18.02.2015,
Author: Paul Rascagnères

# Babar: espionage software finally found and put under the microscope

## G DATA experts analyze malware mentioned in CSEC documents leaked by Snowden

**Almost a year after Operation SNOWGLOBE was publicly mentioned for the first time by the famous French newspaper Le Monde, security experts have now laid hands on malware samples that match the descriptions made by the Communication Security Establishment Canada (CSEC). The following analysis is the first report about the espionage malware dubbed Babar, which the whole computer security community searched for. After the disclosure about EvilBunny [1], Babar is now a second component identified to be related to Operation SNOWGLOBE and is believed to be coded by the same developers. Babar's feature set includes keystroke logging, clipboard logging and, most interesting, the possibility to log audio conversations – the elephant has big ears!**

## Background

The revelation about the existence of yet another potentially nation-state driven spyware occurred in March 2014 when Le Monde first published information about top secret slides originating from 2011 and part of their content . But the slides Le Monde published revealed only a small part of the picture – several slides were cut out, some information was redacted. Germany's Der Spiegel re-published the slide set with far less deletions recently, in January 2015, and therefore gave a deeper insight about what CSEC actually says they have tracked down.

The newly published documents reveal: the so called operation SNOWGLOBE, was discovered in 2009 (slide 9) and consists of three different "implants", two were dubbed snowballs and one "more sophisticated implant, discovered in mid-2010" is tagged as snowman (slide 7). According to slide 22, "CSEC assesses, with moderate certainty, SNOWGLOBE to be a state-sponsored CNO [Cyber Network Operation] effort, put forth by a French intelligence agency." The information given dates back to 2011 and nothing else has been published since. Now that specific Babar samples have been identified and analyzed, there might be new information, also with regards to similarities or differences between the two Remote Administration Tools (RATs) EvilBunny and Babar.

We'd like to express special thanks to Marion Marschalek, Joan Calvet and the CIRCL Luxemburg team for their contributions for this report! We recommend reading Marion's report "Shooting Elephants", a complementary piece of work regarding the Babar malware.

## The samples

### EvilBunny samples (SHA256)

c6a182f410b4cda0665cd792f00177c56338018fbc31bb34e41b72f8195c20cc
7d1e5c4afb1682087d86e793b3fc5a8371dc7c28e27e7196e3b258934f6bafb5
7bfc135194d3e5b85cbe46ed1c6f5e21dbe8f62c0a3ef56245b2d6500fc3a618
be14d781b85125a6074724964622ab05f89f41e6bacbda398bc7709d1d98a2ef

### Babar samples (SHA256; dropper and payload)

c72a055b677cd9e5e2b2dcbba520425d023d906e6ee609b79c643d9034938ebf: dropper
82e6f9c10c7ba737f8c79deae4132b9ff82090ccd220eb3d3739365b5276c3c8: dropper

aa73634ca325022dd6daff2df30484ec9031939044cf4c2a004cbdb66108281d: payload (perf_585.dll)
57437a675cae8e71ac33cd2e001ca7ef1b206b028f3c810e884223a0369d2f8a: payload: (dump21cb.dll)

G DATA's security solutions detect all analyzed samples.

### The malware names: are the coders cartoonists?

Looking at the compilation path stored in the binary, we can identify the internal name of the projects:

C:\Users\user\Desktop**bunny 2.3.2**\Release\Transporter2.pdb
C:\Documents and Settings\admin\Desktop**Babar64****Babar64**\obj\DllWrapper Release\Release.pdb

Furthermore, a command and control server of an EvilBunny sample also mentioned the sample project name:

hxxp://1.9.32.11/**bunny**/test.php?rec=nvista

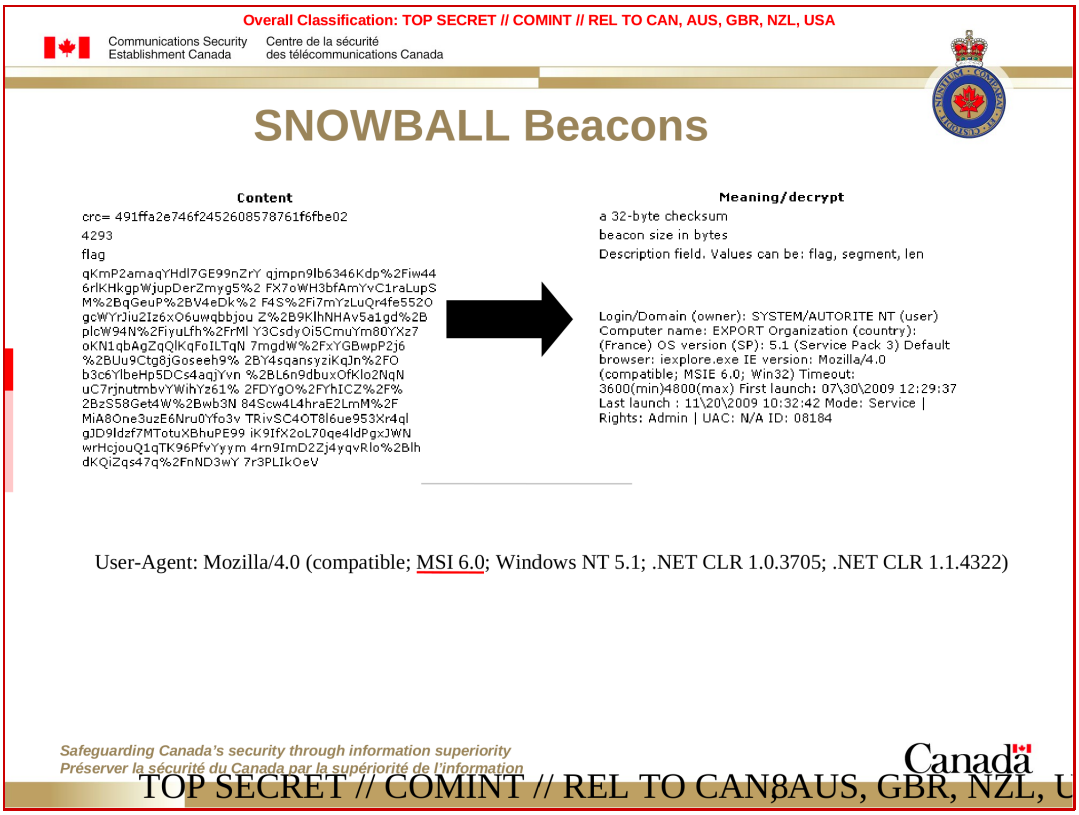
## Identifying the malware described in CSEC slides

The following indicators underline the assumption that the EvilBunny and Babar samples analyzed match the ones described in the leaked Snowden documents, in the order of the slides. Nevertheless, some differences are listed at the end:

### Match: Typographical error – slide 8

CSEC mentioned a typo, committed by the malware authors. In the user agent, instead of using the string MSIE (MicroSoft Internet Explorer), the malware uses the string MSI. The malware does not use the browser to communicate; the request was inserted manually by the developer who made a mistake. We found this exact same mistake in EvilBunny and Babar samples:

```
paul@gdata:~/babar$ strings -a perf_585.dll | grep "MSI "
User-Agent: Mozilla/4.0 (compatible; MSI 6.0; Windows NT 5.1; .NET CLR 1.0.3705; .NET CLR 1.1.4322)
User-Agent: Mozilla/4.0 (compatible; MSI 6.0; Windows NT 5.1; .NET CLR 1.0.3705; .NET CLR 1.1.4322)
```



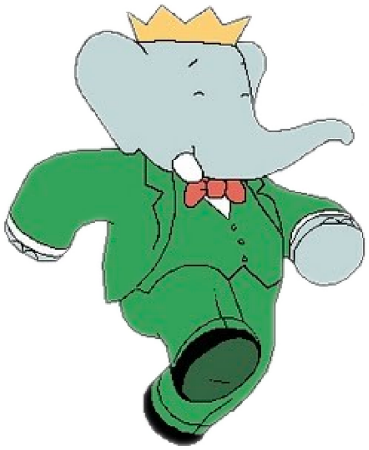

### Match: Internal name: Babar – slide 18

Within the documents leaked by Snowden, the CSEC mentioned the internal name of the malware: Babar. In the malware samples analyzed the internal name is the same, as mentioned before.

Communications Security Establishment Canada | Centre de la sécurité des télécommunications Canada

## Attribution: Binary Artifacts

- ntrass.exe
  - DLL Loader uploaded to a victim as part of tasking seen in collection
  - Internal Name: Babar
  - Developer username: titi
- Babar is a popular French children's television show
- Titi is a French diminutive for Thiery, or a colloquial term for a small person





### Match: Locale option – slide 19

The CSEC mentioned the locale option "**fr_FR**" during the spear-phishing attack. In the EvilBunny samples, during the HTTP queries to the command and control servers the Accept-Language parameter is set to "**fr**".

Overall Classification: TOP SECRET // COMINT // REL TO CAN, AUS, GBR, NZL, USA

Communications Security Establishment Canada | Centre de la sécurité des télécommunications Canada

## Attribution: Language

- ko used instead of kB – a quirk of the French technical community
- English used throughout C2 interface, BUT phrasing and word choice are not typical of a native English speaker
  - An attempt at obfuscation?
- Locale option of artifact within spear-phishing attack set to "fr_FR"

Safeguarding Canada's security through information superiority
Préserver la sécurité du Canada par la supériorité de l'information
Canada
TOP SECRET // COMINT // REL TO CAN, AUS, GBR, NZL, U

### Match: English language – slide 19

Also on the slide 19, the CSEC mentioned that the command and control interface is in English but the choice of

words is not typical for a native English speaker.  We found English mistakes in EvilBunny and Babar samples, such as this example from Babar:

```
!!!EXTRACT ERROR!!!File Does Not Exists-->[%s]
```

### Difference: Infrastructure – slide 10

The CSEC documents reveal that scripts called "**outbase.php**" and "**register.php**" were found on infrastructure domains, "in a directory under root domain". The scripts found in the samples analyzed were named "**index.php**" in a deeper directory.

### Difference: Developer username: titi – slide 18

The developer's username in the Babar samples analyzed is admin instead of titi as mentioned in the Snowden documents. The Bunny samples reveal user as the developer's username.

## Comparing EvilBunny to Babar

We believe that both malware species belong to the mentioned operation SNOWGLOBE and the following chapter will describe similarities and differences:

### Typing error

As mentioned previously, a typo has been found within EvilBunny and Babar, within the user agent string. This mistake can be the result of a copy/paste error or due to the use of the same library inside the two samples.

### Antivirus detection

The first task for both, EvilBunny and Babar, is to list the installed antivirus software. They use the exact same technique to fulfill this task: WMI, the Windows Management Instrumentation.

WMI is an interface provided by Microsoft to get information about and notifications from the system. The users can use WMI by using VBScript, PowerShell or C++ language. To detect the name of the antivirus solution installed and registered, the malware opens one of the following Windows Security Center WMI providers:

- **ROOT\SecurityCenter** (for operating systems before Windows Vista)
- **ROOT\SecurityCenter2** (Windows Vista and newer OS)

The analyzed malware includes the two providers and the two versions of operating system (pre-Vista and post-Vista). Microsoft provides an SQL-like system to perform queries using the WMI. This system is called WMI Query Language (short WQL). The malware performs the following query:

```
SELECT * FROM AntiVirusProduct
```

Here is the description of the antivirus object:

```
class AntiVirusProduct
{
  string  companyName;                  // Vendor name
  string  displayName;                  // Application name
  string  instanceGuid;                 // Unique identifier
  boolean onAccessScanningEnabled;   // Real-time protection
  boolean productUptoDate;              // Definition state
  string  versionNumber;                // Application version
}
```

The malware checks the following entries: **productUpToDate**, **versionNumber** and the**displayName**. The malware checks whether the SHA-256 of the first word of the**displayName** is equal to a predefined list. Looking at several samples, the content of this list varies. Here is one example of a list [Updated on February 19, 2015):

- ab6ed3db3c243254294cfe431a8aeada28e5741dfa3b9c8aeb54291fddc4f8c3 (AhnLab)

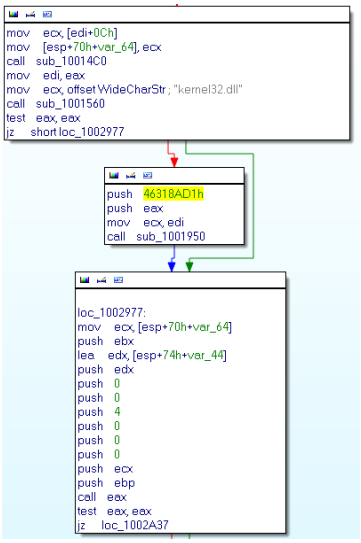
- b3fe0e3a3e3befa152c4237b0f3a96ffaa44a2d7e1aa6d379d3a1ab4659e1676 (AntiVir)
- c0ffcaf63c2ca2974f44138b0956fed657073fde0adeb0b1c940b5c45e8a5cab (avast!)
- 249a90b07ed10bd0cd2bcc9819827267428261fb08e181f43e90807c63c65e80 (AVG)
- 4b650e5c4785025dee7bd65e3c5c527356717d7a1c0bfef5b4ada8ca1e9cbe17 (CA)
- c8e8248940830e9f1dc600c189640e91c40f95caae4f3187fb04427980cdc479 (DoctorWeb)
- 97010f4c9ec0c01b8048dbad5f0c382a9269e22080ccd6f3f1d07e4909fac1a5 (F-PROT)
- aa0ad154f949a518cc2be8a588d5e3523488c20c23b8eb8fafb7d8c34fa87145 (F-Secure)
- 333e0a1e27815d0ceee55c473fe3dc93d56c63e3bee2b3b4aee8eed6d70191a3 (G)
- d4634c9d57c06983e1d2d6dc92e74e6103c132a97f8dc3e7158fa89420647ec3 (InternetSecurity)
- 977781971f7998ff4dbe47f3e1d679f1941b3237d0ba0fdca90178a15aec1f52 (Jiangmin)
- f1761a5e3856dceb3e14d4555af92d3d1ac47604841f69fc72328b53ab45ca56 (Kaspersky)
- a48be88bed64eff941be52590c07045b896bc3e87e7cf62985651bbc8484f945 (McAfee)
- 2bc42b202817bdab7d49506d291e3d9624ae0069087a8949c8fcb583c73772b1 (Norton)
- 0d21bd52022ca7f7e97109d28d327da1e68cc0bedd9713b2dc2b49d3aa104392 (Online)
- 0d21bd52022ca7f7e97109d28d327da1e68cc0bedd9713b2dc2b49d3aa104392 (Online)
- f7d9ea7f3980635237d6ea58048057c33a218f2670e0ff45af5f4f670e9aa6f4 (Panda)
- 522e5549af01c747329d923110c058b7bb7e112816de64bd7919d7b9194fba5b (Rising)
- 4db3801a45802041baa44334303e0498c2640cd5dfd6892545487bf7c8c9219f (ThreatFire)
- 9e217716c4e03eee7a7e44590344d37252b0ae75966a7f8c34531cd7bed1aca7 (Trend)
- e1625a7f2f6947ea8e9328e66562a8b255bc4d5721d427f943002bb2b9fc5645 (VirusBuster)
- 588730213eb6ace35caadcb651217bfbde3f615d94a9cca41a31ee9fa09b186c (ZoneAlarm)
- b39be67ae54b99c5b05fa82a9313606c75bfc8b5c64f29c6037a32bf900926dd ()
- a7f9b61169b52926bb364e557a52c07b34c9fbdcd692f249cd27de5f4169e700 ()
- 1ba035db418ad6acc8e0c173a49d124f3fcc89d0637496954a70e28ec6983ad7 ()

The identified hashes correspond to the strings of well-known commercial antivirus products. The hash (G) stands for G DATA software solutions. The hashes mentioned before the empty brackets have not yet been identified.

## API obfuscation

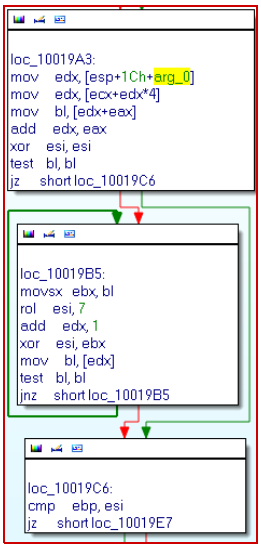
### Both cases:

The two malicious programs both use API obfuscation in order to make the analysis more complicated. The purpose is to execute a Microsoft Windows API without naming it. On our cases, the approach is the same in both malware families: when the malware needs to execute an external function (from a dynamic library), it uses a kind of “hash” instead of using the function name. The “hash” is provided to an internal function, this function establishes the relation between the “hash” and the address of the function. At the end, the address is executed. The only difference between EvilBunny and Babar, when it comes to API obfuscation, is the internal function used to establish the relation. An example below (where the “hash” is 0x46318AD1):




## EvilBunny case:

On EvilBunny samples, the malware realized a kind of Cyclic Redundancy Check (short CRC) of every exported function name of the desired dynamic library. If the "CRC" of a function's name matches the value of the "hash", the malware knows that it is the function to be executed. Here is the "CRC" loop:




This loop can be represented by the following python script:

```
#!/usr/bin/python
CRC = 0
function = "CreateProcessW"
for i in list(function)
  key = rol32(CRC, 7)
  CRC = ord(i)^key
print function+": 0x%08x" % (CRC)
```

Here is the output of the script:

```
CreateProcessW: 0x46318ad1
```

The hexadecimal value is the same as the value in our screenshot, so the executed function will be **CreateProcessW()**. With this script, we can easily create a correlation table to generate the hexadecimal value for each and every function available in the library**kernel32.dll**:

```
paul@gdata:~/$ cat API.py
#!/usr/bin/python
import sys
import pefile

def rol32(num, count):
      num1 = (num << count) & 0xFFFFFFFF
      num2 = (num >> (0x20 - count)) & 0xFFFFFFFF
      return num1 | num2

pe =  pefile.PE(sys.argv[1])
for exp in pe.DIRECTORY_ENTRY_EXPORT.symbols:
  cpt = 0
  for i in list(exp.name):
      key = rol32(cpt,7)
      cpt=ord(i)^key
  print exp.name+": 0x%08x" % (cpt)

paul@gdata:~/babar$ ./API.py kernel32.dll
ActivateActCtx: 0x5147f60f
AddAtomA: 0x1e1865e5
AddAtomW: 0x1e1865f3
AddConsoleAliasA: 0x06dc97e5
AddConsoleAliasW: 0x06dc97f3
AddLocalAlternateComputerNameA: 0xedbafee8
AddLocalAlternateComputerNameW: 0xedbafefe
...
```

## Babar case:

The Babar malware does not perform a kind of “CRC” regarding the function name. The algorithm is more complex. However, the philosophy is the same: for each exported function name, the malware applies an algorithm in order to verify if the calculated “hash” matches the wanted “hash”.

To create the correlation table in this case, our approach was to instrument the debugger using Python. On our samples, the instruction at 0x10040930 (**CMP ECX, [EAX]**) is really interesting because ECX contains the desired “hash”, [EAX] contains the calculated “hash” of the current exported function and finally [EBX] contains the current exported function name. So we can create a short Immunity Debugger Python script to calculate these values for each exported function name and create the table:

```
from immlib import *
from immutils import *

def main(args):
  imm = Debugger()
  imm.setBreakpoint(0x10040930)
  imm.run()
  while True:
     regs=imm.getRegs()
     fct = imm.readString(regs['EBX'])
     value = imm.readMemory(regs['EAX'], 4)[::-1]
     imm.log(fct+":"+value.encode('hex'))
     imm.run()
```

And the output:

```
AcquireSRWLockExclusive:333bab35
AcquireSRWLockShared:567cb604
```

```
ActivateActCtx:4e17a661
AddAtomA:3b9ce8fb
AddAtomW:236e73a4
AddConsoleAliasA:42b5c543
AddConsoleAliasW:e566de2b
AddDllDirectory:94debd22
AddIntegrityLabelToBoundaryDescriptor:b4107a12
AddLocalAlternateComputerNameA:1f6ed911
…
```

If you are interested in the complete correlation tables, please contact us at intelligence@gdata.de

## Babar configuration extraction and analysis

The configuration of the malware is encrypted with the AES algorithm. The key and the offset where the configuration is stored are located at the end of the Babar payloads (the .dll files). Once decrypted, we can identify the following content, which reveals information about command and control servers as well as certain process names and file name extensions the malware will keep an eye on:

Sample: 5da5079754d975d5b04342abf9d60bd0bae181a0:

```
excel.exe, winword.exe, powerpnt.exe, visio.exe, acrord32.exe, notepad.exe, wordpad.exe
txt, rtf, xls, xlsx, ppt, ppts, doc, docx, pdf, vsd
skype.exe, msnmsgr.exe, oovoo.exe, nimbuzz.exe, googletalk.exe, yahoomessenger.exe, x-lite.exe
hxxp://www.alexpetro.com/images/training/courses/bb212/index.php
hxxp://www.etehadyie.ir/images/public/bb212/index.php
```

Sample: efbe18eb8a66e4b6289a5c53f22254f76e3a29db:

```
excel.exe, winword.exe, powerpnt.exe, visio.exe, acrord32.exe, notepad.exe, wordpad.exe
txt, rtf, xls, xlsx, ppt, ppts, doc, docx, pdf, vsd
skype.exe, msnmsgr.exe, oovoo.exe, nimbuzz.exe, googletalk.exe, yahoomessenger.exe, x-lite.exe
hxxp://www.horizons-tourisme.com/_vti_bin/_vti_msc/bb/index.php
hxxp://www.gezelimmi.com/wp-includes/misc/bb/index.php
```

The first line contains document viewer processes, the second line contains media document extensions and the third line contains instant messaging processes. The use of this information will be described below in the chapter Babar’s spy features.

Finally, the last line contains the URLs of the command and control servers. The following information is available at the time of writing this article:

- www.alexpetro.com
  - Website topic: Service company for drilling equipment (oil and gas), located in Egypt
  - Domain registrant's origin: -
  - Website hosted in: Texas, USA
- www.etehadyie.ir (not available during investigation)
  - Website topic: Home appliances (according to Google translator)
  - Domain registrant's origin: Tehran, Iran
  - Website hosted in: -
- www.horizons-tourisme.com
  - Website topic: travel agency, located in Algeria
  - Domain registrant's origin: Algiers, Algeria
  - Website hosted in: Ohio, USA
- www.gezelimmi.com (not available during investigation)
  - Website topic: Turkish website to promote tourism in Turkey

- Domain registrant's origin: Merkez, Turkey
- Website hosted in: New York, USA

We do not know whether the command and controls were compromised legitimate websites, during the campaign, or servers dedicated to the attacks. Slide 23 mentions that C&C nodes were found “worldwide (including Canada, US, UK)”.

## Babar’s espionage features

The RAT has common features such as code execution, code injection into running processes, file stealing (the extensions listed in the configuration file come into play at this pint). However, Babar has additional features such as being a key logger in order to record key strokes and it also has the possibility to steal the clipboard content (frequently used to store passwords in case the user uses password storage application such as **KeePass**). The data is stored in the file **%COMMON_APPDATA%\MSI\update.msi**. Here are two screenshots of the key logger API:

```
keyboard_API proc near

var_10= dword ptr -10h
var_4= dword ptr -4

push    4
mov     eax, offset sub_10065A71
call    __EH_prolog3
mov     esi, ecx
mov     [ebp+var_10], esi
push    offset aUser32_dll_0 ; "user32.dll"
call    loadlibrary
and     [ebp+var_4], 0
push    0A843095h       ; GetKeyboardLayout
mov     ecx, esi
mov     dword ptr [esi], offset off_1006B2CC
call    API_obfuscation
push    5566A9EAh       ; GetKeyboardState
mov     ecx, esi
mov     [esi+0Ch], eax
call    API_obfuscation
push    6575CE06h       ; GetForegroundWindow
mov     ecx, esi
mov     [esi+10h], eax
call    API_obfuscation
mov     [esi+14h], eax
mov     eax, esi
call    __EH_epilog3
retn
keyboard_API endp
```

```
keyboard_API3 proc near

var_10= dword ptr -10h
var_4= dword ptr -4

push    4
mov     eax, offset sub_10065A71
call    __EH_prolog3
mov     esi, ecx
mov     [ebp+var_10], esi
push    offset aUser32_dll_0 ; "user32.dll"
call    loadlibrary
and     [ebp+var_4], 0
push    0DADEE24Ah      ; RegisterRawInputDevices
mov     ecx, esi
mov     dword ptr [esi], offset off_1006B2CC
call    API_obfuscation
push    21BC600Ah       ; GetRawInputData
mov     ecx, esi
mov     [esi+0Ch], eax
call    API_obfuscation
mov     [esi+10h], eax
mov     eax, esi
call    __EH_epilog3
retn
keyboard_API3 endp
```

And the following is a snippet of the clipboard stealer API:

```
clipboard_API proc near

var_10= dword ptr -10h
var_4= dword ptr -4

push    4
mov     eax, offset sub_10065A71
call    __EH_prolog3
mov     esi, ecx
mov     [ebp+var_10], esi
push    offset aUser32_dll_0 ; "user32.dll"
call    loadlibrary
and     [ebp+var_4], 0
push    0C1CC96EAh      ; IsClipboardFormatAvailable
mov     ecx, esi
mov     dword ptr [esi], offset off_1006B2CC
call    API_obfuscation
push    23FA1765h       ; OpenClipboard
mov     ecx, esi
mov     [esi+0Ch], eax
call    API_obfuscation
push    0FA1318DCh      ; GetClipboardData
mov     ecx, esi
mov     [esi+10h], eax
call    API_obfuscation
push    424878DEh       ; CloseClipboard
mov     ecx, esi
mov     [esi+14h], eax
call    API_obfuscation
mov     [esi+18h], eax
```

Babar is also able to take screenshots of the infected desktop (thanks to the **GdiPlus** API). Here is a snippet of the **GdiPlus** API:

```
call    __EH_prolog3
mov     esi, ecx
mov     [ebp+var_10], esi
push    offset aGdiplus_dll ; "gdiplus.dll"
call    loadlibrary
and     [ebp+var_4], 0
push    0A7359146h      ; GdiplusStartup
mov     ecx, esi
mov     dword ptr [esi], offset off_1006B2CC
call    API_obfuscation
push    6C423770h       ; GdiplusShutdown
mov     ecx, esi
mov     [esi+0Ch], eax
call    API_obfuscation
push    569734A3h       ; GdiplusCreateBitmapFromHBITMAP
mov     ecx, esi
mov     [esi+10h], eax
call    API_obfuscation
push    86344474h       ; GdipDisposeImage
mov     ecx, esi
mov     [esi+14h], eax
call    API_obfuscation
push    27832D82h       ; GdipSaveImageToStream
mov     ecx, esi
mov     [esi+18h], eax
call    API_obfuscation
mov     [esi+1Ch], eax
mov     eax, esi
call    __EH_epilog3
retn
screenshot_api endp
```

And finally, as every elephant, Babar has big ears and the malware is able to listen to conversations and log them by using the **dsound** and **winmm** libraries. We assume that the process list of the instant messaging services, seen in the configuration, is used to identify when the malware should enable this feature. The following screenshot shows the use of the**wave*** API to record the audio flow:

```
call    __EH_prolog3
mov     esi, ecx
mov     [ebp+var_10], esi
push    offset aWinmm_dll ; "winmm.dll"
call    loadlibrary
and     [ebp+var_4], 0
push    860EC87Fh       ; waveInGetNumDevs
mov     ecx, esi
mov     dword ptr [esi], offset off_1006B2CC
call    API_obfuscation
push    84BBAE3Ch       ; waveInGetDevCapsA
mov     ecx, esi
mov     [esi+0Ch], eax
call    API_obfuscation
push    201CA672h       ; waveInOpen
mov     ecx, esi
mov     [esi+10h], eax
call    API_obfuscation
push    0C4C74E68h      ; waveInClose
mov     ecx, esi
mov     [esi+14h], eax
call    API_obfuscation
push    0A1F3183Fh      ; waveInStart
mov     ecx, esi
mov     [esi+18h], eax
call    API_obfuscation
push    3CFFB6D8h       ; waveInReset
mov     ecx, esi
mov     [esi+1Ch], eax
call    API_obfuscation
push    2F582023h       ; waveInStop
mov     ecx, esi
mov     [esi+20h], eax
call    API_obfuscation
push    7C159B03h       ; waveInPrepareHeader
```

Looking at the feature list, we can identify that this malware is meant to be a pure espionage tool. It is, regarding the current information, not harming the computer system itself but represents an elaborate instrument to function as wiretap and to exfiltrated data from computers infected. This leads to the assumption that the number of infected machines is rather small and chosen.

## Conclusion

After having more information about the malware attributed to operation SNOWGLOBE, taken from the re-

published slides, the G DATA experts are sure to have found samples which match the descriptions. EvilBunny and Babar might correspond to two of the three "implants" mentioned as Snowballs and Snowman.

The G DATA SecurityLabs are convinced that the number of similarities identified between EvilBunny and Babar show that both malware families originate from the same developers. The evil cartoon malware families share part of their code. The analyses suggest that the samples identified are newer versions of the malware CSEC described in the slides. This may be one reason for the absence of certain indications CSEC has mentioned.

Nevertheless, unfortunately, the experts cannot contribute further information with regards to the malware's origin nor the list of victims. The information CSEC provided was partly supported by indications found in the code, but no clue has been identified. The assertion of a "French intelligence community" being responsible remains unchanged.  Attributing malware to any origin, especially when dealing with specialized and professional malware, has always been difficult.

With a possible nation-state background, this espionage software would not be spread as mass malware but activated against specific and chosen targets only. The main functions of this malware are data exfiltration and wiretapping.

Even if many questions still remain unanswered, the analyses present mark an important step towards the validation of the slides leaked.

---

[1] 0x1338.blogspot.co.at/2014/11/hunting-bunnies.html