CrackMe – A Little VM : Solution

(Solution écrite avec radare2)

Le binaire est protégé par une machine virtuelle simple qui suit le modèle :
fetch → execute.

[Le main()]

Une fois dans le main, on aperçoit seulement 2 appels de fonction.

- La 1ère fonction s'occupe de récupérer un octet en fonction de la valeur de **EAX** avec comme adresse de base : **0x8048880**.
- La 2ème fonction exécute une action selon la valeur de **EAX** *(sur 1 octets)*.

[La boucle principale]

La 1ère valeur récupérer est **0x20**, une valeur est ajouter dans une variable *(qui fait office de **stack**)*, et ensuite récupérer et afficher par l'itération suivante de la boucle, correspondant à l'octet **0x23**.

Après l'analyse de plusieurs boucles en exécution pas à pas (*), on note que les octets suivant sont particulièrement intéressant pour la compréhension du programme :

- **0x24**
  - C'est à ce moment que le mot de passe est demandé à l'utilisateur avec **fgets()**.
- **0x30**
  - Pour cet octet, le programme va effectuer une comparaison avec la valeur hexa du caractère que l'on a rentré précédemment *(ici = root)* et **0x41** *(situé dans une suite d'octets en **0x0804a060**)*, puis **0x6f** *("o")* et **0x6e**.

```
eip 0x08048705    oeax 0xffffffff     eax 0x00000072     ebx 0xf77a1000
ecx 0x0804a0e0     edx 0x00000002     esp 0xffb04880     ebp 0xffb048a8
esi 0x00000000     edi 0x080483a0     eflags = 1I
          0x080486cb      8b048520a104.  mov eax, dword [eax*4 + 0x804a120] ; [0x804a120:4]=12
          0x080486d2      89c2           mov edx, eax
          0x080486d4      a160a50408     mov eax, dword [0x804a560]  ; [0x804a560:4]=0x804a0e0 ecx
          0x080486d9      01d0           add eax, edx
          0x080486db      0fb600         movzx eax, byte [eax]
          0x080486de      8845da         mov byte [ebp - 0x26], al
          0x080486e1      a140a00408     mov eax, dword [0x804a040]  ; [0x804a040:4]=1
          0x080486e6      8b048520a104.  mov eax, dword [eax*4 + 0x804a120] ; [0x804a120:4]=12
          0x080486ed      83c001         add eax, 1
          0x080486f0      01c0           add eax, eax
          0x080486f2      89c2           mov edx, eax
          0x080486f4      a16ca50408     mov eax, dword [0x804a56c]  ; [0x804a56c:4]=0x804a060
          0x080486f9      01d0           add eax, edx
          0x080486fb      0fb600         movzx eax, byte [eax]
          0x080486fe      8845db         mov byte [ebp - 0x25], al
          0x08048701      0fb645da       movzx eax, byte [ebp - 0x26]
          ;-- eip:
          0x08048705 b    3a45db         cmp al, byte [ebp - 0x25]
```

Si on va voir ce qu'il se trouve à cette adresse :

```
:> px@0x804a060
- offset -   0 1  2 3  4 5  6 7  8 9  A B  C D  E F  0123456789ABCDEF
0x0804a060  190a 416f 6e5a 304f 740a 6808 6509 7264  ..AonZ0Ot.h.e.rd
0x0804a070  5f5f 4655 4c48 3457 4700 0000 0000 0000  __FULH4WG.......
```

On voit clairement une suite de caractères hexa qui semblent faire partie de la tables **ASCII**. Logiquement, la première lettre du passe paraît être un "**A**".

Dans la chaîne récupérer juste avant, on constate que le caractère comparé est situé 2 caractère plus loin que le premier.

[Le Flag]

On peut essayer un script python qui récupère 1 caractère sur 2 à partir du "**A**" et voir si sa correspond au mot de passe.

```
madmath@Mathrix:~/Documents/Root Me/Crack-Me/My Challs/Chall 2 - ELF VM/VM$ ./reverse_pass.py
Pass : An0ther_FL4G
madmath@Mathrix:~/Documents/Root Me/Crack-Me/My Challs/Chall 2 - ELF VM/VM$ ./vm
Password : An0ther_FL4G
Well done!
```

(*) : Avec radare2, une exécution pas à pas peut s'effectuer en mode visuel
1) "**V**" dans la console pour entrer en mode visuel.
2) Puis "**s**" pour avancer d'une instruction.