# Spam, Phishing, and the Looming Challenge of Big Botnets

Hemmingsen, Ren´e H.; Aycock, John; Jacobson Jr., Michael

# Spam, Phishing, and the Looming Challenge of Big Botnets

René H. Hemmingsen
Department of Computer Science
University of Copenhagen, Denmark
rhaahr@gmail.com

John Aycock and Michael Jacobson, Jr.
Department of Computer Science
University of Calgary, Canada
{aycock,jacobs}@cpsc.ucalgary.ca

TR 2007-865-17, May 2007,
Department of Computer Science, University of Calgary
Presented at the EU Spam Symposium 2007, Vienna, Austria

**Abstract**

What could a spammer or phisher do with a botnet of a thousand machines? a hundred thousand? a million? "Send lots of email" is the least worrisome answer to these questions. As anti-spam and anti-phishing defenses improve, there is more than sufficient financial motivation for spammers and phishers to consider what they can accomplish with enormous scale.

We begin by looking at a wide range of anti-spam defenses. Many of these, like rate limiting and port 25 blocking, will simply no longer work against big botnets; we explain why.

Further, the basic cryptographic assumptions underlying the implementation of SSL certificates and DomainKeys/DKIM need re-examination in light of the massive computing power of big botnets. We describe possible attacks by spammers and phishers, and the implications these attacks have in terms of defense.

## 1 Introduction

It is probably fair to say that a lot of spam, if not the majority of spam, is sent either by or through compromised computers. These compromised computers, called "zombies" or "bots," are connected together into a network, usually referred to as a "botnet," which can be controlled from afar by a spammer.[1]

[1]For convenience, we generically use the term "spammer" in this paper to refer to spammers, phishers, and malware authors propagating malware via spam.

Some botnet researchers have begun to look at the threats posed by future botnets. So far, this work has indicated that future botnets can be highly robust against defensive measures [34, 35] and potentially massive, with multiple small botnets being joined together into one large "super-botnet" [34].

This is the starting point for our work. We assume that a spammer has access to a large botnet – how it is acquired does not matter, nor is it important whether or not the spammer has created the botnet themselves. What *is* important is determining what the spammer can do with it.

We consider a model with four parties involved:

1. Spammer. Obviously, the originator of the spam.
2. Target. The intended recipient of spam.
3. Zombie. A compromised computer the spammer can use to send spam to the target.
4. Victim. The actual owner or user of a zombie computer. As pointed out in previous work [2], the victim will have saved email that the zombie can use. The saved email may not be stored on the zombie machine, e.g., a victim may be a webmail user, but a zombie can simply steal the webmail password and peruse the victim's email at its leisure.

Normally the focus would be just on the spammer and the target, with the zombie as a mere conduit for spam transmission. As we discuss below, zombies (and unwillingly, their victims) play an important role in circumventing some anti-spam defenses. In the next section, we look at a variety of these defenses that can be skirted by big botnets; Section 3 examines some new attacks. Section 4 is the conclusion. We tried to make Section 4 into the introduction again and turn this paper into a Möbius strip, but to no avail.

# 2 Current Anti-Spam Techniques

We have grouped current anti-spam techniques based on their underlying assumption about spammer behavior. We have added citations where available, but many of these techniques are well-known; see [39] for a general reference.

Some of the ideas in this section with respect to authentication have been discussed in [9], which unfortunately had no written record; other ideas were discussed in a different context in [2]. This is, as far as we know, the first attempt to bring these ideas together and categorize them.

## 2.1 Assumption: No Existing Relationship

These defenses operate on the assumption that there is no existing relationship between the spammer and the target.

**Whitelisting.** The target only receives email from senders they know.

**Blacklisting.** The target does not receive email from specific senders. Normally this would be applied by the target's email system to sending IP addresses, to refuse email connections from known spam senders. Ideally, legitimate senders are not blacklisted.

**Spam traps.** Spam traps are fake email addresses that are deliberately meant to the harvested by spammers. Because they don't correspond to real targets, any email sent to spam traps can be considered spam.

**Challenge/response.** The target's email system issues a challenge to a sender – legitimate or spammer – and the sender's email is not delivered to the target until the challenge is correctly replied to. Senders who have passed the test are whitelisted, at least for a while, to prevent a challenge being issued for each exchange in an ongoing email conversation.

A zombie can extract out email addresses from a victim's address book and from saved email, and send the spammer's message to those targets. Effectively the spam travels along social networks, and the spammer exploits the preexisting relationship between victim and target. Challenge/response systems present slightly more of a challenge to a spammer – the spam must be sent to a target where the victim has already replied to the challenge and been temporarily whitelisted. If this is a concern, the zombie would have to limit the search for email addresses to recent entries in saved email.

## 2.2 Assumption: Spam Sent in High Volumes

The assumption here is that spammers send large amounts of email.

**Rate limiting.** The number of emails sent within a given span of time can be throttled [37], which may not prevent spam but can slow it down. A related idea involves limiting the number of concurrent outbound connections to port 25 from a computer, rather than preventing them entirely [23].

**Tarpits.** A tarpit is a mail system that responds very slowly, to try and bog down spam transmission.

**Proof of work.** One of several suggestions to add some kind of cost to sending email, proof of work systems require an email sender to solve a computationally-intensive but tractable problem for each piece of email sent [7].

**Challenge/response.** Appearing twice, the challenges of C/R systems are presumed to be hard enough that a human being must solve them. If so, this limits the volume that a spammer can send, or at least changes the economics if humans have to be paid to solve challenges.

This assumption is endemic even in recent work, e.g., [3]. With large botnets, though, each zombie need only send a small amount of spam to produce the same overall amount of spam. In addition, large botnets allow the computational problems of proof-of-work to be solved using what is effectively a compute farm. It also opens

up ways to trick humans into solving challenges: for example, a zombie could intercept a victim's browser, telling the victim they have to solve the presented challenge before they can visit a web page.[2]

(As an aside, proof-of-work has also been criticized elsewhere [15], although they too make the assumption about high spam volumes.)

## 2.3 Assumption: Legitimate Mail System Not Used

Other defenses assume that spammers do not send email using "legitimate" mail systems, opting instead to send in quick-and-dirty ways like direct-to-MX transmission.

**Blacklisting.** Blacklisting also appears here, because zombies sending email directly are frequent occupants of RBL lists. By contrast, much if not most legitimate email is sent via an ISP's mail system rather then directly.[3]

**Greylisting.** Initial attempts at email delivery, regardless of sender, are told try-again-later by a target's email system, on the belief that spammers will not retry transmission [16].

**Verification.** Verification has two flavors. First, it can be verified that a message came from where it says it did, and that it hasn't been altered in transit (DKIM/DomainKeys [1, 6]). Second, it can be verified that a sending machine is authorized to send mail for a domain (SPF [38]).

**Reputation systems.** Reputation systems classify email based on a sending machine's past spam-sending behavior; the email content is of secondary importance [28].

**Port 25 blocking.** Outgoing port 25 (SMTP) connections can be blocked by an ISP to force the use of their legitimate email system, and completely prevent direct email delivery by zombies.

**Simulated tarpits.** Recent work suggests simulating the behavior of a tarpit, expecting real spammers to time out quickly and give up, whereas legitimate mail systems will persist long enough to overcome the initial slowness. Connections that stick around long enough are ramped back up to full speed [8].

A zombie can simply piggyback on their victim's legitimate email system, transmitting spam through it instead of directly. In the case of reputation systems, the victim's reputation (or that of the victim's ISP) is inherited by the zombie.

## 2.4 Assumption: Many Emails, Same Content

**Checksumming.** Checksums of incoming email messages are computed and compared against the checksums of known spam messages. Done in large scale,

[2]Note that this is distinct from the "porn proxy attack" because the spammer does not have to establish an enticing website.

[3]The comments about ISPs in this paper can also be taken to apply to corporate networks, or any large enterprise network.

this also permits the ability to detect patterns, like the same piece of email being sent to thousands of people.

Techniques to try to evade checksumming have been around for some time, from simple random number "hashbusters" to more recent random alterations of image spam (e.g., [25]). This has opened up yet another anti-spam front, where checksumming algorithms are tweaked to be tolerant to random changes. There is no reason why spam cannot be customized extensively by each zombie in a botnet, either randomly or based on content on the victim's machine; in many ways, this is the spam equivalent of polymorphic viruses, creating millions of different variants of a single spam message.

## 2.5 Assumption: High Spammy Content

**Statistical filtering.** Emails can be classified as spam or ham based on whether or not their words (for some definition of what a "word" is) look "spammy" compared to past emails. Words that have not been seen previously are assigned a neutral score, and these filters can learn through retraining.

Spammers who randomly add words ("word salad" [12]), excerpts from online sources, or news articles to their spam can be seen to be hashbusting. It can also be seen as trying to skew statistical filters away from a spam classification. This is not very sophisticated or directed yet, and the opinion has been expressed that attacks against statistical filters would be too hard to mount [11, 39]. However, we argue that a zombie *can* perform a directed attack automatically: ham words that pass through a target's filter are sitting in the victim's saved email.

## 2.6 Assumption: Mail is Spam or Ham

One final assumption about spammer behavior is that a spammer only sends spam, and email can be correspondingly classified in a binary way, as either spam or ham. Research has demonstrated that this need not be the case in two scenarios, where a single email can be both ham *and* spam. Zombies play a part in both of these.

First, parasitic spam relies on a zombie to tag a brief spam signature onto the victim's legitimate email, or changes URLs in legitimate email to malicious ones [26, 27]. The ham portion is legitimate, although possibly altered, and the message cannot be filtered out completely as a result.

Second, "more effective spam" is where a zombie custom-tailors email to a target, using the victim's saved email and (optionally) legitimate incoming email from the target [2]. Again classification is not straightforward, because real ham messages are mixed with the spam content. However, in this case, the ham is a red herring and the entire message can be flagged as spam without loss of legitimate content.

## 2.7 Discussion

The bad news is that all the applications of zombies we have presented scale to big botnets. Big botnets just present more opportunities for spammers: more social networks to exploit, more saved email, more targets.

The good news is that these uses of big botnets by spammers drive more spam through ISPs' mail systems, creating more potential for ISP filtering of outbound email. It also allows the zombies to be discovered in two ways. First, a zombie sending spam using a victim's account can be traced. Second, sending spam along social networks should increase the number of victims who are notified about zombie infestations from their friends and colleagues.

However, we would urge caution. With enough zombies, a spammer can potentially have enough to send spam, lose zombies to detection, *and* infect enough new machines to maintain the botnet's size. The assumption that the spammer fears detection of their zombies may be incorrect at a large scale.

# 3 New Attacks

The cryptographic assumptions underlying some defenses against spam (DKIM/DomainKeys) and phishing (SSL certificates) are not often questioned. However, considering the potential size of spammers' botnets, it is prudent to re-examine these ideas; spammers may possess enough computing power for new forms of attacks.

## 3.1 Fake Certificate Authority

This first attack can be performed without a big botnet, but serves to illustrate that certificates are not a silver bullet, and attacks involving them can occur. As a simple example of an attack on certificates, consider the following scenario:

1. A worm is released whose payload installs a new root certificate for the spammer in a victim's browser or, alternatively, a zombie installs one. Once the certificate is installed, the malware can remove itself, as it is no longer needed for the attack.

2. Phishing emails can now be sent by the spammer to a victim. When victims go to the spammer's web site, they are greeted by a valid SSL connection with no warning messages from the browser. The spammer's root certificate allows them to create real SSL certificates, a process that can continue until the spammer's root certificate is removed.

We tested this attack by creating a root certificate for the fictitious "VeríSign Trust Network" (note the accent over the "i") and inserted it into our Firefox installation.

In this case, we used a pharming attack in conjunction with the root certificate, redirecting DNS lookups for `www.bank.com` on our machine to another machine we had access to – call it `example.com`. For pharming, we simply inserted an appropriate entry into `/etc/hosts`, but other techniques are possible [19, 32].

On `example.com`, we started an HTTPS server which was equipped with a certificate attesting to it being `www.bank.com`, signed by our "VeríSign Trust Network."

The result was as anticipated. When we pointed our copy of Firefox to `https://www.bank.com/`, it went to our HTTPS server on `example.com`. There were no certificate warnings popped up, and all of Firefox's secure connection indicators were present. Mousing over the padlock icon produced a tooltip saying "Signed by VeríSign Trust Network"

in a small enough font that the accent over the "i" was next to impossible to see with a casual inspection. Even if a user had been trained to look for the secure connection indicators, it would not have helped in this case. This suggests that security software should be guarding certificates as well.

## 3.2 Certificate Forgery

In this section, we examine whether it is probable, or even possible, that spammers will be able to use the processing power present in their botnets to take on a calculation-heavy problem such as breaking the digital signature on a digital certificate. If successful, spammers would be capable of creating SSL server certificates that are indistinguishable from the legitimate ones. The impact this would have on security in relation to e-commerce and other online services is profound.

We consider X.509 certificates, the most common digital certificate standard in use today. If a spammer is to successfully spoof a public key certificate, i.e., forge a certificate that is indistinguishable from a legitimate certificate, then they need to break the digital signature and recover the signing CA's private key. If the spammer accomplishes this, then they can, for example, make a phishing site, `www.bank.com`, appear legitimate by creating a certificate that has the elements of the subject field set accordingly, i.e., `CN=www.bank.com`, `O=BigBank`, and then subsequently sign the fabricated certificate with the recovered private key. Breaking the digital signature and recovering the signing CA's private key is, however, not trivial. The most widely-used digital signature encryption algorithm in this context is RSA [20]. RSA is normally used with 1024-bit keys so a brute force attack is not computationally feasible since the key space is extremely large, $2^{1024} > 1.8 \times 10^{308}$. A number of attacks on RSA have been proposed that are more advanced than brute-force approaches; [17] describes some of them.

The problem with public key cryptography is that it is based on computational problems – factorization of integers in the case of RSA – and this makes it more vulnerable to attacks. Here, the spammer would know the CA's digital signature in the certificate, and the CA's public key. Based on this information, finding the CA's private key is computationally equivalent to the problem of factoring $n$, the modulus in the RSA algorithm [17]. The best-known algorithm used to factor $n$, the number field sieve [4], is very easy to distribute, and it is highly resilient to computing nodes going offline.

Since the security of the RSA algorithm rests on the difficulty of factoring large values of $n$, RSA Laboratories maintains a list of "RSA Challenge Numbers" [22]. Researchers are challenged to factor the numbers on the list for a symbolic cash reward. Currently, the list consists of eight numbers of lengths from 576 bits to 2048 bits, each chosen as if it were to be used as the modulus $n$ in the RSA algorithm. The two smallest numbers, RSA-576 and RSA-640, have both been factored. The latter was factored in 2005, and reputedly took approximately 30 2.2 GHz Opteron-CPU years [21].

In [18], a 2.6 GHz Opteron CPU was benchmarked, giving a result for CPU arithmetic of 38,228 MIPS. We will thus use the conservative estimate of 35,000 MIPS for a 2.2 GHz Opteron CPU. Based on this estimate, the computing power required to factor RSA-640 would correspond to $30 \times 35,000 = 1 \times 10^6$ MIPS-years. Table 1 summarizes the MIPS-year estimates for four of the RSA challenge numbers. Note that the

| Number | Bit length | Factored | MIPS years |
|---|---|---|---|
| RSA-140 | 256 | Feb 1999 | $2 \times 10^3$ [30] |
| RSA-155 | 512 | Aug 1999 | $8 \times 10^3$ [29] |
| RSA-640 | 640 | Nov 2005 | $1 \times 10^6$ |
| RSA-1024 | 1024 | not yet factored | $5.6 \times 10^{10}$ [24] |

Table 1: MIPS-year estimates for selected RSA challenges

estimates we have used here are older ones; we have deliberately chosen these to be conservative, knowing that they do not take into account recent improvements in the field.

How much processing power could a spammer accumulate? We begin by quantifying the average power of individual zombies in a botnet. In [31], the rating of a 2.8 GHz Pentium 4 CPU is estimated at 6,775 MIPS. Since a 2.8 GHz Pentium 4 is probably still high-end compared to an average PC, we take the average processing power of a single zombie to be only 5,500 MIPS.

A zombie computer may not be able to join in the factoring process continuously, due to:

- Availability. A zombie may not always be available to a spammer; a home PC might only be powered on for brief periods every day, whereas a corporate computer may be online 24/7. Here the availability is conservatively set to be 20%.
- Utilization. Even when online, zombies will not spend all their CPU cycles factoring – the victims would undoubtedly notice the performance deterioration and take steps to fix the problem. We take the utilization to be 25%, assuming most users would not notice a one-quarter drop in CPU performance.

This gives us the following equation for an average zombie's processing power, $P_{avg}$.

$$\begin{aligned} P_{avg} &= \textit{processing power} \times \textit{availability} \times \textit{utilization} \\ &= 5,500\,\text{MIPS} \times 20\% \times 25\% \\ &= 275\,\text{MIPS} \end{aligned}$$

Now we simply need to multiply $P_{avg}$ by the botnet's size. [13] claims that some botnets only have a few hundred zombies, so we take our "small" botnet to be of size 300, giving $8.25 \times 10^4$ MIPS. Some sources attest to botnets with over 100,000 zombies [5, 14, 33], and consequently our "large" botnet has 100,000 zombies with $2.75 \times 10^7$ MIPS. Finally, to represent the "super" botnets of [34], we use 1,000,000 zombies yielding $2.75 \times 10^8$ MIPS.

Table 2 gives the approximate time it would take to factor the RSA numbers given in Table 1 using the different-sized botnets. Clearly at least a large botnet is required for keys of length 640, although even small botnets could handle 512-bit keys. None would be able to break a 1024-bit RSA key. Since no major CA uses a key less than

| Number | Small botnet | Large botnet | Super-botnet |
|---|---|---|---|
| RSA-140 | 9 days | 38 minutes | 4 minutes |
| RSA-155 | 35 days | 153 minutes | 15 minutes |
| RSA-640 | 12 years | 13 days | 1.3 days |
| RSA-1024 | $6.8 \times 10^5$ years | 2036 years | 204 years |

Table 2: Estimated time for botnets to factor RSA numbers

1024 bits, we must conclude that this particular attack is practically infeasible given current processor technology and cryptographic knowledge.[4]

### 3.3 DKIM/DomainKeys Forgery

Even if certificates are safe, perhaps there is some algorithm using RSA with shorter keys, where 512-bit keys are required to be accepted, where relatively recent examples of 512-bit keys in use have been seen. Like DKIM/DomainKeys.

Both DKIM and DomainKeys mandate support for 512-bit keys, although DKIM requires at least 1024 bits for 'long-lived keys' [1, page 13]. A spammer who can use their botnet's computing power to recover an organization's private key for DKIM/DomainKeys can forge that organization's signatures on email. Obviously 512-bit keys, even if permitted, are not a wise choice in the face of big botnets.

### 3.4 Hash Collisions

Both certificates and DKIM/DomainKeys employ the SHA-1 cryptographic hash function. SHA-1 is currently in a weakened state, and increasingly better methods are being discovered for finding collisions in it [36]. For now, this only means that two messages can be found that result in the same hash value. What a spammer would want instead is to be able to find message whose hash value matches a *specific* legitimate message's hash value; this is a stronger requirement. A spammer would then be able to substitute their message for a legitimate one without having to crack RSA. The consensus seems to be that it is only a matter of time before cryptography researchers are able to solve this problem, necessitating a switch to a stronger hash function. The implications of hash collision attacks are examined in [10].

## 4 Conclusion

Big botnets present a variety of challenges to spam and phishing defenses. A key part of the problem, and the solution, is understanding the magnitude of the computing power and information that these botnets put within reach of spammers.

[4]There are additional hurdles in terms of memory availability on the zombies [24], but the time requirements alone are enough to discount this attack.

## Acknowledgment

The second and third authors' research is supported in part by grants from the Natural Sciences and Engineering Research Council of Canada.

## References

[1] E. Allman, J. Callas, M. Delany, M. Libbey, J. Fenton, and M. Thomas. DomainKeys identified mail (DKIM) signatures. http://www.ietf.org/internet-drafts/draft-ietf-dkim-base-10.txt, February 2007. Internet draft.

[2] J. Aycock and N. Friess. Spam zombies from outer space. In *15th Annual EICAR Conference*, pages 164–179, 2006.

[3] A. Brodsky and D. Brodsky. A distributed content independent methods for spam detetion. In *First Workshop on Hot Topics in Understanding Botnets*, 2007.

[4] J. P. Buhler, H. W. Lenstra, Jr., and C. Pomerance. Factoring integers with the number field sieve. In *The Development of the Number Field Sieve*, pages 50–94, 1993.

[5] D. Dagon, G. Gu, C. Zou, J. Grizzard, S. Dwivedi, W. Lee, and R. Lipton. A taxonomy of botnets. Unpublished paper, c. 2005.

[6] M. Delany. Domain-based email authentication using public keys advertised in the DNS (DomainKeys). http://www.ietf.org/internet-drafts/draft-delany-domainkeys-base-06.txt, July 2006. Internet draft.

[7] C. Dwork and M. Naor. Pricing via processing or combatting junk mail. In *Proceedings of CRYPTO '92*, pages 139–147, 1993.

[8] T. Eggendorfer. Reducing spam to 20% of its original value with a SMTP tar pit simulator. In *MIT Spam Conference*, 2007.

[9] N. FitzGerald. Why 'user authentication' is a bad idea. In *Virus Bulletin Conference*, page 226, 2005. Abstract only.

[10] P. Gauravaram, A. McCullagh, and E. Dawson. Collision attacks on MD5 and SHA-1: Is this the "Sword of Damocles" for electronic commerce? In *Proceedings of AusCERT Asia Pacific Information Technology Security Conference (AusCERT2006): Refereed R&D Stream*, pages 73–88, 2006.

[11] J. Graham-Cumming. Fooling and poisoning adaptive spam filters. Sophos white paper, November 2003.

[12] J. Graham-Cumming. How to beat an adaptive spam filter. In *MIT Spam Conference*, 2004.

[13] Honeynet Project & Research Alliance. Know your enemy: Tracking botnets. http://www.honeynet.org/papers/bots/, 2005.

[14] A. Householder and R. Danyliw. Increased activity targeting Windows shares. CERT Advisory CA-2003-08, 11 March 2003.

[15] B. Laurie and R. Clayton. "proof-of-work" proves not to work. In *Third Annual Workshop on Economics and Information Security*, 2004.

[16] J. R. Levine. Experiences with greylisting. In *Second Conference on Email and Anti-Spam*, 2005.

[17] A. Menezes, P. van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[18] S. Mohammadi and A. Roos. Intel Xeon and AMD Opteron battle head to head. http://www.tomshardware.com/2006/10/26/intel_woodcrest_and_amd_opteron_battle_head_to_head/, 2006.

[19] G. Ollmann. The pharming guide. NGSSoftware white paper, 2005.

[20] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[21] RSA Laboratories. RSA-640 is factored! http://www.rsa.com/rsalabs/node.asp?id=2964, 2005.

[22] RSA Laboratories. The RSA challenge numbers. http://www.rsa.com/rsalabs/node.asp?id=2093, 2007.

[23] J. E. Schmidt. Dynamic port 25 blocking to control spam zombies. In *Third Conference on Email and Anti-Spam*, 2006.

[24] R. Silverman. A cost-based security analysis of symmetric and asymmetric key lengths. http://www.rsa.com/rsalabs/node.asp?id=2088, 2001.

[25] J. Stewart. SpamThru Trojan analysis. http://www.secureworks.com/research/-threats/spamthru, 2006.

[26] D. Swimmer. The spectre of parasitic spam. *Virus Bulletin*, pages S2–S4, December 2005.

[27] M. Swimmer, B. Leiba, I. Whalley, and N. Borenstein. Breaking anti-spam systems with parasitic spam. In *Third Conference on Email and Anti-Spam*, 2006.

[28] B. Taylor. Sender reputation in a large webmail service. In *Third Conference on Email and Anti-Spam*, 2006.

[29] H. te Riele. Factorization of a 512-bits RSA key using the number field sieve. http://ftp.cwi.nl/herman/NFSrecords/RSA-155, 1999.

[30] H. te Riele. Factorization of RSA-140 using the number field sieve. http://ftp.cwi.nl/herman/NFSrecords/RSA-140, 1999.

[31] Tom's Hardware Guide. Cpu charts. http://www23.tomshardware.com/cpu.html, 2007.

[32] A. Tsow, M. Jakobsson, L. Yang, and S. Wetzel. Warkitting: The drive-by subversion of wireless home routers. *Journal of Digital Forensic Practice*, 1(3):179–192, 2006.

[33] United States v. Ancheta. Case CR05-1060, Indictment, U.S. District Court, Central District of California, February 2005.

[34] R. Vogt, J. Aycock, and M. Jacobson, Jr. Army of botnets. In *14th Annual Network and Distributed System Security Symposium*, pages 111–123, 2007.

[35] P. Wang, S. Sparks, and C. C. Zou. An advanced hybrid peer-to-peer botnet. In *First Workshop on Hot Topics in Understanding Botnets*, 2007.

[36] X. Wang, Y. L. Yin, and H. Yu. Finding collisions in the full SHA-1. In *CRYPTO 2005*, pages 17–36, 2005.

[37] M. M. Williamson. Design, implementation and test of an email virus throttle. In *19th Annual Computer Security Applications Conference*, 2003.

[38] M. Wong and W. Schlitt. Sender policy framework (SPF) for authorizing use of domains in e-mail, version 1. RFC 4408 (Experimental), 2006.

[39] J. A. Zdziarski. *Ending Spam*. No Starch Press, 2005.