Cours de Cracking

(1ième Partie)

Mon objectif : tout vous expliquer de A à Z pour que vous puissiez comprendre comment fonctionne le cracking. Il va falloir que vous lisiez bien tout attentivement de façon à bien comprendre les cours. Chaque cours est la suite du précdent et de niveau croissant en difficulté (rassurez vous on va faire dans la simplicité en expliquant au maximum les points pouvant paraître obscurs).

1/ Qu'est ce qu'un crack ?

Un crack, est un petit programme (quelques ko en général) qui va modifier certains octets d'une application à certains emplacements du fichier cible de façon transformer son comportement. Par exemple jouer à votre jeux préféré sans CD ou encore utiliser le programme en tant qu'utilisateur enregistré.

Pour faire votre premier crack, on a choisi un programme super facile à cracker pour vous montrer un truc super classique, le saut conditionnel ;))

Comme son nom l'indique, un saut conditionnel est un endroit dans le programme où une condition (on parle encore de test) est vérifiée. Si cette condition n'est pas remplie par exemple lors la tentative d'enregistrement de **Start Clean v1.2** le programme sautera vers une partie du code qui affichera un message d'erreur sinon il continuera normalement son exécution pour afficher un message de succès (pour l'enregistrement).

2/ Les logiciels pour le cracking.

Pour craquer un programme il vous faut d'abord choisir une cible (le programme lui-même à craquer). Typiquement il s'agit dans la majorité des cas une fichier exécutable (d'extension .exe) ; plus rarement un fichier dll. Ici la cible choisie qu'on a choisi pour sa simplicité et sa taille c'est **Start Clean v1.2** (il ne fait que 14 ko). Ce programme sert à supprimer les raccourcis orphelins dans le menu Démarrer (c'est à dire qui ne mènent nulle part).

Ensuite il vous faut un désassembleur qui comme son nom l'indique sert à désassembler un fichier (ici **Start Clean v1.2**) de façon à le rendre lisible et compréhensible. Pour ma part, j'utilise **W32Dasm 8.93** facile à trouver sur le net. Il existe aussi pour ceux qui ont un niveau un peu plus avancé en cracking le désassembleur **ollydbg 1.10**. Ces 2 désassembleurs intègrent un débugger : c'est un module qui permet

d'exécuter ligne de code par ligne de code le programme à craquer.

Par ailleurs il vous faut aussi ce qu'on appelle un éditeur hexadécimal. Comme son nom l'indique aussi, il sert à faire apparaître un fichier sous forme héxadécimale. C'est cet éditeur qui nous permet de modifier physiquement le comportement du programme et donc de le craquer en remplaçant une suite d'octets par une autre. Mon préferé c'est **WinHex 10.2** , mais il y a ausi HexWorks 3.X. N'importe quel éditeur hexadécimal fera l'affaire.

Enfin, il vous faudra dans tous les cas (enfin presque) un cerveau réveillé et en état de marche ;)

3/ Réglages préliminaires de W32dasm.

D'abord, lancez **Start Clean v1.2** (STARTCLN.EXE). Au démarrage, une fenêtre s'affiche pour pouvoir s'enregistrer. Mais le problème c'est que vous avez peut être pas que ça à faire que d'envoyer de l'argent à ce cher monsieur Fira El-Hasans, le concepteur du logiciel **Start Clean v1.2**, d'où la necessité de réaliser un crack ;) .

Donc, pour pouvoir voir le programme de façon lisible, il faut l'ouvrir avec notre désassembleur **W32Dasm 8.93** (W32DSM89.EXE).

Avant tout, faite une copie de STARTCLN.EXE et lancez W32DSM89.EXE.

Comme c'est le premier lancement de W32DSM89.EXE, il faut choisir la font par défaut que l'on va utiliser. Faites :
Disassembler -> Font ... -> Select Font.

Et pour sauvegarder la font choisie, faite :
Disassembler -> Font -> Save Default Font.

Allez cherchez ensuite la copie du fichier executable STARTCLN.EXE de la façon suivante :
Disassembler -> Open File to Disassemble ..

Double-cliquez sur la copie de STARTCLN.EXE

4/ Le listing de désassemblage.

Ben oui, ce que vous voyez maintenat à l'écran c'est du code en assembleur. Ce sont de petites instructions qui s'enchaînent pour former un tout, un programme.

L'assembleur est le language de programmation le plus bas. Quand vous programmez que se soit en C ou en Delphi, tout est traduit en assembleur pour que votre programme soit compréhensible par l' ordinateur,

car ne l'oublions pas l'assembleur, c'est le language machine par exellence. Ses avantages résident dans le fait qu'il est tout de suite compris par la machine, ainsi, un programme, même écrit en C sera exécuté moins rapidement que s'il avait été fait en assembleur (de quelques dixièmes de secondes, mais bon ...).

Ok, vous devez peut-être vous dire "c'est quoi ce charabia incompréhensible, je quitte ça et je retourne télécharger des cracks tous faits sur internet". Si c'est ce que vous vous dites, alors quittez ce tutorial, et vous serez obligé d'attendre sur le net que les cracks sortent, et surtout, vous ratterez le plus important, la satisfaction personnelle et le monde superbe qu'est l'assembleur.

Par contre si vous vous dites "ce truc, c'est bizarre mais j'ai envie d'appronfondir et de savoir comment on s'y prend pour cracker" ; alors la suite de ce tutorial est pour vous ;)).

Bon, ce qui apparaît à l'écran ça s'appelle un listing de désassemblage. C'est là dessus que l'ont va travailler.

Avant de poursuivre ayez à l'esprit que les instructions successives d'un programme s'exécutent toujours de bas en haut et de gauche à droite : c'est le sens naturel de lecture.

5/ Faisons un peu d'assembleur.

Revenons à W32DSM89.EXE.

-> Avant de lancer une recherche sur un mot, on va se placer directement à l'entrée du prog, car ce qu'il y a avant on s'en fiche un peu ;). Pour ca, faites: Goto -> Goto Code Start...

-> Lancez ensuite une recherche automatique sur le mot **"**name**"**; (ne rentrez pas les guïllemets), car il fait partie de la boîte de dialogue se chargeant de la saisie du code. On pourrait aussi rechercher "code". Pour faire cette recherche, faites la commande suivant dans Search -> Find Text..

Et là, vous allez forcement tomber sur quelque chose. Cependant, il se peut qu'il y ait plusieurs fois ce mot dans le programme... Donc, par mesure de prudence, faites Suivant. Vous verrez alors que vous arrivez à d'autres endroits du programme. Alors, que choisir ?

-> Revenez au debut du code (Goto -> Goto Code Start ou bien clic sur le bouton avec une lampe marqué Cd Start en dessous dans la barre de bouton de w32dasm).

-> Refaite une recherche et arrêtez vous à la seconde occurence de "name " (qui doit etre trouvée a la ligne 423 comme indiqué en bas de cette photo qui représente une partie de listing de désassemblage crée plus haut).

On se trouve alors à un endroit particulièrement intérressant. Mais alors pourquoi cet endroit est t-il si intéressant me direz-vous ?

Parce que si l'on observe bien cette petite partie (ce qu'il y a avant et après...), on remarque que l'indication *Reference To: ADVAPI32. RegCreateKey, Ord: 00C6h est affichée.

Cette indication de fonction indique que STARTCLN.EXE va inscrire un truc dans la base de registre de Windows, sans doute votre nom et code ***a condition que*** votre code soit bon :). En français pour ceux qui sont allergiques à l'anglais RegCreateKey veux dire créer une clef dans le registre (Reg pour Registry).

L'instruction que l'on recherche est un saut conditionnel. En assembleur, ça se traduit par :
JNE, JNZ... (jump if not equal (to zéro), jump if not zéro) qui sont des sauts appelés sauts de "non egalité"...
JE, JZ.... (jump if equal (to zéro), jump if zéro) qui sont des sauts qui s'effectuent en cas d'égalité"...

Egalité signifie ici "deux valeur identique". Lorsque que le programme va comparer votre code avec le vrai, il va indiquez s'il sont "egaux" ou non.

Si le saut est de forme JE, alors on dira que "le saut se fait **si** les codes sont égaux".
Si le saut est de la forme JNE, alors on dira que "le saut se fait **si** les codes ne sont pas egaux"...

Il existe un type de saut appelé *inconditionnels*, c'est a dire que le saut s'effectue sans conditions (on saute tout le temps).Ce saut s'ecrit avec une instruction **JMP** en assembleur, mais cela ne nous servira pas pour ce tutorial.

Donc, pour revenir a STARTCLN.EXE, il va falloir qu'on trouve tout les *sauts* qu'il y a dans le bout de code ci dessus, puis qu'on analyse où est-ce qu'ils nous font sauter.

On est chanceux, ici il n'y a qu'un seul saut qui est a l'adresse 004011EB, et qui est située juste apres un test (un TEST est l'instruction qui determine si une valeur est égale à zéro) et juste au dessus de la chaîne "name" trouvée.On a donc :

L'instruction assembleur jne 00401271 signifie : "Sauter à l'adresse 00401271 si la valeur testée ici eax n'est pas égale à zéro". Cette instruction est intéressante car il y a fort à parier que la valeur testée soit le numéro de série que vous avez rentré au pif. En effet au dessus on voit que le programme fait appel à une API (Application Interfcace Programming) nommée Kernel32 qui est un fichier sur votre disque dur (le

fichier kernel32.dll dans C:\windows\system32). C'est dans cette dll (Dynamic Link Librairie <=> bibliothèque de liens dynamiques) qu'est encodée la fonction de comparaison de chaînes nommée lstrcmpA (l' API Kernel32.lstrcmpA va comparer votre numéro de série encore appelé serial dans la terminologie anglo-saxone avec le vrai serial calculé par l'application).

Il faut bien comprendre que souvent, un numéro de série est considéré comme valide lorsque la valeur du test est égale a zéro. Autrement, dans le cas où le test n'est pas égal à zéro, alors ce numéro est reconnu comme incorrect. C'est un peu comme le courant electrique : 0 = ouvert et 1 = fermé. Compris ?

Donc, "sauter a l'adresse 00401271 si la valeur testée n'est pas egale a zéro" siginfie que si le code est mauvais, on va "sauter" tout ce qu'il y a entre l'adresse 004011EB (l'endroit d'où l'on saute) et l'adresse 00401271 (l'endroit où le saut nous amène).

Regardez sur le listing.Concrètement, le saut va passer au dessus de toutes la partie du programme qui inscrit notre nom et notre code dans la base de registre.

```
* Reference To: ADVAPI32.RegCreateKeyExA, Ord:00C6h
:00401218    FF1500924000            Call dword ptr [00409200]
...

* Reference To: ADVAPI32.RegSetValueExA, Ord:00ECh
:0040122C    8B35FC914000            mov esi, dword ptr [004091FC]
...

* Possible StringData Ref from Data Obj ->"Name"
:00401236    6838624000              push 00406238
...

* Possible StringData Ref from Data Obj ->"Code"
:00401250    6830624000              push 00406230
...

* Reference To: ADVAPI32.RegCloseKey, Ord:00C2h
:0040125D    FF15F0914000            Call dword ptr [004091F0]
```

Par contre, **si le code est bon, le saut ne s'effectue pas** et le programme va continuer son listing en passant sur ls lignes du dessus.

Là encore c'est une API qui est utilisée ADVAPI32.dll qui est utilisée
-> pour créer une clé dans le registre de windows on utilise la fonction RegCreateKeyExA).
-> pour assigner une valeur à cette clef on utilise la fonction RegSetValueExA.

Le mot Reg est une abréviation de Registry (registre en français. Le registre est la base de paramétrage du système windows, du matétriel et des applications installés sur votre machine; en d'autres termes c'est une

grosse base de données et le coeur des systèmes windows).

Cette étape est **importante** à comprendre et si vous ne l'avez pas comprise relisez la plus attentivement...

Donc, il va falloir que l'on modifie le saut jne 00401271.Dans ce cas de figure, il faut faire ce que l'on appelle "nopper" une valeur :)
Mais avant, un petit cour sur l'hexadécimal s'impose :))

6/ L'hexadécimal.

D'abord oubliez la façon dons vous avez appris à compter. Parce qu' en hexadécimal, on compte le la façon suivante :

Décimal	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Hexadécimal	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

Par exemple 15 en décimal s' ecrit F en hexadécimal. Mais pouquoi dit-on héxadécimal ? Tout simplement parce qu' après les neufs premier chiffres il y à six lettres, 6 d'où hexa, comme un héxagone a six cotés ;)
Sinon pour ce qui est des conversions et tout, nous verrons ça dans un autre tutorial ;)

7/ Conversion assembleur / hexadécimal.

Voici une petite liste qui vous indique la valeur hexadécimale de quelques instructions assembleur.

Assembleur	Hexadécimal
NOP	90
JNE	75 ou 0F85
JE	74 ou 0F84
JMP	EB ou E9
XOR	33
RET	C3

Il existe encore beaucoup d'autres instructions, mais qui sont peu intéressantes à notre stade ;)

Pour illustrer ce tableau prenons un exemple. Regardons le code hexadécimal 0F8580000000 se trouvant à la gauche de l'instruction jne 00401271 dans le listing à l'adresse 004011EB. C'est en fait la valeur traduite en hexadécimale de l'instruction assembleur jne 00401271. 0F85 c'est jne et 80000000 c'est la zone mémoire mémoire où l'on saute. Une zone mémoire, si vous ne savez pas, c'est un peut comme un petit carré de mémoire éléctronique qui va contenir une information. Par exemple si vous collez un post-it avec écrit dessus "achète du pain pour ce soir" sur un porte, et bien la porte c'est le bloc mémoire et le post-it l'information ;)

8/ Nopper une valeur.

Qu'est ce que ça veut dire "**Nopper une valeur**" ? Ca veut dire que l'on va remplacer la valeur d'un octet par la valeur 90 en hexadécimal, ce qui correspond à une instruction assembleur qui veut dire **ne fait rien** (Nop = **N**o **OP**eration). L'instruction nop en assembleur est un peu bizarre, puisqu'elle ne sert à rien. Sauf dans certains cas pour faire perdre du temps CPU (Central Processing Unit = micro-processeur) en encombrant ses entrées/sorties.

9/ Modifier l'exécutable STARTCLN.EXE.

La question qu'on se pose maintenant c'est Comment nopper le saut conditionnel en 004011EB ?

Il suffit d' ouvrir le fichier STARTCLN.EXE avec votre éditeur hexadécimal **WinHex 10.2** (commande fichier -> ouvrir -> STARTCLN.EXE), de lancer une recherche automatique sur la valeur 0F 85 80 00 00 00 (avec CTRL ALT F -> 0F8580000000 qui represente en assembleur l'instruction jne 00401271). Une fois vérifié que cette valeur est bien la seule occurence trouvée en faisant suivant avec F3, on écrit au milieu de la fenêtre 909090909090 (6 fois 90) puis on enregistre les modifications (n'oubliez pas de faire une sauvegarde du fichier d'origine au préalable, au cas où vous vous tromperiez...(on ne sait jamais)).

Remarque : message de pifoman : si votre éditeur héxadécimal vous dit qu'il ne peut enregistrer les modifications enlevez l'attribut lecture seule de **STARTCLN.EXE**. Si cela ne suffit pas fermez le désassembleur W32DSM89.EXE qui verrouille toujours un fichier en écriture quand celui-ci est ouvert dedans. Il empêche ainsi d'écrire sur ce fichier avec un autre logiciel comme notre éditeur hexadécimal winhex par exemple.

10/ Vérifications après les modifications.

Quand on craque un logiciel en modifiant un ou plusieurs octets avec l'éditeur hexadécimal il faut s'assurer que l'effet attendu est bien là : ici on cherche à s'enregistrer avec un numéro de série bidon. Pour cela lançons le programme ainsi modifié.

Quoi ? La boite de dialogue d'enregistrement est toujours présente ? Normal puisque le programme n'a pas encore enregistré votre code dans la base de registre de Windows. Il faut donc cliquer sur le bouton REGISTER et rentrer un nom par exemple le mien pifoman et un code bidon 123456 puis cliquer sur le bouton OK (c'est moi pifoman qui écrit ici).

Et là ca marche ! Car le saut jne n'existe plus, donc le prog continue tranquillement son listing en passant sur les fonctions ADVAPI32.RegCreateKeyExA et ADVAPI32.RegSetValueExA. Le programme est donc maintenat complet et illimité. Vous pourrez aussi observer qu'il a inscrit un code personnel qu'il s'est auto-généré dans la base de registre de Windows sous la clef HKEY_CURRENT_USER\Software\Start Clean \Configuration (on acède au registre de windows avec le menu démarrer -> exécuter -> regedit). Pourquoi cette clé ? Parceque comme on pouvait le constater au dessus de notre Kernel32.lstrcmpA au paragraphe 5 on avait en plus des valeurs name et Code la chaîne

Miracle ? Non, c'est ça la magie de l'assembleur ;)))

PS : je ne dis pas ça pour vous décourager, mais pour un cracker conséquent, un tel raisonnement, observations et crack compris, ce type de sécurité à retirer ne prend que deux à trois minutes et encore. Alors entraînez vous bien et vous verrez, c'est pas si compliqué que ça en à l'air ;)

11/ Quelques conseils et astuces.

-> Lire pleins de tutoriaux fait sur de petit progs, les comprendre et faire l'effort de tout lire jusqu'à la fin.
-> Etre à l'aise avec l'anglais technique (avoir un dictionnaire comme ultralingua dans un coin de l'écran).
-> Ne pas se prendre la tête et laisser son esprit logique tourner seul, ça donne plus d'endurance.
-> Faire une pause de 5 minutes minimum toutes les heures, quand on commence à cracker, c'est mieux.
-> Faire ses devoirs avant ;) , car craquer, surtout pour la première fois prend du temps.
-> S'acheter un petit bouquin sur l'assembleur.
. Pour moi la réference, c'est L' assembleur de Bernard Fabrot, éditions Marabout, environ 50 francs.
. Pour moi pifoman ma référence L'assembleur une découverte pas à pas aux éditions Marabout de Philippe Mercier (57.95 F)
en français, imprimé le 26/02/1998 et bien expliqué [ISBN 2-501-01176-7].
-> Ne pas s'auto proclamer bon crackeur parce qu'on à cracker 1 ou 2 progs.
-> Avoir un bon écran (type tube cathodique et pas un écran plat) car pour le confort de lecture du listing, c'est primordial.

-> Ne pas rester tout le temps dans le désassembleur et naviguer de façon équilibré entre ce dernier et l'éditeur héxa.
-> Faire pleins d'essais avec l'éditeur hexadécimal.
-> Désassemblez le fichier d'origine et travailler sur une copie du fichier dans l'éditeur hexadécimal.
-> Toujours vérifier si l'effet d'une modification dans l'éditeur hexdécimal est bien celui attendu dans le programme.


and


Nombre de visites depuis le 15/02/2003