Bunny CrackMe 1

1. Première analyse

On commence par lancer pied, on voit direct que ça a été codé en Visual Basic.

On test le crackme, et on retiends plusieurs choses :

- Titre de la fenêtre principale : "CrackMe 1"
- Un champ de text et un bouton "Check Serial"
- Une nouvelle fenêtre indique qu'on se plante dans le serial. Titre : "Try again!" Texte : "Wrong Serial"

On continu dans la routine : on ouvre OllyDbg, on jette un oeil aux Referenced Text String. On trouve 5 lignes plutôt interressantes :

```
0040251C - UNICODE "12345"
0040256E - UNICODE "Good job"
00402580 - UNICODE "Well Done!"
004025C7 - UNICODE "Try again!"
004025D9 - UNICODE "Wrong Serial!"
```

On remarque directement le rapport avec ce qu'on a noté plus haut. On suppose que les lignes 0040256E et 00402580 sont donc respectivement le titre et le texte de la fenêtre goodBoy, et on a 12345 qui ressemble à un serial. En le testant, on peut voir qu'on avait raison sur les premières supposition.

On a trouvé le serial, on s'arrête là. C'est fini. Fini? On peut toujours fouiller un peu plus loin pour comprendre comment tout ça fonctionne.

2. Rappel des bases de l'ASM

Je vais passer très vite là desssus. C'est juste un petit rappel des instructions qu'on va croiser plus bas.

CMP

```
cmp EAX, EBX;
```

On positionne ZF à 1 en cas d'égalité. ZF c'est le Zero Flag, il permet de savoir si la dernière opération effectuée donnait un résultat nul (1) ou non (0). Donc, on déduit que CMP effectue une soustraction des deux paramètres mais sans en garder le résultat.

JE

```
JE 004089E1
```

Signifie "Jump if equal". Soit, saute si ZF=1.

MOV

```
MOV EAX, EBX
```

Place la valeur de EBX dans EAX.

LEA

```
LEA EAX, DWORD PTR:SS[EBP-18]
```

On ne peut passer que des valeur numériques par MOV, donc pas question de jouer avec l'offset. On utilise l'instruction LEA à la place. On place ici un double mot (Dword, 32 bits) venant de l'adresse EBP-18.

SUB

```
SUB EAX, EBX
```

Soustraction. EAX = EAX-EBX

SBB

```
SBB EAX, EBX
```

Soustraction en prenant en compte l'indicateur de retenue (CF). EAX = EAX - (EBX + CF)

3.Les mains dans le cambouis

A partir des chaines de caractères qui nous interressent, on retrouve la portion de code qui gère les good/bad boys. On remarque un saut en 40255B commandé par l'instruction de test en 40254C. Si SI et DI sont égaux, on saute à 4025B6. C'est très classique.

```
0040254C   . 66:3BF7        CMP SI,DI
0040254F   . 894D AC        MOV DWORD PTR SS:[EBP-54],ECX
00402552   . 8945 A4        MOV DWORD PTR SS:[EBP-5C],EAX
00402555   . 894D BC        MOV DWORD PTR SS:[EBP-44],ECX
00402558   . 8945 B4        MOV DWORD PTR SS:[EBP-4C],EAX
0040255B   . 74 59          JE SHORT defi_ach.004025B6
0040255D   . 8B35 74104000  MOV ESI,DWORD PTR DS:[<&MSVBVM60.__vbaVa>;  MSVBVM60.__vbaVarDup
00402563   . BB 08000000    MOV EBX,8
00402568   . 8D55 84        LEA EDX,DWORD PTR SS:[EBP-7C]
0040256B   . 8D4D C4        LEA ECX,DWORD PTR SS:[EBP-3C]
0040256E   . C745 8C 102340>MOV DWORD PTR SS:[EBP-74],defi_ach.00402>;  UNICODE "Good job"
00402575   . 895D 84        MOV DWORD PTR SS:[EBP-7C],EBX
00402578   . FFD6           CALL ESI                                   ;  <&MSVBVM60.__vbaVarDup>
0040257A   . 8D55 94        LEA EDX,DWORD PTR SS:[EBP-6C]
0040257D   . 8D4D D4        LEA ECX,DWORD PTR SS:[EBP-2C]
00402580   . C745 9C F42240>MOV DWORD PTR SS:[EBP-64],defi_ach.00402>;  UNICODE "Well Done!"
00402587   . 895D 94        MOV DWORD PTR SS:[EBP-6C],EBX
0040258A   . FFD6           CALL ESI
0040258C   . 8D4D A4        LEA ECX,DWORD PTR SS:[EBP-5C]
0040258F   . 8D55 B4        LEA EDX,DWORD PTR SS:[EBP-4C]
00402592   . 51             PUSH ECX
00402593   . 8D45 C4        LEA EAX,DWORD PTR SS:[EBP-3C]
00402596   . 52             PUSH EDX
00402597   . 50             PUSH EAX
00402598   . 8D4D D4        LEA ECX,DWORD PTR SS:[EBP-2C]
0040259B   . 6A 06          PUSH 6
0040259D   . 51             PUSH ECX
0040259E   . FF15 20104000  CALL DWORD PTR DS:[<&MSVBVM60.#595>]     ;  MSVBVM60.rtcMsgBox
004025A4   . 8D55 A4        LEA EDX,DWORD PTR SS:[EBP-5C]
004025A7   . 8D45 B4        LEA EAX,DWORD PTR SS:[EBP-4C]
004025AA   . 52             PUSH EDX
004025AB   . 8D4D C4        LEA ECX,DWORD PTR SS:[EBP-3C]
004025AE   . 50             PUSH EAX
004025AF   . 8D55 D4        LEA EDX,DWORD PTR SS:[EBP-2C]
004025B2   . 51             PUSH ECX
004025B3   . 52             PUSH EDX
004025B4   . EB 57          JMP SHORT defi_ach.0040260D
004025B6   > 8B35 74104000  MOV ESI,DWORD PTR DS:[<&MSVBVM60.__vbaVa>;  MSVBVM60.__vbaVarDup
004025BC   . BB 08000000    MOV EBX,8
```

```
004025C1   . 8D55 84        LEA EDX,DWORD PTR SS:[EBP-7C]
004025C4   . 8D4D C4        LEA ECX,DWORD PTR SS:[EBP-3C]
004025C7   . C745 8C 482340>MOV DWORD PTR SS:[EBP-74],defi_ach.00402>;  UNICODE "Try again!"
004025CE   . 895D 84        MOV DWORD PTR SS:[EBP-7C],EBX
004025D1   . FFD6           CALL ESI                                   ;  <&MSVBVM60.__vbaVarDup>
004025D3   . 8D55 94        LEA EDX,DWORD PTR SS:[EBP-6C]
004025D6   . 8D4D D4        LEA ECX,DWORD PTR SS:[EBP-2C]
004025D9   . C745 9C 282340>MOV DWORD PTR SS:[EBP-64],defi_ach.00402>;  UNICODE "Wrong Serial!"
```

On remonte encore un peu plus haut pour trouver ce que contiennent SI et DI. La portion de code étudiée est située juste au dessus de notre test entre SI et DI. Notre intuition ne nous avait pas trompé en pensant que "12345" était le bon code à entrer. Et, surprise, on retrouve ce code dans cette portion de code. Et, re-surprise, on croise une fonction vbaStrCmp en 402521 qui compare deux chaines de caractères entrées en arguments. Ces deux chaines sont passées dans les deux lignes qui précèdent la fonction. Cette fonction donne une valeur 1 à EAX si les chaines sont différentes et 0 si ce sont les deux mêmes.On imagine facilement que EAX contiends le serial que nous avons entré.

On passe le retour de la fonction dans ESI (402527). La ligne ECX récupère le serial entré par l'utilisateur(402529). Les lignes suivantes vont travailler sur ESI.

On prend l'opposé de la valeur stockée dans ESI. Petit rappel : en cas de valeur négatif, on inverse les bit et on ajout 1. De plus, on passe l'indicateur de signe (SF) à 1. Dans notre cas, on aura 0 ou 1.

- Si on a 0, ESI reste à 0 et CF est aussi à 0.
- Si on a 1, on se retrouve avec FFFFFFFF (vérifiez par vous même). De plus, CF passe à 1.

On passe à la ligne 40252E. Au final, cette instruction place la valeur du CF dans ESI.

A la fin de la ligne 402531, on aura donc ESI = FFFFFFFF si on a rentré le bon code et 0 si on a le mauvais.

```
00402518   > 8B45 E8        MOV EAX,DWORD PTR SS:[EBP-18]
0040251B   . 50             PUSH EAX
0040251C   . 68 E4224000    PUSH defi_ach.004022E4                     ;  UNICODE "12345"
00402521   . FF15 38104000  CALL DWORD PTR DS:[<&MSVBVM60.__vbaStrCm>;  MSVBVM60.__vbaStrCmp
00402527   . 8BF0           MOV ESI,EAX
00402529   . 8D4D E8        LEA ECX,DWORD PTR SS:[EBP-18]
0040252C   . F7DE           NEG ESI
0040252E   . 1BF6           SBB ESI,ESI
00402530   . 46             INC ESI
00402531   . F7DE           NEG ESI
00402533   . FF15 88104000  CALL DWORD PTR DS:[<&MSVBVM60.__vbaFreeS>;  MSVBVM60.__vbaFreeStr
00402539   . 8D4D E4        LEA ECX,DWORD PTR SS:[EBP-1C]
0040253C   . FF15 8C104000  CALL DWORD PTR DS:[<&MSVBVM60.__vbaFreeO>;  MSVBVM60.__vbaFreeObj
00402542   . B9 04000280    MOV ECX,80020004
00402547   . B8 0A000000    MOV EAX,0A
0040254C   . 66:3BF7        CMP SI,DI
```

Tout au long de cette séquence, EDI était à 0. On comprend mieux à présent l'origine des deux lignes :

```
0040254C   . 66:3BF7        CMP SI,DI
0040255B   . 74 59          JE SHORT defi_ach.004025B6     ; saute vers badBoy
```

On sauté pas mal de lignes qui ne nous interressaient pas. Pourquoi? Parce qu'elle n'influaient pas sur les registres qui nous interressaient (ex : les lignes 0040254F à 00402558). De plus, on peut déterminer leur fonction, et donc leur interet, par leur nom. vbaFreeStr et vbaFreeObj sont sans doutes de fonctions de libération de mémoires. Leur routine nous importe peut

Conclusion

C'est un crackme très simple et qui ne demande pas de grande connaissance pour le cracker. Mais étudier de plus près le code reste toujours un bon exercice pour les débutants.