Cours N° 1

Ce cours vous expliquera les bases du Cracking ainsi que les outils requis pour Cracker.

Le Cracking c'est quoi ?

Le Cracking est l'art de modifier un programme pour changer son comportement, il y a 3 grandes techniques :

- Le Patching : C'est une technique de modification du programme. On peut le modifier de tel sorte que le programme nous enregistre avec n'importe quel code, ou qu'un jeu démarre sans cd par exemple. Ce type de Crack est relativement simple car en général, car il suffit de modifier un saut ou d'inverser une valeur pour que le logiciel soit Cracké sans pour autant avoir besoin de vraiment comprendre le code dans les moindres détails. Mais cette technique s'utilise le moins possible car Patcher un programme n'est jamais vraiment propre.

- Le Sérial Fishing : Cela signifie littéralement "Pêche au Sérial". On essaie de trouver un code valide dans le programme en rapport d'un nom entré. Cela se complique un peu plus car on doit retrouver un Sérial en Hexadécimal (Base 16) dans un registre ou une adresse mémoire.

- Le Keygenning : Cette technique consiste à retrouver l'algorithme de génération du Sérial dans tout le code du programme. Quand on entre un code, le programme " Crée " un bon code et le compare au Sérial entré, s'il y a des différences, le code est faux et s'il est égal au code généré par le programme c'est que le code est juste. Le Keygenning consiste à retrouver la petite partie du code qui crée le code et ensuite de la reproduire pour créer un " Générateur de Codes valides ". Elle est assez compliquée car elle requiert de nombreuses connaissances en ASM en aussi beaucoup de logique. La difficulté principale est de comprendre l'algorithme et ensuite de le reproduire dans n'importe quel langage de programmation. Cette technique est vraiment le fin du fin pour un Cracker car elle a l'avantage par rapport au Sérial Fishing de générer un code valide à partir de n'importe quel nom et permet donc un enregistrement pour tous.

De quels outils ai-je besoin ?

Les outils de "Base" sont :

- Un Débuggeur / Désassembleur (permet de lire le code du programme en assembleur).
- Un Editeur Hexadécimal (permet de modifier des octets en hexadécimal).

Mais il y a aussi tous les outils qui aident pour des taches très précises

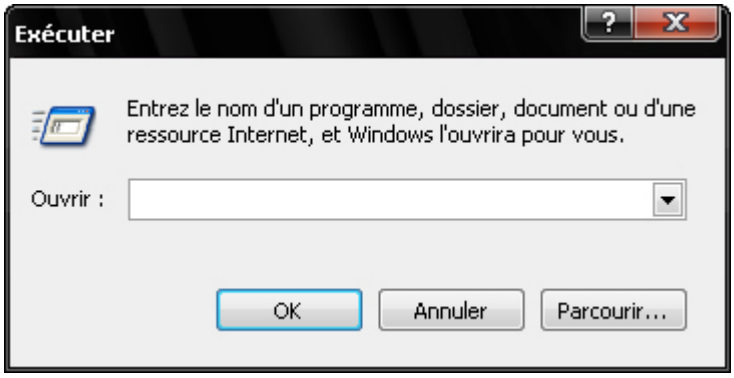
- HexDecCharEditor : Un éditeur Hexadécimal très complet.
- OllyDBG : Surement le meilleur Débuggeur / Désassembleur.
- Peid 0.93 : Un petit logiciel qui permet d'analyser un logiciel pour savoir si il est packé et / ou crypté.
- ProcDump : Un programme qui vous permet de faire une sauvegarde d'un logiciel en exécution.
- Regmon : Un très bon logiciel qui permet de surveiller la base de registre.
- Stripper : Permet de faire un Dump d'un exécutable
- Stud PE 1.7 : Un éditeur de ressource sympa
- UPX : Un très bon logiciel de décompression UPX

Quelles sont les connaissances requises pour Cracker ?

Les connaissances requises ne sont pas très élevés donc pas besoin d'être informaticien pour Cracker, mais il vous faudra quand même en connaître un minimum comme :

Les extensions de fichier exécutables courants et leurs dérivés (.exe, .dll, .ini ...) et savoir avec quel programme peut on les lire et / ou les modifier.

Les principales commandes à entrer dans la boîte de dialogue "Exécuter" :



Comme " Regedit " (accès à la base de registre de Windows, c'est l'endroit où sont stocké toutes les clés d'enregistrement et autres), " Calc " (la calculatrice Windows, nous l'utiliserons beaucoup pour les conversions Hexadécimales -> Décimales / Binaire), " Notepad " (éditeur de texte très pratique pour noter des informations et créer des fichiers pour toutes les extensions, .reg par exemple).

Il faudra aussi vous habituer aux expressions et termes courants, qui sont presque toujours en Anglais :) mais rassurez-vous si vous n'êtes pas anglophone, les mots seront du style de : Jump, If, Compare, Not, And, Or, Call donc pas trop de soucis de ce point de vue là, car vous vous y habituerais très vite, vous verrez :)

Et connaître bien sur aussi les bases de Windows, changer l'extension d'un fichier, copie / déplacement, savoir dans quel dossier le programme est installé ...

Et bien sur une bonne tête et aussi Beaucoup de patience ^^

Conversions Décimale à Hexadécimale

A quoi sert l'Hexadécimale ? Eh bien cela sert surtout en informatique pour afficher de grands nombres. Par exemple 1'265'789 en Décimal ne sera que 13 507D en Hexa. L'Hexadécimal est en base 16, ce qui veut dire qu'elle se compose de chiffres et de lettres allant de 1 à 15 (de 1 à F en Hexa). Il n'y a pas besoin de connaitre par cœur les conversions car tout est fait Automatiquement via " Calc.exe ", donc sachez juste les grandes lignes et ça suffira. Voici un petit tableau de conversion entre Décimale, Hexadécimale et binaire.

Décimale	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	...
Hexadécimale	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	...
Binaire	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111	...

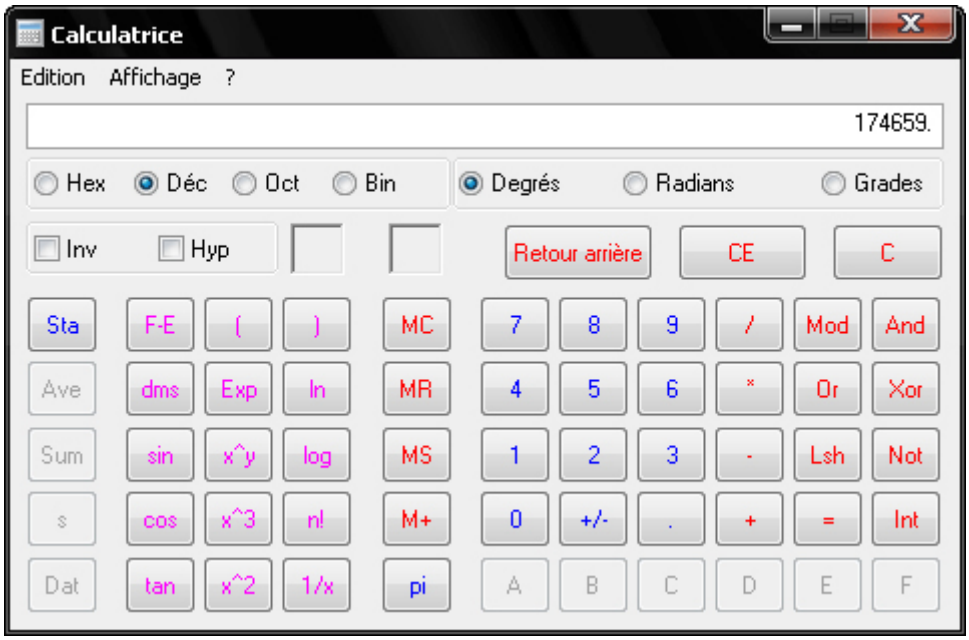
Et vous allez voir plus tard que l'Hexadécimale nous sert à modifier le programme, car chaque instruction d'un programme a son pareil en Hexa. Pour faire de rapides Conversions allez dans :

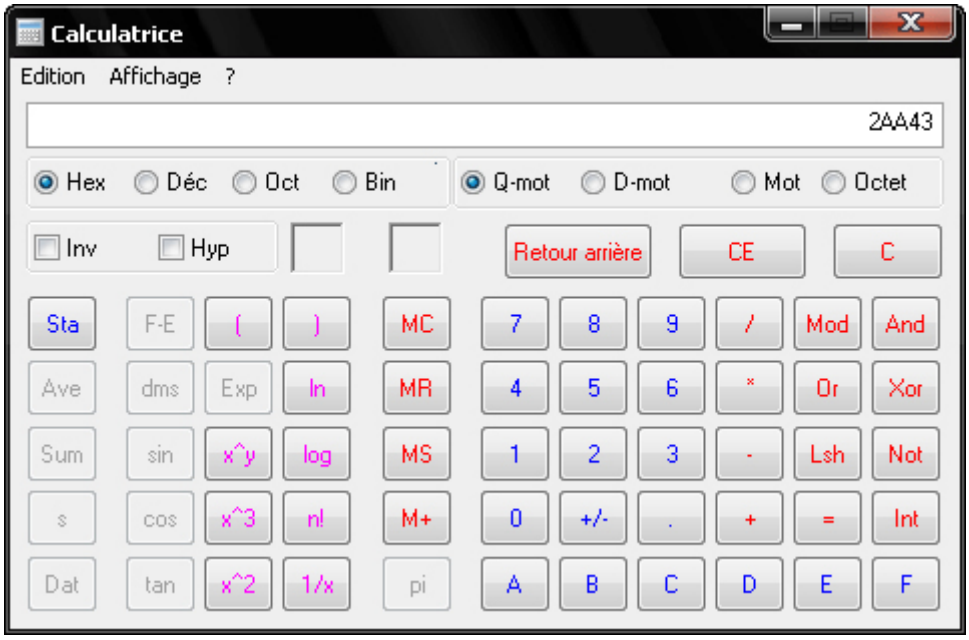
- Démarrer ->> Exécuter ->> Calc ->> Ok.

Quand la Calculatrice Windows s'ouvre mettez le en scientifique, pour cela aller dans :

- " Affichage " ->> Scientifique.

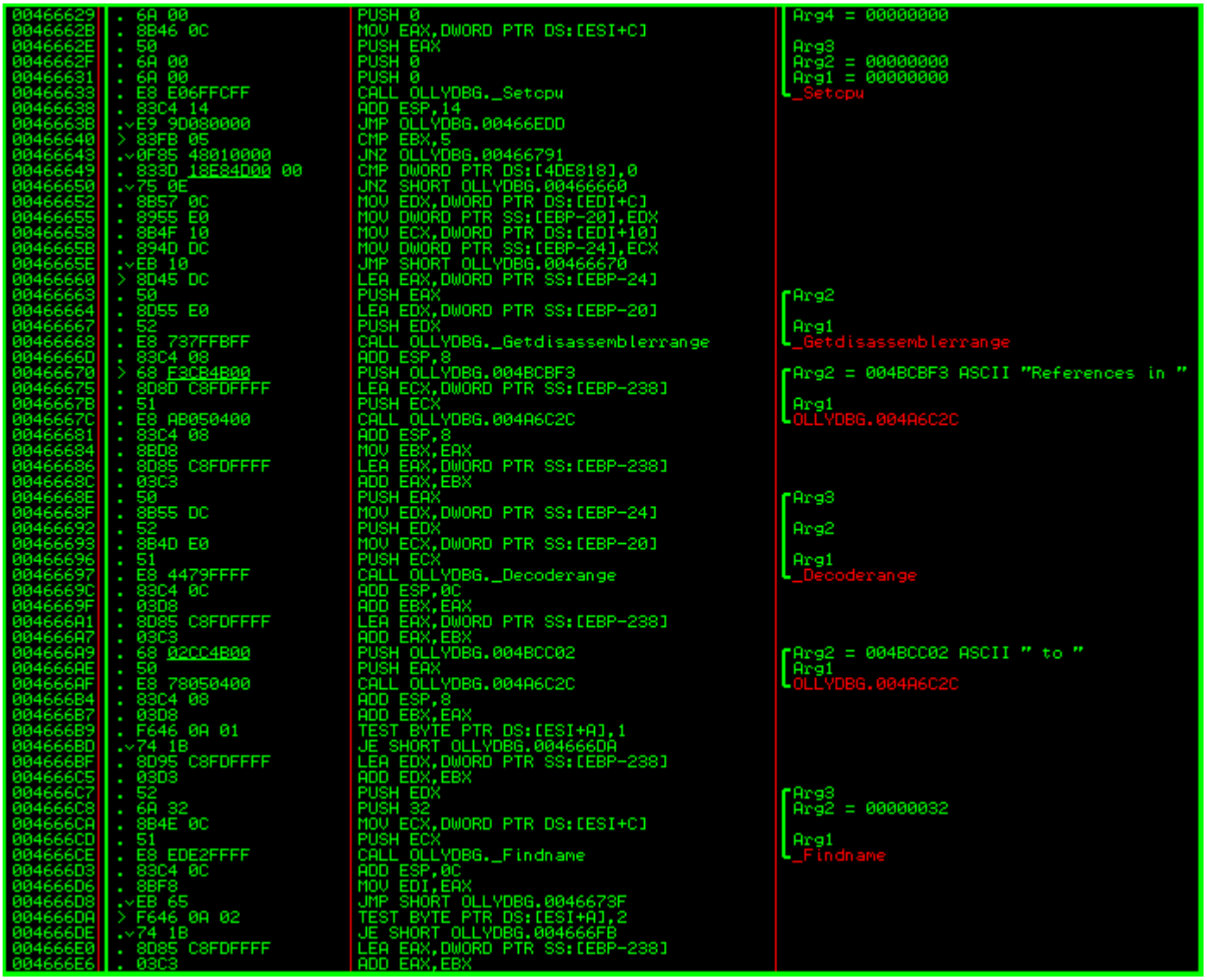
Eh maintenant rien de plus simple pour convertir : Mettez n'importe quel valeur en Décimal et quand vous aller cliquer sur le bouton " Hex ", les valeurs entrées seront cette fois affichés en Hexa.





Qu’est-ce que l'Assembleur ?

L'Assembleur (ASM) est un langage de programmation plutôt compliqué, mais vous n'aurez pas besoin de le connaître sur le bout des doigts. Quand on crée on programme dans n'importe quel langage de programmation, on compile ce que l'on appelle un " code source " qui est un fichier que l'on peut très bien modifier avec un éditeur de texte classique. Pour le " Convertir " en .exe, on fait une étape que l'on appelle " Compiler ", chaque langages de programmation à son compilateur, par exemple pour le C/C++ on peut utiliser Dev-C++ ou Code::Block et pour l'Assembleur par exemple c'est MASM. Une fois compiler on obtient un .exe C'est justement pour pouvoir lire le code source que l'on Désassemble un programme. Un programme Désassemblé su lit toujours en Assembleur car si on prend trois mêmes programmes fait en trois langages différents et qu'on les Désassemble, on verra à peu de choses près la même chose en Assembleur. Voici un exemple d'un programme désassemblé dans OllyDBG (Programme qui permet de désassembler n'importe quel logiciel écrit dans n'importe quel langage) :



Vous y voyez pleins de nombres bizarres et plein d'inscriptions suspectes mais ne vous découragez pas ! Après quelques explications ça ira déjà mieux ...

Alors la première colonne n'est que le numéro des lignes du programme, dans la suite de mes cours, je dirais d'aller à la ligne 0042EB58 ou 004458D9 par exemple. Ce n'est qu'un numéro de ligne alors pas besoin de se prendre la tête pour ça, il n'y a rien de spécial à savoir là-dessus. Alors prenons un exemple de ligne :

- 00466650 |. 75 0E JNZ SHORT OLLYDBG.00466660

La première information est le numéro de la ligne donc ici 00466650. La deuxième est l'instruction en Assembleur traduit en Hexadécimal. 75 se rapporte au JNZ et 0E se rapporte au numéro de la ligne sur laquelle le JNZ va sauter. La troisième information est l'instruction en Assembleur. Pour faire clair, le JNZ SHORT OLLYDBG.00466660 est une instruction en Assembleur qui veut dire de Sauter à la ligne 00466660 ->> JNZ = Jump if Not égual to Zéro (Saute si ce n'est pas égal à 0), donc si ce n'est pas égal à 0, saute à la ligne 0466660, sinon continue à la ligne 00466652 (pourquoi continuer à la ligne 00466652 et pas 00466651 ? Tous simplement car la ligne 00566651 n'existe pas du moins en tant que numéro de ligne (elle peut exister comme numéro dans la pile ou du Dump))

Pour comprendre l'Assembleur, il n'y a malheureusement pas de miracle, il faut l'apprendre ! Pas besoin non plus de savoir toutes les instructions par cœur mais sachez au moins :

- Jump // Saut vers un endroit du programme.
- CMP // Un comparaison par ex CMP EAX, EDX (compare EAX à EDX).
- PUSH // Met le registre concerné sur la pile (PUSH ECX = met ECX sur la pile).
- POP // Récupère la valeur qui se trouve en haut de la pile (POP EAX) met la valeur en haut de la pile dans EAX.
- CALL // Sous-routines (de vérifications par exemple).
- MOV // Instruction de déplacement de valeurs (MOV AL,CL).
- ADD // Instruction d'ajout de valeur.
- LEA // Instruction d'ajout de valeur.

Et les différents registres :

- EAX
- ECX
- EDX
- EBX
- ESP
- EBP
- ESI
- EDI

Maintenant que vous avez vu les rouages du Cracking, il ne vous reste plus qu'à apprendre l'Assembleur dans mon Cours ASM :) :

- **http://www.KromCrack.com/Pdf/Assembleur.pdf**

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/**