King of Spam: Festi Botnet Analysis

Eugene Rodionov

Malware Researcher

Aleksandr Matrosov

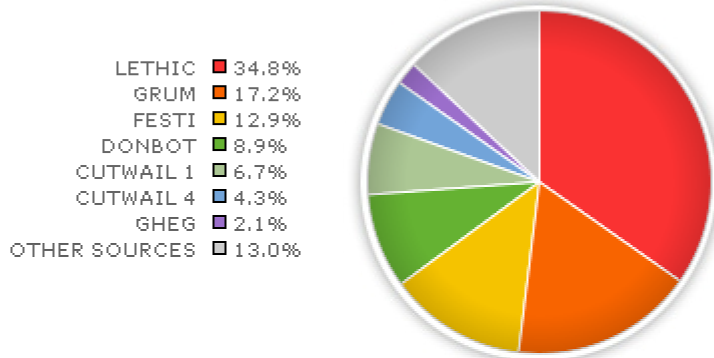
Security Intelligence Team Lead

Table of Contents

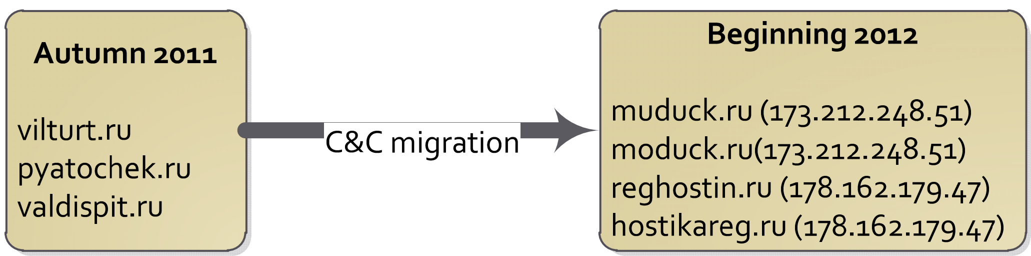
Introduction

The botnet Win32/Festi has been in business since the autumn of 2009 and is currently one of the most powerful and active botnets for sending spam and performing DDoS attacks. The bot consists of two parts: the dropper and the main module, the kernel-mode driver, which is detected by ESET as Win32/Rootki.Festi.

In 2009 and early 2010 the bot was leased out for spam sending but was then restricted to major spam partners. According to statistics from M86 Security Labs, shown on the right, Win32/Festi is one of the three most active spam botnets in the world.



In the autumn of 2011 the botnet migrated its C&C (Command & Control) servers to new domain names. All the previously-used domains are still alive and are kept in reserve in case the primary domain/servers don't respond.

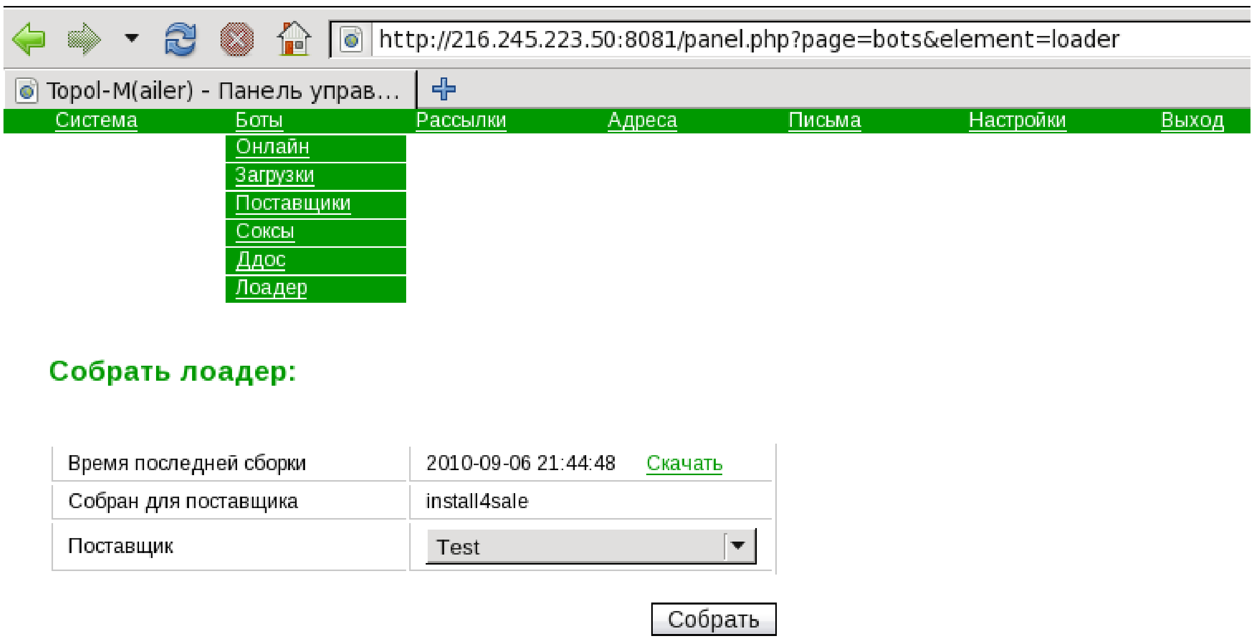


The botnet periodically migrates to new hosting and domain names in order to decrease the rate at which it is detected using C&C URLs and corresponding IP addresses. There are only C&C domain names inside the bot's binary with no IP addresses.

The previous versions of the bot communicated with C&C servers over HTTP (Hypertext Transfer Protocol) by encrypting POST requests. At the beginning of 2012 an updated version of the bot employed a new communication protocol which is capable of bypassing IPS and IDS systems operating at the network layer. In this report we analyze the latest version of the bot which appeared in February, and is detected by ESET products as Win32/Rootkit.Agent.NVG.

Investigation

In 2010 we managed to get access to one of the botnet's administrative consoles. Here is how the process of assembling the downloader for distributing the dropper looks:



In our recent analysis we noticed that one of the most recent droppers we received was built in March of 2012, as shown on the right. This suggests that the dropper is being updated frequently.

Field Name	Data Value	Description
Machine	014Ch	i386®
Number of Sections	0007h	
Time Date Stamp	4F6ACE0Ch	22/03/2012 07:00:28
Pointer to Symbol Table	00000000h	
Number of Symbols	00000000h	
Size of Optional Header	00E0h	
Characteristics	0103h	
Magic	010Bh	PE32
Linker Version	0008h	8.0

Win32/Rootkit.Agent.NVG is distributed mainly through a PPI (Pay-Per-Install) scheme. The dropper is specially optimized for propagation using this approach. The main intention of the dropper is to install into the system the kernel-mode driver which implements the main logic of the bot.

As you can see on the right, at the beginning of April the kernel-mode driver was also rebuilt. However, there were no significant changes to its functionality.

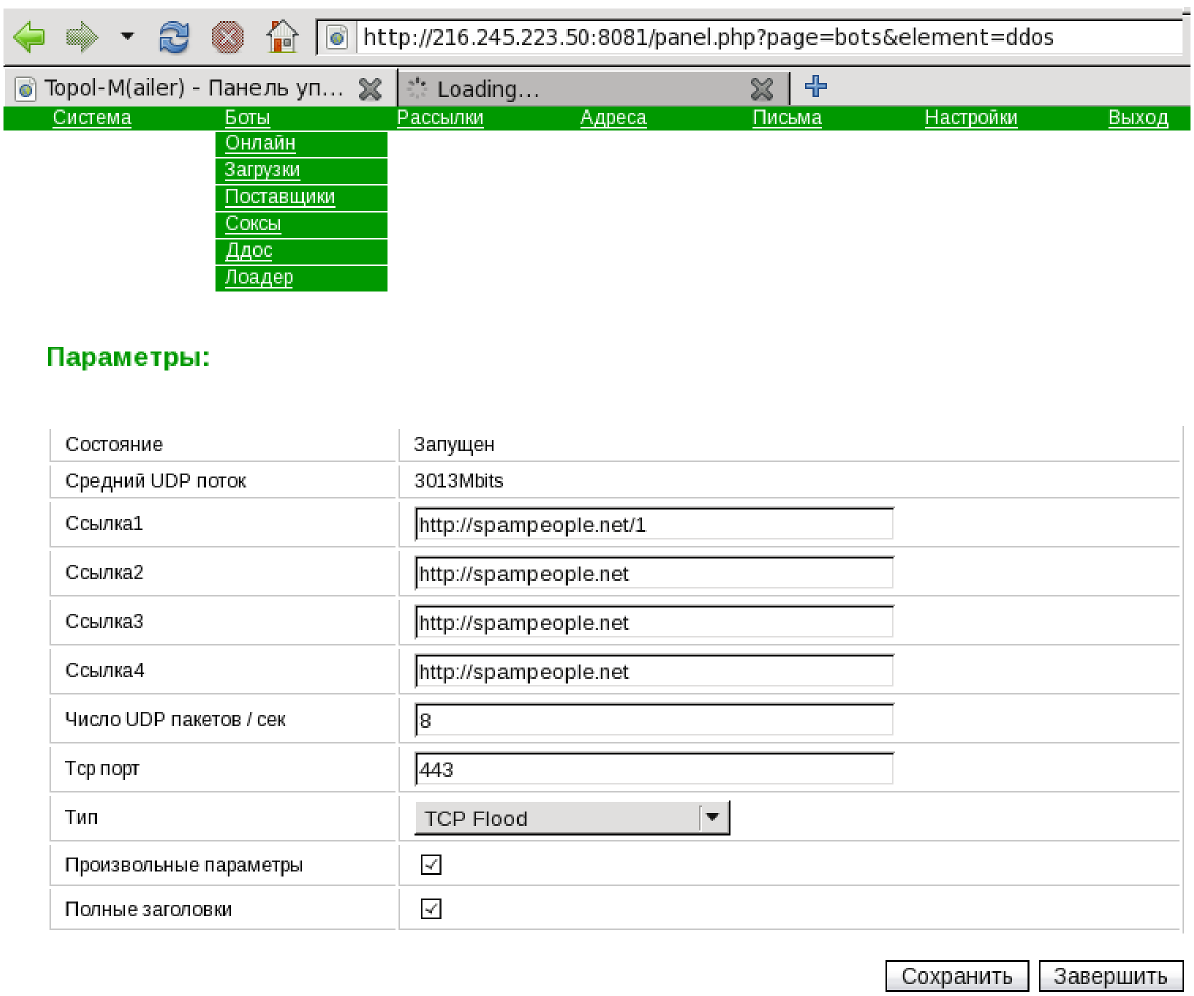
Field Name	Data Value	Description
Machine	014Ch	i386®
Number of Sections	0007h	
Time Date Stamp	4F7ACBC4h	03/04/2012 10:07:00
Pointer to Symbol Table	00000000h	
Number of Symbols	00000000h	
Size of Optional Header	00E0h	
Characteristics	0102h	
Magic	010Bh	PE32
Linker Version	0008h	8.0

Another interesting feature of the kernel-mode driver is this string constant, found in the binary, as shown in the following image:

```
                dd 2                    ; SEHandlerCount
                dd 53445352h, 0B501DD16h, 4987F879h, 0FFF14ABh, 0AED6E286h
                dd 1
aEEclipseBotnet db 'e:\eclipse\botnet\drivers\Bin\i386\kernel.pdb',0
                align 10h
___safe_se_handler_table dd rva sub_205D0
                                        ; DATA XREF: .text:0001C6E8↑o
                dd rva loc_2080B
                dd 3 dup(0)
dword_1C754     dd 0FF8B0000h           ; DATA XREF: .data:00020D84↓o
```

This means that it is highly probable the bot was developed using the Eclipse IDE. This theory is supported by the detection of object code that does not appear to have been produced by a Microsoft compiler. Probably some part of the bot was compiled and tested separately from other components then linked to the bot's main module later.

The bot implements a very powerful DoS (Denial of Service) engine, which is examined in further detail in this report. Here you can see how DoS tasks are handled by Win32/Festi.



We have not seen the DoS bot being leased out and at the present time it is used only for targeted attacks. For instance, one such attack performed by this bot in 2010 targeted Assist, the company that was processing payments for Aeroflot, Russia's largest airline. (For more information see Brian Krebs on Security: http://krebsonsecurity.com/2011/06/financial-mogul-linked-to-ddos-attacks.)

Field Name	Data Value	Description
Machine	014Ch	i386
Number of Sections	0006h	
Time Date Stamp	4F857C87h	11/04/2012 12:43:51
Pointer to Symbol Table	00000000h	
Number of Symbols	00000000h	
Size of Optional Header	00E0h	
Characteristics	0102h	
Magic	010Bh	PE32
Linker Version	0008h	8.0

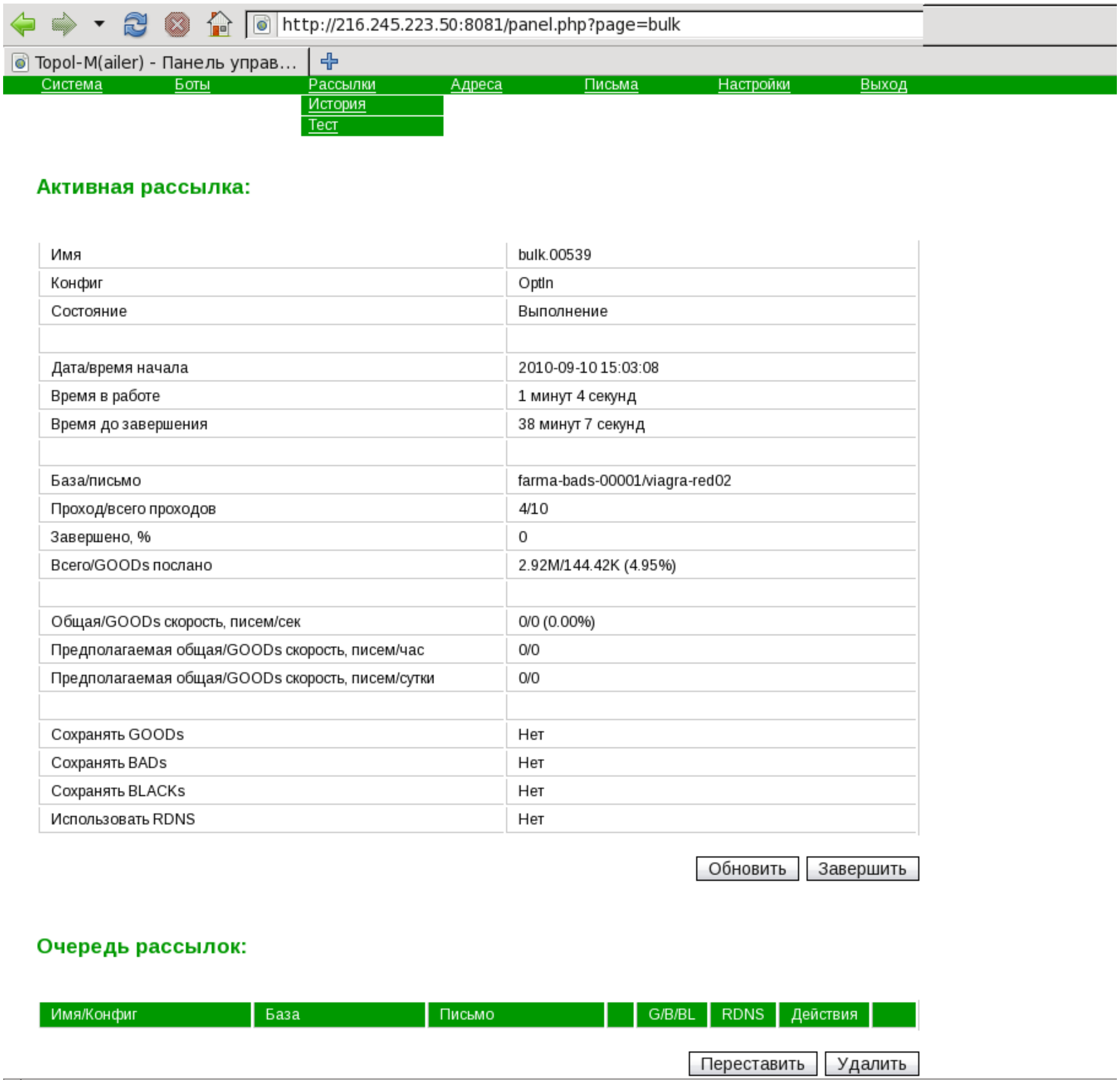
The DDoS plugin that we analyze in the report was updated in April of 2012.

Here you can see what a network dump of an HTTP flood attack by this particular bot looks like.

Src IP	Dest IP	Src Port	Dest Port
173.21...	10.0.2.15	https	1036
10.0.2....	2.101.151.21	1037	http
10.0.2....	86.198.12.13	1038	http
10.0.2....	212.225.171.108	1039	http
10.0.2....	188.3.32.59	1040	http
10.0.2....	2.101.151.21	1041	http
10.0.2....	2.101.151.21	1042	http
10.0.2....	94.189.132.59	1043	http
10.0.2....	188.3.32.59	1044	http
10.0.2....	94.189.132.59	1045	http
10.0.2....	24.212.208.233	1046	http
10.0.2....	86.198.12.13	1047	http
10.0.2....	189.169.133.71	1048	http
10.0.2....	24.212.208.233	1049	http
10.0.2....	189.169.133.71	1050	http
10.0.2....	90.49.232.70	1051	http
10.0.2....	2.101.151.21	1052	http
10.0.2....	188.3.32.59	1053	http
10.0.2....	189.169.133.71	1054	http
10.0.2....	189.157.91.179	1055	http
10.0.2....	188.3.32.59	1056	http
10.0.2....	94.189.132.59	1057	http
10.0.2....	86.198.12.13	1058	http
10.0.2....	189.169.133.71	1059	http
10.0.2....	188.3.32.59	1060	http
10.0.2....	94.189.132.59	1061	http
10.0.2....	86.198.12.13	1062	http
10.0.2....	189.157.91.179	1063	http
10.0.2....	189.157.91.179	1064	http
10.0.2....	188.3.32.59	1065	http
10.0.2....	24.212.208.233	1066	http
10.0.2....	24.212.208.233	1067	http
10.0.2....	94.189.132.59	1068	http
10.0.2....	2.101.151.21	1069	http
10.0.2....	189.169.133.71	1070	http
10.0.2....	24.212.208.233	1071	http

The main purpose of the Win32/Festi bot is the sending of spam. The screen on the following page shows how scheduling of spam jobs is distributed among the bots.

Figure 1: How scheduling of spam jobs is distributed among the bots.



On the following page you can see what a spam template used by the Win32/Festi botnet looks like:

Figure 2: Spam template used by the Win32/Festi botnet.

```
220 smtp.mail.ru ESMTP
EHLO cgrn.com
250-smtp.mail.ru
250-PIPELINING
250-8BITMIME
250-XCLIENT NAME HELO
250-XFORWARD NAME ADDR PROTO HELO
250-ENHANCEDSTATUSCODES
250
MAIL FROM:<cecilija.lihachjova@gesundheit.ru>
250 2.1.0 Ok
RCPT TO:<intruderss450@atmofresh.ru>
250 2.1.5 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
Received: from gesundheit.ru ([89.110.144.102]) by cgrn.com with SMTP; Sat, 05 May 2012 05:19:51 +0400
Message-ID: <000e01cd2a5d$2b46d580$621d5058@gesundheit.ru>
From: "Cecilija Lihachjova" <cecilija.lihachjova@gesundheit.ru>
To: <intruderss450@atmofresh.ru>
Subject: =?windows-1251?B?6Pno8uUg8err40Q/?=
Date: Sat, 05 May 2012 05:18:12 +0400
MIME-Version: 1.0
Content-Type: text/plain;
        charset="windows-1251"
Content-Transfer-Encoding: 8bit
X-Priority: 3
X-MSMail-Priority: Normal
X-Mailer: Microsoft Outlook Express 5.00.2615.200
X-MimeOLE: Produced By Microsoft MimeOLE V5.00.2615.200

СКЛАДЫ В АРЕНДУ г. Москва, ЮАО, м. Кантемировская.

Удобные отдельные помещения:
250 кв.м., 530 кв.м.

Склад сухой,  на 1 этаже с пандусом.
Территории административно-складского комплекса круглосуточно охраняется.

При необходимости предоставим:

Ответственное хранение грузов от 10 кв.м.
Офисные помещения в аренду от 25 кв.м.,

Погрузо-разгрузочные работы,
Ведение складского учета.

Подробная информация по тел.:
(495) 798-5505, 726-1777, 514-0475


.
250 2.0.0 Ok
QUIT
221 Bye
```

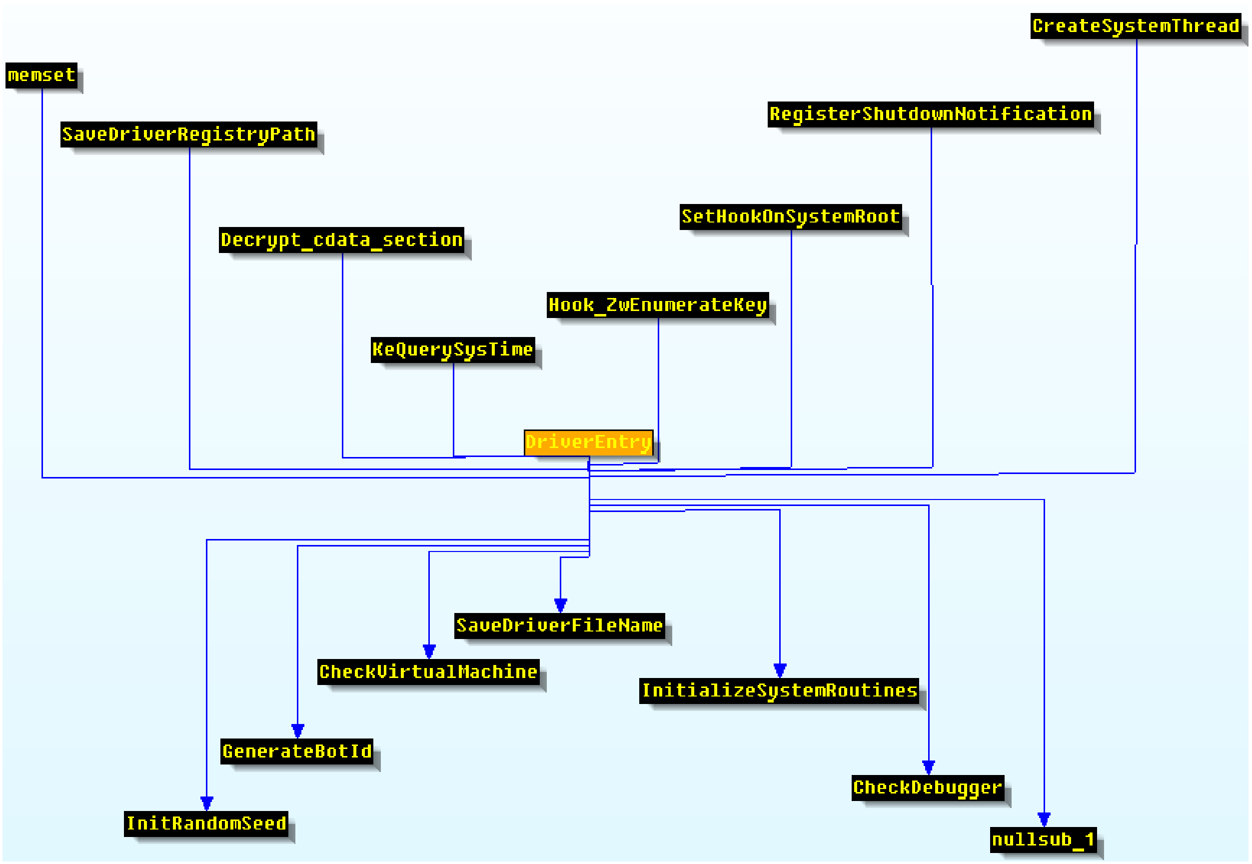
The plugin for sending spam was most recently updated in October of 2011.

One of the most interesting aspects of researching Win32/Festi is reverse engineering of the main module of the kernel-mode driver which implements a framework for plugins performing dedicated tasks. We examine this architecture in the next section of our analysis.

Field Name	Data Value	Description
Machine	014Ch	i386®
Number of Sections	0007h	
Time Date Stamp	4E9D0E1Ah	18/10/2011 05:26:50
Pointer to Symbol Table	00000000h	
Number of Symbols	00000000h	
Size of Optional Header	00E0h	
Characteristics	0102h	
Magic	010Bh	PE32
Linker Version	0008h	8.0

Win32/Festi architecture

The malware consists of a single kernel-mode driver which is installed into the victim systems by its dropper. The kernel-mode component is registered in the system as SYSTEM_START kernel-mode driver with a randomly generated name. As a result, it is loaded and receives control in the *IoInitSystem* routine during the system initialization process. Here you can see the driver's entry point call graph.

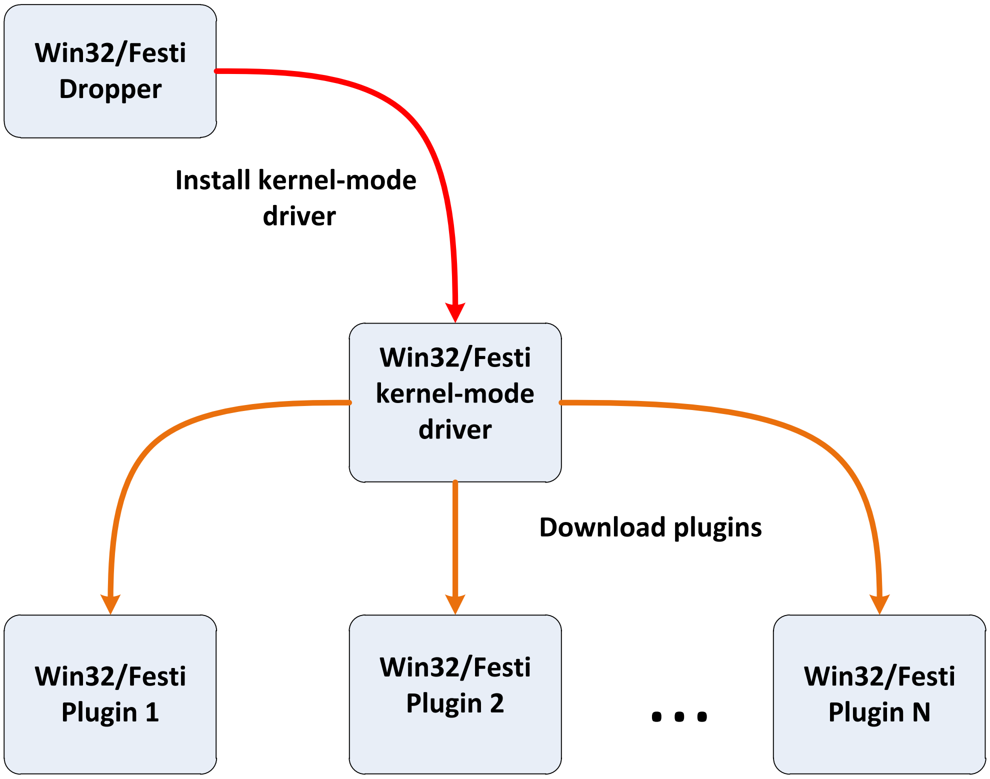


The kernel-mode driver implements backdoor functionality and is capable of:

1) Updating configuration data from C&C;
2) Downloading additional dedicated plugins.

The kernel-mode driver periodically contacts the C&C server and requests plugins and configuration information. The downloaded plugins perform the bot's main tasks, such as sending spam. The plugins are kernel-mode drivers which aren't saved on any storage device in the system and are volatile in memory. Thus, when the infected computer is switched off or rebooted the plugins have vanished from system memory. This makes forensic analysis of the malware significantly harder since the only file stored on the hard drive is the main kernel-mode driver, and this contains neither the payload nor information regarding which sites to attack or target with spam.

Each plugin is dedicated to performing certain kinds of job such as performing DDoS attacks against a specified network resource or sending spam. The plugins communicate with the main driver through a well-defined interface. The next figure illustrates how the bot penetrates into the system and performs malicious activity.



Configuration information

The bot's configuration information is hardcoded into the driver's binary. If we look at section table of kernel-mode driver we notice a writable section with the name *".cdata"*:

Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations N...	Linenumber...	Characteristics
Byte[8]	Dword	Dword	Dword	Dword	Dword	Dword	Word	Word	Dword
.text	00003B27	00001000	00003C00	00000400	00000000	00000000	0000	0000	68000020
.rdata	000007C8	00005000	00000800	00004000	00000000	00000000	0000	0000	48000040
.data	00001098	00006000	00001000	00004800	00000000	00000000	0000	0000	C8000040
pagecode	0000A84C	00008000	0000AA00	00005800	00000000	00000000	0000	0000	C8000040
.cdata	00000582	00013000	00000600	00010200	00000000	00000000	0000	0000	C8000040
INIT	000008D8	00014000	00000A00	00010800	00000000	00000000	0000	0000	E2000020
.reloc	00000992	00015000	00000A00	00011200	00000000	00000000	0000	0000	42000040

The .cdata section contains encrypted configuration information which specifies:

- URLs of C&C servers;
- Key to encrypt data transmitted between the bot and C&C;
- Bot version information and so on.

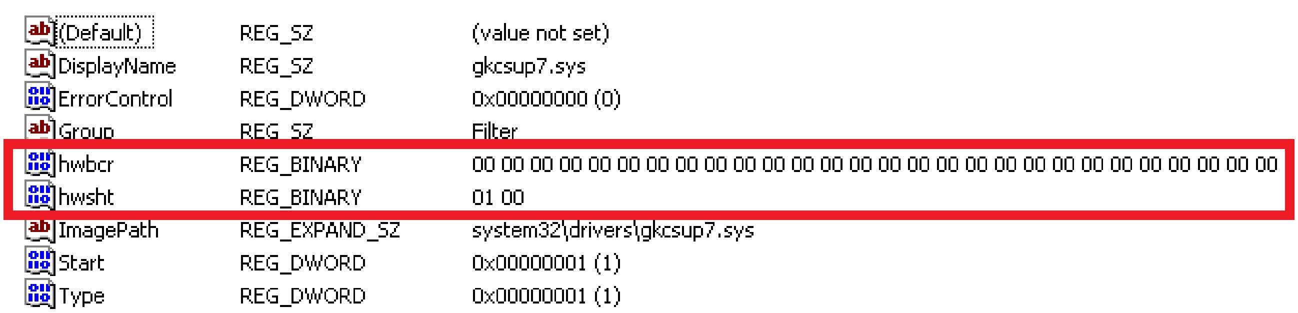
The section also contains some strings used by the malware to perform various system-specific operations. Encrypting these strings helps the bot to evade AV software. The following table lists some of the strings as well the purpose they serve in the malware:

Table 1 – List of encrypted strings in bot's binary

String	Purpose
\Device\Tcp \Device\Udp	Sending/receiving data over the network
\REGISTRY\MACHINE\SYSTEM\CurrentControlSet\Services\SharedAccess\Parameters\FirewallPolicy\StandardProfile\GloballyOpenPorts\List	To disable local firewall
ZwDeleteFile; ZwQueryInformationFile; ZwLoadDriver; KdDebuggerEnabled, etc.	Using corresponding exported symbols

To be able to access configuration data as soon as the driver is loaded by the system, Win32/Festi decrypts the ".cdata" section using the simple encryption/decryption algorithm presented below:

The bot also uses its driver's registry key (HKLM\System\CurrentControlSet\Services\Driver_Name) to store some information on the system and receive data from C&C in plain text. It keeps data in two binary registry key's values named respectively "*hwbcr*" and "*hwsht*" as shown here:



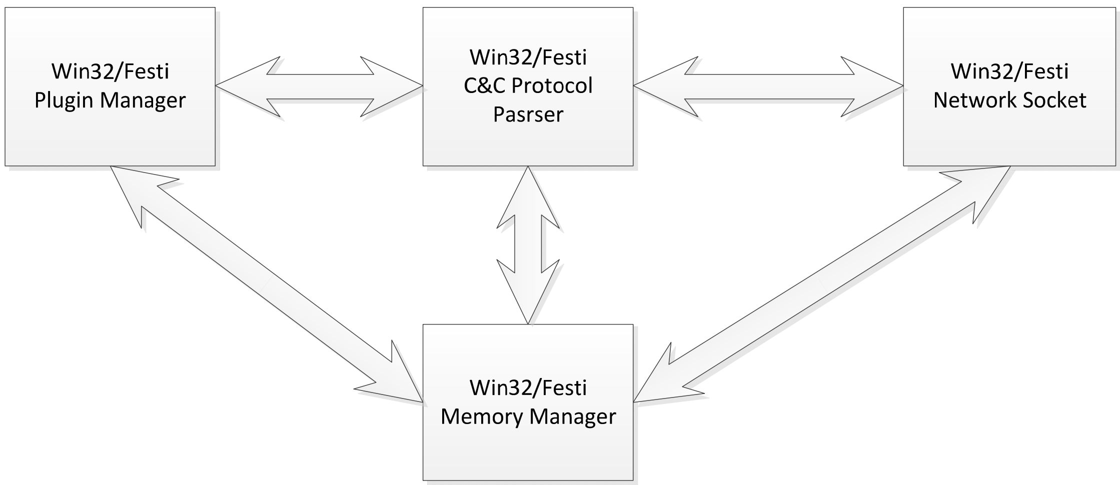
OOP Framework

One of the remarkable features of the bot is that its driver is developed using an object oriented programming (OOP) language and has a corresponding architecture. This is very unusual for kernel-mode drivers as they are typically written in plain C.

Here is the list of main components (classes) implemented by the malware:

- Memory manager – to allocate/release memory buffers;
- Network sockets – to send/receive data over the network;
- C&C protocol parser – to parse C&C messages and execute received commands;
- Plugin manager – to efficiently manage downloaded plugins;

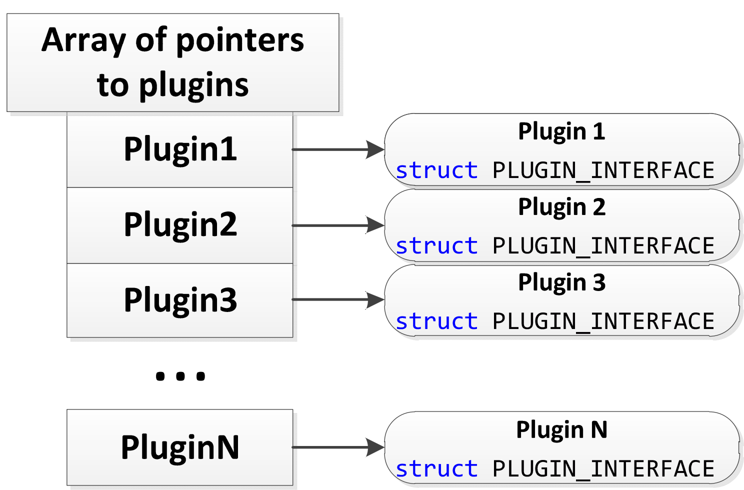
The interconnection of the components listed above is presented in the following figure. As we can see Memory Manager is the central component used throughout the bot.



The design principles of the malware make it extremely portable to other platforms like Linux, for instance. The system-specific code is isolated by the component's interface and may be easily changed to support other platforms. For instance, downloaded plugins that are dedicated to performing a specific task rely almost completely on the interfaces provided by the main module. They rarely use routines provided by the system to perform system-specific operations.

Managing plugins

To be able to manage downloaded plugins efficiently the bot maintains an array of pointers to a specially defined structure. The structure describes a plugin and provides the bot with specific entry points for plugins– routines responsible for handling data received from C&C.



Here you can see the layout of the structure describing the interface that a plugin should make available to the main module:

```
struct PLUGIN_INTERFACE
{
        // Initialize plugin
        PVOID Initialize;
        // Release plugin, perform cleanup operations
        PVOID Release;
        // Get plugin version information
        PVOID GetVersionInfo_1;
        // Get plugin version information
        PVOID GetVersionInfo_2;
        // Write plugin specific information into tcp stream
        PVOID WriteIntoTcpStream;
        // Read plugin specific information from tcp stream and parse data
        PVOID ReadFromTcpStream;
        // Reserved fields
        PVOID Reserved_1;
        PVOID Reserved_2;
};
```

When the bot transmits data to the C&C server it runs through the array of pointers to the plugin interface and executes the *WriteIntoTcpStream* routine of each registered plugin passing a pointer to a tcp stream object as a parameter. On receiving data from the C&C server the bot executes the plugins' *ReadFromTcpStream* routine, so that the registered plugins can get parameters and plugin-specific configuration information from the network stream. As a result the data sent over the network are structured as follows:

head of the message | | | | tail of the message

Message Header	Plugin1 Data	Plugin2 Data	...	Trailing Bytes

Built in plugins

When the bot is installed into the system, the main kernel-mode driver already contains the implementation of two built-in plugins, namely:

- The configuration information manager;
- The bot plugin manager.

Configuration manager

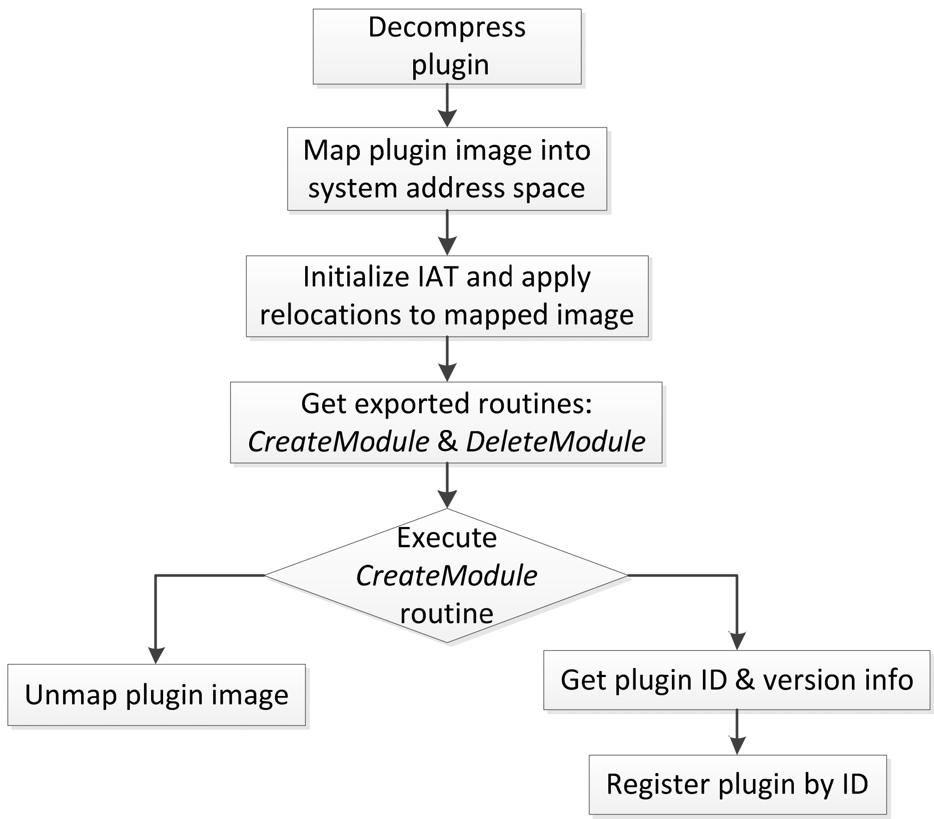
This plugin is responsible for requesting configuration information from the C&C server.

Plugin manager

The plugin manager is responsible for maintaining an array of downloaded plugins for the bot. It is able to load/unload a specific plugin onto the system when it receives a remote command. It receives compressed plugins from the C&C server. Each plugin is a DLL exporting two routines:

- *PLUGIN_INTERFACE *CreateModule(PVOID DriverInterfaces)*;
- *VOID DeleteModule()*.

The *CreateModule* routine is executed on plugin initialization and returns a pointer to the interface described above. It takes as a parameter a pointer to the set of interfaces provided by the main module. The plugin uses these interfaces to interact with the main module and C&C servers over the network. The *DeleteModule* routine is executed when the plugin is unloaded and is used to free all the previously allocated resources. in the figure below you can see a description of the algorithm for loading a downloaded plugin:



Network Communication

In order to communicate with C&C servers and perform some malicious activity Win32/Festi employs a custom network protocol which is protected against eavesdropping.

C&C Protocol description

The botnet uses client-server topology, meaning that there is a set of dedicated servers managing the botnet. In particular, in the course of investigation we managed to figure out that some C&C servers handle bots sending spam while other C&C servers manage bots performing DDoS attacks.

The Win32/Festi communication protocol consists of 2 phases:

- Initialization phase – obtaining addresses of C&C IP addresses;
- Work phase – requesting job description from C&C.

Initialization phase

During the initialization phase the malware obtains the IP addresses of the C&C so as to communicate with it. The bot binary contains URLs of C&C servers. The point here is that the malware manually resolves C&C IP address: namely, it constructs a UDP packet with the DNS name to resolve and sends the packet to either of two hosts : 8.8.8.8 or 8.8.4.4 at port 53 (DNS service). Both hosts are Google DNS servers. In reply they receive an IP address which they use in subsequent communication.

Work phase

The communication with the C&C is performed over TCP protocol. As shown on the figure below the message to be sent consists of:

- Message header;
- Plugin specific data;

Plugin-specific data consist of an array of “tag-value-term” entries:

- 16-bit integer specifying particular value.
- value – specific data which might be: byte, word, dword, null-terminated string, binary array;
- term – terminating word: 0xABDC, signifying the end of entry.

Tag	Value	0xABDC

The data are encrypted on sending with a simple encryption algorithm:

```
key = (0x17, 0xFB, 0x71,0x5C)
def decr_data(data):
    for ix in xrange(len(data)):
        data[ix] ^= key[ix % 4]
```

Bypassing security software and antiforensics

Win32/Festi uses some interesting techniques to defeat personal firewalls and HIPS systems installed on the infected machine, and to obstruct forensics.

Personal firewalls & HIPS

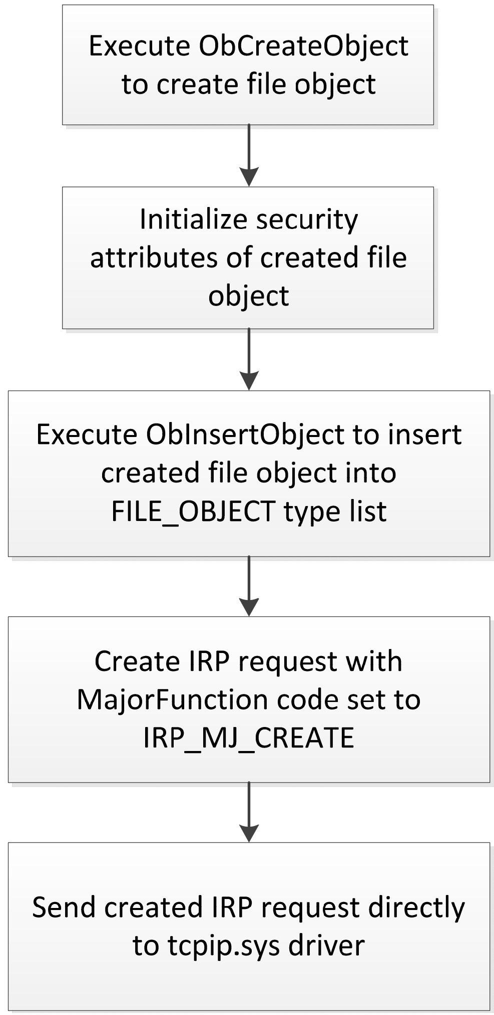
One of the interesting things about Win32/Festi is that it is able to bypass personal firewalls and HIPS systems installed on the infected machine. So as to be able to communicate over the network with C&C servers and send spam and perform DDoS attacks, it relies on a TCP/IP stack implemented in Microsoft Windows OS in kernel-mode. By doing it this way the malware is able to send TCP/UDP packets and at the same time is not burdened with constructing packets of these types manually (as is the case with NDIS drivers that operate at link layer).

In order to send/receive packets the malware opens *\Device\Tcp* or *\Device\Udp* devices depending on the protocol type being used. Most personal firewalls and HIPS systems intercept IRP_MJ_CREATE_FILE requests sent to the transport driver on opening these devices. This allows the security software to learn who (which process) is going to communicate over the network. Generally speaking, there are two ways of achieving this:

- Hooking the ZwCreateFile system service handler to intercept all attempts to open the devices;
- Attaching to *\Device\Tcp* or *\Device\Udp* in order to intercept all the IRP requests sent.

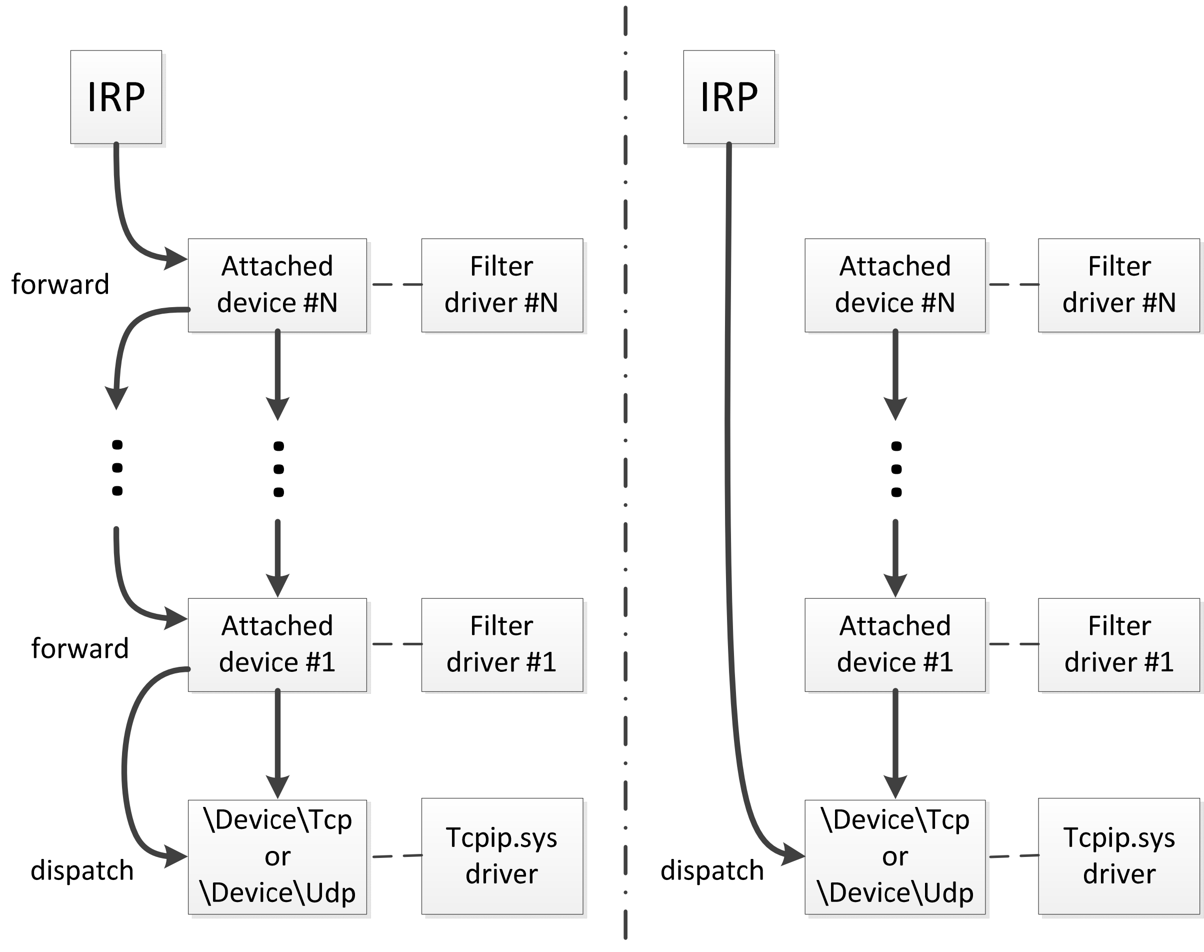
Let's see how Win32/Festi bypasses both techniques to establish connection with a remote host over the network. Instead of using a system implementation of the *ZwCreateFile* system service, Win32/Festi implements its own, with almost the same functionality as the original. The figure on the next page describes the custom implementation of the *ZwCreateFile* routine.

Figure 3: The custom implementation of the ZwCreateFile routine.



From the figure we can see that Win32/Festi manually creates a file object to communicate with the device being opened and sends an IRP_MJ_CREATE request directly to the transport driver. Thus, all the devices attached to *\Device\Tcp* or *\Device\Udp* will miss the request and as a result this operation is unnoticed by security software. This is clarified in the next figure.

Figure 4: IRP request sent directly to the transport driver.



In order to be able to send the request directly to *\Device\Tcp* or *\Device\Udp* the malware requires pointers to corresponding device objects. It obtains a pointer to the *Tcpip.sys* driver object by executing

NTSTATUS
ObReferenceObjectByName (
IN PUNICODE_STRING ObjectName,
IN ULONG Attributes,
IN PACCESS_STATE AccessState OPTIONAL,
IN ACCESS_MASK DesiredAccess OPTIONAL,
IN POBJECT_TYPE ObjectType,
IN KPROCESSOR_MODE AccessMode,
IN OUT PVOID ParseContext OPTIONAL,
*OUT PVOID *Object*
);

This is an undocumented system routine passing it as parameter pointer to a Unicode string with target driver name. Then the malware iterates through the list of device objects corresponding to the driver object and compares its names with *"\Device\Tcp"* or *"\Device\Udp"*. The fragment of code responsible for this maneuver looks like this:

```
_this = this;
if ( !this->DeviceTcp )
{
  if ( this->tcp_drv_ver < 6u )
    RtlInitUnicodeString(&DriverName, L"\\Driver\\Tcpip");
  else
    RtlInitUnicodeString(&DriverName, L"\\Driver\\tdx");
  RtlInitUnicodeString(&tcp_name, L"\\Device\\Tcp");
  RtlInitUnicodeString(&udp_name, L"\\Device\\Udp");
  if ( !ObReferenceObjectByName(
          &DriverName,
          64,
          0,
          0x1F01FFu,
          IoDriverObjectType,
          0,
          0,
          &TcpipDriver) )                     // get pointer to DRIVER_OBJECT by its name
  {
    ms_exc.disabled = 0;
    DevObj = TcpipDriver->DeviceObject;
    _DevObj = TcpipDriver->DeviceObject;
    while ( DevObj )                          // iterate through DEVICE_OBJECT linked list
    {
      if ( !ObQueryNameString(DevObj, &Objname, 256, &v8) )// Get DEVICE_OBJECT name
      {
        if ( RtlCompareUnicodeString(&tcp_name, &Objname, 1u) )// Check names
        {
          if ( !RtlCompareUnicodeString(&udp_name, &Objname, 1u) )
          {
            ObfReferenceObject(DevObj);
            _this->DeviceUdp = DevObj;        // Save pointer to \Device\Udp
            DevObj = _DevObj;
          }
        }
        else
        {
          ObfReferenceObject(DevObj);
          _this->DeviceTcp = DevObj;          // Save pointer to \Device\Tcp
        }
      }
      _DevObj = DevObj->NextDevice;           // get pointer to next DEVICE_OBJECT in the list
      DevObj = DevObj->NextDevice;
    }
    ms_exc.disabled = -2;
    ObfDereferenceObject(TcpipDriver);
  }
  this = _this;
}
return this->DeviceTcp != 0;
```

When the malware obtains a handle for the opened device in this manner it uses the handle to send/receive data over the network. Although the malware can avoid security software, we can see packets sent by the malware with network traffic filters operating at a lower level (NDIS level) than Win32/Festi.

Detecting Virtual Machines

Win32/Festi detects whether it is running inside a VMware virtual machine. It employs a rather well-documented technique: it executes the following instructions:

```
mov eax, 'VMXh'
mov ebx, 0
mov ecx, 0Ah
mov edx, 'VX'
in  eax, dx
```

If the code is executed inside a VMware virtual environment the *ebx* register will contain the 'VMX' dword.

Anti-debugging

Win32/Festi also checks for the presence of a kernel debugger in the system by examining the *KdDebuggerEnabled* symbol. It also periodically zeroes debugging registers so as to remove the hardware breakpoint, if any:

```
char __thiscall ProtoHandler_1(STRUCT_4_4 *this, PKEVENT a1)
{
  _EDX = 0;
  __asm
  {
    mov     dr0, edx
    mov     dr1, edx
    mov     dr2, edx
    mov     dr3, edx
  }
  return _ProtoHandler(&this->struct43, a1);
}
```

OOP Reversing problem

In this section we want highlight some of the problems that researchers face while reversing object-oriented code. Currently, there is a lot of malware written in C++ and other high-level languages which employ an object-oriented approach to implementing complex logic.

Take, for instance, such threats as Duqu or Stuxnet that make extensive use of object-oriented structures. As the size and number of implemented objects in the code grows, then the task of reversing is more and more challenging.

In programs written in procedural languages like C it is usually straightforward to build a control flow graph until special measures are taken to obfuscate it. Although C supports dynamic pointers to routines, which tend to complicate things, it’s normally quite easy to ascertain control transfer direction.

In the case of code written in object-oriented languages, the task of building a control flow graph is not so easy. For instance, virtual functions, which implement polymorphism in the C++ language, are called by pointers. This is depicted here:

```
pagecode:0001BC80 68 09 46 00 00      push    4609h
pagecode:0001BC85 58                  pop     eax
pagecode:0001BC86 8B D0               mov     edx, eax
pagecode:0001BC88 8B 4F 0C            mov     ecx, [edi+0Ch]
pagecode:0001BC8B 8B 01               mov     eax, [ecx]            Load pointer to table
                                                                    of virtual methods
pagecode:0001BC8D 8D 55 F8            lea     edx, [ebp-8]
pagecode:0001BC90 52                  push    edx
pagecode:0001BC91 FF 77 10            push    dword ptr [edi+10h]
pagecode:0001BC94 FF 50 0C            call    dword ptr [eax+0Ch]   Call virtual method
pagecode:0001BC97 84 C0               test    al, al
pagecode:0001BC99 75 0E               jnz     short loc_1BCA9
pagecode:0001BC9B 8B CF               mov     ecx, edi        ; this
pagecode:0001BC9D E8 5E FF FF FF      call    CloseTcpSocket
pagecode:0001BCA2 B8 CF 44 00 00      mov     eax, 44CFh
pagecode:0001BCA7 50                  push    eax
pagecode:0001BCA8 5B                  pop     ebx
```

Given this information it is difficult to get the exact address of the routine being called. Static analysis doesn’t provide a researcher with information as to the location register that *eax* points to. To be able to get the address, one needs to figure out where the object of specified type is created. At the time of its creation an object is initialized with a pointer to table of virtual methods like this:

```
STRUCT_4_3 *__thiscall CSocket_Ctor(STRUCT_4_3 *this)
{
  STRUCT_4_3 *v1; // esi@1

  v1 = this;
  this->vTable = &csocket_v_table;
  this->struct44 = 0;
  this->DeviceTcp = 0;
  this->DeviceUdp = 0;
  sub_19664(&this->struct41);
  sub_13E22(&v1->struct2);
  v1->SocketNumber = 0;
  v1->RefNo = 0;
  return v1;
}
```

In the figure above you can see a constructor of *CSocket* class implemented in the malware. We can see that its opaque CSocket::*vTable* field is initialized with a pointer to a table of virtual methods which has the following layout:

```
.rdata:000156E4 C4 E3 01 00  csocket_v_table dd offset InitializeTransport
.rdata:000156E8 48 0B 02 00                  dd offset OpenTransport
.rdata:000156EC C1 0C 02 00                  dd offset CloseTransport
.rdata:000156F0 BD F3 01 00                  dd offset TcpConnect
.rdata:000156F4 FC F5 01 00                  dd offset TcpDisconnect
.rdata:000156F8 EF E4 01 00                  dd offset sub_1E4EF
.rdata:000156FC 10 E5 01 00                  dd offset sub_1E510
.rdata:00015700 0A F8 01 00                  dd offset ReleaseNodeFromList
.rdata:00015704 86 F8 01 00                  dd offset TcpListen
.rdata:00015708 B8 0D 02 00                  dd offset TcpAccept
.rdata:0001570C 28 FA 01 00                  dd offset TcpSend
.rdata:00015710 DF FC 01 00                  dd offset TcpReceive
.rdata:00015714 BD FF 01 00                  dd offset UdpSend
.rdata:00015718 B3 02 02 00                  dd offset ReceiveDataFromUdp
.rdata:0001571C 7B 05 02 00                  dd offset GetTcpAddressInfo
.rdata:00015720 A8 E5 01 00                  dd offset sub_1E5A8
.rdata:00015724 2E E5 01 00                  dd offset SetTimeout
.rdata:00015728 4F E5 01 00                  dd offset SendOverUdp
.rdata:0001572C 7F E5 01 00                  dd offset ret_0
.rdata:00015730 84 E5 01 00                  dd offset GetErrorCode
.rdata:00015734 89 E5 01 00                  dd offset GetIrpStatus
```

Thus, when we encounter a virtual function call in static analysis, we are unable to get address of the called routine unless we already have type information for the object. To get this info we need to find where the object is created, and this task is quite challenging. As a result, reversing object oriented code is usually a difficult and time consuming task.

Plugins

During our investigation of Win32/Festi we noticed that different bots download different set of plugins. We managed to identify two sets of bots:

- spammers – those that send spam;
- DDoS – the bots designated to perform DDoS attacks.

Spam module (BotSpam.dll)

This plugin is responsible for sending junk emails. The plugin receives a list of email addresses that it should send spam letters to, and the actual text for sending. There is nothing unusual about the algorithm for sending spam. The plugin merely runs through the list of recipients and sends mails to corresponding email addresses.

The interesting thing about the spam plugin is the way it checks the status of sent email. The plugin scans responses from the SMTP servers for specific string constants signifying that there are problems with sending email (the mail wasn't received or was classified as junk). In the course of the research we obtained two almost identical versions of the plugins where the plugin looks for different strings in the server response. Both sets of the strings are used for server reply verification and presented in appendixes B and C, correspondingly. In the event that the plugin finds any of the strings in the server's reply it stops sending messages to that address and fetches the next address in its list.

DDos module (BotDos.dll)

The DDoS plugin allows the bot to perform DDoS attacks against specified hosts. The plugin supports several types of DDoS attacks, depending on configuration data received from C&C. It is highly configurable and, as a result, may be used to mount attacks on remote hosts with different kinds of software installed, and of different architecture. Here are types of attack implemented by the plugin:

- Tcp flood;
- Udp flood;
- DNS flood;
- HTTP flood.

Tcp flood

In the case of TCP flooding the bot initiates by default a large number of connections to port 80 (HTTP port) on the target machine. The port to connect to might be changed by corresponding configuration information from the C&C server.

Udp flood

For UDP flooding the bot sends UDP packets of randomly generated length and filled with random data. The length of the packet lies in the range from 256 up to 1024 bytes. The target port is also generated at random and therefore is unlikely to be open. As a result the attack causes the target host to generate enormous amount of ICMP Destination Unreachable packets in reply to UDP requests. Thus, the target machine becomes unavailable.

DNS flood

The bot is also able to perform a DNS flood attack. In such a case it sends high volumes of UDP packets to port 53 (DNS service) on the target host. The packets contain requests to resolve a randomly generated domain name in the ".com" domain zone.

HTTP flood

Another feature implemented in the bot is an HTTP flood attack against web servers. The bot contains a many different user-agent strings in the binary. You can find all the user-agent strings that the bot uses to attack web servers in appendix A. These strings are used to create a large number of HTTP sessions with the Web server, thus overloading the remote host. On the following page you can see the code for assembling the HTTP request to be sent.

Figure 5: Assembling the HTTP request to be sent.

```
user_agent_index = gen_rnd() % 0x64;
str_cpy(http_request, "GET ");
str_cat(http_request, (char *)v4 + 204 * *(_DWORD *)(v2 + 4) + 2796);
str_cat(http_request, " HTTP/1.0\r\n");
if ( *((_BYTE *)v4 + 2724) & 2 )
{
  str_cat(http_request, "Accept: */*\r\n");
  str_cat(http_request, "Accept-Language: en-US\r\n");
  str_cat(http_request, "User-Agent: ");
  str_cat(http_request, user_agent_str[user_agent_index]);
  str_cat(http_request, "\r\n");
}
str_cat(http_request, "Host: ");
str_cat(http_request, (char *)v4 + 204 * *(_DWORD *)(v2 + 4) + 2732);
str_cat(http_request, "\r\n");
if ( *((_BYTE *)v4 + 2724) & 2 )
  str_cat(http_request, "Connection: Keep-Alive\r\n");
str_cat(http_request, "\r\n");
result = str_len(http_request);
*(_DWORD *)(v2 + 16) = result;
return result;
```

To send the packets the plugin employs network sockets implemented by the main module of the bot. As a result the attack is performed in kernel-mode, which makes it quite stealthy.

Win32/Festi is also to send IP packets with the protocol field set to a random value.

This concludes our analysis of Win32/Festi. The following three appendices provide the HTTP DoS user-agent strings and the two different sets of strings we found being used for checking SMTP server replies when sending spam.

The Authors

Eugene Rodionov, Malware Researcher

Aleksandr Matrosov, Security Intelligence Team Lead

Appendix A: HTTP DoS user-agent strings

Mozilla/5.0 (Linux; U; Android 2.3.3; ko-kr; LG-LU3000 Build/GRI40) AppleWebKit/533.1 (KHTML, like Gecko) Version/4.0 Mobile Safari/533.1
Mozilla/5.0 (Linux; U; Android 2.3.3; zh-tw; HTC Pyramid Build/GRI40) AppleWebKit/533.1 (KHTML, like Gecko) Version/4.0 Mobile Safari/533.1
Mozilla/5.0 (Linux; U; Android 2.3.3; zh-tw; HTC_Pyramid Build/GRI40) AppleWebKit/533.1 (KHTML, like Gecko) Version/4.0 Mobile Safari
Mozilla/5.0 (Linux; U; Android 2.3.3; zh-tw; HTC_Pyramid Build/GRI40) AppleWebKit/533.1 (KHTML, like Gecko) Version/4.0 Mobile Safari/533.1
Mozilla/5.0 (Linux; U; Android 2.3.4; en-us; T-Mobile myTouch 3G Slide Build/GRI40) AppleWebKit/533.1 (KHTML, like Gecko) Version/4.0 Mobile Safari/533.1
Mozilla/5.0 (Linux; U; Android 2.3.4; fr-fr; HTC Desire Build/GRJ22) AppleWebKit/533.1 (KHTML, like Gecko) Version/4.0 Mobile Safari/533.1
Mozilla/5.0 (Linux; U; Android 2.3.5; en-us; HTC Vision Build/GRI40) AppleWebKit/533.1 (KHTML, like Gecko) Version/4.0 Mobile Safari/533.1
Mozilla/5.0 (Linux; U; Android 2.3; en-us) AppleWebKit/999+ (KHTML, like Gecko) Safari/999.9
Mozilla/5.0 (Windows; U; Windows NT 5.1; de; rv:1.8.0.10) Gecko/20070221 Thunderbird/1.5.0.10
Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.8.0.10) Gecko/20070306 Thunderbird/1.5.0.10
Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.8.0.10) Gecko/20070403 Thunderbird/1.5.0.10
Mozilla/5.0 (Windows; U; Windows NT 5.1; en-GB; rv:1.8.0.14) Gecko/20071210 Thunderbird/1.5.0.14
Mozilla/5.0 (Windows; U; Windows NT 5.1; fr; rv:1.8.0.14) Gecko/20071210 Thunderbird/1.5.0.14
Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.1.3) Gecko/20070326 Thunderbird/2.0.0.0
Mozilla/5.0 (Windows; U; Windows NT 5.1; de; rv:1.8.1.12) Gecko/20080213 Thunderbird/2.0.0.12
Mozilla/5.0 (Windows; U; Windows NT 5.1; en-GB; rv:1.8.1.12) Gecko/20080213 Thunderbird/2.0.0.12
Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.1.12) Gecko/20080213 Thunderbird/2.0.0.12
Mozilla/5.0 (Windows; U; Windows NT 6.0; en-GB; rv:1.8.1.12) Gecko/20080213 Thunderbird/2.0.0.12
Mozilla/5.0 (Windows; U; Windows NT 6.0; en-US; rv:1.8.1.12) Gecko/20080213 Thunderbird/2.0.0.12
Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.8.1.12) Gecko/20080227 Thunderbird/2.0.0.12
Mozilla/5.0 (X11; U; Linux i686; nl; rv:1.8.1.12) Gecko/20080213 Thunderbird/2.0.0.12
Mozilla/5.0 (Macintosh; U; Intel Mac OS X; en-GB; rv:1.8.1.14) Gecko/20080421 Thunderbird/2.0.0.14
Mozilla/5.0 (Macintosh; U; Intel Mac OS X; en-US; rv:1.8.1.14) Gecko/20080421 Thunderbird/2.0.0.14
Mozilla/5.0 (Windows; U; Windows NT 5.1; en-GB; rv:1.8.1.14) Gecko/20080421 Thunderbird/2.0.0.14
Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.1.14) Gecko/20080421 Thunderbird/2.0.0.14
Mozilla/5.0 (Windows; U; Windows NT 6.0; en-GB; rv:1.8.1.14) Gecko/20080421 Thunderbird/2.0.0.14
Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.8.1.14) Gecko/20080502 Thunderbird/2.0.0.14
Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.8.1.14) Gecko/20080505 Thunderbird/2.0.0.14
Mozilla/5.0 (Macintosh; U; Intel Mac OS X; fr; rv:1.8.1.16) Gecko/20080707 Thunderbird/2.0.0.16
Mozilla/5.0 (Windows; U; Windows NT 5.1; de; rv:1.8.1.16) Gecko/20080708 Thunderbird/2.0.0.16
Mozilla/5.0 (Windows; U; Windows NT 5.1; en-GB; rv:1.8.1.16) Gecko/20080708 Thunderbird/2.0.0.16
Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.1.16) Gecko/20080708 Thunderbird/2.0.0.16
Mozilla/5.0 (Windows; U; Windows NT 5.1; fr; rv:1.8.1.16) Gecko/20080708 Thunderbird/2.0.0.16
Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.8.1.16) Gecko/20080723 Fedora/2.0.0.16-1.fc8 Thunderbird/2.0.0.16
Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.8.1.16) Gecko/20080724 Thunderbird/2.0.0.16
Mozilla/5.0 (Macintosh; U; Intel Mac OS X; en-GB; rv:1.8.1.17) Gecko/20080914 Thunderbird/2.0.0.17
Mozilla/5.0 (Macintosh; U; Intel Mac OS X; en-US; rv:1.8.1.17) Gecko/20080914 Thunderbird/2.0.0.17
Mozilla/5.0 (Windows; U; Windows NT 5.1; en-GB; rv:1.8.1.17) Gecko/20080914 Thunderbird/2.0.0.17

Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.1.17) Gecko/20080914 Thunderbird/2.0.0.17
Mozilla/5.0 (Windows; U; Windows NT 5.1; fr; rv:1.8.1.17) Gecko/20080914 Thunderbird/2.0.0.17
Mozilla/5.0 (Windows; U; Windows NT 6.0; de; rv:1.8.1.17) Gecko/20080914 Thunderbird/2.0.0.17
Mozilla/5.0 (Windows; U; Windows NT 6.0; en-GB; rv:1.8.1.17) Gecko/20080914 Thunderbird/2.0.0.17
Mozilla/5.0 (Windows; U; Windows NT 6.0; en-US; rv:1.8.1.17) Gecko/20080914 Thunderbird/2.0.0.17
Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.8.1.17) Gecko/20080925 Thunderbird/2.0.0.17
Mozilla/5.0 (X11; U; Linux x86_64; en-US; rv:1.8.1.17) Gecko/20081018 Thunderbird/2.0.0.17
Mozilla/5.0 (Macintosh; U; Intel Mac OS X; en-GB; rv:1.8.1.18) Gecko/20081105 Thunderbird/2.0.0.18
Mozilla/5.0 (Macintosh; U; Intel Mac OS X; en-US; rv:1.8.1.18) Gecko/20081105 Thunderbird/2.0.0.18
Mozilla/5.0 (Windows; U; Windows NT 5.1; en-GB; rv:1.8.1.18) Gecko/20081105 Thunderbird/2.0.0.18
Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.1.18) Gecko/20081105 Thunderbird/2.0.0.18
Mozilla/5.0 (Windows; U; Windows NT 6.0; de; rv:1.8.1.18) Gecko/20081105 Thunderbird/2.0.0.18
Mozilla/5.0 (Windows; U; Windows NT 6.0; en-GB; rv:1.8.1.18) Gecko/20081105 Thunderbird/2.0.0.18
Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.8.1.18) Gecko/20081125 Thunderbird/2.0.0.18
Mozilla/5.0 (Macintosh; U; Intel Mac OS X; en-US; rv:1.8.1.19) Gecko/20081209 Thunderbird/2.0.0.19
Mozilla/5.0 (Windows; U; Windows NT 5.1; de; rv:1.8.1.19) Gecko/20081209 Thunderbird/2.0.0.19
Mozilla/5.0 (Windows; U; Windows NT 5.1; en-GB; rv:1.8.1.19) Gecko/20081209 Thunderbird/2.0.0.19
Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.1.19) Gecko/20081209 Thunderbird/2.0.0.19
Mozilla/5.0 (Windows; U; Windows NT 6.0; de; rv:1.8.1.19) Gecko/20081209 Thunderbird/2.0.0.19
Mozilla/5.0 (Windows; U; Windows NT 6.0; en-GB; rv:1.8.1.19) Gecko/20081209 Thunderbird/2.0.0.19
Mozilla/5.0 (Windows; U; Windows NT 6.0; en-US; rv:1.8.1.19) Gecko/20081209 Thunderbird/2.0.0.19
Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.8.1.19) Gecko/2008120920 Thunderbird/2.0.0.19
Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.8.1.19) Gecko/20090105 Thunderbird/2.0.0.19
Mozilla/5.0 (Windows; U; Windows NT 5.1; de; rv:1.8.1.21) Gecko/20090302 Thunderbird/2.0.0.21
Mozilla/5.0 (Windows; U; Windows NT 5.1; en-GB; rv:1.8.1.21) Gecko/20090302 Thunderbird/2.0.0.21
Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.1.21) Gecko/20090302 Thunderbird/2.0.0.21
Mozilla/5.0 (Windows; U; Windows NT 5.1; fr; rv:1.8.1.21) Gecko/20090302 Thunderbird/2.0.0.21
Mozilla/5.0 (Windows; U; Windows NT 6.0; de; rv:1.8.1.21) Gecko/20090302 Thunderbird/2.0.0.21
Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.8.1.21) Gecko/20090318 Thunderbird/2.0.0.21
Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.8.1.21) Gecko/20090409 Thunderbird/2.0.0.21
Mozilla/5.0 (Windows; U; Windows NT 5.1; en-GB; rv:1.8.1.24) Gecko/20100228 Thunderbird/2.0.0.24
Mozilla/5.0 (Windows; U; Windows NT 6.0; de; rv:1.8.1.24) Gecko/20100228 Thunderbird/2.0.0.24
Mozilla/5.0 (Windows; U; Windows NT 6.0; en-GB; rv:1.8.1.24) Gecko/20100228 Thunderbird/2.0.0.24
Mozilla/5.0 (Windows; U; Windows NT 6.1; en-GB; rv:1.8.1.24) Gecko/20100228 Thunderbird/2.0.0.24
Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.8.1.24) Gecko/20100317 Thunderbird/2.0.0.24
Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.8.1.24) Gecko/20100411 Thunderbird/2.0.0.24
Mozilla/5.0 (Windows; U; Windows NT 5.0; es-ES; rv:1.9.1.7) Gecko/20100111 Thunderbird/3.0.1 ThunderBrowse/3.2.8.1
Mozilla/5.0 (Windows; U; Windows NT 5.0; fr-CA; rv:1.9.1.7) Gecko/20100111 Thunderbird/3.0.1 ThunderBrowse/3.2.8.1
Mozilla/5.0 (Windows; U; Windows NT 5.1; es-ES; rv:1.9.1.7) Gecko/20100111 Thunderbird/3.0.1 ThunderBrowse/3.2.8.1
Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.1.15) Gecko/20101027 Lightning/1.0b1 Thunderbird/3.0.10
Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.1.15) Gecko/20101027 Lightning/1.0b1 Thunderbird/3.0.10 ThunderBrowse/3.3.2
Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10.5; en-US; rv:1.9.1.9) Gecko/20100317 Thunderbird/3.0.4
Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10.5; pl; rv:1.9.1.9) Gecko/20100317 Thunderbird/3.0.4
Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10.6; en-US; rv:1.9.1.9) Gecko/20100317 Thunderbird/3.0.4

Mozilla/5.0 (Windows; U; Windows NT 5.1; de; rv:1.9.1.9) Gecko/20100317 Thunderbird/3.0.4
Mozilla/5.0 (Windows; U; Windows NT 5.1; en-GB; rv:1.9.1.9) Gecko/20100317 Lightning/1.0b1 Thunderbird/3.0.4
Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.1.9) Gecko/20100423 Thunderbird/3.0.4
Mozilla/5.0 (Windows; U; Windows NT 5.0; de; rv:1.9.2.17) Gecko/20110414 Thunderbird/3.1.10
Mozilla/5.0 (Windows; U; Windows NT 5.0; de; rv:1.9.2.17) Gecko/20110414 Thunderbird/3.1.10 ThunderBrowse/3.3.5
Mozilla/5.0 (Windows; U; Windows NT 5.0; en; rv:1.9.2.17) Gecko/20110414 Thunderbird/3.1.10
Mozilla/5.0 (Windows; U; Windows NT 5.1; en-GB; rv:1.9.2.17) Gecko/20110414 Thunderbird/3.1.10
Mozilla/5.0 (Windows; U; Windows NT 6.0; de; rv:1.9.2.17) Gecko/20110414 Thunderbird/3.1.10 ThunderBrowse/3.3.5
Mozilla/5.0 (Windows; U; Windows NT 6.1; de; rv:1.9.2.17) Gecko/20110414 Lightning/1.0b2 Thunderbird/3.1.10
Mozilla/5.0 (Windows; U; Windows NT 6.1; de; rv:1.9.2.17) Gecko/20110414 Thunderbird/3.1.10
Mozilla/5.0 (Windows; U; Windows NT 6.1; en-GB; rv:1.9.2.17) Gecko/20110414 Lightning/1.0b2 Thunderbird/3.1.10
Mozilla/5.0 (Windows; U; Windows NT 6.1; en-GB; rv:1.9.2.17) Gecko/20110414 Thunderbird/3.1.10
Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US; rv:1.9.2.17) Gecko/20110414 Lightning/1.0b2 Thunderbird/3.1.10
Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US; rv:1.9.2.17) Gecko/20110414 Thunderbird/3.1.10
Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.2.17) Gecko/20110424 Lightning/1.0b2 Thunderbird/3.1.10
Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.2.17) Gecko/20110424 Thunderbird/3.1.10
Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.2.17) Gecko/20110516 Lightning/1.0b2 Thunderbird/3.1.10
Mozilla/5.0 (X11; U; Linux x86_64; de-DE; rv:1.9.2.17) Gecko/20110424 Lightning/1.0b2 Thunderbird/3.1.10

Appendix B: Strings for checking SMTP server reply (1)

RP-001
RP-002
RP-003
SC-001
SC-002
SC-003
SC-004
DY-001
DY-002
OU-001
OU-002
temporarily deferred
not allowed
Spamhaus PBL
Spamhaus SBL
Spamhaus XBL
permanently deferred
(RTR:BB)
(RTR:CH)
(RTR:BG)
(RTR:RD)
(RTR:SC)
(RTR:DU)
(RTR:GE)
(RTR:BL)
(DNS:B1)
(DNS:B2)
(DNS:NR)
(RLY:B1)
(RLY:B2)
(RLY:B3)
(RLY:BD)
(RLY:CH)
(RLY:CH2)
(RLY:CS4)
(RLY:IR)
(RLY:NW)
(RLY:SN)
(DYN:T1)
(CON:B1)

Appendix C: Strings for checking SMTP server reply (2)

concurrent connections limit
dsbl.org
blocked
reject mail
administrative prohibition
too many invalid recipients
not allowed
reject
deny
rbl
access denied
blackhole
blacklist
resolve
service not available
smtp service not available
denied
domain
dns
blocked by
connection from
valid host name
spam
invalid
4.1.8
4.4.4
4.7.7
5.4.4
mailfrom
invalid
could not resolve
unacceptable mail address
invalid
unable to find
domain
reaches maximum limit
mail failed
ip address denied
blocked by
spam
5.7.1
access denied
spam
unresolved
junk
blocked
rejected
blacklist
timeout
t connect to
resolve
unresolvable address
{mailfrom}
badmailfrom
received this hour
sender
user unknown
unknown user
no such user
bad address
0 address does not
sorry,
user not found
user does not
rejected by
rejected due to spam
5.1.0 address rejected
not accepted
spam source blocked
relaying denied
mailbox currently suspended
no mailbox here
spam
mailbox unavailable
not our customer
mailbox name not
unroutable address
invalid mailbox
quota excedida
mailbox not available
no such mailbox
quota exceeded
mail for that address
cannot send mail to this address
relay denied
no such recipient
no such address
blacklist

5.1.1 unknown or illegal alias
not local
does not exist at this domain
blocked
unknown local
banned
sorry !!:
is not an active address
unknown or illegal alias
5.1.2 unknown host or domain:
addressee unknown
relaying not allowed
invalid address
listed at
access denied
unable to deliver to
cannot deliver
address changed
address invalid
not known here
does not have an
you are not allowed to send mail
inactive mbox
rbl
refused by
ptr
see: http://
error, el dominio no esta en mi
account
rcpt to:
is restricted
blacklist
>... >
email address disabled
mail from
: host
access not allowed
destinataire inconnu
bogus
connection refused
address unknown
destinatario desconocido
helo/ehlo
badhelo
refused
rejected
restricted
ip address
user unkown
usuario
default blocking message
quota
bad rcpts
invalid user
listed
blocking message
proxy
thank you for your kind message
denied by access list complain
relay
ehlo/he
address not valid
black list
target address
local policy
black
header error
no such
local policy
no mailbox here
unknown mailbox
spam
user unknown
recipient
blocked
mail from
block
required stand
prohibition
sorry
unsolicited bulk
is not match with your ip
filter
bloc
spam
scored
message content
refused
uce
policy
junk
deliver
url listed
rejected
not accept
5.0.0

unsolicited
spam
denied
unknown recipient
content
user not found
invalid address
hostname
cannot find your hostname
bad from mx record
greylist
authentic
unresolvable host name
reverse
dynamic
name lookup failed
not resolve
client host
is listed
you rejected
listed in
host not found
is blocked
abuseat.org
spamhaus
relay access denied
abuse
is blacklisted
blackholes.us
see http:
spamcop.net
blocked http:
your ip
dnsbl.sorbs.net
smtp rejected from
blacklisted for
\"helo\"
open proxy
dsbl.org/listing
blacklisted ip
?ip=
dsbl.org
mail from block-listed site
ip address rejec
mail from pool
error, el dominio no esta en mi
no host name
ordb.org
host has incomplete dns
administrative prohibition
try
temp fail
recipient address
later
too many
dialup
address of sender
dynablock
bad helo
njabl.org
badhelo