Cours de Cracking

(13ième Partie)

Mon objectif : craquer le CD-CHECK de Myth II

1/ Les logiciels utiles pour ce cours

-> Un désassembleur : **W32dasm 8.93**
-> Un éditeur hexadécimal : **Winhex 10.2**

2/ Présentation

Quand vous lancez le jeu sans le CD, une image vous demande de l'inserer !Il n'y a pas de MessageBox, donc il faut procéder autrement que d'habitude... Pas de probleme, on va le cracker.

-> Faites deux copies de MythII.exe. Appellez la premiere WDASMyth2.exe et l'autre BAKUP.EXE...

-> Maintenant lancez W32DASM et desassemblez WDASMyth2.exe.

-> Quand le fichier est decompilé cliquez sur le bouton Imports Functions.

Pourquoi 'Imports Funtions' and pas 'String Data reference' comme d'hab ?

Parceque le message d'erreur n'est pas sous forme de MessageBox,mais est integré dans le jeu sous forme d'image. Dans les 'Imports functions', double-cliquez sur KERNEL32.GetDriveTypeA. Cette fonction determine si un support est un CD-Rom ou un Disque Dur,et permet donc de realiser un **Cd-CHECK** (en sus, il y a d'autres infos du style Nom_du_volume, taille_du_disque...).C'est donc cette fonction qui va nous interresser !

Au premier double-clic, vous tombez sur un petit bout de code... Ce n'est pas l'endroit qui nous interesse.

Pourquoi ?

Eh ben, sachez que la fonction GetDriveTypeA differencie le type de support suivant un code (de 1 à 7 je crois..) et le CD-Rom a le code '5'. Regardez donc en 4870D6... Si vous voyer une ligne comme ca a coté d'un GetDriveTypeA,vous pouvez être quasi sûr que c'est la routine du CD-Check !

Recliquez donc sur KERNEL32.GetDriveTypeA afin d'atterir sur cette partie du code :

```
* Referenced by a (U)nconditional or (C)onditional Jump at Address:
|:00487170(C)
* Possible Reference to String Resource ID=00001: "Myth II"
:004870B2 BA01000000              mov edx, 00000001
:004870B7 8ACB                    mov cl, bl
:004870B9 D3E2                    shl edx, cl
:004870BB 8AC3                    mov al, bl
:004870BD 0441                    add al, 41
:004870BF 88442410                mov byte ptr [esp+10], al
:004870C3 85D5                    test ebp, edx
:004870C5 0F84A0000000            je 0048716B       <-- jump vers le debut du CD-CHECK
:004870CB 8D442410                lea eax, dword ptr [esp+10]
:004870CF 50                      push eax


* Reference To: KERNEL32.GetDriveTypeA, Ord:00DFh
:004870D0 FF1520495B00            Call dword ptr [005B4920]
:004870D6 83F805                  cmp eax, 00000005     <-- Verifie qu'il y a le CD.
:004870D9 0F858C000000            jne 0048716B          <-- Si non, jump au debut du
'CD-Check'
:004870DF 8A0D103E5500            mov cl, byte ptr [00553E10]
:004870E5 33C0                    xor eax, eax          <-- Le test 'Cd-Check' a échoué
:004870E7 884C2418                mov byte ptr [esp+18], cl
:004870EB B93F000000              mov ecx, 0000003F
:004870F0 8D7C2419                lea edi, dword ptr [esp+19]
:004870F4 6A00                    push 00000000
:004870F6 F3                      repz
:004870F7 AB                      stosd
:004870F8 66AB                    stosw
:004870FA 6A00                    push 00000000
:004870FC 6A00                    push 00000000
:004870FE 6A00                    push 00000000
:00487100 6A00                    push 00000000
:00487102 AA                      stosb
:00487103 8D54242C                lea edx, dword ptr [esp+2C]
:00487107 68FF000000              push 000000FF
:0048710C 8D442428                lea eax, dword ptr [esp+28]
:00487110 52                      push edx
:00487111 50                      push eax
.
..
...
....

//Un peu plus loin dans le listing, vers la fin de la routine du 'CD-Check'...
```

```
* Referenced by a (U)nconditional or (C)onditional Jump at Address:
|:00487156(U)
:0048715D 85C0                  test eax, eax    <-- Test si eax=0
:0048715F 741B                  je 0048717C      <-- jump si vous avez le CD (eax=0)
:00487161 EB08                  jmp 0048716B     <-- Jump vers le 'CD-CHECk' (eax<>0)


* Referenced by a (U)nconditional or (C)onditional Jump at Addresses:
|:004870C5(C), :004870D9(C), :00487161(U)
:0048716B 43                    inc ebx       <- ici, ebx est incrementé a chaque
passage
:0048716C 6683FB19              cmp ebx, 0019 <- au bout de 20 passage on quitte le
Cd Check
:00487170 0F8E3CFFFFFF          jle 004870B2  <- Appelle GetDriveTypeA
:00487176 8B442414              mov eax, dword ptr [esp+14]
:0048717A EB05                  jmp 00487181  <-- Le 'CD-check' a échoué.


* Referenced by a (U)nconditional or (C)onditional Jump at Address:
|:0048715F(C)

* Possible Reference to String Resource ID=00001: "Myth II"
:0048717C B801000000            mov eax, 00000001            <-- Ok, le Cd est
reconnu !


* Referenced by a (U)nconditional or (C)onditional Jump at Address:
|:0048717A(U)
:........ ..........            ................. => vers la suite du listing...
```

Bon, maintenant je vous explique...

C'est assez simple : l'histoire commence en 487170 .... On arrive d'on-ne-sais-ou et voici donc que le programme va se demander si oui ou non vous avez le CD...Alors en 4870C5 il y a un jump vers la boucle du 'Cd-Check'. Une boucle ?? Ben oui, à chaque passage on incremente 'ebx', et au bout de 20 fois, on rend le 'verdict' :) Ne me demandez pas pourquoi ca se passe comme ça, c'est pas important de toute façon...

Donc, après avoir incrementé 'ebx', s'il est inferieur à 19 on repart en 487170.

Sauf que cette fois, le saut en 4870C5 ne se fera pas car 'ebp' n'est plus egale a 'edx'
(si vous regardez bien, on peut supposer que c'est 'ebp' qui change pendant la routine du CD-Check puisque 'edx' est initialisé a 1 des le debut de 487170...'ebp' doit etre modifier dans les lignes que je vous ai coupé, de toute facon c'est pas important).

Donc comme on saute pas ce coup si, on passe a GetDriveTypeA...Arrivé la, la fonction regarde s'il y a un CD ou non...Et donc on va se taper le saut car on a pas de CD ! Il faudra donc penser a remplacer ce 0F85 (jne) par un 0F84 (je) qui annulera ce saut...

Imaginons qu'on est annulé ce saut: on continue notre epopée et arrivé a un moment de la routine (dans les lignes que j'ai coupé), le programme determine si le support correspond à un CD original de Myth 2 (voir 48715D). On voit qu'il saute de facon inconditionnel en 48716B si le premier saut n'est pas executé. Or 48716B c'est la 'boucle des 20 passages', la ou il faut pas aller !!
On en deduit donc que le saut en 48715F, c'est la ou il faut aller :)

Alors la, c'est simple, il suffit de forcer le 741B (JE) en EB1B (JMP) (cf 48715F).

Bon, ben voila si on recapitule, il faut d'abord annuler le saut en 4870D9 puis forcer le saut en 48715F...

Voulez-vous une autre methode pour niquer le CD-Check ?? (ouiiiiiii) Mmm...mais alors c'est parceque c'est vous :) Bon, sans rire, regardez la difference entre le saut de 48715F et celui qui est en 48717A, a la fin du 'CD-Check'...Ben la seule difference c'est qu'avec le premier vous passez par une ligne supplementaire :

```
:0048717C B801000000 mov eax, 00000001
```

Donc, en fait, on pourrait tout simplement nopper le saut en 48717A et on irai toujours sur ce mov eax, 00000001...valavalou :)) Il suffirait alors de mettre un 9090 a la place du EB05 (cf 48717A) et le jeu serait cracké... Cool non ?? (Au fait, cette méthode marche avec toutes les versions de Myth 2.)

Bon, alors la vous venez d'apprendre a cracker un jeu commercial d'une grosse boîte...Vous voyez, c'est pas si dur :)) En tout cas, soyez sûr que c'est pas toujours aussi simple. Ceci dit, vous pouvez cracker pas mal de jeu comme ca :)

Bon, je vous laisse !! Happy New Year a tous !!