

DAVID DITTRICH AND SVEN DIETRICH

## command and control structures in malware

### FROM HANDLER/AGENT TO P2P



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](mailto:dittrich@u.washington.edu)



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

[spock@cs.stevens.edu](mailto:spock@cs.stevens.edu)

### INTERNET ATTACK TOOLS HAVE

evolved, similarly to the way that operating systems and applications themselves have evolved. We will focus on one particular aspect, the mechanisms to allow control of the increasing number of hosts being exploited. The result is an increase in efficiency that allows attackers today to rapidly marshal the computing resources of millions of personal computers across the globe in order to use them for a wide range of criminal activities. In particular, we consider the impact of P2P command and control mechanisms and other features of distributed attack tools that result in a distributed attack network resilient to current methods of detection, monitoring, and take-down by any individual defender or rival.

We look at the impact these structures have on incident response and muse about the trends for the years to come.

### Structure of Distributed Intruder Tool Networks

One of the central problems of a distributed intruder tool network is the topology of the network and the implications for traffic monitoring, enumeration of the entire network, and traceback to the attacker. The latter problem, traceback, is often made more difficult through the use of *stepping stones* (hosts used to redirect connections and “bounce” off a third-party system or network), which leave attackers many “hops” away from their victims.

### SPECIFIC DISTRIBUTED SYSTEM NETWORK STRUCTURES

Since the advent of distributed intruder tools [1, 9], there have been three principal structures employed by distributed system intruder tools: *handler/agent* relationships, central command and control mechanisms (e.g., IRC channels and *botnets*), and *P2P* networks.

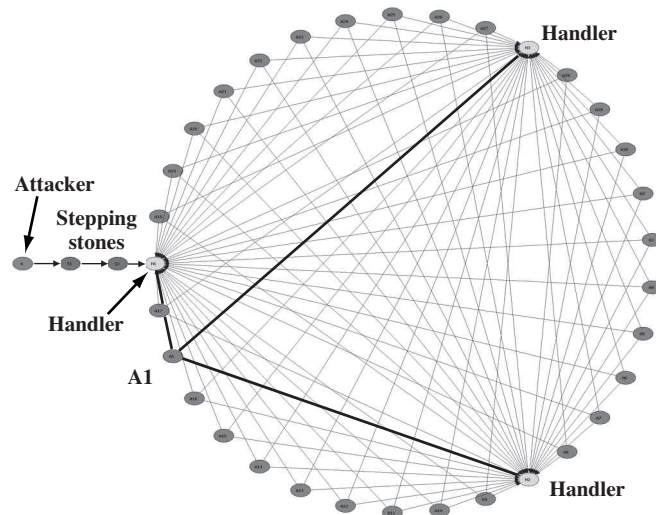
To compare and contrast the various structures of the three principal attack network topologies, we depict each using the same circular layout for highlighting the command and control relationship with attack agents in Figures 1, 2, and 3. The number of nodes (30) in these diagrams was chosen

simply to be manageable in visualizing these attack network structures. In each diagram, nodes depict computer hosts and edges depict the direction of establishing connections, along with two stepping stones and the attacker's primary host (the three nodes that line up on the left of the diagrams). Figures 1 and 2 each has controlling hosts (handlers or IRC servers), whereas Figure 3 does not involve central handlers of either type. To illustrate how traceback is performed from a single known attack agent, one agent or peer (e.g., node A1 or P1) has been selected and all incoming/outgoing connections associated with that host are shown by bold black arrows; all other arrows are in grey.

Typical real-world distributed networks can range from several hundred up to several hundred thousand at a time, with a medium to large botnet in 2006 being on the order of 10,000 hosts. The controversy on how to accurately enumerate botnets remains [12].

#### HANDLER/AGENT NETWORKS

The UNIX-based *Stacheldraht* DDoS tool employed the *handler/agent* model of command and control. In the handler/agent model, the attacker uses a computer with a special malware command/control program (the handler) that coordinates a set of hosts running a different malware attack component (the agents). The attacker connects to the handler, optionally through stepping stones, and controls the attack network. (There are typically no connections between the handlers themselves.) This topology is depicted in Figure 1.



**FIGURE 1: HANDLER/AGENT ATTACK NETWORK**

In most handler/agent DDoS networks, the agents had a predefined list of handlers compiled into the executable program image to which the agent would initially connect on startup. The list would often include at least three handlers, in case one or two had been found and disabled. If all of the handlers had been disabled, the agent would become “lost” and cease to be of use to the attacker, unless or until it was updated with a new copy of the agent that included new handlers. For this reason, a backdoor was sometimes also set up on the compromised host. Some tools allowed for dynamic updating of the handler hosts, but that information would get lost upon restart of the tool. The update affected the memory copy but not the binary stored on disk. For malware only existing in memory and spreading copies of itself, this would of course propagate the correct copy.

If a defender or rival group were to identify an agent (e.g., by noticing the initial attack through which the agent was installed or when local network bandwidth was exhausted during a DDoS attack), it would be easy to trace connections directly back to the handler(s). Figure 1 shows these command and control connections as bold lines from node A1 to the three handler nodes. In the same figure, all three handler connections are shown; however, in practice only one of them may be active at a time. One could then get in touch with someone responsible for security operations at the site that hosted the handler and then monitor network traffic to identify the agents being controlled by that handler. Alternatively, one could seize the host, copy the file system, and recover the list of agents being maintained by the handler. Later versions of handler/agent malware used encryption or obfuscation of the list of agents to make it harder to identify them through file system analysis. Monitoring of network traffic at the site hosting a handler would also allow identification of incoming command and control connections from the stepping stones, allowing traceback to begin toward the attacker.

Even with encryption being used, it was easy to identify the role of handler, agent, and stepping stone in this structure and to act accordingly. (In 1999 and 2000, handler/agent networks of several hundred hosts could be identified from the point of one attacking agent, and the entire network could be identified and taken down in several hours, in the best case.)

One of the primary limiting factors in handler/agent networks resulted from the use of TCP sockets connecting each agent to its handler. Normal UNIX systems intended for desktop use would have their default kernels configured to support a very limited number of concurrently open file handles. Rarely if ever did an attacker stumble upon a highly tuned host that could handle more than the typical default of 1024 file handles, which meant these type of attack networks could not exceed just over 1000 agents total.

Another drawback to handler/agent DDoS tools was that each had its own specialized command and control protocol, which the author would have to maintain in addition to adding code to perform the new functions. This prevents interoperability of handlers and agents from different attack tools. For example, a *Stacheldraht* handler could not be used to control *trinoo* agents, or vice versa.

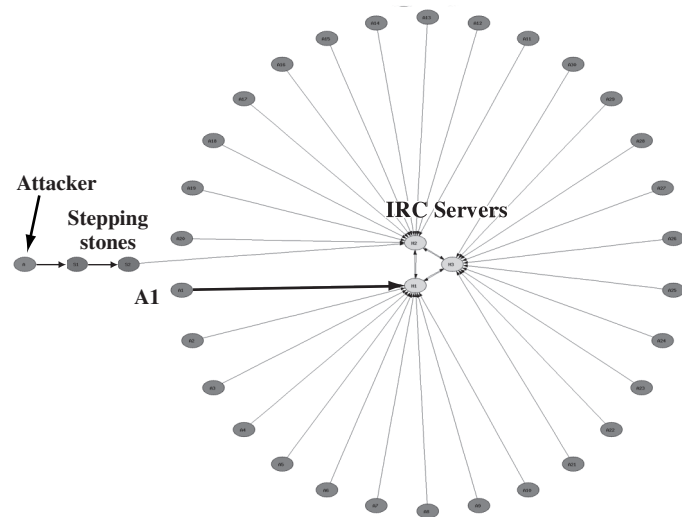
---

#### INTERNET RELAY CHAT NETWORKS

---

Internet Relay Chat (IRC) networks can scale much larger than the handler/agent model will allow, owing to the use of peering servers that can be spread around the globe, each capable of handling a much larger number of concurrent connections (as described in the previous section), in turn relaying chat messages from server to server. This is depicted in Figure 2.

IRC not only serves as a client/server communication network but also provides a defined protocol that can be used by special programs designed to identify specific individuals and commands that appear in IRC channels. The programs that identify these commands and act on them are called *bots*, which is short for *robot*. A set of bots acting together as a group in a single IRC channel is referred to as a *botnet*. (The “topic” of the channel, a string that usually advertises to humans the central chat subject focused on in the channel, sometimes serves as a default command for the bots to act upon when initially joining the channel.) Bots can be implemented in one of two ways: (a) by loading new modules to an existing IRC client (e.g., TCL command scripts executed by the *mIRC* client [7]) or general-purpose bot (e.g.,

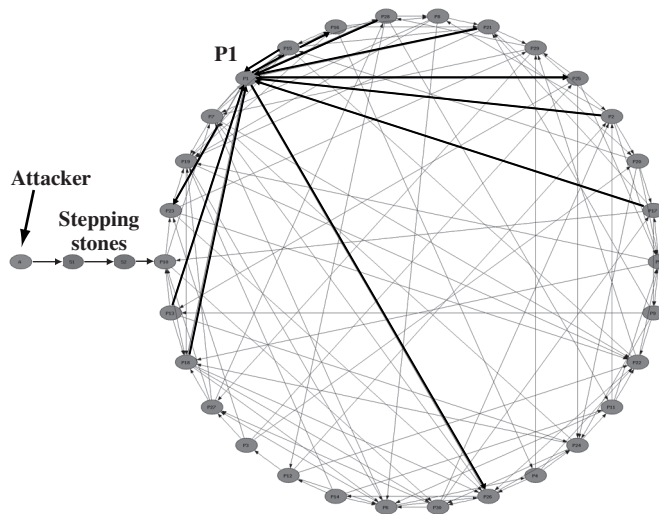


**FIGURE 2: IRC ATTACK NETWORK**

*eggdrop* [16]) or (b) by programming a new bot that “speaks” the IRC protocol and allows direct connection to an IRC server (e.g., knight and Agobot.)

IRC operators (IRC ops) are well aware of botnets and the disruption they cause, so they are constantly on the lookout for thousands of “users” showing up in a short period of time in a given channel or moving from channel to channel *en masse*. More advanced bot herders are skilled at moving bots around, sheltering or combining bots as needed to perform the acts the attacker wishes (such as sending spam, spreading malware, retrieving product keys, or performing distributed denial of service attacks). Bot herders may employ one or more tactics: use of dynamic DNS entries, or short TTLs in DNS records, to point to specific servers for short periods of time (one variation of this is known as *fast flux* [5]); having all or some bots switch IRC channels (*channel hopping*); having all or some bots switch IRC servers (*server hopping*); being redirected to an IRC server or port by downloading a file with HTTP protocol; use of proxies that use port numbers other than the standard IRC server ports (e.g., 6667/tcp); and avoiding the standard IRC networks altogether by setting up customized botnet-tuned IRC server programs on compromised third-party hosts (often called “rogue IRC servers”).

The principal difference from a response perspective between the handler/agent and the IRC command and control structure is the fact that the three IRC server nodes in Figure 2 themselves act similar to stepping stones, so that any node (such as A1 in Figure 2) is only connected to one, while the last stepping stone used by the attacker can be connected to another node, preventing direct traceback from agent to handler to stepping stone. If, for example, the IRC bots were all using encryption for the traffic going over the IRC channel, it would be nearly impossible to trace a connection from our known bot back to the final stepping stone, because there may be hundreds of thousands of connections in total across all three IRC servers and no information that ties any one flow to other flows. Even when encryption is not being used, many bots obfuscate their identity in the chat channel, preventing a direct association between a bot’s name and its actual IP address.



**FIGURE 3: SAMPLE OF IRC BOT COMMAND TRAFFIC**

```

Feb 19 13:36:40 <~foobar> FRA|XXXXXX .login toldo
Feb 19 13:36:40 < FRA|XXXXXX> [r[X]-Sh0[x]]: :( Password Accettata ): .
Feb 19 13:36:41 <~foobar> .opencmd
Feb 19 13:36:42 < FRA|XXXXXX> [CMD]: Remote shell already running.
Feb 19 13:36:54 <~foobar> .cmd mkdir c:\windows\system32\kernel
Feb 19 13:36:55 < FRA|XXXXXX> mkdir c:\windows\system32\kernel
Feb 19 13:36:56 < FRA|XXXXXX> C:\Documents and Settings\KiM>
Feb 19 13:37:00 <~foobar> .cmd cd c:\windows\system32\kernel
Feb 19 13:37:01 < FRA|XXXXXX> cd c:\windows\system32\kernel
Feb 19 13:37:02 <~foobar> .cmd dir
Feb 19 13:37:03 < FRA|XXXXXX> C:\WINDOWS\system32\kernel>dir
Feb 19 13:37:04 < FRA|XXXXXX> Le volume dans le lecteur C n'a pas de nom
Feb 19 13:37:05 < FRA|XXXXXX> Le numero de serie du volume est A443-2CAF
Feb 19 13:37:07 < FRA|XXXXXX> Repertoire de C:\WINDOWS\system32\kernel
Feb 19 13:37:09 < FRA|XXXXXX> 19/02/2005 13:37 .
Feb 19 13:37:10 < FRA|XXXXXX> 19/02/2005 13:37 ..
Feb 19 13:37:11 < FRA|XXXXXX> 0 fichier(s) 0 octets
Feb 19 13:37:13 < FRA|XXXXXX> 2 Rep(s) 8'990'302'208 octets libres

```

**FIGURE 4: PEER-TO-PEER ATTACK NETWORK**

**PEER TO PEER NETWORKS**

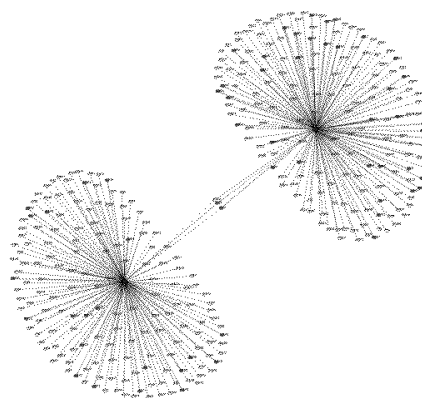
Visually, there is a clear difference between the P2P model and the first two topologies. Figures 1 and 2 show a distinct symmetry, whereas Figure 4 shows a randomness between the number and the direction of connections between peers. In the P2P model, all attack agents form a randomly connected network, where no single host or network of hosts is responsible for central communication. Unlike either of the two previous models, the P2P model does not need any central host or hosts responsible for command and control, or even for joining the P2P network. The SpamThru Trojan [14], which also uses a P2P model for some command and control, also employs a central server for its spam templates. This is a hybrid of the IRC and P2P models. (See the analysis of the Storm and Nugache trojans in this issue, pp. 18–27.)

Commands are retransmitted through the P2P network a limited number of times, enough for all peers to see and act on the command. The attacker is able to connect to any of the peers using a special client program and initiate

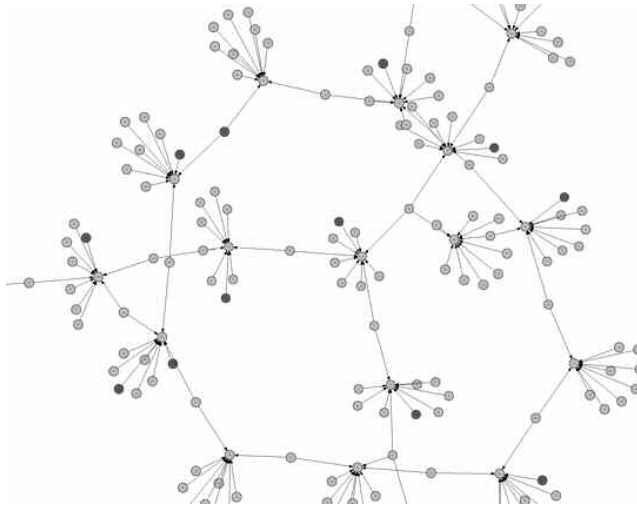
commands, which are then relayed throughout the P2P network. Although we have included the two stepping stones in each diagram for consistency, the stepping stones become unnecessary in the case of P2P networks. In fact, the P2P network becomes both the stepping stones and the command and control channel at the same time, very effectively hiding the IP address of the originator of the commands as well as the complete set of peers. Responses are similarly routed through the network until they reach the intended recipient (or are dropped because they have exceeded a “time-to-live” threshold).

Use of P2P command and control in malware has been attempted before and thought of as early as 2000 [17], but with limited success until 2006. The 2002 Linux Slapper worm source code [3] claimed to use a P2P algorithm that could support 16 million peers; however, the propagation itself was so noisy that the P2P mechanism was never really tested. In 2003 Agobot began to see widespread use. Agobot included a rudimentary P2P mechanism that does not appear to have been popular. This assumption is based on the fact that over 20 versions of Agobot/Phatbot source code analyzed by the authors had no difference at all in any of the P2P related source files, whereas other sections of the code underwent regular and extensive enhancement and bug fixing. In 2004, a version of Phatbot used the WASTE protocol [10] to communicate among the bots. According to Stewart’s Phatbot analysis [6], the encryption capabilities of WASTE were removed to avoid either the problem of key distribution or the weakness of use of a commonly shared key and so that the largest usable P2P network could not exceed much more than 50 peers. In practice, Agobot/Phatbot is almost always controlled using clear-text IRC. When this occurs, all command and control traffic is visible, as is seen in Figure 3. In all these instances, the goal of assembling very large P2P botnets was not realized; however, use of P2P as a means of *spreading* has been successful. A 2006 report indicated that 35% of successful malware infections involved spreading through email, P2P networks, and instant messaging [8].

In the arena of file sharing, specifically anonymous file sharing, P2P mechanisms have also been pursued [11]. Looking at just those P2P networks that use some form of public key exchange and strong cryptography, we see that there are at least five such P2P networks in development.



**FIGURE 5: POSSIBLE CONNECTIVITY GRAPH FOR TWO P2P NODES**



**FIGURE 6: POSSIBLE STRUCTURE OF A SURVIVABLE P2P NETWORK**

### Impact of Structure on Traceback and Response

**TABLE 1: COMPARISON BETWEEN ATTACK NETWORK STRUCTURES**

When the three structures just described are compared as far as traceback and mitigation are concerned, we can see a progression of hardening and in-

Structure	C2 Links per Agent	C2 Links per Handler	C2 Method	Impact of Crypto on Traceback
<i>Handler/Agent</i>	1 per handler	1 (from attacker)	Direct	Low
<i>IRC</i>	1	1 per IRC server	Indirect	Moderate
<i>P2P</i>	Random	NA	Indirect	High

direction that tends toward more resilience as each new structure is employed. This comparison is illustrated by Table 1.

The advent of peer-to-peer command and control of botnets has many implications for incident response and mitigation. In the early days of DDoS, the handler/agent model of command and control was prevalent. To completely mitigate an entire DDoS network, it was necessary to perform manual traceback from one or more DDoS agents to a handler, where an incident responder would then do one of two things:

- Do further manual traceback through traffic analysis to identify all agents in communication with that handler.
- Do host forensics and retrieve the list of agents from a file within the handler's file system, possibly also having to decrypt that file first.

Once all of the agents were identified, the responder would need to send a series of reports and cleanup requests to the owners (as identified by WHOIS records) of all of these IP addresses.

Those steps alone are very time-consuming and complicated, take relatively advanced skills and understanding of DDoS attack tools, methods, operating systems involved, and host- and network-based analysis, and are complicated by differences in time zone, language, legal structures, available re-

sources, available tools, and available skills. For this reason, DDoS-related incidents can last for many months [15].

In case of IRC-based command and control, one could try to detect activity on the IRC port 6667/tcp, assuming the attackers are using a standard IRC server port. Even if they chose not to do so, one could look with `ngrep` for IRC commands, such as joining a channel (`JOIN`). Mitigation would then follow.

The current detection and mitigation approaches, such as DNS-based or text-based signature detection, will be impeded by the features described in this article. There are, however, still actions the user can take, such as using a variety of antivirus programs, rootkit detection programs, or the Microsoft Malicious Software Removal Tool (MSRT). On the network, one would have to look for anomalous behavior on the local networks (e.g., spreading of the malware, spamming, or denial of service). These methods are not foolproof, as the cat-and-mouse game between rootkit authors and defenders continues.

## Trends

More recently, botnets have become the principal subject of research and investigation in academic, commercial, and legal circles. The principal means of fighting botnets is to focus on identifying and removing command and control channels on IRC servers [2, 13]. These efforts have proven effective against the typical botnet in use on standard IRC networks, but they are less effective against “rogue” IRC servers (such as those in [4, 15]) that are either established at sites under control of the person(s) using the botnet, are friendly to them, or are on hosts where little assistance is available to take the system off-line.

Command and control traffic that *does not* utilize a single IRC channel and is heavily encrypted significantly increases the difficulty and time it takes for the entire attack network to be identified and mitigated.

Attackers are realizing the value of sophisticated botnets, if it means that their network can evade detection and perform longer, sometimes for many months. The increased use of various topologies for the botnet (as illustrated in Figures 5 and 6), encryption to evade direct detection, and P2P command and control to avoid a central point of command and control that could be taken down are all characteristics of this development. For examples of two P2P bots, we suggest reading the article by Sam Stover et al. beginning on p. 18 in this issue.

## Conclusion

We have seen the implications of the use of a P2P command and control mechanism, as opposed to either the classic handler/agent or the more contemporary IRC-based central server mechanism, on detection and reaction aspects of investigating and mitigating malicious botnets.

We have also seen the increase in sophistication and breadth of features in malicious software, so it should be expected that more powerful command and control mechanisms will increase the flexibility and dynamism of distributed intruder tool networks in the future. The motivation for this comes from the economic interest in maintaining and retaining control of these attack networks in the face of response or competitors. Two types appear to be

emerging: one that is easy to use by the attacker, characterized by a simplistic or throwaway network, and another, more sophisticated and economically lucrative, characterized by resiliency and survivability of the botnet.

Coming soon to a networked computer near you. Perhaps it's already there?

---

#### ACKNOWLEDGMENTS

The authors would like to thank John Hernandez for his valuable contributions.

---

#### REFERENCES

- [1] CERT Coordination Center, Results of the Distributed-Systems Intruder Tools Workshop, December 1999: [http://www.cert.org/reports/dsit\\_workshop.pdf](http://www.cert.org/reports/dsit_workshop.pdf).
- [2] E. Cooke, F. Jahanian, and D. McPherson, "The Zombie Roundup: Understanding, Detecting, and Disrupting Botnets," *Proceedings of SRUTI '05: Steps to Reducing Unwanted Traffic on the Internet Workshop*, 2005: <http://www.usenix.org/events/sruti05/tech/cooke.html>.
- [3] The HoneyNet Project, Scan of the Month 25: Slapper Worm .unlock.c source file, 2002: <http://www.honeynet.org/scans/scan25/.unlock.nl.c>.
- [4] The HoneyNet Project, "Know Your Enemy: Tracking Botnets," 2005: <http://www.honeynet.org/papers/bots/>.
- [5] The HoneyNet Project, "Know Your Enemy: Fast-Flux Service Networks," 2007: <http://www.honeynet.org/papers/ff/fast-flux.html>.
- [6] LURHQ, "Phatbot Trojan Analysis," June 2004: <http://www.lurhq.com/phantbot.html>.
- [7] K. Mardam-Bey, mIRC homepage, 2006: <http://www.mirc.com/>.
- [8] Microsoft Research, "The Windows Malicious Software Removal Tool: Progress Made, Trends Observed," June 2006: <http://www.microsoft.com/downloads/details.aspx?FamilyId=47DDCFA9-645D-4495-9EDA-92CDE33E99A9&displaylang=en>.
- [9] J. Mirković, S. Dietrich, D. Dittrich, and P. Reiher, *Internet Denial of Service: Attack and Defense Mechanisms* (New York: Prentice Hall PTR, 2004).
- [10] Nullsoft, WASTE, 2003: <http://waste.sourceforge.net/>.
- [11] PlanetPeer.de, PlanetPeer anonymous filesharing wiki: [http://www.planetpeer.de/wiki/index.php/Main\\_Page](http://www.planetpeer.de/wiki/index.php/Main_Page).
- [12] M. A. Rajab, J. Zarfoss, F. Monrose, and A. Terzis, "My Botnet Is Bigger Than Yours (Maybe, Better Than Yours): Why Size Estimates Remain Challenging," in First Workshop on Hot Topics in Understanding Botnets (HotBots '07), April 2007: [http://www.usenix.org/events/hotbots07/tech/full\\_papers/rajab/rajab\\_html/](http://www.usenix.org/events/hotbots07/tech/full_papers/rajab/rajab_html/).
- [13] R. Naraine, "Botnet Hunters Search for 'Command and Control' Servers," eWeek.com, 2005: <http://www.eweek.com/article2/0,1759,1829347,00.asp>.
- [14] SecureWorks, "SpamThru Trojan Analysis," October 2006: <http://www.secureworks.com/analysis/spamthru/>.

[15] United States Department of Justice, U.S. v. James Jeanson Ancheta: <http://news.findlaw.com/hdocs/docs/cyberlaw/usanchetaind.pdf>.

[16] Wikipedia, "Eggdrop," 2006: <http://en.wikipedia.org/wiki/Eggdrop>.

[17] M. Zalewski, "I Don't Think I Really Love You": <http://seclists.org/lists/vuln-dev/2000/May/0159.html>.