

SECURINETS



**CLUB DE LA SECURITE INFORMATIQUE
INSAT**

SECURILIGHT 2011

Crack des logiciels

Chef Atelier : Moez KHORCHENI (GL3)

Prénom Amira ABID (GL3)

Hana CHARRADI (GL3)

Nesrine DRIWECH (RT3)

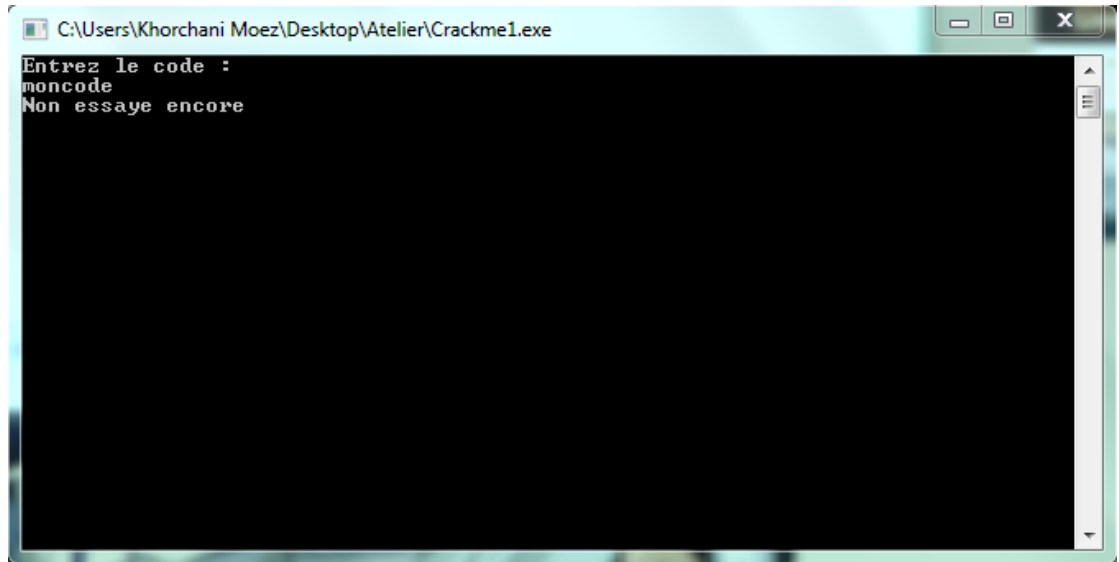
Selma KAROUI (GL3)

Souha KASSAB (RT3)

30/11/2011

I. Crack me1 :

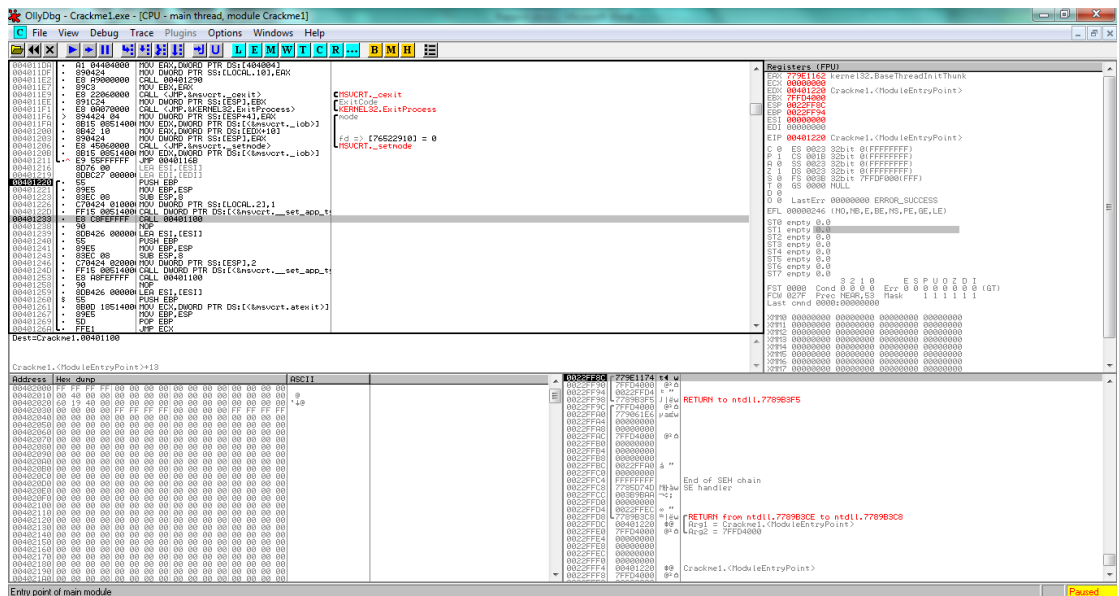
Exécuter, observer et analyser le comportement du programme :



Le programme demande un code, le teste et nous renvoie le résultat du test

On cherche donc à déterminer le code (valide) qui sera accepté par le programme

Importation de l'exécutable : crack-me1.exe dans ollydbg :



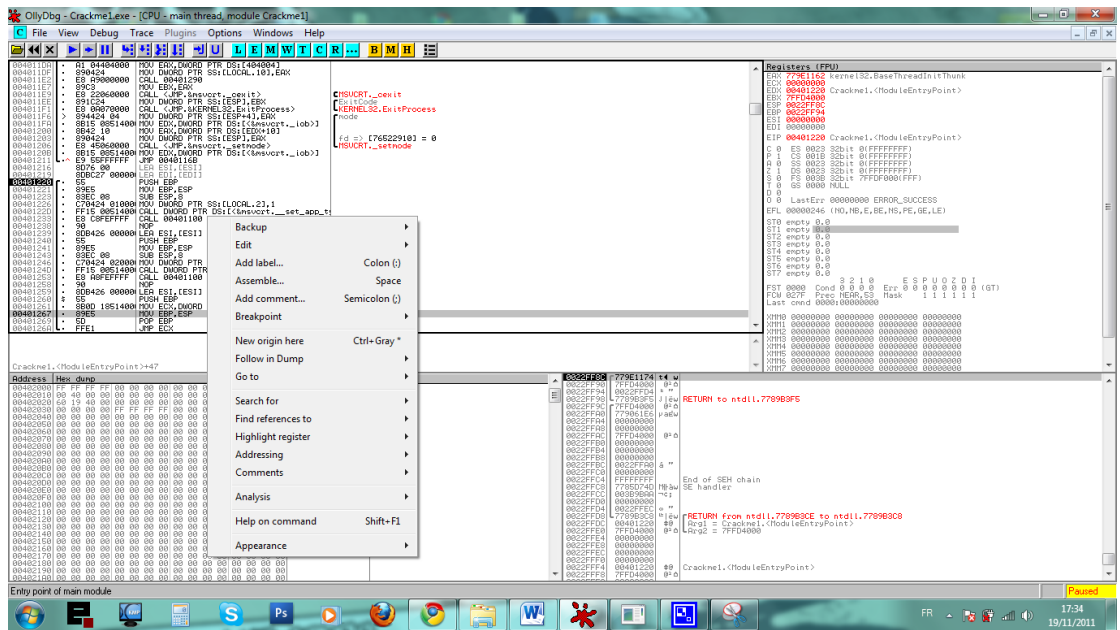
Ollydbg analyse cet exécutable et nous affiche son code assembleur.

Il faut que nous trouvions le bout de code qui compare le code que nous entrons avec le vrai code (dont le programme a besoin)

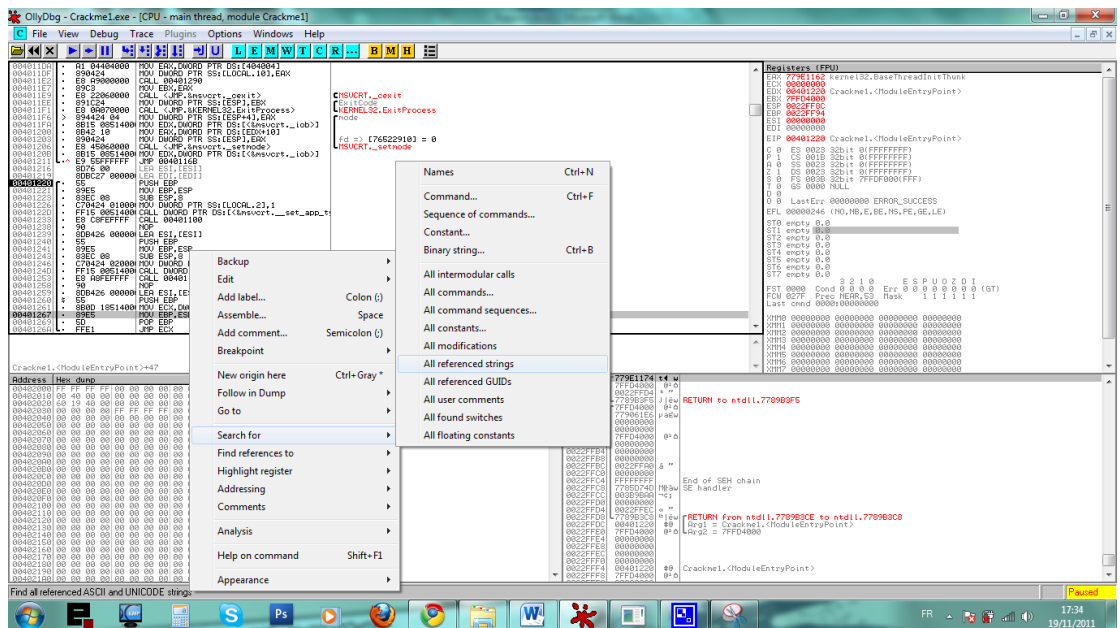
On remarque tout d'abord, qu'en entrant un code invalide, le programme affiche un message « Non essaye encore », on va alors « intuitivement » chercher ce message dans le programme.

En fait, ollydbg permet de chercher toutes les chaines de caractères manipulées par le programme, pour le faire on procède comme suit :

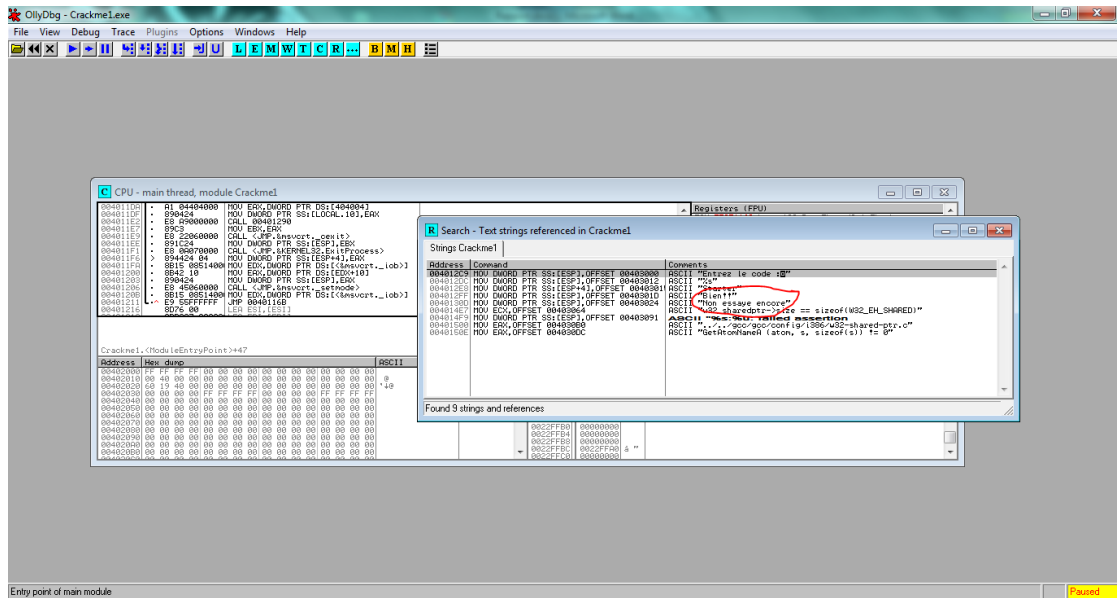
Un clic droit :



Search for → all referenced strings



Une nouvelle fenêtre s'affiche :



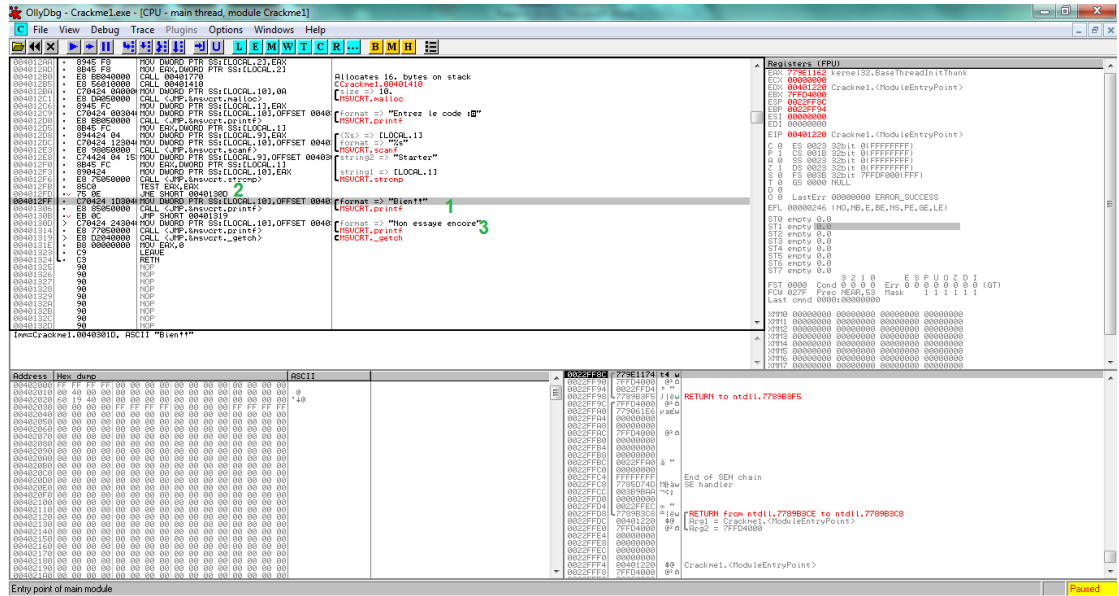
Là on remarque bien deux messages importants :

Le message qu'on cherche : « Non essaye encore »

Un message : « Bien !! »

On devine facilement que le dernier message est celui qui va être affiché si on entre le code valide !!!!

Un double clic sur le message « Bien !! » nous ramène au code qui affiche ce message dans l'autre fenêtre (CPU) :



Analyse :

On regarde ici :

A droite on lit :

string1 → « **Starter** » : ça paraît très important, hein ?!

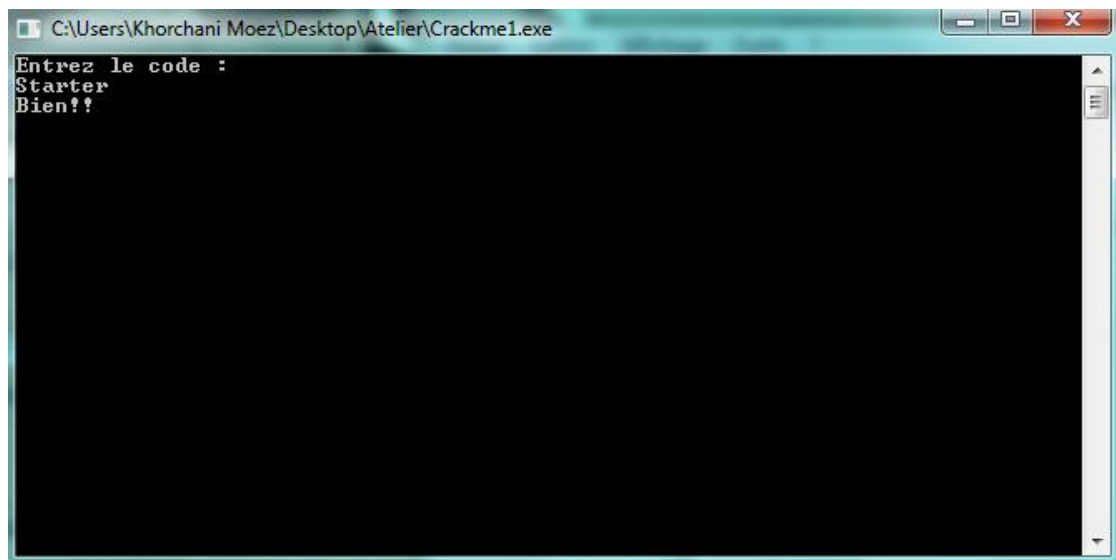
string2 → [local.1] : un message variable ? Ça sent le message que nous entrons à chaque fois, non ?

MSVCRT.strcmp : Oh, on connaît bien la fonction strcmp du C (Rappel : elle permet de comparer deux chaînes de caractères)

Et voilà !! Donc on fait l'hypothèse que « Starter » est notre fameux code.

Pour être sûr :

On n'a qu'à exécuter le programme en lui donnant « Starter » comme code :



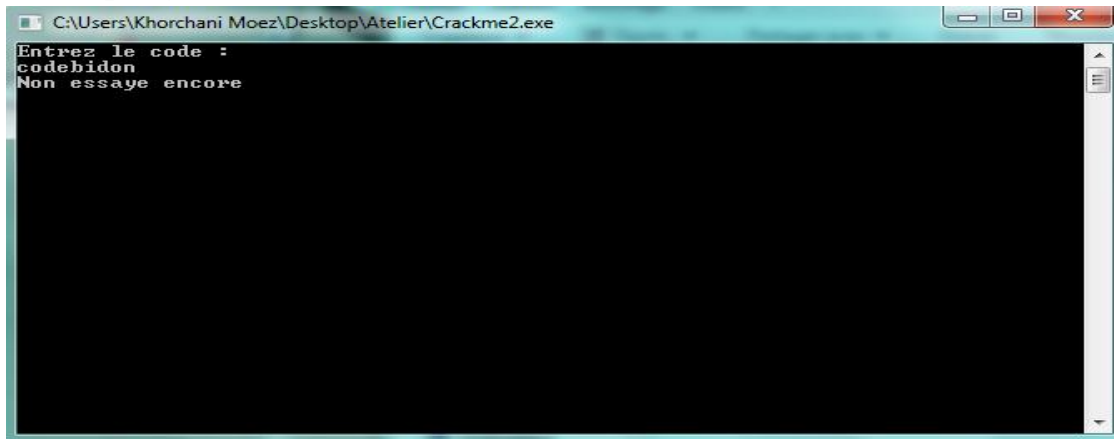
```
C:\Users\Khorchani Moez\Desktop\Atelier\Crackme1.exe
Entrez le code :
Starter
Bien!!
```

ET VOILA TOUT EST BIEN QUI FINIT BIEN, ON A DETERMINE LE CODE VALIDE.

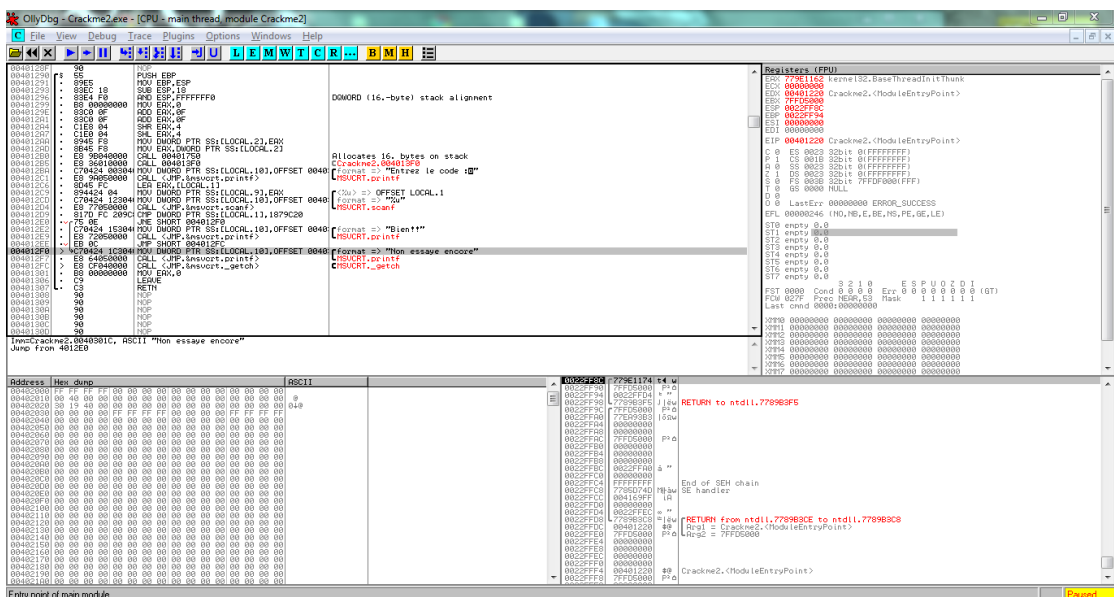
REMARQUE : Il y a une infinité de méthodes permettant de trouver ce code, on en a choisi juste une 😊

II. Crack me2 :

Exécuter, observer et analyser le comportement du programme :



On suit la même démarche que précédemment :



On voit les messages « Bien » et « Non essaye encore »

On voit aussi la fameuse : JNE SHORT 004012F0 qui décide d'afficher « bien » ou « non... ». Il va y avoir sûrement une comparaison juste avant ce JNE. En regardant on ne trouve plus cette fois le strcmp., mais on trouve une autre ligne :

CMP DWORD PTR SS:[EBP-4], 1879C20

Un Securinetsien ayant une connaissance basique en assembleur pourra comprendre aisément que CMP est une comparaison ☺

Une comparaison, oui, mais entre quoi et quoi (?) :

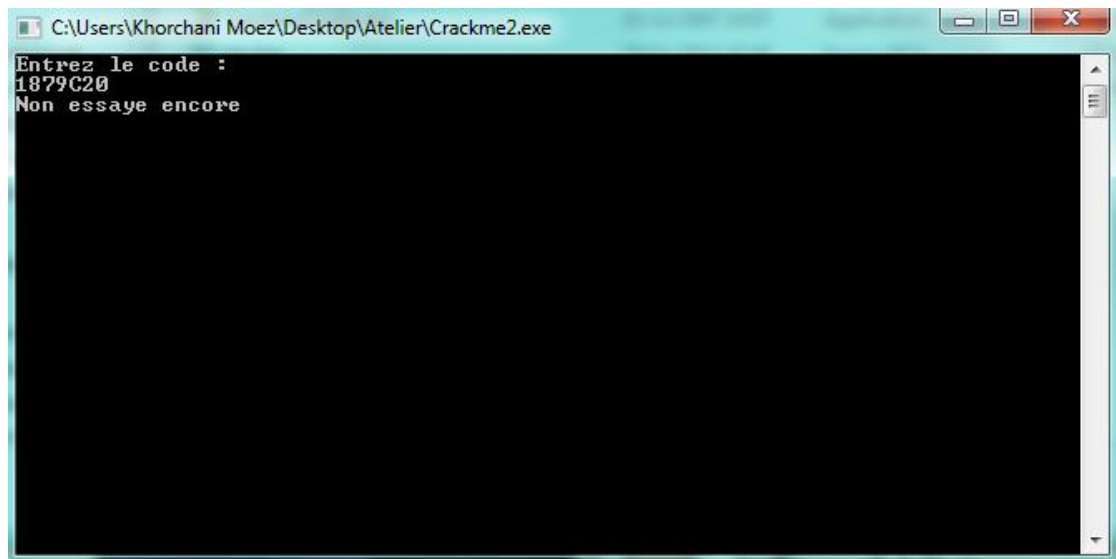
Entre : DWORD PTR SS:[EBP-4] : C'est une donnée stockée en mémoire à l'adresse EPB-4

1879C20 : Une constante

➔ Comparaison entre une variable et une constante, qui décide si le code est vrai ou pas

La constante est sans doute le code voulu

On teste l'hypothèse que notre code est bien «1879C20 » :



??? Mais pourquoi il n'accepte pas ce code ?? J'étais pourtant sûr de moi !!!

Ah oui, on sait bien que toutes les constantes en assembleur sont représentées en HEXA.

Donc avec la calculatrice Windows on convertit cette constante en décimal.

On obtient : 25664544

ET VOILA UNE DEUXIEME FOIS ON RETROUVE LE CODE VALIDE. 😊