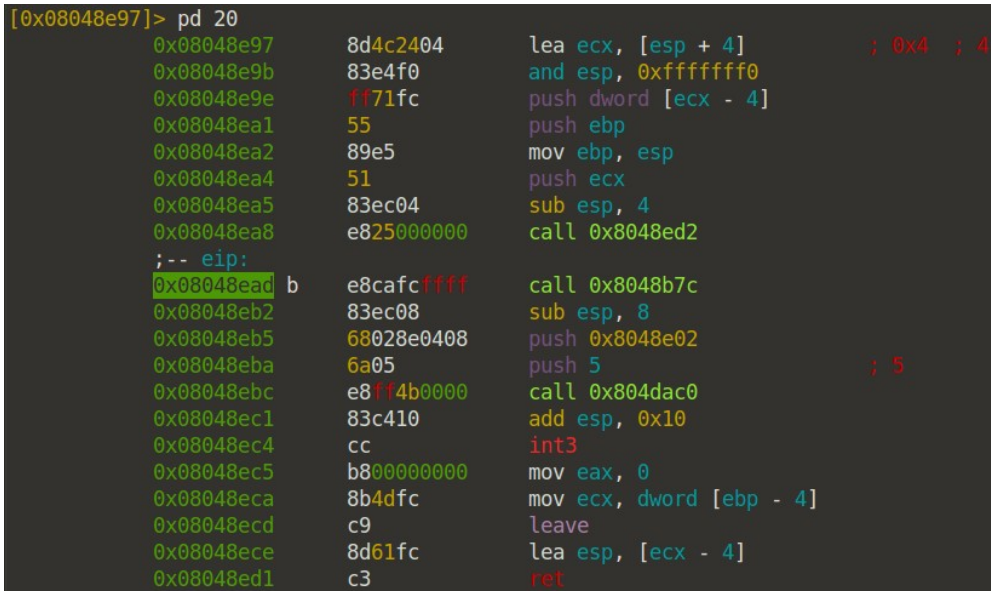
CrackMe – ELF x86 : Solution

(Solution écrite avec radare2)

[Le main()] :

→ La 1ère fonction s'occupe de dé-xorer une partie du code.

On peut donc placer un **breakpoint** à l'appel de la fonction suivante et continuer l'exécution afin de pouvoir examiner le vrai code des 2 fonctions suivantes.



→ La 2ème fonction effectue différentes opérations arithmétique en fonction des octets du code afin de calculer une sorte de **checksum**.

→ L'appel à la 3ème fonction est protégé par un faux point d'arrêt grâce à l'instruction **int3** qui génère un signal **SIGTRAP** et quitte le programme lorsqu'il est lancé dans un debugger, ou continue dans le **handler** du **SIGTRAP** en exécution normale. Le **PUSH** juste avant l'appel nous montre l'adresse du **handler**.

[Le sighanlder()] :

→ Dans ce dernier on observe 5 appels de fonctions :

- **call 0x8048c34** : affiche la demande de mot de passe
- **call 0x804f820** : attend une saisie de l'utilisateur
- **call 0x8048bd8** : re-calcul le **checksum**

  (Comparaison entre les 2 checksum)
- **call 0x8048da5** : déchiffre une partie du code

- **call 0x8048c8d** : vérifie le mot de passe saisie
- **call 0x804e760** : quitte le programme

Comme radare2 ne gère pas très bien les signaux, on doit XOR-er manuellement la fonction de vérification du mot de passe, successivement avec **0xAC** et **0xDE** *(XOR-er une première fois dans le **main()** puis dans le **sighandler()**)*, avec la commande *''**wox**''* afin d'obtenir le code original.

Le mot de passe saisie est ensuite comparé avec un l'octet présent tous les 3 octets (le 1er octet du mot de passe si situe au 4ème de la liste) de la suite d'octets que l'on voit ci-dessous.

```
0x08048ca4      c745ae190a09.  mov dword [ebp - 0x52], 0x6f090a19 ; [0x6f090a19:4]=-1
0x08048cab      c745b2595a53.  mov dword [ebp - 0x4e], 0x4f535a59 ; [0x4f535a59:4]=-1
0x08048cb2      c745b6580a5e.  mov dword [ebp - 0x4a], 0x85e0a58 ; [0x85e0a58:4]=-1
0x08048cb9      c745ba480951.  mov dword [ebp - 0x46], 0x64510948 ; [0x64510948:4]=-1
0x08048cc0      c745be515f0a.  mov dword [ebp - 0x42], 0x550a5f51 ; [0x550a5f51:4]=-1
0x08048cc7      c745c2504854.  mov dword [ebp - 0x3e], 0x57544850 ; [0x57544850:4]=-1
0x08048cce      c745c6645752.  mov dword [ebp - 0x3a], 0x6f525764 ; [0x6f525764:4]=-1
0x08048cd5      c745ca5e5053.  mov dword [ebp - 0x36], 0x5f53505e ; [0x5f53505e:4]=-1
0x08048cdc      c745ce510855.  mov dword [ebp - 0x32], 0x5a550851 ; [0x5a550851:4]=-1
0x08048ce3      c745d2644e58.  mov dword [ebp - 0x2e], 0x7d584e64 ; [0x7d584e64:4]=-1
0x08048cea      c745d64b5e57.  mov dword [ebp - 0x2a], 0x52575e4b ; [0x52575e4b:4]=-1
0x08048cf1      c745da530f5f.  mov dword [ebp - 0x26], 0x5c5f0f53 ; [0x5c5f0f53:4]=-1
0x08048cf8      c745de5c4e5e.  mov dword [ebp - 0x22], 0x1a5e4e5c ; [0x1a5e4e5c:4]=-1
0x08048cff      c745e2424200.  mov dword [ebp - 0x1e], 0x4242 ; [0x4242:4]=-1
0x08048d06      c745e6000000.  mov dword [ebp - 0x1a], 0
0x08048d0d      c745ea000000.  mov dword [ebp - 0x16], 0
0x08048d14      c745ee000000.  mov dword [ebp - 0x12], 0
0x08048d1b      66c745f20000   mov word [ebp - 0xe], 0
0x08048d21      c745a8000000.  mov dword [ebp - 0x58], 0
```

Une fois la chaîne reconstitué en une seule dans l'ordre (**LIFO**), un script python va donner le bon mot de passe. La clé étant **0x3B**.

"\x19\x0a\x09\x6f\x59\x5a\x53\x4f\x58\x0a\x5e\x08\x48\x09\x51\x64\x51\x5f\x0a\x55\x50\x48\x54\x57\x64\x57\x52\x6f\x5e\x50\x53\x5f\x51\x08\x55\x5a\x64\x4e\x58\x7d\x4b\x5e\x57\x52\x53\x0f\x5f\x5c\x5c\x4e\x5e\x1a\x42\x42"

```
madmath@Mathrix:~/Documents/Root Me/Crack-Me/My Challs/Chall 1 - ELF x86/verion final$ ./reverse_pass.py
Pass : Th1s_1s_Th3_Fl4g!
madmath@Mathrix:~/Documents/Root Me/Crack-Me/My Challs/Chall 1 - ELF x86/verion final$ ./crackme
Password : Th1s_1s_Th3_Fl4g!
Connected !
```