Router-Level Spam Filtering Using TCP Fingerprints: Architecture and Measurement-Based Evaluation

Holly Esquivel
University of Wisconsin - Madison
1210 W. Dayton St.
Madison, WI 53706-1685
esquivel@cs.wisc.edu

Tatsuya Mori
NTT Service Integration Laboratories
3-9-11 Midoricho Musashino
Tokyo, Japan 180-8585
mori.tatsuya@lab.ntt.co.jp

Aditya Akella
University of Wisconsin - Madison
1210 W. Dayton St.
Madison, WI 53706-1685
akella@cs.wisc.edu

ABSTRACT

Email spam has become costly and difficult to manage in recent years. Many of the mechanisms used for controlling spam are located at local SMTP servers and end-host machines. These mechanisms can place a significant burden on mail servers and end-host machines as the number spam messages received continues to increase. We propose a preliminary architecture that applies spam detection filtering at the router-level using light-weight signatures for spam senders. We argue for using TCP headers to develop fingerprint signatures that can be used to identify spamming hosts based on the specific operating system and version from which the email is sent. These signatures are easy to compute in a light-weight, stateless fashion. More importantly, only a small amount of fast router memory is needed to store the signatures that contribute a significant portion of spam.

We present simple heuristics and architectural enhancements for selecting signatures which result in a negligible false positive rate. We evaluate the effectiveness of our approach on data sets collected at two different vantage points simultaneously, the University of Wisconsin-Madison and a corporation in Tokyo, Japan over a one month period. We find that by targeting 100 fingerprint signatures, we can reduce the amount of received spam by 28-59% with false positive ratio less than 0.05%. Thus, our router-level approach works effectively to decrease the workload of subsequent anti-spam filtering mechanisms, such as, DNSBL look up, and content filtering. Our study also leverages the AS numbers of spam senders to discover the origin of the majority of spam seen in our data sets. This information allows us to pin-point effective network locations to place our router-level spam filters to stop spam close to the source. As a byproduct of our study, the extracted TCP fingerprints reveal signatures which originate all over the world but only send spam indicating the potential existence of global-scale spamming infrastructures.

1. INTRODUCTION

Throughout the world, Internet users are being targeted by spammers. Spam traffic is costly not only on infrastructure entities such as mail servers but also on end-users. While there are several approaches to controlling spam, many of these apply spam detection at local SMTP servers or at a user's email inbox. In this paper, we explore the role of network routers in filtering spam email. Filtering spam at routers can help reduce both the network footprint of spam messages, as well as the overhead imposed by spam filtering on mail servers and end-users.

A key issue in designing router-based spam filters is the cost of implementing the technique, versus the overall benefits it offers. We present a preliminary architecture and empirical evaluation of a router-level filtering system that offers substantial spam filtering benefits while imposing a minimal load and low implementation costs on network routers.

Our approach is collaborative in nature. SMTP servers who have a more complete view of whether an email is spam or not, compute light-weight signatures for spam senders and send signature updates to network routers periodically. We assume a trust relationship between the SMTP servers and network routers, as might exist between the servers and border routers of a single enterprise, or the mail servers and network routers of an ISP. The routers apply filtering to all traffic on port 25 and drop traffic matching the stored signatures.

We consider a list of several candidate signatures that routers could check against. A good set of signatures requires a small amount of storage at the routers (and hence can be stored in high-speed memory such as SRAM), is easy to compute and check against, and is effective at filtering spam with a negligible false positive ratio. We argue that signatures based on the TCP fingerprint of the operating system of the email sender fully satisfies the above requirements. The fingerprinting can be done passively without the sender's knowledge.

TCP-based operating system fingerprints form the cornerstone of our router-based approach. There are many advantages to using TCP fingerprints as signatures: (1) Signatures can be computed based on a single TCP SYN packet destined to the SMTP port. Thus, signature computation is lightweight and requires no state maintenance. (2) Since the signature can be computed from the TCP connection establishment packets, spammers whose signatures match a stored signature can be prevented from establishing SMTP connections at the onset. (3) Matching TCP fingerprints is a much simpler task than content-based spam message signature matching and filtering. As well, numerous regular expressions and content phrases need not be maintained in order for the filtering to be effective. (4) Signatures are very few in number and thus, routers incur very little overhead in storing them. In contrast, IP address and prefix blacklists can contain millions of entries. Finally, (5) since our TCP fingerprinting approach is light-weight and completely stateless, it can precede the other filtering mechanisms.

We address two key challenges that prevail with our router-level mechanism: How do we identify a candidate set of signatures for filtering at routers? And what is the overall effectiveness of router-level filtering given a negligible false positive ratio? We address

CEAS 2009 - Sixth Conference on Email and Anti-Spam July 16-17, 2009, Mountain View, California USA

these challenges through an empirical study of email and *tcpdump* data collected at two different vantage points, the University of Wisconsin-Madison, USA and a corporation in Tokyo, Japan. We propose that receiving SMTP servers compute a list of TCP fingerprints observed in the received emails, and the corresponding *ham ratio* or the fraction of ham messages received out of the total amount of email received from all hosts with that particular signature. Signatures whose ham ratios are below a certain threshold (e.g. 0.001) are pushed to the routers after being observed in logs for a given amount of time(e.g. a month) and surpassing a minimum number of emails received. We find that the signatures of offending operating systems are fairly stable across time. Our analysis shows that this approach to router-level filtering can filter 28-59% of spam messages in the network with false positive rates of less than 0.05%.

Our light-weight router-level filtering mechanism is meant to supplement not supplant existing techniques, in order to reduce the burden of spam on local entities. Effective signatures reduce the amount of email that needs to be processed by subsequent anti-spam mechanisms. Our research discusses the effects of router-level filtering locally, but it could be applied universally across the world. Placing light-weight router-level filters strategically throughout the Internet could reduce the impact of spam worldwide. Although resources to mitigate spam may be adequate in some places, in others, spam remains a prevalent problem. Places where up-to-date filters have failed to be installed or users who primarily use dial-up connections are breeding grounds for spam to reach its intended targets. Given additional monitoring vantage points, a more comprehensive list of offending signatures could be developed that would significantly improve the benefits of a system such as ours. Our work indicates how to collect these signatures effectively.

Our study also leverages the IP addresses of spam senders to discover the origin of the majority of spam seen in our data sets. From the IP addresses we find the country from which the email was sent as well as the Autonomous System (AS) number. This information allows an ISP to pin point the most effective locations to place our router-level spam filters to prevent spam from reaching end-users. The extracted TCP fingerprints reveal signatures which originate all over the world but only send spam indicating the potential existence of global-scale spamming infrastructures.

The remainder of this paper is structured as follows. In Section 2, we discuss background information and prior studies that are related to our approach. In Section 3, we present the system architecture of our router-level filtering technique. Section 4 overviews the data sets used in this work. In Section 5, we show how to extract appropriate signatures to be stored in routers and discuss the performance of the filtration mechanism. In Section 6, we analyze the origin and robustness of the extracted signatures. Section 7 discusses various deployment issues. We summarize our work and outline future directions for this work in Section 8.

2. BACKGROUND

In this section, we first review router-level spam filtering techniques and other common *pre-acceptance* filtering mechanisms, e.g., DNSBL and greylisting, for spam mitigation. We also review TCP fingerprinting techniques to see how they are used in the context of spam filtering.

Pre-acceptance filtering tries to derive characteristics of the email sender that can help in identifying an email as spam. These tests apply to the initial handshake, prior to message reception. It has been empirically validated that pre-acceptance filtering works quite effectively if filters are managed carefully [15]. We note that post-acceptance filtering is always preceded by pre-acceptance filtering. Thus, effective and accurate pre-acceptance filtering can significantly reduce the load on SMTP servers, and enhance their ability to identify and thwart spam. Since our TCP fingerprinting approach is light-weight and completely stateless, it can precede the other pre- and post-acceptance filtering mechanisms.

2.1 Previous Techniques

Most spam filters are positioned at local SMTP servers or end-host machines. We observe that few router-level filters exist, as this research proposes. Deep packet inspection (DPI) is used in router-level spam filtering schemes in commercial products such as [1]. In this approach, many rule-based scoring and other heuristics are applied to each message. Although DPI filtering works well, it requires a lot of computational resources; thus, as the number of spam messages increases the amount of resources available must scale as well. Another router-level mechanism proposed by Agrawal et al. in [11], limits the bandwidth of spam flows by separating bulk streams from other traffic and applying a Bayesian classifier on the stream to determine if it is spam. Research studies show this approach to be extremely effective, but it requires the state of all the incoming SMTP flows to be monitored. Depending on the router location, the number of unique flows could be in the hundreds of thousands, thus reducing the overall appeal of such a mechanism.

IP address reputation services are widely used to create pre-acceptance filtering tools. DNS Blacklist (DNSBL) such as Spamhaus DNSBL [10] is an example of IP reputation service that can be used as a tool to detect potential spammers. These services collect and publish a list of IP addresses that are linked to spamming activity. These lists can be queried through a DNS interface and used for spammer detection by network administrators. It is also possible that a local SMTP server loads the list directly to drop connections from IP addresses on these lists. However, the number of IP addresses is generally quite large and variable over time. Thus, it is not a practical to have a router that implements the entire DNSBL process inside of it.

Greylisting is another pre-acceptance spam filtering approach, which rejects initial attempts to deliver a piece of email. Also located on SMTP servers, it logs *triplets* containing the sender's IP address, and recipient and sender addresses of the halted email. If the sender is a legitimate source, it is likely that the sender or mail server will attempt to resend the email. It will then be matched to the original entry in the log, resulting in the email being accepted and subsequent delivery to the recipient being completed. This is effective since spamming senders generally do not attempt to resend messages, because they blindly send to a variety of recipients, many of which do not exist. We note that greylisting needs to handle and keep track of the state of all incoming SMTP connections; thus, it is unfeasible to implement this approach at routers.

2.2 TCP Fingerprinting

TCP fingerprinting, also known as operating system (OS) fingerprinting, is used to identify the operating system and version of a remote host without their knowledge for analysis. Fingerprinting involves observing TCP packets and analyzing packet header fields. Since different operating systems implement their own TCP stack differently, a unique signature is created in TCP packets from each individual operating system (and sometimes each version).

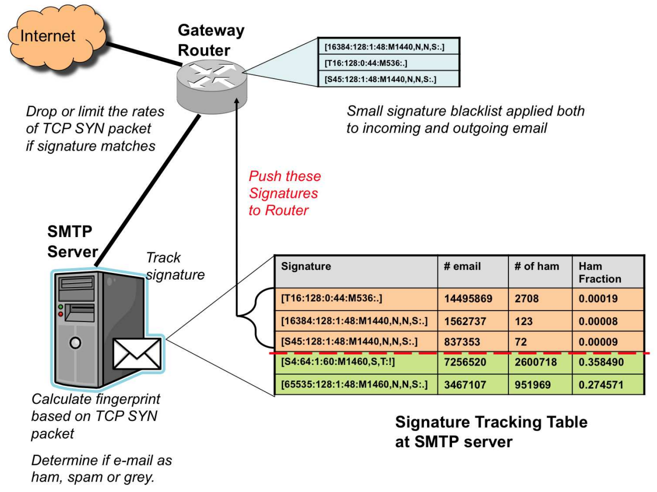
Examples of passive TCP fingerprinting tools are *p0f*, *Ettercap* and *Siphon* [2, 9, 21]. In this work, we employ *p0f*, which uses a single TCP SYN or SYN/ACK packet to identify the remote host's fingerprint. P0f comes with fingerprints for many different operating systems and allows one to add fingerprints. Although p0f is unable to identify all operating systems, it provides a good foundation on which many systems can be fingerprinted.

TCP fingerprinting has already been used in efforts to further understand the environments in which spam is sent. Ramachandran and Feamster [17] analyzed SMTP traffic destined to their *spam sinkhole* server and found that approximately 95% of the identified spam-sending hosts were running Windows. Similarly to Ramachandran et al., we employ the tool p0f and analyze ASN numbers. Our main differentiating contribution is that we present an anti-spam filtering architecture, which is able to solely leverage TCP fingerprints for spam filtering based on a statistical signature selection mechanism. While their study analyzes the data collected at *spam sinkhole*, which only receives spam messages, our study analyzes two *live* email data sets, which include both ham and spam. This allows us to analyze the fraction of spam/ham seen from particular signatures as well as the fraction they contribute to all spam/ham seen. We present similar statistics for the fractions of spam/ham sent by operating system genres, but our use of the more fine grain fingerprints is what makes our approach successful at filtering spam.

Calais et.al [13] proposes a methodology for characterizing email based on spamming strategies. They group spam messages into campaigns based on a technique which builds a frequent pattern tree. This structure is able to capture invariants of the original spam message content. They found a strong correlation between various operating systems and types of abuse seen. Li et. al [14] studied the clustering structures of spammers empirically. They investigated OS information of the spam host machine, using TCP fingerprinting. They found that among the total spam messages, 74% of them were sent from Windows, around 10% were from Linux, about 5% originated from BSD and Solaris machines, and about 11% were from unclassified hosts. Although not exactly TCP fingerprinting, Beverly and Sollins [12] proposed a technique to detect spam based on the transport-layer characteristics of SMTP flows, i.e., initial RTT estimate, congestion window size, number of packets, etc. They noticed that the values of these fields typically differed from that of a legitimate email.

There are several anti-spam mechanisms that adopt TCP fingerprinting to improve spam filtering. *Amavisd-new* has provided a *p0f* interface to *SpamAssassin* [8] which utilizes *p0f* to weight spam probability scores based on three primary operating systems: Windows, Unix and unrecognized systems [18, 20]. *Milter*, which is an extension to the popular mail transfer agent (MTA) such as *sendmail* [7] and *postfix* [6], has also provided an interface between the MTA and *p0f*, which runs as a daemon [5]. These tools classify spam into only a handful of operating systems genres, instead of analyzing more detailed fingerprints as our architecture proposes.

As we shall show in this work, building accurate signatures is crucial when fingerprinting is used as a technique for spam source filtration. For example, some signatures that are primarily based on Windows operating systems exhibit a high fraction of ham messages. Thus, the naive selection of certain fingerprints, e.g., selecting versions of Windows OS, can cause a significant number of legitimate messages to be dropped. For example, a legitimate email server such as Microsoft Exchange Server could be running on a version of the Windows OS, whose signature could be penalized for the bad behavior of other bot-infected hosts that also are running on that specific version. While many previous studies seemed to be aware that coarse-grained information could be extracted about a *classified* sender's operating system, to the best of our knowledge none of them paid a lot of attention to both the *classified* and *unclassified* signatures and their characteristics as a whole. Although the *classified* operating systems genres, such as Windows, Linux, FreeBSD, enables us to better understand the chances of a host being a spam sender, they don't reveal potential *new* signatures that could be associated with a new spamming entities, e.g., spamming botnet with a dedicated TCP/IP engine.


Internet
Gateway Router
[16384:128:1:48:M1440,N,N,S:.]
[T16:128:0:44:M536:.]
[S45:128:1:48:M1440,N,N,S:.]
Drop or limit the rates of TCP SYN packet if signature matches
Small signature blacklist applied both to incoming and outgoing email
Push these Signatures to Router
SMTP Server
Track signature
Calculate fingerprint based on TCP SYN packet
Determine if e-mail as ham, spam or grey.
Signature Tracking Table at SMTP server

Signature	# email	# of ham	Ham Fraction
[T16:128:0:44:M536:.]	14495869	2708	0.00019
[16384:128:1:48:M1440,N,N,S:.]	1562737	123	0.00008
[S45:128:1:48:M1440,N,N,S:.]	837353	72	0.00009
[S4:64:1:60:M1460,S,T:!]	7256520	2600718	0.358490
[65535:128:1:48:M1460,N,N,S:.]	3467107	951969	0.274571

Figure 1: Architecture for router-level spam filtering.

3. SYSTEM ARCHITECTURE

Our proposed architecture for router-level filtering is simple and is as illustrated in Figure 1. This protocol is a feedback-based architecture, where the SMTP server for the network computes a small list of candidate signatures and pushes signature updates to border routers on a periodic basis. The initial set of candidate signatures is constructed over some reasonable amount of time, for example, a month, and only applies to signatures who have a combined total of over 2,000 entries in the master log of all incoming connections. The minimum number of entries can be adjusted to ensure enough emails have been seen to adequately calculate the ham ratio for a given signature. As presented later, the feedback loop is crucial to cope with the introduction of new signatures which may have sparse locality at start-up.

We assume that the router has a small amount of fast memory dedicated to storing signature fingerprints and that look ups to the memory take just a few nano-seconds (current SRAMs offer 5 ns latency per access). The number of fingerprints is bounded, to ensure that memory look-ups meet this requirement. The router runs TCP header fingerprinting on all incoming TCP SYN packets in order to compute the fingerprint of each sender. The signature is hashed for looked up in the set of offending signatures. Assuming the fingerprints are stored in a hash table, each fingerprint look up will need one memory access (5 ns). If the signature is found in the hash table, the SYN packet is immediately dropped, preventing TCP channel communication establishment. If we prefer a more relaxed mitigation policy, the matched packets can be throttled by router instead of dropping all of them.

We also require that the fingerprinting mechanism be installed on the local SMTP server. The SMTP server fingerprints all incoming TCP SYN packets in a similar fashion to the router. In contrast with the router, however, the SMTP server creates a table which tracks the number of spam and legitimate emails, i.e., ham, received from senders using that specific operating system signature.

Here we assume that the classification of the email as spam or ham can be done on the SMTP server with a certain degree of accuracy. We use the output of an existing spam filtering software such as SpamAssassin or other third party commercial software to do this. We note though that outputs can be error-prone, but these filtering techniques are still useful because of skewed statistics of extracted signatures. We investigate this property later. If the SMTP server performs greylisting, this output log is also useful because we can regard the filtered connections as spam-sending hosts in general until retransmission is attempted. As we mentioned earlier, our approach is collaborative in nature. That is, our router-level filtering mechanism is meant to supplement not supplant existing techniques.

After identifying the sender's fingerprint from the TCP SYN packet, the packet is then dropped or forwarded to the SMTP server if the signature is not found in the list. The SMTP server accepts the incoming SMTP connection from the sender and then apply traditional post-filtering mechanisms. The final outcome is used to update the spam/ham tally for the corresponding TCP signature stored at the SMTP server.

The SMTP server tracks this information over a pre-determined period of time, T, at the end of which, it computes the ham ratio for the seen signatures. This is simply the number of ham messages relative to all email messages sent by this particular signature. All TCP fingerprints for which the ham ratio is less than a threshold, e.g., 10^{-4}, are considered to be offending fingerprints. The list of offending fingerprints is pushed to the router at the end of the time-period T and old signatures are flushed from the router. To prevent the vacillation of certain signatures longer term history would need to be tracked. We suspect the overhead of storing this information on a local server would be minimal. If an offending signature was noted for several months, this signature could be added to a default list of signatures that would be pushed to the router with every flush and update. Occasionally it would be necessary to confirm that the signature was still offending by dropping it back off the list, but this would need to occur infrequently.

Although signatures can be updated at frequencies ranging from daily to bimonthly, we anticipate updates will be infrequent since the addition of new operating systems or versions that alter the TCP/IP stack are rare. Our system also allows a network administrator to set an appropriate spam threshold for their own network to ensure that legitimate traffic is not being filtered. Lower thresholds introduce the possibility of ham messages being discarded, while very conservative thresholds fail to filter a sufficient amount of spam. The network operator may want to adjust the spam threshold according to balance between filter accuracy and the conservation of device resources that are required to run subsequent spam filtering mechanisms, e.g., DNSBL look up, greylisting, domain authentication, and content filtering. In Section 5, we present an analysis of the trade-offs between false positives and false negatives.

One advantage of our architecture is that it continuously monitors signatures. With infrequent updates, it is natural to assume that a threat could arise in between the updates. The network administrator could set local policies within the system architecture, which would alert them of suspect signatures that suddenly send an abnormally high number of spam messages. These signatures could then immediately be pushed manually by the admin to the router. We can also leverage anomaly detection techniques to automate this procedure.

Although our architecture is described as existing on a local ISP or SMTP server, it is feasible that this specific type of router-level filtering could be placed at the level of boarder routers. As explained in Section 5, the origin of spam senders can be discovered using our techniques, and this information provides us with an idea of where the placement of spam filters should be to stop spam close to the source. In order for a feasible deployment of this router-level mechanism by a regional or backbone ISP, a set of representative signatures would need to be constructed. In order for this to occur a subset of local sites would need to be *trusted* to provide accurate feedback of the history of signatures seen to a centralized database. Prior to signature submission, sites would need to conform to a set of signature scoring standards set by the larger authority. These standards would undoubtedly need to include what anti-spam scoring mechanism was to be used with what thresholds, and what the minimum number of emails seen needs to be in order for the ham ratio to be considered conclusive. Before being added to the centralized list the larger authority could impose a standard that a minimum number of sites must have identified this signature as suspect to ensure low false positive rates. Like local signature lists, this centralized list would need to be updated periodically and flushed accordingly. This would undoubtedly cause extra communication to occur within the network, but we assume that updates are infrequent and that only two messages would need to be sent per contributing site. One message to report the new signatures to the central authority, and one message from the central authority with the new signature list. Finally, this distributed filtering mechanism could save a lot of unwanted spam traffic that is disseminated to a number of leaf sites.

Another alternative to our architecture would be to replace the router with a middle box appliance. When routers are infeasible choice, being able to implement this mechanism prior to other pre- and post-acceptance filtering will still provide added benefit, just at a slightly higher processing cost. Since the focus of this paper is the original architecture, we leave the evaluation of this alternative for future work.

4. DATA DESCRIPTION

The data used in this study was collected at two vantage points located at different organizations and countries in April 2008. The first was collected at the University of Wisconsin - Madison, USA. The second collected at a corporation in Tokyo, Japan. In this work, we call the data sets collected at the two vantage points UW and CORP, respectively.

Each vantage point collects two data sets for the analysis of spam. The first data set data consists of a *tcpdump* of all incoming TCP SYN packets to the SMTP servers. The signatures of TCP headers are then extracted by running *p0f* [21] over the tcpdump files. From *p0f*, we are able to analyze specific operating characteristics about the sending host. Our second set of data contains all email delivery records for each vantage point for all respective email servers. At both sites greylisting[1] is performed as part of the email delivery system, and commercial anti-spam filtering is applied to all messages which pass the greylist filtering. The greylisted connection information is then constructed by taking the tcpdump logs and subtracting out all the connections which later appear in the delivery logs (when retransmission is attempted). The remaining connection information reveals the amount of *attempted* spam messages sent to the email servers, which was filtered by the greylisting mechanism. Note that most of spam messages were filtered at the greylisting stage. The delivered emails, which passed the greylisting filter, are assigned a spam probability score by the commercial spam filtering software. To avoid the risk of errors introduced by the filtering software, we used conservative thresholds to classify emails into spam, or ham, based on the score. For example, a spam email must have a spam probability score of greater

[1]At the UW collection site, greylisting is applied to blacklisted IP addresses while in the CORP setting, greylisting is applied to all the IP addresses that are *not* whitelisted.

Table 1: Characteristics of SMTP logs.

Dataset	#senders	#delivered emails	#delivered spam	#greylisted	#delivered ham
UW	7385521	26205650	13294554	87837109	12265296
CORP	3117901	2040443	1302582	18804706	545686

than 0.95 out of 1.0 in order to be considered spam, while a ham or legitimate email must have a score of smaller than 0.05. We note that software-based filtering is error-prone and thus, it could effect the derived statistics. However, despite the existence of potential errors in the classified messages, the information derived from the scores assigned by the software are promising because of the skewed distributions observed from in our data sets. As we shall see shortly, some signatures exhibit a very high fraction of spam messages over the total amount of messages sent by the signatures, e.g., 99.99%. Thus, even though the anti-spam software has some errors, say, a 5% false positive rate[2], it is very likely that the signatures with high spamming histories are strictly spam senders. We also note that for both vantage points, the majority of spam messages were filtered at the greylisting stage.

A major challenge in utilizing these data sets (the tcpdump and delivered logs) together is correlating the data across them. To accomplish this, we developed an algorithm that searched through the tcpdump logs and extracted every distinct signature on a per IP address basis for the entire month. In the typical case an IP address had one signature associated with it; in this case, all entries for that IP address in the SMTP logs were simply associated with that particular signature. If multiple fingerprint signatures were seen for the same IP, the algorithm attempted to match the SMTP entry to the closest entry in the tcpdump log for that day. To do this, we first look for an entry in the tcpdump log within five minutes of the greylisting for the SMTP entry we are comparing to. If one was found, then it was assigned to that signature. In most cases, multiple signatures were not seen in a single day, making the chances for misclassification slim.

The general statistics of the information gathered from the SMTP data collections are shown in Table 1. The table shows the size and diversity of senders that compose our data collection. In the table, "#greylisting" stands for the number of greylisted connections that did *not* attempt to resend the message(s). In this work, we count the number of spam messages as the summation of the number of delivered spam messages, i.e., "#delivered spam" in the table, and the number of connections filtered by greylisting, i.e., "#greylisting" in the table. The number of legitimate emails is listed in "#delivered ham". Further details of our data sets are presented in subsequent sections.

5. EXTRACTION OF SIGNATURES

In this section, we present simple heuristics that enable us to extract effective signatures. We also use the collected data sets to show the established results.

5.1 Statistics of Individual Signatures

After combining the tcpdump log with the SMTP log, we collect statistics for each signature. These statistics include the number of delivered messages, delivered spam messages, delivered ham messages, completely greylisted messages, and the number of distinct IP addresses for each signature. As a basic fingerprinting tool to analyze our TCP headers, we employ the *p0f* program [21]. The format of the extracted signature is `[W:T:D:S:O...:Q]`, where `W` stores the information about the window size, `T` is the initial value of TTL, `D` is the do not fragment bit, `S` is overall SYN packet size, `O` is the option value and order specification, and `Q` is a list of miscellaneous information. A full description of the option values and miscellaneous information can be found in the *p0f* manual [21].

Table 2: Top 10 Spam Sending Signatures for UW.

Signature	#Spam	#ham	#senders	OS genre
`[T16:128:0:44:M536:.]`	14495869	2708	260955	UNKNOWN
`[16384:128:1:48:M1440,N,N,S:.]`	1562732	123	20308	Windows
`[S45:128:1:48:M1440,N,N,S:.]`	837353	72	12270	Windows
`[65535:64:1:52:M1452,N,W2,N,N,S:.]`	679216	54	7537	UNKNOWN
`[65535:128:1:48:M1442,N,N,S:.]`	468074	14	8328	Windows
`[65535:128:1:48:M1352,N,N,S:.]`	361652	22	7843	Windows
`[65535:64:1:52:M1440,N,W2,N,N,S:.]`	298878	37	4331	Windows
`[T16:128:0:44:M1360:.]`	262077	21	3147	UNKNOWN
`[T16:128:0:44:M528:.]`	223246	3	2662	UNKNOWN
`[65535:128:1:52:M1460,N,W1,N,N,S:.]`	210267	45	3261	Windows

Table 3: Top 10 Spam Sending Signatures for CORP.

Signature	#Spam	#ham	#senders	OS genre
`[T16:128:0:44:M536:.]`	7252084	41	1139778	UNKNOWN
`[65535:128:1:48:M1440,N,N,S:.]`	1729670	56	165024	Windows
`[16384:128:1:48:M1452,N,N,S:.]`	327224	31	47412	Windows
`[16384:128:1:48:M1440,N,N,S:.]`	284660	17	43938	Windows
`[S44:128:1:48:M1452,N,N,S:.]`	166357	6	24545	Windows
`[65535:128:1:48:M1420,N,N,S:.]`	132957	16	18771	Windows
`[T16:128:0:44:M1360:.]`	126955	0	21329	UNKNOWN
`[65535:128:1:48:M1400,N,N,S:.]`	94770	26	13910	Windows
`[T16:128:0:44:M528:.]`	90518	0	9463	UNKNOWN
`[S45:128:1:48:M1440,N,N,S:.]`	71594	1	16083	Windows

Tables 2 and 3 show the top 10 spam-sending signatures by total number of messages in the two data sets with a negligible ham ratio. Here, the value of initial TTL is corrected with the formula $2^{\lceil \log_2(t)+1 \rceil}$, where t is the value of observed TTL[3]. We notice that for both data sets, the top signature is in common. Senders with this signature contribute a significant amount of spam messages, and a very small number of ham messages. We also notice that many of signatures are common across both data sets. Finally, we note that many of signatures are variants of the Windows operating system. This observation agrees with previous studies, such as [17], which found that approximately 95% of identified spam-sending hosts were running Windows. It is noteworthy that the top signature and their variants, which start with `T16:128:0:44`, are unclassified hosts; meaning, these signatures cannot be identified with the latest version of *p0f*. To the best of our knowledge, ours is the first work that finds the existence of these intrinsic signatures that are not known versions of OSes[4]. The characteristics of these signatures are presented in the next section.

For comparison, we also investigate the top-ham sending signatures by number of messages sent (see Table 4, 5). Unlike in the case of spam senders, we see no common signatures among the two data sets. This may reflect the difference in the two vantage point locations. One is an academic organization in US while the other is a private company in Japan; thus, end-users are likely receiving a different set of e-mails from a different set of senders. It is noteworthy that a large amount of spam messages are sent from hosts with signatures on the top-ham sending signatures list. This is especially prevalent when the number of senders with a given signature is large, indicating that spammers have piggybacked some of their op-

[2]This number may be very pessimistic. We manually checked the accuracy of the software using sampled messages delivered to our mailbox and found that the number of misclassifications was quite low for both data sets.

[3]If the corrected value is 256, we apply another heuristic so that the corrected value is 255.

[4]We manually updated the signatures of p0f for newer versions of OSes such as variants of Windows Vista, Mac OS X 10.4 and later, FreeBSD 7.0 and later etc. However, none of them matched to the unknown signatures mentioned above.

Table 4: Top 10 Ham Sending Signatures for UW.

Signature	#Spam	#ham	#senders	OS genre
`[S4:64:1:60:M1460,S,T:!]`	5761617	2600718	125635	UNKNOWN
`[65535:128:1:48:M1460,N,N,S:.]`	18761354	951969	186748	Windows
`[S4:256:1:48:M1460,N,W9:.]`	1356	889390	9	UNKNOWN
`[S4:64:1:60:M1380,S,T:!]`	187572	877897	10195	UNKNOWN
`[S4:64:0:60:M1430,S,T:!]`	171279	851829	397	UNKNOWN
`[S4:64:1:44:M1460:.]`	89709	515808	5315	Linux
`[65535:128:1:48:M1380,N,N,S:.]`	789546	396238	19900	Windows
`[65535:64:1:64:M1460,N,W1,N,N,T:T!]`	154175	384603	6854	UNKNOWN
`[65535:64:1:60:M1460,N,W1,N,N,T:T]`	40735	347468	3858	UNKNOWN
`[5792:256:1:60:M1460,S,T:T!]`	475	278431	9	UNKNOWN

Table 5: Top 10 Ham Sending Signatures for CORP.

Signature	#Spam	#ham	#senders	OS genre
`[S4:64:1:60:M1460,S,T,N,W0:.]`	84138	84124	9432	Linux
`[S4:64:1:60:M1460,S,T,N,W2:.]`	102542	46000	6613	Linux
`[S34:64:1:52:M1460,N,W0,N,N,S:.]`	21656	32184	578	Solaris
`[S17:64:1:48:N,N,S,M1460:.]`	534461	31368	707	Solaris
`[S4:64:1:60:M1460,S,T,N,W7:.]`	20619	29507	3310	Linux
`[16384:64:0:60:M1460,N,W0,N,N,T0:.]`	4412	24852	5	QNX
`[32768:64:0:48:M536,W0,N:.]`	1281	22628	202	HP-UX
`[57344:64:1:60:M1460,N,W0,N,N,T:.]`	79590	21401	1054	FreeBSD
`[65535:64:1:64:M1460,N,W1,N,N,T,S,E:P]`	19936	18619	1591	FreeBSD

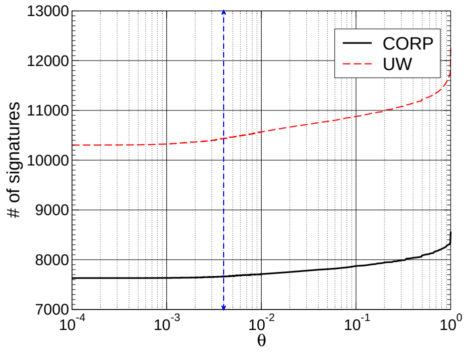


Figure 2: Number of signatures under the various thresholds.

erations off default operating system configurations. These signatures would need to be filtered using other anti-spam mechanisms. Another important finding here is that signatures in the UW data set that are Windows-based sent out a significant fraction of ham messages, where Windows-based signatures on the spam list had the exact opposite characteristic. This indicates that specific versions of operating systems are highly susceptible to spam sending operations and/or have been altered specifically for this type of operation. The classification assigned to these ham-sending Windows-based signatures by *p0f* suggests that these are machines running versions of Windows 2000 Server. Given that this classification is in fact true, many might be running legitimate MTAs such as Microsoft Exchange Server, which would send ham e-mails. Thus, the naive approach of selecting Windows-based signatures as suspect could largely fail. This shows the need for TCP fingerprinting that is highly specific to carefully avoid the risk of misclassification.

5.2 Extracting Signatures

The next part of the analysis involves extracting effective signatures for spam filtering. Good signatures will have a high spam coverage ratio when applied to the message set while having negligible rate at which ham messages are selected. A good set of signatures is also fairly small (less than 200) so that the entire list can be loaded onto fast SRAM memory in routers. For this purpose, we first introduce a threshold, θ, which is the fraction of filtered ham messages over the total number messages sent by a given signature. As the threshold gets higher, we get more signatures (see Figure 2) and senders that are filtered (see Figure 3). Accordingly, the number of potential ham messages filtered also increases (graph omitted for brevity), thus affecting the performance of signatures as we shall see in short. The threshold, θ, turns out to be a valuable

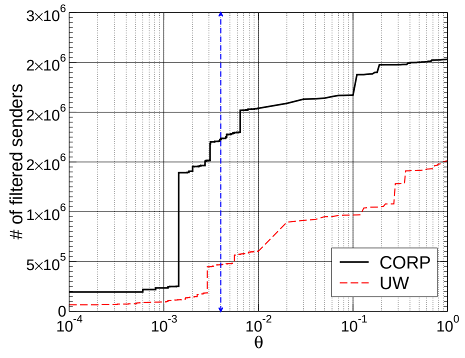


Figure 3: Number of filtered senders under the various thresholds.

control parameter in extracting effective signatures.

The actual extraction procedure is composed of two-stages. In the first stage, the candidate signature list is created based on signatures that surpass a set θ value. The network administrator can then adjust the θ value or add/remove signatures to construct a list that is representative of the goals of this anti-spam mechanism. In our case, we pick a set of signatures that satisfies a good trade-off between the FPR (false positive ratio) and FNR (false negative ratio). FPR is the fraction of misclassified ham messages over the total ham messages, while FNR is the fraction of missed spam messages over the total spam messages. Figure 4 shows the FPR and FNR for UW and CORP data sets. For CORP, there is a drastic decrease in FNR at a threshold of around $\theta = 0.0014$. This indicates that signatures that are identified as being suspect with this threshold, have a much higher chance of filtering spam messages than if the threshold was more restrictive. We will look at the details of this signature later.

In the second stage, we pick a small set signatures that have a good coverage ratio when applied to spam messages. This stage is aimed to reduce the amount of signatures that are kept on the SRAM of routers, which is generally expensive. Lets assume that we need FPR of less than 5×10^{-4}. If we pick the threshold of $\theta = 0.004$, which is shown with dashed arrows in Figure 4, the FPRs are 0.00030 and 0.00045, and the FNRs are 0.72 and 0.41 for UW and CORP, respectively. Figure 5 shows the contributions of the top signatures to the FPR and FNR. Here, the signatures are sorted by the number of spam messages they sent in descending order. We see that a very few number of signatures can establish the good balance between filtering and falsely dropping ham e-mails. For example, if we use the top 100 signatures, the FPRs 0.00028 and 0.00045, FNRs are 0.75 and 0.44 for UW and CORP, respectively. We also note that the accuracy and the coverage of the top signature is quite striking. Just one signature covers more than 15-35% of spam messages with a false positive ratio of less than 0.00025. This outstanding signature, `[T16:128:0:44:M536:.]`, is shown in the first row of Tables 2 and 3. For further reference we denote this signature and its variants as *X*. According to recent studies [16, 19], these signatures are associated with the spamming botnet, Srizbi, which has caused significant amount of spam all over the world since mid 2007.

We note that as we mentioned before, the "#Spam" in the tables is the summation of the number of delivered spam messages and greylisted connections, i.e., unique triplets that did not resend messages. We found that most spam originating from the signature *X* is effectively stopped through greylisting. In the next section, we will further look into the origin of the signature X.

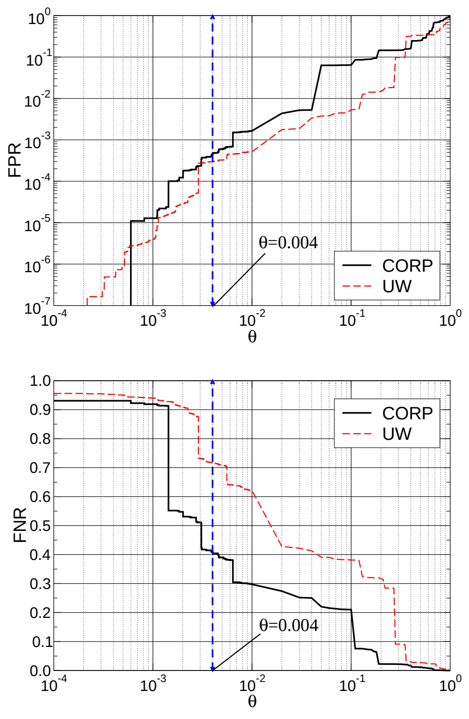


Figure 4: Performance of extracted signatures under the various thresholds (the first stage): False positive ratio (top) and False negative ratio (bottom).

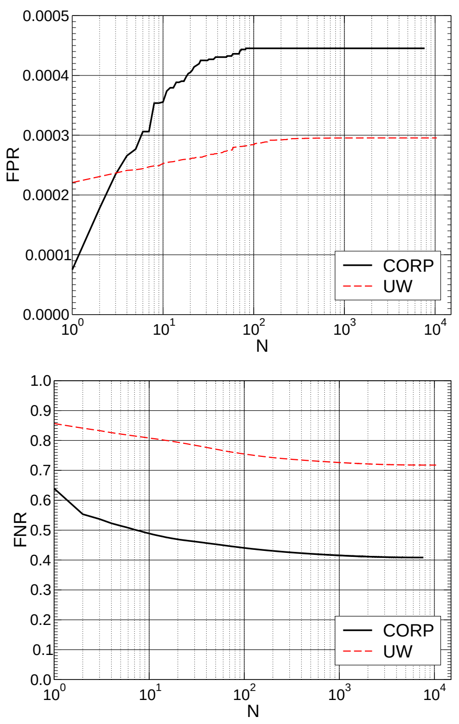


Figure 5: Performance of top N signatures, which were extracted with $\theta = 0.004$ at the first stage. These signatures are sorted by the number of spam messages they sent in descending order. False positive ratio (top) and False negative ratio (bottom).

6. ANALYSIS OF EXTRACTED SIGNATURES

In this section, we analyze various properties of the spam sending signatures extracted by our technique.

6.1 Origin of Spam Sending Signatures

We first study the origin of the top 100 signatures with the threshold of $\theta = 0.004$ for each data set that have sent more than our $2,000$ minimum number of emails. For comparison purposes, the origin of the special case, signature *X*, is investigated separately.

Tables 6-9 show the top 10 ASes from which spam senders originate with these top 100 signatures. We first notice that many of the origins of the senders are in common among the two vantage points. This reflects the fact that spammers target recipients all over the world, thus two entities on opposite sides of the world may see spam originating from a single source. We also notice that the signatures are originated from various regions mainly in Asia, Europe, and South America. These findings suggest that individual spam sending signatures seen across these regions may construct a larger global-scale spamming infrastructure. It is noteworthy that the main origin AS of signature *X* is slightly different from other signatures. Since the signature is associated with a spamming botnet, it is likely that the number of hosts infected with the spamming bot has increased popularity in these regions over other parts of the world. Thus, by tracking extracted signatures we see correlations related to the physical origin of spam. The outgoing links from these top 10 ASes from Tables 6 and 7 would be ideal sites for the placement for our router-level spam filters. This would stop spam close to the source preventing it reaching destinations worldwide.

Next, we use commercially created DNSBLs to study the characteristics of senders that were identified by the top 100 extracted spamming signatures. Three DNSBLs created by the Spamhaus project [10] are used for this analysis, namely PBL, SBL, and XBL. SBL is a verified list of spam sources, including spammers, spam gangs and spam support services. PBL is a list of IP addresses which are designated as being part of the dynamic and DHCP address space, thus they are not supposed to make direct SMTP connections. Lastly, XBL gives us a list of exploited hosts. These compromised hosts now include open proxies, worms/viruses with built-in spam engines, and other types of exploitable trojan-horses for remote users to take advantage of. All DNSBLs were collected during the same month as our spam measurement period, April 2008.

The procedure for our analysis of senders by their IP address is detailed here. We first compile a list of all IP addresses that are seen in our set of extracted top 100 spam sending signatures including signature *X*. Each IP address is then classified based on which DNSBLs it is on. Finally, we reanalyze the logs for these selected IP addresses to calculate the statistics of each IP address. Table 10 shows the number of spam messages sent and senders classified by each of the three lists for our collected data sets. For both cases, the majority of spam messages and senders are matched by PBL. This list encompasses more than 82% of IP addresses which sent mail using the signature *X*, and these hosts sent out more than 65% of the spam messages sent by this signature. Since PBL is the list of dynamic IP addresses, this result supports the fact that signature *X* is actually associated with spam activity spawning from the botnet, Srizbi.

We also find that a certain amount of addresses and spam messages are *not* covered by the DNSBLs. In our study more than 15% of spam senders that contributed more than 25% of spam messages were not covered by DNSBLs for both data sets. This finding suggests that our TCP fingerprinting approach can assist in improving the coverage of existing DNSBLs. Since the results for the remainder of top 100 extracted spam signatures are similar, we omit them for brevity.

Table 6: Top 10 ASes that originate the senders with the signature *X* ($\theta = 0.004$) for UW data set.

ASN	AS name	# of senders
9121	TTNET TTnet Autonomous System	208405
4134	CHINANET-BACKBONE No.31,Jin-rong Street	146338
4837	CHINA169-BACKBONE CNCGROUP China169 Backbone	86983
5617	TPNET Polish Telecom's commercial IP network	62940
3269	ASN-IBSNAZ TELECOM ITALIA	56020
22927	Telefonica de Argentina	51563
8359	COMSTAR COMSTAR-Direct Moscow region network	49336
7738	Telecomunicacoes da Bahia S.A.	47939
7470	ASIAINFO-AS-AP ASIA INFONET Co.,Ltd.	46739
8167	TELESC - Telecomunicacoes de Santa Catarina SA	37059

Table 7: Top 10 ASes that originate the senders with the signature *X* for CORP data set.

ASN	AS name	# of senders
9121	TTNET TTnet Autonomous System	234182
4134	CHINANET-BACKBONE No.31,Jin-rong Street	87427
5617	TPNET Polish Telecom's commercial IP network	60694
7470	ASIAINFO-AS-AP ASIA INFONET Co.,Ltd.	60663
4837	CHINA169-BACKBONE CNCGROUP China169 Backbone	52539
22927	Telefonica de Argentina	52334
8359	COMSTAR COMSTAR-Direct Moscow region network	36200
7738	Telecomunicacoes da Bahia S.A.	36040
6147	Telefonica del Peru S.A.A.	31332
3269	ASN-IBSNAZ TELECOM ITALIA	29811

6.2 Robustness of Signatures

In this section, we investigate the robustness of the extracted signatures relative to both the temporal and spacial domains. Figure 6 shows the fraction of signatures in common for the top N signatures from the two vantage points. As we mentioned earlier, the top signature, signature X is common among the two vantage points. On the other hand, other top-signatures exhibit locality here. That is, the fraction of common signatures is not close to one. The locality suggests the need to apply a local feedback loop to obtain effective signatures, while other signatures can be derived locally or from a centralized database of high spam sending signatures. Tables 11 and 12 show the performance of the two sets of top 100 signatures based on the number of spam and ham emails filtered. These tables also reflect how locality can effect the performance of the protocol by analyzing the intersection and union of the two sets. The locality of the signatures collected at the other site can increase the number of false positives seen when union filtering is applied. On the other hand, by taking the intersection of the two signature sets, we can decrease the false positive ratio, but at the cost of a certain amount of spam messages being missed. If a network operator prefers less false positives to less false negatives, she/he may take the intersection of signatures that are compiled at different locations.

Table 13 shows the fraction of connections and senders that are covered by the original UW top 100 spam signatures (from April 2008) when applied to later months. The CORP data exhibits similar properties, hence it is omitted. Since SMTP logs were only collected for the month of April, we analyze the number of connections and senders that appeared in the tcpdump files collected. We also confirmed that the coverage of DNSBLs were same as were revealed in Table 10. Again, for each month about 80% of the senders extracted by the signatures collected in April 2008 were listed in the DNSBLs collected. Thus, we conclude that the obtained spam sending signatures were stable over of a period of several months for both data sets.

7. DISCUSSION

In this section we discuss some final details related to utilizing TCP fingerprints for signatures for router-level spam filtering.

Table 8: Top 10 ASes that originate the senders with the top 100 signatures *except* signature *X* ($\theta = 0.004$) for UW data set.

ASN	AS name	# of senders
3269	ASN-IBSNAZ TELECOM ITALIA	90004
4134	CHINANET-BACKBONE No.31,Jin-rong Street	72019
4837	CHINA169-BACKBONE CNCGROUP China169 Backbone	43611
9121	TTNET TTnet Autonomous System	32565
9050	RTD RTD-ROMTELECOM Autonomous System Number	28590
3320	DTAG Deutsche Telekom AG	24205
7738	Telecomunicacoes da Bahia S.A.	23723
22927	Telefonica de Argentina	21742
13184	HANSENET HanseNet Telekommunikation GmbH	20458
3209	ARCOR-AS Arcor IP-Network	18684

Table 9: Top 10 ASes that originate the senders with the top 100 signatures *except* signature *X* ($\theta = 0.004$) for CORP data set.

ASN	AS name	# of senders
4134	CHINANET-BACKBONE No.31,Jin-rong Street	68635
4837	CHINA169-BACKBONE CNCGROUP China169 Backbone	55704
3269	ASN-IBSNAZ TELECOM ITALIA	20479
9121	TTNET TTnet Autonomous System	18433
22927	Telefonica de Argentina	13478
6147	Telefonica del Peru S.A.A.	11364
9829	BSNL-NIB National Internet Backbone	11211
7418	Terra Networks Chile S.A.	10503
7470	ASIAINFO-AS-AP ASIA INFONET Co.,Ltd.	9887
27699	TELECOMUNICACOES DE SAO PAULO S/A - TELESP	9576

Table 10: Classification and statistics of email senders with the signature *X* for UW (left) and CORP (right).

DNSBL	#Spam	#Senders	DNSBL	#Spam	#Senders
SBL	1538633	27854	SBL	662555	21510
XBL	922	114	XBL	677	87
PBL	10009504	1407027	PBL	5112101	973058
unmatched	3669080	279825	unmatched	2039988	186714

7.1 Sharing Fingerprints

As we have seen in the previous section, signatures collected at different locations can improve the accuracy of our TCP fingerprinting framework. Based upon the collected signatures, we can create a set of fingerprints that are shared among participating network operators. We should note, however, that these signatures can exhibit locality. Thus, in addition to the global list, it is preferable to run the signature extraction mechanism locally to compile an effective and compact list of signatures. The benefit of utilizing a global list is that if many sites are able to confirm that a particular signature is only sending spam, local administrators can be reassured that they are not accidentally filtering ham emails.

As with other research efforts related to fingerprint sharing [3], an active fingerprint sharing infrastructure is beneficial when security threats escalate quickly. This is particularly true with such

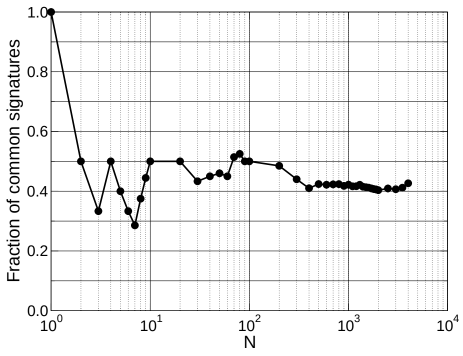


Figure 6: Fraction of common signatures in the top N signatures for the two vantage points ($\theta = 0.004$).

Table 11: Performance of the two sets of signatures compiled at each location including their intersection and union when applied to the UW data set.

Set of signatures	#Spam	#Ham	#Senders
CORP Top 100	34378320	33756	561278
UW Top 100	24797823	3485	403568
INTERSECTION	21329958	3211	360627
UNION	37846185	34030	604219

Table 12: Performance of the two sets of signatures compiled at each location including their intersection and union when applied to the CORP data set.

Set of signatures	#Spam	#Ham	#Senders
CORP Top 100	11249690	243	1639667
UW Top 100	8676986	443	1361959
INTERSECTION	8383147	89	1316314
UNION	11543529	597	1685312

Table 13: Stability of the top 100 signatures collected in April 2008 (UW).

Month	fraction of connections	fraction of senders
April 2008	0.74	0.68
May 2008	0.77	0.67
June 2008	0.78	0.69
July 2008	0.63	0.61

Table 14: Stability of the signature *X* collected in April 2008 (UW).

Month	fraction of connections	fraction of senders
April 2008	0.65	0.52
May 2008	0.68	0.51
June 2008	0.71	0.53
July 2008	0.53	0.41

events as distributed denial of service attacks. When an attack occurs, the site should be able to fingerprint the offending connection, and an automatic alert can be sent network administrator. Once validated, this signature can be sent to other sites via the shared infrastructure, and these sites can temporarily drop connections matching that signature.

In this work, we assumed that the classification of emails as spam or ham could be done on a local SMTP server. While this assumption holds true for some networks, it is not always the case. For such networks, we can still derive useful information for signature fingerprints from greylisting logs and email bouncing errors, but the picture of what is spam or ham will be less clear. In this case, it is also possible to utilize other DNSBLs or DNSWLs to give weight to the scores to each signature. Finally, if none of the above mentioned techniques are available, the site can utilize the shared global fingerprints.

Since signatures themselves do not contain any private information, we conjecture that sharing the signatures will not cause any serious privacy problem. By collecting signatures from many trusted networks, it is possible to compile an useful set of signatures for use worldwide.

7.2 Signature Updates

In order for a router-level spam filter to be effective it must require updates infrequently and have low false positive rates. Infrequent updates help maintain high filter performance and reduce the amount of required update downtime. During the period in which we collected data, we monitored the number of signatures that surpassed our established spam threshold and therefore, needed to be pushed to the routers. As we showed in the previous section, 100 signatures needed to be push to the router to establish a false positive less than 0.0005 and a coverage ratio of more than 28% (59% for CORP). We also showed that the collected signatures were stable over a period of several months at each location. We note, that finding an appropriate θ value is not a clear cut process, but that it must be set carefully based on the goal of the anti-spam mechanism as it will effect updates to the filter list.

7.3 Threat Analysis

Although our research has shown signatures to be fairly consistent over a few months worth of data, this might not always be the case. Here, we briefly consider some threats that our router-level filtering approach faces from clever spammers that would try to thwart detection if they knew their fingerprints were being targeted.

There are generally two spamming scenarios which arise: (1) A spammer hijacks or assembles machines from which it sends out spam through the default operating system network driver. (2) A spammer maliciously installs a new network driver or modifies the existing one for spam sending purposes. In the first scenario, spammers have much less control, if any over the TCP field values observed from the emails they send. They can take advantage of tools, such as IPPersonality [4], so that they can masquerade as a different operating system, but the choices they have are limited. In this case, these spam emails generally contribute to the statistics of signatures that have already been observed, and likely can not be filtered because legitimate mail is also sent from them. The second scenario is more problematic.

With a dedicated network driver a spammer can change almost any TCP field resulting in spam messages with any number of possible signatures. A spammer might target fields, such as initial TTL value, window size, default flags and NOP options, for change. Although changing these values would allow a spammer to initially avoid detection, the new signature would be added to the drop list if it is consistently observed as sending spam. Fingerprints could be altered on a daily or even hourly basis, but this would require significant changes to current spamming infrastructures. For example, Mori et al. [16] noted the rise and fall of the Srizbi version one signature in August 2007 and November 2008, respectively. Srizbi version two was first observed in October 2008, but its rise has been slower than version one; indicating that it takes time to spread changes to spamming sources or create new ones. A router-level filtering approach, such as we propose, could stop many of these spamming infrastructures that have specific network drivers. Even with the ability to randomize values, spammers would need to wisely choose values to ensure their packets would not be dropped by low TTL values or by systems that drop obscure looking packets.

Most likely, legitimate hosts would keep consistent signatures, thus the set of suspect signatures to be dropped at routers would simply adapt with the dynamically-changing seen signature set. If at some point, spamming signatures were completely being randomized, the system could be altered to store the top ham sending signatures and probabilistically drop all other signatures seen. This approach would result in mechanism similar to whitelisting.

Each of these aforementioned issues would result in a spam email being misclassified as legitimate traffic by our filtering mechanism. In these cases, the third-party spam filtering mechanism would be in place to help identify the message as spam.We also argue that the potential for false positives is very low in our approach, and that spammers cannot induce false positives. In particular, we consider it unlikely that spam-sending hosts could send enough spam using a forged legitimate signature to skew the spam ratio enough to exceed the thresholds, and thus cause legitimate traffic using that signature to be filtered. In our analysis, we used a spam threshold of .004, which translates into a spam sending host needing to send roughly 250 times the amount of legitimate mail in spam. In the unlikely event that this did happen, all the network administrator would have to do is manually remove this signature from the filter list. Signatures like this would be easily identified because of the detailed history kept for each signature. A large amount (say a few tens of thousands) of ham emails from a specific signature - no matter what the ham ratio - should alert the human operator of the legitimacy of some of the emails sent from this signature.

8. CONCLUSIONS AND FUTURE WORK

In this paper we have presented a preliminary approach for router-level spam filtering which utilizes passive TCP fingerprinting to form candidate signatures. Our architecture is simple and easy to implement at either a local or ISP gateway router. As an alternative, our mechanism could be placed in existing middle box devices.

Router-level spam filtering would reduce overhead placed on MTAs whose resources are heavily consumed as the number of spam messages continues to increase. This approach will limit the number of incoming TCP sessions by immediately dropping connections which match a stored TCP fingerprint signature. The router-level mechanism can and should be used in conjunction with a variety of other spam filtration techniques to ensure accurate spam detection.

In our study, we were able to identify several candidate signatures that would reduce the amount of spam processed by secondary anti-spam filters. These signatures have several distinguishing features, and we note that our top signature *X* is associated with Sribzi botnet. The locations of these spamming entities are diverse, but our study reveals several specific ASNs, from Asia, Europe and South America, from which much of the spam we see today is originating. We also found that the naive approach of selecting Windows-based signatures as suspect could largely fail. This fact shows the need for highly accurate signature selection mechanism to avoid the risk of misclassification. We validated that our method can meet such a requirement.

We argue that TCP fingerprint filtering, as a supplementary filtering mechanism based on fine granularity signatures, results in low false positives. Although fingerprints could be randomized or mimic legitimate operating systems, this would only cause messages that are currently being analyzed to be analyzed with the new system. The remaining messages that would be filtered would help reduce the load on MTAs.

Our future work includes a more detailed analysis of the top 100 spam signatures. We are particularly interested in discovering what global-scale spamming infrastructures exist that contribute the spam signatures we have seen in our logs. A longer-term study of TCP signatures also will help to solidify our conclusion that signatures are relatively stable over time, and allow us to analyze which signatures in fact do change. We hope to implement and deploy a prototype version of our the system, to evaluate the performance of our architecture in terms of spam filtering and feasibility for large-scale deployment.

Acknowledgements

We thank Mike Blodgett, Jesse Thompson, Jeffrey Savoy, David Plonka, Mitsuhiro Shigematsu, Masashi Mitsuda, Kanako Nozue and Hideko Mills for their assistance in collecting the SMTP traffic. This measurement was performed in cooperation with the University of Wisconsin - Madison's Division of Information Technology. Finally, we thank Archit Gupta and Lydia Sidrak for their great contribution to the early stage of this work. This work was supported in part by an NSF CAREER Award (CNS-0746531) and an NSF NeTS FIND Award (CNS-0626889).

9. REFERENCES

[1] Barracuda spam firewall. `http://www.barracudanetworks.com/ns/products/spam_overview.php`.

[2] Ettercap. `http://ettercap.sourceforge.net/`.

[3] Fingerprint Sharing Alliance. `http://www.arbornetworks.com/en/fingerprint-sharing-alliance.html`.

[4] IPPersonality. `http://ippersonality.sourceforge.net/`.

[5] Milter. `http://www.milter.org/`.

[6] Postfix. `http://www.postfix.org/`.

[7] Sendmail. `http://www.sendmail.org/`.

[8] SpamAssassin. `http://spamassassin.apache.org/`.

[9] The Siphon Project. `http://siphon.datanerds.net/`.

[10] The Spamhaus Project. `http://www.spamhaus.org/`.

[11] B. Agrawal, N. Kumar, and M. Molle. Controlling spam emails at the routers. *Communications, 2005. ICC 2005. 2005 IEEE International Conference on*, 3:1588–1592 Vol. 3, 16-20 May 2005.

[12] R. Beverly and K. Sollins. Exploiting transport-level characteristics of spam. In *CEAS*, 2008.

[13] P. Calais, D. Pires, D. Guedes, W. M. Jr., C. Hoepers, and K. Steding-Jessen. A campaign-based characterization of spamming strategies. In *CEAS*, 2008.

[14] F. Li and M.-H. Hseih. An empirical study of clustering behavior of spammers and group-based anti-spam strategies. In *CEAS*, 2006.

[15] T. Mori, H. Esquivel, A. Akella, Z. M. Mao, Y. Xie, and F. Yu. On the effectiveness of pre-acceptance spam filtering. *University of Wisconsin Madison Tech Report TR1650*, Mar. 2009.

[16] T. Mori, H. Esquivel, A. Akella, A. Shimoda, and S. Goto. Understanding the world.s worst spamming botnet. *University of Wisconsin Madison Tech Report TR1660*, June 2009.

[17] A. Ramachandran and N. Feamster. Understanding the network-level behavior of spammers. In *SIGCOMM '06: Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 291–302, New York, NY, USA, 2006. ACM.

[18] A. Schwartz. amavisd-new and p0f. http://www.amazon.com/gp/blog/post/PLNK3F6QB0NKZ1DZT, 2006.

[19] H. Stern. The rise and fall of reactor mailer. In *Proc. MIT Spam Conference 2009*, Mar 2009.

[20] D. Webber. Spamassassin p0f plugin catches bot spam. http://advosys.ca/viewpoints/2007/07/spamassassin-p0f-plugin-catches-bot-spam/, 2007.

[21] M. Zalewski. the new p0f: 2.0.8. http://lcamtuf.coredump.cx/p0f.shtml, 2006.