Detecting Spam Zombies by Monitoring Outgoing Messages

Zhenhai Duan, Peng Chen, Fernando Sanchez
Florida State University
{*duan, pchen, sanchez*}*@cs.fsu.edu*

Yingfei Dong
University of Hawaii
yingfei@hawaii.edu

Mary Stephenson, James Barker
Florida State University
{*mstephenson, jmbarker*}*@fsu.edu*

***Abstract*—Compromised machines are one of the key security threats on the Internet; they are often used to launch various security attacks such as DDoS, spamming, and identity theft. In this paper we address this issue by investigating effective solutions to automatically identify compromised machines in a network. Given that spamming provides a key economic incentive for attackers to recruit the large number of compromised machines, we focus on the subset of compromised machines that are involved in the spamming activities, commonly known as spam zombies. We develop an effective spam zombie detection system named SPOT by monitoring outgoing messages of a network. SPOT is designed based on a powerful statistical tool called Sequential Probability Ratio Test, which has bounded false positive and false negative error rates. Our evaluation studies based on a two-month email trace collected in a large U.S. campus network show that SPOT is an effective and efficient system in automatically detecting compromised machines in a network. For example, among the 440 internal IP addresses observed in the email trace, SPOT identifies 132 of them as being associated with compromised machines. Out of the 132 IP addresses identified by SPOT, 126 can be either independently confirmed (110) or highly likely (16) to be compromised. Moreover, only 7 internal IP addresses associated with compromised machines in the trace are missed by SPOT.**

I. Introduction

A major security challenge on the Internet is the existence of the large number of compromised machines. Such machines have been increasingly used to launch various security attacks including DDoS, spamming, and identity theft [1]. Two natures of the compromised machines on the Internet—sheer volume and wide spread—render many existing security countermeasures less effective and defending attacks involving compromised machines extremely hard. On the other hand, identifying and cleaning compromised machines in a network remain a significant challenge for system administrators of networks of all sizes.

In this paper we focus on the subset of compromised machines that are used for sending spam messages, which are commonly referred to as spam zombies. Given that spamming provides a critical economic incentive for the controllers of the compromised machines to recruit these machines, it has been widely observed that many compromised machines are involved in spamming [2]. A number of recent research efforts have studied the aggregate global characteristics of spamming botnets (networks of compromised machines involved in spamming) such as the size of botnets and the spamming patterns of botnets, based on the sampled spam messages received at a large email service provider [2], [3].

Rather than the aggregate global characteristics of spamming botnets, we aim to develop a tool for system administrators to automatically detect the compromised machines in their networks in an online manner. We consider ourselves situated in a network and ask the following question: How can we automatically identify the compromised machines in the network as outgoing messages pass the monitoring point sequentially? The approaches developed in the previous work [2], [3] cannot be applied here. The locally generated outgoing messages in a network normally cannot provide the aggregate large-scale spam view required by these approaches. Moreover, these approaches cannot support the online detection requirement in the environment we consider.

The nature of sequentially observing outgoing messages gives rise to the sequential detection problem. In this paper we will develop a spam zombie detection system, named SPOT, by monitoring outgoing messages. SPOT is designed based on a statistical method called Sequential Probability Ratio Test (SPRT), developed by Wald in his seminal work [4]. SPRT is a powerful statistical method that can be used to test between two hypotheses (in our case, a machine is compromised vs. the machine is not compromised), as the events (in our case, outgoing messages) occur sequentially. As a simple and powerful statistical method, SPRT has a number of desirable features. It minimizes the expected number of observations required to reach a decision among all the sequential and non-sequential statistical tests with no greater error rates. This means that the SPOT detection system can identify a compromised machine quickly. Moreover, both the false positive and false negative probabilities of SPRT can be bounded by user-defined thresholds. Consequently, users of the SPOT system can select the desired thresholds to control the false positive and false negative rates of the system.

In this paper we develop the SPOT detection system to assist system administrators in automatically identifying the compromised machines in their networks. We also evaluate the performance of the SPOT system based on a two-month email trace collected in a large U.S. campus network. Our evaluation studies show that SPOT is an effective and efficient system in automatically detecting compromised machines in a network. For example, among the 440 internal IP addresses observed in the email trace, SPOT identifies 132 of them as being

associated with compromised machines. Out of the 132 IP addresses identified by SPOT, 126 can be either independently confirmed (110) or are highly likely (16) to be compromised. Moreover, only 7 internal IP addresses associated with compromised machines in the trace are missed by SPOT. In addition, SPOT only needs a small number of observations to detect a compromised machine. The majority of spam zombies are detected with as little as 3 spam messages.

The remainder of the paper is organized as follows. In Section II we discuss related work in the area of botnet detection. We formulate the spam zombie detection problem in Section III. Section IV provides the necessary background on SPRT for developing the SPOT spam zombie detection system. In Section V we provide the detailed design of SPOT. Section VI evaluates the SPOT detection system based on the two-month email trace. We briefly discuss the practical deployment issues, potential evasion techniques, and limitations of the current work in Section VII, and conclude the paper in Section VIII.

II. Related Work

In this section we discuss related work, focusing on the studies that utilize spamming activities to detect bots.

Based on email messages received at a large email service provider, two recent studies [2], [3] investigated the aggregate global characteristics of spamming botnets including the size of botnets and the spamming patterns of botnets. These studies provided important insights into the aggregate global characteristics of spamming botnets by clustering spam messages received at the provider into spam campaigns using embedded URLs and near-duplicate content clustering, respectively. However, their approaches are better suited for large email service providers to understand the aggregate global characteristics of spamming botnets instead of being deployed by individual networks to detect internal compromised machines. Moreover, their approaches cannot support the online detection requirement in the network environment considered in this paper. We aim to develop a tool to assist system administrators in automatically detecting compromised machines in their networks in an online manner.

Xie, *et al.* developed an effective tool DBSpam to detect proxy-based spamming activities in a network relying on the packet symmetry property of such activities [5]. We intend to identify all types of compromised machines involved in spamming, not only the spam proxies that translate and forward upstream non-SMTP packets (for example, HTTP) into SMTP commands to downstream mail servers as in [5].

BotHunter [6], developed by Gu *et al.*, detects compromised machines by correlating the IDS dialog trace in a network. It was developed based on the observation that a complete malware infection process has a number of well-defined stages including inbound scanning, exploit usage, egg downloading, outbound bot coordination dialog, and outbound attack propagation. By correlating inbound intrusion alarms with outbound communications patterns, BotHunter can detect the potential infected machines in a network. Unlike BotHunter which relies on the specifics of the malware infection process, SPOT focuses on the economic incentive behind many compromised machines and their involvement in spamming. Compared to BotHunter, SPOT is a light-weight spam zombie detection system; it does not need the support from the network intrusion detection system as required by BotHunter.

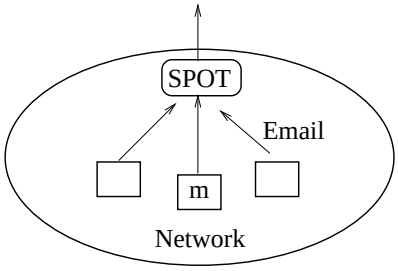


Fig. 1. Network model.

As a simple and powerful statistical method, Sequential Probability Ratio Test (SPRT) has been successfully applied in many areas. In the area of networking security, SPRT has been used to detect portscan activities [7], proxy-based spamming activities [5], and MAC protocol misbehavior in wireless networks [8].

III. Problem Formulation

In this section we formulate the spam zombie detection problem in a network. In particular, we discuss the network model and assumptions we make in the detection problem.

Figure 1 illustrates the logical view of the network model. We assume that messages originated from machines inside the network will pass the deployed spam zombie detection system. This assumption can be achieved in a few different scenarios. First, in order to alleviate the ever-increasing spam volume on the Internet, many ISPs and networks have adopted the policy that all the outgoing messages originated from the network must be relayed by a few designated mail servers in the network. Outgoing email traffic (with destination port number of 25) from all other machines in the network is blocked by edge routers of the network [9]. In this situation, the detection system can be co-located with the designated mail servers in order to examine the outgoing messages. Second, in a network where the aforementioned blocking policy is not adopted, the outgoing email traffic can be replicated and redirected to the spam zombie detection system. We note that the detection system does not need to be on the regular email traffic forwarding path; the system only needs a replicated stream of the outgoing email traffic. Moreover, as we will show in Section VI, the proposed SPOT system works well even if it cannot observe all outgoing messages. SPOT only requires a reasonably sufficient view of the outgoing messages originated from the network in which it is deployed.

A machine in the network is assumed to be either compromised or normal (that is, not compromised). In this paper we only focus on the compromised machines that are involved in spamming. Therefore, we use the term a *compromised*

machine to denote a *spam zombie*, and use the two terms interchangeably. Let X_i for $i = 1, 2, \ldots$ denote the successive observations of a random variable X corresponding to the sequence of messages originated from machine m inside the network. We let $X_i = 1$ if message i from the machine is a spam, and $X_i = 0$ otherwise. The detection system assumes that the behavior of a compromised machine is different from that of a normal machine in terms of the messages they send. Specifically, a compromised machine will with a higher probability generate a spam message than a normal machine. Formally,

$$Pr(X_i = 1|H_1) > Pr(X_i = 1|H_0), \quad (1)$$

where H_1 denote that machine m is compromised and H_0 that the machine is normal.

We assume that a sending machine m as observed by the spam zombie detection system is an end-user client machine. It is not a mail relay server. This assumption is just for the convenience of our exposition. The proposed SPOT system can handle the case where an outgoing message is forwarded by a few internal mail relay servers before leaving the network. We discuss practical deployment issues in Section VII. We further assume that a (content-based) spam filter is deployed at the detection system so that an outgoing message can be classified as either a spam or nonspam. The spam filter does not need to be perfect in terms of the false positive rate and the false negative rate. From our communications with network operators, an increasing number of networks have started filtering outgoing messages in recent years. Based on the above assumptions, the spam zombie detection problem can be formally stated as follows. As X_i arrives sequentially at the detection system, the system determines with a high probability if machine m has been compromised. Once a decision is reached, the detection system reports the result, and further actions can be taken, e.g., to clean the machine.

IV. Background on Sequential Probability Ratio Test

In this section we provide the necessary background on the Sequential Probability Ratio Test (SPRT) for understanding the proposed spam zombie detection system. Interested readers are directed to [4] for a detailed discussion on the topic of SPRT.

In its simplest form, SPRT is a statistical method for testing a simple null hypothesis against a single alternative hypothesis. Intuitively, SPRT can be considered as an one-dimensional random walk with two user-specified boundaries corresponding to the two hypotheses. As the samples of the concerned random variable arrive sequentially, the walk moves either upward or downward one step, depending on the value of the observed sample. When the walk hits or crosses either of the boundaries for the first time, the walk terminates and the corresponding hypothesis is selected. In essence, SPRT is a variant of the traditional probability ratio tests for testing under what distribution (or with what distribution parameters), it is more likely to have the observed samples. However, unlike traditional probability ratio tests that require a pre-defined number of observations, SPRT works in an online manner and updates as samples arrive sequentially. Once sufficient evidence for drawing a conclusion is obtained, SPRT terminates.

As a simple and powerful statistical tool, SPRT has a number of compelling and desirable features that lead to the wide-spread applications of the technique in many areas. First, both the actual false positive and false negative probabilities of SPRT can be bounded by the user-specified error rates. This means that users of SPRT can pre-specify the desired error rates. A smaller error rate tends to require a larger number of observations before SPRT terminates. Thus users can balance the performance and cost of an SPRT test. Second, it has been proved that SPRT minimizes the average number of the required observations for reaching a decision for a given error rate, among all sequential and non-sequential statistical tests. This means that SPRT can quickly reach a conclusion to reduce the cost of the corresponding experiment, without incurring a higher error rate. In the following we present the formal definition and a number of important properties of SPRT. The detailed derivations of the properties can be founded in [4].

Let X denote a Bernoulli random variable under consideration with an unknown parameter θ, and $X_1, X_2, \ldots$ the successive observations on X. As discussed above, SPRT is used for testing a simple hypothesis H_0 that $\theta = \theta_0$ against a single alternative H_1 that $\theta = \theta_1$. That is,

$$\begin{aligned} Pr(X_i = 1|H_0) &= 1 - Pr(X_i = 0|H_0) = \theta_0 \\ Pr(X_i = 1|H_1) &= 1 - Pr(X_i = 0|H_1) = \theta_1. \end{aligned}$$

To ease exposition and practical computation, we compute the logarithm of the probability ratio instead of the probability ratio in the description of SPRT. For any positive integer $n = 1, 2, \ldots$, define

$$\Lambda_n = ln \frac{Pr(X_1, X_2, \ldots, X_n|H_1)}{Pr(X_1, X_2, \ldots, X_n|H_0)}. \quad (2)$$

Assume that X_i's are independent (and identically distributed), we have

$$\Lambda_n = ln \frac{\prod_1^n Pr(X_i|H_1)}{\prod_1^n Pr(X_i|H_0)} = \sum_{i=1}^{n} ln \frac{Pr(X_i|H_1)}{Pr(X_i|H_0)} = \sum_{i=1}^{n} Z_i \quad (3)$$

where $Z_i = ln \frac{Pr(X_i|H_1)}{Pr(X_i|H_0)}$, which can be considered as the step in the random walk represented by Λ. When the observation is one ($X_i = 1$), the constant $ln \frac{\theta_1}{\theta_0}$ is added to the preceding value of Λ. When the observation is zero ($X_i = 0$), the constant $ln \frac{1-\theta_1}{1-\theta_0}$ is added.

The Sequential Probability Ratio Test (SPRT) for testing H_0 against H_1 is then defined as follows. Given two user-specified constants A and B where $A < B$, at each stage n of the Bernoulli experiment, the value of Λ_n is computed as

in Eq. (3), then

$$\begin{aligned} \Lambda_n \le A &\implies \text{accept } H_0 \text{ and terminate test,} \\ \Lambda_n \ge B &\implies \text{accept } H_1 \text{ and terminate test,} \\ A < \Lambda_n < B &\implies \text{take an additional observation.} \end{aligned} \quad (4)$$

In the following we describe a number of important properties of SPRT. If we consider H_1 as a detection and H_0 as a normality, an SPRT process may result in two types of errors: false positive where H_0 is true but SPRT accepts H_1 and false negative where H_1 is true but SPRT accepts H_0. We let α and β denote the user-desired false positive and false negative probabilities, respectively. There exist some fundamental relations among α, β, A, and B [4],

$$A \ge ln\frac{\beta}{1-\alpha},\ B \le ln\frac{1-\beta}{\alpha},$$

for most practical purposes, we can take the equality, that is,

$$A = ln\frac{\beta}{1-\alpha},\ B = ln\frac{1-\beta}{\alpha}. \quad (5)$$

This will only slightly affect the actual error rates. Formally, let α' and β' represent the actual false positive rate and the actual false negative rate, respectively, and let A and B be computed using Eq. (5), then the following relations hold,

$$\alpha' \le \frac{\alpha}{1-\beta},\ \beta' \le \frac{\beta}{1-\alpha}, \quad (6)$$

and

$$\alpha' + \beta' \le \alpha + \beta. \quad (7)$$

Eqs. (6) and (7) provide important bounds for α' and β'. In all practical applications, the desired false positive and false negative rates will be small, for example, in the range from 0.01 to 0.05. In these cases, $\frac{\alpha}{1-\beta}$ and $\frac{\beta}{1-\alpha}$ very closely equal the desired α and β, respectively. In addition, Eq. (7) specifies that the actual false positive rate and the false negative rate cannot be both larger than the corresponding desired error rate in a given experiment. Therefore, in all practical applications, we can compute the boundaries A and B using Eq. (5). This will provide at least the same protection against errors as if we use the precise values of A and B for a given pair of desired error rates. The precise values of A and B are hard to obtain.

Another important property of SPRT is the number of observations, N, required before SPRT reaches a decision. The following two equations approximate the average number of observations required when H_1 and H_0 are true, respectively.

$$E[N|H_1] = \frac{\beta \ln\frac{\beta}{1-\alpha} + (1-\beta)\ln\frac{1-\beta}{\alpha}}{\theta_1 \ln\frac{\theta_1}{\theta_0} + (1-\theta_1)\ln\frac{1-\theta_1}{1-\theta_0}} \quad (8)$$

$$E[N|H_0] = \frac{(1-\alpha)\ln\frac{\beta}{1-\alpha} + \alpha\ln\frac{1-\beta}{\alpha}}{\theta_1 \ln\frac{\theta_1}{\theta_0} + (1-\theta_1)\ln\frac{1-\theta_1}{1-\theta_0}} \quad (9)$$

From the above equations we can see that the average number of required observations when H_1 or H_0 is true depends on four parameters: the desired false positive and negative rates (α and β), and the distribution parameters θ_1 and θ_0 for hypotheses H_1 and H_0, respectively. We note that SPRT does not require the precise knowledge of the distribution parameters θ_1 and θ_0. As long as the true distribution of the underlying random variable is sufficiently close to one of hypotheses compared to another (that is, θ is closer to either θ_1 or θ_0), SPRT will terminate with the bounded error rates. An imprecise knowledge of θ_1 and θ_0 will only affect the number of required observations for SPRT to reach a decision.

Algorithm 1 SPOT spam zombie detection system

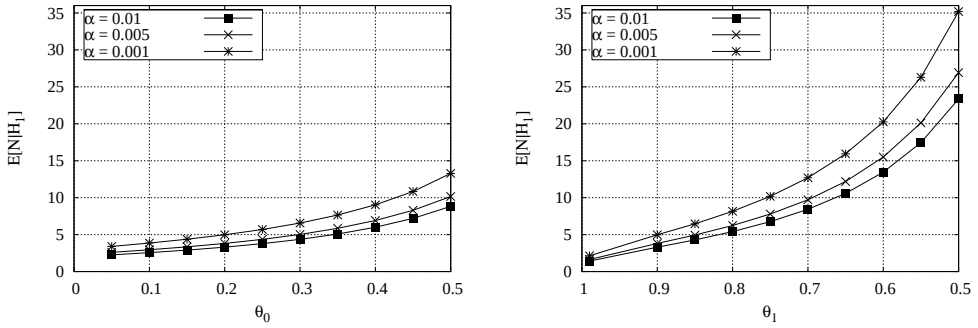
```
An outgoing message arrives at SPOT
Get IP address of sending machine m
// all following parameters specific to machine m
Let n be the message index
Let X_n = 1 if message is spam, X_n = 0 otherwise
if (X_n == 1) then
    // spam, Eq. 3
    Λ_n += ln(θ_1/θ_0)
else
    // nonspam
    Λ_n += ln((1-θ_1)/(1-θ_0))
end if
if (Λ_n ≥ B) then
    Machine m is compromised. Test terminates for m.
else if (Λ_n ≤ A) then
    Machine m is normal. Test is reset for m.
    Λ_n = 0
    Test continues with new observations
else
    Test continues with an additional observation
end if
```

V. Detecting Spam Zombies

In this section we develop the spam zombie detection system SPOT, which utilizes the Sequential Probability Ratio Test (SPRT) presented in the last section. We discuss the impacts of SPRT parameters on SPOT in the context of spam zombie detection. To ease exposition of the algorithm, we ignore the potential impact of dynamic IP addresses [10] and assume that an IP address corresponds to a unique machine. We will informally discuss the impact of dynamic IP addresses at the end of this section. We will formally evaluate the performance of SPOT and the potential impact of dynamic IP addresses in the next section, based on a two-month email trace collected on a large U.S. campus network.

A. SPOT Detection Algorithm

In the context of detecting spam zombies in SPOT, we consider H_1 as a detection and H_0 as a normality. That is, H_1 is true if the concerned machine is compromised, and H_0 is true if it is not compromised. In addition, we let $X_i = 1$ if the ith message from the concerned machine in the network is a spam, and $X_i = 0$ otherwise. Recall that SPRT requires four configurable parameters from users, namely, the desired false positive probability α, the desired false negative probability β,



(a) $\theta_1 = 0.9$ (b) $\theta_0 = 0.2$

Fig. 2. Average number of required observations when H_1 is true ($\beta = 0.01$)

the probability that a message is a spam when H_1 is true (θ_1), and the probability that a message is a spam when H_0 is true (θ_0). We discuss how users configure the values of the four parameters in the next subsection. Based on the user-specified values of α and β, the values of the two boundaries A and B of SPRT are computed using Eq. (5).

In the following we describe the SPOT detection algorithm. Algorithm 1 outlines the steps of the algorithm. When an outgoing message arrives at the SPOT detection system, the sending machine's IP address is recorded, and the message is classified as either spam or nonspam by the (content-based) spam filter. For each observed IP address, SPOT maintains the logarithm value of the corresponding probability ratio Λ_n, whose value is updated according to Eq. (3) as message n arrives from the IP address (lines 6 to 12 in Algorithm 1). Based on the relation between Λ_n and A and B, the algorithm determines if the corresponding machine is compromised, normal, or a decision cannot be reached.

We note that in the context of spam zombie detection, from the viewpoint of network monitoring, it is more important to identify the machines that have been compromised than the machines that are normal. After a machine is identified as being compromised (lines 13 and 14), it is added into the list of potentially compromised machines that system administrators can go after to clean. The message-sending behavior of the machine is also recorded should further analysis be required. Before the machine is cleaned and removed from the list, the SPOT detection system does not need to further monitor the message sending behavior of the machine.

On the other hand, a machine that is currently normal may get compromised at a later time. Therefore, we need to continuously monitor machines that are determined to be normal by SPOT. Once such a machine is identified by SPOT, the records of the machine in SPOT are re-set, in particular, the value of Λ_n is set to zero, so that a new monitoring phase starts for the machine (lines 15 to 18).

B. Parameters of SPOT Algorithm

SPOT requires four user-defined parameters: α, β, θ_1, and θ_0. In this subsection we discuss how a user of SPOT configures these parameters, and how these parameters may affect the performance of SPOT. As discussed in the previous section α and β are normally small values in the range from 0.01 to 0.05, which users can easily specify independent of the behaviors of the compromised and normal machines in the network.

Ideally, θ_1, and θ_0 should indicate the true probability of a message being spam from a compromised machine and a normal machine, respectively. However, as we have discussed in the last section, θ_1 and θ_0 do not need to accurately model the behaviors of the two types of machines. Instead, as long as the true distribution is closer to one of them than another, SPRT can reach a conclusion with the desired error rates. Inaccurate values assigned to these parameters will only affect the number of observations required by the algorithm to terminate. Moreover, SPOT relies on a (content-based) spam filter to classify an outgoing message into either spam or nonspam. In practice, θ_1 and θ_0 should model the detection rate and the false positive rate of the employed spam filter, respectively. We note that all the widely-used spam filters have a high detection rate and low false positive rate.

To get some intuitive understanding of the average number of required observations for SPRT to reach a decision, Figures 2 (a) and (b) show the value of $E[N|H_1]$ as a function of θ_0 and θ_1, respectively, for different desired false positive rates. In the figures we set the false negative rate $\beta = 0.01$. In Figure 2 (a) we assume the probability of a message being spam when H_1 is true to be 0.9 ($\theta_1 = 0.9$). From the figure we can see that it only takes a small number of observations for SPRT to reach a decision. For example, when $\theta_0 = 0.2$ (the spam filter has 20% false positive rate), SPRT requires about 3 observations to detect that the machine is compromised if the desired false positive rate is 0.01. As the behavior of a normal machine gets closer to that of compromised machine (or rather, the false positive rate of the spam filter increases),

i.e., θ_0 increases, a slightly higher number of observations are required for SPRT to reach a detection.

In Figure 2 (b) we assume the probability of a message being spam from a normal machine to be 0.2 ($\theta_0 = 0.2$). This can be caused, for example, by a spam filter with a false positive rate of 20%. From the figure we can see that it also only takes a small number of observations for SPRT to reach a decision. As the behavior of a compromised machine gets closer to that of a normal machine (or rather, the detection rate of the spam filter decreases), i.e., θ_1 decreases, a higher number of observations are required for SPRT to reach a detection.

From the figures we can also see that, as the desired false positive rate decreases, SPRT needs a higher number of observations to reach a conclusion. The same observation applies to the desired false negative rate. These observations illustrate the trade-offs between the desired performance of SPRT and the cost of the algorithm. In the above discussion, we only show the average number of required observations when H_1 is true because we are more interested in the speed of SPOT in detecting compromised machines. The study on $E[N|H_0]$ shows a similar trend (not shown).

C. Impact of Dynamic IP addresses

In the above discussion of the SPOT algorithm we have for simplicity ignored the potential impact of dynamic IP addresses and assumed that an observed IP corresponds to a unique machine. This needs not to be the case for the algorithm to work correctly. SPOT can work extremely well in the environment of dynamic IP addresses. To understand the reason we note that SPOT can reach a decision with a small number of observations as illustrated in Figure 2, which shows the average number of observations required for SPRT to terminate. In practice, we have noted that 3 or 4 observations are sufficient for SPRT to reach a decision for the vast majority of cases. If a machine is compromised, it is likely that more than 3 or 4 spam messages will be sent before the (unwitty) user shutdowns the machine. Therefore, dynamic IP addresses will not have any significant impact on SPOT. We formally evaluate the impact of dynamic IP addresses on SPOT in the next section.

VI. Performance Evaluation

In this section we evaluate the performance of the SPOT detection system based on a 2-month email trace collected on a large U.S. campus network. We also study the potential impact of dynamic IP addresses on SPOT.

A. Overview of the Email Trace and Methodology

The email trace was collected at a mail relay server deployed in the Florida State University (FSU) campus network between 8/25/2005 and 10/24/2005, excluding 9/11/2005 (we do not have trace on this date). During the course of the email trace collection, the mail server relayed messages destined for 53 subdomains in the FSU campus network. The mail relay server ran SpamAssassin [11] to detect spam messages.

TABLE I
SUMMARY OF THE EMAIL TRACE.

Measure	Non-spam	Spam	Aggregate
Period	8/25/2005 – 10/24/2005 (excld. 9/11/2005)		
# of emails	6,712,392	18,537,364	25,249,756
# of FSU emails	5,612,245	6,959,737	12,571,982
# of infected emails	60,004	163,222	223,226
# of infected FSU emails	34,345	43,687	78,032

TABLE II
SUMMARY OF SENDING IP ADDRESSES.

	Non-spam only	Spam only	Mixed
# of IP (%)	121,103 (4.9)	2,224,754 (90.4)	115,257 (4.7)
# of FSU IP (%)	175 (39.7)	74 (16.8)	191 (43.5)

The email trace contains the following information for each incoming message: the local arrival time, the IP address of the sending machine, and whether or not the message is spam. In addition, if a message has a known virus/worm attachment, it was so indicated in the trace by an anti-virus software. The anti-virus software and SpamAssassin were two independent components deployed on the mail relay server. Due to privacy issues, we do not have access to the content of the messages in the trace.

Ideally we should have collected all the outgoing messages in order to evaluate the performance of SPOT. However, due to logistical constraints, we were not able to collect all such messages. Instead, we identified the messages in the email trace that have been forwarded or originated by the FSU internal machines, that is, the messages forwarded or originated by an FSU internal machine and destined to an FSU account. We refer to this set of messages as the *FSU emails* and perform our evaluation of SPOT based on the FSU emails. We note the set of FSU emails does not contain all the outgoing messages originated from inside FSU, and the compromised machines identified by SPOT based on the FSU emails will likely be a lower bound on the true number of compromised machines inside FSU campus network.

An email message in the trace is classified as either *spam* or *non-spam* by SpamAssassin [11] deployed in the FSU mail relay server. For ease of exposition, we refer to the set of all messages as the *aggregate* emails including both spam and non-spam. If a message has a known virus/worm attachment, we refer to such a message as an *infected message*. We refer to an IP address of a sending machine as a *spam-only* IP address if only spam messages are received from the IP. Similarly, we refer to an IP address as *non-spam only* and *mixed* if we only receive non-spam messages, or we receive both spam and non-spam messages, respectively. Table I shows a summary of the email trace. As shown in the table, the trace contains more than 25 M emails, of which more than 18 M, or about 73%, are spam. During the course of the trace collection, we observed more than 2 M IP addresses ($2,461,114$) of sending machines, of which more than 95% sent at least one spam message. During the same course, we observed 440 FSU internal IP addresses. Table II shows the classifications of the observed

IP addresses. More detailed analysis of the email trace can be found in [12], including the daily message arrival patterns, and the behaviors of spammers at both the mail-server level and the network level.

In order to study the potential impacts of dynamic IP addresses on the SPOT system, we obtain the subset of FSU IP addresses in the trace whose domain names contain "wireless", which normally have dynamically allocated IP addresses. For each of the IP addresses, we group the messages sent from the IP address into clusters, where the messages in each cluster are likely to be from the same machine (before the IP address is re-assigned to a different machine). We group messages according to the inter-arrival times between consecutive messages, as discussed below. Let m_i for $i = 1, 2, \ldots$ denote the messages sent from an IP address, and t_i denote the time when message i is received. Then messages m_i for $i = 1, 2, \ldots, k$ belong to the same cluster if $|t_i - t_{i-1}| \leq T$ for $i = 2, 3, \ldots, k$, and $|t_{k+1} - t_k| > T$, where T is an user-defined time interval. We repeat the same process to group other messages. Let m_i for $i = j, j+1, \ldots, k$ be the sequence of messages in a cluster, arriving in that order. Then $|t_k - t_j|$ is referred to as the *duration* of the cluster. Figure 3 illustrates the message clustering process. The intuition is that, if two messages come closely in time from an IP address (within a time interval T), it is unlikely that the IP address has been assigned to two different machines within the short time interval.

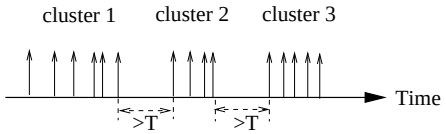


Fig. 3. Illustration of message clustering.

In the evaluation studies, we whitelist the known mail servers deployed on the FSU campus network, given that they are unlikely to be compromised. If a deployed mail server forwards a large number of spam messages, it is more likely that machines behind the mail server are compromised. However, just based on the information available in the email trace we cannot decide which machines are responsible for the large number of spam messages, and consequently, determine the compromised machines. Section VII discusses how SPOT can handle this case in practical deployment.

B. Performance Evaluation of SPOT

In this section, we evaluate the performance of SPOT based on the collected FSU emails. In all the studies, we set $\alpha = 0.01$, $\beta = 0.01$, $\theta_1 = 0.9$, and $\theta_0 = 0.2$. Many widely-deployed spam filters have much better performance than what we assume here.

TABLE III
PERFORMANCE OF SPOT.

Total # FSU IP	Detected	Confirmed (%)	Missed (%)
440	132	126 (94.7)	7 (5.3)

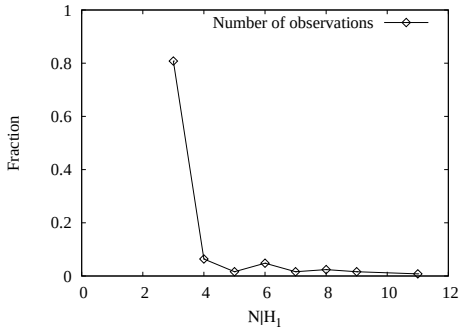


Fig. 4. Number of actual observations

Table III shows the performance of the SPOT spam zombie detection system. As discussed above, there are 440 FSU internal IP addresses observed in the email trace. SPOT identifies 132 of them to be associated with compromised machines. In order to understand the performance of SPOT in terms of the false positive and false negative rates, we should ideally examine each FSU internal physical machines included in the trace; however, we do not have the logistics to do so. Instead, we rely on a number of heuristics to verify if a machine is indeed compromised. We discuss the limitations of these heuristics in Section VII. First, we check if any message sent from an IP address carries a known virus/worm attachment. If this is the case, we say we have a confirmation. Out of the 132 IP addresses identified by SPOT, we can confirm 110 of them to be compromised in this way. For the remaining 22 IP addresses, we manually examine the spam sending patterns from the IP addresses and the domain names of the corresponding machines. If the fraction of the spam messages from an IP address is high (greater than 98%), we also claim that the corresponding machine has been confirmed to be compromised. We can confirm 16 of them to be compromised in this way. We note that the majority (62.5%) of the IP addresses confirmed by the spam percentage are dynamic IP addresses, which further indicates the likelihood of the machines to be compromised.

For the remaining 6 IP addresses that we cannot confirm by either of the above means, we have also manually examined their sending patterns. We note that, they have a relatively overall low percentage of spam messages over the two month of the collection period. However, they sent substantially more spam messages towards the end of the collection period. This indicates that they may get compromised towards the end of our collection period. However, we cannot independently confirm if this is the case.

Evaluating the false negative rate of SPOT is a bit tricky by noting that SPOT focuses on the machines that are potentially compromised, but not the machines that are normal (see Section V). In order to have some intuitive understanding of the false negative rate of the SPOT system, we consider the machines that SPOT does not identify as being compromised at the end of the email collection period, but for which SPOT has re-set the records (lines 15 to 18 in Algorithm 1). That is,

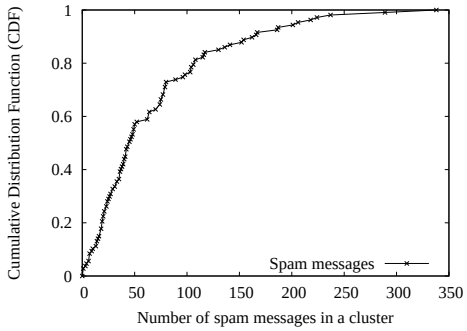


Fig. 5. Distribution of spam messages in each cluster.

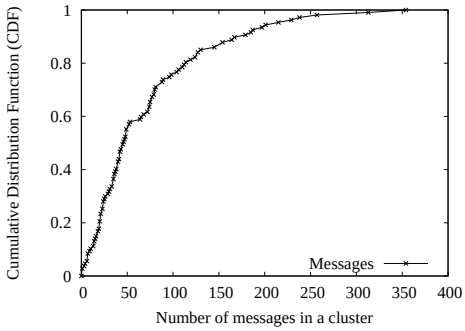


Fig. 6. Distribution of total messages in each cluster.

such machines have been claimed as being normal by SPOT (but have continuously been monitored). We also obtain the list of IP addresses that have sent at least a message with a virus/worm attachment. 7 of such IP addresses have been claimed as being normal, i.e., missed, by SPOT.

We emphasize that the infected messages are only used to confirm if a machine is compromised in order to study the performance of SPOT. Infected messages are not used by SPOT itself. SPOT relies on the spam messages instead of infected messages to detect if a machine has been compromised to produce the results in Table III. We make this decision by noting that, it is against the interest of a professional spammer to send spam messages with a virus/worm attachment. Such messages are more likely to be detected by anti-virus softwares, and hence deleted before reaching the intended recipients. This is confirmed by the low percentage of infected messages in the overall email trace shown in Table I. Infected messages are more likely to be observed during the spam zombie recruitment phase instead of spamming phase. Infected messages can be easily incorporated into the SPOT system to improve its performance.

We note that both the actual false positive rate and the false negative rate are higher than the specified false positive rate and false negative rate, respectively. One possible reason is that the evaluation was based on the FSU emails, which can only provide a partial view of the outgoing messages originated from inside FSU.

Figure 4 shows the distributions of the number of actual observations that SPOT takes to detect the compromised machines. As we can see from the figure, the vast majority of compromised machines can be detected with a small number of observations. For example, more than 80% of the compromised machines are detected by SPOT with only 3 observations. All the compromised machines are detected with no more than 11 observations. This indicates that, SPOT can quickly detect the compromised machines. We note that SPOT does not need compromised machines to send spam messages at a high rate in order to detect them. Here, "quick" detection does not mean a short duration, but rather a small number of observations. A compromised machine can send spam messages at a low rate (which, though, works against the interest of spammers), but it can still be detected once enough observations are obtained by SPOT.

C. Dynamic IP Addresses

In order to understand the potential impacts of dynamic IP addresses on the performance of SPOT, we group messages from a dynamic IP address (with domain names containing "wireless") into clusters with a time interval threshold of 30 minutes. Messages with a consecutive inter-arrival time no greater than 30 minutes are grouped into the same cluster. Given the short inter-arrival duration of messages within a cluster, we consider all the messages from the same IP address within each cluster as being sent from the same machine. That is, the corresponding IP address has not been re-assigned to a different machine within the concerned cluster. (It is possible that messages from multiple adjacent clusters are actually sent from the same machine.)

Figure 5 shows the cumulative distribution function (CDF) of the number of spam messages in each cluster. In particular, we note that more than 90% of the clusters have no less than 10 spam messages, and more than 96% no less than 3 spam messages. Given the large number of spam messages sent within each cluster, it is unlikely for SPOT to mistake one compromised machine as another when it tries to detect spam zombies. Indeed, we have manually checked that, spam messages tend to be sent back to back in a batch fashion when a dynamic IP address is observed in the trace. Figure 6 shows the CDF of the number of all messages (including both spam and non-spam) in each cluster. Similar observations can be made to that in Figure 5.

Figure 7 shows the CDF of the durations of the clusters. As we can see from the figure, more than 75% and 58% of the clusters last no less than 30 minutes and one hour (corresponding to the two vertical lines in the figure), respectively. The longest duration of a cluster we observe in the trace is about 3.5 hours.

Given the above observations, in particular, the large number of spam messages in each cluster, we conclude that dynamic IP addresses will not have any important impact on the performance of SPOT. SPOT can reach a decision within the vast majority (96%) of the clusters in the setting we used in the current performance study. It is unlikely for SPOT to mistake a compromised machine as another.

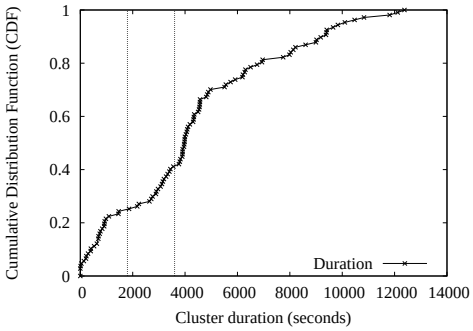


Fig. 7. Distribution of the cluster duration.

VII. Discussion

A. Practical Deployment

To ease exposition we have assumed that a sending machine m (Figure 1) is an end-user client machine. In practice, a network may have multiple subdomains and each has its own mail servers. A message may be forwarded by a number of mail relay servers before leaving the network. SPOT can work well in this kind of network environments. In the following we outline two possible approaches. First, SPOT can be deployed at the mail servers in each subdomain to monitor the outgoing messages so as to detect the compromised machines in that subdomain. Second, and possibly more practically, SPOT is only deployed at the designated outgoing mail servers for the domain as discussed in Section III. SPOT relies on the `Received` header fields to identify the originating machine of a message in the network [13]. Given that the `Received` header fields can be spoofed by spammers, SPOT should only use the `Received` header fields inserted by the known mail servers in the network.

B. Possible Evasion Techniques and Limitations

Given that SPOT relies on (content-based) spam filters to classify messages into spam and non-spam, spammers may try to evade the developed SPOT system by evading the deployed spam filters. They may send completely meaningless non-spam messages (as classified by spam filters). However, this will reduce the real spamming rate, and hence, the financial gains, of the spammers. More importantly, as shown in Figure 2 (b), even if a spammer reduces the spam percentage to 50%, SPOT can still detect the spam zombie with a relatively small number of observations (25 when $\alpha = 0.01$, $\beta = 0.01$, and $\theta_0 = 0.2$). Moreover, in certain environment where user feedback is reliable, for example, feedback from users of the same network in which SPOT is deployed, SPOT can also rely on classifications from end users in addition to the spam filters. As we have discussed in the previous section, trying to send spam at a low rate will also not evade the SPOT system. SPOT relies on the number of (spam) messages, not the sending rate, to detect spam zombies.

The current work has a number of limitations. First, we assumed that message arrivals are a random process, independent of each other. This may not be true, especially for spam messages, which tend to be generated in a batch with identical or similar content. Second, SPOT depends on spam filters to classify messages and no spam filters are perfect. The impacts of these limitations will be better studied in a future work.

VIII. Conclusion

In this paper we developed SPOT, an effective spam zombie detection system by monitoring outgoing messages in a network. SPOT was designed based on a simple and powerful statistical tool named Sequential Probability Ratio Test to detect the subset of compromised machines that are involved in the spamming activities. Our evaluation studies based on a 2-month email trace collected on the FSU campus network showed that SPOT is an effective and efficient system in automatically detecting compromised machines in a network. We have also evaluated two alternative designs based on spam count and spam fraction. We do not report the results due to the page limit. In summary, they are not as effective as SPOT.

Acknowledgments

We thank Mr. Michael Hodges and Mr. Thang Nguyen at the Information Technology Services of UH for their support of this project. Zhenhai Duan and Yingfei Dong were supported in part by NSF Grant CCF-0541096 and CNS-0649950, respectively. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of National Science Foundation.

References

[1] J. Markoff, "Russian gang hijacking PCs in vast scheme," *The New York Times*, Aug. 2008, http://www.nytimes.com/2008/08/06/technology/06hack.html.

[2] Y. Xie, F. Xu, K. Achan, R. Panigrahy, G. Hulten, and I. Osipkov, "Spamming botnets: Signatures and characteristics," in *Proc. ACM SIGCOMM*, Seattle, WA, Aug. 2008.

[3] L. Zhuang, J. Dunagan, D. R. Simon, H. J. Wang, I. Osipkov, G. Hulten, and J. D. Tygar, "Characterizing botnets from email spam records," in *Proc. of 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats*, San Francisco, CA, Apr. 2008.

[4] A. Wald, *Sequential Analysis*. John Wiley & Sons, Inc, 1947.

[5] M. Xie, H. Yin, and H. Wang, "An effective defense against email spam laundering," in *ACM Conference on Computer and Communications Security*, Alexandria, VA, October 30 - November 3 2006.

[6] G. Gu, P. Porras, V. Yegneswaran, M. Fong, and W. Lee, "Bothunter: Detecting malware infection through ids-driven dialog correlation," in *Proc. 16th USENIX Security Symposium*, Boston, MA, Aug. 2007.

[7] J. Jung, V. Paxson, A. Berger, and H. Balakrishnan, "Fast portscan detection using sequential hypothesis testing," in *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, CA, May 2004.

[8] S. Radosavac, J. S. Baras, and I. Koutsopoulos, "A framework for MAC protocol misbehavior detection in wireless networks," in *Proceedings of 4th ACM workshop on Wireless security*, Cologne, Germany, Sep. 2005.

[9] S. Linford, "Increasing spam threat from proxy hijacking," http://www.spamhaus.org/news.lasso?article=156.

[10] Y. Xie, F. Xu, K. Achan, E. Gillum, M. Goldszmidt, and T. Wobber, "How dynamic are IP addresses?" in *Proc. ACM SIGCOMM*, Kyoto, Japan, Aug. 2007.

[11] SpamAssassin, "The apache spamassassin project," http://spamassassin.apache.org/.

[12] Z. Duan, K. Gopalan, and X. Yuan, "Behavioral characteristics of spammers and their network reachability properties," in *IEEE International Conference on Communications (ICC)*, Jun. 2007.

[13] J. Klensin, "Simple Mail Transfer Protocol," RFC 5321, Oct. 2008.