Spamalytics: An Empirical Analysis of Spam Marketing Conversion

Chris Kanich* Christian Kreibich† 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

ABSTRACT

The "conversion rate" of spam — the probability that an unsolicited e-mail will ultimately elicit a "sale" — underlies the entire spam value proposition. However, our understanding of this critical behavior is quite limited, and the literature lacks any quantitative study concerning its true value. In this paper we present a methodology for measuring the conversion rate of spam. Using a parasitic infiltration of an existing botnet's infrastructure, we analyze two spam campaigns: one designed to propagate a malware Trojan, the other marketing on-line pharmaceuticals. For nearly a half billion spam e-mails we identify the number that are successfully delivered, the number that pass through popular anti-spam filters, the number that elicit user visits to the advertised sites, and the number of "sales" and "infections" produced.

Categories and Subject Descriptors

K.4.1 [**Public Policy Issues**]: ABUSE AND CRIME INVOLVING COMPUTERS

General Terms

Measurement, Security, Economics

Keywords

Spam, Unsolicited Email, Conversion

1. INTRODUCTION

Spam-based marketing is a curious beast. We all receive the advertisements — "Excellent hardness is easy!" — but few of us have encountered a person who admits to following through on this offer and making a purchase. And yet, the relentlessness by which such spam continually clogs Internet inboxes, despite years of energetic deployment of anti-spam technology, provides undeniable testament that spammers find their campaigns profitable. Someone is clearly buying. But how many, how often, and how much?

Unraveling such questions is *essential* for understanding the economic support for spam and hence where any structural weaknesses may lie. Unfortunately, spammers do not file quarterly financial reports, and the underground nature of their activities makes third-party data gathering a challenge at best. Absent an empirical foundation, defenders are often left to speculate as to how successful spam campaigns are and to what degree they are profitable. For example, IBM's Joshua Corman was widely quoted as claiming that spam sent by the Storm worm alone was generating "millions and millions of dollars every day" [2]. While this claim could in fact be true, we are unaware of any public data or methodology capable of confirming or refuting it.

The key problem is our limited visibility into the three basic parameters of the spam value proposition: the cost to send spam, offset by the "conversion rate" (probability that an e-mail sent will ultimately yield a "sale"), and the marginal profit per sale. The first and last of these are self-contained and can at least be estimated based on the costs charged by third-party spam senders and through the pricing and gross margins offered by various Internet marketing "affiliate programs".[1] However, the conversion rate depends fundamentally on group actions — on what hundreds of millions of Internet users do when confronted with a new piece of spam — and is much harder to obtain. While a range of anecdotal numbers exist, we are unaware of any well-documented measurement of the spam conversion rate.[2]

In part, this problem is methodological. There are no apparent methods for indirectly measuring spam conversion. Thus, the only obvious way to extract this data is to build an e-commerce site, market it via spam, and then record the number of sales. Moreover, to capture the spammer's experience with full fidelity, such a study must also mimic their use of illicit botnets for distributing e-mail and proxying user responses. In effect, the best way to measure spam is to be a spammer.

In this paper, we have effectively conducted this study, though *sidestepping* the obvious legal and ethical problems associated with sending spam.[3] Critically, our study makes use of an *existing* spam-

CCS'08, October 27–31, 2008, Alexandria, Virginia, USA.
Copyright 2008 ACM 978-1-59593-810-7/08/10 ...$5.00.

[1] Our cursory investigations suggest that commissions on pharmaceutical affiliate programs tend to hover around 40-50%, while the *retail* cost for spam delivery has been estimated at under $80 per million [22].

[2] The best known among these anecdotal figures comes from the Wall Street Journal's 2003 investigation of Howard Carmack (a.k.a the "Buffalo Spammer"), revealing that he obtained a 0.00036 conversion rate on ten million messages marketing an herbal stimulant [4].

[3] We conducted our study under the ethical criteria of ensuring *neutral actions* so that users should never be worse off due to our ac-

ming botnet. By infiltrating its command and control infrastructure parasitically, we convinced it to modify a subset of the spam it *already* sends, thereby directing any interested recipients to servers under our control, rather than those belonging to the spammer. In turn, our servers presented Web sites mimicking those actually hosted by the spammer, but "defanged" to remove functionality that would compromise the victim's system or receive sensitive personal information such as name, address or credit card information.

Using this methodology, we have documented three spam campaigns comprising over 469 million e-mails. We identified how much of this spam is successfully delivered, how much is filtered by popular anti-spam solutions, and, most importantly, how many users "click-through" to the site being advertised (*response rate*) and how many of those progress to a "sale" or "infection" (*conversion rate*).

The remainder of this paper is structured as follows. Section 2 describes the economic basis for spam and reviews prior research in this area. Section 3 describes the Storm botnet, and Section 4 describes our experimental methodology using Storm. Section 5 describes our spam filtering and conversion results, Section 6 analyzes the effects of blacklisting on spam delivery, and Section 7 analyzes the possible influences on spam responses. We synthesize our findings in Section 8 and conclude.

2. BACKGROUND

Direct marketing has a rich history, dating back to the 19th century distribution of the first mail-order catalogs. What makes direct marketing so appealing is that one can directly measure its return on investment. For example, the Direct Mail Association reports that direct mail sales campaigns produce a response rate of 2.15 percent on average [5]. Meanwhile, rough estimates of direct mail *cost per mille* (CPM) – the cost to address, produce and deliver materials to a thousand targets – range between $250 and $1000. Thus, following these estimates it might cost $250,000 to send out a million solicitations, which might then produce 21,500 responses. The cost of developing these prospects (roughly $12 each) can be directly computed and, assuming each prospect completes a sale of an average value, one can balance this revenue directly against the marketing costs to determine the profitability of the campaign. As long as the product of the conversion rate and the marginal profit per sale exceeds the marginal delivery cost, the campaign is profitable.

Given this underlying value proposition, it is not at all surprising that bulk direct e-mail marketing emerged very quickly after e-mail itself. The marginal cost to send an e-mail is tiny and, thus, an e-mail-based campaign can be profitable even when the conversion rate is negligible. Unfortunately, a perverse byproduct of this dynamic is that sending as much spam as possible is likely to maximize profit.

The resulting social nuisance begat a vibrant anti-spam community, eventually producing a multi-billion dollar industry focused on the same problem. However, with each anti-spam innovation spammers adapted in kind and, while the resulting co-evolution has not significantly changed the spam problem, it has changed how spam is purveyed. For example, the advent of real-time IP blacklisting deployed in Mail Transfer Agents (MTAs) forced spammers to relay their messages through "untainted" third-party hosts — driving the creation of modern large-scale botnets. Similarly, content-based anti-spam filters in turn forced spammers to create sophisticated polymorphism engines, modifying each spam message to be distinct. As well, it forced them to send even more spam. Thus, it has been estimated that over 120 billion spam messages are now sent each day [11].

However, while spam has long been understood to be an economic problem, it is only recently that there has been significant effort in modeling spam economics and understanding the value proposition from the spammer's point of view. Rarely do spammers talk about financial aspects of their activities themselves, though such accounts do exist [14, 21]. Judge et al. describe a prototypical model of spam profitability, including both the basic value proposition as well as the impact of anti-spam filtering and law enforcement. They speculate that response rates as low as 0.000001 are sufficient to maintain profitability [17]. Khong [13] likewise employs an economic cost model of spam, comparing the success of several anti-spam strategies. Goodman and Rounthwaite construct a more complex model, aimed at deriving the cost factors for sending spam, and conclude depressingly that the optimal strategy for sending spam is to send as fast as possible [9]. Serjantov and Clayton explore these issues from the standpoint of an ISP and try to understand how to place appropriate incentives around the use of anti-spam blacklists [19].

However, the work that is most closely related to our own are the several papers concerning "Stock Spam" [7, 8, 10]. Stock spam refers to the practice of sending positive "touts" for a low-volume security in order to manipulate its price and thereby profit on an existing position in the stock. What distinguishes stock spam is that it is monetized through price manipulation and not via a sale. Consequently, it is not necessary to measure the conversion rate to understand profitability. Instead, profitability can be inferred by correlating stock spam message volume with changes in the trading volume and price for the associated stocks.

The work of Ma and Chen is similar to ours in that it analyzes in detail the structure of a spamming operation. However, their focus is on redirection chains employed by spammers as a search engine optimization strategy [20].

3. THE STORM BOTNET

The measurements in this paper are carried out using the Storm botnet and its spamming agents. While a complete technical description of Storm is outside the scope of this paper, we review key mechanisms in Storm's communication protocols and organizational hierarchy.

Storm is a peer-to-peer botnet that propagates via spam (usually by directing recipients to download an executable from a Web site). Storm communicates using two separate protocols: the first is an encrypted version of the UDP-based Overnet protocol (in turn based on the Kademlia DHT [16]) and is used primarily as a directory service to find other nodes. As well, Storm uses a custom TCP-based protocol for managing command and control — the directions informing each bot what actions it should take. We describe each of these below.

3.1 Overnet protocol

There are four basic messages to facilitate the basic functioning of Overnet: *Connect*, *Search*, *Publicize*, and *Publish*. During the bootstrap phase, a Storm node only has the initial list of peers that it was shipped with. To gather more peers Storm chooses a OID pseudo-randomly from the 128-bit Overnet address space and proceeds to *Connect* to all the peers in its bootstrap list. Each available peer contacted returns a list of up to 20 peers. Storm does this for a few rounds until it has gathered enough peers to be adequately connected in Overnet. Once a new node has learned about enough peers it switches to *Publicizing* its presence to nearby peers and

tivities, while strictly *reducing harm* for those situations in which user property was at risk.

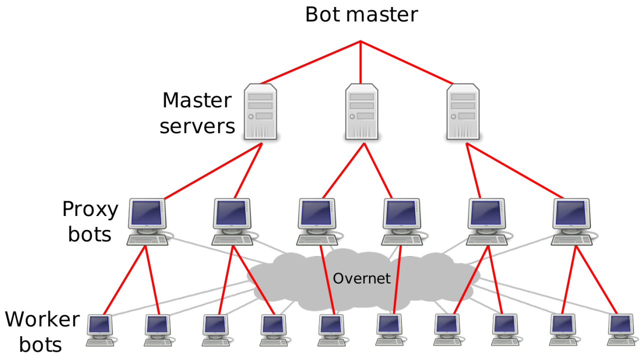


Figure 1: The Storm botnet hierarchy.

periodically searching for its own OID to stay connected and learn about new close-by peers to keep up with churn.

Overnet also provides two messages for storing and finding content in the network: *Publish* and *Search* which export a standard DHT (key,value) pair interface. However, Storm uses this interface in an unusual way. In particular, the keys encode a dynamically changing rendezvous code that allow Storm nodes to find each other on demand.

A Storm node generates and uses three rendezvous keys simultaneously: one based on the current date, one based on the previous date, and one based on the next date. To determine the correct date, Storm first sets the system clock using NTP.

In particular, each key is based on a combination of the time (with 24-hour resolution) mixed with a random integer between 0 and 31. Thus there are 32 unique Storm keys in use per day but a single Storm bot will only use 1 of the 32. Because keys are based on time, Storm uses NTP to sync a bot's clock and attempts to normalize the time zone. Even so, to make sure bots around the world can stay in sync, Storm uses 3 days of keys at once, the previous, current, and next day.

In turn, these keys are used to rendezvous with Storm nodes that implement the command and control (C&C) channel. A Storm node that wishes to offer the C&C service will use the time-based hashing algorithm to generate a key and encode its own IP address and TCP port into the value. It will then search for the appropriate peers close to the key and publish its (key, value) pair to them. A peer wishing to locate a C&C channel can generate a time-based key and search for previously published values to decode and connect to the TCP network.

3.2 Storm hierarchy

There are three primary classes of Storm nodes involved in sending spam (shown in Figure 1). Worker bots make requests for work and, upon receiving orders, send spam as requested. Proxy bots act as conduits between workers and master servers. Finally, the master servers provide commands to the workers and receive their status reports. In our experience there are a very small number of master servers (typically hosted at so-called "bullet-proof" hosting centers) and these are likely managed by the botmaster directly.

However, the distinction between worker and proxy is one that is determined automatically. When Storm first infects a host it tests if it can be reached externally. If so, then it is eligible to become a proxy. If not, then it becomes a worker.

3.3 Spam engine

Having decided to become a worker, a new bot first checks whether it can reach the SMTP server of a popular Web-based mail provider on TCP port 25. If this check fails the worker will remain active but not participate in spamming campaigns.[4]

Figure 2 outlines the broad steps for launching spam campaigns when the port check is successful. The worker finds a proxy (using the time-varying protocol described earlier) and then sends an update request (via the proxy) to an associated master server (Step 1), which will respond with a spam workload task (Step 2). A spam workload consists of three components: one or more spam templates, a delivery list of e-mail addresses, and a set of named "dictionaries". Spam templates are written in a custom macro language for generating polymorphic messages [15]. The macros insert elements from the dictionaries (e.g., target e-mail addresses, message subject lines), random identifiers (e.g., SMTP message identifiers, IP addresses), the date and time, etc., into message fields and text. Generated messages appear as if they originate from a valid MTA, and use polymorphic content for evading spam filters.

Upon receiving a spam workload, a worker bot generates a unique message for each of the addresses on the delivery list and attempts to send the message to the MX of the recipient via SMTP (Step 3). When the worker bot has exhausted its delivery list, it requests two additional spam workloads and executes them. It then sends a delivery report back to its proxy (Step 4). The report includes a result code for each attempted delivery. If an attempt was successful, it includes the full e-mail address of the recipient; otherwise, it reports an error code corresponding to the failure. The proxy, in turn, relays these status reports back to the associated master server.

To summarize, Storm uses a three-level self-organizing hierarchy comprised of worker bots, proxy bots and master servers. Command and control is "pull-based", driven by requests from individual worker bots. These requests are sent to proxies who, in turn, automatically relay these requests to master servers and similarly forward any attendant responses back to to the workers.

4. METHODOLOGY

Our measurement approach is based on *botnet infiltration* — that is, insinuating ourselves into a botnet's "command and control" (C&C) network, passively observing the spam-related commands and data it distributes and, where appropriate, actively changing individual elements of these messages in transit. Storm's architecture lends itself particularly well to infiltration since the proxy bots, *by design*, interpose on the communications between individual worker bots and the master servers who direct them. Moreover, since Storm compromises hosts indiscriminately (normally using malware distributed via social engineering Web sites) it is straightforward to create a proxy bot on demand by infecting a globally reachable host under our control with the Storm malware.

Figure 2 also illustrates our basic measurement infrastructure. At the core, we instantiate eight unmodified Storm proxy bots within a controlled virtual machine environment hosted on VMWare ESX 3 servers. The network traffic for these bots is then routed through a centralized gateway, providing a means for blocking unanticipated behaviors (e.g., participation in DDoS attacks) and an interposition point for parsing C&C messages and "rewriting" them as they pass from proxies to workers. Most critically, by carefully rewriting the spam template and dictionary entries sent by master servers, we arrange for worker bots to replace the intended site links in their spam with URLs of our choosing. From this basic capability we synthesize experiments to measure the click-through and conversion rates for several large spam campaigns.

[4] Such bots are still "useful" for other tasks such as mounting coordinated DDoS attacks that Storm perpetrates from time to time.

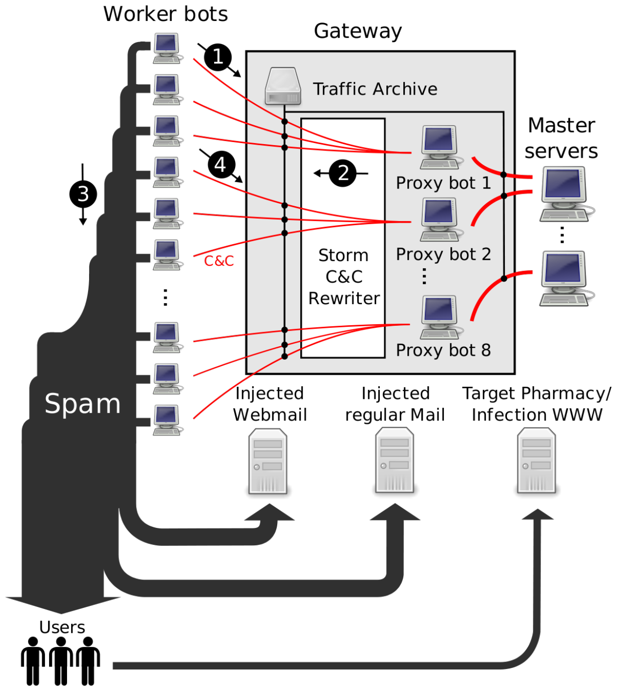


Figure 2: The Storm spam campaign dataflow (Section 3.3) and our measurement and rewriting infrastructure (Section 4). (1) Workers request spam tasks through proxies, (2) proxies forward spam workload responses from master servers, (3) workers send the spam and (4) return delivery reports. Our infrastructure infiltrates the C&C channels between workers and proxies.

In the remainder of this section we provide a detailed description of our Storm C&C rewriting engine, discuss how we use this tool to obtain empirical estimates for spam delivery, click-through and conversion rates and describe the heuristics used for differentiating real user visits from those driven by automated crawlers, honeyclients, etc. With this context, we then review the ethical basis upon which these measurements were conducted.

4.1 C&C protocol rewriting

Our runtime C&C protocol rewriter consists of two components. A custom Click-based network element redirects potential C&C traffic to a fixed IP address and port, where a user-space proxy server implemented in Python accepts incoming connections and impersonates the proxy bots. This server in turn forwards connections back into the Click element, which redirects the traffic to the intended proxy bot. To associate connections to the proxy server with those forwarded by the proxy server, the Click element injects a SOCKS-style destination header into the flows. The proxy server uses this header to forward a connection to a particular address and port, allowing the Click element to make the association. From that point on, traffic flows transparently through the proxy server where C&C traffic is parsed and rewritten as required. Rules for rewriting can be installed independently for templates, dictionaries, and e-mail address target lists. The rewriter logs all C&C traffic between worker and our proxy bots, between the proxy bots and the master servers, and all rewriting actions on the traffic.

Since C&C traffic arrives on arbitrary ports, we designed the proxy server so that it initially handles any type of connection and falls back to passive pass-through for any non-C&C traffic. Since the proxy server needs to maintain a connection for each of the (many) workers, we use a preforked, multithreaded design. A pool of 30 processes allowed us to handle the full worker load for the eight Storm proxy bots at all times.

4.2 Measuring spam delivery

To evaluate the effect of spam filtering along the e-mail delivery path to user inboxes, we established a collection of test e-mail accounts and arranged to have Storm worker bots send spam to those accounts. We created multiple accounts at three popular free e-mail providers (Gmail, Yahoo!, and Hotmail), accounts filtered through our department commercial spam filtering appliance (a Barracuda Spam Firewall Model 300 with slightly more permissive spam tagging than the default setting), and multiple SMTP "sinks" at distinct institutions that accept any message sent to them (these served as "controls" to ensure that spam e-mails were being successfully delivered, absent any receiver-side spam filtering). When worker bots request spam workloads, our rewriter appends these e-mail addresses to the end of each delivery list. When a worker bot reports success or failure back to the master servers, we remove any success reports for our e-mail addresses to hide our modifications from the botmaster.

We periodically poll each e-mail account (both inbox and "junk/spam" folders) for the messages that it received, and we log them with their timestamps. However, some of the messages we receive have nothing to do with our study and must be filtered out. These messages occur for a range of reasons, including spam generated by "dictionary bots" that exhaustively target potential e-mail addresses, or because the addresses we use are unintentionally "leaked" (this can happen when a Storm worker bot connects to our proxy and then leaves before it has finished sending its spam; when it reconnects via a new proxy the delivery report to the master servers will include our addresses). To filter such e-mail, we validate that each message includes both a subject line used by our selected campaigns and contains a link to one of the Web sites under our control.

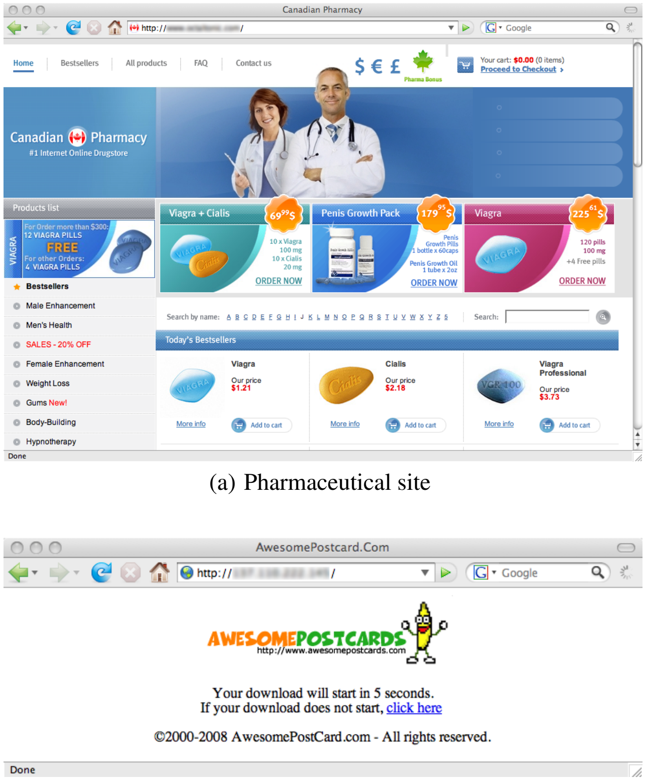
4.3 Measuring click-through and conversion

To evaluate how often users who receive spam actually visit the sites advertised requires monitoring the advertised sites themselves. Since it is generally impractical to monitor sites not under our control, we have arranged to have a fraction of Storm's spam advertise sites of our creation instead.

In particular, we have focused on two types of Storm spam campaigns, a self-propagation campaign designed to spread the Storm malware (typically under the guise of advertising an electronic postcard site) and the other advertising a pharmacy site. These are the two most popular Storm spam campaigns and represent over 40% of recent Storm activity [15].

For each of these campaigns, the Storm master servers distribute a specific "dictionary" that contains the set of target URLs to be inserted into spam e-mails as they are generated by worker bots. To divert user visits to our sites instead, the rewriter replaces any dictionaries that pass through our proxies with entries only containing URLs to our Web servers.

In general, we strive for verisimilitude with the actual Storm operation. Thus, we are careful to construct these URLs in the same manner as the real Storm sites (whether this is raw IP addresses, as used in the self-propagation campaigns, or the particular "noun-noun.com" naming schema used by the pharmacy campaign) to ensure the generated spam is qualitatively indistinguishable from the "real thing". An important exception, unique to the pharmacy campaign, is an identifier we add to the end of each URL by modi-



(a) Pharmaceutical site

Your download will start in 5 seconds.
If your download does not start, click here

©2000-2008 AwesomePostCard.com - All rights reserved.

(b) Postcard-themed self-propagation site

Figure 3: Screenshots of the Web sites operated to measure user click-through and conversion.

fying the associated spam template. This identifier allows us to unambiguously associate individual spam messages with subsequent accesses to the site. We did not add this identifier to the self-propagation campaigns since their URLs typically consist entirely of raw IP addresses. The addition of a text identifier suffix might thus appear out of place, reducing verisimilitude, and perhaps bias user click behavior.

Finally, we created two Web sites to mimic those used in the associated campaigns (screenshots of these sites are shown in Figure 3). The pharmaceutical site, primarily marketing "male-enhancement" drugs such as Viagra, is a nearly-precise replica of the site normally advertised by Storm down to using the same naming convention for the domains themselves. Our site mirrors the original site's user interface, the addition of products advertised for sale to a "shopping cart", and navigation up to, but not including, the input of personal and payment information (there are a range of complex regulatory, legal and ethical issues in accepting such information). Instead, when a user clicks on "Checkout" we return a 404 error message. We log all accesses to the site, allowing us to determine when a visitor attempts to make a purchase and what the content of their shopping cart is at the time. We assume that a purchase attempt is a conversion, which we speculate is a reasonable assumption, although our methodology does not allow us to validate that the user would have actually completed the purchase or that their credit card information would have been valid.

The self-propagation campaign is Storm's key mechanism for growth. The campaign entices users to download the Storm malware via deception; for example by telling them it is postcard software essential for viewing a message or joke sent to them by a friend. Unlike the pharmacy example, we were not able to mirror the graphical content of the postcard site, since it was itself stolen from a legitimate Internet postcard site. Instead, we created a close analog designed to mimic the overall look and feel. We also "defanged" our site by replacing its link to the Storm malware with that of a benign executable. If run, our executable is designed to performs a simple HTTP POST with a harmless payload ("`data=1`") to a server under our control, and then exit. As a rough timeout mechanism, the executable will not send the message if the system date is 2009 or later. Since the postcard site we impersonated served three different executables under different names, we served three executables with different target filenames in the POST command as well. Again, all accesses to the site are logged and we are able to identify when our binary has been downloaded. Moreover, by correlating with the POST signal, we are able to determine if a particular download is ultimately executed on the visitor's machine (and hence is a conversion). Downloads and executions can differ because the user has second thoughts about allowing an execution or because the user's security software prevents it from executing (indeed, we observed that several anti-virus vendors developed signatures for our benign executable within a few days of our introducing it).

4.4 Separating users from crawlers

As with our e-mail accounts, not all visits to our Web site are prospective conversions. There is a range of automated and semi-automated processes that visit our sites, ranging from pure Web crawlers, to "honeyclient" systems designed to gather intelligence on spam advertised sites, to security researchers trying to identify new malware.

To filter out such visits (which we generically call "crawlers") from intentful ones, we have developed a series of heuristics to identify crawlers and use this data to populate a global IP blacklist across all of our Web sites. We outline these heuristics below.

First, we consider all hosts that access the pharmacy site that do not use a URL containing the unique identifier discussed in Section 4.3 to be crawlers. Second, we blacklist hosts that access `robots.txt` (site-specific instructions meant only for Web crawlers) and hosts that make malformed requests (most often exploit attempts). Third, we blacklist all hosts that disable javascript and do not load embedded images. We assume that typical users do not browse under these conditions, whereas some large-scale anti-spam honeypots that follow embedded links in suspected spam exhibit this behavior to reduce load.

In addition to blacklisting based on the behavior of individual site visits, another common pattern we observed was the same IP address accessing the pharmacy site using several different unique identifiers, presumably as part of a spam defense or measurement mechanism. Consequently, we blacklist an IP address seen accessing the pharmacy site with more than one unique identifier with the same `User-Agent` field. This heuristic does not filter users browsing behind larger Web proxy services, but does filter the homogeneous accesses seen from spam honeyclients. Similarly, we also blacklist any host that requests the downloaded executable from the postcard site ten or more times, under the assumption that such hosts are used by researchers or other observers interested in tracking updates to the Storm malware.

Finally, it has become common for anti-malware researchers to find new versions of the Storm malware by directly accessing the self-propagation dictionary entries. To detect such users we injected new IP addresses (never advertised in spam messages) into the self-propagation dictionary during a period of inactivity (i.e., when no self-propagation spam was being sent). Any visitors to

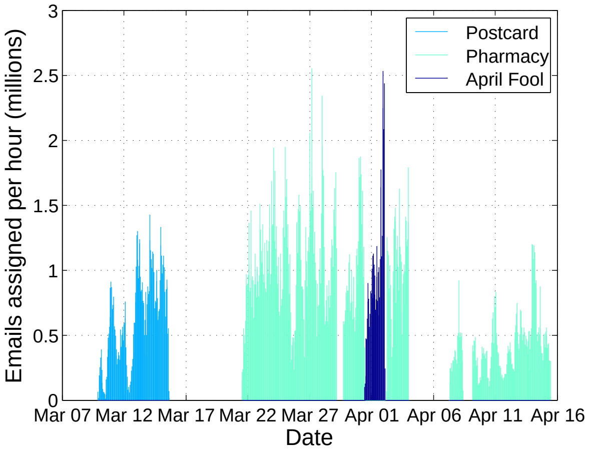


Figure 4: Number of e-mail messages assigned per hour for each campaign.

CAMPAIGN	DATES	WORKERS	E-MAILS
Pharmacy	Mar 21 – Apr 15	31,348	347,590,389
Postcard	Mar 9 – Mar 15	17,639	83,665,479
April Fool	Mar 31 – Apr 2	3,678	38,651,124
		Total	469,906,992

Table 1: Campaigns used in the experiment.

these IP addresses could not have resulted from spam, and we therefore also added them to our crawler blacklist.

It is still possible that some of the accesses were via full-featured, low-volume honeyclients, but even if these exist we believe they are unlikely to significantly impact the data.

4.5 Measurement ethics

We have been careful to design experiments that we believe are both consistent with current U.S. legal doctrine and are fundamentally ethical as well. While it is beyond the scope of this paper to fully describe the complex legal landscape in which active security measurements operate, we believe the ethical basis for our work is far easier to explain: *we strictly reduce harm*. First, our instrumented proxy bots do not create any new harm. That is, absent our involvement, the same set of users would receive the same set of spam e-mails sent by the same worker bots. Storm is a large self-organizing system and when a proxy fails its worker bots automatically switch to other idle proxies (indeed, when our proxies fail we see workers quickly switch away). Second, our proxies are passive actors and do not themselves engage in any behavior that is intrinsically objectionable; they do not send spam e-mail, they do not compromise hosts, nor do they even contact worker bots asynchronously. Indeed, their only function is to provide a conduit between worker bots making requests and master servers providing responses. Finally, where we do modify C&C messages in transit, these actions themselves strictly reduce harm. Users who click on spam altered by these changes will be directed to one of our innocuous doppelganger Web sites. Unlike the sites *normally* advertised by Storm, our sites do not infect users with malware and do not collect user credit card information. Thus, no user should receive more spam due to our involvement, but some users will receive spam that is less dangerous that it would otherwise be.

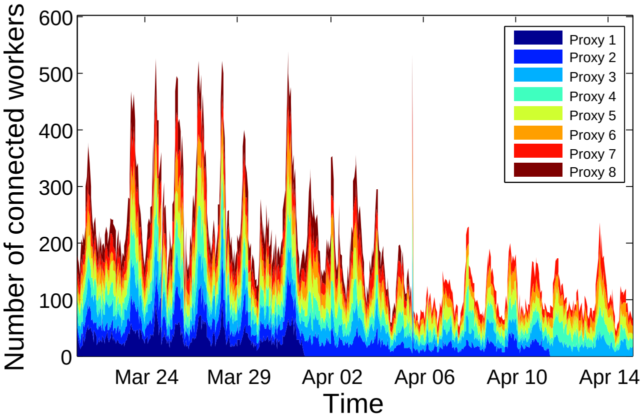


Figure 5: Timeline of proxy bot workload.

DOMAIN	FREQ.
hotmail.com	8.47%
yahoo.com	5.05%
gmail.com	3.17%
aol.com	2.37%
yahoo.co.in	1.13%
sbcglobal.net	0.93%
mail.ru	0.86%
shaw.ca	0.61%
wanadoo.fr	0.61%
msn.com	0.58%
Total	23.79%

Table 2: The 10 most-targeted e-mail address domains and their frequency in the combined lists of targeted addresses over all three campaigns.

5. EXPERIMENTAL RESULTS

We now present the overall results of our rewriting experiment. We first describe the spam workload observed by our C&C rewriting proxy. We then characterize the effects of filtering on the spam workload along the delivery path from worker bots to user inboxes, as well as the number of users who browse the advertised Web sites and act on the content there.

5.1 Campaign datasets

Our study covers three spam campaigns summarized in Table 1. The "Pharmacy" campaign is a 26-day sample (19 active days) of an on-going Storm campaign advertising an on-line pharmacy. The "Postcard" and "April Fool" campaigns are two distinct and serial instances of self-propagation campaigns, which attempt to install an executable on the user's machine under the guise of being postcard software. For each campaign, Figure 4 shows the number of messages per hour assigned to bots for mailing.

Storm's authors have shown great cunning in exploiting the cultural and social expectations of users — hence the April Fool campaign was rolled out for a limited run around April 1st. Our Web site was designed to mimic the earlier Postcard campaign and thus our data probably does not perfectly reflect user behavior for this campaign, but the two are similar enough in nature that we surmise that any impact is small.

We began the experiment with 8 proxy bots, of which 7 survived until the end. One proxy crashed late on March 31. The total number of worker bots connected to our proxies was 75,869.

Figure 5 shows a timeline of the proxy bot workload. The number of workers connected to each proxy is roughly uniform across

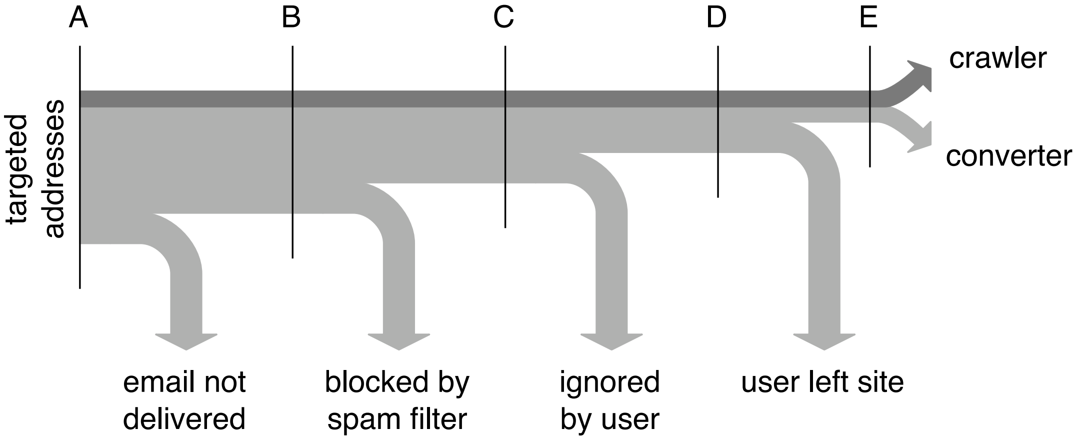


Figure 6: The spam conversion pipeline.

Stage	Pharmacy		Postcard		April Fool	
A – Spam Targets	347,590,389	100%	83,655,479	100%	40,135,487	100%
B – MTA Delivery (est.)	82,700,000	23.8%	21,100,000	25.2%	10,100,000	25.2%
C – Inbox Delivery	—	—	—	—	—	—
D – User Site Visits	10,522	0.00303%	3,827	0.00457%	2,721	0.00680%
E – User Conversions	28	0.0000081%	316	0.000378%	225	0.000561%

Table 3: Filtering at each stage of the spam conversion pipeline for the self-propagation and pharmacy campaigns. Percentages refer to the conversion rate relative to Stage A.

all proxies (23 worker bots on average), but shows strong spikes corresponding to new self-propagation campaigns. At peak, 539 worker bots were connected to our proxies at the same time.

Most workers only connected to our proxies once: 78% of the workers only connected to our proxies a single time, 92% at most twice, and 99% at most five times. The most prolific worker IP address, a host in an academic network in North Carolina, USA, contacted our proxies 269 times; further inspection identified this as a NAT egress point for 19 individual infections. Conversely, most workers do not connect to more than one proxy: 81% of the workers only connected to a single proxy, 12% to two, 3% to four, 4% connected to five or more, and 90 worker bots connected to all of our proxies. On average, worker bots remained connected for 40 minutes, although over 40% workers connected for less than a minute. The longest connection lasted almost 81 hours.

The workers were instructed to send postcard spam to a total of 83,665,479 addresses, of which 74,901,820 (89.53%) are unique. The April Fool campaign targeted 38,651,124 addresses, of which 36,909,792 (95.49%) are unique. Pharmacy spam targeted 347,590,389 addresses, of which 213,761,147 (61.50%) are unique. Table 2 shows the 15 most frequently targeted domains of the three campaigns. The individual campaign distributions are identical in ordering and to a precision of one tenth of a percentage, therefore we only show the aggregate breakdown.

5.2 Spam conversion pipeline

Conceptually, we break down spam conversion into a pipeline with five "filtering" stages in a manner similar to that described by Aycock and Friess [6]. Figure 6 illustrates this pipeline and shows the type of filtering at each stage. The pipeline starts with delivery lists of target e-mail addresses sent to worker bots (Stage A). For a wide range of reasons (e.g., the target address is invalid, MTAs refuse delivery because of blacklists, etc.), workers will successfully deliver only a subset of their messages to an MTA (Stage B). At this point, spam filters at the site correctly identify many messages as spam, and drop them or place them aside in a spam folder. The remaining messages have survived the gauntlet and appear in a user's inbox as valid messages (Stage C). Users may delete or otherwise ignore them, but some users will act on the spam, click on the URL in the message, and visit the advertised site (Stage D). These users may browse the site, but only a fraction "convert" on the spam (Stage E) by attempting to purchase products (pharmacy) or by downloading and running an executable (self-propagation).

Spam Filter	Pharmacy	Postcard	April Fool
Gmail	0.00683%	0.00176%	0.00226%
Yahoo	0.00173%	0.000542%	none
Hotmail	none	none	none
Barracuda	0.131%	N/A	0.00826%

Table 4: Number of messages delivered to a user's inbox as a fraction of those injected for test accounts at free e-mail providers and a commercial spam filtering appliance. The test account for the Barracuda appliance was not included in the Postcard campaign.

We show the spam flow in two parts, "crawler" and "converter", to differentiate between real and masquerading users (Section 4.4). For example, the delivery lists given to workers contain honeypot e-mail addresses. Workers deliver spam to these honeypots, which then use crawlers to access the sites referenced by the URL in the messages (e.g., our own Spamscatter project [3]). Since we want to measure the spam conversion rate for actual users, we separate out the effects of automated processes like crawlers — a necessary aspect of studying an artifact that is also being actively studied by other groups [12].

Table 3 shows the effects of filtering at each stage of the conversion pipeline for both the self-propagation and pharmaceutical campaigns. The number of targeted addresses (A) is simply the total number of addresses on the delivery lists received by the worker bots during the measurement period, excluding the test addresses we injected.

We obtain the number of messages delivered to an MTA (B) by relying on delivery reports generated by the workers. Unfortunately, an exact count of successfully delivered messages is not possible because workers frequently change proxies or go offline, causing both extraneous (resulting from a previous, non-interposed proxy session) and missing delivery reports. We can, however, estimate the aggregate delivery ratio (B/A) for each campaign using the success ratio of all observed delivery reports. This ratio allows us to then estimate the number of messages delivered to the MTA and even to do so on a per-domain basis.

The number of messages delivered to a user's inbox (C) is a much harder value to estimate. We do not know what spam filtering, if any, is used by each mail provider, and then by each user individually, and therefore cannot reasonably estimate this number in total. It is possible, however, to determine this number for individual mail providers or spam filters. The three mail providers and the spam filtering appliance we used in this experiment had a method for separating delivered mails into "junk" and inbox categories. Table 4 gives the number of messages delivered a user's inbox for the free e-mail providers, which together accounted for about 16.5% of addresses targeted by Storm (Table 2), as well as our department's commercial spam filtering appliance. It is important to note that these are results from one spam campaign over a short period of time and should not be used as measures of the relative effectiveness for each service. That said, we observe that the popular Web mail providers all do a very a good job at filtering the campaigns we observed, although it is clear they use different methods to get there (for example, Hotmail rejects most Storm spam at the MTA-level, while Gmail accepts a significant fraction only to filter it later as junk).

The number of visits (D) is the number of accesses to our emulated pharmacy and postcard sites, excluding any crawlers as determined using the methods outlined in Section 4.2. We note that crawler requests came from a small fraction of hosts but accounted for the majority of all requests to our Web sites. For the pharmacy site, for instance, of the 11,720 unique IP addresses seen accessing the site with a valid unique identifier, only 10.2% were blacklisted as crawlers. In contrast, 55.3% of all unique identifiers used in requests originated from these crawlers. For all non-image requests made to the site, 87.43% were made by blacklisted IP addresses.

The number of conversions (E) is the number of visits to the purchase page of the pharmacy site, or the number of executions of the fake self-propagation program.

Our results for Storm spam campaigns show that the spam conversion rate is quite low. For example, out of 350 million pharmacy campaign e-mails only 28 conversions resulted (and no crawler ever completed a purchase so errors in crawler filtering plays no role). However, a very low conversion rate does not necessary imply low revenue or profitability. We discuss the implications of the conversion rate on the spam conversion proposition further in Section 8.

5.3 Time to click

The conversion pipeline shows what fraction of spam ultimately resulted visits to the advertised sites. However, it does not reflect the latency between when the spam was sent and when a user clicked on it. The longer it takes users to act, the longer the scam hosting infrastructure will need to remain available to extract revenue from the spam [3]. Put another way, how long does a spam-advertised site need to be available to collect its potential revenue?

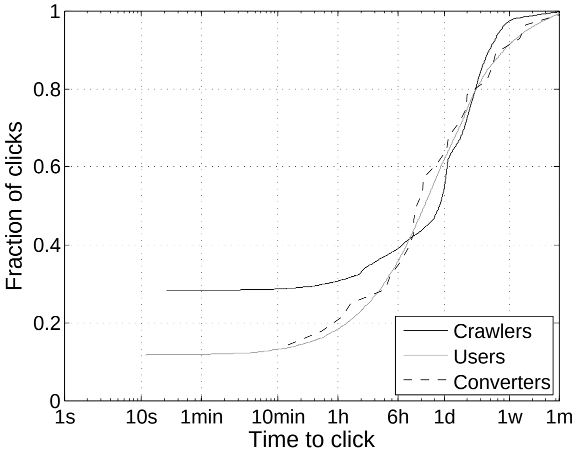


Figure 7: Time-to-click distributions for accesses to the pharmacy site.

Figure 7 shows the cumulative distribution of the "time-to-click" for accesses to the pharmacy site. The time-to-click is the time from when spam is sent (when a proxy forwards a spam workload to a worker bot) to when a user "clicks" on the URL in the spam (when a host first accesses the Web site). The graph shows three distributions for the accesses by all users, the users who visited the purchase page ("converters"), and the automated crawlers (14,716 such accesses). Note that we focus on the pharmacy site since, absent a unique identifier, we do not have a mechanism to link visits to the self-propagation site to specific spam messages and their time of delivery.

The user and crawler distributions show distinctly different behavior. Almost 30% of the crawler accesses are within 20 seconds of worker bots sending spam. This behavior suggests that these crawlers are configured to scan sites advertised in spam immediately upon delivery. Another 10% of crawler accesses have a time-to-click of 1 day, suggesting crawlers configured to access spam-advertised sites periodically in batches. In contrast, only 10% of the user population accesses spam URLs immediately, and the remaining distribution is smooth without any distinct modes. The distributions for all users and users who "convert" are roughly similar, suggesting little correlation between time-to-click and whether a user visiting a site will convert. While most user visits occur within the first 24 hours, 10% of times-to-click are a week to a month, indicating that advertised sites need to be available for long durations to capture full revenue potential.

6. EFFECTS OF BLACKLISTING

A major effect on the efficacy of spam delivery is the employment by numerous ISPs of address-based blacklisting to reject e-mail from hosts previously reported as sourcing spam. To assess the impact of blacklisting, during the course of our experiments we monitored the *Composite Blocking List* (CBL) [1], a blacklist source used by the operators of some of our institutions. At any given time the CBL lists on the order of 4–6 million IP addresses that have sent e-mail to various spamtraps. We were able to monitor the CBL from March 21 – April 2, 2008, from the start of the Pharmacy campaign until the end of the April Fool campaign. Although the monitoring does not cover the full extent of all campaigns, we believe our results to be representative of the effects of CBL during the time frame of our experiments.

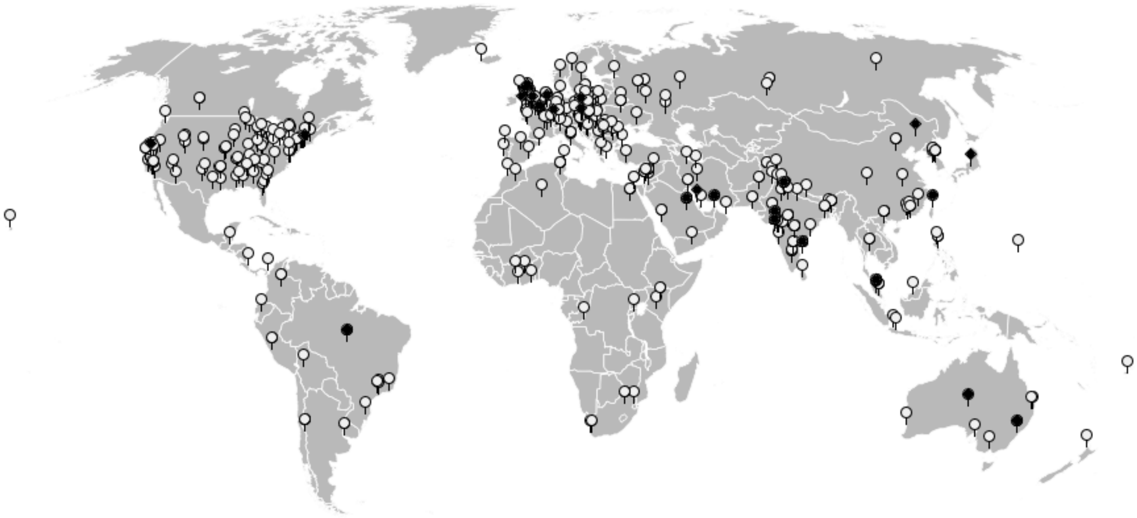


Figure 9: Geographic locations of the hosts that "convert" on spam: the 541 hosts that execute the emulated self-propagation program (light grey), and the 28 hosts that visit the purchase page of the emulated pharmacy site (black).

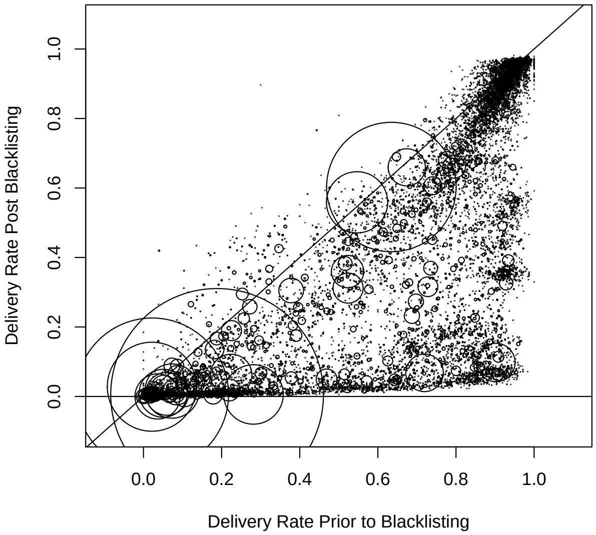


Figure 8: Change in per-domain delivery rates as seen prior to a worker bot appearing in the blacklist (x-axis) vs. after appearing (y-axis). Each circle represents a domain targeted by at least 1,000 analyzable deliveries, with the radius scaled in proportion to the number of delivery attempts.

We downloaded the current CBL blacklist every half hour, enabling us to determine which worker bots in our measurements were present on the list and how their arrival on the list related to their botnet activity. Of 40,864 workers that sent delivery reports, fully 81% appeared on the CBL. Of those appearing at some point on the list, 77% were on the list prior to our observing their receipt of spamming directives, appearing first on the list 4.4 days (median) earlier. Of those not initially listed but then listed subsequently, the median interval until listing was 1.5 hours, strongly suggesting that the spamming activity we observed them being instructed to conduct quickly led to their detection and blacklisting. Of hosts never appearing on the list, more than 75% never reported successful delivery of spam, indicating that the reason for their lack of listing was simply their inability to effectively annoy anyone.

One confounding factor is that the CBL exhibits considerable flux once an address first appears on the blacklist: the worker bots typically (median) experience 5 cycles of listing-followed-by-delisting. Much of this churn comes from a few periods of massive delistings, which appear to be glitches in maintenance (or propagation) of the blacklist rather than a response to external events. (If delistings arose due to botmasters using the delisting process to render their workers more effective for a while, then it might be possible to monitor the delisting process in order to conduct botnet counterintelligence, similar to that developed previously for blacklisting lookups [18].) Due to caching of blacklist entries by sites, we thus face ambiguity regarding whether a given worker is viewed as blacklisted at a given time. For our preliminary analysis, we simply consider a worker as blacklisted from the point where it first appears on the CBL onwards.

We would expect that the impact of blacklisting on spam delivery strongly depends on the domain targeted in a given e-mail, since some domains incorporate blacklist feeds such as the CBL into their mailer operations and others do not. To explore this effect, Figure 8 plots the per-domain delivery rate: the number of spam e-mails that workers reported as successfully delivered to the domain divided by number attempted to that domain. The x-axis shows the delivery rate for spams sent by a worker prior to its appearance in the CBL, and the y-axis shows the rate after its appearance in the CBL. We limit the plot to the 10,879 domains to which workers attempted to deliver at least 1,000 spams. We plot delivery rates for the two different campaigns as separate circles, though the overall nature of the plot does not change between them. The radius of each plotted circle scales in proportion to the number of delivery attempts, the largest corresponding to domains such as hotmail.com, yahoo.com, and gmail.com.

From the plot we clearly see a range of blacklisting behavior by different domains. Some employ other effective anti-spam filtering, indicated by their appearance near the origin — spam did not get through even prior to appearing on the CBL blacklist. Some make heavy use of either the CBL or a similar list (y-axis near zero, but

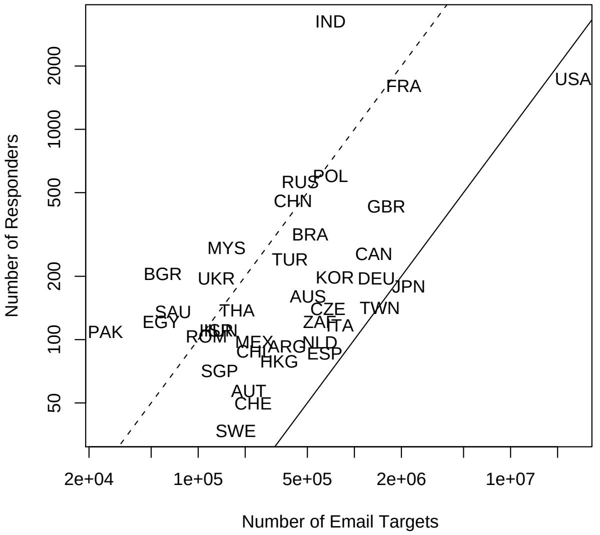


Figure 10: Volume of e-mail targeting (x-axis) vs. responses (y-axis) for the most prominent country-code TLDs. The x and y axes correspond to Stages A and D in the pipeline (Figure 6), respectively.

x-axis greater than zero), while others appear insensitive to blacklisting (those lying on the diagonal). Since points lie predominantly below the diagonal, we see that either blacklisting or some other effect related to sustained spamming activity (e.g., learning content signatures) diminishes the delivery rate seen at most domains. Delisting followed by relisting may account for some of the spread of points seen here; those few points above the diagonal may simply be due to statistical fluctuations. Finally, the cloud of points to the upper right indicates a large number of domains that are not much targeted individually, but collectively comprise a significant population that appears to employ no effective anti-spam measures.

7. CONVERSION ANALYSIS

We now turn to a preliminary look at possible factors influencing response to spam. For the present, we confine our analysis to coarse-grained effects.

We start by mapping the geographic distribution of the hosts that "convert" on the spam campaigns we monitored. Figure 9 maps the locations of the 541 hosts that execute the emulated self-propagation program, and the 28 hosts that visit the purchase page of the emulated pharmacy site. The map shows that users around the world respond to spam.

Figure 10 looks at differences in response rates among nations as determined by prevalent country-code e-mail domain TLDs. To allow the inclusion of generic TLDs such as `.com`, for each e-mail address we consider it a member of the country hosting its mail server; we remove domains that resolve to multiple countries, categorizing them as "international" domains. The x-axis shows the volume of e-mail (log-scaled) targeting a given country, while the y-axis gives the number of responses recorded at our Web servers (also log-scaled), corresponding to Stages A and D in the pipeline (Figure 6), respectively. The solid line reflects a response rate of 10^{-4} and the dashed line a rate of 10^{-3}. Not surprisingly, we see that the spam campaigns target e-mail addresses in the United States substantially more than any other country. Further, India, France and the United States dominate responses. In terms of response *rates*, however, India, Pakistan, and Bulgaria have the highest response rates than any other countries (furthest away from the diagonal). The United States, although a dominant target and responder, has the lowest resulting response rate of any country, followed by Japan and Taiwan.

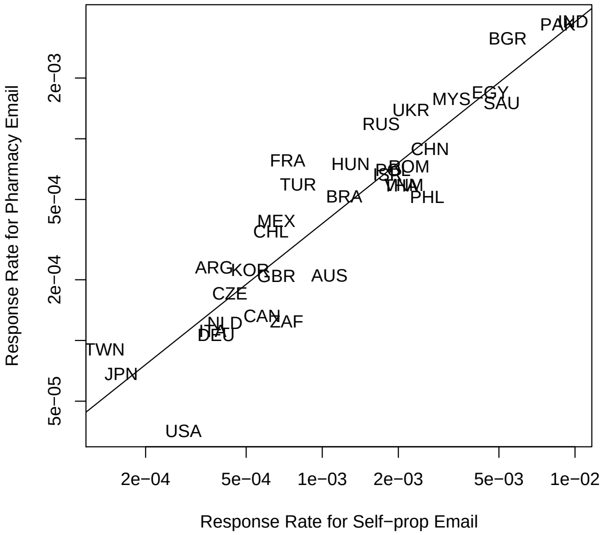


Figure 11: Response rates (stage D in the pipeline) by TLD for executable download (x-axis) vs. pharmacy visits (y-axis).

However, the countries with predominant response rates do not appear to reflect a heightened interest in users from those countries in the specific spam offerings. Figure 11 plots the rates for the most prominent countries responding to self-propagation vs. pharmacy spams. The median ratio between these two rates is 0.38 (diagonal line). We see that India and Pakistan in fact exhibit almost exactly this ratio (upper-right corner), and Bulgaria is not far from it. Indeed, only a few TLDs exhibit significantly different ratios, including the US and France, the two countries other than India with a high number of responders; users in the US respond to the self-propagation spam substantially more than pharmaceutical spam, and vice-versa with users in France. These results suggest that, for the most part, per-country differences in response rate are due to *structural* causes (quality of spam filtering, general user anti-spam education) rather than differing degrees of cultural or national interest in the particular promises or products conveyed by the spam.

8. CONCLUSIONS

This paper describes what we believe is the first large-scale quantitative study of spam conversion. We developed a methodology that uses botnet infiltration to indirectly instrument spam e-mails such that user clicks on these messages are taken to replica Web sites under our control. Using this methodology we instrumented almost 500 million spam messages, comprising three major campaigns, and quantitatively characterized both the delivery process and the conversion rate.

We would be the first to admit that these results represent a single data point and are not necessarily representative of spam as a whole. Different campaigns, using different tactics and marketing different products will undoubtedly produce different outcomes. Indeed, we caution *strongly* against researchers using the conversion rates we have measured for these Storm-based campaigns to justify assumptions in any other context. At the same time, it is tempting to speculate on what the numbers we have measured might *mean*. We succumb to this temptation below, with the understanding that few of our speculations can be empirically validated at this time.

After 26 days, and almost 350 million e-mail messages, only 28 sales resulted — a conversion rate of well under 0.00001%. Of these, all but one were for male-enhancement products and the average purchase price was close to $100. Taken together, these conversions would have resulted in revenues of $2,731.88 — a bit over $100 a day for the measurement period or $140 per day for periods when the campaign was active. However, our study interposed on only a small fraction of the overall Storm network — we estimate roughly 1.5 percent based on the fraction of worker bots we proxy. Thus, the total daily revenue attributable to Storm's pharmacy campaign is likely closer to $7000 (or $9500 during periods of campaign activity). By the same logic, we estimate that Storm self-propagation campaigns can produce between 3500 and 8500 new bots per day.

Under the assumption that our measurements are representative over time (an admittedly dangerous assumption when dealing with such small samples), we can extrapolate that, were it sent continuously at the same rate, Storm-generated pharmaceutical spam would produce roughly 3.5 million dollars of revenue in a year. This number could be even higher if spam-advertised pharmacies experience repeat business. A bit less than "millions of dollars every day", but certainly a healthy enterprise.

The next obvious question is, "How much of this revenue is profit"? Here things are even murkier. First, we must consider how much of the gross revenue is actually recovered on a sale. Assuming the pharmacy campaign drives traffic to an affiliate program (and there are very strong anecdotal reasons to believe this is so) then the gross revenue is likely split between the affiliate and the program (a annual net revenue of $1.75M using our previous estimate). Next, we must subtract business costs. These include a number of incidental expenses (domain registration, bullet-proof hosting fees, etc) that are basically fixed sunk costs, and the cost to distribute the spam itself.

Anecdotal reports place the *retail* price of spam delivery at a bit under $80 per million [22]. This cost is an order of magnitude less than what legitimate commercial mailers charge, but is still a significant overhead; sending 350M e-mails would cost more than $25,000. Indeed, given the net revenues we estimate, retail spam delivery would only make sense if it were 20 times cheaper still.

And yet, Storm continues to distribute pharmacy spam — suggesting that it is in fact profitable. One explanation is that Storm's masters are vertically integrated and the purveyors of Storm's pharmacy spam are none other than the operators of Storm itself (i.e., that Storm does not deliver these spams for a third-part in exchange for a fee). There is some evidence for this, since the distribution of target e-mail domain names between the self-propagation and pharmacy campaigns is virtually identical. Since the self-propagation campaigns fundamentally must be run by the botnet's owners, this suggests the purveyor of the pharmacy spam is one and the same. A similar observation can be made in the harvesting of e-mail addresses from the local hard drives of Storm hosts. These e-mail addresses subsequently appear in the target address lists of the pharmacy campaign and self-propagation campaigns alike. Moreover, neither of these behaviors is found in any of the other (smaller) campaigns distributed by Storm (suggesting that these may in fact be fee-for-service distribution arrangements). If true, then the cost of distribution is largely that of the labor used in the development and maintenance of the botnet software itself. While we are unable to provide any meaningful estimates of this cost (since we do not know which labor market Storm is developed in), we surmise that it is roughly the cost of two or three good programmers.

If true, this hypothesis is heartening since it suggests that the third-party *retail* market for spam distribution has not grown large or efficient enough to produce competitive pricing and thus, that profitable spam campaigns require organizations that can assemble complete "soup-to-nuts" teams. Put another way, the profit margin for spam (at least for this one pharmacy campaign) may be meager enough that spammers must be sensitive to the details of how their campaigns are run and are economically susceptible to new defenses.

9. ACKNOWLEDGMENTS

This was one of the most complex measurement studies our group has ever conducted and would have been impossible without the contributions of a large and supportive cast. Here we offer our thanks for their insightful feedback and individual contributions to our effort.

Jordan Hayes provided decidedly non-trivial help with site domain registration. Peter Blair, Paul Karkas, Jamie Knight and Garrick Lau at Tucows supported this activity (once we convinced them we weren't spammers) and allowed us to use reputable registrars. Randy Bush provided overall guidance and help concerning Internet operations and policy issues while Erin Kenneally advised us on legal issues. Brian Kantor set up and managed our DNS, Web and SMTP servers, while Scott Campbell and Stephen Chan performed massive DNS lookups for us. Jef Poskanzer provided data access for debugging our experiment, Stephen Chenette provided technical assistance and Fallon Chen was our in-house graphic designer. Bill Young and Gregory Ruiz-Ade set up target e-mail accounts in UCSD's CSE department. Special thanks to Gabriel Lawrence and Jim Madden of UCSD's ACT for supporting this activity on UCSD's systems and networks. Finally, our thanks to the anonymous reviewers for their time and commentary.

This work was made possible by the National Science Foundation grants NSF-0433702 and NSF-0433668 and by generous research, operational and in-kind support from Cisco, Microsoft, HP, Intel, VMWare, ESnet, the Lawrence Berkeley National Laboratory, and UCSD's Center for Networked Systems. 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 these organizations.

10. REFERENCES

[1] Composite Blocking List (CBL). http://cbl.abuseat.org/, March 2008.

[2] C. Akass. Storm worm 'making millions a day'. http://www.pcw.co.uk/personal-computer-world/news/2209293/strom-worm-making-millions-day, February 2008.

[3] 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.

[4] J. Angwin. Elusive Spammer Sends EarthLink on Long Chase. http://online.wsj.com/article_email/SB10522559338237260.html, May 2003.

[5] D. M. Association. DMA Releases 5th Annual 'Response Rate Trends Report'. http://www.the-dma.org/cgi/disppressrelease?article=1008, October 2007.

[6] J. Aycock and N. Friess. Spam Zombies from Outer Space, January 2006.

[7] R. Boehme and T. Ho. The Effect of Stock Spam on Financial Markets. In *Proceedings of the Fifth Workshop on the Economics of Information Security (WEIS)*, June 2006.

[8] L. Frieder and J. Zittrain. Spam Works: Evidence from Stock Touts and Corresponding Market Activity. *Berkman Center Research Publication*, 2006.

[9] J. Goodman and R. Rounthwaite. Stopping Outgoing Spam. *Proceedings of the 5th ACM conference on Electronic commerce*, pages 30–39, 2004.

[10] M. Hanke and F. Hauser. On the Effects of Stock Spam E-mails. *Journal of Financial Markets*, 11(1):57–83, 2008.

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

[12] C. Kanich, K. Levchenko, B. Enright, G. M. Voelker, and S. Savage. The Heisenbot Uncertainty Problem: Challenges in Separating Bots from Chaff. In *First USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET'08)*, April 2008.

[13] D. Khong. An Economic Analysis of Spam Law. *Erasmus Law and Economics Review*, 1(1), 2004.

[14] J. Kirk. Former spammer: 'I know I'm going to hell'. http://www.macworld.com/article/58997/2007/07/spammer.html, July 2007.

[15] C. Kreibich, C. Kanich, K. Levchenko, B. Enright, G. M. Voelker, V. Paxson, and S. Savage. On the Spam Campaign Trail. In *First USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET'08)*, April 2008.

[16] P. Maymounkov and D. Mazières. Kademlia: A Peer-to-Peer Information System Based on the XOR Metric. *First International Workshop on Peer-To-Peer Systems (IPTPS), Cambridge, MA, USA*, March 2002.

[17] W. Y. P. Judge, D. Alperovitch. Understanding and Reversing the Profit Model of Spam. In *Workshop on Economics of Information Security 2005. (WEIS 2005)*, Boston, MA, USA, June 2005.

[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] A. Serjantov and R. Clayton. Modeling Incentives for Email Blocking Strategies. *Workshop on the Economics of Information Security (WEIS05)*, 2005.

[20] Y. Wang, M. Ma, Y. Niu, and H. Chen. Spam Double-Funnel: Connecting Web Spammers with Advertisers. *Proceedings of the 16th international conference on World Wide Web*, pages 291–300, 2007.

[21] D. Watson. All Spammers Go to Hell (posting to funsec list). http://www.mail-archive.com/funsec%40linuxbox.org/msg03346.html, July 2007.

[22] T. Wilson. Competition May be Driving Surge in Botnets, Spam. http://www.darkreading.com/document.asp?doc_id=142690, 2008.