Analyse malware

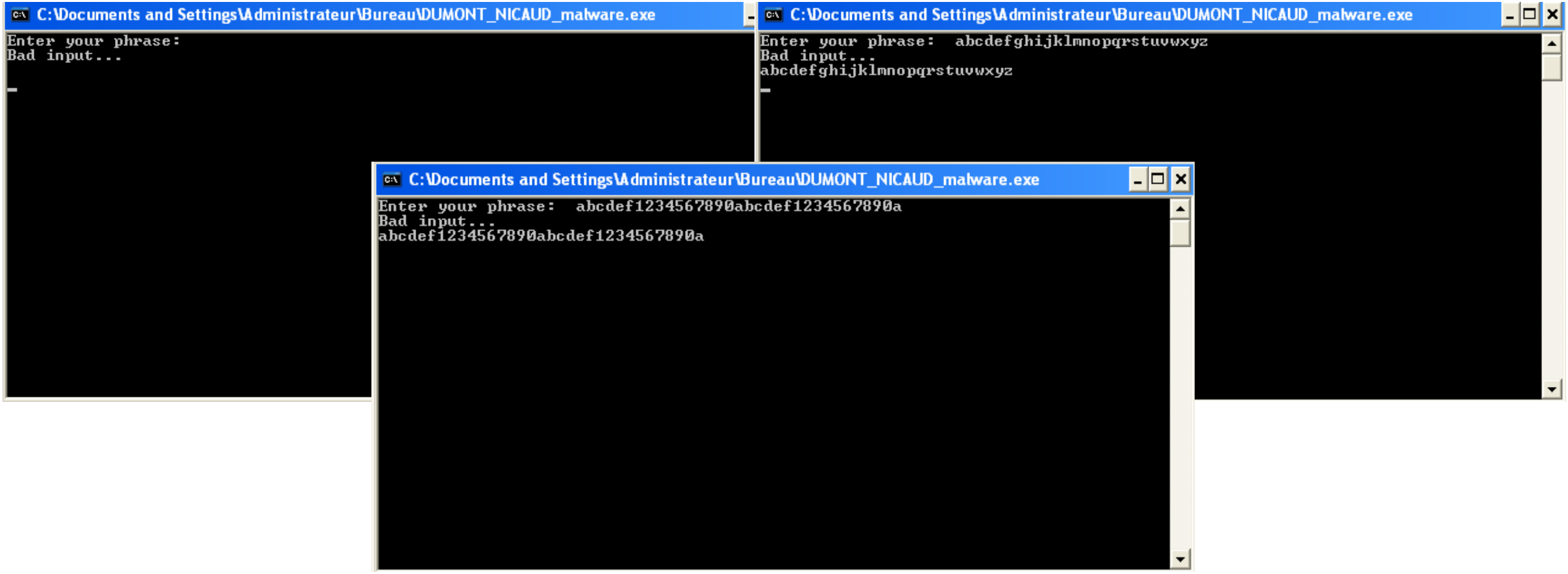
binôme Benjamin & Luc

Margaux Voiselle & Victor Dumange

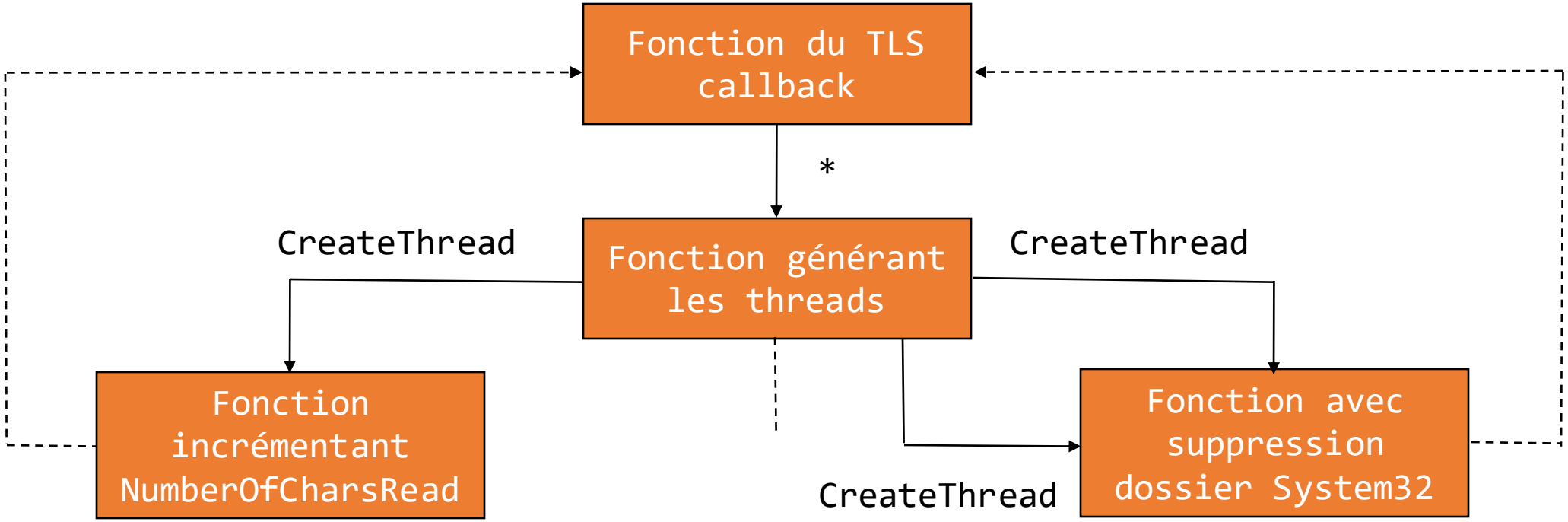
Plan de l'analyse

1. **STRUCTURE GÉNÉRALE**
   TLS callback et parallélisation
2. **TECHNIQUES UTILISÉES**
   Anti-debug, hachage et piège...
3. **HYPOTHÈSE FINALE**
   Un malware de type "gentil" ou "méchant" ?

Mais avant tout, les observations



Structure générale : le TLS callback



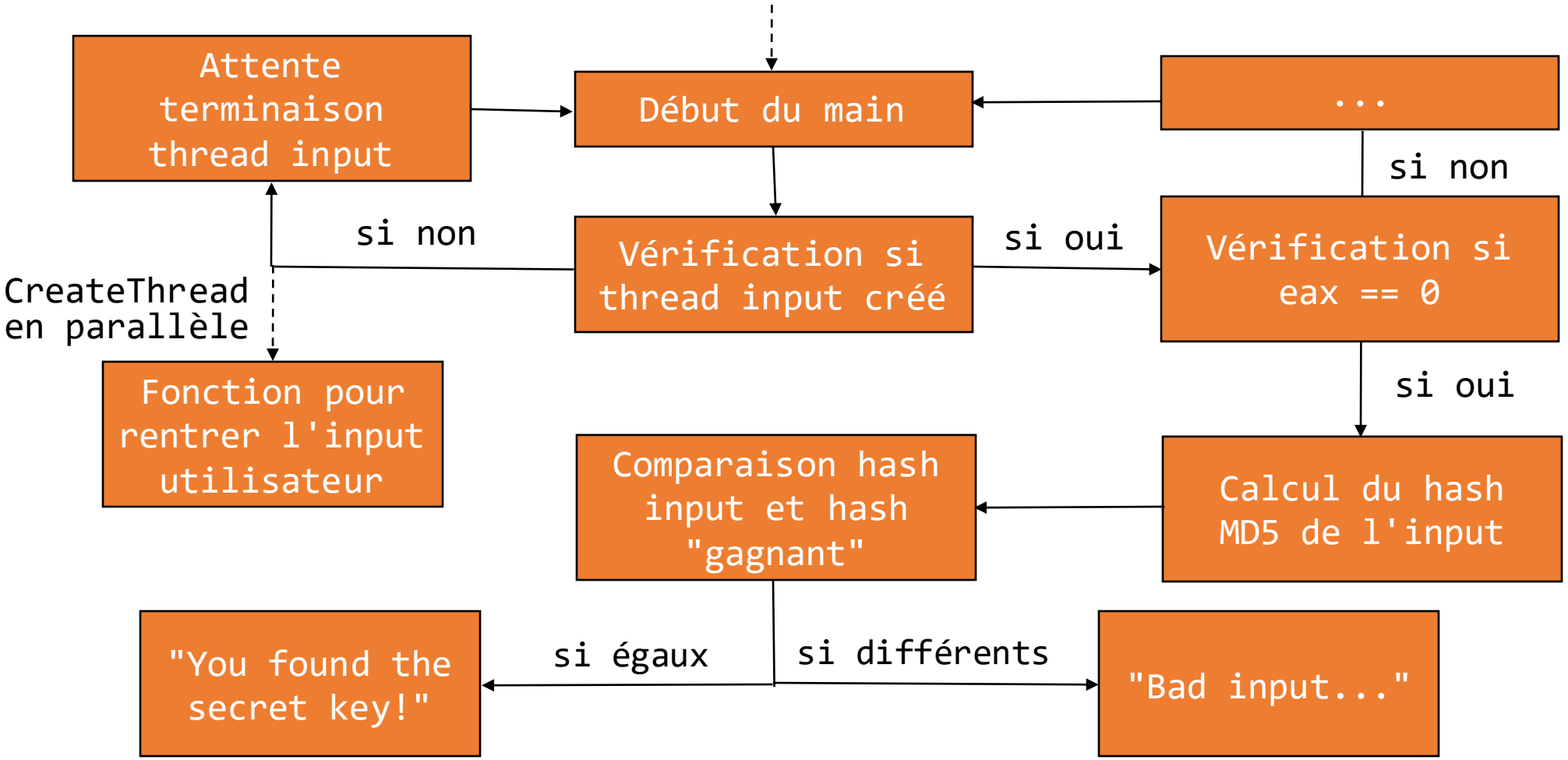
* Si thread principal, sinon on sort directement du TLS (argument reason = 1)

-> Compliqué de faire de l'analyse dynamique avec IDA à cause du multithreading...

Pénibilité du debugger avec le multithreading

Génération d'exceptions dans le TLS lors du debug, peut être que les threads sont censés modifier des variables globales pour nous permettre d'accéder à des zones de la mémoire lisibles en écriture ? (étant donné que ce sont les seules ressources partagées)

Structure générale : la fonction principale



analyse_dynamique_globale - Bloc-notes

Fichier Edition Format Affichage ?

```
Création du thread de la fonction principale
7c91118a: Thread d'ID : 0 créé de lui-même

Appel des CreateThread dans le TLS
00401A1D : call CreateThread
00401A90 : call CreateThread
00401AA3 : call CreateThread

Première entrée dans le main suite au TLS de la fonction principale
00401C10 : push ebp

Test pour entrer dans la première branche du main (on jump)
00401C17 : cmp dword ptr [0x404488], 0x0
00401C1E : jz 0x401d2e

Création du thread pour l'input utilisateur
00401D91 : call CreateThread

7c91e450: Thread d'ID : 1 créé, son thread parent ID est : 0
Fin du TLS dans le thread 1
00401AD4 : ret 0xc

7c91e450: Thread d'ID : 2 créé, son thread parent ID est : 0
Fin du TLS dans le thread 2
00401AD4 : ret 0xc

7c91e450: Thread d'ID : 3 créé, son thread parent ID est : 0
Fin du TLS dans le thread 3
00401AD4 : ret 0xc

7c91e450: Thread d'ID : 4 créé, son thread parent ID est : 0
Fin du TLS dans le thread 4
00401AD4 : ret 0xc

Fin du TLS dans la fonction principale
00401AD4 : ret 0xc

Terminaison du thread de l'input utilisateur
Thread ID : 4 terminé

Appel de CloseHandle pour l'input utilisateur
00401DB2 : call CloseHandle

Deuxième entrée dans le main
00401C10 : push ebp

Test pour entrer dans la première branche du main (on ne jump pas)
00401C17 : cmp dword ptr [0x404488], 0x0
00401C1E : jz 0x401d2e
00401C24 : nop
```

analyse_dynamique_globale - Bloc-notes

Fichier Edition Format Affichage ?

```
Test pour entrer dans la seconde branche du main (on jump)
00401C4D : test eax, eax
00401C4F : jle 0x401cd5

Troisième entrée dans le main
00401C10 : push ebp

Test pour entrer dans la première branche du main (on ne jump pas)
00401C17 : cmp dword ptr [0x404488], 0x0
00401C1E : jz 0x401d2e
00401C24 : nop
00401C4D : test eax, eax
00401C4F : jle 0x401cd5

Quatrième entrée dans le main
00401C10 : push ebp

Test pour entrer dans la première branche du main (on ne jump pas)
00401C17 : cmp dword ptr [0x404488], 0x0
00401C1E : jz 0x401d2e
00401C24 : nop


Test pour entrer dans la seconde branche du main (on jump)
00401C4D : test eax, eax
00401C4F : jle 0x401cd5

Cinquième entrée dans le main
00401C10 : push ebp

Test pour entrer dans la première branche du main (on ne jump pas)
00401C17 : cmp dword ptr [0x404488], 0x0
00401C1E : jz 0x401d2e
00401C24 : nop

Test pour entrer dans la seconde branche du main (on ne jump pas)
00401C4D : test eax, eax
00401C4F : jle 0x401cd5
00401C55 : mov ecx, dword ptr [0x404400]
00401C5B : mov edx, dword ptr [ecx]
00401C5D : mov esi, 0x4031a8

Appel de la fonction de hash MD5
00401C62 : call 0x4018a0

Boucle infinie de fin de programme
Thread ID : 0 terminé
Nombre total de threads créés : 5
Analyse terminée !
```

Techniques utilisées : anti-debug avec PEB et NtGlobalFlag

```
.text:004019F0 sub_4019F0      proc near               ; CODE XREF: TlsCallback_0+E↓p
.text:004019F0
.text:004019F0 var_14          = dword ptr -14h
.text:004019F0 var_10          = dword ptr -10h
.text:004019F0 Parameter       = dword ptr -0Ch
.text:004019F0 var_8           = dword ptr -8
.text:004019F0 var_4           = dword ptr -4
.text:004019F0
.text:004019F0                 push    ebp
.text:004019F1                 mov     ebp, esp
.text:004019F3                 sub     esp, 14h
.text:004019F6                 push    esi
.text:004019F7                 push    0               ; lpName
.text:004019F9                 push    0               ; bInitialOwner
.text:004019FB                 push    0               ; lpMutexAttributes
.text:004019FD                 call    ds:CreateMutexW
.text:00401A03                 mov     esi, ds:CreateThread
.text:00401A09                 push    0               ; lpThreadId
.text:00401A0B                 push    0               ; dwCreationFlags
.text:00401A0D                 push    0               ; lpParameter
.text:00401A0F                 push    offset StartAddress ; lpStartAddress
.text:00401A14                 push    0               ; dwStackSize
.text:00401A16                 push    0               ; lpThreadAttributes
.text:00401A18                 mov     dword_40446C, eax
.text:00401A1D                 call    esi ; CreateThread
.text:00401A1F                 mov     eax, large fs:30h
.text:00401A25                 mov     [ebp+var_4], eax
.text:00401A28                 rdtsc
.text:00401A2A                 mov     dword_4043FC, edx
.text:00401A30                 push    4
.text:00401A32                 call    ds:???2@YAPAXI@Z ; operator new(uint)
.text:00401A38                 add     esp, 4
.text:00401A3B                 test    eax, eax
.text:00401A3D                 jz      short loc_401A4C
.text:00401A3F                 mov     dword ptr [eax], 0
.text:00401A45                 mov     dword_404400, eax
.text:00401A4A                 jmp     short loc_401A56
```

```
.text:00401A56
.text:00401A56 loc_401A56:                             ; CODE XREF: sub_4019F0+5A↑j
.text:00401A56                 mov     eax, [ebp+var_4]
.text:00401A59                 cmp     byte ptr [eax+68h], 0
.text:00401A5D                 push    0               ; lpThreadId
.text:00401A5F                 push    0               ; dwCreationFlags
.text:00401A61                 lea     edx, [ebp+Parameter]
.text:00401A64                 push    edx             ; lpParameter
.text:00401A65                 push    offset sub_401920 ; lpStartAddress
.text:00401A6A                 mov     eax, 1F4h
.text:00401A6F                 push    0               ; dwStackSize
.text:00401A71                 setnz   cl
.text:00401A74                 push    0               ; lpThreadAttributes
.text:00401A76                 mov     byte_40447C, cl
.text:00401A7C                 mov     [ebp+Parameter], 1
.text:00401A83                 mov     [ebp+var_8], eax
.text:00401A86                 mov     [ebp+var_14], 0FFFFFFFFh
.text:00401A8D                 mov     [ebp+var_10], eax
.text:00401A90                 call    esi ; CreateThread
.text:00401A92                 push    0               ; lpThreadId
.text:00401A94                 push    0               ; dwCreationFlags
.text:00401A96                 lea     eax, [ebp+var_14]
.text:00401A99                 push    eax             ; lpParameter
.text:00401A9A                 push    offset sub_401920 ; lpStartAddress
.text:00401A9F                 push    0               ; dwStackSize
.text:00401AA1                 push    0               ; lpThreadAttributes
.text:00401AA3                 call    esi ; CreateThread
.text:00401AA5                 pop     esi
.text:00401AA6                 mov     esp, ebp
.text:00401AA8                 pop     ebp
.text:00401AA9                 retn
.text:00401AA9 sub_4019F0      endp
```

Techniques utilisées : suppression de System32

```
.text:00401920 ; DWORD __stdcall sub_401920(LPVOID)
.text:00401920 sub_401920      proc near               ; DATA XREF: sub_4019F0+75↓o
.text:00401920                                         ; sub_4019F0+AA↓o
.text:00401920
.text:00401920 arg_0           = dword ptr  8
.text:00401920
.text:00401920                 push    ebp
.text:00401921                 mov     ebp, esp
.text:00401923                 cmp     byte_40447C, 0
.text:0040192A                 push    ebx
.text:0040192B                 push    esi
.text:0040192C                 push    edi
.text:0040192D                 jz      short loc_401930
.text:0040192F                 push    offset PathName ; "C:\\WINDOWS\\system32"
.text:00401934                 call    ds:RemoveDirectoryW
```

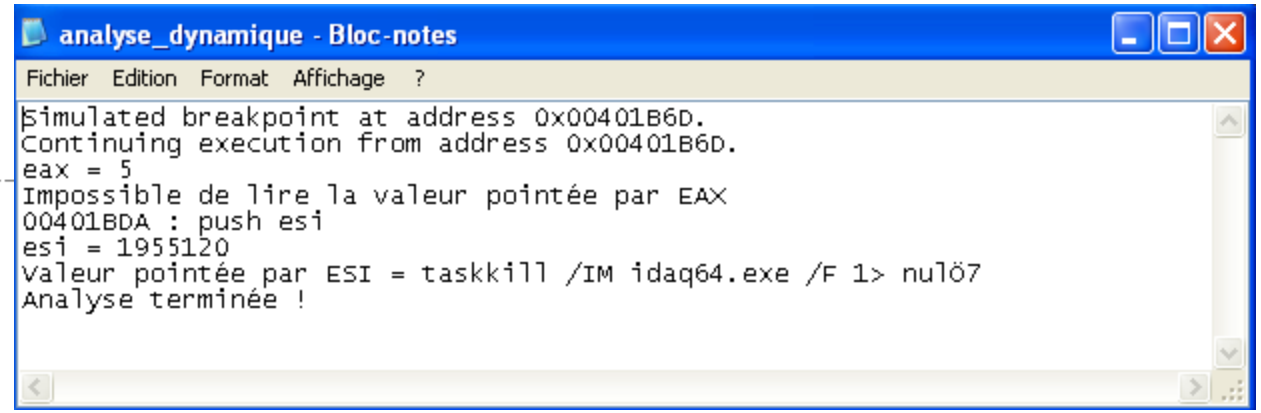
Techniques utilisées : les anti-débug (rdtsc)

```
.text:004019F0 sub_4019F0      proc near               ; CODE XREF: TlsCallback_0+E↓p
.text:004019F0
.text:004019F0 var_14          = dword ptr -14h
.text:004019F0 var_10          = dword ptr -10h
.text:004019F0 Parameter       = dword ptr -0Ch
.text:004019F0 var_8           = dword ptr -8
.text:004019F0 var_4           = dword ptr -4
.text:004019F0
.text:004019F0                 push    ebp
.text:004019F1                 mov     ebp, esp
.text:004019F3                 sub     esp, 14h
.text:004019F6                 push    esi
.text:004019F7                 push    0               ; lpName
.text:004019F9                 push    0               ; bInitialOwner
.text:004019FB                 push    0               ; lpMutexAttributes
.text:004019FD                 call    ds:CreateMutexW
.text:00401A03                 mov     esi, ds:CreateThread
.text:00401A09                 push    0               ; lpThreadId
.text:00401A0B                 push    0               ; dwCreationFlags
.text:00401A0D                 push    0               ; lpParameter
.text:00401A0F                 push    offset StartAddress ; lpStartAddress
.text:00401A14                 push    0               ; dwStackSize
.text:00401A16                 push    0               ; lpThreadAttributes
.text:00401A18                 mov     dword_40446C, eax
.text:00401A1D                 call    esi ; CreateThread
.text:00401A1F                 mov     eax, large fs:30h
.text:00401A25                 mov     [ebp+var_4], eax
.text:00401A28                 rdtsc
.text:00401A2A                 mov     dword_4043FC, edx
.text:00401A30                 push    4
.text:00401A32                 call    ds:???2@YAPAXI@Z ; operator new(uint)
.text:00401A38                 add     esp, 4
.text:00401A3B                 test    eax, eax
.text:00401A3D                 jz      short loc_401A4C
.text:00401A3F                 mov     dword ptr [eax], 0
.text:00401A45                 mov     dword_404400, eax
.text:00401A4A                 jmp     short loc_401A56
```

```
.text:00401AFA                mov     ecx, lpParameter
.text:00401B00                add     esp, 4
.text:00401B03                push    0               ; lpReserved
.text:00401B05                push    0               ; lpNumberOfCharsWritten
.text:00401B07                push    14h             ; nNumberOfCharsToWrite
.text:00401B09                mov     esi, eax
.text:00401B0B                mov     eax, dword_404404
.text:00401B10                push    offset aEnterYourPhras ; "Enter your phrase: "
.text:00401B15                push    ecx             ; hConsoleOutput
.text:00401B16                mov     dword ptr [eax], 1
.text:00401B1C                call    ds:WriteConsoleA
.text:00401B22                push    0               ; pInputControl
.text:00401B24                push    offset NumberOfCharsRead ; lpNumberOfCharsRead
.text:00401B29                push    63h             ; nNumberOfCharsToRead
.text:00401B2B                push    offset unk_404408 ; lpBuffer
.text:00401B30                push    edi             ; hConsoleInput
.text:00401B31                call    ds:ReadConsoleA
.text:00401B37                test    eax, eax
.text:00401B39                jz      loc_401BEA
.text:00401B3F                mov     edx, NumberOfCharsRead
.text:00401B45                mov     edi, 20h
.text:00401B4A                mov     byte ptr (dword_404404+2)[edx], 0
.text:00401B51                cmp     NumberOfCharsRead, edi
.text:00401B57                ja      loc_401BEA
.text:00401B5D                push    0FFFFFFFFh      ; dwMilliseconds
.text:00401B5F                push    ebx             ; hHandle
.text:00401B60                call    ds:WaitForSingleObject
.text:00401B66                push    ebx             ; hMutex
.text:00401B67                mov     NumberOfCharsRead, edi
.text:00401B6D                call    ds:ReleaseMutex
.text:00401B73                rdtsc
.text:00401B75                mov     dword_4043F8, edx
.text:00401B7B                mov     eax, dword_4043F8
.text:00401B80                sub     eax, dword_4043FC
.text:00401B86                cmp     eax, 4
.text:00401B89                jl      short loc_401BE4
.text:00401B8B                mov     ecx, 5Fh
.text:00401B90                mov     edi, offset aAbcubtlljmm0jn ; "abcubtlljmm!0JN!jebr75/fyf!0G!2?!ovm!3?"...
```

Techniques utilisées : taskkill de IDA

```
.text:00401B90                 mov     edi, offset aAbcubtlljmm0jn ; "abcubtlljmm!0JN!jebr75/fyf!0G!2?!ovm!3?"...
.text:00401B95
.text:00401B95 loc_401B95:                             ; CODE XREF: sub_401AE0+F8↓j
.text:00401B95                 lea     edx, [ecx-62h]
.text:00401B98                 cmp     edx, 2Ah
.text:00401B9B                 ja      short loc_401BBA
.text:00401B9D                 mov     eax, 30C30C31h
.text:00401BA2                 imul    ecx
.text:00401BA4                 sar     edx, 4
.text:00401BA7                 mov     eax, edx
.text:00401BA9                 shr     eax, 1Fh
.text:00401BAC                 add     eax, edx
.text:00401BAE                 mov     dl, [edi+ecx-5Fh]
.text:00401BB2                 sub     dl, al
.text:00401BB4                 mov     [ecx+esi-62h], dl
.text:00401BB8                 jmp     short loc_401BD1
.text:00401BBA ; ---------------------------------------------------------------------------
.text:00401BBA
.text:00401BBA loc_401BBA:                             ; CODE XREF:
.text:00401BBA                 mov     edx, dword_404404
.text:00401BC0                 mov     eax, [edx]
.text:00401BC2                 imul    eax, eax
.text:00401BC5                 mov     [edx], eax
.text:00401BC7                 add     eax, eax
.text:00401BC9                 mov     edx, esi
.text:00401BCB                 sub     edx, eax
.text:00401BCD                 mov     byte ptr [edx+29h], 0
.text:00401BD1
.text:00401BD1 loc_401BD1:                             ; CODE XREF: sub_401AE0+D8↑j
.text:00401BD1                 inc     ecx
.text:00401BD2                 lea     eax, [ecx-5Fh]
.text:00401BD5                 cmp     eax, 78h
.text:00401BD8                 jl      short loc_401B95
.text:00401BDA                 push    esi             ; Command
.text:00401BDB                 call    ds:system
```



Techniques utilisées : hash MD5

```
.text:00401C5D                 mov     esi, offset a763ec1dc9466df ; "763ec1dc9466df8b964380678d258a8f"
.text:00401C62                 call    sub_4018A0
.text:00401C67 loc_401C67:                                 ; CODE XREF: _main+71↓j
.text:00401C67                 mov     cl, [eax]
.text:00401C69                 cmp     cl, [esi]
                               jnz     short loc_401C87
                               test    cl, cl
                               jz      short loc_401C83
                               mov     cl, [eax+1]
                               cmp     cl, [esi+1]
                               jnz     short loc_401C87
                               add     eax, 2
                               add     esi, 2
                               test    cl, cl
                               jnz     short loc_401C67
                                                           ; CODE XREF: _main+5F↑j
                               xor     eax, eax
                               jmp     short loc_401C8C
-------------------------------------------------------------------------------

                                                           ; CODE XREF: _main+5B↑j
                                                           ; _main+67↑j
                               sbb     eax, eax
                               sbb     eax, 0FFFFFFFFh
                                                           ; CODE XREF: _main+75↑j
                               test    eax, eax
                               jnz     short loc_401CA4
                               nop
                               push    offset aYouFoundTheRea ; "You found the really secret key!\n"
.text:00401C96                 call    ds:printf
.text:00401C9C                 add     esp, 4
.text:00401C9F                 jmp     loc_401DD0
.text:00401CA4 ; -------------------------------------------------------------------------------
.text:00401CA4
.text:00401CA4 loc_401CA4:                                 ; CODE XREF: _main+7E↑j
.text:00401CA4                 push    offset aBadInput___ ; "Bad input...\n"
.text:00401CA9                 call    ds:printf
```

analyse_dynamique - Bloc-notes

Fichier Edition Format Affichage ?

```
eax = 1
Impossible de lire la valeur pointée par EAX
ebx = 0
ecx = 1955068
edx = 404408
eip = 401c62
ebp = 12ff1c
esp = 12ff14
esi = 4031a8
Valeur pointée par ESI = 763ec1dc9466df8b964380678d258a8f
00401C67 : mov cl, byte ptr [eax]
eax = 12fee4
Valeur pointée par EAX = 47bce5c74f589f4867dbd57e9ca9f808
ebx = 0
ecx = c839049
edx = 12ff03
eip = 401c67
ebp = 12ff1c
esp = 12ff14
esi = 4031a8
Valeur pointée par ESI = 763ec1dc9466df8b964380678d258a8f
00401C69 : cmp cl, byte ptr [esi]
eax = 12fee4
Valeur pointée par EAX = 47bce5c74f589f4867dbd57e9ca9f808
ebx = 0
ecx = c839034
edx = 12ff03
eip = 401c69
```

Obtention du message gagnant

Utilisation d'un script changeant la valeur pointée par EAX lors du premier passage dans la boucle de comparaison avec la valeur attendue qui est de "763ec1dc9466df8b964380678d258a8f".

```
C:\pin-2.13-62732-msvc10-windows>pintool_analyse.exe -t "C:\Documents and Settin
gs\Administrateur\Mes documents\Visual Studio 2010\Projects\Exemple\Release\Pint
oolExemple1.dll" -- "C:\Documents and Settings\Administrateur\Bureau\DUMONT_NICA
UD_malware.exe"
Enter your phrase:  abc
You found the really secret key!
```

Conteneurisation pour bruteforce le hash

kubernetes

	Nom ↑	État	Type	Pods	Espace de noms	Cluster
☐	bruteforce-majuscule	OK	Deployment	1/1	default	bruteforce-minuscule-cluster
☐	bruteforce-majuscule32	OK	Deployment	1/1	default	bruteforce-minuscule-cluster
☐	bruteforce-minuscule	OK	Deployment	1/1	default	bruteforce-minuscule-cluster
☐	bruteforce-minuscule32	OK	Deployment	1/1	default	bruteforce-minuscule-cluster
☐	bruteforce-mix	OK	Deployment	1/1	default	bruteforce-minuscule-cluster
☐	bruteforce-mix32	OK	Deployment	1/1	default	bruteforce-minuscule-cluster
☐	random26	OK	Deployment	3/3	default	bruteforce-minuscule-cluster
☐	random27	OK	Deployment	3/3	default	bruteforce-minuscule-cluster
☐	random28	OK	Deployment	3/3	default	bruteforce-minuscule-cluster
☐	random29	OK	Deployment	3/3	default	bruteforce-minuscule-cluster
☐	random30	OK	Deployment	3/3	default	bruteforce-minuscule-cluster
☐	random31	OK	Deployment	3/3	default	bruteforce-minuscule-cluster
☐	random32	OK	Deployment	3/3	default	bruteforce-minuscule-cluster

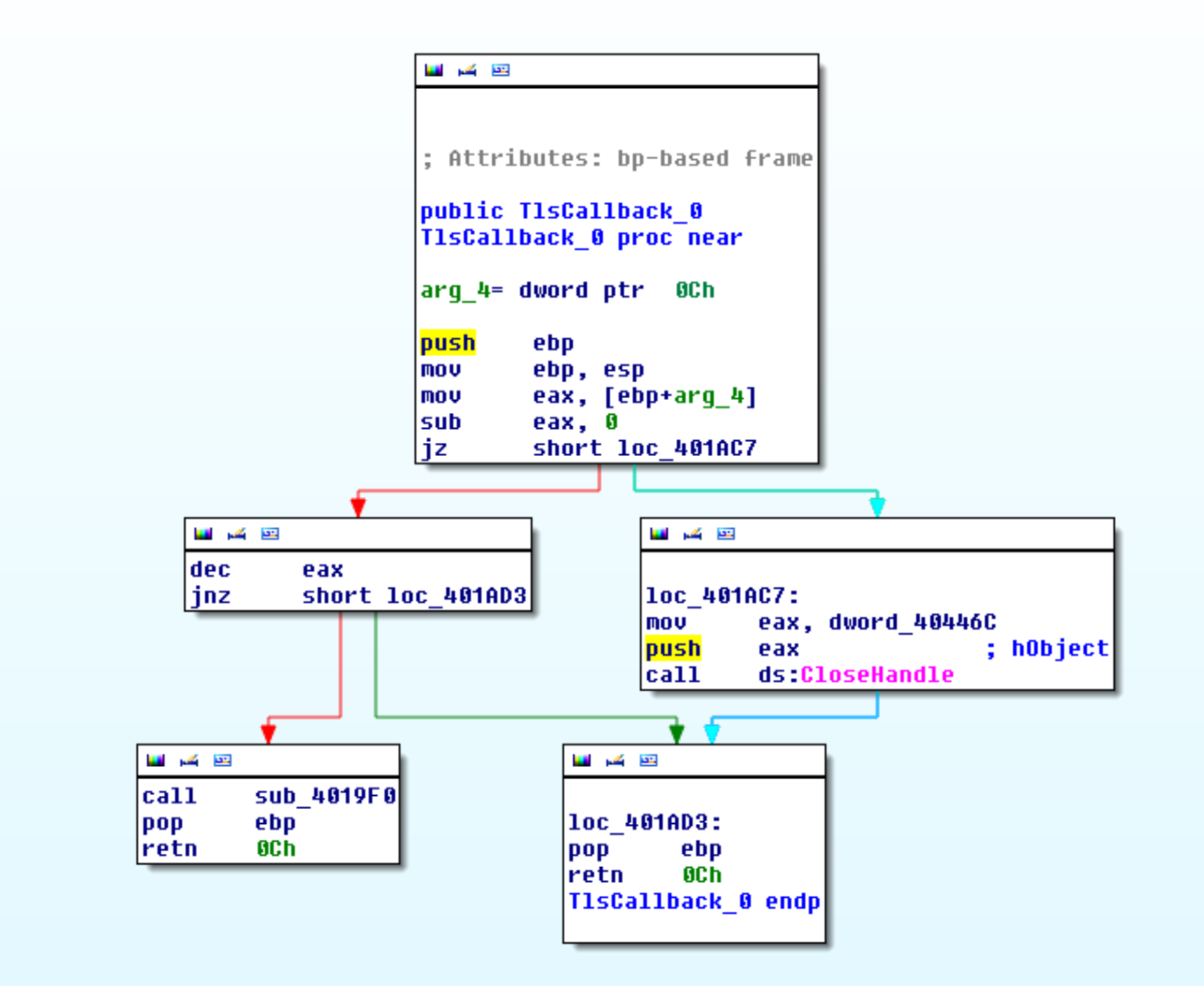
```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <openssl/evp.h>
#include <openssl/md5.h>
#include <time.h>

int compare(char *, int);

int main() {
    char* maxDigitEnv = getenv("MAXDIGIT");
    int MAXDIGIT = (maxDigitEnv != NULL) ? atoi(maxDigitEnv) : 1;

    printf("Début de la recherche avec des chaînes de %d caractères\n", MAXDIGIT);
    fflush(stdout);
    char cset[] = "abcdefABCDEF0123456789";
    int csetSize = strlen(cset);

    char value[MAXDIGIT + 1];

    // Variables pour le timer
    clock_t start = clock(), end;
    double cpu_time_used;

    srand((unsigned int)time(NULL));

    while (1) {
        // Générer une entrée aléatoire de 32 caractères
        for (int i = 0; i < MAXDIGIT; ++i) {
            value[i] = cset[rand() % csetSize];
        }
        value[MAXDIGIT] = '\0';

        int tmp = compare(value, MAXDIGIT);
        if (tmp == 0) {
            break;
        }

        end = clock();
        cpu_time_used = ((double)(end - start)) / CLOCKS_PER_SEC / 60;

        if (cpu_time_used >= 30) {
            printf("1 heure écoulée\n");
            fflush(stdout);
            start = clock(); // Réinitialiser le timer
        }
    }

    return 0;
}
```

Hypothèse finale

Le programme réagit de la même manière si l'entrée est licite ou illicite. Peut-être que le hash correspondant à l'entrée gagnante est un hash "illicite" obtenu par une entrée "illicite". Dans ce cas, l'exécutable peut ne pas avoir un comportement normal d'après les règles du jeu et donc il n'y a pas de clé dite "gagnante".

La seule manière de déterminer si l'entrée permettant de calculer le hash gagnant est licite est de brute force le hash "763ec1dc9466df8b964380678d258a8f" or c'est impossible pour la durée du projet.

Arbre de la fonction du TLS Callback

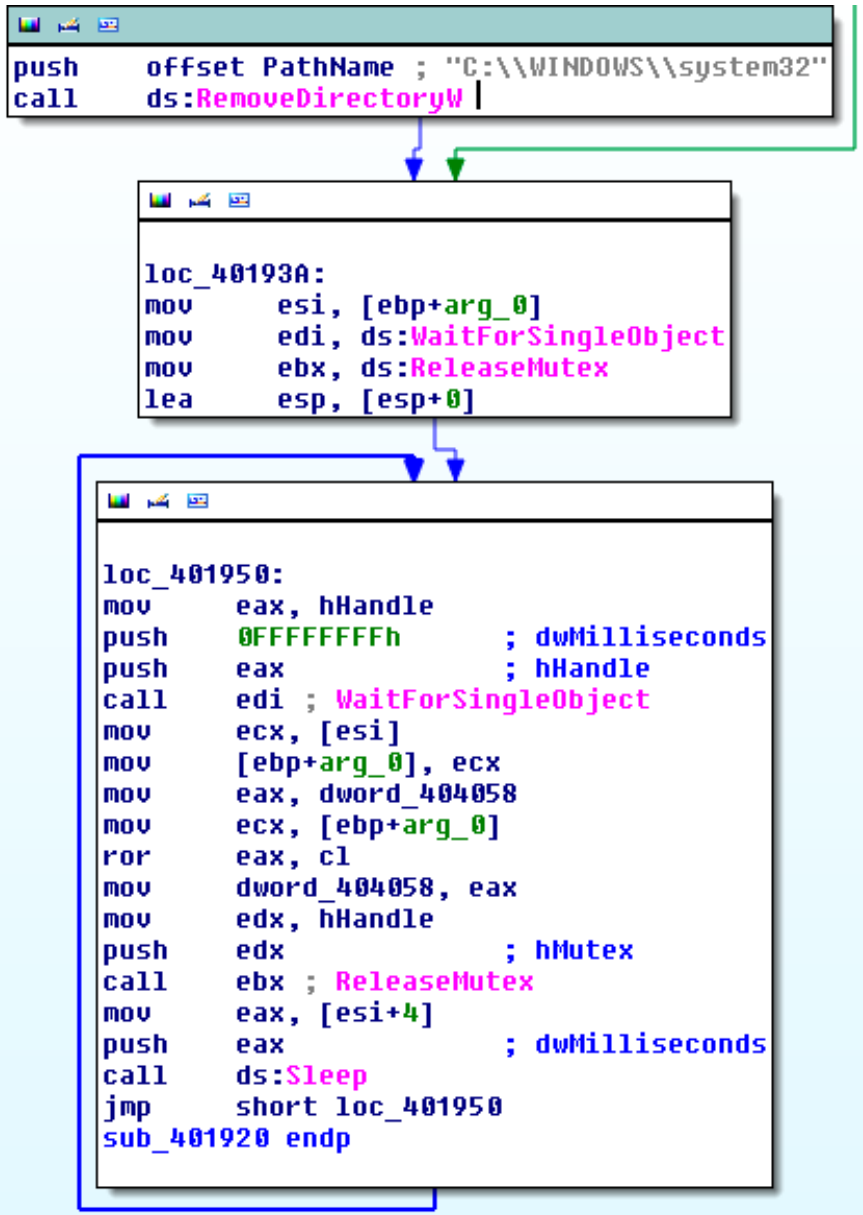


Variables globales

```
.rdata:0040315A                 db    0
.rdata:0040315B                 db    0
.rdata:0040315C ; struct _EXCEPTION_POINTERS ExceptionInfo
.rdata:0040315C ExceptionInfo   _EXCEPTION_POINTERS <offset dword_4040C0, offset dword_404118>
.rdata:0040315C                                         ; DATA XREF: ___report_gsfailure+D6↑o
.rdata:00403164 ; char Format[]
.rdata:00403164 Format          db '%02x',0             ; DATA XREF: sub_4017A0+C8↑o
.rdata:00403169                 align 4
.rdata:0040316C ; const WCHAR PathName
.rdata:0040316C PathName:                               ; DATA XREF: sub_401920+F↑o
.rdata:0040316C                 unicode 0, <C:\WINDOWS\system32>,0
.rdata:00403194 aEnterYourPhras db 'Enter your phrase: ',0 ; DATA XREF: sub_401AE0+30↑o
.rdata:004031A8 a763ec1dc9466df db '763ec1dc9466df8b964380678d258a8f',0
.rdata:004031A8                                         ; DATA XREF: _main+4D↑o
.rdata:004031C9                 align 4
.rdata:004031CC ; char aYouFoundTheRea[]
.rdata:004031CC aYouFoundTheRea db 'You found the really secret key!',0Ah,0
.rdata:004031CC                                         ; DATA XREF: _main+81↑o
.rdata:004031EE                 align 10h
.rdata:004031F0 ; char aBadInput___[]
.rdata:004031F0 aBadInput___    db 'Bad input...',0Ah,0 ; DATA XREF: _main:loc_401CA4↑o
.rdata:004031FE                 align 10h
.rdata:00403200 __load_config_used dd 48h               ; Size
.rdata:00403204                 dd 0                    ; Time stamp
.rdata:00403208                 dw 2 dup(0)             ; Version: 0.0
.rdata:0040320C                 dd 0                    ; GlobalFlagsClear
.rdata:00403210                 dd 0                    ; GlobalFlagsSet
.rdata:00403214                 dd 0                    ; CriticalSectionDefaultTimeout
.rdata:00403218                 dd 0                    ; DeCommitFreeBlockThreshold
.rdata:0040321C                 dd 0                    ; DeCommitTotalFreeThreshold
```

```
.data:00404058                                          ; sub_401920+49↑w
.data:0040405C                  dd offset NumberOfCharsRead
.data:00404060                  dd offset NumberOfCharsRead
.data:00404064 aAbcubtlljmm0jn  db 'abcubtlljmm!0JN!jebr75/fyf!0G!2?!ovm!3?ovmcba',0
.data:00404064                                          ; DATA XREF: sub_401AE0+B0↑o
```

Les threads


push offset PathName ; "C:\\WINDOWS\\system32"
call ds:RemoveDirectoryW
loc_40193A:
mov esi, [ebp+arg_0]
mov edi, ds:WaitForSingleObject
mov ebx, ds:ReleaseMutex
lea esp, [esp+0]
loc_401950:
mov eax, hHandle
push 0FFFFFFFFh ; dwMilliseconds
push eax ; hHandle
call edi ; WaitForSingleObject
mov ecx, [esi]
mov [ebp+arg_0], ecx
mov eax, dword_404058
mov ecx, [ebp+arg_0]
ror eax, cl
mov dword_404058, eax
mov edx, hHandle
push edx ; hMutex
call ebx ; ReleaseMutex
mov eax, [esi+4]
push eax ; dwMilliseconds
call ds:Sleep
jmp short loc_401950
sub_401920 endp