Execution konkolik

for dummies

Exécution concolique ?

On a du code

```
.text:004012DF loc_4012DF:                             ; CODE XREF: sub_4012B0+1C↑j
.text:004012DF                 push    ds:dword_590010
.text:004012E5                 mov     eax, Dst
.text:004012EA                 push    dword ptr [eax+28h]
.text:004012ED                 push    edx
.text:004012EE                 call    ecx
.text:004012F0                 add     esp, 0Ch
.text:004012F3                 test    eax, eax
.text:004012F5                 jz      short loc_401340
.text:004012F7                 cmp     eax, ds:dword_590010
.text:004012FD                 jle     short loc_401340
.text:004012FF                 cmp     dword ptr [ebx+4], 0
.text:00401303                 mov     ds:dword_590010, eax
.text:00401308                 mov     off_48D020, ebx
.text:0040130E                 mov     dword ptr [ebx], 0
.text:00401314                 jnz     short loc_40131E
.text:00401316                 mov     eax, off_48D034
.text:0040131B                 mov     [ebx+4], eax
.text:0040131E
.text:0040131E loc_40131E:                             ; CODE XREF: sub_4012B0+64↑j
.text:0040131E                 mov     edx, off_48D020
.text:00401324                 cmp     dword ptr [edx+8], 0
.text:00401328                 jnz     short loc_401332
.text:0040132A                 mov     eax, off_48D038
.text:0040132F                 mov     [edx+8], eax
```

Exécution concolique ?

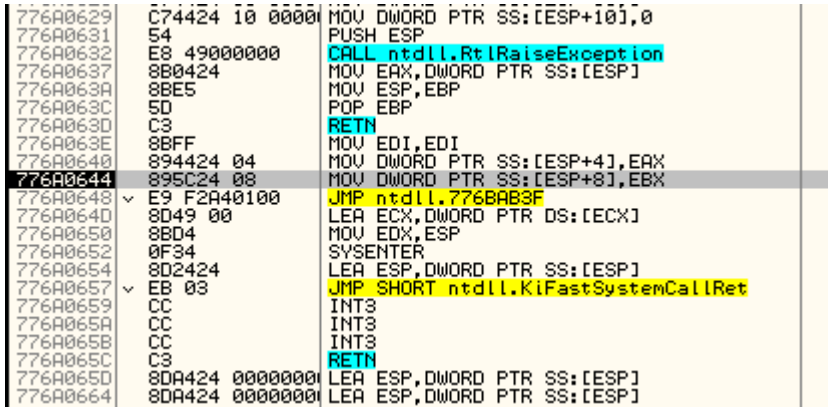
On a de la symbolique

« add dword ptr [eax+0x28], ebx »

{Indirection[[:eax, :+, 40], 4, 0]=>
Expression[Indirection[[:eax, :+, 40], 4, 0], :+, :ebx],
:eflag_z=>
Expression[[[[Indirection[[:eax, :+, 40], 4, 0], :&, 4294967295], :+, [:ebx, :&, 4294967295]], :&, 4294967295], :==, 0],
:eflag_s=>
Expression[[[[[Indirection[[:eax, :+, 40], 4, 0], :&, 4294967295], :+, [:ebx, :&, 4294967295]], :&, 4294967295], :>>, 31], :!=, 0],
:eflag_c=>
Expression[[[Indirection[[:eax, :+, 40], 4, 0], :&, 4294967295], :+, [:ebx, :&, 4294967295]], :>, 4294967295],
:eflag_o=>
Expression[[[[[Indirection[[:eax, :+, 40], 4, 0], :&, 4294967295], :>>, 31], :!=, 0], :==, [[[:ebx, :&, 4294967295], :>>, 31], :!=, 0]], :"&&", [[[[Indirection[[:eax, :+, 40], 4, 0], :&, 4294967295], :>>, 31], :!=, 0], :!=, [[[[[Indirection[[:eax, :+, 40], 4, 0], :&, 4294967295], :+, [:ebx, :&, 4294967295]], :&, 4294967295], :>>, 31], :!=, 0]]]}

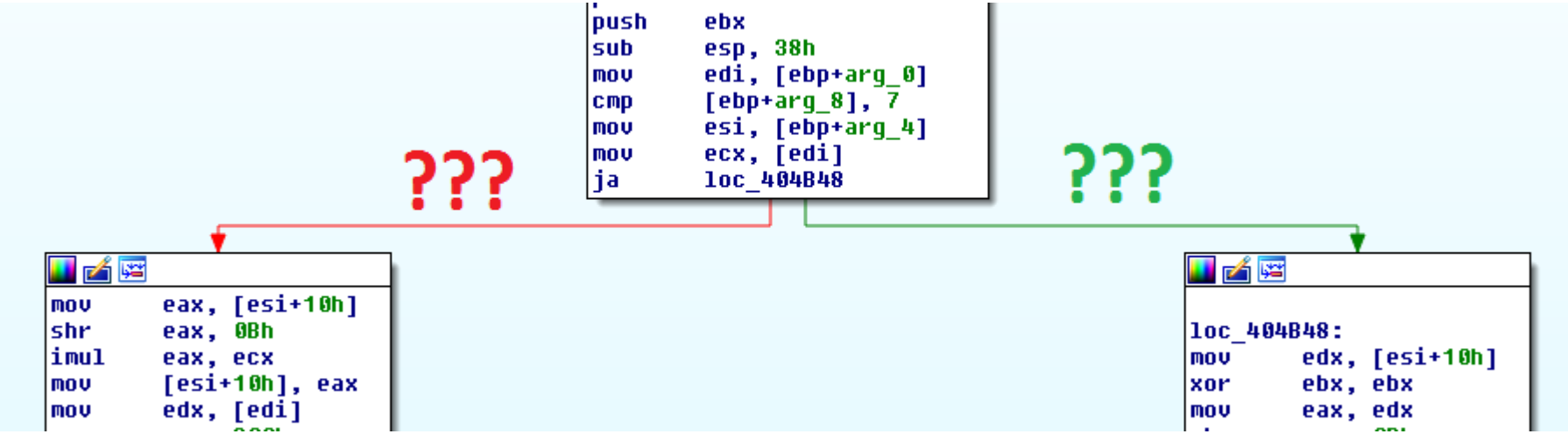
Exécution concolique ?

On a de l’exécution de code



Exécution concolique ?

Objectif : prédire des chemins d'exécution !



Pourquoi un tool ?

Il existe déjà des choses (exemple : Triton)

J’ai rien vu qui me convient, c’est-à-dire:

- Tainte tout seul mes datas
- Identifie mes conditions
- Résout mes conditions
- Continue l’exploration

Pourquoi un tool ?

Pour se coder son tool : Metasm (ofc) <3

- Il a un désassembleur
- Il a une symbolique
- Il a un débugger

Pourquoi un tool ?

Pour se coder son tool : Metasm (ofc) <3

- Il a un désassembleur
- Il a une symbolique
- Il a un débugger

Bon par contre faut ajouter les breakpoints mémoire...

Pourquoi un tool ?

Pour se coder son tool : Metasm (ofc) <3

- Il a un désassembleur
- Il a une symbolique
- Il a un débugger

Bon par contre faut ajouter les breakpoints mémoire...

Et un système de tainte mémoire…

Pourquoi un tool ?

Pour se coder son tool : Metasm (ofc) <3

- Il a un désassembleur
- Il a une symbolique
- Il a un débugger

Bon par contre faut ajouter les breakpoints mémoire...

Et un système de tainte mémoire…

Et un solveur de contraintes…

Comment on fait ?

On sélectionne un fichier et un exécutable à tester.

On va tracer le fichier dès qu'il est load :

- ZwCreateFile
- ZwReadFile
- ZwCreateSection
- ZwMapViewOfSection
- ZwSetInformationFile

Comment on fait ?

Quand le fichier est chargé on met un breakpoint mémoire dessus.

On créé aussi une relation

Adresse mémoire -> Offset dans le fichier

Comment on fait ?

Quand la mémoire est lue on a une exception.

A ce moment on tainte le registre et on remplit (encore) des structures :

- Registre -> Offset du fichier
- Registre -> Opérations effectuées (contraintes)
- Stack d’exécution <- Instruction

Comment on fait ?

Tainte des registres et mémoire

```
Mov Eax, dword ptr [Ebx] ; BPM
Mov Ecx, 5
Add Eax, Ecx
Cmp Esi, Eax
Jz loc_next
```

Comment on fait ?

Tainte des registres et mémoire

```
Mov Eax, dword ptr [Ebx]
Mov Ecx, 5
Add Eax, Ecx
Cmp Esi, Eax
Jz loc_next
```

Comment on fait ?

Tainte des registres et mémoire

```
Mov Eax, dword ptr [Ebx]
Mov Ecx, 5
Add Eax, Ecx ; (reg + 5)
Cmp Esi, Eax
Jz loc_next
```

Comment on fait ?

Tainte des registres et mémoire

```
Mov Eax, dword ptr [Ebx]
Mov Ecx, 5
Add Eax, Ecx
Cmp Esi, Eax
Jz loc_next
```

Comment on fait ?

Tainte des registres et mémoire

```
Mov Eax, dword ptr [Ebx]
Mov Ecx, 5
Add Eax, Ecx
Cmp Esi, Eax
Jz loc_next
```

Comment on fait ?

Tant qu'un registre est tainté on singlestep.

Lors des Jcc (Jz, Jnz, Js, ...) on remonte la stack d'exécution pour trouver qui a placé le flag.

`Cmp Esi, Eax`

Comment on fait ?

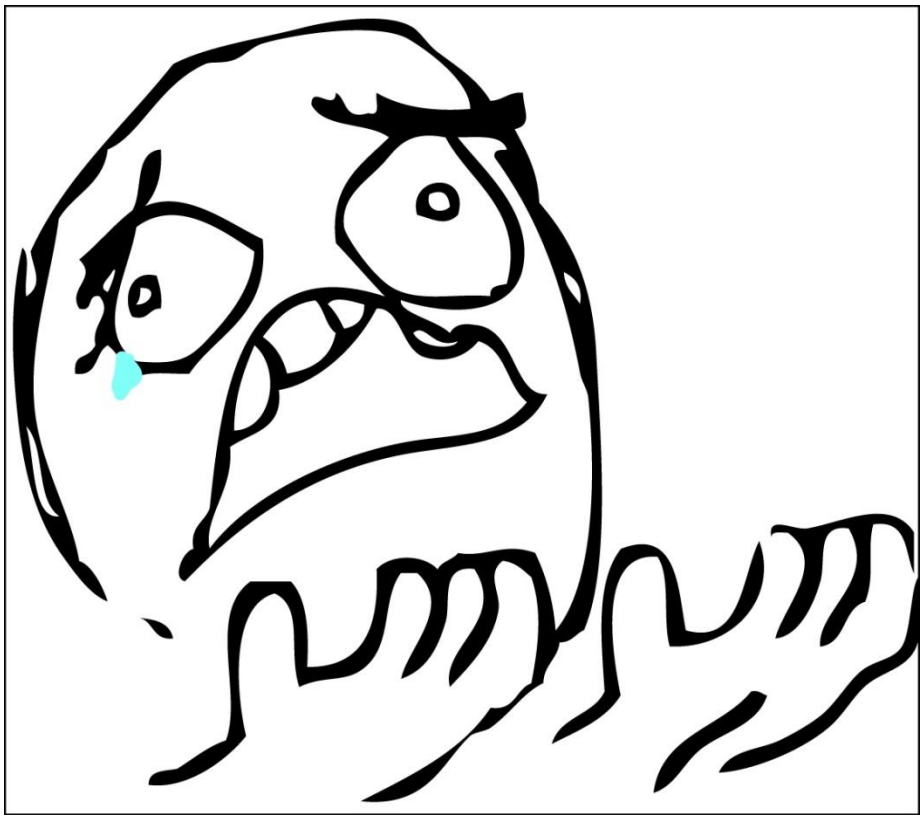
On résout la contrainte !

Z3 ?

Comment on fait ?

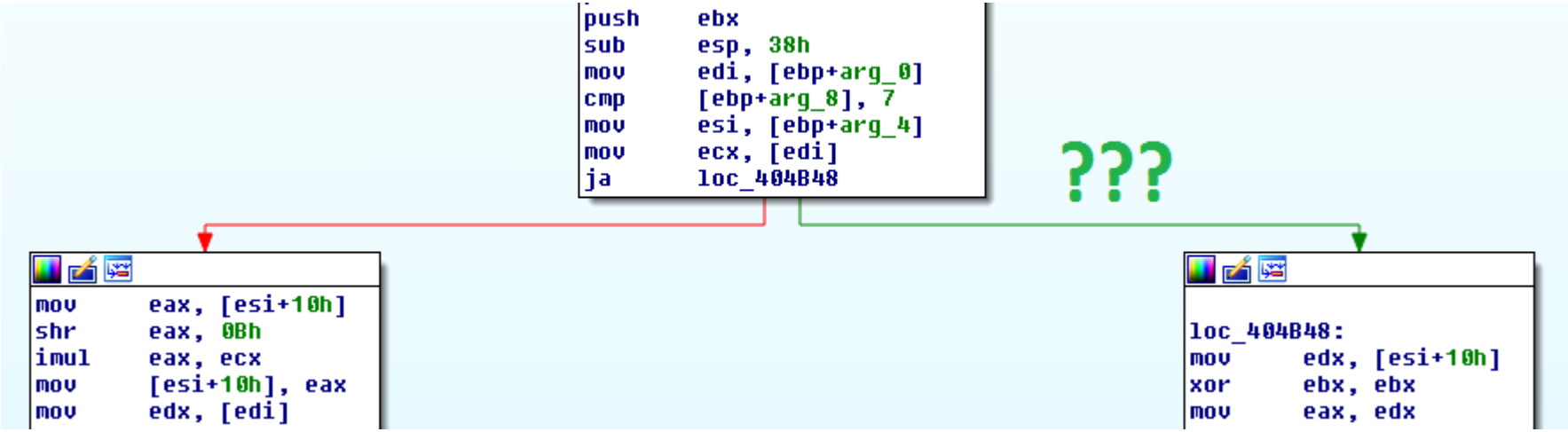
On résout la contrainte !

Z3 ?



Comment on fait ?

Dans quel cas on est ?



On a donc déjà un chemin.

Comment on fait ?

/me est trop fort en math !

On a pas besoin de résoudre le 1er chemin.

Pour le deuxième on a juste à inverser une « équation ».

8 = 5+3

7 = 5+x

Comment on fait ?

En fait on a même pas à résoudre...

```
Add Eax, Ecx ; (reg + 5)
Add Eax, Ecx ; (reg - 5)
```

Comment on fait ?

En fait on a même pas à résoudre...

```
Add Eax, 4 ; (reg - 4)
Shl Eax, 2 ; ((reg >> 2) - 4)
Cmp Eax, 0x20
```

```
((0x20 >> 2) - 4)
```

Comment on fait ?

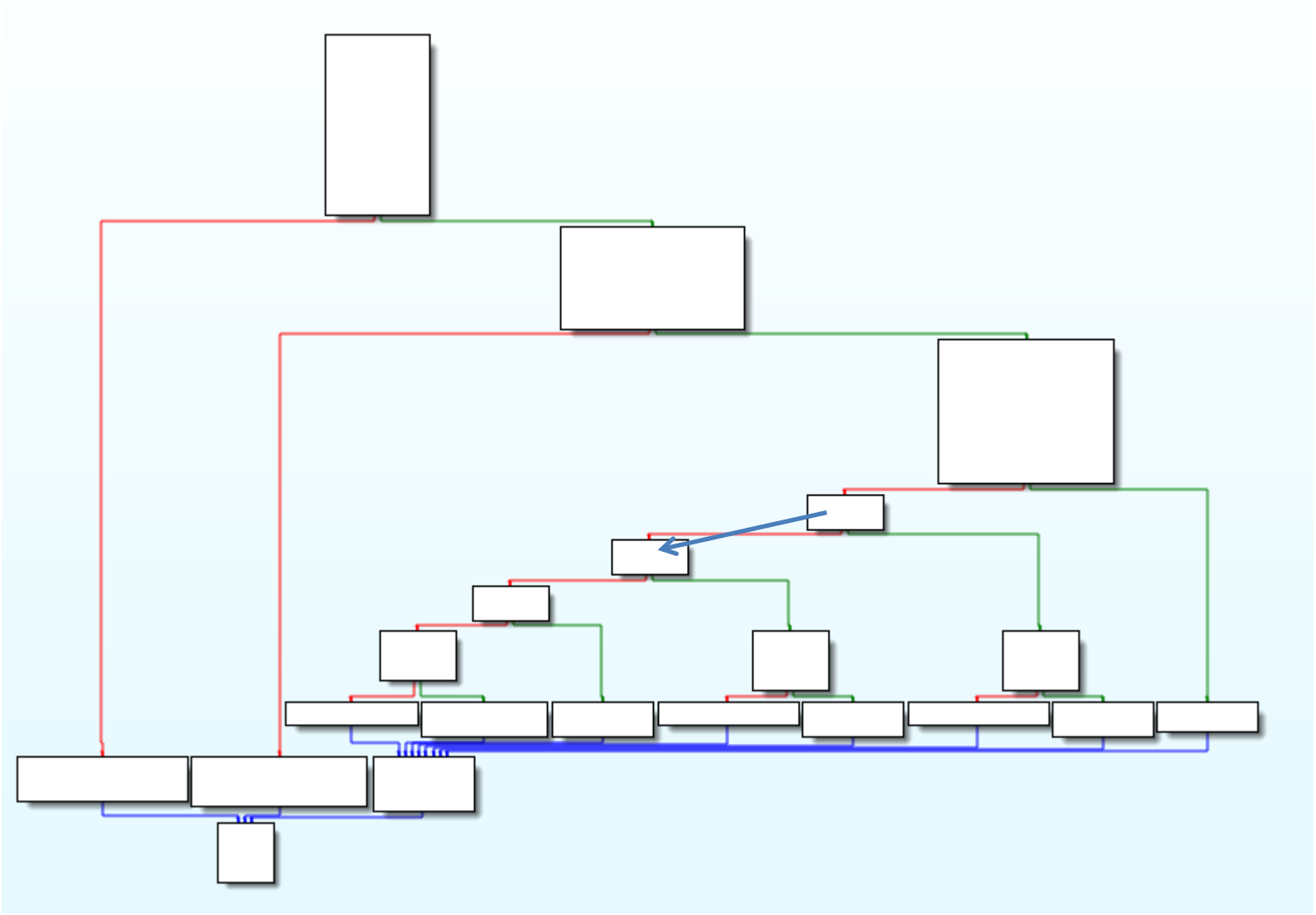
On a l’expression résolue !

On a l’offset affecté au registre.

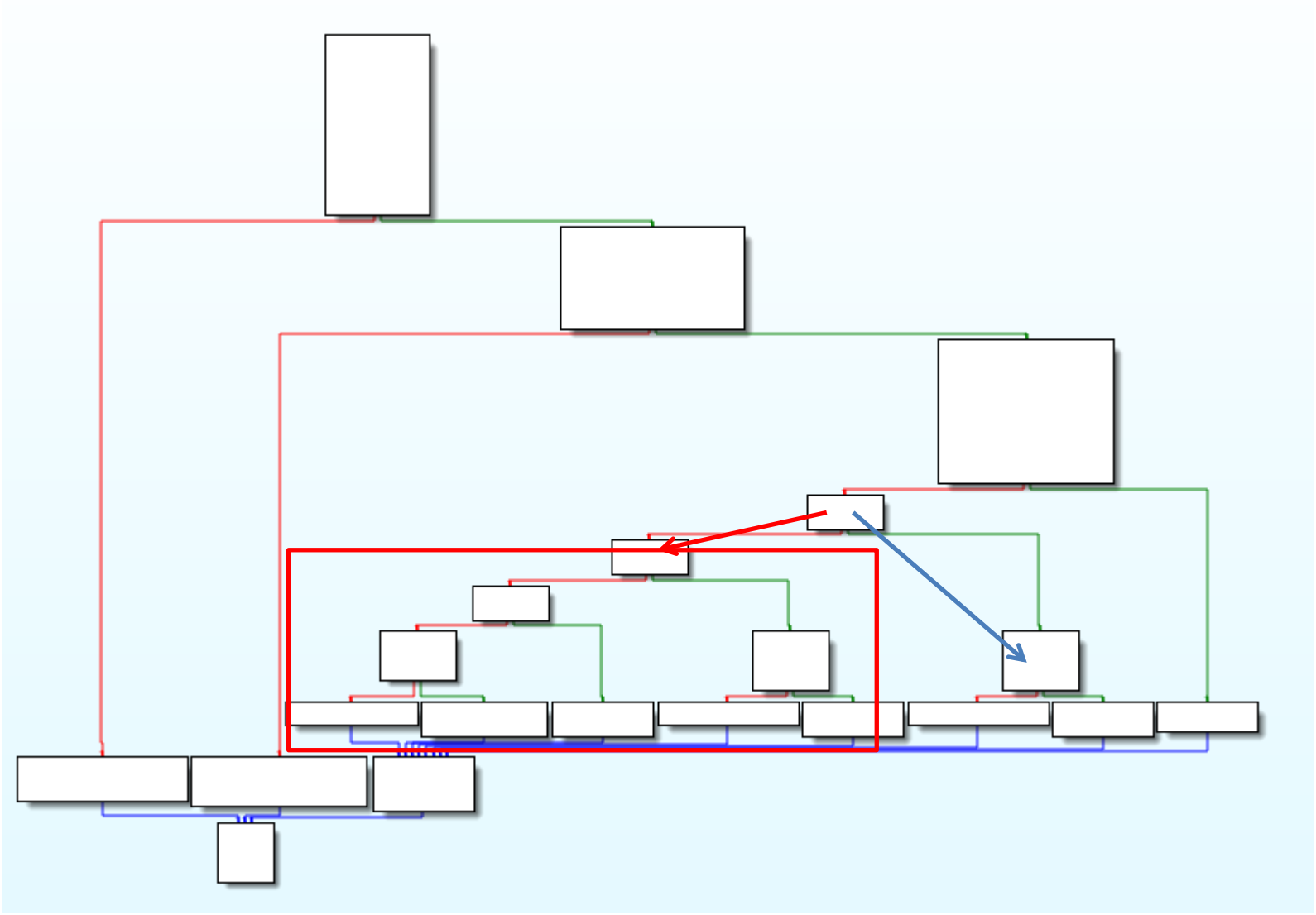
Plus qu’à écrire dans le fichier.

Problème…

Comment on fait ?



Comment on fait ?



Comment on fait ?

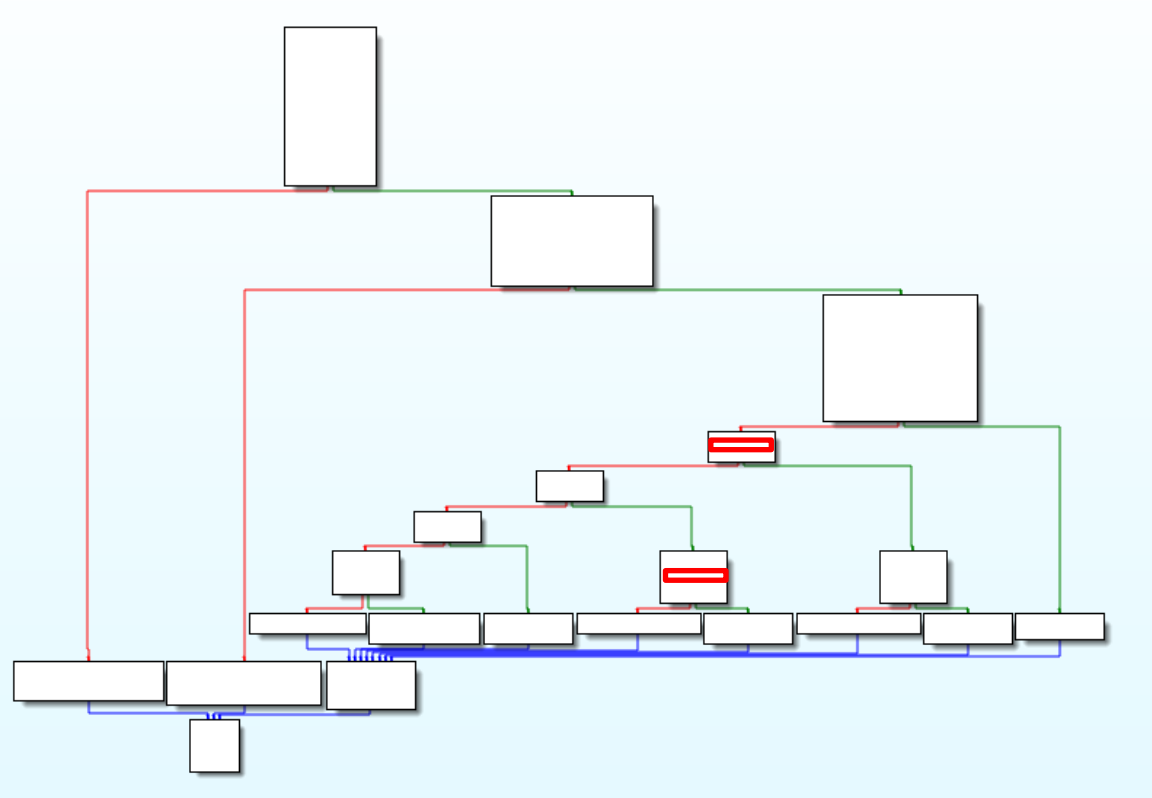
On stock :

- L'adresse de la condition
- Le layer dans lequel il est intervenu (jcc n°5)
  - On tentera d'aller plus en profondeur dans chaque branche
  - On inverse les conditions de chaque jcc
  - Si on n'est pas allé plus loin on remonte d'un layer
- On stocke le nombre de passages dans la branche (de 0 à 4, 4 étant tout a été testé)

Comment on fait ?

MAIS...

Si deux opérations touchent le même offset ?



Dans ce cas on ne retournera pas dans certaines branches.

Comment on fait ?

Donc on stock les fichiers ayant menés à chaque jcc.

Si une branche devient perdue on reprendra le fichier utilisé et on repartira de là pour faire muter.

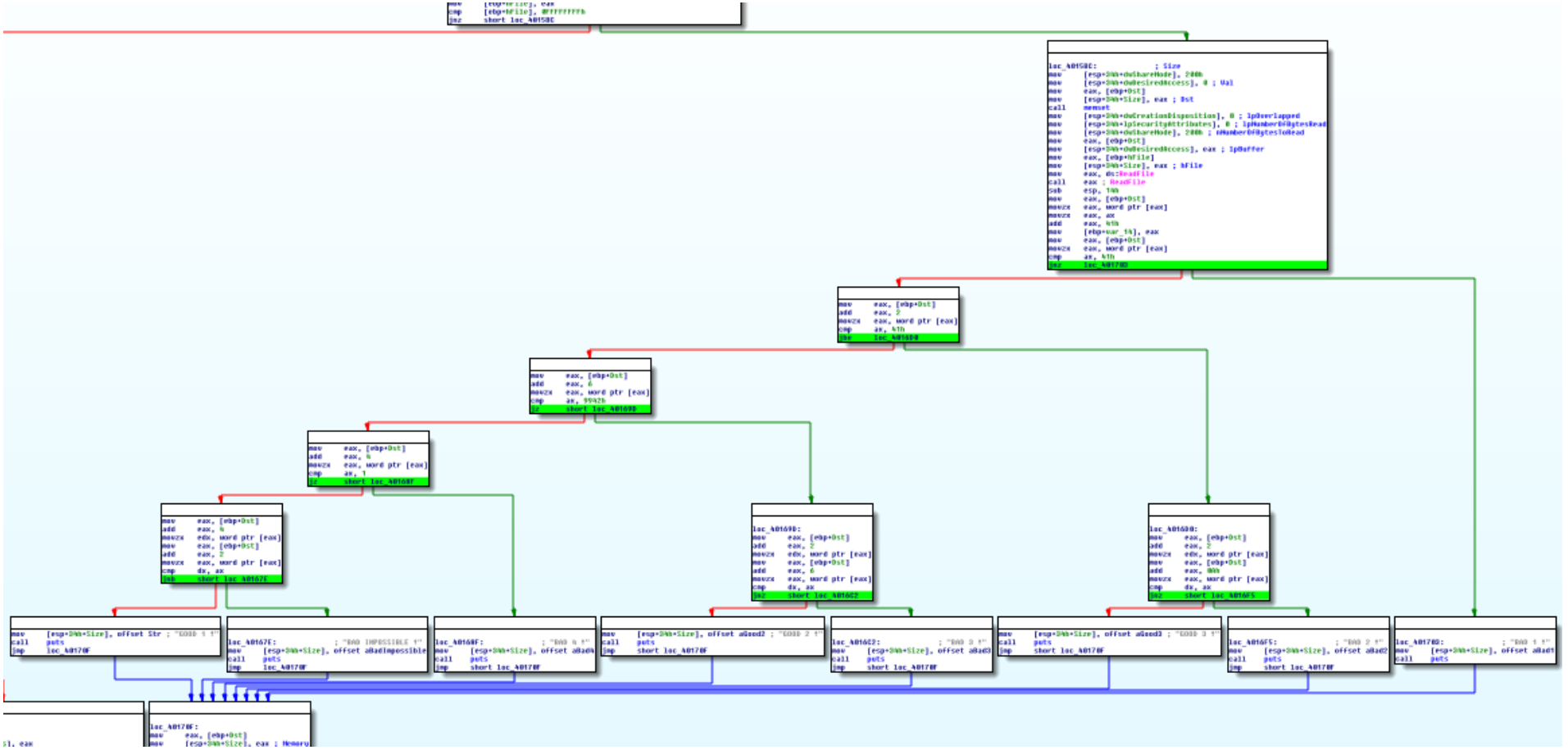
Ca marche !

```
ReadFile(bin1, data1, 512, 0, 0);

   bin2 = data1[0]+0x41;

   if (data1[0] == 0x41){
       if (data1[1] >= 0x42){
           if (data1[3] != 0x9942){
               if (data1[2] != 1){
                   if (data1[2] < data1[1]){
                       printf("GOOD 1 !\n");
                   }else{printf("BAD 5 !\n");}
               }else{printf("BAD 4 !\n");}
           }else{
               if (data1[1] == data1[3]){
                   printf("GOOD 2 !\n");
               }else{printf("BAD 3 !\n");}}
       }else{
           if (data1[1] == data1[5]){
               printf("GOOD 3 !\n");
           }else{printf("BAD 2 !\n");}}
   }else{printf("BAD 1 !\n");}
```

Ca marche !

```
echo
.....................................................................................
.....................................................................................
.....................................................................................
.....................................................................................
.....................................................................................
.....................................................................................
.....................................................................................
.....................................................................................
.....................................................................................
.....................................................................................
.....................................................................................
.....................................................................................
.....................................>toto.txt
```

Ca marche !



Ca marche !

```
 [-] Run he process
  [*] CreateFile on UPXnslookup.exe
  [*] Handle to surveil : 0x40
   [*] ReadFile(0x8022b0,0x200) on surveilled handle
401609h movzx eax, word ptr [eax] ; eax => 8022b0
40160ch movzx eax, ax ; eax => 2e2e
40160fh add eax, 41h ; eax => 2e2e
401612h mov [ebp-14h], eax ; eax => 2e6f  ebp => 28fea8
401615h mov eax, [ebp-0ch] ; eax => 2e6f  ebp => 28fea8
401618h movzx eax, word ptr [eax] ; eax => 8022b0
40161bh cmp ax, 41h ; eax => 2e2e
40161fh jnz loc_401703h ; x:loc_401703h ;
 [B] 40161bh cmp ax, 41h
  ARG1 is controled
  Tainted eax&0ffffh :
   [S] val1 is controled = 0x2e2e
   [S] val2 is const = 0x41
   [S] size of operation : 2
    [S] already jump to dest
     [S] val1 is at 0x0 offset
     [S] now offset 0x0 = 0x41
BAD 1 !
  0x40161f -> 2
@layers_passed = 0
@layers_to_pass = 1
```

Ca marche !

```
 [-] Run he process
  [*] CreateFile on UPXnslookup.exe
  [*] Handle to surveil : 0x34
   [*] ReadFile(0x5122b0,0x200) on surveilled handle
401609h movzx eax, word ptr [eax] ; eax => 5122b0
40160ch movzx eax, ax ; eax => 41
40160fh add eax, 41h ; eax => 41
401612h mov [ebp-14h], eax ; eax => 82  ebp => 28fea8
401615h mov eax, [ebp-0ch] ; eax => 82  ebp => 28fea8
401618h movzx eax, word ptr [eax] ; eax => 5122b0
40161bh cmp ax, 41h ; eax => 41
40161fh jnz loc_401703h ; x:loc_401703h ;
401625h mov eax, [ebp-0ch] ; eax => 41  ebp => 28fea8
401628h add eax, 2 ; eax => 5122b0
40162bh movzx eax, word ptr [eax] ; eax => 5122b2
40162eh cmp ax, 41h ; eax => 2e2e
401632h jbe loc_4016d0h ; x:loc_4016d0h ;
 [B] 40162eh cmp ax, 41h
  ARG1 is controled
  Tainted eax&0ffffh :
   [S] val1 is controled = 0x2e2e
   [S] val2 is const = 0x41
   [S] size of operation : 2
    [S] already continue execution
     [S] val1 is at 0x2 offset
     [S] now offset 0x2 = 0x40
BAD 5 !
  0x40161f -> 2
  0x401632 -> 2
@layers_passed = 1
@layers_to_pass = 2
```

Ca marche (IRL -> UPX) !

```
 Ultimate Packer for eXecutables
                          Copyright (C) 1996 - 2013
UPX 3.91w       Markus Oberhumer, Laszlo Molnar & John Reiser   Sep 30th 2013

        File size         Ratio      Format      Name
   --------------------   ------   -----------   -----------
  [*] CreateFile on UPXnslookup.exe
  [*] Handle to surveil : 0x40
   [*] SetInformationFile(0x0) on surveilled handle
   [*] SetInformationFile(0x0) on surveilled handle
   [*] ReadFile(0x28f720,0x1c) on surveilled handle
41dbf6h movzx eax, word ptr [ebp-28h] ; eax => 1c  ebp => 28f748
41dbfah add esp, 0ch ; esp => 28f70c
41dbfdh cmp ax, 5a4dh ; eax => 2e2e
41dc01h jnz loc_41dcbbh ; x:loc_41dcbbh ;
 [B] 41dbfdh cmp ax, 5a4dh
  ARG1 is controled
  Tainted eax&0ffffh :
   [S] val1 is controled = 0x2e2e
   [S] val2 is const = 0x5a4d
   [S] size of operation : 2
    [S] already jump to dest
     [S] val1 is at 0x0 offset
     [S] now offset 0x0 = 0x5a4d
upx: UPXnslookup.exe: NotPackedException: not packed by UPX

Unpacked 0 files.
  0x41dc01 -> 2
@layers_passed = 0
@layers_to_pass = 1
```

Ca marche (IRL -> UPX) !

```
00000000- 4D 5A 0D 00  00 00 2E 2E  2E 2E 2E 2E  2E 2E 2E 2E [MZ..............]
00000001- 2E 2E 2E 2E  2E 2E 2E 2E  2E 2E 2E 2E  2E 2E 2E 2E [................]
00000002- 2E 2E 2E 2E  2E 2E 2E 2E  2E 2E 2E 2E  2E 2E 2E 2E [................]
00000003- 2E 2E 2E 2E  2E 2E 2E 2E  2E 2E 2E 2E  2E 2E 2E 2E [................]
00000004- 2E 2E 2E 2E  2E 2E 2E 2E  2E 2E 2E 2E  2E 2E 2E 2E [................]
00000005- 2E 2E 2E 2E  2E 2E 2E 2E  2E 2E 2E 2E  2E 2E 2E 2E [................]
00000006- 2E 2E 2E 2E  2E 2E 2E 2E  2E 2E 2E 2E  2E 2E 2E 2E [................]
[...]
0000001B- 2E 2E 2E 2E  2E 2E 2E 2E  2E 2E 2E 2E  2E 2E 2E 2E [................]
0000001C- 2E 2E 2E 2E  2E 2E 2E 2E  2E 2E 2E 2E  2E 2E 2E 2E [................]
0000001D- 2E 2E 2E 2E  2E 2E 2E 2E  2E 2E 2E 2E  2E 2E 2E 2E [................]
0000001E- 2E 2E 2E 2E  2E 2E 2E 2E  2E 2E 2E 2E  2E 2E 2E 2E [................]
0000001F- 2E 2E 2E 2E  2E 2E 2E 2E  2E 2E 2E 2E  2E 2E 2E 2E [................]

00000020- 67 6F 33 32  73 74 75 62  2E 2E 2E 2E  2E 2E 2E 2E [go32stub........]
00000021- 2E 2E 2E 2E  2E 2E 2E 2E  2E 2E 2E 2E  2E 2E 2E 2E [................]
00000022- 2E 2E 2E 2E  2E 2E 2E 2E  2E 2E 2E 2E  2E 2E 2E 2E [................]
```

Questions ?