Detecting Spam Zombies by Monitoring Outgoing Messages

Zhenhai Duan, *Senior Member*, *IEEE*, Peng Chen, Fernando Sanchez, Yingfei Dong, *Member*, *IEEE*, Mary Stephenson, and James Michael Barker

Abstract—Compromised machines are one of the key security threats on the Internet; they are often used to launch various security attacks such as spamming and spreading malware, DDoS, and identity theft. Given that spamming provides a key economic incentive for attackers to recruit the large number of compromised machines, we focus on the detection of the compromised machines in a network 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. In addition, we also evaluate the performance of the developed SPOT system using a two-month e-mail trace collected in a large US 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 e-mail 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 seven internal IP addresses associated with compromised machines in the trace are missed by SPOT. In addition, we also compare the performance of SPOT with two other spam zombie detection algorithms based on the number and percentage of spam messages originated or forwarded by internal machines, respectively, and show that SPOT outperforms these two detection algorithms.

Index Terms—Compromised machines, spam zombies, compromised machine detection algorithms.

1 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 spamming and spreading malware, DDoS, and identity theft [1], [10], [14]. Two natures of the compromised machines on the Internet—sheer volume and widespread—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 detection of the compromised machines in a network 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 [17], [19], [25]. 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 e-mail service provider [25], [26].

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 [25], [26] 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 [21]. SPRT is a powerful statistical method that can be used to test between two hypotheses (in our case, a machine is

- *Z. Duan and F. Sanchez are with the Department of Computer Science, Florida State University, Tallahassee, FL 32306-4530. E-mail: {duan, sanchez}@cs.fsu.edu.*
- *P. Chen is with Juniper Networks, Sunnyvale, CA 94089. E-mail: g_chenpeng@hotmail.com.*
- *Y. Dong is with the Department of Electrical Engineering, University of Hawaii, 2540 Dole Street, Holmes Hall 483, Honolulu, HI 96822. E-mail: yingfei@hawaii.edu.*
- *M. Stephenson is with Information Technology Services, Florida State University, 2035 E. Paul Dirac Drive, 200 BFS, Tallahassee, FL 32310. E-mail: mstephenson@fsu.edu.*
- *J.M. Barker is with Information Technology Services, The University of North Carolina at Chapel Hill, 211 Manning Drive, Chapel Hill, NC 27599-3420. E-mail: michael_barker@unc.edu.*

Manuscript received 27 Aug. 2010; revised 9 May 2011; accepted 9 June 2011; published online 30 Sept. 2011.
For information on obtaining reprints of this article, please send e-mail to: tdsc@computer.org, and reference IEEECS Log Number TDSC-2010-08-0149.
Digital Object Identifier no. 10.1109/TDSC.2011.49.

compromised versus 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 nonsequential 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 e-mail trace collected in a large US 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 e-mail 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 seven 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 three spam messages. For comparison, we also design and study two other spam zombie detection algorithms based on the number of spam messages and the percentage of spam messages originated or forwarded by internal machines, respectively. We compare the performance of SPOT with the two other detection algorithms to illustrate the advantages of the SPOT system.

The remainder of the paper is organized as follows: in Section 2, we discuss related work in the area of botnet detection (focusing on spam zombie detection schemes). We formulate the spam zombie detection problem in Section 3. Section 4 provides the necessary background on SPRT for developing the SPOT spam zombie detection system. In Section 5, we provide the detailed design of SPOT and the two other detection algorithms. Section 6 evaluates the SPOT detection system based on the two-month e-mail trace, and contrasts its performance with the two other detection algorithms. We briefly discuss the practical deployment issues and potential evasion techniques in Section 7, and conclude the paper in Section 8.

2 Related Work

In this section, we discuss related work in detecting compromised machines. We first focus on the studies that utilize spamming activities to detect bots and then briefly discuss a number of efforts in detecting general botnets.

Based on e-mail messages received at a large e-mail service provider, two recent studies [25], [26] 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 e-mail 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 named DBSpam to detect proxy-based spamming activities in a network relying on the packet symmetry property of such activities [23]. 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 [23].

In the following, we discuss a few schemes on detecting general botnets. BotHunter [8], 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.

An anomaly-based detection system named BotSniffer [9] identifies botnets by exploring the spatial-temporal behavioral similarity commonly observed in botnets. It focuses on IRC-based and HTTP-based botnets. In BotSniffer, flows are classified into groups based on the common server that they connect to. If the flows within a group exhibit behavioral similarity, the corresponding hosts involved are detected as being compromised. BotMiner [7] is one of the first botnet detection systems that are both protocol and structure independent. In BotMiner, flows are classified into groups based on similar communication patterns and similar malicious activity patterns, respectively. The intersection of the two groups is considered to be compromised machines. Compared to general botnet detection systems such as BotHunter, BotSniffer, and BotMiner, SPOT is a lightweight compromised machine detection scheme, by exploring the economic incentives for attackers to recruit the large number of compromised machines.

As a simple and powerful statistical method, Sequential Probability Ratio Test has been successfully applied in many areas [22]. In the area of networking security, SPRT has been used to detect portscan activities [12], proxy-based spamming activities [23], anomaly-based botnet detection [9], and MAC protocol misbehavior in wireless networks [16].

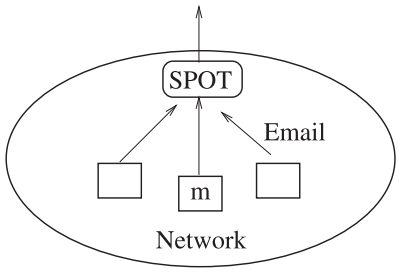


Fig. 1. Network model.

3 Problem Formulation and Assumptions

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.

Fig. 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. For example, the outgoing e-mail traffic (with destination port number of 25) can be replicated and redirected to the spam zombie detection system.

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,

$$Pr(X_i = 1|H_1) > Pr(X_i = 1|H_0), \tag{1}$$

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

We 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 [20]. None of existing spam filters can achieve perfect spam detection accuracy, and they all suffer from both false positive and false negative errors. The false negative rate of a spam filter measures the percentage of spam messages that are misclassified, and the false positive rate measures the percentage of nonspam messages that are misclassified. We note that all widely deployed spam filters have very low false negative and false positive rates, and such small spam classification errors will only have a marginal impact on the performance of the spam zombie detection algorithms (see Section 6).

We also 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 7.

In addition, we assume that an IP address corresponds to a unique machine and ignores the potential impacts of dynamic IP addresses on the detection algorithms in the presentation of the algorithms [3], [24]. We will investigate the potential impacts of dynamic IP addresses on the detection algorithms in Sections 5 and 6.

4 Background on Sequential Probability Ratio Test

In this section, we provide the necessary background on the Sequential Probability Ratio Test for understanding the proposed spam zombie detection system. Interested readers are directed to [21] 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.

As a simple and powerful statistical tool, SPRT has a number of compelling and desirable features that lead to the widespread applications of the technique in many areas [22]. First, both the actual false positive and false negative probabilities of SPRT can be bounded by the user-specified error rates. A smaller error rate tends to require a larger number of observations before SPRT terminates. Thus, users can balance the performance (in terms of false positive and false negative rates) and cost (in terms of number of required observations) 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 nonsequential statistical tests. In the following, we present the formal definition and a number of important properties of SPRT. The detailed derivations of the properties can be found in [21].

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,

$$Pr(X_i = 1|H_0) = 1 - Pr(X_i = 0|H_0) = \theta_0$$
$$Pr(X_i = 1|H_1) = 1 - Pr(X_i = 0|H_1) = \theta_1.$$

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 1 ($X_i = 1$), the constant $ln\frac{\theta_1}{\theta_0}$ is added to the preceding value of Λ. When the observation is 0 ($X_i = 0$), the constant $ln\frac{1-\theta_1}{1-\theta_0}$ is added.

The Sequential Probability Ratio Test 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 (3), then

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

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 [21],

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

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

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

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 (5), then the following relations hold:

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

and

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

Equations (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, (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 (5), given the user-specified false positive and false negative rates. 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.

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

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

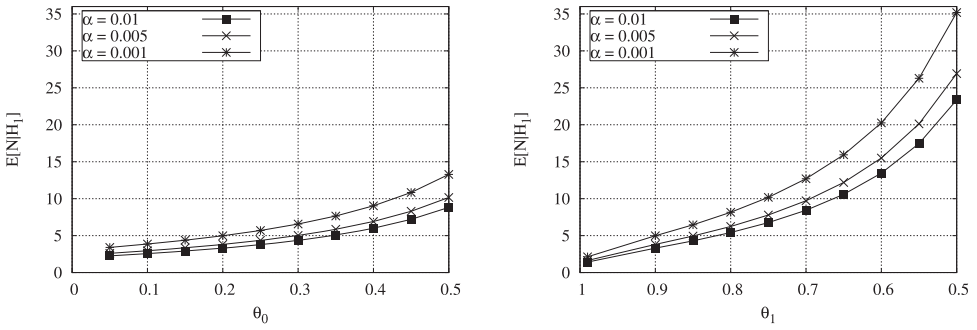
From the above equations, we can see that the average number of required observations when H_1 or H_0 is true depends on four parameters: the desired false positive and negative rates (α and β), and the distribution parameters θ_1 and θ_0 for hypotheses H_1 and H_0, respectively. We note that SPRT does 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.

5 Spam Zombie Detection Algorithms

In this section, we will develop three spam zombie detection algorithms. The first one is SPOT, which utilizes the Sequential Probability Ratio Test presented in the last section. We discuss the impacts of SPRT parameters on SPOT in the context of spam zombie detection. The other two spam zombie detection algorithms are developed based on the number of spam messages and the percentage of spam messages sent from an internal machine, respectively.

5.1 SPOT Detection Algorithm

SPOT is designed based on the statistical tool SPRT we discussed in the last section. 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, 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



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

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

parameters after we present the SPOT algorithm. Based on the user-specified values of α and β, the values of the two boundaries A and B of SPRT are computed using (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 (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 and additional observations are needed (lines 13 to 21).

Algorithm 1. SPOT spam zombie detection system

1: An outgoing message arrives at SPOT
2: Get IP address of sending machine m
3: // all following parameters specific to machine m
4: Let n be the message index
5: Let $X_n = 1$ if message is spam, $X_n = 0$ otherwise
6: **if** $(X_n == 1)$ **then**
7: // spam, 3
8: $\Lambda_n += ln\frac{\theta_1}{\theta_0}$
9: **else**
10: // nonspam
11: $\Lambda_n += ln\frac{1-\theta_1}{1-\theta_0}$
12: **end if**
13: **if** $(\Lambda_n \geq B)$ **then**
14: Machine m is compromised. Test terminates for m.
15: **else if** $(\Lambda_n \leq A)$ **then**
16: Machine m is normal. Test is reset for m.
17: $\Lambda_n = 0$
18: Test continues with new observations
19: **else**
20: Test continues with an additional observation
21: **end if**

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 reset, 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).

5.2 Parameters of SPOT Algorithm

SPOT requires four user-defined parameters: α, β, θ_1, and θ_0. In the following, we discuss how a user of the SPOT algorithm configures these parameters, and how these parameters may affect the performance of SPOT. As discussed in the previous section, α and β are the desired false positive and false negative rates. They are normally small values in the range from 0.01 to 0.05, which users of SPOT can easily specify independent of the behaviors of the compromised and normal machines in the network. As we have shown in Section 4, the values of α and β will affect the cost of the SPOT algorithm, that is, the number of observations needed for the algorithm to reach a conclusion. In general, a smaller value of α and β will require a larger number of observations for SPOT to reach a detection.

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, which are hard to obtain. A practical way to assign values to θ_1 and θ_0 is to use the detection rate and the false positive rate of the spam filter deployed together with the spam zombie detection system, respectively. Given that all the widely used spam filters have a high detection rate and low false positive rate [20], values of θ_1 and θ_0 assigned in this way should be very close to the true probabilities.

To get some intuitive understanding of the average number of required observations for SPRT to reach a decision, Figs. 2a and 2b 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 Fig. 2a, 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 $\theta_0 = 0.2$, SPRT requires about three 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, i.e., θ_0 increases, a slightly higher number of observations are required for SPRT to reach a detection.

In Fig. 2b, we assume the probability of a message being spam from a normal machine to be 0.2 ($\theta_0 = 0.2$). 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, 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).

We note that the statistical tool SPRT assumes events (in our cases, outgoing messages) are independently and identically distributed. However, it is well known that spam messages belonging to the same campaign are likely generated using the same spam template and delivered in batch; therefore, spam messages observed in time proximity may not be independent with each other. This can adversely affect the performance of SPOT in detecting compromised machines. As a consequence, both the detection accuracy of SPOT and the number of observations required to reach a detection decision can differ from the theoretical bounds provided in the statistical tool SPRT. We study the performance of SPOT using a real-world e-mail trace in Section 6.

5.3 Spam Count and Percentage-Based Detection Algorithms

For comparison, in this section, we present two different algorithms in detecting spam zombies, one based on the number of spam messages and another the percentage of spam messages sent from an internal machine, respectively. For simplicity, we refer to them as the count-threshold (CT) detection algorithm and the percentage-threshold (PT) detection algorithm, respectively.

In CT, the time is partitioned into windows of fixed length T. A user-defined threshold parameter C_s specifies the maximum number of spam message that may be originated from a normal machine in any time window. The system monitors the number of spam messages n originated from a machine in each window. If $n > C_s$, then the algorithm declares that the machine has been compromised.

Similarly, in the PT detection algorithm, the time is partitioned into windows of fixed length T. PT monitors two e-mail sending properties of each internal machine in each time window: one is the percentage of spam messages sent from a machine, another the total number of messages. Let N and n denote the total messages and spam messages originated from a machine m within a time window, respectively, then PT declares machine m as being compromised if $N \geq C_a$ and $\frac{n}{N} > P$, where C_a is the minimum number of messages that a machine must send, and P is the user-defined maximum spam percentage of a normal machine. The first condition is in place for preventing high false positive rates when a machine only generates a small number of messages. For example, in an extreme case, a machine may only send a single message and it is a spam, which renders the machine to have a 100 percent spam ratio. However, it does not make sense to classify this machine as being compromised based on this small number of messages generated.

In the following, we briefly compare the two spam zombie detection algorithms CT and PT with the SPOT system. The three algorithms have the similar running time and space complexities. They all need to maintain a record for each observed machine and update the corresponding record as messages arrive from the machine. However, unlike SPOT, which can provide a bounded false positive rate and false negative rate, and consequently, a confidence how well SPOT works, the error rates of CT and PT cannot be a priori specified.

In addition, choosing the proper values for the four user-defined parameters (α, β, θ_1, and θ_0) in SPOT is relatively straightforward (see the related discussion in the previous section). In contrast, selecting the "right" values for the parameters of CT and PT is much more challenging and tricky. The performance of the two algorithms is sensitive to the parameters used in the algorithm. They require a thorough understanding of the different behaviors of the compromised and normal machines in the concerned network and a training based on the behavioral history of the two different types of machines in order for them to work reasonably well in the network. For example, it can be challenging to select the "best" length of time windows in CT and PT to obtain the optimal false positive and false negative rates. We discuss how an attacker may try to evade CT and PT (and SPOT) in Section 7.

5.4 Impact of Dynamic IP Addresses

In the above discussion of the spam zombie detection algorithms, we have for simplicity ignored the potential impact of dynamic IP addresses and assumed that an observed IP corresponds to a unique machine. In the following, we informally discuss how well the three algorithms fair with dynamic IP addresses. We formally evaluate the impacts of dynamic IP addresses on detecting spam zombies in the next section using a two-month e-mail trace collected on a large US campus network.

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 Fig. 2, which shows the average number of observations required for SPRT to terminate with a conclusion. In practice, we have noted that three or four observations are sufficient for SPRT to reach a decision for the vast majority of cases (see the

TABLE 1
Summary of the E-Mail 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

performance evaluation of SPOT in the next section). If a machine is compromised, it is likely that more than three or four spam messages will be sent before the (unwitting) user shutdowns the machine and the corresponding IP address gets reassigned to a different machine. Therefore, dynamic IP addresses will not have any significant impact on SPOT.

Dynamic IP addresses can have a greater impact on the other two detection algorithms CT and PT. First, both require the continuous monitoring of the sending behavior of a machine for at least a specified time window, which in practice can be on the order of hours or days. Second, CT also requires a relatively larger number of spam messages to be observed from a machine before reaching a detection. By properly selecting the values for the parameters of CT and PT (for example, a shorter time window for machines with dynamic IP addresses), they can also work reasonably well in the environment of dynamic IP addresses.

6 Performance Evaluation

In this section, we evaluate the performance of the three detection algorithms based on a two-month e-mail trace collected on a large US campus network. We also study the potential impact of dynamic IP addresses on detecting spam zombies.

6.1 Overview of the E-Mail Trace and Methodology

The e-mail 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 e-mail trace collection, the mail server relayed messages destined for 53 subdomains in the FSU campus network. The mail relay server ran SpamAssassin [20] to detect spam messages. The e-mail trace contains the following information for each incoming message: the local arrival time, the IP address of the sending machine (i.e., the upstream mail server that delivered the message to the FSU mail relay server), 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 antivirus software. The antivirus 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 the detection algorithms. However, due to logistical constraints, we were not able to collect all such messages. Instead, we identified the messages in the e-mail 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 e-mails* and perform our evaluation of the detection algorithms based on the FSU e-mails. We note the set of FSU e-mails does not contain all the outgoing messages originated from inside FSU, and the compromised machines identified by the detection algorithms based on the FSU e-mails will likely be a lower bound on the true number of compromised machines inside FSU campus network.

An e-mail message in the trace is classified as either *spam* or *nonspam* by SpamAssassin [20] deployed in the FSU mail relay server. For ease of exposition, we refer to the set of all messages as the *aggregate* e-mails including both spam and nonspam. 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 address. Similarly, we refer to an IP address as *nonspam only* and *mixed* if we only receive nonspam messages, or we receive both spam and nonspam messages, respectively, from the IP address.

Table 1 shows a summary of the e-mail trace. As shown in the table, the trace contains more than 25 M e-mails, of which more than 18 M, or about 73 percent, are spam. About half of the messages in the e-mail trace were originated or forwarded by FSU internal machines, i.e., contained in the set of FSU e-mails. Table 2 shows the classifications of the observed IP addresses. As shown in the table, 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 percent sent at least one spam message. During the same course, we observed 440 FSU internal IP addresses.

Table 3 shows the classification of the observed IP addresses that sent at least one message carrying a virus/worm attachment. We note that a higher proportion of FSU internal IP addresses sent e-mails with a virus/worm

TABLE 3
Summary of IP Addresses Sending Virus/Worm

	Total	Non-spam only	Spam only	Mixed
# of IP	10,385	1,032	6,705	2,648
# of FSU IP	204	19	42	143

TABLE 2
Summary of Sending IP Addresses

	Total	Non-spam only	Spam only	Mixed
# of IP (%)	2,461,114	121,103 (4.9)	2,224,754 (90.4)	115,257 (4.7)
# of FSU IP (%)	440	175 (39.7)	74 (16.8)	191 (43.5)

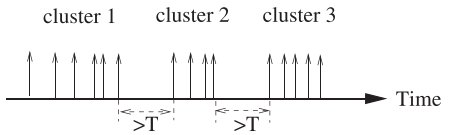


Fig. 3. Illustration of message clustering.

attachment than the overall IP addresses observed (all e-mails were destined to FSU accounts). This could be caused by a few factors. First, a (compromised) e-mail account in general maintains more e-mail addresses of friends in the same domain than other remote domains. Second, an (e-mail-propagated) virus/worm may adopt a spreading strategy concentrating more on local targets [2]. More detailed analysis of the e-mail trace can be found in [5] and [6], 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 detection algorithms, 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 reassigned to a different machine). We group messages according to the interarrival 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 a 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, and $|t_{k+1} - t_k|$ is referred to as the *time interval* between two clusters.

Fig. 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.

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 e-mail trace, we cannot decide which machines are responsible for the large number of spam messages, and consequently, determine the compromised machines. Section 7 discusses how we can handle this case in practical deployment.

6.2 Performance of SPOT

In this section, we evaluate the performance of SPOT based on the collected FSU e-mails. In all the studies, we set $\alpha = 0.01$, $\beta = 0.01$, $\theta_1 = 0.9$, and $\theta_0 = 0.2$.

Table 4 shows the performance of the SPOT spam zombie detection system. As discussed above, there are

TABLE 4
Performance of SPOT

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

440 FSU internal IP addresses observed in the e-mail 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 rely on a number of ways to verify if a machine is indeed compromised. 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 percent), 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 percent) 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 six 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 toward the end of the collection period. This indicates that they may get compromised toward 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 5). 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 e-mail collection period, but for which SPOT has reset the records (lines 15 to 18 in Algorithm 1). That is, 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. Seven 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 4. 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

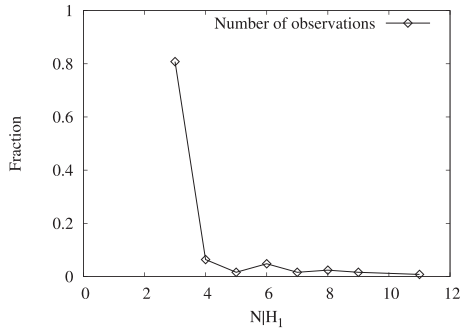


Fig. 4. Number of actual observations.

be detected by antivirus softwares, and hence deleted before reaching the intended recipients. This is confirmed by the low percentage of infected messages in the overall e-mail trace shown in Table 1. 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. A number of factors can contribute to this observation. As we have discussed in Section 5, the IID assumption of the underlying statistical tool SPRT may not be satisfied in the spam zombie detection problem. In addition, the spam filter used to classify messages did not have a perfect spam detection accuracy, and as a consequence, events may be misclassified. Moreover, the FSU e-mails can only provide a rather limited view of the outgoing messages originated from inside FSU, which will also affect the performance of SPOT in detection spam zombie machines.

Fig. 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 percent of the compromised machines are detected by SPOT with only three 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.

6.3 Performance of CT and PT

In this section, we evaluate the performance of CT and PT and compare their performance with that of SPOT, using the same two-month e-mail trace collected on the FSU campus network. Recall that CT is a detection algorithm based on the number of spam messages originated or forwarded by an internal machine, and PT based on the percentage of spam messages originated or forwarded by an internal machine (see Section 5.3). For comparison, we also include a simple spam zombie detection algorithm that identifies any machine sending at least a single spam message as a compromised machine. We note that such a simple scheme may not be deployed due to a potential high false positive rate; however, it provides us with the insights into the performance gain we can obtain by employing a more sophisticated spam zombie detection system.

TABLE 5
Performance of CT and PT

	Total # FSU IP	Detected	Confirmed (%)	Missed (%)
CT	440	81	79 (59.8)	53 (40.2)
PT	440	84	83 (61.9)	51 (38.1)
Simple	440	210	157 (89.7)	18 (10.3)

In this evaluation study, we set the length of time windows to be 1 hour, that is, $T = 1$ hour, for both CT and PT. For CT, we set the maximum number of spam messages that a normal machine can send within a time window to be 30 ($C_s = 3$), that is, when a machine sends more than 30 spam messages within any time windows, CT concludes that the machine is compromised. In PT, we set the minimum number of (spam and nonspam) messages within a time window to be 6 ($C_a = 6$), and the maximum percentage of spam messages within a time window to be 50 percent ($P = 50\%$). That is, if more than 50 percent of all messages sent from a machine are spam in any time window with at least six messages in the window, PT will conclude that the machine is compromised. We choose the values for the parameters of PT in this way so that it is relatively comparable with SPOT. Recall that based on our empirical study in the last section, the minimum number of observations needed by SPOT to reach a detection is 3 (when $\alpha = 0.01$, $\beta = 0.01$, $\theta_0 = 0.2$, and $\theta_1 = 0.9$).

Table 5 shows the performance of CT and PT, which includes the number of compromised IP addresses detected, confirmed, and missed. We use the same methods to confirm a detection or identify a missed IP address as we have done with the SPOT detection algorithm. From the table we can see that, CT and PT have a worse performance than SPOT. For example, CT only detects 81 IP addresses as being compromised. Among the 81 IP addresses, 79 can be confirmed to be associated with compromised machines. However, CT missed detecting 53 IP addresses associated with compromised machines. The detection rate and false negative rate of CT is 59.8 and 40.2 percent, respectively, much worse than that of SPOT, which are 94.7 and 5.3 percent, respectively. We also note that all the compromised IP addresses detected (confirmed) using CT or PT are also detected (confirmed) using the SPOT detection algorithm. That is, the IP addresses detected (confirmed) using CT and PT are a subset of compromised IP addresses detected (confirmed) using the SPOT detection algorithm. The IP addresses associated with compromised machines that are missed by SPOT are also missed by CT and PT. We conclude that SPOT outperforms both CT and PT in terms of both detection rate and miss rate.

In the table, we also show the performance of the simple spam zombie detection algorithm. We first note that the methods to confirm a detection or identify a missed

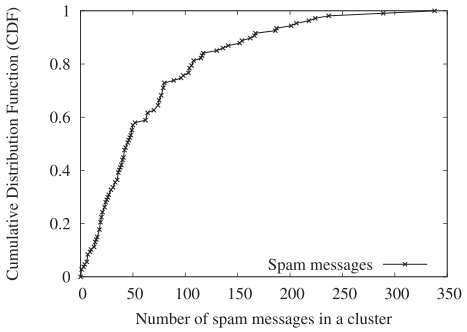


Fig. 5. Distribution of spam messages in each cluster.

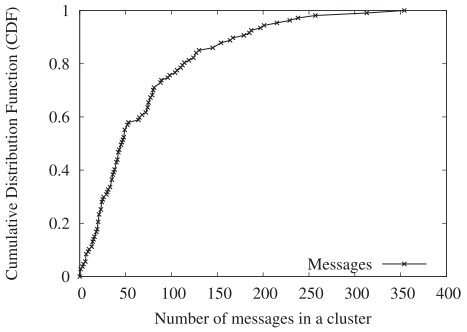


Fig. 6. Distribution of total messages in each cluster.

IP address are different from the ones used in SPOT, CT, and PT, due to the special property of the simple algorithm. For this simple algorithm, we confirm the detection of a compromised machine if the machine sent at least one outgoing message carrying a virus/worm attachment. Similarly, if a machine sent out at least one outgoing message carrying a virus/worm attachment, but the simple algorithm fails to detect it as a compromised machine, then we have identified a missed IP address. In this process, we have similarly excluded all the whitelisted mail-server machines. As we have discussed early that, the antivirus software and SpamAssassin were two independent components deployed at the FSU mail relay server, and a small number of messages carrying virus/worm attachments were not detected as spam by the spam filter. Due to the difference in the methods of confirming a detection or identifying a missed IP address, the four detection algorithms observe different number of confirmed and missed IP addresses.

From the table we can see that, the simple detection algorithm can detect more machines (210) as being compromised than SPOT, CT, and PT. It also has better performance than CT and PT in terms of both detection rate (89.7 percent) and false negative rate (10.3 percent). The simple algorithm has worse performance than the SPOT algorithm (see Table 4). To a degree, this result is surprising in that the simple detection algorithm can have better performance than both the CT and PT algorithms. However, we caution that this observation could be caused by the limitation of the FSU e-mails used in the study. Recall that this e-mail trace only contains outgoing messages destined to internal FSU accounts, and as we have discussed, a higher portion of FSU internal IP addresses sent e-mails with a virus/worm attachment to FSU internal accounts than the overall IP addresses observed, which results in a higher confirmation rate in the compromised machine detection. We expect the percentage of FSU machines sending e-mails with a virus/worm attachment will drop if the e-mail trace contains all outgoing messages (instead of only the ones destined to internal FSU accounts), which will adversely affect the observed performance of the simple detection algorithm.

6.4 Dynamic IP Addresses

In this section, we conduct studies to understand the potential impacts of dynamic IP addresses on the performance of the three detection algorithms. Given that SPOT outperforms both CT and PT, our discussion will focus on the impacts on SPOT; similar observations also apply to CT and PT.

In order to understand the potential impacts of dynamic IP addresses on the detection algorithms, 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 interarrival time no greater than 30 minutes are grouped into the same cluster. Given the short interarrival 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 reassigned to a different machine within the concerned cluster. (It is possible that messages from multiple adjacent clusters are actually sent from the same machine.)

Fig. 5 shows the cumulative distribution function (CDF) of the number of spam messages in each cluster. From the figure, we can see that more than 90 percent of the clusters have no less than 10 spam messages, and more than 96 percent no less than three 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. Fig. 6 shows the CDF of the number of all messages (including both spam and nonspam) in each cluster. Similar observations can be made to that in Fig. 5.

Fig. 7 shows the CDF of the durations of the clusters. As we can see from the figure, more than 75 and 58 percent of the clusters last no less than 30 minutes and 1 hour

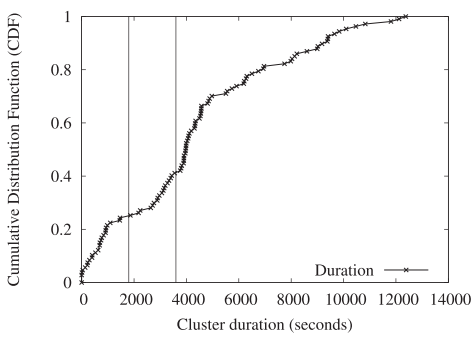


Fig. 7. Distribution of the cluster duration.

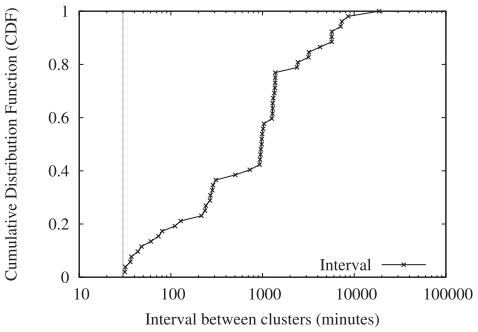


Fig. 8. Distribution of time intervals between clusters.

(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. Fig. 8 shows the CDF of the time intervals between consecutive clusters. As we can see from the figure, the minimum time interval between two consecutive clusters is slightly more than 30 minutes (31.38 minutes), and the longest one is close to 13 days (18,649.38 minutes). Moreover, more than 88 percent of all intervals between clusters are longer than 1 hour.

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 percent) 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.

7 Discussion

In this section, we discuss the practical deployment issues and possible techniques that spammers may employ to evade the detection algorithms. Our discussions will focus on the SPOT detection algorithm.

7.1 Practical Deployment

To ease exposition, we have assumed that a sending machine m (Fig. 1) is an end-user client machine. It cannot be a mail relay server deployed by the network. 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 mail servers, which forward all outgoing messages (or SPOT gets a replicated stream of all outgoing messages), as discussed in Section 3. SPOT relies on the `Received` header fields to identify the originating machine of a message in the network [13], [18]. Given that the `Received` header fields can be spoofed by spammers [19], SPOT should only use the `Received` header fields inserted by the known mail servers in the network. SPOT can determine the reliable `Received` header fields by backtracking from the last known mail server in the network that forwards the message. It terminates and identifies the originating machine when an IP address in the `Received` header field is not associated with a known mail server in the network. The similar practical deployment methods also apply to the CT and PT detection algorithms.

7.2 Possible Evasion Techniques

Given that the developed compromised machine detection algorithms rely on (content-based) spam filters to classify messages into spam and nonspam, spammers may try to evade the detection algorithms by evading the deployed spam filters. They may send completely meaningless "nonspam" messages (as classified by spam filters). However, this will reduce the real spamming rate, and hence, the financial gains, of the spammers [4]. More importantly, as shown in Fig. 2b, even if a spammer reduces the spam percentage to 50 percent, SPOT can still detect the spam zombie with a relatively small number of observations (25 when $\alpha = 0.01$, $\beta = 0.01$, and $\theta_0 = 0.2$). So, trying to send nonspam messages will not help spammers to evade the SPOT system.

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 rely on classifications from end users (in addition to the spam filter). Although completely meaningless messages may evade the deployed spam filter, it is impossible for them to remain undetected by end users who receive such messages. User feedbacks may be incorporated into SPOT to improve the spam detection rate of the spam filter. 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 performance evaluation of SPOT was carried out using the FSU e-mails collected in the year of 2005. However, based on the above discussion, we expect that SPOT will work equally well in today's environment. Indeed, as long as the spam filer deployed together with SPOT can provide a reasonable spam detection rate, the SPOT system should be effective in identifying compromised machines. We also argue that a spam zombie detection system such as SPOT is in an even greater need, given that a large percentage of spam messages were originated from spamming bots in recent years. Various studies in recent years have shown that spam messages sent from botnets accounted for above 80 percent of all spam messages on the Internet [11], [15]. For example, the MessageLabs Intelligence 2010 annual security report showed that approximately 88.2 percent of all spam in 2010 were sent from botnets.

As we have discussed in Section 5.3, selecting the "right" values for the parameters of CT and PT is much more challenging and tricky than those of SPOT. In addition, the parameters directly control the detection decision of the two detection algorithms. For example, in CT, we specify the maximum number of spam messages that a normal machine can send. Once the parameters are learned by the spammers, they can send spam messages below the configured threshold parameters to evade the detection algorithms. One possible countermeasure is to configure the

algorithms with small threshold values, which helps reduce the spam sending rate of spammers from compromised machines, and therefore, the financial gains of spammers. Spammers can also try to evade PT by sending meaningless "nonspam" messages. Similarly, user feedback can be used to improve the spam detection rate of spam filters to defeat this type of evasions.

8 Conclusion

In this paper, we developed an effective spam zombie detection system named SPOT 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 compromised machines that are involved in the spamming activities. SPOT has bounded false positive and false negative error rates. It also minimizes the number of required observations to detect a spam zombie. Our evaluation studies based on a two-month e-mail trace collected on the FSU campus network showed that SPOT is an effective and efficient system in automatically detecting compromised machines in a network. In addition, we also showed that SPOT outperforms two other detection algorithms based on the number and percentage of spam messages sent by an internal machine, respectively.

Acknowledgments

The authors thank the anonymous reviewers of IEEE INFOCOM 2009 and the *IEEE Transactions on Dependable and Secure Computing*. Their insightful and constructive comments helped improve both the technical details and presentation of the paper. Zhenhai Duan and Fernando Sanchez were supported in part by US National Science Foundation (NSF) Grant CNS-1041677. Yingfei Dong was supported in part by NSF Grants CNS-1041739 and CNS-1018971. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the NSF. A preliminary version of this paper appeared in the *Proceedings of IEEE INFOCOM 2009* with the same title. This project was carried out while Peng Chen was a graduate student at the Florida State University. This project was carried out while James Michael Barker was with the Florida State University.

References

[1] P. Bacher, T. Holz, M. Kotter, and G. Wicherski, "Know Your Enemy: Tracking Botnets," http://www.honeynet.org/papers/bots, 2011.
[2] Z. Chen, C. Chen, and C. Ji, "Understanding Localized-Scanning Worms," *Proc. IEEE Int'l Performance, Computing, and Comm. Conf. (IPCCC '07)*, 2007.
[3] R. Droms, "Dynamic Host Configuration Protocol," IETF RFC 2131, Mar. 1997.
[4] Z. Duan, Y. Dong, and K. Gopalan, "DMTP: Controlling Spam through Message Delivery Differentiation," *Computer Networks*, vol. 51, pp. 2616-2630, July 2007.
[5] Z. Duan, K. Gopalan, and X. Yuan, "Behavioral Characteristics of Spammers and Their Network Reachability Properties," Technical Report TR-060602, Dept. of Computer Science, Florida State Univ., June 2006.
[6] Z. Duan, K. Gopalan, and X. Yuan, "Behavioral Characteristics of Spammers and Their Network Reachability Properties," *Proc. IEEE Int'l Conf. Comm. (ICC '07)*, June 2007.
[7] G. Gu, R. Perdisci, J. Zhang, and W. Lee, "BotMiner: Clustering Analysis of Network Traffic for Protocol- and Structure-Independent Botnet Detection," *Proc. 17th USENIX Security Symp.*, July 2008.
[8] G. Gu, P. Porras, V. Yegneswaran, M. Fong, and W. Lee, "BotHunter: Detecting Malware Infection through Ids-Driven Dialog Correlation," *Proc. 16th USENIX Security Symp.*, Aug. 2007.
[9] G. Gu, J. Zhang, and W. Lee, "BotSniffer: Detecting Botnet Command and Control Channels in Network Traffic," *Proc. 15th Ann. Network and Distributed System Security Symp. (NDSS '08)*, Feb. 2008.
[10] N. Ianelli and A. Hackworth, "Botnets as a Vehicle for Online Crime," *Proc. First Int'l Conf. Forensic Computer Science*, 2006.
[11] J.P. John, A. Moshchuk, S.D. Gribble, and A. Krishnamurthy, "Studying Spamming Botnets Using Botlab," *Proc. Sixth Symp. Networked Systems Design and Implementation (NSDI '09)*, Apr. 2009.
[12] J. Jung, V. Paxson, A. Berger, and H. Balakrishnan, "Fast Portscan Detection Using Sequential Hypothesis Testing," *Proc. IEEE Symp. Security and Privacy*, May 2004.
[13] J. Klensin, "Simple Mail Transfer Protocol," IETF RFC 2821, Apr. 2001.
[14] J. Markoff, "Russian Gang Hijacking PCs in Vast Scheme," *The New York Times*, http://www.nytimes.com/2008/08/06/technology/06hack.html, Aug. 2008.
[15] P. Wood et al., "MessageLabs Intelligence: 2010 Annual Security Report," 2010.
[16] S. Radosavac, J.S. Baras, and I. Koutsopoulos, "A Framework for MAC Protocol Misbehavior Detection in Wireless Networks," *Proc. Fourth ACM Workshop Wireless Security*, Sept. 2005.
[17] A. Ramachandran and N. Feamster, "Understanding the Network-Level Behavior of Spammers," *Proc. ACM SIGCOMM*, pp. 291-302, Sept. 2006.
[18] P. Resnick, "Internet Message Format," IETF RFC 2822, Apr. 2001.
[19] F. Sanchez, Z. Duan, and Y. Dong, "Understanding Forgery Properties of Spam Delivery Paths," *Proc. Seventh Ann. Collaboration, Electronic Messaging, Anti-Abuse and Spam Conf. (CEAS '10)*, July 2010.
[20] SpamAssassin, "The Apache SpamAssassin Project," http://spamassassin.apache.org, 2011.
[21] A. Wald, *Sequential Analysis*. John Wiley & Sons, 1947.
[22] G.B. Wetherill and K.D. Glazebrook, *Sequential Methods in Statistics*. Chapman and Hall, 1986.
[23] M. Xie, H. Yin, and H. Wang, "An Effective Defense against Email Spam Laundering," *Proc. ACM Conf. Computer and Comm. Security*, Oct./Nov. 2006.
[24] Y. Xie, F. Xu, K. Achan, E. Gillum, M. Goldszmidt, and T. Wobber, "How Dynamic Are IP Addresses?" *Proc. ACM SIGCOMM*, Aug. 2007.
[25] Y. Xie, F. Xu, K. Achan, R. Panigrahy, G. Hulten, and I. Osipkov, "Spamming Botnets: Signatures and Characteristics," *Proc. ACM SIGCOMM*, Aug. 2008.
[26] L. Zhuang, J. Dunagan, D.R. Simon, H.J. Wang, I. Osipkov, G. Hulten, and J.D. Tygar, "Characterizing Botnets from Email Spam Records," *Proc. First Usenix Workshop Large-Scale Exploits and Emergent Threats*, Apr. 2008.

Zhenhai Duan received the BS degree from Shandong University, China, in 1994, the MS degree from Beijing University, China, in 1997, and the PhD degree from the University of Minnesota, in 2003, all in computer science. He joined the faculty of the Department of Computer Science at the Florida State University in 2003, where he is now an associate professor. His research interests include computer networks and network security. He was a corecipient of Best Paper Award of the 2002 IEEE International Conference on Network Protocols (ICNP), the 2006 IEEE International Conference on Computer Communications and Networks (ICCCN), and the IEEE GLOBECOM 2008. He has served as a TPC cochair for CEAS 2011, the IEEE GLOBECOM 2010 Next-Generation Networking Symposium, and the IEEE ICCCN 2007 and 2008 Network Algorithms and Performance Evaluation Track. He is a member of the ACM and a senior member of the IEEE.

Peng Chen received the BEng degree in management information systems from Tianjin University, China, in 1999, the MEng degree in computer science from Beihang University, China, in 2006, and the MS degree in computer science from the Florida State University, in 2008. He is currently a software engineer at Juniper Networks, Sunnyvale, California. His research interests include computer networks, Internet routing protocols, and networking security.

Fernando Sanchez received the BS degree from the Universidad San Francisco de Quito, Ecuador, in 2002, and the MS degree from Florida State University in 2007, all in computer science. Currently, he is working toward the PhD degree in the Department of Computer Science at the Florida State University. His research interests include network security with interest in networking protocols security and network services security.

Yingfei Dong received the BS and MS degrees in computer science from the Harbin Institute of Technology, P.R. China, in 1989 and 1992, the doctor degree in engineering from Tsinghua University in 1996, and the PhD degree in computer and information science from the University of Minnesota in 2003. He is currently an associate professor in the Department of Electrical Engineering at the University of Hawaii at Manoa. His current research mostly focuses on computer networks, especially in network security, real-time networks reliable communications, multimedia content delivery, Internet services, and distributed systems. His work has been published in many referred journals and conferences. He has served as both organizer and program committee member for many IEEE/ACM/IFIP conferences. His current research is supported by US National Science Foundation. He is a member of the IEEE.

Mary Stephenson received BS degrees in meteorology and in mathematics/computer science from the Florida State University in 1983 and the MS degree in meteorology from the University of Oklahoma in 1987. She is currently an associate director in Information Technology Services at Florida State University, within the Infrastructure and Operations Services Group. She is currently responsible for the University's enterprise Unix and Linux systems, storage, backup, and virtualization technologies and the University's data center facility.

James Michael Barker received the BA degree in philosophy from the College of William and Mary and the MA and PhD degrees in philosophy from the Florida State University. He is currently employed at the University of North Carolina at Chapel Hill (UNC Chapel Hill) as an assistant vice chancellor and chief technology officer. He is responsible for the Infrastructure and Operations Division as well as the Communication Technologies Division. His areas of operational oversight include UNC Chapel Hill's network, telephony, datacenters, messaging/communication services, operating systems hosting, identity management, middleware services, storage systems, and workload automation. In this role, he is responsible for the core information technology infrastructure services of UNC Chapel Hill's "enterprise" as well as building block services for campus units. He previously served as the director of the University Computing Services at the Florida State University. His dominant research interests include Kant, the history of modern philosophy, and philosophy of technology.