

# **Pushdo / Cutwail**

A study of the Pushdo / Cutwail botnet

## **Pushdo / Cutwail Botnet**

A study of the Pushdo / Cutwail Botnet

by

<b>Alice Decker:</b>	Malware Testing
<b>David Sancho:</b>	Reverse Engineering
<b>Loucif Kharouni:</b>	Network Analysis
<b>Max Goncharov:</b>	Underground Research
<b>Robert McArdle:</b>	Project Coordinator

**Release Date:** 22 May 2009

**Classification:** External Use

# Pushdo / Cutwail

A study of the Pushdo / Cutwail botnet

## Table of Contents

<b>Introduction.....</b>	<b>3</b>
<b>Pushdo / Cutwail Analysis.....</b>	<b>4</b>
Stage 1: Initial Installer.....	4
Stage 2: Downloader Module.....	7
Stage 3: Downloaded Modules - TempFile.....	11
Stage 4: Downloaded Modules – Cutwail Spam Engine.....	13
Cutwail Initialization: Stage 1 – Configuration Settings.....	15
Cutwail Initialization: Stage 2 – Handshake.....	16
Cutwail Initialization: Stage 3 – Retrieve spam content.....	14
Stage 5: Downloaded Modules – Pushdo Kernel Module.....	20
Stage 6: Downloaded Modules – Pushdo Sniffer Module.....	22
<b>Campaign Modules.....</b>	<b>23</b>
Campaign Modules: Example: Popup Ad.....	23
<b>Spam Analysis.....</b>	<b>26</b>
Spam Waves.....	26
Spam Wave 1: Self Promotion Spam.....	26
Spam Wave 2: Porn Sites.....	28
Spam Wave 3: Pharmacy Spam.....	29
Spam Wave 4: Prestige Replica Spam.....	30
Spam Wave 5: Local Advertising Spam.....	31
Spam Statistics.....	33
<b>Behind the Malware – Botnet Owners.....</b>	<b>34</b>
<b>Propagation of Pushdo.....</b>	<b>38</b>
<b>References.....</b>	<b>39</b>

## INTRODUCTION

The Pushdo botnet has been with us since January 2007<sup>1</sup>. The botnet is also known as Pandex or Cutwail. While it does not grab as many headlines as its attention-seeking peers such as Storm or Conficker, according to recent reports it is the 2<sup>nd</sup> largest SPAM botnet on the planet<sup>2</sup> – sending approximately 7.7 Billion emails per day, making it single-handedly responsible for about 1 out of every 25 emails sent<sup>3</sup>. This percentage is likely to be a lot higher in Russia, the target of the majority of Pushdo's spam.

There are several reasons for Pushdo's lack of notoriety – the authors have actively used several techniques to help keep its activity “under the radar”

- Not only is Pushdo responsible for a huge amount of spam activity, it also is one of the primary conduits for other criminal gangs to spread their malware creations. As a result many different detections exist for variants of this threat, the majority of which are so called “generic detections”. This confusion has actually helped the botnet keep a lower profile than its more famous competitors, and makes our role as researchers more difficult.
- Pushdo components are almost all memory resident, with very few being written to disk. This makes the job of security companies much more difficult when attempting to detect them.
- Pushdo does not contain any means of self replication. Unlike other well-known botnets such as Conficker and Storm, which spread via vulnerability exploitation and mass mailing, Pushdo appears to exhibit no Worm-like behaviour.
- Adding to this confusion is the tendency of the botnet owners to frequently change Pushdo's functionality and code. It is perhaps better to think of Pushdo as a “criminal operation”, rather than a single piece of malware.

For the purposes of clarification in this report we will refer to the various components of this threat as follows:

- **Pushdo:** This is the name given to the main malware binaries. Pushdo is essentially an advanced downloader which will first infect the system and then download the Cutwail spam module (also owned by the same criminal gang).

In addition to installing the Cutwail module it will also normally install 1 or more different “Campaign Modules” – third party malware from other malware groups. These modules account for the large number of observable differences between infections.

- **Cutwail:** The name Cutwail refers to the spamming module of the Pushdo botnet.

The modules which refer to the core of the Pushdo threat are in the “Pushdo / Cutwail Analysis” section of this report.

Some information on third party malware associated with Pushdo is included in the section entitled “Campaign Modules”

Note we do not use the name Pandex in this report.

<sup>1</sup> [http://threatinfo.trendmicro.com/vinfo/virusencyclo/default5.asp?VName=TROJ\\_PANDEX.A&Vsect=Sn](http://threatinfo.trendmicro.com/vinfo/virusencyclo/default5.asp?VName=TROJ_PANDEX.A&Vsect=Sn)

<sup>2</sup> [http://www.message-labs.com/mlireport/MLIRreport\\_2009.01\\_Jan\\_Final.pdf](http://www.message-labs.com/mlireport/MLIRreport_2009.01_Jan_Final.pdf)

<sup>3</sup> <http://royal.pingdom.com/2009/01/22/internet-2008-in-numbers/>

## PUSHDO / CUTWAIL ANALYSIS

Like most malware, Pushdo is distributed as a binary file. Our first step in the analysis of this malware is to use IDA Pro to disassemble the binary file, and then interpret the resulting assembler code. Additionally we execute the malware in a test environment and monitor all System and Network Activity.

**Note:** As part of our analysis we uncovered the malware authors original names for each module and the malware project associated with it. We have included this information at the start of each module for reference.

### STAGE 1: INITIAL INSTALLER

**Debug Name:** UMLoader  
**Project Name:** Siberia2

Unusually for malware the initial installer binary is not actually packed. Once executed the malware first tests to see if it's currently running as the hardcoded value "rs32net.exe" in the system folder (C:\Windows\System32 by default). If not, the malware creates a copy of itself as rs32net.exe in the system folder, and executes it. In some other variants we have analyzed the name is not hard coded – rather it uses the name of the currently logged in user as the executable name.

It then removes the original installer using the following call to the command line, and then exits:

```
cmd /c del [ORIGINAL_INSTALLER_LOCATION] >> NUL
```

Once it is clear that the malware is running with the correct name, and from the correct folder, the installer's main routine starts. The process first creates a new thread. This thread is responsible for creating, and recreating the following two system load points every 10 seconds:

```
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\Rs32net  
HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\Rs32net
```

Both of these keys point to the copy of the threat in the windows system folder, and ensure that the threat will execute automatically when the machine reboots.

The threat next executes a copy of svchost.exe (hereafter referred to as *svchost(1)*), starting the process in suspended mode. It then injects executable code into svchost by repeatedly calling the **WriteProcessMemory** windows API, writing each section of the injected PE executable in turn into the memory of *svchost(1)* starting at memory offset 0x90000000, before patching the entry point of *svchost(1)* to instead point to the entry point of the injected executable. Next the main thread of *svchost(1)* is resumed, causing the injected code to execute. In the past the malware authors have been seen to use Internet Explorer instead of svchost for these purposes<sup>4</sup>.

This injected executable code is located, encrypted, between Offsets 0x3000 and 0x5C00 in the data section of *Rs32net.exe*. Before injection this code is first decrypted in 4 byte (1 dword) chunks using the following algorithm:

```
Decrypted_Dword = Encrypted_Dword XOR (0x87654321 XOR Dword_Offset)
```

<sup>4</sup> <http://www.virusbtn.com/virusbulletin/archive/2008/03/vb200803-pandex>

# Pushdo / Cutwall

A study of the Pushdo / Cutwall botnet

```
C:\ Select View: 1.exe
1.exe                                ↓FRO    PE.08003000                                18944|H
000025F0: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00
00003000: 6C 19 F5 87-26 43 65 87-2D 43 65 87-D2 BC 65 87  11Jc&Cec-CecπJec
00003010: 89 43 65 87-35 43 65 87-79 43 65 87-3D 43 65 87  2Cec5CecyCec=Cec
00003020: 01 43 65 87-05 43 65 87-09 43 65 87-0D 43 65 87  0Cec&Cec0CecJCec
```

Figure 1.1: First Encrypted DWORD [6C19F587]

```
C:\ View: 1.exe
1.exe                                ↓FWO    PE 00001A04    <Editor>                                18944|H
000019F0: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00
00001A00: 4D 5A 90 00-26 43 65 87-2D 43 65 87-D2 BC 65 87  MZ& &Cec-CecπJec
00001A10: 89 43 65 87-35 43 65 87-79 43 65 87-3D 43 65 87  2Cec5CecyCec=Cec
00001A20: 01 43 65 87-05 43 65 87-09 43 65 87-0D 43 65 87  0Cec&Cec0CecJCec
```

Figure 1.2: Decrypted DWORD - 6C19F587 XOR (87654321 XOR 0)

```
C:\ Select View: 1.exe
1.exe                                ↓FWO    PE 00001A04    <Editor>                                18944|H
000019F0: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00
00001A00: 4D 5A 90 00-26 43 65 87-2D 43 65 87-D2 BC 65 87  MZ& &Cec-CecπJec
00001A10: 89 43 65 87-35 43 65 87-79 43 65 87-3D 43 65 87  2Cec5CecyCec=Cec
00001A20: 01 43 65 87-05 43 65 87-09 43 65 87-0D 43 65 87  0Cec&Cec0CecJCec
```

Figure 1.3: 2nd Encrypted DWORD. Decrypted\_DWORD = 26436587 XOR (87654321 XOR 4)

Once fully decrypted a small section of the data is patched before injection. Depending on whether *Rs32net.exe* was executed with a */n* parameter, this patching will vary slightly. The patched data consists of parameters to be used in a HTTP GET request issued by the injected code, which is described in more detail in the next section.

Lastly, *Rs32net.exe* will create a second copy of itself in memory, this time running with the */n* parameter which results in the creation of another *svchost* process (hereafter *svchost(2)*) with slightly different behaviour (more information below). Upon completion of this routine, *Rs32net.exe* will remain running in memory, constantly recreating the registry keys as explained above.

A result of this technique is that the malware components are never written to disk making it much more difficult for anti-virus software to scan for and detect Pushdo. This technique is used by the majority of Pushdo components.

# Pushdo / Cutwail

A study of the Pushdo / Cutwail botnet

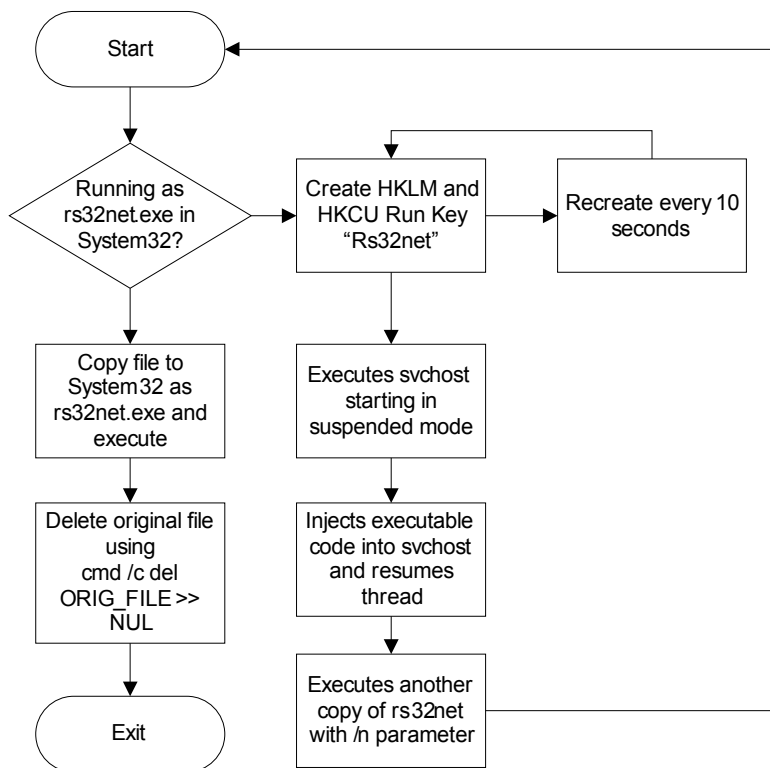


Figure 1.4: Stage 1 – Initial Installer

# Pushdo / Cutwail

## A study of the Pushdo / Cutwail botnet

### STAGE 2: DOWNLOADER MODULE

**Debug Name:** loader  
**Project Name:** Siberia2

After having been injected by the initial installer the code in *svchost(1).exe* and *svchost(2).exe* behave in almost identical ways. Both processes act as Downloaders – downloading additional Pushdo, and third party, components onto the machine.

They both start by first checking their own MZ header to ensure it is correctly formatted. Once this sanity check is out of the way, they proceed onto their main routines.

First they query the following registry values to retrieve the infected PC's system bios date, video bios date and processor type.

**HKLM\Hardware\Description\System\SystemBiosDate**  
**HKLM\Hardware\Description\System\VideoBiosDate**  
**HKLM\Hardware\Description\System\CentralProcessor\0**

Information about the hard disk is also obtained using the **GetVolumeInformation** API. This technique can be easily used to identify if Pushdo is running in a virtual environment, as they tend to report back predictable responses to this API call. Pushdo does not seem to be concerned about this however, running the same in virtual and normal environments.

Next they decrypt an area of their .data section which yields 3 parameter values, and a list of six IP addresses. This is done via a simple XOR decryption using the value 0x78d616b2. The IP addresses are the same for both *svchost(1)* and *svchost(2)*, however the parameters are slightly different based on whether the initial installer was run with the /n parameter. While the hardcoded IPs are generally the same between variants, there is anecdotal evidence that they are updated fairly regularly.

Drop-site IP Addresses	
70.38.68.137	69.64.67.194
91.211.64.117	74.54.77.82
94.247.3.46	74.54.135.202
192.8.75.216	216.55.176.45
216.195.63.22	72.167.49.117
66.45.246.146	92.62.101.118

Figure 2.1: C&C IP Addresses

The code then attempts to connect to each of these drop-site IP addresses in turn. In our testing we found that most of these drop-sites or C&C servers are located in North America. Also each IP address comes from a different provider and each is located on a different ASN<sup>5</sup>. This adds a lot of redundancy to the Pushdo botnet, as six separate network providers need to be contacted to shutdown any one variant of the malware.

Once it has successfully established a connection it issues a specially-crafted HTTP GET request. The GET request differs from *svchost(1)* to *svchost(2)*, but the formatting is the same.



<sup>5</sup> [http://en.wikipedia.org/w/index.php?title=Autonomous\\_system\\_\(Internet\)&oldid=281446510](http://en.wikipedia.org/w/index.php?title=Autonomous_system_(Internet)&oldid=281446510)

## Pushdo / Cutwail

### A study of the Pushdo / Cutwail botnet

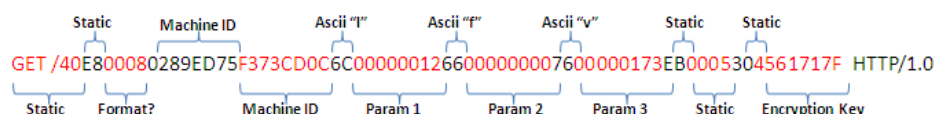


Figure 2.2: HTTP GET request

Some sections of the request are static, coming from hardcoded values in the executable. Others have different meanings:

- **GET /40E8:** This seems to order the server to return 1 or more executables to run on the victims machine
- **Format:** If this value is less than 0015, the downloaded executable will be encrypted using the last 8 Chars of the GET request as a key. If the value is 0015 or over, the executable is returned unencrypted, but needs to be patched before it can be run (each section offset is off by 1 byte)
- **Machine ID:** These 2 machine IDs are used as an identifier for the infected machine, and are calculated based on the system bios date, video bios date, processor type and the system volume information
- **Parameters:** Parameter 3 appears to specify which files to return. In *svchost(1)* parameter 3 is set to 00000173, whereas the value is 00000177 for *svchost(2)*.
- **Encryption Key:** This is the key used to encrypt and decrypt the downloaded executable.

The formatting of these GET requests is one of the many features of the Pushdo botnet that is constantly evolving, and there is evidence that they changed from an earlier format in order to avoid detection by Intrusion Detection Systems<sup>6</sup>.

The server responds by sending one or more executables. These executables can be either encrypted or unencrypted as described above.

If parameter 3 is 00000177 (*svchost(2)*), the server returns several. One of these is the main Cutwail spamming engine, another is Pushdo's kernel driver, and the rest are third party malware (described in the section "Campaign Modules"). During our testing we consistently received 4 to 5 executables every time. These executables are not encrypted, but they are attached one after the other in a single package with special formatting.

If parameter 3 is 00000173 (*svchost(1)*) the server returns a single executable. This may be another copy of one of the files returned from *svchost(2)*'s request, or alternatively an executable belonging to another malware family (in testing this was normally another copy of the kernel driver). This executable is normally encrypted.

The malware next checks the MZ header of each downloaded executable for the phrase "This Program cannot be run in DOS Mode". If the word "This" is present, the malware creates a temporary file in users "Local Settings\Temp" directory, copies the downloaded executable there and executes it. We hereafter call this executable *TempFile*.

<sup>6</sup> <http://www.virusbtn.com/virusbulletin/archive/2008/03/vb200803-pandex>



## Pushdo / Cutwail

A study of the Pushdo / Cutwail botnet

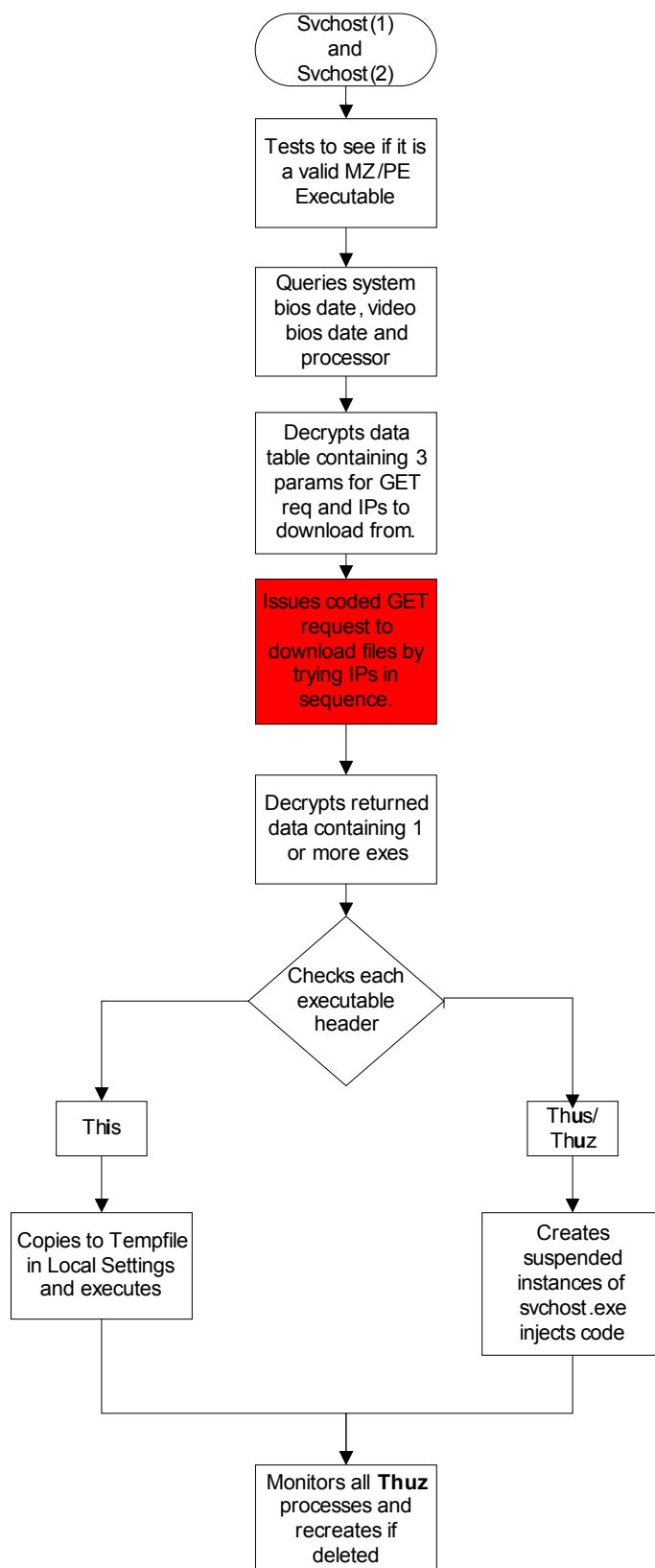


Figure 2.1: Stage 2 – Downloader Module. Red sections differs in svchost(1) and svchost(2)

## Pushdo / Cutwail

### A study of the Pushdo / Cutwail botnet

If the word “This” instead reads “Thus” (the third character in the sentence is the value actually tested by the malware) a new instance of svchost is created in, and the executable code is injected in the same manner as before, using **WriteProcessMemory** and patching the entry point. We will describe each of the downloaded executables in the following sections of this report.

The downloader module keeps information related to each of these processes in a structure in memory, containing the PID associated with each. If a certain character of the executable header is actually “z” (e.g. Thuz) the structure marks that particular injected module as critical. The downloader module constantly checks the list of running processes and compares the PIDs to this memory structure, to see if a “critical” module has been deleted. When this occurs the downloader module relaunches it by once more creating a new svchost.exe process and injecting the module code.

In our testing the only module marked as “critical” in this manner is the Cutwail spam engine. As such, the downloader adds a layer of protection to this module by constantly recreating it in memory.

## Pushdo / Cutwall

A study of the Pushdo / Cutwall botnet

### STAGE 3: DOWNLOADED MODULES - TEMPFILE

Debug Name: Install  
Project Name: Siberia2

This module is a simple dropper responsible for installing Pushdo's kernel module on the system.

The *TempFile* is dropped by the downloader module (*svchost(1)* or *svchost(2)*) in the current user's Local Settings\Temp folder. The name will be different each time, and is based on the **GetTempFileName** API Function, but will have the .tmp extension.

The *TempFile* first creates a name for the kernel file using the following observed formats:

**[SYSFILENAME] = ati + [3 Random characters] + xx.sys**

e.g ati0bsxx.sys.

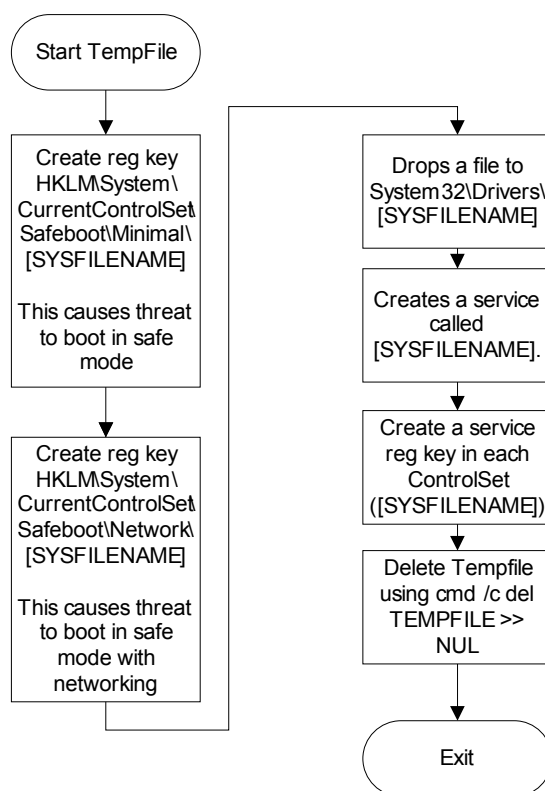


Figure 3.1: Stage 3: Downloaded Module - TempFile

Next it sets up the kernel driver so that it will be loaded into memory even if the computer starts in Safe Mode. It does this by creating the following 2 registry keys

**HKLM\SYSTEM\CurrentControlSet\Safeboot\Minimal\[SYSFILENAME]**  
**HKLM\SYSTEM\CurrentControlSet\Safeboot\Network\[SYSFILENAME]**

Once these registry keys are created, the kernel executable is copied to the Windows system folder, and creates a kernel driver service using the standard APIs.

## Pushdo / Cutwail

### A study of the Pushdo / Cutwail botnet

To complete the setup the *TempFile* finishes by creating the following registry entry in each ControlSet, which is needed to execute the kernel driver service on system startup.

```
HKLM\SYSTEM\ControlSet001\Services\[SYSFILENAME]
    Group:      SCSI Class
    ImagePath:  System32\Drivers\[SYSFILENAME]
    Start:      0x00000000 (Boot – Loaded by Boot Loader)
    Type:       0x00000001 (Kernel Device Driver)
```

**Note:** The kernel driver will be loaded early in the boot process as a result of the Start and Group Values<sup>7</sup> allowing it to load before certain security products have the chance to detect it.

Finally the *TempFile* will delete itself before exiting, using the same method as the initial installer module:

```
cmd /c del [TEMPFILE] >> NUL
```

<sup>7</sup> <http://support.microsoft.com/kb/115486>

## Pushdo / Cutwail

A study of the Pushdo / Cutwail botnet

### STAGE 4: DOWNLOADED MODULES – CUTWAIL SPAM ENGINE

Debug Name: MailApp  
Project Name: bot15

The *Cutwail* Spam engine is one of the principal modules of the overall Pushdo family. It is a highly efficiently coded, optimized spam engine which is easily updatable as new spam campaigns are pushed out by the botnet owners.

Firstly *Cutwail* creates seven mutexes, which will be used by the threat to properly coordinate its highly multithreaded communication

- **gangrenb**
- **germeonb**
- **garbagb**
- **crypt32LogOffPortEvent**
- **memoryallocblock**
- **MACLink0**

The module also checks the contents of the hosts file (%SYSTEM\etc\driver\hosts) before starting its main subroutine – thread creation.

*Cutwail* sets up 60 new threads, all in suspended mode. After this, *Cutwail* starts its main communication with the C&C server.

The main execution thread opens a connection to a hardcoded IP address and port (in our testing this was 216.195.63.72 on port 5432 but its shown to vary quite frequently). This channel is then used for an encrypted conversation, allowing *Cutwail* to receive the settings needed for the spam run.

All information sent from the infected host to the C&C server consists of 32 bytes of information divided in 9 separate fields as follows:

Field #1 dd OSVer  
Field #2 dd Internal\_IP\_Address  
Field #3 dd Req  
Field #4 dd Control  
Field #5 dd ??  
Field #6 dd Windows version  
Field #7 dw DNSQuery  
Field #8 dw ??  
Field #9 dd ??

All server responses also follow a set structure:

Response\_Type dd  
Data\_Size dd  
...  
[Encrypted\_Data]

Beyond these first two double words the main content of the response is encrypted. *Cutwail*'s encryption is relatively straightforward. It uses a static key hardcoded in the *Cutwail* executable. The value of the key is the following string, which is a Russian phrase in reverse – roughly translating to "Screw you my friend":

## Pushdo / Cutwail

### A study of the Pushdo / Cutwail botnet

#### “reva gurd iuh an it ak-lehsoP”

The word Aver (reva reversed) in the above phrase may refer to the company Avermedia (<http://www.avermedia.com/>), distributors of computer hardware, who also have an office in Moscow. The attackers may simply dislike Avers hardware, or may even be disgruntled former employees.

Decryption occurs in 3 stages:

- Firstly *Cutwail* divides the encrypted string into blocks of length equal to the length of the current key (29 bytes in this case).
- Each block is then XORed with the key.
- The result is reversed (byte 1 and 29 are swapped, 2 and 28 etc)
- Even-numbered blocks (e.g. Block 2,4...) are also NOTed
- Finally the remaining bytes which do not fit into a full sized block are simply NOTed

Once decrypted each response has a different structure depending on the Response\_Type. In total 3 main exchanges of information occur in setting up *Cutwail*.

### Cutwail Initialisation: Stage 1 – Configuration Settings

The client's initial request is as follows: The first field is initially set to 0x00000000 unless the OSVersion registry entry was previously created by Pushdo (this registry key is described in Stage 2). The second field contains the internal IP address and this will remain the same during all subsequent communication. The Req field is initialized at 74h and will remain the same also. The Windows version is taken straight from the **GetVersion** API. In Windows XP SP2 (internal version 5.1.2600) this field has a value of 05 01 280A (280A is hex for 2600). The rest of the fields are zeroed out in this first request as shown in Figure 4.1.1:

```
00000000: 00 00 00 00-C0 A8 C7 81-74 00 00 00-00 00 00 00
00000010: 00 00 00 00-05 01 28 0A-00 00 00 00-00 00 00 00
```

Figure 4.1.1: Cutwail Initialization Request

The start of the server's response is shown in Figure 4.1.2. The initial response type is always 7. Please note the little-endian nature of these fields (0x00000007 is stored as 0x07000000 in Intel processors). The second field is the size of the data that follows. In initialization requests this is usually 194h bytes.

```
00000000: 07 00 00 00-94 01 00 00-1D 15 76 53-17 49 46 44
00000010: 4A 15 50 44-46 16 50 5C-20 1B 10 44-00 6B 2D 6C
00000020: D1 68 73 6F-51 E6 F4 FC-FB A9 F9 F9-8D AB E9 96
```

Figure 4.1.2: Cutwail Initialization Response

The rest of the information in the response is an encrypted payload. When decrypted it reveals configuration settings used for initialising the *Cutwail* module (Figure 4.1.3). These include a new IP address and port. Other settings include connection timeouts, maximum numbers of attempts, delays etc. These settings will replace the current control channel settings if they are different from the current hardcoded values. The default hardcoded values are:

- **Addr:** Varies (216.195.63.72 in our samples)
- **Port:** Varies (5432 in our samples)
- **Checksmtpdelay:** 120
- **Maxconn:** 512
- **Constconnect:** 1
- **Udpsockcount:** 16
- **Maxudptry:** 5
- **Udpptimeout:** 20
- **Knockdelay:** 20



## Pushdo / Cutwail

A study of the Pushdo / Cutwail botnet

```
00000000: 01 00 00 00-B4 00 00 00-61 64 64 72-00 32 31 36 00 01 addr 216
00000010: 2E 31 39 35-2E 36 33 2E-37 32 00 70-6F 72 74 00 00 .195.63.72 port
00000020: 35 34 33 32-00 6B 6E 6F-63 6B 64 65-6C 61 79 00 00 5432 knockdelay
00000030: 36 30 00 73-61 76 65 75-6E 6B 61 6E-73 77 00 00 60 saveunkansw
00000040: 74 75 72 62-6F 6D 69 6E-00 31 30 00-74 75 72 62 62 turbomin 10 turb
00000050: 6F 6D 61 78-00 31 32 00-6D 78 72 65-63 76 74 69 69 omax 12 mxrecuti
00000060: 6D 65 6F 75-74 00 31 32-30 00 6D 78-63 6F 6E 6E 6E meout 120 mxconn
00000070: 74 69 6D 65-6F 75 74 00-31 32 30 00-6D 61 78 74 74 timeout 120 maxt
00000080: 72 79 62 61-64 66 72 6F-6D 00 35 00-6D 61 78 74 74 rybadfrom 5 maxt
00000090: 72 79 63 6F-6E 6E 00 35-00 6D 61 78-74 72 79 65 65 ryconn 5 maxtrye
000000A0: 72 72 00 35-00 6D 61 78-74 72 79 62-6C 61 63 6B 6B rr 5 maxtryblack
000000B0: 00 33 00 6D-61 78 64 6F-6D 63 6F 6E-6E 00 32 00 00 3 maxdomconn 2
000000C0: 6D 61 78 63-6F 6E 6E 00-34 30 30 00-6D 61 78 75 75 maxconn 400 maxu
000000D0: 64 70 74 72-79 00 35 00-75 64 70 72-65 63 76 74 74 dptry 5 udprecvt
000000E0: 69 6D 65 6F-75 74 00 32-30 00 63 68-65 63 6B 73 73 imeout 20 checks
000000F0: 6D 74 70 64-65 6C 61 79-00 31 32 30-00 75 64 70 70 mtpdelay 120 udp
00000100: 73 6F 63 6B-63 6F 75 6E-74 00 34 00-63 6F 6E 73 73 sockcount 4 cons
00000110: 74 63 6F 6E-6E 65 63 74-00 31 00 68-65 6C 6F 73 73 tconnect 1 helos
00000120: 65 6C 65 63-74 69 66 68-6F 73 74 00-33 00 74 72 72 electifhost 3 tr
00000130: 79 70 69 70-65 6C 69 6E-69 6E 67 00-31 00 62 6F 6F ypipelining 1 bo
00000140: 74 5F 63 6F-6E 74 72 5F-64 69 76 00-00 64 69 65 65 t_contr_div die
00000150: 69 66 6E 6F-73 70 61 6D-00 30 00 64-69 65 69 66 66 ifnoscam 0 dieif
00000160: 6E 6F 73 70-61 6D 00 31-30 00 64 69-65 69 66 6E 6E nospam 10 dieifn
00000170: 6F 73 70 61-6D 00 30 00-64 69 65 69-66 6E 6F 73 73 ospam 0 dieifnos
00000180: 70 61 6D 00-30 00 64 69-65 69 66 6E-6F 73 70 61 61 pam 0 dieifnospa
00000190: 6D 00 30 00-00 00 00 00 00 00 00 00 00 00 00 m 0
```

Figure 4.1.3: Blob 1: Cutwail Initialization Settings

The first field of this response is usually equal to the value 1. The second is a continuation code which will be used in later messages – in this case the C&C server is requesting the client to follow up a control code of 0x000000B4.

### Cutwail Initialisation: Stage 2 - Handshake

The second request is similar to the first, but the Cutwail client uses the control code received previously. In the example, we can see that the code is 0x000000B4. The only other difference is the 00 04 in Field #8.

```
00000000: 00 00 00 00-C0 A8 C7 81-74 00 00 00-B4 00 00 00
00000010: 00 00 00 00-05 01 28 0A-00 00 04 00-00 00 00 00
```

Figure 4.2.1: Cutwail Request 2

The server response is also very similar to the initialization at first sight. In this case the Response\_Type code is 5 and the size of the encrypted data is only 16 bytes (0x00000010h)

```
00000000: 05 00 00 00-10 00 00 00-91 71 F2 FF-AB 34 86 10
00000010: 00 7C 2B B6-FF FF FF FF-00 00 00 00 00 00
```

Figure 4.2.3: Cutwail Response 2

The decrypted message starts out with 4 bytes giving us a bot id that will later be written to the following registry key as an infection marker.

**HKCU\SOFTWARE\Microsoft\OSVersion**

In the example, the value was 0x000D8E6E. Next is the external IP address of the client from the server's perspective. The third field is another id for the bot.

```
00000000: 6E 8E 0D 00-54 CB 79 EF-FF 83 D4 49-00 00 00 00
```

Figure 4.2.3: Cutwail ID Settings

## Pushdo / Cutwail

### A study of the Pushdo / Cutwail botnet

After receiving this message, the *Cutwail* client makes a reverse DNS connection to see if there is a domain name associated with the external IP address provided by the C&C server.

### Cutwail Initialisation: Stage 3 – Retrieve SPAM Content

With the third request packet, the client already has a bot ID and an infection marker so now it needs information to start spamming.

This time the new Infection Marker (OSVersion) is used in Field #1 and the reverse DNS response in Field #7 (in our testing we had no name associated with our IP, so this value was set to 2), with the rest stays the same as the last message.

```
00000000: 6E 8E 0D 00-C0 A8 C7 81-74 00 00 00-B4 00 00 00
00000010: 00 00 00 00-05 01 28 0A-02 00 04 00-00 00 00 00
```

Figure 4.3.1: Cutwail Request 3

The C&C server responds with the standard 2-field message with encrypted data following, this time with a Response\_Type of 8.

```
00000000: 08 00 00 00-C7 1B 03 00-07 17 58 0F-49 0B 14 1A
00000010: 0F 41 1A 5B-1E 4F 13 1A-53 06 34 41-0B 0C 4C 06
00000020: 65 68 73 6F-30 E2 EE E3-F7 B5 ED ED-8D 80 25 DE
```

Figure 4.3.2: Cutwail Response 2

The encrypted information is quite large (several hundred Kb) and contains a list of all of the email addresses to be targeted by the current campaign. The list of emails is organised in sets depending on which package the botnet's customer has opted to pay for – for example the list of emails may be for people in Moscow only, or businesses in St.Petersburg. More details on these packages can be seen in the “Self Promotion Spam” segment of our Spam Analysis discussion.

```
00000000: 60 00 00 00-00 6A 61 67-6A 61 40 6F-73 74 72 6F jagja@ostro
00000010: 76 2E 73 61-6B 68 61 6C-69 6E 2E 72-75 00 6D 78 v.sakhalin.ru mx
00000020: 31 2E 73 61-6B 68 61 6C-69 6E 2E 72-75 00 C3 48 i.sakhalin.ru H
00000030: FA 1B 00 67-75 6A 69 6A-74 6F 40 6F-6E 6C 69 6E + gujiito@onlin
00000040: 65 2E 72 75-00 72 65 6C-61 79 31 2E-6F 6E 6C 69 e.ru relay1.onli
00000050: 6E 65 2E 72-75 00 C2 43-01 0A 00 61-6B 6F 70 73 ne.ru rC@ akops
00000060: 65 69 6C 75-40 6E 76 70-74 75 73 2E-72 75 00 6D eilu@nuptus.ru m
00000070: 61 69 6C 2E-6E 76 70 74-75 73 2E 72-75 00 50 52 ail.nuptus.ru PR
00000080: A3 03 00 62-61 74 6E 65-75 40 69 6E-74 73 2E 72 u batneu@ints.r
00000090: 75 00 6D 61-69 6C 2E 69-70 2E 6E 63-6E 65 74 2E u mail.ip.ncnet.
000000A0: 72 75 00 4D-25 FE EE 00-61 6E 6F 61-76 61 40 6E ru Mz= anoava@n
000000B0: 5F 76 6F 73-68 69 70 63-72 65 77 69-6E 67 2E 72 vovoshipcrewin.r
000000C0: 75 00 6E 6F-76 6F 73 68-69 70 63 72-65 77 69 6E u novoshipcrewin
000000D0: 67 2E 72 75-00 51 1D 70-14 00 69 72-75 66 6E 65 g.ru Q*pfl irufne
000000E0: 6D 6F 40 6D-61 69 6C 2E-72 63 6F 6D-2E 72 75 00 mo@mail.rcom.ru
000000F0: 72 65 6C 61-79 31 2E 72-63 6F 6D 2E-73 70 62 2E relay1.rcom.spb.
00000100: 73 75 00 C3-F2 02 63 00-71 75 61 68-65 79 74 6B su l=gc quahcutk
```

Figure 4.3.3: Cutwail Targets

The first field is 60h (96) and this is followed directly by the spam recipients in a structure of three members each

Email\_address string  
Email\_server string  
Server\_IP\_Address dd



## Pushdo / Cutwail

### A study of the Pushdo / Cutwail botnet

In the example above, we can see the first email address ([jagja@ostrov.sakhalin.ru](mailto:jagja@ostrov.sakhalin.ru)), the associated Email Server (mx1.sakhalin.ru) and the IP address of this server (0xC348FA1B => 195.72.250.27).

This pre-computation of email address -> MX server -> IP address is one of the key reasons that *Cutwail* is capable of generating the sheer volume of spam that we have observed in our tests – it cuts out a lot of the overhead involved in sending a spam email message as the client does not need to send a DNS request for each email server

The next step in the information request phase is sending a fourth request with the only difference being a new value in Field #5.

```
00000000: 6E 8E 0D 00-C0 A8 C7 81-74 00 00 00-B4 00 00 00
00000010: 60 00 00 00-05 01 28 0A-02 00 04 00-00 00 00 00
```

Figure 4.3.5: Cutwail Request 4

By adding a value of 60h in Field #5, the server is instructed to provide additional information. This time the resulting response has a Response\_Type of 6. Once more the size of the encrypted data is quite large.

```
00000000: 06 00 00 00-2C 18 03 00-13 13 17 0D-73 14 14 3F
00000010: 1F 20 69 75-6A 20 61 6F-0B 69 74 20-74 6B 2D 6C
00000020: 65 68 73 7C-9F B1 BA A9-E3 EE F1 E6-E4 F6 BE F0
```

Figure 4.3.6: Cutwail Response 4

The encrypted data this time contains all of the templates needed for the spam runs. These are very customizable as can be seen in Figure 4.3.7 below – with subject lines, bodies, attachments, from/to fields etc all being easily customized.

A large amount of *Cutwail*'s spam is targeted at the Russian market and as such we see a large number of Cyrillic characters in these templates.

```
1 10 0 +0 0 {MasSlavas} {rusname1} {rusfamilii} <{generic_ru_mail}> {HTML_DECODE}{_BODY_HTML}/{HTML_DECODE}
2 <table width="737" border="2" cellpadding="15" cellspacing="6" bordercolor="#000000">
3 <tr>
4 <td width="707" bordercolor="#000000"><table width="711" border="0" cellspacing="0" cellpadding="10">
5 <tr>
6 <td width="128"><div align="center"><p><font size="+2"><strong><font color="#FE960A"><font color="#CC3300" face="Georgia,
8 <p><font color="#CC3300" size="+2"><strong><font face="Georgia, Times New Roman, Times, serif">Xioëôâ Riôôâiëôû Ââs
9 <p><font color="#CC3300" size="+2"><strong><font face="Georgia, Times New Roman, Times, serif">Xioëôâ eiâôû iôâiââë
10 </tr>
11 </td>
12 </tr>
13 <p align="center"><strong><font color="#0099FF" size="+3" face="Georgia, Times New Roman, Times, serif">ôîâââ - yôi
14 <p align="center"><strong><font color="#666666" face="Georgia, Times New Roman, Times, serif">Ñôî-iây, <font color=
15 <div align="center">
16 <table width="721" border="0" cellspacing="0" cellpadding="0">
17 <tr>
18 <td width="541" valign="top"><p align="center"><font size="+1"><strong><font color="#666666" face="Georgia, Times
19 iiiiâôô Ââi iâiëôë éiëôôâiôîâ è ââôîôôû iôâiââë!</font></strong></font></p>
20 <p align="center"><font size="+3"><strong><font color="#CC3300" face="Georgia, Times New Roman, Times, serif">ÂÊÂÊ
21 <td width="170"><div align="center">
23 </td>
24 </tr>
25 <p align="right"><font size="+2"><strong><font color="#CC3300" face="Georgia, Times New Roman, Times, serif">Dâqââ
26 <p><strong><font size="+1" face="Georgia, Times New Roman, Times, serif">ôâë:</font><font size="+3"> </font><tel5423
27 </tr>
28 </table>
29 </div>
30 Date: {DATE}
31 From: {TAGMAILFROM}
32 Return-Path: <{generic_ru_mail}>
33 X-Priority: 3 (Normal)
34 Message-Id: <{DIGIT[10]}.{DIGIT[14]}@{MAILFROM_DOMAIN}>
35 To: {MAIL_TO}
36 In-Reply-To: <{nHEX[44]}@{MAILTO_DOMAIN}>
37 References: <{nHEX[44]}@{MAILTO_DOMAIN}> <{nHEX[32-44]}@{MAILFROM_DOMAIN}>
```

Figure 4.3.7: Cutwail spam Templates

## Pushdo / Cutwail

### A study of the Pushdo / Cutwail botnet

After this, the client asks for the last piece of information needed. The fifth request is the same as the fourth with one small addition: Field #9 has now a new value of 14h (20).

```
00000000: 6E 8E 0D 00-C0 A8 C7 81-74 00 00 00-B4 00 00 00
00000010: 60 00 00 00-05 01 28 0A-02 00 04 00-14 00 00 00
```

Figure 4.3.8: Cutwail Request 5

The last response is the same format as before, with a Response\_Type of 1, and a size of 5cc9 (23,753) bytes

```
00000000: 01 00 00 00-C9 5C 00 00-0E 65 76 61-20 67 00 00
00000010: 4A 4E 0A 00-1D 20 61 6E-20 96 8B DF-9E 6B 2D 30
00000020: A4 68 73 7C-9F 8D 9A 8B-9E BE F5 EF-FE E7 DF 96
```

Figure 4.3.9: Cutwail response 5

This final piece of information, once decrypted, reveals lists of first names, surnames and domains – which are used to create the fake email address used in the “From” field of the spam emails.

```
00000000: CF 13 00 00-C1 5C 00 00-FF FF FF FF-00 00 00 00 x!! 1\
00000010: 75 75 63 6E-2E 72 75 00-00 00 00 00-7C 61 6E 64 ucn.ru land
00000020: 72 65 79 5F-76 6F 69 74-65 6E 6B 6F-00 01 00 00 reg_voitenko @
00000030: 00 7C 73 65-6D 61 00 02-00 00 00 7C-62 6F 72 67 isena @ iborg
00000040: 6F 6C 6F 76-61 00 FF FF-FF FF 00 00-00 00 6D 74 olova mt
00000050: 74 2E 72 75-00 03 00 00-00 7C 61 73-64 00 04 00 t.ru iasd
00000060: 00 00 7C 70-6B 75 64 72-69 61 73 68-6F 76 61 00 ipkudriashova
00000070: 05 00 00 00-7C 67 75 6C-6E 61 72 61-00 FF FF FF igulnara
00000080: FF 00 00 00-00 73 63 61-6E 2D 6E 65-76 61 2E 72 scan-neva.r
00000090: 75 00 06 00-00 00 7C 73-6B 76 6F 72-74 73 6F 76 u iskvoortsov
000000A0: 2E 61 00 FF-FF FF FF 00-00 00 00 75-63 61 72 2E .a ucar.
000000B0: 65 64 75 00-07 00 00 00-7C 6D 70 6F-77 65 72 73 edu impowers
000000C0: 00 FF FF FF-FF 00 00 00-00 72 6D 62-6C 2E 72 75 rmb1.ru
000000D0: 00 08 00 00-00 7C 33 64-6B 69 6E 69-61 70 69 6E idkiniapin
000000E0: 00 FF FF FF-FF 00 00 00-00 6E 65 74-7A 65 72 2E netzer.
000000F0: 72 75 00 09-00 00 00 7C-6C 65 74 6F-00 FF FF FF ru ileto
00000100: FF 00 00 00-00 6B 6C 6C-76 6F 7A 2E-72 75 00 0A klbooz.ru
00000110: 00 00 00 7C-61 6C 6C 61-31 39 36 38-00 FF FF FF ialla1968
00000120: FF 00 00 00-00 6D 6E 64-75 6C 75 2E-61 6E 67 2E mdulu.ang.
00000130: 61 66 2E 6D-69 6C 00 0B-00 00 00 7C-6E 69 65 6C af.mil iniel
00000140: 6C 65 2E 6C-75 75 6B 6B-6F 6E 65 6E-00 FF FF FF le.luukkonen
00000150: FF 00 00 00-00 70 65 6F-2E 67 64 2E-6D 6F 73 65 peo.gd.mose
00000160: 6E 65 72 67-6F 2E 65 6C-65 6B 74 72-61 2E 72 75 nergo.elektra.ru
00000170: 70 0C 00 00-00 7C 33 64-6E 69 6B 6E-6C 73 6B 69 8 idnikolski
```

Figure 4.3.10: Cutwail Sender details

## Cutwail Main Routine:

Having received all of its setup information *Cutwail* finally sets up all of its spam threads.

- Some of these threads are used to carry out DNS requests, which unusually will bypass the local DNS server, instead making queries directly to the root DNS nameservers<sup>8</sup>.
- One thread constantly resets the “OSVersion” registry entry, used as an infection marker for the threat.
- One thread attempts to connect to several hardcoded MX servers such as mx.google.com, msx.mail.ru and mx2.messagingengine.com. This appears to be used by the threat to ensure that it is online, and that such connections are not being blocked by a firewall.
- Provided the connection attempts of the previous thread are successful, all other threads are responsible for the main spam sending of the *Cutwail* module, with the full spam run being distributed evenly across each of them.

These threads will continue to run until the entire spam run has completed. Then the *Cutwail* module will enter a cycle of requesting a new spam run and sleeping for a period of time. Once settings for a new spam run are received, *Cutwail* will restart its execution with the new settings. These waves can clearly be seen in Figure 8.6.2 in the section of this report detailing spam analysis.

<sup>8</sup> [http://en.wikipedia.org/w/index.php?title=Root\\_nameserver&oldid=284118253](http://en.wikipedia.org/w/index.php?title=Root_nameserver&oldid=284118253)

## Pushdo / Cutwail

### A study of the Pushdo / Cutwail botnet

In our testing we noted that different infected nodes will receive separate spam campaign runs. When starting two independent nodes at the same time we found that one was ordered to send pharmacy spam and the other to spam advertising pornographic websites. Both of these are covered in the section of this report covering spam analysis.

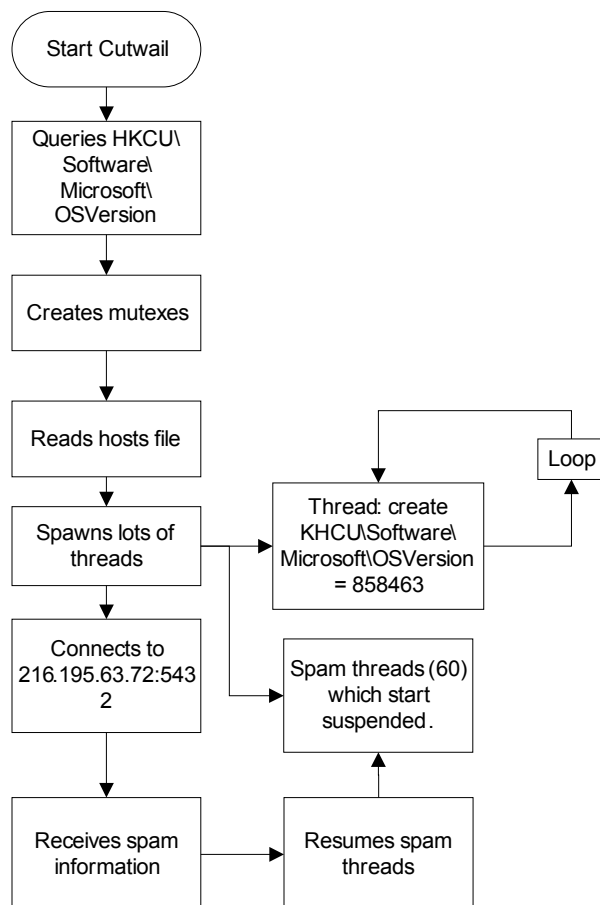


Figure 4.4: Stage 4: Downloaded Module – Cutwail spam Module

## Pushdo / Cutwail

A study of the Pushdo / Cutwail botnet

### STAGE 5: DOWNLOADED MODULES – PUSHDO KERNEL MODULE

**Debug Name:** protect (drops innerdriver)  
**Project Name:** Siberia2

Pushdo's *Kernel Module* is launched very early in the system boot process (more details are available in the section explaining Stage 3). The *Kernel Module* is responsible for ensuring that Pushdo remains installed on the machine and executing in memory. It also acts as an additional load point for the threat.

The actual *Kernel Module* is itself a dropper - on execution, it first checks the current Windows Version. Next, it unpacks a section of its own code and injects this code into *services.exe*. This code, which we will call *Inner Kernel Driver*, is responsible for the main Kernel operations of Pushdo.

*Inner Kernel Driver* checks all of the Registry keys associated with the kernel module to ensure that they have not been removed from the system. It also creates an object named **Prot3** which is used as a means of communication with the Pushdo user mode processes.

Once these setup components complete, the module starts its main subroutines. It sets up three callback procedures so that the driver will be notified of any system changes, allowing it to monitor almost every event on the system. This is done via the following three kernel API calls:

- **IoRegisterFsRegistrationChange** – This routine registers a file system filter driver that will be notified whenever a file system registers or unregisters itself. These events trigger during the Windows startup process or for example anytime a USB stick / removable drive is attached to the machine. Whenever one of these events triggers the *Inner Kernel Module* decodes the main Downloader module and injects it (in the normal manner) into *services.exe*.
- **CmRegisterCallback** – This routine registers a registry callback routine, allowing the calling driver to monitor, block, or modify a registry operation. The driver intercepts registry manipulation attempts and blocks those associated with the Pushdo threat. This acts as a means to protect the registry entries that autostart the threat.
- **PsSetCreateProcessNotifyRoutine** – This routine adds a driver-supplied callback routine which notifies the driver each time a process is created or deleted. When one of these events triggers, the driver checks if it is a creation event for *services.exe*. If so, it will also decode the main Downloader module and inject it (in the normal manner) into that *services.exe* process.

Having setup these callback hooks, the *Kernel Module* checks the registry entries associated with the kernel once more, ensuring that the threat is still correctly installed.

## Pushdo / Cutwail

A study of the Pushdo / Cutwail botnet

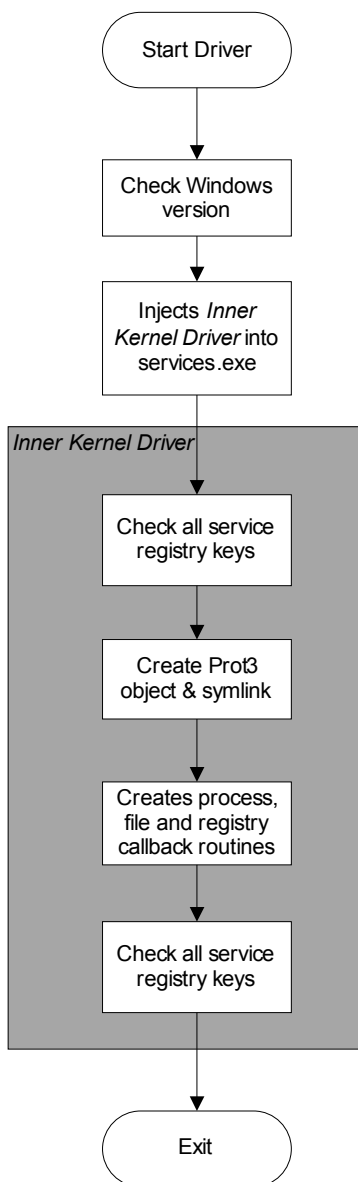


Figure 5.1: Stage 5: Downloaded Modules – Pushdo Kernel Module

## Pushdo / Cutwail

A study of the Pushdo / Cutwail botnet

### STAGE 6: DOWNLOADED MODULES – PUSHDO SNIFFER MODULE

Debug Name: MailSniff  
Project Name: Sniff4

Pushdo's sniffer module is responsible for helping the botnet owners maintain statistics on the spam email being sent by the infected machine, and possibly also as a means to harvest new email addresses.

The Downloader module first saves the sniffer module to the following location:

**%WINDOWS%\systems\drivers\tcpsr.sys**

The downloader then executes the sniffer as a kernel device driver. This allows the sniffer module to hook the networking functions of ndis.sys, which it use to sniff all SMTP traffic going to external machines on port 25. It then extracts and stores all email recipients.

Have executed the sniffer, the main downloader module removes the tcpsr.sys file from disk to further increase the stealthy nature of Pushdo.

The downloader module actively communicates with the sniffer driver in order to recover this recipient data. Once the data gathered reaches a certain amount, it is sent to a server hosted on **216.195.58.115** via a HTTP POST connection. This IP address is hardcoded but could be changed over time. This sequence continues as long as the machine is generating spam emails.

This method is used by the threat in order to keep detailed statistics on not only the volumes of spam sent, but also which recipients have already been emailed. This allows Pushdo to ensure it has delivered the pre-agreed volumes of spam for its customers.

One interesting side effect of this method is that the sniffer module does not discriminate between spam email and any legitimate mail traffic being sent by the machine. As a result and legitimate email recipients will be harvested, and can be added to the criminal gang's database of email addresses.

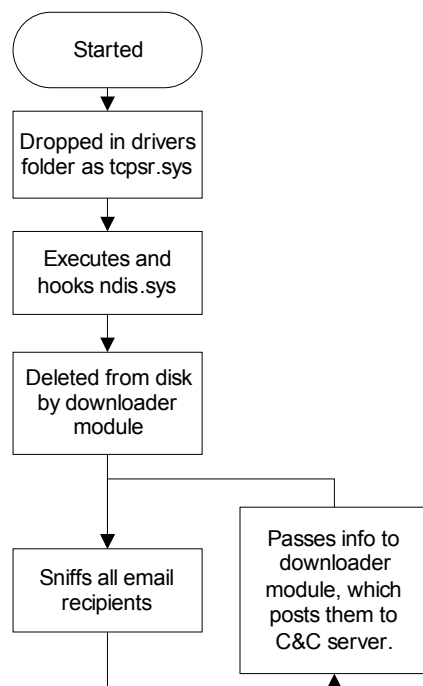


Figure 6.1: Stage 6: Downloaded Modules – Pushdo Sniffer Module



## CAMPAIGN MODULES

One of Pushdo's defining features as a threat is its ability to serve up malware belonging to other malware groups. During our research we observed many examples of downloaded third party malware, all with quite different behaviour. During each major update Pushdo will normally reinfect the machine with a new wave of third party malware, sometimes uninstalling the previous malware first (most likely depending on whatever financial arrangement the authors of the third party malware have agreed with the Pushdo owners).

We have included an example of these “Campaign Modules” below. Other malware families commonly installed by Pushdo include TROJ\_RENOS, TROJ\_FAKEAV and a variety of droppers, downloaders and backdoors.

## CAMPAIGN MODULES: EXAMPLE: POPUP AD

One example of the third Party malware modules which are served by Pushdo is this “Popup Ad” module, which Trend Micro detect as BKDR\_INJECT.SZ.

The module is injected into a svchost process by the original Pushdo downloader module. Once executed it starts by creating a mutex with a value similar to:

**1.5.2-53657468—466041896**

Next it sets up a connection from a random port on the infected machine to a random port on the IP address resolved to by <http://sys215.3fn.net> (this changes regularly due to a Fast Flux backend), which acts as it's C&C server. All communication is then sent across this connection, starting with a setup request via HTTP (but using these two pre-established ports instead of the default port 80). The full URL is formatted as follows:

**http://sys215.3fn.net:3110/?bot\_id=0&mode=1**

In this case the random port used was 3110. This server will then respond with a series of commands for the malware module, which are sent in an encrypted format:

```
1 <!doctype html public "-//W3C//DTD HTML 4.0 Transitional//EN"><html><body bgcolor = #DEDEDE>  
2 <br><h5 align = center><font color = green>incoming data has been passed</font></h5>  
3 <br><form name = "request" action = "/?bot_id=1170714744" method="POST">  
4 <input type=hidden name="bot_id" value="1170714744">  
5 <input type=hidden name="mode" value="11">  
6 <table width = "100%" border = 1 align = center>  
7 <tr>  
8 <th align = center> Campaign <nbsp;   <input type = text size = 5 value = "??"></tr>  
9 <tr><td align = center >Task</td></tr>  
10 <tr><td align = left>  
11 <!-- script [??]-->  
12 <textarea wrap = off cols=140 rows = 15 name=script ?>  
13 X9bc1wgrKRNTEtguXUj1RfRDYR5E8TGyUlH12aCcXXmTQ3xV/BINJEjSBTWkWRM8OgcqClOhoI0lkHal1EYhBGeTS8UGoaPEFfbasKJRwkCCkStWBgV  
14 Tdws6msWhnL1ZNI2N9Heho6P3pjJ5YMcS4UlnToTKfESJQCZCIxy1vSRiVSQGzNBHSWTABSHBxpGQOrAeEAcmtuHXODAhITxtPnjjiBE1EOxyDZocEicp  
15 UiVoZgQyZGsWTOo+AxN6NUOYzG4aTF1ECE9kBAsOfTxpNLWhy&zdcQTQM1UgkUZfg4G12OawY2UVOFNVNRSRGZYblLWEYkL2SZWj96Jxx7TOhjThNsCtNPDo/  
16 e1cc0c07ek9bQRyIRkobghaksakk/J19HWRoYK2NOASjRDBVl2TNzfEofAyt4RRSzbfKeWeC0glROaEMzbGAEGlf4EmJ2NFMTSHAZnWhFF18MatJYOITVHRn  
17 B32BWjYqtckSe2tBYU1atAgrQXYGEF8Q3JioAFNWmaIKOPFGBHffDVozigldVFpsAhpBaAqrAiRcvTEGkdFYxgrTDw6K2WSDuLiIE1HcGwbIX88GC1GHG3  
18 SzdzRE5fSpfZYehck3GSZU84TUqvYe1zGEVIMkIPDG2IKBhpdmMYKWO8oitUrX45jtsT3caB6pjXKEYbYcnZz4rUodFZB43c0IdGEYkEO1PRFN5THIKFOF0  
19 GctYL1fhZ1vlVmMaJPoCDZAG9GBEa2OKwRYBSOEUA2YEPmJYBlEXdx4WahKcULIME1IQNNLCVvtIOYD4vaHccUW8bYmRuOpdvZW43AU873tSPSEUN
```

**Figure 7.1: Campaign Module: Popup Ad – Setup Response**

## Pushdo / Cutwail

### A study of the Pushdo / Cutwail botnet

Two interesting values are highlighted above - the Bot ID and main payload of the threat. The bot\_ID is responsible for uniquely identifying this malware. The main payload however is more interesting – to decrypt it the malware first decodes it using BASE64, before XORing the payload in blocks with the following 16 byte HEX decryption key:

**7B2BOC712A1D1D2A717B2B0C717B2B48**

This decoded content contains a large amount of settings in an XML format, which is parsed by the threat. This first instructs the threat to download three image files from <http://leznha.bay.livefilestore.com/>. Next the threat visits a Russian site <http://www.pochtamt.ru>, which provides free anonymous web hosting services.

It signs up for a new account using a random name and password. Unusually for web hosting, no email address is required to open an account, but there is however a captcha which needs to be correctly input. The threat sends the captcha image back to its C&C server, which appears to be running software capable of breaking this particular type of captcha – as it replies with the correct value.

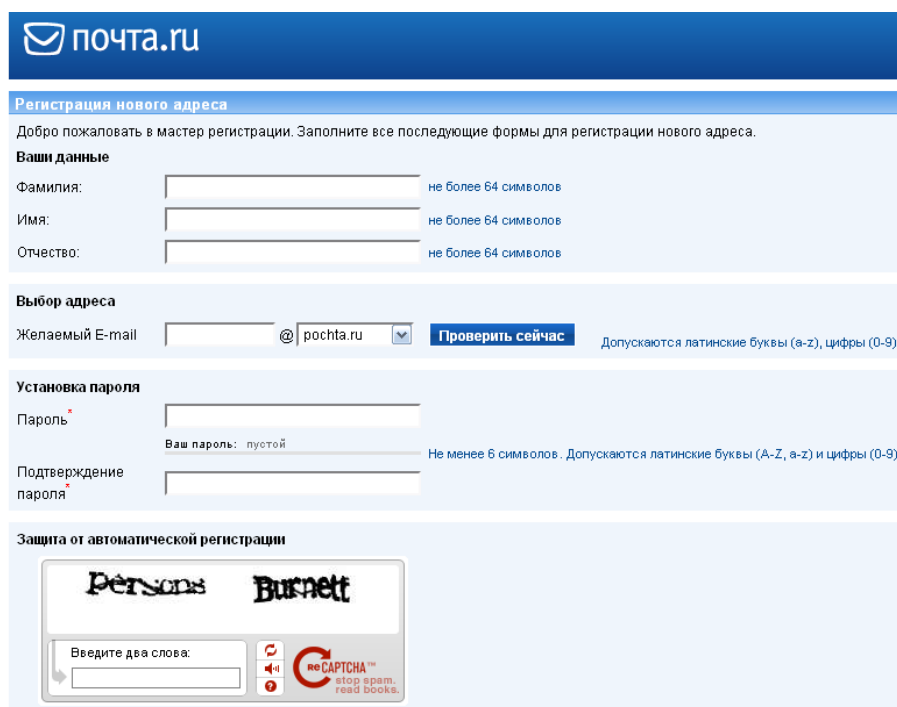


Figure 7.2: Campaign Module: PopUp Ad – Pochtamt.ru

Once it has successfully broken the captcha and setup a new account, it uploads the three images downloaded previously, along with a randomly named HTML file to its new web hosting account.

Finally it opens a visible Internet Explorer Window displaying the randomly named HTML file on the pochta.ru server, which appears as follows:



## Pushdo / Cutwail

### A study of the Pushdo / Cutwail botnet



Figure 7.3: Campaign Module: Popup Ad – Popup advertisement

Clicking anywhere on the image presented (actually made up from all three images) will result in the user being redirected to the site being advertised by the malware, [www.fresh-serial.ru](http://www.fresh-serial.ru), which in turn will set off a whole chain of nasty exploits and web threats

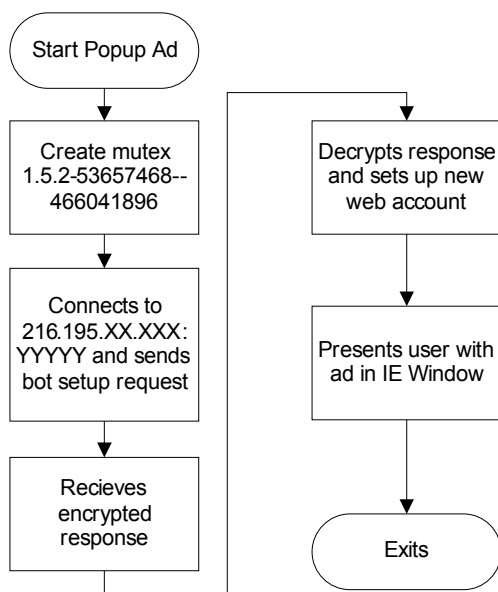


Figure 7.4: Campaign Module: Popup Ad – Execution Flow

## SPAM ANALYSIS

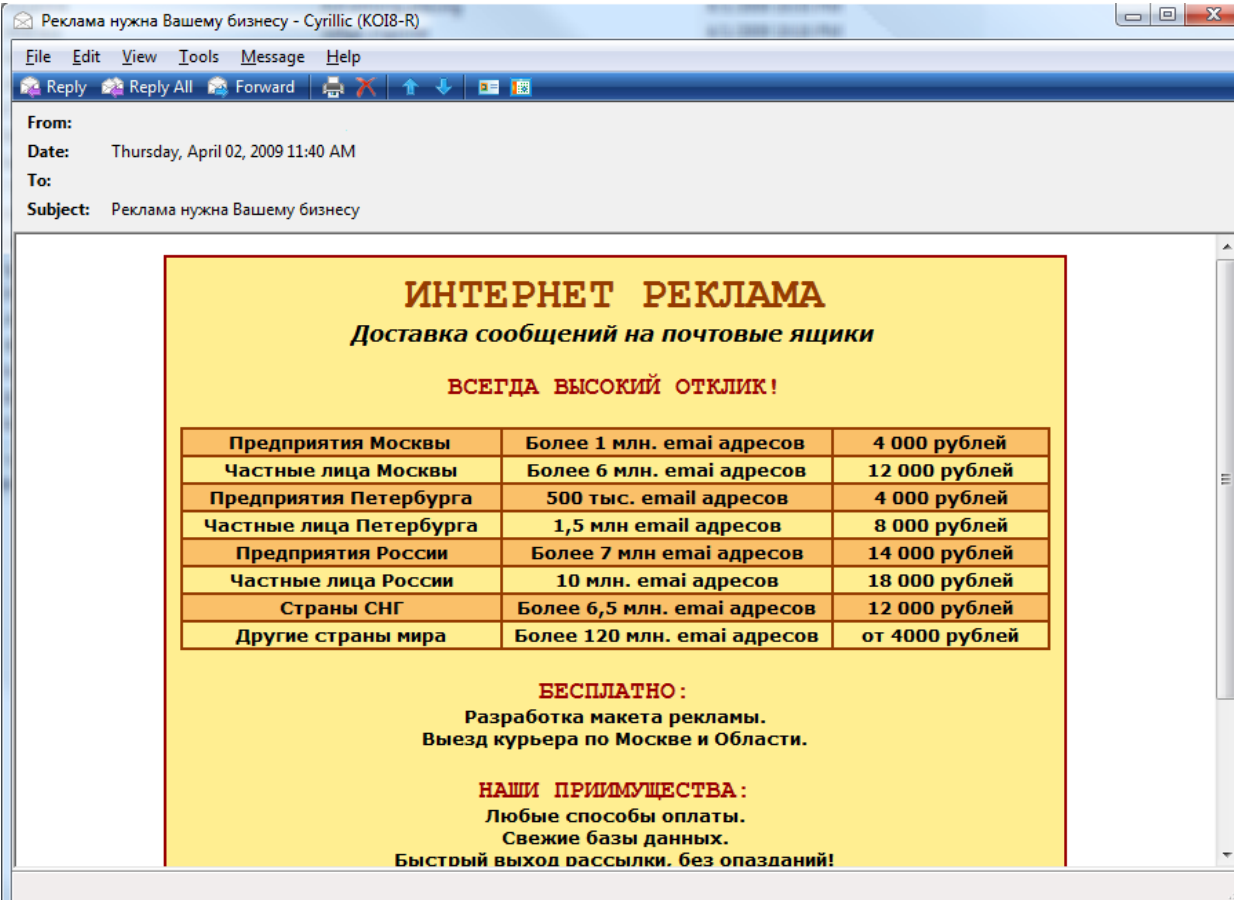
According to reports by SecureWorks from April 2008<sup>9</sup> and Messagelabs in January 2009<sup>10</sup> Cutwail is one of the most aggressive spam generating botnets on the planet. Messagelabs estimated that the botnet sent an average of almost 8 billion spam emails per day, putting it second in terms of botnet spam volume, and ahead of such big names as Rustock and Xarvester. As part of our analysis of the network behaviour of the threat, we have identified the following spam trends.

### SPAM WAVES

#### Spam Wave 1: Self Promotion Spam

These waves send mails advertising the botnet's own ability to spread spam as a medium for advertising. The botnet's owners use this as a method to attract new advertising partners.

Prices for this service start from as little as 4000 rubles (about €90), and are very customizable for the client (see below):



**ИНТЕРНЕТ РЕКЛАМА**  
*Доставка сообщений на почтовые ящики*

**ВСЕГДА ВЫСОКИЙ ОТКЛИК!**

Предприятия Москвы	Более 1 млн. email адресов	4 000 рублей
Частные лица Москвы	Более 6 млн. email адресов	12 000 рублей
Предприятия Петербурга	500 тыс. email адресов	4 000 рублей
Частные лица Петербурга	1,5 млн email адресов	8 000 рублей
Предприятия России	Более 7 млн email адресов	14 000 рублей
Частные лица России	10 млн. email адресов	18 000 рублей
Страны СНГ	Более 6,5 млн. email адресов	12 000 рублей
Другие страны мира	Более 120 млн. email адресов	от 4000 рублей

**БЕСПЛАТНО :**  
Разработка макета рекламы.  
Выезд курьера по Москве и Области.

**НАШИ ПРИМУЩЕСТВА :**  
Любые способы оплаты.  
Свежие базы данных.  
Быстрый выход рассылки. без опозданий!

Figure 8.1.1: Self Promotion spam

<sup>9</sup> <http://www.secureworks.com/research/threats/topbotnets/?threat=topbotnets>

<sup>10</sup> [http://www.messagelabs.com/mlireport/MLIReport\\_2009.01\\_Jan\\_Final.pdf](http://www.messagelabs.com/mlireport/MLIReport_2009.01_Jan_Final.pdf)

## Pushdo / Cutwail

### A study of the Pushdo / Cutwail botnet

The above email translates to:

INTERNET ADVERTISING  
Delivery of messages to mailboxes  
STAY HIGH RESPONSE!

Companies in Moscow	more than 1 million email addresses	4 000 rubles
Individuals in Moscow	more than 6 million email addresses	12 000 rubles
Enterprises of St. Petersburg	500 thousand email addresses	4 000 rubles
Individuals of St. Petersburg	1.5 million email	8 000 rubles
Companies Russia	Over 7 million email addresses	14 000 rubles
Individuals Russia	10 million email addresses	18 000 rubles
The CIS countries	over 6.5 million email addresses	12 000 rubles
Other countries in the world	Over 120 million email addresses	From 4000 rubles

FREE:  
Graphic Design, Advertising.  
Check courier to Moscow and the region.

OUR PROMISE:  
Any forms of payment.  
Fresh data base.  
Quick access subscribers, without delay!  
Always a high response rate of advertising.  
If you order 2 shots, 3-for FREE!

Figure 8.1.2: Self Promotion spam (Translated)

Above emails also contain contact numbers and ICQ numbers by which a customer can contact the botnet owners to avail of their services. As part of our investigations we had a member of our team pose as an interested customer and contact the group, more details of which can be found in the section "Behind the Malware – Botnet Owners"

## Pushdo / Cutwail

A study of the Pushdo / Cutwail botnet

### Spam Wave 2: Porn Sites

Another type of spam sent by the threat is the very common practice of sending links to sites with pornographic content. The advertised porn sites are generally Russian language sites, which have most likely paid the botnet group.

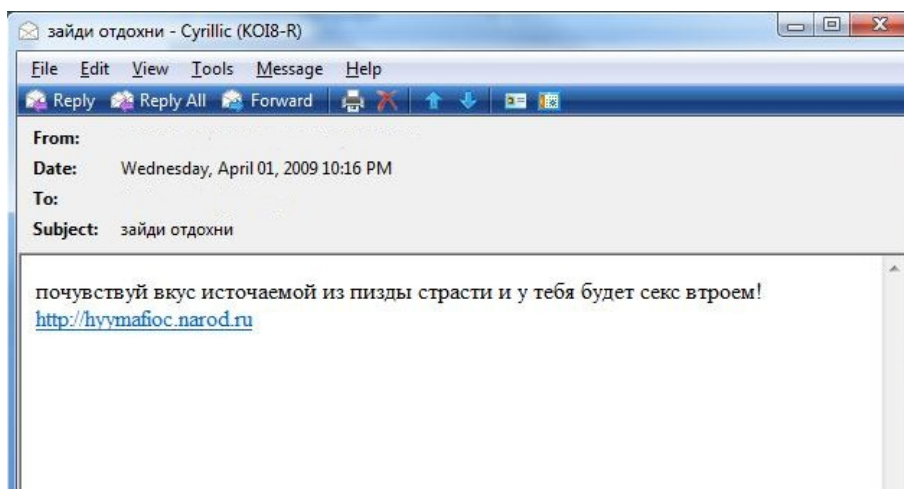


Figure 8.2.1: Porn Site Spam

In our testing we came across many different URLs for these pornographic sites. Normally upon accessing these URLs the browser is automatically redirected to the real porn site via a HTTP 503 redirect message.

### Spam Wave 3: Pharmacy Spam

Cutwail is also one of several botnets involved in the spread of the very popular “Canadian Pharmacy Spam”. These spam runs have also been used by several other botnets such as Storm. Unlike porn websites which make money out of paid membership areas, pharmacy spam uses the lure of cheap drugs to extract money from users.

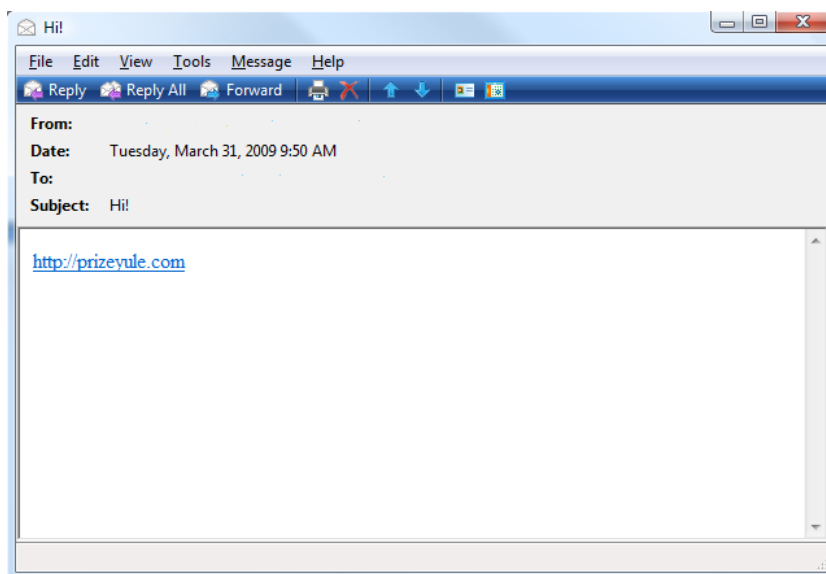


Figure 8.3.1: Canadian Pharmacy Spam Mail



Figure 8.3.2: Canadian Pharmacy Site

A full detailed description of such spam waves can be found on the SpamTrackers.eu wiki<sup>11</sup>

<sup>11</sup> [http://www.spamtrackers.eu/wiki/index.php?title=Canadian\\_Pharmacy&oldid=5842](http://www.spamtrackers.eu/wiki/index.php?title=Canadian_Pharmacy&oldid=5842)

### Spam Wave 4: Prestige Replica Spam

Again this type of spam is not unique to the Cutwail Family, and is again very well documented on SpamTrackers.eu<sup>12</sup>. These spam waves offer users high quality replica watches (Rolex, Cartier, etc) for low prices.

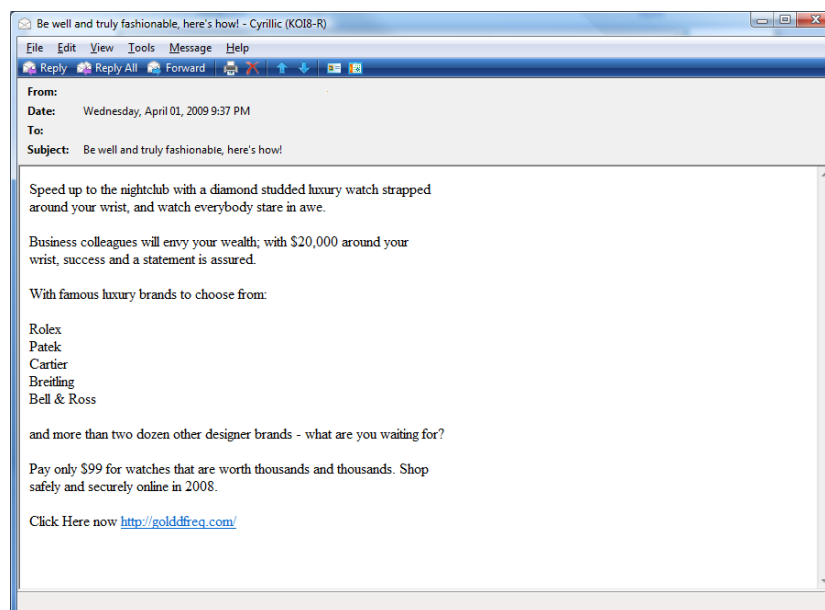


Figure 8.4.1: Sample Replica spam

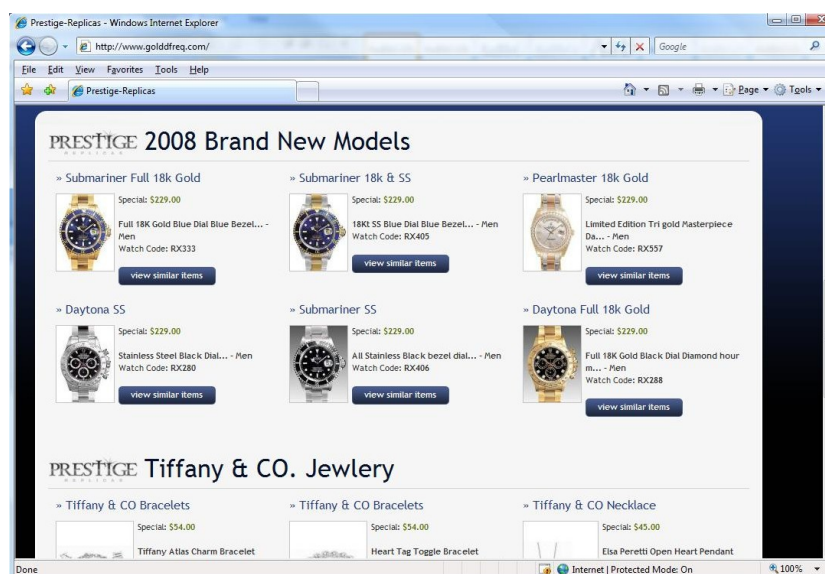


Figure 8.4.2: Sample Replica Site

<sup>12</sup> [http://www.spamtrackers.eu/wiki/index.php?title=Prestige\\_Replicas&oldid=5171](http://www.spamtrackers.eu/wiki/index.php?title=Prestige_Replicas&oldid=5171)



## Spam Wave 5: Local Advertising Spam

One part of *Cutwail's* spam that we found particularly interesting is their spammed advertisements for local businesses. We assume these emails are sent on behalf of local businesses who have contacted the botnet owners, having received one of their Self-Promotion emails, and decided to pay for their services.

We have seen example of everything from local salsa classes, and tickets for the upcoming Eurovision song contest, to lawyers services.

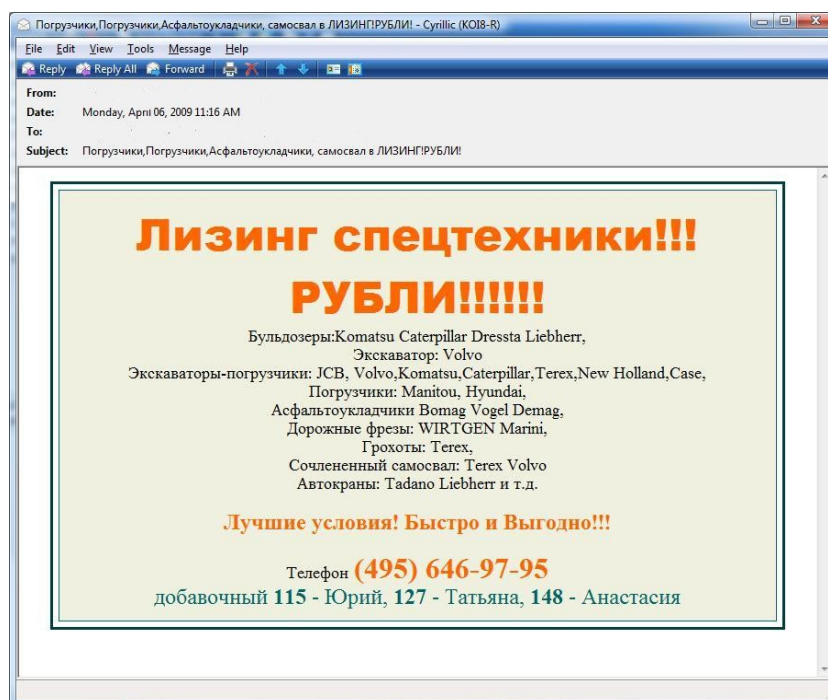


Figure 8.5.1: Construction Machinery Suppliers

Interestingly, when we contacted one of the lawyers advertising their services this way, and told them that we had received their advertisement via spam, they claimed to have no idea what we were talking about.

In this case it is possible that the individual in the lawyer's office did not wish to admit to such a form of advertising or equally likely that the firm had paid a third party to advertise their services – and it was this third party which in turn employed the services of the botnet owners. These third party resellers are fairly common in online advertising circles.

# Pushdo / Cutwail

## A study of the Pushdo / Cutwail botnet

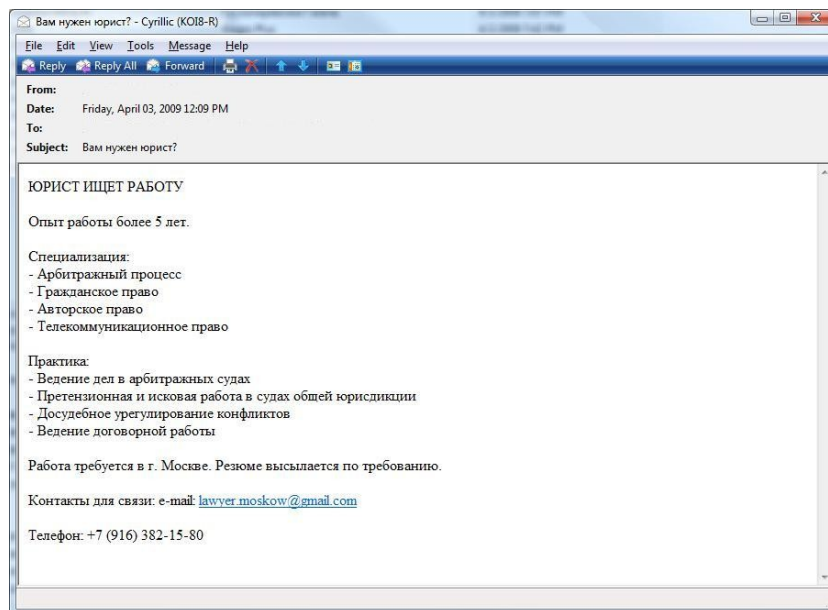


Figure 8.5.2: Lawyers Firm



## Pushdo / Cutwail

A study of the Pushdo / Cutwail botnet

### Spam Statistics

On our monitoring system for *Cutwail* we generated statistics on the number of emails the threat would send per hour, per day etc – to better understand how the threat operates.

As can be seen in the graph below, Cutwail sends spam in waves of approximately 5 hours in length, with breaks of about 1 hour in between. This behaviour averages at 117 msgs/min over the course of a day, but the volume of emails is often double that during periods of high activity. In some cases (where the length of the spam emails is low) we have seen volumes in excess of 800 msgs/min.

At these rates our single infected botnet node would send approximately 60 million mails per year. The reason Cutwail can send spam at these rates is due to the efficient, multithreaded nature of its spam engine, which has been described in more detail in the “Malware Analysis” section of this report

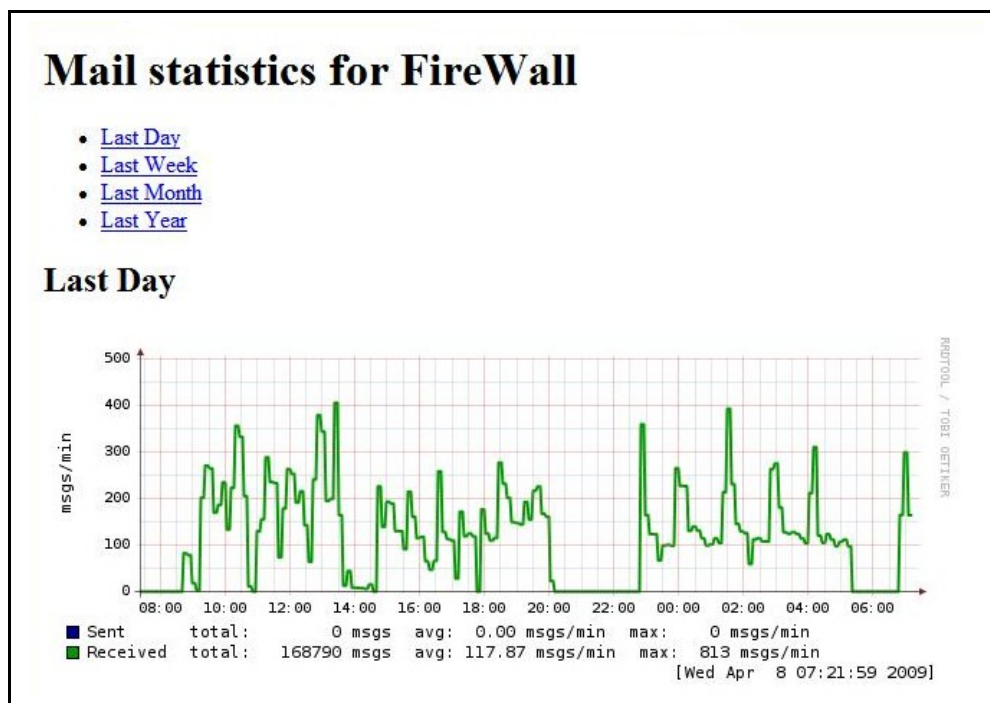


Figure 8.6.2: Email Statistics

In our testing we found that different infected nodes are configured to send separate campaigns – so it appears that the workload is split over subsets of the botnet (as opposed to each node sending the same content). On an average week our nodes sent approximately 35 different spam runs each. At the basic cost of 4000 rubles per spam run this would net the botnet creators a minimum of \$3500 per week, although it is likely that the real figure is far in excess of this.

### BEHIND THE MALWARE – BOTNET OWNERS

As part of our research into the motivations and financial workings behind the botnet, we decided to directly contact the botnet owners and ask them. Contact details were easily obtained from one of the self promotion spam messages – in this case an ICQ number and a direct landline phone number.

We decided it would be safer to contact the ICQ number first. We tried several times (and several of the advertised ICQ numbers) to get information on pricing by sending Russian messages via ICQ – but in each case we were simply sent a message containing all of the same details found in the original spammed email. It appears that the botnet owners have simply setup an automated bot, which replies to any messages sent to the list of ICQ accounts with their latest pricing information. Seeing as this was fruitless we decided to directly call the botnet owners.

Firstly we acquired a normal Russian mobile phone from which to make the call. The number we contacted was +7(495)585-69-60, which is a landline number registered in Moscow. We made 2 separate calls – the first time posing as a private person wishing to advertise an erotic website, and the second time wishing to advertise transportation services.

The person who responded had no problems catering for either business. They first pointed us to the website <http://advert1.ru> which contains all of their pricing information (See Figure 9.1). This is far more complete than the shorter version which can be seen in Figure 8.1.2. We have translated the “Region” column into English, but it appears in Russian on the website. Note all prices are in Rubles. At the time of writing 1000 Rubles was worth approximate €22.

	Region	Number of Addresses	Price per Million	Price for All
	All Russia	16.000.000	4.000 p	25.000 p
	Russia Private	9.840.000	4.000 p	18.000 p
	Russia Business	6.101.000	4.000 p	10.000 p
	Moscow & Moscow Region	6.840.000	4.000 p	12.000 p
	Moscow Private	5.93700.000	4.000 p	10.000 p
	Moscow Business	908.000	4.000 p	3.500 p
	All St Petersburg	1.599.000	4.000 p	9.500 p
	Petersburg Private	292.000	4.000 p	3.500 p
	Petersburg Business	1.307..000	4.000 p	7.000 p
	Austrailia	10.000.000	4.000 p	20.000 p

# Pushdo / Cutwail

A study of the Pushdo / Cutwail botnet

	Argentina	1.000.000	4.000 p	2.500 p
	United Kingdom	8.000.000	4.000 p	22.000 p
	Germany	10.000.000	4.000 p	25.000 p
	Denmark	2.000.000	4.000 p	5.000 p
	Israel	1.000.000	4.000 p	4.000 p
	Italy	2.000.000	4.000 p	6.000 p
	Canada	2.000.000	4.000 p	6.500 p
	Crimea	100.000	-	3.000 p
	MIX	500.000.000	4.000 p	Contract
	Norway	1.000.000	4.000 p	3000 p

## Pushdo / Cutwall

A study of the Pushdo / Cutwall botnet

	<b>Singapore</b>	<b>100.000</b>	<b>-</b>	<b>4.000 p</b>
	<b>USA</b>	<b>50.000.000</b>	<b>4.000 p</b>	<b>100.000 p</b>
	<b>Turkey</b>	<b>2.000.000</b>	<b>4.000 p</b>	<b>5.500 p</b>
	<b>Ukraine</b>	<b>1.000.000</b>	<b>4.500 p</b>	<b>4.500 p</b>
	<b>Florida</b>	<b>100.000</b>	<b>4.000 p</b>	<b>2.500 p</b>
	<b>France</b>	<b>3.000.000</b>	<b>4.000 p</b>	<b>7.000 p</b>
	<b>Switzerland</b>	<b>1.000.000</b>	<b>4.500 p</b>	<b>4.500 p</b>
	<b>Japan</b>	<b>2.000.000</b>	<b>4.000 p</b>	<b>6.000 p</b>

Figure 9.1: Comprehensive Cutwall Pricing

We were offered two alternative methods of payment should we wish to avail of the groups services. Firstly they could send a courier to our address (if we were in the Moscow area), who would collect payment and any materials we wished to advertise. The alternative was via direct bank transfer to a particular account. Regardless of the method used, as part of their service they would provide us with official receipts etc so that we prove to tax authorities that we had spent money on advertising.

In the case of our guise as an owner of a transportation group, the botnet owners also offered to design a simple website for our (fake) company, stating that this would increase the success rate of our spam email campaign. In both cases they also offered to craft the mail in such a way that it would avoid most anti-spam software while still being very effective at attracting the recipient.

## Pushdo / Cutwail

### A study of the Pushdo / Cutwail botnet

When we expressed our concern about the legal issues involved in sending millions of unsolicited emails we were assured that the practice is not illegal, at least not in Russia. We were told that it is illegal to stick advertising posters to walls without permission, but that spamming was not illegal as its “just email”. In truth the laws governing unsolicited mail are a lot weaker in Russia than in other countries.

All evidence points to Moscow being the origins of the Pushdo botnet. All of their contact numbers are for the Moscow area. As previously mentioned they offer to collect payment in person for individuals in the Moscow area. The IP addresses of their C&C servers seem to be spread across different countries, but most of their websites (including <http://Advert1.ru>) are registered to Moscow numbers.

As part of our research into the Advert1.ru domain we found a large list of other underground sites using the same authoritative name servers ( ns1.buildhost.ru ). While these sites do not all seem to be associated with the Pushdo gang, they are engaged in similar activities (selling email spam, “bullet-proof” hosting, porn etc). It is clear that the owner of this name server is yet another key player in the Russian Malware underground



Figure 9.2: Other Underground sites – Spam for sale



Figure 9.3: Other Underground sites – “Bullet Proof Hosting”

## Pushdo / Cutwail

A study of the Pushdo / Cutwail botnet

### PROPAGATION OF PUSHDO

Pushdo is unusual for a botnet in that it has no means of self-replication. Other botnets use worm-like behavior to spread from machine to machine across the network, or alternatively use their spamming engines to spread via links and attachments. Pushdo however uses neither of these techniques making it difficult to understand where the initial installation / propagation vector of the threat originates.

Without spam or worm-like behaviour as a means of propagation the only alternative for Pushdo is via the web, and anecdotal evidence appears that this is actually the case. Some studies<sup>13</sup> have found that the initial installers of Pushdo are being dropped by other well known malware families such as PE\_VIRUT, TROJ\_EXCHANGER and TROJ\_BREDOLAB.

What are equally interesting are the other binaries that are dropped along with Pushdo. These read as a who's who of modern malware with names such as Storm, Srizbi, Rustock, AntispywareXP2009 all in attendance.

It appears that just as Pushdo have agreements with several other malware groups to install those groups software, they also have agreements for separate malware groups to install Pushdo itself. This would be in keeping with Pushdo's practice of staying under the radar – by not having self-propagation it further adds to the confusion around the Pushdo / Cutwail threat.

<sup>13</sup> <http://blog.fireeye.com/research/2009/04/botnetweb.html#>



# Pushdo / Cutwail

A study of the Pushdo / Cutwail botnet

## REFERENCES

- [1] [http://threatinfo.trendmicro.com/vinfo/virusencyclo/default5.asp?VName=TROJ\\_PANDEX.A&VSect=Sn](http://threatinfo.trendmicro.com/vinfo/virusencyclo/default5.asp?VName=TROJ_PANDEX.A&VSect=Sn)
- [2][10] [http://www.message-labs.com/mlireport/MLIRreport\\_2009.01\\_Jan\\_Final.pdf](http://www.message-labs.com/mlireport/MLIRreport_2009.01_Jan_Final.pdf)
- [3] <http://royal.pingdom.com/2009/01/22/internet-2008-in-numbers/>
- [4][6] <http://www.virusbtn.com/virusbulletin/archive/2008/03/vb200803-pandex>
- [5] [http://en.wikipedia.org/w/index.php?title=Autonomous\\_system\\_\(Internet\)&oldid=281446510](http://en.wikipedia.org/w/index.php?title=Autonomous_system_(Internet)&oldid=281446510)
- [7] <http://support.microsoft.com/kb/115486>
- [8] [http://en.wikipedia.org/w/index.php?title=Root\\_nameserver&oldid=284118253](http://en.wikipedia.org/w/index.php?title=Root_nameserver&oldid=284118253)
- [9] <http://www.secureworks.com/research/threats/topbotnets/?threat=topbotnets>
- [11] [http://www.spamtrackers.eu/wiki/index.php?title=Canadian\\_Pharmacy&oldid=5842](http://www.spamtrackers.eu/wiki/index.php?title=Canadian_Pharmacy&oldid=5842)
- [12] [http://www.spamtrackers.eu/wiki/index.php?title=Prestige\\_Replicas&oldid=5171](http://www.spamtrackers.eu/wiki/index.php?title=Prestige_Replicas&oldid=5171)
- [13] <http://blog.fireeye.com/research/2009/04/botnetweb.html#>

