THE ELEPHANT IN THE ROOM

Marion Marschalek
Cyphort Inc., Austria

Email marion@cyphort.com

ABSTRACT

At the beginning of 2015, the first espionage software designed for large-scale operations with allegedly European roots was reported. Researchers uncovered malware dubbed 'Babar' by its creators and determined that it fitted the profile of an espionage campaign initially documented by the Communications Security Establishment Canada (CSEC). *PowerPoint* slides created by CSEC and leaked through whistleblower Edward Snowden link Babar to an operation named 'SNOWGLOBE' and attribute it 'with moderate certainty' to French intelligence.

This paper puts the spotlight on Babar and other malware families related to SNOWGLOBE, including NBOT, Bunny and Casper. The technical finesse of the espionage toolkit will be examined, outlining implementation details and peculiarities. In the second part the paper will discuss possibilities and impossibilities of attribution, explaining the inter-family relations between the different binaries as well as analysing the link to the CSEC documents and the credibility of attribution conducted by CSEC.

Finally, the examination will close with a proposal for future research on what more is to uncover of the SNOWGLOBE operation.

INTRODUCTION

In the past year more and more cases of nation-state espionage involving malware have surfaced and have been discussed publicly. The United States, United Kingdom, Russia and China, among others, are known to operate a cyber espionage apparatus in order to aid military operations. At the beginning of 2015, researchers uncovered a collective of malware families which is potentially one of the rare approaches to malware-aided espionage conducted by a European nation-state actor outside of the 'Five Eyes' alliance. The uncovered malware fits the profile of an operation dubbed 'SNOWGLOBE', first mentioned in a leaked presentation of the Communications Security Establishment Canada (CSEC).

The slides describe a CNO (covert network operation) involving different malware strains which were used to target entities of political interest in Canada, Norway and Iran among other nations. Of special interest to SNOWGLOBE seems to have been Iran, where a number of universities along with the Atomic Energy Organization were among the targets. Furthermore, CSEC documents malware with the internal project name 'Babar', which happens to be a French cartoon character. CSEC assesses 'with moderate certainty' that operation SNOWGLOBE is likely to be conducted by a French intelligence agency.

The mystery around Babar was first brought to the public by French newspaper *Le Monde*, which published an article referencing the CSEC slides in March 2014 [1]. Almost a year later, researchers finally identified what comes close to the Babar malware mentioned by CSEC. Babar is an espionage tool, the most sophisticated one among a menagerie of related families. Other SNOWGLOBE tools are NBOT, Bunny and Casper, all of which were uncovered along with Babar. The SNOWGLOBE malware has previously been analysed by security company *Kaspersky Lab*, which dubbed it 'Animal Farm' [2]. Next to the malware discussed in this paper, *Kaspersky* reports two more 'cartoonesque' miscreants, namely Tafacalou and Dino.

SNOWGLOBE'S MALWARE FAMILIES

The first family examined was NBOT, a simple denial of service bot which was active from around 2010 to 2012. The malware includes modules to perform flooding of a specified URL with packages of different protocols, such as plain TCP, HTTP GET or POST with different configurations. Besides that, not much functionality can be found in NBOT binaries.

What makes NBOT interesting is the way it is implemented, as it does not give the impression of regular crimeware. The samples are well designed, come with a handful of clear text strings and have no binary protection whatsoever. On deeper inspection it becomes clear that this family really sticks out from the norm, and on following the breadcrumbs one eventually stumbles over related samples with very similar traits.

The second identified family was Bunny, a multi-threaded bot with an integrated scripting engine [3]. The family name is derived from a project name embedded in the malware dropper. Bunny incorporates a Lua interpreter and downloads and executes Lua scripts to reach a certain level of polymorphism. The Lua scripts can call back into the C++ code of the malware and alter its behaviour at runtime. Bunny was seen being spread in a spear-phishing campaign in December 2011, in which a PDF document exploiting CVE-2011-4369 was used to install the malware [4].

Also among the stash of related binaries we discovered a DLL, itself part of a fairly sophisticated piece of espionage malware. Further investigation led to the dropper of said DLL, which came with the internal project name Babar64 [5]. Babar is a French cartoon character (an elephant), as well as the name of a malware family mentioned in a classified slide deck from CSEC [1]. The slides were leaked by Edward Snowden and first discussed in an article published by French newspaper *Le Monde* [6] in April 2014. The CSEC slides mention SNOWGLOBE as the internal name of the campaign involving Babar and related malware and attribute it 'with moderate certainty' [1, p.22] to an unspecified French intelligence agency.

From an analyst's perspective, Babar is somewhat more complex than the other families. It comes with solid espionage capabilities, it performs keylogging and invades *Windows* processes to steal data from instant messengers, softphones, browsers and office applications. The Babar implant comes with a userland rootkit component in order to hook APIs of interest in dedicated remote processes and steal data on the fly.

While Babar could be called the crown jewel within the espionage toolset, it is not the latest creation. CSEC mentioned Babar for the first time in 2009, the analysed binaries were probably compiled at a later point in time, but probably not later than 2012. It is assumed that the Babar binaries at hand are a later version of the malware described by CSEC. Researchers uncovered one of the newer malware families of

the same group in March 2015 [7]. Casper (as in Casper the Friendly Ghost) matches with TFC, Bunny and Babar on a binary level and doubtlessly comes from the same authors. It is so-called reconnaissance malware, used to collect data from an infected machine in order to identify the owner and/or machine-specific settings.

Casper is known to have been spread through a watering hole attack. The attackers had taken control of the server hosting the website of the Syrian Ministry of Justice in April 2014 to deploy their malicious infrastructure. Two Flash zero-day exploits were involved in spreading Casper to potential targets [8].

All uncovered families can be linked with each other, and several binary attributes match with traits described in the CSEC documents.

THE ACTORS IN THE SPOTLIGHT

NBOT

NBOT is a family of denial-of-service bots which, according to compilation time stamps, was created in 2010. The binaries are well structured and multi-threaded, written in C++ and not heavily protected as one would expect non-targeted malware to be. The bots connect to a C&C server and exchange data in clear text via HTTP. The C&C domains are hard coded in the binaries. Currently, two different domains are known, http://callientefever.info/ and http://fullapple.net/, both of which were registered in 2009 and active until 2010. From 2010 onwards, the domains were sinkholed by security vendor *Kaspersky Lab*.

As a means of stealth, the bots create an svchost.exe process and inject a remote thread to execute their binary payload in the context of svchost.exe. NBOT also implements a mechanism to dynamically load APIs at runtime, identified by hashes of API names. The hash function is simple, using left-sided rotation and XOR with the key AB34CD77h to calculate the hashes:

```
for (i=0; i<expcount; i++) {
   elen = strlen(exports[i]);
   result = 0;
   otherresult = 0xAB34CD77;
      for (j=0; j<elen; j++) {
         result = (rotl(result,7) ^ exports[i][j]);
   }
   otherresult ^= result;
   otherresult ^= 0xAB34CD77;
   printf("%8x\t\t%s\n", otherresult, exports[i]);
   }
```

The bots come with a custom configuration embedded as strings in the binaries, which will be stored as a linked list in memory during start up. A large portion of the configuration attributes' names start with the string 'NBOT_' (see Table 1). It is notable that two configuration terms start with 'TFC_', an abbreviation for 'Tafacalou', according to another piece of malware used by the same actors [2]. Tafacalou is reconnaissance malware, a platform used in the first stage of an attack, dedicated to installing high-profile malware on selected target machines. It is assumed that NBOT inherited these values from the TFC code base.

The binaries are written in C++, and for every action the bots can take they generate an object of an 'action class' at start time. NBOT shows extensive flooding capabilities, as well as functionality to generate statistics on performance and a means for self-update. The dedicated actions can be of the kinds shown in Table 2.

ATCLEAR	TCPFLOOD	SET
PING	WEBFLOOD	UPLOAD
EXEC	POSTFLOOD	UPDATE
HTTPF	STATISTICS	PLUGIN
ASPFLOOD	KILL	

Table 2: Dedicated actions.

NBOT does not show capabilities of espionage or data exfiltration, and no system reconnaissance is performed. It is not perfectly clear what the intentions of the attackers were. Building a botnet out of unprotected binaries implies it will be a short-lived one. Also, denial-of-service attacks are not known to meet the common interests of nation-state attackers. An interesting side note to NBOT is that the binaries come with the Accept-Language of all HTTP requests set to 'fr' (French).

Bunny

The Bunny malware family got its name from a link to debug information embedded in the dropper malware. The path to the associated .pdb file contains the respective project name 'bunny 2.3.2'. The malware is multi-threaded with an integrated Lua engine, making it a scriptable bot which can change its behaviour to a certain extent. Bunny shows a number of interesting anti-analysis features, most of which seem intended for evasion of anti-virus engine emulators and sandboxes. The following features were found:

- An emulator check is made by searching the module file name for strings such as 'TESTAPP' (known to be used by the *Bitdefender* engine), 'klavme', 'myapp' and 'afyjevmv.exe' (assumed to be used by *Kaspersky*).
- The module path name must be more than five characters long and contain either 'msapps\' or 'Perf Manager\',

NBOT_VER	NBOT_PORT	NBOT_BOOL_SEND_DATA
NBOT_ACTION	NBOT_MAX_REQ	NBOT_DATA
NBOT_URL	NBOT_MAX_DATA	NBOT_COMMAND_LINE
NBOT_FORCE_ACTION	NBOT_LAST_IMAGE	TFC_Config_Key
NBOT_TIME	NBOT_UUID	TFC_Config_Name
NBOT_TIMEOUT	NBOT_STATS_END	NotifUrl
NBOT_HOST	NBOT_STATS_URL	SyncUrl

Table 1: Configuration attribute names.

which is the directory into which the Bunny dropper drops its payload.

- The creation timestamp of the dropped payload is changed to the creation timestamp of the system's explorer.exe to hinder forensic analysis.
- Using the EnumProcesses API, Bunny checks whether fewer than 15 processes are running on the system. If that is the case, execution is aborted.
- Bunny performs hook detection on time retrieval APIs, namely NtQuerySystemTime, GetSystemTimeAsFileTime and GetTickCount. Every API is called twice to calculate a delta, while performing a sleep(1000) operation between iteration one and iteration two. The final condition is that, if any of the three deltas is below 998 milliseconds, execution will abort. This can only be the case if any of the three APIs' return values is modified by a system monitoring solution, like a sandbox.
- A subset of API names is obfuscated using the same hashing algorithm and key as NBOT.
- Installed anti-virus products are enumerated by querying the Windows Management Interface (WMI) and adapting the system infiltration method, for example, either injecting the malicious payload into an existing svchost.exe process or creating a new process into which to inject.
- The dropped implant is not started by the dropper, merely a registry key for loading at boot time is created. This is effective in tricking sandboxes, as a reboot is required to invoke the implant. This is curious though, as the fact that deletion of the dropper is the implant's task means that the dropper will remain on the system until reboot as well.

The Bunny implant is compiled with Visual Studio compiler, with performance optimization options set. It is not clear whether this was intended for purposes of obfuscation or if the authors did indeed aim for performance-optimized binaries. Clearly, though, a tremendous amount of inlined code and repeated constants make analysis of the binaries significantly more difficult than usual.

Remote servers and configuration

At initialization Bunny decrypts an XML-format configuration file stored in its resource section, revealing three URLs among timeout settings and encryption keys:

- http://le-progres.net/images/php/test.php?rec=11206-01
- http://ghatreh.com/skins/php/test.php?rec=11206-01
- http://www.usthb-dz.org/includes/php/test.php?rec=11206-01

All three of these URLs served as C&C contacts, sending commands or Lua scripts to the infected host. It is interesting that two of the domains are actually fake domains resembling legitimate websites (le-progres.net and usthb-dz.org), while one is a legitimate domain (ghatreh.com).

Multi-threading model and operation mode

The Bunny implant comes with a solid multi-threading model. The malware runs a main thread, which manages four worker threads and performs C&C command parsing and Lua script execution. The worker threads are dedicated to receiving commands and scripts. Each worker has a dedicated method for receiving instructions, which is either separately via HTTP from the server, aggregated through a downloaded data file or as tasks to be configured as scheduled tasks. Next to that, the main thread also runs sub threads to maintain log files created by the malware during execution and to keep track of the overall system load created by the malware.

The threads are coordinated via named events, global flag variables, and in some cases mutexes or semaphores are also used. The main action of the malware is carried out in the main thread, which parses commands and executes Lua scripts provided by the worker threads via command files.

Commands & Lua integration

The bot supports a total of 20 commands (see Table 3), which are received from the C&C as encrypted data but contained in the binaries in clear text.

mainfrequency	restarthearer	crontaskr
getconfig	Restart	crontaskl
ftpput	cleanhearer	Maxpostdata
ftpget	timeout	seturl
sendfile	Waitfor	Stop
getfile	updatedietime	setcpulimit
uninstall	crontaska	

Table 3: Commands supported by the bot.

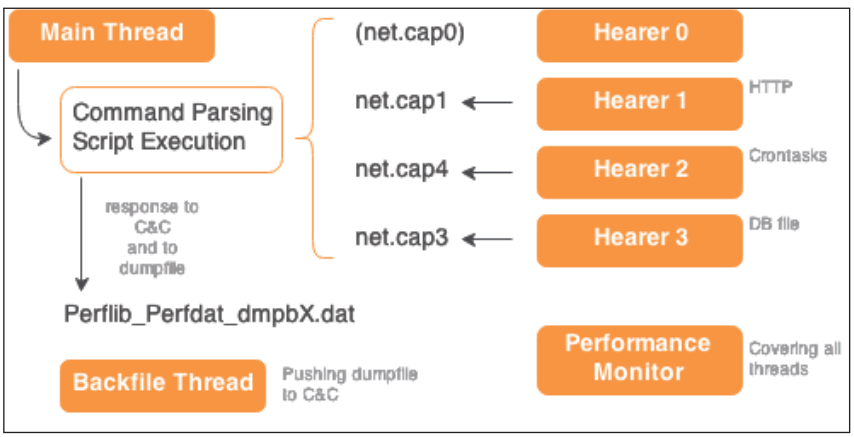

Figure 1: Bunny's mode of operation.

The most notable actions of the bot can be summarized as follows:

- Downloads and executes Lua scripts to instrument its own code.
- Can install managed tasks (named 'crontask') for its integrated engine.
- Can maintain FTP connections.
- Can send and receive files via HTTP.
- Writes runtime information to local files.
- Provides encryption for local data and network communication.

Lua execution is achieved by an integrated Lua 5.1 interpreter. C/C++ bindings provided through the C/Invoke library enable the Lua scripts to call back into the Lua interpreter and instrument the engine. The Lua interpreter is very small (roughly 180KB compiled), thus it can easily be integrated into an application. The C/Invoke bindings enable Lua to be completely independent from the C/C++ application, so injected scripts can be pure Lua code.

The Lua interpreter is a powerful code base which enables Bunny to change functionality on the fly, as different scripts are downloaded and executed. The scripts define the functionality as they perform callbacks to the C/C++ code in the malware binary. The potential of the Lua interpreter can be investigated on the project homepage [9].

Babar

Babar is a fully fledged piece of espionage software, able to invade running processes on the machine, hook API calls to steal sensitive data on the fly, log keystrokes and exfiltrate the stolen information to its remote server. The dropper binary contains a link to debug information, giving away the internal project name 'Babar64'.

Babar, like all other related binaries, is not packed or protected by a crypter. For C&C communication and for data handling the malware uses 128-bit AES encryption, while the keys are hard coded in the binaries. Interestingly, Babar uses the same technique as Bunny for obfuscation of a subset of APIs, which are loaded dynamically at runtime. A different hashing algorithm is used though: the authors have adapted the SHA-1 algorithm to generate 32-bit hashes instead of the usual 160-bit hashes.

As seen before in Bunny, enumeration of installed security solutions is performed by querying WMI. The SHA-256 hashes of the names of installed products are compared against a hard-coded set of hashes.

System infiltration

The Babar dropper drops an implant to the application data folder of the current user and spawns a regsvr32.exe process to load the implant. The implant will inject itself into a randomly chosen desktop process, and from there propagate further to a maximum of two more victim processes as a means of persistence. Babar operates from a main instance which will be the first invaded process, and keeps two backup instances. Should the carrier process of the main instance be terminated, one of the child instances will take over as the main one.

Code injection is performed in the classical file infector way, by mapping a shared object to the memory of the victim process and invoking a function stub as remote thread. The function stub loads the Babar DLL and calls one of its exports, with the export name communicated via shared memory. This way the Babar DLL can propagate silently among remote processes.

Regsvr32.exe, along with the according parameters to load the Babar implant, will be invoked through a registry key at startup. It is curious, though, that the loading process remains in memory and is never terminated.

Functionality

The spying activities are performed either locally through the Babar instance or via a global *Windows* hook invading all processes running on the same desktop. Instance-local capabilities are basic, spying on window names or snooping on the clipboard data, while the global hooks manage to steal information directly from *Windows* API calls.

A summary of the capabilities is as follows:

- Logging keystrokes
- Taking screenshots
- Capture of audio streams from softphone applications

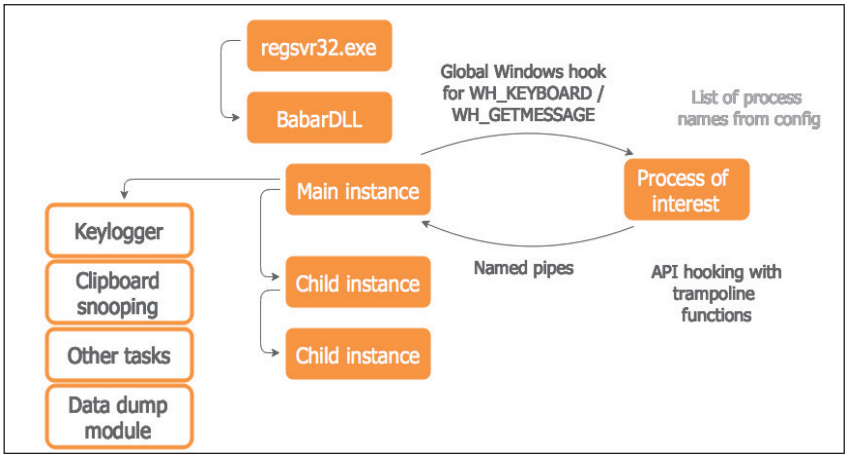

Figure 2: Babar's mode of operation.

- Stealing of clipboard data
- Capture of system and user default language, keyboard layout
- Capture of names of desktop windows.

The rootkit component

The Babar implant applies a global *Windows* hook to load its DLL into the process space of other processes. This effectively means that code provided by the hooking DLL gets executed whenever an arbitrary desktop process receives an event of a type specified by the hook.

Babar installs hooks for types 2 and 3, which are WH_KEYBOARD and WH_GETMESSAGE. This way Babar has control over all keyboard and message events received by any application on the same *Windows* desktop.

Once in the context of a desired target process, the malware goes on to hook specific APIs of interest. This is achieved by applying the detours technique, which implements trampoline functions to be invoked every time a hooked API is called [10]. To achieve this, Babar rewrites the in-memory code for target APIs. A call to a hooked API then results in the calling application invoking a trampoline function, which performs the malicious activity and then passes control on to the legitimate API.

Babar supports trampoline functions for APIs involved in Internet communication, file creation and sound processing.

WSARecv	DirectSoundCaptureCreate	waveOutClose
Send	DirectSoundCreate8	waveOutWrite
Closesocket	DirectSoundCaptureCreate8	waveInOpen
CreateFileW	CoCreateInstance	waveInClose
DirectSoundCreate	waveOutOpen	waveInAddBuffer

Table 4: Supported functions.

Babar's hooking attempts are limited to a list of processes and document extensions, defined in the malware configuration:

- Internet communication: iexplore.exe, firefox.exe, opera.exe, chrome.exe, Safari.exe, msnmsgr.exe
- File creation: excel.exe, winword.exe, powerpnt.exe, visio.exe, acrord32.exe, notepad.exe, wordpad.exe.txt, .rtf, .xls, .xlsx, .ppt, .pptx, .doc, .docx, .pdf, .vsd
- Media: skype.exe, msnmsgr.exe, oovoo.exe, nimbuzz.exe, googletalk.exe, yahoomessenger.exe, x-lite.exe.

Calling home

The Internet communication module of Babar is located in a separate export, which will be invoked through remote thread injection at runtime. The analysed sample of Babar comes with two hard-coded C&C server domains:

- http://www.horizons-tourisme.com/_vti_bin/_vti_msc/bb/index.php
- http://www.gezelimmi.com/wp-includes/misc/bb/index.php

At the time of analysis the server pointed to by horizons-tourisme.com was still hosting bits of C&C infrastructure used by Babar. With directory traversal activated, researchers from *ESET* were able to pull a minimalistic directory structure, showing directories named as follows:

- bb28
- d13
- tfc422

Clearly, the directory belonging to Babar is 'bb28'. Meanwhile, 'tfc422' matches with strings found in the NBOT malware, i.e. 'TFC_'. The purpose of the 'd13' directory remains unknown, although it is assumed that it serves for requests of a third malware family named 'Dino'. The only script inside the bb28 directory is a .php script named config.inc, which contains variables that look familiar from Babar's configuration, such as 'user', 'id' or 'seq' (see Figure 3).

Casper

Thorough investigation led to the discovery of a handful of other related binaries, among which was an implant which obviously carries the signature of the infamous cartoonists. After Bunny and Babar, a third family with a cartoon character name caught our attention, named Casper, as in Casper, the Friendly Ghost.

The analysed binaries share several parts of source code with the other cartoon families and also show some techniques we

```php
<?php
  $uninstall = false; //true to uninstall
  $buninstall_id = false; //true to uninstall from ID
  $uninstall_id = "0C124D55"; //ID to uninstall in hex
  //$uninstall = true;
  $debug = false; //true to see errors messages FOR TEST ONLY!!
  $writelogs = false; //true to create logs file in logs directory
  $version = 3;
  $get_varname_user = "user";
  $get_varname_id  = "id";
  $get_varname_seq = "seq";
  $storefile = "storage/file";
  $ext = ".kv";
  $maxsize = 1024*1024*2; //0 = unlimited size if we choose predifined files
  $UOK_V3 = 0x7345d346;

  function disableCache() {
    $expires = 0;
```

Figure 3: Excerpt from PHP script found on Babar's C&C.

have already seen before. The list of features that some or all of the families have in common is as follows:

- Proxy bypass code
- Enumeration of installed anti-virus solutions through WMI
- Embedded and encrypted configuration in XML format
- Partial API name hashing, Casper sharing the algorithm with NBOT and Bunny
- Payload deployment by remote thread injection through mapping of section objects
- Use of unhandled exception filters, calling ExitProcess in the case of an exception.

An interesting twist to Casper is that its binaries have been seen before, spread in a watering hole attack in Syria in April 2014. Researchers from *Kaspersky* have reported on the watering hole and two zero-day exploits that were involved [8], and researchers from *ESET* provided the necessary link between Casper and the watering hole [7]. The attack had been launched from http://jpic.gov.sy/, a website operated by the Syrian Ministry of Justice.

Plainly spoken, Casper is reconnaissance malware aiming to gather sensitive information about the target system and loading second-stage malware should the target be of interest. It is a typical approach in targeted attacks to operate in several stages while using dedicated types of malware for different tasks.

In fact, the analysed binaries have a very similar mode of operation to related families. The malware dropper will place an infector binary named 'aiomgr.exe' into a directory named 'INTEL Audio Interface Device Manager', which is located in the system's common program files folder. The naming convention, which resembles that of *Windows* services and applications, has been seen before in related families, with artefact names such as 'netmgr', 'ntrass', 'IPSec' and 'MSSecurity'. Once started, the Casper infector spawns a svchost.exe process and injects its malicious payload. This is achieved by mapping a shared memory object to the remote process, which is then invoked as a remote thread. Casper then goes on to collect information about the target system, including the following:

- Operating system version and system architecture
- Default web browser
- Country and organization info from the system settings
- Running processes
- Applications registered for auto-run
- Installed applications.

Casper's C&C server is the same as the watering hole server, http://jpic.gov.sy/. This is a rather uncommon practice, but given that acquisition of compromised web servers in a region like Syria is considerably difficult, the decision of the attackers seems plausible.

Casper, like Bunny and Babar, comes with the 'AV strategy' feature, where the installed anti-virus product is queried through WMI and, based on the specific product, a dedicated system infiltration technique is chosen. This means the process injection technique differs between strategies, as well as persistence technique implementation and the overall decision as to whether or not to proceed with infection. A detailed analysis of Casper's strategy can be found at *ESET*'s research blog [7].

INTER-FAMILY RELATIONS

From a binary analyst's perspective, there is little room for doubt that all the discussed binaries stem from the same authors. In practice, this is hard to prove though – explanation of relationships among binaries is often tedious and hard to understand, even for individuals with expertise in binary analysis.

In general, a binary itself does not give away who wrote it, who controlled it, who the infected victims were or what the aim of the operation that involved it was. A standalone binary does not even give away which operation involved it. We cannot conclude from a binary its context. What we can do, though, is determine related binaries, which in most cases helps a great deal in the investigation.

In the case of the SNOWGLOBE malware, a number of binary attributes from different domains can be derived, which in conjunction proves that the families all stem from the same authors. The matrix shown in Table 5 provides an overview.

	Attribute	NBOT	Bunny	Babar	Casper
Shared code	Proxy bypass	x	x	x	x
	AV enumeration	-	x	x	x
	Timestamp formatting	x	x	x	x
Shared techniques	Encrypted config	-	x	x	x
	Partial API hashing	x	x	x	x
	Dynamic API loading technique	x	x	x	x
	AV evasion 'strategies'	-	x	x	x
	Persistence technique	x	x	x	-
	Exception handling	x	x	x	x
String constants	Typo in WMI handling error message	-	x	x	-
	English grammar mistakes	x	x	x	x
	Capital letter string constants	x	x	-	x
Artefacts	File/Regkey naming scheme	x	x	x	x
	Shared C&Cs	x	x	x	-
	French-language domains/websites	-	x	x	-

Table 5: Overview of attributes shared between families.

The families share code, which despite differences in compiler settings can certainly be identified as identical on a source-code level. The discovery of proxy settings, enumeration of anti-virus products via WMI, and the formatting of timestamps for log files are three modules which were frequently found. On the list of shared techniques are: the hashing of a subset of APIs, occasionally even with the same algorithm and key; the 'strategies' in system infiltration based on installed anti-virus products; and the habit of encrypted configurations hard coded in the binaries, which in clear text resemble XML format. String constants formatted in the same fashion or coming with equally misspelled vocabularies are considered a strong link as well. Last but not least, shared C&C server domains are a clear indication that not only the same authors but also the same operators are behind the malware families.

As seen in this example, the attribute extraction and matching technique shows an obvious link among the families. This is perhaps as close as research can get to binary stylometry, the objective of prooving that a set of binaries stems from the same authors. A credible link can only be built by leveraging attributes from different domains. By choosing attributes from domains such as implementation details, techniques used, coding habits and operational infrastructure, the risk of being caught up in comparing unrelated entities such as compiler behaviour or sole infrastructure is minimized.

ATTRIBUTION ASPECTS

Little doubt remains that the analysed binaries match with descriptions of operation SNOWGLOBE, conducted by Canadian intelligence [1]. CSEC mentions that one of the implants comes with the internal name 'Babar', which matches with the .pdb path in the Babar64 binaries at hand. Also, the beaconing contains the misspelled 'MSI' instead of 'MSIE' in the User-Agent string, as outlined in the leaked document. Furthermore, the slides mention a 'locale option of artefact within spear-phishing attack set to "fr_FR"'. While no details of spear-phishing attacks are known, the same locale has been found within the NBOT binaries as a setting for HTTP request headers.

Slide number 10 of the CSEC document says that the C&C infrastructure 'seems to be found primarily, but not exclusively on French-language sites'. This holds true for a number of domains, especially for Bunny's and Babar's remote servers. Keep in mind though, that this helps solely in underlining the statement that CSEC was analysing the same binaries, not that the actor was French.

All three families give the impression of having been developed by a team of skilled software developers, rather than being the product of a malware author operating in the criminal underground. What's more, none of the binaries makes any attempt to hide its intentions, which is a common trait among targeted malware. Heavy obfuscation or the use of crypters easily raises the suspicions of heuristics-based malware scanners. Another interesting fact is that the attackers used at least one 0-day exploit in the spreading of Bunny in 2011 and at least two more in 2014 in the spreading of Casper. The use of 0-day exploits itself does not indicate whether or not a nation state is involved, but it does suggest that the actor had a good amount of resources available, as well as a certain amount of determination.

However, besides the CSEC document there was no obvious indication that the discussed malware families were created by or for French intelligence services. As is often the case with digital crime, the chances are high that no proof will ever be found and research will be limited to educated guesses.

PECULIARITIES AND FUN FACTS

Bunny possesses the capability to upload and download files via HTTP. Successful downloads are handled with status code 418, which is not defined in the RFC of HTTP, but is specified in the RFC April Fools Day edition RFC2324. HTTP status code 418 says 'I'm a teapot'.

The partial obfuscation of API names with a weak hashing algorithm does not serve well as protection from binary analysts as it is easy to bypass. However, it does make sense to trick malware detection solutions which apply heuristics based on static analysis of imports.

There is a typo in the registry key name 'isakmpAutoNegociate' (should be 'isakmpAutoNegotiate'), which gives away the non-legitimate use of what seems to be a standard *Windows* key. For malware writers it makes sense to use legitimate-looking names for artefacts on the system to trick analysts. In order to make full use of the stealth effect though, correct spelling of the names is crucial.

One of the Babar C&C domains is an Algerian travel agency, with office in Bir Mourad Raïs. A contact with close ties to Algeria has noted that the existence of this travel agency is highly unlikely, and that even if it did exist, the possession of a domain hosted in the United States is highly unlikely for an Algerian company. Thus we conclude that this domain, and maybe other seemingly legitimate domains, are fake web representations not operated by real-world companies.

FUTURE RESEARCH

Bits of information from the leaked CSEC slides indicate the existence of so far unknown malware and that the entire operation SNOWGLOBE is much bigger than currently known. It is clear that further investigation will eventually lead to the uncovering of more pieces of the puzzle.

Also, the analysis of Tafacalou, Dino and the likely related NGBD malware remains for future research. Further victimology needs to be conducted in order to determine the outreach of the SNOWGLOBE operation and currently targeted individuals.

ACKNOWLEDGEMENTS

My deepest gratitude goes to a number of fellow researchers involved in the investigation of the SNOWGLOBE malware: Paul Rascagnéres, Joan Calvet, Morgan Marquis-Boire, Sebastien Larinier, Matthieu Suiche, Alexandre Dulaunoy, Raphäel Vinot, Fred Arbogast, Alexei Bulazel and Michael Shalyt.

REFERENCES

[1] SNOWGLOBE: From Discovery to Attribution. Communications Security Establishment Canada. http://www.spiegel.de/media/media-35683.pdf.

[2] Animals in the APT Farm. Kaspersky Securelist. https://securelist.com/blog/research/69114/animals-in-the-apt-farm/.

[3] Marschalek, M. EvilBunny: Suspect #4. https://drive.google.com/file/d/0B9Mrr-en8FX4M2lXN1B4eElHcE0/view.

[4] Dillon, B. Analysing CVE-2011-4369. http://blog.9bplus.com/analysing-cve-2011-4369-part-one/

[5] Marschalek, M. Shooting Elephants. https://netzpolitik.org/wp-upload/Elephantosis.pdf.

[6] Untersinger, M.; Follorou, J. Quand les Canadiens partent en chasse de « Babar ». Le Monde. http://www.lemonde.fr/international/article/2014/03/21/quand-les-canadiens-partent-en-chasse-de-babar_4387233_3210.html.

[7] Calvet, J. Casper Malware: After Babar and Bunny another Espionage Cartoon. http://www.welivesecurity.com/2015/03/05/casper-malware-babar-bunny-another-espionage-cartoon/.

[8] Zakorzhevsky, V. New Flash Player 0-day (CVE-2014-0515) used in watering-hole attacks. Kaspersky Securelist. http://old.securelist.com/en/blog/8212/New_Flash_Player_0_day_CVE_2014_0515_used_in_watering_hole_attacks.

[9] The Lua Project. http://www.lua.org/.

[10] Hunt, G.; Brubacher, D. Detours: Binary Interception of Win32 Functions. Proceedings of Usenix conference '99. http://research.microsoft.com/pubs/68568/huntusenixnt99.pdf.

APPENDIX

Sample hashes

	MD5	SHA-256
Nbot		
	8132ee00f64856cf10930fd72505cebe	82daadf1558692587f82d6ad545b7e6ba2c98a88a20604e2f074f39248aa2712
	2a64d331964dbdec8141f16585f392ba	a1973790e277b489110ae8ed625c13c2e9a79afaa2aed5d27904dcfa46481ae3
	e8a333a726481a72b267ec6109939b0d	4f4b484acc053687d6e4365f0f19e926b1a44cc665182aa6b9417fa43264b240
	51cd931e9352b3b8f293bf3b9a9449d2	43861501e7e7bb546b55a9323d1cff132dddb750b3e86823af7ea08d45357e28
	d5a80844c54059654688ae7abfc1fad9	5ef78d7c2d3b7201e540e6c1909174c7133fca8bafc44ac53ce275e04559c393
Bunny		
	3bbb59afdf9bda4ffdc644d9d51c53e7	be14d781b85125a6074724964622ab05f89f41e6bacbda398bc7709d1d98a2ef
	b8ac16701c3c15b103e61b5a317692bc	7d1e5c4afb1682087d86e793b3fc5a8371dc7c28e27e7196e3b258934f6bafb5
	c40e3ee23cf95d992b7cd0b7c01b8599	c6a182f410b4cda0665cd792f00177c56338018fbc31bb34e41b72f8195c20cc
	eb2f16a59b07d3a196654c6041d0066e	c9197a1fa5f911d11c51a66cfc1424063ed80547ca3f8637b77d06f36fc96e7d
Babar		
	4525141d9e6e7b5a7f4e8c3db3f0c24c	aa73634ca325022dd6daff2df30484ec9031939044cf4c2a004cbdb66108281d
	9fff114f15b86896d8d4978c0ad2813d	c72a055b677cd9e5e2b2dcbba520425d023d906e6ee609b79c643d9034938eb
	8b3961f7f743daacfd67380a9085da4f	82e6f9c10c7ba737f8c79deae4132b9ff82090ccd220eb3d3739365b5276c3c8
	4582D9D2120FB9C80EF01E2135FA3515	57437a675cae8e71ac33cd2e001ca7ef1b206b028f3c810e884223a0369d2f8a
	4592f10c654d613080fe9fa3c1591f20	213bddc35d737867bb168aaf0fb4165bf1afd2216d7662344402f08318650038
Casper		
	4d7ca8d467770f657305c16474b845fe	8e6402c8703e9f10493222a26afeb0fc575bb879d6c82d89c1a79aa75be645d0
	cc87d090a1607b4dde18730b79b78632	daa56e7acd5fb69ecefdbf5179c5ef4776ccc41ebe7e14920f11b84678c83a00