SAM STOVER, DAVE DITTRICH,
JOHN HERNANDEZ, AND SVEN DIETRICH

analysis of the Storm and Nugache trojans: P2P is here

Sam Stover is the Director of Tech Ops for iSIGHT Partners, a startup which produces human and electronic intelligence-fused products and services. Research interests include detection and mitigation methods for vulnerabilities and malware.

sam.stover@gmail.com

Dave Dittrich is an affiliate information security researcher in the University of Washington's Information School. He focuses on advanced malware threats and the ethical and legal framework for responding to computer network attacks.

dittrich@u.washington.edu

John Hernandez, a recent graduate of the University of Washington, is currently working for Casaba Security LLC as a security consultant. Research interests include reverse engineering and malware forensics.

lafkuku@msn.com

Sven Dietrich is an assistant professor in the Computer Science Department at Stevens Institute of Technology in Hoboken, NJ. His research interests are computer security and cryptography.

spock@cs.stevens.edu

SINCE THE ADVENT OF DISTRIBUTED intruder tools in the late 1990s, defenders have striven to identify and take down as much of the attack network as possible, as fast as possible. This has never been an easy task, owing in large part to the wide distribution of attacking agents and command and control (C2) servers, often spread across thousands of individual networks, or Autonomous Systems in routing terms, around the globe. Differentials in the abilities and capabilities of these sites, as well as knowledge of what role the site plays in distributed attack networks (potentially many active at one time), make mitigation harder, as do differences in legal regimes, etc. [1]. Still, there has grown a huge population of researchers, security vendors, and organizations focused on identifying and mitigating distributed attack networks.

In another article in this issue ("Command and Control Structures in Malware from Handler/Agent to P2P"), the authors discuss how topologies for C2 have changed over time, for various reasons. The most popular method of C2 in recent years has been the use of Internet Relay Chat (IRC), either in standard form or through use of customized implementations of IRC servers and clients intended to thwart mitigation efforts. Programs that use IRC for C2 are known as "bots" (short for "robot"), and a distributed network of bots is known as a "botnet." The terms "bot" and "botnet" are now becoming so widely used that they are losing much of their original meaning. Botnets will likely be around for some time, causing a huge amount of grief for network operators, victims of DDoS attacks, and other victims, but IRC-based bots are not the be-all and end-all, and the advent of Peer-to-Peer (P2P) mechanisms for C2 may spell the eventual death of IRC as a means of C2. At that point, when IRC is secondary (at best) and possibly not involved at all, will the terms "bot" and "botnet" still have their original meaning, or will they become general terms that are synonymous with "trojan," "worm," etc.?

There are some obvious advantages to decentralizing the C2 mechanism, and P2P networking fits the bill. The Overnet protocol is particularly attractive because of the transient, "self-healing" nature of the network, along with "servers" that actually share files. Each peer is constantly advertising its

presence, as well as requesting updates from other peers. Throughout this article, there are references to eDonkey and Overnet portions and fields in the packets. This is because the Overnet protocol was based on eDonkey—the primary difference is that pure eDonkey clients are all equal (i.e., there are no "servers"). Overnet expanded on eDonkey to allow peers to communicate information about the P2P network via eDonkey methods, but it added the capability to introduce servers that could host files, if necessary [2]. As we'll see, Storm requires the server file-sharing technique offered by Overnet networks, so technically Storm does not use just the eDonkey protocol, but the Overnet protocol. To reduce confusion, when generically discussing the P2P capabilities, the term "Overnet" will be used. However, when a legacy eDonkey protocol characteristic is being discussed specifically, for example an eDonkey Connect request, the specific name will be used.

Our goal was to compare two different trojans, with a particular emphasis on how they rely on P2P mechanisms. We infected a number of fully patched Windows XP SP2 test systems with samples of Storm and Nugache trojans and analyzed the resulting network traffic captures. Because a large volume of research exists on the impact to the host, we decided not to focus on this (although we did monitor some environmental changes to track the progress of the infecting malware), but more on the communication methods. We hoped to be able to define some mitigation methods at both network and host levels, which will be presented at the end of the article.

To adequately describe the similarities and differences between the P2P capabilities of Storm and Nugache, a brief walkthrough of both is required. The Storm trojan is primarily designed to send spam, but because of its modular nature, it can easily acquire other capabilities such as the DDoS module that was used to great effect early in 2007 [3]. Once installed, the trojan joins and participates in an extensive network that utilizes the Overnet protocol to distribute information and eventually supply the infected peer with the tools it needs. As we will see, the Overnet P2P mechanisms it uses to propagate information provide a very effective means to this end.

Several different Storm binaries were collected and used to infect the test systems, but behavior was consistent throughout the exercise. A complication referred to by one of the authors as the "P2P chicken and egg problem" exists when a new P2P peer is created: It has no knowledge of the current state of the network. A certain amount of information must be delivered with, or in, the binary, which directs the trojan to other infected peers. In truth, only one peer is needed, but the self-healing nature of P2P networks requires a very transient state. If all newly infected systems were given one "superpeer" to connect to, it would only be a matter of time before that IP address was discovered and addressed via firewall rules, IDS signatures, legal action, etc. Storm takes the opposite tack, in that it seeds each new peer with approximately 300 static peers in a text file called spooldr.ini, although this name may change (e.g., it was wincom.ini previously). This file contained approximately 300 rows, with two fields per row. Figure 1 shows the first few lines of the spooldr.ini file delivered with our trojan:

```
[config]
[local]
uport=11873
[peers]
00000000000000009C2DB8A6F34A9C69=452FC581466700
00010CED75C2E4C6222534E6BD5BB4A1=D5868ADE16C900
0001351DE60D58519C2DB8A6F34A9C69=452FC581466700
00037A3051FE23B6BE8B8C79BE6DD56A=41FF4E35835600
```

FIGURE 1: BEGINNING OF CONFIGURATION FILE FOR STORM

The first field in each line contains the peer hash, which uniquely identifies the peer node. The second field contains the IP, port, and peer type, in hex, for the original set of peers with which the trojan will start. These hosts are the only ones currently known to the trojan, and they will be contacted in the hope of receiving up-to-date information about the network. The first packet that our infected system sent out was an eDonkey Publicize packet, shown in Figure 2, which is designed to alert existing peers that a new system is available to the network:

```
06:14:22.949925 IP 192.168.168.152.2506 > 81.248.26.210.20136: UDP,
length 25
    0x0000:  00c0 4f1e 2844 5254 0012 3456 0800 4500  ..O.(DRT..4V..E.
    0x0010:  0035 014c 0000 8011 6361 c0a8 a898 51f8  .5.L....ca....Q.
    0x0020:  1ad2 09ca 4ea8 0021 77bd e30c 89ae f92f  ....N..!w....../
    0x0030:  20bb bac5 dce0 e6ee 6d51 1cda 0000 0000  ........mQ......
    0x0040:  ca09 00
```

FIGURE 2: EDONKEY PUBLICIZE PACKET

Converting the destination IP address (81.248.26.210) into hex yields 0x51F81AD2. Searching for that string in the spooldr.ini file gives:

```
F3032DA7F7C1E94A4FE9D59838C67D40=51F81AD24EA800
                                 ^^^^^^^
```

Logic suggests that the destination port follows the IP, and sure enough 0x4EA8 is 20136, the port that our packet went to. The final two digits are the Overnet Peer Type designation, which we will see later.

Another important aspect of the "chicken and egg" problem is that the new peer is unsure of its external IP address. A breakdown of the Publicize packet will demonstrate this, as well as lay the groundwork for all of the eDonkey fields. The eDonkey portion of the packet starts with 0xe3 (byte offset 0x002A), which designates the eDonkey protocol, followed by the type of eDonkey packet, which in this case is 0x0c for Publicize. The remaining portion of the Publicize data represents characteristics of the Overnet Peer:

Hash Identifier:	0x89ae f92f 20bb bac5 dce0 e6ee 6d51 1cda
IP Address:	0x0000 0000 (0.0.0.0)
Port:	0xca09 (2506)
Peer Type:	0x00 (0)

Since the trojan does not know its external IP address, the value for the IP address field is 0.0.0.0. Our system must rely on a replying peer to provide that information. When a live peer receives a Publicize packet, it responds with a Publicize ACK (0xe30d), a very simple packet with a 2-byte payload that informs the publicizing peer that it is willing to communicate. Upon receiving a Publicize ACK, the new system now has someone to talk to and quickly sends out two very important packets; an eDonkey Connect (0xe30a) and an IP Query (0xe31b). The purpose of the Connect request is to gain current information about the P2P network, while the goal of the IP Query is to determine its own routable IP address. Upon receiving a Connect Reply (0xe30b) and an IP Query Answer (0xe31c), the new peer has conquered the chicken and egg problem: It is now aware of new peers, and it knows its routable IP address.

Until now, the sole purpose of the malware was to find its place in the network by learning about other peers and advertising itself to them. Now that it has integrated into the network it is time to get to work, and for this trojan, that means spam. This is not to say that the malware stops advertising and learning—that process continues throughout the life of the infection. But once a certain steady state in the network is reached, the traffic shifts

from pure Publicize and Connect requests to searches for specific kinds of data, followed by the initiation of TCP sessions. Until this point, all traffic has been UDP, which is consistent with the eDonkey protocol description: "In the Edonkey [*sic*] network the clients are the nodes sharing data. Their files are indexed by the servers. If they want to have a piece of data (a file), they have to connect using TCP to a server or send a short search request via UDP to one or more servers to get the necessary information about other clients sharing that file"[4]. In the case of the Storm network, TCP connections are made to servers that hold the keys to the spam kingdom. There are email lists, mailserver names, and email templates to be had for the asking. Once a peer knows where to look, it initiates a TCP session to that server, or in our case, group of servers, looking for the goods. In one test session, our system repeatedly attempted to establish TCP sessions with ten IP addresses, three of which were successful: one registered in Kiev, one in Romania, and one in Illinois. Of the three sessions, one was very short, only 9 bytes, which implies that the server, although active, did not have the information being searched for. The other two sessions, however, were exactly 52,806 bytes each, and after these sessions were completed, the trojan began to spam.

First, before actually sending any spam, MX record queries were made. Once the response was received from the DNS, numerous short conversations were held with the mailservers (see Fig. 3).

```
06:17:59.971764 IP 209.191.89.172.25 > 192.168.168.152.1092: P 1:136(135) ack 1 win 65535
E....'@./..u..Y.......DS..at..aP.......421 Message from (76.2.252.62) temporarily
deferred
06:18:00.018835 IP 65.54.244.72.25 > 192.168.168.152.1096: P 1:311(310) ack 1 win 65535
E..^..@.o...A6.H.......HcK......P.......220 bay0-mc5-f1.bay0.hotmail.com Sending
Unsolicited commercial or bulk e-mail to Microsoft's computer network is prohibited.
06:18:00.091036 IP 67.91.84.250.25 > 192.168.168.152.1097: P 1:120(119) ack 1 win 65535
E....F@.m..|C[T........IK.8....uP.......220 mail1.nuparts.com Microsoft ESMTP MAIL
Service, Version: 6.0.3790.3959 ready at Sun, 23 Sep 2007 03:43:24 -0700
```

FIGURE 3: SNIFFED SPAM COMMUNICATIONS WITH MAILSERVERS

After a number of these mailserver tests, the infected system begins to send spam using standard SMTP.

Nugache

As far as Nugache is concerned, our evaluation is based on examination of several binaries, spread over many months. The functionality of this trojan evolved over time, with its command set increasing in conjunction with its attack and spreading capabilities. Although one of its main purposes is DDoS, it is also capable of acting as a password-protected proxy, propagating itself, downloading and running arbitrary programs, and also collecting and returning keystroke information, possibly compromising user-entered data. In the early stages of development, Nugache, also known misleadingly as the "tcp/8 bot," used fixed ports and IRC for command and control. Later on, it dropped below the radar, as its command and control communications moved to random high-numbered ports and evaded detection in most cases. Keeping track of only a few neighbors, a single trojan would not be aware of the entire network.

Nugache is simpler in architecture than Storm, and in some ways it is simpler in its use of P2P C2 communications. This simplicity comes partly from handling the seeding of peers in a different way from Storm (either by preseeding the compromised host's Windows Registry with a list of peers prior to first running the malware or through bootstrapping the list from a small set of hard-coded default hosts within the binary itself). The default list is in binary form (not ASCII "IP:PORT" strings) and packed into the binary to further conceal them.

Once the list of peers is produced, Nugache peers can join the network using an RSA key exchange to share seed material used to generate and return a Rijndael 256-bit OFB (Output Feedback) mode key. This exchange is shown in Figure 4. Once encryption is set up, an internal protocol is used to perform other tasks. One of these tasks is to negotiate a "connect-back" process to determine the initiating peer's routable IP address and listening port number, at the same time determining whether this peer can act as a "servent" or is only capable of being a client. As with Storm, knowing whether a peer is capable of being a servent is important to determine.

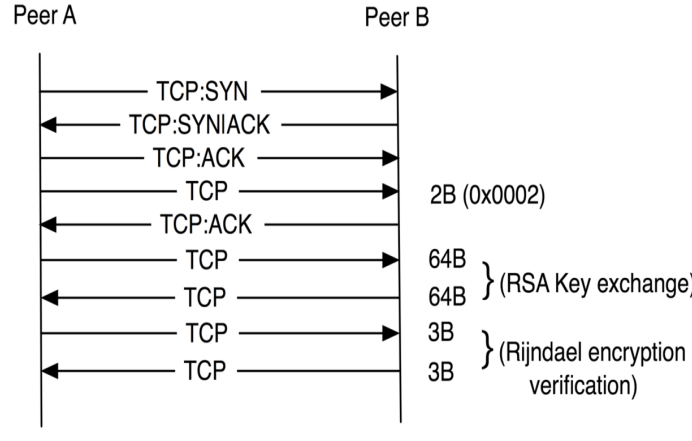

FIGURE 4: SAMPLE KEY EXCHANGE AND ENCRYPTION SETUP

Other tasks involve checking to see if the peer should have its binary instance updated, or whether an updated peer list is needed. Nugache suffers from the same problem as Storm in regard to needing an up-to-date list of peers to use for rejoining the P2P network. After that, a "PING" and "PONG" keep-alive exchange is engaged, and peers inform their connected siblings of any newly discovered servant peers that join the network. This latter information is used to "refill" the seed list in the Windows Registry in the event that excessively nonresponsive peers have been dropped from the list.

Now that we've briefly discussed Nugache and Storm, we can identify the similarities and differences between the two trojans, specifically with respect to how they utilize P2P to function. We've broken the types of comparisons into nine different categories: primary command and control mechanisms, initial peer list seeding, use of cryptography to secure the C2 channel, use of DNS, connectivity, updates, listening port for P2P connections, architecture, and detection.

PRIMARY COMMAND AND CONTROL MECHANISM

Nugache almost entirely handles C2 via the encrypted P2P channel established between connected client and servant peers. There is an IRC capabili-

ty built in, and Nugache peers could respond to (or direct) classic IRC bots in standard IRC channels, but IRC sessions have very rarely been seen in practice.

Storm's C2 is handled via TCP initiated to servers that were not discovered via pure P2P client integration communications. That is, when the peer joins the network, there is a lot of communication with other Overnet peers, but these peers are utilized purely to (a) determine the state of the network and (b) join it. No file exchanges take place via P2P communications; however, IP addresses of "servers" hosting email templates, email lists, and mailservers are disseminated using P2P communications. As such, Storm is more of a "pull" C2 technology—servers do not "push" commands down to the clients; rather, the clients "pull" data from servers.

INITIAL PEER LIST SEEDING

For a Nugache peer to join the network, it must first know of at least one active servant peer that will accept an incoming connection. Once connected, peers will be told of other servant peers and will maintain a list of up to 100 such peers for future use in rejoining the network. How that list of 100 peers is first loaded is controlled by a "bootstrap" process. How it is kept up-to-date is a function of the Nugache P2P algorithm.

The initial seeding is done one of two ways. Either a helper program is run to fill the Windows Registry, which then starts the Nugache program, or the Nugache program uses an internal list to attempt to find an active servant peer, which then provides a current list of up to 100 recently seen servant peers.

Once connected to one or more peers, those peers will report new connecting servant peers. This allows connected peers to learn of recently active servant peers, which have a high probability of being available for reconnecting after a system reboot or shutdown.

Storm seeds its initial peer list via a text file ("spooldr.ini" in this investigation). The infected systems sends eDonkey Publicize packets to all of the peers located in the spooldr.ini file. Once a Publicize ACK is received from a live peer, Connect requests that ask for up-to-date information on peers are delivered. Publicize and Connect packets are consistently sent to any and all peers throughout the life of the infected system.

USE OF CRYPTOGRAPHY TO SECURE THE C2 CHANNEL

Nugache uses a variable bit length RSA key exchange, which is used to seed symmetric Rijndael-256 session keys for each peer connection. Rijndael is also used to encrypt keystroke log files prior to transfer, using a key that is derived from information unique to the peer sending the keylog data. IRC sessions (when used, which is rarely) are not encrypted in any known version of Nugache.

Storm uses a hash mechanism for encrypting data requests to peers and servers. In previous examinations of this trojan, the encrypted string was stored in the meta-tag field of an eDonkey Search Result packet. In the version we investigated, the meta-tag field was either empty or contained a cleartext string (e.g., "20765.mpg;size=78092;"). This indicates that the obfuscation and encryption method is evolving.

USE OF DNS

Aberer and Hauswirth [5] cite the description of P2P as given by Clay Shirky (The Accelerator Group), which states that, "P2P nodes must operate outside the DNS system, and have significant or total autonomy from central servers" [6]. In this respect, Nugache is a true P2P malware artifact. DNS is almost entirely unused, save for immediately prior to DDoS events, joining IRC channels on rare occasions, or for other activity that peers are tasked with via commands, and there is no central C2 server (i.e., peers operate fully autonomously). As there is no use of DNS for seeding peer lists, for identifying C2 channels, or for joining the network, any DNS-based detection or mitigation mechanism will be entirely blind and useless in dealing with Nugache.

Similarly, Storm relies on DNS purely for MX record requests. Mailserver names are passed to the trojan (e.g., "gmail.com"), it performs an MX record query, and upon receiving the answer, it proceeds to connect to port 25 and send spam. Although no DDoS activity was observed, it seems logical to assume that a similar process would be followed: A domain name would be passed to the trojan, DNS queries would be made, and then DDoS activity would commence. At no time was any DNS activity observed that was related to the P2P or to the TCP/C2 communications.

CONNECTIVITY

Nugache peers maintain an in-degree of connections that totals no more than ten clients at any time. The out-degree varies, but it is typically less than half of the ten-client limit. The result is a typical peer with at most about 13–15 connections active at any given time.

Storm seems to collect as many active peers as possible. Because connections to these peers utilize UDP, the malware is chatty, but the traffic is lightweight. Constant Publicize and Connect packets are delivered while the system is active. There were approximately 300 peers in the original spooldr.ini file, but the file would grow as new peers were discovered. In one case, after less than 30 minutes, the spooldr.ini file contained over 430 entries. Theoretically, this number could, if monitored, be used to infer the size of the P2P network.

UPDATES

Nugache uses an internal release number to indicate the current version of the currently running code. When peers connect to the P2P network, they compare version numbers, and a peer with a lower version number will request an update from the peer it just connected with. This allows the entire network to continually upgrade itself as peers come back online after an absence. (Nugache stopped using the fixed port 8/tcp well over a year ago, yet a handful of connection attempts to port 8/tcp can still be observed occurring today.)

Storm updates itself in two main ways. Peer updates and information queries utilize UDP P2P communications, whereas TCP sessions are used to download important data such as new functionality (i.e., the DDoS module), Mailservers, and email templates.

LISTENING PORT NUMBER FOR P2P CONNECTIONS

Although Nugache is known to some analyses as the "tcp port 8 bot," it has not used this fixed port since June 2006. Each peer chooses its own randomly generated high-numbered port to listen on, ranging from 1025 to 65535. On connecting to the Nugache P2P network, the connecting peer will report what port it listens on, and the connected (servant) peer will check to see if a connection can be made back to the connecting peer. If a connection can be made, the routable address of the connecting peer is sent to that peer and the IP:PORT combination is reported to other active peers for future reference. (See the section on "Initial Peer List Seeding.")

Storm picks a random high port for communications and advertises that port in every packet that it sends out. Publicize and Publicize ACK packets are used to establish initial communications as well as to verify the proper IP address and port number. In the case of a peer receiving a Publicize packet that contains the 0.0.0.0 IP, that peer will transmit an Identify Packet requesting the proper IP.

ARCHITECTURE

Nugache is a monolithic binary executable, written in Visual C++ and packed with a simple home-grown packer. Other helper programs have been observed (e.g., used for seeding the initial peer list and installing the Nugache binary on infected systems), but these programs are secondary and not required to infect a host with Nugache. State is maintained primarily through the use of the Windows Registry; however, a keystroke log is kept until retrieved via commands from the attacker controlling the Nugache P2P network.

Storm is packed åusing a more complex method. The first stage of unpacking is decrypted using the XOR function; then a TEA decryption algorithm is applied; finally, the binary is reconstructed using the TIBS unpacker. After the binary is extracted it drops a copy of the original binary plus the spooldr.ini file into the Windows directory and also extracts a rootkit driver called spooldr.sys to the system32 directory. Then it loads this driver into the kernel via an unexpired call from tcpip.sys called SfcFileException, allowing the binary to hide from the OS [7]. After the binary is set, it attempts to communicate to the peers located in the spooldr.ini file and establish itself in the P2P network. Storm is a multicomponent modular set of programs. Each component has a different purpose, and the programs are installed in a set after the initial infection has occurred and the Storm program has successfully found a C2 server.

DETECTION

There is no static IDS signature that will detect Nugache P2P flows. The RSA key exchange is dynamic enough that one cannot get a 100% successful hit rate on detecting the key exchange. (The "Bleeding-Snort" [8] signatures for Nugache that appeared in May 2006 are insufficient to detect all Nugache flows.) More research is necessary to come up with an effective means of IDS detection of Nugache flows.

Nugache can be detected on hosts through various signatures of the infection itself, including Windows Registry keys in HKCU\SOFTWARE\GNU\, a MUTEX lock "d3kb5sujs50lq2mr," the keystroke log file (e.g., C:\Docu-

ments and Settings\user\Application Data\FNTCACHE.BIN), and one of at least three known names for the binary itself (C:\WINDOWS\system32\mstc.exe, system32\mvwatvx.exe, and system32\wmipvs.exe).

Storm can be detected in several different manners, none of which is foolproof.

On the host, one way (outside of noticing the trojan installing itself) to identify Storm is to find the spooldr.ini file. A host-based IDS could be configured to look for that file and understand its contents. Once the file is found, it can be removed, and the system cannot function. When we cleared that file of peers, the trojan was unable to complete the initial Publicize advertisements and was effectively neutralized. Obviously, this is not a long-term solution, as once it becomes public, the format of the file will change. As with Nugache, signatures could be written for different stages of the initial infection. One example of this kind of detection would be to install a system-monitoring tool such as Capture-BAT [9], which was utilized during this investigation to determine what the malware was doing (e.g., the many "writes" to spooldr.ini were one indication that the file was being updated). Incorporating the Capture-BAT output file into a HIDS strategy would provide a good source of intel for which signatures could be written.

On the network side, it's much more difficult to differentiate Storm P2P traffic from legitimate P2P communications. It would be much easier to detect the voluminous amount of outbound TCP/25 traffic from an infected system; however, this is a very reactive strategy. User education is likely the only mitigation method to prevent installation of the malware.

Conclusion

We have just seen a comparison of two recently successful distributed malware networks that employ P2P concepts, in slightly different roles and degrees, for command and control. We must assume that these are just two of the first successful attempts to move away from the central C2 mechanism of IRC botnets, toward distributed attack networks that are significantly harder to detect, to shut down, or to trace back to the attackers who are controlling them. Many papers and articles have predicted this eventuality, and the task now is to understand how the threat landscape has shifted and to adjust mitigation strategies accordingly. As we have seen in the past, the old tools and tactics do not entirely go away, but are joined by new tools and tactics. Likewise, defenses are not to be thrown out, but they must expand to accommodate the new reality of a multiplicity of attack tools and tactics, as well as the old. The trick is to adjust fast enough to avoid giving the attackers the advantage for long, and that is the challenge that we revel in rising to meet.

ACKNOWLEDGMENTS

The authors would like to thank Christian Seifert for his assistance with Capture-BAT.

REFERENCES

[1] D. Dittrich and K. E. Himma, "Active Response to Computer Intrusions," in *The Handbook of Information Security*, edited by H. Bidgoli (Wiley, New York, 2005).

[2] http://en.wikipedia.org/wiki/EDonkey2000.

[3] http://www.secureworks.com/research/threats/storm-worm/?threat=storm-worm.

[4] "The eDonkey 2000 Protocol": ftp://ftp.kom.e-technik.tu-darmstadt.de/pub/papers/HB02-1-paper.pdf.

[5] "An Overview on Peer-to-Peer Information Systems": http://www.p-grid.org/publications/papers/WDAS2002.pdf.

[6] http://www.scripting.com/davenet/2000/11/15/clayShirkyOnP2p.html.

[7] http://www.reconstructer.org/papers/Peacomm.C - Cracking the nutshell.zip.

[8] http://www.bleedingsnort.com/.

[9] http://www.nz-honeynet.org/capture-standalone.html.