March 5, 2015

In March 2014, French newspaper Le Monde revealed that France is suspected by the Communications Security Establishment Canada (CSEC) of having developed and deployed malicious software for espionage purposes. This story was based on presentation slides leaked by Edward Snowden, which were then published by Germany's Der Spiegel in January 2015.

According to the CSEC presentation, the malicious software in question is called "Babar" by its creators, likely after the famous French cartoon character "Babar The Elephant". Since then, several malware researchers have begun to work on the enigma that is Babar. Marion Marschalek (Cyphort) struck first, with her report on the "Bunny" malware. Bunny shares some characteristics with the Babar malware described by CSEC. In mid-February, Marion published another report, this time on the actual Babar case, explaining in great detail its spying features. At the same time, Paul Rascagnères (G Data) published a blog post on the similarities between Babar and Bunny, and showed that they were very probably related to the malware described in the CSEC's slides.

In this blog post, we lift the veil on another piece of software that we believe to have been created by the same organization that is behind Babar and Bunny. This component is called "Casper" by its authors – presumably named after yet another famous cartoon character.

Casper was used against Syrian targets in April 2014, which makes it the most recent malware from this group publicly known at this time. To attack their targets, Casper's operators used zero-day exploits in Adobe Flash, and these exploits were – surprisingly – hosted on a Syrian governmental website. Casper is a well-developed reconnaissance tool, making extensive efforts to remain unseen on targeted machines. Of particular note are the specific strategies adopted against antimalware software.

Context

In mid-April 2014, Vyacheslav Zakorzhevsky (Kaspersky) observed that the website "jpic.gov.sy" was hosting two Flash zero-day exploits, targeting the vulnerability later labeled CVE-2014-0515. This website was set up in 2011 by the Syrian Justice Ministry apparently to allow Syrian people to ask for reparation for the damage of the civil war. The website is still online and apparently currently clean, although it was defaced in September 2014 by some "hacktivist".

At the time of the events, Zakorzhevsky could not retrieve the payloads distributed by these Flash zero-days exploits. ESET researchers were able to find two of these payloads, thanks to ESET LiveGrid® threat telemetry systems. The URLs of these payloads and the dates when they were seen correspond to Zakorzhevsky's description.

In a joint effort with Marion Marschalek, Paul Rascagnères, and researchers from the Computer Incident Response Center Luxemboug (CIRCL), we were recently able to determine that the payloads distributed were very likely developed by the same actors who developed the Babar and Bunny software.

Casper Binary Analysis

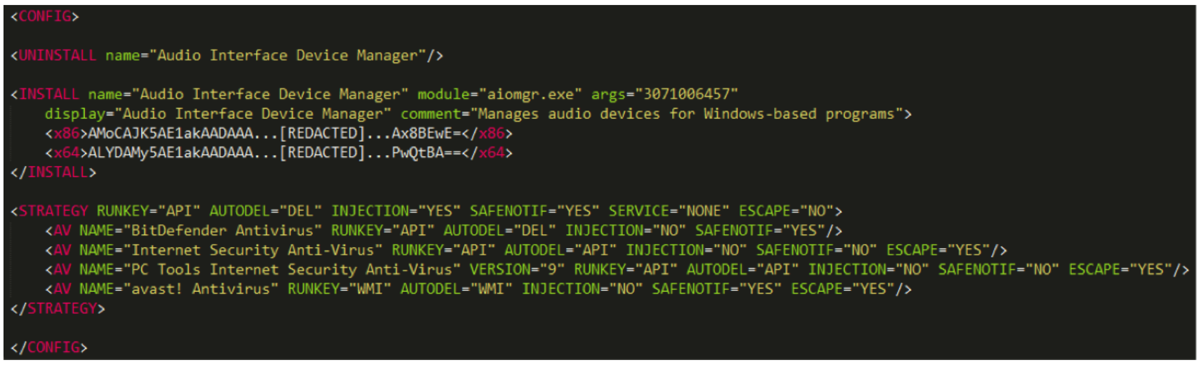
The two samples we found are the same core program but differently packaged. The first sample is an executable dropping the core program and making it persistent on the machine. The second is a Windows library that deploys the core program directly into memory, also in the form of a library. In this latter case, the name of the core program library was left visible by its creators: "**Casper_DLL.dll**".

Throughout this blog, we will focus on the first of these two payloads, the second one being similar in terms of behavior.

Dropper

The dropper is named "domcommon.exe" and its compilation date is set to the June 18th, 2010. This is very likely a forged date, as we will explain later.

Its execution is based on an XML configuration file decrypted at runtime with the RC4 algorithm and a hardcoded 16-byte key. Before the decryption, the program uses a checksum computation to make sure the memory area containing the decryption key has not been modified. Figure 1 shows the dropper's decrypted configuration file.



```
<CONFIG>

<UNINSTALL name="Audio Interface Device Manager"/>

<INSTALL name="Audio Interface Device Manager" module="aiomgr.exe" args="3071006457"
    display="Audio Interface Device Manager" comment="Manages audio devices for Windows-based programs">
    <x86>AMoCAJK5AE1akAADAAA...[REDACTED]...Ax8BEwE=</x86>
    <x64>ALYDAMy5AE1akAADAAA...[REDACTED]...PwQtBA==</x64>
</INSTALL>

<STRATEGY RUNKEY="API" AUTODEL="DEL" INJECTION="YES" SAFENOTIF="YES" SERVICE="NONE" ESCAPE="NO">
    <AV NAME="BitDefender Antivirus" RUNKEY="API" AUTODEL="DEL" INJECTION="NO" SAFENOTIF="YES"/>
    <AV NAME="Internet Security Anti-Virus" RUNKEY="API" AUTODEL="API" INJECTION="NO" SAFENOTIF="NO" ESCAPE="YES"/>
    <AV NAME="PC Tools Internet Security Anti-Virus" VERSION="9" RUNKEY="API" AUTODEL="API" INJECTION="NO" SAFENOTIF="NO" ESCAPE="YES"/>
    <AV NAME="avast! Antivirus" RUNKEY="WMI" AUTODEL="WMI" INJECTION="NO" SAFENOTIF="YES" ESCAPE="YES"/>
</STRATEGY>

</CONFIG>
```

Figure 1 – Casper Dropper Configuration File

Casper Playing Chess against Antivirus

Firstly, the dropper extracts the tag from its configuration file. This tag defines precisely how the malware should behave, depending on which antivirus is present on the machine.

Choosing the appropriate strategy

First, the dropper retrieves the name of any antivirus that may be running on the machine by executing the Windows Management Instrumentation (WMI) request “SELECT * FROM AntiVirusProduct” and fetching the “displayName” field from the result. If an tag exists in the configuration file with a “NAME” attribute matching the name of an installed antivirus product, it will be set as the execution strategy. In this case, four antivirus products have a defined strategy.

If no strategy is found for the running antivirus, or if no antivirus is protecting the computer, the default strategy described in the tag’s attributes will be applied. Alternatively, if a file named “strategy.xml” is present in the dropper’s folder, it will override the strategy from the configuration file.

Possible Moves

A strategy is a set of attributes that influences both the dropper and the payload execution. Some of these attributes define *how* to realize certain actions, whereas the others define *whether* to perform certain actions. The following array describes the various “moves” offered by these attributes.

Attribute	Attribute Purpose	Possible Value	Value Meaning
RUNKEY	Defines how the dropper will interact with the Windows registry in order to be persistent on the machine	API	Calls to Windows API functions (RegOpenKeyEx, RegQueryValueEx…)
		BAT	Execution of a batch file containing “reg” commands
		REG	Execution of “reg” commands in a command prompt process
		WMI	Calls to methods of the StdRegProv WMI class
AUTODEL	Defines how the dropper will remove itself from the machine after its execution	DEL	Execution of a command line in a command prompt process
		API	Call to MoveFileEx API function to delete the dropper during the next restart of the system
		WMI	Execution of a command line in a command prompt process created through the Create method of the Win32_Process WMI class
INJECTION	Defines whether the dropper and the payload will inject their code into a new process, or execute it in the initial process	YES/NO	N/A
SAFENOTIF	Defines whether or not the payload will contact the C&C server	YES/NO	N/A

Attribute	Attribute Purpose	Possible Value	Value Meaning
SERVICE	Likely defines how to interact with Windows services, but the code managing this attribute is missing in these Casper samples	API	N/A
		SC	N/A
ESCAPE	Defines whether the dropper will execute normally, or simply exit	YES/NO	N/A
SCHEDULER	Unknown. The code managing this attribute is missing in these Casper samples	CMD	N/A

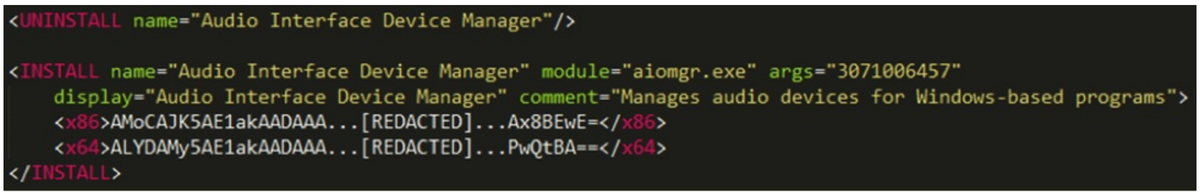
The possibilities offered by this tag show that Casper's authors have acquired an in-depth knowledge of behavioral detections in certain antivirus products.

For example, process injection will only happen on machines with none of the four defined antiviruses running, since in such a case the "INJECTION" attribute will be set to "NO". Interestingly, three antiviruses have the "ESCAPE" attribute set to "YES", which means the dropper will simply uninstall itself in their presence **without** deploying Casper's payload.

As the list of tags is pretty short, we can speculate that these are the antiviruses Casper's authors expect to find on their targets. For the record, the "VERSION" attribute present in one tag is actually never used in the code, but it still indicates the intention to distinguish different versions of the same antivirus product. We very rarely see this level of precision employed in malware in order to bypass antivirus.

Time To Drop The Payload

In the event that the "ESCAPE" attribute is set to "NO" in the chosen strategy – as is the case with the default strategy – the dropper will then execute the commands provided in the form of XML tags in the configuration file, as shown in Figure 2.



```
<UNINSTALL name="Audio Interface Device Manager"/>

<INSTALL name="Audio Interface Device Manager" module="aiomgr.exe" args="3071006457"
    display="Audio Interface Device Manager" comment="Manages audio devices for Windows-based programs">
    <x86>AMoCAJK5AE1akAADAAA...[REDACTED]...Ax8BEwE=</x86>
    <x64>ALYDAMy5AE1akAADAAA...[REDACTED]...PwQtBA==</x64>
</INSTALL>
```

Figure 2 – Casper Dropper's Commands

Uninstalling previous versions

The first command instructs the dropper to remove other Casper instances that could possibly be running on the system. The corresponding tag comes with a "name" attribute, which will be prefixed with the BIOS constructor name retrieved from the

Windows registry (Intel, NEC…) before being used as an identifier. This prefixing is likely meant to avoid drawing the user's attention if he or she happened to notice the identifier.

The program is uninstalled in two steps, each step addressing different methods of persistence employed by Casper:

- If it exists, the scheduled task whose name matches the identifier is removed from the system
- If it exists, the application registered with the identifier in the Windows registry is removed from the system

Payload installation

The payload installation is then directed by the tag, which provides two versions of the payload, one for 32-bit machines () and another one for 64-bit machines ().

The attributes of the tag will then be used by one of the two installation methods previously mentioned. If the operating system is Windows 7 or newer, persistence will be set through a scheduled task; otherwise it will be set through the Windows registry key

"HKLM\Software\Microsoft\Windows\CurrentVersion\Run".

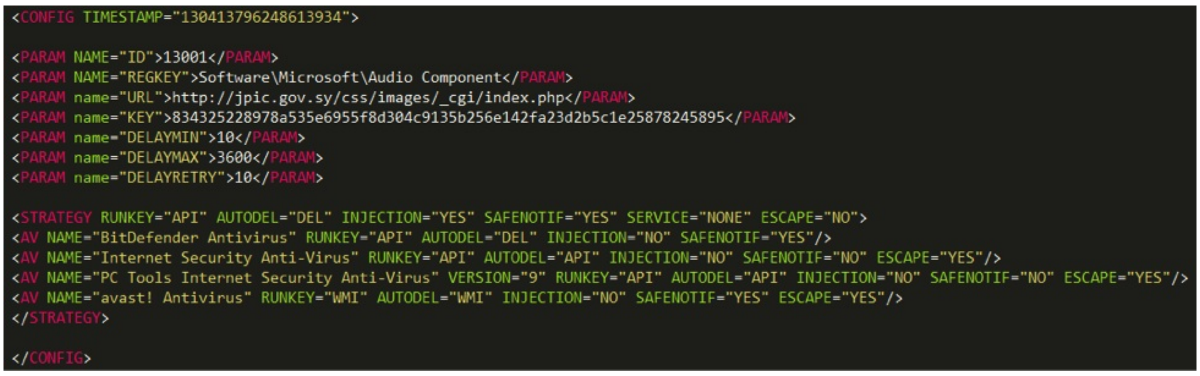
The tag provides an argument to give to the payload. The exact value of the argument is critical to the "correct" execution of the payload. The actual verification in the payload is subtle: the argument is used in a custom algorithm to find library functions in memory. Unless the value is correct, the addresses of these library functions will be wrong, resulting in a random-looking crash of the payload.

Dropper cleans itself

Before terminating its execution, the dropper removes itself from the system, using the method defined in the AUTODEL attribute. It should be noted that the payload is not launched at this moment: it will be run only at the next startup thanks to the previous persistence method.

Payload

Similarly to the dropper, Casper payload's execution is based on an XML configuration file decrypted at run-time, and shown in Figure 3.



```
<CONFIG TIMESTAMP="130413796248613934">

<PARAM NAME="ID">13001</PARAM>
<PARAM NAME="REGKEY">Software\Microsoft\Audio Component</PARAM>
<PARAM name="URL">http://jpic.gov.sy/css/images/_cgi/index.php</PARAM>
<PARAM name="KEY">834325228978a535e6955f8d304c9135b256e142fa23d2b5c1e25878245895</PARAM>
<PARAM name="DELAYMIN">10</PARAM>
<PARAM name="DELAYMAX">3600</PARAM>
<PARAM name="DELAYRETRY">10</PARAM>

<STRATEGY RUNKEY="API" AUTODEL="DEL" INJECTION="YES" SAFENOTIF="YES" SERVICE="NONE" ESCAPE="NO">
<AV NAME="BitDefender Antivirus" RUNKEY="API" AUTODEL="DEL" INJECTION="NO" SAFENOTIF="YES"/>
<AV NAME="Internet Security Anti-Virus" RUNKEY="API" AUTODEL="API" INJECTION="NO" SAFENOTIF="NO" ESCAPE="YES"/>
<AV NAME="PC Tools Internet Security Anti-Virus" VERSION="9" RUNKEY="API" AUTODEL="API" INJECTION="NO" SAFENOTIF="NO" ESCAPE="YES"/>
<AV NAME="avast! Antivirus" RUNKEY="WMI" AUTODEL="WMI" INJECTION="NO" SAFENOTIF="YES" ESCAPE="YES"/>
</STRATEGY>

</CONFIG>
```

Figure 3 – Casper's Payload Configuration File

This configuration file starts with a timestamp, which corresponds to **Monday, the 7th April 2014 at 21:27:05 GMT**. Therefore, the compilation timestamps – set to 2010 – have very likely been forged.

A series of tags will then control the payload's behavior, as described in the following array.

attribute	Purpose
ID	Unknown. It could be used to distinguish operations, as the value is the same in the two payloads hosted on "jpic.gov.sy".
REGKEY	Path in the Windows registry that will be used as storage area
URL	C&C server's URL
KEY	Cryptographic key for the communications with the C&C server
DELAYMIN DELAYMAX DELAYRETRY	Timers to configure the frequency of the contacts with the C&C server

The payload then generates a unique identifier for the machine and inserts it at the end of the configuration in a tag. Finally, the configuration is RC4-encrypted and stored in the Windows registry.

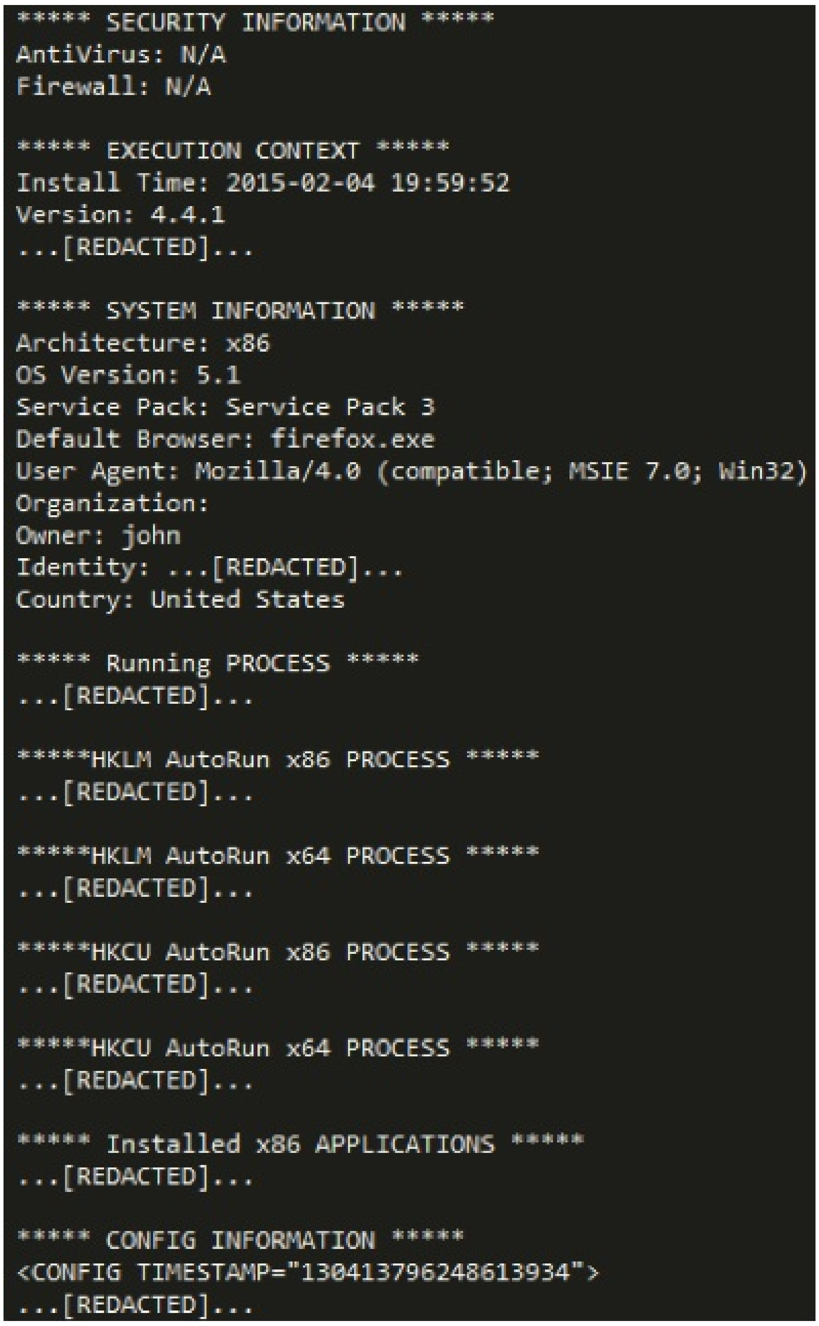
The code handling the configuration shows certain capabilities not exploited in these Casper samples, for example a TIMETOLIVE attribute to plan the termination of Casper after a certain amount of time, or a DELAYED_START attribute to wait before interacting with the system.

Finally, the payload's configuration contains the exact same as the dropper.

Report to C&C

During its first execution, Casper's payload executes the following XML file:

The handler of the "SYSINFO" command retrieves information about the system and builds a report containing several sections, as shown in Figure 4.



```
***** SECURITY INFORMATION *****
AntiVirus: N/A
Firewall: N/A

***** EXECUTION CONTEXT *****
Install Time: 2015-02-04 19:59:52
Version: 4.4.1
...[REDACTED]...

***** SYSTEM INFORMATION *****
Architecture: x86
OS Version: 5.1
Service Pack: Service Pack 3
Default Browser: firefox.exe
User Agent: Mozilla/4.0 (compatible; MSIE 7.0; Win32)
Organization:
Owner: john
Identity: ...[REDACTED]...
Country: United States

***** Running PROCESS *****
...[REDACTED]...

*****HKLM AutoRun x86 PROCESS *****
...[REDACTED]...

*****HKLM AutoRun x64 PROCESS *****
...[REDACTED]...

*****HKCU AutoRun x86 PROCESS *****
...[REDACTED]...

*****HKCU AutoRun x64 PROCESS *****
...[REDACTED]...

***** Installed x86 APPLICATIONS *****
...[REDACTED]...

***** CONFIG INFORMATION *****
<CONFIG TIMESTAMP="130413796248613934">
...[REDACTED]...
```

Figure 4 – SYSINFO Command's Result

The titles of the report sections are self-explanatory. Interestingly, the version of the malware is clearly mentioned: **4.4.1**. This report is then base64-encoded and sent to the C&C server in the body of an HTTP POST request. It will also be written into a temporary file named "perfaudio.dat".

The network request will also have a cookie named "PREF" filled with the concatenation of the machine UID, the configuration ID, the version of Casper and the hardcoded character "R", all base64-encoded.

C&C's possible answers

Due to the C&C being down at the time of the investigation we can only speculate on the rest of the execution based on Casper's known capabilities.

At this point, the binary regularly contacts the C&C server with a cookie similar to the one in the SYSINFO request, but this time with "G" as the hardcoded character instead of "R". Our analysis of the binary reveals that the server can then send back a PNG image – with the correct header and format for a PNG file — from which a XML command file will be decrypted and executed.

In addition to the "SYSINFO" command, Casper can handle tags with the following values:

- "EXEC" to execute a program on the machine from its local path
- "SYSTEM" to execute commands in a Windows command prompt

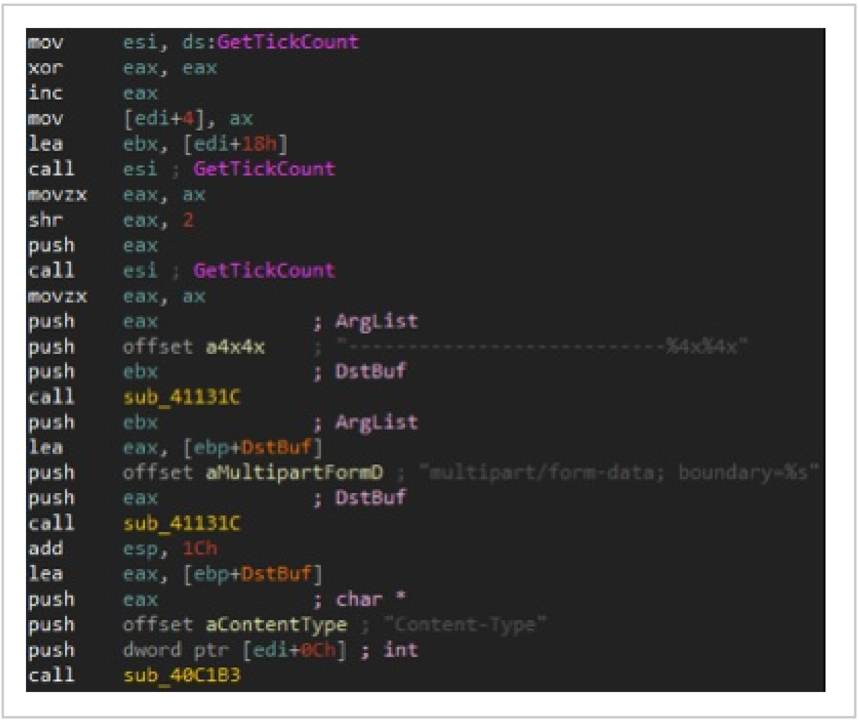
Finally, Casper can also handle tags, whose content is a Windows executable to deploy on the machine.

How Does Casper Relate to the Other Cartoons?

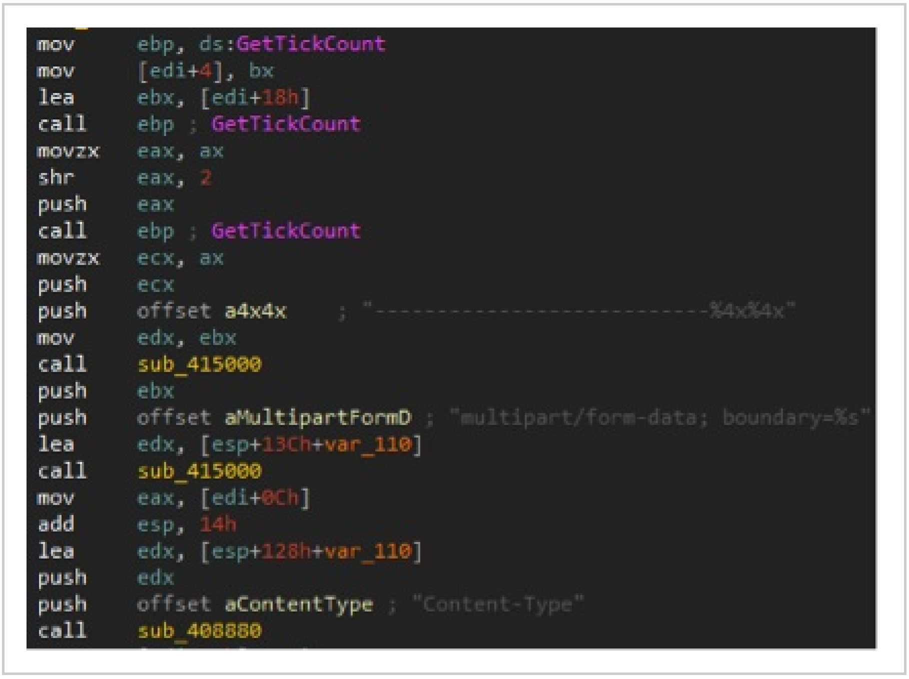
Our best chance of establishing that the same developers are behind Bunny, Babar and Casper is to identify unusual code or algorithms shared between these various programs. In our comparison we also take into account the so-called "NBOT" malware (also known as the "TFC" malware), whose link with Babar and Bunny was established by Marion Marschalek in her Babar report. Here is a **non-exhaustive** list of such shared features we observed:

Casper hides its calls to API functions by using a hash calculated from the functions' names, rather than the names themselves. The hashing algorithm is a combination of rotate-left (ROL) of 7 bits and exclusive-or (XOR) operations. NBOT uses the exact same algorithm for the same purpose, whereas Babar hides its API calls in a similar manner but with a different algorithm.

- Casper fetches information about the running antivirus in a way similar to Bunny, Babar, and NBOT, namely through the same WMI request. Moreover, all these malwares compute the SHA-256 hash of the first word of the antivirus name, although in Casper it is actually never used.
- Casper generates delimiters for its HTTP requests by filling a specific format string with the results of calls to the GetTickCount API function. The same code is present in some NBOT samples, as shown in the following array.



```
mov     esi, ds:GetTickCount
xor     eax, eax
inc     eax
mov     [edi+4], ax
lea     ebx, [edi+18h]
call    esi ; GetTickCount
movzx   eax, ax
shr     eax, 2
push    eax
call    esi ; GetTickCount
movzx   eax, ax
push    eax             ; ArgList
push    offset a4x4x    ; "--------------------------%4x%4x"
push    ebx             ; DstBuf
call    sub_41131C
push    ebx             ; ArgList
lea     eax, [ebp+DstBuf]
push    offset aMultipartFormD ; "multipart/form-data; boundary=%s"
push    eax             ; DstBuf
call    sub_41131C
add     esp, 1Ch
lea     eax, [ebp+DstBuf]
push    eax             ; char *
push    offset aContentType ; "Content-Type"
push    dword ptr [edi+0Ch] ; int
call    sub_40C1B3
```



```
mov     ebp, ds:GetTickCount
mov     [edi+4], bx
lea     ebx, [edi+18h]
call    ebp ; GetTickCount
movzx   eax, ax
shr     eax, 2
push    eax
call    ebp ; GetTickCount
movzx   ecx, ax
push    ecx
push    offset a4x4x    ; "--------------------------%4x%4x"
mov     edx, ebx
call    sub_415000
push    ebx
push    offset aMultipartFormD ; "multipart/form-data; boundary=%s"
lea     edx, [esp+13Ch+var_110]
call    sub_415000
mov     eax, [edi+0Ch]
add     esp, 14h
lea     edx, [esp+128h+var_110]
push    edx
push    offset aContentType ; "Content-Type"
call    sub_408880
```

Casper removes its dropper by executing a Windows command created from the following format string:

```
cmd.exe /C FOR /L %%i IN (1,1,%d) DO IF EXIST “%hs” (DEL “%hs” & SYSTEMINFO) ELSE
EXIT
```

In some NBOT samples we can find the following similar syntax:

```
cmd.exe /C FOR /L %%i IN (1,1,%d) DO IF EXIST “%s” (DEL “%s” & PING 127.0.0.1 -n 3) ELSE
EXIT
```

Casper uses an “ID” value set to “13001”, whereas Babar samples contain an ID of “12075-01”. Also, the malware discovered in 2009 by the CSEC possesses an ID of “08184” (slide 8 of the CSEC slides). This similar format, and the increasing value in decimal, could indicate a familial link.

None of these signs alone is enough to establish a strong link but **all the shared features together make us assess with high confidence that Bunny, Babar, NBOT and Casper were all developed by the same organization.**

Victimology

According to our telemetry data, all the people targeted during this operation were located in Syria. These targets may have been the visitors of the “jpic.gov.sy” website — Syrian citizens who want to file a complaint. In this case they could have been redirected to the exploits from a legitimate page of this website.

But we were actually unable to determine if this were indeed the case. In other words, it is just as likely that the targets have been redirected to the exploits from another location, for example from a hacked legitimate website or from a link in an email. What is known for sure is that the exploits, the Casper binaries and the C&C component were all hosted on this website’s server.

This leads us to a second hypothesis: the “jpic.gov.sy” website could have been hacked to serve as a storage area. This would have at least two advantages for the attackers: firstly, hosting the files on a Syrian server can make them more easily accessible from Syria, a country whose Internet connection to the outside world has been unstable since the beginning of the civil war, as shown in Google Transparency Report. Secondly, it would mislead attribution efforts by raising suspicion against the Syrian government.

Conclusion

As previously explained, we are confident that the same group developed Bunny, Babar and Casper. The detailed analysis of Babar in the CSEC slides from 2009 indicates this group is not a newcomer to the espionage business. The use of zero-day exploits is another indication that Casper’s operators belong to a powerful organization. Finally, the narrow targeting of people in Syria shows a likely interest in geopolitics.

Nevertheless, we did not find any evidence in Casper itself to point a finger at a specific country. In particular, no signs of French origin, as suggested by CSEC for Babar, were found in the binaries.

Hashes

SHA1	Note	ESET Detection Name
75BF51709B913FDB4086DF78D84C099418F0F449	DLL Dropper	Win32/ProxyBot.B
7F266A5E959BEF9798A08E791E22DF4E1DEA9ED5	DLL Dropper	Win32/ProxyBot.B

SHA1	Note	ESET Detection Name
E4CC35792A48123E71A2C7B6AA904006343A157A	Executable Dropper	Win32/ProxyBot.B
F4C39EDDEF1C7D99283C7303C1835E99D8E498B0	X86 Executable Payload	Win32/ProxyBot.B
C2CE95256206E0EBC98E237FB73B68AC69843DD5	X64 Executable Payload	Win32/ProxyBot.A

Indicators of Compromise

Indicator	Value
Dropper's file name	domcommon.exe
Payload's file name	aiomgr.exe
C&C URLs	hXXp://jpic.gov.sy/css/images/_cgi/index.php
Mutex name	{4216567A-4512-9825-7745F856}
Key for configuration decryption	7B 4B 59 DE 37 4A 42 26 59 98 63 C6 2D 0F 57 40
Temporary file name	perfaudio.dat

Image: PAISAN HOMHUAN / Shutterstock.com

Joan Calvet 5 Mar 2015 - 01:55PM