Cours N° 4

Ce cours va vous apprendre comment faire un Keygen pour Hexa.exe.

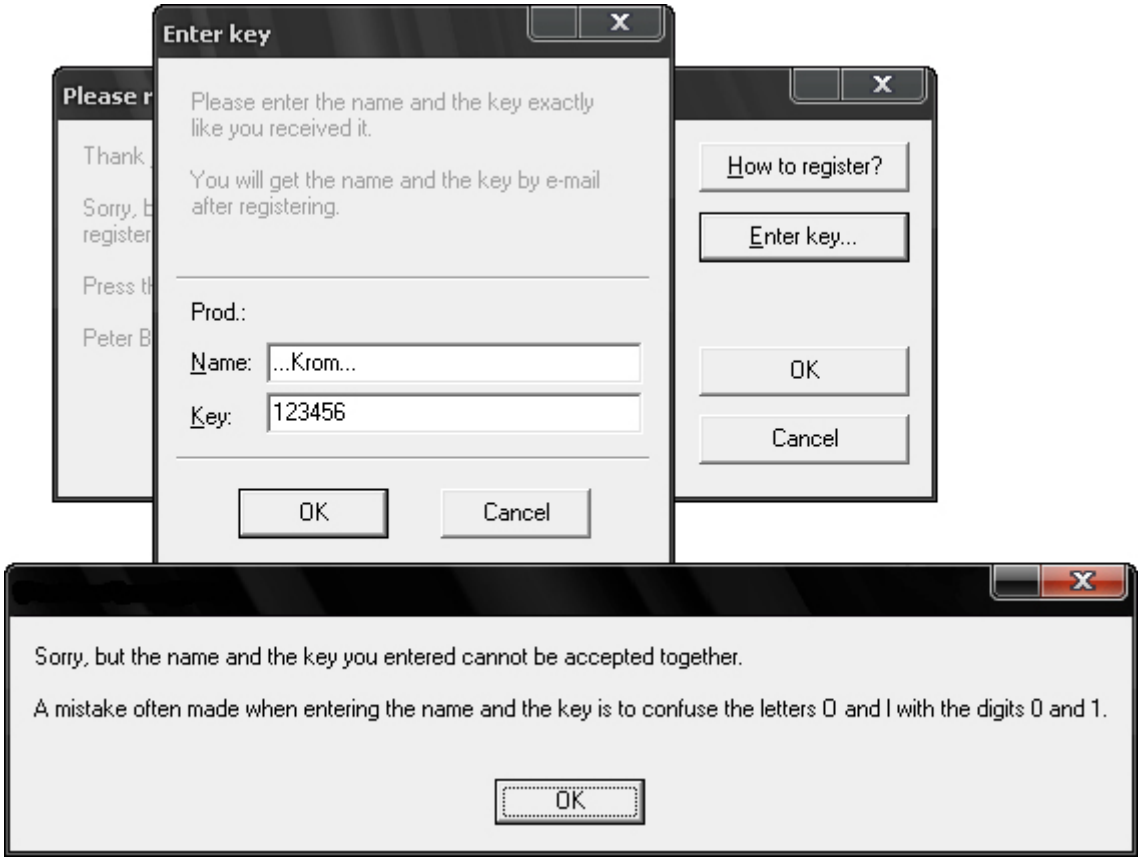
Il est téléchargeable ici :

- http://www.KromCrack.com/prog/HexaCours4.exe

(Password : KrommorK)

Un Keygen, aussi appelé KeyGenerator ou Générateur de Clés est un logiciel générant des numéros de série afin d'installer / déverrouiller / lancer une application payante. La difficulté de ce type de Crack est de comprendre l'algorithme utilisé par le programme pour ensuite le reproduire dans n'importe quel langage de programmation, (C, C++, ASM ...). La compréhension de l'algorithme et la programmation d'un Keygen est le fin du fin pour un Cracker car il n'y a aucune modification du programme.

La première étape consiste à trouver la routine de calcule du Sérial. Pour la localiser dans le programme, on doit trouver le message d'erreur et remonter ensuite dans le code pour voir a quel moment le code est créé a partir du nom entré.



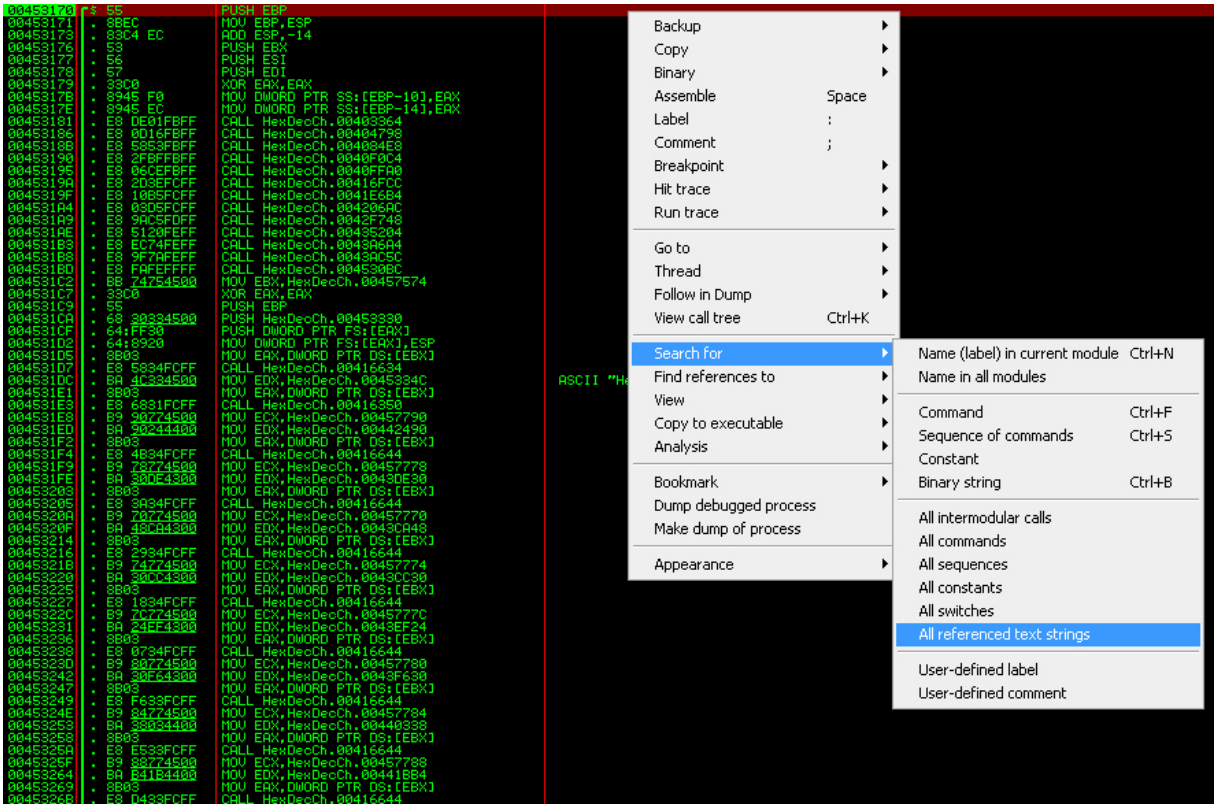
Alors quand vous entrez un nom de plus de 10 caractères, cf. Cours N° 3

- http://www.KromCrack.com/cours3.php

Vous voyez que le message d'erreur indiquant qu'un code est invalide est:

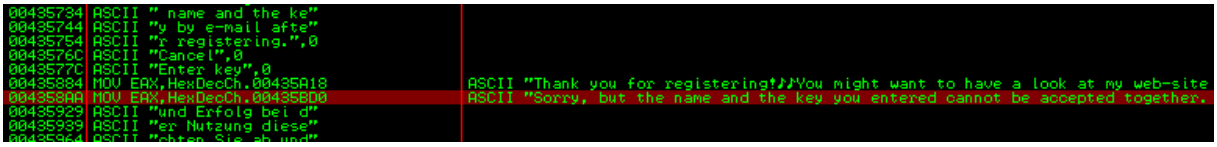
"Sorry, but the name and the key you entered cannot be accepted together".

Alors ouvrez "Hexa.exe" dans OllyDBG et rechercher le message d'erreur : clic droit puis "Search for" ->> "All referenced text strings".

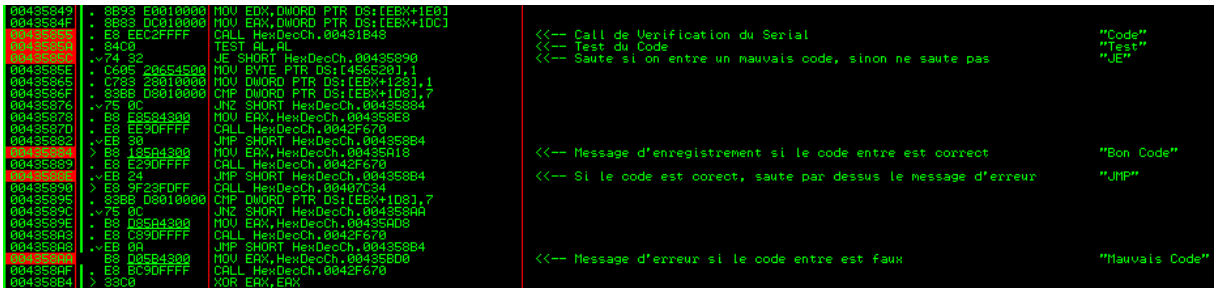


Dans cette nouvelle fenêtre, cherchez le contenu du message d'erreur, a savoir "Sorry, but the name and the key you entered cannot be accepted together".

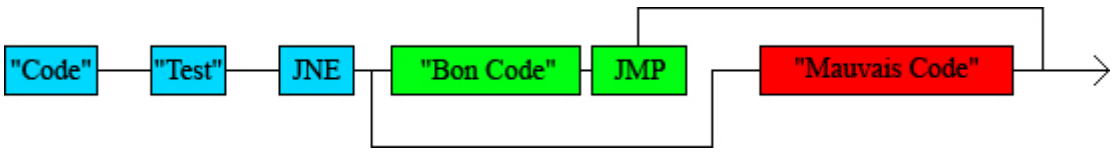
Vous trouverez 2 occurrences, une en 00435BD0 et une en 004358AA et c'est cette dernière qui nous intéresse :



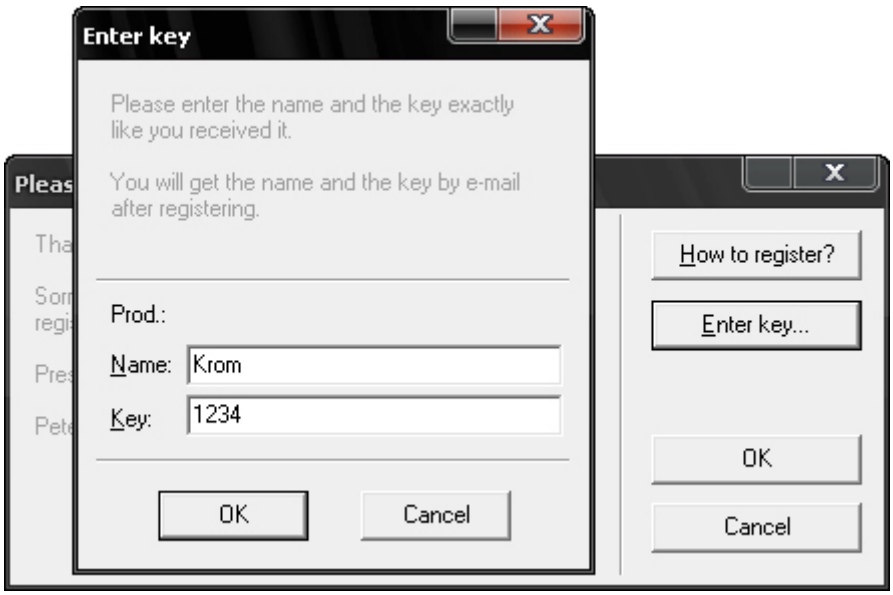
Alors, après avoir double cliquer dessus, vous arriverez là :



Nous voyons ici que le fait d'être enregistré ou non dépend du saut JE SHORT HexDecCh.00435890 en 0043585C. le fait de sauter ou non dépend du résultat du TEST AL,AL en 0043585A et c'est dans le CALL qui le précède que l'on va trouver notre routine. Mettez un breakpoint sur ce CALL avec F2.

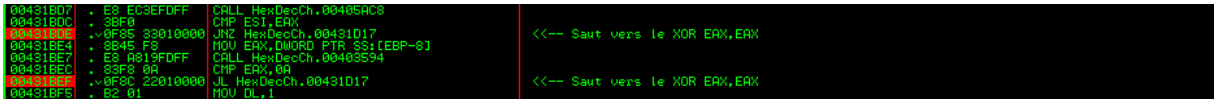


Alors lancez le programme dans OllyDBG avec F9 et enregistrez-vous (Krom / 1234) puis cliquez sur "Ok".

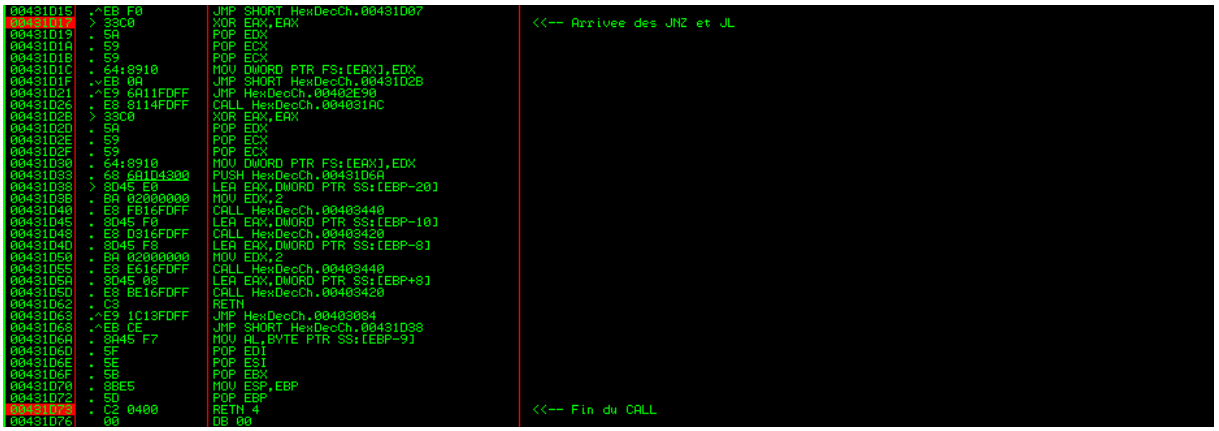


OllyDBG break en 00435855. Entrez dans le CALL avec F7 :

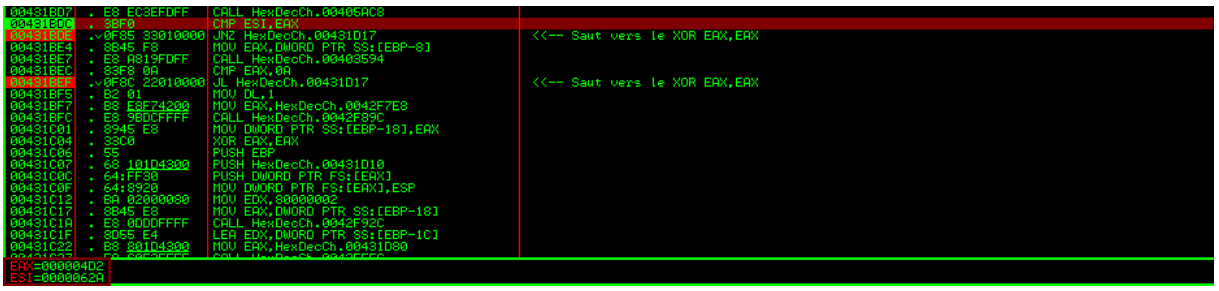
...



...



On voit ici que l'on saute en 00431BDE (JNZ HexDecCh.00431D17) et en 00431BEF (JL HexDecCh.00431D17) pour arriver en 00431D17 ce qui a pour but de remettre EAX (donc indirectement) AL à 0. Si AL est à 0 en 0043585C, le JE (Jump if Equal to 0) Sautera par dessus le bon message pour afficher le message d'erreur. Il faut donc que les 2 sauts (JNZ HexDecCh.00431D17 et JL HexDecCh.00431D17) ne sautent pas. Intéressons-nous d'abord au premier saut (JNZ HexDecCh.00431D17).



Pour qu'il ne saute pas, on doit avoir ESI égal à EAX donc comme ca il n'y aura aucune différence entre ESI et EAX et le JNZ ne sautera pas car Jump if Not Zéro, pas 0 -> pas de saut ;).

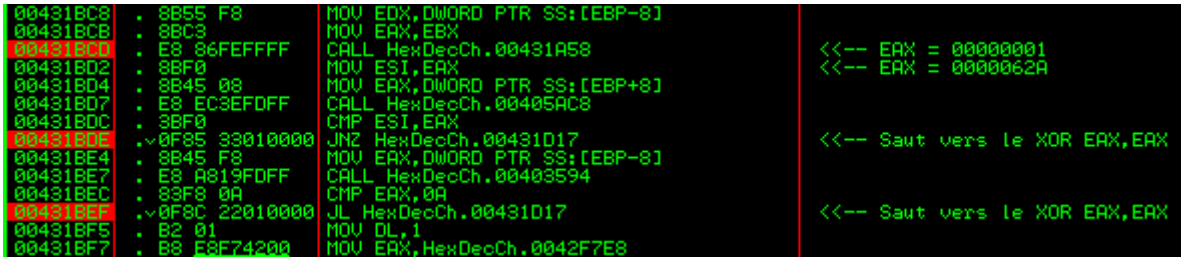
Donc pour qu'il n'y ait aucune différence entre le bon et le mauvais code, le code entré doit être bon logique non ? ;) Il ne suffit plus qu'a convertir 4D2 (EAX) et 62A (ESI) en Décimal car les registres sont en Hexadécimal

Registre	Valeur Hexadécimale	Valeur Décimale
EAX	000004D2	1234
ESI	0000062A	1578

Le Sérial pour Krom est donc 1578. Voyons maintenant ou ce Sérial est généré. Pour savoir d'ou viens ce Sérial il suffit de débugger ligne par ligne et dès que l'on verra notre Sérial (62A) apparaître dans un EAX (EAX = Sérial entré), on saura que le Sérial aura été généré dans le CALL qui le précède ... C'est assez difficile à expliquer sans exemple, donc en voici un qui vous sera surement utile !

Redémarrez avec CTRL + F2 -> Oui, lancez le ensuite en enregistrez vous avec Krom / 1234. Le programme break sur le CALL HexDecCh.00431B48 en 00435855. Entrez-y avec F7, puis faites une série de F8 jusqu'a voir 62A dans EAX. la première fois que l'on voit 62A est à l'instruction MOV ESI,EAX en 00431BD2, donc le Sérial est généré dans ce CALL :

00431BCD . E8 86FEFFFF CALL HexDecCh.00431A58



Mettez un breakpoint en 00431BDC avec F2 puis redémarrez avec CTRL + F2 -> Oui, lancez le programme avec F9 et en enregistrez vous avec Krom / 1234. Le programme break sur le CALL HexDecCh.00431B48 en 00435855. Entrez-y avec F7 puis une série de F8 jusqu'au CALL 00431BDC. Entrez-y avec F7. En faisant quelques F8, vous arriverez à une boucle qui est exécutée 4 fois (Krom contient 4 lettres, la boucle sera exécutée le nombre de fois que de lettre contenue dans le nom) et si vous faites un peu attention à ce qui se passe dans cette boucle (au niveau des registres) vous y verrez Krom lettre par lettre, multiplié, divisé, mis dans un emplacement mémoire ... et donc on est sur que c'est ici qu'on a notre routine de calcule du Sérial.

```
00431B46   . 8BC0          MOV EAX,EAX
00431B48   $ 55            PUSH EBP                 <<-- Debut du CALL
00431B49   . 8BEC          MOV EBP,ESP
```

...

```
00431BD7   . E8 EC3EFDFF   CALL HexDecCh.00405AC8
00431BDC   . 3BF0          CMP ESI,EAX
00431BDE   .v0F85 33010000 JNZ HexDecCh.00431D17    <<-- Saut vers le XOR EAX,EAX
00431BE4   . 8B45 F8       MOV EAX,DWORD PTR SS:[EBP-8]
00431BE7   . E8 A819FDFF   CALL HexDecCh.00403594
00431BEC   . 83F8 0A       CMP EAX,0A
00431BEF   .v0F8C 22010000 JL HexDecCh.00431D17     <<-- Saut vers le XOR EAX,EAX
00431BF5   . B2 01         MOV DL,1
```

...

```
00431D15   .^EB F0         JMP SHORT HexDecCh.00431D07
00431D17   > 33C0          XOR EAX,EAX              <<-- Arrivee des JNZ et JL
00431D19   . 5A            POP EDX
00431D1A   . 59            POP ECX
00431D1B   . 59            POP ECX
00431D1C   . 64:8910       MOV DWORD PTR FS:[EAX],EDX
00431D1F   .vEB 0A         JMP SHORT HexDecCh.00431D2B
00431D21   .^E9 6A11FDFF   JMP HexDecCh.00402E90
00431D26   . E8 8114FDFF   CALL HexDecCh.004031AC
00431D2B   > 33C0          XOR EAX,EAX
00431D2D   . 5A            POP EDX
00431D2E   . 59            POP ECX
00431D2F   . 59            POP ECX
00431D30   . 64:8910       MOV DWORD PTR FS:[EAX],EDX
00431D33   . 68 6A1D4300   PUSH HexDecCh.00431D6A
00431D38   > 8D45 E0       LEA EAX,DWORD PTR SS:[EBP-20]
00431D3B   . BA 02000000   MOV EDX,2
00431D40   . E8 FB16FDFF   CALL HexDecCh.00403440
00431D45   . 8D45 F0       LEA EAX,DWORD PTR SS:[EBP-10]
00431D48   . E8 D316FDFF   CALL HexDecCh.00403420
00431D4D   . 8D45 F8       LEA EAX,DWORD PTR SS:[EBP-8]
00431D50   . BA 02000000   MOV EDX,2
00431D55   . E8 E616FDFF   CALL HexDecCh.00403440
00431D5A   . 8D45 08       LEA EAX,DWORD PTR SS:[EBP+8]
00431D5D   . E8 BE16FDFF   CALL HexDecCh.00403420
00431D62   . C3            RETN
00431D63   .^E9 1C13FDFF   JMP HexDecCh.00403084
00431D68   .^EB CE         JMP SHORT HexDecCh.00431D38
00431D6A   . 8A45 F7       MOV AL,BYTE PTR SS:[EBP-9]
00431D6D   . 5F            POP EDI
00431D6E   . 5E            POP ESI
00431D6F   . 5B            POP EBX
00431D70   . 8BE5          MOV ESP,EBP
00431D72   . 5D            POP EBP
00431D73   . C2 0400       RETN 4                   <<-- Fin du CALL
00431D76   . 00            DB 00
```

On se concentrera sur la boucle au milieu du CALL, qui est celle qui génère le Sérial. Voici le détail de la boucle de calcule du Sérial :

C'est maintenant la partie la plus délicate, comprendre l'algorithme et dans ce cas, il vous faut des notions en Assembleur. Je vous conseille de lire mon cours ASM :

- http://www.kromCrack.com/pdf/Assembleur.pdf

Il est indispensable de comprendre les instructions pour pouvoir les reproduirent. Alors on va débugger le programme ligne par ligne et regarder ce qu'il fait a chaque fois.

Redémarrez avec CTRL + F2 ->> Oui, puis F9. enregistrez vous avec Krom / 1234 puis "Ok". OllyDBG Break en 00435855, entrez dans le CALL avec F7, puis F9. Là, Olly break en 00431BCD (Le CALL de calcule du Sérial) entrez dans ce CALL avec F7 et faites une série de F8 jusqu'à arriver au MOV EBX,1 en 00431AC4. C'est maintenant qu'il faudra de la concentration et de la patience car nous allons le faire ensemble et lignes par lignes ce qui prendra surement un moment ...

Boucle N° 1

- Ligne 00431AC4, EBX est mis à 1.
- Ligne 00431AC9, ECX vaut maintenant 1 à cause du MOV ECX,EBX.
- Ligne 00431ACB, ECX vaut maintenant 0 à cause du DEC ECX.
- Ligne 00431ACC, AND ECX, 8000000F compare ECX (00000000) et 8000000F en binaire, et met 1 si les 2 opérandes sont 1, sinon met 0

Registre	Hexadécimal	Décimal	Binaire
ECX	00000000	0	00000000000000000000000000000000
	8000000F	2'147'483'663	10000000000000000000000000001111
ECX	00000000	0	00000000000000000000000000000000

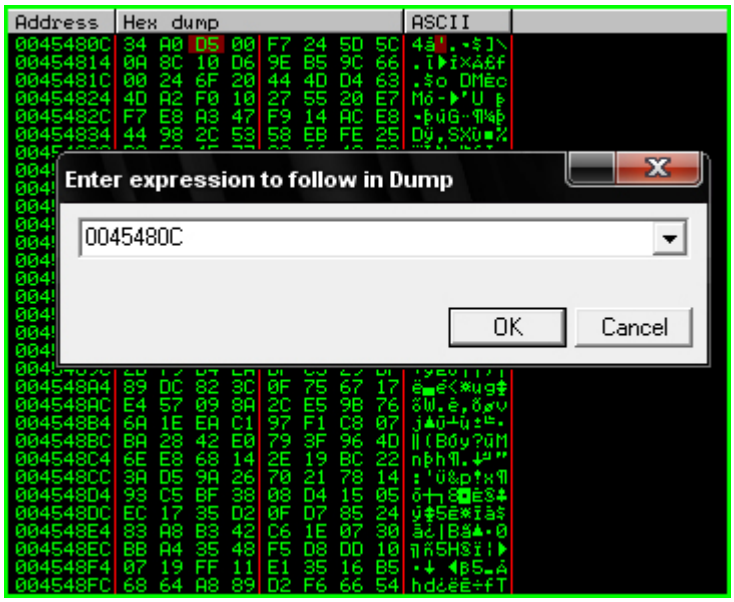
- Ligne 00431AD2, JNS SHORT HexDecCh.00431AD9 va sauter ver le LEA EAX,DWORD PTR DS:[EBX+EDI] en 00431AD9.
- Ligne 00431AD9, place dans EAX le contenu de [EBX+EDI], donc place 2 dans EAX.

Registre	Hexadécimal	Décimal	Binaire
EBX	00000001	1	01
EDI	80000001	1	01
EAX	00000002	2	10

- Ligne 00431ADC, on met ECX sur la pile (ECX vaut toujours 0) avec l'instruction PUSH ECX.
- Ligne 00431ADD, on met 112 dans ECX, d'ou le MOV ECX,112.
- Ligne 00431AE2, convertit EAX en un quadruple mot dans <EDX:EAX>.
- Ligne 00431AE3, IDIV ECX, IDIV effectue une division non signée de EAX par ECX. En divisant 2 (EAX) par 112 (ECX) le résultat sera mis dans EAX (0).

Registre	Hexadécimal	Décimal	Binaire
EAX	00000002	2	000000010
ECX	00000112	274	100010010
EAX	00000000	0	000000000

- Ligne 00431AE5, 00000000 est mis dans ECX à cause du POP ECX.
- Ligne 00431AE6, Voila une ligne intéressante ! Elle a pour but de mettre dans AL le contenu de [EDX+45480C]. Pour voir ce qu'il y a à cette adresse faites CTRL + G dans la fenêtre de Dump en bas à gauche et entrez-y 0045480C.



- On voit que 45480C vaut 34 mais comme EDX vaut 2 on regarde en fait sur l'adresse 0045480E (0045480C + 2 = 0045480E), on a donc AL qui vaut D5.
- Ligne 00431AEC, met [EBP-4] dans EDX, [EBP-4] est l'emplacement mémoire où est stocké le nom entré "Krom".
- Ligne 00431AEF, met dans DL, [EDX+EBX-1], EDX vaut " Krom ", EBX vaut 1 -1 donc met 4B (K) dans DL.
- Ligne 00431AF3, voici surement une des lignes les plus importantes, car elle compare AL et DL en binaire, et met dans AL (EAX) 1 si les 2 opérandes sont différentes sinon met 0.

Registre	Hexadécimal	Décimal	Binaire
AL	000000D5	213	11010101
DL	0000004B	75	01001011
EAX	0000009E	158	10011110

- Ligne 00431AF5, on a un AND EAX,0FF qui compare EAX (9E) à 0FF et met 1 uniquement quand les 2 opérandes sont 1 sinon met 0.

Registre	Hexadécimal	Décimal	Binaire
EAX	0000009E	158	10011110
	000000FF	255	11111111
EAX	0000009E	158	10011110

- Ligne 00431AFA, EAX est multiplié par CL. SHL est une multiplication par puissances de 2 mais comme CL vaut 0 2^0 = 1 donc 1 x 9E = 9E.
- Ligne 00431AFC, ajoute le contenu de EAX (9E) dans [EBP-8] et c'est à cette adresse que le Sérial sera stocké.
- Ligne 00431AFF, INC EBX = EBX + 1 donc EBX vaut 2.
- Ligne 00431B00, DEC ESI = ESI - 1 donc ESI vaut 3.
- Ligne 00431B01, et pour finir cette boucle, le JNZ SHORT HexDecCh.00431AC9 en 00431B01 saute en 00431AC9 pour recommence la boucle.

Boucle N° 2

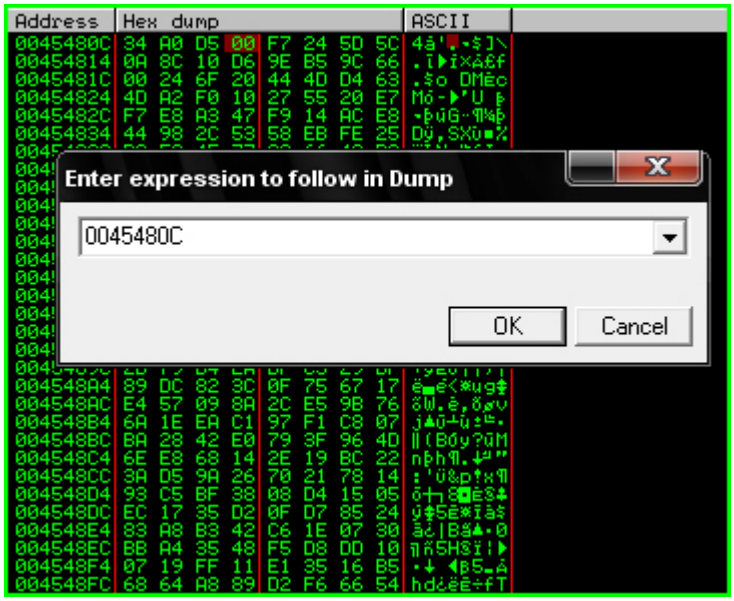
- Ligne 00431AC9, EBX (2) est mis dans ECX (0) on a donc EBX = 2 et ECX = 2.
- Ligne 00431ACB, on a un DEC ECX, ECX = 1.
- Ligne 00431ACC, on a ici un AND ECX, 8000000F, on compare ECX (1) et 8000000F et met 1 si les 2 opérandes sont 1 sinon met 0 et stock le résultat dans la première opérande.

Registre	Hexadécimal	Décimal	Binaire
ECX	00000000	0	00000000000000000000000000000000
	8000000F	2'147'483'663	10000000000000000000000000001111
ECX	00000000	0	00000000000000000000000000000000

- Ligne 00431AD2, JNS SHORT HexDecCh.00431AD9 va sauter ver le LEA EAX,DWORD PTR DS:[EBX+EDI] en 00431AD9.
- Ligne 00431AD9, arrivée du JNS SHORT HexDecCh.00431AD9 en 00431AD2. Cette instruction LEA EAX,DWORD PTR DS:[EBX+EDI] est comme un MOV XXX, XXX, et comme EAX vaut 9E, EBX = 2 et EDI = 1 on aura EAX qui vaudra 3.
- Ligne 00431ADC, met ECX (00000001) sur la pile.
- Ligne 00431ADD, met 112 dans ECX.
- Ligne 00431AE2, convertit EAX en un quadruple mot dans <EDX:EAX>.
- Ligne 00431AE3, IDIV ECX à pour but de diviser EAX par ECX donc 3:112 = 0.

Registre	Hexadécimal	Décimal	Binaire
EAX	00000003	3	000000011
ECX	00000112	274	100010010
EAX	00000000	0	000000000

- Ligne 00431AE5, récupère la valeur au haut de la pile et la met dans ECX (00000001) ECX = 1.
- Ligne 00431AE6, cette ligne a pour but de mettre dans AL le contenu de [EDX+45480C]. Pour voir ce qu'il y a à cette adresse faites CTRL + G dans la fenêtre de Dump en bas à gauche et entrez-y 0045480C.



On voit que 45480C vaut 34 mais comme EDX vaut 3 on regarde en fait sur l'adresse 0045480F (0045480C + 3 = 0045480F), on a donc AL qui vaut 00.

- Ligne 00431AEC, met [EBP-4] dans EDX, [EBP-4] est l'emplacement mémoire où est stocké le nom entré "Krom".
- Ligne 00431AEF, met dans DL, [EDX+EBX-1], EDX vaut "Krom", EBX vaut 2 -1 donc met 72 (r) dans DL (à chaque boucle, EBX vaut 1 de plus pour prendre une lettre différente à chaque passages).
- Ligne 00431AF3, voici surement une des lignes les plus importantes, car elle compare AL et DL en binaire, et met dans AL (EAX) 1 si les 2 opérandes sont différentes sinon met 0.

Registre	Hexadécimal	Décimal	Binaire
AL	00000000	0	0000000
DL	00000072	114	1110010
EAX	00000072	114	1110010

- Ligne 00431AF5, on a un AND EAX,0FF qui compare EAX (72) à 0FF et met 1 uniquement quand les 2 opérandes sont 1 sinon met 0.

Registre	Hexadécimal	Décimal	Binaire
EAX	00000072	114	01110010
	000000FF	255	11111111
EAX	00000072	114	01110010

- Ligne 00431AFA, à cette ligne, on multiplie EAX (72) par CL (1) en puissance de ^2.
  Voici le calcul : 72 x 2^1 = E4. donc on a un EAX = E4.
- Ligne 00431AFC, c'est à cette ligne que le Sérial est stocké, on met EAX (E4) dans [EBP-8]. ([EBP-8] contient déjà 9E, il nous suffira de l'additionner à notre E4 pour avoir [EBP-8] = 182).
- Ligne 00431AFF, ici on incrémente EBX de 1 donc EBX = 3.
- Ligne 00431B00, ici on décrémente ESI de 1 donc ESI = 2.
- Ligne 00431B01, et pour finir cette boucle, le JNZ SHORT HexDecCh.00431AC9 en 00431B01 saute en 00431AC9 pour recommence la boucle.

Boucle N° 3

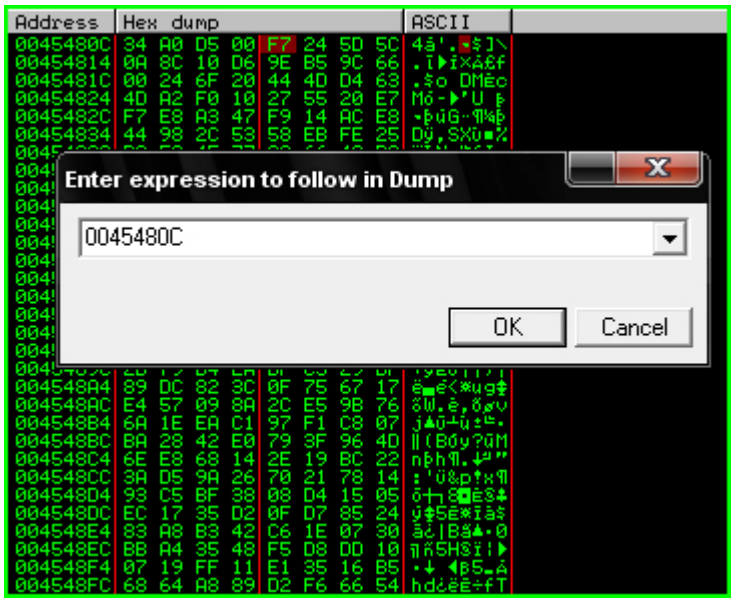
- Ligne 00431AC9, MOV ECX,EBX (EBX = 3 et ECX = 1), ECX va passer à 3.
- Ligne 00431ACB, DEC ECX, ECX = 2.
- Ligne 00431ACC, AND ECX,8000000F, compare ECX (2) à 8000000F en binaire, et met 1 si les 2 opérandes sont 1 sinon met 0. (Le résultat est mis dans la première opérande).

Registre	Hexadécimal	Décimal	Binaire
ECX	00000002	2	00000000000000000000000000000010
	8000000F	2'147'483'663	10000000000000000000000000001111
ECX	00000002	2	00000000000000000000000000000010

- Ligne 00431AD2, le JNS SHORT HexDecCh.00431AD9 en 00431AD2 saut vers le LEA EAX,DWORD PTR DS:[EBX+EDI] en 00431AD9.
- Ligne 00431AD9, LEA EAX,DWORD PTR DS:[EBX+EDI], place dans EAX, le contenu de [EBX+EDI] donc place dans EAX, 3 (EBX) + 1 (EDI), EAX = 4.
- Ligne 00431ADC, PUSH ECX, place ECX (2) sur la pile.
- Ligne 00431ADD, MOV ECX,112, met 112 dans ECX.
- Ligne 00431AE2, CDQ, convertit EAX en un quadruple mot dans <EDX:EAX>.
- Ligne 00431AE3, IDIV ECX à pour but de diviser EAX par ECX donc 4:112 = 0.

Registre	Hexadécimal	Décimal	Binaire
EAX	00000004	4	000000100
ECX	00000112	274	100010010
EAX	00000000	0	000000000

- Ligne 00431AE5, POP ECX, récupère la valeur qui est en haut de la pile et la met dans ECX = 2.
- Ligne 00431AE6, cette ligne a pour but de mettre dans AL le contenu de [EDX+45480C]. Pour voir ce qu'il y a à cette adresse faites CTRL + G dans la fenêtre de Dump en bas à gauche et entrez-y 0045480C.



On voit que 45480C vaut 34 mais comme EDX vaut 4 on regarde en fait sur l'adresse 00454810 (0045480C + 4 = 00454810), on a donc AL qui vaut F7.

- Ligne 00431AEC, MOV EDX,DWORD PTR SS:[EBP-4], met dans EDX le contenu de [EBP-4] ([EBP-4] = Nom entré = Krom).
- Ligne 00431AEF, met dans DL, [EDX+EBX-1], EDX vaut " Krom ", EBX vaut 3 -1 donc met 6F (o) dans DL. (à chaque boucle, EBX vaut 1 de plus pour prendre une lettre différente à chaque passages).
- Ligne 00431AF3, voici surement une des lignes les plus importantes, car elle compare AL et DL en binaire, et met dans AL (EAX) 1 si les 2 opérandes sont différentes sinon met 0.

Registre	Hexadécimal	Décimal	Binaire
AL	000000F7	274	11110111
DL	0000006F	111	01101111
EAX	00000098	152	10011000

- Ligne 00431AF5, on a un AND EAX,0FF qui compare EAX (98) à 0FF et met 1 uniquement quand les 2 opérandes sont 1 sinon met 0.

Registre	Hexadécimal	Décimal	Binaire
EAX	00000098	152	10011000
	000000FF	255	11111111
EAX	00000098	152	10011000

- Ligne 00431AFA, à cette ligne, on multiplie EAX (98) par CL (2) en puissance de ^2.
  Voici le calcul : 98 x 2^2 = 260. donc on a un EAX = 260.
- Ligne 00431AFC, c'est à cette ligne que le Sérial est stocké, on met EAX (260) dans [EBP-8]. ([EBP-8] contient déjà 182, il nous suffira de l'additionner à notre 260 pour avoir [EBP-8] = 3E2).
- Ligne 00431AFF, ici on incrémente EBX de 1 donc EBX = 4.
- Ligne 00431B00, ici on décrémente ESI de 1 donc ESI = 1.
- Ligne 00431B01, et pour finir cette boucle, le JNZ SHORT HexDecCh.00431AC9 en 00431B01 saute en 00431AC9 pour recommencer la boucle.

Boucle N° 4

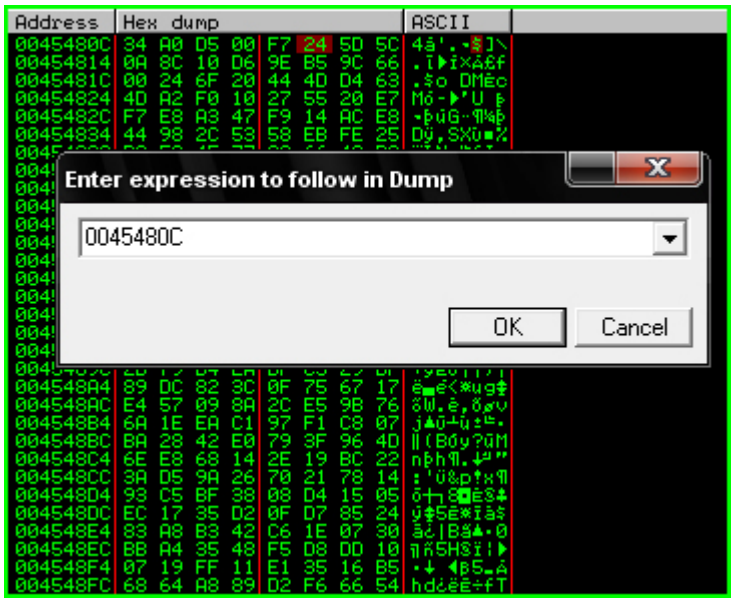
- Ligne 00431AC9, met EBX dans ECX donc ECX = 4 et EBX = 4.
- Ligne 00431ACB, DEC ECX, ECX = 3.
- Ligne 00431ACC, on a ici un AND ECX, 8000000F, on compare ECX (3) à 8000000F et met 1 si les 2 opérandes sont 1 sinon met 0 et stock le résultat dans la première opérande.

Registre	Hexadécimal	Décimal	Binaire
ECX	00000003	3	00000000000000000000000000000011
	8000000F	2'147'483’663	10000000000000000000000000001111
ECX	00000003	3	00000000000000000000000000000011

- Ligne 00431AD2, le JNS SHORT HexDecCh.00431AD9 en 00431AD2 saut vers le LEA EAX,DWORD PTR DS:[EBX+EDI] en 00431AD9.
- Ligne 00431AD9, place dans EAX, le contenu de [EBX+EDI] donc EAX = 5.
- Ligne 00431ADC, met ECX dur la pile (ECX = 3).
- Ligne 00431ADD, MOV ECX,112, met 112 dans ECX.
- Ligne 00431AE2, CDQ à pour but de convertir EAX en un quadruple mot dans <EDX:EAX>.
- Ligne 00431AE3, IDIV ECX à pour but de diviser EAX par ECX donc 5:112 = 0.

Registre	Hexadécimal	Décimal	Binaire
EAX	00000005	5	000000101
ECX	00000112	274	100010010
EAX	00000000	0	000000000

- Ligne 00431AE5, récupère la valeur qui est en haut de la pile et la met dans ECX (ECX = 3).
- Ligne 00431AE6, cette ligne a pour but de mettre dans AL le contenu de [EDX+45480C]. Pour voir ce qu'il y a à cette adresse faites CTRL + G dans la fenêtre de Dump en bas à gauche et entrez-y 0045480C.



On voit que 45480C vaut 34 mais comme EDX vaut 5 on regarde en fait sur l'adresse 00454811 (0045480C + 5 = 00454811), on a donc AL qui vaut 24.

- Ligne 00431AEC, met [EBP-4] dans EDX, [EBP-4] est l'emplacement mémoire où est stocké le nom entré "Krom".
- Ligne 00431AEF, met dans DL, [EDX+EBX-1], EDX vaut "Krom", EBX vaut 4 -1 donc met 6D (m) dans DL (à chaque boucle, EBX vaut 1 de plus pour prendre une lettre différente à chaque passages).
- Ligne 00431AF3, voici surement une des lignes les plus importantes, car elle compare AL et DL en binaire, et met dans AL (EAX) 1 si les 2 opérandes sont différentes sinon met 0.

Registre	Hexadécimal	Décimal	Binaire
AL	00000024	36	0100100
DL	0000006D	109	1101101
EAX	00000049	73	1001001

- Ligne 00431AF5, on a un AND EAX,0FF qui compare EAX (49) à 0FF et met 1 uniquement quand les 2 opérandes sont 1 sinon met 0.

Registre	Hexadécimal	Décimal	Binaire
EAX	00000049	73	01001001
	000000FF	255	11111111
EAX	00000049	73	01001001

- Ligne 00431AFA, à cette ligne, on multiplie EAX (49) par CL (3) en puissance de ^2.
  Voici le calcul : 49 x 2^3 = 248. donc on a un EAX = 248.
- Ligne 00431AFC, c'est à cette ligne que le Sérial est stocké, on met EAX (248) dans [EBP-8]. ([EBP-8] contient déjà 3E2, il nous suffira de l'additionner à notre 248 pour avoir [EBP-8] = 62A), 62A enfin ! car ce fameux 62A donne 1578 en décimal, c'est donc notre Sérial !
- Ligne 00431AFF, incrémente EBX de 1, EBX vaut alors 5.
- Ligne 00431B00, décrémente ESI de 1, ESI vaut alors 0.
- Ligne 00431B01, comme ESI vaut 0, le JNZ SHORT HexDecCh.00431AC9 en 00431B01 ne saute pas en 00431AC9 car ESI vaut 0 et continue simplement en 00431B03.

Ca a été long mais on y est arrivé, mais l'algorithme qui est dans ces boucle n'est pas vraiment lisible donc en voici un petit résumé :

1) Met à chaque boucle :

   D5, 00, F7, 24, 5D, 5C, 0A, 8C, 10, D6, 9E, B5, 9C, 66, 00, 24, 6F, 20, 44, 4D, D4, 63, 4D, A2, F0,
   10, 27, 55, 20, E7, F7, E8, A3, 47, F9, 14, AC, E8, 44, 98, 2C, 53, 58, EB, FE, 25, B0, FC, 4E, 77, CC,
   66, 49, DA, 7C, C6, 8E, D1, 5C, 22, D9, C5, C8, 90, E4, BF, B4, 79, 87, 7E, 3E, 3D, 76, A3, D6, 08, 43, 1A
   dans AL.

2) Met le nom entré dans EDX.

3) Met a chaque passage une lettre du nom entré dans DL, 1. 4B (K) 2. 72 (r) 3. 67 (o) 4. 6D (m).

4) XOR AL,DL compare AL et DL en binaire, met 1 si différent et 0 si égale et place le résultat dans AL.

Registre	Boucle 1	Boucle 2	Boucle 3	Boucle 4
AL	D5	00	F7	24
DL	4B	70	6F	6D

5) Multiplie EAX par CL en puissance de ^2 (2^0, 2^1, 2^2, 2^3 ...).

6) Met le résultat dans un emplacement mémoire (dans ce cas c'est [EBP-8]).

7) Et on recommence la boucle.

La création du Keygen pour Hexa.exe en DOS
se trouve dans le Cours N° 5.

- **http://www.KromCrack.com/cours5.php**

Et la création du Keygen pour Hexa.exe en GUI
se trouve dans le Cours N° 6.

- **http://www.KromCrack.com/cours6.php**

J'espère que ce cours a été clair ;)

Si vous avez rencontré une erreur ou que quelque chose ne marche pas, vous pouvez m'envoyer un mail à **Admin@KromCrack.com** ou en parler sur le forum :

- **http://www.KromCrack.com/forum/**