On the Spam Campaign Trail

Christian Kreibich[†] Chris Kanich[*] Kirill Levchenko[*] Brandon Enright[*]
Geoffrey M. Voelker[*] Vern Paxson[†] Stefan Savage[*]

[†]International Computer Science Institute
Berkeley, USA
christian@icir.org,vern@cs.berkeley.edu

[*]Dept. of Computer Science and Engineering
University of California, San Diego, USA
{ckanich,klevchen,voelker,savage}@cs.ucsd.edu
bmenrigh@ucsd.edu

1 Introduction

Over the last decade, unsolicited bulk email, or *spam*, has transitioned from a minor nuisance to a major scourge, adversely affecting virtually every Internet user. Industry estimates suggest that the total daily volume of spam now exceeds 120 *billion messages per day* [10]; even if the actual figure is 10 times smaller, this means thousands of unwanted messages annually for every Internet user on the planet. Moreover, spam is used not only to shill for cheap pharmaceuticals, but has also become the de facto delivery mechanism for a range of criminal endeavors, including phishing, securities manipulation, identity theft and malware distribution. This problem has spawned a multi-billion dollar anti-spam industry that in turn drives spammers to ever greater sophistication and scale. Today, even a single *spam campaign* may target hundreds of millions of email addresses, sent in turn via hundreds of thousands of compromised "bot" hosts, with polymorphic "message templates" carefully crafted to evade widely used filters.

However, while there is a considerable body of research focused on spam from the recipient's point of view, we understand considerably less about the *sender's* perspective: how spammers test, target, distribute and deliver a large spam campaign in practice. At the heart of this discrepancy is the limited vantage point available to most research efforts. While it is straightforward to collect individual spam messages at a site (e.g., via a "spam trap"), short of infiltrating a spammer organization it is difficult to observe a campaign being orchestrated in its full measure. We believe ours is the first study to approach the problem from this direction.

In this paper, we explore a new methodology—distribution infiltration—for measuring spam campaigns *from the inside*. This approach is motivated by the observation that as spammers have migrated from open relays and open proxies to more complex malware-based "botnet" email distribution, they have unavoidably opened their infrastructure to outside observation. By hooking into a botnet's command-and-control (C&C) protocol, one can infiltrate a spammer's distribution platform and measure spam campaigns as they occur.

In particular, we present an initial analysis of spam campaigns conducted by the well-known Storm botnet, based on data we captured by infiltrating its distribution platform. We first look at the system components used to support spam campaigns. These include a work queue model for distributing load across the botnet, a modular campaign framework, a template language for introducing per-message polymorphism, delivery feedback for target list pruning, per-bot address harvesting for acquiring new targets, and special test campaigns and email accounts used to validate that new spam templates can bypass filters. We then also look at the dynamics of how such campaigns unfold. We analyze the address lists to characterize the targeting of different campaigns, delivery failure rates (a metric of address list "quality"), and estimated total campaign sizes as extrapolated from a set of samples. From these estimates, one such campaign—focused on perpetuating the botnet itself—spewed email to around 400 million email addresses during a three-week period.

2 Background

The origins of spam date back to the early-1990s when legitimate advertisers and scammers alike began to realize the capability of email to reach large numbers of potential customers or marks. As spam's prevalence increased, so too did attempts to stop it, whether by maintaining "blacklists" of IP addresses or filtering on spam content itself. In their quest to reach their targets spammers have ever adapted to these methods. Where they once were able to send spam from a few servers under their control, IP blacklists have forced the development of bot-based distribution networks that use compromised PC's to relay messages and launder their true origin. Similarly, while spammers could once send the same message to all their targets, the use of filters based on statistical learning have in turn caused spammers to dynamically add textual polymorphism to their spam, thus evading the filters.

The research community has come relatively late to the study of spam, but in recent years there has been considerable attention focused on characterizing important aspects of the spam enterprise including address harvesting [13], the network behavior of spam relays [17], the hosting of scam sites [1] and advances in filter evasion [14]. Similarly, botnets themselves have enjoyed considerable attention from security researchers, including both case studies and analyses of size, number, activity, membership, and dynamics [4, 18, 6, 15, 21, 2, 11, 16]. Here too, advances in defenses have provoked improvements in the underlying C&C technology. While early botnets depended exclusively on centralized C&C channels (typically IRC), modern botnets have developed increasingly sophisticated methods for obfuscating or minimizing their C&C exposure. Most recently, the Storm botnet has become the first that implements a directory service upon a distributed hash table [12, 9, 19]. In turn, this is used to bootstrap a custom C&C protocol that constructs a multi-level distribution hierarchy with a layer of message relays isolating the "foot soldier" bots, who blindly poll for commands, from a smaller set of servers that control them. While Storm represents the current technological front for spammers, at its core the basic methods are structurally the same as all spam distribution platforms of which we are aware.

Spammers divide their efforts into individual campaigns that are focused on a particular goal, whether it is selling a product, committing financial fraud, or distributing malware. Abstractly, we think of each *spam campaign* as consisting of a *target list* of email addresses—either harvested via crawling or malware or purchased outright via underground markets [5]—along with a set of subject and body text *templates* that are combined mechanically to create an individual message for each targeted address. A campaign may consist of one or more such runs and thus can vary in length from only a few hours to as long as months (as evidenced by Storm's e-card campaign of mid-2007 [3]). In turn, a spam campaign is executed by some *distribution platform*—typically a botnet—and this infrastructure can be reused by multiple campaigns (conversely there is anecdotal evidence of individual campaigns moving to using different botnets at different points in time). For reasons of scalability, this infrastructure is typically responsible for the task of evading textual spam filters and thus must generate each message algorithmically based on the campaign's text templates and a set of evasion rules, or *macros*. Also for scalability, the load of delivering a spam campaign must be balanced across the infrastructure. While the exact method can vary (e.g., pull-based vs push-based bots), the logic is nearly universal: a campaign is quantized into individual work requests—consisting of a subset of the target list and, optionally, updated templates and macros—that are distributed as updates from the spammer to the individual distribution nodes. Finally, the infrastructure can report back failures, allowing the spammer to weed out addresses from their target list that are not viable.

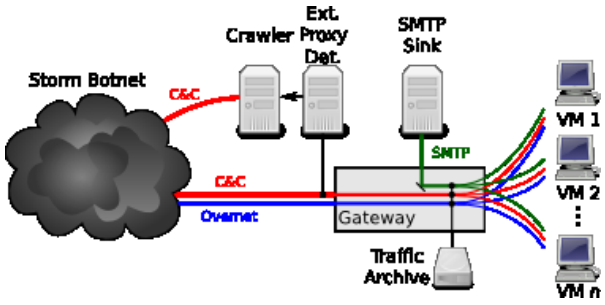


Figure 1: Experimental setup.

3 Data collection

Our measurements come from both instrumentation and probing of the Storm botnet [12, 20]. For instrumentation we run bots in controlled environments, and for probing we use crawling of Storm proxies, as described below. The key enabler for our infiltration effort was our development of the capability to analyze the different forms of Storm communication traffic, which required a significant reverse-engineering effort.

Storm employs a tiered coordination mechanism. At the lowest level, *worker* bots access a form of the Overnet peer-to-peer network to locate C&C *proxy* bots. Workers relay through the proxies requests for instructions and the results of executed commands, receiving from them their subsequent C&C. The proxies in turn interact with "bullet-proof hosting" sites under control of the botmaster. (Note that our work here focuses entirely on the proxy-based C&C mechanism; we do not employ any form of Overnet monitoring or probing.)

From late Dec. 2007 through early Feb. 2008 we ran 16 instances of Storm bots[1] in virtual machines hosted on VMware ESX 3 servers. Depending on the machines' configuration, these bots could run as either workers or proxies. As workers, they would contact remote proxies and receive instructions such as spamming directives. As proxies, they would themselves be contacted by remote workers requesting instructions, which they recorded and then relayed back into Storm.

In addition, we analyze outgoing C&C requests from our workers in order to discover the external proxies to which they attempt to communicate. We feed these proxy identities to a custom crawler that mimics the presence of additional workers, repeatedly querying each active proxy for the latest spamming instructions.

Figure 1 summarizes the experimental setup (here, Overnet is shown only due to its use by our workers to

Bot-based Dataset		
Timeframe	26 Dec 07 – 04 Feb 08	
C&C messages	1.03M	
SMTP traffic	3.36M	(to Sink)
Contacts from ext. workers	145,585	
Contacts to ext. proxies	1,086	
Total update messages	208,463	
Ext. delivery reports	50,131	
Ext. harvest reports	272,546	(9.9% non-empty)
Total spam templates	172,498	(13.3% unique)
Total targeted addresses	66,698,722	(97.7% unique)
Ext. harvested addresses	463,580	

Crawl-based Dataset		
Timeframe	20 Nov 07 – 13 Feb 08	
Proxies contacted	45,909	(722 distinct)
Spam templates extracted	16,977	
Email addresses extracted	11,487,402	

Table 1: Summary of datasets used in the study.

locate proxies), and Table 1 summarizes the contents of data we gathered with the setup for our study. The first group of figures refers to measurements from the bot-based components of our setup (workers and proxies we ran in our controlled environment). Here, the term "External" refers to information sent to us by remote workers contacting our local proxies; "Total" refers to volumes summed across both such external reports plus the activities of our local workers. (Our local workers do not generate meaningful harvesting figures, which they are unable to perform in an effective fashion. Also, our local workers did not attempt to send spam until Jan 7.) The second group of figures refers to measurements extracted from the crawler component of our setup.

4 Campaign mechanics

In this section we briefly describe how Storm worker bots are instructed to construct the individual spam messages that they then attempt to propagate.

4.1 Message structure & propagation

In general, workers acquire new tasks in a pull-based fashion, by actively requesting *update* messages from their proxies. (Similarly, they send back delivery and harvest reports asynchronously, at a time of their choosing.) Update messages consist of three sections, each possibly empty (if modifying elements of a previous update). These are: (i) template material; (ii) sets of *dictionaries* containing raw text material to substitute into templates; and (iii) lists of target email addresses. These lists typically provide roughly 1,000 addresses per update message. Templates and target lists are labeled with small integers, which we term a *slot*. Spam constructed from a given template is sent to targets in lists labeled with the corresponding slot numbers. Delivery reports mirror the target list structure of update messages, with addresses listed in full upon success, and otherwise an error code in its stead, reporting the cause of failure. Harvest reports contain zero or more addresses, not further structured.

4.2 Templating mechanism

For spam construction, Storm implements a fairly elaborate template language, supporting formatting macros with input arguments for text insertion and formatting, generation of random numbers, computation of MTA message identifiers, dates, and reuse of results of previous macro invocations. Macros are delineated by a start marker "`%^`" and a corresponding end marker "`^%`". We use parentheses below instead of Storm's markup to ease readability. A single letter after the initial marker identifies the macro's functionality. It is followed by zero or more macro input arguments, which may consist of the output of nested macros. We verified the meaning of the different macro types seen in real traffic by feeding suitably crafted templates to the workers in our setup and observing the resulting spam they attempted to send. In addition, we also tested the letters of the alphabet not encountered in real templates, to see whether they would provide any functionality. This way, we discovered ten additional language features. Table 2 summarizes the language features we identified.

Figure 2 shows the header part of a template as used by Storm, together with the resulting email message header. The `from`-part of the `Received` header serves as a good example of macro use. The content is constructed as follows:

$$(C0\ \ (P(R2-6) : qwertyuiopasdfghjklzxcvbnm)\ .\\ \quad (P(R2-6) : qwertyuiopasdfghjklzxcvbnm))$$

This macro builds two character strings of length between 2 and 6 characters, randomly taken from the given character string, places them around a dot character, and labels the output string "0." The result is later picked up in the `Message-ID` header. The `From:` line in the template illustrates the generation of text via selection from a dictionary, thereby randomizing the apparent sender of the message.

5 Measurements of bot spam activity

We now turn to characterizing the observed behavior of workers as they construct, transmit, and report back on spam batches. Note that characterizations can reflect observation of our local workers; remote workers that reported to our local proxies; or remote proxies sending instructions to our crawler. When unclear from context, we clarify which subsets of our data apply to each discussion.

Overall, we observed bots sending spam to 67M target addresses with very little redundancy: over 65M of

MACRO	SEEN LIVE	FUNCTIONALITY
(0)	✓	Spam target email address.
(A)	✓	FQDN of sending bot, as reported to the bot as part of the preceding C&C exchange.
(B)		Creates content-boundary strings for multi-part messages.
(C*num*)	✓	Labels a field's resulting content, so it can be used elsewhere through (V); see below.
(D)	✓	Date and time, formatted per RFC 2822.
(E)		ROT-3–encodes the target email address.
(F*string*)	✓	Random value from the dictionary named *string*.[2]
(G*string*)	✓	Line-wrap *string* into 72 characters per line.
(H*string*)		Defines hidden text snippets with substitutions, for use in HTML- and plain-text parts.
(I)	✓	Random number between 1 and 255, used to generate fake IP addresses.
(J*string*)		Produces quoted-printable "=20" linewrapping.
(K)		IP address of SMTP client.
(M)	✓	6-character string compatible with Exim's message identifiers (keyed on time).
(N)		16-bit prefix of SMTP client's IP address.
(O*string*:*num*)	✓	Randomized message identifier element compatible with Microsoft SMTPSVC.
(Pnum_1[-num_2]:*string*)	✓	Random string of num_1 (up to num_2, if provided) characters taken from *string*.
(Q*string*)		Quoted-printable "=" linewrapping.
(Rnum_1-num_2)	✓	Random number between num_1 and num_2. Note, special-cased when used with (D).
(U*string*)		Randomized percent-encoding of *string*.
(V*num*)	✓	Inserts the value of the field identified by (C*num*).
(W)		Time and date as plain numbers, e.g. "20080225190434".
(X)		Previously selected member of the "names" dictionary.
(Y*num*)	✓	8-character alphanumeric string, compatible with Sendmail message identifiers.
(Z)	✓	Another Sendmail-compatible generator for message identifiers.

Table 2: Storm's spam-generation templating language.

the addresses were unique. The target addresses were heavily concentrated (60%) in the `.com` TLD, reflecting a similar concentration in harvested addresses. We also observed over 170K templates (22K unique) used by the bots to generate spam, far more than the number of campaigns observed. The high template dynamism suggests constant tuning by the spammers to continually subvert filtering.

In the remainder of this section, we discuss preliminary measurements of the bot life-cycle (the dynamics of how bots send spam), delivery efficacy (how many targets actually have mail sent to them), the dictionaries used to programatically construct messages, the prevalence of different kinds of campaigns, address harvesting behavior, and the presence of test accounts employed by the botmaster.

5.1 Bot life-cycle

Worker bots begin searching for proxies (via Overnet) upon boot-up, sending a request for an update message upon successful contact with one. (The worker then attempts a TCP port 25 connectivity check to one of Google's SMTP servers. However, its subsequent behavior does not appear to change whether or not the check succeeds.) Upon receiving an update with spam instructions, the worker then attempts to send a spam to each member of target lists in slots for which the bot has templates. It targets each address only once per occurrence in the list, working through it in sequence. The only pauses in spamming occur when a worker runs out of targets, at which point spamming goes idle until the worker next requests and receives additional targets.

We analyzed 72 bot lifespans to better understand spamming rates and bot reliability. On average, it takes our worker bots just over 4 min after boot-up until receiving the first update message. We have found that workers are short-lived: they generally fall silent (on both Overnet as well as C&C) within 24h of start-up, and on average remain functional only for a little under 4 hr. A reboot spurs them to resume their activity. This observation can explain the tendency of spam sent by the Botnet to peak in the mid-morning hours of the local timezone, shortly after infected machines are powered on [8]. The overall average spamming rate was 152 messages per minute, per bot.

5.2 Delivery efficacy

We analyzed a total of 30,186 delivery reports that our proxy bots received from remote workers. This allowed us to form an estimate of how successfully Storm delivers spam. *Delivery succeeds for only 1/6th of the target addresses provided.* Understanding the causes of failure required investigation of the error status codes reported by the bots. By combining reverse-engineering of the binary with failure-introducing configurations of our internal setup (DNS failures, SMTP server refusal, etc.) at different stages of the spam-delivery process, we could observe the codes returned by our local workers when encountering these difficulties.

We find that DNS lookup failures account for about 10% of reported failures; 8% of delivery attempts fail to establish a TCP connection to the SMTP server, 28%

```
Received: from %^C0%^P%^R2-6^%:qwertyuiopasdfghjklzxcvbnm^%.%^P%^R2-6^%:qwertyuiopasdfghjkl ▷
                                        zxcvbnm^%^% ([%^C6%^I^%.%^I^%.%^I^%.%^I^%^%]) by ▷
                                        %^A^% with Microsoft SMTPSVC(%^Fsvcver^%); %^D^%
Message-ID: <%^O%^V6^%:%^R3-50^%^%%^V0^%>
From: <%^Fnames^%@%^Fdomains^%>
To: <%^0^%>
Subject: JOB $1800/WEEK - CANADIANS WANTED!
Date: %^D-%^R30-600^%^%
```

```
Received: from auz.xwzww ([132.233.197.74]) by dsl-189-188-79-63.prod-infinitum.com.mx with ▷
                        Microsoft SMTPSVC(5.0.2195.6713); Wed, 6 Feb 2008 16:33:44 -0800
Message-ID: <002e01c86921$18919350$4ac5e984@auz.xwzww>
From: <katiera@experimentalist.org>
To: <voelker@cs.ucsd.edu>
Subject: JOB $1800/WEEK - CANADIANS WANTED!
Date: Wed, 6 Feb 2008 16:33:44 -0800
```

Figure 2: Snippet of a spam template, showing the transformation of an email header from template (top) to resulting content (bottom). The ▷-symbol indicates line continuations. Bold text corresponds to the formatting macros and their evaluation.

NAME	TOTAL	REDUNDANCY (%)	AVERAGE SIZE	SAMPLE
linksh	67,808	86.13	99.96	76.119.95.66
pharma_links	67,771	92.11	9.03	`http://iygom.tryyoung.cn/?625112501432`
domains	39,936	96.94	1040.94	013.net
names	39,695	96.72	875.54	steven88
mynamesl	6,880	99.93	98.97	Ada
trunver	6,876	99.99	4.00	2.0.0.6 (Windows/20070728)
svcver	6,871	99.97	10.00	6.0.3790.0
ronsubj	6,813	99.96	6.00	Best job for you
eximver	6,706	99.99	19.00	4.04
mynames	6,700	95.60	616.02	Abel

Table 3: Properties of the 10 most frequent dictionaries found in the bot-based dataset.

fail because the SMTP session does not begin with the expected 220 banner, 1% fail due to errors in the `HELO` stage, 18% at the `RCPT TO` stage, 14% at the `MAIL FROM` stage, and 4% at the `DATA` stage.

5.3 Dictionaries

Storm's spamming operation relies heavily on the use of F-macros to generate polymorphic messages and to create URLs for the recipients to click on. In the bot-based dataset, we observed a total of 356,279 dictionaries, clustered under 33 different names. Dictionaries repeat heavily, with only 2% being unique across the overall set of dictionaries bearing the same name.

The group with the least redundancy is "linksh" (14% unique), whose purpose is Storm's *self-preservation*: the spam content includes the raw IP addresses of proxy bots that serve up web content designed to trick the email recipient into downloading and executing a Storm executable.

Table 3 summarizes the main aspects of the dictionaries we observed.

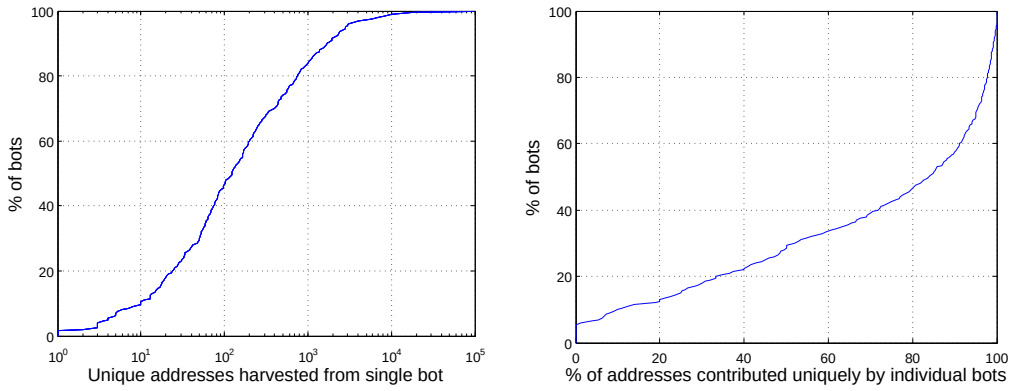
5.4 Campaign Topics

Many of the campaigns are identifiable by the templates the use. For example, self-propagation spam uses the `wormsubj` dataset for the subject line, while pharmaceutical spam the `pharma` dataset. By identifying certain template keywords, we were able to classify many of the 16,977 templates retrieved by the crawler. Unfortunately, many were image-based, which could not be classified easily. Table 4 shows the topic breakdown for templates collected during the measurement period (November 20, 2007 to February 13, 2008).

5.5 Address harvesting

Our local proxies received a total of 272,546 harvest reports from 522 remote workers. Only 10% of these contained any addresses.

The reports reflected a total harvest of 929,976 email addresses, of which 463,580 were unique. Figure 3(a) shows the distribution of overall harvest size per bot, with duplicates removed. Workers on average reported nearly a thousand distinct addresses, with the top harvest—seemingly a regular home customer with an Indian ISP—contributing 96,053 addresses. In addition,



(a) (b)

Figure 3: Harvest properties. (a) Distribution of the number of unique addresses reported by each bot. (b) Distribution of the percentage of addresses uniquely reported by the bots

TOPIC	FRACTION
Self-propagation	22%
Pharmaceutical	22%
Stocks	11%
Job offer	1%
Phishing	1%
Unknown image	39%
Unknown other	4%

Table 4: Campaign topic breakdown for the 16,977 templates retrieved by the crawler during the collection period (November 20, 2007 to February 13, 2008).

as seen in Figure 3(b), addresses reported by individual workers are often unique across the entire harvest (that is, had that worker not reported, those addresses would not have otherwise been reaped). For example, for 50% of the workers, 84% or more of their reported addresses were not otherwise harvested.

Unsurprisingly, the most frequently harvested domains all correspond to major email services: `hotmail.com`, `yahoo.com`, `aol.com`, `mail.ru`, `gmail.com`, `mynet.com`, `msn.com`, `rediffmail.com`, etc. These for the most part closely match the most prevalent domains in email address lists, as well.

However, about 10% of the harvested addresses do not correspond to a valid top-level domain. Frequent errors include `.gbl`, `.jpg`, `.msn`, `.hitbox`, `.yahoo`, `.com0`, `.dll`; clearly, some of these reflect inadequate pattern-matching when scouring the local filesystem, or files that contain slightly mangled addresses. Interestingly, unlike for correctly harvested addresses, the prevalent patterns for error here differ to a greater degree from the errors seen in address lists used during campaigns, suggesting that some additional pruning/filtering is likely applied to harvested addresses.

We also note the possibility of seeding machines with *honeytoken* addresses encoded to uniquely identify the machines on which they are planted. Unlike harvesting errors, these addresses could look completely legitimate, and in fact would be functional; any subsequent use of them would would then strongly indicate that the machine encoded within the address had been compromised and its file system scoured for email harvesting.

5.6 Botmaster test accounts

(Note: we have left details vague in this section because the specifics have implications for current law enforcement activity.)

In our analysis we uncovered the presence of a template slot which sent email to a set of just 4 email addresses. Upon inspecting the message template, we discovered that a modified version of the spam body appeared a few hours later, again sent to the same set of addresses. All four addresses have a similar structure and correspond to large email providers. We speculate that the botmaster uses these addresses to test the degree to which the structure of their new campaign is vulnerable to detection by the spam filters that large sites employ. This possibility suggests an opportunity for a form of counter-intelligence (similar in spirit to that employed in [18]): perhaps at the vantage point of such a provider we can detect the pattern of an initial message sent to a small number of addresses, and flagged as spam, followed by a subsequent similar message sent more broadly. Such an approach might be able to both detect the onset of a new spam campaign and help unmask the spammer via their access to the test account.

6 Estimating Campaign Size

In this section we attempt to estimate the total size of a campaign mailing list based on a small sample of the

overall botnet activity. Specifically, we estimate the total mailing list size used in one of Storm's self-propagation campaigns. We use an estimation method called *Mark and Recapture*, which is widely employed to estimate the sizes of animal populations. As a check on the above, we also compute an estimate based on sample intersection with a list of addresses known to have received Storm spam.

Both estimation methods are grounded in a set of statistical assumptions about the underlying sampling process, namely that:

1. the address list does not change,
2. addresses are sampled with equal probability, and
3. addresses are sampled independently.

We emphasize that none of these assumptions are believed to hold absolutely: it is reasonable to expect, for example, that address harvesting (Section 5.5) is used to augment the distribution list, and that delivery reports (Section 5.2) are used to prune the list. We have also seen addresses that differ only in their capitalization, suggesting that the list may contain duplicates. Nevertheless, we believe our techniques provide reasonable first-order estimates of the total list size.

6.1 The `wormsubj` Campaign

One of the longest-running campaigns in our crawl-based dataset is identified by a Subject header of the form "`Subject: %^Fwormsubj^%`." It is a self-propagation campaign: the body of the message contains a URL for downloading the Storm executable. Our crawler downloaded this template 3,777 times, with an observed total distribution list of 1,797,458 addresses. Because the subject line is drawn from a known set of strings (the contents of the "wormsubj" dictionary), it is also possible to identify messages generated by this template. We use this feature of the campaign to identify such messages in our spam trap (which was also used in the Spamscatter study [1]).

6.2 Mark and Recapture

Mark and Recapture is a widely used technique for estimating animal populations. In its simplest form, it involves capturing and tagging a small sample of an animal population, and counting the number of tagged specimens in another sample taken some time later. An estimate of the total population is then given by $C_0 C_1 / R$, where C_0 is the number initially captured and tagged, C_1 is the number captured in the second sample, and R is the number of animals in the second sample with tags. This is the so-called Lincoln-Petersen estimator.

For our analysis, we use the Bayesian multiple-capture model described by Gazey and Staley [7]. In this setting, email addresses are the individual animals, the complete mailing list is the population, and each batch of 900–1,000 addresses retrieved with a template is a sample of the population. We use two estimators: the standard estimator described in [7], which assumes that all addresses are chosen independently, and a modified estimator that only requires that the *first* addresses of the batch included with each template be independent. We refer to them as the aggressive and conservative estimators, respectively. Figure 4 shows the estimate CDF for both. The aggressive estimator estimate is 437 million and the conservative estimator estimate is 376 million, with the 95% confidence interval for the conservative estimate between 206 million and 790 million addresses.

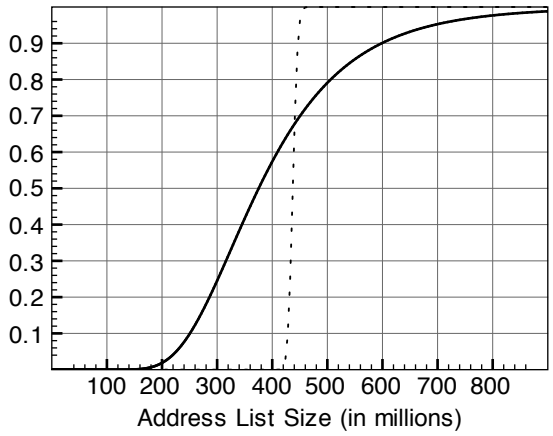


Figure 4: Cumulative distribution function for the `wormsubj` campaign total mailing list size: aggressive (dotted) and conservative (solid) estimators.

6.3 Sample Coverage

To compute the second estimate, we monitored an email domain for Storm spam and used the proportion of the domain covered by our sample to estimate the fraction of the complete list included in our sample. In other words, if D email addresses in a given domain receive `wormsubj` spam, and R of those occur in our sample of size C, then we estimate the entire list to consist of DC/R addresses. In fact, this method is just the two-capture case of the Mark and Recapture method, where the monitored domain is considered the first (tagged) sample, and the set of addresses we receive from Storm the second sample.

Over the campaign time period (late 2007 to early 2008), a total of $D = 3015$ distinct addresses received `wormsubj`. Of those, only $R = 8$ occurred in the list of $C = 1,797,458$ addresses retrieved by our crawler for this campaign. This gives an estimate of the total list size of approximately 677 million addresses, which is just within the 95% confidence interval of the conservative estimator. The 95% confidence interval for the Sample Coverage estimator itself is 410 million to 1,880 million.

Because this estimator is unreliable when R is small, we consider it a "sanity check" for the Mark and Recapture estimate.

7 Conclusion

Understanding how the enemy thinks is a requirement for anticipating their next move; doubtless there's an apt Sun Tzu quote describing this. Thus, we are ever motivated to improve our understanding of the spammer's approach and methods: *spamcraft*, to hijack a Cold War term. However, a key limitation in this goal is how well we can measure the spammer's activities. While spam is a near-universal problem, each spam victim can only bear witness to the teeniest sliver of the larger picture. Thus, outside of a few large anti-spam companies, researchers have lacked any means to obtain a sufficiently large and distributed view of spam activity from which to form representative conclusions. Moreover, since such activity is heterogeneous by its nature (consisting of all received spams), it is difficult to begin to isolate the activity of a particular spam crew, or to extract the dynamics, targeting, and evolution of a single spam campaign during its lifetime. Finally, spammers rely on intelligence from their delivery infrastructure—including target liveness and the harvesting of new addresses—that is inherently invisible to a spam recipient.

Even though these problems might each be accommodated to varying degrees, the opportunities of instead measuring spam at its source are strong. An ideal solution would be to infiltrate spam organizations themselves, observe the address lists they construct, the commands they deliver, and the feedback they receive. However, as few in the research community have a strong aptitude (and legal standing) for covert operations, we believe the next best opportunity is presented by the networks of bots used to distribute the vast majority of today's spam—*distribution infiltration*. In this setting, reverse-engineering can suffice to infiltrate the distribution infrastructure and "drink from the spam firehose" at will. In this paper, we have demonstrated this approach in the context of the Storm network, extracting millions of samples pre-sorted by campaign, and observing the same intelligence that the botnet delivers to the spammers themselves. In so doing, we have documented a number of interesting aspects of such campaigns, including automated error reporting, address harvesting, a sophisticated template language, and dedicated test accounts. We have further constructed a well-grounded estimate of the size of one of the address lists in the hundreds of millions. While both our data and our experiences are preliminary, we believe the underlying technique is sound, and offers an unprecedented level of detail about spammer activities—virtually impossible to otherwise obtain using traditional methods.

8 Acknowledgments

Our thanks to Joe Stewart of Secureworks for offering his insight into the workings of Storm, Erin Kenneally for advising us on legal issues, and to Gabriel Lawrence and Jim Madden for supporting this activity on UCSD's systems and networks. Our data collection was made possible by generous support and gifts from ESnet, the Lawrence Berkeley National Laboratory, Cisco, Microsoft Research, VMware, HP, Intel and UCSD's Center for Networked Systems, for which we are very grateful. This work was made possible by the National Science Foundation grants NSF-0433702 and NSF-0433668. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors or originators and do not necessarily reflect the views of the National Science Foundation.

Notes

1. The binary used for most of experiments was a `withlove.exe` version with MD5 hash `cbb4dedbfa7b57bb1e47063467c14e4f`.

2. Spam does get sent even when no dictionary content is available for a particular name. The output that would normally result is simply omitted.

References

[1] D. S. Anderson, C. Fleizach, S. Savage, and G. M. Voelker. Spamscatter: Characterizing Internet Scam Hosting Infrastructure. In *Proceedings of the USENIX Security Symposium*, Boston, MA, Aug. 2007.

[2] P. Barford and V. Yegneswaran. An Inside Look at Botnets. In *Special Workshop on Malware Detection, Advances in Information Security, Springer Verlag*, 2006.

[3] V. Bulletin. Storm e-card malware keeps on coming. `http://www.virusbtn.com/news/2007/08_17.xml`, August 2007.

[4] D. Dagon, C. Zou, and W. Lee. Modeling Botnet Propagation Using Time Zones. In *Proceedings of NDSS 2006*, February 2006.

[5] J. Franklin, V. Paxson, A. Perrig, and S. Savage. An Inquiry into the Nature and Causes of the Wealth of Internet Miscreants. In *ACM Computer and Communications Security*, Washington, D.C., Nov. 2007.

[6] F. C. Freiling, T. Holz, and G. Wicherski. Botnet Tracking: Exploring a Root-Cause Methodology to Prevent Distributed Denial-of-Service Attacks. In *Proceedings of ESORICS 2005*, pages 319–335, September 2005.

[7] W. J. Gazey and M. J. Staley. Population estimation from mark-recapture experiments using a sequential Bayes algorithm. *Ecology*, 67(4):841–951, Aug 1986.

[8] Government Technology. Storm spam peaks at 10 each morning. `http://www.govtech.com/gt/260128?topic=117671`, January 2008.

[9] J. Grizzard, V. Sharma, C. Nunnery, B. Kang, and D. Dagon. Peer-to-Peer Botnets: Overview and Case Study. In

Proceedings of the First USENIX Workshop on Hot Topics in Understanding Botnets, April 2007.

[10] Ironport. 2008 Internet Security Trends. http://www.ironport.com/securitytrends/, 2008.

[11] Paul Bäher, Thorsten Holz, Markus Kötter and Georg Wicherski. Know your Enemy: Tracking Botnets. In *The Honeynet Project & Research Alliance*, March 2005.

[12] P. Porras, H. Saïdi, and V. Yegneswaran. A Multi-perspective Analysis of the Storm (Peacomm) Worm. Technical report, Computer Science Laboratory, SRI International, October 2007.

[13] M. Prince, B. Dahl, L. Holloway, A. Keller, and E. Langheinrich. Understanding How Spammers Steal Your E-Mail Address: An Analysis of the First Six Months of Data from Project Honey Pot. In *Second Conference on Email and Anti-Spam (CEAS 2005)*, 2005.

[14] C. Pu and S. Webb. Observed Trends in Spam Construction Techniques: A Case Study of Spam Evolution. In *Third Conference on Email and Anti-Spam (CEAS 2006)*, 2006.

[15] M. Rajab, J. Zarfoss, F. Monrose, and A. Terzis. My Botnet is Bigger than Yours (Maybe, Better than Yours). In *Proceedings of the First USENIX Workshop on Hot Topics in Understanding Botnets*, April 2007.

[16] M. A. Rajab, J. Zarfoss, F. Monrose, and A. Terzis. A Multifaceted Approach to Understanding the Botnet Phenomenon. In *Proceedings of the ACM Internet Measurement Conference*, Rio de Janeiro, Brazil, Oct. 2006.

[17] A. Ramachandran and N. Feamster. Understanding the Network-Level Behavior of Spammers. In *Proceedings of the ACM SIGCOMM Conference*, Pisa, Italy, Sept. 2006.

[18] A. Ramachandran, N. Feamster, and D. Dagon. Revealing botnet membership using DNSBL counter-intelligence. In *USENIX 2nd Workshop on Steps to Reducing Unwanted Traffic on the Internet (SRUTI '06)*, July 2006.

[19] S. Sarat and A. Terzis. Measuring the Storm Worm Network. Technical Report 01-10-2007, HiNRG Johns Hopkins University, 2007.

[20] J. Stewart. Storm worm DDoS attack. http://www.secureworks.com/research/threats/storm-worm/?threat=storm-worm, February 2007.

[21] J. Zhuge, T. Holz, X. Han, J. Guo, and W. Zou. Characterizing the IRC-based Botnet Phenomenon. Technical Report TR-2007-010, University of Mannheim, December 2007.