

Solution du défi Test Your Level de CarosVonMagor par skirby

<i>Aperçu du défi</i>	2
<i>Analyse du des fonctions de l'API</i>	2
<i>Analyse du des String Data Reference</i>	2
<i>Debuggage de l'application</i>	3
<i>Code source du Brute Force :</i>	5
<i>Epilogue</i>	8
<i>Remerciements</i>	8
<i>Note de l'auteur du défi :</i>	9



Aperçu du défi

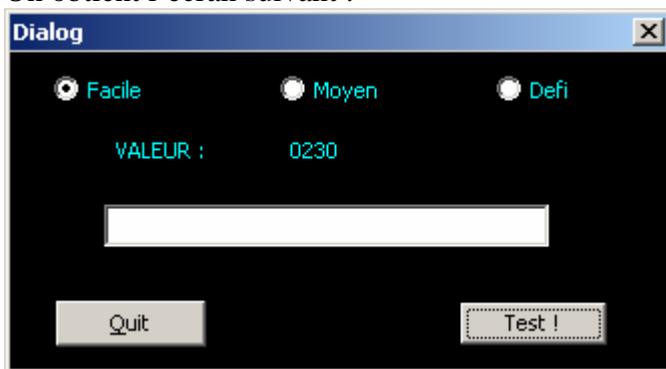
Pour résoudre ce défi vous aurez besoin de OllyDbg et potentiellement d'un compilateur C si vous souhaitez compiler le code source du brute force..

Premier réflexe avant de débiter 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 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 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.

Address	Hex	Disassembly	Comment
004048CC	.	PUSH 3E9	ButtonID = 3E9 (1001.)
004048D1	.	PUSH EBX	hWnd
004048D2	.	CALL DWORD PTR DS:[&USER32.IsDlgButtonChecked]	IsDlgButtonChecked
004048D8	.	CMP EAX,ESI	
004048DA	.	JNZ SHORT 00404948	; Test si le radio bouton du niveau 1 est coché
004048DC	.	PUSH 5	Count = 5
004048DE	.	LEA EDX,DWORD PTR SS:[ESP+1F8]	
004048E5	.	PUSH EDX	Buffer
004048E6	.	PUSH 3EC	ControlID = 3EC (1004.)
004048EB	.	PUSH EBX	hWnd
004048EC	.	CALL DWORD PTR DS:[&USER32.GetDlgItemTextA]	GetDlgItemTextA
004048F2	.	XOR EAX,EAX	; Debut du code pour le niveau 1
004048F4	>	MOVSX ECX,BYTE PTR SS:[ESP+EAX+1F4]	; Prend un caractere du serial
004048FC	.	SUB ECX,30	; Soustrait 30h
004048FF	.	MOV DWORD PTR SS:[ESP+EAX*4+190],ECX	; Stock le caractere dans un tableau d'entiers
00404906	.	INC EAX	
00404907	.	CMP EAX,EDI	
00404909	^	JL SHORT 004048F4	; Boucle sur le serial
0040490B	.	JMP SHORT 00404910	
0040490D	.	LEA ECX,DWORD PTR DS:[ECX]	
00404910	>	XOR EDI,EDI	
00404912	>	XOR ECX,ECX	
00404914	>	MOV EAX,DWORD PTR SS:[ESP+ECX*4+190]	; Prend un caractere du tableau d'entier (serial modifie)
0040491B	.	CDQ	
0040491C	.	IDIV ESI	; Indice du compteur de la boucle principale
0040491E	.	ADD EDI,EDX	; EDI = reste de la division
00404920	.	INC ECX	
00404921	.	CMP ECX,4	; Boucle sur les 4 caracteres du serial saisi
00404924	^	JL SHORT 00404914	
00404926	.	MOV EAX,EDI	; EAX = somme des restes de la division
00404928	.	CDQ	
00404929	.	MOV ECX,0A	
0040492E	.	IDIV EAX	; Divise la somme des restes pour ecx = i par 10
00404930	.	CMP EDX,DWORD PTR SS:[ESP+ESI*4+10]	; Test si le reste de la division = caractere du serial 0230 en position esi
00404934	.	MOV DWORD PTR SS:[ESP+ESI*4+19C],EDX	
0040493B	.	JE SHORT 00404942	
0040493D	.	MOV BYTE PTR SS:[ESP+13],0	
00404942	>	INC ESI	; Passe au caractere du serial de comparaison suivant
00404943	.	CMP ESI,5	
00404946	^	JL SHORT 00404910	; On boucle 4 fois

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.

Address	Hex	Disassembly	Comment
00404948	>	PUSH 3EA	ButtonID = 3EA (1002.)
0040494D	.	PUSH EBX	hWnd
0040494E	.	CALL DWORD PTR DS:[&USER32.IsDlgButtonChecked]	IsDlgButtonChecked
00404954	.	CMP EAX,1	
00404957	.	JNZ SHORT 004049C8	; Test si le radio bouton du niveau 2 est coché
00404959	.	PUSH 14	Count = 14 (20.)
0040495B	.	LEA EDX,DWORD PTR SS:[ESP+1F8]	
00404962	.	PUSH EDX	Buffer
00404963	.	PUSH 3EC	ControlID = 3EC (1004.)
00404968	.	PUSH EBX	hWnd
00404969	.	CALL DWORD PTR DS:[&USER32.GetDlgItemTextA]	GetDlgItemTextA
0040496F	.	XOR EAX,EAX	; Debut du code pour le niveau 2
00404971	>	MOVSX ECX,BYTE PTR SS:[ESP+EAX+1F4]	
00404979	.	SUB ECX,30	
0040497C	.	MOV DWORD PTR SS:[ESP+EAX*4+168],ECX	
00404983	.	INC EAX	
00404984	.	CMP EAX,0A	
00404987	^	JL SHORT 00404971	
00404989	.	MOV ESI,1	
0040498E	>	MOV EDI,EDI	
00404990	>	XOR EDI,EDI	
00404992	>	XOR ECX,ECX	
00404994	>	MOV EAX,DWORD PTR SS:[ESP+ECX*4+168]	
0040499B	.	CDQ	
0040499C	.	IDIV ESI	
0040499E	.	ADD EDI,EDX	
004049A0	.	INC ECX	
004049A1	.	CMP ECX,0A	
004049A4	^	JL SHORT 00404994	
004049A6	.	MOV EAX,EDI	
004049A8	.	CDQ	
004049A9	.	MOV ECX,0A	
004049AE	.	IDIV EAX	
004049B0	.	CMP EDX,DWORD PTR SS:[ESP+ESI*4+00]	
004049B7	.	MOV DWORD PTR SS:[ESP+ESI*4+1AC],EDX	
004049BE	.	JE SHORT 004049C5	
004049C0	.	MOV BYTE PTR SS:[ESP+13],0	
004049C5	>	INC ESI	
004049C6	.	CMP ESI,0B	
004049C9	^	JL SHORT 00404990	

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.

Address	Hex	Disassembly	Comment
004049CB	>	PUSH 3EB	ButtonID = 3EB (1003.)
004049D0	.	PUSH EBX	hWnd
004049D1	.	CALL DWORD PTR DS:[&USER32.IsDlgButtonC	IsDlgButtonChecked
004049D7	.	CMP EAX,1	
004049DA	.	JNZ 00404A70	; Test si le radio bouton du niveau 3 est coche
004049E0	.	PUSH 14	Count = 14 (20.)
004049E2	.	LEA EDX, DWORD PTR SS:[ESP+1F8]	
004049E9	.	PUSH EDX	
004049EA	.	PUSH 3EC	Buffer
004049EF	.	PUSH EBX	ControlID = 3EC (1004.)
004049F6	.	CALL DWORD PTR DS:[&USER32.GetDlgItemTe	GetDlgItemTextA
004049F8	.	XOR EAX, EAX	; Debut du code pour le niveau 3
004049F8	.	JMP SHORT 00404A00	
004049FA	.	LEA EBX, DWORD PTR DS:[EBX]	
00404A00	>	MOVSX ECX, BYTE PTR SS:[ESP+EAX+1F4]	; Meme code que pour les deux precedents niveaux
00404A08	.	SUB ECX, 30	
00404A0B	.	MOV DWORD PTR SS:[ESP+EAX*4+160], ECX	
00404A12	.	INC EAX	
00404A13	.	CMP EAX, 0A	; Le serial doit faire 10 caracteres (comme le niveau 2)
00404A16	^	JL SHORT 00404A00	
00404A18	.	MOV ESI, 1	
00404A1D	.	LEA ECX, DWORD PTR DS:[ECX]	
00404A20	>	XOR EDI, EDI	
00404A22	.	XOR ECX, ECX	
00404A24	.	JMP SHORT 00404A30	
00404A26	.	LEA ESP, DWORD PTR SS:[ESP]	
00404A2D	.	LEA ECX, DWORD PTR DS:[ECX]	
00404A30	>	MOV EDX, DWORD PTR SS:[ESP+ECX*4+160]	
00404A37	.	MOV EAX, DWORD PTR SS:[ESP+ECX*4+160]	
00404A3E	.	ADD EAX, EDX	; On fait la somme des caracteres ecx et ecx + 1
00404A40	.	CDQ	
00404A41	.	IDIV ESI	; Divise la somme des caracteres par esi (compteur de boucle principale)
00404A43	.	ADD EDI, EDX	; EDI = la somme des restes
00404A45	.	INC ECX	
00404A46	.	CMP ECX, 9	
00404A49	^	JL SHORT 00404A30	
00404A4B	.	MOV EAX, EDI	; EAX = EDI = la somme des restes
00404A4D	.	CDQ	
00404A4E	.	MOV ECX, 0A	
00404A53	.	IDIV ECX	; Divise la somme des restes par 10
00404A55	.	CMP EDX, DWORD PTR SS:[ESP+ESI*4+C9]	; Test si le reste de la division = caractere du serial du niveau 3 en position esi
00404A5C	.	MOV DWORD PTR SS:[ESP+ESI*4+1AC], EDX	
00404A63	.	JE SHORT 00404A6A	
00404A65	.	MOV BYTE PTR SS:[ESP+13], 0	; Si cette variable est mise a 0 alors c'est perdu
00404A66	>	INC ESI	
00404A6B	.	CMP ESI, 0B	
00404A6E	^	JL SHORT 00404A20	
00404A70	.	CMP BYTE PTR SS:[ESP+13], 1	; Si cette variable est egale a 1 alors c'est gagne
00404A75	.	JNZ 00404CB2	
00404A78	.	MOV EDX, DWORD PTR DS:[414F8C]	TestYour.00400000
00404A81	.	PUSH EBP	IParam
00404A82	.	PUSH 004010E0	DlgProc = TestYour.004010E0
00404A87	.	PUSH EBP	hOwner
00404A88	.	PUSH 66	pTemplate = 66
00404A89	.	PUSH EDX	hInst => 00400000
00404A8E	.	CALL DWORD PTR DS:[&USER32.DialogBoxPar	DialogBoxParamA
00404A91	.	JMP 00404CB2	

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 {
Debut2:
        mov     esi, 1

        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 {
Debut3:
        mov     esi, 1

        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.

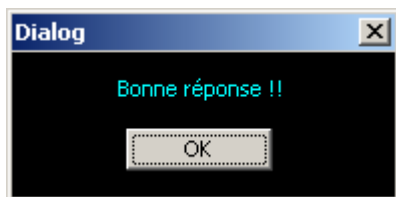
Les serials trouvés sont :

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 goûts).

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.