SAME BOTNET, SAME GUYS, NEW CODE

Pierre-Marc Bureau
ESET, spol. s r.o., Aupark Tower, 16th Floor, Einsteinova 24, 851 01 Bratislava, Slovak Republic

Email bureau@eset.sk

ABSTRACT

Win32/Kelihos first appeared at the very end of 2010. There are many things that make this malware stand out. First of all, it uses a custom peer-to-peer network protocol. It also shares many similarities in terms of code and endgame with Win32/Nuwar (the infamous Storm worm) and Win32/Waledac. These resemblances lead us to think the same gang is behind the creation, and possibly the operation, of all three pieces of malware.

The first variants of Win32/Kelihos to be discovered were, at best, in the alpha stage of development. Over recent months, we were able to closely track the new iterations, which allowed us to determine the primary purposes of this malware: spam campaigns and information stealing.

In this paper, we give an overview of the functionalities of Win32/Kelihos and expose its peer-to-peer network protocol. We also describe the evolution of the malware functionalities over time and conclude with a comparison between this malware and its two predecessors.

FUNCTIONALITY OVERVIEW

Win32/Kelihos is programmed in C++ and uses the object-oriented programming paradigm. Each executable is statically linked with many libraries, including OpenSSL and Boost. The average size of an unpacked executable is 1.3 megabytes. Considering the size of its code, it is hard to make an exhaustive list of the characteristics of this malware.

In this section, we give a general overview of the different characteristics of the Win32/Kelihos malware family. These characteristics help understand the objectives fulfilled by this malware family and put some context around its evolution.

Software protection

The first binaries of Win32/Kelihos that were discovered used the UPX packer to reduce the size of the binary executable. A few days later, the malware switched to a custom packer. We think the new software protection layer was outsourced to someone with deep knowledge of anti-virus engines and with the ability to program a packer straight in assembly language. This skill set seems distant from the one shown by the main developers of Win32/Kelihos.

The packer includes various anti-debugging and anti-emulation tricks. For example, Figure 1 shows a trick seen in many variants. The first instructions of the program try to load a file using the LoadLibraryExA API call. The file that gets loaded is not a standard DLL but rather an executable file. The file to load varies from one version of the packer to the other but is usually present on standard *Windows* installations. If the LoadLibraryExA call fails, the program simply exits without unpacking its code, potentially evading emulators used in anti-virus products.

Propagation

It appears that the initial infection vector used by Win32/Kelihos was through propagation waves of malicious links embedded in emails advertising fake greeting cards. When a user follows the malicious link, he is directed to a web page urging him to download a player for *Flash* files. Of course, the downloaded file is a fresh variant of the malware. Figure 2 shows an example of such a web page being displayed to the user. The malicious links are based on domains controlled by the botmaster and generally use fast flux techniques. The actual payload (HTML content and executable files) is served by infected hosts acting as proxies for the command and control server.

Can't view this greeting? Download Flash Player!

Figure 2: Malicious links propagated by email.

At the end of February, Win32/Kelihos started using a new propagation mechanism: the LNK parsing vulnerability that was previously exploited by Stuxnet (CVE-2010-2568). Later variants added the creation of malicious LNK files on removable drives in an effort to spread to other computers. More information on the effect of using this exploit usage can be found in the 'Code evolution' section of this paper.

Information stealing

Win32/Kelihos steals various types of information from infected hosts. It will analyse the hard drive of an infected machine, parsing a wide range of file formats looking for email addresses. We believe these addresses are later supplied to other bots to feed their spam campaigns. Another option would be that they are sold to other groups. More details on Win32/Kelihos spam operation can be found in the next section.

Win32/Kelihos also monitors network traffic for HTTP and FTP credentials. This information-stealing capability is implemented with the libpcap library. The stolen credentials are later used to upload content promoting fake pharmaceutical material.

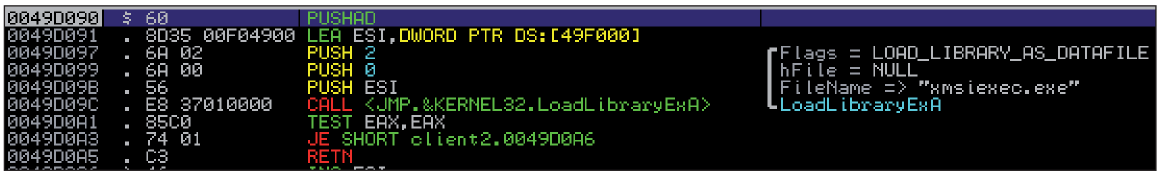
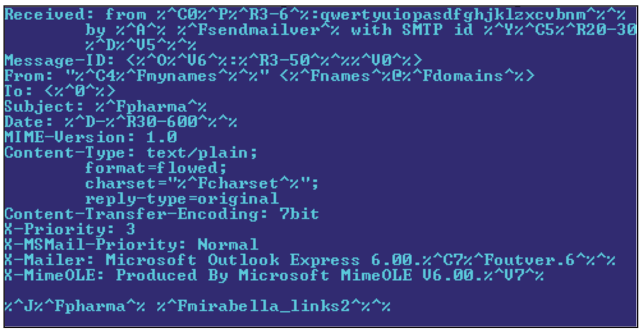

```
0049D090  $ 60              PUSHAD
0049D091  . 8D35 00F04900   LEA ESI,DWORD PTR DS:[49F000]
0049D097  . 6A 02           PUSH 2                                  ┌Flags = LOAD_LIBRARY_AS_DATAFILE
0049D099  . 6A 00           PUSH 0                                  │hFile = NULL
0049D09B  . 56              PUSH ESI                                │FileName => "xmsiexec.exe"
0049D09C  . E8 37010000     CALL <JMP.&KERNEL32.LoadLibraryExA>     └LoadLibraryExA
0049D0A1  . 85C0            TEST EAX,EAX
0049D0A3  . 74 01           JE SHORT client2.0049D0A6
0049D0A5  . C3              RETN
```

Figure 1: Anti-emulation trick used in the packer.

Spam

Win32/Kelihos uses a template-based spam engine. For each spam run, an infected host receives a template with a list of words to be used for each variable found in the template. Each variable's name starts with the '%^F' marker. Figure 3 shows a spam template to advertise pharmaceutical products.

The spam engine also receives a list of email addresses to spam. We believe these email addresses are gathered by the malware when it inspects the hard drives of infected systems.

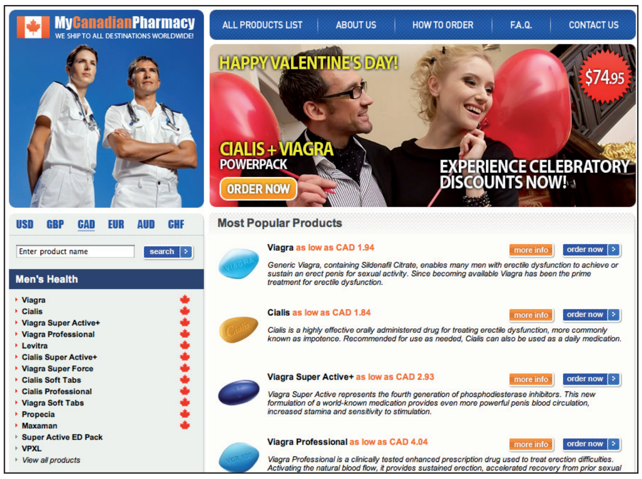


```
Received: from %^C0%^P%^R3-6^%:qwertyuiopasdfghjklzxcvbnm^%^%
          by %^A^% %^Fsendmailver^% with SMTP id %^Y%^C5%^R20-30
          %^D%^V5^%^%
Message-ID: <%^O%^V6^%:%^R3-50^%^%%^V0^%>
From: "%^C4%^Fmynames^%^%" <%^Fnames^%@%^Fdomains^%>
To: <%^0^%>
Subject: %^Fpharma^%
Date: %^D-%^R30-600^%^%
MIME-Version: 1.0
Content-Type: text/plain;
          format=flowed;
          charset="%^Fcharset^%";
          reply-type=original
Content-Transfer-Encoding: 7bit
X-Priority: 3
X-MSMail-Priority: Normal
X-Mailer: Microsoft Outlook Express 6.00.%^C7%^Foutver.6^%^%
X-MimeOLE: Produced By Microsoft MimeOLE V6.00.%^V7^%

%^J%^Fpharma^% %^Fmirabella_links2^%^%
```

Figure 3: Spam template.

In most cases, the spam is promoting fake pharmaceutical material with a compelling message and a link to an HTML page. These HTML pages are usually hosted on compromised websites that merely act as a redirection to the real sites selling the products. Figure 4 shows an example of a website promoted by Win32/Kelihos.



Viagra as low as CAD 1.94

Generic Viagra, containing Sildenafil Citrate, enables many men with erectile dysfunction to achieve or sustain an erect penis for sexual activity. Since becoming available Viagra has been the prime treatment for erectile dysfunction.

Cialis as low as CAD 1.84

Cialis is a highly effective orally administered drug for treating erectile dysfunction, more commonly known as impotence. Recommended for use as needed, Cialis can also be used as a daily medication.

Viagra Super Active+ as low as CAD 2.93

Viagra Super Active represents the fourth generation of phosphodiesterase inhibitors. This new formulation of a world-known medication provides even more powerful penis blood circulation, increased stamina and sensitivity to stimulation.

Viagra Professional as low as CAD 4.04

Viagra Professional is a clinically tested enhanced prescription drug used to treat erection difficulties. Activating the natural blood flow, it provides sustained erection, accelerated recovery from prior sexual

Figure 4: Pharmaceutical website advertised through spam.

Besides pharmaceutical spam, we observed that the botnet has been used to send spam advertising some stocks, probably for a pump-and-dump scheme. We also observed that an infected host will send a feedback report on the spam run to its command and control server, reporting which email addresses were reachable.

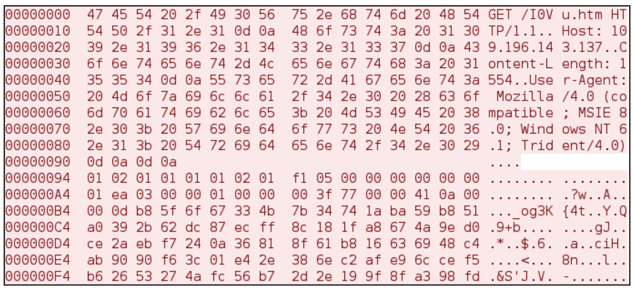
Peer-to-peer network protocol

Win32/Kelihos's peer-to-peer network protocol uses TCP port 80 and is only used by a subset of hosts that have a public IP address, also called proxies. The peer-to-peer protocol is used to exchange the following information:

- Information on 'Job Servers' which deliver tasks to infected systems
- Information on other peers participating on the network
- Information on domain names used by the botnet.

Hosts with a private IP address (workers) will only communicate with infected hosts having a public IP address (proxies).

The peer-to-peer network protocol imitates the HTTP protocol. Peers communicate with each other by sending an HTTP 'GET' request, requesting an HTML file with a random name. Data that is sent to the peer is appended to the GET request, which is not HTTP-compliant. Figure 5 shows a network capture of the peer-to-peer communication between two peers.



```
00000000  47 45 54 20 2f 49 30 56  75 2e 68 74 6d 20 48 54 GET /I0V u.htm HT
00000010  54 50 2f 31 2e 31 0d 0a  48 6f 73 74 3a 20 31 30 TP/1.1.. Host: 10
00000020  39 2e 31 39 36 2e 31 34  33 2e 31 33 37 0d 0a 43 9.196.14 3.137..C
00000030  6f 6e 74 65 6e 74 2d 4c  65 6e 67 74 68 3a 20 31 ontent-L ength: 1
00000040  35 35 34 0d 0a 55 73 65  72 2d 41 67 65 6e 74 3a 554..Use r-Agent:
00000050  20 4d 6f 7a 69 6c 6c 61  2f 34 2e 30 20 28 63 6f  Mozilla /4.0 (co
00000060  6d 70 61 74 69 62 6c 65  3b 20 4d 53 49 45 20 38 mpatible ; MSIE 8
00000070  2e 30 3b 20 57 69 6e 64  6f 77 73 20 4e 54 20 36 .0; Wind ows NT 6
00000080  2e 31 3b 20 54 72 69 64  65 6e 74 2f 34 2e 30 29 .1; Trid ent/4.0)
00000090  0d 0a 0d 0a                                      ....
00000094  01 02 01 01 01 01 02 01  f1 05 00 00 00 00 00 00 ........ ........
000000A4  01 ea 03 00 00 01 00 00  00 3f 77 00 00 41 0a 00 ........ .?w..A..
000000B4  00 0d b8 5f 6f 67 33 4b  7b 34 74 1a ba 59 b8 51 ..._og3K {4t..Y.Q
000000C4  a0 39 2b 62 dc 87 ec ff  8c 18 1f a8 67 4a 9e d0 .9+b.... ....gJ..
000000D4  ce 2a eb f7 24 0a 36 81  8f 61 b8 16 63 69 48 c4 .*..$.6. .a..ciH.
000000E4  ab 90 90 f6 3c 01 e4 2e  38 6e c2 af e9 6c ce f5 ....<... 8n...l..
000000F4  b6 26 53 27 4a fc 56 b7  2d 2e 19 9f 8f a3 98 fd .&S'J.V. -.......
```

Figure 5: Peer-to-peer packet capture.

The data exchanged in the peer-to-peer protocol is encrypted. The encryption algorithms used are DES and Blowfish. The keys used are hard coded in the malware. After decryption, the data is deserialized into objects. The serialization algorithm packs data into a list of items that can be nested. The type and size of each item is also serialized. The serialization code used in Win32/Kelihos is publicly documented on the following webpage: http://www.rsdn.ru/article/files/Classes/Serialization2.xml.

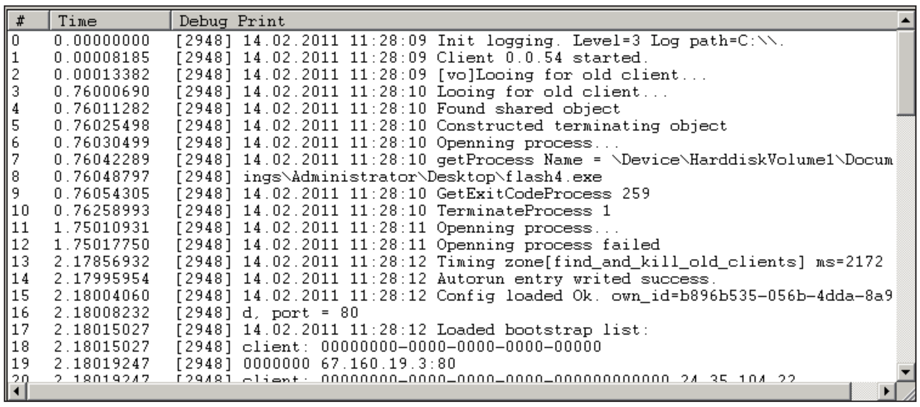
To keep up-to-date information on the peer-to-peer network after a reboot, the list of peers known to an infected system is stored in the HCKU\Software\Google\ID3 registry key. Peers frequently exchange peer-related information in order to keep good connectivity.

Alongside the workers and proxies, a third part of the botnet architecture plays a major role: the job servers. These servers are used to deliver tasks to infected hosts (mostly spam jobs) and collect information back from them (list of reachable email addresses, stolen information, etc.). Each time a worker wants to contact a job server, it will first send an HTTP request to one of the proxies in its peer list. The proxy will simply forward this request to a job server and send back the server's response. Proxies will also use other proxies to contact a job server, instead of contacting them directly.

CODE EVOLUTION

The first variants of Win32/Kelihos released on the Internet were in an early stage of development. The code included hundreds of debug messages that were printed to the debug output. Figure 6 shows the log of Win32/Kelihos infecting a new host. We can see the version of the bot as well as compatibility checks for earlier versions of the code that could have already been on the system.

The infection ratio of Win32/Kelihos has been very limited compared to large infections like Win32/Conficker and other big malware families. On the other hand, we have been able to see the impact of code modifications on the detection ratio



```
#     Time          Debug Print
0     0.00000000    [2948] 14.02.2011 11:28:09 Init logging. Level=3 Log path=C:\\.
1     0.00008185    [2948] 14.02.2011 11:28:09 Client 0.0.54 started.
2     0.00013382    [2948] 14.02.2011 11:28:09 [vo]Looing for old client...
3     0.76000690    [2948] 14.02.2011 11:28:10 Looing for old client...
4     0.76011282    [2948] 14.02.2011 11:28:10 Found shared object
5     0.76025498    [2948] 14.02.2011 11:28:10 Constructed terminating object
6     0.76030499    [2948] 14.02.2011 11:28:10 Openning process...
7     0.76042289    [2948] 14.02.2011 11:28:10 getProcess Name = \Device\HarddiskVolume1\Docum
8     0.76048797    [2948] ings\Administrator\Desktop\flash4.exe
9     0.76054305    [2948] 14.02.2011 11:28:10 GetExitCodeProcess 259
10    0.76258993    [2948] 14.02.2011 11:28:10 TerminateProcess 1
11    1.75010931    [2948] 14.02.2011 11:28:11 Openning process...
12    1.75017750    [2948] 14.02.2011 11:28:11 Openning process failed
13    2.17856932    [2948] 14.02.2011 11:28:12 Timing zone[find_and_kill_old_clients] ms=2172
14    2.17995954    [2948] 14.02.2011 11:28:12 Autorun entry writed success.
15    2.18004060    [2948] 14.02.2011 11:28:12 Config loaded Ok. own_id=b896b535-056b-4dda-8a9
16    2.18008232    [2948] d, port = 80
17    2.18015027    [2948] 14.02.2011 11:28:12 Loaded bootstrap list:
18    2.18015027    [2948] client: 00000000-0000-0000-0000-00000
19    2.18019247    [2948] 0000000 67.160.19.3:80
20    2.18019247    [2948] client: 00000000-0000-0000-0000-000000000000 24.35.104.22
```

Figure 6: Debug output of an early version of Win32/Kelihos.

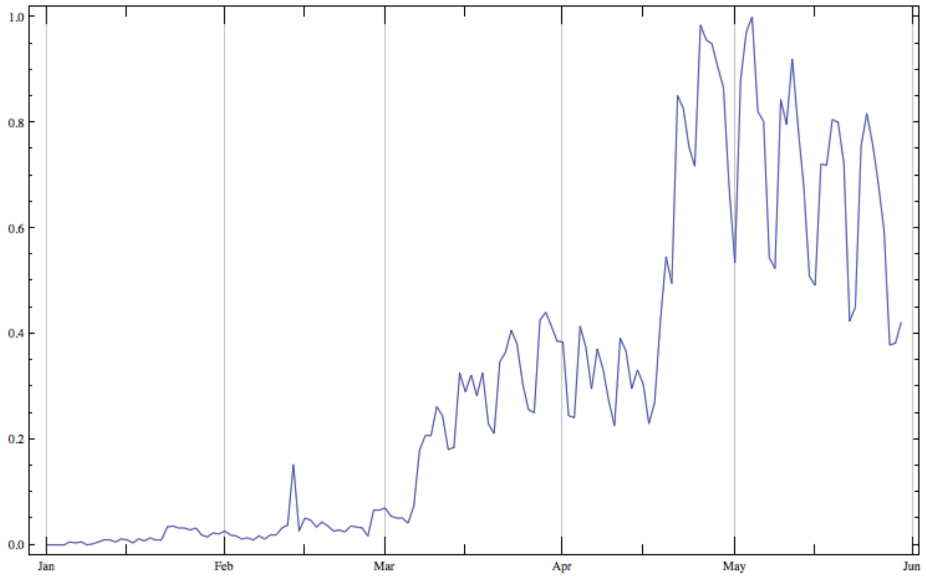


Figure 7: Detection ratio for Win32/Kelihos over time (2011).

for this malware family. Figure 7 shows the evolution of the detection statistics collected from *ESET*'s *ThreatSense* system from 1 January 2011 until 31 May 2011. This figure shows that the malware propagation increased significantly after the inclusion of the CVE-2010-2568 (LNK) exploit into the malware at the end of February.

The encryption of the peer-to-peer protocol used by Win32/Kelihos also evolved over time. The first versions of the protocol did not use encryption. Encryption was gradually added, starting with one layer of Blowfish and two additional layers of DES later on. To make the data harder to decipher, the labels for the variable names have also been hashed. With this new hashing, it becomes much harder to follow which variable does what. A work-around for this obfuscation is to reuse labels found in older versions of the code and hash them to create a dictionary of words corresponding to hashes. It is then possible to replace the hash values with their original text form when analysing the 'obfuscated' data.

WIN32/NUWAR, WIN32/WALEDAC, WIN32/KELIHOS; SAME MALWARE?

In this section, we compare the functionalities and peer-to-peer protocols used by the Win32/Nuwar, Win32/Waledac and Win32/Kelihos families to understand the commonalities between the three.

Functionality comparison

There are many similarities between Win32/Kelihos and its two predecessors. Table 1 shows a summary of similarities found in the binaries.

	Win32/Nuwar	Win32/Waledac	Win32/Kelihos
Propagation	Spam & links	Spam & links	Spam & links
Information stealing		FTP / SMTP	FTP / SMTP
Fast flux	X	X	X
Kernel mode component	X		
Persistence	Registry key	Registry key	Registry key
Spamming	Pump and dump Pharmaceutical	Pharmaceutical	Stock Pharmaceutical

Table 1: Summary of similarities found in the binaries.

We see that the objective of all three malware families is to send unsolicited emails; this is clearly how the operators are funding their operation and profiting from it. Furthermore, the variable names and spam templates are the same for all three families – this suggests that the same gang is behind all three operations.

To our knowledge, there are not many botnets using fast-flux nowadays; it is interesting to note that all three families make use of this technique. Finally, we note some strong similarities in the material that is spammed by the botnet.

Peer-to-peer comparison

The peer-to-peer network protocol used by Win32/Kelihos is one of its most interesting characteristics. Table 2 shows a comparison of the peer-to-peer protocol used by the three malware families.

	Win32/Nuwar	**Win32/Waledac**	**Win32/Kelihos**
Peer-to-peer protocol	Kademlia	XML over HTTP	Serialized over HTTP
Peer-to-peer network traffic	UDP, variable port	TCP 80	TCP 80
Peer-to-peer protection	XOR	Zlib + AES + base64	Zlib + DES + blowfish
Peer-to-peer 'keys'	Static	Rotating	Static
Multi-tiered architecture	X	X	X

Table 2: Comparison of the peer-to-peer protocol used by the three malware families.

From a theoretical point of view, the reliability against attacks and data distribution efficiency make Kademlia the best protocol of all three. On the other hand, this protocol uses UDP and can easily be blocked in an enterprise environment. From a stealth perspective, exchanging data over TCP port 80 is better and we think this is why the programmers have moved to this new protocol.

We observe that the programmers have also improved their usage of cryptography to reduce the possibility of data poisoning or even interception. Moving from XOR to AES is clearly an evolution. In terms of key exchange, we observed that it was too computationally expensive to change cryptographic keys in Win32/Waledac [3]. We think this is why Win32/Kelihos is also using hard-coded keys. This is a clear trade-off between performance and security.

Finally, it is worth mentioning that all three botnets use a multi-tiered architecture where multiple command and control proxies are used to share the load and increase reliability of the botnet.

CONCLUSIONS

The analysis of the Win32/Kelihos malware family, and its similarities to Win32/Nuwar and Win32/Waledac teach us many things. First of all, the evolution of malware does not follow a strict rule of performance. Malware developers must find the best equilibrium between costs, level of stealth, security and performance. This is shown by the fact that the programmers first tried using the Kademlia peer-to-peer protocol and then moved to a less efficient but much stealthier custom protocol using TCP port 80.

Our study also shows that malware authors won't hesitate to use third-party software components in their programs. In this case, we observed OpenSSL, zlib and libpcap being used by the malware. It is also likely that the programmers rented the service of another group for their packer. This shows that malware creators and operators are now specializing their services into 'niches' where they are the experts and outsourcing other areas of development to other groups.

To conclude, Win32/Kelihos should be considered as a separate malware family from Win32/Nuwar and Win32/Waledac because it doesn't have the same code, and the binaries are entirely different. From a higher-level perspective though, we should classify all three families as being part of the same operation where similar strategies and tactics are being used to infect systems, maintain the botnet, and bank juicy profits.

ACKNOWLEDGEMENTS

We would like to thank the following people for their help during the research and in the redaction of this paper: Joan Calvet, Alexis Dorais-Joncas, Sébastien Duquette, Dávid Gábriš and Witold Gerstendorf.

REFERENCES

[1] Stewart, J. Inside the Storm: Protocols and Encryption of the Storm Botnet. Black Hat 2008. http://www.blackhat.com/presentations/bh-usa-08/Stewart/BH_US_08_Stewart_Protocols_of_the_Storm.pdf.

[2] Sinclair, G.; Nunnery, C.; ByungHoon Kang, B. The Waledac Protocol: The How and Why. Malware 2009.

[3] Calvet, J.; Bureau, P-M.; Fernandez, J.; Marion, J-Y. Large-scale malware experiments, why, how, and so what? Proceedings of the Virus Bulletin International Conference 2010.

[4] Maymounkov, P.; Mazières, D. Kademlia: A Peer-to-Peer Information System Based on the XOR metric. Peer-to-Peer Systems, 2002.