

Report

# Write Up TryHackMe Challenge Reversing ELF

---

Akmal Ilmi

14th April, 2021

## Crackme1

This is the first challenge

### Task 1 Crackme1

Let's start with a basic warmup, can you run the binary?

What is the flag?

With using strings in the crackme1 file, we can see the flag

```
frost@kali:~/Downloads$ file crackme1
crackme1: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically li
nked, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 2.6.32, BuildID[
sha1]=672f525a7ad3c33f190c060c09b11e9ffd007f34, not stripped
frost@kali:~/Downloads$ chmod +x crackme1
frost@kali:~/Downloads$ ./crackme1
flag{not that kind of elf}
```

## Crackme2

### Task 2 Crackme2

Find the super-secret password! and use it to obtain the flag

What is the super secret password ?

What is the flag ?

In this second challenge, got two challenge, the super secret password and search the flag

Using strings again, maybe i can get something

```
frost@kali:~/Downloads$ file crackme2
crackme2: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamical
ly linked, interpreter /lib/ld-linux.so.2, for GNU/Linux 2.6.32, BuildID[sha1
]=b799eb348f3df15f6b08b3c37f8feb269a60aba7, not stripped
frost@kali:~/Downloads$ chmod +x crackme2
frost@kali:~/Downloads$ ./crackme2
Usage: ./crackme2 password
frost@kali:~/Downloads$ strings crackme2
/lib/ld-linux.so.2
libc.so.6
_IO_stdin_used
puts
printf
memset
strcmp
__libc_start_main
/usr/local/lib:$ORIGIN
__gmon_start__
GLIBC_2.0
PTRh
j3jA
[^_]
UWVS
t$,U
[^_]
Usage: %s password
super_secret_password
Access denied.
Access granted.
;*2$(
GCC: (Ubuntu 5.4.0-6ubuntu1~16.04.9) 5.4.0 20160609
crtstuff.c
```

The super password is `super_secret_password`, that is the password, i try and put the password, and the flag is shown up

```
frost@kali:~/Downloads$ ./crackme2 super_secret_password
Access granted.
flag{if_i_submit_this_flag_then_i_will_get_points}
```

## Crackme3

### Task 3 Crackme3

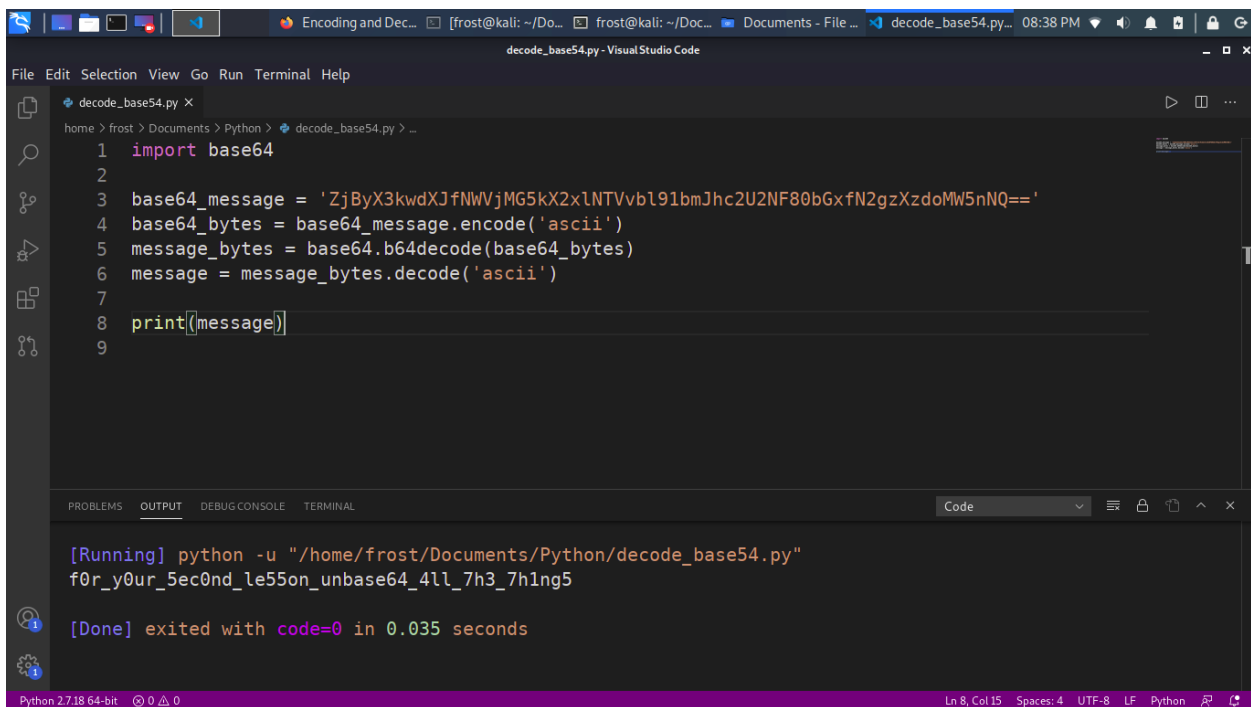
Use basic reverse engineering skills to obtain the flag

What is the flag?

The question is similar with crackme2, search it using strings crackme3

```
104$  
D$;,D$  
UWVS  
[^_]  
Usage: %s PASSWORD  
malloc failed  
ZjByX3kwdXJfNWVjMG5kX2xlNTVvb191bmJhc2U2NF80bGxfN2gzXzdoMW5nNQ==  
Correct password!  
Come on, even my aunt Mildred got this one!  
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/  
;*2$"8
```

Got the flag, but it encrypted in base64, let's decode



```
1 import base64  
2  
3 base64_message = 'ZjByX3kwdXJfNWVjMG5kX2xlNTVvb191bmJhc2U2NF80bGxfN2gzXzdoMW5nNQ=='  
4 base64_bytes = base64_message.encode('ascii')  
5 message_bytes = base64.b64decode(base64_bytes)  
6 message = message_bytes.decode('ascii')  
7  
8 print(message)  
9
```

```
[Running] python -u "/home/frost/Documents/Python/decode_base54.py"  
f0r_y0ur_5ec0nd_le55on_unbase64_4ll_7h3_7hing5  
[Done] exited with code=0 in 0.035 seconds
```

the flag : f0r\_y0ur\_5ec0nd\_le55on\_unbase64\_4ll\_7h3\_7hing5

## Crackme4

### Task 4 Crackme4

Analyze and find the password for the binary?

What is the password ?

Using radare2 to see the password, the command is `r2 -d crackme4`

```
[0x7f6f9a2a2090]> pdf @main
; DATA XREF from entry0 @ 0x40055d
74: int main (int argc, char **argv, char **envp);
; var int64_t var_10h @ rbp-0x10
; var int64_t var_4h @ rbp-0x4
; arg int argc @ rdi
; arg char **argv @ rsi
0x00400716 55          push rbp
0x00400717 4889e5      mov rbp, rsp
0x0040071a 4883ec10    sub rsp, 0x10
0x0040071e 897dfc      mov dword [var_4h], edi ; argc
0x00400721 488975f0    mov qword [var_10h], rsi ; argv
0x00400725 837dfc02    cmp dword [var_4h], 2
0x00400729 741b        je 0x400746
0x0040072b 488b45f0    mov rax, qword [var_10h]
0x0040072f 488b80      mov rax, qword [rax]
0x00400732 4889c6      mov rsi, rax
0x00400735 bf10084000 mov edi, str.Usage: __s_password__This_time_the_string_is_hidden_and_we_used_strcmp ; 0x400810 ; "Usage :
time the string is hidden and we used strcmp\n"
0x0040073a b800000000 mov eax, 0
0x0040073f e8bcbfdfff call sym.imp.printf ; int printf(const char *format)
0x00400744 eb13        jmp 0x400759
0x00400746 488b45f0    mov rax, qword [var_10h]
0x0040074a 4883c008    add rax, 8
0x0040074e 488b80      mov rax, qword [rax]
0x00400751 4889c7      mov rdi, rax
0x00400754 e821ffff call sym.compare_pwd
; CODE XREF from main @ 0x400744
0x00400759 b800000000 mov eax, 0
0x0040075e c9          leave
0x0040075f c3          ret
```

Can see `sym.compare_pwd`, maybe can see there is `strcmp` function

```
[0x7f6f9a182639]> pdf @sym.compare_pwd
; CALL XREF from main @ 0x400754
156: sym.compare_pwd (int64_t arg1);
; var int64_t var_28h @ rbp-0x28
; var int64_t var_20h @ rbp-0x20
; var int64_t var_18h @ rbp-0x18
; var int64_t var_10h @ rbp-0x10
; var int64_t var_0eh @ rbp-0xe
; var int64_t var_8h @ rbp-0x8
; arg int64_t arg1 @ rdi
0x0040067a 55 push rbp
0x0040067b 4889e5 mov rbp, rsp
0x0040067e 4883ec30 sub rsp, 0x30
0x00400682 48897dd8 mov qword [var_28h], rdi ; arg1
0x00400686 64488b042528 mov rax, qword fs:[0x28]
0x0040068f 488945f8 mov qword [var_8h], rax
0x00400693 31c0 xor eax, eax
0x00400695 48b8495d7b49 movabs rax, 0x7b175614497b5d49
0x0040069f 488945e0 mov qword [var_20h], rax
0x004006a3 48b857414751 movabs rax, 0x547b175651474157
0x004006ad 488945e8 mov qword [var_18h], rax
0x004006b1 66c745f05340 mov word [var_10h], 0x4053 ; 'Sq'
0x004006b7 c645f200 mov byte [var_0eh], 0
0x004006bb 488d45e0 lea rax, qword [var_20h]
0x004006bf 4889c7 mov rdi, rax
0x004006c2 e866ffff call sym.get_pwd
0x004006c7 488b55d8 mov rdx, qword [var_28h]
0x004006cb 488d45e0 lea rax, qword [var_20h]
0x004006cf 4889d6 mov rsi, rdx
0x004006d2 4889c7 mov rdi, rax
0x004006d5 b e846feffff call sym.imp.strcmp ; int strcmp(const char *s1, const char *s2)
0x004006da 85c0 test eax, eax
```

Yeah and that's the strcmp function, let's set a breakpoint, db 0x004006d5 and then check the rax and rdx

```
[0x7f6f9a182639]> ood 'argument'
PTRACE_CONT: No such process
child received signal 9
Process with PID 14703 started...
= attach 14703 14703
File dbg:///home/frost/Downloads/crackme4 'argument' reopened in read-write mode
Unable to find filedescriptor 3
Unable to find filedescriptor 3
14703
[0x7f50e6819090]> dc
hit breakpoint at: 4006d5
```

We are using ood 'argument' to pass argument into the strcmp and compare it with the password that stored in rax

```
0x004006b7 c645f200 mov byte [var_0eh], 0
0x004006bb 488d45e0 lea rax, qword [var_20h]
0x004006bf 4889c7 mov rdi, rax
0x004006c2 e866ffff call sym.get_pwd
0x004006c7 488b55d8 mov rdx, qword [var_28h]
0x004006cb 488d45e0 lea rax, qword [var_20h]
0x004006cf 4889d6 mov rsi, rdx
0x004006d2 4889c7 mov rdi, rax
;-- rip:
0x004006d5 b e846feffff call sym.imp.strcmp ; int strcmp(const char *s1, const char *s2)
0x004006da 85c0 test eax, eax
0x004006dc 750c jne 0x4006ea
0x004006de bfe8074000 mov edi, str.password_OK ; 0x4007e8 ; "password OK"
0x004006e3 e8f8fdffff call sym.imp.puts ; int puts(const char *s)
0x004006e8 eb16 jmp 0x400700
0x004006ea 488b55d8 mov rax, qword [var_28h]
```


```

[0x004006d5]> px @rax
- offset -      0 1 2 3 4 5 6 7 8 9 A B C D E F 0123456789ABCDEF
0x7ffeddfb1160 6d79 5f6d 3072 335f 7365 6375 7233 5f70 my_m0r3_secur3_p
0x7ffeddfb1170 7764 0000 0000 0000 0064 9c2a 10da eb86 wd.....d.*....
0x7ffeddfb1180 a011 fbdd fe7f 0000 5907 4000 0000 0000 .....Y.ð.....
0x7ffeddfb1190 9812 fbdd fe7f 0000 0000 0000 0200 0000 .....
0x7ffeddfb11a0 6007 4000 0000 0000 0a4d 65e6 507f 0000 `..ð.....Me.P..
0x7ffeddfb11b0 9812 fbdd fe7f 0000 0000 0000 0200 0000 .....
0x7ffeddfb11c0 1607 4000 0000 0000 0000 0000 0000 0000 ..ð.....
0x7ffeddfb11d0 0000 0000 0000 0000 d4be badd eba3 a6b8 .....
0x7ffeddfb11e0 4005 4000 0000 0000 0000 0000 0000 0000 ð.ð.....
0x7ffeddfb11f0 0000 0000 0000 0000 0000 0000 0000 0000 .....
0x7ffeddfb1200 d4be 1af0 9d18 5b47 d4be fc4a a16f 0746 .....[G...J.o.F
0x7ffeddfb1210 0000 0000 0000 0000 0000 0000 0000 0000 .....
0x7ffeddfb1220 0000 0000 0000 0000 0200 0000 0000 0000 .....
0x7ffeddfb1230 9812 fbdd fe7f 0000 b012 fbdd fe7f 0000 .....
0x7ffeddfb1240 8041 84e6 507f 0000 0000 0000 0000 0000 .A..P.....
0x7ffeddfb1250 0000 0000 0000 0000 4005 4000 0000 0000 .....ð.ð.....

```

And that's it, The password is `my_m0r3_secur3_pwd`

## Crackme5

Task 5  Crackme5

What will be the input of the file to get output `Good game ?`

What is the input ?

AS usually using radare2 to see what is the password, and got some interesting words

```

0x0040078b 488945f8 mov qword [var_8h], rax
0x0040078f 31c0 xor eax, eax
0x00400791 c645d04f mov byte [var_30h], 0x4f ; 'O' ; 79
0x00400795 c645d166 mov byte [var_2fh], 0x66 ; 'f' ; 102
0x00400799 c645d264 mov byte [var_2eh], 0x64 ; 'd' ; 100
0x0040079d c645d36c mov byte [var_2dh], 0x6c ; 'l' ; 108
0x004007a1 c645d444 mov byte [var_2ch], 0x44 ; 'D' ; 68
0x004007a5 c645d553 mov byte [var_2bh], 0x53 ; 'S' ; 83
0x004007a9 c645d641 mov byte [var_2ah], 0x41 ; 'A' ; 65
0x004007ad c645d77c mov byte [var_29h], 0x7c ; '|' ; 124
0x004007b1 c645d833 mov byte [var_28h], 0x33 ; '3' ; 51
0x004007b5 c645d974 mov byte [var_27h], 0x74 ; 't' ; 116
0x004007b9 c645da58 mov byte [var_26h], 0x58 ; 'X' ; 88
0x004007bd c645db62 mov byte [var_25h], 0x62 ; 'b' ; 98
0x004007c1 c645dc33 mov byte [var_24h], 0x33 ; '3' ; 51
0x004007c5 c645dd32 mov byte [var_23h], 0x32 ; '2' ; 50
0x004007c9 c645de7e mov byte [var_22h], 0x7e ; '~' ; 126
0x004007cd c645df58 mov byte [var_21h], 0x58 ; 'X' ; 88
0x004007d1 c645e033 mov byte [var_20h], 0x33 ; '3' ; 51
0x004007d5 c645e174 mov byte [var_1fh], 0x74 ; 't' ; 116
0x004007d9 c645e258 mov byte [var_1eh], 0x58 ; 'X' ; 88
0x004007dd c645e340 mov byte [var_1dh], 0x40 ; '@' ; 64
0x004007e1 c645e473 mov byte [var_1ch], 0x73 ; 's' ; 115
0x004007e5 c645e558 mov byte [var_1bh], 0x58 ; 'X' ; 88
0x004007e9 c645e660 mov byte [var_1ah], 0x60 ; '`' ; 96
0x004007ed c645e734 mov byte [var_19h], 0x34 ; '4' ; 52
0x004007f1 c645e874 mov byte [var_18h], 0x74 ; 't' ; 116
0x004007f5 c645e958 mov byte [var_17h], 0x58 ; 'X' ; 88
0x004007f9 c645ea74 mov byte [var_16h], 0x74 ; 't' ; 116
0x004007fd c645eb7a mov byte [var_15h], 0x7a ; 'z' ; 122
0x00400801 bf54094000 mov edi, str.Enter_your_input ; 0x400954 ; "Enter your input:"
0x00400806 e865fdffff call sym.imp.puts ; int puts(const char *s)
0x0040080b 488d45b0 lea rax, qword [var_50h]

```

Just lineup and sort from top to bottom, and got the password :

OfdIDSAI3txb32~x3tx@sx'4tXtz

Or you can set breakpoint in strcmp function

```

;-- rip:
0x0040082f b e8a2feffff call sym.strcmp ; int strcmp(const char *s1, const char *s2)
0x00400834 8945ac mov dword [var_54h], eax
0x00400837 837dac00 cmp dword [var_54h], 0
0x0040083b 750c jne 0x400849
0x0040083d bf69094000 mov edi, str.Good_game ; 0x400969 ; "Good game"
0x00400842 e829fdffff call sym.imp.puts ; int puts(const char *s)
0x00400847 eb0a jmp 0x400853
0x00400849 bf73094000 mov edi, str.Always_dig_deeper ; 0x400973 ; "Always dig deeper"
0x0040084e e81dfdffff call sym.imp.puts ; int puts(const char *s)
; CODE XREF from main @ 0x400847
0x00400853 b800000000 mov eax, 0
0x00400858 488b4df8 mov rcx, qword [var_8h]
0x0040085c 6448330c2528 xor rcx, qword fs:[0x28]
0x00400865 7405 je 0x40086c
0x00400867 e824fdffff call sym.imp.__stack_chk_fail ; void __stack_chk_fail(void)
0x0040086c c9 leave
0x0040086d c3 ret

```



And px the rdx, to see the password

```
[0x0040082f]> px @rdx
- offset -      0 1  2 3  4 5  6 7  8 9  A B  C D  E F  0123456789ABCDEF
0x7ffc0bf03fd0  4f66 646c 4453 417c 3374 5862 3332 7e58 0fdlDSA|3tXb32~X
0x7ffc0bf03fe0  3374 5840 7358 6034 7458 747a 0000 0000 3tX@sX`4tXtz....
0x7ffc0bf03ff0  f040 f00b fc7f 0000 00b5 2b89 0be6 0d63 .@... ..+. ...c
0x7ffc0bf04000  d008 4000 0000 0000 0afd 33f3 8b7f 0000 .. @..... 3 .. ..
0x7ffc0bf04010  f840 f00b fc7f 0000 0000 0000 0000 0100 0000 .@... ..
0x7ffc0bf04020  7307 4000 0000 0000 0000 0000 0000 0000 s.@.....
0x7ffc0bf04030  0000 0000 0000 0000 9953 94d4 0247 8b22 .....S... G."
0x7ffc0bf04040  e005 4000 0000 0000 0000 0000 0000 0000 .. @.....
0x7ffc0bf04050  0000 0000 0000 0000 0000 0000 0000 0000 .....
0x7ffc0bf04060  9953 1445 6250 73dd 9953 b23c e5a1 9cdd .S.EbPs.. S.<....
0x7ffc0bf04070  0000 0000 0000 0000 0000 0000 0000 0000 .....
0x7ffc0bf04080  0000 0000 0000 0000 0100 0000 0000 0000 .....
0x7ffc0bf04090  f840 f00b fc7f 0000 0841 f00b fc7f 0000 .@... ..A... ..
0x7ffc0bf040a0  80f1 52f3 8b7f 0000 0000 0000 0000 0000 .. R.. .....
0x7ffc0bf040b0  0000 0000 0000 0000 e005 4000 0000 0000 ..... @.....
0x7ffc0bf040c0  f040 f00b fc7f 0000 0000 0000 0000 0000 .@... ..
```

Same but more convenient not to sort from top to bottom, the password :

OfdlDSA|3tXb32~x3tx@sx'4tXtz

## Crackme6

### Task 6 Crackme6

Analyze the binary for the easy password

What is the password ?

```
[0x7f9a11180090]> pdf @main
; DATA XREF from entry0 @ 0x4004ad
74: int main (int argc, char **argv, char **envp);
; var int64_t var_10h @ rbp-0x10
; var int64_t var_4h @ rbp-0x4
; arg int argc @ rdi
; arg char **argv @ rsi
0x00400711 55          push rbp
0x00400712 4889e5      mov rbp, rsp
0x00400715 4883ec10    sub rsp, 0x10
0x00400719 897dfc      mov dword [var_4h], edi ; argc
0x0040071c 488975f0    mov qword [var_10h], rsi ; argv
0x00400720 837dfc02    cmp dword [var_4h], 2
0x00400724 741b        je 0x400741
0x00400726 488b45f0    mov rax, qword [var_10h]
0x00400728 488b00      mov rax, qword [rax]
0x0040072d 4889c6      mov rsi, rax
0x00400730 bf10084000 mov edi, str.Usage: __s_password__Good_luck__read_the_source ; 0x400810 ; "Usage : %s password\nGood luck"
; ""
0x00400735 b800000000 mov eax, 0
0x0040073a e821fdffff call sym.imp.printf ; int printf(const char *format)
0x0040073f eb13        jmp 0x400754
0x00400741 488b45f0    mov rax, qword [var_10h]
0x00400745 4883c008    add rax, 8
0x00400749 488b00      mov rax, qword [rax]
0x0040074c 4889c7      mov rdi, rax
0x0040074f e87dffff call sym.compare_pwd
; CODE XREF from main @ 0x40073f
0x00400754 b800000000 mov eax, 0
0x00400759 c9          leave
0x0040075a c3          ret
```

Got some interesting compare\_pwd function. Let see what is the function

```
[0x7f9a11180090]> pdf @sym.compare_pwd
; CALL XREF from main @ 0x40074f
64: sym.compare_pwd (int64_t arg1);
; var int64_t var_8h @ rbp-0x8
; arg int64_t arg1 @ rdi
0x004006d1 55          push rbp
0x004006d2 4889e5      mov rbp, rsp
0x004006d5 4883ec10    sub rsp, 0x10
0x004006d9 48897df8    mov qword [var_8h], rdi ; arg1
0x004006dd 488b45f8    mov rax, qword [var_8h]
0x004006e1 4889c7      mov rdi, rax
0x004006e4 e894feffff call sym.my_secure_test
0x004006e9 85c0        test eax, eax
0x004006eb 750c        jne 0x4006f9
0x004006ed bfe8074000 mov edi, str.password_OK ; 0x4007e8 ; "password OK"
0x004006f2 e859fdffff call sym.imp.puts ; int puts(const char *s)
0x004006f7 eb16        jmp 0x40070f
0x004006f9 488b45f8    mov rax, qword [var_8h]
0x004006fd 4889c6      mov rsi, rax
0x00400700 bff4074000 mov edi, str.password__s_not_OK ; 0x4007f4 ; "password \"%s\" not OK\n"
0x00400705 b800000000 mov eax, 0
0x0040070a e851fdffff call sym.imp.printf ; int printf(const char *format)
; CODE XREF from sym.compare_pwd @ 0x4006f7
0x0040070f c9          leave
0x00400710 c3          ret
```

Well got interesting my\_secure\_test function, let's check it

```

[0x7f198b80d090]> pdf @sym.my_secure_test
; CALL XREF from sym.compare_pwd @ 0x4006e4
340: sym.my_secure_test (int64_t arg1);
; var int64_t var_8h @ rbp-0x8
; arg int64_t arg1 @ rdi
0x0040057d 55 push rbp
0x0040057e 4889e5 mov rbp, rsp
0x00400581 4889df8 mov qword [var_8h], rdi ; arg1
0x00400585 488b45f8 mov rax, qword [var_8h]
0x00400589 0fb600 movzx eax, byte [rax]
0x0040058c 84c0 test al, al
0x0040058e 740b je 0x40059b
0x00400590 488b45f8 mov rax, qword [var_8h]
0x00400594 0fb600 movzx eax, byte [rax]
0x00400597 3c31 cmp al, 0x31 ; 49
0x00400599 740a je 0x4005a5
0x0040059b b8ffffff mov eax, 0xffffffff ; -1
0x004005a0 e92a010000 jmp 0x4006cf
0x004005a5 488b45f8 mov rax, qword [var_8h]
0x004005a9 4883c001 add rax, 1
0x004005ad 0fb600 movzx eax, byte [rax]
0x004005b0 84c0 test al, al
0x004005b2 740f je 0x4005c3
0x004005b4 488b45f8 mov rax, qword [var_8h]
0x004005b8 4883c001 add rax, 1
0x004005bc 0fb600 movzx eax, byte [rax]
0x004005bf 3c33 cmp al, 0x33 ; 51
0x004005c1 740a je 0x4005cd
0x004005c3 b8ffffff mov eax, 0xffffffff ; -1
0x004005c8 e902010000 jmp 0x4006cf

```

It's look like some of password that must be sort, can use command :

```
[0x7f9a11180090]> VV @sym.my_secure_test
```

The command is to see the flow of the assembly, after that can sort it from top do down to get the password/flag

```

0x400590 [ob]
mov rax, qword [var_8h]
movzx eax, byte [rax]
; 49
cmp al, 0x31
je 0x4005a5

```

```

0x4005b4 [oe]
mov rax, qword [var_8h]
add rax, 1
movzx eax, byte [rax]
; 51
cmp al, 0x33
je 0x4005cd

```

```

0x4005dc [oh]
mov rax, qword [var_8h]
add rax, 2
movzx eax, byte [rax]
; 51
cmp al, 0x33
je 0x4005f5

```

```

0x400604 [ok]
mov rax, qword [var_8h]
add rax, 3
movzx eax, byte [rax]
; 55
cmp al, 0x37
je 0x40061d

```

```

0x40062c [on]
mov rax, qword [var_8h]
add rax, 4
movzx eax, byte [rax]
; 95
cmp al, 0x5f
je 0x400645

```

```

0x400654 [oq]
mov rax, qword [var_8h]
add rax, 5
movzx eax, byte [rax]
; 112
cmp al, 0x70
je 0x40066a

```

```

0x400679 [ot]
mov rax, qword [var_8h]
add rax, 6
movzx eax, byte [rax]
; 119
cmp al, 0x77
je 0x40068f

```

```

0x40069e [ow]
mov rax, qword [var_8h]
add rax, 7
movzx eax, byte [rax]
; 100
cmp al, 0x64
je 0x4006b4

```

The password is : 313333375f707764, let's convert to ascii

```

>>> "313333375f707764".decode("hex")
'1337_pwd'

```

The password is : 1337\_pwd

## Crackme7

### Task 7 Crackme7

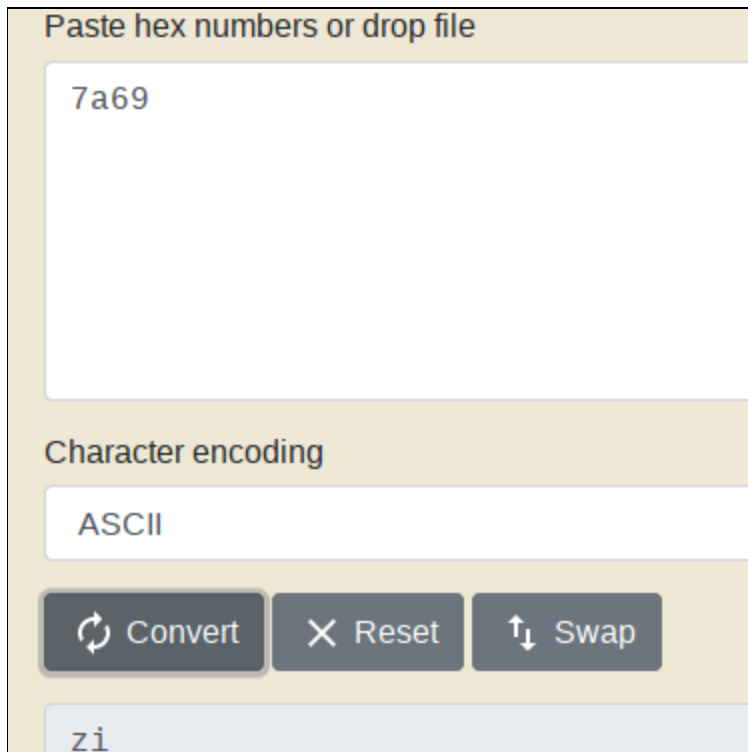
Analyze the binary to get the flag

What is the flag ?

```
frost@kali:~/Downloads$ ./crackme7
Menu:
[1] Say hello
[2] Add numbers
[3] Quit
```

Some sort of program, but the task is to search the flag, let's reverse engineer this code

Got some weird and interesting string wow such h4xor, got cmp eax and 0x7a69, let's convert to ascii text



The image shows a web-based hex-to-ASCII conversion tool. It has a title bar that says "Paste hex numbers or drop file". Below this is a large text input area containing the hex string "7a69". Underneath the input area is a section titled "Character encoding" with a dropdown menu currently set to "ASCII". At the bottom of this section are three buttons: "Convert" (with a circular arrow icon), "Reset" (with an 'X' icon), and "Swap" (with a double-headed arrow icon). Below these buttons is a small output box containing the text "zi".

Let's input in the program

```
frost@kali:~/Downloads$ ./crackme7
Menu:

[1] Say hello
[2] Add numbers
[3] Quit

[>] zi
Unknown input!
```

Still unknown, let's convert to decimal, the decimal is 31337, let's try the input again

```
frost@kali:~/Downloads$ ./crackme7
Menu:

[1] Say hello
[2] Add numbers
[3] Quit

[>] 31337
Wow such h4x0r!
flag{much_reversing_very_ida_wow}
```

The flag is `flag{much_reverse_very_uda_wow}`

## Crackme8

### Task 8 Crackme8

Analyze the binary and obtain the flag

What is the flag ?

```
[0xf7f060b0]> pdf @main
; DATA XREF from entry0 @ 0x80483b7
137: int main (int argc, char **argv, char **envp);
; var int32_t var_4h @ ebp-0x4
; arg int32_t arg_4h @ esp+0x24
0x0804840b 8d4c2404 lea ecx, dword [arg_4h]
0x0804840f 83e4f0 and esp, 0xffffffff
0x080484a2 ff71fc push dword [ecx - 4]
0x080484a5 55 push ebp
0x080484a6 89e5 mov ebp, esp
0x080484a8 51 push ecx
0x080484a9 83ec04 sub esp, 4
0x080484ac 89c8 mov eax, ecx
0x080484ae 833802 cmp dword [eax], 2
0x080484b1 741d je 0x80484d0
0x080484b3 8b4004 mov eax, dword [eax + 4]
0x080484b6 8b00 mov eax, dword [eax]
0x080484b8 83ec08 sub esp, 8
0x080484bb 50 push eax
0x080484bc 6860860408 push str.Usage: __s_password ; 0x8048660 ; "Usage: %s password\n"
0x080484c1 e87afeffff call sym.imp.printf ; int printf(const char *format)
0x080484c6 83c410 add esp, 0x10
0x080484c9 b801000000 mov eax, 1
0x080484ce eb4c jmp 0x804851c
0x080484d0 8b4004 mov eax, dword [eax + 4]
0x080484d3 83c004 add eax, 4
0x080484d6 8b00 mov eax, dword [eax]
0x080484d8 83ec0c sub esp, 0xc
0x080484db 50 push eax
0x080484dc e89ffeffff call sym.imp.atoi ; int atoi(const char *str)
0x080484e1 83c410 add esp, 0x10
0x080484e4 3d0df0feca cmp eax, 0xcafef00d
```

As you can see there is compare eax and 0xcafef00d, maybe that is the password, let's

convert to decimal

The image shows a web-based converter interface with a light yellow background. At the top, it says "Enter hex number" above a text input field containing "cafef00d". To the right of the input field is a small grey box with the number "16". Below the input field are three buttons: a blue button with a blue border labeled "= Convert", a grey button labeled "× Reset", and a grey button labeled "↕ Swap". Below these buttons, it says "Decimal number" above a large light grey text input field containing "3405705229". To the right of this field is a small grey box with the number "10". Below that, it says "Decimal from signed 2's complement" above another large light grey text input field containing "-889262067". To the right of this field is a small grey box with the number "10".

Let's try it!

```
frost@kali:~/Downloads$ ./crackme8 3405705229
Access denied.
frost@kali:~/Downloads$ ./crackme8 -889262067
Access granted.
flag{at_least_this_cafe_wont_leak_your_credit_card_numbers}
```

The flag is : `flag{at_least_this_cafe_wont_leak_your_credit_card_numbers}`