✅ ReversingELF

Author: Hubert 'hubertf' Feyrer, 2026-04-03
Link: https://tryhackme.com/room/reverselfiles

General note:
All reversing and source-code extraction done with ghidra under MacOS,
Only 1x ltrace was used on an x86 machine to run a binary.

crackme1

```
neuland% cat 1crackme1-sol.c
#include <stdio.h>
#include <string.h>

#define uint unsigned int

int main(void)

{
  char local_98 [32];
  uint local_78 [28];

  local_78[0] = 0x25;
  local_78[1] = 0x2b;
  local_78[2] = 0x20;
  local_78[3] = 0x26;
  local_78[4] = 0x3a;
  local_78[5] = 0x2d;
  local_78[6] = 0x2e;
  local_78[7] = 0x33;
  local_78[8] = 0x1e;
  local_78[9] = 0x33;
  local_78[10] = 0x27;
  local_78[0xb] = 0x20;
  local_78[0xc] = 0x33;
  local_78[0xd] = 0x1e;
  local_78[0xe] = 0x2a;
  local_78[0xf] = 0x28;
  local_78[0x10] = 0x2d;
  local_78[0x11] = 0x23;
  local_78[0x12] = 0x1e;
  local_78[0x13] = 0x2e;
  local_78[0x14] = 0x25;
  local_78[0x15] = 0x1e;
  local_78[0x16] = 0x24;
  local_78[0x17] = 0x2b;
```

```
  local_78[0x18] = 0x25;
  local_78[0x19] = 0x3c;
  local_78[0x1a] = 0xffffffbf;
  memset(local_98,0x41,0x1b);
  for (local_78[0x1b] = 0; local_78[0x1b] < 0x1b; local_78[0x1b] =
local_78[0x1b] + 1) {
    local_98[(int)local_78[0x1b]] =
         (char)local_78[(int)local_78[0x1b]] +
local_98[(int)local_78[0x1b]];
  }
  puts(local_98);
  return 0;
}

neuland% gcc 1crackme1-sol.c -o 1crackme1-sol
neuland% ./1crackme1-sol
flag{not_that_kind_of_elf}
```

Summary:
Extracted the relevant code from ghidra and got it compiled to print the flag.

crackme2

```
neuland% cat 2crackme2-sol.c
#include <stdio.h>
#include <string.h>

#define uint unsigned int

char DAT_080486c0[] = { 0x25,  0x2b,  0x20,  0x26,  0x3a,  0x28,  0x25,
0x1e,  0x28,  0x1e,  0x32,  0x34,  0x21,  0x2c,  0x28,  0x33,  0x1e,
0x33,  0x27,  0x28,  0x32,  0x1e,  0x25,  0x2b,  0x20,  0x26,  0x1e,
0x33,  0x27,  0x24,  0x2d,  0x1e,  0x28,  0x1e,  0x36,  0x28,  0x2b,
0x2b,  0x1e,  0x26,  0x24,  0x33,  0x1e,  0x2f,  0x2e,  0x28,  0x2d,
0x33,  0x32,  0x3c,  0xbf, 0xff, 0xff, 0xff, 0x01, 0x1b, 0x03, 0x3b };

int main(void)
{
  int i;
  char *puVar1;
  char *puVar2;
  char local_11f [51];
  char local_ec [51];
  uint j;

  puVar1 = DAT_080486c0;
  puVar2 = local_ec;
  for (i = 0x33; i != 0; i = i + -1) {
```

```
    *puVar2 = *puVar1;
    puVar1 = puVar1 + 1;
    puVar2 = puVar2 + 1;
  }
  memset(local_11f,0x41,0x33);
  for (j = 0; j < 0x33; j = j + 1) {
    local_11f[j] = (char)local_ec[j] + local_11f[j];
  }
  puts(local_11f);
  return 0;
}

neuland% gcc 2crackme2-sol.c -o 2crackme2-sol
neuland% ./2crackme2-sol
flag{if_i_submit_this_flag_then_i_will_get_points}
```

Summary:
Extracted the relevant code from ghidra and got it compiled to print the flag.
The data was copied as "C Array" from Ghidra's data segment.

crackme3

```
ghidra:

      if ((argv1len == 0x40) &&
         (iVar1 =
strcmp(buf,"ZjByX3kwdXJfNWVjMG5kX2xlNTVvbl91bmJhc2U2NF80bGxfN2gzXzdoMW5nNQ
=="),
          iVar1 == 0)) {
        puts("Correct password!");



neuland% echo -n
ZjByX3kwdXJfNWVjMG5kX2xlNTVvbl91bmJhc2U2NF80bGxfN2gzXzdoMW5nNQ== | base64
-d
f0r_y0ur_5ec0nd_le55on_unbase64_4ll_7h3_7h1ng5
```

Summary:
After understanding the only interesting function really does base64 on the input, this is easily decoded.

crackme4

```
neuland% cat 4sol.c
#include <stdio.h>
#include <string.h>
```

```
#define uint unsigned int
#define byte char

char s[] = "I]{I\x14V\x17{WAGQV\x17{TS@";

int main(void)
{
  int i = -1;
  while (i = i + 1, s[i] != '\0') {
     s[i] = s[i] ^ 0x24;
  }

  printf("s: \"%s\"\n", s);

  return 0;
}

neuland% gcc 4sol.c -o 4sol
neuland% ./4sol
s: "my_m0r3_secur3_pwd"
```

Summary:
Extracted the relevant code from ghidra and got it compiled to print the flag using XOR from C.

crackme5

```
ltrace:

$ ltrace ./5crackme5
__libc_start_main(0x400773, 1, 0x7ffe821ab108, 0x4008d0 <unfinished ...>
puts("Enter your input:"Enter your input:
)                       = 18
__isoc99_scanf(0x400966, 0x7ffe821aaf80, 0x7f6537083750, 0x7f6537083750x
) = 1
strlen("x")                                     = 1
strlen("x")                                     = 1
strncmp("x", "OfdlDSA|3tXb32~X3tX@sX`4tXtz", 28) = 41
puts("Always dig deeper"Always dig deeper
)                       = 18
+++ exited (status 0) +++


OfdlDSA|3tXb32~X3tX@sX`4tXtz
```

Summary:
I didn't get this built properly in C, so I ran the binary to see what strncmp() is comparing the

input with. After that, the garbage-like looking output from my C program made sense.

crackme6

```
ghidra:

undefined8 my_secure_test(char *argv1)

{
  undefined8 uVar1;

  if ((*argv1 == '\0') || (*argv1 != '1')) {
    uVar1 = 0xffffffff;
  }
  else if ((argv1[1] == '\0') || (argv1[1] != '3')) {
    uVar1 = 0xffffffff;
  }
  else if ((argv1[2] == '\0') || (argv1[2] != '3')) {
    uVar1 = 0xffffffff;
  }
  else if ((argv1[3] == '\0') || (argv1[3] != '7')) {
    uVar1 = 0xffffffff;
  }
  else if ((argv1[4] == '\0') || (argv1[4] != '_')) {
    uVar1 = 0xffffffff;
  }
  else if ((argv1[5] == '\0') || (argv1[5] != 'p')) {
    uVar1 = 0xffffffff;
  }
  else if ((argv1[6] == '\0') || (argv1[6] != 'w')) {
    uVar1 = 0xffffffff;
  }
  else if ((argv1[7] == '\0') || (argv1[7] != 'd')) {
    uVar1 = 0xffffffff;
  }
  else if (argv1[8] == '\0') {
    uVar1 = 0;
  }
  else {
    uVar1 = 0xffffffff;
  }
  return uVar1;
}


goal: return 0

=> 1337_pwd
```

Summary:
The password's single characters are obvious from looking at the Ghidra source.

crackme7

```
neuland% cat 7sol.c
#include <stdio.h>
#include <string.h>

#define uint unsigned int

int DAT_080488e0[] = { 0x25,  0x2b,  0x20,  0x26,  0x3a,  0x2c,  0x34,
0x22,  0x27,  0x1e,  0x31,  0x24,  0x35,  0x24,  0x31,  0x32,  0x28,
0x2d,  0x26,  0x1e,  0x35,  0x24,  0x31,  0x38,  0x1e,  0x28,  0x23,
0x20,  0x1e,  0x36,  0x2e,  0x36,  0x3c,  0xbf, 0xff, 0xff, 0xff };

int main(void)
{
  int i;
  int *puVar2;
  int *piVar1;
  char local_ca [34];
  int local_a8 [34];
  uint j;

  puVar2 = DAT_080488e0;
  piVar1 = local_a8;
  for (i = 34; i != 0; i = i + -1) {
    *piVar1 = *puVar2;
    puVar2 = puVar2 + 1;
    piVar1 = piVar1 + 1;
  }
  memset(local_ca,0x41,0x22);
  for (j = 0; j < 34; j = j + 1) {
    local_ca[j] = (char)local_a8[j] + local_ca[j];
  }
  puts(local_ca);
  return 0;
}

neuland% gcc 7sol.c -o 7sol
neuland% ./7sol
flag{much_reversing_very_ida_wow}
```

Summary:
Again, extracted the relevant code from ghidra and got it compiled to print the flag.

crackme8

```
neuland% cat 8sol.c
#include <stdio.h>
#include <string.h>

#define uint unsigned int

int DAT_080486a0[] =
{ 0x25,  0x2b,  0x20,  0x26,  0x3a,  0x20,  0x33,  0x1e,  0x2b,  0x24,
0x20,  0x32,  0x33,  0x1e,  0x33,  0x27,  0x28,  0x32,  0x1e,  0x22,
0x20,  0x25,  0x24,  0x1e,  0x36,  0x2e,  0x2d,  0x33,  0x1e,  0x2b,
0x24,  0x20,  0x2a,  0x1e,  0x38,  0x2e,  0x34,  0x31,  0x1e,  0x22,
0x31,  0x24,  0x23,  0x28,  0x33,  0x1e,  0x22,  0x20,  0x31,  0x23,
0x1e,  0x2d,  0x34,  0x2c,  0x21,  0x24,  0x31,  0x32,  0x3c,  0xbf, 0xff,
0xff, 0xff };

int main(void)
{
  int i;
  int *puVar2;
  int *piVar1;
  char local_14c [60];
  int local_110 [60];
  uint j;

  puVar2 = DAT_080486a0;
  piVar1 = local_110;
  for (i = 60; i != 0; i = i + -1) {
    *piVar1 = *puVar2;
    puVar2 = puVar2 + 1;
    piVar1 = piVar1 + 1;
  }
  memset(local_14c,0x41,0x3c);
  for (j = 0; j < 60; j = j + 1) {
    local_14c[j] = (char)local_110[j] + local_14c[j];
  }
  puts(local_14c);
  return 0;
}

neuland% !gc
gcc 8sol.c -o 8sol
neuland% ./8sol
flag{at_least_this_cafe_wont_leak_your_credit_card_numbers}
```

Summary:
A last time extracted the relevant code from ghidra and got it compiled to print the flag.