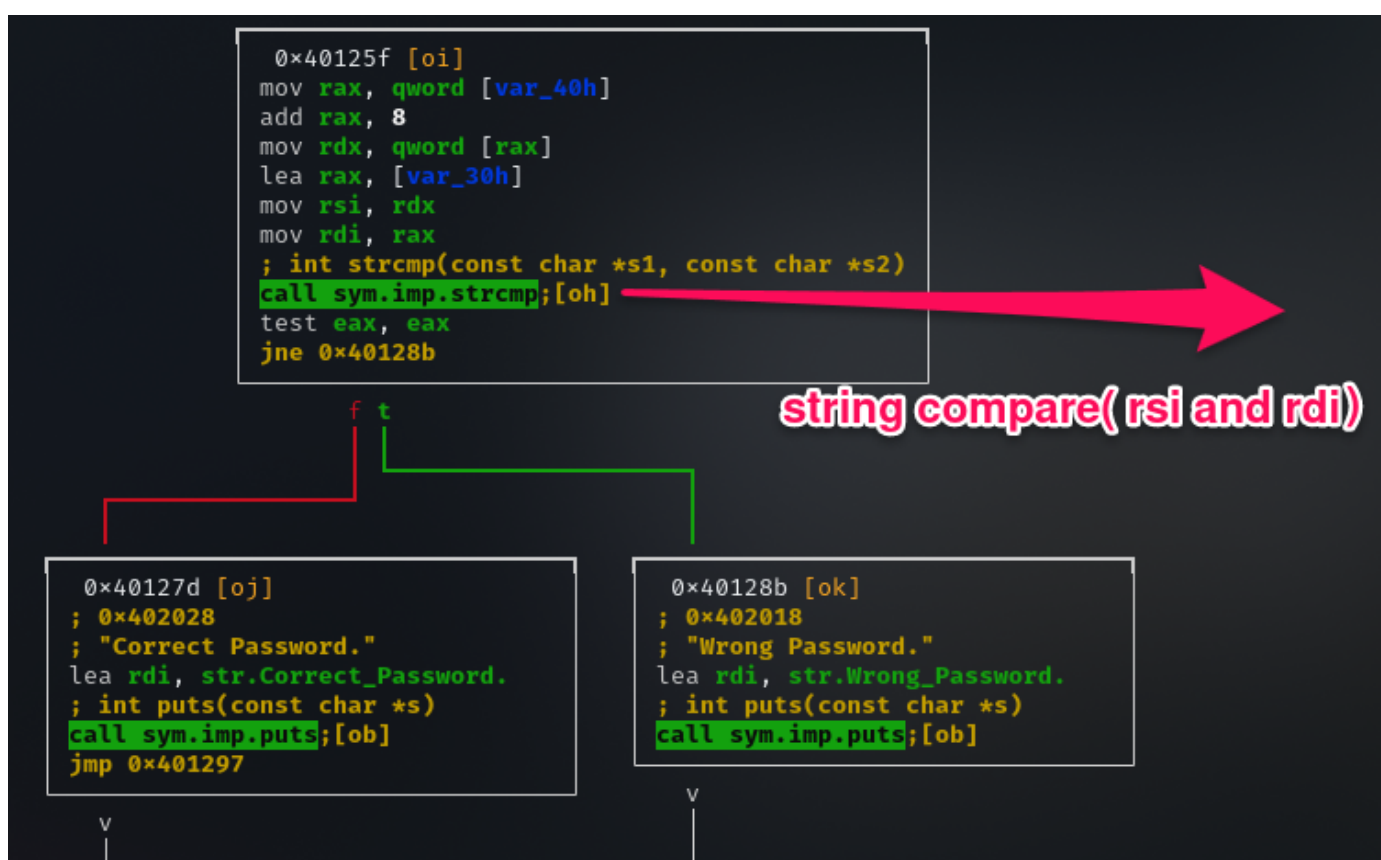
r10 = 0x00001000
r11 = 0x00000202
r12 = 0x00000000
r13 = 0x7ffcde19de70
r14 = 0x00000000
r15 = 0x7f1d61a66000
rsi = 0x00f492a0
rdi = 0x7f1d61a0da30
rsp = 0x7ffcde19dd00
rbp = 0x7ffcde19dd40
rip = 0x00401255
rflags = 0x00000202
orax = 0xffffffffffffffff

```

'rax' is our input string length value and 'rbx' is password length value.(10 and 22 in decimal)

now re-run the program with 22 string argument

now we go to the next level.



if password is same it print correct password if is wrong it prints wrong password.

now we have to know the rsi and rdi value.

we can use `->dr` command.

```

-> dr
role reg value refstr
R0 rax 7ffdc066ba70 [stack] rax,rdi stack R W 0x535f726570753353 S3uper_S3cr3t_Pa33w0rd
R1 rbx 16 22 .comment rbx
A3 rcx 7f33c4b24840 /usr/lib/x86_64-linux-gnu/libc.so.6 rcx library R W 0x7f33c4b26300
A2 rdx 7ffdc066cf0f [stack] rdx,rsi stack R W 0x3837363534333231 1234567890123456789012
A4 r8 401330 4199216 /home/luchifer/ctf/reverse/crackme1 .text __libc_csu_fini,r8 sym.__libc_csu_fini program R X 'endbr64' 'crackme1'
A5 r9 7f33c4b51b10 /usr/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2 r9 library R X 'push r15' 'ld-linux-x86-64.so.2'
r10 7f33c4960328 /usr/lib/x86_64-linux-gnu/libc.so.6 r10 library R 0x10001a00007068 hp
r11 7f33c4aa6140 /usr/lib/x86_64-linux-gnu/libc.so.6 r11 library R X 'mov eax, edi' 'libc.so.6'
r12 0 0
r13 7ffdc066bbd0 [stack] r13 stack R W 0x7ffdc066cf26
r14 0 0
r15 7f33c4b7f000 /usr/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2 r15 library R W 0x7f33c4b802d0
A1 rsi 7ffdc066cf0f [stack] rdx,rsi stack R W 0x3837363534333231 1234567890123456789012
A0 rdi 7ffdc066ba70 [stack] rax,rdi stack R W 0x535f726570753353 S3uper_S3cr3t_Pa33w0rd
SP rsp 7ffdc066ba60 [stack] rsp stack R W 0x7ffdc066bbb8
BP rbp 7ffdc066baa0 [stack] rbp stack R W 0x2
PC rip 401274 4199028 /home/luchifer/ctf/reverse/crackme1 .text rip main program R X 'call 0x4010b0' 'crackme1'
cs 33 51 .symtab .asciiz ('3')
rflags 216 534 .symtab .rflags

```

```

SN  orax  ffffffff
    ss    2b      43 .symtab .ascii ('+')
    fs    7f33c49e740 unk0 R W 0x7f33c49e740
    gs    0
    ds    0
    es    0
    fs_base 0
    gs_base 0

```

here rsi value is **1234567890123456789012**

and rdi value is **S3uper_S3cr3t_Pa33w0rd**

so, we get our password. it is **"S3uper_S3cr3t_Pa33w0rd"**

the program we reverse is

```

#include <stdio.h>
#include <string.h>
#include <stdlib.h>

int main(int argc, char const *argv[])
{
    char password[] = "S3uper_S3cr3t_Pa33w0rd";
    if (argc < 2) {
        puts("Argument 1 Missing.");
        exit(1);
    }
    if (strlen(password) != strlen(argv[1])) {
        puts("Wrong Password.");
        exit(1);
    }
    if (!(strcmp(password, argv[1]))) {
        puts("Correct Password.");
    }
    else {
        puts("Wrong Password.");
    }
    return 0;
}

```

credit: <https://www.youtube.com/watch?v=IXNWmc1u3Kg&t=1481s>