Solution du défi Test Your Level de CarosVonMagor
par skirby

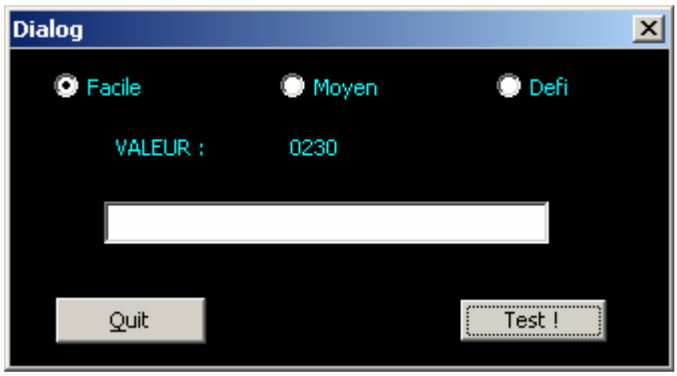
Aperçu du défi

Pour résoudre ce défi vous aurez besoin de OllyDbg et potentiellement d'un compilateur C si vous souhaiter compiler le code source du brute force..

Premier réflexe avant de débuguer un défi, on le test avec PEiD ou RDG Packer Detector.
Dans le cas présent, aucun packer n'est détecté.
En revanche, on voit ceci : Microsoft Visual C++ 7.0 [Debug]

Debug signifie qu'il va y avoir pas mal de code inutile dans le source une fois désassemblé.
Décidemment, **CarosVonMagor** est le champion de la compilation en mode debug (cf. précédents défis et tutorial associés)

Allez hop, on charge le défi dans OllyDbg et on fait F9.
On obtient l'écran suivant :



A priori, aucun anti-debugger ni aucun trick anti-ollydbg au lancement de l'application.

Analyse du des fonctions de l'API

Un deuxième bon réflexe consiste à jeter un œil rapide sur les fonctions de l'API Windows.
Clic droit sur la fenêtre du code puis « Search For » puis « All Intermodular Calls »

On peut voir des appels à IsDlgButtonChecked et GetDlgItemTextA
On ne constate aucun appel à MessageBox (comme ces précédents défis où le message de réussite s'affiche dans une boîte de dialogue.

IsDlgButtonChecked permet de savoir si un radio bouton est coché et GetDlgItemTextA de récupérer le texte contenu dans une zone de texte.

On va donc poser un breakpoint sur ces fonctions.
A coup sûr, elles devraient nous mener dans la fonction de vérification.

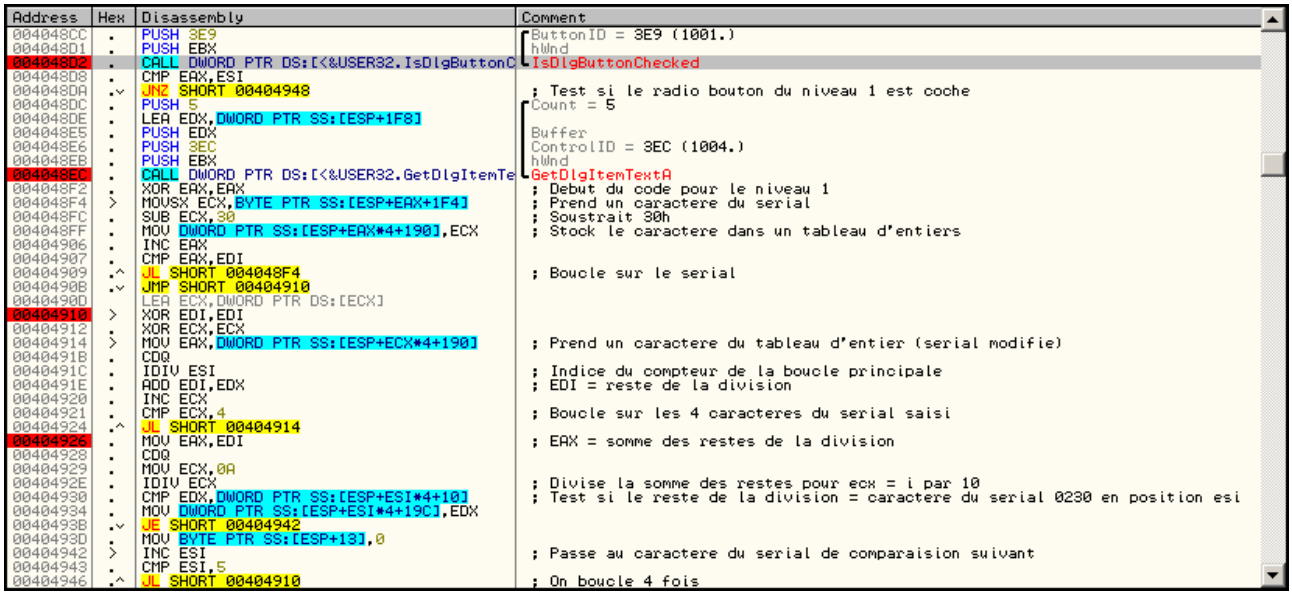
Analyse du des String Data Reference

En règle générale, il faut pouvoir se passer de SDR quand on veut cracker un programme.
J'en veux pour preuve le défi présent qui ne contient aucune SDR intéressante.
On va donc utiliser simplement les API trouvées ci-dessus.

Debuggage de l’application

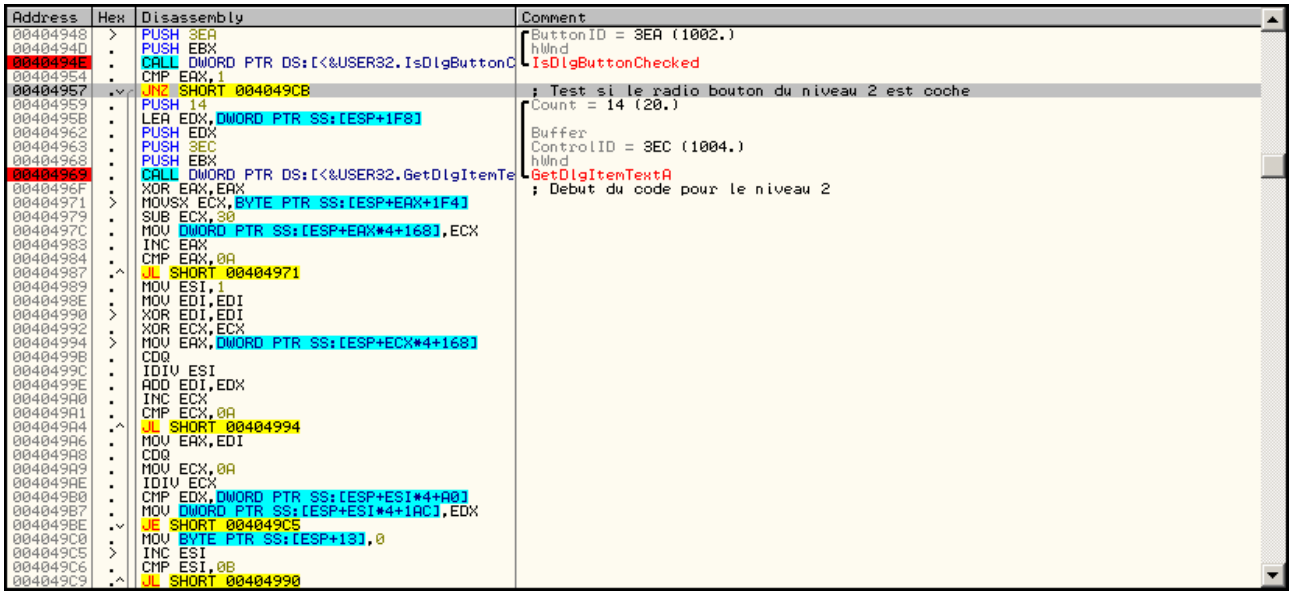
On lance le programme (F9) et la boîte de dialogue nous invite à saisir le nom et le serial.
On entre 12345678 puis on clique sur le bouton Test !.

On break sur la fonction IsDlgButtonChecked pour arriver ici.



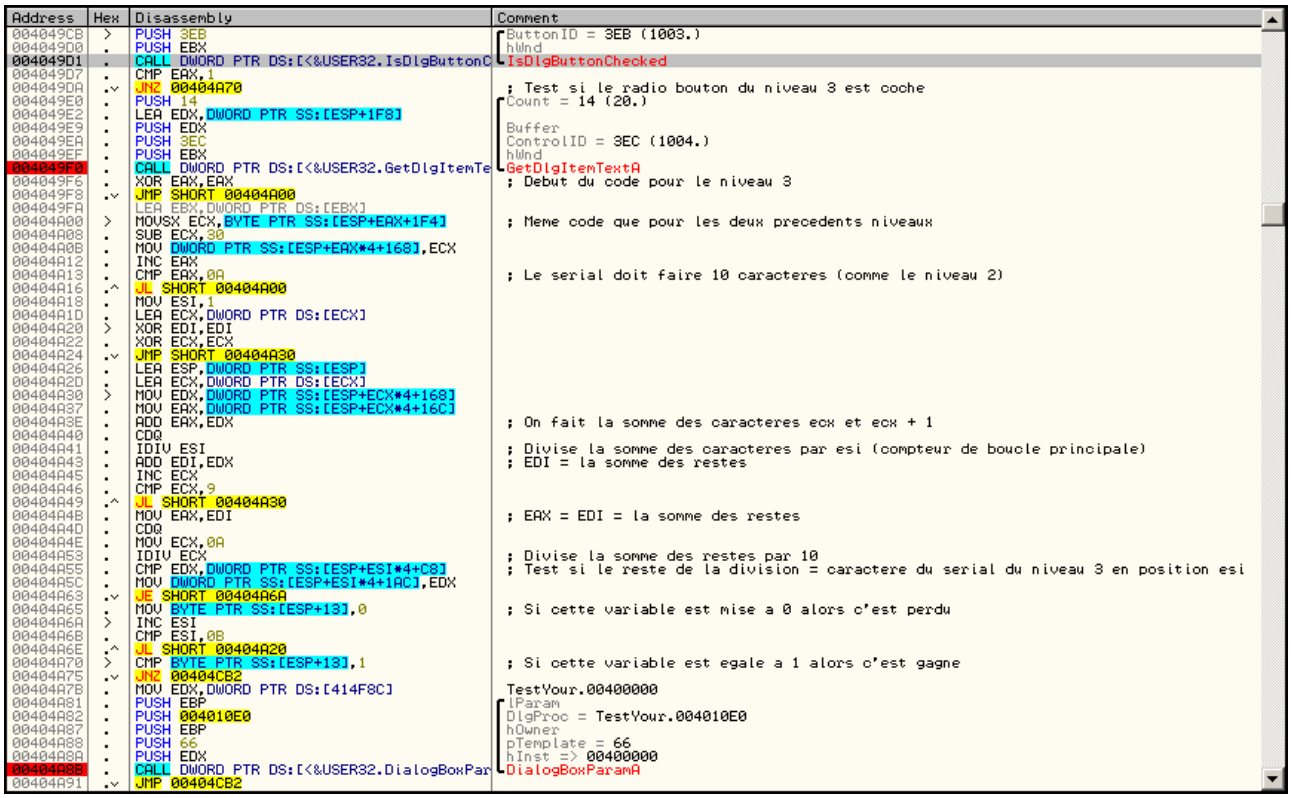
Les commentaires devraient permettre de comprendre facilement ce que fait le code.
Il n’y a aucune difficulté particulière à part que l'on peut d'ores et déjà se demander comment on va keygener cet algorithme.

Juste en dessous, on retrouve quasiment le même code. Il s'agit du code de vérification du niveau 2. La seule différence si situe au niveau du nombre de caractères du serial qui seront utilisés : 10 contre 4 pour le niveau 1.



Encore un peu plus en dessous, on retrouve encore quasiment le même code mais avec quelques variantes. Il s'agit du code de vérification du niveau 3.

Je vous laisse analyser le code pour comprendre ce qu'il fait.



Maintenant, la vrai question est comment keygener ceci.
Et bien je n'en sait rien :o(

C'est pourquoi, je vous propose la méthode de brute qui s'appelle : Le brute force.

Un brute force va s'appuyer le plus souvent sur le code du programme et va calculer TOUTES les combinaisons possible jusqu'à trouver la bonne.
Ca peut être très long (plusieurs heures et même plusieurs jours) mais dans notre cas, le code est assez court et le champ d'action est limité. En effet, on a un nombre fini de caractères (4 pour le niveau 1 et 10 pour les niveaux 2 et 3) et ils doivent être compris entre 0 et 9.

Vous trouverez ci-dessous le code source du brute force.
Je l'ai commenté donc il devrait être facile à comprendre.

Code source du Brute Force :

```
#include <windows.h>
#include <stdio.h>
#include <time.h>

#define debug FALSE

int sCode[15];  // tableau contenant le code à chercher
int iCodeCmp[10];  // tableau contenant le code de comparaison

////////////////////////////////////////////////////////////////////////////
// Main function
int main(void)
{
  int i;                      // indice tableau
  int longueur;               // Nombre de caractère du serial
  unsigned int iFound;        // Permet de savoir si un serial à été trouvé
  int begin = 0, end = 9;     // Caractère de début et de fin du serial
  time_t tStart, tEnd;        // Compteur de temps

  int niveau = 1;   // Ici on détermine le niveau à exécuter

  if (niveau == 1) {
    // Niveau 1
    iCodeCmp[0] = 0; iCodeCmp[1] = 2; iCodeCmp[2] = 3; iCodeCmp[3] = 0;
    longueur = 4;
  } else if (niveau == 2) {
    // Niveau 2
    iCodeCmp[0] = 0; iCodeCmp[1] = 6; iCodeCmp[2] = 1; iCodeCmp[3] = 4; iCodeCmp[4] = 0; iCodeCmp[5]
= 6; iCodeCmp[6] = 9; iCodeCmp[7] = 4; iCodeCmp[8] = 1; iCodeCmp[9] = 0;
    longueur = 10;
  } else {
    // Niveau 3
    iCodeCmp[0] = 0; iCodeCmp[1] = 4; iCodeCmp[2] = 1; iCodeCmp[3] = 2; iCodeCmp[4] = 7; iCodeCmp[5]
= 8; iCodeCmp[6] = 9; iCodeCmp[7] = 0; iCodeCmp[8] = 9; iCodeCmp[9] = 2;
    longueur = 10;
  }

  // Initialise la chaine qui va recevoir le serial final
  for (i=0; i < longueur; i++)
    sCode[i] = 0;

  time(&tStart);  // Début du brute force

  while(sCode[longueur-1] <= end)
  {
    i = 0; // init indice
    while(sCode[i] > end && i+1 != longueur)
    {
      sCode[i] = begin;
      sCode[++i]++;
    }

    ////////////////////////////////////////////////////////////////
    // Algo à bruteforcer
    if (niveau == 1) {
      ////////////////////////
      // Niveau 1
      _asm {
          mov     esi, 1
Debut1:
          xor     edi, edi
          xor     ecx, ecx

Boucle1:
          mov     eax, dword ptr sCode[ecx*4]
          cdq
          idiv    esi
          add     edi, edx
          inc     ecx
          cmp     ecx, 4
          jl      short Boucle1
          mov     eax, edi
          cdq
          mov     ecx, 0Ah
          idiv    ecx
```

```
          cmp     edx, iCodeCmp[esi*4 - 4]
          jz      Next1
          jmp End1

Next1:
          inc     esi
          cmp     esi, 5
          jl      short Debut1

          mov iFound, 1
End1:
      }
    }
    else if (niveau == 2) {
      ////////////////////////
      // Niveau 2
      _asm {
          mov     esi, 1
Debut2:
          xor     edi, edi
          xor     ecx, ecx

Boucle2:
          mov     eax, dword ptr sCode[ecx*4]
          cdq
          idiv    esi
          add     edi, edx
          inc     ecx
          cmp     ecx, 0Ah
          jl      short Boucle2
          mov     eax, edi
          cdq
          mov     ecx, 0Ah
          idiv    ecx
          cmp     edx, iCodeCmp[esi*4 - 4]
          jz      Next2
          jmp End2

Next2:
          inc     esi
          cmp     esi, 0Bh
          jl      short Debut2

          mov iFound, 1
End2:
      }
    }
    else {
      ////////////////////////
      // Niveau 3
      _asm {
          mov     esi, 1
Debut3:
          xor     edi, edi
          xor     ecx, ecx

BoucleSomme3:
          mov     edx, dword ptr sCode[ecx*4]
          mov     eax, dword ptr sCode[ecx*4+4]
          add     eax, edx
          cdq
          idiv    esi
          add     edi, edx
          inc     ecx
          cmp     ecx, 9
          jl      short BoucleSomme3

          mov     eax, edi
          cdq
          mov     ecx, 0Ah
          idiv    ecx
          cmp     edx, iCodeCmp[esi*4 - 4]
          jz      Next3
          jmp End3

Next3:
          inc     esi
          cmp     esi, 0Bh
          jl      short Debut3
```

```
          mov iFound, 1
End3:
      }
    }
    ////////////////////////////////////////////////////////////////

#if debug == TRUE
    // Mettre debug = true permet de tracer les serials testé à l'écran
    for(int cpt = 0; cpt < longueur; cpt++)
      printf("%d", sCode[cpt]);
    printf("\n");
#endif

    if (iFound == 1) goto EndFunction;  // Sort si on trouvé un serial

    sCode[0]++; // caractere suivant
  }

EndFunction:
  if (iFound == 1) {
    time(&tEnd);  // Fin du brute force

    printf("Solution : ");
    for(int cpt = 0; cpt < longueur; cpt++)
      printf("%d", sCode[cpt]);

    printf("\nTemps total : %d\n", tEnd - tStart);

    MessageBox(0, "Fini", "Yeah!!!", MB_ICONINFORMATION);
  }
  else {
    printf("Aucun serial trouve :o(\n");
  }

  return(0);
}
```

Epilogue

Vous pouvez compiler ce code source avec Visual C++ ou Pelles C (excellent IDE + compilateur et très léger comparé à Visual C++).

Si vous voulez utiliser Code::Blocks avec le compilateur GCC, il vous faudra modifier le code source en Assembleur car la syntaxe n'est pas la même.

Le brute force ci-dessus me génère des codes pour les 3 niveaux avec les temps suivants :
Niveau 1 : moins d'une seconde (en fait, c'est instantané)
Niveau 2 : 643 secondes => soit 10 minutes et 43 secondes
Niveau 3 : 118 secondes => soit 1 minute et 58 secondes

Les calculs ont été réalisés sur un P4 à 2.8 Ghz avec mon Brute Force qui disons le franchement n'est pas vraiment optimisé.

Les serials trouvés sont :
Niveau 1 : 7632
Niveau 2 : 9876554321
Niveau 3 : 8982919040

Vous devriez avoir la boîte de dialogue suivante si le code saisi est correct :

J’espère avoir été le plus clair possible.
Si vous avez des questions, n’hésitez pas à venir les poser sur FORUMCRACK.

Remerciements

Merci à tous les membres de ForumCrack.
Et bien sûr, merci à **CarosVonMagor** pour ce défi.

skirby

Note de l'auteur du défi :

Attention : pour le mode moyen et défi je ne sais pas si une solution existe pour reverser en un temps raisonnable ou keygenner.

Donc voici le ranking:
Niveau Debutant: Patcher
Niveau Intermediaire: Résoudre le premier crackme
Niveau Intermediaire flemmard: Solution BruteForce (donner au moins l'algo sinon c'est pas la peine)
Niveau Pro: Solution du 2eme ou BruteForce intelligent. (Donner le temps de bruteforcing)
Niveau Legende: Solution du 3eme ou BruteForce intelligent. (Idem)
Niveau Dieu: Keygenner le niveau Facile
Niveau Zeus: Keygenner le niveau Moyen
Niveau Dieu du Reversing: Keygenner le niveau Difficile

Promotion : Niveau Zeus + Dieu du Reversing un voyage aller retour tout frais payé sur Singapour Airlines pour l'Asie avec hotesses nues et avion rempli de demoiselles/hommes sexuellement intelligents (chacun ses gouts).

Comme vous pouvez le constater, je me suis arrêté au niveau légende (ce qui est déjà pas mal)
Libre à vous de pousser l'analyse et de réussir le niveau Dieu.