# Décoder les images de charme des CD-ROM de magazines

Introduction:

Si vous avez 30 ans ou plus, vous avez surement connu les magazines d'informatique fournis avec CD-ROM dans les kiosques de presse.
Sur ces CDs on retrouvait la plupart du temps une interface graphique qui servait de loader pour installer divers programmes stockés sur le CD.
Souvent les CDs étaient accompagnés d'une zone charme avec quelques sets de photos de strip-teases, et si vous vouliez voir le contenu un peu plus charnel, il fallait payer via le 3617, ou par téléphone.


Revenons à notre époque maintenant, 2021, le numéro de tel n'est plus attribué et on peut oublier le minitel. On va donc devoir trouver par nous même un moyen d'accéder au reste du prOn chiffré stocké sur le cd.

Les outils

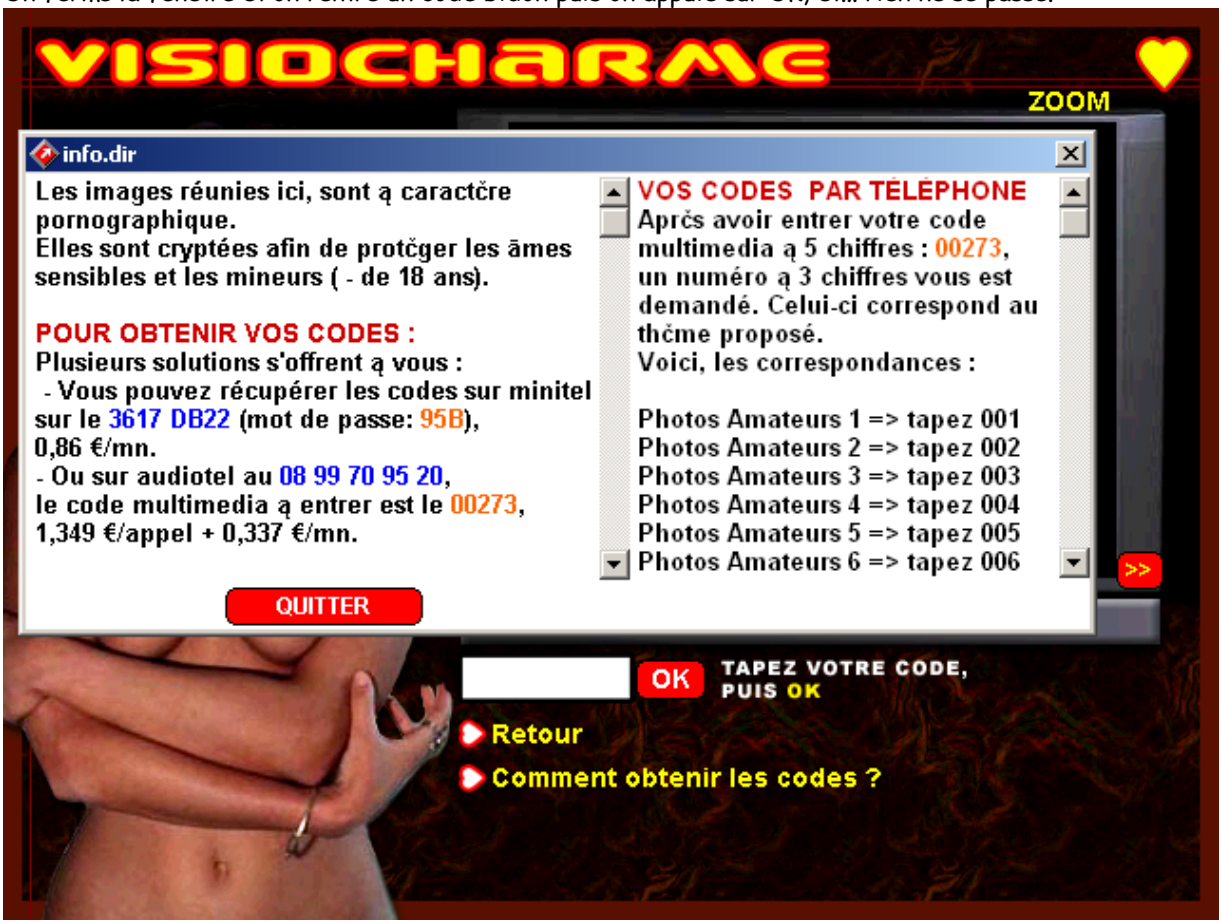
Pour appliquer ce tutoriel il vous faudra...

• OllyDbg v2.01 (http://Ollydbg.de/)

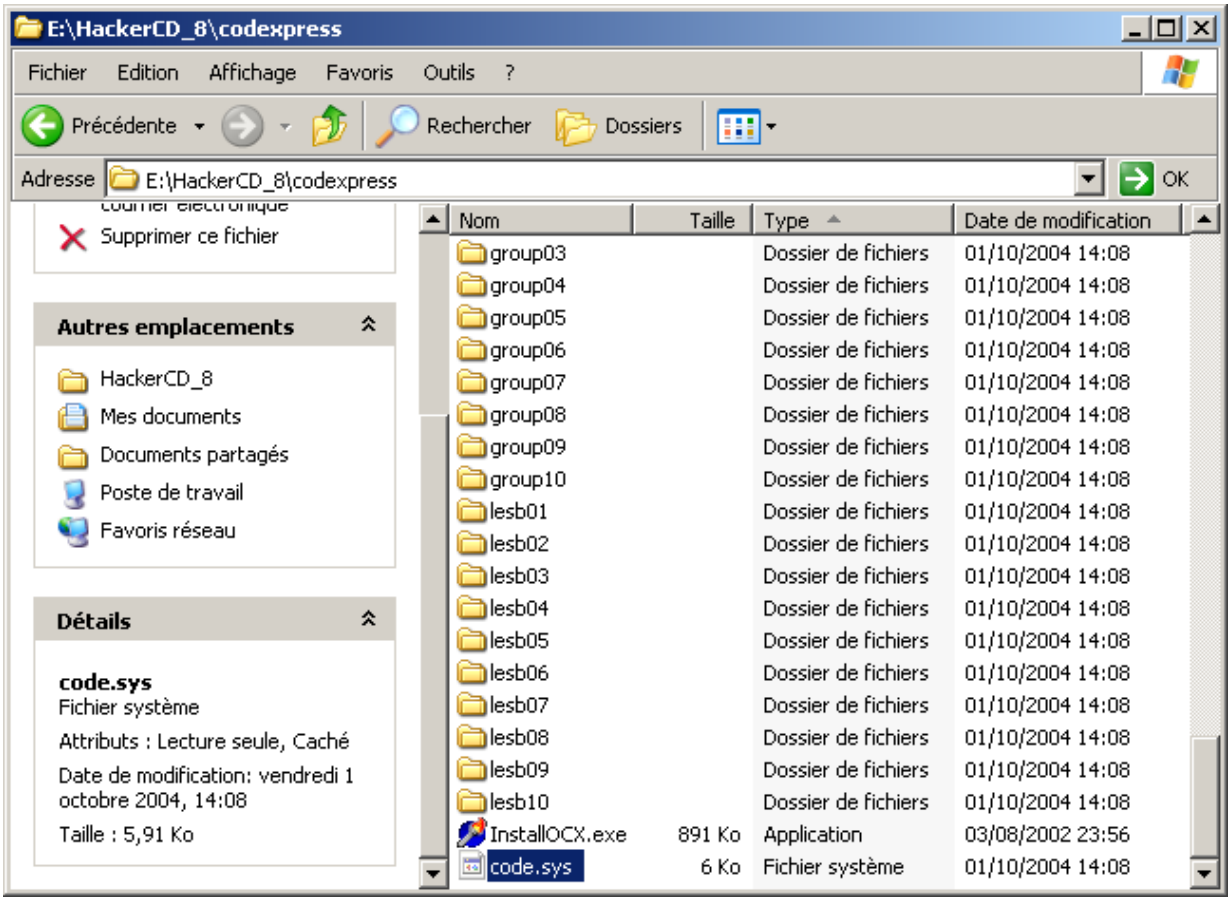
• Hacker CD 8 (https://archive.org/details/hackercd-8 - 472.8MB)
• PC Pirate 22 (https://archive.org/details/pcpr-22 - 653.1MB)

I: Démarrage:

On va commencer par voir à quoi ça ressemble, et après on ira explorer la galette.
On va dans l'espace charme, un disclaimer nous demande de confirmer notre âge en tapant 'Oui', puis on arrive sur 'Visocharme'
Un genre de nag-screen surgit par-dessus l'interface de visiocharme pour nous informer de comment obtenir un code.
On remarquera aussi l'icône de la fenêtre qui laisse pensé qu'il y-a une technologie Macromedia (maintenant Adobe) derrière l'application.
On ferme la fenêtre et on rentre un code bidon puis on appuie sur OK, et... rien ne se passe.



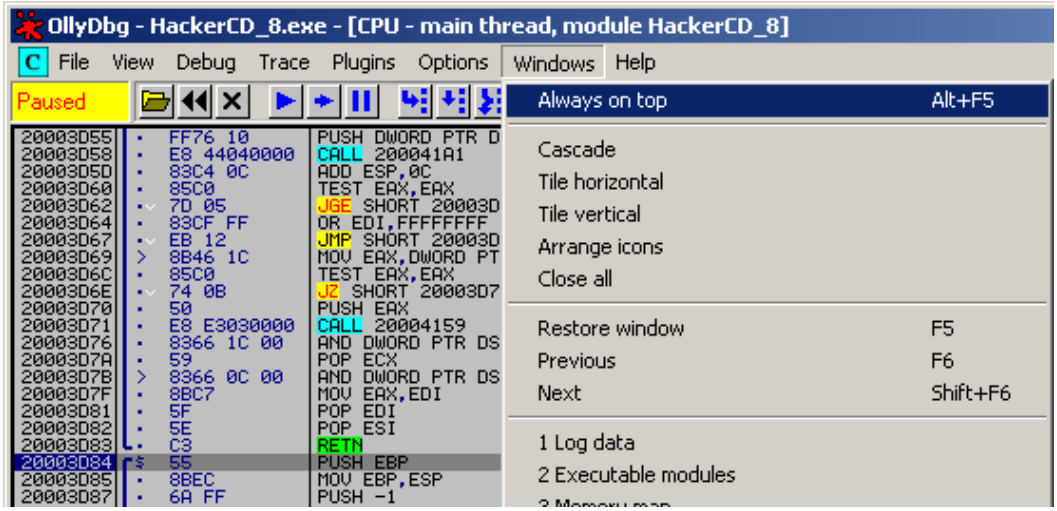

Parcourons le CD-ROM pour voir comment tout ça est construit.
À la racine on retrouve notre loader "HackerCD_8.exe" et a coté des fichiers en .DXR, on a donc du Macromedia Projector.
Le dossier 'codexpress' semble contenir les photos de l'espace charme.
On y retrouve aussi un fichier "code.sys" avec l'attribut 'caché' ainsi que InstallOCX.exe, surement pour installer les dépendances du lecteur d'image.



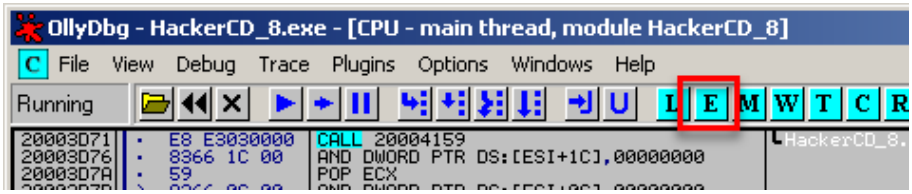

Pas d'exécutable pour visiocharme, celui-ci doit être intégré dans l'exécutable principal ou dans un des fichiers .dxr
Le fichier "code.sys" et chiffré, les photos aussi. Sous un éditeur hexa on notera que la structure des premiers octets sont identique entre les photos.
Ça me laisse penser que ça pourrait être un XOR ou quelque chose du genre. (Ça se retrouvait beaucoup à cette époque.)..
On verra bien.

II: Trouver la routine de vérification

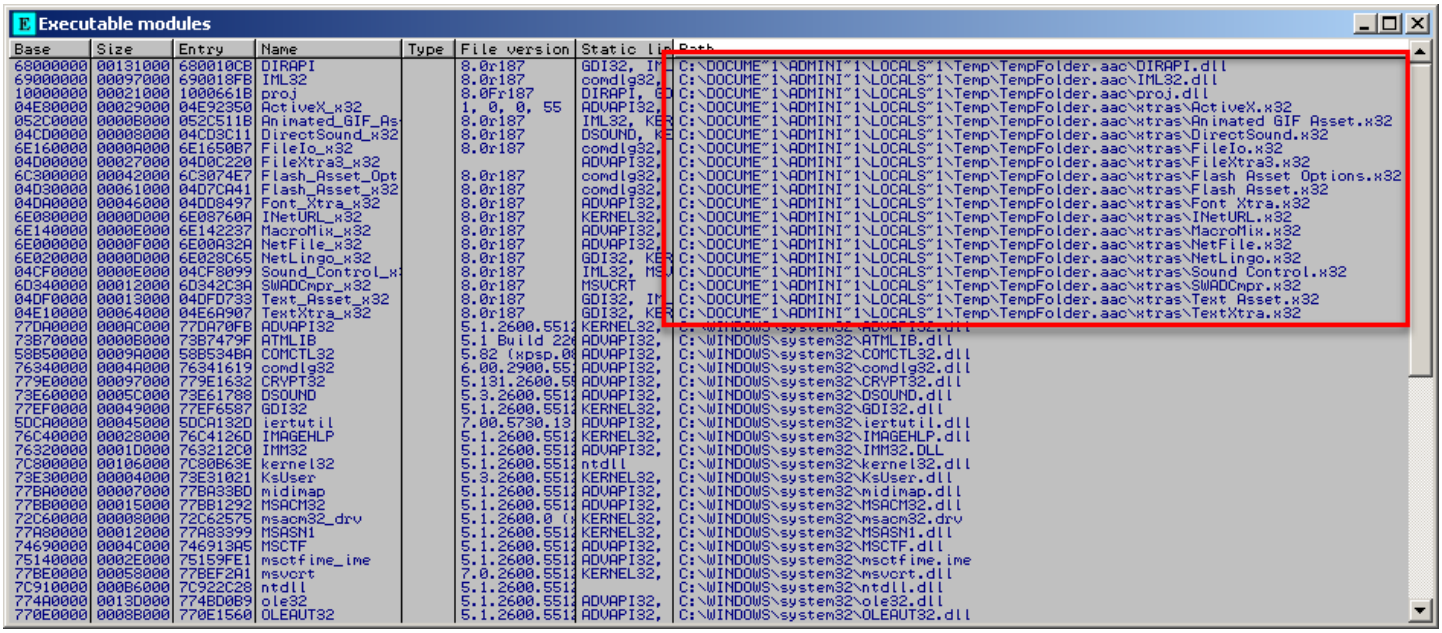
On lance le débogueur puis on charge HackerCD_8.exe, vu que l'application utilise tout l'écran, on va faire ALT+F5 ou bien depuis le menu du débogueur: Windows>Alway on top.
Histoire d'avoir notre débogueur toujours au premier plan, même si Alt+Tab va bien aussi pour passer de l'un a l'autre, y'a deux écoles.



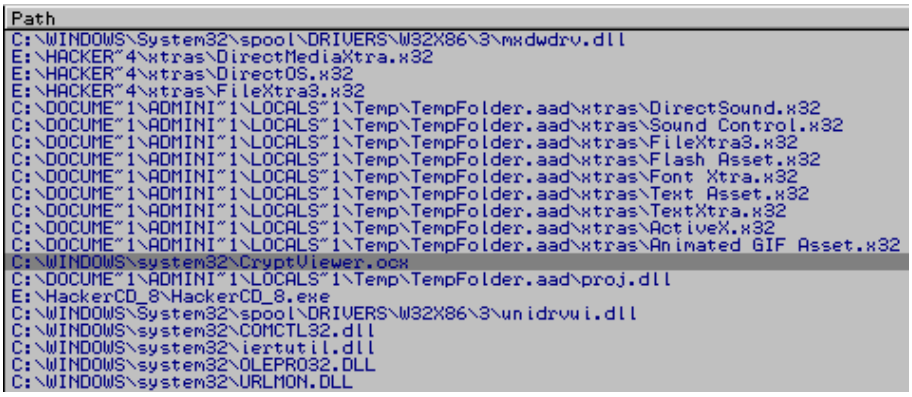

Une fois tranquille avec ça on lance complètement le programme avec F9, puis on bascule sur les modules avec Alt+E, ou bien depuis le menu.



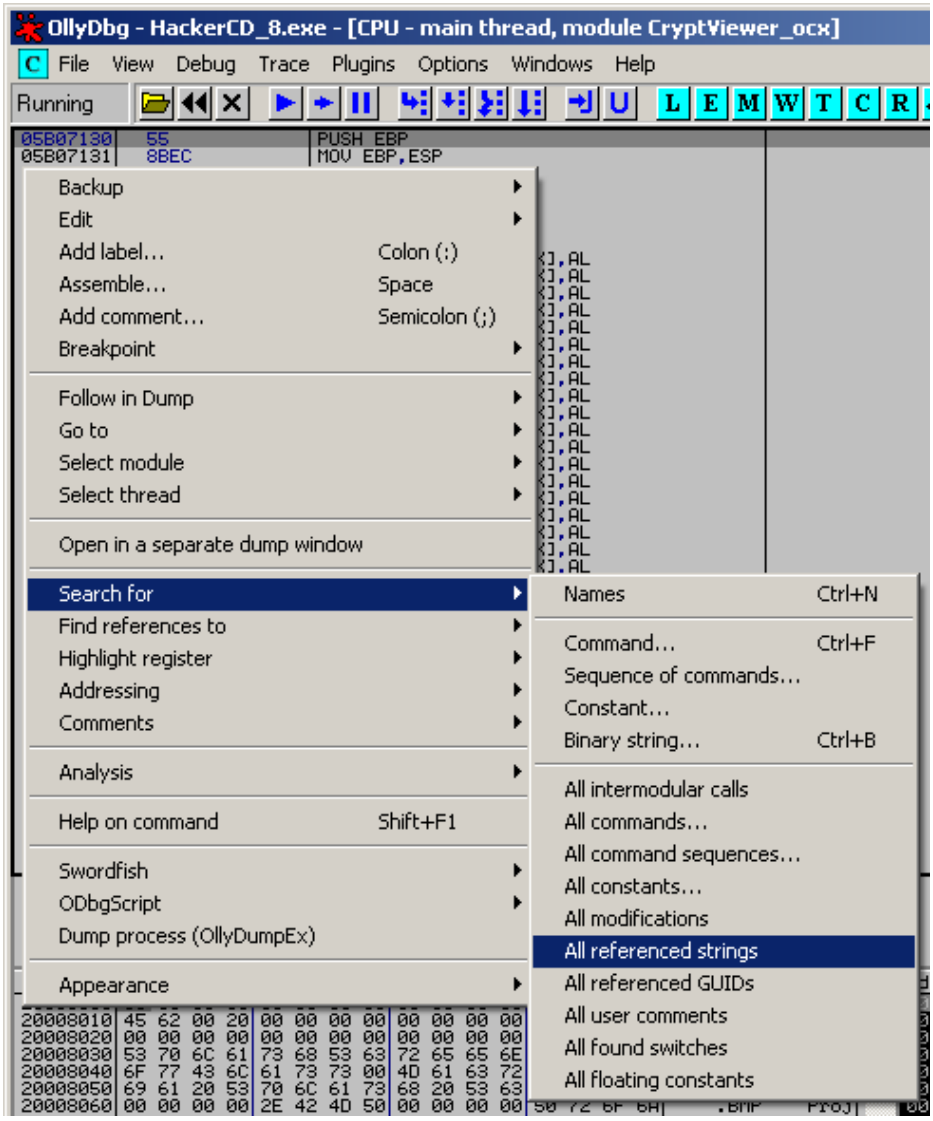

Macromedia projector a unpack sur notre disque dans un dossier temporaire tout sont bazar pour géré le son, les images, le texte...




On retourne sur l'application puis on va sur visiocharme, ensuite on retourne dans olly voir les modules si y'a du neuf.
Et effectivement, y'a de nouvelles choses, notamment "CryptViewer.ocx" exécuté depuis.. /system32/ ?!
Sans doute que "InstallOCX.exe" a été exécuté en caché.
Vraiment les mecs on honte de rien a installé des ocx dans un répertoire système sans en informé l'utilisateur...

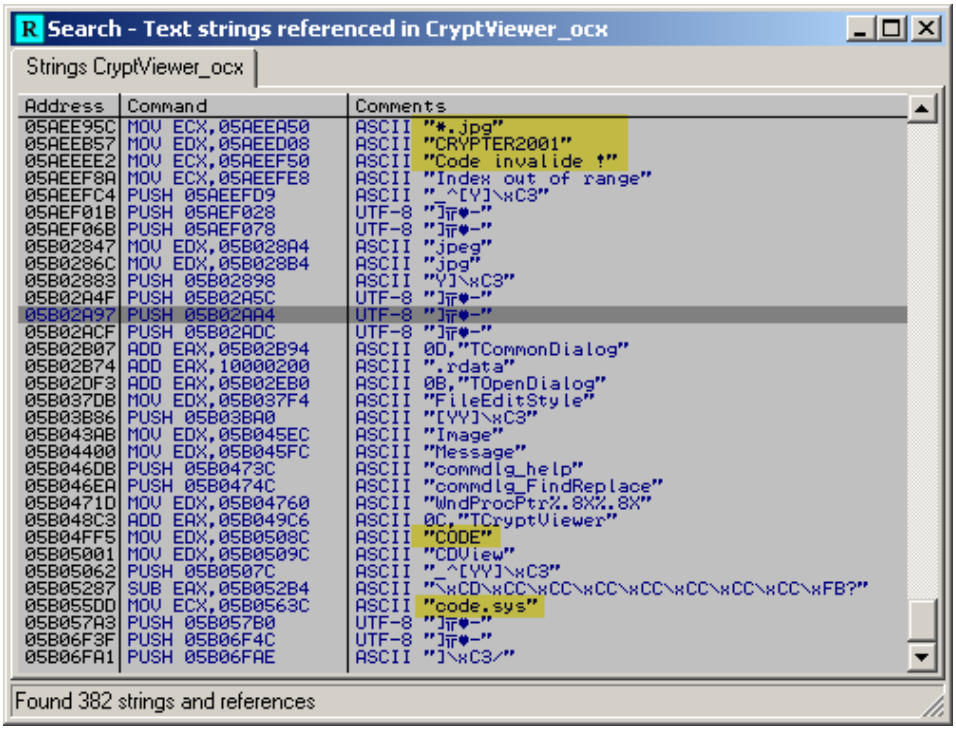



Bon.. voyont voir ce fichier qui semble prometteur vu sont nom, peut être que ça nous éclairera sur le chiffrement des images.
Allez hop, on double click sur la ligne "CryptViewer.ocx" dans les modules et on se retrouve sur ça plage.
On lance un scan de la plage pour y voir plus clair avec Ctrl+A ou bien: clic droit>Analysis>Analyse code
Une fois ça fait on va regarder les strings pour se faire une idée de ce fichier.
Clic droit>Search for>All referenced strings




Les strings nous confirment qu'on est dans la bonne direction.

```
ASCII "*.jpg"
ASCII "CRYPTER2001"
ASCII "Code invalide !"
ASCII "CODE"
ASCII "code.sys"
```




Aller hop, on double-clic sur "Code invalide !" et on arrive sur ça position.
On remonte un peut et on place un breakpoint au début de ça procédure avec F2, ensuite on retourne sur visiocharm rentré un code bidon puis on valide..
Le débogueur break!
Et ce qu'on est déjà trop bas dans le code? il n'y a plus qu'a tracer pour le savoir.

```
05AEEE5C  $  55           PUSH EBP
05AEEE5D  .  8BEC         MOV EBP,ESP
05AEEE5F  .  83C4 F4      ADD ESP,-0C
05AEEE62  .  53           PUSH EBX
05AEEE63  .  56           PUSH ESI
05AEEE64  .  57           PUSH EDI
05AEEE65  .  8955 FC      MOV DWORD PTR SS:[LOCAL.1],EDX
05AEEE68  .  8BD8         MOV EBX,EAX
05AEEE6A  .  8B45 FC      MOV EAX,DWORD PTR SS:[LOCAL.1]
05AEEE6D  .  E8 9A50FBFF  CALL 05AA3F0C
05AEEE72  .  33C0         XOR EAX,EAX
05AEEE74  .  55           PUSH EBP
05AEEE75  .  68 37EFAE05  PUSH 05AEEF37
05AEEE7A  .  64:FF30      PUSH DWORD PTR FS:[EAX]
05AEEE7D  .  64:8920      MOV DWORD PTR FS:[EAX],ESP              Installs SE handler 5AEEF37
05AEEE80  .  33C0         XOR EAX,EAX
05AEEE82  .  8945 F8      MOV DWORD PTR SS:[LOCAL.2],EAX
05AEEE85  .  8B43 14      MOV EAX,DWORD PTR DS:[EBX+14]
05AEEE88  .  8B78 08      MOV EDI,DWORD PTR DS:[EAX+8]
05AEEE8B  .  4F           DEC EDI
05AEEE8C  .  85FF         TEST EDI,EDI
05AEEE8E  .v 7C 4C        JL SHORT 05AEEEDC
05AEEE90  .  47           INC EDI
05AEEE91  .  33F6         XOR ESI,ESI
05AEEE93  >  8BD6         MOV EDX,ESI
05AEEE95  .  8B43 14      MOV EAX,DWORD PTR DS:[EBX+14]
05AEEE98  .  E8 8738FCFF  CALL 05AB2724                           CryptViewer_ocx.05AB2724
05AEEE9D  .  8B15 E0E2AE0 MOV EDX,DWORD PTR DS:[5AEE2E0]
05AEEEA3  .  E8 E440FBFF  CALL 05AA2F8C
05AEEEA8  .  8945 F4      MOV DWORD PTR SS:[LOCAL.3],EAX
05AEEEAB  .  8B45 F4      MOV EAX,DWORD PTR SS:[LOCAL.3]
05AEEEAE  .  8B40 04      MOV EAX,DWORD PTR DS:[EAX+4]
05AEEEB1  .  8B40 04      MOV EAX,DWORD PTR DS:[EAX+4]
05AEEEB4  .  8B55 FC      MOV EDX,DWORD PTR SS:[LOCAL.1]
05AEEEB7  .  E8 AC4FFBFF  CALL 05AA3E68                           CryptViewer_ocx.05AA3E68
05AEEEBC  .v 75 1A        JNZ SHORT 05AEEED8
05AEEEBE  .  8BD6         MOV EDX,ESI
05AEEEC0  .  8B43 14      MOV EAX,DWORD PTR DS:[EBX+14]
05AEEEC3  .  E8 5C38FCFF  CALL 05AB2724                           CryptViewer_ocx.05AB2724
05AEEEC8  .  8B15 E0E2AE0 MOV EDX,DWORD PTR DS:[5AEE2E0]
05AEEECE  .  E8 B940FBFF  CALL 05AA2F8C
05AEEED3  .  8945 F8      MOV DWORD PTR SS:[LOCAL.2],EAX
05AEEED6  .v EB 04        JMP SHORT 05AEEEDC
05AEEED8  >  46           INC ESI
05AEEED9  .  4F           DEC EDI
05AEEEDA  .^ 75 B7        JNZ SHORT 05AEEE93
05AEEEDC  >  837D F8 00   CMP DWORD PTR SS:[LOCAL.2],0
05AEEEE0  .v 75 16        JNE SHORT 05AEEEF8
05AEEEE2  .  B9 50EFAE05  MOV ECX,05AEEF50                        ASCII "Code invalide !"
05AEEEE7  .  B2 01        MOV DL,1
05AEEEE9  .  A1 7CE8AE05  MOV EAX,DWORD PTR DS:[5AEE87C]          ASCII 0C,"EInvalidCode"
05AEEEEE  .  E8 C5ACFBFF  CALL 05AA9BB8                           CryptViewer_ocx.05AA9BB8
05AEEEF3  .  E8 7046FBFF  CALL 05AA3568
05AEEEF8  >  8D43 0C      LEA EAX,[EBX+0C]
05AEEEFB  .  8B55 FC      MOV EDX,DWORD PTR SS:[LOCAL.1]
05AEEEFE  .  E8 2D4CFBFF  CALL 05AA3B30
05AEEF03  .  8B45 F8      MOV EAX,DWORD PTR SS:[LOCAL.2]
05AEEF06  .  8B40 04      MOV EAX,DWORD PTR DS:[EAX+4]
05AEEF09  .  8B40 08      MOV EAX,DWORD PTR DS:[EAX+8]
05AEEF0C  .  8943 18      MOV DWORD PTR DS:[EBX+18],EAX
05AEEF0F  .  8B45 F8      MOV EAX,DWORD PTR SS:[LOCAL.2]
05AEEF12  .  8B40 08      MOV EAX,DWORD PTR DS:[EAX+8]
05AEEF15  .  8943 1C      MOV DWORD PTR DS:[EBX+1C],EAX
05AEEF18  .  33D2         XOR EDX,EDX
05AEEF1A  .  8BC3         MOV EAX,EBX
05AEEF1C  .  E8 3F000000  CALL 05AEEF60
05AEEF21  .  33C0         XOR EAX,EAX
05AEEF23  .  5A           POP EDX
05AEEF24  .  59           POP ECX
05AEEF25  .  59           POP ECX
05AEEF26  .  64:8910      MOV DWORD PTR FS:[EAX],EDX
05AEEF29  .  68 3EEFAE05  PUSH 05AEEF3E                           Entry point
05AEEF2E  >  8D45 FC      LEA EAX,[EBP-4]
05AEEF31  .  E8 A64BFBFF  CALL 05AA3ADC                           CryptViewer_ocx.05AA3ADC
05AEEF36  .v C3           RETN                                    Jump to 5AEEF3E
```

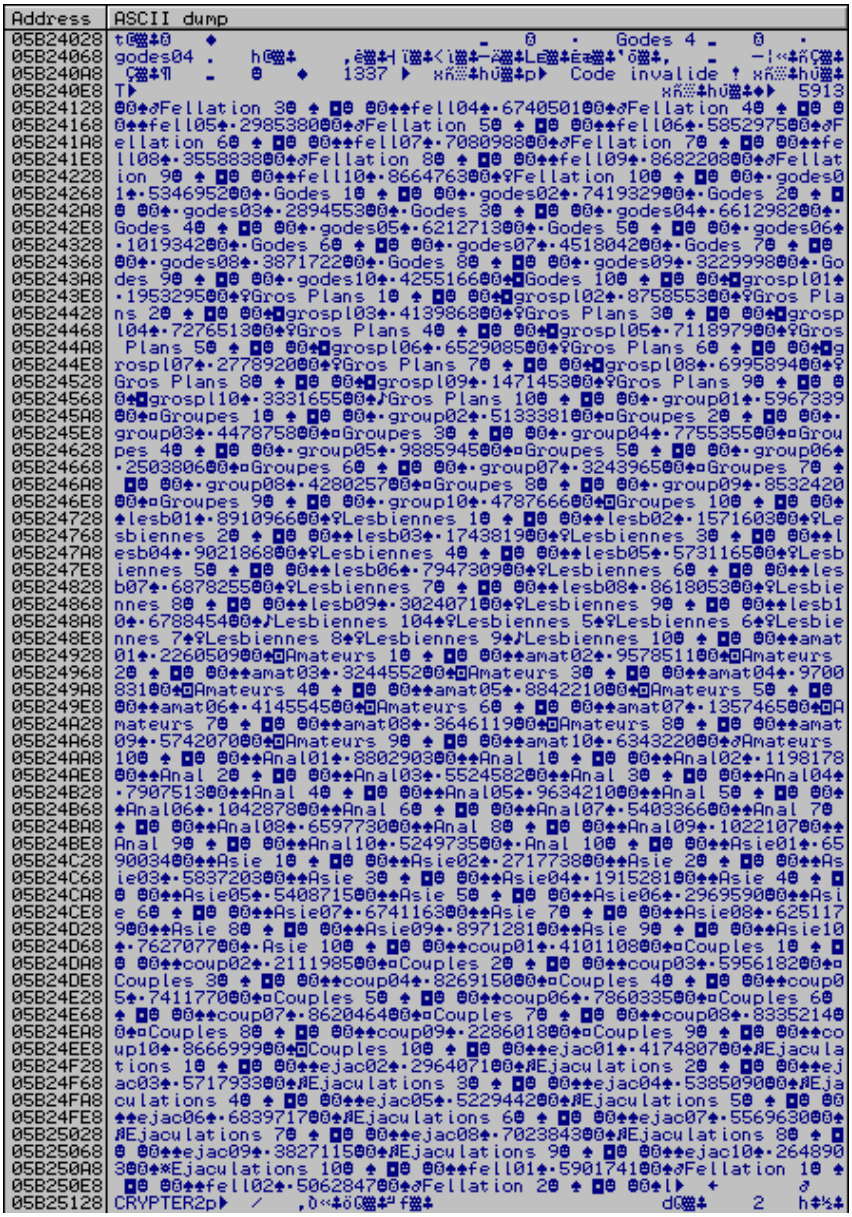
III: Comprendre la routine de vérification

On trace tranquillement avec Step over (F8)
On observe des mouvements de notre code entre les registres et des buffers puis on arrive dans une boucle.
Un string qui ressemble a un code et envoyer dans EAX, sur la ligne suivante notre code bidon et transféré d'un buffer a EDX, puis on entre dans un CALL.

```
05AEEE93  >  8BD6         MOV EDX,ESI
05AEEE95  .  8B43 14      MOV EAX,DWORD PTR DS:[EBX+14]
05AEEE98  .  E8 8738FCFF  CALL 05AB2724                           CryptViewer_ocx.05AB2724
05AEEE9D  .  8B15 E0E2AE0 MOV EDX,DWORD PTR DS:[5AEE2E0]
05AEEEA3  .  E8 E440FBFF  CALL 05AA2F8C
05AEEEA8  .  8945 F4      MOV DWORD PTR SS:[LOCAL.3],EAX
05AEEEAB  .  8B45 F4      MOV EAX,DWORD PTR SS:[LOCAL.3]
05AEEEAE  .  8B40 04      MOV EAX,DWORD PTR DS:[EAX+4]
05AEEEB1  .  8B40 04      MOV EAX,DWORD PTR DS:[EAX+4]            ASCII "6154855"
05AEEEB4  .  8B55 FC      MOV EDX,DWORD PTR SS:[LOCAL.1]
05AEEEB7  .  E8 AC4FFBFF  CALL 05AA3E68                           CryptViewer_ocx.05AA3E68
05AEEEBC  .v 75 1A        JNZ SHORT 05AEEED8
05AEEEBE  .  8BD6         MOV EDX,ESI
05AEEEC0  .  8B43 14      MOV EAX,DWORD PTR DS:[EBX+14]
05AEEEC3  .  E8 5C38FCFF  CALL 05AB2724                           CryptViewer_ocx.05AB2724
05AEEEC8  .  8B15 E0E2AE0 MOV EDX,DWORD PTR DS:[5AEE2E0]
05AEEECE  .  E8 B940FBFF  CALL 05AA2F8C
05AEEED3  .  8945 F8      MOV DWORD PTR SS:[LOCAL.2],EAX
05AEEED6  .v EB 04        JMP SHORT 05AEEEDC
05AEEED8  >  46           INC ESI
05AEEED9  .  4F           DEC EDI
05AEEEDA  .^ 75 B7        JNZ SHORT 05AEEE93
```

Nous aussi on va rentrer dans ce call avec Step Into (F7) une fois dans le call, pas de surprise, EAX et comparer a EDX.

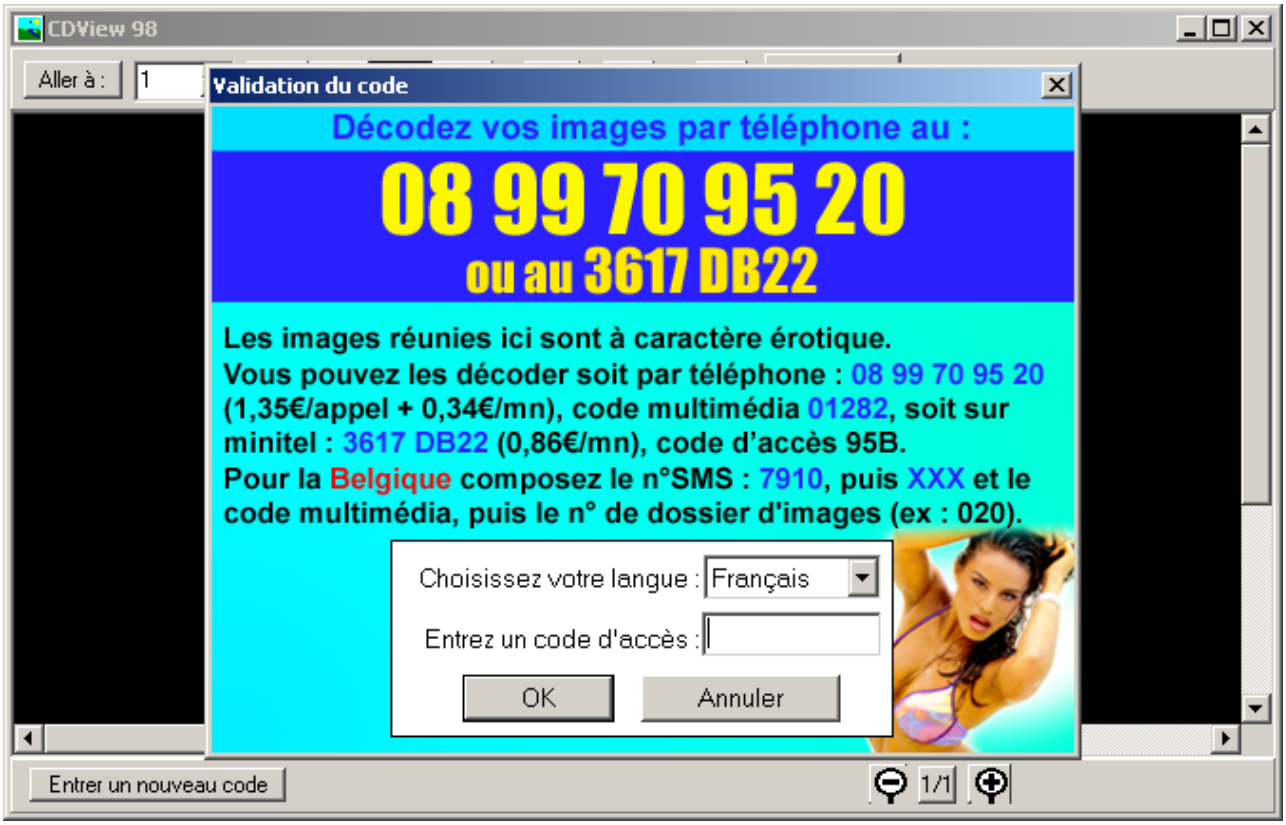

On continue a tracer pour sortir du call, puis la boucle reloop, ça recommence, mais cette fois avec un autre serial.
On va regarder dans la fenêtre de dump l'adresse du serial, vu qu'il est envoyé via un pointeur qui fait EAX+4, il y en a certainement d'autres.
Et effectivement il y en a plein. On se balade un peu dans le dump et voilà.




Ça fait un paquet de code, 1 par set.
Il n'y a plus qu'a testé ça dans l'interface voir ce qu'il se passe, on essaye un code un hazard..
Et... Oh yeah boy!


Voilà, donc c'est simplement de la protection par serial, ou les bons serials sont stockés dans le fichier 'code.sys' sans doute déchiffré puis comparés.

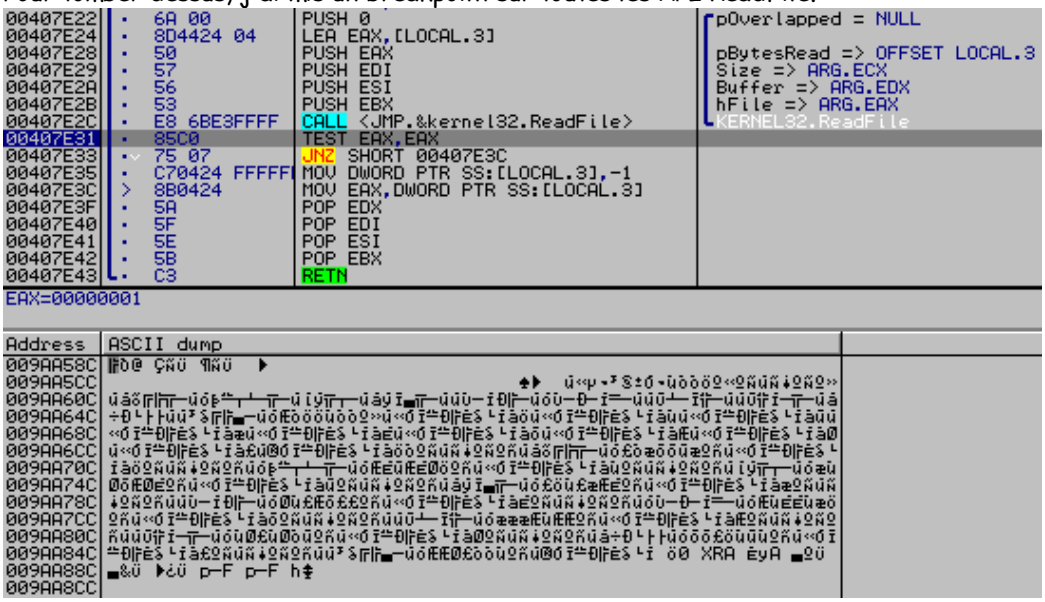
Du coup par curiosité j'ai testé avec un CD un peu plus récent. (PC Pirate 22)
Dans celui-ci, un exécutable additionnel et fournit avec: 'viewer.exe' dans le même dossier on retrouve aussi un fichier "code.sys" avec attribut caché.



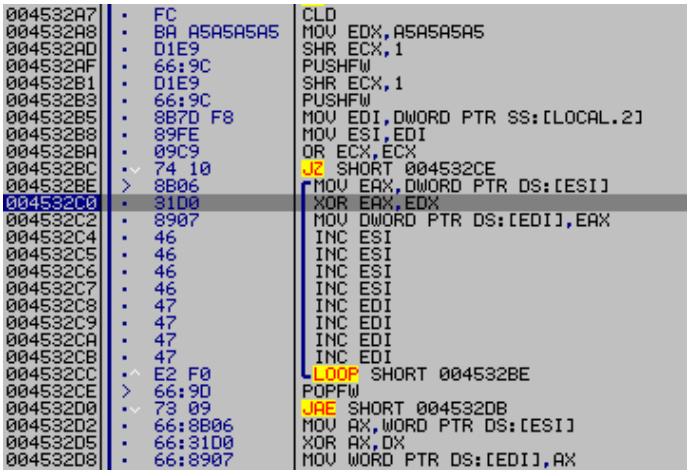
La routine et également assez reconnaissable a ce que l'on a vu précédemment.
On a une fonctionnalité en plus: au bout de trois mauvaises tentatives un appel sur PostQuitMessage et effectuer pour terminer le processus.

```
00468C5C  $  55              PUSH EBP
00468C5D  .  8BEC            MOV EBP,ESP
00468C5F  .  33C9            XOR ECX,ECX
00468C61  .  51              PUSH ECX
00468C62  .  51              PUSH ECX
00468C63  .  51              PUSH ECX
00468C64  .  51              PUSH ECX
00468C65  .  51              PUSH ECX
00468C66  .  51              PUSH ECX
00468C67  .  51              PUSH ECX
00468C68  .  53              PUSH EBX
00468C69  .  56              PUSH ESI
00468C6A  .  57              PUSH EDI
00468C6B  .  8945 FC         MOV DWORD PTR SS:[LOCAL.1],EAX
00468C6E  .  33C0            XOR EAX,EAX
00468C70  .  55              PUSH EBP
00468C71  .  68 948E4600     PUSH 00468E94
00468C76  .  64:FF30         PUSH DWORD PTR FS:[EAX]
00468C79  .  64:8920         MOV DWORD PTR FS:[EAX],ESP          Installs SE handler 468E94
00468C7C  .  8B45 FC         MOV EAX,DWORD PTR SS:[LOCAL.1]
00468C7F  .  C680 5803000(   MOV BYTE PTR DS:[EAX+358],1
00468C86  .  33D2            XOR EDX,EDX
00468C88  .  55              PUSH EBP
00468C89  .  68 DA8D4600     PUSH 00468DDA
00468C8E  .  64:FF32         PUSH DWORD PTR FS:[EDX]
00468C91  .  64:8922         MOV DWORD PTR FS:[EDX],ESP          Installs SE handler 468DDA
00468C94  .  8B45 FC         MOV EAX,DWORD PTR SS:[LOCAL.1]
00468C97  .  8B80 CC02000(   MOV EAX,DWORD PTR DS:[EAX+2CC]
00468C9D  .  33D2            XOR EDX,EDX
00468C9F  .  E8 440DFCFF     CALL 004299E8                       Cviewer.004299E8
00468CA4  .  C745 F4 0500(   MOV DWORD PTR SS:[LOCAL.3],5
00468CAB  .  B0 01           MOV AL,1
00468CAD  .v E9 B6000000     JMP 00468D68
00468CB2  >  A1 70B64600     MOV EAX,DWORD PTR DS:[46B670]       ASCII "0\xAA\x9A"
00468CB7  .  8B00            MOV EAX,DWORD PTR DS:[EAX]
00468CB9  .  8B10            MOV EDX,DWORD PTR DS:[EAX]
00468CBB  .  FF92 CC00000(   CALL DWORD PTR DS:[EDX+0CC]
00468CC1  .  C645 FB 00      MOV BYTE PTR SS:[LOCAL.2+3],0
00468CC5  .  A1 70B64600     MOV EAX,DWORD PTR DS:[46B670]       ASCII "0\xAA\x9A"
00468CCA  .  8B00            MOV EAX,DWORD PTR DS:[EAX]
00468CCC  .  83B8 2C02000(   CMP DWORD PTR DS:[EAX+22C],2
00468CD3  .v 0F84 9D000000   JE 00468D76
00468CD9  .  8B45 FC         MOV EAX,DWORD PTR SS:[LOCAL.1]
00468CDC  .  8B80 2803000(   MOV EAX,DWORD PTR DS:[EAX+328]
00468CE2  .  8B70 08         MOV ESI,DWORD PTR DS:[EAX+8]
00468CE5  .  85F6            TEST ESI,ESI
00468CE7  .v 7E 77           JLE SHORT 00468D60
00468CE9  .  BB 01000000     MOV EBX,1
00468CEE  >  8BD3            MOV EDX,EBX
00468CF0  .  4A              DEC EDX
00468CF1  .  8B45 FC         MOV EAX,DWORD PTR SS:[LOCAL.1]
00468CF4  .  8B80 2803000(   MOV EAX,DWORD PTR DS:[EAX+328]
00468CFA  .  E8 2564FAFF     CALL 0040F124                       Cviewer.0040F124
00468CFF  .  8B15 2433450(   MOV EDX,DWORD PTR DS:[453324]
00468D05  .  E8 DAA1F9FF     CALL 00402EE4
00468D0A  .  8BF8            MOV EDI,EAX
00468D0C  .  8D55 F0         LEA EDX,[LOCAL.4]
00468D0F  .  A1 70B64600     MOV EAX,DWORD PTR DS:[46B670]       ASCII "0\xAA\x9A"
00468D14  .  8B00            MOV EAX,DWORD PTR DS:[EAX]
00468D16  .  8B80 C402000(   MOV EAX,DWORD PTR DS:[EAX+2C4]
00468D1C  .  E8 D70DFCFF     CALL 00429AF8
00468D21  .  8B55 F0         MOV EDX,DWORD PTR SS:[LOCAL.4]
00468D24  .  8B47 04         MOV EAX,DWORD PTR DS:[EDI+4]
00468D27  .  8B40 04         MOV EAX,DWORD PTR DS:[EAX+4]        ASCII "7011200"
00468D2A  .  E8 65AFF9FF     CALL 00403C94
00468D2F  .v 75 2B           JNZ SHORT 00468D5C
00468D31  .  8BD3            MOV EDX,EBX
00468D33  .  4A              DEC EDX
00468D34  .  8B45 FC         MOV EAX,DWORD PTR SS:[LOCAL.1]
00468D37  .  8B80 2803000(   MOV EAX,DWORD PTR DS:[EAX+328]
00468D3D  .  E8 E263FAFF     CALL 0040F124                       Cviewer.0040F124
00468D42  .  8B15 2433450(   MOV EDX,DWORD PTR DS:[453324]
00468D48  .  E8 97A1F9FF     CALL 00402EE4
00468D4D  .  8B55 FC         MOV EDX,DWORD PTR SS:[LOCAL.1]
00468D50  .  8982 2003000(   MOV DWORD PTR DS:[EDX+320],EAX
00468D56  .  C645 FB 01      MOV BYTE PTR SS:[LOCAL.2+3],1
00468D5A  .v EB 04           JMP SHORT 00468D60
00468D5C  >  43              INC EBX
00468D5D  .  4E              DEC ESI
00468D5E  .^ 75 8E           JNZ SHORT 00468CEE
00468D60  >  8A45 FB         MOV AL,BYTE PTR SS:[LOCAL.2+3]
00468D63  .  34 01           XOR AL,01
00468D65  .  FF4D F4         DEC DWORD PTR SS:[LOCAL.3]
00468D68  >  84C0            TEST AL,AL
00468D6A  .v 74 0A           JZ SHORT 00468D76
00468D6C  .  837D F4 00      CMP DWORD PTR SS:[LOCAL.3],0
00468D70  .^ 0F8F 3CFFFFFF   JG 00468CB2
00468D76  >  807D FB 00      CMP BYTE PTR SS:[LOCAL.2+3],0
00468D7A  .v 74 4A           JE SHORT 00468DC6
00468D7C  .  8B45 FC         MOV EAX,DWORD PTR SS:[LOCAL.1]
00468D7F  .  8B80 2003000(   MOV EAX,DWORD PTR DS:[EAX+320]
00468D85  .  8B50 08         MOV EDX,DWORD PTR DS:[EAX+8]
00468D88  .  8B45 FC         MOV EAX,DWORD PTR SS:[LOCAL.1]
00468D8B  .  8B80 CC02000(   MOV EAX,DWORD PTR DS:[EAX+2CC]
00468D91  .  8B80 EC01000(   MOV EAX,DWORD PTR DS:[EAX+1EC]
00468D97  .  8B08            MOV ECX,DWORD PTR DS:[EAX]
00468D99  .  FF51 08         CALL DWORD PTR DS:[ECX+8]
00468D9C  .  8B45 FC         MOV EAX,DWORD PTR SS:[LOCAL.1]
00468D9F  .  8B80 CC02000(   MOV EAX,DWORD PTR DS:[EAX+2CC]
00468DA5  .  33D2            XOR EDX,EDX
00468DA7  .  E8 50A8FBFF     CALL 004235FC                       Cviewer.004235FC
00468DAC  .  8B45 FC         MOV EAX,DWORD PTR SS:[LOCAL.1]
00468DAF  .  E8 50F9FFFF     CALL 00468704                       Cviewer.00468704
00468DB4  .  8B45 FC         MOV EAX,DWORD PTR SS:[LOCAL.1]
00468DB7  .  E8 30EDFFFF     CALL 00467AEC                       Cviewer.00467AEC
00468DBC  .  8B45 FC         MOV EAX,DWORD PTR SS:[LOCAL.1]
00468DBF  .  E8 D4F7FFFF     CALL 00468598
00468DC4  .v EB 07           JMP SHORT 00468DCD
00468DC6  >  6A 00           PUSH 0                              ExitCode = 0
00468DC8  .  E8 EFD9F9FF     CALL <JMP.&user32.PostQuitMessage>  USER32.PostQuitMessage
00468DCD  >  33C0            XOR EAX,EAX
00468DCF  .  5A              POP EDX
00468DD0  .  59              POP ECX
00468DD1  .  59              POP ECX
00468DD2  .  64:8910         MOV DWORD PTR FS:[EAX],EDX
00468DD5  .v E9 8D000000     JMP 00468E67
00468DDA  $^ E9 B1A4F9FF     JMP 00403290                        SE handling routine
```

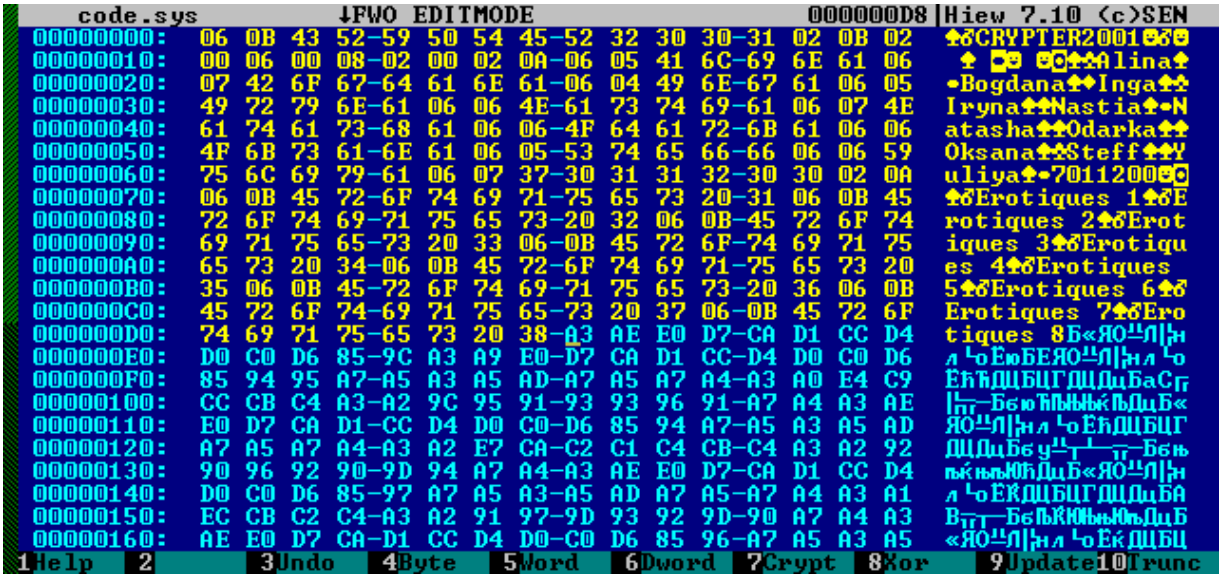
Du coup j'en profite pour regarder comment la routine de décodage fonctionne, et...
C'est du xor pour le code.sys.
Pour tomber dessus, j'ai mis un breakpoint sur toutes les API ReadFile.




Après avoir récupéré le contenu de code.sys via ReadFile, on tombe plus tard dans la routine de décodage du fichier.
On a un MOV EDX,A5A5A5A5 puis on entre dans une boucle de xor, qui au démarrage place les bytes a décoder dans EAX, puis il effectue un XOR des bytes avec EDX (A5).




Vous pouvez utiliser Hiew, et mettre un masque xor avec A5 pour décoder les fichiers code.sys.



J'ai testé avec Hacker CD 8, PC Pirate 22, aini que l'encyclopédie des utilitaires PC 19, la clé était la même.
Pour le décodage des images et bien… c'est la même clé/routine qui est utilisée.
Maintenant que l'on sait ça, on peut écrire un décodeur, ça nous évitera de devoir rentrer tous les codes et d'utiliser leur visionneuse d'image.

Ah, chose rigolote en plus: on a un serial maitre qui déverrouille toutes les images dans PC Pirate 22.
Chose qu'on ne retrouve pas pour Hacker CD 8.

IV: Faire un décodeur d'image

ça change un peut des keygens :)
Alors, on construit une boucle avec FindFirstFile/FindNextFile et un filtre "*.jpg"
Puis dans cette boucle, on a juste a ajouté après FindFirstFile un call a une procédure de xor qui va traité le fichier puis le réenregistrer.
Ensuite il n'y a plus qu'à passer au fichier suivant avec FindNextFile.

xordecoder.asm:

```
.486
.model flat, stdcall
option casemap :none ; case sensitive

comment /*
Tested successfully with:
- ENCYCLO_UT_19_CD (L'encyclopédie des utilitaires PC 19)
- UTPM_3 (Utilitaire PC Pratique)
- HACKERCD_8 (Hacker CD 8)
- PC_PIRATE_5 (PC Pirate 5)
- PCPR_22 (PC Pirate 22)
- PCPR_28 (PC Pirate 28)
- JPMN_4 (Japan Mania 4)
/

include                 \masm32\include\windows.inc
include                 \masm32\include\user32.inc
include                 \masm32\include\kernel32.inc
include                 \masm32\macros\macros.asm

includelib              \masm32\lib\user32.lib
includelib              \masm32\lib\kernel32.lib

WndProc                 proto  :DWORD,:DWORD,:DWORD,:DWORD
List                    proto  :DWORD,:DWORD
ScanForParticularFiles  proto  :HWND
CryptFile               proto  :HWND
Crypt                   proto  :DWORD,:DWORD,:BYTE

.const
IDD_DIALOG              equ 1000
IDB_QUIT                equ 1001
IDB_CRYPT               equ 1002
IDC_GROUPRESULT         equ 1003
IDC_GROUPPATH           equ 1004
IDC_STATICFound         equ 1005
IDC_LISTBOXJPG          equ 1006
IDC_EDITPATH            equ 1007
IDC_COUNTER             equ 1008

.data
szTitle                 db "3617 DB22 pr0n Dcoder v0.1",0
szIDBCrypt              db "dÉCHIFFRER LES IMAGES *.jpg",0
szIDCGroupFound         db "iMAGES TRAiTÉES",0
szIDCGroupPath          db "cHEMiN DU DOSSiER CONTENANT LES iMAGES",0
szStaticFound           db "TOTAL:",0
szIDBQUit               db "QUiTTER",0
szIDCcounter            db "0",0

Dect                    db "%d",0
szFilter                db "*.jpg",0
cnt                     db 0
```

```
bKey                    db 0A5h ; ¥

.data?
hInstance               HINSTANCE ?
ThreadID                dd ?
CountJPG                dd ?
hthread                 dd ?
hFile                   dd ?
dwFileSize              dd ?
dwBytesDone             dd ?
hMemory                 dd ?
hBuffer                 dd ?
dpath                   db 256 dup (?)
hDir                    db 256 dup (?)
fpath                   db 256 dup (?)
cuntBuffer              db 8 dup(?)

.code
start:
    invoke GetModuleHandle, NULL
    mov hInstance,eax
    invoke DialogBoxParam,hInstance,IDD_DIALOG,NULL,addr WndProc,NULL
    invoke ExitProcess,eax

WndProc proc hWin:DWORD, uMsg:DWORD, wParam:DWORD, lParam:DWORD
    .if uMsg == WM_INITDIALOG
        ; Set the dialog controls texts. Done here in the code instead of resource
        ; file to reduce the required bytes (strings in the rc file are UNICODE not ANSI)
        invoke SetWindowText,hWin,addr szTitle
        invoke SetDlgItemText,hWin,IDB_CRYPT,addr szIDBCrypt
        invoke SetDlgItemText,hWin,IDC_GROUPRESULT,addr szIDCGroupFound
        invoke SetDlgItemText,hWin,IDC_GROUPPATH,addr szIDCGroupPath
        invoke SetDlgItemText,hWin,IDC_STATICFound,addr szStaticFound
        invoke SetDlgItemText,hWin,IDB_QUIT,addr szIDBQUit
        invoke GetCurrentDirectory,1024,addr hDir
        invoke SetDlgItemText,hWin,IDC_EDITPATH,addr hDir
        invoke SetDlgItemText,hWin,IDC_COUNTER,addr szIDCcounter
        xor eax, eax
        ret
    .elseif uMsg == WM_COMMAND
        .if wParam == IDB_CRYPT
            invoke RtlZeroMemory,addr dpath,sizeof dpath
            invoke GetDlgItemText,hWin,IDC_EDITPATH,addr dpath,sizeof dpath
            invoke SetCurrentDirectory,ADDR dpath
            mov CountJPG,0
            invoke wsprintf,addr cuntBuffer,addr Dect,CountJPG
            invoke SetDlgItemText,hWin,IDC_COUNTER,addr cuntBuffer
            invoke CreateThread,NULL,NULL,offset ScanForParticularFiles,hWin,NULL,ADDR ThreadID
            mov hthread,eax
        .elseif wParam == IDB_QUIT
            invoke EndDialog,hWin,0
        .endif
    .elseif uMsg == WM_CLOSE
        invoke EndDialog,hWin,0
    .endif
    xor eax, eax
     ret
WndProc endp

List proc hWin:HWND, pMsg:DWORD
    invoke SendDlgItemMessage,hWin,IDC_LISTBOXJPG,LB_ADDSTRING,0,pMsg
    invoke SendDlgItemMessage,hWin,IDC_LISTBOXJPG,WM_VSCROLL,SB_BOTTOM,0
    Ret
List EndP

ScanForParticularFiles proc hWnd:HWND
    LOCAL fnd      :WIN32_FIND_DATA
    LOCAL hFind    :DWORD
    invoke FindFirstFile,addr szFilter,addr fnd
    .if eax != INVALID_HANDLE_VALUE
        mov hFind, eax
        .repeat
            invoke GetCurrentDirectory,1024,ADDR hDir
```

```
            invoke lstrcat,addr fpath,addr hDir
            invoke lstrcat,addr fpath,chr$("\")
            invoke lstrcat,addr fpath,addr fnd.cFileName
            invoke List,hWnd,addr fpath
            inc CountJPG
            invoke wsprintf,addr cuntBuffer,addr Dect,CountJPG
            invoke SetDlgItemText,hWnd,IDC_COUNTER,addr cuntBuffer
            invoke CryptFile,hWnd
            invoke RtlZeroMemory,addr fpath,sizeof fpath
            inc cnt
            .if cnt == 25
                mov cnt,0
            .endif
            invoke FindNextFile,hFind,addr fnd
        .until eax == 0
        invoke FindClose,hFind
    .endif
    ret
ScanForParticularFiles endp

CryptFile proc hWnd:HWND
    invoke CreateFile,addr fpath,GENERIC_READ OR GENERIC_WRITE,0,0,OPEN_EXISTING,0,0
    mov hFile,eax
    .if eax == INVALID_HANDLE_VALUE
        jmp err0rz
    .else
        invoke GetFileSize,eax,0
        mov [dwFileSize],eax
        push eax
        invoke GlobalAlloc,GMEM_MOVEABLE,eax
        mov [hMemory],eax
        invoke GlobalLock,eax
        mov [hBuffer],eax
        push eax
        invoke ReadFile,[hFile],eax,dwFileSize,OFFSET dwBytesDone,0
        pop edx
        pop ecx
        mov ah,bKey
        invoke Crypt,[hBuffer],[dwFileSize],[bKey]
        invoke SetFilePointer,[hFile],0,0,FILE_BEGIN
        invoke WriteFile,[hFile],[hBuffer],[dwFileSize],OFFSET dwBytesDone,0
        invoke GlobalUnlock,[hMemory]
        invoke GlobalFree,[hMemory]
        invoke CloseHandle,[hFile]
    .endif
    ret
err0rz:
    invoke MessageBox,NULL,chr$("Error opening file!"),chr$("Error"),MB_ICONERROR
    ret
CryptFile endp

Crypt proc pszBuffer:DWORD, dwSize:DWORD, Key:BYTE
doxor:
    mov al,[edx+ecx]
    xor al,ah
    mov [edx+ecx],al
    dec ecx
    .if ecx == 0FFFFFFFFh ; -1
       ret
    .else
       jmp doxor
    .endif
Crypt endp

end start
```

xordecoder.rc:

```
;This Resource Script was generated by WinAsm Studio.

#define IDD_DIALOG 1000
#define IDB_QUIT 1001
#define IDB_CRYPT 1002
#define IDC_GROUPRESULT 1003
```

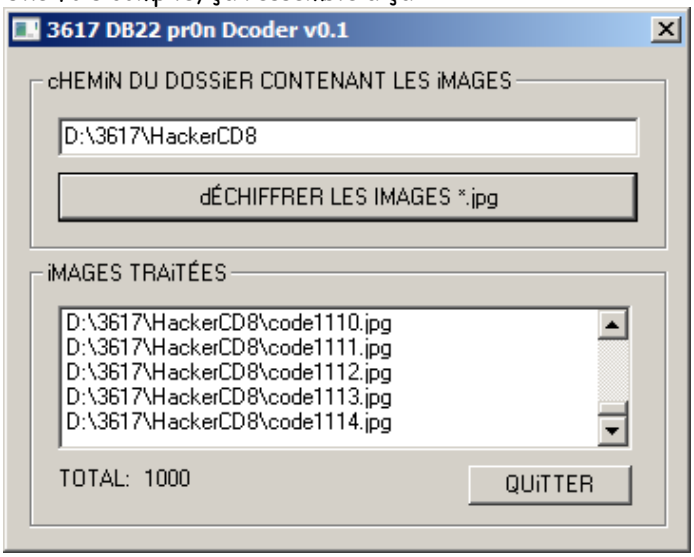
```
#define IDC_GROUPPATH 1004
#define IDC_STATICFound 1005
#define IDC_LISTBOXJPG 1006
#define IDC_EDITPATH 1007
#define IDC_COUNTER 1008

IDD_DIALOG DIALOGEX 0,0,237,161
FONT 8,"MS Sans Serif"
STYLE 0x10c80800
EXSTYLE 0x00000010
BEGIN
    CONTROL "",IDB_CRYPT,"Button",0x50010000,17,40,203,16,0x00000000
    CONTROL "",IDB_QUIT,"Button",0x50010000,161,135,57,13,0x00000000
    CONTROL "",IDC_GROUPRESULT,"Button",0x50000007,7,68,224,87,0x00000000
    CONTROL "",IDC_GROUPPATH,"Button",0x50000007,7,6,224,59,0x00000000
    CONTROL "",IDC_LISTBOXJPG,"ListBox",0x50010140,17,83,201,47,0x00000200
    CONTROL "",IDC_STATICFound,"Static",0x50000000,17,135,27,10,0x00000000
    CONTROL "",IDC_EDITPATH,"Edit",0x50010080,17,22,204,13,0x00000200
    CONTROL "",IDC_COUNTER,"Static",0x50000000,47,135,43,10,0x00000000
END
```

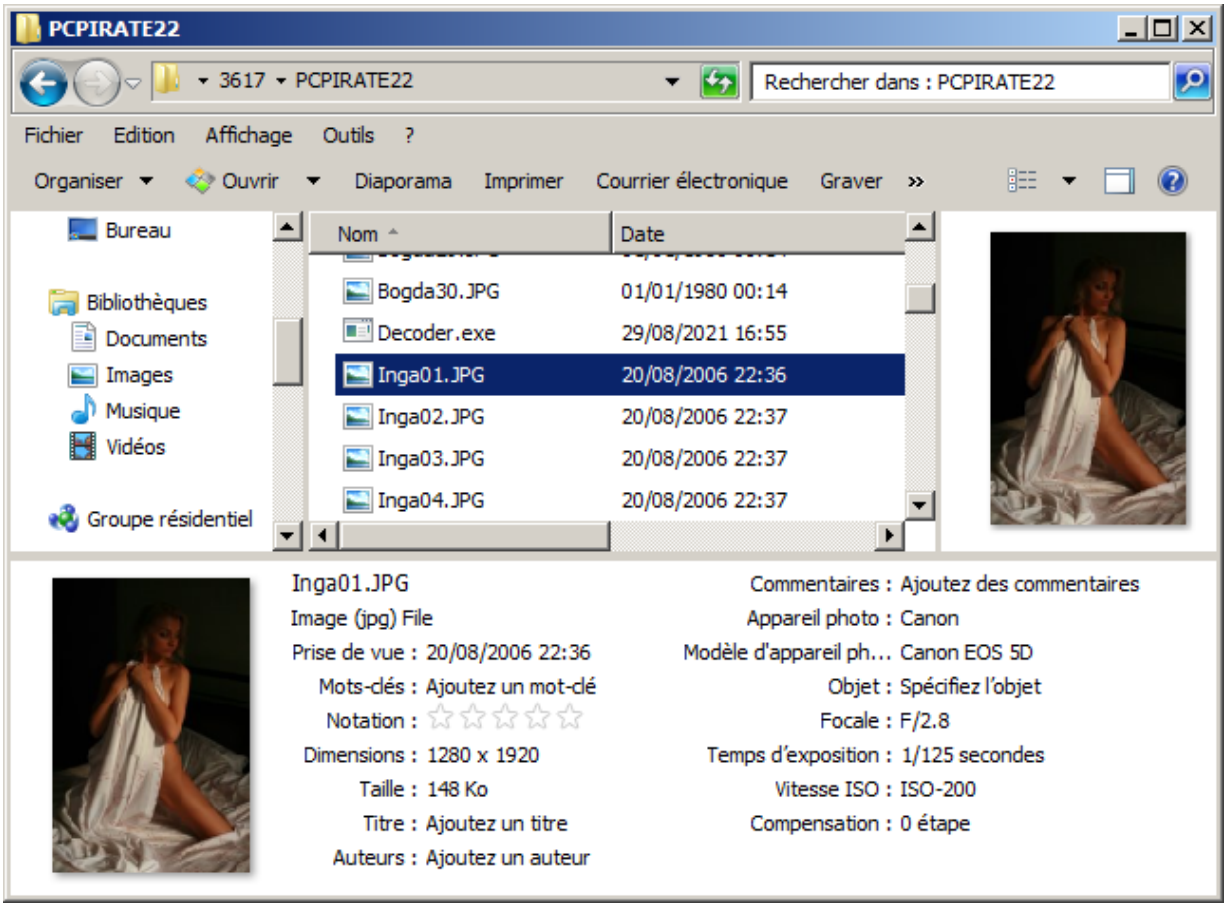
make.bat:

```
@echo off
\masm32\bin\rc /v xordecoder.rc
\masm32\bin\ml.exe /c /coff /Cp /nologo xordecoder.asm
\masm32\bin\link.exe /SUBSYSTEM:WINDOWS /RELEASE /VERSION:4.0 /OUT:Decoder.exe xordecoder.obj xordecoder.res
del xordecoder.res
del xordecoder.obj
pause
```

Une fois compilé, ça ressemble à ça:




Placer ce programme dans le même dossier contenant vos images et appuyer sur déchiffrer.
Il n'y a pas de récursivité pour les sous-dossiers.
Vous pouvez maintenant admirée les filles du CD.




Regrettable tout de même que pour de la presse Française tout les séance photos contenus dans ses CD sont des filles de l'Europe de l'Est.

Xylitol, 29/08/2021