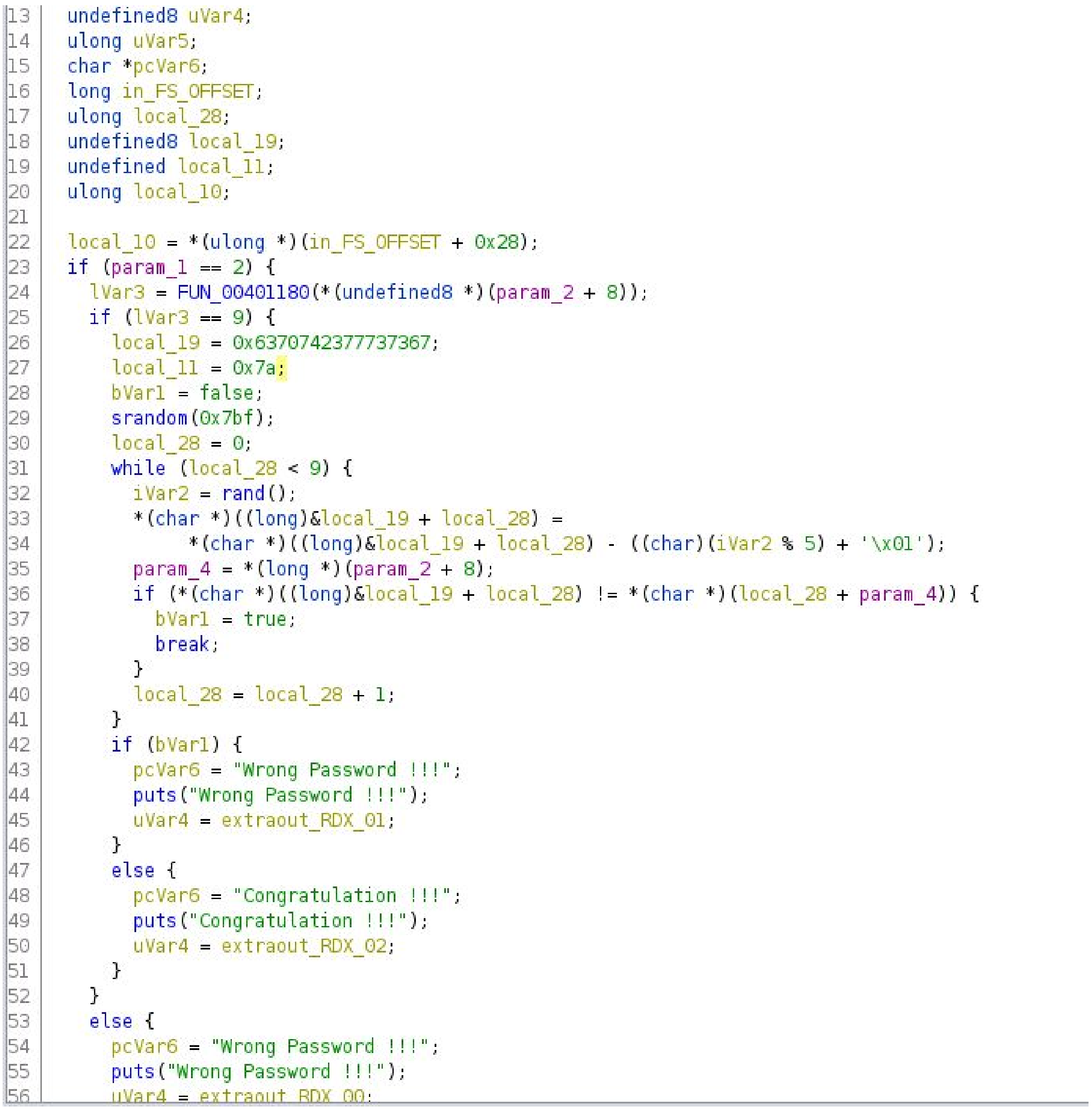
Crackmes.one Wargames

Link: Crackmes
Description: Use ./WarGames pass
Language: C/C++
Level:2

Ghidra Analysis Main



```
  undefined8 uVar4;
  ulong uVar5;
  char *pcVar6;
  long in_FS_OFFSET;
  ulong local_28;
  undefined8 local_19;
  undefined local_11;
  ulong local_10;

  local_10 = *(ulong *)(in_FS_OFFSET + 0x28);
  if (param_1 == 2) {
    lVar3 = FUN_00401180(*(undefined8 *)(param_2 + 8));
    if (lVar3 == 9) {
      local_19 = 0x6370742377737367;
      local_11 = 0x7a;
      bVar1 = false;
      srandom(0x7bf);
      local_28 = 0;
      while (local_28 < 9) {
        iVar2 = rand();
        *(char *)((long)&local_19 + local_28) =
             *(char *)((long)&local_19 + local_28) - ((char)(iVar2 % 5) + '\x01');
        param_4 = *(long *)(param_2 + 8);
        if (*(char *)((long)&local_19 + local_28) != *(char *)(local_28 + param_4)) {
          bVar1 = true;
          break;
        }
        local_28 = local_28 + 1;
      }
      if (bVar1) {
        pcVar6 = "Wrong Password !!!";
        puts("Wrong Password !!!");
        uVar4 = extraout_RDX_01;
      }
      else {
        pcVar6 = "Congratulation !!!";
        puts("Congratulation !!!");
        uVar4 = extraout_RDX_02;
      }
    }
    else {
      pcVar6 = "Wrong Password !!!";
      puts("Wrong Password !!!");
      uVar4 = extraout_RDX_00;
```

Important Lines

Line 23: Check argc count

Line 22: Checks The Len of the password and sees if its equal to 9

Line 24: Calls FUN_004011890

Line 26: Random String maybe ?

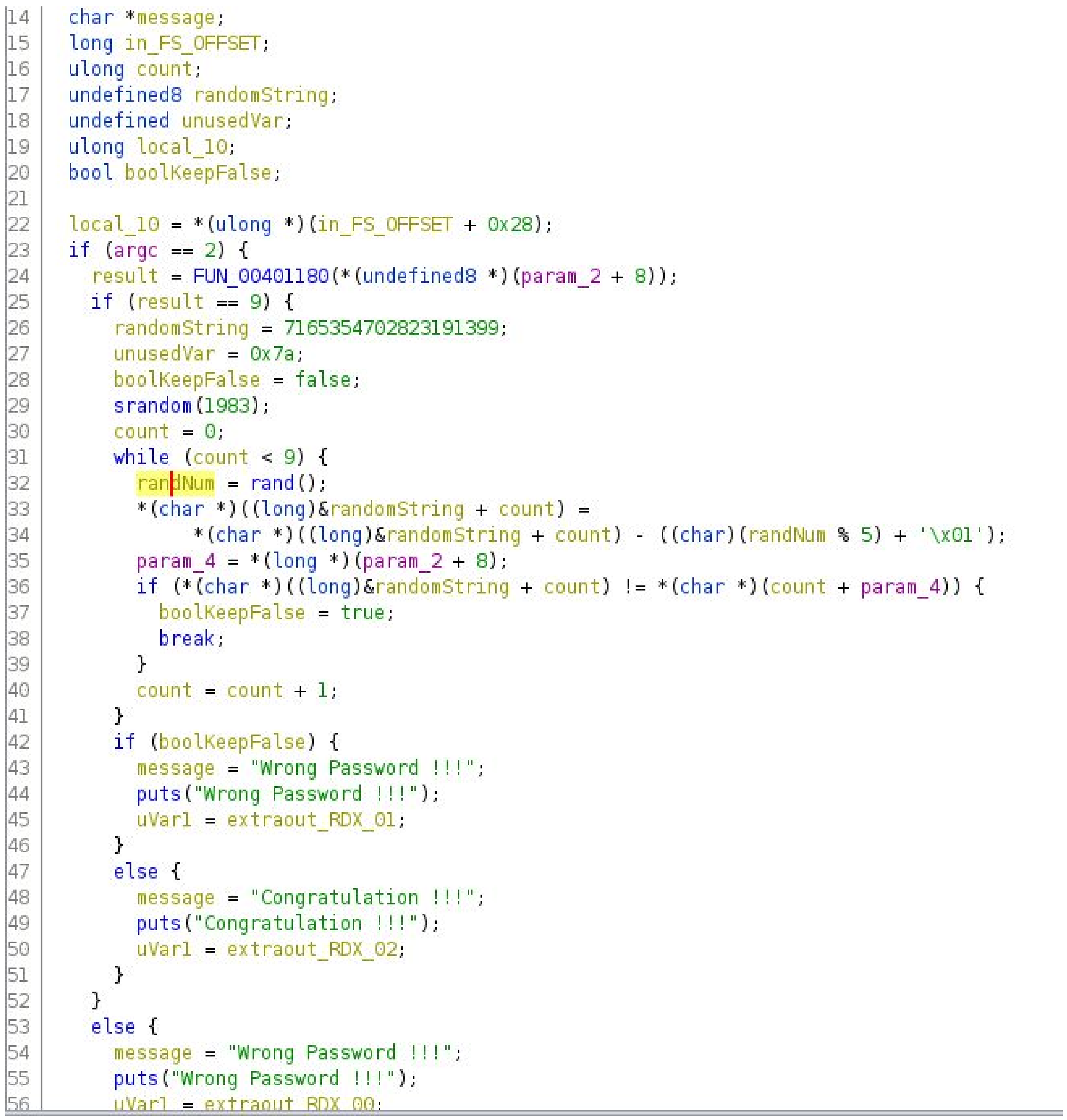
Line: 29: void srandom(uint __seed) (set random seed) to 0x7bf == 1983

Line 30,31: local_28 looks like a counter variable

Line 32: calls rand() and stores result in iVar2

Line 33-35 Looks like some type of encryption

Ghidra Analysis With renamed Variables



```
14    char *message;
15    long in_FS_OFFSET;
16    ulong count;
17    undefined8 randomString;
18    undefined unusedVar;
19    ulong local_10;
20    bool boolKeepFalse;
21    
22    local_10 = *(ulong *)(in_FS_OFFSET + 0x28);
23    if (argc == 2) {
24      result = FUN_00401180(*(undefined8 *)(param_2 + 8));
25      if (result == 9) {
26        randomString = 7165354702823191399;
27        unusedVar = 0x7a;
28        boolKeepFalse = false;
29        srandom(1983);
30        count = 0;
31        while (count < 9) {
32          randNum = rand();
33          *(char *)((long)&randomString + count) =
34               *(char *)((long)&randomString + count) - ((char)(randNum % 5) + '\x01');
35          param_4 = *(long *)(param_2 + 8);
36          if (*(char *)((long)&randomString + count) != *(char *)(count + param_4)) {
37            boolKeepFalse = true;
38            break;
39          }
40          count = count + 1;
41        }
42        if (boolKeepFalse) {
43          message = "Wrong Password !!!";
44          puts("Wrong Password !!!");
45          uVar1 = extraout_RDX_01;
46        }
47        else {
48          message = "Congratulation !!!";
49          puts("Congratulation !!!");
50          uVar1 = extraout_RDX_02;
51        }
52      }
53      else {
54        message = "Wrong Password !!!";
55        puts("Wrong Password !!!");
56        uVar1 = extraout_RDX_00;
```

Radare2 Check Random Values

```
r2 -d WarGames aaaaaaaaa
```

Rand function:

```
;-- rand:
;-- panel.addr:
0x00410d80     f30f1efa      endbr64
0x00410d84     4883ec08       sub rsp, 8
0x00410d88     e8e3faffff    call sym.random
0x00410d8d     4883c408       add rsp, 8
0x00410d91      c3             ret
```

```
db 0x00410d91
```

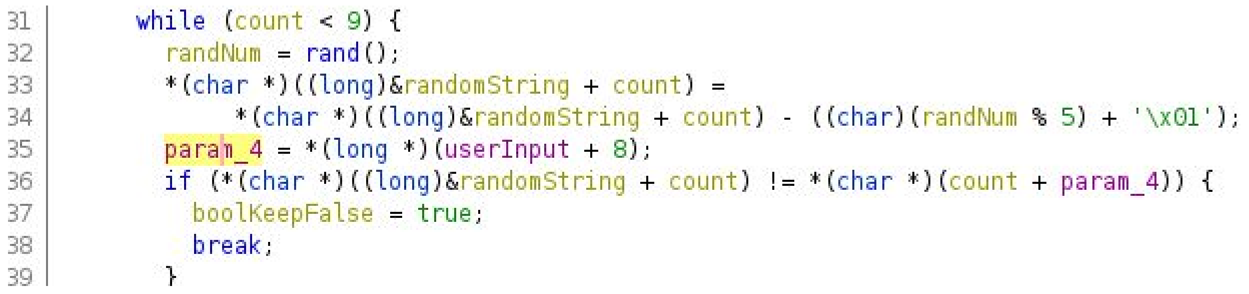
Setting a breakpoint on the return address will allow us to see what value is getting returned.

```
[0x00401c10]> dc
hit breakpoint at: 0x410d91
[0x00410d91]> drr
R0   rax     13329ea0        322084512 rax,rdx

[0x00410d91]> dc
Wrong Password !!!
(36635) Process exited with status=0x0
```

I ran it again and got the same input: 322084512

Lines 33-36



```
31      while (count < 9) {
32        randNum = rand();
33        *(char *)((long)&randomString + count) =
34             *(char *)((long)&randomString + count) - ((char)(randNum % 5) + '\x01');
35        param_4 = *(long *)(userInput + 8);
36        if (*(char *)((long)&randomString + count) != *(char *)(count + param_4)) {
37          boolKeepFalse = true;
38          break;
39        }
```

Line: 36 checks to see if the correct character matches the inputted password.

What Are We Looking For

This program seems to look for a specific password rather than multiple that fit a certain criteria (like a keygen). If we set a breakpoint at line 35 we can see what the correct character is.

Radare2 Finding The Correct Characters

I'm going to set a breakpoint on line 35

```
0x00401e3e     488b45c0      mov rax, qword [rbp - 0x40]
0x00401e42     4883c008      add rax, 8
 ;-- rip:
0x00401e46     488b08        mov rcx, qword [rax]
0x00401e49     488b45e0      mov rax, qword [rbp - 0x20]
```

```
db 0x0401e46
[0x00410d91]> dc
hit breakpoint at: 0x401e46
[0x00401e46]> drr
role reg     value           refstr
―――――――――――――――――――――――――――――――――――――――――――――――――――――――
R0   rax     7ffebe238500     [stack] rax stack R W 0x7ffebe23a40f
     rbx     400518           4195608 /home/drew/Challenges/crackmes.one/WarGames
__rela_iplt_end,rbx program R 0x0
A3   rcx     64               100 .note.stapsdt rcx,rdx ascii ('d')
A2   rdx     64               100 .note.stapsdt rcx,rdx ascii ('d')
```

If we do this nine more times we can figure out the values.

1 →d
2 →o
3 →n
4 → t
5 →(space)
6 →p
7 →l
8 →a
9 →y

```
drew@ubuntu:~/Challenges/crackmes.one$ ./WarGames "dont play"
Congratulations !!!
```