virus BULLETIN Covering the global threat landscape

CUSTOM PACKER DEFEATS MULTIPLE AUTOMATION SYSTEMS

Ke Zhang
Symantec, Japan

Automation systems are an enormous help to malware analysts, enabling them both to identify malicious files and determine their functionalities quickly. However, malware authors are constantly working on new ways to defeat automation systems, in order to increase the length of time it takes for the malware to be detected. In my recent work, I came across a custom packer combining anti-automation, anti-VM and anti-reverse-engineering abilities. I studied the packer in some depth, although I was not able to determine the name under which it is being sold.

ANTI-AUTOMATION

In a normal system, the foreground window changes when the user switches between different tasks. In an automation system, however, there is usually only a single task: running a potentially malicious sample and monitoring its behaviour.

The custom packer makes clever use of this difference between the two types of system. First, it calls GetForegroundWindow() and saves the window handle, then it continuously calls the same function and checks whether the foreground window has changed (see Figure 1). The rest of the code won't be executed until the window has changed.

It also calls GetModuleFileNameW() and GetUserNameExW() to examine whether the file and user name contain the keywords 'sample', 'sandbox' or 'virus', which are all likely to appear in automation systems.

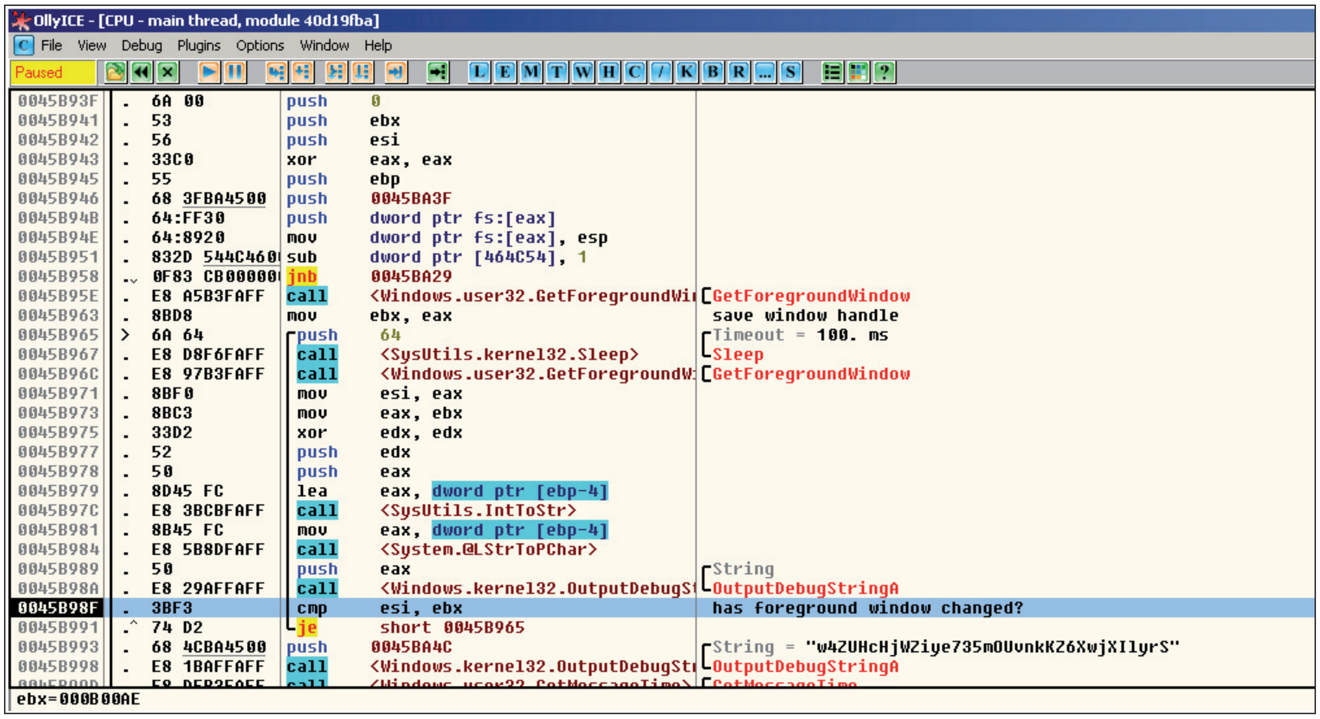
ANTI-VM

The packer detects whether it is being run inside a virtual machine by checking registry values, checking for the existence of certain files, a certain process module, and IO control code.

1. Registry value

The packer reads the following registry value and checks whether it contains the substrings 'vmware', 'qemu' or 'vbox', which are all used by popular virtual machines:

```
HKEY_LOCAL_MACHINE\HARDWARE\DEVICEMAP\Scsi\Scsi Port
0\Scsi Bus 0\Target Id 0\LogicIdentifier
```


OllyICE - [CPU - main thread, module 40d19fba]
File View Debug Plugins Options Window Help
Paused
L E M T W H C / K B R ... S

```
0045B93F  .   6A 00           push    0
0045B941  .   53              push    ebx
0045B942  .   56              push    esi
0045B943  .   33C0            xor     eax, eax
0045B945  .   55              push    ebp
0045B946  .   68 3FBA4500     push    0045BA3F
0045B94B  .   64:FF30         push    dword ptr fs:[eax]
0045B94E  .   64:8920         mov     dword ptr fs:[eax], esp
0045B951  .   832D 544C460    sub     dword ptr [464C54], 1
0045B958  .v  0F83 CB00000    jnb     0045BA29
0045B95E  .   E8 A5B3FAFF     call    <Windows.user32.GetForegroundWi [GetForegroundWindow
0045B963  .   8BD8            mov     ebx, eax                         save window handle
0045B965  >   6A 64          [push     64                             [Timeout = 100. ms
0045B967  .   E8 D8F6FAFF     call     <SysUtils.kernel32.Sleep>      [Sleep
0045B96C  .   E8 97B3FAFF     call     <Windows.user32.GetForegroundW [GetForegroundWindow
0045B971  .   8BF0            mov      esi, eax
0045B973  .   8BC3            mov      eax, ebx
0045B975  .   33D2            xor      edx, edx
0045B977  .   52              push     edx
0045B978  .   50              push     eax
0045B979  .   8D45 FC         lea      eax, dword ptr [ebp-4]
0045B97C  .   E8 3BCBFAFF     call     <SysUtils.IntToStr>
0045B981  .   8B45 FC         mov      eax, dword ptr [ebp-4]
0045B984  .   E8 5B8DFAFF     call     <System.@LStrToPChar>
0045B989  .   50              push     eax                            [String
0045B98A  .   E8 29AFFAFF     call     <Windows.kernel32.OutputDebugS [OutputDebugStringA
0045B98F  .   3BF3            cmp      esi, ebx                        has foreground window changed?
0045B991  .^  74 D2          [je       short 0045B965
0045B993  .   68 4CBA4500     push    0045BA4C                        [String = "w4ZUHcHjWZiye735mOUvnkKZ6XwjXIlyrS"
0045B998  .   E8 1BAFFAFF     call    <Windows.kernel32.OutputDebugSt [OutputDebugStringA
ebx=000B00AE
```

Figure 1: Checking the foreground window.

It then reads another registry value (shown below) and checks whether it contains the substrings ‘qemu’ or ‘vbox’ – again, the presence of either of these indicates that the machine is running inside a virtual machine.

```
HARDWARE\Description\System\SystemBiosVersion\
SystemBiosVersion
```

2. File existence

The packer checks for the existence of the following files.

- %system32%\drivers\vmmouse.sys
- %system32%\drivers\vmhgfs.sys
- %system32%\drivers\VBoxMouse.sys
- %system32%\drivers\VBoxGuest.sys

The first two are used by *VMware*; the latter two are used by *VirtualBox*.

3. Process module

The packer checks whether the process module ‘sbiedll.dll’ is loaded. This DLL is used by the *Sandboxie* sandbox.

4. IO control code 0x2D1400

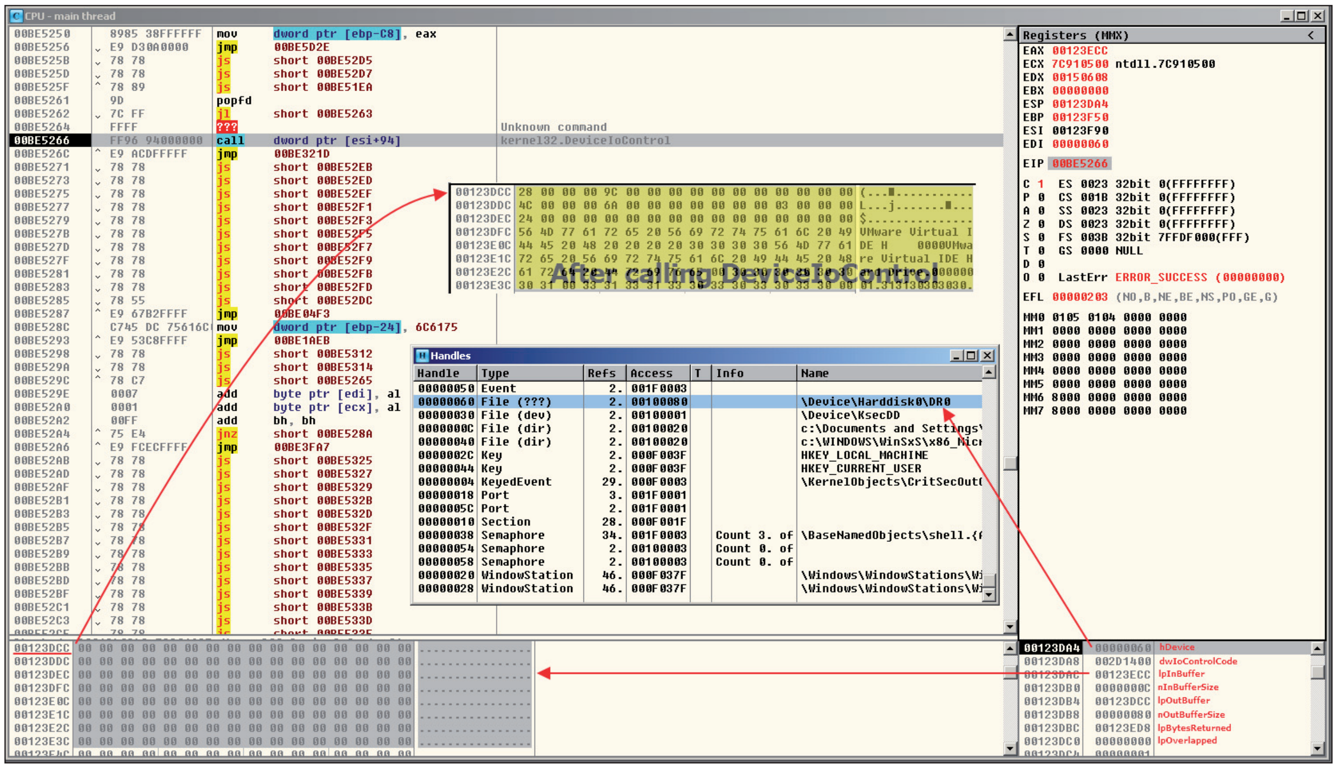
The packer opens hard drive ‘\\.\PhysicalDrive0’ and sends control code 0x2D1400 to it (see Figure 2). It then checks whether the output buffer contains any of the following strings:

- ‘vbox’
- ‘qemu’
- ‘vmware’
- ‘virtual’
- ‘qm00001’
- ‘array’
- ‘00000000000000000001’

ANTI-REVERSE-ENGINEERING

The packer uses a lot of forward and backward unconditional jump instructions to split the sequential code. This makes it difficult for an analyst to get the whole picture, especially when following the code flow in a debugger. Figure 3 demonstrates the packer’s obfuscated strstr() function.

As shown in Figure 4, after all the redundant jumps have been removed, the number of blocks decreases by more than half (from 23 to 10) and the logic looks much simpler.



```
00BE5250    8985 38FFFFFF   mov    dword ptr [ebp-C8], eax
00BE5256  ⌄ E9 D30A0000     jmp    00BE5D2E
00BE525B  ⌄ 78 78           js     short 00BE52D5
00BE525D  ⌄ 78 78           js     short 00BE52D7
00BE525F  ^ 78 89           js     short 00BE51EA
00BE5261    9D              popfd
00BE5262  ⌄ 7C FF           jl     short 00BE5263
00BE5264    FFFF            ???                                Unknown command
00BE5266    FF96 94000000   call   dword ptr [esi+94]          kernel32.DeviceIoControl
00BE526C  ^ E9 ACDFFFFF     jmp    00BE321D
00BE5271  ⌄ 78 78           js     short 00BE52EB
00BE5273  ⌄ 78 78           js     short 00BE52ED
00BE5275  ⌄ 78 78           js     short 00BE52EF
00BE5277  ⌄ 78 78           js     short 00BE52F1
00BE5279  ⌄ 78 78           js     short 00BE52F3
00BE527B  ⌄ 78 78           js     short 00BE52F5
00BE527D  ⌄ 78 78           js     short 00BE52F7
00BE527F  ⌄ 78 78           js     short 00BE52F9
00BE5281  ⌄ 78 78           js     short 00BE52FB
00BE5283  ⌄ 78 78           js     short 00BE52FD
00BE5285  ⌄ 78 55           js     short 00BE52DC
00BE5287  ^ E9 67B2FFFF     jmp    00BE04F3
00BE528C    C745 DC 75616C( mov    dword ptr [ebp-24], 6C6175
00BE5293  ^ E9 53C8FFFF     jmp    00BE1AEB
00BE5298  ⌄ 78 78           js     short 00BE5312
00BE529A  ⌄ 78 78           js     short 00BE5314
00BE529C  ^ 78 C7           js     short 00BE5265
00BE529E    0007            add    byte ptr [edi], al
00BE52A0    0001            add    byte ptr [ecx], al
00BE52A2    00FF            add    bh, bh
00BE52A4  ^ 75 E4           jnz    short 00BE528A
00BE52A6  ^ E9 FCECFFFF     jmp    00BE3FA7
00BE52AB  ⌄ 78 78           js     short 00BE5325
00BE52AD  ⌄ 78 78           js     short 00BE5327
00BE52AF  ⌄ 78 78           js     short 00BE5329
00BE52B1  ⌄ 78 78           js     short 00BE532B
00BE52B3  ⌄ 78 78           js     short 00BE532D
00BE52B5  ⌄ 78 78           js     short 00BE532F
00BE52B7  ⌄ 78 78           js     short 00BE5331
00BE52B9  ⌄ 78 78           js     short 00BE5333
00BE52BB  ⌄ 78 78           js     short 00BE5335
00BE52BD  ⌄ 78 78           js     short 00BE5337
00BE52BF  ⌄ 78 78           js     short 00BE5339
00BE52C1  ⌄ 78 78           js     short 00BE533B
00BE52C3  ⌄ 78 78           js     short 00BE533D
```

```
00123DCC 28 00 00 00 9C 00 00 00 00 00 00 00 00 00 00 00 (...█...........
00123DDC 4C 00 00 00 6A 00 00 00 00 00 00 00 03 00 00 00 L...j.......█...
00123DEC 24 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 $...............
00123DFC 56 4D 77 61 72 65 20 56 69 72 74 75 61 6C 20 49 VMware Virtual I
00123E0C 44 45 20 48 20 20 20 20 30 30 30 30 56 4D 77 61 DE H    0000VMwa
00123E1C 72 65 20 56 69 72 74 75 61 6C 20 49 44 45 20 48 re Virtual IDE H
00123E2C 61 72 64 20 44 72 69 76 65 00 30 30 30 30 30 30 ard Drive.000000
00123E3C 30 31 00 33 31 33 31 33 30 33 30 33 30 33 30 00 01.3131303030.
```

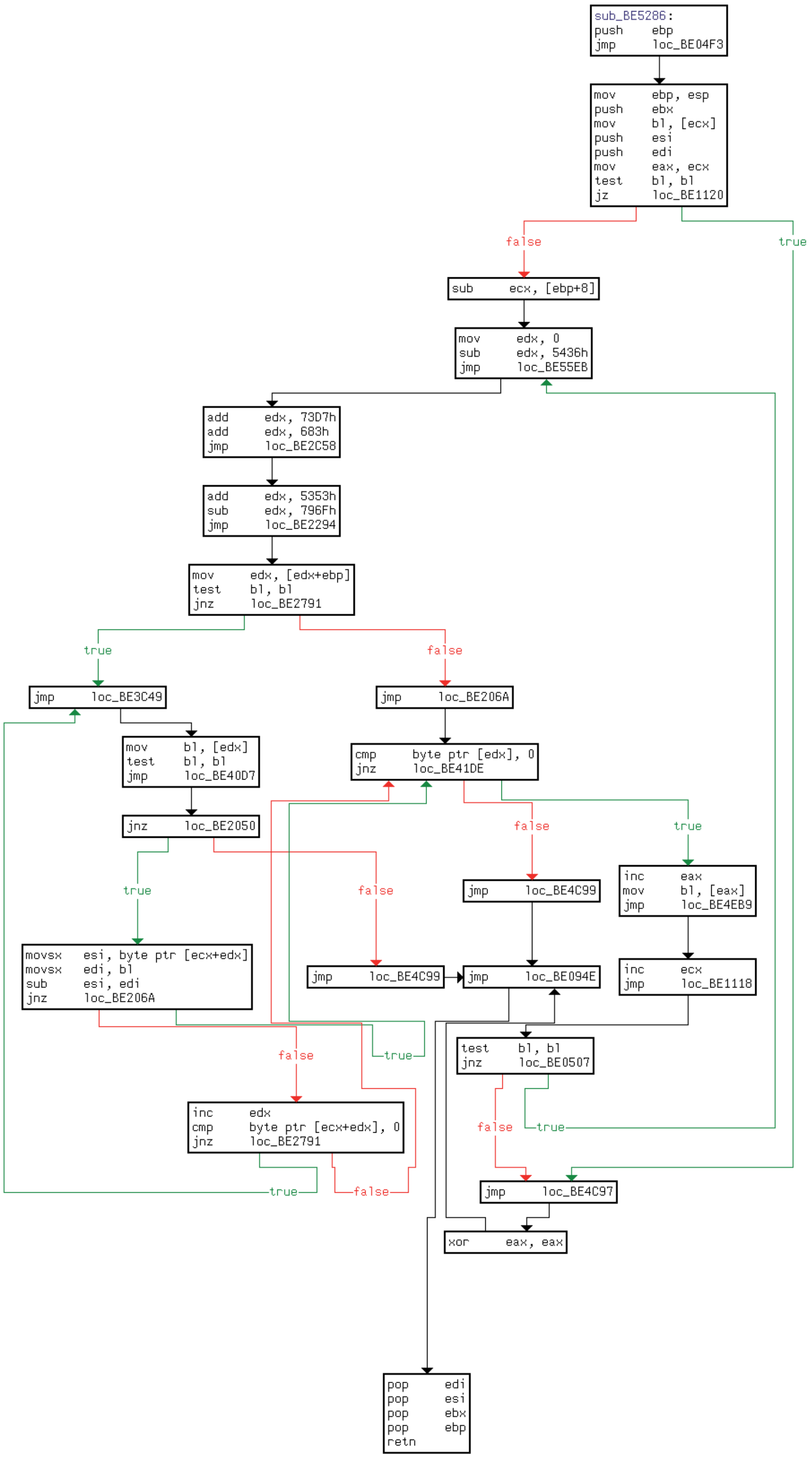
```
Handle   Type           Refs  Access   T Info          Name
00000050 Event            2.  001F0003
00000060 File (???)       2.  00100080                 \Device\Harddisk0\DR0
00000030 File (dev)       2.  00100001                 \Device\KsecDD
0000000C File (dir)       2.  00100020                 c:\Documents and Settings\
00000040 File (dir)       2.  00100020                 c:\WINDOWS\WinSxS\x86_Micr
0000002C Key              2.  000F003F                 HKEY_LOCAL_MACHINE
00000044 Key              2.  000F003F                 HKEY_CURRENT_USER
00000004 KeyedEvent      29.  000F0003                 \KernelObjects\CritSecOut(
00000018 Port             3.  001F0001
0000005C Port             2.  001F0001
00000010 Section         28.  000F001F
00000038 Semaphore       34.  001F0003   Count 3. of   \BaseNamedObjects\shell.{(
00000054 Semaphore        2.  00100003   Count 0. of
00000058 Semaphore        2.  00100003   Count 0. of
00000020 WindowStation   46.  000F037F                 \Windows\WindowStations\Wi
00000028 WindowStation   46.  000F037F                 \Windows\WindowStations\Wi
```

```
Registers (MMX)
EAX 00123ECC
ECX 7C910500 ntdll.7C910500
EDX 00150608
EBX 00000000
ESP 00123DA4
EBP 00123F50
ESI 00123F90
EDI 00000060

EIP 00BE5266

C 1  ES 0023 32bit 0(FFFFFFFF)
P 0  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_SUCCESS (00000000)

EFL 00000203 (NO,B,NE,BE,NS,PO,GE,G)

MM0 0105 0104 0000 0000
MM1 0000 0000 0000 0000
MM2 0000 0000 0000 0000
MM3 0000 0000 0000 0000
MM4 0000 0000 0000 0000
MM5 0000 0000 0000 0000
MM6 8000 0000 0000 0000
MM7 8000 0000 0000 0000
```

```
00123DCC 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00123DDC 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00123DEC 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00123DFC 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00123E0C 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00123E1C 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00123E2C 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00123E3C 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
```

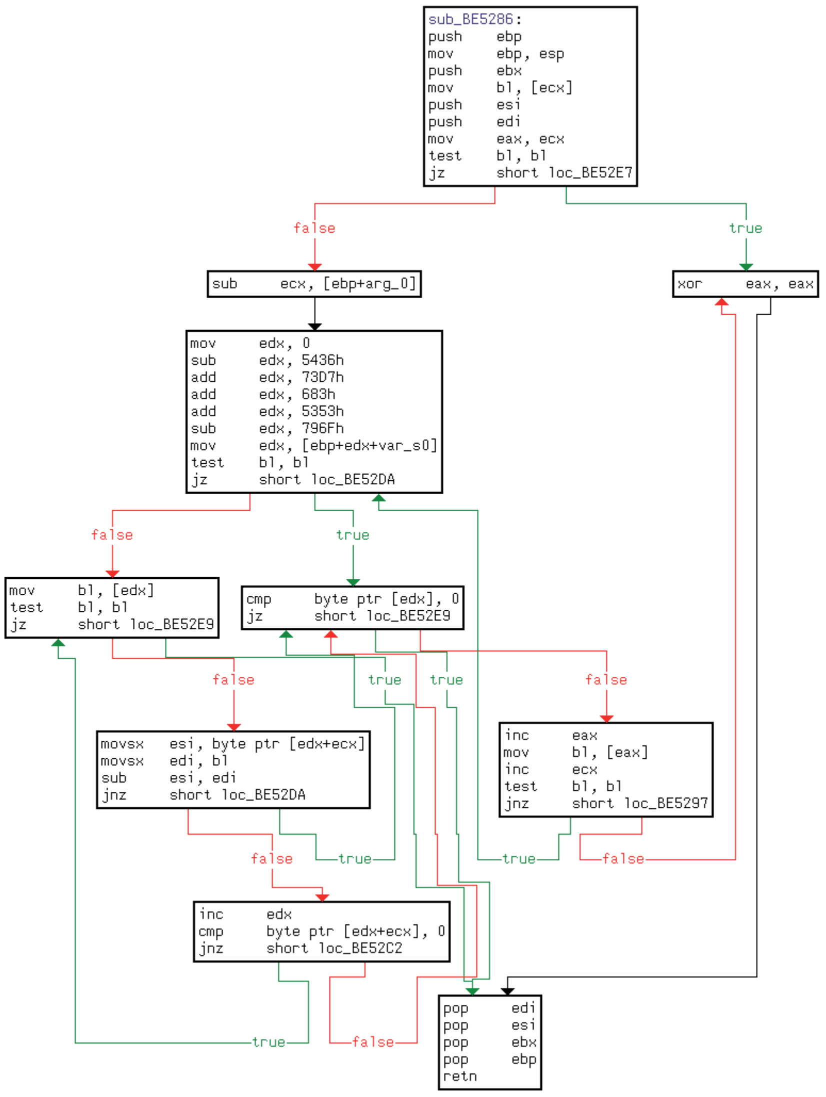
```
00123DA4  00000060  hDevice
00123DA8  002D1400  dwIoControlCode
00123DAC  00123ECC  lpInBuffer
00123DB0  0000000C  nInBufferSize
00123DB4  00123DCC  lpOutBuffer
00123DB8  00000080  nOutBufferSize
00123DBC  00123ED8  lpBytesReturned
00123DC0  00000000  lpOverlapped
```

Figure 2: Sending IO control code.



```
sub_BE5286:
push    ebp
jmp     loc_BE04F3

mov     ebp, esp
push    ebx
mov     bl, [ecx]
push    esi
push    edi
mov     eax, ecx
test    bl, bl
jz      loc_BE1120

sub     ecx, [ebp+8]

mov     edx, 0
sub     edx, 5436h
jmp     loc_BE55EB

add     edx, 73D7h
add     edx, 683h
jmp     loc_BE2C58

add     edx, 5353h
sub     edx, 796Fh
jmp     loc_BE2294

mov     edx, [edx+ebp]
test    bl, bl
jnz     loc_BE2791

jmp     loc_BE3C49

jmp     loc_BE206A

mov     bl, [edx]
test    bl, bl
jmp     loc_BE40D7

cmp     byte ptr [edx], 0
jnz     loc_BE41DE

jnz     loc_BE2050

jmp     loc_BE4C99

inc     eax
mov     bl, [eax]
jmp     loc_BE4EB9

movsx   esi, byte ptr [ecx+edx]
movsx   edi, bl
sub     esi, edi
jnz     loc_BE206A

jmp     loc_BE4C99

jmp     loc_BE094E

inc     ecx
jmp     loc_BE1118

test    bl, bl
jnz     loc_BE0507

inc     edx
cmp     byte ptr [ecx+edx], 0
jnz     loc_BE2791

jmp     loc_BE4C97

xor     eax, eax

pop     edi
pop     esi
pop     ebx
pop     ebp
retn
```

Figure 3: Flow chart of the obfuscated strstr() function.



```
sub_BE5286:
push    ebp
mov     ebp, esp
push    ebx
mov     bl, [ecx]
push    esi
push    edi
mov     eax, ecx
test    bl, bl
jz      short loc_BE52E7
```

```
sub     ecx, [ebp+arg_0]
```

```
xor     eax, eax
```

```
mov     edx, 0
sub     edx, 5436h
add     edx, 73D7h
add     edx, 683h
add     edx, 5353h
sub     edx, 796Fh
mov     edx, [ebp+edx+var_s0]
test    bl, bl
jz      short loc_BE52DA
```

```
mov     bl, [edx]
test    bl, bl
jz      short loc_BE52E9
```

```
cmp     byte ptr [edx], 0
jz      short loc_BE52E9
```

```
movsx   esi, byte ptr [edx+ecx]
movsx   edi, bl
sub     esi, edi
jnz     short loc_BE52DA
```

```
inc     eax
mov     bl, [eax]
inc     ecx
test    bl, bl
jnz     short loc_BE5297
```

```
inc     edx
cmp     byte ptr [edx+ecx], 0
jnz     short loc_BE52C2
```

```
pop     edi
pop     esi
pop     ebx
pop     ebp
retn
```

Figure 4: Flow chart after removing the redundant jump instructions.

PERFORMANCE OF PUBLIC AUTOMATION SYSTEMS

The techniques discussed here make things difficult for both human and machine analysis. I have tested several well known automation systems against one of the packed samples (sample 'A' in the Appendix) and none of them produced any information about its payload (dropping files to the %temp% folder and then running these), as shown in Figures 5–9.

However, with the relentless contribution from malware analysts, we expect to see ongoing improvements in those systems.

Editor: Martijn Grooten

Chief of Operations: John Hawes

Security Test Engineers: Scott James, Tony Oliveira, Adrian Luca, Lewis Jones

Sales Executive: Allison Sketchley

Editorial Assistant: Helen Martin

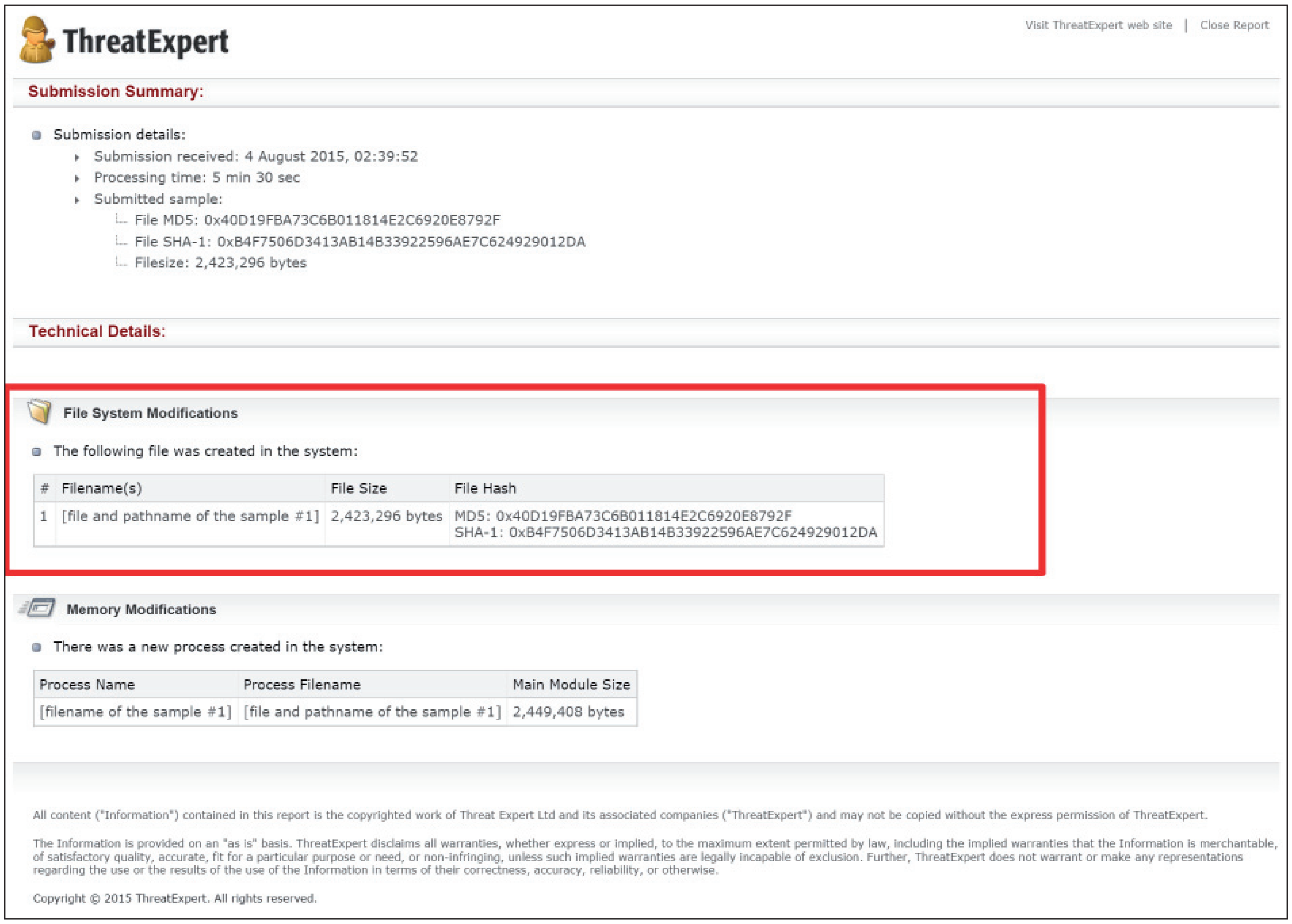
Developer: Lian Sebe

Consultant Technical Editor: Dr Morton Swimmer

© 2015 Virus Bulletin Ltd, The Pentagon, Abingdon Science Park, Abingdon, Oxfordshire OX14 3YP, England

Tel: +44 (0)1235 555139 Fax: +44 (0)1865 543153

Email: editorial@virusbtn.com Web: http://www.virusbtn.com/



ThreatExpert

Visit ThreatExpert web site | Close Report

Submission Summary:

- Submission details:
  - Submission received: 4 August 2015, 02:39:52
  - Processing time: 5 min 30 sec
  - Submitted sample:
    - File MD5: 0x40D19FBA73C6B011814E2C6920E8792F
    - File SHA-1: 0xB4F7506D3413AB14B33922596AE7C624929012DA
    - Filesize: 2,423,296 bytes

Technical Details:

File System Modifications

- The following file was created in the system:

#	Filename(s)	File Size	File Hash
1	[file and pathname of the sample #1]	2,423,296 bytes	MD5: 0x40D19FBA73C6B011814E2C6920E8792F SHA-1: 0xB4F7506D3413AB14B33922596AE7C624929012DA

Memory Modifications

- There was a new process created in the system:

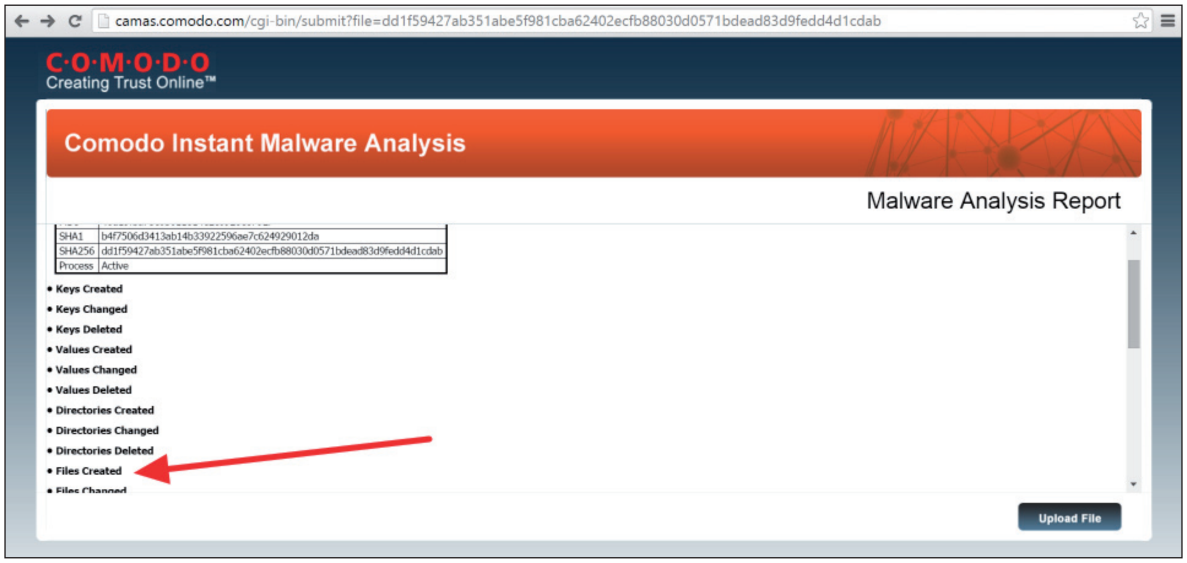
Process Name	Process Filename	Main Module Size
[filename of the sample #1]	[file and pathname of the sample #1]	2,449,408 bytes

All content ("Information") contained in this report is the copyrighted work of Threat Expert Ltd and its associated companies ("ThreatExpert") and may not be copied without the express permission of ThreatExpert.

The Information is provided on an "as is" basis. ThreatExpert disclaims all warranties, whether express or implied, to the maximum extent permitted by law, including the implied warranties that the Information is merchantable, of satisfactory quality, accurate, fit for a particular purpose or need, or non-infringing, unless such implied warranties are legally incapable of exclusion. Further, ThreatExpert does not warrant or make any representations regarding the use or the results of the use of the Information in terms of their correctness, accuracy, reliability, or otherwise.

Copyright © 2015 ThreatExpert. All rights reserved.

Figure 5: ThreatExpert.



camas.comodo.com/cgi-bin/submit?file=dd1f59427ab351abe5f981cba62402ecfb88030d0571bdead83d9fedd4d1cdab

C·O·M·O·D·O
Creating Trust Online™

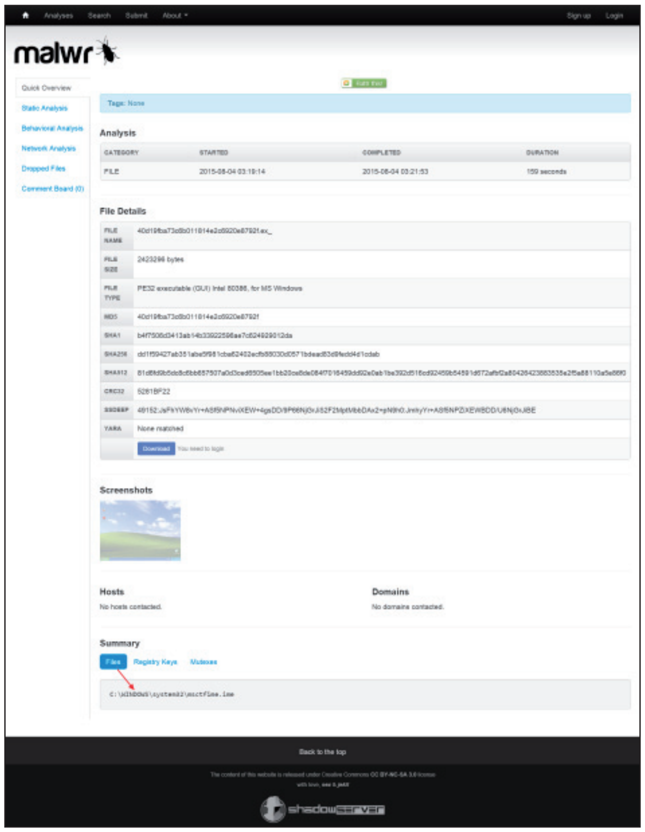
Comodo Instant Malware Analysis

Malware Analysis Report

SHA1	b4f7506d3413ab14b33922596ae7c624929012da
SHA256	dd1f59427ab351abe5f981cba62402ecfb88030d0571bdead83d9fedd4d1cdab
Process	Active

- Keys Created
- Keys Changed
- Keys Deleted
- Values Created
- Values Changed
- Values Deleted
- Directories Created
- Directories Changed
- Directories Deleted
- Files Created
- Files Changed

Figure 6: Comodo Camas.



Analysis

CATEGORY	STARTED	COMPLETED	DURATION
FILE	2015-08-04 03:19:14	2015-08-04 03:21:53	159 seconds

File Details

FILE NAME	40d19fba73c6b011814e2c6920e8792f.ex_
FILE SIZE	2423296 bytes
FILE TYPE	PE32 executable (GUI) Intel 80386, for MS Windows
MD5	40d19fba73c6b011814e2c6920e8792f
SHA1	b4f7506d3413ab14b33922596ae7c624929012da
SHA256	dd1f59427ab351abe5f981cba62402ecfb88030d0571bdead83d9fedd4d1cdab
SHA512	81d8fd9b5dc8c6bb657507a0d3ced6505ee1bb20ce8de084f7016459dd92e0ab1be392d516cd92459b54591d672afbf2a8042842388353Se2f5a88110a5e86f0
CRC32	5261BF22
SSDEEP	49152:JsFkYW6vYr+AS5NPNvXEW+4gsDD/9P66NjGxJi52F2MptMbbDAx2+pN9h0:JmkyYr+AS5NPZXEWBDD/U6NjGxJiBE
YARA	None matched

Screenshots

Hosts

No hosts contacted.

Domains

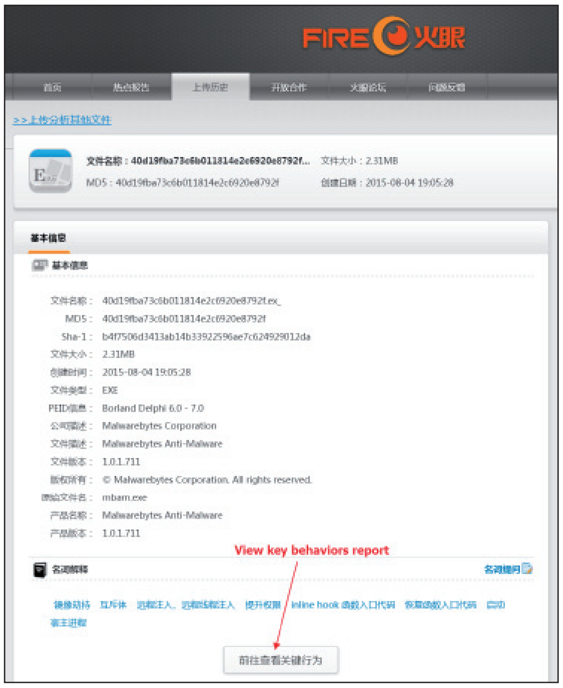
No domains contacted.

Summary

```
C:\WINDOWS\system32\msctfime.ime
```

The content of this website is released under Creative Commons CC BY-NC-SA 3.0 license
with love, see & jekil

Figure 7: Malwr.



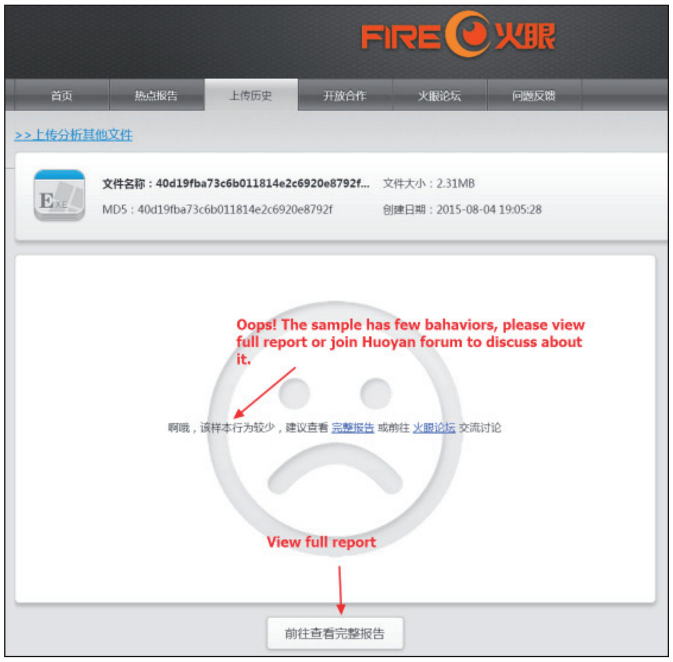
文件名称：40d19fba73c6b011814e2c6920e8792f... 文件大小：2.31MB
MD5：40d19fba73c6b011814e2c6920e8792f 创建日期：2015-08-04 19:05:28

基本信息

文件名称：	40d19fba73c6b011814e2c6920e8792f.ex_
MD5：	40d19fba73c6b011814e2c6920e8792f
Sha-1：	b4f7506d3413ab14b33922596ae7c624929012da
文件大小：	2.31MB
创建时间：	2015-08-04 19:05:28
文件类型：	EXE
PEID信息：	Borland Delphi 6.0 - 7.0
公司描述：	Malwarebytes Corporation
文件描述：	Malwarebytes Anti-Malware
文件版本：	1.0.1.711
版权所有：	© Malwarebytes Corporation. All rights reserved.
原始文件名：	mbam.exe
产品名称：	Malwarebytes Anti-Malware
产品版本：	1.0.1.711

镜像劫持 互斥体 远程注入、远程线程注入 提升权限 inline hook 函数入口代码 恢复函数入口代码 启动宿主进程

Figure 8: Full report from Kingsoft Huoyan (Huoyan’s literal meaning is ‘fire eye’).



文件名称：40d19fba73c6b011814e2c6920e8792f... 文件大小：2.31MB
MD5：40d19fba73c6b011814e2c6920e8792f 创建日期：2015-08-04 19:05:28

啊哦，该样本行为较少，建议查看 完整报告 或前往 火眼论坛 交流讨论

Figure 9: Key behaviours report from Kingsoft Huoyan.

APPENDIX: MD5 OF RELATED SAMPLES (ALL WITH THE IDENTICAL PACKER)

	MD5	**Payload**	**Notes**
A	40D19FBA73C6B011814E2C6920E8792F	Trojan.Dropper drops samples B, C, and D to %temp% folder and executes them.	N/A
B	FBDEC6F2A565E5B6844A7DE2F785EC88	Galaxy Logger V3	Official site: http://galaxysproducts.com
C	BA2A65C19C961A51739E28DF238FB0EA	Backdoor.Trojan	C&C server: degreat248.no-ip.org:9003
D	9C306303F6656435500A6A3C53793758	Instant Password Stealer 1.0	Official site: http://www.nuclearwintercrew.com