# Xtreme RAT analysis

**malware.lu**/articles/2012/07/22/xtreme-rat-analysis.html

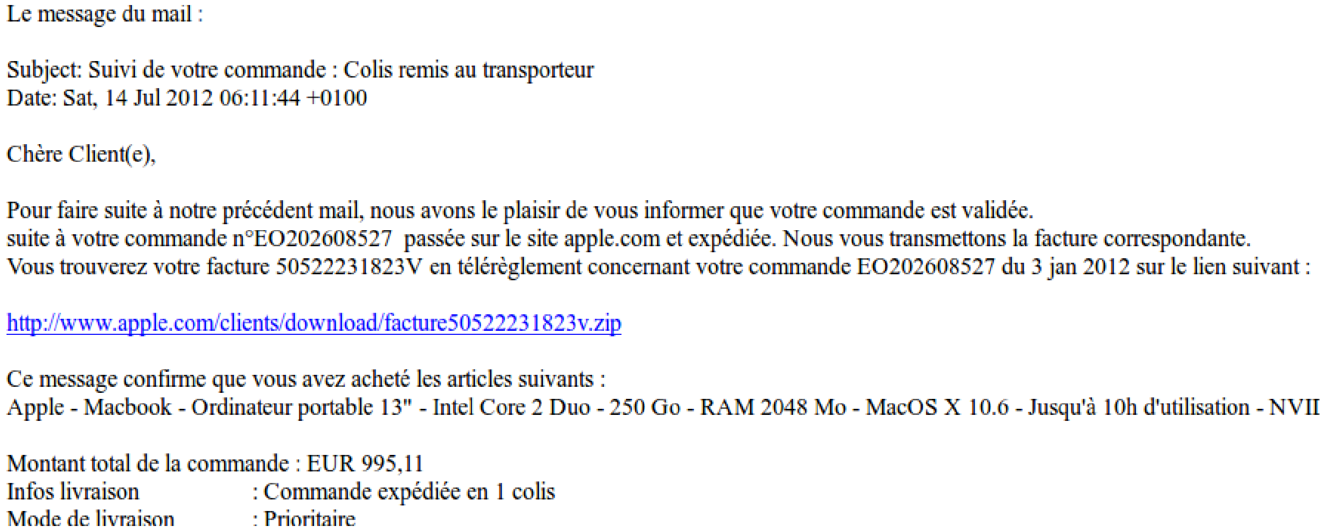
Published on 2012-07-22 14:00:00.
We received an email with an invoice from Apple (in french).

Of course we never bought something from Apple!!!!

The link of the invoice seems to be :
http://www.apple.com/clients/download/facture50522231823v.zip

But when we put our mouse on the link we can see the real link:
http://editionslabonte.com/plugins/Facture147778.zip

We think that the Website “editionslabonte.com” was compromised and the attacker puts the malware on it. We sent an email to the administrator and we do not have a feedback for the moment.



Le message du mail :

Subject: Suivi de votre commande : Colis remis au transporteur
Date: Sat, 14 Jul 2012 06:11:44 +0100

Chère Client(e),

Pour faire suite à notre précédent mail, nous avons le plaisir de vous informer que votre commande est validée.
suite à votre commande n°EO202608527 passée sur le site apple.com et expédiée. Nous vous transmettons la facture correspondante.
Vous trouverez votre facture 50522231823V en télérèglement concernant votre commande EO202608527 du 3 jan 2012 sur le lien suivant :

http://www.apple.com/clients/download/facture50522231823v.zip

Ce message confirme que vous avez acheté les articles suivants :
Apple - Macbook - Ordinateur portable 13" - Intel Core 2 Duo - 250 Go - RAM 2048 Mo - MacOS X 10.6 - Jusqu'à 10h d'utilisation - NVII

Montant total de la commande : EUR 995,11
Infos livraison : Commande expédiée en 1 colis
Mode de livraison : Prioritaire

## Tools

- A debugger for dynamic analysis (in our case OllyDbg)
- LordPE in order to dump a memory page
- Volatility in order to analyse memory dump

## Zip archive

The md5 of the archive is e0aa33dc57aa3eee43cb61933eb3241c.

Virustotal score : 5/42

So we downloaded the .zip file.

```
rootbsd@alien:~/Samples$ unzip -l Facture147778.zip
Archive:  Facture147778.zip
  Length      Date    Time    Name
---------  ---------- -----   ----
   176128  2012-07-14 03:05   Facture147778.pdf            .scr
---------                     -------
   176128                     1 file
```

The .zip contains one file. To trick the user, the attacker adds several space before the extension .scr, some users may thought that the file is really a .pdf.

## First binary

```
rootbsd@alien:~/Samples$ yara -r packer.yara Facture147778.pdf\ \ \ \ \ \ \ \ \ \ \ \
.scr
java Facture147778.pdf            .scr
NETexecutableMicrosoft Facture147778.pdf            .scr
```

The file is a .NET binary.

With the strings command, we find somethink that looks like a base64.

We extract the base64 :

```
rootbsd@alien:~/Samples$ cat base64.dmp
TVqQAAMAAAAEAAAA//8AALgAAAAAAAAAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAA4AAAAA4fug4AtAnNIbgBTM0hVGhpcyBwcm9ncmFtIGNhbm5vdCBiZSBydW4gaW4gRE9TIG1v
ZGUuDQ0KJAAAAAAAAACZtmjHqtcGlN3XBpTd1waUpssKlNzXBpReywiU3NcGlDXIDJTW1waUNcgC
lNnXBpQe2FuU1NcGlN3XB5Tg1waUNcgNlNzXBpRl0QCU3NcGlFJpY2jd1waUAAAAAAAAAABQRQAA
TAEEAKYPc0oAAAAAAAAAAOAADwELAQYAAEIAAACUAAAAAAAAdE8AAAAQAAAAYAAAAABAAAAQAAAA
AgAABAAAAAAAAAAEAAAAAAAAAAAQAQAABAAAAAAAAAIAAAAAABAAABAAAAAAEAAAEAAAAAAAAABAA
[...]
W1EPulAAAAIAAQAgAEAAAQABADQBAAAFAAAAAQAEACAgEAABAAQA6AIAAAEAEBAQAAEABAAoAQAA
AgAgIAAAAQAgAKgQAAADABAQAAABACAAaAQAAAMAUEEAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAA=
```

We decode this file.

```
rootbsd@alien:~/Samples$ cat base64.dmp | base64 -d > base64.out
rootbsd@alien:~/Samples$ file base64.out
base64.out: PE32 executable for MS Windows (GUI) Intel 80386 32-bit
```

This base64 is a PE32 executable.

## Second binary

We use yara to identify the binary:

```
rootbsd@alien:~/Samples$ yara -r packer.yara base64.out
rootbsd@alien:~/Samples$
```

This binary doesn’t use a well-known packer. So we decided to unpack it manually.

To unpack it, we use OllyDBG.

We are suprised by a lot of exception when we tried to debug the sample.

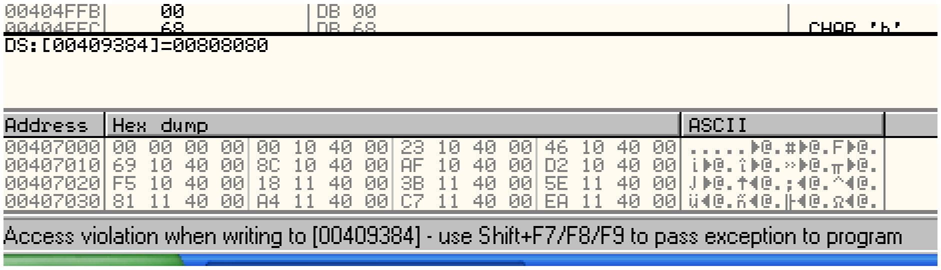
In fact this malware volontary uses and traps exceptions to be unpacked.

So as usual, we add breakpoint on VirtualAlloc & VirtualAllocEx calls:

- View
- Executable modules
- right click on kernel32.dll -> View names
- F2 on VirtualAlloc & VirtualAllocEx

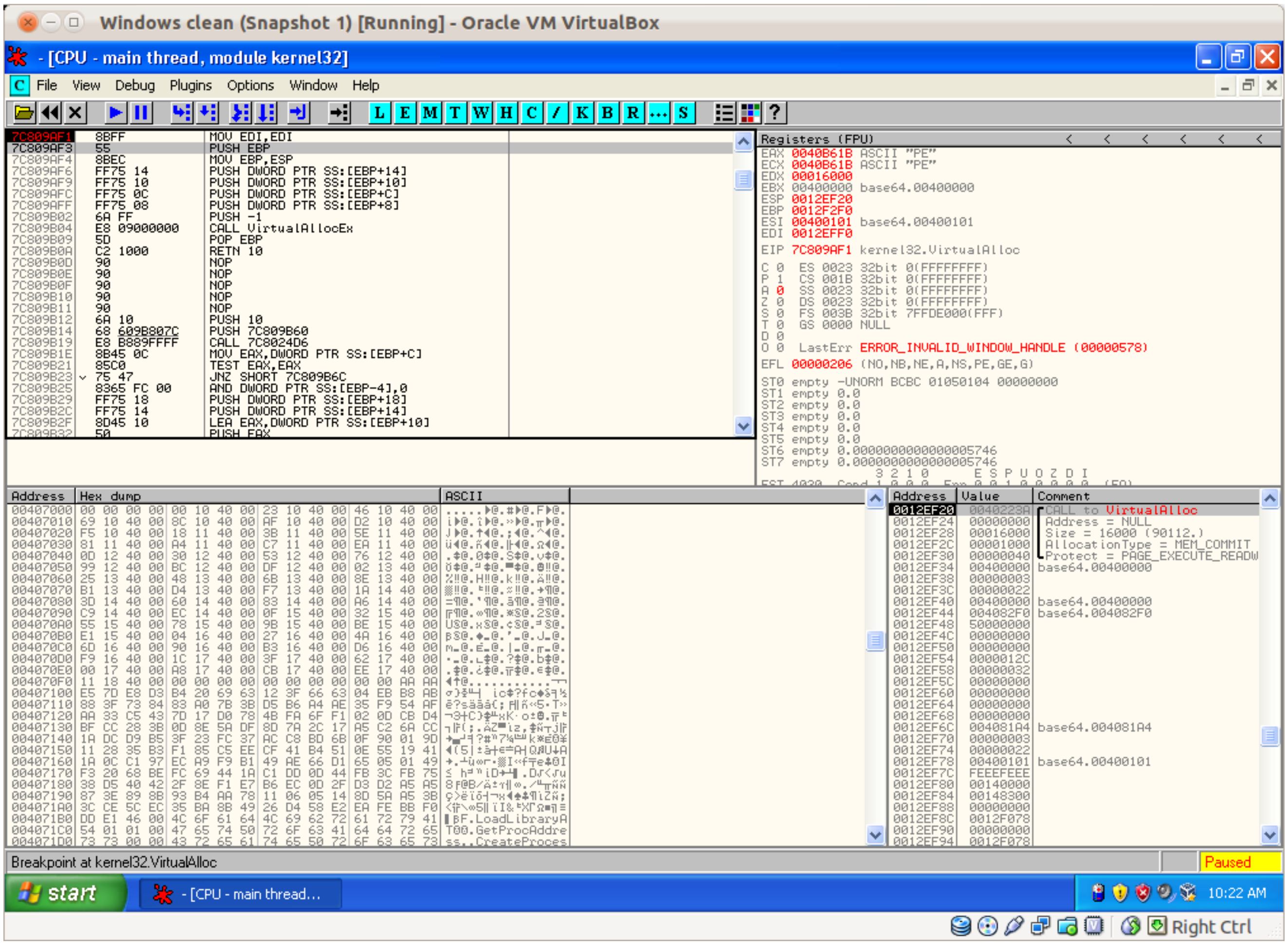
Now we run the malware with F9

A lot of exception must be pass. Use shift+F9 to pass it.



```
00404FFB      00          DB 00
00404FFC      68          DB 68                                   CHAR 'h'
DS:[00409384]=00808080

Address  Hex dump                                         ASCII
00407000 00 00 00 00 00 10 40 00 23 10 40 00 46 10 40 00 .....▶@.#▶@.F▶@.
00407010 69 10 40 00 8C 10 40 00 AF 10 40 00 D2 10 40 00 i▶@.î▶@.»▶@.π▶@.
00407020 F5 10 40 00 18 11 40 00 3B 11 40 00 5E 11 40 00 J▶@.↑◀@.;◀@.^◀@.
00407030 81 11 40 00 A4 11 40 00 C7 11 40 00 EA 11 40 00 ü◀@.ñ◀@.╟◀@.Ω◀@.

Access violation when writing to [00409384] - use Shift+F7/F8/F9 to pass exception to program
```

Now the application is break at kernel32.VirtualAllocEx :


Windows clean (Snapshot 1) [Running] - Oracle VM VirtualBox
- [CPU - main thread, module kernel32]
C File View Debug Plugins Options Window Help
L E M T W H C / K B R ... S
Paused
start
- [CPU - main thread...
10:22 AM
Right Ctrl

```
7C809AF1   8BFF           MOV EDI,EDI
7C809AF3   55             PUSH EBP
7C809AF4   8BEC           MOV EBP,ESP
7C809AF6   FF75 14        PUSH DWORD PTR SS:[EBP+14]
7C809AF9   FF75 10        PUSH DWORD PTR SS:[EBP+10]
7C809AFC   FF75 0C        PUSH DWORD PTR SS:[EBP+C]
7C809AFF   FF75 08        PUSH DWORD PTR SS:[EBP+8]
7C809B02   6A FF          PUSH -1
7C809B04   E8 09000000    CALL VirtualAllocEx
7C809B09   5D             POP EBP
7C809B0A   C2 1000        RETN 10
7C809B0D   90             NOP
7C809B0E   90             NOP
7C809B0F   90             NOP
7C809B10   90             NOP
7C809B11   90             NOP
7C809B12   6A 10          PUSH 10
7C809B14   68 609B807C    PUSH 7C809B60
7C809B19   E8 B889FFFF    CALL 7C8024D6
7C809B1E   8B45 0C        MOV EAX,DWORD PTR SS:[EBP+C]
7C809B21   85C0           TEST EAX,EAX
7C809B23 v 75 47          JNZ SHORT 7C809B6C
7C809B25   8365 FC 00     AND DWORD PTR SS:[EBP-4],0
7C809B29   FF75 18        PUSH DWORD PTR SS:[EBP+18]
7C809B2C   FF75 14        PUSH DWORD PTR SS:[EBP+14]
7C809B2F   8D45 10        LEA EAX,DWORD PTR SS:[EBP+10]
7C809B32   50             PUSH EAX
```

```
Registers (FPU)
EAX 0040B61B ASCII "PE"
ECX 0040B61B ASCII "PE"
EDX 00016000
EBX 00400000 base64.00400000
ESP 0012EF20
EBP 0012F2F0
ESI 00400101 base64.00400101
EDI 0012EFF0

EIP 7C809AF1 kernel32.VirtualAlloc

C 0  ES 0023 32bit 0(FFFFFFFF)
P 1  CS 001B 32bit 0(FFFFFFFF)
A 0  SS 0023 32bit 0(FFFFFFFF)
Z 0  DS 0023 32bit 0(FFFFFFFF)
S 0  FS 003B 32bit 7FFDE000(FFF)
T 0  GS 0000 NULL
D 0
O 0  LastErr ERROR_INVALID_WINDOW_HANDLE (00000578)

EFL 00000206 (NO,NB,NE,A,NS,PE,GE,G)

ST0 empty -UNORM BCBC 01050104 00000000
ST1 empty 0.0
ST2 empty 0.0
ST3 empty 0.0
ST4 empty 0.0
ST5 empty 0.0
ST6 empty 0.0000000000000005746
ST7 empty 0.0000000000000005746
               3 2 1 0      E S P U O Z D I
FST 4020  Cond 1 0 0 0  Err 0 0 1 0 0 0 0 0  (EQ)
```

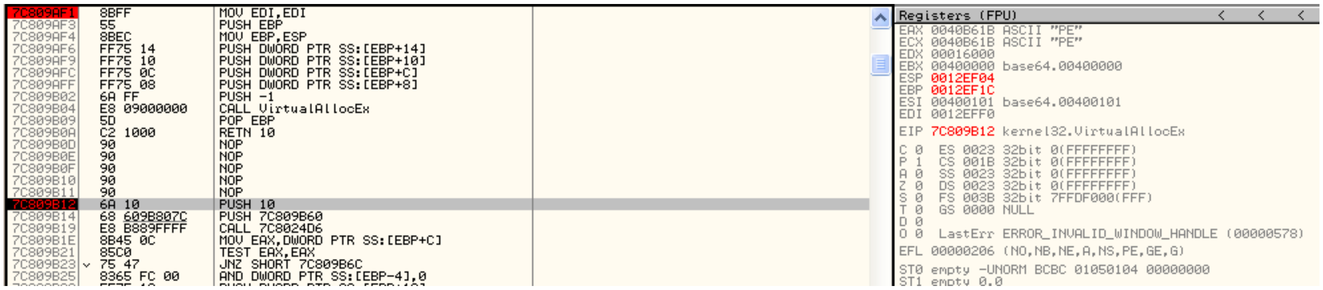
```
Address  Hex dump                                         ASCII
00407000 00 00 00 00 00 10 40 00 23 10 40 00 46 10 40 00  .....►@.#►@.F►@.
00407010 69 10 40 00 8C 10 40 00 AF 10 40 00 D2 10 40 00  i►@.î►@.»►@.╥►@.
00407020 F5 10 40 00 18 11 40 00 3B 11 40 00 5E 11 40 00  J►@.↑◄@.;◄@.^◄@.
00407030 81 11 40 00 A4 11 40 00 C7 11 40 00 EA 11 40 00  ü◄@.ñ◄@.╟◄@.Ω◄@.
00407040 0D 12 40 00 30 12 40 00 53 12 40 00 76 12 40 00  .↕@.0↕@.S↕@.v↕@.
00407050 99 12 40 00 BC 12 40 00 DF 12 40 00 02 13 40 00  Ö↕@.╝↕@.▀↕@.☻‼@.
00407060 25 13 40 00 48 13 40 00 6B 13 40 00 8E 13 40 00  %‼@.H‼@.k‼@.Ä‼@.
00407070 B1 13 40 00 D4 13 40 00 F7 13 40 00 1A 14 40 00  ▒‼@.╘‼@.≈‼@.→¶@.
00407080 3D 14 40 00 60 14 40 00 83 14 40 00 A6 14 40 00  =¶@.`¶@.â¶@.ª¶@.
00407090 C9 14 40 00 EC 14 40 00 0F 15 40 00 32 15 40 00  ╔¶@.∞¶@.☼§@.2§@.
004070A0 55 15 40 00 78 15 40 00 9B 15 40 00 BE 15 40 00  U§@.x§@.¢§@.╛§@.
004070B0 E1 15 40 00 04 16 40 00 27 16 40 00 4A 16 40 00  ß§@.♦▬@.'▬@.J▬@.
004070C0 6D 16 40 00 90 16 40 00 B3 16 40 00 D6 16 40 00  m▬@.É▬@.│▬@.╓▬@.
004070D0 F9 16 40 00 1C 17 40 00 3F 17 40 00 62 17 40 00  ·▬@.∟↨@.?↨@.b↨@.
004070E0 00 17 40 00 A8 17 40 00 CB 17 40 00 EE 17 40 00  .↨@.¿↨@.╦↨@.ε↨@.
004070F0 11 18 40 00 00 00 00 00 00 00 00 00 00 00 AA AA  ◄↑@...........¬¬
00407100 E5 7D E8 D3 B4 20 69 63 12 3F 66 63 04 EB B8 AB  σ}Φ╙┤ ic↕?fc♦δ╕½
00407110 88 3F 73 84 83 A0 7B 3B D5 B6 A4 AE 35 F9 54 AF  ê?säâá{;╒╢ñ«5·T»
00407120 AA 33 C5 43 7D 17 D0 78 4B FA 6F F1 02 0D CB D4  ¬3┼C}↨╨xK·o±☻.╦╘
00407130 BF CC 28 3B 0D 8E 5A DF 8D 7A 2C 17 A5 C2 6A CC  ┐╠(;.ÄZ▀ìz,↨Ñ┬j╠
00407140 1A DC D9 B5 3F 23 FC 37 AC C8 BD 6B 0F 90 01 9D  →▄┘╡?#ⁿ7¼╚╜k☼É☺¥
00407150 11 28 35 B3 F1 85 C5 EE CF 41 B4 51 0E 55 19 41  ◄(5│±à┼ε╧A┤Q♫U↓A
00407160 1A 0C C1 97 EC A9 F9 B1 49 AE 66 D1 65 05 01 49  →.┴ù∞⌐·▒I«f╤e♣☺I
00407170 F3 20 68 BE FC 69 44 1A C1 DD 0D 44 FB 3C FB 75  ≤ h╛ⁿiD→┴▌.D√<√u
00407180 38 D5 40 42 2F 8E F1 E7 B6 EC 0D 2F D3 D2 A5 A5  8╒@B/Ä±τ╢∞./╙╥ÑÑ
00407190 87 3E 89 8B 93 B4 AA 78 11 06 05 14 8D 5A A5 3B  ç>ëïô┤¬x◄♠♣¶ìZÑ;
004071A0 3C CE 5C EC 35 BA 8B 49 26 D4 58 E2 EA FE BB F0  <╬\∞5║ïI&╘XΓΩ■╗≡
004071B0 DD E1 46 00 4C 6F 61 64 4C 69 62 72 61 72 79 41  ▌ßF.LoadLibraryA
004071C0 54 01 01 00 47 65 74 50 72 6F 63 41 64 64 72 65  T☺☺.GetProcAddre
004071D0 73 73 00 00 43 72 65 61 74 65 50 72 6F 63 65 73  ss..CreateProces
```

```
Address  Value    Comment
0012EF20 0040223A ┌CALL to VirtualAlloc
0012EF24 00000000 │Address = NULL
0012EF28 00016000 │Size = 16000 (90112.)
0012EF2C 00001000 │AllocationType = MEM_COMMIT
0012EF30 00000040 └Protect = PAGE_EXECUTE_READW
0012EF34 00400000 base64.00400000
0012EF38 00000003
0012EF3C 00000022
0012EF40 00400000 base64.00400000
0012EF44 004082F0 base64.004082F0
0012EF48 50000000
0012EF4C 00000000
0012EF50 00000000
0012EF54 0000012C
0012EF58 00000032
0012EF5C 00000000
0012EF60 00000000
0012EF64 00000000
0012EF68 00000000
0012EF6C 004081A4 base64.004081A4
0012EF70 00000003
0012EF74 00000022
0012EF78 00400101 base64.00400101
0012EF7C FEEEFEEE
0012EF80 00140000
0012EF84 00148300
0012EF88 00000000
0012EF8C 0012F078
0012EF90 00000000
0012EF94 0012F078
```

Breakpoint at kernel32.VirtualAlloc

Execute the binary until the next RET with Ctrl+F9.

Now we can see the allocated address of the memory in the EAX register: 0x40B61B.

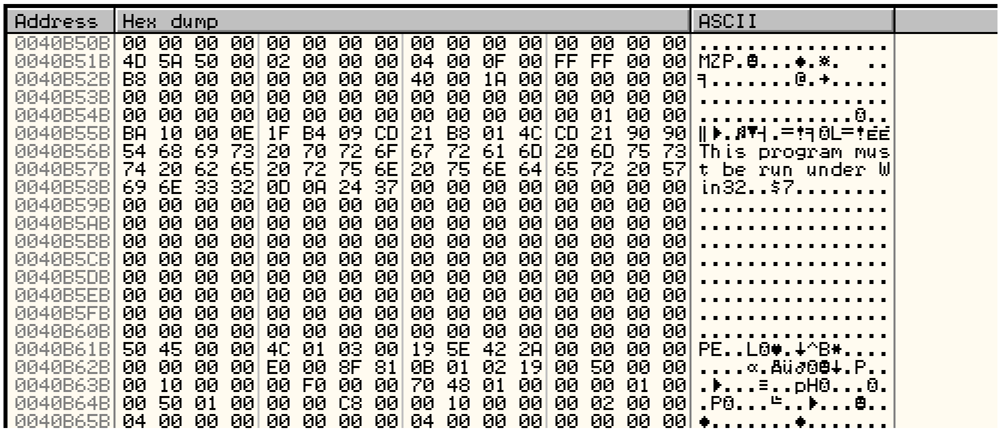


```
7C809AF1   8BFF           MOV EDI,EDI
7C809AF3   55             PUSH EBP
7C809AF4   8BEC           MOV EBP,ESP
7C809AF6   FF75 14        PUSH DWORD PTR SS:[EBP+14]
7C809AF9   FF75 10        PUSH DWORD PTR SS:[EBP+10]
7C809AFC   FF75 0C        PUSH DWORD PTR SS:[EBP+C]
7C809AFF   FF75 08        PUSH DWORD PTR SS:[EBP+8]
7C809B02   6A FF          PUSH -1
7C809B04   E8 09000000    CALL VirtualAllocEx
7C809B09   5D             POP EBP
7C809B0A   C2 1000        RETN 10
7C809B0D   90             NOP
7C809B0E   90             NOP
7C809B0F   90             NOP
7C809B10   90             NOP
7C809B11   90             NOP
7C809B12   6A 10          PUSH 10
7C809B14   68 609B807C    PUSH 7C809B60
7C809B19   E8 B889FFFF    CALL 7C8024D6
7C809B1E   8B45 0C        MOV EAX,DWORD PTR SS:[EBP+C]
7C809B21   85C0           TEST EAX,EAX
7C809B23 v 75 47          JNZ SHORT 7C809B6C
7C809B25   8365 FC 00     AND DWORD PTR SS:[EBP-4],0
```

```
Registers (FPU)
EAX 0040B61B ASCII "PE"
ECX 0040B61B ASCII "PE"
EDX 00016000
EBX 00400000 base64.00400000
ESP 0012EF04
EBP 0012EF1C
ESI 00400101 base64.00400101
EDI 0012EFF0

EIP 7C809B12 kernel32.VirtualAllocEx

C 0  ES 0023 32bit 0(FFFFFFFF)
P 1  CS 001B 32bit 0(FFFFFFFF)
A 0  SS 0023 32bit 0(FFFFFFFF)
Z 0  DS 0023 32bit 0(FFFFFFFF)
S 0  FS 003B 32bit 7FFDF000(FFF)
T 0  GS 0000 NULL
D 0
O 0  LastErr ERROR_INVALID_WINDOW_HANDLE (00000578)

EFL 00000206 (NO,NB,NE,A,NS,PE,GE,G)

ST0 empty -UNORM BCBC 01050104 00000000
ST1 empty 0.0
```

Right click on the EAX value, and click on “Follow in dump”.

We can see a PE value in the bottom left. If we scroll we can see the complete MZ :



```
Address  Hex dump                                         ASCII
0040B50B 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0040B51B 4D 5A 50 00 02 00 00 00 04 00 0F 00 FF FF 00 00 MZP.☻...♦.☼.  ..
0040B52B B8 00 00 00 00 00 00 00 40 00 1A 00 00 00 00 00 ╕.......@.→.....
0040B53B 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0040B54B 00 00 00 00 00 00 00 00 00 00 00 00 00 01 00 00 ..............☺..
0040B55B BA 10 00 0E 1F B4 09 CD 21 B8 01 4C CD 21 90 90 ║►.♫▼┤.═!╕☺L═!ÉÉ
0040B56B 54 68 69 73 20 70 72 6F 67 72 61 6D 20 6D 75 73 This program mus
0040B57B 74 20 62 65 20 72 75 6E 20 75 6E 64 65 72 20 57 t be run under W
0040B58B 69 6E 33 32 0D 0A 24 37 00 00 00 00 00 00 00 00 in32..$7........
0040B59B 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0040B5AB 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0040B5BB 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0040B5CB 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0040B5DB 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0040B5EB 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0040B5FB 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0040B60B 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0040B61B 50 45 00 00 4C 01 03 00 19 5E 42 2A 00 00 00 00 PE..L☺♥.↓^B*....
0040B62B 00 00 00 00 E0 00 8F 81 0B 01 02 19 00 50 00 00 ....α.Åü♂☺☻↓.P..
0040B63B 00 10 00 00 00 F0 00 00 70 48 01 00 00 00 01 00 .►...≡..pH☺...☺.
0040B64B 00 50 01 00 00 00 C8 00 00 10 00 00 00 02 00 00 .P☺...╚..►...☻..
0040B65B 04 00 00 00 00 00 00 00 04 00 00 00 00 00 00 00 ♦.......♦.......
```

Now we can use lordPE to make a partial dump: - launch LordPE

- right click on the process
- Dump partial
- set the start address to 40B51B
- set the size to 411000 - 40B51B = 5AE5

Now we have a binary with the md5: 18e5ff1d0610341257f33e6fefe4f9a7

## Third binary

We used yara to identify the binary:

```
rootbsd@alien:~/Samples$ yara -r packer.yara base64.stage2.dmp
UPXv20MarkusLaszloReiser base64.stage2.dmp
UPXV200V290MarkusOberhumerLaszloMolnarJohnReiser base64.stage2.dmp
UPX20030XMarkusOberhumerLaszloMolnarJohnReiser base64.stage2.dmp
```

The binary is simply pack with UPX.

```
rootbsd@alien:~/Samples$ upx -o base64.stage2.exe -d base64.stage2.dmp
                       Ultimate Packer for eXecutables
                          Copyright (C) 1996 - 2010
UPX 3.07        Markus Oberhumer, Laszlo Molnar & John Reiser   Sep 08th 2010

        File size         Ratio      Format      Name
   --------------------   ------   -----------   -----------
     46821 <-     23269   49.70%    win32/pe     base64.stage2.exe

Unpacked 1 file.
rootbsd@alien:~/Samples$ file base64.stage2.exe
base64.stage2.exe: PE32 executable for MS Windows (GUI) Intel 80386 32-bit
```

We have got the final binary.

## Fourth binary

We easily identify a well-known RAT:

```
rootbsd@alien:~/Samples$ strings -el base64.stage2.exe  | grep RAT
Xtreme RAT SOFTWARE\XtremeRAT
```

After a quick search on Google, we discovered that the RAT could be buy here: https://sites.google.com/site/nxtremerat/.

The second interesting think is that fact that the RAT is used in Syria : https://www.eff.org/deeplinks/2012/03/how-find-syrian-government-malware-your-computer-and-remove-it/

We can use 3 methods to analyse the binary: the simple, the semi talented method and the full talented method.

### Simple

We execute it, and launch netstat.exe on Windows. The IP of the C&C is 41.103.186.12 and port 2013.

It’s an IP from Alger:

```
rootbsd@alien:~/Samples$ whois 41.103.186.12

% This is the AfriNIC Whois server.

% Note: this output has been filtered.

%Information related to '41.103.0.0 - 41.103.255.255'

inetnum:        41.103.0.0 - 41.103.255.255
netname:        RegAlg1
descr:          Region Alger 1
country:        DZ
admin-c:        SD6-AFRINIC
tech-c:         SD6-AFRINIC
status:         ASSIGNED PA
mnt-by:         DJAWEB-MNT
source:         AFRINIC # Filtered
parent:         41.96.0.0 - 41.111.255.255

person:         Security Departement
address:        Alger
phone:          +21321922004
fax-no:         +21321922004
e-mail:         security@djaweb.dz
nic-hdl:        SD6-AFRINIC
source:         AFRINIC # Filtered
```

To be persitent, the malware adds a value (antivirus) in the registry: Software\Microsoft\Windows\CurrentVersion\Run

The malware is stored in the directory: C:\Windows\Browser\Web.exe

A configuration file is available here: C:\Documents and Settings\rootbsd\Application Data\Microsoft\Windows\S5tVn.cfg

## Semi talented

We can use a memory dump to analyse the binary. We use volatility to analyse the binary:

```
rootbsd@alien:~/Samples$ volatility/vol.py -f output pslist
Volatile Systems Volatility Framework 2.0
 Offset(V)  Name                 PID    PPID   Thds   Hnds   Time
---------- -------------------- ------ ------ ------ ------ -------------------
0x812ed020 System                    4      0     54    247 1970-01-01 00:00:00
0xffbaeb10 smss.exe                368      4      3     19 2012-05-21 15:20:54
0x811248e0 csrss.exe               584    368     10    379 2012-05-21 15:20:54
0x81197248 winlogon.exe            608    368     21    514 2012-05-21 15:20:54
0x811275a8 services.exe            652    608     16    253 2012-05-21 15:20:54
0x8112d7e0 lsass.exe               664    608     23    338 2012-05-21 15:20:54
0xffbd7a78 VBoxService.exe         820    652      8    106 2012-05-21 15:20:54
0x81180c30 svchost.exe             864    652     19    197 2012-05-21 06:20:56
0x811a6b28 svchost.exe             952    652      9    237 2012-05-21 06:20:56
0xffac4218 svchost.exe            1044    652     79   1367 2012-05-21 06:20:56
0xffabbd08 svchost.exe            1092    652      6     76 2012-05-21 06:20:56
0x8116cda0 svchost.exe            1132    652     13    172 2012-05-21 06:20:56
0x8112eca8 spoolsv.exe            1544    652     14    111 2012-05-21 06:20:57
0xffa93b00 explorer.exe           1556   1504     17    477 2012-05-21 06:20:57
0x8112fda0 VBoxTray.exe           1700   1556      6     58 2012-05-21 06:20:57
0xffb95da0 svchost.exe            1904    652      4    106 2012-05-21 06:21:05
0xffa01a98 alg.exe                1076    652      6    107 2012-05-21 06:21:09
0x81178278 wscntfy.exe            1188   1044      1     31 2012-05-21 06:21:11
0x81188da0 wuauclt.exe            1956   1044      8    180 2012-05-21 06:21:51
0x811323c0 wuauclt.exe             248   1044      4    133 2012-05-21 06:22:05
0x8119ada0 svchost.exe            2000   1488      2     41 2012-07-20 19:15:47
0x8118b888 svchost.exe            1404   1488      8    188 2012-07-20 19:15:47
```

The 2 last svchost.exe are stange. The date is not logic.

When you list the dll you can see that the malware change his name to svchost.exe:

```
rootbsd@alien:~/Samples$ ../Pentest/volatility/vol.py -f output -p 2000 dlllist
Volatile Systems Volatility Framework 2.0
************************************************************************
svchost.exe pid:   2000
Command line : svchost.exe
Service Pack 3

Base         Size         Path
0x00400000   0x038000     E:\essai\svchost.exe
0x7c900000   0x0b2000     C:\WINXP\system32\ntdll.dll
0x7c800000   0x0f6000     C:\WINXP\system32\kernel32.dll
0x7e410000   0x091000     C:\WINXP\system32\user32.dll
0x77f10000   0x049000     C:\WINXP\system32\GDI32.dll
0x76390000   0x01d000     C:\WINXP\system32\IMM32.DLL
0x77dd0000   0x09b000     C:\WINXP\system32\ADVAPI32.dll
0x77e70000   0x093000     C:\WINXP\system32\RPCRT4.dll
0x77fe0000   0x011000     C:\WINXP\system32\Secur32.dll
0x7c9c0000   0x818000     C:\WINXP\system32\shell32.dll
0x77c10000   0x058000     C:\WINXP\system32\msvcrt.dll
0x77f60000   0x076000     C:\WINXP\system32\SHLWAPI.dll
0x773d0000   0x103000     C:\WINXP\WinSxS\x86_Microsoft.Windows.Common-
Controls_6595b64144ccf1df_6.0.2600.6028_x-ww_61e65202\comctl32.dll
0x5d090000   0x09a000     C:\WINXP\system32\comctl32.dll
```

We make a memory dump of the process 1404 :

```
rootbsd@alien:~/Samples$ volatility/vol.py -f output -p 1404 memdump -D .
Volatile Systems Volatility Framework 2.0
************************************************************************
Writing svchost.exe [  1404] to 1404.dmp
```

In the .dmp we have got all necessary information:

```
rootbsd@alien:~/Samples$ strings -a  1404.dmp | grep http://
[...]
http://baloobadjamel.hopto.org:2013/1234567890.functions
[...]
rootbsd@alien:~/Samples$ nslookup baloobadjamel.hopto.org
Server:         192.168.0.254
Address:        192.168.0.254#53

Non-authoritative answer:
Name:   baloobadjamel.hopto.org
Address: 41.103.186.12
```

And we find the IP.

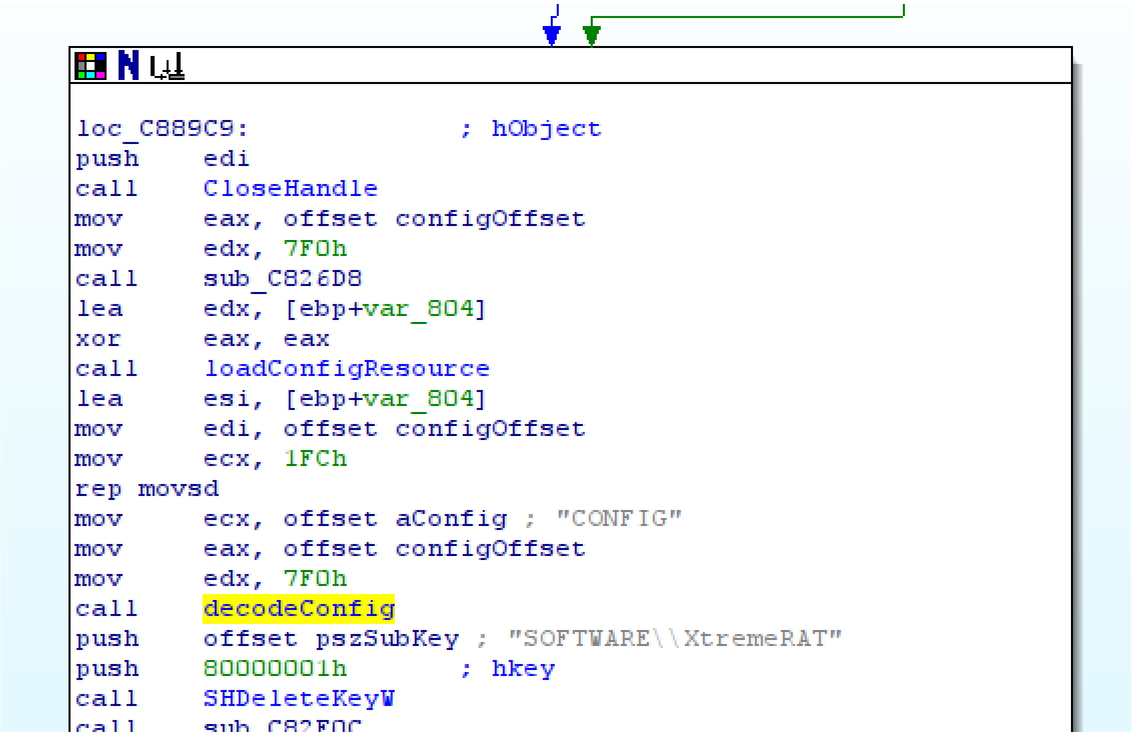
We hope that Djamel Baloodad is not the real name of the owner of the C&C ;)

## Talented

We open the final binary on IDA.

To help us you can find the .idb here

At loc_C889C9, we find two functions sub_C93B1C (loadConfigResource) and sub_C82914 (decondeConfig).



```
loc_C889C9:              ; hObject
push    edi
call    CloseHandle
mov     eax, offset configOffset
mov     edx, 7F0h
call    sub_C826D8
lea     edx, [ebp+var_804]
xor     eax, eax
call    loadConfigResource
lea     esi, [ebp+var_804]
mov     edi, offset configOffset
mov     ecx, 1FCh
rep movsd
mov     ecx, offset aConfig ; "CONFIG"
mov     eax, offset configOffset
mov     edx, 7F0h
call    decodeConfig
push    offset pszSubKey ; "SOFTWARE\\XtremeRAT"
push    80000001h        ; hkey
call    SHDeleteKeyW
call    sub_C82F0C
```

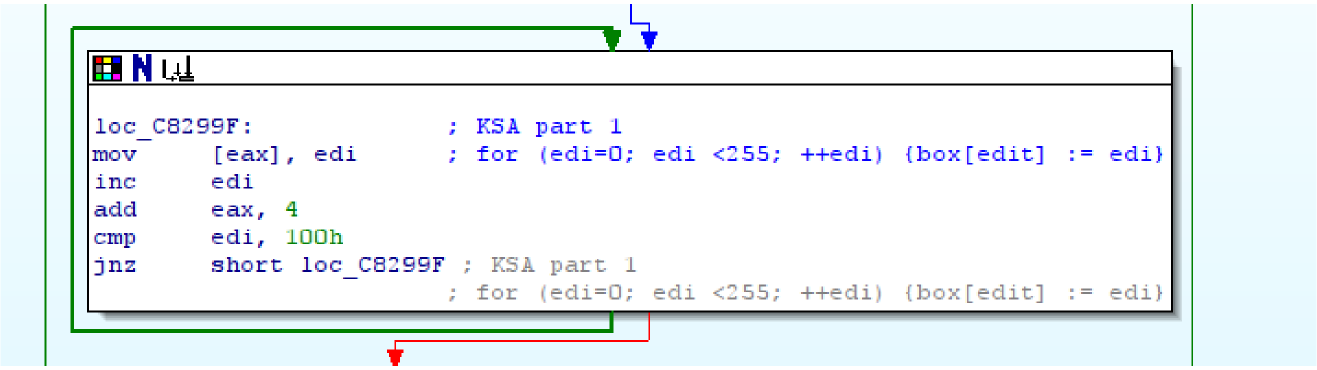
The fisrt function extracts a resource. This resource is the config file (in this case S5tVn.cfg).

The second function decode the configuration file. Two interesting arguments are passed ton the function: the offset of the config file & the word “CONFIG” (in unicode).

This function is composed of 3 loops. This kind of layout looks like RC4 (RC4) :

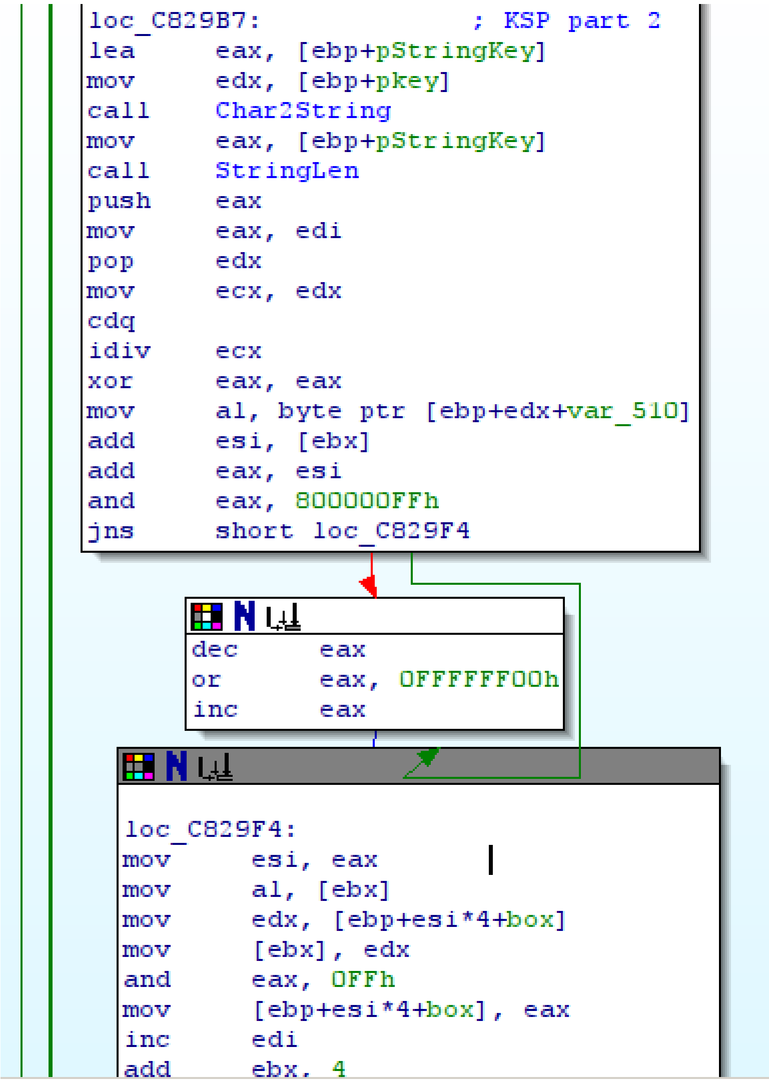
- 2 loops KSA (KSA)
- 1 loop for PRGA (PRGA).

The first loop:



```
loc_C8299F:              ; KSA part 1
mov     [eax], edi       ; for (edi=0; edi <255; ++edi) {box[edit] := edi}
inc     edi
add     eax, 4
cmp     edi, 100h
jnz     short loc_C8299F ; KSA part 1
                         ; for (edi=0; edi <255; ++edi) {box[edit] := edi}
```

The second loop:

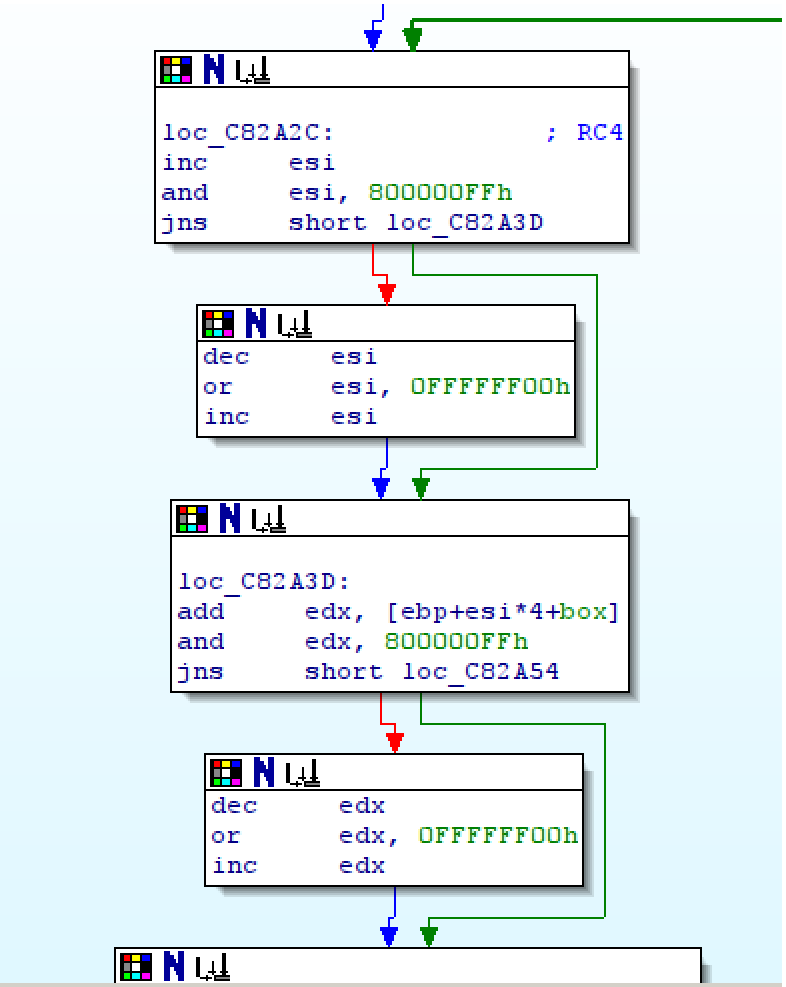


```
loc_C829B7:             ; KSP part 2
lea     eax, [ebp+pStringKey]
mov     edx, [ebp+pkey]
call    Char2String
mov     eax, [ebp+pStringKey]
call    StringLen
push    eax
mov     eax, edi
pop     edx
mov     ecx, edx
cdq
idiv    ecx
xor     eax, eax
mov     al, byte ptr [ebp+edx+var_510]
add     esi, [ebx]
add     eax, esi
and     eax, 800000FFh
jns     short loc_C829F4
```

```
dec     eax
or      eax, 0FFFFFF00h
inc     eax
```

```
loc_C829F4:
mov     esi, eax
mov     al, [ebx]
mov     edx, [ebp+esi*4+box]
mov     [ebx], edx
and     eax, 0FFh
mov     [ebp+esi*4+box], eax
inc     edi
add     ebx, 4
```

And the final loop:



```
loc_C82A2C:                ; RC4
inc     esi
and     esi, 800000FFh
jns     short loc_C82A3D
```

```
dec     esi
or      esi, 0FFFFFF00h
inc     esi
```

```
loc_C82A3D:
add     edx, [ebp+esi*4+box]
and     edx, 800000FFh
jns     short loc_C82A54
```

```
dec     edx
or      edx, 0FFFFFF00h
inc     edx
```

So the config file is crypted with RC4 with the key “CONFIG”.

To perform a RC4 encryption we need the length of the key. To have this size the developer mades his own function sub_C81AF8 (StringLen) but this function does not support unicode, it returns 6 and not 12. So we must implemente this bug in our tool to decrypt the config file.

A script to decode the config file is available here

```
rootbsd@alien:~/Samples$ ./xtremerat_config.py xtreme.exe | strings -el
baloobadjamel.hopto.org
Spam2013
teSpam2013
Web.exe
Browser
svchost.exe
Antivirus
Antivirus
 P8CWY65J-GY7I-CD3S-7K6Q-BD3A60R037L3
Server
3.5 Private
S5tVn
S5tVnEXIT
S5tVnPERSIST
ftp.ftpserver.com
pData\Local
ftpuser
ftppass
Error
ivateAn unexpected error occurred when starting the program.
Please try again later.
```

We can already see the C&C, the port, etc…

We are working on the format on the configuration file, for the moment we identify this format:

```
rootbsd@alien:~/Samples$ ./xtremerat_config.py -d xtreme.exe
name10: 3.5 PrivateS5tV
name11: stS5tVnEXI
name6: Antivirus
name7: Antivirus
host: baloobadjamel.hopto.org
num: 101
name2: teSpam2013
name3: Web.exe
port: 2013
name8:  P8CWY65J-GY7I-CD3S-7K6Q-BD3A60R037L3
name9: Server
name: Spam2013
name4: Browser
name5: svchost.exe
```