

Les Articles | Non-Geek By Thxer

For Commu N-PN

IDA L'usine a Gaz

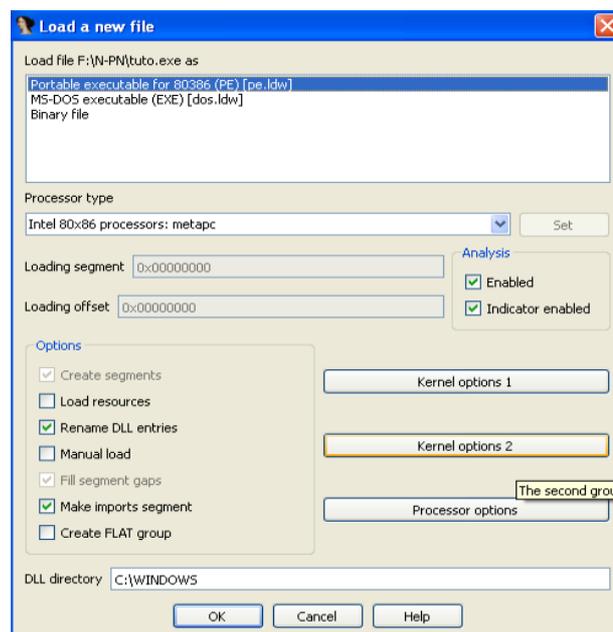
Intro

Il est 1:00 du mat', et suite à une suggestion de petit « tuto » sur ida je me tords dans mon lit, pleins d'idées ... ou plutôt ce tuto me prends la tête alors je m'y attelle.

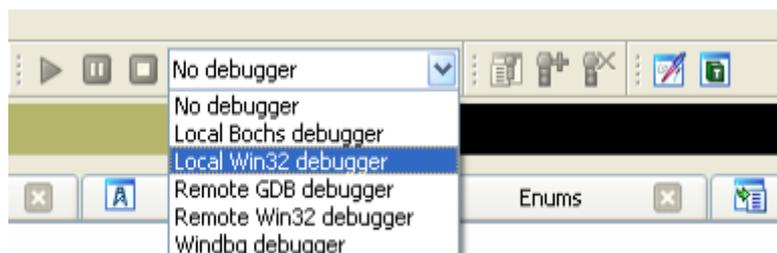
Alors IDA, certains disent « C'est une usine à gaz », et ben ... oui carrément, personnellement je pense que c'est la hotte du père Noël ! Je vous propose donc une petite visite guidée. (nocturne :()

Mettre en place le debugger

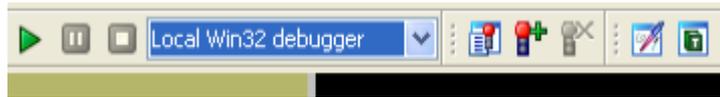
IDA c'est bien mais lorsqu'on ouvre notre premier PE, on ne peut pas le debug ensuite. Il faut choisir le bon debugger.



Ici On desass le .exe



Comme c'est un PE on choisit « Local Win32 debugger »



Yeah ! C'est vert on peut Debug !!!

Oui mais on peut aussi poser des Breakpoints (bp) car avant c'était pas possible.

Du coup pour poser un bp = F2.

La suite c'est comme pour OLLY :

F9 = Run , F8=Fonction suivante, F7= Pas à Pas (ça donne des crampes)

Là =>
J'ai posé un super
bp au pif sur un xor

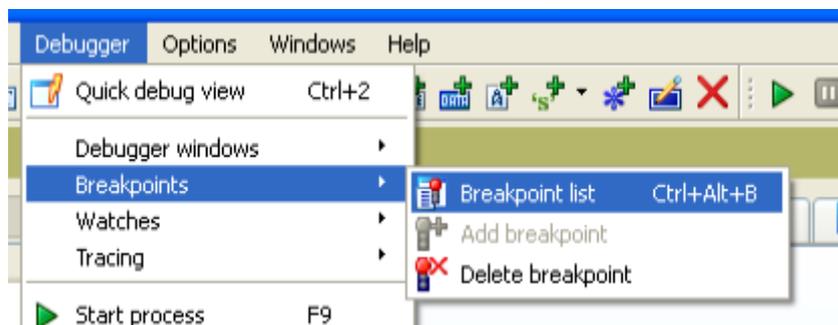
```
sub_4021E5 proc near
arg_0= dword ptr 4
push    edi
xor     ecx, ecx
mov     edi, [esp+4+arg_0]
not     ecx
xor     al, al
cld
repne  scasb
not     ecx
pop     edi
```

Bref quand c'est rouge y'a un bp.
(F2 pour enlever)

Vous voulez lister les Bp ?

Voilà

=>



Bon okey, jusque là ça reste ... basique

La Graph View !

C'est une des nombreuses qualités d'ida, le mode « graph »

En général vous êtes sur cette vue de base.

Mais le hic c'est que quand vous voulez vous balader dans le code bah ... vous savez plus comment revenir le (-) de OLLY ne marche pas.

Il suffit d'utiliser la touche « espace » pour basculer entre les vues.

Vue « Graph »



```
loc_40220E:  
push    eax  
xor     al, al  
mov     eax, [ebx+ecx]  
rol     eax, 4  
shl     eax, 2  
xor     eax, 13374266h  
push    ebx  
mov     ebx, [ebp+arg_4]  
mov     [ebx+ecx], eax  
mov     byte ptr [ebx+ecx+1], 0  
pop     ebx  
pop     eax  
add     ecx, 4  
cmp     ecx, eax  
jbe     short loc_40220E
```

Vue « Inside »

code: 0040220F	xor di, di
code: 00402211	mov eax, [ebx+ecx]
code: 00402214	rol eax, 4
code: 00402217	shl eax, 2
code: 0040221A	xor eax, 13374266h
code: 0040221F	push ebx
code: 00402220	mov ebx, [ebp+arg_4]
code: 00402223	mov [ebx+ecx], eax
code: 00402226	mov byte ptr [ebx+ecx+1], 0
code: 0040222B	pop ebx
code: 0040222C	pop eax
code: 0040222D	add ecx, 4
code: 00402233	cmp ecx, eax

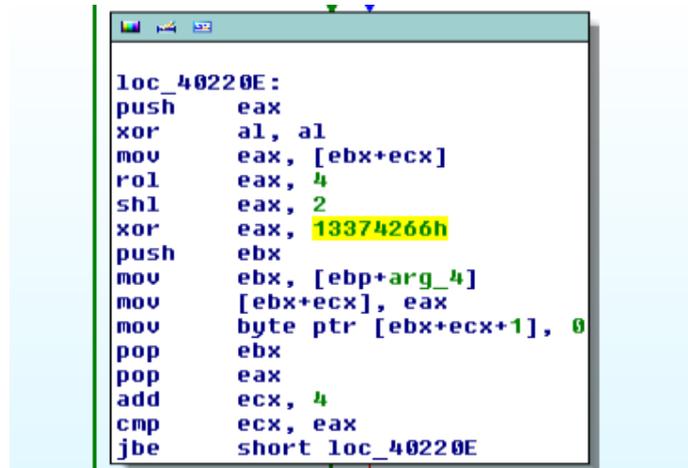
Pour rentrer dans un call il suffit de cliquer deux fois dessus.

On commence à toucher des choses cool non ?

Le Pseudo-Code

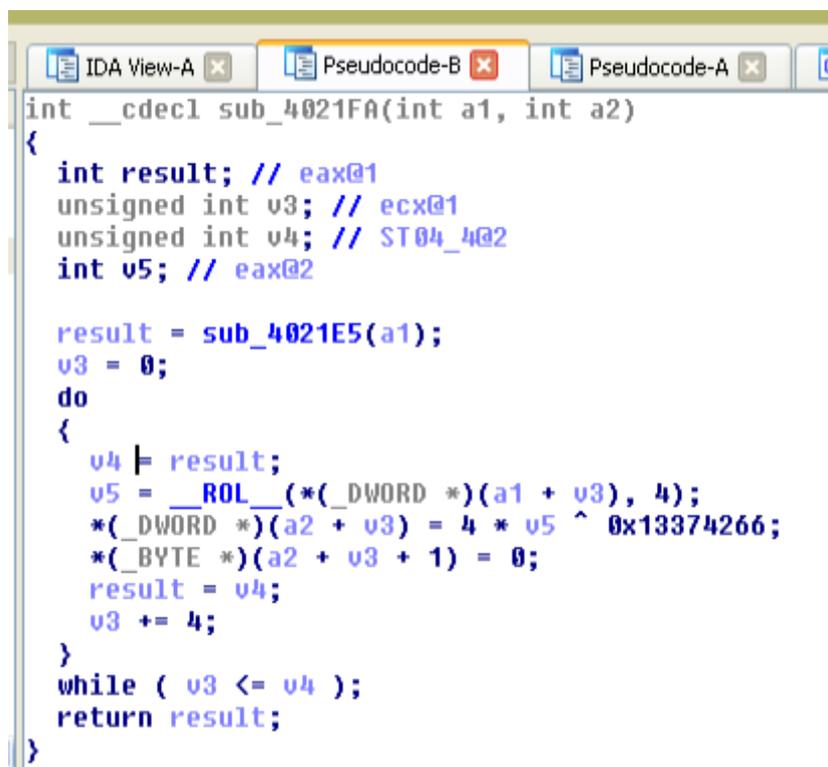
Parfois on se mélange un peu dans l'asm, et un pseudo code-c nous aidera à comprendre la mécanique interne, du coup IDA le fait pour nous.

Allez dans votre fonction.



```
loc_40220E :
push    eax
xor     al, al
mov     eax, [ebx+ecx]
rol     eax, 4
shl     eax, 2
xor     eax, 13374266h
push    ebx
mov     ebx, [ebp+arg_4]
mov     [ebx+ecx], eax
mov     byte ptr [ebx+ecx+1], 0
pop     ebx
pop     eax
add     ecx, 4
cmp     ecx, eax
jbe     short loc_40220E
```

Puis faire F5 :



```
int __cdecl sub_4021FA(int a1, int a2)
{
    int result; // eax@1
    unsigned int v3; // ecx@1
    unsigned int v4; // ST04_4@2
    int v5; // eax@2

    result = sub_4021E5(a1);
    v3 = 0;
    do
    {
        v4 = result;
        v5 = __ROL__((_DWORD *) (a1 + v3), 4);
        *(_DWORD *) (a2 + v3) = 4 * v5 ^ 0x13374266;
        *(_BYTE *) (a2 + v3 + 1) = 0;
        result = v4;
        v3 += 4;
    }
    while ( v3 <= v4 );
    return result;
}
```

Magique non ? Pour revenir ? => Cliquez sur IDA View-A

Je sais pas vous ... mais perso je ne veux pas m'arrêter là.

Le Renommage Ou Façonner son code !

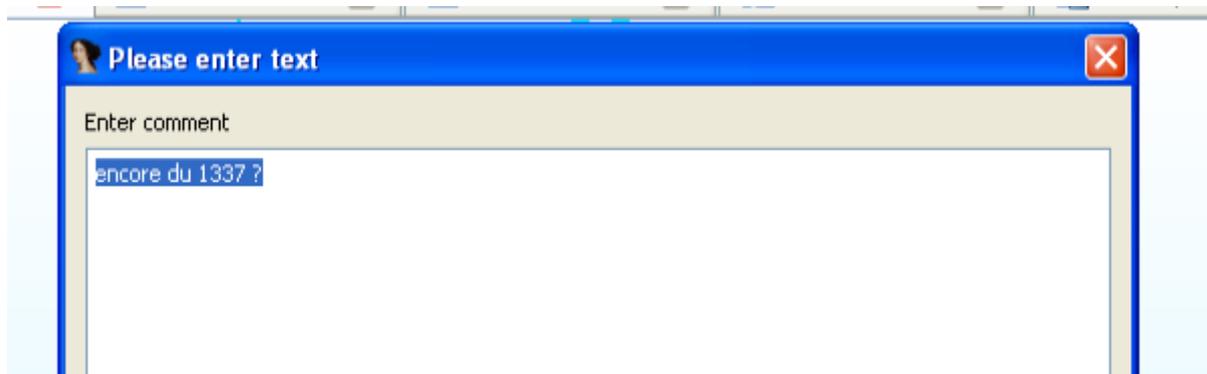
Quand on dessas prendre des notes c'est bien, mais commenter et s'appropriier le code c'est encore mieux !

IDA propose de pouvoir tout renommer, les variables, les fonctions etc ... mais elle permet aussi de commenter le code !

Let's run !

Commenter le code

Les commentaire en asm c'est « : », du coup taper « : » sur votre ligne, une fenêtrre s'ouvre commentez !



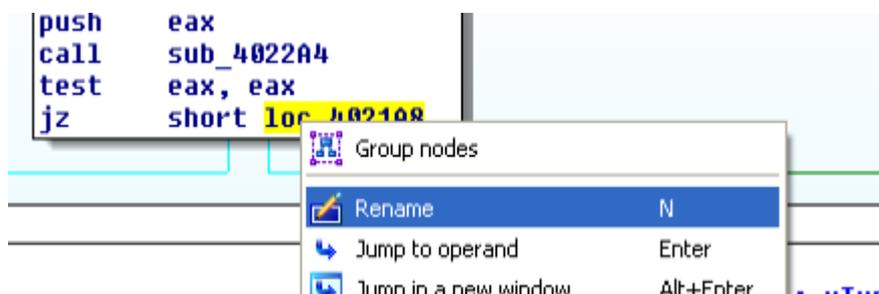
Donne =>

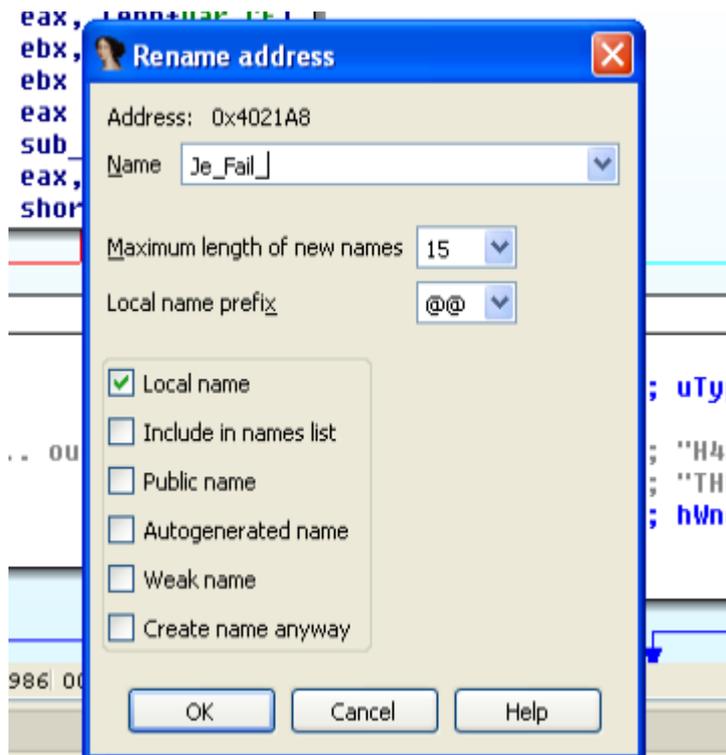
```
rol    eax, 4
shl    eax, 2
xor    eax, 13374266h ; encore du 1337 ?
push   ebx
mov    ebx, [ebp+arg 4]
```

Renommer le reste

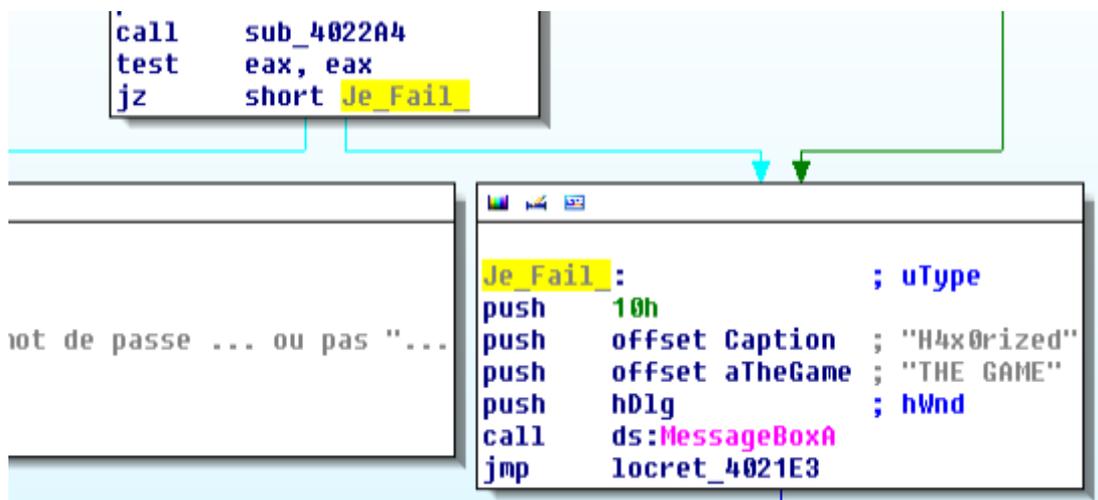
Vous venez de trouver ce que fait une fonction ou autre ? Oubliez le papier IDA est là !

Clic droit
=>



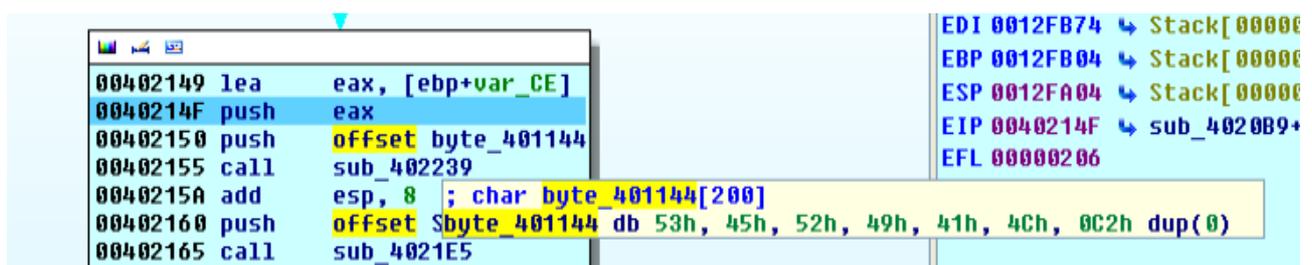


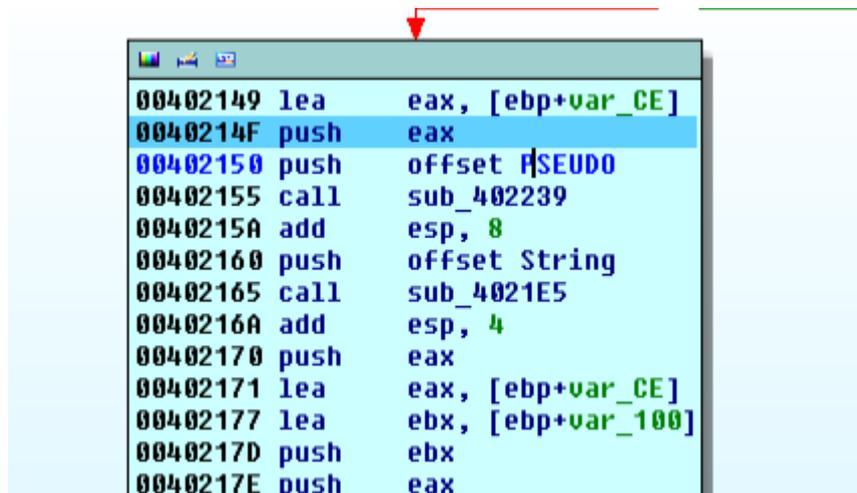
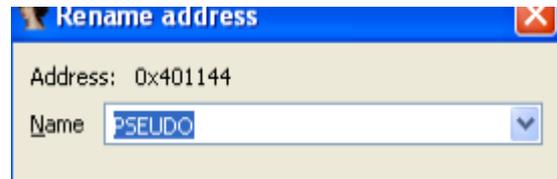
Donne :



Autre exemple :

Ici je sais qu'a cet offset j'ai le PSEUDO autant le dire non ?

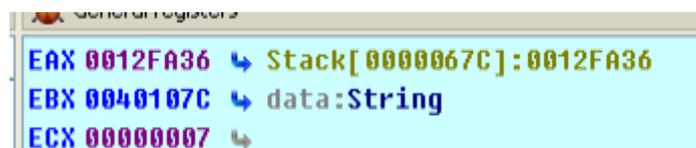




Pas mal non ?

Ce qui donne dans les registres :

Avant



Après

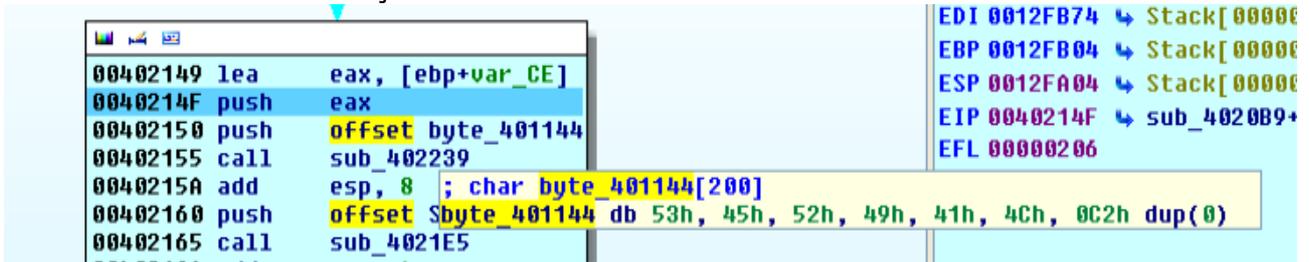


Héhé ! Je vous laisse imaginer la suite ...

Encore plus ? Ok .. je vous livre mes découvertes :) .

Afficher les Strings

Tout à l'heure on avait ça :



```
00402149 lea  eax, [ebp+var_CE]
0040214F push eax
00402150 push offset byte_401144
00402155 call sub_402239
0040215A add  esp, 8 ; char byte_401144[200]
00402160 push offset Sbyte_401144 db 53h, 45h, 52h, 49h, 41h, 4Ch, 0C2h dup(0)
00402165 call sub_4021E5
```

Stack information:
EDI 0012FB74 Stack[00000000]
EBP 0012FB04 Stack[00000000]
ESP 0012FA04 Stack[00000000]
EIP 0040214F sub_4020B9+...
EFL 00000206

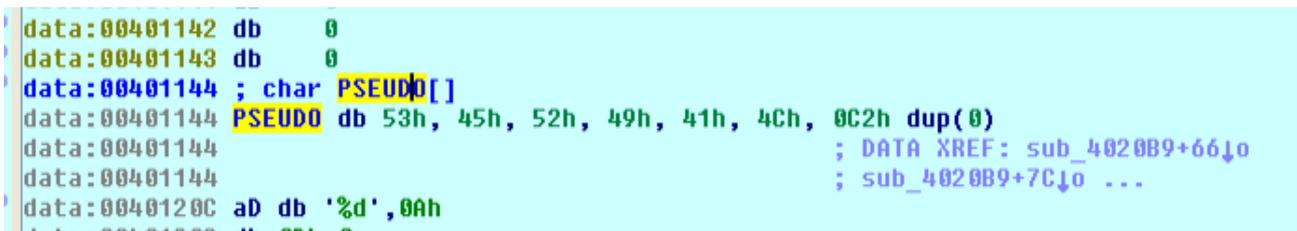
On ne parle pas tous (encore) l'hexa couramment.

```
echo -e "\x53\x45\x52\x49\x41\x4c"  
SERIAL
```

Donc on voit bien qu'en allant vite tout à l'heure j'ai renommé l'offset du serial en pseudo ... passons : p .

Donc c'est assez lourd de devoir « convertir » en string à chaque fois.

Donc on va sur notre bidule et on appuie sur « a »

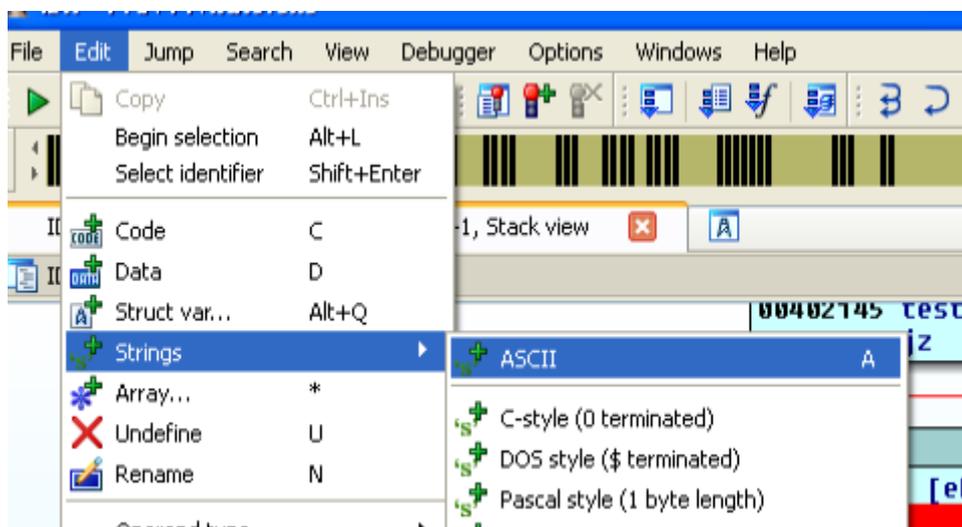


```
data:00401142 db 0
data:00401143 db 0
data:00401144 ; char PSEUDO[]
data:00401144 PSEUDO db 53h, 45h, 52h, 49h, 41h, 4Ch, 0C2h dup(0)
data:00401144 ; DATA XREF: sub_4020B9+66↓o
data:00401144 ; sub_4020B9+7C↓o ...
data:0040120C aD db '%d',0Ah
```



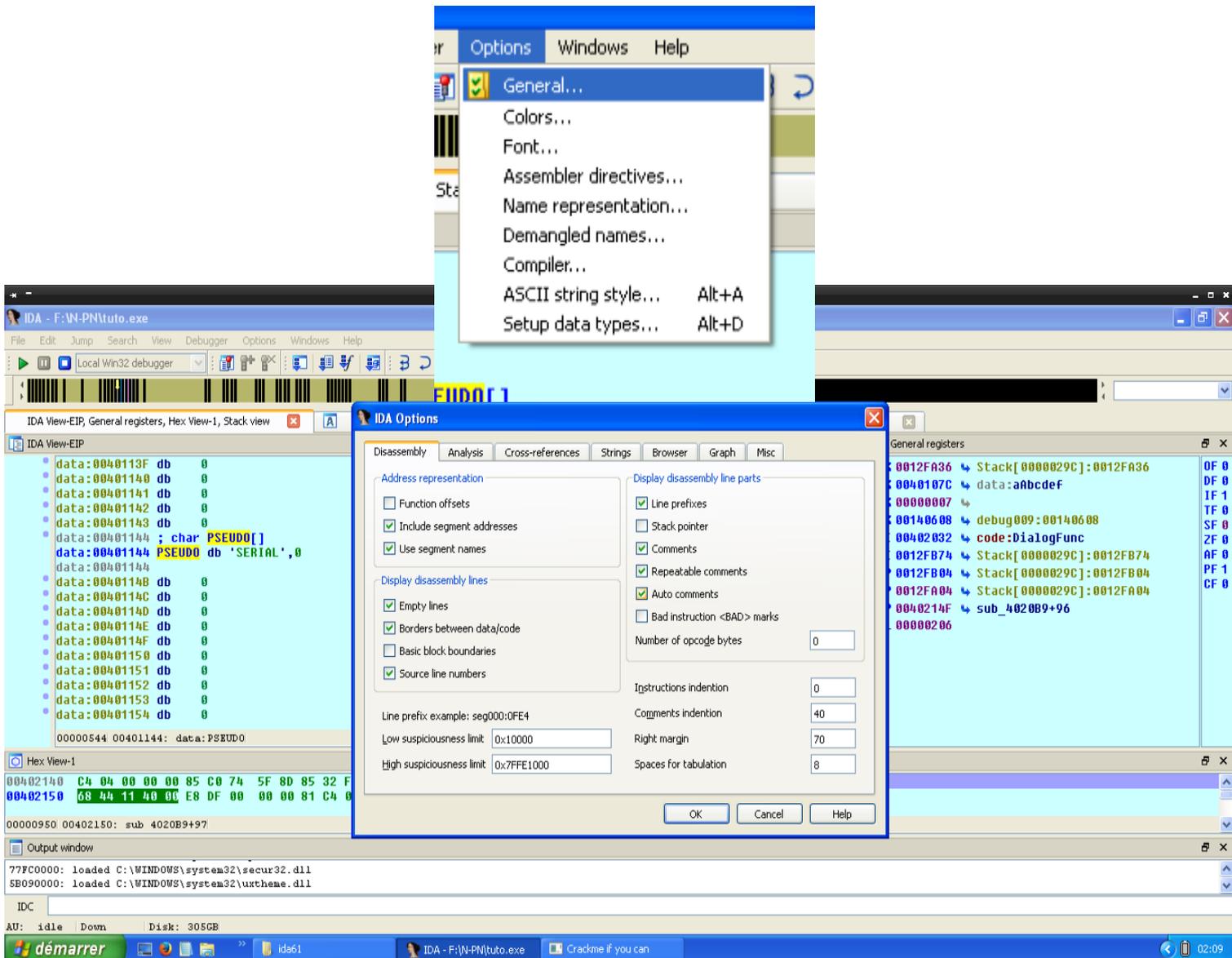
```
00401143 db 0
00401144 ; char PSEUDO[]
00401144 PSEUDO db 'SERIAL',0
00401144
```

On peut aussi :



Une aide bienvenue !

Encore autre chose, parfois on se confond dans les JMP , JZ , JBE , XOR , ROL alors IDA nous traduit .. si si je vous jure !



Cochez « Auto-Comments »

J'admets le résultat est un peu lourd, mais bon ça peut aider sur un trou de mémoire .

```

lea    eax, [ebp+var_100] ; Load Effective Address
push   eax
push   offset aAbcdef    ; ""
call   sub_4021FA        ; Call Procedure
add    esp, 8            ; Add
push   31h               ; cchMax
push   offset PSEUDO    ; lpString
push   3EBh              ; nIDDlgItem
push   hDlg              ; hDlg
call   ds:GetDlgItemTextA ; Indirect Call Near Procedure
push   offset PSEUDO    ; ""
call   sub_4021E5        ; Call Procedure
add    esp, 4            ; Add
test   eax, eax          ; Logical Compare
jz     short Je_Fail_    ; Jump if Zero (ZF=1)

```

IDA Ne Perd pas le Nord (lui)

Envie de savoir en plein debug ce que vaut un registre , ou ce que va donner le résultat de quelque chose ? Laissez votre souris dessus !

Esp ?

```

esp, 4 ; A
h     eax
eax   esp=Stack[00000590]:0012FA04 ; L
ebx   db 36h ; 6 ; L
h     ebx db 0
h     eax db 0E7h ; p
l     sub db 3 ; C
      db 26h ; &
sub_4020B9 db 0
      db 26h ; &
      db 13h
68 44 1   á 00
85 C0 7   ...

```

Voilà une petite idée quand on bosse sur un truc :

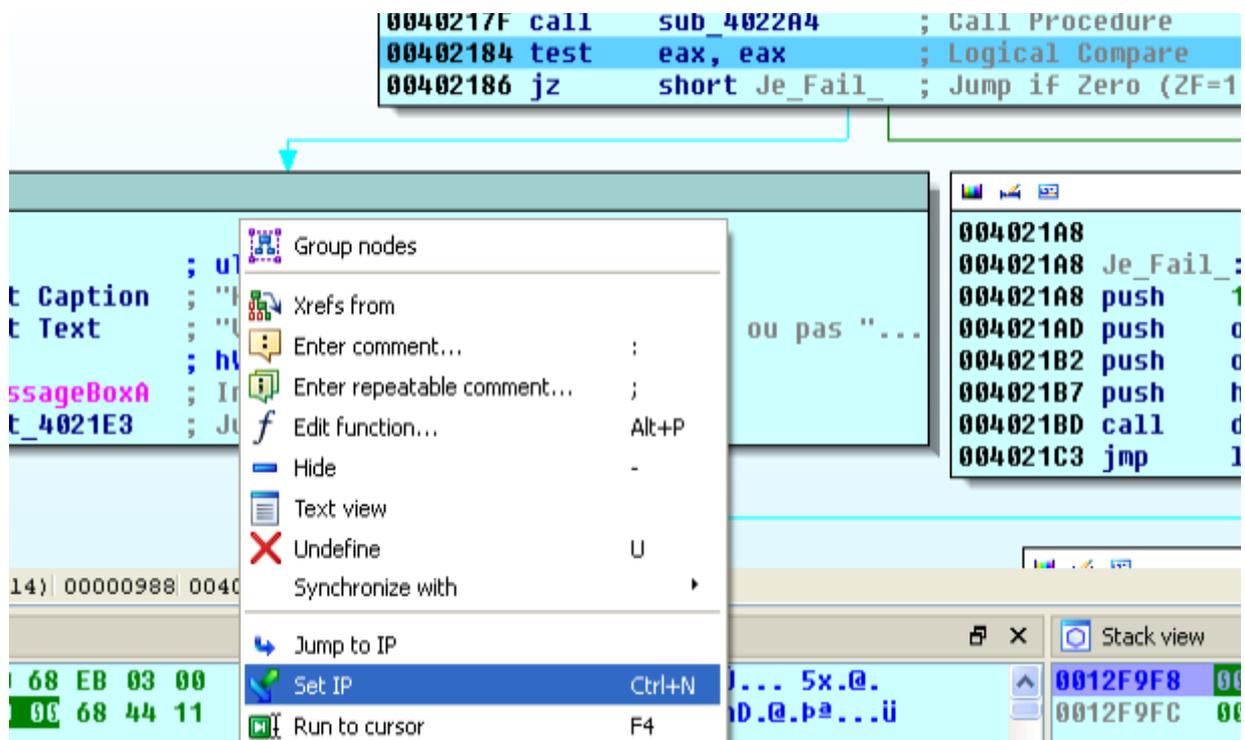
```

104 lea    eax, [ebp+var_CE] ; Load Effective Address
104 push   eax
108 push   offset aSerial    ; ""
10C call   avant_cmp         ; Call Procedure
10C add    esp, 8            ; Add
104 push   offset Pseudo    ; ""
108 call   Rec_pseu_strlen   ; Call Procedure
108 add    esp, 4            ; Add
104 push   eax
108 lea    eax, [ebp+var_CE] ; Load Effective Address
108 lea    ebx, [ebp+var_100] ; Load Effective Address
108 push   ebx
10C push   eax
110 call   Check_serial     ; Call Procedure
110 test   eax, eax          ; Logical Compare
110 jz     short The_Game_Fail ; Jump if Zero (ZF=1)

```

L'édition Hexa et le Jump To

Vous voulez tester un nop ou autre sans toucher à l'hexa ? Faites clic droit « Set IP ».

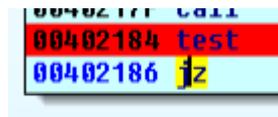


On jump plus à fail.

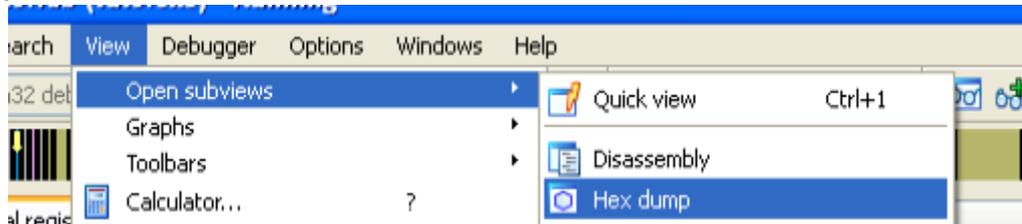


Éditer l'Hexa

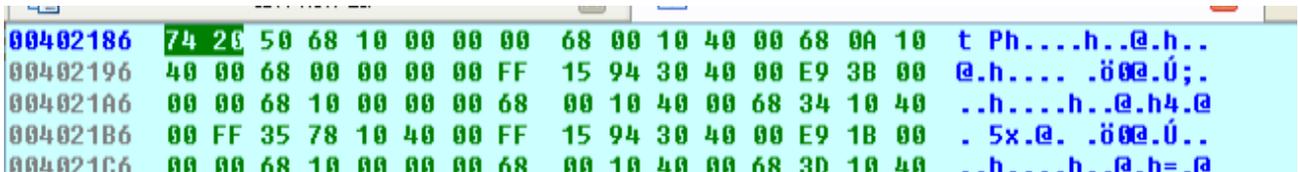
On met en jaune (1 clic)



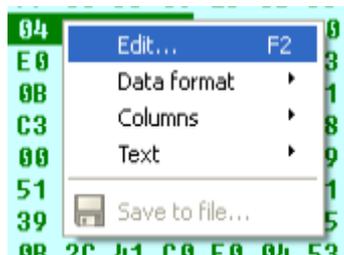
Si jamais on ouvre la « Hex view »



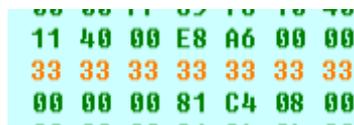
« Notre Jump est déjà sélectionné »



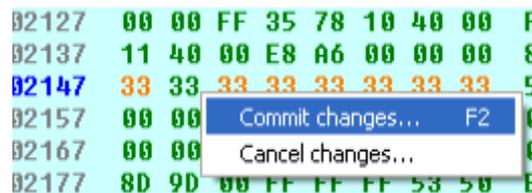
On clic droit « edit » :



On met ce que l'on souhaite :



On enregistre :



--

Alors Heureux ? Il y'a encore beaucoup de choses mais c'est déjà pas mal.
A une prochaine !

Thxer ;)