An Effective Defense Against Email Spam Laundering

Mengjun Xie, Heng Yin, Haining Wang
Department of Computer Science
The College of William and Mary, Williamsburg, VA 23187
{mjxie, hyin, hnw}@cs.wm.edu

ABSTRACT

Laundering email spam through open-proxies or compromised PCs is a widely-used trick to conceal real spam sources and reduce spamming cost in underground email spam industry. Spammers have been plaguing the Internet by exploiting a large number of spam proxies. The facility of breaking spam laundering and deterring spamming activities close to their sources, which would greatly benefit not only email users but also victim ISPs, is in great demand but still missing. In this paper, we reveal one salient characteristic of proxy-based spamming activities, namely packet symmetry, by analyzing protocol semantics and timing causality. Based on the packet symmetry exhibited in spam laundering, we propose a simple and effective technique, DBSpam, to on-line detect and break spam laundering activities inside a customer network. Monitoring the bi-directional traffic passing through a network gateway, DBSpam utilizes a simple statistical method, Sequential Probability Ratio Test, to detect the occurrence of spam laundering in a timely manner. To balance the goals of promptness and accuracy, we introduce a noise-reduction technique in DBSpam, after which the laundering path can be identified more accurately. Then, DBSpam activates its spam suppressing mechanism to break the spam laundering. We implement a prototype of DBSpam based on *libpcap*, and validate its efficacy through both theoretical analyses and trace-based experiments.

Categories and Subject Descriptors: C.2.0 [Computer Communication Networks]: Security and protection

General Terms: Security.

Keywords: Spam, Proxy, SPRT.

1. INTRODUCTION

As a side-product of free email services, spam has become a serious problem that afflicts every Internet user in recent years. According to MessageLabs [1], currently over 60% email traffic is spam. Although a number of anti-spam mechanisms have been proposed and deployed to foil spammers, spam messages continue swarming into Internet users' mailboxes. A more effective spam detection and suppression mechanism close to spam sources is critical to dampen the dramatically-grown spam volume.

At present, proxies such as off-the-shelf SOCKS and HTTP proxies play an important role in the spam epidemic. Spammers launder email spam through these proxies to conceal their real identities and reduce spamming cost. The popularity of proxy-based spamming is mainly due to the anonymous characteristic of a proxy and the availability of a large number of spam proxies. The IP address of a spammer is obfuscated by a spam proxy during the protocol transformation, which hinders the tracking of real spam origins. According to Composite Blocking List [7] which is a highly-trusted DNSBL (DNS-based Blackhole List), the number of available spam proxies and bots in August 2006 is more than 3,200,000. Such numerous spam proxies facilitate the formation of email spam laundering, by which a spammer has great flexibility to change spam paths and bypass anti-spam barriers.

To break this spam laundering, we propose a simple and effective mechanism, called *DBSpam*, which detects and blocks spam proxies' activities inside a customer network and further traces the corresponding spam sources outside the network. DBSpam is designed to be placed at a network vantage point such as the edge router or gateway that connects a customer network to the Internet. The customer network could be a regional broadband (cable or DSL) customer network, a regional dialup network, or a campus network. It detects ongoing proxy-based spamming by monitoring bi-directional traffic. Due to the protocol semantics of SMTP and timing causality, the behavior of proxy-based spamming demonstrates the unique characteristics of connection correlation and packet symmetry. Utilizing this distinctive spam laundering behavior, we can easily identify the suspicious TCP connections involved in spam laundering. Then, we can single out the spam proxies, trace the spam sources behind them, and block the spam traffic. Based on *libpcap*, we implement a prototype of DBSpam and evaluate its effectiveness against email spam laundering through theoretical analyses and trace-based experiments.

In general, DBSpam is distinctive from all previous anti-spam approaches in the following two aspects.

- First, DBSpam pushes the defense line towards spam sources. DBSpam enables an ISP (Internet Service Provider) to on-line detect spam laundering activities and spam proxies inside its customer networks. The quick responsiveness of DBSpam offers the ISP an opportunity to suppress laundering activities and quarantine the identified spam proxies.

- Second, DBSpam has no need to scan message contents, and has very few assumptions about the connections between a spammer and its proxies. DBSpam works even if (1) these connections are encrypted and the message contents are compressed; and (2) a spammer uses proxy chains inside the monitored network.

One additional benefit of DBSpam is that once spam laundering is detected, fingerprinting spam messages at the sender side is viable and spam signatures may be distributed to accelerate spam detection at other places. In addition to all these advantages, DBSpam is complementary to existing anti-spam techniques and can be incrementally deployed over the Internet.

The remainder of the paper is organized as follows. Section 2 briefly presents spam laundering mechanisms. Section 3 surveys commonly-used anti-spam techniques. Section 4 describes the unique behavior of proxy-based spamming. Section 5 details the working mechanism of DBSpam. Section 6 evaluates the effectiveness of DBSpam through the trace-based experiments. Section 7 discusses the robustness of DBSpam against potential evasions. Finally, we conclude the paper with Section 8.

2. SPAM LAUNDERING MECHANISMS

Spam laundering studied in this paper refers to the spamming process, in which only proxies are involved in origin disguise. The proxy refers to the application such as SOCKS that simply performs "protocol translation" (i.e., rewrite IP addresses and port numbers) and tunnels packets through. Different from an email relay, which first receives the whole message and then forwards it to the next mail server, an email proxy requires that the connections on both sides of the proxy synchronize during the message transferring. More importantly, unlike an email relay which inserts the information—"Received From" that records the IP address of sender and the timestamp when the message is received—in front of the message header before relaying the message, an email proxy does not record such trace information during protocol transformation. Thus, from a recipient's perspective, the email proxy, instead of the original sender, becomes the source of the message. It is this identity replacement that makes email proxy a favorite choice of spammers.

Initially, spammers just seek open proxies on the Internet, which usually are mis-configured proxies allowing anyone to access their services. There are many Web sites and free software providing open proxy search function. However, once such mis-configurations are corrected by system administrators, spammers have to find other available "open" proxies.

It is ideal for a spammer to own many "private" and stable proxies. Unsecured home PCs with broadband connections are the best candidates for this purpose. To achieve this, malicious software including specially-designed worms and viruses, such as SoBig and Bagle, is used to hijack home PCs. Equipped with Trojan horse or backdoor programs, these compromised machines are available zombies. After proxy programs such as SOCKS or Wingate are installed, these zombies are ready to be used as proxies to pump out email spam. Without serious performance degradation, most non-professional Windows users are not aware of the ongoing spamming. Recent research on the network-level behavior of spammers [28] also confirms that most sinked spam is originated from compromised Windows hosts.

To counter the soaring growth of spam volume, many ISPs have adopted the policy of blocking port 25 (SMTP port), in which outbound email from a subscriber must be relayed by the ISP-designated email server. In other words, the ISP's edge routers only forward the SMTP traffic from some designated IP addresses to the outside. However, spammers have easily evaded such simple SMTP port blocking mechanisms. The spam laundry is simple: having zombies send spam messages to their ISP email servers first. In February 2005, Spamhaus [2] reported that over the past few months a number of major ISPs had witnessed far more spam messages coming directly from the email servers of other ISPs. This change in proxy-based spamming activity is mainly caused by the use of new stealth spamware, which instructs the hijacked proxy (e.g., zombie) to send spam messages via the legitimate email server of the proxy's ISP.

3. ANTI-SPAM TECHNIQUES

Many anti-spam techniques have been proposed and deployed to counter email spam from different perspectives. Based on the placement of anti-spam mechanisms, these techniques can be divided into two categories: recipient-oriented and sender-oriented. In terms of fighting spam at the source, HoneySpam [13] might be the closest work to ours. In the following, we briefly describe recipient-oriented and sender-oriented techniques, respectively, and then compare our work with HoneySpam.

3.1 Recipient-oriented Techniques

This class of techniques either (1) block/delay email spam from reaching the recipient's mailbox or (2) remove/mark spam in the recipient's mailbox. Due to the flourish of techniques in this category, we further divide them into content-based and non-content-based sub-categories.

3.1.1 Content-based Techniques

The techniques in this sub-category detect and filter spam by analyzing the content of received messages, including both message header and message body.

Email address filters: Email address filters are simply whitelists or blacklists. Whitelists consist of all acceptable email addresses and blacklists are the opposite. Blacklists can be easily broken when spammers forge new email addresses, but using whitelists alone makes the world enclosed. Garriss *et al.* [18] developed a new whitelisting system, which can automatically populate whitelists by exploiting friend-of-friend relationships among email correspondents.

Heuristic filters: The features that are rare in normal messages but appear frequently in spam, such as non-existing domain names and spam-related keywords, can be used to distinguish spam from normal email. SpamAssassin [3] is such an example. Each received message is verified against the heuristic filtering rules. Compared with a predefined threshold, the verification result decides whether the message is spam or not.

Machine learning based filters: Since spam detection can be converted into the problem of text classification, many content-based filters utilize machine-learning algorithms for filtering spam. Among them, Bayesian-based approaches [15, 19, 24, 35] have achieved outstanding accuracy and have been widely used. As these filters can adapt their classification engines with the change of message content, they outperform heuristic filters.

3.1.2 Non-content-based Techniques

The techniques in this sub-category use non-content spam characteristics, such as source IP address, message sending rate, and violation of SMTP standards, to detect email spam.

DNSBLs: DNSBLs are distributed blacklists, which record IP addresses of spam sources and are accessed via DNS queries. When an SMTP connection is being established, the receiving MTA (Mail Transfer Agent) can verify the sending machine's IP address by querying the subscribed DNSBL. Even DNSBLs have been widely used, their effectiveness [22, 28] and responsiveness [27] are still under study.

MARID: MARID (MTA Authorization Records In DNS) is a class of techniques to counter forged email addresses by enforcing sender authentication. MARID is also based on DNS and can be seen as a distributed whitelist of authorized MTAs. Multiple MARID drafts [10] have been proposed, in which [8, 12] have been deployed in some places.

Challenge-Response (C-R): C-R is used to keep the merit of whitelist without losing important messages. Incoming messages, whose sender email addresses are not in the recipient's whitelist, are bounced back with a challenge that needs to be solved by a human being. After a proper response is received, the sender's address can be added into the whitelist.

Tempfailing: Tempfailing [30] is based on the fact that legitimate SMTP servers have implemented the retry mechanism as required by SMTP, but a spammer seldom retries if sending fails. It usually works with a greylist that records the failed messages and the MTAs failed on their first tries.

Delaying: As a variation of rate limiting, delaying is triggered by an unusually high sending rate. Most delaying mechanisms, such as [17, 20, 33, 34] are applied at receiving MTAs.

Sender Behavior Analysis: This technique distinguishes spam from normal email by examining behavior of incoming SMTP connections. Messages from the machine exhibiting characteristics of malicious behavior such as directory harvest are blocked before reaching mailbox [4].

3.2 Sender-oriented Techniques

Usage Regulation: To effectively throttle spam at the source, ISPs and ESPs (Email Service Providers) have taken various measures such as blocking port 25, SMTP authentication, to regulate the usage of email services. Message submission protocol [9] has been proposed to replace SMTP, when a message is submitted from an MUA (Mail User Agent) to its MTA.

Cost-based approaches: Borrowing the idea of postage from regular mail systems, many cost-based anti-spam techniques [5, 11, 14, 23, 32] attempt to shift the cost of thwarting spam from receiver side to sender side. All these techniques assume that the average email cost for a normal user is trivial and negligible, but the accumulative charge for a spammer will be high enough to drive them out of business. Cost concept may have different forms in different proposals. Bonded Sender [5] advocates associating email with real money, while SHRED [23] proposes affixing electronic stamps to messages. Both centralized [11, 23] and distributed [32] cost enforcement mechanisms have been proposed.

3.3 HoneySpam

HoneySpam [13] is a specialized honeypot framework based on honeyd [25] to deter email address harvesters, poison spam address databases, and intercept or block spam traffic that goes through the open relay/proxy decoys set by HoneySpam. With the network virtualization offered by honeyd, HoneySpam can set up multiple fake web servers, open proxies, and open relays. Fake web servers provide specially crafted webpages to trap email address harvesting bots. Fake open proxies or open relays are used to track spammers exploiting them and block spam going through them.

HoneySpam shares the same motivation of countering spam at the source as DBSpam, and both deal with spam proxies. However, the role of proxy and anti-spam approaches in HoneySpam are quite different from those in DBSpam. The proxies of HoneySpam are intentionally set on end hosts, and spam sources are logged by HoneySpam. Thus, spam tracking is very easy. In contrast, detecting spam proxies is the major task of DBSpam, and proxy identification and spam tracking can only be accomplished through traffic analysis. On the other hand, these two tracing and blocking systems are complementary to each other. Moreover, both of them can be used for spam signature generation, spam forensic and law enforcement.

4. PROXY-BASED SPAM BEHAVIOR

In this section, we delineate the distinct behavior of proxy-based spamming, which directly inspire the design of our detecting algorithm. Figure 1 depicts a typical scenario of proxy-based spamming in a customer network such as a Cox regional residential network. Although spammers can conceal their real identities from destination MTAs by exploiting spam proxies, they cannot make the connection between a spam source and its proxy *invisible* to the edge router or gateway that sits in between. Here we assume that there is a network vantage point where we can monitor all the bi-directional traffic passing through the customer network, and the location of the gateway (or firewall) of the customer network (e.g. edge router R in Figure 1) that connects to the Internet is such a point.

4.1 Laundry Path of Proxy Spamming

As shown in Figure 1, there is a customer network N, in which spam proxies reside. Both spammer S and receiving MTA M are connected to customer network N via edge router R. S may be the original spam source or just another spam proxy (but it must be closer to the real spam source). M is the outside MTA.

Note that for the customer network that has its own mail server(s) such as campus (or enterprise) networks, the monitored network N may not be the whole network, but one of its protected sub-networks. Usually such campus/enterprise networks are divided into multiple sub-networks for security and management concerns. Their mail servers are placed in DMZ (DeMilitarized Zone) or a special sub-network that is separated from other sub-networks such as wireless, dormitory, or employee sub-networks. It is one of these loosely-

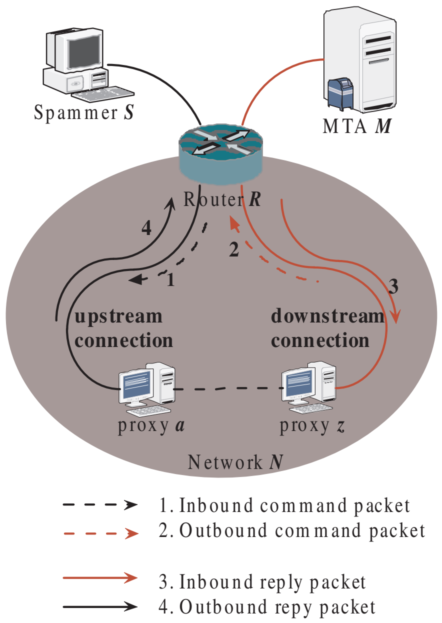

Figure 1: Scenario of Proxy-based Spamming

managed sub-networks that becomes the monitored network N and the router/gateway connecting the sub-networks becomes the vantage point R. Thus, the assumption of exterior MTA M is valid even when the MTA is under the the same administration domain as network N.

Inside monitored network N, S may use a single or multiple spam proxies. If multiple proxies are employed, they may either launder spam messages individually or be organized into one or multiple proxy chains, depending on the spammer's strategy. Without loss of generality, only one chain is shown in Figure 1. Spammer S usually communicates with spam proxies through SOCKS or HTTP. The spam message sent from S to a may even be encrypted. If it is a proxy chain, the spam message can be conveyed by different proxy protocols at different hops. For instance, SOCKS 4 is used between S and a, while HTTP is employed between a and z. However, all these protocol variations and message content encryptions cannot change the fact: it is last-hop proxy z [1] that does the protocol transformation and forwards the spam message to the MTA via SMTP.

We define the connection between spammer S and first-hop proxy a as the *upstream* connection, and define the connection between last-hop proxy z and MTA M as the *downstream* connection. The upstream and downstream connections plus the proxy chain form the spam laundry path, which is shown in Figure 1.

4.2 Connection Correlation

There is a one-to-one mapping between the upstream and downstream connections along the spam laundry path. While this kind of connection mapping is common for proxy-based spamming, it is very unusual for normal email transmission. In normal email delivery, there is only one connection, i.e., the connection between sender and receiving MTA. The existence of such connection correlation is a strong indication of spam laundering and provides valuable clue for spammer

[1]proxy z and proxy a are the same in the single proxy scenario.

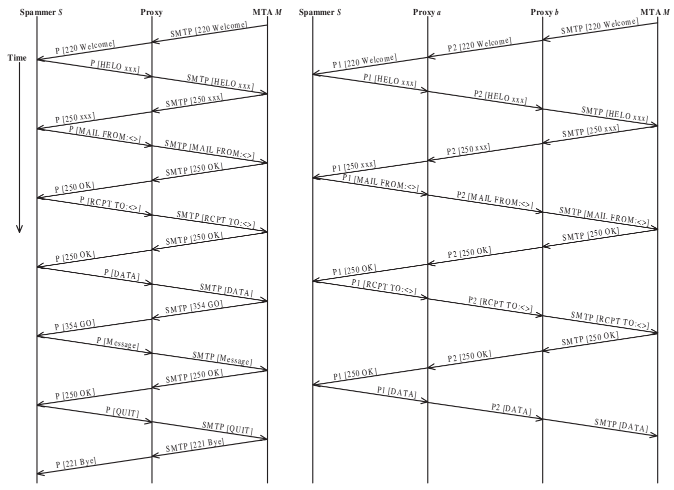

Figure 2: Time-line of Spamming Processes for Single Proxy (left) and Proxy Chain (right)

tracking. Here we assume that the downstream connection is an SMTP connection. For the upstream connection we have no restriction except that it should be a TCP connection. The packets in the upstream connection may be encrypted and even compressed.

The detection of such spam-proxy-related connection correlation is challenging due to the following three reasons. First, content-based approaches could be ineffective as spammers may use encryption to evade content examination. Second, because such a detection mechanism is usually deployed at network vantage points, the induced overhead should be affordable, which is critical to the success of its deployment. Third, since spam traffic is machine-driven and could be delayed by proxy at will, those timing-based correlation detection algorithms such as [36] may not work well in this environment.

4.3 Packet Symmetry

Figure 2 illustrates the detailed communication processes of spam laundering for both single proxy and proxy chain cases at the application layer, in which the message format is "PROTOCOL [content]". For simplicity, P/P1/P2 stands for different application protocols, including SOCKS (v4 or v5), HTTP, etc. For SMTP, its packet content is in plain-text. But for application protocols P/P1/P2, their packet contents may be encrypted. For ease of presentation, the small delays introduced by message processing at end hosts and intermediate proxies are ignored. The initial proxy handshaking process is also omitted as it has no effect on email transactions. Without losing any generality, here we only show the shortest SMTP transaction process for the single-proxy case and parts of SMTP transaction process for the proxy-chain case.

Due to protocol semantics, the process of proxy-based spamming is similar to that of an interactive communication. The appearance of one inbound SOCKS-encapsulated (or HTTP-encapsulated) [2] SMTP command message on the upstream connection will trigger the occurrence of one outbound SMTP command message on the downstream connection later. Similarly, for each inbound SMTP reply message

[2]For the ease of presentation, we only use SOCKS in the rest of paper, although HTTP can be used as well.

on the downstream connection, later on there will be one corresponding outbound SOCKS-encapsulated reply message carried by TCP on the upstream connection. We term this communication pattern as *message symmetry*.

This message symmetry leads to the *packet symmetry* at the network layer with a few exceptions, in which the one-to-one packet[3] mapping between the upstream and downstream connections may be violated. The exceptions can be caused by (1) packet fragmentation, (2) packet compression, (3) packet retransmission occurring along the laundry path. However, due to the fact that SMTP reply messages are very short (usually less than 300 bytes including packet header) and Path MTUs for most customer networks are above 500 bytes, the occurrence of (1) and (2) is very rare. Moreover, the packet retransmission problem can be easily resolved by checking TCP sequence numbers. In general, the packet symmetry between the inbound and outbound reply packets holds most of time.

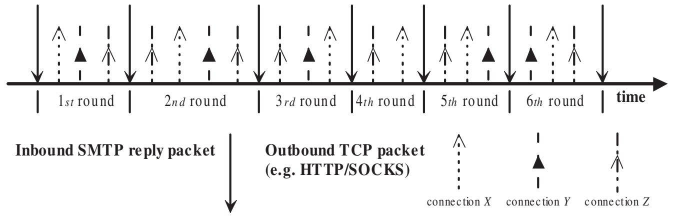

Figure 3: Example of Reply Round and TCP Correlation

Such packet symmetry is exemplified in Figure 3, where the arrow with long solid line stands for the arrival of an inbound SMTP reply packet of the suspicious SMTP connection. In addition to the inbound SMTP connection, there are three outbound TCP connections X, Y, and Z, as shown in Figure 3. Three kinds of arrows with different dotted lines stand for the arrivals of outbound TCP packets belonging to these outbound TCP connections, respectively. The upward arrow indicates that the packet is leaving the monitored network, while the downward arrow indicates the packet is entering the network.

All of the inbound SMTP reply packets shown in Figure 3 belong to the same suspicious SMTP connection. We define a *reply round* as the time interval between the arrivals of two consecutive reply packets on an SMTP connection. Thus, the n_{th} reply round is the time interval between the arrival of the n_{th} reply packet and that of the $(n+1)_{th}$ reply packet. Even for the simplified SMTP transaction, it has six reply rounds as shown in Figure 3. Within one reply around, the number of arrows with a specific dotted line indicates the number of outbound TCP packets of the corresponding TCP connection.

According to the one-to-one mapping of packet symmetry, each SMTP reply packet observed on the downstream SMTP connection should cause *one and only one* TCP packet appeared on the upstream connection. As Figure 3 shows, if one connection among X, Y, and Z is the suspicious upstream connection, one and only one outbound TCP packet must be observed from that connection in every reply round. Based on this rule, only TCP connection X meets this "one and only one" requirement and can be classified as the suspicious upstream connection with high probability. In the second reply round, more than one packets appear on connection Z; and in the fourth round, no packet occurs on connection Y. Thus, we can easily filter out TCP connections Y and Z as normal background traffic. Note that the order of packet arrivals in a reply round does not affect the checking result of packet symmetry.

This *packet symmetry* is the key to distinguish the suspicious upstream and downstream connections along the spam laundry path from normal background traffic. It simply captures the fundamental feature of chained interactive communications, and does not assume any specific time distribution of packet arrivals. We use this simple rule to detect the laundry path of proxy-based spamming, and the detection scheme is robust against any possible time perturbation induced by spammers. Note that the *one and only one* mapping of packet symmetry can be relaxed, which we will discuss in Section 7.

5. WORKING MECHANISM OF DBSPAM

DBSpam consists of two major components: spam detection module and spam suppression module, in which the detection module is the core of DBSpam. To the best of our knowledge, so far there is no effective technique which can on-line detect both spam proxies and the corresponding spammers behind them. We envisage that DBSpam may achieve the following goals: (1) fast detection of spam laundering with high accuracy; (2) breaking spam laundering via throttling or blocking after detection; (3) support for spammer tracking and law enforcement; (4) support for spam message fingerprinting; and (5) support for global forensic analysis.

In essence, the detection module of DBSpam is a simple and efficient *connection correlation detection* algorithm to identify the laundry path of spam messages (i.e., the suspicious downstream and upstream connections) and the spam source[4] that drives spamming behind the proxies.

5.1 Deployment of DBSpam

Like other network intrusion detection systems, DBSpam needs to be placed at a network vantage point that connects a customer network to the Internet, where it can monitor the bi-directional traffic of the customer network. For a single-homed network, it is easy to locate such a network vantage point (an edge router or a firewall) and deploy DBSpam on it. For a multi-homed network, it may not be possible to locate a single network vantage point that can monitor all the bi-directional traffic passing through the customer network.

However, on one hand, many customer networks use multi-homing not for load-balance, but for reliability and fault-tolerance. Therefore, in case of the backup multi-homing, DBSpam works well if deployed at the primary ISP edge router. On the other hand, even in the load-balance multi-homing scenario, as long as the packets that belong to the same proxy chain go through the same ISP edge router or firewall, DBSpam still can work at different ISP edge routers or firewalls without coordination. Moreover, there are special network devices (e.g., [6]) which can passively aggregate traffic from multiple network segments. By hooking up to

[3]TCP control packets such as SYN, ACK are not counted here.

[4]Or just another spam proxy that is outside the customer network but at least one more step closer to the real source.

such devices, DBSpam can still have the complete view of network traffic.

5.2 Design Choices and Overview

Our goal is to detect the spam laundry path promptly and accurately, once a proxy-based spamming activity occurs on the monitored network. We show in the previous section that packet symmetry is the inherent characteristic of proxy-based spamming behavior. Since legitimate messages are rarely delivered along the path illustrated in Figure 1, the possibility of a normal SMTP connection being consistently correlated with an unrelated TCP connection is very small in terms of packet symmetry. Hence, frequent observations of connection correlation is a strong indication of occurrence of spam laundering.

According to the packet symmetry rule, for the upstream TCP connection along a spam laundry path, its outbound packet[5] number in each reply round of the downstream SMTP connection is always one. For a normal TCP connection, however, this rule can only be satisfied with a very small probability. Thus, a simple and intuitive correlation detection method is to count the number of outbound packets observed on suspicious TCP connections in sequential reply rounds of an SMTP connection. Given the characteristic of successive arrival of observations, this correlation detection problem is well suited for the statistical method of *Sequential Probability Ratio Test* (SPRT) developed by Wald [31].

As a simple and powerful mathematical tool, SPRT has been used in many areas such as portscan detection [21] and wireless MAC protocol misbehavior detection [26]. Basically, an SPRT can be viewed as an one-dimensional random walk. The walk starts from a point between two boundaries and can go either upward or downward with different probabilities. With each arrival of observation, the walk makes one step in the direction determined by the result of observation. Once the walk firstly hits or crosses either the upper boundary or the lower boundary, it terminates and the corresponding hypothesis is selected. For SPRT, its actual false positive probability and false negative probability are bounded by predefined values. It has been proved that SPRT minimizes the average number of required observations to reach a decision among all sequential and non-sequential tests, which do not have larger error probabilities than SPRT.

We utilize the packet symmetry of SMTP reply packets to detect proxy-based spamming activity. Basically, we monitor the inbound SMTP traffic first, then apply the rule of packet symmetry for detecting the spam laundry path inside the customer network. In other words, DBSpam focuses on the clock-wise reply packet flow as shown in Figure 1, instead of the counter-clock-wise command packet flow, for connection correlation detection. The arrivals of inbound SMTP reply packets, which delimit the reply rounds and drive the progress of connection correlation detection, become a self-setting clock of the detection algorithm. SPRT terminates by either selecting the hypothesis that $C_{\texttt{tcp}}$ is correlated with $C_{\texttt{smtp}}$ or choosing the opposite hypothesis.

There are two benefits of using SMTP reply messages to drive SPRT. First, as mentioned earlier, SMTP reply messages are very small, which minimizes the occurrence of packet fragmentation; and we can significantly increase the processing capacity of DBSpam by monitoring small packets only. Second, being either the spam target or the relay, the remote SMTP servers are usually very reliable; and the implementation and listening port of these servers strictly follow the SMTP protocol semantics. Thus, the packet symmetry rule always holds, and SMTP packets can be easily identified based on the port number of TCP header.

In the rest part of the section, we first briefly describe the basic concept of SPRT, then present the detection module of DBSpam, which include two phases: SPRT detection and noise reduction.

5.3 Sequential Probability Ratio Testing

Let X_i, $i = 1, 2, \ldots$, be random variables representing the events observed sequentially. The SPRT for a simple hypothesis H_0 against a simple alternative H_1 has the following form:

$$\begin{aligned} \Lambda_n \geq B &\Longrightarrow \text{ accept } H_1 \text{ and terminate test,} \\ \Lambda_n \leq A &\Longrightarrow \text{ accept } H_0 \text{ and terminate test,} \\ A < \Lambda_n < B &\Longrightarrow \text{ conduct another observation.} \end{aligned} \tag{1}$$

where two constants or boundaries A and B satisfy $0 < A < B < \infty$, and Λ_n is the log-likelihood ratio defined as follows:

$$\Lambda_n = \lambda(X_1, \ldots, X_n) = \ln \frac{\Pr(X_1, \ldots, X_n | H_1)}{\Pr(X_1, \ldots, X_n | H_0)}. \tag{2}$$

Assume $X_1, \ldots, X_i$ are independent and identically distributed (i.i.d) Bernoulli random variables with

$$\Pr(X_i = 1 | \theta) = 1 - \Pr(X_i = 0 | \theta) = \theta, \tag{3}$$

then

$$\Lambda_n = \ln \frac{\prod_1^n \Pr(X_i | H_1)}{\prod_1^n \Pr(X_i | H_0)} = \sum_1^n \ln \frac{\Pr(X_i | H_1)}{\Pr(X_i | H_0)} = \sum_1^n Z_i, \tag{4}$$

where $Z_i = \ln \frac{\Pr(X_i|H_1)}{\Pr(X_i|H_0)}$. Λ_n can be viewed as a random walk (or more properly a family of random walks[6]) with steps Z_i which proceeds until it first crosses boundary A or B. Suppose the distributions for H_1 and H_0 are θ_1 and θ_0, respectively. Λ_n moves up with step length $\ln \frac{\theta_1}{\theta_0}$ when $X_i = 1$, and goes down with step length $\ln \frac{1-\theta_1}{1-\theta_0}$ when $X_i = 0$.

In SPRT, we define two types of error

$$\alpha = \Pr(S_1 | H_0), \qquad \beta = \Pr(S_0 | H_1),$$

where $\Pr(S_i | H_j)$ denotes the probability of selecting H_i but in fact H_j is true. If we call the selection of H_1 detection and the selection of H_0 normality, the event of $S_1 | H_0$ can be viewed as a false positive. So, α represents the false positive probability. Likewise, the event of $S_0 | H_1$ can be termed a false negative and β represents false negative probability.

Let α^* and β^* be user-desired false positive and false negative probabilities, respectively. According to (1), we can derive[7] the *Wald boundaries* as follows:

$$A = \ln \frac{\beta^*}{1 - \alpha^*}, \qquad B = \ln \frac{1 - \beta^*}{\alpha^*}, \tag{5}$$

[5]Here packets refer to non-retransmitted, non-zero-payload TCP packets.

[6]It is a family of random walks, since the distribution of the steps depends on which hypothesis is true.

[7]Due to the space limitation, the derivations of (5), (6), and (7) are omitted here. See [21, 31] for details.

and the derived relationships between actual error probabilities and user-desired error probabilities are:

$$\alpha \leq \frac{\alpha^*}{1-\beta^*}, \qquad \beta \leq \frac{\beta^*}{1-\alpha^*}, \tag{6}$$

$$\alpha + \beta \leq \alpha^* + \beta^*. \tag{7}$$

Inequality (6) suggests that the actual error probabilities α and β can only be slightly larger than their expected values α^* and β^*. For example, if the desired α^* and β^* are both 0.01, then their actual values α and β will be no greater than 0.0101. Inequality (7) can be interpreted as that the sum of actual error probabilities is bounded by the sum of their desired values.

According to Wald's theory, $E[N] = E[\Lambda_N]/E[Z_i]$. Suppose hypothesis H_1 is true and Bernoulli variable X_i has distribution θ_1 which implies that Λ_n steps up with probability θ_1 or goes down with probability $1-\theta_1$, we have

$$E[Z_i|H_1] = \theta_1 \ln\frac{\theta_1}{\theta_0} + (1-\theta_1)\ln\frac{1-\theta_1}{1-\theta_0}. \tag{8}$$

If the user-desired false negative probability of the test is β^*, then the true positive probability is $1-\beta^*$ and

$$\begin{aligned} E[\Lambda_N|H_1] =& \beta^* A + (1-\beta^*)B \\ =& \beta^* \ln\frac{\beta^*}{1-\alpha^*} + (1-\beta^*)\ln\frac{1-\beta^*}{\alpha^*}. \end{aligned} \tag{9}$$

With (8) and (9), we have

$$E[N|H_1] = \frac{\beta^* \ln\frac{\beta^*}{1-\alpha^*} + (1-\beta^*)\ln\frac{1-\beta^*}{\alpha^*}}{\theta_1 \ln\frac{\theta_1}{\theta_0} + (1-\theta_1)\ln\frac{1-\theta_1}{1-\theta_0}}. \tag{10}$$

Likewise, we can derive

$$E[N|H_0] = \frac{(1-\alpha^*) \ln\frac{\beta^*}{1-\alpha^*} + \alpha^*\ln\frac{1-\beta^*}{\alpha^*}}{\theta_0 \ln\frac{\theta_1}{\theta_0} + (1-\theta_0)\ln\frac{1-\theta_1}{1-\theta_0}}. \tag{11}$$

Apparently the average observation number $E[N]$ of SPRT is determined by four parameters: predefined error probabilities α^*, β^* and distribution parameters θ_0 and θ_1. The determination of these values and their effect on $E[N]$ will be discussed with our correlation detection algorithm in the following.

5.4 SPRT Detection Algorithm

According to the principle of packet symmetry, within each reply round, there must be one and only one outbound TCP packet appearing on the corresponding upstream connection. By contrast, those connections that have none or more than one TCP packet can be classified as innocent connections. Within the framework of SPRT, this correlation detection problem can be easily transformed into an SPRT, in which we test the hypothesis H_1 that $C_{\texttt{tcp}}$ is correlated with $C_{\texttt{smtp}}$ against the hypothesis H_0 that two connections are uncorrelated by counting the number of TCP packets appearing on $C_{\texttt{tcp}}$ in each reply round of $C_{\texttt{smtp}}$.

If we use a Bernoulli random variable X_i to represent the observation result on $C_{\texttt{tcp}}$ in i-th reply round of $C_{\texttt{smtp}}$ and assume that these variables in different rounds are i.i.d, we have the following distribution:

$$X_i|H_1 = \begin{cases} \theta_1 & \text{if one outbound TCP packet observed} \\ 1-\theta_1 & \text{otherwise} \end{cases}$$

Algorithm 1 Detect-Correlation

```
 1: Input: $C_{\texttt{tcp}}$, $C_{\texttt{smtp}}$
 2: Para: $A, B$
 3: Output: $C_{\texttt{tcp}}$ is correlated with $C_{\texttt{smtp}}$ or not
 4: repeat
 5:   for each reply round of $C_{\texttt{smtp}}$ do
 6:     if # of outbound packets on $C_{\texttt{tcp}}$ is 1 then
 7:       $\Lambda_n \leftarrow \Lambda_n + \ln\frac{\theta_1}{\theta_0}$
 8:     else
 9:       $\Lambda_n \leftarrow \Lambda_n + \ln\frac{1-\theta_1}{1-\theta_0}$
10:     end if
11:     if $\Lambda_n \geq B$ then
12:       $C_{\texttt{tcp}}$ is correlated with $C_{\texttt{smtp}}$ and the test stops
13:     else if $\Lambda_n \leq A$ then
14:       $C_{\texttt{tcp}}$ is not correlated with $C_{\texttt{smtp}}$ and the test
          stops
15:     else
16:       wait for observation in next reply round
17:     end if
18:   end for
19: until $C_{\texttt{smtp}}$ is closed
```

$$X_i|H_0 = \begin{cases} \theta_0 & \text{if one outbound TCP packets observed} \\ 1-\theta_0 & \text{otherwise} \end{cases}$$

The algorithm of detecting connection correlation can be expressed in Algorithm 1.

For proxy-based spamming, given that packet symmetry holds most of time, the major reason that correlation cannot be detected without is mainly attributed to the packet misses by the monitoring system. For example, when the traffic volume exceeds the capacity that the monitoring system can handle, packets may be dropped by the monitoring system. If the packet conveying SMTP reply message is dropped on either the downstream connection or the upstream connection, the correlation detection will fail in this reply round. So we can use packet miss rate to estimate the probability of a proxy connection being correlated when spamming occurs, i.e. θ_1. From the conservative perspective, we take 0.01 as the packet miss rate which in fact is fairly high[8] considering only small packets (say less than 300 bytes) need attention and only packet header information is required for detection algorithm. So θ_1 is 0.99 in this case.

To estimate θ_0, we employ the mathematical model given in [16]. We assume that the uni-directional packet arrivals of a normal TCP connection can be modeled as a non-homogeneous Poisson process, which can be approximated by a sequence of Poisson processes with varying rates, and over varying time periods that could be arbitrarily small. For example, let $M(t)$ denote the number of packets sent in an outbound TCP connection during time interval t. Process $\{M(t), t \geq 0\}$ can be represented by a sequence of Poisson processes $(\lambda_1, \Delta t_1)$, $(\lambda_2, \Delta t_2)$, $\cdots$, where $t = \Delta t_1 + \Delta t_2 + \cdots$. The advantage of this model is to approximate almost any distribution. More importantly, the number of packets observed during any given time interval T, can be represented by a Poisson process M with a single rate $\hat{\lambda}_T$. Here $\hat{\lambda}_T$ is the weighted mean of the rates of all the Poisson processes during T.

With this model, we can easily compute the probability of

[8]In practice, the miss rate is usually below 0.005 in our campus network.

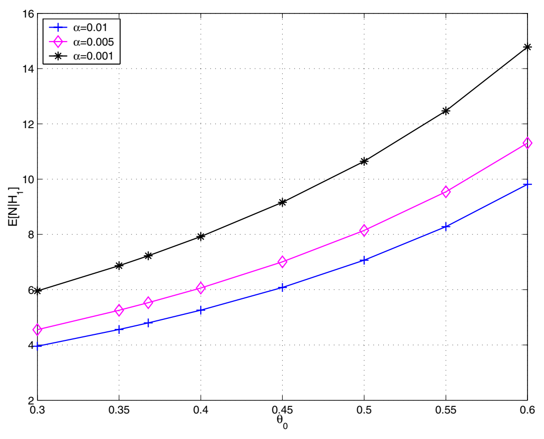

Figure 4: $E[N|H_1]$ **vs.** θ_0 **and** α^* $(\theta_1 = 0.99, \beta^* = 0.01)$

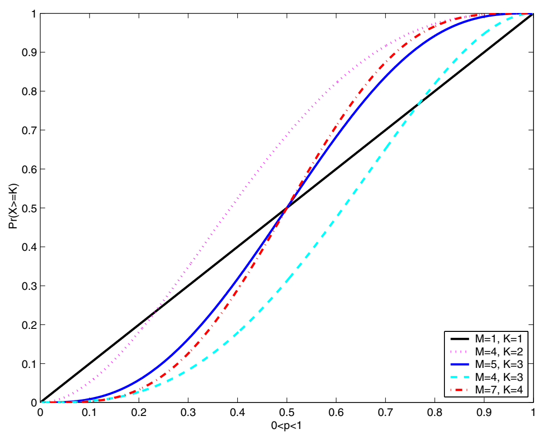

Figure 5: $\Pr(X \geq K)$ **vs.** p **and** (M, K)

one and only one packet sent in a reply round if T denotes the duration of a reply round. From

$$\Pr(M = i) = e^{-(\hat{\lambda}_T T)} \frac{(\hat{\lambda}_T T)^i}{i!}, \tag{12}$$

we have

$$\Pr(M = 1) = e^{-(\hat{\lambda}_T T)} (\hat{\lambda}_T T) \leq e^{-1}. \tag{13}$$

In (13) $\Pr(M = 1)$ reaches its maximum value e^{-1} when $\hat{\lambda}_T T = 1$. Although this is a theoretical derivative, we find that it is valid on almost all of the evaluated traces. Thus, we set $\theta_0 = e^{-1}$.

If we choose 0.005 for false positive probability α^* and 0.01 for false negative probability β^*, with $\theta_0 = e^{-1}$ and $\theta_1 = 0.99$, $E[N|H_1]$ is 5.5 and $E[N|H_0]$ is 2.02, respectively. Figure 4 shows how $E[N|H_1]$ varies with the changes of α^* and θ_0, when β^* and θ_1 are fixed. In general, $E[N|H_1]$ increases when θ_0 gets bigger or α^* gets smaller. Intuitively, this prolonged random walk is a natural result of smaller step length $\ln \frac{\theta_1}{\theta_0}$ or enlarged distance $\ln \frac{1-\beta^*}{\alpha^*}$ for the walk towards upper threshold.

From the perspective of anomaly detection, it is desirable that error probabilities, especially the false positive probability, can be as low as possible. In the framework of SPRT, this implies that $E[N|H_1]$ goes up, i.e., the average detection time is prolonged. However, given that not all SMTP transactions (the shortest one has only 6 reply rounds) can be longer enough to make the SPRT reach a decision when α is too small, a tradeoff between lowering false positive and false negative has to be made. In DBSpam, we set $\alpha^* = 0.005$ so that even the shortest spam transactions can be captured.

5.5 Noise Reduction

To further lower the false positives of SPRT, we introduce a simple and effective noise reduction technique in DBSpam. In a series of correlation tests, we define the active spam sources and proxies that are prone to be identified many times as signals, and define those innocent IP addresses that may be accidentally captured as noises. We utilize the dichotomy between signal and noise to distinguish spam sources and proxies from innocent end-hosts. We call this procedure *noise reduction*. The noise reduction are executed in two steps: first, we maintain a set S_i of external IP addresses that appear in the correlation results for each time window Δ; second, in the consecutive M time windows, we single out the external IP addresses, which appear no fewer than K times, as the spam sources and the corresponding proxy addresses as the spam proxies.

The time window Δ is determined by the lower-bound of spamming rate υ (in replies/s) and the number of reply rounds N:

$$\Delta \geq N/\upsilon. \tag{14}$$

Hence, a spammer sending spam faster than υ must appear in S_i at least once in each time window Δ. Assume that the appearance of an IP address in S_i is independent, with a constant probability p. Then, the number of occurrences of the IP address among M time windows follows the *binomial* distribution.

$$\Pr(X = i) = \binom{M}{i} p^i (1-p)^{M-i}. \tag{15}$$

The probability of having no fewer than K occurrences in the binomial distribution is:

$$\Pr(X \geq K) = \sum_{i=K}^{M} \binom{M}{i} p^i (1-p)^{M-i}. \tag{16}$$

Figure 5 illustrates the dynamics of $\Pr(X \geq K)$ with the variation of probability p for several pre-determined tuples of (M, K). The diagonal line shows the case of tuple $(M = 1, K = 1)$, in which $\Pr(X \geq K)$ is equal to p. Clearly, if p is smaller than 0.2, all other curves are below this diagonal line, indicating that their values of $\Pr(X \geq K)$ are smaller than that of tuple $(M = 1, K = 1)$. In contrast, if p is larger than 0.8, these curves are above the diagonal line, indicating that their values of $\Pr(X \geq K)$ are larger than that of tuple $(M = 1, K = 1)$.

The value of p for an innocent address depends on the false positive rate of the correlation detection, which should be closer to zero than one. The left part of Figure 5 illustrates the noise reduction can further lower the chance of an innocent address being mis-classified as a spam source. On the other hand, the value of p for a spam source is related to the complementary of the false negative rate of the correlation detection, which should be closer to one than zero as shown in the right part of Figure 5. This indicates that noise reduction increases the probability of a spam source being identified as well. Therefore, both false positives and false negatives are reduced after noise reduction. Figure 5

Table 1: Trace Information

Attribute	S-1-A	S-1-B	S-1-C	S-2-A	S-2-B	S-2-C	N-1	N-2
duration (sec)	770	674	756	654	1,385	1,398	5,116	14,944
# of packets	3,872,550	4,178,567	4,509,336	12,036,413	26,422,563	26,172,898	24,434,518	297,733,228
avg packet/sec	5,029	6,200	5,965	18,404	19,078	18,722	4,776	19,923
trace size	295MB	318MB	343MB	931MB	2,044MB	2,018MB	1,851MB	22.4GB
packet miss rate	< 0.001	< 0.001	< 0.001	0.008	0.005	0.005	<0.001	0.006
# of threads/spammer	1	3	1	1	3	1	-	-

shows that when M is fixed, the probability $\Pr(X \geq K)$ goes smaller with bigger K. For example, $\Pr(X \geq 3|M = 4)$ is much smaller than $\Pr(X \geq 2|M = 4)$. Moreover, the noise reduction algorithm works very well even with very small M and K. For example, with $(M = 4, K = 3)$, pre-noise-reduction false positive rate, which is 0.1, can be significantly lowered to 0.0037 after noise reduction. These two rules of thumb may guide the selection of (M, K) in practice. We will further discuss the parameter setup of Δ, M and K, and demonstrate the effectiveness of the noise reduction technique in Section 6.3.2.

6. SYSTEM EVALUATION

We implemented a prototype of DBSpam using *libpcap* on Linux. Due to access limitation, we cannot deploy our prototype in an ISP network environment to evaluate its on-line performance. Alternatively, we collected traces from a middle-sized campus network and conducted a series of trace-based experiments to validate the efficacy of DBSpam.

By replaying the collected traces with our prototype, we attempt to answer the following questions: (1) how fast DBSpam can detect spam laundering; (2) how accurate the detection result of DBSpam is; (3) how many system resources DBSpam consumes.

6.1 Data Collection

The campus network is connected to the Internet via an OC-3 data link. A Snort-based NIDS [29] is deployed on the edge router of the campus network to block any suspicious proxy traffic (e.g. SOCKS and HTTP) via signature checking. All outgoing email messages must go through the main email server and secure authentication is enforced.

This well-protected campus network provides an ideal platform to assess the false positive ratio of DBSpam on normal network traffic. According to the IT department, proxy-based spamming activities on this campus network are very rare. To evaluate the detection time and accuracy of DBSpam on spam laundering, we generate "spam" traffic, including both plain-text and encrypted proxy traffic, with the cooperation of the IT department. Although the monitoring systems of IT can detect plain-text proxy traffic by checking content, our encrypted proxy traffic successfully evades their detection.

The generated spamming scenario is similar to the one shown in Figure 1. The campus network plays the role of network N. We use two home PCs outside the campus network, which are located in two different ISP broadband networks, to emulate two spam sources. The spam sink (MTA M in Figure 1) is located in the dark net of the campus network. The dark net is a special subnet that directly links to the edge router and is used to dump all malicious traffic. Two SOCKS and HTTP proxies run in two different subnets of the campus network to form a proxy chain. We use a common spamware and *sockschain* [9] to emulate proxy-chain spamming. The spam messages are sent from the two home PCs, through the proxy chain and destined to the spam sink. The data collection point is just before the edge router and can see all the traffic passing through the edge router. We use *tcpdump* to capture all small bi-directional TCP packets with the `snaplen` set to 75 bytes.

We collected multiple traces of normal and spam traffic in two different months. The detailed information of the traces is listed in Table 1, and additional explanations are given below. First, we only captured small TCP packets with packet length less than 300 bytes as DBSpam only utilizes the SMTP reply messages for detection, which are usually conveyed by TCP packets with length less than 300 bytes. Second, We collected two kinds of traces to evaluate the performance of DBSpam, one with generated spam traffic and the other without generated spam traffic. All traces include the normal background SMTP traffic passing through the campus network. The name of a trace follows the format "{S|N}-{1|2}-{A|B|C}". S (N) indicates that the trace has Spam (No spam) traffic. 1 (2) refers to the different month of trace collection. A (B, C) is only for spam traces and stands for different spam scenario. Third, in order to validate DBSpam for detecting both plain-text and encrypted spam traffic, we injected encrypted and compressed spam traffic through SSH tunneling into traces S-*-C (* is either 1 or 2), and injected plain-text spam traffic into S-*-A and S-*-B. Fourth, a multi-threaded spamming technique was used in S-*-B to validate the efficacy of DBSpam in a multi-threaded spamming scenario. The N-threaded spamming means up to N upstream connections may be issued simultaneously from the spam source to a proxy for spam laundering.

6.2 Detection Time

The overall detection time of DBSpam is determined by SPRT detection time, the noise-reduction time window Δ, and the number of consecutive windows M. Among these three factors, SPRT detection time is the fundamental one, which bounds the value of time window Δ. In the following, we focus on the estimation of SPRT detection time.

6.2.1 SPRT Detection Time

We evaluate SPRT detection time from two perspectives: the number of observations needed to reach a decision and the actual time spent by SPRT.

Number of Observations N: The theoretical average number of observations under spam hypothesis ($E[N|H_1]$) and non-spam hypothesis ($E[N|H_0]$) can be easily computed based on Equations (10) and (11). In our evaluation, they are rounded to 6 and 3, respectively, with $\alpha^* = 0.005$, $\beta^* = 0.01$, $\theta_0 = e^{-1}$, and $\theta_1 = 0.99$. Table 2 shows the distribution of $N|H_1$ in six spam traces. The results clearly

[9]Both are binary Windows programs so that we cannot modify any code.

Table 2: Distribution of $N|H_1$

Trace	$N = 6$	$N = 11$	$N >= 16$
S-1-A	970 (100%)	0	0
S-1-B	5019 (96.9%)	139 (2.7%)	21 (0.4%)
S-1-C	2245 (92.8%)	169 (7.0%)	6 (0.2%)
S-2-A	433 (99.1%)	3 (0.7%)	1 (0.2%)
S-2-B	4298 (94.7%)	198 (4.4%)	40 (0.9%)
S-2-C	1758 (98.9%)	16 (1.0%)	3 (0.1%)

demonstrate the dominance of ($N = 6$) in all traces. The comparatively low percentage of ($N = 6$) in trace S-1-C is mainly caused by the abnormally high packet-miss-rate of the spam traffic but not the whole traffic. Note that due to the characteristics of SPRT, the detection of connection correlation (H_1) can only be reached after certain number of observations, such as 6 and 11.

Figure 6 shows the distribution of $N|H_0$ for non-spam traces N-1 and N-2. The curves indicate that SPRT can filter out at least 95% of normal connections within four observations. The distributions of $N|H_0$ for spam traces are similar to those for non-spam traces.

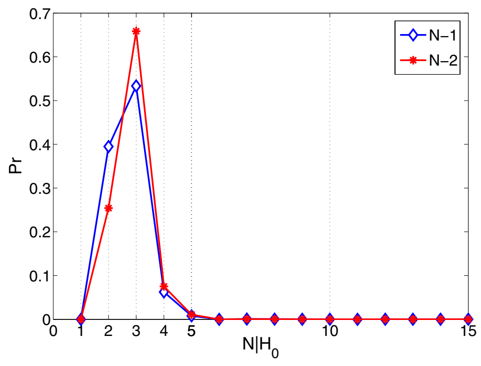

Figure 6: Distribution of $N|H_0$

Actual Detection Time of SPRT: After recording the start and end points for each SPRT on six spam traces, we derive all the detection time in these traces and draw their CDF (cumulative distribution function) in Figure 7. The detection time is approximated by ceiling for CDF drawing, e.g., 1.2s is ceiled to 2s. We classify the results from six traces into two groups: "S-1" and "S-2", since the results in each group are very similar. As shown in Figure 7, 95% detections are made within 5 seconds. Note that the actual detection time is roughly the duration of 6 reply rounds of SMTP connection, since the computation overhead of SPRT is negligible. The curve difference between "S-1" and "S-2" is due to the inferior link quality in "S-2" experiments.

6.3 Detection Accuracy

Since the detection module of DBSpam has two phases—SPRT detection and noise reduction, we first evaluate the false positive and false negative of SPRT detection, and then present the overall detection accuracy of DBSpam after noise reduction.

6.3.1 *Accuracy of SPRT*

False Positives: The upper part of Table 3 shows the false positives of SPRT in different traces. The "detection" row is the total number of correlations reported by SPRT, and "True Positives (TP)" and "False Positives (FP)" rows list the outcome of detections. The "True Negatives (TN)"

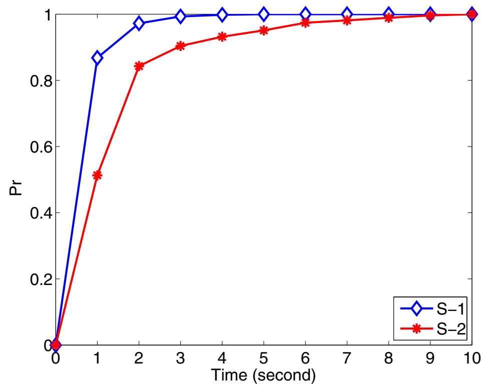

Figure 7: CDF of Detection Time for SPRT

row lists the number of tests on normal connections that are correctly identified. According to the definition of false positive probability $\alpha = \frac{FP}{FP+TN}$, the probabilities in all traces are well below 0.0002, indicating that the false positive probability of SPRT is fairly small in practice.

False Negatives: We estimate the false negatives by counting the number of proxy connections that are missed by SPRT, and compute the ratio of missed spam connections, which are shown in the lower part of Table 3. The false negatives of SPRT are attributed to the missed packets in the spam traces. The three spam traces S-2-A/B/C contain both long SMTP connections (more than 10 reply rounds) and short SMTP connections (six reply rounds). More than 70% of the total connections are short SMTP connections. For those short spam connections with only six reply rounds, if any packet on either the upstream connection or the downstream connection is missed in the trace, SPRT cannot reach a decision, leading to a false negative. A simple estimation shows the feasibility of the missing ratio of spam connections. For simplicity, we assume that the packet miss rate p is constant through the trace. Then, the probability of one packet missing in six reply rounds is $\binom{12}{1}p(1-p)^{11}$. If $p = 0.005$ (the packet miss rate of traces S-2-B/C), the probability is around 0.057, which is more than the miss ratio as shown in Table 3.

6.3.2 *DBSpam Accuracy after Noise Reduction*

To investigate the efficacy of noise-reduction, we first need to determine the value of time window Δ. Figure 7 shows that over 80% of all SPRTs on spam traces terminate within 2 seconds. So, we set the time window Δ to 2 seconds. For (M, K), we test several combinations and the final detection results are shown in Table 4, where the data format is "number of FP/number of overall detections". From the table, we can see that noise reduction eliminates the majority of false positives of SPRT, due to the fact that most of wrongly-classified correlations only occur sporadically. The false positive of DBSpam approaches zero, when (1) M and K are relatively large and (2) the gap between M and K is small. Such dynamics of false positive reduction fits well with the analysis in Section 5.5. For our traces, any combination with 4/5 for M and 3/4 for K can achieve fairly high accuracy. Of course, the high detection accuracy is achieved at the cost of lowering detection sensitivity. It always exists a tradeoff between accuracy and sensitivity in network anomaly detection. However, even when the time window

Table 3: False Positives and False Negatives of SPRT

Attribute	S-1-A	S-1-B	S-1-C	S-2-A	S-2-B	S-2-C	N-1	N-2
Detection	970	5,179	2,420	437	4,536	1,777	66	2,368
True Positives	966	5,108	2,369	320	3,510	1,558	-	-
False Positives	4	71	51	117	1,026	219	66	2,368
True Negatives	290,889	1,156,085	596,979	1,634,307	8,895,993	4,266,100	687,390	15,941,150
FP/(FP+TN)	1.4e-5	6.1e-5	8.5e-5	7.2e-5	1.2e-4	5.1e-5	9.6e-5	1.5e-4
Spam Connections	958	570	324	329	1,351	969	-	-
Missed Connections	8	2	0	6	27	13	-	-
Missed Conn Ratio	0.008	0.004	0	0.018	0.020	0.013	-	-

Δ is set to 2 seconds and M is set to 5, the overall delay of DBSpam detection is just 10 seconds but with much higher accuracy.

Currently most false positives of DBSpam are induced by P2P applications. The capacity of spawning thousands of connections in a second and the behavior of periodic PING/PONG communications make P2P applications have a much higher probability of being correlated than any other applications. Due to its hog overwhelming proportion in bandwidth consumption, many ISPs and university networks in US have restricted the maximal connections that P2P applications can establish, which helps reduce the false positives of DBSpam.

Table 4: Overall False Positives of DBSpam ($\Delta = 2s$)

	(M, K)			
Trace	(3, 2)	(4, 3)	(5, 3)	(5, 4)
S-1-A	0/188	0/138	0/124	0/110
S-1-B	0/162	0/126	0/103	0/103
S-1-C	0/194	0/150	0/124	0/123
S-2-A	0/65	0/36	0/52	0/27
S-2-B	13/335	3/243	4/216	0/186
S-2-C	0/193	0/124	0/135	0/94
N-1	0/0	0/0	0/0	0/0
N-2	7/7	1/1	2/2	0/0

*Data Format: # of false positives / # of total detections

6.4 Resource Consumption

According to Table 1, the arrival rate of small TCP packets at the edge router can reach around 20,000 packets per second (pps), at which DBSpam must be able to handle. Current high-end PCs can meet this requirement without much difficulty. Using a Dell Precision 360 machine with Pentium-4 3GHz CPU and 512MB memory, we run the prototype of DBSpam on each trace multiple times. We use *time* and *ps* to measure the CPU and memory usage. The results are listed in Table 5. The average packet processing rate of DBSpam is computed by dividing the total packet number of the trace over the processing time ("CPU Time"). The processing rates clearly demonstrate the capability of DBSpam working at high-speed networks. Even in the worst case, DBSpam still can handle 241,965 pps, which is over 10 times more than the required processing speed.

Table 5: Resource Consumption

Trace	CPU Util	CPU Time	pps	Peak Mem
S-1-A	36.3%	9.0s	430,283	2.2MB
S-1-B	37.7%	9.8s	426,384	1.6MB
S-1-C	24.0%	9.3s	484,875	1.2MB
S-2-A	58.0%	36.8s	327,076	11.9MB
S-2-B	84.3%	109.2s	241,965	10.5MB
S-2-C	57.1%	78.6s	332,989	2.8MB
N-1	21.7%	51.1s	478,171	5.6MB
N-2	32.1%	789.9s	376,925	8.4MB

Memory consumption of DBSpam is mainly determined by two factors: the number of active SMTP connections and the number of outbound TCP connections during each SMTP reply rounds. So, the peak memory consumption is not necessarily determined by the network traffic volume. As DBSpam only needs to maintain very few states, and only a very small portion (false positive probability) of connections need to maintain states for relatively long time (lifespan of SMTP connections), the overall memory consumption should not be a problem. Also note that the memory management of our prototype is quite naive since our focus is mainly on the correctness, not on the performance.

6.5 Suppressing Spam Activities

Once the spam laundering activities are identified, DBSpam can effectively stifle them by activating the suppression module. Since spam suppression mechanisms such as blocking and throttling are straightforward to implement, the evaluation results of the suppression module are not included due to space limit.

7. POTENTIAL EVASIONS

In such an ongoing arms race between spammers and anti-spammers, we envision that sufficiently aggressive spammers will seek sophisticated techniques to evade DBSpam. This is especially true for a spammer who is able to fully control remote spam proxy machines and deploy arbitrarily customized software. It may use non-off-the-shelf proxy programs, which can manipulate the traffic between the spam source and the first-hop proxy, to break packet symmetry. One possible way is to split a single reply packet from SMTP server into n fragmented packets on the first-hop proxy and then to transfer them back to the spam source.

However, as long as enough observations are collected, DBSpam can still capture such potential evasions. Recall that the effect of this packet splitting on SPRT model is just the change of the value of θ_0, which measures the probability of 1 to n outbound TCP packets observed in a reply round. So, instead of $\theta_0 = \Pr(M = 1)$, now $\theta_0 = \Pr(M = 1) + \ldots + \Pr(M = n)$. According to Equation (10), without changing other parameters, the augmented value of θ_0 renders more average number of observations needed to detect a spam proxy. On the other hand, not all SMTP transactions have enough reply rounds for detection. Due to enlonged observations, short-living spamming activities may not be detected.

To demonstrate the capability of DBSpam in capturing such evasions, we relax the definition of packet symmetry, in which one or two data packets may appear in one reply round, and adjust θ_0 to 0.5.[10] Then, we estimate the overall false positives of DBSpam, which are listed in Table 6 under the parameter setting of $M = 5$, $K = 4$, and $\Delta = 2s$. For comparison, the results without relaxation are listed in the first row, while the results with relaxation are listed in

[10]Note that θ_0 never exceeds 0.5 in all our traces with various packet lengths from 150 to 300 bytes.

Table 6: False Positive Comparisons ($M = 5$, $K = 4$, $\Delta = 2s$)

θ_0	α^*	$E[N\|H_1]$	S-1-A	S-1-B	S-1-C	S-2-A	S-2-B	S-2-C	N-1	N-2
e^{-1}	0.005	5.5	0/110	0/103	0/123	0/27	0/186	0/94	0/0	0/0
0.5	0.005	8.1	0/0	0/103	0/120	0/0	0/97	0/32	0/0	8/8
0.5	0.02	6.0	0/110	2/105	0/121	0/27	7/194	1/94	0/0	21/21

the second row. Clearly, the short-living spamming activities are missed by DBSpam, with zero detection for S-*-A traces and much fewer detections for S-2-B and S-2-C traces. However, those spamming activities with more reply rounds can still be accurately detected. Since parameter α^* is tunable, we vary its value, from 0.005 to 0.02, to accommodate the shortest SMTP transactions for the examination of DBSpam. The third row in Table 6 lists the results after this adjustment, showing that DBSpam can capture almost all spamming activities as same as before but at the cost of slightly more false positives, which is the necessary tradeoff in capturing evasive spam proxy traffic.

Moreover, instead of employing off-the-shelf proxy software, any advanced evasion technique will inevitably induce the modifications on the current spam methods and degrade the spam laundering efficiency. The customized proxy software also increases the cost of spamming. Overall, DBSpam indeed significantly raises the protection bar against email spam, breaking the laundering and tracing out the real spam sources, in the anti-spam-vs-spam arms race.

8. CONCLUSION

In this paper, we present a simple and effective system, DBSpam, to detect and break proxy-based email spam laundering activities inside a customer network and to trace out the corresponding spam sources outside the network. Instead of content checking, DBSpam leverages the protocol semantics and timing causality of proxy-based spamming to identify spam proxies and real spam sources behind them. Based on connection correlation and packet symmetry principles, DBSpam monitors the bi-directional traffic passing through a network gateway, and utilizes a simple statistical method, Sequential Probability Ratio Test, to quickly filter out innocent connections and identify the spam laundry path with high probability. To further reduce false positives and false negatives, we propose a noise reduction technique to make spammer-tracking more accurate after gathering consecutive correlation detection results. We implement a prototype of DBSpam using *libpcap* on Linux, and conduct trace-based experiments to evaluate its effectiveness. Our experimental results reveal that DBSpam can be tuned to detect spam proxies and sources with low false positives and false negatives in seconds. After detecting spam proxies and related spam sources, DBSpam can effectively throttle or block spam traffic.

9. REFERENCES

[1] http://www.messagelabs.com/Threat_Watch/.

[2] http://www.spamhaus.org/news.lasso?article=156.

[3] http://spamassassin.apache.org/.

[4] http://www.postini.com.

[5] http://www.senderscorecertified.com.

[6] http://www.toplayer.com.

[7] Composite blocking list. http://cbl.abuseat.org.

[8] Domainkeys: Proving and protecting email sender identity. http://antispam.yahoo.com/domainkeys.

[9] http://www.rfc-editor.org/rfc/rfc2476.txt.

[10] MTA Authorization Records in DNS. http://www.ietf.org/html.charters/OLD/marid-charter.html.

[11] The penny black project. http://www.research.microsoft.com/research/sv/PennyBlack/.

[12] SPF. http://www.openspf.org.

[13] M. Andreolini, A. Bulgarelli, M. Colajanni, and F. Mazzoni. Honeyspam: Honeypots fighting spam at the source. In *Proc. USENIX SRUTI 2005*, Cambridge, MA, July 2005.

[14] A. Back. Hashcash. http://www.hashcash.org/.

[15] J. Blosser and D. Josephsen. Scalable centralized bayesian spam mitigation with bogofilter. In *Proc. USENIX LISA 2004*, Atlanta, GA, November 2004.

[16] A. Blum, D. X. Song, and S. Venkataraman. Detection of interactive stepping stones: Algorithms and confidence bounds. In *Proc. RAID 2004*, Sophia Antipolis, France, September 2004.

[17] L. Donnerhacke. Teergrubing faq. http://www.iks-jena.de/mitarb/lutz/usenet/teergrube.en.html.

[18] S. Garriss, M. Kaminsky, M. J. Freedman, B. Karp, D. Mazieres, and H. Yu. Re: Reliable email. In *Proc. USENIX NSDI 2006*, San Jose, CA, May 2006.

[19] P. Graham. A plan for spam. http://www.paulgraham.com/spam.html.

[20] T. Hunter, P. Terry, and A. Judge. Distributed tarpitting: Impeding spam across multiple servers. In *Porc. USENIX LISA 2003*, San Diego, CA, October 2003.

[21] J. Jung, V. Paxson, A. W. Berger, and H. Balakrishnan. Fast portscan detection using sequential hypothesis testing. In *Proc. IEEE Symposium on Security and Privacy 2004*, Oakland, CA, May 2004.

[22] J. Jung and E. Sit. An empirical study of spam traffic and the use of dns black lists. In *Proc. ACM SIGCOMM Internet Measurement Conference*, Taormina, Italy, October 2004.

[23] B. Krishnamurthy and E. Blackmond. SHRED: Spam harassment reduction via economic disincentives. http://www.research.att.com/~bala/papers/shred-ext.pdf.

[24] K. Li and Z. Zhong. Fast statistical spam filter by approximate classifications. In *Proc. ACM SIGMETRICS 2006*, St. Malo, France, June 2006.

[25] N. Provos. A virtual honeypot framework. In *Proc. USENIX Security 2004*, San Diego, CA, August 2004.

[26] S. Radosavac, J. S. Baras, and I. Koutsopoulos. A framework for mac protocol misbehavior detection in wireless networks. In *Proc. 4th ACM workshop on Wireless security*, Cologne, Germany, September 2005.

[27] A. Ramachandran, D. Dagon, and N. Feamster. Can DNS-based blacklists keep up with bots? In *CEAS 2006*, Mountain View, CA, July 2006.

[28] A. Ramachandran and N. Feamster. Understanding the network-level behavior of spammers. In *Proc. ACM SIGCOMM 2006*, Pisa, Italy, September 2006.

[29] M. Roesch. Snort - lightweight intrusion detection for networks. In *Proc. USENIX LISA 1999*, Seattle, WA, November 1999.

[30] R. D. Twining, M. M. Williamson, M. Mowbray, and M. Rahmouni. Email prioritization: Reducing delays on legitimate mail caused by junk mail. In *Proc. USENIX Annual Technical Conference 2004*, Boston, MA, June 2004.

[31] A. Wald. *Sequential Analysis*. Dover Publications, 2004.

[32] M. Walfish, J. Zamfirescu, H. Balakrishnan, D. Karger, and S. Shenker. Distributed quota enforcement for spam control. In *Proc. USENIX NSDI 2006*, San Jose, CA, May 2006.

[33] M. M. Williamson. Design, implementation and test of an email virus throttle. In *Proc. 19th Annual Computer Security Applications Conference*, Las Vegas, Nevada, December 2003.

[34] D. Woolridge, J. Law, and M. Kawasaki. The qmail spam throttle mechanism. http://spamthrottle.qmail.ca/man/qmail-spamthrottle.5.html.

[35] B. Yerazunis. Crm114 - the controllable regex mutilator. http://crm114.sourceforge.net.

[36] Y. Zhang and V. Paxson. Detecting stepping stones. In *Proc. USENIX Security 2000*, Denver, CO, August 2000.