Открываем программу в гидре, ищем определенные в файле строки

Интерес для нас представляют следующие:

140008280	I have "cazzima"	u"I have \"cazzima\" "	unicode
1400082a8	Non ci riuscirai mai	u"Non ci riuscirai mai"	unicode
1400082d8	Good Job Bros	u"Good Job Bros"	unicode
1400082f8	Azz O_O	u"Azz O_O"	unicode

Переходи в xref-функцию

```
void UndefinedFunction_140001940(HWND param_1,int param_2,ushort param_3)

{
  HWND hWnd;
  longlong lVar1;
  undefined1 auStack_78 [40];
  WCHAR aWStack_50 [24];
  int iStack_20;
  uint uStack_1c;
  ulonglong uStack_18;

  uStack_18 = DAT_14000b008 ^ (ulonglong)auStack_78;
  iStack_20 = param_2;
  if (param_2 == 0x10) {
    EndDialog(param_1,0);
  }
  else {
    if (param_2 == 0x110) {
      hWnd = GetDlgItem(param_1,1000);
      SendMessageW(hWnd,0xc5,0x14,0);
      goto LAB_14000la50;
    }
    if (param_2 != 0x111) goto LAB_14000la50;
  }
  uStack_1c = (uint)param_3;
  if (uStack_1c == 1) {
    GetDlgItemTextW(param_1,1000,aWStack_50,0x14);
    lVar1 = (*(code *)PTR_FUN_14000b000)(aWStack_50);
    if (lVar1 == 0) {
      MessageBoxW((HWND)0x0,L"Non  ci riuscirai mai",L"I have \"cazzima\" ",0);
    }
    else {
      MessageBoxW((HWND)0x0,L"Azz O_O",L"Good Job Bros",0);
    }
  }
LAB_14000la50:
  __security_check_cookie(uStack_18 ^ (ulonglong)auStack_78);
  return;
}
```

Есть буфер, есть форма. Текст из формы считывается в буфер (28 строка), а по статусу lVar1 выводится нужное сообщение. Так как в 29 строке присваивается значения другой функции, то пойдем в неё, посмотрим, что там происходит

```
                    //
                    // .data
                    // ram:14000b000-ram:14000d19f
                    //

                    PTR_FUN_14000b000                       XREF[3]:    14000024c(*), 140001a11(R),
                                                                        FUN_140001a70:140001aab(W)
    14000b000 00 10 00      addr        FUN_140001000
              40 01 00
              00 00
```

А это указатель на функцию, но в xref указаны места использования

```
undefined8 FUN_140001a70(void)

{
  DAT_14000c040 = ExitProcess_exref + 2;
  if (*(char *)((longlong)ProcessEnvironmentBlock + 2) == '\0') {
    PTR_FUN_14000b000 = FUN_140001ad0;
  }
  else {
    (*DAT_14000c040)(0);
  }
  return 0;
}
```

И далее

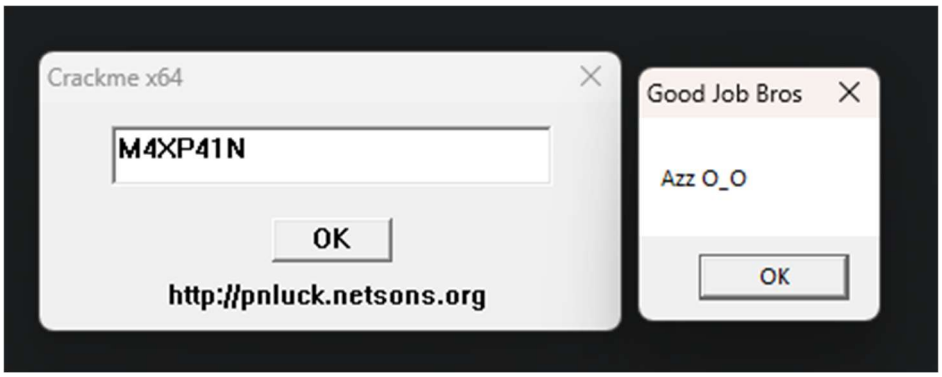
```
undefined4 FUN_140001ad0(short *param_1)

{
  undefined4 local_18;

  local_18 = 0;
  if (((((*param_1 == 0x4d) && (param_1[1] == 0x34)) && (param_1[2] == 0x58)) &&
      ((param_1[3] == 0x50 && (param_1[4] == 0x34)))) &&
     ((param_1[5] == 0x31 && ((param_1[6] == 0x4e && (param_1[7] == 0)))))) {
    local_18 = 1;
  }
  return local_18;
}
```

Вот и цикл проверки пароля. Аски-коды символов в hex-формате

Декодируем

```
if (((((*param_1 == L'M') && (param_1[1] == L'4')) && (param_1[2] == L'X')) &&
    ((param_1[3] == L'P' && (param_1[4] == L'4')))) &&
   ((param_1[5] == L'1' && ((param_1[6] == L'N' && (param_1[7] == 0)))))) {
```

Проверяем


Crackme x64
M4XP41N
OK
http://pnluck.netsons.org
Good Job Bros
Azz O_O
OK