Défi #1 d'elooo

(version superexplicado)

par miamasse, le père noël d'été.

Trouvons la partie où va s'effectuer la vérification.

Ce crackme est du type Dialogbox : les différentes composantes de l'interface sont éditables avec des programmes comme resource hacker. Les API servant à récupérer les diverses informations entrées sont généralement de la forme GetDlgItem*. On peut poser un breakpoint pour s'arrêter à l'appel d'une de ces fonctions ou en chercher une dans le listing d'ollybbg.

```
004010AF  |. 6A 2D          PUSH 2D                              ; /Count = 2D (45.)
004010B1  |. 68 00304000    PUSH elooo1.00403000                 ; |Buffer = elooo1.00403000
004010B6  |. 68 EB030000    PUSH 3EB                             ; |ControlID = 3EB (1003.)
004010BB  |. FF75 08        PUSH DWORD PTR SS:[EBP+8]            ; |hWnd
004010BE  |. E8 CB000000    CALL <JMP.&user32.GetDlgItemTextA>   ; \GetDlgItemTextA
004010C3  |. E8 47000000    CALL <elooo1.CheckSerial>
004010C8  |. 6A 00          PUSH 0                               ; /Style = MB_OK|MB_APPLMODAL
004010CA  |. 68 37304000    PUSH OFFSET <elooo1.aTitre>          ; |Title = "Hehe"
004010CF  |. 68 58304000    PUSH OFFSET <elooo1.aMessage>        ; |Text = ""
004010D4  |. FF75 08        PUSH DWORD PTR SS:[EBP+8]            ; |hOwner
004010D7  |. E8 BE000000    CALL <JMP.&user32.MessageBoxA>       ; \MessageBoxA
```

J'ai déjà renommé la fonction appelée en 0x004010C3 car il apparaît évident que c'est elle qui va déterminer si le serial est bon ou non puisque les instructions suivantes ne servent qu'à afficher la boite de dialogue.

la chaîne contenue à l'adresse aMessage, en blanc dans le listing, est initialisée lors de l'exécution par ceci :

```
004010A0  |. 68 49304000    PUSH OFFSET <elooo1.aGrmbl>        ; /String2 = "Grmbl"
004010A5  |. 68 58304000    PUSH OFFSET <elooo1.aMessage>      ; |String1 = OFFSET <elooo1.aMessage>
004010AA  |. E8 09010000    CALL <JMP.&kernel32.lstrcpyA>      ; \lstrcpyA
```

Maintenant que l'on a trouvé l'appel vers la fonction qui sert à vérifier la validité du serial, nous allons étudier cette fonction.

Etude de la fonction de vérification : CheckSerial

```
0040110F >/$ 60             PUSHAD
00401110  |. 68 00304000    PUSH elooo1.00403000
00401115  |. E8 32000000    CALL elooo1.0040114C
0040111A  |. B9 07000000    MOV ECX,7
0040111F  |. 99             CDQ
00401120  |. F7F9           IDIV ECX
00401122  |. 83E9 03        SUB ECX,3
00401125  |. 99             CDQ
00401126  |. F7E9           IMUL ECX
00401128  |. 05 37130000    ADD EAX,1337
0040112D  |. 83E8 09        SUB EAX,9
00401130  |. 3B05 32304000  CMP EAX,DWORD PTR DS:[403032]
00401136  |. 75 0F          JNZ SHORT elooo1.00401147
00401138  |. 68 3C304000    PUSH OFFSET <elooo1.aCestGood>   ; /String2 = "C'est good !"
0040113D  |. 68 58304000    PUSH OFFSET <elooo1.aMessage>    ; |String1 = OFFSET <elooo1.aMessage>
00401142  |. E8 71000000    CALL <JMP.&kernel32.lstrcpyA>    ; \lstrcpyA
00401147  |> 61             POPAD
00401148  \. C3             RETN
```

La ligne en 0x00401136 est la plus significative de CheckSerial : c'est elle qui va servir à diriger la suite de l'exécution du programme en fonction de la validité de notre serial. C'est ce que

l'on appelle le saut vers le goodboy ou le badboy.

```
00401130  |. 3B05 32304000  CMP EAX,DWORD PTR DS:[403032]           ;  EAX = 0xBADCAFE ??
00401136  |. 75 0F          JNZ SHORT elooo1.00401147
```

Ce test compare EAX à 0x0BADCAFE, et si EAX est différent alors on saute l'étape qui consiste à écrire la chaîne "C'est good !" à l'adresse aMessage.

Étudions le code pour trouver le serial.

Comme nous l'avons vu dans la fonction CheckSerial, le code que l'on a entré est comparé à 0xBADCAFE, il va falloir trouver les différentes étapes qui vont modifier ce code.

Le fonction de traitement de la chaîne serial.

```
0040114C  /$ 55             PUSH EBP
0040114D  |. 8BEC           MOV EBP,ESP
0040114F  |. 56             PUSH ESI
00401150  |. 57             PUSH EDI
00401151  |. 33C0           XOR EAX,EAX
00401153  |. 8B75 08        MOV ESI,DWORD PTR SS:[EBP+8]
00401156  |. 33C9           XOR ECX,ECX
00401158  |. 33D2           XOR EDX,EDX
0040115A  |. 8A06           MOV AL,BYTE PTR DS:[ESI]
0040115C  |. 46             INC ESI
0040115D  |. 3C 02          CMP AL,2
0040115F  |. 75 12          JNZ SHORT elooo1.00401173
00401161  |. 8A06           MOV AL,BYTE PTR DS:[ESI]
00401163  |. F7D2           NOT EDX
00401165  |. 46             INC ESI
00401166  |. EB 0B          JMP SHORT elooo1.00401173
00401168  |> 2C 30          /SUB AL,30
0040116A  |. 8D0C89         |LEA ECX,DWORD PTR DS:[ECX+ECX*4]
0040116D  |. 8D0C48         |LEA ECX,DWORD PTR DS:[EAX+ECX*2]
00401170  |. 8A06           |MOV AL,BYTE PTR DS:[ESI]
00401172  |. 46             |INC ESI
00401173  |> 0AC0            OR AL,AL
00401175  |.^75 F1          \JNZ SHORT elooo1.00401168
00401177  |. 8D0411         LEA EAX,DWORD PTR DS:[ECX+EDX]
0040117A  |. 33C2           XOR EAX,EDX
0040117C  |. 5F             POP EDI
0040117D  |. 5E             POP ESI
0040117E  |. C9             LEAVE
0040117F  \. C2 0400        RETN 4
```

Le code peut paraître complexe mais il est en fait très simple. ESI va contenir un pointeur sur notre serial, puis les opérations suivantes vont être effectuées :

- le code ASCII du caractère serial[ESI] va être soustrait de 0x30 pour obtenir la valeur numérique correspondante (ChiffreSerial).
- ECX va être multiplié par 5 (ECX + 4 * ECX) puis à nouveau multiplié par 2 pour enfin être ajouter de EAX (la valeur numérique correspondante). Ce qui donne SerialTemp = SerialTemp * 10 + ChiffreSerial[ESI] : une simple conversion ASCII -> décimale.
- On incrémente ESI et on reboucle tant que serial[ESI]<>0.

Lorsque la boucle est finie, on met le résultat de l'addition de ECX et de EDX dans EAX

La notation serial[ESI] correspond au caractère de la chaîne serial à l'adresse contenu dans ESI. On considère la chaîne ASCII comme un tableau de caractère.

.

EDX peut avoir deux valeurs car un test est effectué sur pour savoir comment initialiser la boucle à l'adresse 0x0040115D. Cette condition est satisfaite si le premier caractère du serial a pour code

ASCII la valeur 2. Dans ce cas EDX est 0xFFFFFF.

Revenons maintenant aux opérations effectuées avant la comparaison.

La valeur numérique du serial est contenu dans EAX, il va subir ces opérations :

serial = serial / 7
serial = serial * 4
serial = serial + 0x1337
serial = serial – 9

Puis comparaison de SERIAL avec 0xBADCAFE.

Les calculs se font sur des entiers naturels, le reste de la division n'est pas pris en compte.

Il suffit d'appliquer les opérations inverses dans le sens inverse pour trouver la bonne valeur numérique du serial.

Avec Ollydbg et le plugin commandbar, il suffit de taper l'opération « (0xBADCAFE + 9 - 0x1337) / 4 * 7 » pour que s'affiche le résultat, soit 342884780 (0x147001AC).

Extra :

Pour le deuxième serial débutant par le code ascii 0x2, on doit tenir compte de deux autres opérations : lea [ecx + edx] et xor eax, edx.

Le serial avant ces opérations doit être 0xEB8FFE54, soit en décimal non signé 3952082516. À vous de trouver les calculs correspondants pour arriver à ce résultat.

Remerciements :

Merci Elooo pour ce crackme plein de petites références, même le caractère au code ASCII 2 est poilant.
Un merci aux lecteurs sans qui ce texte ne serait rien, et aux auteurs des tutoriels.