

Malware.lu - Xtreme RAT analysis

Archived: 2026-04-05 14:43:01 UTC

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 télégramme 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

- 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
-----
-----
```



```
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 surprised by a lot of exception when we tried to debug the sample.

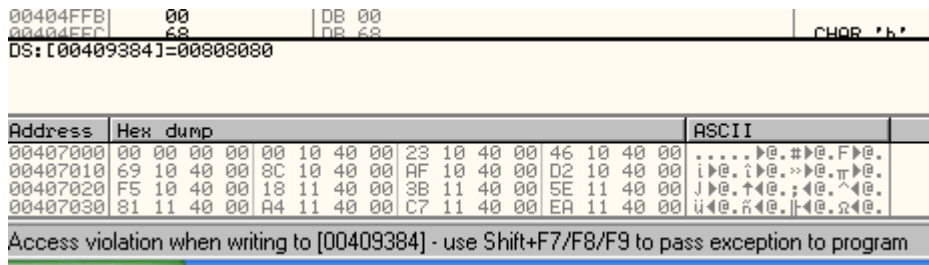
In fact this malware voluntary uses and traps exceptions to be unpacked.

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

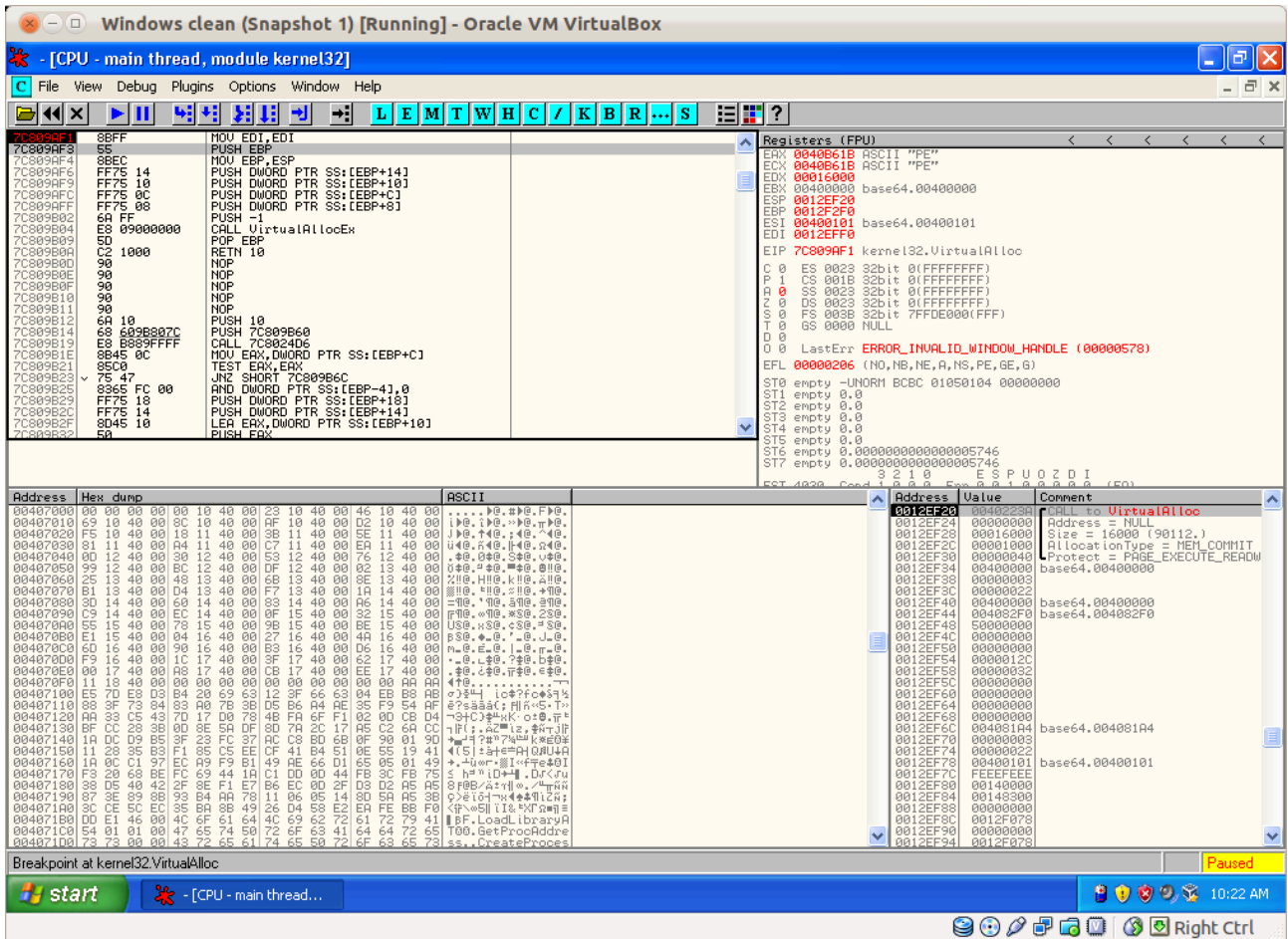
- View
- Executable modules
- right click on kernel32.dll -> View names
- F2 on VirtualAlloc & VirtualAllocEx

Now we run the malware with F9

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

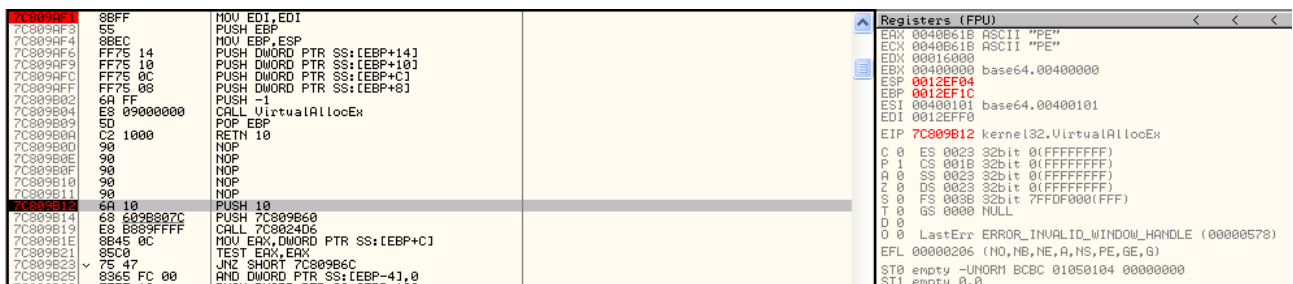


Now the application is break at kernel32.VirtualAllocEx :



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.



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	7..... @. +.
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	0A 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 60 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 00 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. Au@ @. 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 persistent, 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
0xffabb08 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 strange. 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_
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 (decodeConfig).



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

The second function decodes the configuration file. Two interesting arguments are passed to 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:

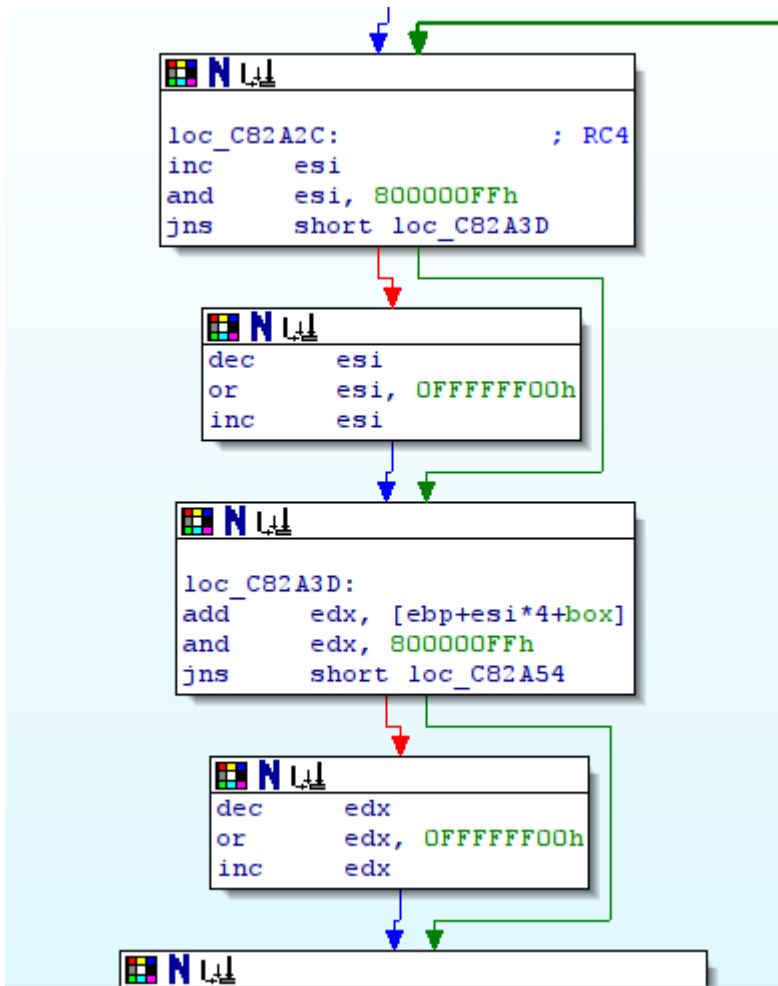
```
loc_C8299F:                ; KSA part 1
mov     [eax], edi        ; for (edi=0; edi <255; ++edi) (box[edit] := edi)
inc     edi
add     eax, 4
cmp     edi, 100h
jnz     short loc_C8299F ; KSA part 1
                                ; for (edi=0; edi <255; ++edi) (box[edit] := edi)
```

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

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:



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

To perform a RC4 encryption we need the length of the key. To have this size the developer makes 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
```

Source: <https://malware.lu/articles/2012/07/22/xtreme-rat-analysis.html>