L’appel de la fonction check_credential commence à 0x1216

```
[0x1216]
  ; CALL XREF from sym.ask_credentials @ +0x189(x)
  ; CALL XREF from main @ 0x140e(x)
377: sym.ask_credentials ();
; var int32_t var_4h @ ebp-0x4
; var int32_t var_ch @ ebp-0xc
; var char *s @ ebp-0x4c
; var int32_t var_8ch @ ebp-0x8c
; var int32_t var_90h @ ebp-0x90
; var int32_t var_94h @ ebp-0x94
push ebp
; '\xff\xff\xff\xff\xff\xff\xff\xff'
mov ebp, esp
push ebx
sub esp, 0x94
call sym.__x86.get_pc_thunk.bx;[oa]
add ebx, 0x2d9f
; '\xff\xff\xff\xff\xff\xff\xff\xff'
mov eax, dword gs:[0x14]
; '\xff\xff\xff\xff\xff\xff\xff\xff'
mov dword [var_ch], eax
xor eax, eax
sub esp, 0xc
lea eax, [ebx - 0x1fad]
; const char *format
push eax
; int printf(const char *format)
call sym.imp.printf;[ob]
add esp, 0x10
sub esp, 8
lea eax, [var_94h]
push eax
lea eax, [ebx - 0x1fa2]
; const char *format
push eax
```

Par la suite, on vient ramasser l'input de l'utilisateur pour le nom avec sym.imp.__isoc99_scanf. Par la suite, on fait un check sur le register eax

```
mov ebp, esp
push ebx
sub esp, 0x94
call sym.__x86.get_pc_thunk.bx;[oa]
add ebx, 0x2d9f
; '\xff\xff\xff\xff\xff\xff\xff\xff'
mov eax, dword gs:[0x14]
; '\xff\xff\xff\xff\xff\xff\xff\xff'
mov dword [var_ch], eax
xor eax, eax
sub esp, 0xc
lea eax, [ebx - 0x1fad]
; const char *format
push eax
; int printf(const char *format)
call sym.imp.printf;[ob]
add esp, 0x10
sub esp, 8
lea eax, [var_94h]
push eax
lea eax, [ebx - 0x1fa2]
; const char *format
push eax
; int scanf(const char *format)
call sym.imp.__isoc99_scanf;[oc]
add esp, 0x10
cmp eax, 1
je 0x127d
```

```
0x00001225    81c39f2d0000    add ebx, 0x2d9f
0x0000122b    65a114000000    mov eax, dword gs:[0x14]
0x00001231    8945f4          mov dword [var_ch], eax
0x00001234    31c0            xor eax, eax
0x00001236    83ec0c          sub esp, 0xc
0x00001239    8d8353e0ffff    lea eax, [ebx - 0x1fad]
0x0000123f    50              push eax
0x00001240    e80bfeffff      call sym.imp.printf
0x00001245    83c410          add esp, 0x10
0x00001248    83ec08          sub esp, 8
0x0000124b    8d856cffffff    lea eax, [var_94h]
0x00001251    50              push eax
0x00001252    8d835ee0ffff    lea eax, [ebx - 0x1fa2]
0x00001258    50              push eax
0x00001259    e842feffff      call sym.imp.__isoc99_scan
0x0000125e    83c410          add esp, 0x10
0x00001261    83f801          cmp eax, 1
0x00001264    7417            je 0x127d
0x00001266    83ec0c          sub esp, 0xc
0x00001269    8d8362e0ffff    lea eax, [ebx - 0x1f9e]
0x0000126f    50              push eax
0x00001270    e80bfeffff      call sym.imp.puts
```

Ensuite on refait le même processus pour le mot de passe

Finalement, il y a une fonction qui semble refaire le processus. Je dirais donc qu'on fait deux check_credentials et il y en a un de faux et un de vrai.

```
 0x12c4 [om]
; CODE XREF from sym.ask_credentials @ 0x12ab(x)
; '\xff\xff\xff\xff\xff\xff\xff\xff'
mov eax, dword [var_94h]
sub esp, 8
lea edx, [var_8ch]
; int32_t arg_ch
push edx
; int32_t arg_8h
push eax
call sym.get_flag;[oi]
add esp, 0x10
; '\xff\xff\xff\xff\xff\xff\xff\xff'
mov dword [var_90h], eax
; '\xff\xff\xff\xff\xff\xff\xff\xff'
mov eax, dword [var_94h]
sub esp, 0xc
lea edx, [var_8ch]
push edx
;  ...
push eax
lea eax, [ebx - 0x1f8a]
; const char *format
push eax
; size_t size
; '@'
push 0x40
lea eax, [s]
; char *s
push eax
; int snprintf(char *s, size_t size, const char *for
call sym.imp.snprintf;[oj]
add esp, 0x20
sub esp, 8
lea eax, [s]
```

```
; int32_t arg_8h
push eax
call sym.check_credentials;[ol]
add esp, 0x10
test eax, eax
jne 0x134d
```

Pour avoir le username et le flag, il faudrait décoder la logique derrière les manipulations de registrer et trouver les valeurs