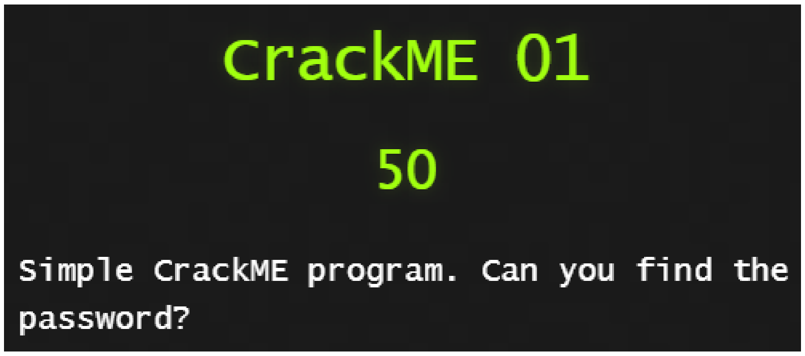
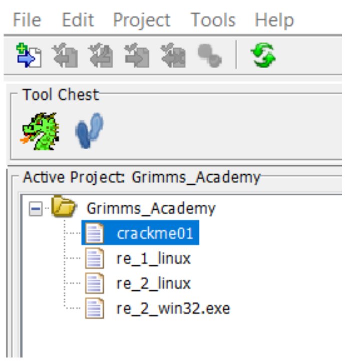


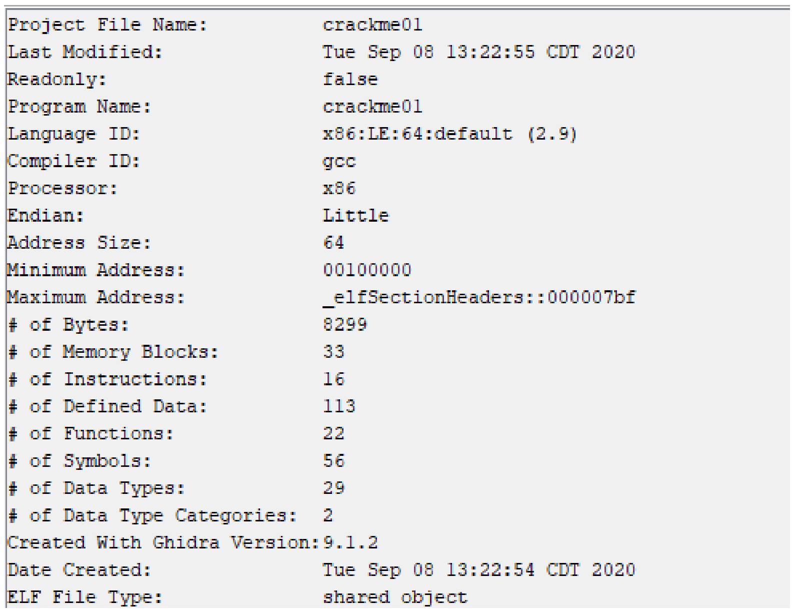
CrackME 01

50

Simple CrackME program. Can you find the password?

Import into Ghidra:

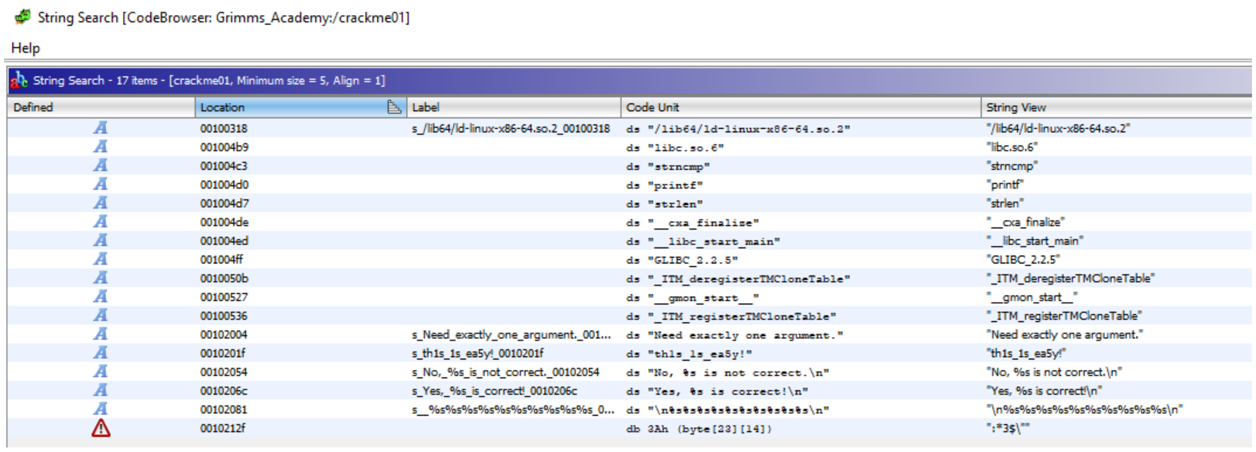




```
Project File Name:            crackme01
Last Modified:                Tue Sep 08 13:22:55 CDT 2020
Readonly:                     false
Program Name:                 crackme01
Language ID:                  x86:LE:64:default (2.9)
Compiler ID:                  gcc
Processor:                    x86
Endian:                       Little
Address Size:                 64
Minimum Address:              00100000
Maximum Address:              _elfSectionHeaders::000007bf
# of Bytes:                   8299
# of Memory Blocks:           33
# of Instructions:            16
# of Defined Data:            113
# of Functions:               22
# of Symbols:                 56
# of Data Types:              29
# of Data Type Categories:    2
Created With Ghidra Version:9.1.2
Date Created:                 Tue Sep 08 13:22:54 CDT 2020
ELF File Type:                shared object
```

Open it up and analyze the program.

Look at the strings to see if there are any clues.



String Search [CodeBrowser: Grimms_Academy:/crackme01]

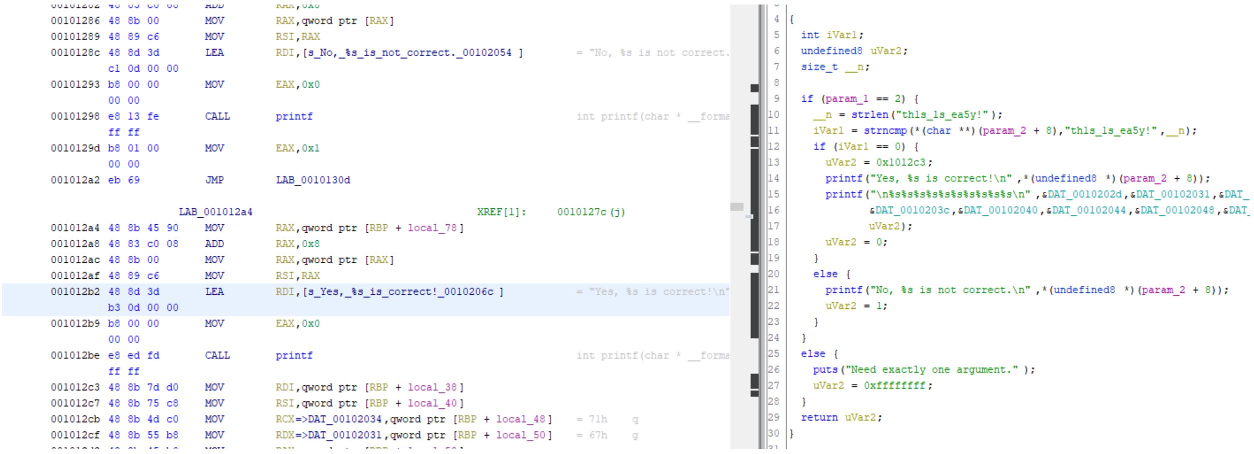
Help

String Search - 17 items - [crackme01, Minimum size = 5, Align = 1]

Defined	Location	Label	Code Unit	String View
A	00100318	s_/lib64/ld-linux-x86-64.so.2_00100318	ds "/lib64/ld-linux-x86-64.so.2"	"/lib64/ld-linux-x86-64.so.2"
A	001004b9		ds "libc.so.6"	"libc.so.6"
A	001004c3		ds "strncmp"	"strncmp"
A	001004d0		ds "printf"	"printf"
A	001004d7		ds "strlen"	"strlen"
A	001004de		ds "__cxa_finalize"	"__cxa_finalize"
A	001004ed		ds "__libc_start_main"	"__libc_start_main"
A	001004ff		ds "GLIBC_2.2.5"	"GLIBC_2.2.5"
A	0010050b		ds "_ITM_deregisterTMCloneTable"	"_ITM_deregisterTMCloneTable"
A	00100527		ds "__gmon_start__"	"__gmon_start__"
A	00100536		ds "_ITM_registerTMCloneTable"	"_ITM_registerTMCloneTable"
A	00102004	s_Need_exactly_one_argument._001...	ds "Need exactly one argument."	"Need exactly one argument."
A	0010201f	s_th1s_1s_ea5y!_0010201f	ds "th1s_1s_ea5y!"	"th1s_1s_ea5y!"
A	00102054	s_No,_%s_is_not_correct._00102054	ds "No, %s is not correct.\n"	"No, %s is not correct.\n"
A	0010206c	s_Yes,_%s_is_correct!_0010206c	ds "Yes, %s is correct!\n"	"Yes, %s is correct!\n"
A	00102081	s__%s%s%s%s%s%s%s%s%s%s_0...	ds "\n%s%s%s%s%s%s%s%s%s%s\n"	"\n%s%s%s%s%s%s%s%s%s%s\n"
⚠	0010212f		db 3Ah (byte[23][14])	":*3$\""

Tried the th1s_1s_ea5y! and that was not the correct answer.

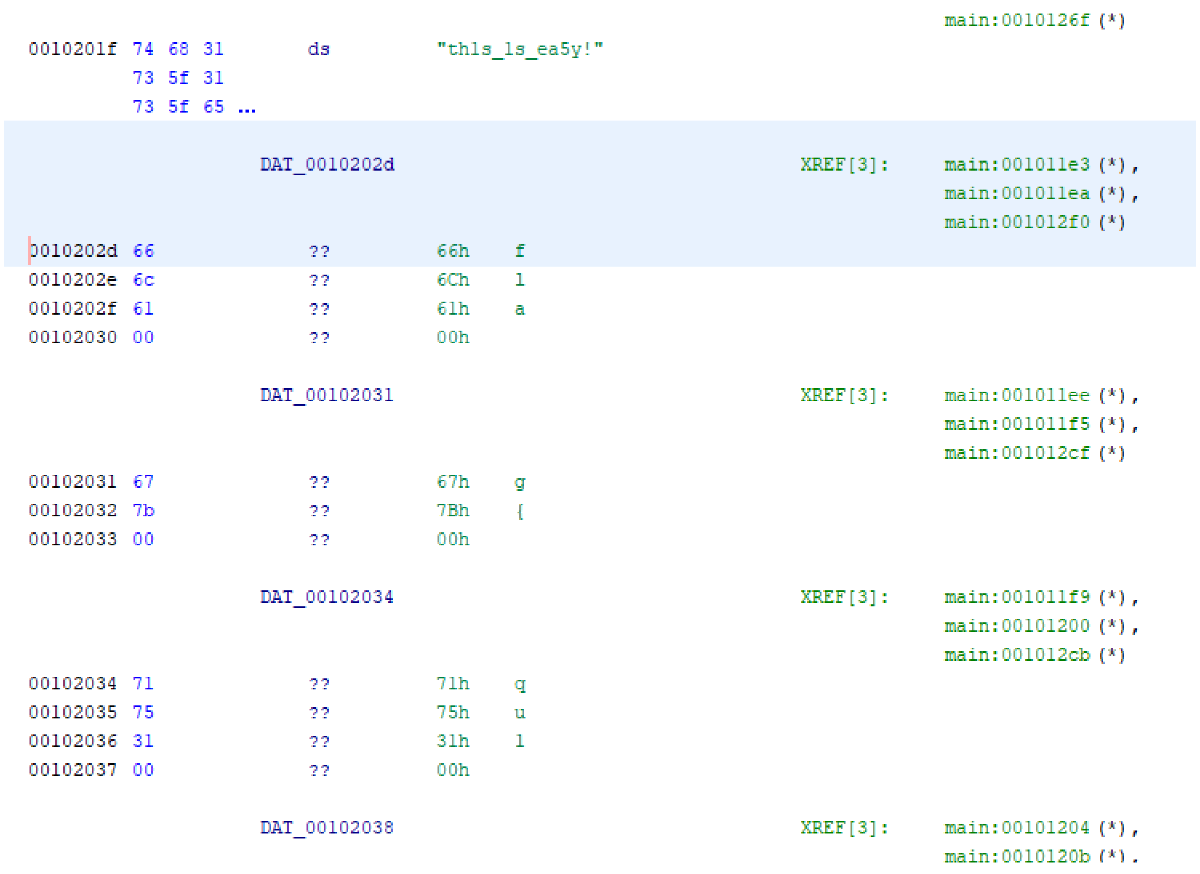
Looking at the code a little closer we see that undefined8 main() may give us a better look a the answer.



```
00101286 48 8b 00       MOV     RAX,qword ptr [RAX]
00101289 48 89 c6       MOV     RSI,RAX
0010128c 48 8d 3d       LEA     RDI,[s_No,_%s_is_not_correct._00102054 ]     = "No, %s is not correct.
         c1 0d 00 00
00101293 b8 00 00       MOV     EAX,0x0
         00 00
00101298 e8 13 fe       CALL    printf                                       int printf(char * __forma
         ff ff
0010129d b8 01 00       MOV     EAX,0x1
         00 00
001012a2 eb 69          JMP     LAB_0010130d

                 LAB_001012a4                          XREF[1]:     0010127c(j)
001012a4 48 8b 45 90    MOV     RAX,qword ptr [RBP + local_78]
001012a8 48 83 c0 08    ADD     RAX,0x8
001012ac 48 8b 00       MOV     RAX,qword ptr [RAX]
001012af 48 89 c6       MOV     RSI,RAX
001012b2 48 8d 3d       LEA     RDI,[s_Yes,_%s_is_correct!_0010206c ]        = "Yes, %s is correct!\n"
         b3 0d 00 00
001012b9 b8 00 00       MOV     EAX,0x0
         00 00
001012be e8 ed fd       CALL    printf                                       int printf(char * __forma
         ff ff
001012c3 48 8b 7d d0    MOV     RDI,qword ptr [RBP + local_38]
001012c7 48 8b 75 c8    MOV     RSI,qword ptr [RBP + local_40]
001012cb 48 8b 4d c0    MOV     RCX=>DAT_00102034,qword ptr [RBP + local_48]   = 71h    q
001012cf 48 8b 55 b8    MOV     RDX=>DAT_00102031,qword ptr [RBP + local_50]   = 67h    g
```

```
{
  int iVar1;
  undefined8 uVar2;
  size_t __n;

  if (param_1 == 2) {
    __n = strlen("th1s_1s_ea5y!");
    iVar1 = strncmp(*(char **)(param_2 + 8),"th1s_1s_ea5y!",__n);
    if (iVar1 == 0) {
      uVar2 = 0x1012c3;
      printf("Yes, %s is correct!\n",*(undefined8 *)(param_2 + 8));
      printf("\n%s%s%s%s%s%s%s%s%s%s\n",&DAT_0010202d,&DAT_00102031,&DAT_
             &DAT_0010203c,&DAT_00102040,&DAT_00102044,&DAT_00102048,&DAT_
             uVar2);
      uVar2 = 0;
    }
    else {
      printf("No, %s is not correct.\n",*(undefined8 *)(param_2 + 8));
      uVar2 = 1;
    }
  }
  else {
    puts("Need exactly one argument.");
    uVar2 = 0xffffffff;
  }
  return uVar2;
}
```

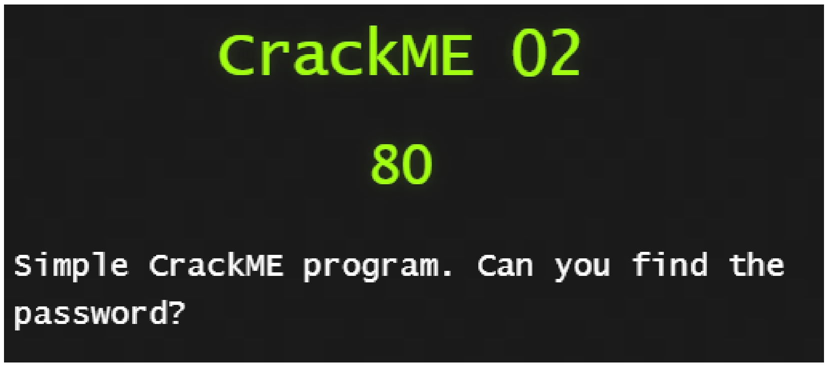
The DAT in the printf statement take use to a bunch of strings broken up:



```
                                                                      main:0010126f (*)
0010201f 74 68 31        ds         "th1s_1s_ea5y!"
         73 5f 31
         73 5f 65 ...
                    DAT_0010202d                         XREF[3]:     main:001011e3 (*),
                                                                      main:001011ea (*),
                                                                      main:001012f0 (*)
0010202d 66              ??         66h    f
0010202e 6c              ??         6Ch    l
0010202f 61              ??         61h    a
00102030 00              ??         00h
                    DAT_00102031                         XREF[3]:     main:001011ee (*),
                                                                      main:001011f5 (*),
                                                                      main:001012cf (*)
00102031 67              ??         67h    g
00102032 7b              ??         7Bh    {
00102033 00              ??         00h
                    DAT_00102034                         XREF[3]:     main:001011f9 (*),
                                                                      main:00101200 (*),
                                                                      main:001012cb (*)
00102034 71              ??         71h    q
00102035 75              ??         75h    u
00102036 31              ??         31h    1
00102037 00              ??         00h
                    DAT_00102038                         XREF[3]:     main:00101204 (*),
                                                                      main:0010120b (*),
```

So if you put them all together you will get: flag{qu1ck_k1ll_w1th_str1ngs}

Flag: flag{qu1ck_k1ll_w1th_str1ngs}

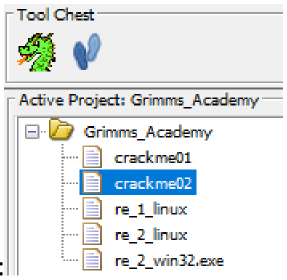


CrackME 02

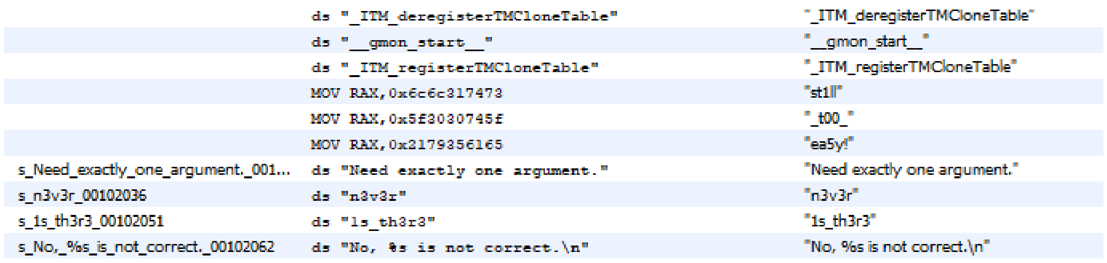
80

Simple CrackME program. Can you find the password?

Import and analyze the program in Ghidra:

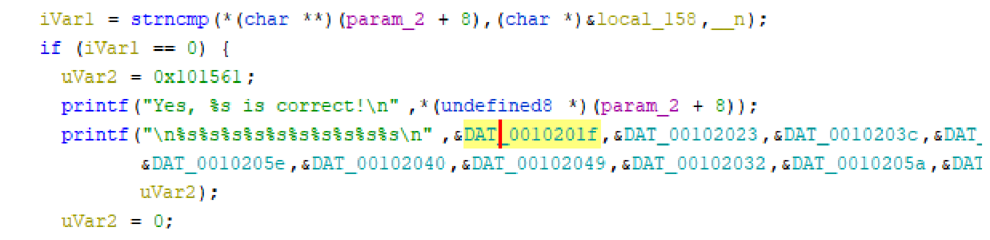


Looking at strings we see the following:



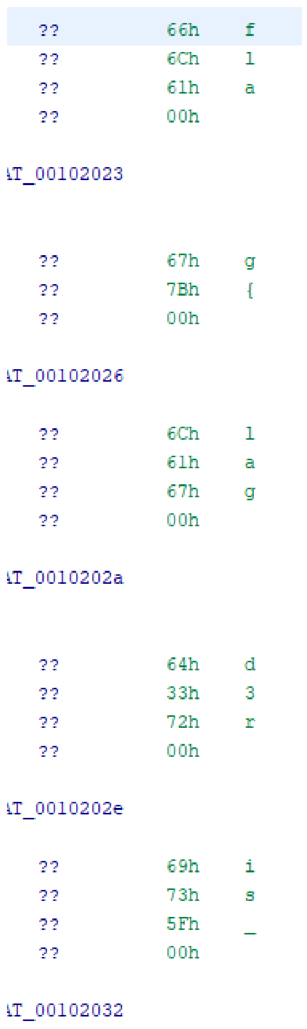
	ds "_ITM_deregisterTMCloneTable"	"_ITM_deregisterTMCloneTable"
	ds "__gmon_start__"	"__gmon_start__"
	ds "_ITM_registerTMCloneTable"	"_ITM_registerTMCloneTable"
	MOV RAX,0x6c6c317473	"st1ll"
	MOV RAX,0x5f3030745f	"_t00_"
	MOV RAX,0x2179356165	"ea5y!"
s_Need_exactly_one_argument._001...	ds "Need exactly one argument."	"Need exactly one argument."
s_n3v3r_00102036	ds "n3v3r"	"n3v3r"
s_1s_th3r3_00102051	ds "1s_th3r3"	"1s_th3r3"
s_No,_%s_is_not_correct._00102062	ds "No, %s is not correct.\n"	"No, %s is not correct.\n"

St1ll_t00_ea5y! looks promising and that takes us to the main function that we see a bunch of DAT fields on the correct printf.



```
  iVar1 = strncmp(*(char **)(param_2 + 8),(char *)&local_158,__n);
  if (iVar1 == 0) {
    uVar2 = 0x101561;
    printf("Yes, %s is correct!\n",*(undefined8 *)(param_2 + 8));
    printf("\n%s%s%s%s%s%s%s%s%s%s%s\n",&DAT_0010201f,&DAT_00102023,&DAT_0010203c,&DAT_
           &DAT_0010205e,&DAT_00102040,&DAT_00102049,&DAT_00102032,&DAT_0010205a,&DAT
           uVar2);
    uVar2 = 0;
```

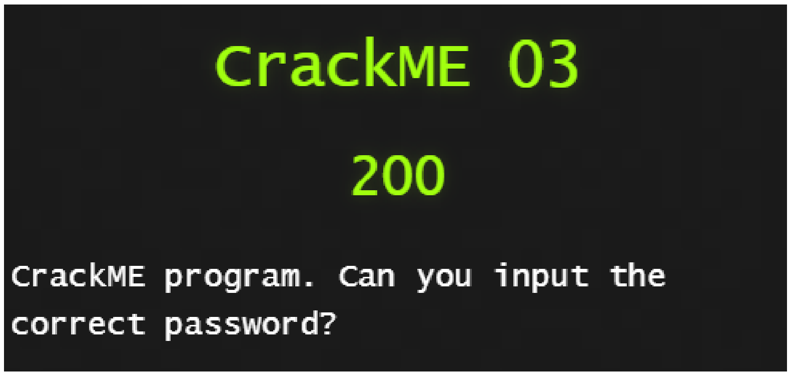
The DATs take us to the following:



```
   ??          66h    f
   ??          6Ch    l
   ??          61h    a
   ??          00h

T_00102023


   ??          67h    g
   ??          7Bh    {
   ??          00h

T_00102026

   ??          6Ch    l
   ??          61h    a
   ??          67h    g
   ??          00h

T_0010202a


   ??          64h    d
   ??          33h    3
   ??          72h    r
   ??          00h

T_0010202e

   ??          69h    i
   ??          73h    s
   ??          5Fh    _
   ??          00h

T_00102032
```

Beware the flag is out of order!

Flag: Flag{k1nd3rgard3n_waz_hard3r}

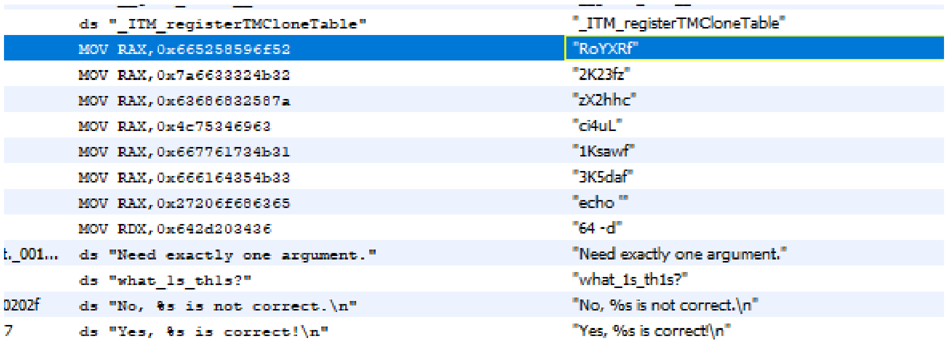


CrackME 03

200

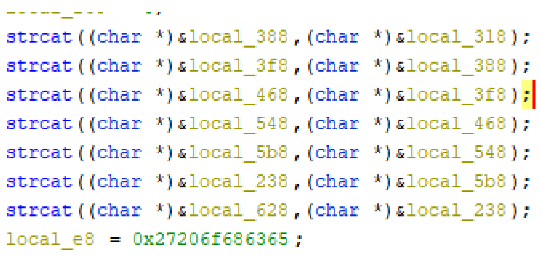
CrackME program. Can you input the correct password?

Import into Ghidra and analyze the program and look at strings. We see Base64 now.



```
            ds "_ITM_registerTMCloneTable"          "_ITM_registerTMCloneTable"
            MOV RAX,0x665258596f52                  "RoYXRf"
            MOV RAX,0x7a6633324b32                  "2K23fz"
            MOV RAX,0x63686832587a                  "zX2hhc"
            MOV RAX,0x4c75346963                    "ci4uL"
            MOV RAX,0x667761734b31                  "1Ksawf"
            MOV RAX,0x666164354b33                  "3K5daf"
            MOV RAX,0x27206f686365                  "echo '"
            MOV RDX,0x642d203436                    "64 -d"
t._001...   ds "Need exactly one argument."         "Need exactly one argument."
            ds "what_1s_th1s?"                      "what_1s_th1s?"
0202f       ds "No, %s is not correct.\n"           "No, %s is not correct.\n"
7           ds "Yes, %s is correct!\n"              "Yes, %s is correct!\n"
```

We can see that there is a string concatenation going on here:



```
  strcat((char *)&local_388,(char *)&local_318);
  strcat((char *)&local_3f8,(char *)&local_388);
  strcat((char *)&local_468,(char *)&local_3f8);
  strcat((char *)&local_548,(char *)&local_468);
  strcat((char *)&local_5b8,(char *)&local_548);
  strcat((char *)&local_238,(char *)&local_5b8);
  strcat((char *)&local_628,(char *)&local_238);
  local_e8 = 0x27206f686365;
```

If we put these together, we get the following b64 code:
ci4uL

mQz

zX2hhc

d2F

RoYXRf

D1M3NzX3

ZmxhZ3tJ

Put into CyberChef and rearrange the base 64 round:
ZmxhZ3tJX2d1M3NzX3RoYXRfd2FzX2hhcmQzci4uLn0K

Flag: flag{I_gu3ss_that_was_hard3r...}