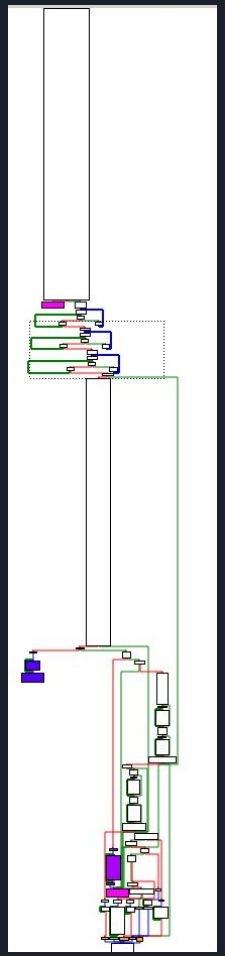
Malware

Groupe Thomas BIGEL, Christian VASAUNE, Matthieu PERRIGOT

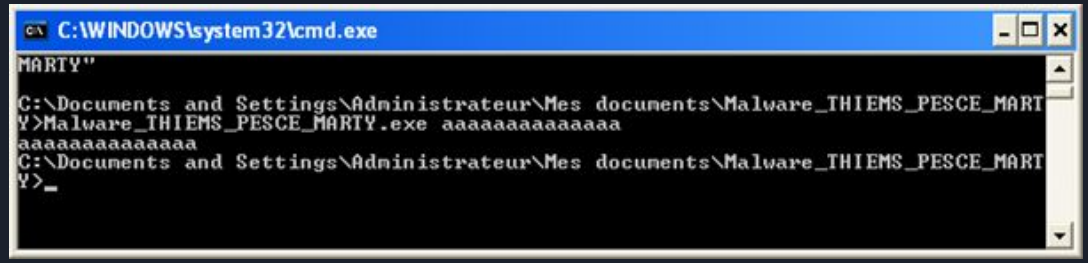
Malware 1

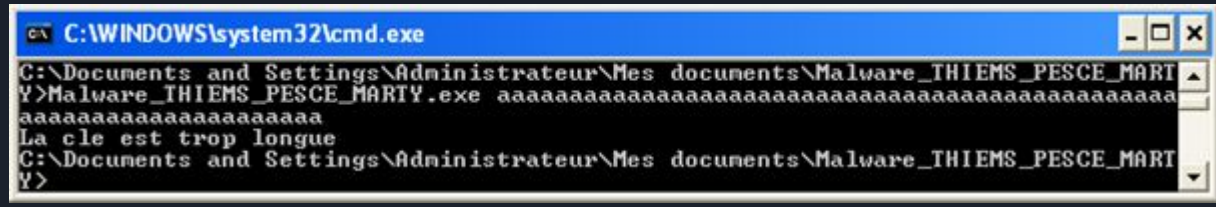
PESCE Teddy
MARTY Pierre
THIEMS Florian

Analyse du malware

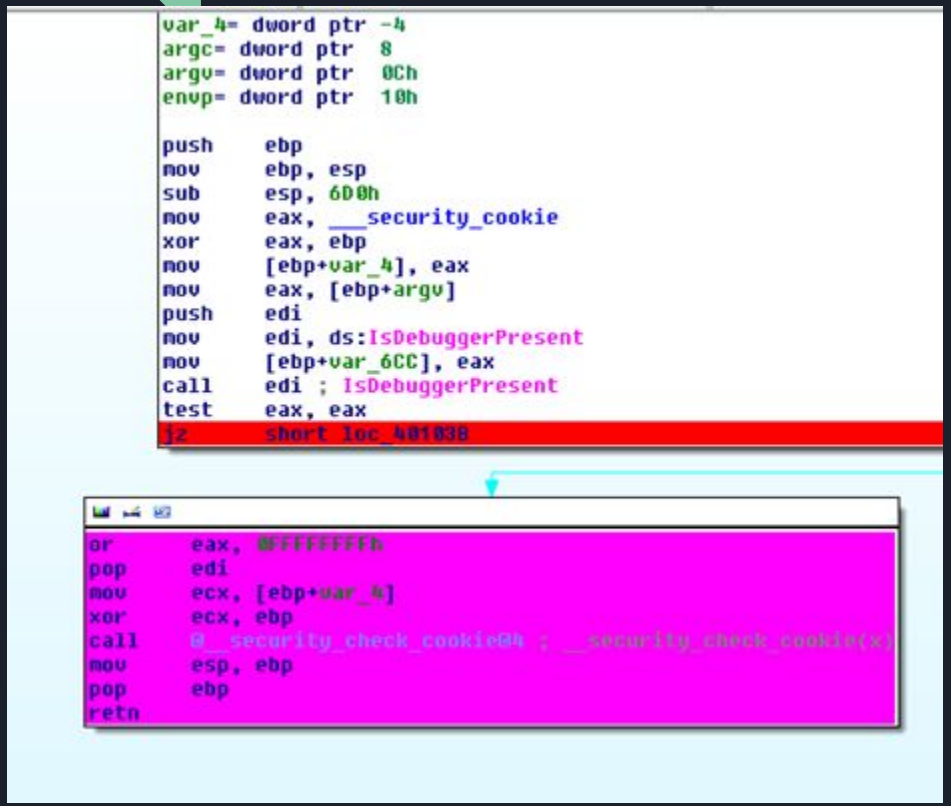


Première exécution :



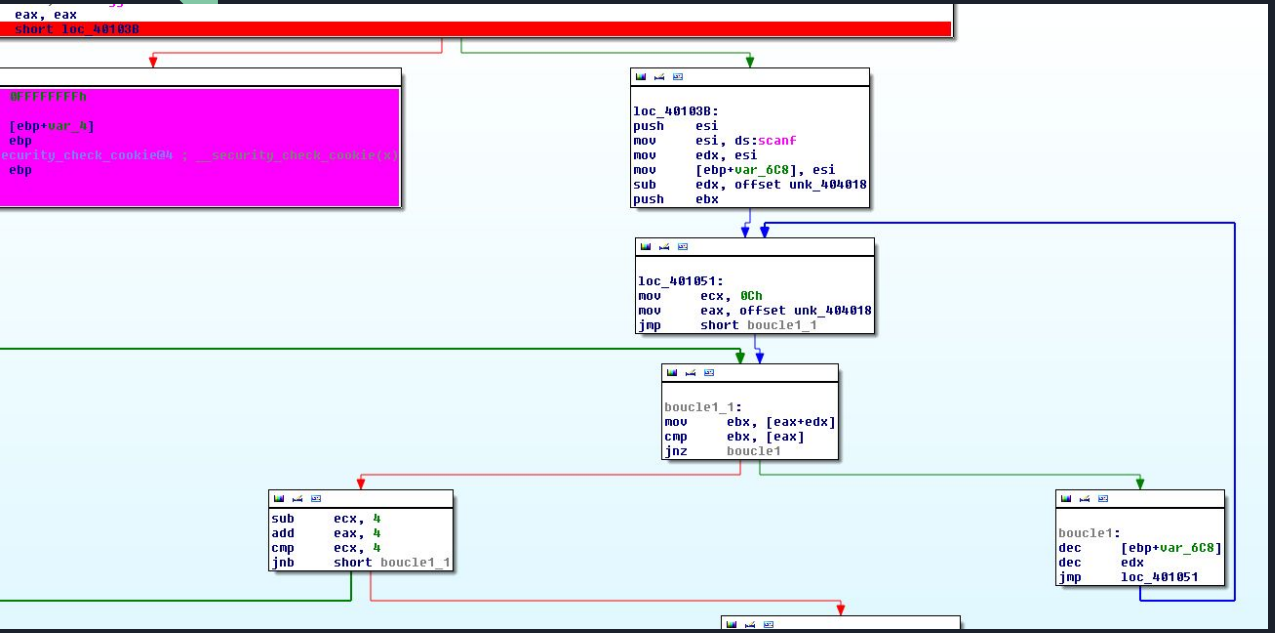


Anti-debug



7 IsDebuggerPresent:
00401017
004010F6
00401DD3
00401F19
0040209D
004021C5
0040224D

Obfuscation

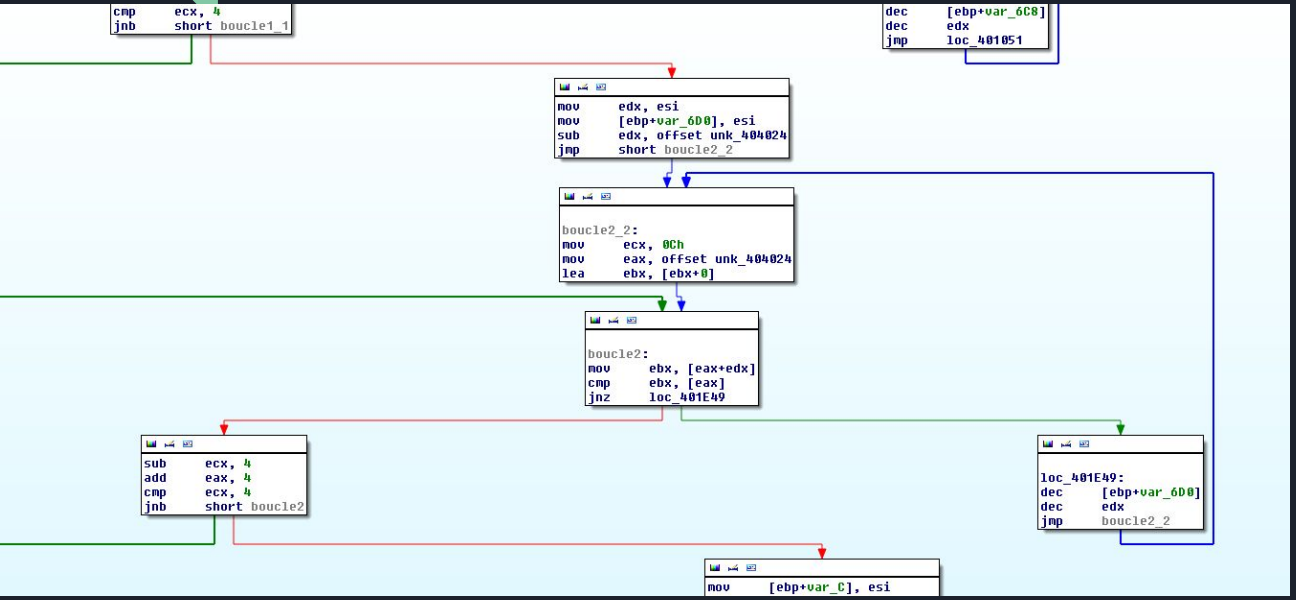


recherche pointeur fonction à partir de scanf:

[ebp+var_6C8]: printf

plages d'adresses:
0040103B à 0040105B
00401060 à 00401074
00401E3D à 00401E44

Obfuscation

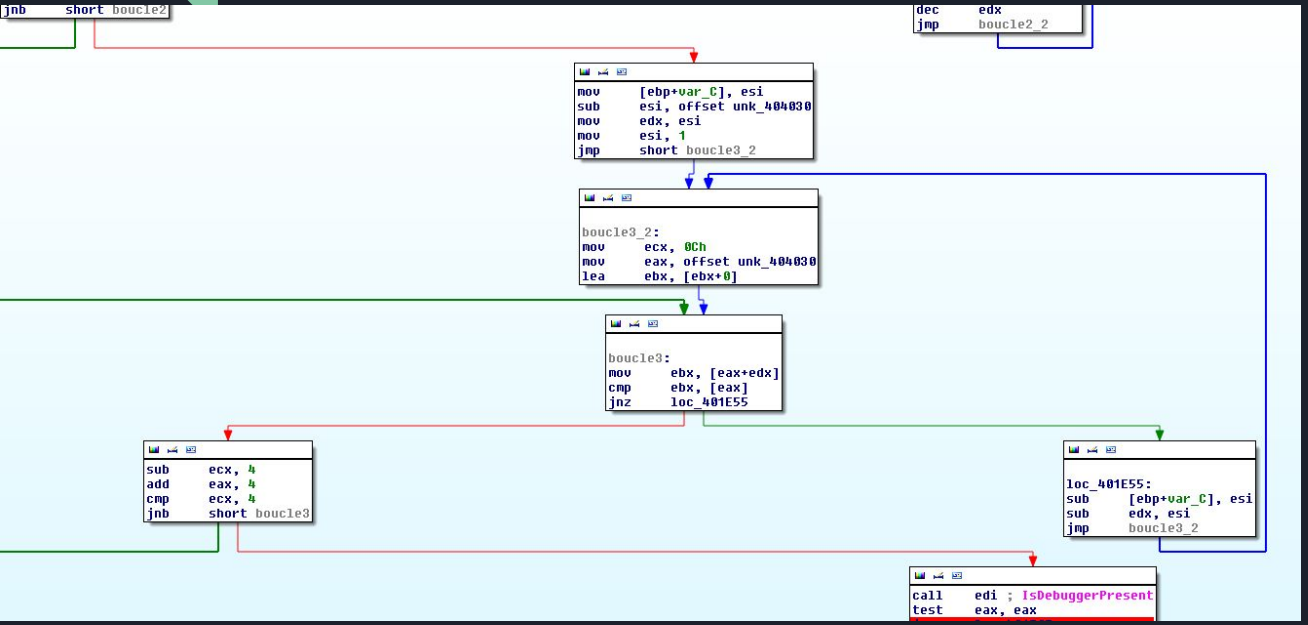


recherche pointeur fonction à partir de scanf:

[ebp+var_6D0]: strcmp

plages d'adresses:
00401076 à 00401084
00401E49 à 00401E50
00401090 à 004010B4

Obfuscation



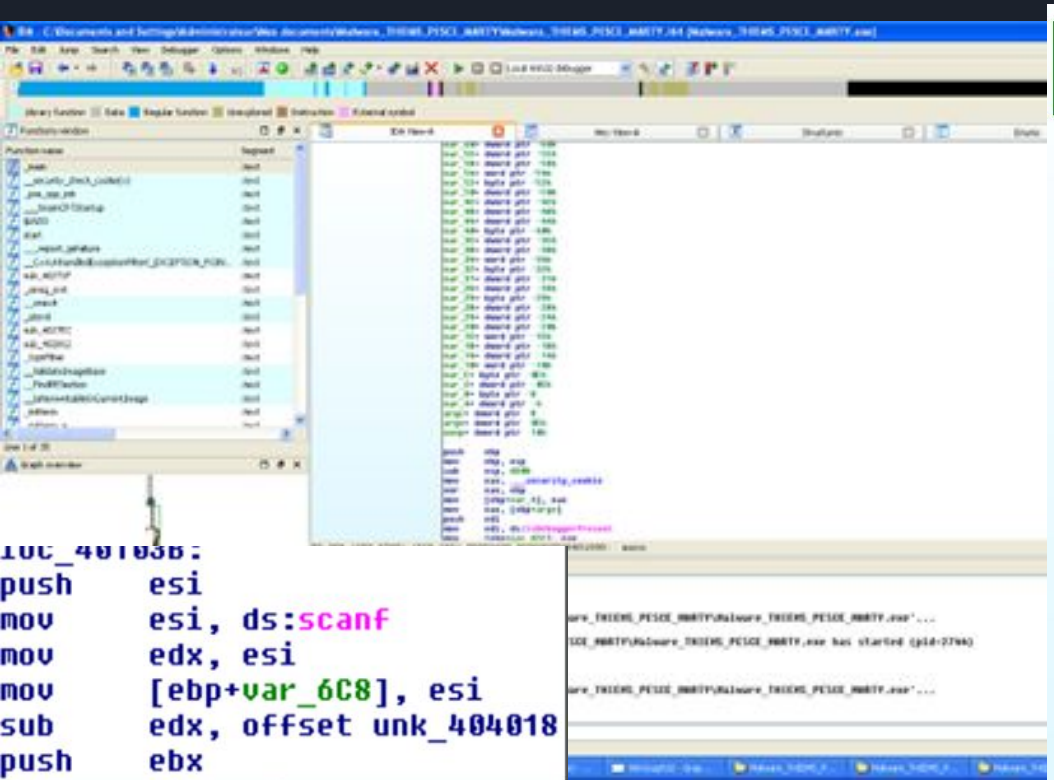
recherche pointeur fonction à partir de scanf:

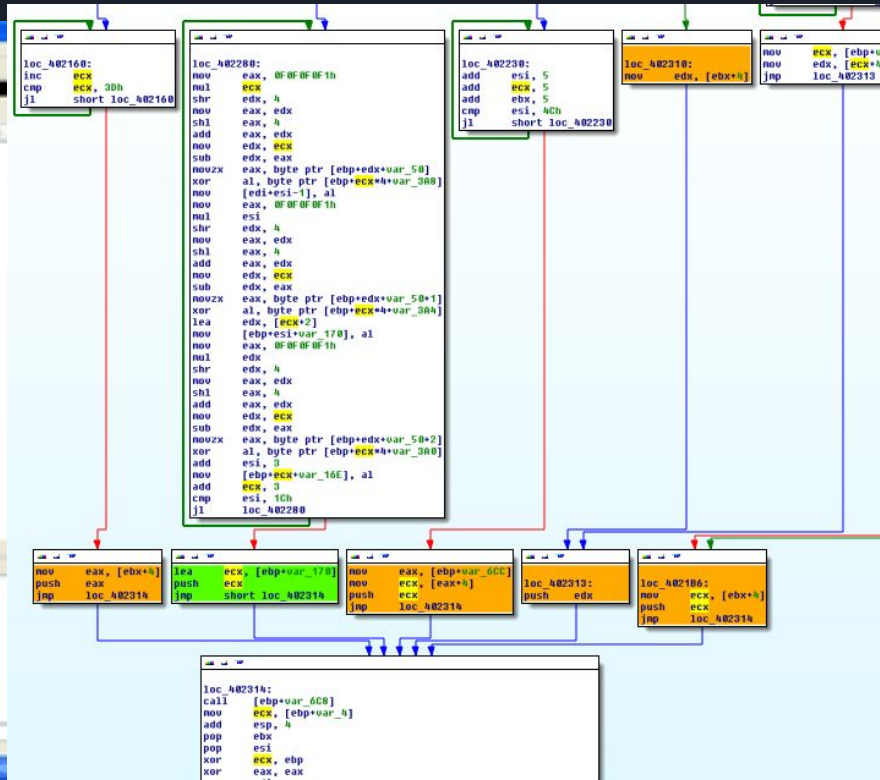
[ebp+var_C]: strlen

plages d'adresses:
004010B6 à 004010C6
00401E55 à 00401E5A
004010D0 à 004010F4

Utilisation de beaucoup de variables

- Utilisation de plein de variables négatives pour remonter à partir de ebp





```
push    3Fh             ; Size
push    eax             ; Val
lea     edx, [ebp+Dst]
mov     bl, 30h
push    edx             ; Dst
mov     [ebp+var_D8], 74201904h
mov     [ebp+var_D4], 21705113h
mov     [ebp+var_D0], 62441157h
mov     [ebp+var_CC], 78018696h
mov     [ebp+var_C8], 73795763h
mov     [ebp+var_18], 22549262h
mov     [ebp+var_14], 99442371h
mov     byte ptr [ebp+var_10], 47h
mov     [ebp+var_28], 40071410h
mov     [ebp+var_24], 56871223h
mov     [ebp+var_20], 206h
mov     [ebp+var_C], 15243110h
mov     [ebp+var_8], 36h
mov     [ebp+var_21C], bl
call    memset
push    3Fh             ; Size
lea     eax, [ebp+var_1D8]
push    0               ; Val
push    eax             ; Dst
mov     [ebp+var_1DC], bl
call    memset
push    3Fh             ; Size
lea     ecx, [ebp+var_298]
push    0               ; Val
push    ecx             ; Dst
mov     [ebp+var_29C], bl
call    memset
push    3Fh             ; Size
lea     edx, [ebp+var_258]
push    0               ; Val
push    edx             ; Dst
mov     [ebp+var_25C], bl
call    memset
mov     eax, [ebp+var_6CC]
mov     ecx, [eax+4]
push    ecx
xor     esi, esi
call    edi
add     esp, 34h
test    eax, eax
jle     short loc_402036
```

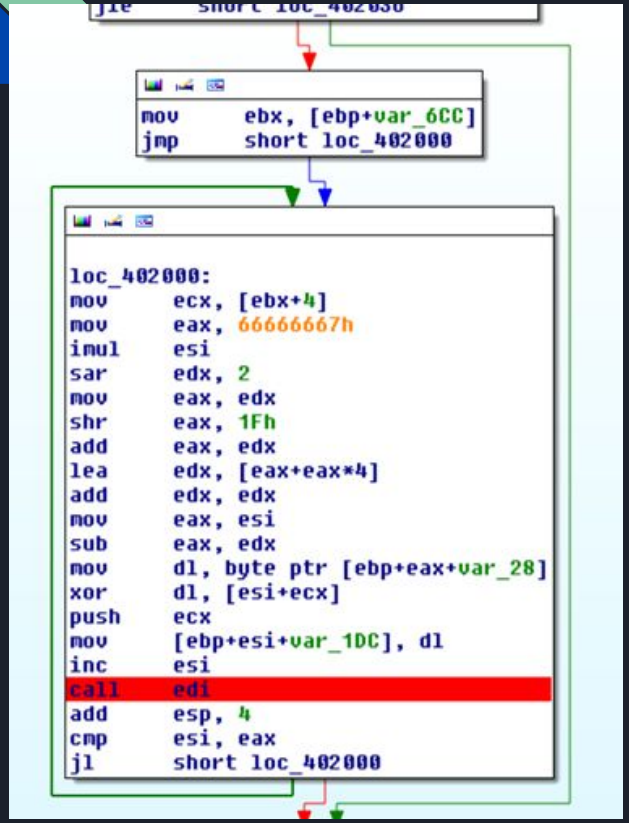
Chiffrement

Préparation des quatres chiffrements

4 memset

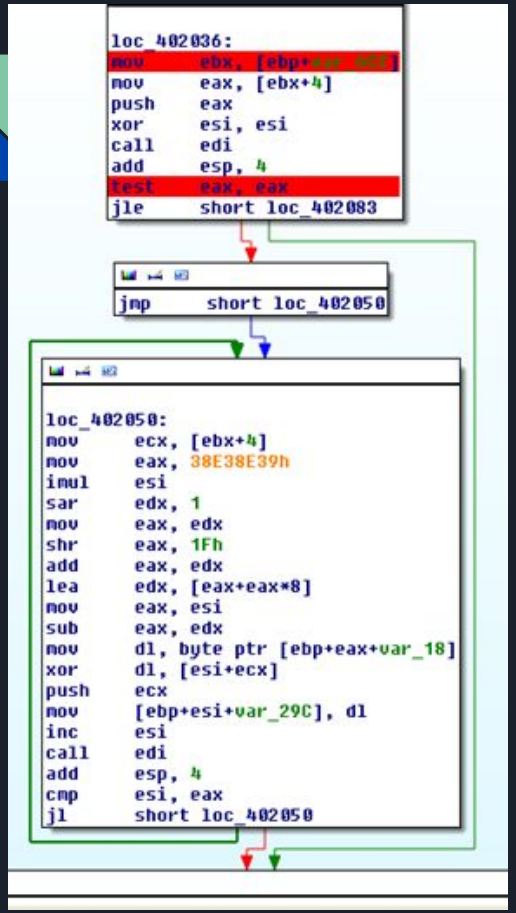
Plage d’adresses:
00401F23 à 00401FF3

Chiffrement



Chiffrement de l'entrée
plage d'adresses:
00401FF5 à 00401FFB
00402000 à 00402034

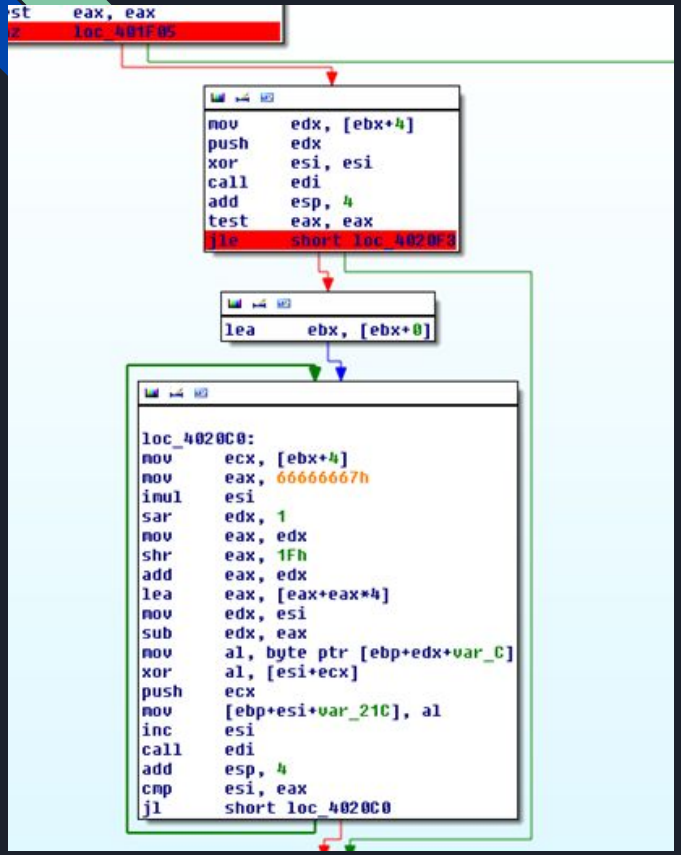
plage adresse liste utilisé:
000000000012FF54 à 000000000012FF5D
{10h, 14h, 7, 40h, 23h, 12h, 87h, 56h, 6, 2}



Chiffrement

Chiffrement de l'entrée
plage d'adresses:
00402050 à 00402081

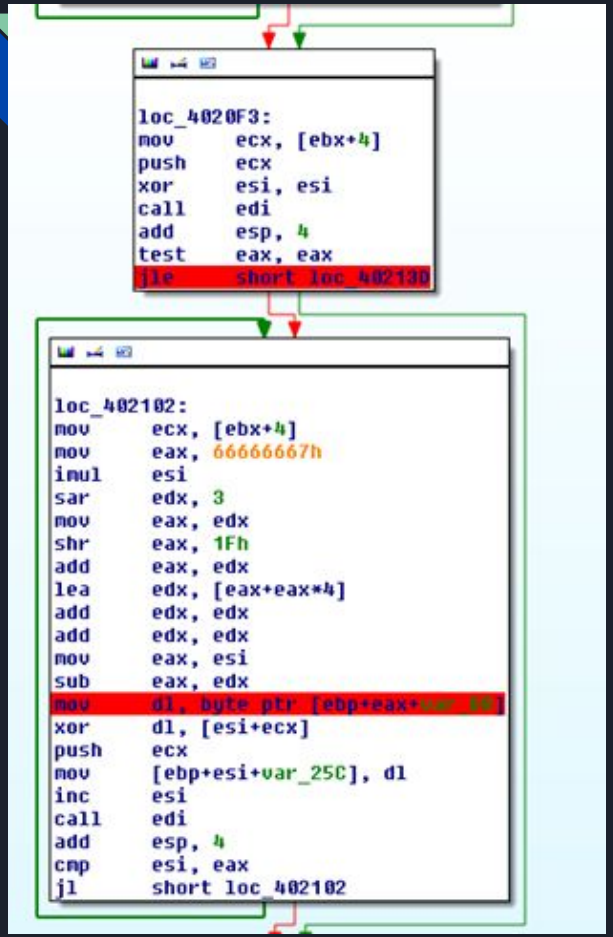
plage adresse liste utilisé:
000000000012FF64 à 000000000012FF6C
{62h, 92h, 54h, 22h, 71h, 23h, 44h, 99h, 47h}



Chiffrement

Chiffrement de l'entrée
plage d'adresses:
004020C0 à 004020F1

plage adresse liste utilisé:
000000000012FF70 à 000000000012FF74
{10h, 31h, 24h, 15h, 36h}



Chiffrement

Chiffrement de l'entrée
plage d'adresses:
00402102 à 0040213B

plage adresse liste utilisé:
000000000012FEA4 à 000000000012FEB7
{4, 19h, 20h, 74h, 13h, 51h, 70h, 21h, 57h, 11h, 44h, 62h, 96h, 86h, 1, 78h, 63h, 57h, 79h, 73}

Outils

IDA: graphes, analyse des instructions

Debugger de IDA

Python: reproduction de l'algo de chiffrement -> algo de déchiffrement

Utilisation de exrex

Outils: python

chiffrement.py

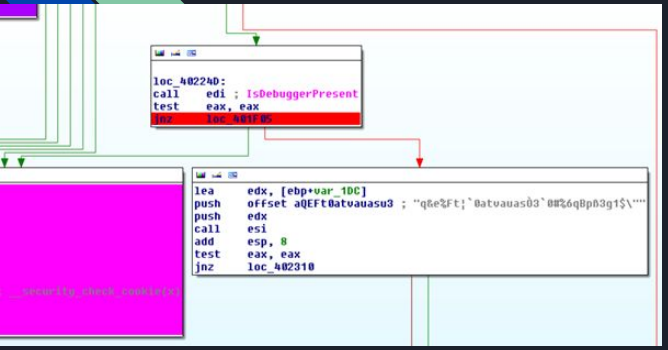
```python
def main():
    listeChiffrement = ['0x10', '0x14', '0x7', '0x40', '0x23', '0x12',
'0x87', '0x56', '0x6', '2']
    i = 0
    cle = ''
    for char in 'q&e%Ft¦`0atvauasÒ3`0#%6qBpß3g1$"':
        print(hex(int(listeChiffrement[i], 16) ^ int(hex(ord(char)), 16)))
        cle += chr(int(listeChiffrement[i], 16) ^ int(hex(ord(char)), 16))
        i += 1
        if i >= len(listeChiffrement):
            i = 0
    print(cle)
if __name__ == '__main__':
    main()
```

brute.py et brutePrec.py

```python
import subprocess
import exrex


def main():
    while 1:
        key = exrex.getone('a2beef.66cdbf5Ba.ef23111ab.ea346')
        output = subprocess.check_output(['C:\Documents and
Settings\Administrateur\Mes
documents\Malware_THIEMS_PESCE_MARTY\Malware_THIEMS_PESCE_MARTY.exe ',
key])
        if output != key:
            print(key + ' ' + output)

if __name__ == '__main__':
    main()
```

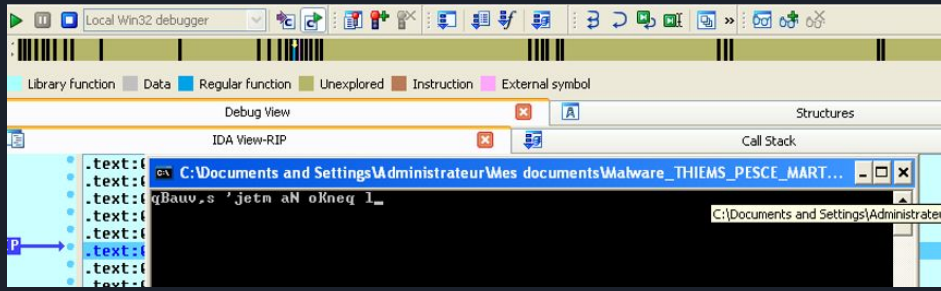
```python
key = exrex.getone('a2beef[3-7]66cdbf5.adef23111abfea346')
```

Trouver la clé secrète



Trouver le bon strcmp

00402257 à 0040226A



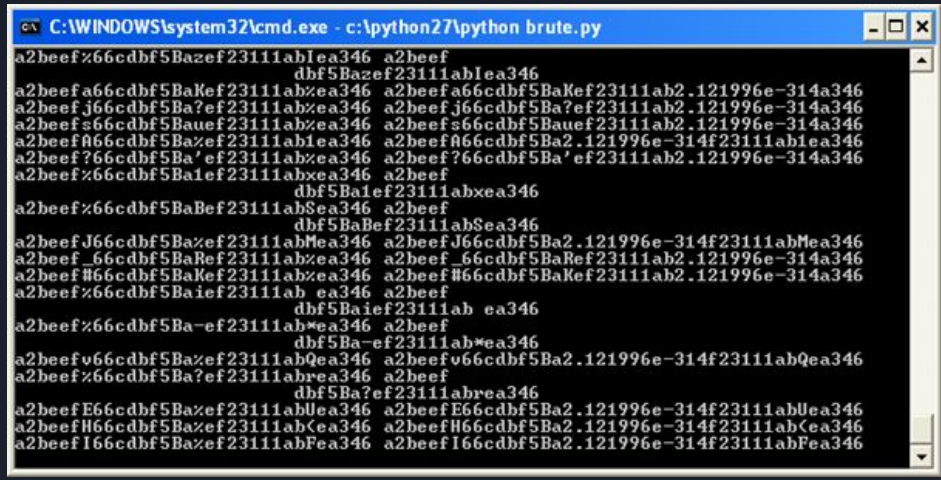
bon hash: q&e%Ft¦`0atvauasÒ3`0#%6qBpß3g1$"

1ere methode:

a2beef!66cdbf5BaUef23111abXea346

a2beefX66cdbf5BaXef23111abXea346

brute force: a2beef666cdbf5Badef23111abfea346



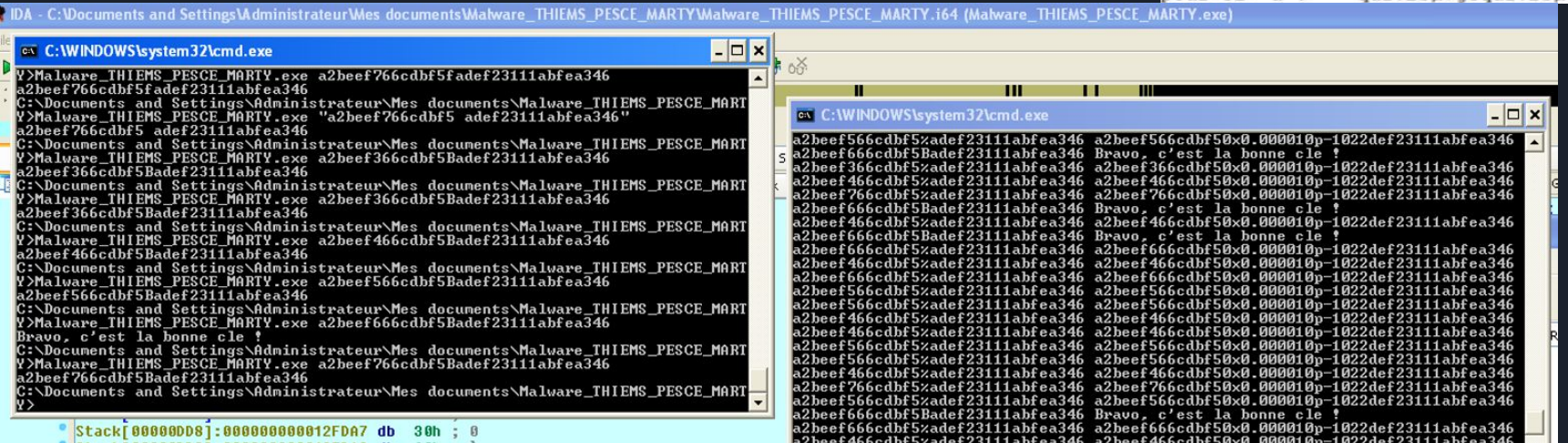
Trouver la clé secrète

bon hash: q&e%Ft¦`0atvauasÒ3`0#%6qBpß3g1$"

2eme methode:

brute force: a2beef666cdbf5Badef23111abfea346

```
clé potentiel   a2beef366cdbf5?adef23111abfea346
clé potentiel   a2beef466cdbf5?adef23111abfea346
clé potentiel   a2beef566cdbf5?adef23111abfea346
clé potentiel   a2beef666cdbf5?adef23111abfea346
clé potentiel   a2beef766cdbf5?adef23111abfea346

clé chiffrée    q&e%Ft¦`0atvauasÒ3`0#%6qBpß3g1$"
pour 32*'f':    vra&Etß0`dvra&Etß0`dvra&Etß0`dvr
pour 32*'e':    uqb%FwÔ3cguqb%FwÔ3cguqb%FwÔ3cguq
pour 32*'d':    tpc$GvÒ2bftpc$GvÒ2bftpc$GvÒ2bftp
pour 32*'c':    swd#@qõ5easwd#@qõ5easwd#@qõ5easw
pour 32*'b':    rve"ApÕ4d`rve"ApÕ4d`rve"ApÕ4d`rv
pour 32*'9':    )->yh+¥o?;)->yh+¥o?;)->yh+¥o?;)-
pour 32*'8':    (,?xh*+n>:(,?xh*+n>:(,?xh*+n>:(,
pour 32*'7':    '#0wh%¦a15'#0wh%¦a15'#0wh%¦a15'#
pour 32*'6':    &"1vh$¦`04&"1vh$¦`04&"1vh$¦`04&"
pour 32*'5':    %!2uh'¦c37%!2uh'¦c37%!2uh'¦c37%!
pour 32*'4':    $h3th&¦b26$h3th&¦b26$h3th&¦b26$h
pour 32*'3':    #'4sh!¦e51#'4sh!¦e51#'4sh!¦e51#'
pour 32*'2':    "&5rhhÁd40"&5rhhÁd40"&5rhhÁd40"&
pour 32*'0':    h$7ph"Àf62h$7ph"Àf62h$7ph"Àf62h$
pour 32*'1':    !%6qh#Âg73!%6qh#Âg73!%6qh#Âg73!%
pour 32*'a':    quf!Bsu7gcquf!Bsu7gcquf!Bsu7gcqu
```



Résumé analyse du malware 1

Anti-debug - utilisation de beaucoup de IsDebuggerPresent() tout le long du malware

Obfuscation de fonction en partant de scanf pour les fonctions printf, strcmp et strlen

Utilisation de plein de **variables négatives** pour remonter à partir de ebp

Utilisation de **leurres** avec strcmp

Beaucoup de mov ou d'instructions inutiles pour essayer de **brouiller** l'analyse

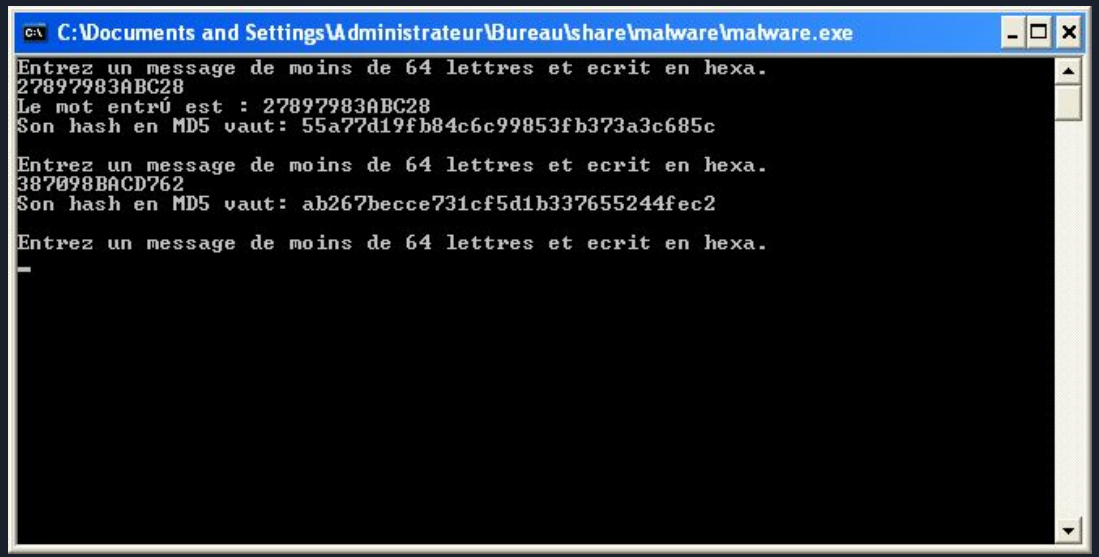
Chiffrement faible caractère à caractère

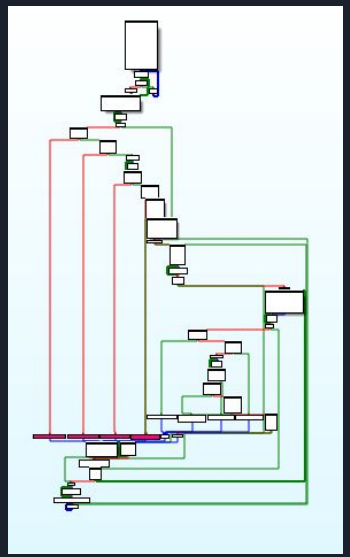
Malware 2

TENEDE TENE Miguel Bryan
DUVERNIER Alban
PORCU Baptiste

Analyse du malware

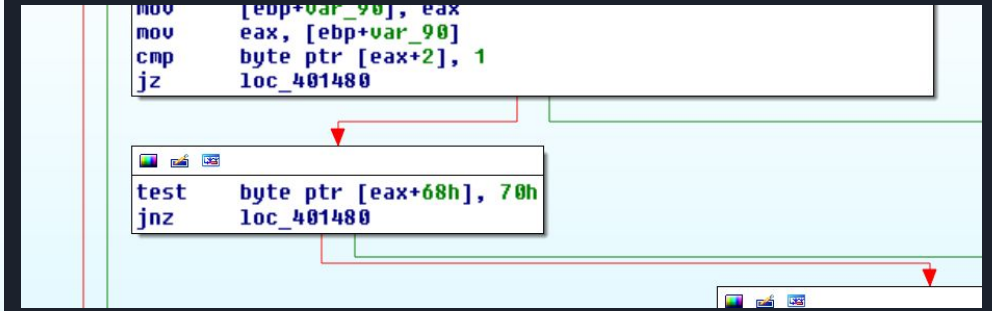
Première exécution :



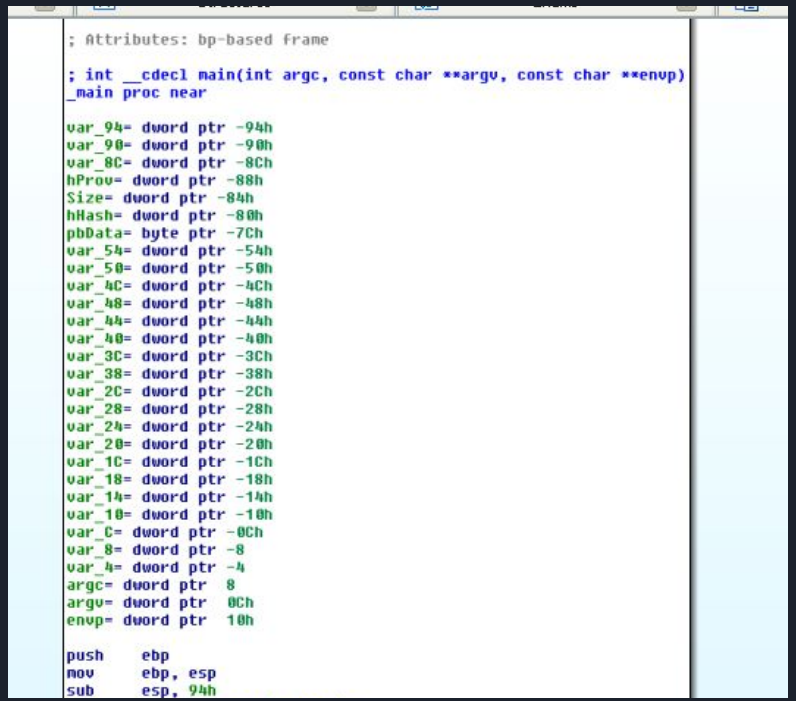


Malware sur IDA

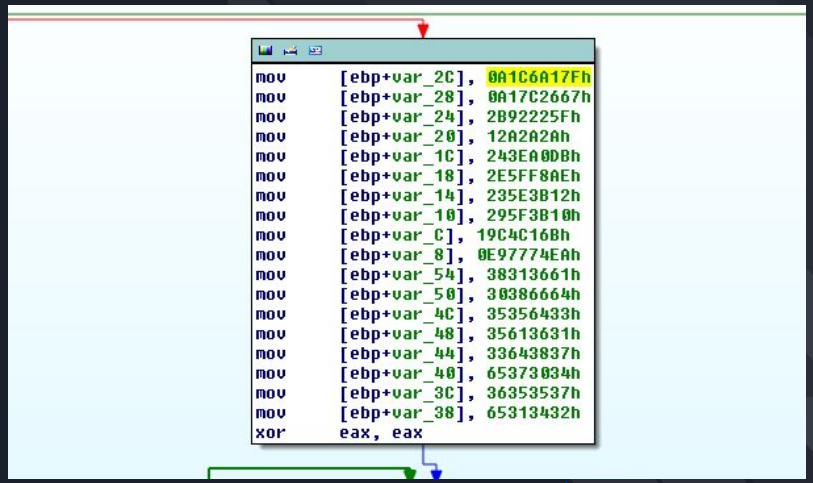
Anti-debug



Utilisation de variables

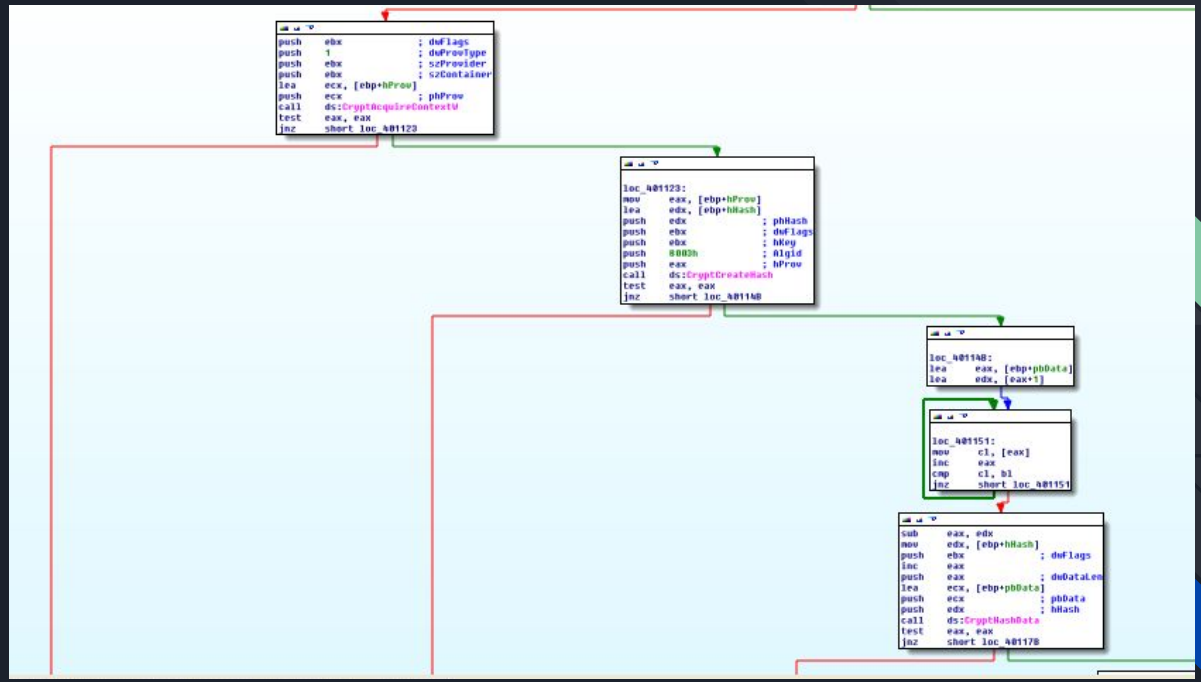


```
; Attributes: bp-based frame

; int __cdecl main(int argc, const char **argv, const char **envp)
_main proc near

var_94= dword ptr -94h
var_90= dword ptr -90h
var_8C= dword ptr -8Ch
hProv= dword ptr -88h
Size= dword ptr -84h
hHash= dword ptr -80h
pbData= byte ptr -7Ch
var_54= dword ptr -54h
var_50= dword ptr -50h
var_4C= dword ptr -4Ch
var_48= dword ptr -48h
var_44= dword ptr -44h
var_40= dword ptr -40h
var_3C= dword ptr -3Ch
var_38= dword ptr -38h
var_2C= dword ptr -2Ch
var_28= dword ptr -28h
var_24= dword ptr -24h
var_20= dword ptr -20h
var_1C= dword ptr -1Ch
var_18= dword ptr -18h
var_14= dword ptr -14h
var_10= dword ptr -10h
var_C= dword ptr -0Ch
var_8= dword ptr -8
var_4= dword ptr -4
argc= dword ptr  8
argv= dword ptr  0Ch
envp= dword ptr  10h

push    ebp
mov     ebp, esp
sub     esp, 94h
```

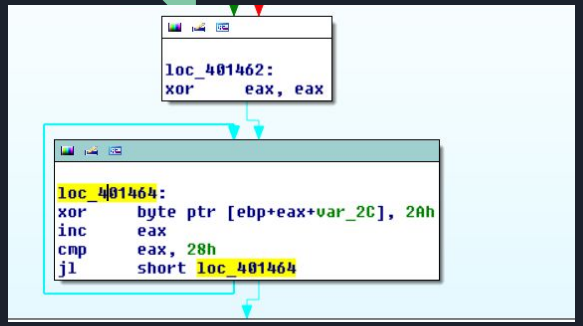


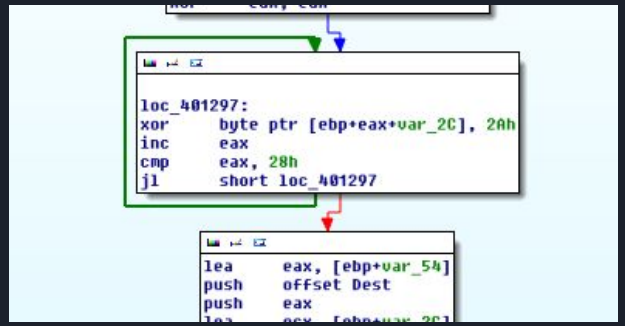
```
mov     [ebp+var_2C], 0A1C6A17Fh
mov     [ebp+var_28], 0A17C2667h
mov     [ebp+var_24], 2B92225Fh
mov     [ebp+var_20], 12A2A2Ah
mov     [ebp+var_1C], 243EA0DBh
mov     [ebp+var_18], 2E5FF8AEh
mov     [ebp+var_14], 235E3B12h
mov     [ebp+var_10], 295F3B10h
mov     [ebp+var_C], 19C4C16Bh
mov     [ebp+var_8], 0E97774EAh
mov     [ebp+var_54], 38313661h
mov     [ebp+var_50], 30386664h
mov     [ebp+var_4C], 35356433h
mov     [ebp+var_48], 35613631h
mov     [ebp+var_44], 33643837h
mov     [ebp+var_40], 6537303Ah
mov     [ebp+var_3C], 36353537h
mov     [ebp+var_38], 65313432h
xor     eax, eax
```

Chiffrement



Obfuscation





permet de cacher une fonction

```
.text:00401297                xor     byte ptr [ebp+eax+var_2C], 2Ah
.text:0040129C                inc     eax
.text:0040129D                cmp     eax, 28h
.text:004012A0                jl      short loc_401297
```

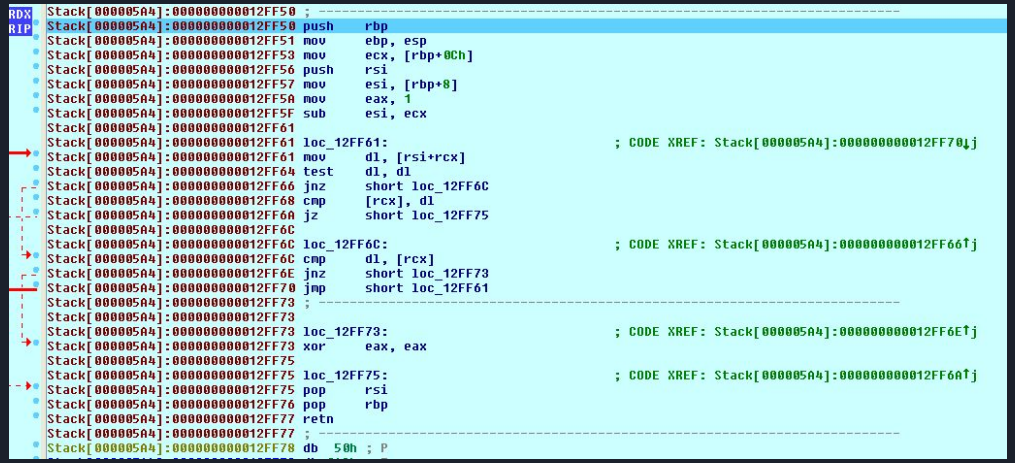
```
.text:00401464                xor     byte ptr [ebp+eax+var_2C], 2Ah
.text:00401469                inc     eax
.text:0040146A                cmp     eax, 28h
.text:0040146D                jl      short loc_401464
.text:0040146F                push    offset aBravoTuAsTrouv ; "Bravo!!! Tu as trouvÚ le mot cachÚ!!!"
```

```
.text:00401217                mov     [ebp+var_2C], 0A1C6A17Fh
```

```
.text:004012AB                lea     ecx, [ebp+var_2C]
.text:004012AE                call    ecx
```

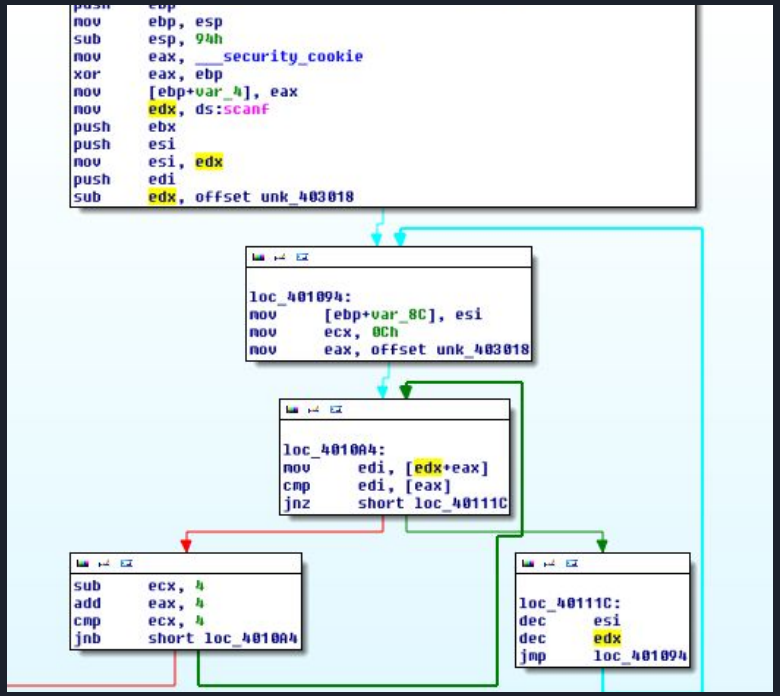
la fonction est chargée dans ecx, puis appeler

Obfuscation



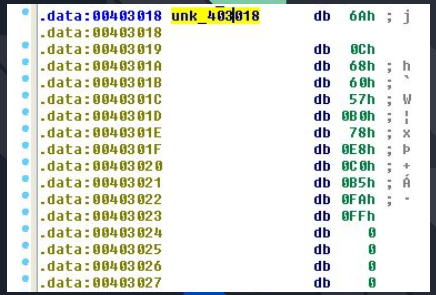
clef md5 : a618df803d5516a578d3407e7556241e

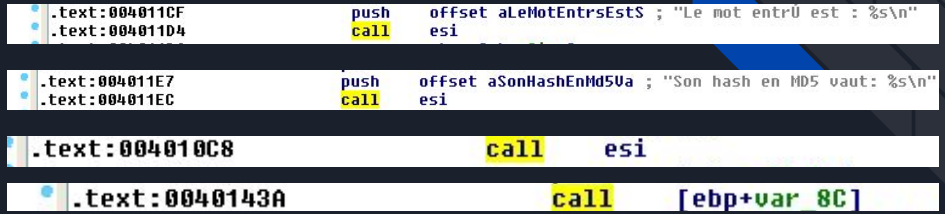
Obfuscation



recherche pointeur fonction à partir de scanf:

edx : printf
esi : edx
[ebp+var_8C] : esi



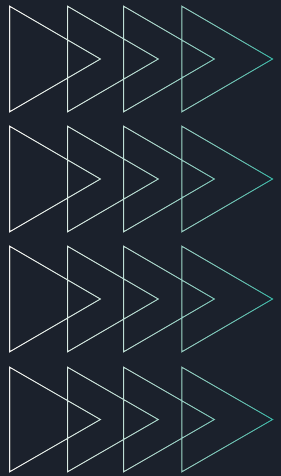


Outils

IDA: graphes, analyse des instructions

Debugger de IDA

Résumé analyse du malware 2



Anti-debug utilisation d'anti debug au début

Obfuscation de fonction en partant de scanf et d'une fonction cacher

Chiffrement MD5 de base via wincrypt.h

Utilisation de plein de **variables négatives** pour remonter à partir de ebp