

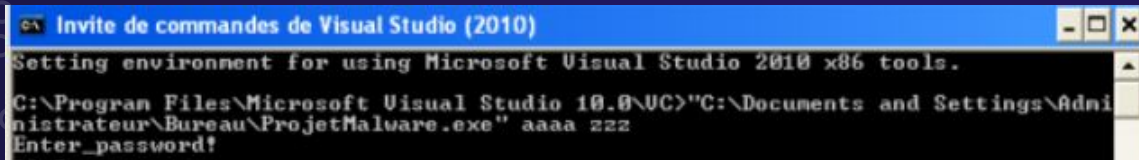
MALO Damien, CIPOLLA Nicolas, LAFOND Louis

# Étude : Malware du groupe 5

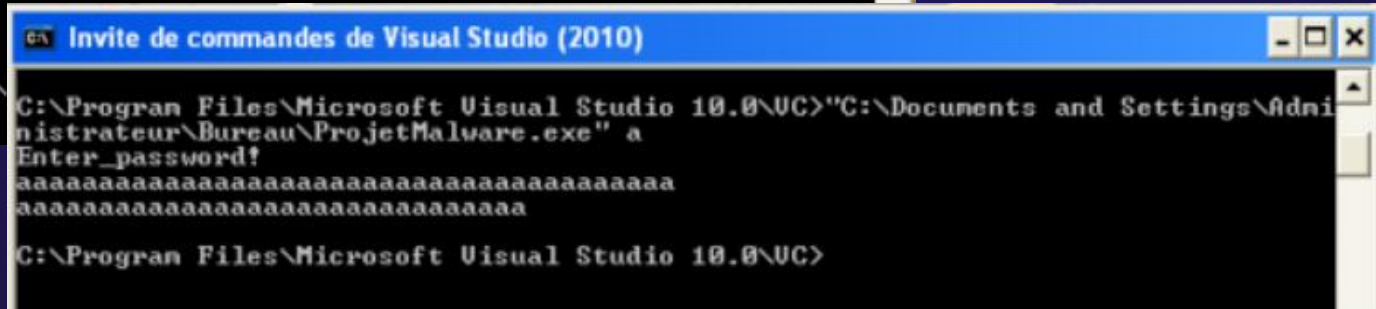


# Comportement Général

- Pas d'argument dans la fonction le programme s'ouvre et on entre le mot de passe
- Le programme met beaucoup de temps à s'exécuter ~ 5 secondes à chaque fois
- Comportements aux limites de la fonction



```
Invite de commandes de Visual Studio (2010)
Setting environment for using Microsoft Visual Studio 2010 x86 tools.
C:\Program Files\Microsoft Visual Studio 10.0\VC>"C:\Documents and Settings\Administrateur\Bureau\ProjetMalware.exe" aaaa zzz
Enter_password!
aa z
aa z
```



```
Invite de commandes de Visual Studio (2010)
C:\Program Files\Microsoft Visual Studio 10.0\VC>"C:\Documents and Settings\Administrateur\Bureau\ProjetMalware.exe" a
Enter_password!
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
C:\Program Files\Microsoft Visual Studio 10.0\VC>
```

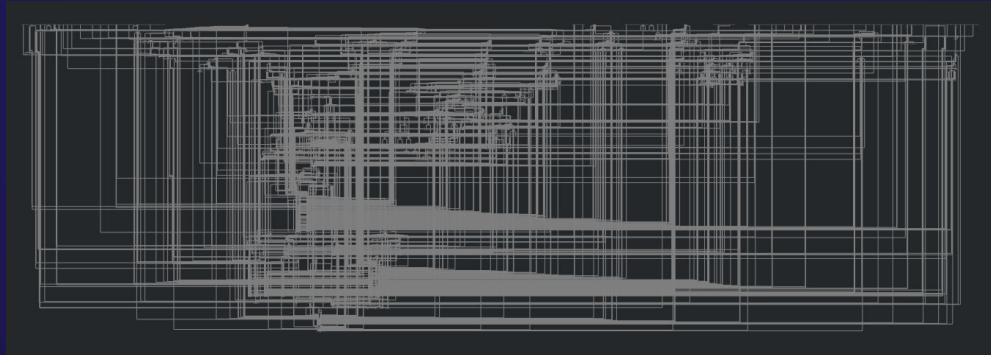
# Analyse Statique

- Quelques strings intéressants notamment le pseudo du créateur du Malware
- Utilisation de VS2022 pour la compilation

0042d4d0	?? 5Fh	_	"_logb"	string	6	false
0042d4d8	ds	"_nextafter"	"_nextafter"	string	11	true
0042d604	u_Alerte_0042d6...	unicode u"Alerte"	u"Alerte"	unico...	14	true
0042d614	u_Etes-vous_sur...	unicode u"Etes-vous sur la b...	u"Etes-vous sur la bonne piste?"	unico...	60	true
0042d650	u_C:\WINDOWS\...	unicode u"C:\\WINDOWS\\syste...	u"C:\\WINDOWS\\system32"	unico...	40	true
0042d680	u_C:\WINDOWS\...	unicode u"C:\\WINDOWS\\syste...	u"C:\\WINDOWS\\system32"	unico...	40	true
0042db54	ds	"C:\\Users\\0xy\\source\\...	"C:\\Users\\Oxy\\source\\repos\\ProjetMalware\\Release\\ProjetMalware.pdb"	string	66	true
0042dbb8	?? 2Eh	,	".text\$mn"	string	9	false

# Analyse Statique

- Beaucoup de fonctions qui sont quasiment toutes appelés par d'autres.
- Les fonctions ne portent pas de noms (sub\_xxxx) => `_attribute__((visibility("hidden")));` ?
- IDA version XP n'affiche pas la fonction main
- Bibliothèques classiques de Windows



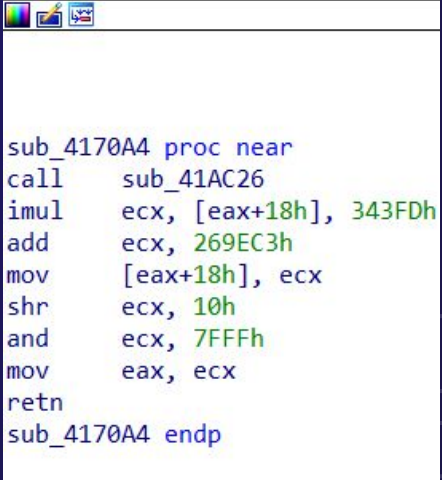
# Le Main:

- Analyse de la fonction StartAddress en premier lieu
  - fausse piste
- Analyse de start sur IDA windows XP
  - Fonction précédent le main et appel du main dans start
- Découverte du main
  - Plus de 1000 Noeuds dans la fonction main
  - Subroutine récurrente : 4170A0



# La Subroutine récurrente : SUB\_4170A4

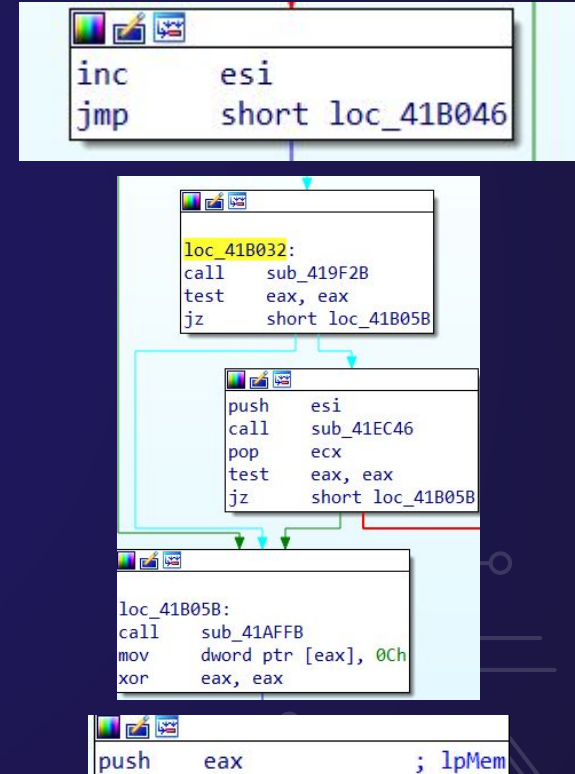
- Etude en profondeur de la fonction
  - But : Déterminer si la fonction est réellement utile ou alors si elle ne représente que du code mort
- La fonction en elle même:
  - Assez courte en apparence (8 lignes)
  - La subroutine 41AC26 est appelé
- Il faut donc analyser la subroutine présente



```
sub_4170A4 proc near
call    sub_41AC26
imul    ecx, [eax+18h], 343FDh
add     ecx, 269EC3h
mov     [eax+18h], ecx
shr     ecx, 10h
and     ecx, 7FFFh
mov     eax, ecx
retn
sub_4170A4 endp
```

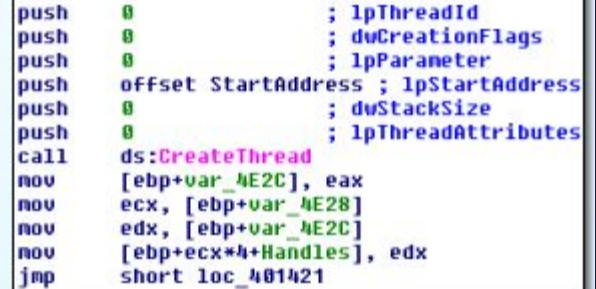
# Codes Mort et résultat de 41B00E

- A la sortie de cette subroutine :
  - Eax a comme valeur 0FFFFFFE/0x364
  - Esi a comme valeur 0x364
- Image à droite : Code mort évacuée grâce à l'analyse de cette subroutine
- Après quelques recherches, cette fonction se révéla être la fonction rand()



# Parcours du main

- A l'aide de méthode de contournement de débbug, nous avons pu nous aventurer dans le main
- Énormément de code mort
- Des fonctions d'anti-debug



```
push    0                ; lpThreadId
push    0                ; dwCreationFlags
push    0                ; lpParameter
push    offset StartAddress ; lpStartAddress
push    0                ; dwStackSize
push    0                ; lpThreadAttributes
call    ds:CreateThread
mov     [ebp+var_4E2C], eax
mov     ecx, [ebp+var_4E28]
mov     edx, [ebp+var_4E2C]
mov     [ebp+ecx*4+Handles], edx
jmp     short loc_401421
```



# Parcours du main : Problème principal

- Le main se compose de plusieurs traps à debugger
- Des boucles infinie (while true par exemple) dont il peut être compliqué d'identifier le point de départ
- De plus, certains while ne s'arrêtent qu'à l'aide d'une fonction random, ralentissant considérablement le debug

```
if (data_440228 s> var_764)
    int32_t var_3ec_1 = 0
    while (true)
        int32_t eax_33
        int32_t edx_20
        edx_20:eax_33 = sx.q(_rand())
        if (var_3ec_1 s>= mods.dp.d(edx_20:eax_33, 0x831))
            break
```

# Les mécanismes de protection

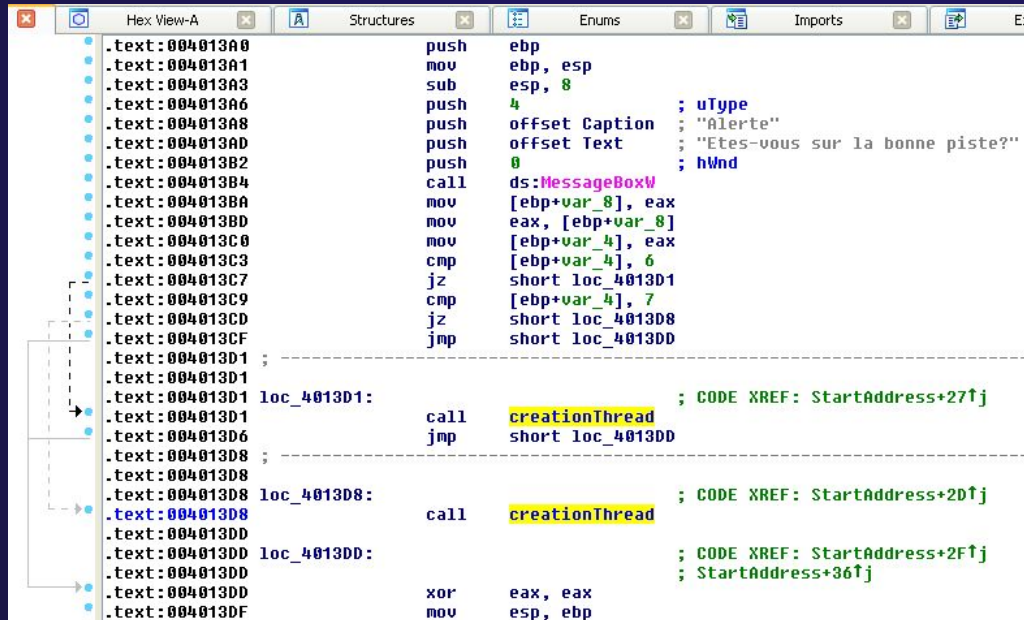
4 mécanismes empêchent de debugger l'exécutable :

- Threads
- MessageBoxes
- Delete system32
- Boucle avec random value



# Threads & MessageBoxes

“Fork bomb” : une infinité de processus sont créés pour saturer la mémoire et le processeur de la machine



```
.text:004013A0      push     ebp
.text:004013A1      mov      ebp, esp
.text:004013A3      sub      esp, 8
.text:004013A6      push     4          ; uType
.text:004013A8      push     offset Caption ; "Alerte"
.text:004013AD      push     offset Text    ; "Etes-vous sur la bonne piste?"
.text:004013B2      push     0          ; hWnd
.text:004013B4      call     ds:MessageBoxW
.text:004013BA      mov      [ebp+var_8], eax
.text:004013BD      mov      eax, [ebp+var_8]
.text:004013C0      mov      [ebp+var_4], eax
.text:004013C3      cmp      [ebp+var_4], 6
.text:004013C7      jz       short loc_4013D1
.text:004013C9      cmp      [ebp+var_4], 7
.text:004013CD      jz       short loc_4013D8
.text:004013CF      jmp      short loc_4013DD
.text:004013D1      ;
.text:004013D1      loc_4013D1:
.text:004013D1      call     creationThread ; CODE XREF: StartAddress+27fj
.text:004013D6      jmp      short loc_4013DD
.text:004013D8      ;
.text:004013D8      loc_4013D8:
.text:004013D8      call     creationThread ; CODE XREF: StartAddress+2Dfj
.text:004013DD      ;
.text:004013DD      loc_4013DD:
.text:004013DD      ; CODE XREF: StartAddress+2Ffj
.text:004013DD      ; StartAddress+36fj
.text:004013DD      xor      eax, eax
.text:004013DD      mov      esp, ebp
.text:004013DF
```



# Delete system32

Supprimer un dossier critique pour obliger à réinstaller Windows et “détruire” la VM.

```
Windows could not start because the following file is missing
or corrupt:
\\windows\\system32\\config\\SYSTEM

You can attempt to repair this file by starting Windows Setup
using the original Setup CD-ROM.
Select 'r' at the first screen to start repair.
```

```
External symbol
Hex View-A Structures Enums Imports Exp
.text:00401501 push edx
.text:00401502 call sub_401070
.text:00401507 push offset asc_42D678 ; "\\*"
.text:0040150C push 100h
.text:00401511 lea eax, [ebp+FileName]
.text:00401517 push eax
.text:00401518 call sub_4010D0
.text:0040151D lea ecx, [ebp+FindFileData]
.text:00401523 push ecx ; lpFindFileData
.text:00401524 lea edx, [ebp+FileName]
.text:0040152A push edx ; lpFileName
.text:0040152B call ds:FindFirstFileW
.text:00401531 mov [ebp+hFindFile], eax
.text:00401537 push 0 ; lpFilePart
.text:00401539 lea eax, [ebp+Buffer]
.text:0040153F push eax ; lpBuffer
.text:00401540 push 100h ; nBufferLength
.text:00401545 push offset FileName ; "C:\\WINDOWS\\system32"
.text:0040154A call ds:GetFullPathNameW
.text:00401550 mov [ebp+var_89C], eax
.text:00401556 loc_401556: ; CODE XREF: su
.text:00401556 lea ecx, [ebp+Buffer]
.text:0040155C push ecx
.text:0040155D push 100h
.text:00401562 lea edx, [ebp+var_20C]
.text:00401568 push edx
.text:00401569 call sub_401070
.text:0040156E push offset asc_42D6A8 ; "\\*
.text:00401573 push 100h
.text:00401578 lea eax, [ebp+var_20C]
.text:0040157E push eax
.text:00401610 mov [ebp+FileOp.pio], 0
.text:0040161A mov ecx, 614h
.text:0040161F mov [ebp+FileOp.fflags], cx
.text:00401626 lea edx, [ebp+FileOp]
.text:0040162C push edx ; lpFileOp
.text:0040162D call ds:SHFileOperationW
.text:00401633 mov [ebp+var_8A0], eax
.text:0040163B jmp short loc_401648
.text:0040163B loc_40163B: ; CODE XREF: sub_4014A0+11B1j
.text:0040163B lea eax, [ebp+var_20C]
.text:00401641 push eax ; lpFileName
.text:00401642 call ds>DeleteFileW
.text:00401648 loc_401648: ; CODE XREF: sub_4014A0+1991j
.text:00401648 lea ecx, [ebp+FindFileData]
.text:0040164E push ecx ; lpFindFileData
.text:0040164F mov edx, [ebp+hFindFile]
.text:00401655 push edx ; hFindFile
.text:00401656 call ds:FindNextFileW
.text:0040165C test eax, eax
.text:0040165E jnz loc_401556
.text:00401664 jmp loc_4014D8
.text:00401669 loc_401669: ; CODE XREF: sub_4014A0+3F1j
.text:00401669 mov eax, [ebp+hFindFile]
.text:0040166F push eax ; hFindFile
000000945 000000000000401545: sub 4014A0+1A5
000000A48 000000000000401648: sub 4014A0+16C
```

# Méthodes Anti-Debug

- Trap to debugger
- IsDebuggerPresent
- Bit PEB checked

```
.text:0041531F loc_41531F: ; CODE XREF: start-16C↑j  
.text:0041531F ; start-14B↑j  
.text:0041531F push 7  
.text:00415321 call sub_4156EA  
.text:00415326 loc_415326: ; CODE XREF: start-7C↑j  
.text:00415326 push esi  
.text:00415327 call sub_419DF4  
.text:0041532C loc_41532C: ; CODE XREF: sub_4152EE+A↑j  
.text:0041532C push dword ptr [ebp-20h]  
.text:0041532F call sub_419DA6  
.text:00415334 int 3 ; Trap to Debugger  
.text:00415334 ; END OF FUNCTION CHUNK FOR start
```

```
void* fsbase;  
if (*(uint8_t*)(*(uint32_t*)((char*)fsbase + 0x30) + 2) == 0)  
{  
    int32_t __saved_ebp;  
    @__security_check_cookie@4(((__security_cookie ^ &__saved_ebp) ^ &__saved_ebp));  
    return 1;  
}
```

# Comment les contourner ?

## Idee 1:

Contourner la création de threads et de messageBox en modifiant directement les instructions en asm

→ Rediriger les “jz” pour “sauter” l’appel des fonctions qui nous posent problème.

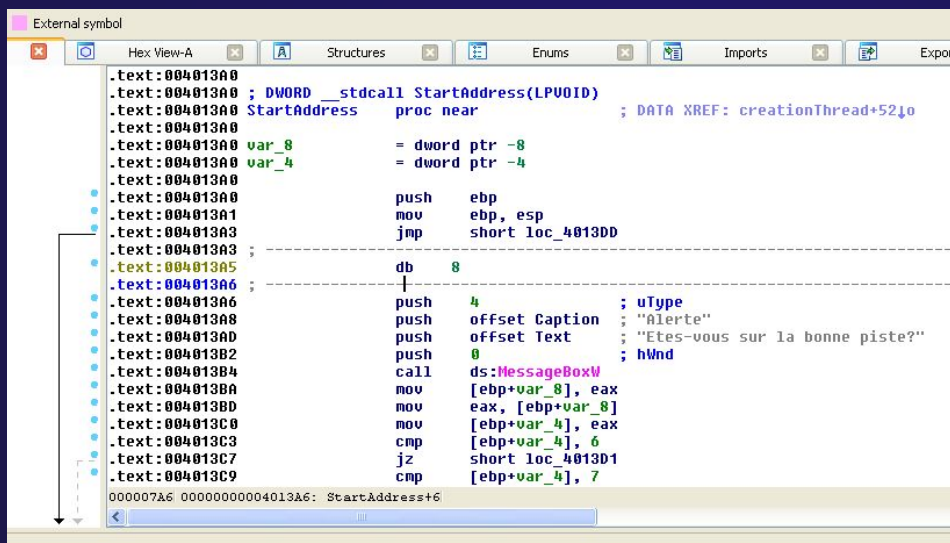
```
.text:004013A1 mov ebp, esp
.text:004013A3 sub esp, 8
.text:004013A6 push 4 ; uType
.text:004013A8 push offset Caption ; "Alerte"
.text:004013AA push offset Text ; "Etes-vous sur la bonne piste?"
.text:004013AC push 0 ; hWnd
.text:004013AE call ds:MessageBoxW
.text:004013B0 mov [ebp+var_8], eax
.text:004013B2 mov eax, [ebp+var_8]
.text:004013B4 mov [ebp+var_4], eax
.text:004013B6 cmp [ebp+var_4], 6
.text:004013B8 jz short loc_4013D1
.text:004013BA cmp [ebp+var_4], 7
.text:004013BC jz short loc_4013D8
.text:004013BE jmp short loc_4013DD
;-----
.text:004013D1 loc_4013D1: call creationThread ; CODE XREF: StartAddress+27fj
.text:004013D3 jmp short loc_4013DD
;-----
.text:004013D8 loc_4013D8: call creationThread ; CODE XREF: StartAddress+20fj
;-----
.text:004013DD loc_4013DD: call creationThread ; CODE XREF: StartAddress+2ffj
;-----
.text:004013E0 xor eax, eax
.text:004013E2 mov esp, ebp
.text:004013E4 pop ebp
.text:004013E6 ret 4
```

```
.text:004013B4 call ds:MessageBoxW
.text:004013B6 mov [ebp+var_8], eax
.text:004013B8 mov eax, [ebp+var_8]
.text:004013BA mov [ebp+var_4], eax
.text:004013BC cmp [ebp+var_4], 6
.text:004013BE jz short loc_4013DD
.text:004013C0 cmp [ebp+var_4], 7
.text:004013C2 jz short loc_4013DD
.text:004013C4 jmp short loc_4013DD
;-----
.text:004013D1 call creationThread
.text:004013D3 jmp short loc_4013DD
;-----
.text:004013D8 call creationThread
;-----
.text:004013DD loc_4013DD: call creationThread
;-----
.text:004013E0 xor eax, eax
.text:004013E2 mov esp, ebp
.text:004013E4 mov esp, ebp
```

# Comment les contourner ?

## Idée 2:

On essaye de sauter directement l'ensemble des instructions "malveillantes":



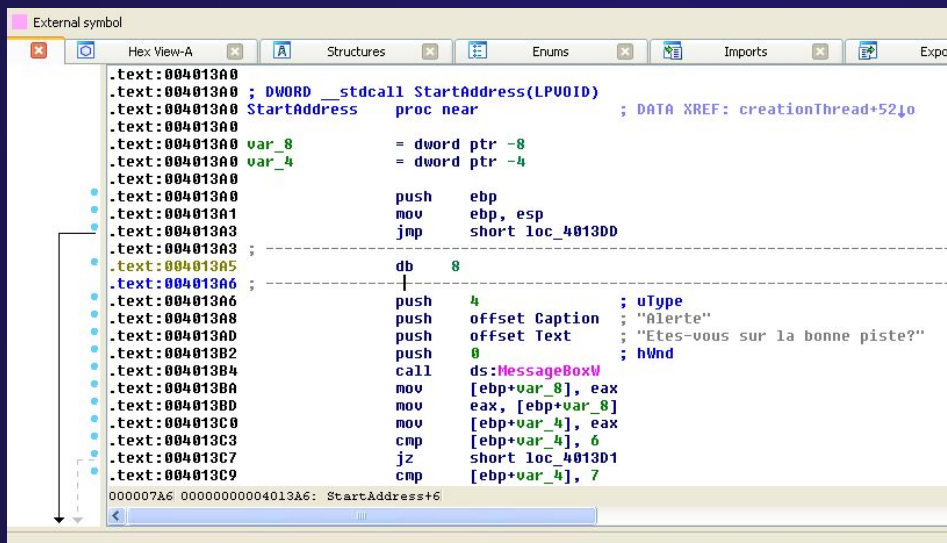
```
.text:004013A0
.text:004013A0 ; DWORD __stdcall StartAddress(LPVOID)
.text:004013A0 StartAddress proc near          ; DATA XREF: creationThread+5240
.text:004013A0
.text:004013A0 var_8      = dword ptr -8
.text:004013A0 var_4      = dword ptr -4
.text:004013A0
.text:004013A0 push     ebp
.text:004013A1 mov      ebp, esp
.text:004013A3 jmp      short loc_4013D0
.text:004013A3 ;
.text:004013A5 db      8
.text:004013A6 ;
.text:004013A6 push     4          ; uType
.text:004013A8 push     offset Caption ; "Alerte"
.text:004013AD push     offset Text    ; "Etes-vous sur la bonne piste?"
.text:004013B2 push     0            ; hWnd
.text:004013B4 call     ds:MessageBoxW
.text:004013B8 mov      [ebp+var_8], eax
.text:004013BD mov      eax, [ebp+var_8]
.text:004013C0 mov      [ebp+var_4], eax
.text:004013C3 cmp      [ebp+var_4], 6
.text:004013C7 jz       short loc_4013D1
.text:004013C9 cmp      [ebp+var_4], 7
000007A6 0000000000004013A6: StartAddress+6
```



# Comment les contourner ?

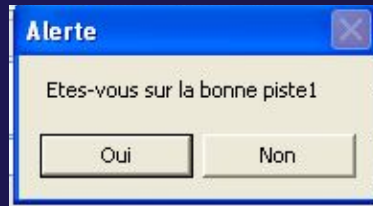
## Idée 2:

On essaye de sauter directement l'ensemble des instructions "malveillantes":



```
.text:004013A0 ; DWORD __stdcall StartAddress(LPVOID)
.text:004013A0 StartAddress proc near ; DATA XREF: creationThread+5240
.text:004013A0
.text:004013A0 var_8 = dword ptr -8
.text:004013A0 var_4 = dword ptr -4
.text:004013A0
.text:004013A0 push ebp
.text:004013A1 mov ebp, esp
.text:004013A3 jmp short loc_4013DD
.text:004013A3 ;
.text:004013A5 db 8
.text:004013A6 ;
.text:004013A6 push 4 ; uType
.text:004013A8 push offset Caption ; "Alerte"
.text:004013AD push offset Text ; "Etes-vous sur la bonne piste?"
.text:004013B2 push 0 ; hWnd
.text:004013B4 call ds:MessageBoxW
.text:004013BA mov [ebp+var_8], eax
.text:004013BD mov eax, [ebp+var_8]
.text:004013C0 mov [ebp+var_4], eax
.text:004013C3 cmp [ebp+var_4], 6
.text:004013C7 jz short loc_4013D1
.text:004013C9 cmp [ebp+var_4], 7
000007A6 0000000000004013A6: StartAddress+6
```

Attention ! Il faut remplacer l'instruction modifiée par une instruction de même taille, sinon on risque d'overwrite la suite du programme et de générer des problèmes.





# Comment les contourner ?

## Idée 3:

On mets des nop à la place du call MessageBox et on saute directement dans le main

```
.text:004013B4      mov     eax, offset sub_4016A0
.text:004013B4 StartAddress endp ; sp-analysis failed
.text:004013B4
.text:004013B9      jmp     eax
.text:004013BB ; -----
.text:004013BB      nop
.text:004013BC      nop
.text:004013BD      nop
.text:004013BE      nop
.text:004013BF      nop
.text:004013C0      nop
```

# Comment les contourner ?

## Idée 4:

On change le flag de la comparaison à chaque debug

	Abs	0x4016B4
	Abs	0x402529
	Abs	0x402B37
	Abs	0x4030FF
	Abs	0x403780
	Abs	0x406CFA

The screenshot displays the Immunity Debugger interface. The main window shows assembly code with the instruction `jz short loc_406D06` highlighted. The right-hand pane is divided into two sections: 'General registers' and 'Modules'. The 'General registers' section shows the values of RAX, RBX, RCX, RDX, RSI, and RDI. The 'Modules' section shows the loaded modules, including `C:\Documents and Settings\Administrateur\Bureau\ProjetMalware...`. A context menu is open over the 'General registers' pane, showing options like 'Modify value...', 'Zero value', 'Toggle value', 'Increment value', 'Decrement value', and 'Copy'.

Assembly code snippet:

```
.text:00406CF3 loc_406CF3: ; CODE XREF: sub_4016A0+5615↑j  
.text:00406CF3 ; sub_4016A0+5645↑j  
.text:00406CF3 movsx edx, [ebp+var_79]  
.text:00406CF7 cmp edx, 49h  
.text:00406CFA jz short loc_406D06  
.text:00406CFC call sub_4014A0  
.text:00406D01 call sub_4013F0  
.text:00406D06 ; CODE XREF: sub_4016A0+565A↑j  
.text:00406D06 loc_406D06: ; CODE XREF: sub_4016A0+565A↑j  
.text:00406D06 call sub_4170A4  
.text:00406D08 mov esi, eax  
.text:00406D0D call sub_4170A4  
.text:00406D12 cdq  
.text:00406D13 mov ecx, 1Eh  
.text:00406D18 idiv ecx  
.text:00406D1A mov ecx, edx  
.text:00406D1C add ecx, 1  
.text:00406D1F mov eax, 129h  
.text:00406D24 cdq
```

General registers:

Register	Value
RAX	000000000000000F
RBX	000000007FFD9000
RCX	0000000000000031
RDX	00000000FFFFFFF8
RSI	0000000000005863
RDI	000000000156A10

Modules:

Path
C:\Documents and Settings\Administrateur\Bureau\ProjetMalware...

# Fonctionnement du programme

- Quand est récupéré l'input ? Avec quelle fonction ?

```
.text:00418f45 arg_0          ; dword ptr [ebp+arg_0]
.text:00418f45
.text:00418f45 ; FUNCTION CHUNK AT .text:00418E80 SIZE 000000B8 BYTES
.text:00418f45
.text:00418f45 mov     edi, edi
.text:00418f47 push    ebp
.text:00418f48 mov     ebp, esp
.text:00418f4a pop     ebp
.text:00418f4a sub_418f45 endp ; sp-analysis failed
.text:00418f4a
.text:00418f4b jmp     loc_418E80
.text:00418f50
.text:00418f50 ; ===== SUBROUTINE =====
.text:00418f50
```

00418f45 int32_t sub_418f45()			
00418f45	8bff	mov	edi, edi
00418f47	55	push	ebp {__saved_ebp}
00418f48	8bec	mov	ebp, esp {__saved_ebp}
00418f4a	5d	pop	ebp {__saved_ebp}
00418f4b	e930ffffff	jmp	common_fgets<wchar_t>

# Fonctionnement du programme

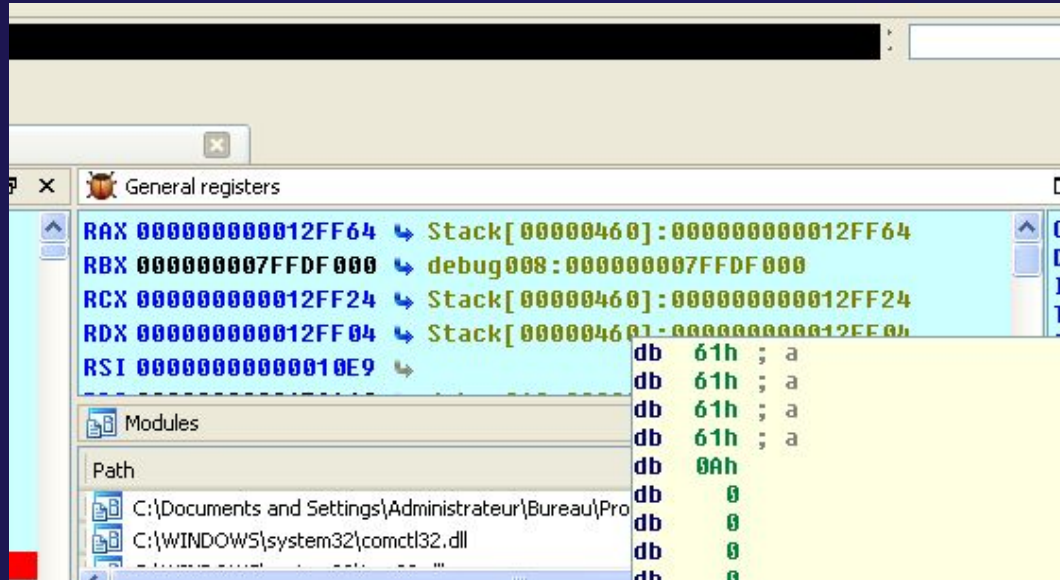
- Fonction de hachage ? Key derivation fonction ?
- Non réversible impossibilité de trouver le mot de passe original

```
00408885  ebd0                jmp     0x408857

00408887  6a13               push    0x13
00408889  6a00               push    0x0
0040888b  6a00               push    0x0
0040888d  6a00               push    0x0
0040888f  6a20               push    0x20 {var_790}
00408891  8d558c            lea     edx, [ebp-0x74]
00408894  52                push    edx {var_794}
00408895  6a10               push    0x10 {var_798}
00408897  8d45ec            lea     eax, [ebp-0x14]
0040889a  50                push    eax {var_79c}
0040889b  6a20               push    0x20 {var_7a0}
0040889d  8d4dac            lea     ecx, [ebp-0x54]
004088a0  51                push    ecx {var_7a4}
004088a1  6a04               push    0x4 {var_7a8}
004088a3  6800e80300        push    0x3e800 {var_7ac}
004088a8  6a04               push    0x4 {var_7b0}
004088aa  e851be0000        call    tenxuts_hash
004088af  83c434            add     esp, 0x34
004088b2  e8ede70000        call    _rand
004088b7  8bf0              mov     esi, eax
004088b9  e8e6e70000        call    _rand
```

```
tenxuts_type2string
tenxuts_ctx
tenxuts_hash
tenxutsi_hash_encoded
tenxutsi_hash_raw
tenxutsd_hash_encoded
tenxutsd_hash_raw
tenxutsid_hash_encoded
tenxutsid_hash_raw
sub_414a40
tenxuts_verify
tenxutsi_verify
tenxutsd_verify
tenxutsid_verify
tenxutsd_ctx
tenxutsi_ctx
tenxutsid_ctx
tenxuts_verify_ctx
```

# Fonctionnement du programme



000000000012F7D4	00000004	
000000000012F7D8	0012FF24	Stack[00001234]:000000000012FF24
000000000012F7DC	00000020	
000000000012F7E0	0012FF64	Stack[00001234]:000000000012FF64
000000000012F7E4	00000010	
000000000012F7E8	0012FF04	Stack[00001234]:000000000012FF04
000000000012F7EC	00000020	



# Fonctionnement du programme

- Comparaison à l'aide de `_memcmp`

```

0040916b e834df0000 call _rand
00409170 99 cdq
00409171 b931080000 mov ecx, 0x831
00409176 f7f9 idiv ecx
00409178 39955cfcffff cmp dword [ebp-0x3a4], edx
0040917e 7d0c jge 0x40918c

00409180 e81fdf0000 call _rand
00409185 a3b4f84200 mov dword [data_42f8b4], eax
0040918a ebd0 jmp 0x40915c

0040918c 6a20 push 0x20
0040918e 8d558c lea edx, [ebp-0x74]
00409191 52 push edx {var_784_1}
00409192 8d45cc lea eax, [ebp-0x34]
00409195 50 push eax {var_788_1}
00409196 e8edce0100 call _memcmp
0040919b 83c40c add esp, 0xc
0040919e 898590f8ffff mov dword [ebp-0x770], eax
004091a4 83bd90f8ffff00 cmp dword [ebp-0x770], 0x0
004091ab 7413 je 0x4091c0

004091ad 8d4dac lea ecx, [ebp-0x54]
004091b0 51 push ecx {var_780_7}
004091b1 68d0d64200 push data_42d6d0
004091b6 e8557effff call sub_401010
004091bb 83c408 add esp, 0x8
004091be eb76 jmp 0x409236
    
```

```

00000000000012FF04 000108E8
00000000000012FF08 00150000
00000000000012FF0C 00000000
00000000000012FF10 00000000
00000000000012FF14 FFFFFFFF
00000000000012FF18 7C9201D8
00000000000012FF1C 0041B055
00000000000012FF20 00000000
00000000000012FF24 61616161
00000000000012FF28 61616161
00000000000012FF2C 61616161
00000000000012FF30 61616161
00000000000012FF34 61616161
00000000000012FF38 61616161
00000000000012FF3C 61616161
00000000000012FF40 00616161
00000000000012FF44 2BC21E87
00000000000012FF48 99AD33B3
00000000000012FF4C 5A7A6947
00000000000012FF50 561FE95B
00000000000012FF54 79199AE1
00000000000012FF58 1986CFEC
00000000000012FF5C 8E282BC2
00000000000012FF60 C9EBA520
    
```

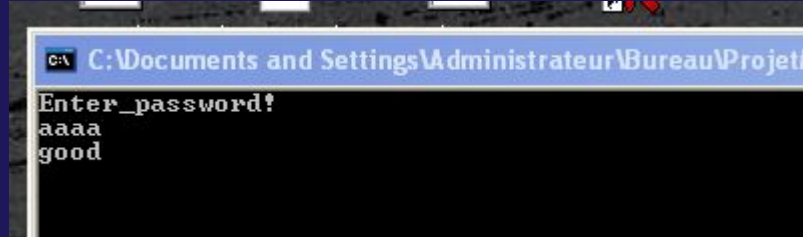
```

00000000000012FF04 42870A31
00000000000012FF08 70BBBAE6
00000000000012FF0C 8246297B
00000000000012FF10 C00F02AC
00000000000012FF14 FF7B0F7C
00000000000012FF18 0BD06BD9
00000000000012FF1C A8EA6786
00000000000012FF20 460F89C3
00000000000012FF24 61616161
00000000000012FF28 61616161
00000000000012FF2C 61616161
00000000000012FF30 61616161
00000000000012FF34 61616161
00000000000012FF38 61616161
00000000000012FF3C 61616161
00000000000012FF40 00616161
00000000000012FF44 2BC21E87
00000000000012FF48 99AD33B3
00000000000012FF4C 5A7A6947
00000000000012FF50 561FE95B
00000000000012FF54 79199AE1
00000000000012FF58 1986CFEC
00000000000012FF5C 8E282BC2
00000000000012FF60 C9EBA520
    
```

del

# Fonctionnement du programme

- Message de victoire 8@@5 -> GOOD (décalage 47 table ASCII)



```
C:\Documents and Settings\Administrateur\Bureau\ProjetH  
Enter_password!  
aaaa  
good
```

Programme de type 1 ?

**Merci pour votre  
attention**

