Crackme : goldo6
Creator : Goldocrack
Type : crackme, un-nagmes
Difficulty : Easy

Voici mon premier tuto, dans celui-ci on va déplomber le crackme de Goldocrack qui m'as bien plu.

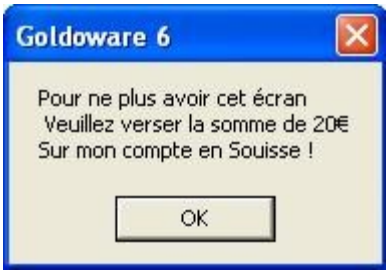
Les outils utilisés :
- OllyDBG
- Une table ASCII
- Un cerveau en état de marche

Ce crackme pour débutant est asser intéressant niveau apprentissage et compréhension.

1 serial à trouver et 4 nags à enlever, on a un programme charger quoi alors au boulot bande de feignasse.

Petits nags ou êtes-vous?

On va d'abord tester le crackme pour repérer les nags à enlever :
1ier nag :
On le lance et déjà un nag :



On clicke ok puis on arrive sur le crackme :

On tape n'importe quoi dans la case pour le sérial (on peut même le laisser vide).
On clicke sur Vérifier et hop :

2ème nag :

Ensuite on clicke sur Quitter et voilà les 2 autres nags :

Bon le travail de reconaissance étant fini, on peut ouvrir OllyDBG et déplomber ça tout gentiment :D .

:D à mort les nags!!!

Tout d'abord ouvrez ce crackme sous OllyDBG.
Click droit puis « Search For » --> « All referenced text strings »
Recherchez les strings des nags : « Goldoware 6 », « Nag a virer », « Pour ne plus avoir cet écran » et « Ce programme a été pirater » .

Je vais vous présenter 2 méthode, la 1 ière c'est la méthode bourrin, pas très esthétique, la 2 ème est mieux.

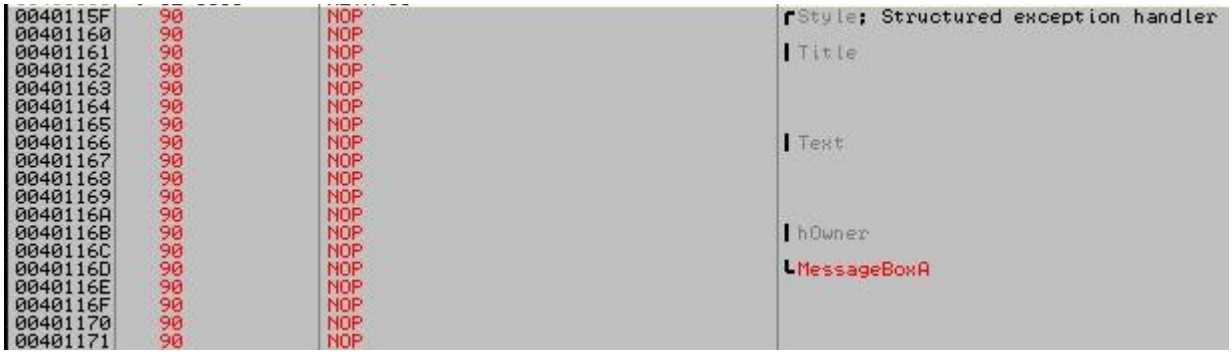
1iere methode : La Methode du bourrin

Avant :
Voilà ce que ça donne lorsqu'on clicke sur « Goldoware 6 ».

```
0040115F    6A 00          PUSH 0                          Style = MB_OK|MB_APPLMODAL; Structured exception handler
00401161    68 00304000    PUSH defi_gol.00403000          Title = "Goldoware 6"
00401166    68 0C304000    PUSH defi_gol.0040300C          Text = "Pour ne plus avoir cet écran♪ Veuillez verser la
0040116B    6A 00          PUSH 0                          hOwner = NULL
0040116D    E8 A6030000    CALL <JMP.&user32.MessageBoxA>  MessageBoxA
```

On va nopper ce MessageBoxA bien gentiment.
Un clique droit --> « Binary » --> « Fill with NOPs »

Vous avez fait ça? :



Il suffisait de nopper en 40116D et pas entièrement lol.

Bon je pense que vous avez compris, on fait pareil pour les autres nags
Ensuite vous faites clicke droit « Copy to executable » --> « All modifications ».
Vous sauvez et oooh miracles plus aucuns nags.

2ème méthode: Une méthode plus esthétique:

On reclicke sur « Goldoware 6 » et on retombe sur ça :

On clique sur 0040116B et on fait un clique droit « Assemble » et vous mettez « PUSH 1 ».
Si une application a déjà ce hOwner à 1 et bah le crackme va bugguer, mais rassurez vous c'est très rarement le cas qu'une application ai ce hOwner égal à 1.

On fait la même chose pour les 3 autres nags.

Vous reconnaitrez que la deuxième méthode est plus jolie car on modifie moins de bits que dans la première.

Ou il est ce sérial?

Et oui vous avez du vous rendre compte en explorant un peu l'executable avec OllyDBG qu'aucuns sérial n'était générer ou stocker en mémoire (si ce n'est celui que vous avez taper).
Mais alors où est-t'il?
On va le voir maintenant :D .

Bon alors on va tout d'abord repérer la routine de comparaison du serial entrer et du serial valide.
On cherche le nag « Nag a virer », on voit ça un peu en dessous de celui-ci :
00401303 . E8 6D000000 CALL defi_gol.00401375
00401308 . EB 65 JMP SHORT defi_gol.0040136F

Il s'avère que lorsqu'on clique sur « Vérifier » on a ce nag puis le message d'Erreur direct!
Pour vérifier ça on pose un BreakPoint sur le CALL, on tape un sérial, « Verifier » et hop on break sur le CALL et paf on a le message d'erreur.

Voyons voir ce CALL :

```
0040136F   > 33C0          XOR EAX,EAX
00401371   . C9            LEAVE
00401372   . C2 1000        RETN 10
00401375  ┌$ E8 01000000    CALL defi_gol.0040137B
0040137A  └. C3            RETN
0040137B  ┌$ 51            PUSH ECX
0040137C  |. 52            PUSH EDX
0040137D  |. 6A 1D          PUSH 1D                                  ┌Count = 1D (29.)
0040137F  |. 68 605C4000    PUSH defi_gol.00405C60                   |Buffer = defi_gol.00405C60
00401384  |. FF35 4B5C4000  PUSH DWORD PTR DS:[405C4B]               |hWnd = 00070300 (class='Edit',parent=000703A6)
0040138A  |. E8 77010000    CALL <JMP.&user32.GetWindowTextA>        └GetWindowTextA
0040138F  |. A3 4F5C4000    MOV DWORD PTR DS:[405C4F],EAX
00401394  |. 833D 4F5C4000  CMP DWORD PTR DS:[405C4F],14
0040139B  |.v0F85 FF000000  JNZ defi_gol.004014A0
004013A1  |. BB 605C4000    MOV EBX,defi_gol.00405C60                 ASCII "fireblast"
004013A6  |. 0FBE03         MOVSX EAX,BYTE PTR DS:[EBX]
004013A9  |. 83F8 47        CMP EAX,47
004013AC  |.v0F85 EE000000  JNZ defi_gol.004014A0
004013B2  |. 43             INC EBX
```

Oho intéressant en remontant un peu on voit 40136F, tiens tiens c'est pas l'adresse de saut?
Le CALL lui est en 401375, à cette adresse on trouve un autre CALL qui va en 40137B.
Réfléchissons : Le CALL qui va en 401375 est avant le JUMP en 40136F, cela voudrait dire que ce JUMP n'est jamais pris, pour vérifier suffit de poser un BreakPoint sur ce JUMP et les autres JUMP.

Je vous fais un rapide commentaire pour que vous compreniez mieux :
0040137D-0040138A : La longueur du serial entrer est stocker en EAX
0040138F : Met la valeur de EAX en 405C4F
00401394 : Compare la longueur de notre serial avec la longueur du bon serial
0040139B : Si les deux serials ont la même longueur (14 en hexadécimal = 21 en décimal) alors on ne saute pas
004013A1 : Met 405C60 dans EBX
004013A6 : Met la première lettre du serial entrer dans EAX
004013A9 : Comparer EAX et la valeur 47
004013AC : Si égal on ne saute pas sinon on va sur le message d'erreur
004013B2 : On incrémente EBX (on va donc à la deuxième lettre du serial entrer)

Comme vous pouvez le deviner en lisant mes commentaires : le serial entrer est stocker en 405C60, pour le savoir c'est facile, on met un breakpoint en 40137F et en lisant le buffer on voit le serial qu'on a entrer.
On sait qu'à la ligne 4013A6 on stocke la première lettre du serial entrer dans EAX puis on compare EAX à 47, en descendant on voit la même façon de faire.
On va tout simplement noter toute les nombres auquels on compare EAX et on obtient ça :
47 6F 6C 64 6F 63 72 61 63 6B 20 48 61 76 65 20 4C 6F 73 74
Ce qui donne en clair :
« Goldocrack_Have_Lost »
A la place des underscore '_' mettez des espace.
Tiens le bon serial ne fait que 20 lettres, ou est la 21 ème lettre? Pour que l'ordinateur sache quant es-ce que termine une chaine de caractères on le lui dit grace à un caractère spécial qui est '\0' donc voilà on a notre serial reste plus qu'à tester.

Voilà j'espère que ce tuto vous a plu, il m'as pris énormément de temps à rédiger car étant moi même débutant.

Si il y a des erreurs n'hésitez pas à me les signaler.

Merci à :
- Crisanar, Demon, Devilz, la ShmeitCorp, etc pour leurs tutos
- Goldocrack pour son superbe crackme (sinon je n'aurais pas passer autant de temps à rediger un tutorial)
- Tous les gars de FC qui m'ont aider (et qui vont encore m'aider je l'espère)
- Tous ceux qui ont lu ce tutos pour l'avoir suivi jusqu'au bout.

FireBlast