BOTMAGNIFIER: Locating Spambots on the Internet

Gianluca Stringhini[§], Thorsten Holz[‡], Brett Stone-Gross[§],
Christopher Kruegel[§], and Giovanni Vigna[§]

[§]University of California, Santa Barbara
{gianluca,bstone,chris,vigna}@cs.ucsb.edu

[‡] Ruhr-University Bochum
thorsten.holz@rub.de

Abstract

Unsolicited bulk email (*spam*) is used by cyber-criminals to lure users into scams and to spread malware infections. Most of these unwanted messages are sent by spam botnets, which are networks of compromised machines under the control of a single (malicious) entity. Often, these botnets are rented out to particular groups to carry out spam campaigns, in which similar mail messages are sent to a large group of Internet users in a short amount of time. Tracking the bot-infected hosts that participate in spam campaigns, and attributing these hosts to spam botnets that are active on the Internet, are challenging but important tasks. In particular, this information can improve blacklist-based spam defenses and guide botnet mitigation efforts.

In this paper, we present a novel technique to support the identification and tracking of bots that send spam. Our technique takes as input an initial set of IP addresses that are known to be associated with spam bots, and learns their spamming behavior. This initial set is then "magnified" by analyzing large-scale mail delivery logs to identify other hosts on the Internet whose behavior is similar to the behavior previously modeled. We implemented our technique in a tool, called BOTMAGNIFIER, and applied it to several data streams related to the delivery of email traffic. Our results show that it is possible to identify and track a substantial number of spam bots by using our magnification technique. We also perform attribution of the identified spam hosts and track the evolution and activity of well-known spamming botnets over time. Moreover, we show that our results can help to improve state-of-the-art spam blacklists.

1 Introduction

Email spam is one of the open problems in the area of IT security, and has attracted a significant amount of research over many years [11, 26, 28, 40, 42]. Unsolicited bulk email messages account for almost 90% of the world-wide email traffic [20], and a lucrative business has emerged around them [12]. The content of spam emails lures users into scams, promises to sell cheap goods and pharmaceutical products, and spreads malicious software by distributing links to websites that perform drive-by download attacks [24].

Recent studies indicate that, nowadays, about 85% of the overall spam traffic on the Internet is sent with the help of *spamming botnets* [20, 36]. Botnets are networks of compromised machines under the direction of a single entity, the so-called *botmaster*. While different botnets serve different, nefarious goals, one important purpose of botnets is the distribution of spam emails. The reason is that botnets provide two advantages for spammers. First, a botnet serves as a convenient infrastructure for sending out large quantities of messages; it is essentially a large, distributed computing system with massive bandwidth. A botmaster can send out tens of millions of emails within a few hours using thousands of infected machines. Second, a botnet allows an attacker to evade spam filtering techniques based on the sender IP addresses. The reason is that the IP addresses of some infected machines change frequently (e.g., due to the expiration of a DHCP lease, or to the change in network location in the case of an infected portable computer). Moreover, it is easy to infect machines and recruit them as new members into a botnet. This means that blacklists need to be updated constantly by tracking the IP addresses of spamming bots.

Tracking spambots is challenging. One approach to detect infected machines is to set up *spam traps*. These are fake email addresses (i.e., addresses not associated with real users) that are published throughout the Internet with the purpose of attracting and collecting spam messages. By extracting the sender IP addresses from the emails received by a spam trap, it is possible to obtain a list of bot-infected machines. However, this approach faces two main problems. First, it is likely that only a subset of the bots belonging to a certain botnet will send emails to the spam trap addresses. Therefore, the analysis of the messages collected by the spam trap can provide only a partial view of the activity of the botnet. Second, some botnets might only target users located in a specific country (e.g., due to the language used in the email), and thus a spam trap located in a different country would not observe those bots.

Other approaches to identify the hosts that are part of a spamming botnet are specific to particular botnets. For example, by taking control of the command & control (C&C) component of a botnet [21, 26], or by analyzing the communication protocol used by the bots to interact with other components of the infrastructure [6, 15, 32], it is possible to enumerate (a subset of) the IP addresses of the hosts that are part of a botnet. However, in these cases, the results are specific to the particular botnet that is being targeted (and, typically, the type of C&C used).

In this paper, we present a novel approach to identify and track spambot populations on the Internet. Our ambitious goal is to track the IP addresses of all active hosts that belong to every spamming botnet. By active hosts, we mean hosts that are online and that participate in spam campaigns. Comprehensive tracking of the IP addresses belonging to spamming botnets is useful for several reasons:

- Internet Service Providers can take countermeasures to prevent the bots whose IP addresses reside in their networks from sending out email messages.
- Organizations can clean up compromised machines in their networks.
- Existing blacklists and systems that analyze network-level features of emails can be improved by providing accurate information about machines that are currently sending out spam emails.
- By monitoring the number of bots that are part of different botnets, it is possible to guide and support mitigation efforts so that the C&C infrastructures of the largest, most aggressive, or fastest-growing botnets are targeted first.

Our approach to tracking spamming bots is based on the following insight: bots that belong to the same botnet share the same C&C infrastructure and the same code base. As a result, these bots will feature similar behavior when sending spam [9, 40, 41]. In contrast, bots belonging to different spamming botnets will typically use different parameters for sending spam mails (e.g., the size of the target email address list, the domains or countries that are targeted, the spam contents, or the timing of their actions). More precisely, we leverage the fact that bots (of a particular botnet) that participate in a *spam campaign* share similarities in the destinations (domains) that they target and in the time periods they are active. Similar to previous work [15], we consider a spam campaign to be a set of email messages that share a substantial amount of content and structure (e.g., a spam campaign might involve the distribution of messages that promote a specific pharmaceutical scam).

Input datasets. At a high level, our approach takes two datasets as input. The first dataset contains the IP addresses of known spamming bots that are active during a certain time period (we call this time period the *observation period*). The IP addresses are grouped by spam campaign. That is, IP addresses in the same group sent the same type of messages. We refer to these groups of IP addresses as *seed pools*. The second dataset is a log of email transactions carried out on the Internet during the same time period. This log, called the *transaction log*, contains entries that specify that, at a certain time, IP address C attempted to send an email message to IP address S. The log does not need to be a complete log of every email transaction on the Internet (as it would be unfeasible to collect this information). However, as we will discuss later, our approach becomes more effective as this log becomes more comprehensive.

Approach. In the first step of our approach, we search the transaction log for entries in which the sender IP address is one of the IP addresses in the seed pools (i.e., the known spambots). Then, we analyze these entries and generate a number of behavioral profiles that capture the way in which the hosts in the seed pools sent emails during the observation period.

In the second step of the approach, the whole transaction log is searched for patterns of behavior that are similar to the spambot behavior previously learned from the seed pools. The hosts that behave in a similar manner are flagged as possible spamming bots, and their IP addresses are added to the corresponding *magnified pool*.

In the third and final step, heuristics are applied to reduce false positives and to assign spam campaigns (and the IP addresses of bots) to specific botnets (e.g., *Rustock* [5], *Cutwail* [35], or *MegaD* [4, 6]).

We implemented our approach in a tool, called BOTMAGNIFIER. In order to populate our seed pools, we used data from a large spam trap set up by an Internet Service Provider (ISP). Our transaction logs were constructed by running a mirror for *Spamhaus*, a popular DNS-based blacklist. Note that other sources of information can be used to either populate the seed pools or to build a transaction log. As we will show, BOTMAGNIFIER also works for transaction logs extracted from netflow data collected from a large ISP's backbone routers.

BOTMAGNIFIER is executed periodically, at the end of each observation period. It outputs a list of the IP addresses of all bots in the magnified pools that were found during the most recent period. Moreover, BOTMAGNIFIER associates with each seed and magnified pool a label that identifies (when possible) the name of the botnet that carried out the corresponding spam campaign. Our experimental results show that our system can find a significant number of additional IP addresses compared to the seed baseline. Furthermore, BotMagnifier is able to detect emerging spamming botnets. As we will show, we identified the resurrection of the *Waledac* spam botnet during the evaluation period, demonstrating the ability of our technique to find new botnets.

In summary, we provide the following contributions:

- We developed a novel method for characterizing the behavior of spamming bots.
- We provide a novel technique for identifying and tracking spamming bot populations on the Internet, using a "magnification" process.
- We assigned spam campaigns to the major botnets, and we studied the evolution of the bot population of these botnets over time.
- We validated our results using ground truth collected from a number of C&C servers used by a large spamming botnet, and we demonstrated the applicability of our technique to real-world, large-scale datasets.

2 Input Datasets

BotMagnifier requires two input datasets to track spambots: *seed pools* and a *transaction log*. In this section, we discuss how these two datasets are obtained.

2.1 Seed Pools

A *seed pool* is a set of IP addresses of hosts that, during the most recent observation period, participated in a specific spam campaign. The underlying assumption is that the hosts whose IP addresses are in the same seed pool are part of the same spamming botnet, and they were instructed to send a certain batch of messages (e.g., emails advertising cheap Viagra or replica watches).

To generate the seed pools for the various spam campaigns, we took advantage of the information collected by a spam trap set up by a large US ISP. Since the email addresses used in this spam trap do not correspond to real customers, all the received emails are spam. We collected data from the spam trap between September 1, 2010 and February 10, 2011, with a downtime of about 15 days in November 2011. The spam trap collected, on average, 924,000 spam messages from 268,000 IP addresses every day.

Identifying similar messages. We identify spam campaigns within this dataset by looking for similar email messages. More precisely, we analyze the subject lines of all spam messages received during the last observation period (currently one day: see discussion below). Messages that share a similar subject line are considered to be part of the same campaign (during this period).

Unfortunately, the subject lines of messages of a certain campaign are typically not identical. In fact, most botnets vary the subject lines of the message they send to avoid detection by anti-spam systems. For example, some botnets put the user name of the recipient in the subject, or change the price of the pills being sold in drug-related campaigns. To mitigate this problem, we extract *templates* from the actual subject lines. To this end, we substitute user names, email addresses, and numbers with placeholder regular expressions. User names are recognized as tokens that are identical to the first part of the destination email address (the part to the left of the @ sign). For example, the subject line "`john, get 90% discounts!`" sent to user `john@example.com` becomes "`\w+, get [0-9]+% discounts!`"

More sophisticated botnets, such as *Rustock*, add random text fetched from Wikipedia to both the email body and the subject line. Other botnets, such as *Lethic*, add a random word at the end of each subject. These tricks make it harder to group emails belonging to the same campaign that are sent by different bots, because different bots will add distinct text to each message. To handle this problem, we developed a set of custom rules for the largest spamming botnets that remove the spurious content from the subject lines.

Once the subjects of the messages have been transformed into templates and the spurious information has been removed, messages with the same template subject line are clustered together. This approach is less sophisticated than methods that take into account more features of the spam messages [22, 40], but we found (by manual investigation) that our simple approach was very effective for our purpose. Our approach, although sufficient, could be refined even further by incorporating these more sophisticated schemes to improve our ability to recognize spam campaigns.

Once the messages are clustered, the IP addresses of the senders in each cluster are extracted. These sets of IP addresses represent the seed pools that are used as input to our magnification technique.

Seed pool size. During our experiments, we found that seed pools that contain a very small number of IP addresses do not provide good results. The reason is that the behavior patterns that can be constructed from only a few known bot instances are not precise enough to represent the activity of a botnet. For example, campaigns involving 200 unique IP addresses in the seed pool produced, on average, magnified sets where 60% of the IP addresses were not listed in *Spamhaus*, and therefore were likely legitimate servers. Similarly, campaigns with a seed pool size of 500 IP addresses still produced magnified sets where 25% of the IP addresses were marked as legitimate by *Spamhaus*. For these reasons, we only consider those campaigns for which we have observed more than 1,000 unique sender IP addresses. The emails belonging to these campaigns account for roughly 84% of the overall traffic observed by our spam trap. It is interesting to notice that 8% of the overall traffic belongs to campaigns carried out by less than 10 distinct IP addresses per day. Such campaigns are carried out by dedicated servers and abused email service providers. The aggressive spam behavior of these servers and their lack of geographic/IP diversity makes them trivial to detect without the need for magnification.

The lower limit on the size of seed pools has implications for the length of the observation period. When this interval is too short, the seed pools are likely to be too small. On the other hand, many campaigns last less than a few hours. Thus, it is not useful to make the observation period too long. Also, when increasing the length of the observation period, there is a delay introduced before BOTMAGNIFIER can identify new spam hosts. This is not desirable when the output is used for improving spam defenses. In practice, we found that an observation period of one day allows us to generate sufficiently large seed pools from the available spam feed. To evaluate the impact that the choice of the analysis period might have on our analysis system, we looked at the length of 100 spam campaigns, detected over a period of one day. The average length of these campaigns is 9 hours, with a standard deviation of 6 hours. Of the campaigns we analyzed, 25 lasted less than four hours. However, only two of these campaigns did not generate large enough seed pools to be considered by BOTMAGNIFIER. On the other hand, 8 campaigns that lasted more than 18 hours would not have generated large enough seed pools if we used a shorter observation period. Also, by manual investigation, we found that campaigns that last more than one day typically reach the threshold of 1,000 IP addresses for their seed pool within the first day. Therefore, we believe that the choice of an observation period of one day works well, given the characteristics of the transaction log we used. Of course, if the volume of either the seed pools or the transaction log increased, the observation period could be reduced accordingly, making the system more effective for real-time spam blacklisting.

Note that it is not a problem when a spam campaign spans multiple observation periods. In this case, the bots that participate in this spam campaign and are active during multiple periods are simply included in multiple seed pools (one for each observation period for this campaign).

2.2 Transaction Log

The transaction log is a record of email transactions carried out on the Internet during the same time period used for the generation of the seed pools. For the current version of BOTMAGNIFIER and the majority of our experiments, we obtained the transaction log by analyzing the queries to a mirror of *Spamhaus*, a widely-used DNS-based blacklisting service (DNSBL). When an email server S is contacted by a client C that wants to send an email message, server S contacts one of the *Spamhaus* mirrors and asks whether the IP address of the client C is a known spam host. If C is a known spammer, the connection is rejected or the email is marked as spam.

Each query to *Spamhaus* contains the IP address of C. It is possible that S may not query *Spamhaus* directly. In some cases, S is configured to use a local DNS server that forwards the query. In such cases, we would mistakenly consider the IP address of the DNS server as the mail server. However, the actual value of the IP address of S is not important for the subsequent analysis. It is only important to recognize when two different clients send email to the *same* server S. Thus, as long as emails sent to server S yield *Spamhaus* queries that always come from the same IP address, our technique is not affected.

Each query generates an entry in the transaction log. More precisely, the entry contains a timestamp, the IP address of the sender of the message, and the IP address of the server issuing the query. Of course, by monitoring a single *Spamhaus* mirror (out of 60 deployed throughout the Internet), we can observe only a small fraction of the global email transactions. Our mirror observes roughly one hundred million email transactions a day, compared to estimates that put the number of emails sent daily at hundreds of billions [13].

Note that even though *Spamhaus* is a blacklisting service, we do not use the information it provides about the blacklisted hosts to perform our analysis. Instead, we use the *Spamhaus* mirror only to collect the transaction logs, regardless of the fact that a sender may be a known spammer. In fact, other sources of information can be used to either populate the seed pools or to collect the transaction log. To demonstrate this, we also ran BOTMAGNIFIER on transaction logs extracted from netflow data collected from a number of backbone routers of a large ISP. The results show that our general approach is still valid (see Section 6.4 for details).

3 Characterizing Bot Behavior

Given the two input datasets described in the previous section, the first step of our approach is to extract the behavior of known spambots. To this end, the transaction log is consulted. More precisely, for each seed pool, we query the transaction log to find all events that are associated with all of the IP addresses in that seed pool (recall that the IP addresses in a seed pool correspond to known spambots). Here, an event is an entry in the transaction log where the known spambot is the *sender* of an email. Essentially, we extract all the instances in the transaction log where a known bot has sent an email.

Once the transaction log entries associated with a seed pool are extracted, we analyze the *destinations* of the spam messages to characterize the bots' behavior. That is, the behavior of the bots in a seed pool is characterized by the set of destination IP addresses that received spam messages. We call the set of server IP addresses targeted by the bots in a seed pool this pool's *target set*.

The reason for extracting a seed pool's target set is the insight that bots belonging to the same botnet receive the same list of email addresses to spam, or, at least, a subset of addresses belonging to the same list. Therefore, during their spamming activity, bots belonging to botnet A will target the addresses contained in list L_A, while bots belonging to botnet B will target destinations belonging to list L_B. That is, the targets of a spam campaign characterize the activity of a botnet.

Unfortunately, the target sets of two botnets often have substantial overlap. The reason is that there are many popular destinations (server addresses) that are targeted by most botnets (e.g., the email servers of Google, Yahoo, large ISPs with many users, etc.) Therefore, we want to derive, for each spam campaign (seed pool), the most *characterizing set* of destination IP addresses. To this end, we remove from each pool's target set all server IP addresses that appear in any target set belonging to another another seed pool.

More precisely, consider the seed pools $P = p_1, p_2, \ldots, p_n$. Each pool p_i stores the IP addresses of known bots that participated in a certain campaign: $i_1, i_2, \ldots, i_m$. In addition, consider that the transaction log L contains entries in the form $\langle t, i_s, i_d \rangle$, where t is a time stamp, i_s is the IP address of the sender of an email and i_d is the IP address of the destination server of an email. For each seed pool p_i, we build this seed pool's target set $T(p_i)$ as follows:

$$T(p_i) := \{i_d | \langle t, i_s, i_d \rangle \in L \wedge i_s \in p_i\}. \qquad (1)$$

Then, we compute the characterizing set $C(p_i)$ of a seed pool p_i as follows:

$$C(p_i) := \{i_d | i_d \in T(p_i) \wedge i_d \notin T(p_j), j \neq i\}. \qquad (2)$$

As a result, $C(p_i)$ contains only the target addresses that are unique (characteristic) for the destinations of bots in seed pool p_i. The characterizing set $C(p_i)$ of each pool is the input to the next step of our approach.

4 Bot Magnification

The goal of the bot magnification step is to find the IP addresses of additional, previously-unknown bots that have participated in a known spam campaign. More precisely, the goal of this step is to search the transaction log for IP addresses that behave similarly to the bots in a seed pool p_i. If such matches can be found, the corresponding IP addresses are added to the *magnification set* associated with p_i. This means that a magnification set stores the IP addresses of additional, previously-unknown bots.

BotMagnifier considers an IP address x_i that appears in the transaction log L as matching the behavior of a certain seed pool p_i (and, thus, belonging to that spam campaign) if the following three conditions hold: (i) host x_i sent emails to at least N destinations in the seed pool's target set $T(p_i)$; (ii) the host never sent an email to a destination that does *not* belong to that target set; (iii) host x_i has contacted *at least one* destination that is unique for seed pool p_i (i.e., an address in $C(p_i)$). If all three conditions are met, then IP address x_i is added to the magnification set $M(p_i)$ of seed pool p_i.

More formally, if we define $D(x_i)$ as the set of destinations targeted by an IP address x_i, we have:

$$\begin{aligned} x_i \in M(p_i) \quad \Longleftrightarrow \quad & |D(x_i) \cap T(p_i)| \geq N \wedge \\ & D(x_i) \subseteq T(p_i) \wedge \\ & D(x_i) \cap C(p_i) \neq \emptyset. \qquad (3) \end{aligned}$$

The intuition behind this approach is the following: when a host h sends a reasonably large number of emails to the same destinations that were targeted by a spam campaign and not to any other targets, there is a strong indication that the email activity of this host is similar to the bots involved in the campaign. Moreover, to assign a host h to at most one campaign (the one that it is most similar), we require that h targets at least one unique destination of this campaign.

Threshold computation. The main challenge in this step is to determine an appropriate value for the threshold N, which captures the minimum number of destination IP addresses in $T(p_i)$ that a host must send emails to in order to be added to the magnification set $M(p_i)$. Setting N to a value that is too low will generate too many bot candidates, including legitimate email servers, and the tool would generate many false positives. Setting N to a value that is too high might discard many bots that should have been included in the magnification set (that is, the approach generates many false negatives). This trade-off between false positives and false negatives is a problem that appears in many security contexts, for example, when building models for intrusion detection.

An additional, important consideration for the proper choice of N is the size of the target set $|T(p_i)|$. Intu-

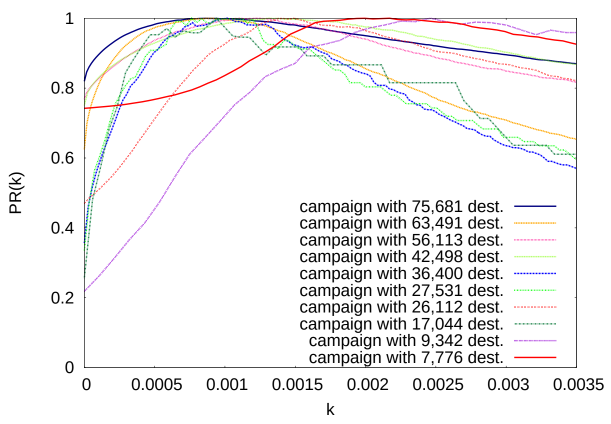


Figure 1: Quality of magnification for varying k using ten *Cutwail* campaigns of different sizes.

itively, we expect that N should be larger when the size of the target set increases. This is because a larger target set increases the chance that a random, legitimate email sender hits a sufficient number of targets by accident, and hence, will be incorrectly included into the magnification set. In contrast, bots carrying out a spam campaign that targets only a small number of destinations are easier to detect. The reason is that as soon as a legitimate email sender sends an email to a server that is *not* in the set targeted by the campaign, it will be immediately discarded by our magnification algorithm. Therefore, we represent the relationship between the threshold N and the size of the target set $|T(p_i)|$ as:

$$N = k \cdot |T(p_i)|, 0 < k \leq 1, \quad (4)$$

where k is a parameter. Ideally, the relation between N and $|T(p_i)|$ would be linear, and k will have a constant value. However, as will be clear from the discussion below, k also varies with the size of $|T(p_i)|$.

To determine a good value for k and, as a consequence, select a proper threshold N, we performed an analysis based on ground truth about the actual IP addresses involved in several spam campaigns. This information was collected from the takedown of more than a dozen C&C servers used by the Cutwail spam botnet. More specifically, each server stored comprehensive records (e.g., target email lists, bot IP addresses, etc.) about spam activities for a number of different campaigns [35]

In particular, we applied BotMagnifier to ten *Cutwail* campaigns, extracted from two different C&C servers. We used these ten campaigns since we had a precise view of the IP addresses of the bots that sent the emails. For the experiment, we varied the value for N in the magnification process from 0 to 300. This analysis yielded different magnification sets for each campaign. Then, using our knowledge about the actual bots B that were part of each campaign, we computed the precision P and recall R values for each threshold setting. Since we want to express the quality of the magnification process as a function of k, independently of the size of a campaign, we use Equation 4 to get $k = \frac{N}{|T(p_i)|}$.

The precision value $P(k)$ represents what fraction of the IP addresses that we obtain as candidates for the magnification set for a given k are actually among the ground truth IP addresses. The recall value $R(k)$, on the other hand, tells us what fraction of the total bot set B is identified. Intuitively, a low value of k will produce high $R(k)$, but low $P(k)$. When we increase k, $P(k)$ will increase, but $R(k)$ will decrease. Optimally, both precision and recall are high. Thus, for our analysis, we use the product $PR(k) = P(k) \cdot R(k)$ to characterize the quality of the magnification step. Figure 1 shows how $PR(k)$ varies for different values of k. As shown for each campaign, $PR(k)$ first increases, then stays relatively level, and then starts to decrease.

The results indicate that k is not a constant, but varies with the size of $|T(p_i)|$. In particular, small campaigns have a higher optimal value for k compared to larger campaigns: as $|T(p_i)|$ increases, the value of k slowly decreases. To reflect this observation, we use the following, simple way to compute k:

$$k = k_b + \frac{\alpha}{|T(p_i)|}, \quad (5)$$

where k_b is a constant value, α is a parameter, and $|T(p_i)|$ is the number of destinations that a campaign targeted. The parameters k_b and α are determined so that the quality of the magnification step PR is maximized for a given ground truth dataset. Using the *Cutwail* campaigns as the dataset, this yields $k_b = 8 \cdot 10^{-4}$ and $\alpha = 10$.

Our experimental results show that these parameter settings yield good results for a wide range of campaigns, carried out by several different botnets. This is because the magnification process is robust and not dependent on an optimal threshold selection. We found that non-optimal thresholds typically tend to decrease recall. That is, the magnification process does not find all bots that it could possibly detect, but false positives are limited. In Section 6.4, we show how the equation of k, with the values we determined for parameters k_b and α, yields good results for any campaign magnified from our *Spamhaus* dataset. We also show that the computation of k can be performed in the same way for different types of transaction logs. To this end, we study how BotMagnifier can be used to analyze netflow records.

5 Spam Attribution

Once the magnification process has completed, we merge the IP addresses from the seed pool and the magnification set to obtain a *campaign set*. We then apply several heuristics to reduce false positives and to assign the different campaign sets to specific botnets. Note that the labeling of the campaign sets does not affect the results of the bot magnification process. BOTMAGNIFIER could be used in the wild for bot detection without these attribution functionalities. It is relevant only for tracking the populations of known botnets, as we discuss in Section 6.2.

5.1 Spambot Analysis Environment

The goal of this phase is to understand the behavior of current spamming botnets. That is, we want to determine the types of spam messages sent by a specific botnet at a certain point in time. To this end, we have built an environment that enables us to execute bot binaries in a controlled setup similarly to previous studies [11, 39].

Our spambot analysis environment is composed of one physical system hosting several virtual machines (VMs), each of which executes one bot binary. The VMs have full network access so that the bots can connect to the C&C server and receive spam-related configuration data, such as spam templates or batches of email addresses to which spam should be sent. However, we make sure that no actual spam emails are sent out by sinkholing spam traffic, i.e., we redirect outgoing emails to a mail server under our control. This server is configured to record the messages, without relaying them to the actual destination. We also prevent other kinds of malicious traffic (e.g., scanning or exploitation attempts) through various firewall rules. Some botnets (e.g., *MegaD*) use TCP port 25 for C&C traffic, and, therefore, we need to make sure that such bots can still access the C&C server. This is implemented by firewall rules that allow C&C traffic through, but prevent outgoing spam. Furthermore, botnets such as *Rustock* detect the presence of a virtual environment and refuse to run. Such samples are executed on a physical machine configured with the same network restrictions. To study whether bots located in different countries show a unique behavior, we run each sample at two distinct locations: one analysis environment is located in the United States, while the other one is located in Europe. In our experience, this setup enables us to reliably execute known spambots and observe their current spamming behavior.

For this study, we analyzed the five different bot families that were the most active during the time of our experiments: *Rustock* [5], *Lethic*, *MegaD* [4, 6], *Cutwail* [35], and *Waledac*. We ran our samples from July 2010 to February 2011. Some of the spambots we ran sent out spam emails for a limited amount of time (typically, a couple of weeks), and then lost contact with their controllers. We periodically substituted such bots with newer samples. Other bots (e.g., *Rustock*) were active for most of the analysis period.

5.2 Botnet Tags

After monitoring the spambots in a controlled environment, we attempt to assign botnet labels to spam emails found in our spam trap. Therefore, we first extract the subject templates from the emails that were collected in the analysis environment with the same technique described in Section 2.1. Then, we compare the subject templates with the emails we received in the spam trap during that same day. If we find a match, we tag the campaign set that contains the IP address of the bot that sent the message with the corresponding botnet name. Otherwise, we keep the campaign set unlabeled.

5.3 Botnet Clustering

As noted above, we ran five spambot families in our analysis environment. Of course, it is possible that one of the monitored botnets is carrying out more campaigns than those observed by analyzing the emails sent by the bots we execute in our analysis environment. In addition, we are limited by the fact that we cannot run all bot binaries in the general case (e.g., due to newly emerging botnets or in cases where we do not have access to a sample), and, thus, we cannot collect information about such campaigns. The overall effect of this limitation is that some campaign sets may be left unlabeled.

The goal of the botnet clustering phase is to determine whether an unlabeled campaign set belongs to one of the botnets we monitored. If an unlabeled campaign set cannot be associated with one of the existing labeled campaign sets, then we try to see if it can be merged with another unlabeled campaign set, which, together, might represent a new botnet.

In both cases, there is a need to determine if two campaign sets are "close" enough to each other in order to be considered as part of the same botnet. In order to represent the distance between campaign sets, we developed three metrics, namely an IP overlap metric, a destination distance metric, and a bot distance metric.

IP overlap. The observation underlying the IP overlap metric is that two campaign sets sharing a large number of bots (i.e., common IP addresses) likely belong to the same botnet. It is important to note that infected machines can belong to multiple botnets, as one machine may be infected with two distinct instances of malware. Another factor one needs to take into account is network address translation (NAT) gateways, which can potentially hide large networks behind them. As a result, the IP address of a NAT gateway might appear as part of multiple botnets. However, a host is discarded from the campaign set related to p_i as soon as it contacts a destination that is *not* in the target set (see Section 4 for a discussion). Therefore, NAT gateways are likely to be discarded from the candidate set early on: at some point, machines behind the NAT will likely hit two destinations that are unique to two different seed pools, and, thus, will be discarded from all campaign sets. This might not be true for small NATs, with just a few hosts behind them. In this case, the IP address of the gateway would be detected as a bot by BOTMAGNIFIER. In a real world scenario, this would still be useful information for the network administrator, who would know what malware has likely infected one or more of her hosts.

Given these assumptions, we merge two campaign sets with a large IP overlap. More precisely, first the intersection of the two campaign sets is computed. Then, if such intersection represents a sufficiently high portion of the IP addresses in *either* of the campaign sets, the two campaign sets are merged.

The fraction of IP addresses that need to match either of the campaign sets to consider them to be part of the same botnet varies with the size of the sets for those campaigns. Intuitively, two small campaigns will have to overlap by a larger percentage than two large campaigns in order to be considered as part of the same botnet. This is done to avoid merging small campaigns together just based on a small number of IP addresses that might be caused by multiple infections or by two different spambots hiding behind a small NAT. Given a campaign c, the fraction of IP addresses that has to overlap with another campaign in order to be merged together is

$$O_c = \frac{1}{\log_{10}(N_c)}, \qquad (6)$$

where N_c is the number of hosts in the campaign set. We selected this equation because the denominator increases slowly with the number of bots carrying out a campaign. Moreover, because of the use of the logarithm, this equation models an exponential decay, which decreases fast for small values of N_c, and much more slowly for large values of it. Applying this equation, a campaign carried out by 100 hosts will require an overlap of 50% or more to be merged with another one, while a campaign carried out by 10,000 hosts will only require an overlap of 25%. When comparing two campaigns c_1 and c_2, we require the smaller one to have an overlap of at least O_c with the largest one to consider them as being carried out by the same botnet.

Destination distance. This technique is an extension of our magnification step. We assume that bots carrying out the same campaign will target the same destinations. However, as mentioned previously, some botnets send spam only to specific countries during a given time frame. Leveraging this observation, it is possible to find out whether two campaign sets are likely carried out by the same botnet by observing the country distribution of the set of destinations they targeted. More precisely, we build a *destination country vector* for each campaign set. Each element of the destination country vector corresponds to the fraction of destinations that belong to a specific country. We determined the country of each IP address using the GEOIP tool [19]. Then, for each pair of campaign sets, we calculate the *cosine distance* between them.

We performed a *precision* versus *recall* analysis to develop an optimal threshold for this clustering technique. By precision, we mean how well this technique can discriminate between campaigns belonging to different botnets. By recall, we capture how well the technique can cluster together campaigns carried out by the same botnet. We ran our analysis on 50 manually-labeled campaigns picked from the ones sent by the spambots in our analysis environment. Similarly to how we found the optimal value of k in Section 4, we multiply precision and recall together. We then searched for the threshold value that maximizes this product. In our experiments, we found that the cosine distance of the destination countries vectors is rarely lower than 0.8. This occurs regardless of the particular country distribution of a campaign, because there will be a significant amount of bots in large countries (e.g., the United States or India). The *precision* versus *recall* analysis showed that 0.95 is a good threshold for this clustering technique.

Bot distance. This technique is similar to the destination distance, except that it utilizes the country distribution of the bot population of the campaign set instead of the location of the targeted servers. For each campaign set, we build a *source country vector* that contains the fraction of bots for a given country.

The intuition behind this technique comes from the fact that malware frequently propagates through malicious web sites, or through legitimate web servers that have been compromised [24, 34]. These sites will not have a uniform distribution of users (e.g., a Spanish web site will mostly have visitors from Spanish-speaking countries) and, therefore, the distribution of compromised users in the world for that site will not be uniform. For this technique, we also performed a *precision* versus *recall* analysis, in the same way as for the destination distance technique. Again, we experimentally found the optimal threshold to be 0.95.

6 Evaluation

To demonstrate the validity of our approach, we first examined the results generated by BOTMAGNIFIER when magnifying the population of a large spamming botnet for which we have ground truth knowledge (i.e., we know which IP addresses belong to the botnet). Then, we ran the system for a period of four months on a large set of real-world data, and we successfully tracked the evolution of large botnets.

6.1 Validation of the Approach

To validate our approach, we studied a botnet for which we had direct data about the number and IP addresses of the infected machines. More precisely, in August 2010, we obtained access to thirteen C&C servers belonging to the *Cutwail* botnet [35]. Note that we only used nine of them for this evaluation, since two had already been used to derive the optimal value of N in Section 4, and two were not actively sending spam at the time of the takedown. As discussed before, these C&C servers contained detailed information about the infected machines belonging to the botnet and the spam campaigns carried out. The whole botnet was composed of 30 C&C servers. By analyzing the data on the C&C servers we had access to, we found that, during the last day of operation, 188,159 bots contacted these nine servers. Of these, 37,914 ($\approx$ 20%) contacted multiple servers. On average, each server controlled 20,897 bots at the time of the takedown, with a standard deviation of 5,478. Based on these statistics, the servers to which we had access managed the operations of between 29% and 37% of the entire botnet. We believe the actual percentage of the botnet controlled by these servers was close to 30%, since all the servers except one were contacted by more than 19,000 bots during the last day of operation. Only a single server was controlling less than 10,000 bots. Therefore, it is safe to assume that the vast majority of the command and control servers were controlling a similar amount of bots ($\approx$ 20,000 each).

We ran the validation experiment for the period between July 28 and August 16, 2010. For each of the 18 days, we first selected a subset of the IP addresses referenced by the nine C&C servers. As a second step, with the help of the spam trap, we identified which campaigns had been carried out by these IP address during that day. Then, we generated seed and magnified pools. Finally, we compared the output magnification sets against the ground truth (i.e., the other IP addresses referenced by the C&C servers) to assess the quality of the results.

Overall, BOTMAGNIFIER identified 144,317 IP addresses as *Cutwail* candidates in the campaign set. Of these, 33,550 ($\approx$ 23%) were actually listed in the C&C servers' databases as bots. This percentage is close to the fraction of the botnet we had access to (since we considered 9 out of 30 C&C servers), and, thus, this result suggests that the magnified population identified by our system is consistent. To perform a more precise analysis, we ran BOTMAGNIFIER and studied the magnified pools that were given as an output on a daily basis. The average size of the magnified pools was 4,098 per day. In total, during the 18 days of the experiment, we grew the bot population by 73,772 IP addresses. Of the IP addresses detected by our tool, 17,288 also appeared in the spam trap during at least one other day of our experiment, sending emails belonging to the same campaigns carried out by the C&C servers. This confirms that they were actually *Cutwail* bots. In particular, 3,381 of them were detected by BOTMAGNIFIER *before* they ever appeared in the spam trap, which demonstrates that we can use our system to detect bots before they even hit our spam trap.

For further validation, we checked our results against the *Spamhaus* database, to see if the IP addresses we identified as bots were listed as known spammers or not. 81% were listed in the blacklist.

We then tried to evaluate how many of the remaining 27,421 IP addresses were false positives. To do this, we used two techniques. First, we tried to connect to the host to check whether it was a legitimate server. Legitimate SMTP or DNS servers can show up in queries on *Spamhaus* due to several reasons (e.g., in cases where reputation services collect information about sender IP addresses or if an email server is configured to query the local DNS server). Therefore, we tried to determine if an IP address that was not blacklisted at the time of the experiment was a legitimate email or DNS server by connecting to port 25 TCP and 53 UDP. If the server responded, we considered it to be a false positive. Unfortunately, due to firewall rules, NAT gateways, or network policies, some servers might not respond to our probes. For this reason, as a second technique, we executed a reverse DNS lookup on the IP addresses, looking for evidence showing that the host was a legitimate server. In particular, we looked for strings that are typical for mail servers in the hostname. These strings are *smtp*, *mail*, *mx*, *post*, and *mta*. We built this list by manually looking at the reverse DNS lookups of the IP address that were not blacklisted by Spamhaus. If the reverse lookup matched one of these strings, we considered the IP address as a legitimate server, i.e., a false positive. In total, 2,845 IP addresses resulted in legitimate servers (1,712 SMTP servers and 1,431 DNS servers), which is 3.8% of the overall magnified population.

We then tried to determine what coverage of the entire *Cutwail* botnet our approach produced. Based on the number of active IP addresses per day we saw on the C&C servers, we estimated that the size of the botnet at the time of the takedown was between 300,000 and 400,000 bots. This means that, during our experiment, we were able to track between 35 and 48 percent of the botnet. Given the limitations of our transaction log (see Section 6.2.1), this is a good result, which could be improved by getting access to multiple *Spamhaus* servers or more complete data streams.

6.2 Tracking Bot Populations

To demonstrate the practical feasibility of our approach, we used BOTMAGNIFIER to track bot populations in the wild for a period of four months. In particular, we ran the system for 114 days between September 28, 2010 and February 5, 2011. We had a downtime of about 15 days in November 2011, during which the emails of the spam trap could not be delivered.

By using our magnification algorithm, our system identified and tracked 2,031,110 bot IP addresses during the evaluation period. Of these, 925,978 IP addresses ($\approx 45.6\%$) belonged to magnification sets (i.e., they were generated by the magnification process), while 1,105,132 belonged to seed pools generated with the help of the spam trap.

6.2.1 Data Streams Limitations

The limited view we have from the transaction log generated by only one DNSBL mirror limits the number of bots we can track each day. BOTMAGNIFIER requires an IP address to appear a minimum number of times in the transaction log, in order to be considered as a potential bot. From our DNSBL mirror, we observed that a medium size campaign targets about 50,000 different destination servers (i.e., $|T(p_i)|$ = 50,000). The value of N for such a campaign, calculated using equation 5, is 50. On an average day, our DNSBL mirror logs activity performed by approximately 4.7 million mail senders. Of these, only about 530,000 ($\approx 11\%$) appear at least 50 times. Thus, we have to discard a large number of potential bots *a priori*, because of the limited number of transactions our *Spamhaus* mirror observes. If we had access to more transaction logs, our visibility would increase, and, thus, the results would improve accordingly.

6.2.2 Overview of Tracking Results

For each day of analysis, BOTMAGNIFIER identified the largest spam campaigns active during that day (Section 2), learned the behavior of a subset of IP addresses carrying out those campaigns (Section 3), and grew a population of IP addresses behaving in the same way (Section 4). This provided us with the ability to track the population of the largest botnets, monitoring how active they were, and determining which periods they were silent.

A challenging aspect of tracking botnets with BOTMAGNIFIER has been assigning the right label to the various spam campaigns (i.e., the name of the botnet that generated them). Tagging the campaigns that we observed in our honeypot environment was trivial, while for the others we used the clustering techniques described in Section 5. In total, we observed 1,475 spam campaigns. We tried to assign a botnet label to each cluster, and every time two clusters were assigned the same label, we merged them together. After this process, we obtained 38 clusters. Seven of them were large botnets, which generated 50,000 or more bot IP addresses in our magnification results. The others were either smaller botnets, campaigns carried out by dedicated servers (i.e., not carried out by botnets), or errors produced by the clustering process.

We could not assign a cluster to 107 campaigns ($\approx 7\%$ of all campaigns), and we magnified these campaigns independently from the others. Altogether, the magnified sets of these campaigns accounted for 20,675 IP addresses ($\approx 2\%$ of the total magnified hosts). We then studied the evolution over time and the spamming capabilities of the botnets we were able to label.

6.2.3 Analysis of Magnification Results

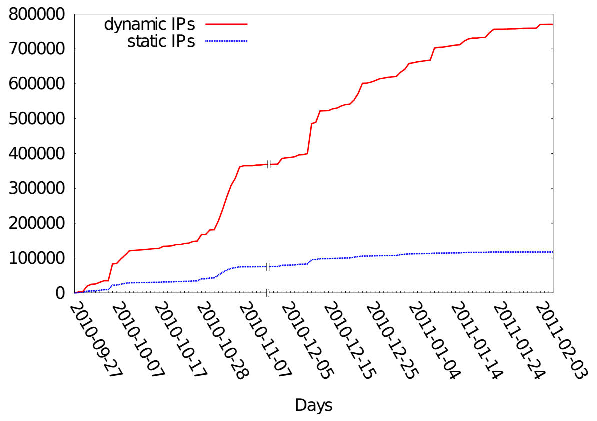
Table 1 shows some results from our tracking. For each botnet, we list the number of IP addresses we obtained from the magnification process. Interestingly, *Lethic*, with 887,852 IP addresses, was the largest botnet we found. This result is in contrast with the common belief in the security community that, at the time of our experiment, *Rustock* was the largest botnet [18]. However, from our observation, *Rustock* bots appeared to be more aggressive in spamming than the *Lethic* bots. In fact, each *Rustock* bot appeared, on average, 173 times per day on our DNSBL mirror logs, whereas each *Lethic* bot showed up only 101 times.

For each botnet population we grew, we distinguished between static and dynamic IP addresses. We considered an IP address as dynamic if, during the testing period, we observed that IP address only once. On the other hand, if we observed the same IP address multiple times, we consider it as static. The fraction of static versus dynamic IP addresses for the botnets we tracked goes from 15% for *Rustock* to 4% for *MegaD*. Note that smaller botnets exceeded the campaign size thresholds required by BOTMAGNIFIER (see Section 5) less often than larger botnets, and therefore it is possible that our system underestimates the number of IP addresses belonging to the *MegaD* and *Waledac* botnets.

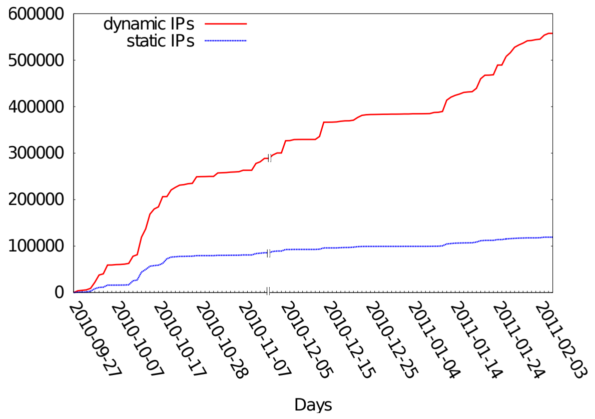
Figures 2(a) and 2(b) show the growth of IP addresses over time for the magnification sets belonging to *Lethic* and *Rustock* (note that we experienced a downtime of the system during November 2010). The figures show that dynamic IP addresses steadily grow over time, while static IP addresses reach saturation after some time. Furthermore, it is interesting to notice that we did not observe much *Rustock* activity between December 24, 2010 and January 10, 2011. Several sources reported that the botnet was (almost) down during this period [14, 37]. BOTMAGNIFIER confirms this downtime of the botnet, which indicates that our approach can effectively track the activity of botnets. After the botnet went back up

Botnet	**Total # of IP addresses**	**# of dynamic IP addresses**	**# of static IP addresses**	**# of events per bot** (per day)
Lethic	887,852	770,517	117,335	101
Rustock	676,905	572,445	104,460	173
Cutwail	319,355	285,223	34,132	208
MegaD	68,117	65,062	3,055	112
Waledac	36,058	32,602	3,450	140

Table 1: Overview of the BotMagnifier results



(a) Growth of *Lethic* IP addresses



(b) Growth of *Rustock* IP addresses

Figure 2: Growth of the dynamic and static IP address populations for the two major botnets

again in January 2011, we observed a steady growth in the number of *Rustock* IP addresses detected by BotMagnifier.

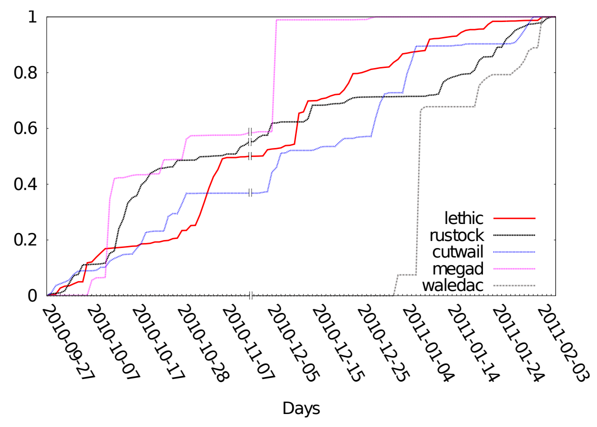
Figures 3(a) and 3(b) show the cumulative distribution functions of dynamic IP addresses and static IP addresses tracked during our experiment for the five largest botnets. It is interesting to see that we started observing campaigns carried out by *Waledac* on January 1, 2011. This is consistent with the reports from several sources, who also noticed that a new botnet appeared at the same time [17, 31]. We also observed minimal spam activities associated with *MegaD* after December 7, 2011. This was a few days after the botmaster was arrested [30].

6.3 Application of Results

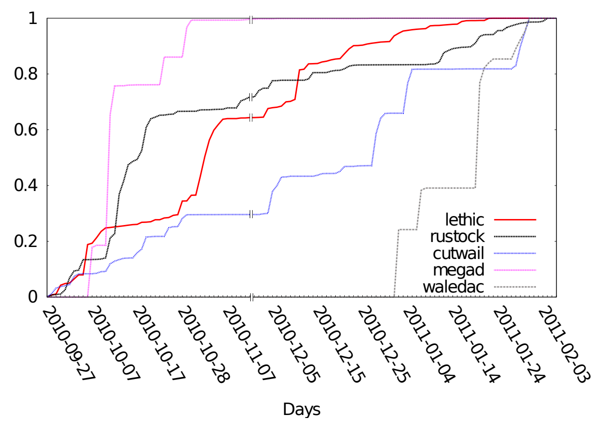
False positives. In Section 4, we showed how the parameter k minimizes the ratio between true positives and false positives. We initially tolerated a small number of false positives because these do not affect the big picture of tracking large botnet populations. However, we want to quantify the false positive rate of the results, i.e., how many of the bot candidates are actually legitimate machines. This information is important, especially if BotMagnifier is used to inform Internet Service Providers or other organizations about infected machines. Furthermore, if we want to use the results to improve spam filtering systems, we need to be very careful about which IP addresses we consider as bots. We use the same techniques outlined in Section 6.1 to check for false positives. We remove each IP address that matches any of these techniques from the magnified sets.

We ran this false positive detection heuristic on all the magnified IP addresses identified during the evaluation period. This resulted in 35,680 ($\approx$1.6% of the total) IP addresses marked as potential false positives. While this might sound high at first, we also need to evaluate how relevant this false positive rate is in practice: our results can be used to augment existing systems and thus we can tolerate a certain rate of false positives. In addition, while deploying BotMagnifier in a production system, one could add a filter that applies the techniques from Section 6.1 to any magnified pool, and obtain clean results that he could use for spam reduction.

Improving existing blacklists. We wanted to understand whether our approach can improve existing blacklists by providing information about spamming bots that are currently active. To achieve this, we analyzed the email logs from the UCSB computer science department over a period of two months, from November 30, 2010 to February 8, 2011. As a first step, the department mail server uses *Spamhaus* as a pre-filtering mechanism, and therefore the majority of the spam gets blocked before



(a) CDF for the dynamic IP addresses



(b) CDF for the static IP addresses

Figure 3: Cumulative Distribution Function for the bot populations grown by BOTMAGNIFIER

being processed. For each email whose sender is not blacklisted, the server runs *SpamAssassin* [3] for content analysis, to find out if the message content is suspicious. *SpamAssassin* assigns a spam score to each message, and the server flags it as spam or ham according to that value. These two steps are useful to evaluate how BOTMAGNIFIER performs, for the following reasons:

- If a mail reached the server during a certain day, it means that at that time its sender was not blacklisted by *Spamhaus*.
- The spam ratios computed by *SpamAssassin* provide a method for the evaluation of BOTMAGNIFIER's false positives.

During the analysis period, the department mail server logged 327,706 emails in total, sent by 228,297 distinct IP addresses. Of these, 28,563 emails were considered as spam by *SpamAssassin*, i.e., they bypassed the first filtering step based on *Spamhaus*. These mails had been sent by 10,284 IP addresses. We compared these IP addresses with the magnified sets obtained by BOTMAGNIFIER during the same period: 1,102 ($\approx$ 10.8%) appeared in the magnified sets. We then evaluated how many of these IP addresses would have been detected before reaching the server if our tool would have been used in parallel with the DNSBL system. To do this, we analyzed how many of the spam sender IP addresses were detected by BOTMAGNIFIER before they sent spam to our server. We found 295 IP addresses showing this behavior. All together, these hosts sent 1,225 emails, which accounted for 4% of the total spam received by the server during this time.

We then wanted to quantify the false positives in the magnified pools generated by BOTMAGNIFIER. To do this, we first searched for those IP addresses that were in one of the magnification pools, but had been considered sending ham by *SpamAssassin*. This resulted in 28 matches. Of these, 15 were blacklisted by *Spamhaus* when we ran the tests, and therefore we assume they are false negatives by *SpamAssassin*. Of the remaining 13 hosts, 12 were detected as legitimate servers by the filters described in Section 6.1. For the remaining one IP address, we found evidence of it being associated with spamming behavior on another blacklist [23]. We therefore consider it as a false negative by *SpamAssassin* as well.

In summary, we conclude that BOTMAGNIFIER can be used to improve the spam filtering on the department email server: the server would have been reached by 4% less spam mails, and no legitimate emails would have been dropped by mistake within these two months. Having access to more *Spamhaus* mirrors would allow us to increase this percentage.

Resilience to evasion. If the techniques introduced by BOTMAGNIFIER become popular, spammers will modify their behavior to evade detection. In this section, we discuss how we could react to such evasion attempts.

The first method that could be used against our system is obfuscating the email subject lines, to prevent BOTMAGNIFIER from creating the seed pools. If this was the case, we could leverage previous work [22,40] that takes into account the body of emails to identify emails that are sent by the same botnet. As an alternative, we could use different methods to build the seed pools, such as clustering bots based on the IPs of the C&C servers that they contact.

Another evasion approach spammers might try is to reduce the number of bots associated with each campaign. The goal would be to stay under the threshold required by BOTMAGNIFIER (i.e., 1,000) to work. This would require more management effort on the botmaster's side, since more campaigns would need to be run. Moreover, we could use other techniques to cluster the spam campaigns. For example, it is unlikely that the spammers would set up a different website for each of the small campaigns they create. We could then cluster the campaigns by looking at the web sites the URLs in the spam emails point to.

Other evasion techniques might be to assign a single domain to each spamming bot, or to avoid evenly distributing email lists among bots. In the first case, BotMagnifier would not be able to unequivocally identify a bot as being part of a specific botnet. However, the attribution requirement could be dropped, and these bots would still be detected as generic spamming bots. The second case would be successful in evading our current systems. However, this behavior involves something that spammers want to avoid: having the same bot sending thousands of emails to the same domain within a short amount of time would most likely result in the bot being quickly blacklisted.

6.4 Universality of k

In Section 4, we introduced a function to determine the optimal N value according to the size of the seed pool's target $|T(p_i)|$. To do this, we analyzed the data from two C&C servers of the *Cutwail* botnet. One could argue that this parameter will work well only for campaigns carried out by that botnet. To demonstrate that the value of k (and subsequently of N) estimated by the function produces good results for campaigns carried out by other botnets, we ran the same precision versus recall technique we used in Section 4 on other datasets. Specifically, we analyzed 600 campaigns observed in the wild, that had been carried out by the other botnets we studied (*Lethic*, *Rustock*, *Waledac*, and *MegaD*). Since we did not have access to full ground truth for these campaigns, we used the IP addresses from the seed pools as true positives, and the set of IP addresses not blacklisted by *Spamhaus* as false positives. For the purpose of this analysis, we ignored any other IP address returned by the magnification process (i.e., magnified IP addresses already blacklisted by *Spamhaus*).

The results are shown in Figure 4. The figure shows the function plot of k in relation to the size of $|T(p_i)|$. The dots show, for each campaign we analyzed, where the optimal value of k lies. As it can be seen, the function of k we used approximates the optimal values for most campaigns well. This technique for setting k might also be used to set up BotMagnifier in the wild, when ground truth is not available.

Data stream independence. In Section 2.2, we claimed that BotMagnifier can work with any kind of transaction log as long as this dataset provides information about which IP addresses sent email to which destination email servers at a given point in time. To confirm this claim, we ran BotMagnifier on an alternative dataset, extracted from *netflow records* [7] collected by the routers of a large Internet service provider. The netflow data is collected with a sampling rate of 1 out of 1,000. To extract the data in a format BotMagnifier understands, we extracted each connection directed to port 25 TCP, and considered the timestamp in which the connection initiated as the time the email was sent. On average, this transaction log contains 1.9 million entries per day related to about 194,000 unique sources.

To run BotMagnifier on this dataset, we first need to correctly dimension k. As explained in Section 4, the equation for k is stable for any transaction log. However, the value of the constants k_b and α changes for each dataset. To correctly dimension these parameters, we ran BotMagnifier on several campaigns extracted from the netflow records. The $PR(k)$ analysis is shown in Figure 5. The optimal point of the campaigns is located at a lower k for this dataset compared to the ones analyzed in Section 4. To address this difference, we set k_b to 0.00008 and α to 1 when dealing with netflow records as transaction logs. After setting these param-

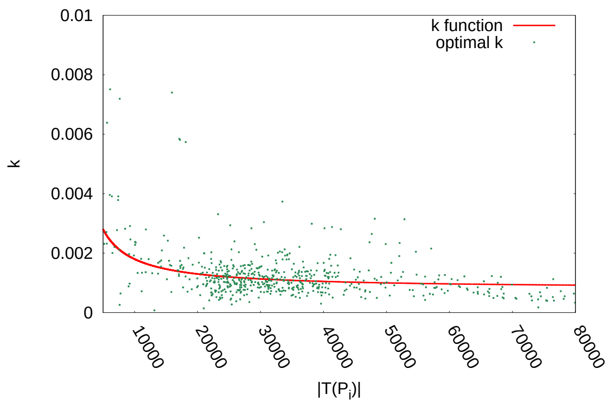


Figure 4: Analysis of our function for k compared to the optimal value of k for 600 campaigns

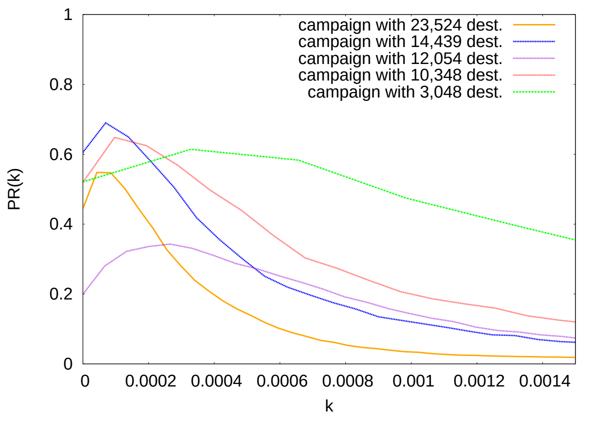


Figure 5: Precision vs. Recall functions for five campaigns observed in the netflow dataset

eters, we analyzed one week of data with BotMagnifier. The analysis period was between January 20 and January 28, 2011. During this period, we tracked 94,894 bots. Of these, 36,739 ($\approx$ 38.7%) belonged to the magnified sets of the observed campaigns. In particular, we observed 40,773 *Rustock* bots, 20,778 *Lethic* bots, 6,045 *Waledac* bots, and 1,793 *Cutwail* bots.

7 Related Work

Spam is one of the major problems on the Internet, and as a result, has attracted a considerable amount of research. In this section, we briefly review related work in this area and discuss the novel aspects of BotMagnifier.

Botnet Tracking. A popular method to gain deeper insights into a particular botnet is *botnet tracking*, i.e., an attempt to learn more about a given botnet by analyzing its inner workings in detail [1, 8]. There are several approaches to conduct the actual analysis, for example by taking over the C&C infrastructure and then performing a live analysis [26,33]. An orthogonal approach is to take down the C&C server and perform an offline analysis of the server to reconstruct information [21]. A less invasive approach is to (automatically) reverse-engineer the communication protocol used by the botnet and then impersonate a bot [4, 6, 15, 32]. This enables a continuous collection of information about the given botnet, e.g., to gather the spam templates used by the bots [6].

BotMagnifier complements these approaches: we are able to track spamming botnets on the Internet in a non-invasive way from a novel vantage point. The information generated by our tool enables us to perform a high-level study of botnets. For example, we can track their size and evolution over time, and obtain a live view of hosts that belong to a particular botnet.

Ramachandran et al. also analyzed queries against a DNSBL to reveal botnet memberships [27], but their motivation is completely different from ours: the intuition behind their approach is that bots might check if their own IP address is blacklisted by a given DNSBL. Such queries can be detected, which discloses information about infected machines. BotMagnifier is complementary with respect to this approach because it analyzes intrinsic traces left by spamming machines (i.e., an email server will query the DNSBL for information), and clustering and enriching this data enables us to find spambots in a generic way. Furthermore, we demonstrated that our approach can also be used on other kinds of transaction logs.

Spam Studies. Several studies analyzed spam and the side-effects of this business [2, 12, 16, 42, 43]. BotLab [11], a tool to correlate incoming spam mails with outgoing spam collected by executing known bots in an analysis environment, shares some characteristics with our approach. The analysis results of BotLab can approximate the relative size of different spamming botnets and provide insights into current spam campaigns based on the information collected at the site running the tool. In contrast, BotMagnifier enables us to detect IP addresses of hosts that belong to spamming botnets at an Internet-wide level. We use the analysis environment only to collect information that enables us to assign labels to spam campaigns, while all other analysis techniques (e.g., the DNSBL analysis) are different compared to BotLab.

Another system that shares some similarities with our approach is AutoRE [40], which examines content-level features in the email body such as URLs to group spam messages into campaigns. The authors performed a large-scale evaluation based on mail messages collected by a large webmail provider to generate signatures to detect polymorphic modifications for individual spam campaigns. Xie et al. also examined characteristics of the spam campaigns, similar to our work. In contrast, our approach focuses primarily on the behavioral similarities between members of a spamming botnet, without requiring knowledge of the actual spam content.

Spam Mitigation. The typical approaches to detect spam either focus on the content of spam messages [3, 22, 40] or on the analysis of network-level features [10, 25, 26, 28, 29, 38]. BotMagnifier generates lists of IP addresses that belong to spamming botnets, which complements both kinds of approaches: the analysis results can be used to improve systems that use network-level features to detect spambots, e.g., by proactively listing such IP addresses in blacklists, or complement existing systems, as demonstrated in Section 6.3. Furthermore, the information can be used to notify ISPs about infected customers within their networks.

8 Conclusion

We presented BotMagnifier, a tool for tracking and analyzing spamming botnets. The tool is able to “magnify” an initial seed pool of spamming IP addresses by learning the behavior of known spamming bots and matching the learned patterns against a (partial) log of the email transactions carried out on the Internet. We have validated and evaluated our approach on a number of datasets (including the ground truth data from a botnet’s C&C hosts), showing that BotMagnifier is indeed able to accurately identify and track botnets.

Future work will focus on finding new data inputs that can either populate our initial seed pools or on obtaining a different, more comprehensive transaction log to be able to identify spamming bots more comprehensively.

Also, analyzing larger data streams might allow us to apply more features for our magnification process, producing more complete results.

Acknowledgments

This work was supported by the ONR under grant N000140911042, the National Science Foundation (NSF) under grants CNS-0845559 and CNS-0905537, and the Ministry of Economic Affairs and Energy of the State of North Rhine-Westphalia (grant 315-43-02/2-005-WFBO-009) as well as Secure Business Austria. We also thank our shepherd Tara Whalen and the anonymous reviewers for their valuable insights and comments.

References

[1] M. Abu Rajab, J. Zarfoss, F. Monrose, and A. Terzis. A Multifaceted Approach to Understanding the Botnet Phenomenon. In *ACM SIGCOMM Conference on Internet Measurement*, 2006.

[2] D. S. Anderson, C. Fleizach, S. Savage, and G. M. Voelker. Spamscatter: Characterizing Internet Scam Hosting Infrastructure. In *USENIX Security Symposium*, 2007.

[3] Apache Foundation. Spamassassin. `http://spamassassin.apache.org`.

[4] J. Caballero, P. Poosankam, C. Kreibich, and D. Song. Dispatcher: Enabling Active Botnet Infiltration Using Automatic Protocol Reverse-engineering. In *ACM Conference on Computer and Communications Security (CCS)*, 2009.

[5] K. Chiang and L. Lloyd. A Case Study of the Rustock Rootkit and Spam Bot. In *USENIX Workshop on Hot Topics in Understanding Botnet*, 2007.

[6] C. Cho, J. Caballero, C. Grier, V. Paxson, and D. Song. Insights from the Inside: A View of Botnet Management from Infiltration. In *USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)*, 2010.

[7] Cisco Inc. Cisco IOS NetFlow. `https://www.cisco.com/en/US/products/ps6601/products_ios_protocol_group_home.html`.

[8] F. C. Freiling, T. Holz, and G. Wicherski. Botnet Tracking: Exploring a Root-Cause Methodology to Prevent Distributed Denial-of-Service Attacks. In *European Symposium on Research in Computer Security (ESORICS)*, 2005.

[9] G. Gu, R. Perdisci, J. Zhang, and W. Lee. BotMiner: Clustering Analysis of Network Traffic for Protocol- and Structure-independent Botnet Detection. In *USENIX Security Symposium*, 2008.

[10] S. Hao, N. A. Syed, N. Feamster, A. G. Gray, and S. Krasser. Detecting Spammers with SNARE: Spatio-temporal Network-level Automatic Reputation Engine. In *USENIX Security Symposium*, 2009.

[11] J. P. John, A. Moshchuk, S. D. Gribble, and A. Krishnamurthy. Studying Spamming Botnets Using Botlab. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2009.

[12] C. Kanich, C. Kreibich, K. Levchenko, B. Enright, G. Voelker, V. Paxson, and S. Savage. Spamalytics: An Empirical Analysis of Spam Marketing Conversion. In *ACM Conference on Computer and Communications Security (CCS)*, 2008.

[13] M. Khmartseva. Email Statistics Report. `http://www.radicati.com/wp/wp-content/uploads/2009/05/email-stats-report-exec-summary.pdf`, 2009.

[14] B. Krebs. Taking Stock of Rustock. `http://krebsonsecurity.com/2011/01/taking-stock-of-rustock/`, 2011.

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

[16] C. Kreibich, C. Kanich, K. Levchenko, B. Enright, G. M. Voelker, V. Paxson, and S. Savage. Spamcraft: An Inside Look at Spam Campaign Orchestration. In *USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)*, 2009.

[17] A. Lelli. Return from the Dead: Waledac/Storm Botnet Back on the Rise. `http://www.symantec.com/connect/blogs/return-dead-waledacstorm-botnet-back-rise`, 2011.

[18] M86. Rustock, the king of spam. `http://www.m86security.com/labs/traceitem.asp?article=1362`, July 2010.

[19] MaxMind. GeoIP. `http://www.maxmind.com/app/ip-location`.

[20] MessageLabs. MessageLabs Intelligence: 2010 Annual Security Report. `http://www.messagelabs.com/mlireport/MessageLabsIntelligence_2010_Annual_Report_FINAL.pdf`, 2010.

[21] C. Nunnery, G. Sinclair, and B. B. Kang. Tumbling Down the Rabbit Hole: Exploring the Id-

iosyncrasies of Botmaster Systems in a Multi-Tier Botnet Infrastructure. In *USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)*, 2010.

[22] A. Pitsillidis, K. Levchenko, C. Kreibich, C. Kanich, G. M. Voelker, V. Paxson, N. Weaver, and S. Savage. Botnet Judo: Fighting Spam with Itself. In *Symposium on Network and Distributed System Security (NDSS)*, 2010.

[23] Project Honeypot. `http://www.projecthoneypot.org/`.

[24] N. Provos, P. Mavrommatis, M. A. Rajab, and F. Monrose. All your iFRAMEs point to Us. In *USENIX Security Symposium*, 2008.

[25] Z. Qian, Z. Mao, Y. Xie, and F. Yu. On Network-level Clusters for Spam Detection. In *Symposium on Network and Distributed System Security (NDSS)*, 2010.

[26] A. Ramachandran and N. Feamster. Understanding the Network-level Behavior of Spammers. *SIGCOMM Comput. Commun. Rev.*, 36, August 2006.

[27] A. Ramachandran, N. Feamster, and D. Dagon. Revealing Botnet Membership using DNSBL Counter-intelligence. In *USENIX Workshop on Steps to Reducing Unwanted Traffic on the Internet (SRUTI)*, 2006.

[28] A. Ramachandran, N. Feamster, and S. Vempala. Filtering Spam with Behavioral Blacklisting. In *ACM Conference on Computer and Communications Security (CCS)*, 2007.

[29] M. B. S. Sinha and F. Jahanian. Improving Spam Blacklisting Through Dynamic Thresholding and Speculative Aggregation. In *Symposium on Network and Distributed System Security (NDSS)*, 2010.

[30] SC Magazine. Accused MegaD operator arrested. `http://www.scmagazineus.com/accused-mega-d-botnet-operator-arrested`, 2011.

[31] Shadowserver. New fast flux botnet for the holidays, 2011.

[32] B. Stock, J. Gobel, M. Engelberth, F. Freiling, and T. Holz. Walowdac Analysis of a Peer-to-Peer Botnet. In *European Conference on Computer Network Defense (EC2ND)*, 2009.

[33] B. Stone-Gross, M. Cova, L. Cavallaro, B. Gilbert, M. Szydlowski, R. Kemmerer, C. Kruegel, and G. Vigna. Your Botnet is My Botnet: Analysis of a Botnet Takeover. In *ACM Conference on Computer and Communications Security (CCS)*, 2009.

[34] B. Stone-Gross, M. Cova, C. Kruegel, and G. Vigna. Peering Through the iFrame. In *IEEE Conference on Computer Communications (INFOCOM)*, 2011.

[35] B. Stone-Gross, T. Holz, G. Stringhini, and G. Vigna. The Underground Economy of Spam: A Botmaster's Perspective of Coordinating Large-Scale Spam Campaigns. In *USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)*, 2011.

[36] Symantec Corp. State of spam & phishing report, 2010.

[37] Symantec. Corp. Rustock hiatus ends with huge surge of pharma spam. `http://www.symantec.com/connect/blogs/rustock-hiatus-ends-huge-surge-pharma-spam`, January 2011.

[38] S. Venkataraman, S. Sen, O. Spatscheck, P. Haffner, and D. Song. Exploiting Network Structure for Proactive Spam Mitigation. In *USENIX Security Symposium*, 2007.

[39] P. Wurzinger, L. Bilge, T. Holz, J. Goebel, C. Kruegel, and E. Kirda. Automatically Generating Models for Botnet Detection. In *European Symposium on Research in Computer Security (ESORICS)*, 2009.

[40] Y. Xie, F. Yu, K. Achan, R. Panigrahy, G. Hulten, and I. Osipkov. Spamming Botnets: Signatures and Characteristics. *SIGCOMM Comput. Commun. Rev.*, 38, August 2008.

[41] T.-F. Yen and M. K. Reiter. Traffic Aggregation for Malware Detection. In *Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA)*, 2008.

[42] Y. Zhao, Y. Xie, F. Yu, Q. Ke, Y. Yu, Y. Chen, and E. Gillum. BotGraph: Large Scale Spamming Botnet Detection. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2009.

[43] L. Zhuang, J. Dunagan, D. R. Simon, H. J. Wang, and J. D. Tygar. Characterizing Botnets From Email Spam Records. In *USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)*, 2008.