SHOOTING ELEPHANTS

SHOOTING ELEPHANTS

DROPPER

MD5	9fff114f15b86896d8d4978c0ad2813d
SHA-1	27a0a98053f3eed82a51cdefbdfec7bb948e1f36
File Size	693.4 KB (710075 bytes)
Compile Time	2011-08-29 11:48:42

IMPLANT

MD5	4525141d9e6e7b5a7f4e8c3db3f0c24c
SHA-1	efbe18eb8a66e4b6289a5c53f22254f76e3a29bd
File Size	585.4 KB (599438 bytes)
Compile Time	2011-08-29 13:02:29

Special thanks for their contribution in the making of this report go to **Morgan Marquis-Boire**, **Joan Calvet**, **Paul Rascagnéres**, **Alex Dulaunoy**, **Raphael Vinot** and the remaining team of **CIRCL Luxembourg**.
You guys rock \m/

Comments or inquiries please direct to Marion Marschalek (0xF372F2CA).

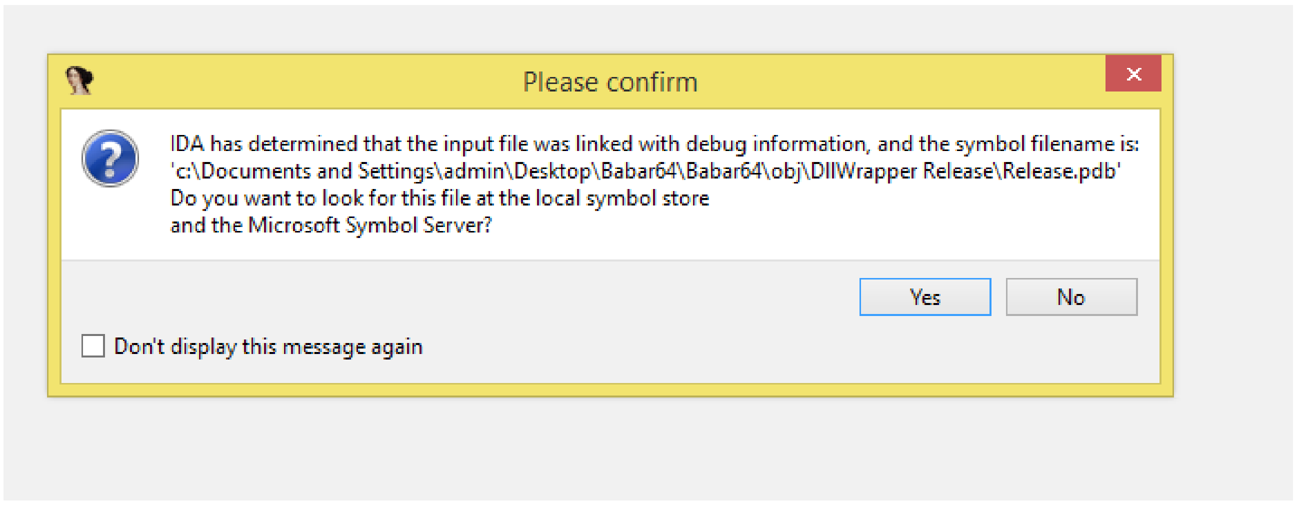
Babar and the related crowd of animalesque malware will be presented at SyScan'15 conference (https://www.syscan.org/) in March 2015. No elephants were harmed in the making of this report.

CONTENT

1. SHOOTING ELEPHANTS

Subject of this analysis is a fascinating piece of malware, which invades Windows desktop machines and aims at.. well, all the things. The analyzed malware consists of a dropper and an implant, which invades Windows processes to steal data from instant messengers, softphones, browsers and office applications. A fully blown espionage kit, so to say, sophisticated almost. The implant is able to hook APIs of interest in dedicated remote processes, to steal data on the fly.

More interesting than the malware itself though is the path to the associated symbol file, which appears embedded in the dropper. The analyzed malware samples come with the internal project name 'Babar64'.



IDA has determined that the input file was linked with debug information, and the symbol filename is:
'c:\Documents and Settings\admin\Desktop\Babar64\Babar64\obj\DllWrapper Release\Release.pdb'
Do you want to look for this file at the local symbol store
and the Microsoft Symbol Server?

Illustration 1 | The .pdb path embedded in Babar's dropper

The myth of Babar has been around for a while in the intelligence community. Questions have been raised since Le Monde published an article on Babar in 2014 [1] and were recently fuelled by a leaked government presentation found among a stash of documents published in January by Spiegel [2]. The leaked document was authored by the Communications Security Establishment Canada (CSEC) and reports about a potential nation state attack involving malware named Babar. The actor behind the attacks is assumed to be French inteligence. Assumptions are based on certain binary attributes, language and location of infrastructure as well as targets.

The binaries at hand fit well with the description CSEC provides, although it is quite clear they are a newer version from what CSEC had uncovered around 2009. It is not clear whether the compilation timestamps are falsified, but an earlier compilation time than the actual stamp seems unlikely.

Doubtlessly though, the Babar binaries match with a malware strain representing itself as Bunny, as well as a family dubbed NBOT or TFC.

2. MISCREANT'S DROPPER

The dropper operates straight forward; it fetches the encrypted DLL from its own binary from outside its resource section, decrypts it and hides it in the file system. Then it goes on to load the DLL using Windows regsvr32.exe. To achieve this it spawns a dedicated process with the command line '"regsvr32.exe" /s /n /i "%APPDATA%\%DLLNAME%"'.

The DLL name is random but legitimately looking and hardcoded for a specific dropper. It is also stored as encrypted string in the binary, and has been seen to decrypt to either 'perf585.dll' or 'dump21cb.dll' depending on the dropper.

The encryption algorithm used is AES (Advanced Encryption Standard) with 128-bit keys. The keys used for decryption are '20 33 AF 73 A9 AC 72 D3 BE E6 A5 73 92 BA 37 6C' for the implant filename and 'A0 0E 3E B3 3A 1C D3 AA A0 BE 3F B3 F9 0A 96 15' for decrypting the binary. AES is the encryption algorithm of choice throughout the dropper and its implant's operations.

Interestingly, the Babar dropper as well as the implant show a similar method of API name obfuscation as the Bunny malware and samples from the NBOT family. Dedicated APIs are loaded through a resolution function, which searches for library exports by comparing hashes of the export names with a given hash. For the Babar project though the attackers used an adapted version of SHA-1, as opposed to the simple XOR/ROL hash algorithm which was used in the Bunny project.

After infection the dropper executes a separate command line through spawning a cmd.exe process, which is instructed to wait for 1 second before deleting the dropper binary on disk. This way the malware reduces forensic artefacts, staying behind after infection. Curiously though, the regsvr32.exe process used to load the dropped implant remains running. This way Babar leaves a visible process on the process list during execution.

As the project name 'Babar64' suggests, dropper and implant are designed to work on 32-bit and on 64-bit Windows versions. However, process hooking and injection can only affect 32-bit processes due to the bitness of the implant itself.

3. MISCREANT'S IMPLANT

Babar's implant is a 32-bit DLL written in C++, which upon start injects itself to running processes and invades desktop applications by applying a global Windows hooks. The implant is capable of logging keystrokes, capture screen shots, spy on installed softphones and instant messengers next to a list of simpler espionage tricks. Babar is a fully blown espionage tool, built to excessively spy on the user's activity on an infected machine.

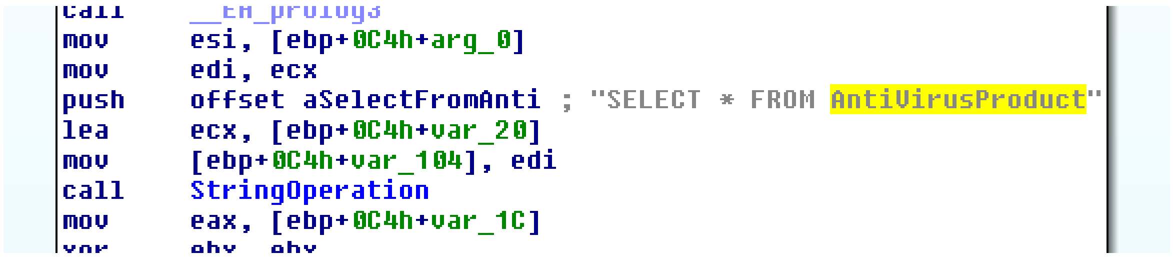
The DLL dropped by Babar is placed into the application data folder, along with a directory named 'MSI' where the runtime data will be stored. Babar operates through multiple instances, by injecting its DLL to a maximum of three desktop processes. This is achieved by loading the Babar DLL to remote processes through a mapped memory object.

Apart from that, Babar comes with a userland rootkit component which applies global Windows hooks to invade all processes on its desktop. This way Babar can install API hooks for various APIs via Windows Detours technique to actively steal data from arbitrary processes.

3.1 ANTI-ANALYSIS MEASURES

Babar's anti-analysis measures are sparse. It is not protected by a runtime packer or crypter, it does not have sandbox detection or anti-debugging measures, most character strings are shown in clear-text.

Babar does show the same obscure treatment for installed anti-virus products as the Bunny malware though. Babar also enumerates the Windows Management Instrumentation for installed AV solutions, while the exact use of this information is not yet known. We do suspect large portions of the product enumeration to be based on the exact same source code as the module Bunny used.



```
call    __EH_prolog3
mov     esi, [ebp+0C4h+arg_0]
mov     edi, ecx
push    offset aSelectFromAnti ; "SELECT * FROM AntiVirusProduct"
lea     ecx, [ebp+0C4h+var_20]
mov     [ebp+0C4h+var_104], edi
call    StringOperation
mov     eax, [ebp+0C4h+var_1C]
xor     ebx, ebx
```

Illustration 2 | Querying anti-virus products installed on the machine

The enumeration function issues the commands „SELECT * FROM AntiVirusProduct“ and retrieves attributes such as 'productState', 'DisplayName', 'VersionNumber' and 'productUptoDate'.

The anti-virus solutions searched for are again identified by 256-bit SHA hashes. Only some of the hashes could be mapped to known names of anti-virus solutions.

```
4db3801a45802041baa44334303e0498c2640cd5dfd6892545487bf7c8c9219f  ThreatFire
522e5549af01c747329d923110c058b7bb7e112816de64bd7919d7b9194fba5b  Rising
2fd5c42d49f9e0fe2daae5b0f78cf9cfde9bfc7b0ed59fc68e0a79a3b16fe05b
06e387bb79584cdff3672feadea0bf6f783ce1ddc1fa91962d1b5bcd94e1a308
f1761a5e3856dceb3e14d4555af92d3d1ac47604841f69fc72328b53ab45ca56  Kaspersky
588730213eb6ace35caadcb651217bfbde3f615d94a9cca41a31ee9fa09b186c  ZoneAlarm
c8e8248940830e9f1dc600c189640e91c40f95caae4f3187fb04427980cdc479
b3fe0e3a3e3befa152c4237b0f3a96ffaa44a2d7e1aa6d379d3a1ab4659e1676  AntiVir
```

Babar also implements an obfuscation technique to hide certain API names. Selected APIs are identified by hashes, which are used to load the APIs dynamically at runtime. The hashes are hardcoded within the binary, and will be compared to hashed library exports by the API resolution routine.

The hashing algorithm is different from the simple XOR/ROL technique Bunny uses. It seems to be based on SHA-1, but generates 32-bit hashes instead of the standard 160-bit length.

What Bunny, Babar and NBOT all have in common though is that the obfuscation technique is easy to break and only applied to a subset of APIs. As an anti-analysis trick this is considerably useless, it does make sense though to trick malware detection solutions which apply heuristics based on static analysis of API calls. This becomes evident when looking at the list of Babar's obfuscated API names, which includes RegisterRawInputDevices, GetRawInputData, GetClipboardData or DirectSoundCaptureCreate.

3.2 BABAR CAME TO STAY

The startup routine of the Babar implant will create a registry key under '[HKU]\..\CurrentVersion\Run' to assure persistence. The key is named 'MSSecurity' and executes the exact same command line as the infector, every time the system boots - '"regsvr32.exe" /s /n /i "%APPDATA%\%DLLNAME%"'. At system boot time the regsvr32.exe process will start, load the Babar DLL and thus 'deploy' the malware to various legitimate processes. As opposed to persistence, Babar also implements functionality to uninstall itself from the affected system.

Interesting again, same as during infection, the regsvr32.exe process remains running even after Babar is readily set up.

3.3 CONFIGURATION DATA

During initialization Babar loads and decrypts a set of configuration parameters which are appended to the DLL's relocation section. ASCII strings within the configuration data reveal a lot about Babar's intentions:

- Office executable names and associated document abbreviations
  excel.exe, winword.exe, powerpnt.exe, visio.exe, acrord32.exe, notepad.exe,

wordpad.exe.txt, rtf, xls, xlsx, ppt, pptx, doc, docx, pdf, vsd

- Softphone executable names
  skype.exe, msnmsgr.exe, oovoo.exe, nimbuzz.exe, googletalk.exe, yahoomessenger.exe, x-lite.exe

- A version number
  12075-01

- Path and filename of the dump file
  %COMMON_APPDATA%\MSI\update.msi

- Path to the runtime data directory
  %COMMON_APPDATA%\MSI

- Two C&C server domains and request parameters
  http://www.horizons-tourisme.com/_vti_bin/_vti_msc/bb/index.php
  http://www.gezelimmi.com/wp-includes/misc/bb/index.php

- Web browser executable names, including MSN messenger
  iexplore.exe,firefox.exe,opera.exe,chrome.exe,Safari.exe,msnmsgr.exe

- The name of the dropped implant
  perf_585.dll

As could be seen already in binaries of related families, next to the hardcoded configuration data Babar maintains a system specific runtime configuration. The additional configuration consists of a set of local attributes and a set of values derived from the Windows environment. The attributes are dynamically generated or requested at runtime and not stored in memory.

%USERHASH% - Hash of the login name
%USER% - Login name
%SEQ% - Assumed to be the sequence number of dump files created by Babar
%KID% - Assumed to be the ID of a subsequently infected process
%ID% - Value retrieved from hardcoded configuration
%SELFDIR% - Executable directory
%SELF% - Executable name

Values retrieved from Windows environment:

%APPDATA%
%USERPROFILE%
%WINDIR%
%COMMON_APPDATA%
%ALLUSERPROFILE%
%CommonProgramFiles%

3.4 MODUS OPERANDI ELEPHANTI

The DLL when loaded in the context of an application seeks to invade a maximum of two more victim processes. This for once is a resilience measure, so if the initially infected process stops running the malware remains in memory through additional instances. On the other hand it is practically a load balancing measure. The C&C communication module is located in an export of the DLL, which will be executed through a remote thread, injected like the child processes, via a memory-mapped file.

The process infection is achieved by mapping a shared object into the victim's process space and invoking a function stub as remote thread. The steps being taken are:

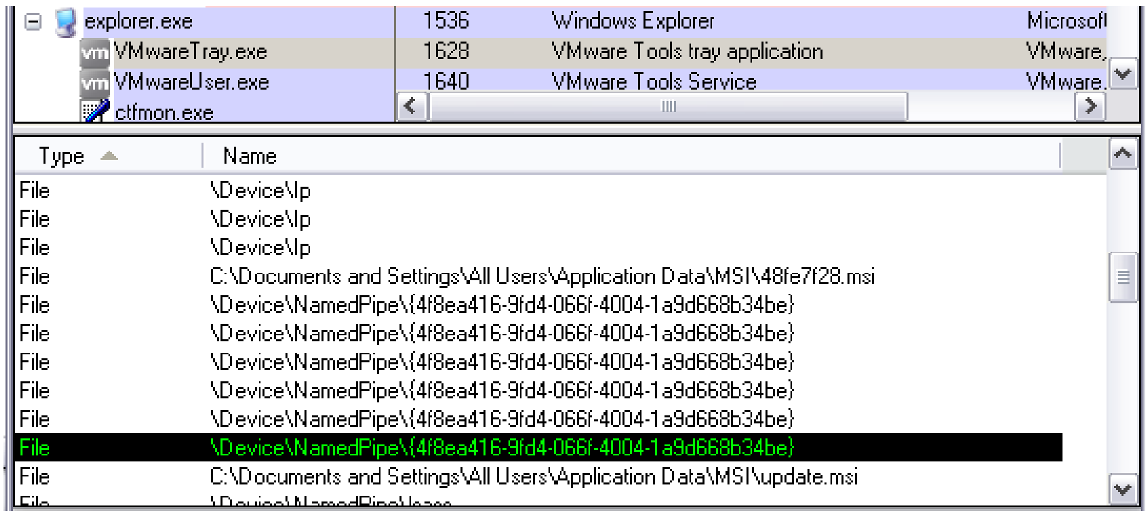
- MapViewOfFile – mapped memory shared with the child instance
- OpenProcess – obtaining a handle to the chosen victim process
- VirtualAllocEx – allocate space in remote memory
- WriteProcessMemory – write function stub to remote memory
- CreateRemoteThread – execute function stub as remote process

The function stub will then go on to load the Babar DLL through LoadLibraryA and execute one of its exports, as indicated in the shared memory. The mapped object contains name and path to the malicious DLL, the name of the pipe being used for communication between the instances, the name of the export from the DLL to be called as well as information about instances already running.

For picking a process to infect the malware randomly picks one from a list of prospects. These have to be 32-bit processes, not already infected and not among a list of processes to avoid such as winlogon.exe, explorer.exe, cmd.exe or regsvr32.exe.

This way Babar always keeps three instances in memory. The first one to start up will be the 'main instance'. If any of the instances dies, a new third instance will be created. If the main instance dies the oldest child will take over.

For inter process communication Babar uses named pipes. The main instance generates a random GUID which is used as name and passed to child instances. At the same time, the main instance creates six named pipes using the very same GUID. These pipes represent the server side for child instances, and also the point to connect to for hooks Babar installs to spy on several system APIs.



```
explorer.exe        1536   Windows Explorer                 Microsoft
  VMwareTray.exe    1628   VMware Tools tray application    VMware,
  VMwareUser.exe    1640   VMware Tools Service             VMware,
  ctfmon.exe

Type    Name
File    \Device\Ip
File    \Device\Ip
File    \Device\Ip
File    C:\Documents and Settings\All Users\Application Data\MSI\48fe7f28.msi
File    \Device\NamedPipe\{4f8ea416-9fd4-066f-4004-1a9d668b34be}
File    \Device\NamedPipe\{4f8ea416-9fd4-066f-4004-1a9d668b34be}
File    \Device\NamedPipe\{4f8ea416-9fd4-066f-4004-1a9d668b34be}
File    \Device\NamedPipe\{4f8ea416-9fd4-066f-4004-1a9d668b34be}
File    \Device\NamedPipe\{4f8ea416-9fd4-066f-4004-1a9d668b34be}
File    \Device\NamedPipe\{4f8ea416-9fd4-066f-4004-1a9d668b34be}
File    C:\Documents and Settings\All Users\Application Data\MSI\update.msi
```

Illustration 3 | Named pipes opened by the main instance

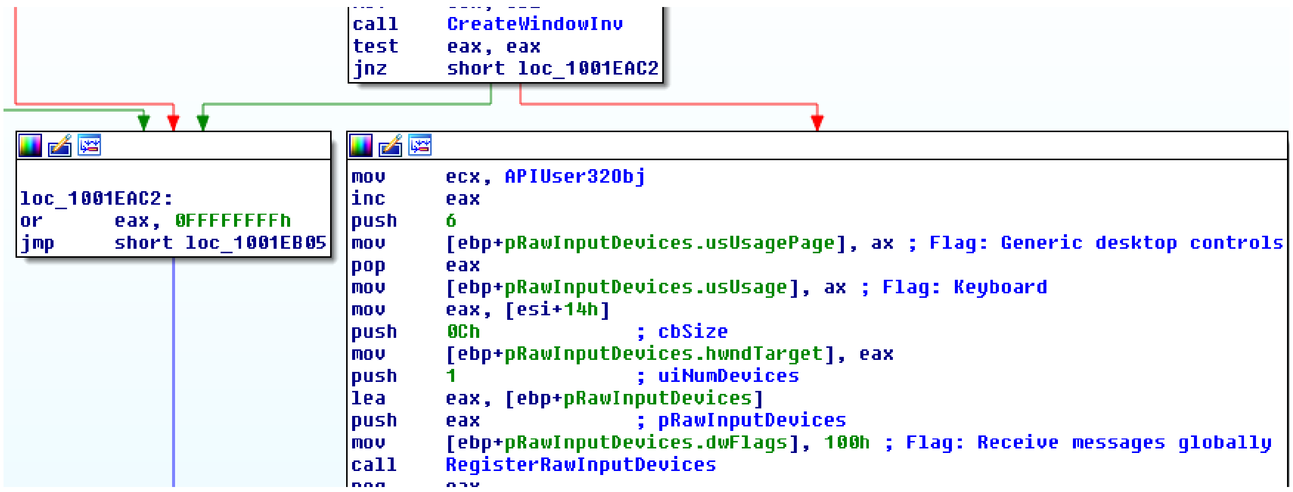
3.5 FUNCTIONALITY

The spying activities are performed either through the Babar instance locally or via a global Windows hook invading all processes running in 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 would be as follows:
- Logging keystrokes
- Taking screenshots
- Capture of audio streams from softphone applications
- Stealing of clipboard data
- System and user default language, keyboard layout
- Names of desktop windows

3.5.1 THE KEYLOGGER MODULE

The keylogger is based on the raw input model. The malware creates an invisible window, with no other purpose than to receive window messages. By processing the window message queue it filters out input events and dispatches them to a raw input device object. Said object is configured to grab keyboard events through GetRawInputData.



```
call    CreateWindowInv
test    eax, eax
jnz     short loc_1001EAC2
```

```
loc_1001EAC2:
or      eax, 0FFFFFFFFh
jmp     short loc_1001EB05
```

```
mov     ecx, APIUser32Obj
inc     eax
push    6
mov     [ebp+pRawInputDevices.usUsagePage], ax ; Flag: Generic desktop controls
pop     eax
mov     [ebp+pRawInputDevices.usUsage], ax ; Flag: Keyboard
mov     eax, [esi+14h]
push    0Ch             ; cbSize
mov     [ebp+pRawInputDevices.hwndTarget], eax
push    1               ; uiNumDevices
lea     eax, [ebp+pRawInputDevices]
push    eax             ; pRawInputDevices
mov     [ebp+pRawInputDevices.dwFlags], 100h ; Flag: Receive messages globally
call    RegisterRawInputDevices
```

Illustration 4 | The keylogger module uses Windows RAWINPUT to achieve its goal

The snooped keystroke data is passed to a thread, which performs encryption and dumps the data to a log file. The file is located in Babar's working directory and named update.msi.

The design of the keylogging component is simple but effective. Babar is able to sniff all keystrokes happening on the same desktop as its invisible window. Interesting though, the code seen in Babar's implant is remarkably similar to an example posted at [4].

3.5.2 INVADING DESKTOP PROCESSES

The Babar implant applies global Windows hooks to load its DLL into the process space of other processes. A global hook is installed by calling SetWindowsHookEx by passing the thread ID zero. It applies to all processes running in the same desktop, having the same 'bitness' as the DLL to be injected. In the given case this applies to all 32-bit processes.

A global Windows hook is installed into the Windows event chain. This effectively means 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 type 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.

3.5.3 I AM ROOT(kit)

Through the hook mechanism Babar can be maximum invasive in the Windows userland. 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 [6]. 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 malicious activity and then passes control on to the legitimate API.

Babar supports trampoline functions for a limited list of APIs and limited to a set of processes of interest, defined by its configuration. The list of APIs to hook is parted in three groups:

- Internet communication
  WSARecv
  send
  closesocket

- File creation
  CreateFileW

- Media
  DirectSoundCreate
  DirectSoundCaptureCreate
  DirectSoundCreate8
  DirectSoundCaptureCreate8
  CoCreateInstance
  waveOutOpen
  waveOutClose
  waveOutWrite
  waveInOpen
  waveInClose
  waveInAddBuffer

Any set of hooks will only be applied to a defined set of processes. The predefined groups of application names retrieved from the hardcoded configuration data set the scope of Babar's hooking advances:

- Internet communication
  iexplore.exe,firefox.exe,opera.exe,chrome.exe,Safari.exe,msnmsgr.exe

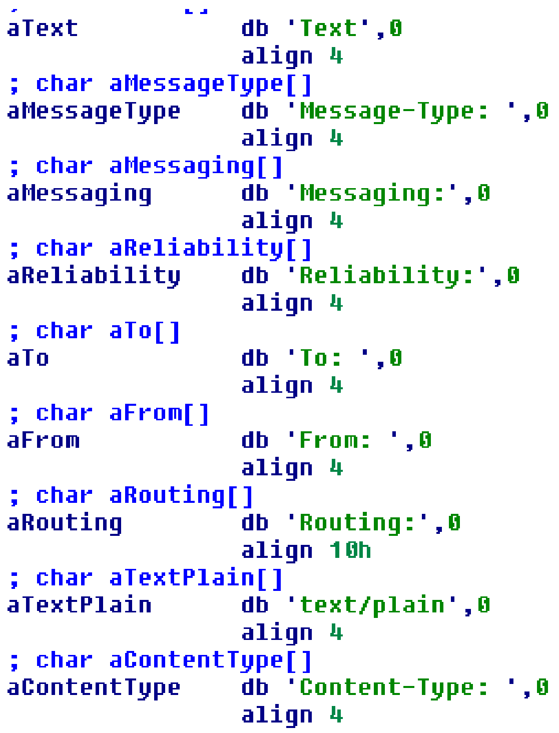
- File creation
  excel.exe, winword.exe, powerpnt.exe, visio.exe, acrord32.exe, notepad.exe, wordpad.exe.txt, rtf, xls, xlsx, ppt, pptx, doc, docx, pdf, vsd

- Media
  skype.exe, msnmsgr.exe, oovoo.exe, nimbuzz.exe, googletalk.exe, yahoomessenger.exe, x-lite.exe

The respective trampolines steal data going in or out of the hooked APIs on the fly. The parsing function for intercepted internet communication searches for chat traffic, more specifically for messages conforming with the MSNP21 standard. Babar is searching for tags such as 'Message-Type', 'Reliability', 'To', 'From' or 'Text' to pick out of the stream. A more detailed explanation concerning the MSNP21 can be found at [7]. MSNP21 is the MSN messenger chat protocol that was introduced with Windows Live 2010 beta. It is unclear though as of why only parsing for the MSN messenger is supported.

The data, like all stolen information, is handled by a separate thread which compresses and encrypts it before dumping it to a file on disk.

For grabbing audio streams Babar includes code from the OpenCORE AMR library [8]. AMR enables the malware to encode and decode raw audio frames as they are passed to or grabbed from the audio devices. The very same malware module performs screen captures while dumping the snooped data. This is assumed to happen to recognize the parties involved in the conversation by capturing the GUI of the softphone application.



```
aText           db 'Text',0
                align 4
; char aMessageType[]
aMessageType    db 'Message-Type: ',0
                align 4
; char aMessaging[]
aMessaging      db 'Messaging:',0
                align 4
; char aReliability[]
aReliability    db 'Reliability:',0
                align 4
; char aTo[]
aTo             db 'To: ',0
                align 4
; char aFrom[]
aFrom           db 'From: ',0
                align 4
; char aRouting[]
aRouting        db 'Routing:',0
                align 10h
; char aTextPlain[]
aTextPlain      db 'text/plain',0
                align 4
; char aContentType[]
aContentType    db 'Content-Type: ',0
                align 4
```

Illustration 6 | MSNP21 tags

3.6 STOLEN GOODS

Babar comes with the Deflate algorithm, as used by Zlib, to be able to compress data before encrypting it and dumping it to disk. The data is encrypted with 128-bit AES using the key 24 FE C5 AD 34 56 F7 F8 12 01 00 AE B6 7C DE AB for reading and writing files. The following files have been seen to be dumped to Babar's working directory under %APPDATA%\MSI:

- update.msi
- 48fe7f28.msi
- 0c6b5d2d.msi
- 31e50daa.msi
- mpavdlta.vdm

Stolen information will be handled by the Babar main instance, which receives data through six named pipes from other instances or hooked processes. The update.msi-file stores data collected by the keylogging module. It posesses a file header which among other attributes keeps system specific data. This is assumed to serve for mapping of logfiles to infected hosts.

Log lines written for stolen data from browsers and the MSN messenger are built by the format string "%s%c%s|%s\n", where the first string is likely participant1 of a conversation, the second string participant2. Both are parted by the characters '<' or '>' depending on the direction of the conversation.

Similar log lines accompany dumped sound data and saved documents. Sound data logs follow the format "A|%u|%S", document logs the format "%I64i|%I64i|%I64i|%s%s|%s\n". For documents three timestamps are included as well as whether read or write access was requested, indicated by 'R' or 'W' or '-'. The last placeholder is reserved for the filename. At the time of writing the content of the sound data log line remains unclear.

3.7 CALLING HOME

The internet communication module of Babar is located in a sparate export called 'FindCtxSectionStringW', which will be invoked through remote thread injection at runtime. Injection is achieved the same way as the infection of child instances described in section 3.4.

The analyzed sample of Babar has two hard coded C&C server addresses which are included in its configuration data:

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

Both servers were used to push spied information onto the remote site. It remains unclear though whether the C&Cs also served to instruct the infected machines per commands, as Babar does not posess obvious command parsing functionality. Data exfiltration is assumed to be time triggered.

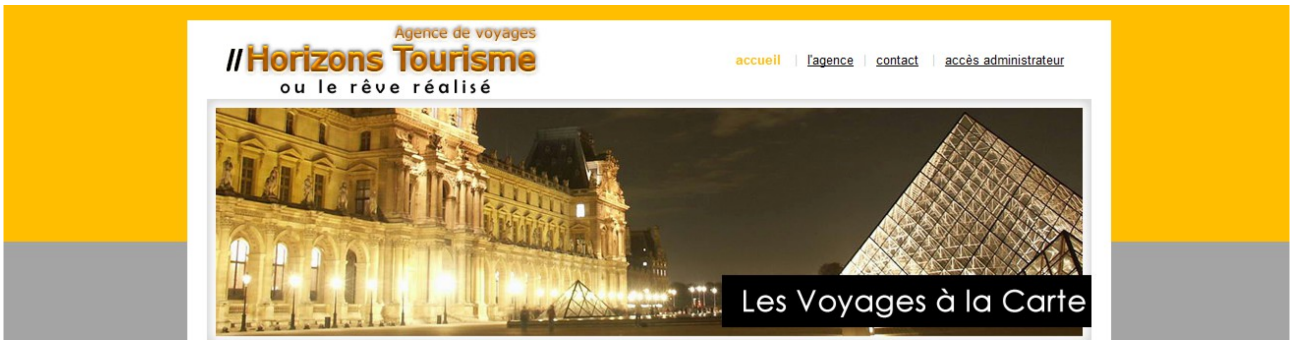


Illustration 7 | The website of horizons-tourisme.com today

The domain horizons-tourisme.com is a legitimate website, operated by an Algerian travel agency, located in Algiers, Algeria. The website is in French and still online today. Gezelimmi.com is a Turkish domain, currently responding with an HTTP error message 403, access not permitted. Both domains appear to be of legitimate use, but compromised and abused to host Babar's server side infrastructure.

At the time of writing the server pointed to by horizons-tourisme.com is still hosting left overs from the C&C infrastructure used by Babar. With directory traversal activated researchers from ESET could pull a minimalistic directory structure, showing directories named as follows:

- bb28
- d13
- tfc422

Obviously, the directory belonging to Babar is 'bb28'. 'tfc422' matches with strings found in the TFC bots (previously named NBOT). The purpose of the 'd13' directory remains unknown, although it is assumed to serve for requests of a third currently unknown family. Most of the directories pulled from horizons-tourisme.com are empty, or contain empty files. The only script inside the bb28 directory though appears to be interesting. Its a .php-script named config.inc, containing variables which look familiar from Babar's config such as 'user', 'id' or 'seq', but also a handful of comments from the authors.

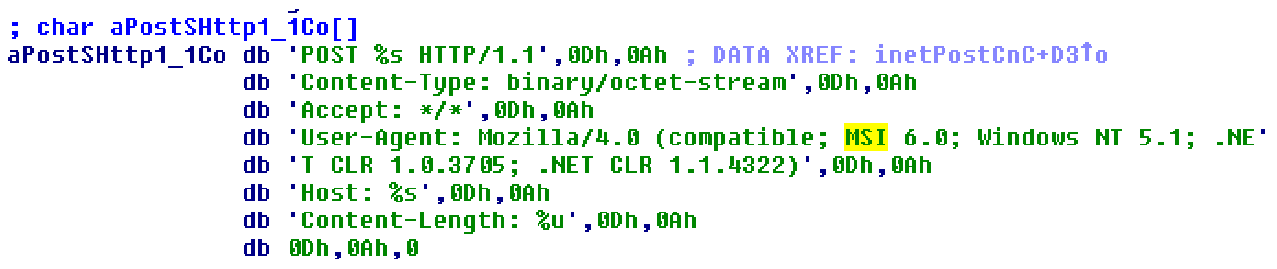
```
<?php
  $uninstall = false; //true to uninstall
  $buninstall_id = false; //true to uninstall from ID
  $uninstall_id = "0C124D55"; //ID to uninstall in hex
  //$uninstall = true;
  $debug = false; //true to see errors messages FOR TEST ONLY!!
  $writelogs = false; //true to create logs file in logs 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;
      $gm_expires = gmdate('D, d M Y H:i:s', time() + $expires);
      header("Cache-control: private, max-age=$expires, pre-check=$expires");
      header("Expires: $gm_expires GMT");
      header("Last-Modified: " . gmdate("D, d M Y H:i:s") . " GMT");
  }
  function listingFile($fn) {
      if (!file_exists($fn)) return false;
      echo "<A HREF= \"$fn\">$fn</A><BR>" ;
      return true;
      }

  function resetStorageFile($fn) {
      if (!file_exists($fn)) return false;
      $fp = fopen($fn, "wb");
      if ($fp) fclose($fp);
      else return false;
      return true;
      }

?>
```

4. BEYOND BABAR

At the time of writing two Babar64 droppers are known, dropping one implant each. They both show traits of the malware described by CSEC, being attributed to French intelligence with „moderate certainty“ [2].

The CSEC document mentions uncovering the attack in 2009. The compilation timestamps of the binaries date back to August 2011. While these can be faked it is still unlikely that the authors would change the 2009-timestamp to a future date. This, and the mentioning of 'Babar' instead of 'Babar64' by CSEC indicates that the samples at hand stem from a more recent campaign.

Besides the project name, Babar64 also shares the malformed user agent string described in the document, where a letter from the MSIE 6.0 name is missing.



```
; char aPostSHttp1_1Co[]
aPostSHttp1_1Co db 'POST %s HTTP/1.1',0Dh,0Ah ; DATA XREF: inetPostCnC+D3↑o
                db 'Content-Type: binary/octet-stream',0Dh,0Ah
                db 'Accept: */*',0Dh,0Ah
                db 'User-Agent: Mozilla/4.0 (compatible; MSI 6.0; Windows NT 5.1; .NE'
                db 'T CLR 1.0.3705; .NET CLR 1.1.4322)',0Dh,0Ah
                db 'Host: %s',0Dh,0Ah
                db 'Content-Length: %u',0Dh,0Ah
                db 0Dh,0Ah,0
```

Illustration 8 | The malformed user agent string showing 'MSI' instead of 'MSIE'

The second Babar64 binary comes with its own set of C&C servers:

- http://www.alexpetro.com/images/training/courses/bb212/index.php
- http://www.etehadyie.ir/images/public/bb212/index.php

The first one is the legitimate website of AlexPetro Technical Services, a company operating in the oil and gas industry in Kairo, Egypt. The second domain is Iranian, allegedly owned by a web design company named Radcom.

What links Babar64 to other families analyzed beforehand, such as Bunny or TFC (also called NBOT) are shared portions of source code and re-used coding techniques. The enumeration of anti-virus software and the code to query the system's proxy settings are almost identical. Furthermore, the samples at hand show the same partial API name obfuscation as Bunny and TFC. What has changed is the hashing algorithm. Also common among all of the families seems to be to hold system specific configuration data in memory at runtime, although this holds true for a lot of malware.

Putting Babar, Bunny and TFC in context a number of conclusion can be drawn. Looking at the compilation timestamps and assuming they are not faked (which is backed by VirusTotal submission dates), the oldest family is TFC being active throughout 2010. These are DDoS bots, used to build a botnet which can be utilized to attack adversaries on the internet.

Next on the timeline is Babar64, allegedly compiled middle of 2011 and first seen on VirusTotal middle of 2012. It is believed that this strain has been active from 2011 until 2013, spying on dedicated targets rather than being spread widely.

Last comes Bunny, compiled second half of 2011 and making its first public appearance during the analysis of the Adobe Reader exploit for CVE-2011-4369 [9]. At the time when Bunny was spread through CVE-2011-4369, before December 2011, the vulnerability was unknown to the public. Bunny itself is a Lua script execution platform, not including any spying capabilities.

All three families leave the impression of being developed by a team of skilled software developers, rather than being the product of a malware author operating in the criminal underground. Also none of the binaries makes attempts of hiding its intentions, which is a common trait among targeted malware. Heavy obfuscation or the use of crypters easily raises suspicion of heuristics based malware scanners.

However, besides the CSEC document there was no obvious pointer found leading to the conclusion that Babar and its companions were set out by French inteligence services. As it is with digital crime, chances are high no proof will ever evolve and research will be limited to educated conclusions. The drawing of personal educated conclusions is left as an exercise for the interested reader.

5. RESOURCES

[1]

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

[2]
SNOWGLOBE: From Discovery to Attribution, CSEC, published by Der Spiegel
http://www.spiegel.de/media/media-35683.pdf

[3]
Slidedeck TS/NOFORN, Keynote Hack.lu 2014 on Bunny / TFC
http://2014.hack.lu/archive/2014/TSNOFORN.pdf

[4]
A Minimal Keylogger using RAWINPUT, CodeProject
http://www.codeproject.com/Articles/297312/Minimal-Key-Logger-using-RAWINPUT

[5]
Implementing Keyloggers in Windows, Securelist
https://securelist.com/analysis/36358/keyloggers-implementing-keyloggers-in-windows-part-two/

[6]
Detours: Binary Interception of Win32 Functions, Proceedings of Usenix conference '99
http://research.microsoft.com/pubs/68568/huntusenixnt99.pdf

[7]
Documentation for changes in MSN Protocol Version 21
https://code.google.com/p/msnp-sharp/wiki/KB_MSNP21

[8]
AMR Audio Encoding, Potluri Suresh
http://www.codeproject.com/Articles/332109/AMR-Audio-Encoding

[9]
Analyzing CVE-2011-4369, 9bplus
http://blog.9bplus.com/analyzing-cve-2011-4369-part-one/

6. APPENDICES

Lists of passive DNS entries for horizons-tourisme.com and gezelimmi.com can be found below. For gezelimmi.com:

Resolve	Location	Network	First	Last
104.153.45.38	N/A	104.153.45.0/24	31.12.2013 00:00	09.11.2014 00:00
199.231.93.221	US	199.231.93.0/24	03.07.2011 00:00	31.12.2013 00:00
199.119.202.195	US	199.119.200.0/21	10.04.2011 00:00	03.07.2011 00:00
208.87.242.66	US	208.87.240.0/22	16.03.2008 00:00	10.04.2011 00:00
209.62.21.228	US	209.62.0.0/17	26.08.2007 00:00	16.03.2008 00:00
83.149.75.58	NL	83.149.64.0/18	24.06.2007 00:00	26.08.2007 00:00
69.25.212.153	US	69.25.208.0/20	24.03.2007 00:00	24.06.2007 00:00
64.20.43.107	US	64.20.32.0/19	18.11.2006 00:00	24.03.2007 00:00
207.189.104.87	US	207.189.96.0/19	12.08.2006 00:00	18.11.2006 00:00
207.189.104.86	US	207.189.96.0/19	05.08.2006 00:00	12.08.2006 00:00
207.189.104.87	US	207.189.96.0/19	08.07.2006 00:00	05.08.2006 00:00
207.189.104.86	US	207.189.96.0/19	01.07.2006 00:00	08.07.2006 00:00
207.189.104.87	US	207.189.96.0/19	17.06.2006 00:00	01.07.2006 00:00
207.189.104.86	US	207.189.96.0/19	10.06.2006 00:00	17.06.2006 00:00
207.189.104.87	US	207.189.96.0/19	06.05.2006 00:00	10.06.2006 00:00
207.189.104.86	US	207.189.96.0/19	29.04.2006 00:00	06.05.2006 00:00
207.189.104.87	US	207.189.96.0/19	18.03.2006 00:00	29.04.2006 00:00
207.189.104.86	US	207.189.96.0/19	05.03.2006 00:00	18.03.2006 00:00
207.189.104.87	US	207.189.96.0/19	25.02.2006 00:00	05.03.2006 00:00
207.189.104.86	US	207.189.96.0/19	12.02.2006 00:00	25.02.2006 00:00
207.189.104.87	US	207.189.96.0/19	27.11.2005 00:00	12.02.2006 00:00
216.152.252.55	US	216.152.240.0/20	21.11.2004 00:00	27.11.2005 00:00

For horizons-tourisme.com:

Resolve	Location	Network	First	Last
192.185.113.148	US	192.185.64.0/18	28.04.11 00:00	10.02.15 00:00
184.172.143.188	US	184.172.128.0/18	23.12.06 00:00	28.04.11 00:00
212.27.35.109	FR	212.27.32.0/19	02.08.05 00:00	23.12.06 00:00
206.41.94.190	CA	206.41.94.0/24	24.06.05 00:00	02.08.05 00:00