

Tutorial #7: CrackMe de Crudd

NIVEAU 2

Outils:

- Le [Crackme](#) de Crudd
- Un désassembleur/debugger: Win32asm
- Un éditeur hexadécimal: WinHex 10.2
- ResHacker

Observation du CrackMe:

- Level 1:
 - Un menu non-valable
 - Un CD-Check normal.
- Level 2:
 - Un menu non-valable un peu plus difficile
 - Un CD-Check plus difficile (pas de string data ref)
 - Un Anti-SI Serial number, ou FrogIce ne nous apprend rien.
- Level 3:
 - Plante sous Windows XP donc pas d'intérêt pour nous.

Objectifs:

Niveau 2

- Rendre les menus bloqués accessibles
- Faire apparaître le message de réussite

Partie 1: Rendre le menu "Enable Me" accessible

Bien, il faut maintenant rendre le menu "Enable Me 2" accessible. On ouvre ResHacker, et on essaie de dégriser le menu comme dans le cours précédent: malheureusement, celui-ci n'est pas grisé. Et oui, celui-ci va nous donner un challenge un peu plus conséquent que le précédent!

Pour ça, on va devoir analyser le code plus en profondeur. On ouvre Wdasm, et on cherche une SDR qui peut nous amener vers quelque chose d'intéressant. On double-clique sur la SDR "*Enable Me 2*" et voila ou on arrive:

```
* Possible Ref to Menu: DICK, Item: "About"
|
:004010E6 6A01 push 00000001

* Possible Ref to Menu: DICK, Item: "Enable Me 2"
|
:004010E8 6A06 push 00000006
:004010EA FF356C324000 push dword ptr [0040326C]
:004010F0 90 nop
:004010F1 90 nop
:004010F2 90 nop
:004010F3 90 nop
:004010F4 90 nop

* Possible Ref to Menu: DICK, Item: "About"
|
:004010F5 6A01 push 00000001

* Possible Ref to Menu: DICK, Item: "Enable Me 3"
|
:004010F7 6A09 push 00000009
:004010F9 FF356C324000 push dword ptr [0040326C]
* Reference To: USER32.EnableMenuItem, Ord:00B4h
|
:004010FF E832080000 Call 00401936

* Possible StringData Ref from Data Obj -> "UN"
|
:00401104 689D304000 push 0040309D

* Possible StringData Ref from Data Obj -> "Crudd Me 3.0"
```

En regardant un peu plus bas, on voit une référence à l'API [USER32.EnableMenuItem](#), [Ord:00B4h](#), celle qui nous intéresse (Enable en Anglais = rendre possible, permettre - suivi de MenuItem = partie du menu).

A partir d'ici, ça devient plus technique donc je vous recommande de bien suivre. C'est le code surligné en rouge qui nous intéresse dans un premier temps.

On aperçoit sous cette API un CALL, qu'il va suffire de nopper (manière bourrin mais on est sur un CrackMe donc on ne cherche pas la finesse, même si en y travaillant un peu plus c'est

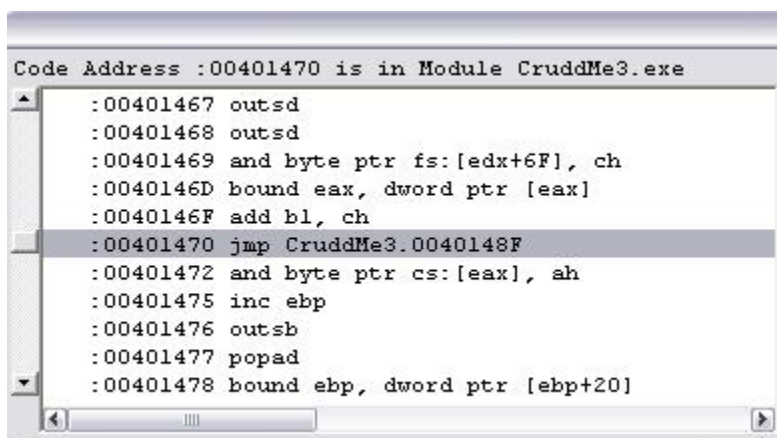
faisable!). Dans WinHex, on remplace donc E832080000 par 9090909090. La on sauvegarde les modif, et on ouvre le CrackMe: le menu n'est plus grisé, mais rien n'apparait lorsqu'on clique dessus.

On sait de manière certaine qu'il existe bien un message de réussite: il est facile de vérifier en faisant une recherche texte par exemple dans WinHex, et en tapant "Level 2", WinHex nous amène à l'offset **875**, et le message est "Enable Level 2 Complete". Il doit donc y avoir moyen d'obtenir ce message en modifiant du code...

Alors regardons maintenant le code surligné en jaune au-dessus: On va chercher ou le programme fait appel à ce message, en recherchant un **0006** et on tombe ici:

```
* Referenced by a (U)nconditional or
(C)onditional Jump at Address:
|:0040141B(C)
|
:00401455 6683F806 cmp ax, 0006
:00401459 0F8583000000 jne 004014E2
:0040145F E8BD030000 call 00401821
:00401464 EB0A jmp 00401470
:00401466 47 inc edi
:00401467 6F outsd
:00401468 6F outsd
:00401469 64206A6F and byte ptr fs:[edx+6F], ch
:0040146D 6200 bound eax, dword ptr [eax]
```

Le JMP en 00401464 vers l'adresse **00401470** ne mène nulle part, mais c'est lui qui nous intéresse. On va faire une opération de debugging avec Wdasm et poser un BreakPoint sur la ligne 00401464. On lance le process avec F9, puis après le lancement du CrackMe on clique sur le menu Enable Me 2, et Wdasm breake. On rentre dans le call (Step Into) avec F7.



```
Code Address : 00401470 is in Module CruddMe3.exe
:00401467 outsd
:00401468 outsd
:00401469 and byte ptr fs:[edx+6F], ch
:0040146D bound eax, dword ptr [eax]
:0040146F add bl, ch
:00401470 jmp CruddMe3.0040148F
:00401472 and byte ptr cs:[eax], ah
:00401475 inc ebp
:00401476 outsb
:00401477 popad
:00401478 bound ebp, dword ptr [ebp+20]
```

On voit alors ce qui se passe à l'adresse 00401470: on Jumps vers **0040148F** donc on va terminer le debugging avec **Terminate** et y aller voir...

```

:0040148F 6A00 push 00000000
:00401491 A066144000 mov al, byte ptr [00401466] <-- Contient le message "Good Job"
:00401496 660FB6C0 movzx ax, al
:0040149A 6650 push ax
:0040149C 6A00 push 00000000
:0040149E A094324000 mov al, byte ptr [00403294] <-- du vide
:004014A3 660FB6C0 movzx ax, al
:004014A7 6650 push ax

* Reference To: KERNEL32.lstrcpyA, Ord:02DCh
|
:004014A9 E812050000 Call 004019C0
:004014AE 6A00 push 00000000
:004014B0 A072144000 mov al, byte ptr [00401472] <-- Correspond offset 875, donc message de réussite
:004014B5 660FB6C0 movzx ax, al
:004014B9 6650 push ax
:004014BB 6A00 push 00000000
:004014BD A094324000 mov al, byte ptr [00403294] <-- du vide a nouveau
:004014C2 660FB6C0 movzx ax, al
:004014C6 6650 push ax

* Reference To: KERNEL32.lstrcatA, Ord:02D3h
|
:004014C8 E8ED040000 Call 004019BA
:004014CD 6894324000 push 00403294
:004014D2 FF3590324000 push dword ptr [00403290]

```

Si l'on observe, on voit que le programme pousse le bon message et remplace par du vide. Pour bien comprendre, je vous renvoie une fois de plus vers les tutos assembleur dispos sur le site...

- Le MOV en **00401491** doit être remplacé par un **PUSH** (donc le **A0** par un **68**).
- Ensuite pour pousser du vide, on doit aussi remplacer le **660FB6C0** à l'adresse **00401496** par un **PUSH (68)**: on le remplace donc par **6894324000** (68=le push / 94324000= le vide).
- Ensuite on veut directement aller vers l'API, donc on remplace le **506A** (un push) par **EB0C** (un JMP).

Mêmes opérations pour les lignes sous **KERNEL32.lstrcpyA**:

- Pour le MOV en **004014B0**: on remplace le **A0** par un **68**.
- Le MOV en **004014B5** est remplacé par **6894324000**.
- Le 506A juste en dessous est remplacé par un **EB0C**

Bref, voilà, après une sauvegarde des modif' dans l'éditeur hexa et un essai, on arrive à un petit message qui nous dit "Good Job".

Partie 2: Virer le CD-Check 2

On passe au CD-Check. On se rend au niveau de la SDR "Level 2 CD Check passed"

(puisqu'on est au Level 2) et on voit ça:

```
* Reference To: KERNEL32.GetFileAttributesA, Ord:00F9h
|
:00401530 E86D040000 Call 004019A2
:00401535 83F801 cmp eax, 00000001
:00401538 7412 je 0040154C
|
* Referenced by a (U)nconditional or (C)onditional Jump at Address:
|:00401556(C)
|
:0040153A 68F3144000 push 004014F3
:0040153F FF3590324000 push dword ptr [00403290]
|
* Reference To: USER32.SetWindowTextA, Ord:0259h
|
:00401545 E828040000 Call 00401972
:0040154A EB1C jmp 00401568
|
* Referenced by a (U)nconditional or (C)onditional Jump at Address:
|:00401538(C)
|
:0040154C 6A00 push 00000000
|
* Reference To: KERNEL32.GetDriveTypeA, Ord:00F0h
|
:0040154E E849040000 Call 0040199C
:00401553 83F805 cmp eax, 00000005
:00401556 75E2 jne 0040153A
|
* Possible StringData Ref from Code Obj ->"Level 2 CD Check Passed."
|
:00401558 6815154000 push 00401515
:0040155D FF3590324000 push dword ptr [00403290]
```

Vous connaissez mes méthodes, et ce sont les lignes surlignées en rouge qui vont nous intéresser, et ici en l'occurrence les sauts qui suivent sont en jaune (simplement pour une meilleure visibilité de ce vers quoi on saute).

On voit que le JE en 00401438 renvoie sur la fonction **Kernel32.GetDriveTypeA**, a l'adresse **0040154C** donc on va rendre ce saut qui est conditionnel ('if equal') définitif et systématique, donc le remplacer par un JMP. Le **74** devient alors un **EB**, mais plus besoin de le rappeler!

Ensuite, il ya un JNE à l'adresse 00401556, qu'il va suffire de changer en **JE** pour qu'il ne saute pas en 0040153A (qui se trouve au-dessus en jaune). Par le fait, le **75** devient un **74**.

Opération terminée, on enregistre et on essaie: le message en vert est bel et bien ce qu'on voit désormais.

Partie 3: Trouver le Serial du Level 2

C'est une partie BIDON, c'en est affligeant. Vous le trouverez tous seuls j'en suis persuadé... faites simplement une opération de debugging avec WinDasm, et ne vous demandez pas trop pourquoi le seul truc que vous voyez dans les résultats API (après avoir cliqué précisément sur le petit carré "Get API Results") c'est votre pseudo.

Il y a une longue série d'opérations (que je n'aurai le courage de décrire ici, c'est sans intérêt) sur chaque caractère de votre pseudo pour vous le rendre dans la partie serial. Ouais, en effet, le serial correct c'est ce que vous rentrez dans la section "Name"!

Conclusion Niveau 2:

Un CD-Check un peu bateau mais formateur tout de même...

Beaucoup plus de difficulté pour l'affichage du message du message de réussite, mais un résultat plutôt satisfaisant. Vous devriez à présent commencer à bien maîtriser certaines fonctions de WinDasm.

Voila, une fois de plus, Félicitations !