Reverse Engineering

By 0xECE

LitCrypt Hack

Challenge | 30 Solves

LITCRYPT HACK

366

blackjack

Les agents du Syndicat ont trouvé une superbe dépendance pour cacher leurs mots de passe, seulement, comme souvent en crypto, la sécurité dépend de la clé utilisé. Nous vous laissons le soin de retrouver le mot de passe.

Syndicat agents have found a superb way to hiding their passwords, but as is often the case in crypto, security depends on the key used... We let you find the password.

md5: 7966294aa963604ca3503de8e3ea46dc

admin_tool

Flag

Submit

```
┌──(kalilinux㉿kalilinux2022)-[~/hackday2024/rev/Crypt]
└─$ file admin_tool
admin_tool: ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV)
, dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, Buil
dID[sha1]=e14e8c111494915a98dd12cef64c813edbca9300, for GNU/Linux 3
.2.0, with debug_info, not stripped
```

On nous donne un binaire linux qui semble tout a fait cool (détrompez vous)

Quand on l'exécute on obtient ça :

```
┌──(kalilinux㉿kalilinux2022)-[~/hackday2024/rev/Crypt]
└─$ ./admin_tool
Please enter password : deeee
Wrong password
```

Une simple (nop) vérification de password.

Utilisons Ghidra pour voir ce qu'il y a dedans :

Decompile: main - (admin_tool)

```
1
2 void main(int param_1,undefined8 param_2)
3
4 {
5   code *local_8;
6
7   local_8 = admin_tool::main;
8   std::rt::lang_start_internal
9             (&local_8,anon.3d5066c3b76da481d363861203e447fc.0.llvm.11389738463635432603,
10             (long)param_1,param_2,0);
11  return;
12 }
13
```

Bon bah c'est du RUST.... aïe aïe aïe ça pique

Bon déjà le "main" n'est pas le vrai main le vrai main est :

Decompile: main - (admin_tool)

```
1
2 void main(int param_1,undefined8 param_2)
3
4 {
5   code *local_8;
6
7   local_8 = admin_tool::main;
8   std::rt::lang_start_internal
9             (&local_8,anon.3d5066c3b76da481d363861203e447fc.0.llvm.11389738463635432603,
10             (long)param_1,param_2,0);
11  return;
12 }
13
```

ça commence bien ...

Decompile: main - (admin_tool)

```
4   void admin_tool::main(void)
5
6   {
7     byte bVar1;
8     byte bVar2;
9     void *__ptr;
10    code **ppcVar3;
11    undefined **__ptr_00;
12    int iVar4;
13    long lVar5;
14    ulong uVar6;
15    byte bVar7;
16    uint uVar8;
17    uint uVar9;
18    uint uVar10;
19    uint uVar11;
20    bool bVar12;
21    void *local_80;
22    long local_78;
23    undefined *puStack_70;
24    undefined **local_68;
25    long local_60;
26    undefined *local_58;
27    undefined local_50 [16];
28    long local_38;
29    undefined8 local_30;
30
31    local_80 = (void *)0x1;
32    local_78 = 0;
33    puStack_70 = (undefined *)0x0;
34    local_68 = (undefined **)&DAT_0015cf40;
35    local_60 = 1;
36    local_58 = &DAT_0014b050;
37    local_50 = ZEXT816(0);
38                        /* try { // try from 00109173 to 00109193 has its CatchHandler @ 001095d7 */
39    std::io::stdio::_print(&local_68);
40    local_68 = (undefined **)std::io::stdio::stdout();
41    lVar5 = <std::io::stdio::Stdout_as_std::io::Write>::flush(&local_68);
42    if (((lVar5 != 0) && (uVar6 = (ulong)((uint)lVar5 & 3), 1 < uVar6 - 2)) && (uVar6 != 0)) {
43      __ptr = *(void **)(lVar5 + -1);
44      ppcVar3 = *(code ***)(lVar5 + 7);
```

```
      }
LAB_0010937d:
      bVar2 = *(byte *)((long)local_80 + (long)(puStack_70 + -2));
      uVar11 = (uint)bVar2;
      uVar8 = (uint)bVar2;
      if (-0x41 < (char)bVar2) goto LAB_0010934b;
LAB_00109388:
      bVar2 = *(byte *)((long)local_80 + (long)(puStack_70 + -3));
      uVar9 = bVar2 & 0xf;
      uVar11 = uVar9;
      if ((char)bVar2 < -0x40) {
        uVar11 = bVar2 & 0x3f | (*(byte *)((long)local_80 + (long)(puStack_70 + -4)) & 7) << 6;
      }
      if (((uVar8 & 0x3f) << 6 | uVar11 << 0xc | (uint)(bVar7 & 0x3f)) != 0xd) goto LAB_00109406;
      if ((char)bVar2 < -0x40) {
        uVar9 = bVar2 & 0x3f | (*(byte *)((long)local_80 + (long)(puStack_70 + -4)) & 7) << 6;
      }
      uVar8 = (uVar8 & 0x3f | uVar9 << 6) << 6 | (uint)(bVar7 & 0x3f);
    }
    else {
      uVar11 = bVar2 & 0x1f;
      if ((uVar11 << 6 | uVar8 & 0x3f) == 10) goto LAB_0010932c;
```

```
LAB_0010933d:
      uVar11 = (uint)*(byte *)((long)local_80 + (long)(puStack_70 + -2));
      uVar8 = uVar11;
      if ((char)*(byte *)((long)local_80 + (long)(puStack_70 + -2)) < -0x40) goto LAB_00109388;
LAB_0010934b:
      if (((uVar11 & 0x1f) << 6 | (uint)(bVar7 & 0x3f)) != 0xd) goto LAB_00109406;
      uVar8 = (uVar11 & 0x1f) << 6 | (uint)(bVar7 & 0x3f);
    }
    if (uVar8 == 0x110000) goto LAB_00109406;
    uVar6 = 0xffffffffffffffff;
    if ((0x7f < uVar8) && (uVar6 = 0xfffffffffffffffe, 0x7ff < uVar8)) {
      uVar6 = (ulong)(uVar8 < 0x10000) | 0xfffffffffffffffc;
    }
  }
  else {
    if (bVar7 == 10) {
      uVar6 = 0xffffffffffffffff;
LAB_00109245:
```

voilà le vrai main...

Bon, à travers tout ce bazard on peut voir une fonction intéressante :

```
  puStack_70 = puStack_70 + uVar6;
LAB_00109406:
  litcrypt_internal::decrypt_bytes(&local_68,&DAT_0014b079,0x27,&DAT_0014b0c1,7);
  __ptr_00 = local_68;
  if (puStack_70 == local_58) {
    iVar4 = bcmp(local_80,local_68,(size_t)puStack_70);
    bVar12 = iVar4 == 0;
  }
  else {
    bVar12 = false;
  }
  if (local_60 != 0) {
    __rust_dealloc(__ptr_00);
  }
  if (bVar12) {
    local_68 = &PTR_DAT_0015cf78;
    local_60 = 1;
    local_58 = &DAT_0014b050;
    local_50 = ZEXT816(0);
```

Un call vers une fonction de chiffrement. Par rapport à l'énoncé on peut se dire qu'on est sur la bonne voie.

```
byte ** admin_tool::litcrypt_internal::decrypt_bytes
                  (byte **param_1,byte *param_2,byte *param_3,byte *param_4,ulong
param_5)

{
  uint *puVar1;
  uint *puVar2;
  byte bVar3;
  byte bVar4;
```

```
  uint uVar5;
  uint uVar6;
  uint uVar7;
  uint uVar8;
  uint uVar9;
  uint uVar10;
  uint uVar11;
  byte *__dest;
  byte *pbVar12;
  byte *pbVar13;
  ulong uVar14;
  ulong uVar15;
  byte *pbVar16;
  bool bVar17;
  uint uVar18;
  undefined auVar19 [16];
  byte *local_70;
  byte *local_68;
  undefined8 local_60;
  byte *local_58;
  char local_50;
  uint local_4f;
  undefined3 uStack_4b;
  uint local_48;
  undefined4 uStack_44;
  byte *local_40;
  byte *local_38;

  pbVar16 = param_3;
  local_40 = param_4;
  if (param_5 == 0) {
    if (param_3 == (byte *)0x0) {
      __dest = (byte *)0x1;
    }
    else {
      bVar17 = (long)param_3 >= 0;
      if ((long)param_3 < 0) {
LAB_00109000:
        alloc::raw_vec::capacity_overflow();
        do {
          invalidInstructionException();
        } while( true );
      }
      __dest = (byte *)__rust_alloc(param_3,bVar17);
      if (__dest == (byte *)0x0) {
LAB_00108d77:
```

```
        alloc::alloc::handle_alloc_error(bVar17,param_3);
        do {
          invalidInstructionException();
        } while( true );
      }
    }
    memcpy(__dest,param_2,(size_t)param_3);
  }
  else {
    if (param_5 == 1) {
      if (param_3 != (byte *)0x0) {
        bVar17 = (long)param_3 >= 0;
        if ((long)param_3 < 0) goto LAB_00109000;
        bVar3 = *param_4;
        __dest = (byte *)__rust_alloc(param_3,bVar17);
        if (__dest == (byte *)0x0) goto LAB_00108d77;
        if (param_3 < (byte *)0x8) {
          pbVar13 = (byte *)0x0;
        }
        else {
          if (param_3 < &DAT_00000020) {
            pbVar13 = (byte *)0x0;
LAB_00108f05:
            auVar19 =
CONCAT142(SUB1614((ZEXT1316(SUB1613((ZEXT1216(SUB1612((ZEXT1116(ZEXT311(0) &

SUB1611((
                                                    undefined
[16])0xffff00ffffffffff >> 0x28,0)) <<
                                                    0x28) >> 0x20,0) &
                                                    SUB1612((undefined
[16])0xffffff00ffffffff >>
                                                            0x20,0)) << 0x20) >>
0x18,0) &
                                                    SUB1613((undefined
[16])0xffffffff00ffffff >>
                                                            0x18,0)) << 0x18) >>
0x10,0) &
                                SUB1614((undefined  [16])0xffffffffff00ffff >>
0x10,0),
                                CONCAT11(bVar3,bVar3));
            auVar19 = pshuflw(auVar19,auVar19,0);
            do {
              *(ulong *)(__dest + (long)pbVar13) =
                   *(ulong *)(param_2 + (long)pbVar13) ^ SUB168(auVar19,0);
              pbVar13 = pbVar13 + 8;
```

```
              pbVar12 = (byte *)((ulong)param_3 & 0xfffffffffffffff8);
              if ((byte *)((ulong)param_3 & 0xfffffffffffffff8) == pbVar13) goto
joined_r0x00108f3c;
            } while( true );
          }
          pbVar13 = (byte *)((ulong)param_3 & 0xffffffffffffffe0);
          auVar19 =
CONCAT142(SUB1614((ZEXT1316(SUB1613((ZEXT1216(SUB1612((ZEXT1116(ZEXT311(0) &

SUB1611((
                                                    undefined
[16])0xffff00ffffffffff >> 0x28,0)) <<
                                                    0x28) >> 0x20,0) &
                                                    SUB1612((undefined
[16])0xffffff00ffffffff >>
                                                            0x20,0)) << 0x20) >>
0x18,0) &
                                                  SUB1613((undefined
[16])0xffffffff00ffffff >> 0x18,
                                                          0)) << 0x18) >> 0x10,0) &
                                SUB1614((undefined  [16])0xffffffffff00ffff >>
0x10,0),
                                CONCAT11(bVar3,bVar3));
          auVar19 = pshuflw(auVar19,auVar19,0);
          uVar18 = SUB164(auVar19,0);
          pbVar12 = (byte *)0x0;
          do {
            puVar1 = (uint *)(param_2 + (long)pbVar12);
            uVar9 = puVar1[1];
            uVar10 = puVar1[2];
            uVar11 = puVar1[3];
            puVar2 = (uint *)(param_2 + 0x10 + (long)pbVar12);
            uVar5 = *puVar2;
            uVar6 = puVar2[1];
            uVar7 = puVar2[2];
            uVar8 = puVar2[3];
            puVar2 = (uint *)(__dest + (long)pbVar12);
            *puVar2 = *puVar1 ^ uVar18;
            puVar2[1] = uVar9 ^ uVar18;
            puVar2[2] = uVar10 ^ uVar18;
            puVar2[3] = uVar11 ^ uVar18;
            puVar1 = (uint *)(__dest + 0x10 + (long)pbVar12);
            *puVar1 = uVar5 ^ uVar18;
            puVar1[1] = uVar6 ^ uVar18;
            puVar1[2] = uVar7 ^ uVar18;
            puVar1[3] = uVar8 ^ uVar18;
```

```
            pbVar12 = pbVar12 + 0x20;
          } while (pbVar13 != pbVar12);
          if (pbVar13 == param_3) goto LAB_00108f5b;
          if (((ulong)param_3 & 0x18) != 0) goto LAB_00108f05;
        }
        do {
          __dest[(long)pbVar13] = param_2[(long)pbVar13] ^ bVar3;
          pbVar12 = pbVar13 + 1;
joined_r0x00108f3c:
          pbVar13 = pbVar12;
        } while (pbVar13 != param_3);
        goto LAB_00108f5b;
      }
    }
    else if (param_3 != (byte *)0x0) {
      bVar3 = *param_4;
      bVar4 = *param_2;
      local_38 = param_3;
      pbVar13 = (byte *)__rust_alloc(8,1);
      if (pbVar13 == (byte *)0x0) {
        alloc::alloc::handle_alloc_error(1,8);
        do {
          invalidInstructionException();
        } while( true );
      }
      *pbVar13 = bVar4 ^ bVar3;
      local_68 = &DAT_00000008;
      local_60 = (byte *)0x1;
      param_3 = local_38;
      pbVar16 = &DAT_00000008;
      __dest = pbVar13;
      local_70 = pbVar13;
      if (local_38 != (byte *)0x1) {
        uVar14 = 1;
        do {
          pbVar16 = local_60;
          uVar15 = uVar14 + 1;
          if (param_5 <= uVar15) {
            uVar15 = 0;
          }
          bVar4 = param_2[(long)local_60];
          bVar3 = local_40[uVar14];
          if (local_60 == local_68) {
                    /* try { // try from 00108e4c to 00108e58 has its CatchHandler
@ 00109034 */
```

```
alloc::raw_vec::RawVec<T,A>::reserve::do_reserve_and_handle(&local_70,local_60);
            pbVar13 = local_70;
          }
          pbVar13[(long)pbVar16] = bVar4 ^ bVar3;
          local_60 = pbVar16 + 1;
          param_3 = local_38;
          uVar14 = uVar15;
          pbVar16 = local_68;
          __dest = local_70;
        } while (local_38 != local_60);
      }
      goto LAB_00108f5b;
    }
    param_3 = (byte *)0x0;
    pbVar16 = (byte *)0x0;
    __dest = (byte *)0x1;
  }
LAB_00108f5b:
                    /* try { // try from 00108f5b to 00108f6b has its CatchHandler
@ 00109029 */
  core::str::converts::from_utf8(&local_70,__dest,param_3);
  if (local_70 != (byte *)0x0) {
    local_48 = local_60._1_4_ & 0xffffff | local_60._4_4_ << 0x18;
    uStack_44._0_3_ = (undefined3)((ulong)local_60 >> 0x28);
    if ((char)local_60 != '\x02') {
      local_58 = local_68;
      local_50 = (char)local_60;
      local_4f = local_48;
      uStack_4b = (undefined3)uStack_44;
                    /* try { // try from 00108fd9 to 00108ffd has its CatchHandler
@ 0010901a */
      local_70 = __dest;
      local_68 = pbVar16;
      local_60 = param_3;
      core::result::unwrap_failed
                ("called `Result::unwrap()` on an `Err` valuesrc/main.rsPlease
enter password : ",

0x2b,&local_70,&PTR_drop_in_place<alloc_string_FromUtf8Error>_0015cee8,
                 &DAT_0015cf08);
      do {
        invalidInstructionException();
      } while( true );
    }
  }
  *param_1 = __dest;
```

```
  param_1[1] = pbVar16;
  param_1[2] = param_3;
  return param_1;
}
```

Bon voilà la fonction, ALED....

On voit certain élément qui nous rappellent un XOR :

```
112         auVar19 = pshuflw(auVar19,auVar19,0);
113         uVar18 = SUB164(auVar19,0);
114         pbVar12 = (byte *)0x0;
115         do {
116           puVar1 = (uint *)(param_2 + (long)pbVar12);
117           uVar9 = puVar1[1];
118           uVar10 = puVar1[2];
119           uVar11 = puVar1[3];
120           puVar2 = (uint *)(param_2 + 0x10 + (long)pbVar12);
121           uVar5 = *puVar2;
122           uVar6 = puVar2[1];
123           uVar7 = puVar2[2];
124           uVar8 = puVar2[3];
125           puVar2 = (uint *)(__dest + (long)pbVar12);
126           *puVar2 = *puVar1 ^ uVar18;
127           puVar2[1] = uVar9 ^ uVar18;
128           puVar2[2] = uVar10 ^ uVar18;
129           puVar2[3] = uVar11 ^ uVar18;
130           puVar1 = (uint *)(__dest + 0x10 + (long)pbVar12);
131           *puVar1 = uVar5 ^ uVar18;
132           puVar1[1] = uVar6 ^ uVar18;
133           puVar1[2] = uVar7 ^ uVar18;
134           puVar1[3] = uVar8 ^ uVar18;
135           pbVar12 = pbVar12 + 0x20;
136         } while (pbVar13 != pbVar12);
137         if (pbVar13 == param_3) goto LAB_00108f5b;
138         if (((ulong)param_3 & 0x18) != 0) goto LAB_00108f05;
```

On peut également repérer un appel de fonction avec la string de début de programme qui nous demande de saisir le password :

```
    local_68 = pbVar16;
    local_60 = param_3;
    core::result::unwrap_failed
              ("called `Result::unwrap()` on an `Err` valuesrc/main.rsPlease enter password : ",
               0x2b,&local_70,&PTR_drop_in_place<alloc_string_FromUtf8Error>_0015cee8,
               &DAT_0015cf08);
    do {
      invalidInstructionException();
    } while( true );
  }
}
*param_1 = __dest;
```

Donc on est dans la bonne fonction.

Si on revient à l'appel de "decrypt_bytes" on a ça comme paramètres :

```
LAB_00109406:
  litcrypt_internal::decrypt_bytes(&local_68,&DAT_0014b079,0x27,&DAT_0014b0c1,7);
```

Allons voir le contenu de DAT_0014b079 et DAT_0014b0c1 :

```
                    DAT_0014b079                                    XREF[1]:     main:00109406(*)
0014b079 6c          ??        6Ch    l
0014b07a 33          ??        33h    3
0014b07b 33          ??        33h    3
0014b07c 74          ??        74h    t
0014b07d 6c          ??        6Ch    l
0014b07e 33          ??        33h    3
0014b07f 33          ??        33h    3
0014b080 5f          ??        5Fh    _
0014b081 35          ??        35h    5
0014b082 20          ??        20h
0014b083 7e          ??        7Eh    ~
0014b084 52          ??        52h    R
0014b085 16          ??        16h
0014b086 09          ??        09h
0014b087 5c          ??        5Ch    \
0014b088 42          ??        42h    B
0014b089 23          ??        23h    #
0014b08a 0a          ??        0Ah
0014b08b 6d          ??        6Dh    m
0014b08c 2b          ??        2Bh    +
0014b08d 1f          ??        1Fh
0014b08e 69          ??        69h    i
0014b08f 36          ??        36h    6
0014b090 29          ??        29h    )
0014b091 5b          ??        5Bh    [
0014b092 5b          ??        5Bh    [
0014b093 40          ??        40h    @
0014b094 59          ??        59h    Y
0014b095 10          ??        10h
0014b096 11          ??        11h
0014b097 47          ??        47h    G
0014b098 07          ??        07h
0014b099 40          ??        40h    @
0014b09a 14          ??        14h
0014b09b 1e          ??        1Eh
0014b09c 45          ??        45h    E
0014b09d 38          ??        38h    8
0014b09e 28          ??        28h    (
```

donc :

```
6c 33 33 74 6c 33 33 5f 35 20 7e 52 16 09 5c 42 23 0a 6d 2b 1f 69 36 29 5b 5b 40 59
10 11 47 07 40 14 1e 45 38 28 42
```

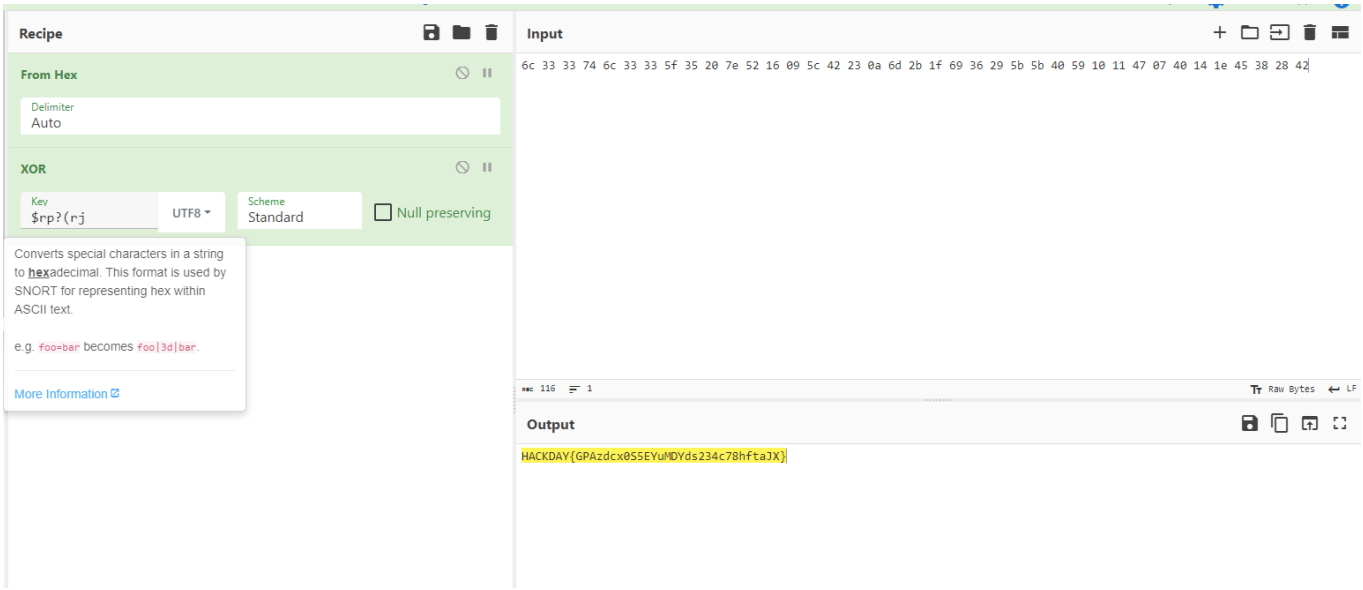
et

```
0014b0c0 0a          ??        0Ah

                    DAT_0014b0c1                                    XREF[1]:     main:0010940d(*)
0014b0c1 24          ??        24h    $
0014b0c2 72          ??        72h    r
0014b0c3 70          ??        70h    p
0014b0c4 3f          ??        3Fh    ?
0014b0c5 28          ??        28h    (
0014b0c6 72          ??        72h    r
0014b0c7 6a          ??        6Ah    j
```

Donc :

```
$rp?(rj
```



On a le flag :

```
HACKDAY{GPAzdcx0S5EYuMDYds234c78hftaJX}
```

CONCLUSION : RUST c'est le souk