Fast Detection of Local Scanners Using Adaptive Methods

Ahren Studer and Chenxi Wang

March 28, 2006
CMU-CyLab-06-004

CyLab
Carnegie Mellon University
Pittsburgh, PA 15213

Fast Detection of Local Scanners Using Adaptive Methods

Ahren Studer and Chenxi Wang
Carnegie Mellon University
{astuder, chenxi}@andrew.cmu.edu

Abstract

Network attacks often employ scanning to locate vulnerable hosts and services. Unimpeded scanning can lead to the subversion of an entire vulnerable population in a matter of minutes. Fast and accurate detection of local scanners is key to contain a spreading epidemic in its early stage. Existing scan detection schemes can detect fast scanners whose behavior can be clearly delineated from that of legitimate traffic. Detecting slow scanners, however, is more difficult. The difficulty arises partially from the fact that these detection schemes use statically determined detection criteria, and as a result do not respond well to traffic perturbations. In this paper, we present two adaptive scan detection schemes, *Success Based* (SB) and *Failure Based* (FB), both of which change detection criteria dynamically based on traffic statistics. FB is designed for fast detection and is particularly well suited for controlled computing environments with well-understood traffic characteristics. SB is more versatile and able to perform well in a wide range of traffic scenarios. We evaluate the proposed schemes analytically as well as empirically using real traffic and attack traces. Our results show that against fast scanners, the adaptive schemes are able to render similar detection precision as the traditional static schemes. For slow scanners, however, the adaptive schemes are much more effective, both in terms of detection precision and speed. Specifically, both SB and FB have non-linear properties not present in other schemes. These properties permit a lower *Sustained Scanning Threshold* and a robustness against perturbations in the background traffic.

Keywords: Scan Detection, Internet Worms, Security

1 Introduction

Modern network attacks commonly employ port scans to locate vulnerable machines and services. A large amount of scan activity is therefore a strong indicator of malicious reconnaissance activities, often to be followed by exploits or infections. As such, an important piece in a network defense is fast and accurate detection of local scanners. With few exceptions, existing scan detectors are exclusively what we call *static rate schemes*. Such a scheme relies on a statically determined arrival rate of suspicious events to delineate the behaviors of legitimate hosts from those of scanners. For instance, NSM [1] permits a host to contact a maximum number of distinct addresses in a given time window. Any host that exceeds this rate is flagged as a potential scanner. These schemes work well for fast scanners whose behaviors are distinctively different from legitimate hosts. Detecting slow scanners, however, is more difficult because slow scans tend to blend in with the background traffic. If you set the rate too low, false positives can occur whilst a large rate will permit a liberal amount of scans.

In this paper, we investigate *adaptive rate* schemes concerning the detection of slow scanners in the presence of background traffic. We show that by adaptively changing the permitted rate of suspicious events, we achieve interesting and powerful "non-linear" properties that are not present in the static schemes, and that these properties lead to more effective and robust detection against various forms of scanning malware. We introduce two adaptive schemes in this paper, one changes the permitted rate (of suspicious events) based on the host's connection success statistics (we call it *SB* for success based) and the other one based on the failure statistics (we call this one *FB*). We show that both SB and FB are able to catch slow scanners while remaining just as effective against fast scanners as static-rate schemes.

Throughout this paper, we use a *token-based framework* to describe and analyze each scan detection scheme. More specifically, in this framework a scan detector allocates a number of tokens to each host in the beginning. Each ensued suspicious event constitutes the removal of some number of tokens, and tokens are rewarded back in an algorithm-specific fashion. The net rate at which tokens are rewarded determines the permitted arrival rate of

suspicious events—a host that exceeds this rate for some length of time is labeled as a scanner. In a static-rate scheme, the permitted arrival rate of suspicious events is constant. In an adaptive scheme, this rate is dynamically determined, based on traffic charactersitics.

It is easily seen how some of the existing scan detectors fit into this framework. For instance, we can use the token balance to represent the state of the random walk in TRW [2]; a step in the walk toward the scanner hypothesis represents the consumption of tokens, and a step in the opposite direction constitutes the accumulation of tokens. Similarly, an NSM scheme that enforces 15 distinct addresses per minute can be translated into a token-based scheme that consumes four tokens per distinct-destination contact and accrues tokens at a rate of 1 per second. The use of this framework simplifies the representation of specific schemes; sometimes a family of algorithms can be described with a single token-based represenation (e.g., TRW and RHT). It abstracts away superfulous details and permits the direct comparison of core design choices.

To contrast and compare the adaptive schemes with others, we focus on these aspects of detection performance; *error rates*, *detection speed*, and *Sustained Scanning Threshold*(SST). Error rates are specifically false positive and false negative rates. We use the metric `Escaped_Count` to measure detection speed. `Escaped_Count` is defined as the number of scans permitted from a scanning host before detection occurs. The Sustained Scanning Threshold (SST) is the maximum failure rate a host can maintain without being labeled as a scanner[1]. SST is an especially important metric concerning slow scanners—the lower the SST, the more effective it is against slow scanners.

To investigate these aspects, we tested each scheme against both real and synthetic network and scan traces. Our analysis shows that both SB and FB give rise to a lower Sustained Scanning Threshold, while maintaining comparable false positive levels to the other detectors. More specifically, SB and FB are equally as effective against fast scanners as the static-rate schemes, but are substantially more so against stealth scanners, both in terms of precision and speed. A sensitivity analysis shows that, even though the adaptive detectors do not strictly render better detection precision, it is robust, in the sense that their performances are only slightly affected by perturbations in the background traffic.

The remainder of the paper is organized as follows. Section 2 discusses related work. In Section 3, we describe the token-based framework. Section 4 covers the static threshold scheme we use as a baseline comparison case. We present the adaptive scan detection schemes in Section 5 and provide an analytical and empirical comparison of the static vs. adaptive schemes in Section 6. We conclude in Section 7.

2 Related Work

Many scan detection schemes have been proposed in the literature. The earlier ones, such as NSM [1], Snort [3], and Bro [4], are all static rate schemes which simply count the number of distinct destinations or failures of each host within a given window of time, and label the host as a scanner if a pre-determined rate is exceeded. These schemes tend to adopt generous permitted rates for fear of false positives. As a result, they are not as effective when slow scanners are concerned.

Jung et al. [5] developed a scheme that uses a threshold random walk (TRW) to detect scanners. In this scheme, a connection success results in a step in one direction, while a failure is a step in the opposite direction. A pre-determined distance traveled in a direction labels the host either as a scanner or a safe host. Reverse Hypothesis Testing (RHT) [6] and the Approximate TRW [2] are variations of TRW. Ganesh et al. [7] developed another scheme where optimal detection is possible if traffic characteristics are known. In this scheme, the time between failures dictates the number of tokens removed or rewarded. These algorithms are closest to our schemes and also belong in the class of adaptive algorithms. We present an analytic treatment of worm detection in a generic token based framework and show that our adaptive algorithms are less susceptible to intelligent gaming, more robust to background perturbations, and more effective against slow scanning worms.

Other defenses against worms include automatic containment and signature generation. Rate limiting such as Williamson's [8], Chen et al's [9], and Wong et al.'s DNS-based scheme [10] are examples of containment schemes. This class of mechanisms focuses on containing potentially anomalous traffic and has different goals and constraints than detection schemes. Signature generation techniques such as Earlybird [11], Autograph [12], and Polygraph [13] have great potential but thus far proved to be difficult against zero-day worms, in particular against slow spreading

[1] This concept was originally defined by Weaver et al. in [2].

Symbol	Meaning
a	token reward rate
γ	token consumed per failure
n	initial token balance (also maximum)
$\eta(t)$	background traffic (non-scan) rate at time t
μ	expected background traffic rate ($E[\eta]$)
r_s	rate of scan connections
p_n	background traffic success probability
p_s	scan traffic success probability

Table 1: Token Equation Parameters

worms.

3 A Token Based Framework

To facilitate analysis, throughout this paper we use a token-based framework to represent the different detection schemes. In this framework, each host has a bank of tokens. Tokens are removed when suspicious events happen (e.g., connection failures), and accrued either at a pre-specified rate or in the absence of suspicious events. As such, the consumption of tokens models the occurrence of suspicious events, and an increase in the token balance is indicative of benign/good behavior. The scan detector regulates the subtraction and addition of tokens and reports that the host is a scanner if the token balance reaches a pre-determined level.

We map the logic of each detection scheme into this framework. To normalize the discussion, token consumption occurs only when outbound connections fail[2]. Exactly how many tokens are removed and the conditions under which tokens are rewarded are algorithm-specific. As we discuss later in the paper, the ways in which each algorithm regulates these two parameters are often the main source of performance differences.

In this study, we represent the connection rate of legitimate traffic with a random variable, η, that follows a probability density function $f_\eta(\cdot)$ with an expected value of μ. We assume that both legitimate and scan traffic exhibit a consistent success probability over time. We further assume that scans are emitted at a constant rate. These assumptions permit us to represent the scan traffic using a simple scan rate, r_s, and a scan success probability, p_s. We assume that the success probability of normal traffic is p_n. Table 1 summarizes the different parameters used in the paper. Equation 1 gives the token balance of a host at time T, assuming the token reward rate, a, and tokens consumed per failure, γ, are both constant.

$$Tokens(T) = \min(n, n + aT - \gamma(r_s(1 - p_s) - \int_0^T \eta(t)(1 - p_n)dt)) \tag{1}$$

Equation 2 shows the condition for no false positive at time T; that is, the tokens consumed within T must be less than the tokens rewarded.

$$\gamma \int_0^T \eta(t)(1 - p_n)dt \leq \; n + aT \tag{2}$$

Equation 3 shows the detection condition—only when the overall failure rate (both scan and non-scan) is greater than the token reward rate, will the scanner be detected. Otherwise a false negative can occur.

$$r_s(1 - p_s)T + \int_0^T \eta(t)(1 - p_n)dt > aT/\gamma \tag{3}$$

The maximum scan rate that a scanner can sustain without being detected is the *Sustained Scanning Threshold*,

[2]The exact conditions for which a connection is considered "failed" can vary for each algorithm

or *SST* [2]. The SST of a detector is an important quantity because it denotes the optimal worm scan rate against the detector. Clearly, a detection mechanism that gives rise to a low SST while maintaining acceptable error rates is desirable. The expected value for SST is shown in Equation 4.

$$E[SST] = \frac{a/\gamma - \mu(1 - p_n)}{1 - p_s} \tag{4}$$

As previously stated, another metric we use is `Escaped_Count`, which measures the timeliness of the detector. `Escaped_Count` is defined as the number of scans permitted before detection. For instance, a scanner that successfully evades detection (a false negative) would have an infinite `Escaped_Count`. Equation 5 gives the expected `Escaped_Count`.

$$E[\texttt{Escaped\_Count}] \approx \frac{\texttt{E[Tokens]}}{\gamma(\texttt{r}_\texttt{s}(1 - \texttt{p}_\texttt{s}) + \mu(1 - \texttt{p}_\texttt{n})) - \texttt{a}} \cdot \texttt{r}_\texttt{s} \tag{5}$$

Given the `Escaped_Count`, we can calculate the "basic reproduction number"; the average number of hosts infected by an instance of the spreading malware before detection occurs. If the basic reproduction number of an infectious spread is greater than one, the infection will achieve an exponential growth. Therefore, the condition under which an epidemic can be contained is,

$$E[\texttt{Escaped\_Count}] \cdot \texttt{p}_\texttt{s} = \texttt{r}_\texttt{s}\texttt{p}_\texttt{s}\frac{\texttt{E[Tokens]}}{\gamma(\texttt{r}_\texttt{s}(1 - \texttt{p}_\texttt{s}) + \mu(1 - \texttt{p}_\texttt{n})) - \texttt{a}} \leq \texttt{1} \tag{6}$$

Setting Equation 6 equal to one and solving for r_s gives another important quantity, the *Critical Scan Threshold*, or CST, such that,

$$E[CST] = \frac{a/\gamma - \mu(1 - p_n)}{1 - p_s(E[Tokens]/\gamma + 1)} \tag{7}$$

CST was originally defined by Ganesh et al. [7]; it is the expected *maximum* scan rate for which the basic reproduction number is one. Scans emitted at a rate higher than the CST will be detected and stopped before infecting one other host, hence successfully contained. When p_s and $E[Tokens]/\gamma$ are relatively small, the Sustained Scanning Threshold (SST) approximately equals the Critical Scanning Threshold (CST). Therefore in this work we focus on SST and leave the analysis of CST for future work.

4 Static Rate Schemes

In this section we explore a generic token-based representation for static rate schemes. A static rate detector stipulates that the permitted rate of suspicious events remains constant throughout time. In a token-based form, this translates to as follows—each connection failure results in the removal of a constant number of tokens, and tokens are accrued at a constant rate, independent of the state of the system[3]. A number of scan detection schemes fall in this category, including NSM [1], Snort [3], Bro [4] and Chen's [9].

To give the best performance, our token-based formulation stipulates that tokens are consumed only when first-contact failures occur. A first-contact failure is the very first connection to a particular destination that resulted in a failure [6]. A connection is considered *failed* when the outgoing SYN elicits a TCP_RST or a timeout without a SYN_ACK. The concept of first contact was first used in TRW [5]. To determine whether a connection is a first-contact connection, the host must maintain statistics of previously contacted addresses. While we do not specify how these statistics should be maintained at the host level, we stipulate that there exist many efficient mechanisms (e.g, hash tables, bloom filters) which will allow a host to store and look up a list of previously seen destination addresses without incurring a high performance overhead. Prior results [6, 10] suggest that a list of 64 or more addresses render sufficiently accurate results.

The algorithm works as follows: Every time the host sends an outgoing SYN packet, the destination IP is checked against the list of maintained addresses. If the destination IP is in the list, the SYN packet is permitted through without further ado. Otherwise the connection is monitored; if the connection generates a failure, a token is removed from the

[3]There is usually a maximum token balance.

host's token pool and the destination IP is added to the list of contacted addresses. Detection occurs when the host exhausts its token pool.

If we model the timing of packets transmitted by the background traffic as a Poisson process with rate parameter μ, then the arrival of non-scan failures is also a Poisson process with rate parameter $\mu(1-p_n)$ (note p_n is the background traffic success probability). The probability for false positive for a period of time τ is therefore

$$P_{FP} = \Sigma_k \frac{e^{-\mu(1-p_n)\tau}(\mu(1-p_n)\tau)^k}{k!} \qquad \text{s.t.} \qquad k > n_0 + a\tau \tag{8}$$

where n_0 is the token balance at the beginning of the interval τ. The false negative probability is,

$$P_{FN} = e^{-\mu(1-p_n)T}(1+\mu(1-p_n)T) \qquad \text{s.t.} \qquad T = \frac{1}{a - r_s(1-p_s)} \tag{9}$$

If $T < 0$, the scan will consume all the tokens independent of the background traffic and $P_{FN} = 0$.

The expected values of the SST and `Escaped_Count` for the static rate scheme can be found using Equations 4 and 5 respectively. To put things in perspective, assume normal traffic success probability $p_n = 0.7$, scan success probability $p_s = 0.02\%$, and the expected rate of non-scan traffic μ = 0.4 (4 connections in 10 seconds). A token reward rate $a = 1$, which translates to one first-contact failure per second, would give rise to an expected value of SST as $E[SST_{static}] = 0.8802$. This means that any worm that performs at least 9 scans every 10 seconds will be detected.

In the remainder of the paper we will use this static rate scheme as a baseline for comparison and contrast it with the adaptive schemes.

5 Adaptive Rate Scan Detection

The main problem with the static rate scheme is that the permitted rate of suspicious events, is statically determined, which leaves little freedom for legitimate traffic perturbation. If one sets the token reward rate (hence the detection threshold) too low, it will result in false positives while a high rate permits a liberal amount of scans to escape the network. In this section, we investigate adaptive rate schemes. In the parlance of the token-based framework, an adaptive rate detector gives rise to a dynamically changing rate of permitted suspicious events. This is in contrast to the constant rate in a static rate scheme. As we shall see later in this section, making this rate dynamic in the manners detailed below has significant consequences to scan detection.

We propose two adaptive scan detectors, *Success Based (SB)* and *Failure Based (FB)*. SB changes the token reward rate based on the connection success characteristics of the host, while FB changes the token penalty, γ, based on the failure statistics. Both give rise to a dynamically changing permitted failure rate. We analyze and contrast them with the static rate scheme described in Section 4 and other dynamic schemes such as RHT [6] and the recent CUSUM based detector by Ganesh et al.[7] in Section 6.

5.1 Success Based (SB)

The fundamental observation behind SB is that a legitimate host (one that is not a scanner) will exhibit a greater percentage of connection successes than a scanning host. As such, SB attempts to adjust the token reward rate based on the connection success statistics of the host. The high-level strategy of SB is simple: more successful hosts are rewarded with a larger token reward rate. This approach is different from RHT [6] and the modified CUSUM detector by Ganesh et al. [7]; RHT uses only the ratio of success to failure while the CUSUM detector uses only the rate of failures. SB attempts to use both, and as a result, SB gives rise to a better overall performance than both RHT and CUSUM.

To keep track of success statistics, we use the concept of *success index*. The success index of a host after the ith first-contact connection, we call it ρ_i, is defined as follows,

$$\rho_i = \mathsf{connectionResult}_i \cdot \alpha + \rho_{i-1}(1-\alpha) \tag{10}$$

α is a smoothing factor, which we set to 0.1. ρ_0 is initialized as the percentage of successful connections within an initialization period (we use ten connections for the initialization period). Note that Equation 10 renders a weighted

Host Success Index (ρ)	Token Reward Rate (a)
$0 \leq \rho < 0.1$	$a =$ Desired SST (σ)
$0.1 \leq \rho < 0.9$	$a \approx \dfrac{\sigma(1-\rho)e^{(10\ln(\omega/\sigma)\rho)/9}}{\epsilon^{1/n} - 0.146}$
$0.9 \leq \rho \leq 1$	$a =$ Maximum-to-be-tolerated failure rate (ω)

Table 2: Success Index to Token Reward Rate Mapping for SB (ϵ: desired false positive rate)

index that slightly favors recent connection results, which is more sensitive to short-term traffic pattern changes than a straightforward success ratio. However, the success index calculation is also robust against short traffic bursts, assuming reasonable values for the smoothing factor, α (typically 0.05 to 0.2). This robustness provides a more accurate overall success index, and prevents false positives for hosts that experience failure bursts.

To put things in perspective, a normal desktop client (with web surfing and email activities) typically has a success index greater than 0.6. We observed, however, hosts involved in P2P applications tend to exhibit a wide range of values for success index, sometimes as low as 0.2. The scan traffic recorded in our traces has a success index less than 0.1.

Once we have the success index, the token reward rate a is determined in a fashion detailed in Table 2. As shown, hosts whose success indices are below 0.1 receive the lowest token reward rate, which is set to be the desired SST. For instance, $a = 0.01$ (1 token every 100 seconds) to match an SST of 1 scan per 100 seconds. Similarly, we set the largest token reward rate to the maximum, to-be-tolerated legitimate failure rate, and allocate this rate to hosts whose success indices are above 0.9. For the mid-range success indices, the token reward rate is determined by the formula shown in Table 2 where ϵ is the desired false positive rate, and n is the maximum token balance.

Table 2 reflects SB's design philosophy that benign hosts are rewarded for good behavior while the failure behavior of potentially malicious hosts are progressively restricted. It is intuitive that for hosts whose success indices are low (and therefore likely scanners), its token reward rate should approximate the Sustained Scanning Threshold. Similarly, when a host's success index is above 0.9 (and therefore likely not a scanner), its token reward rate reflects the largest, to-be-tolerated failure rate for the host, since anything lower could result in false positives. The mid-range index-to-reward-rate mapping in Table 2 was selected with the goal of maintaining a low sustained scanning threshold and low false positives. Appendix A shows the formulation of the equation in Table 2.

Table 2 provides a general guideline for setting token reward rates based on the success index. In practice, however, to avoid adjusting the token reward rate for every little change in the success index, one can set incremental values for the token reward rate based on the relationship guideline laid out in Table 2. An example is shown in Table 3 (the desired SST is 0.01, the max tolerated failure rate is 4.0, and the desired false positive rate is 2%).

Excluding the calculation of the success index and the dynamic token reward rate, SB operates exactly the same as the static rate scheme; one token is removed for each first-contact failure and only first-contact connections are considered in the calculation of the token reward rate.

Again, if we model the background traffic as a Poisson process with rate μ, the probability for false positive with SB is as follows,

$$P_{FP} = \Sigma_k \frac{e^{-\mu(1-p_n)\tau}(\mu(1-p_n)\tau)^k}{k!} \qquad \text{s.t.} \qquad k > n_0 + f(\rho)\tau \tag{11}$$

where $f(\rho)$ represents the mapping between ρ and the token reward rate as defined by Table 2. The false negative probability for SB is,

$$P_{FN} = e^{-\mu(1-p_n)T}(1 + \mu(1-p_n)T) \qquad \text{s.t.} \qquad T = \frac{1}{f(\rho) - r_s(1-p_s)} \tag{12}$$

The Sustained Scan Threshold with SB is as follows.

$$E[SST_{SB}] = \frac{f(\rho) - \mu(1-p_n)}{1-p_s} \tag{13}$$

If the background traffic has a rate parameter of $\mu = 4/10$, probability of success $p_n = 0.7$, and scan success probability $p_s = 0.02\%$, the expected SST is 0.3 scans per second.

Host Success Index	Token Reward Rate	Host Success Index	Token Reward Rate
$0 \leq \rho < 0.1$	0.01	$0.5 \leq \rho < 0.6$	0.75
$0.1 \leq \rho < 0.2$	0.10	$0.6 \leq \rho < 0.7$	1.0
$0.2 \leq \rho < 0.3$	0.25	$0.7 \leq \rho < 0.8$	2.0
$0.3 \leq \rho < 0.4$	0.40	$0.8 \leq \rho < 0.9$	3.0
$0.4 \leq \rho < 0.5$	0.50	$0.9 \leq \rho \leq 1.0$	4.0

Table 3: Example Reward Rates for SB

The expected number of scans permitted before detection is as follows.

$$E[\texttt{Escaped\_Count}_{\texttt{SB}}] \approx \frac{\texttt{E[Tokens]}}{\texttt{r}_\texttt{s}(1-\texttt{p}_\texttt{s}) + \mu(1-\texttt{p}_\texttt{n}) - \texttt{f}(\rho)} \cdot \texttt{r}_\texttt{s} \tag{14}$$

Note that a scanner can purposely inflate ρ by generating successful first-contact connections. This way the host will receive a greater token reward rate, thereby achieving a greater SST. Unlike RHT, however, the Sustained Scanning Threshold of SB increases at a substantially lower rate with the rate of successful connections. A detailed comparison and analysis of SB vs. other schemes appears in Section 6.

5.2 Failure Based (FB)

Instead of changing the token reward rate, FB adjusts the number of tokens consumed for each failure, based on the host's failure behavior. FB is an alternative way to achieve dynamic rates and has a decidedly different focus than SB; by manipulating, γ, the number of tokens removed per failure, FB is more restrictive and achieves faster detection of scanners. This, however, necessitates an increase in false positives, but the success test here is a lower false positive rate than that of a static-rate scheme with a similar Sustained Scanning Threshold (SST).

At a high level, FB's strategy is simple: FB heightens the token penalty—number of tokens removed per failure—as the host's failure rate increases and reduces it as the failure rate decreases. The token reward rate, a, remains constant in FB. We will detail how a is determined below. To determine the token penalty, FB periodically estimates the host's failure rate as follows: after each interval i, the failure rate, ϕ_i, is calculated as

$$\phi_i = \mathsf{current\_failure\_rate} \cdot \alpha + \phi_{i-1} \cdot (1-\alpha) \tag{15}$$

where current_failure_rate is the average failure rate for the current period (i) and α is the smoothing factor. We use an α of 0.25 here. ϕ_0 is the average failure rate for the first interval. We note that to reduce the computation overhead, one can use a sampling method to estimate current_failure_rate. We consider sampling an optimization and do not address this issue further in the paper.

In addition to the failure rate, FB uses a number of other quantities. They are: a) a, the constant token reward rate, is set to the maximum, to-be-tolerated failure rate (a may reflect a legitimate bursty failure rate), b) β, the maximum length of legitimate failure bursts, also the length of estimation interval for failure rates. Both a and β are system configurable parameters. We observed that for our network, failure bursts for legitimate hosts typically last fewer than 5 seconds, and therefore a five-second interval window seems appropriate.

FB adjusts the token penalty per failure, γ, based on the failure rate, ϕ, in a fashion detailed in Table 4. In Table 4, for each range of failure rate, the interval-until-depletion number specifies the desired number of intervals until the complete depletion of tokens, assuming the failure rate remains in the same range. These numbers should be system-specific parameters, which are set based on traffic characteristics and the target false negative and false positive probability. More specifically,

- If the failure rate of the host, ϕ, is less than $0.2a$, the host is considered normal and FB sets the token penalty, γ, to 1 to allow the maximum permitted failure rate a (note that the permitted failure rate for the host is a/γ). For these values, the interval-until-depletion is infinite.
- Hosts whose failure rates are within $[0.2a, 0.4a]$ should primarily be legitimate hosts. However, we increase the penalty to 2 to allow only 50% of the maximum permitted failure rate. This would shorten the detection time should the failure rate continues to increase (due to a scan).

Estimated Failure Rate (ϕ)	Desired intervals until token depletion	Penalty γ(# tokens)
$0 \leq \phi < 0.2a$	∞	1
$0.2a \leq \phi < 0.4a$	∞	2
$0.4a \leq \phi < 0.6a$	x_1	$\left(\frac{n}{x_1\beta} + a\right) 1/\phi$
$0.6a \leq \phi < 0.8a$	x_2	$\left(\frac{n}{x_2\beta} + a\right) 1/\phi$
$\phi \geq 0.8a$	1	$a\beta$

Table 4: Penalties for FB Detection

- If ϕ is close to a ($\phi \geq 0.8a$), the host has been generating failures close to the maximum rate for more than one interval (see Equation 15). Continuned failures at this rate are well outside what is considered acceptable. The token penalty is therefore set to $a\beta$ to ensure that all tokens will be depleted before the end of the next interval.
- For hosts whose failure rate is within $[0.4a,\ 0.8a]$, one should first set the desired number of intervals until depletion; they are system-specific parameters. In Table 4, x_1 denotes the interval-until-depletion for the range of $[0.4a \leq \phi \leq 0.6a]$ and x_2 for $[0.6a \leq \phi \leq 0.8a]$. x_1 and x_2 should be progressively smaller. The token penalty, γ, can be calculated subsequently using the formula in Table 4. Such a penalty will give rise to the desired number of interval-until-depletion, if the failure rate remains within the same range . In our network, for instance, we set x_1 and x_2 to 4 and 2, respectively. The corresponding token penalties are therefore 3 and 4.

When implementing FB in practice, one can adjust the desired intervals-until-depletion value for each failure range and even adjust the granularity of the failure range based on the desired properties of the detector.

Assuming the background traffic as a Poisson process with a rate parameter μ, we can calculate the false positive probability for FB as,

$$P_{FP} = \Sigma_j \left(\frac{e^{-\mu(1-p_n)\tau}(\mu(1-p_n)\tau)^j}{j!} \right) \qquad \text{s.t.} \qquad j \geq \frac{n_0 + a\tau}{g(\phi)} \tag{16}$$

where n_0 is the token balance at the beginning of time period τ, a is the token reward rate (in this case remains constant), $g(\cdot)$ is the function defined by Table 4 such that $\gamma \leftarrow g(\phi)$.

The probability of false negative is,

$$P_{FN} = e^{-\mu(1-p_n)T}(1 + \mu(1-p_n)T) \qquad \text{s.t.} \qquad T = \frac{1}{a/g(\phi) - r_s(1-p_s)} \tag{17}$$

where r_s is the scan rate and p_s is the scan success probability.

The SST for FB is,

$$E[SST_{FB}] = \left(\frac{a}{g(\mu(1-P_n) + r_s(1-p_s))} - \mu(1-p_n) \right) / (1-p_s) \tag{18}$$

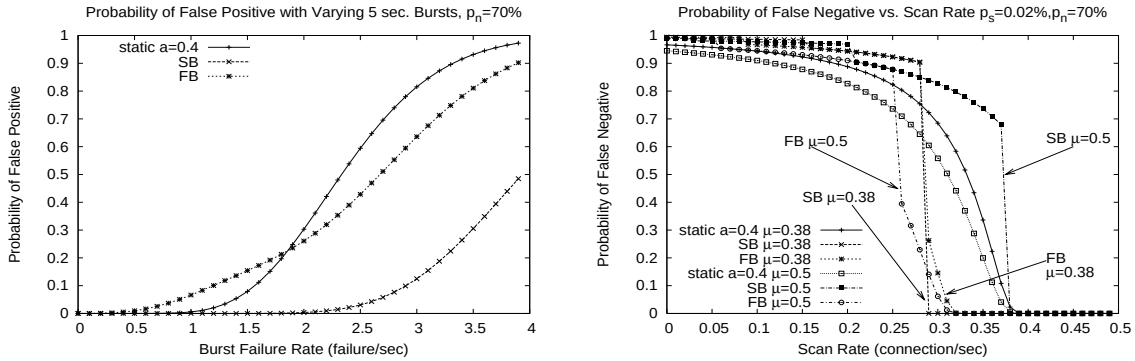
where $\gamma \leftarrow g(\mu(1-p_n) + r_s(1-p_s))$. For any scan with a rate greater than the SST, the expected number of scans that occur before detection is

$$E[\texttt{Escaped\_Count}_{FB}] = \frac{E[Tokens]}{g(\phi)(r_s(1-p_s) + \mu(1-p_n)) - a} \cdot r_s \tag{19}$$

To put things in context, let's consider again the example for which the background traffic rate of the host is 4 connections every ten seconds ($\mu = 0.4$) and its success probability is $p_n = 70\%$. If the scan success probability is $p_s = 0.02\%$ and the token reward rate, a, is 1, then the SST for the host is 0.28 scans per second. A spreading worm that exhibits a connection rate as low as this will see an extremely slow propagation. In contrast, random scanning worms to date typically have scan rates in the neighborhood of tens of scans per second.

FB has an added benefit in that it is difficult, from the scanner's point of view, to game the detector; the only way to evade detection with FB is to reduce the failure rate of the host. Hit-list worms can possibly evade FB. For scanning worms, however, reducing the failure rate necessitates a reduction in the scan rate, which leads to a slower propagation. A more in-depth analysis of FB and comparison with other schemes can be found in Section 6.

5.3 Probability of Error



(a) Prob. of False Positive with varying background failure bursts (τ=5s)

(b) Prob. of False Negative with varying background traffic

Figure 1: Error Probabilities for SB, FB and Static Schemes

Figures 1(a) and (b) show the probability of false positives and false negatives for SB, FB and the static rate scheme discussed in Section 4. Figure 1(a) shows specifically the false positive probability against 5-second traffic bursts. The parameters of the schemes plotted here are selected such that static and FB would have the same SST's.

A number of observations are significant here. First, when the SST's are similar, in Figure 1(a) the static rate scheme exhibits a sharper curve than the others as the burst failure rate increases. In other words, the static scheme is less robust against background failure bursts and therefore more likely to generate false positives. Second, SB exhibits the least chance of false positive. This is not surprising since one of the design goals for SB was the accommodation of such bursts for hosts with a large success index. In our experience, legitimate hosts can generate short bursts with up to 3 or 4 failures/second. The parameter setting plotted in Figure 1(a) stipulates that the static-rate scheme will likely generate false positives with at least 40% probability during such bursts. To remedy this, we can increase the token reward rate, but doing so would increase the Sustained Scanning Threshold (SST). SB, on the other hand, has a visibly lower false positive rate and is in general less sensitive to changes in the background traffic rate.

The false negative plot in Figure 1(b) shows the detection capabilities of the various schemes with respect to scan rates and two different background traffic rates $\mu = 0.38$, and $\mu = 0.5$. We opt for low background traffic rates to better respresent average long term traffic characteristics. When $\mu = 0.38$, the three detectors have approximately the same SST (0.3scans/second). Not surprisingly, both SB and FB have a sharper decreasing false negative probability than the static scheme. This is intentional, since both aim for fast detection of scanners beyond their SSTs. When the background traffic rate, μ, increases, SB's SST pulls away from the others since its success index will likely be higher due to more background successes. We will address the SST inflation issue of SB in Section 6.2.

Overall, FB provides fast detection of scanners but is susceptible to false positives when bursty failures occur, and hence is more appropriate for a controlled environment with well-understood traffic characteristics. SB, on the other hand, is the most robust against background perturbation and is also likely to detect scanners quickly. It would work well in the context of an open network with a wide variety of different applications that may have diverse traffic characteristics.

6 Analysis

In this section we present a detailed analysis of SB and FB, comparing against the static scheme and other adaptive detectors. We show that, both analytically and empirically, SB and FB are capable of rendering lower SSTs than the others while maintaining comparable or better detection precisions.

6.1 Other adaptive detectors

Two scan detectors, *Reverse Hypothesis Test* (RHT) [6] and the CUSUM detector [7], are of particular interest to this work because both fall in the category of adaptive scan detection. In this section, we contrast the performance of FB and SB with these two schemes. Due to space constraints, we give only a brief description of RHT and CUSUM. Interested readers should refer to the original papers for more details [6, 7].

RHT is a random walk based detector that operates in the range of real numbers. The position of the walk is increased by a pre-determined factor for each first-contact failure and decreased also by a pre-determined factor for a first-contact success. If the random walk reaches a state greater than a certain threshold, the algorithm terminates and reports that the host is a scanner. Translating RHT into a token-based form entails taking $\log$ of the step function values as the subtraction and addition operation of tokens. More specifically, if we use the original parameter setting as defined in [6], the token representation of RHT calls for the removal of one token for every first-contact failure and the addition of 1.77 tokens for each first-contact success. The expected value of token reward rate for RHT is therefore, $E[a] = 1.77(r_s p_s + \mu p_n)$, and the expected SST and `Escaped_Count` for RHT are as follows, using the expected token reward rate.

$$E[SST_{RHT}] = \frac{\mu(2.77p_n - 1)}{1 - 2.77p_s} \tag{20}$$

$$E[\texttt{Escaped\_Count}_{\texttt{RHT}}] = \frac{\texttt{E[Tokens]}}{\texttt{r}_\texttt{s}(1 - 2.77\texttt{p}_\texttt{s}) + \mu(1 - 2.77\texttt{p}_\texttt{n})} \cdot \texttt{r}_\texttt{s} \tag{21}$$

Again, let's use the same example for which the non-scan traffic rate, μ, is 4 connections every ten seconds, success probability, p_n, is 70%, and scan success probability, p_s, is 0.02%. For RHT, one can expect that a scanner can perform at most 0.375 scans/second without being detected. As presented earlier in the paper, this SST is smaller than that of the static-rate but larger than those of SB and FB, under the same traffic conditions.

RHT is susceptible to gaming in that a scanner can generate successful cover traffic to accrue more tokens. If the cover traffic is able to generate more tokens than the scan traffic consumes, detection will not occur. Because in RHT each successful first-contact connection adds 1.77 tokens and each failure only consumes one token, the permitted number of failures grows linearly with the amount of background traffic, assuming a constant failure probability for background traffic.

The CUSUM detector by Ganesh et al. [7] is analogous to FB in that both are only concerned with failure characteristics. CUSUM assumes that the detector knows the failure rate of both non-scan and scan traffic, which they call λ_0 and λ_1, respectively. One can also estimate the scan failure rate, λ_1, as follows,

$$\lambda_1 = \lambda_0 \exp\left(\frac{p_s n}{1 - 2p_s}\right) \tag{22}$$

where p_s is the scan success probability and n is the initial and also maximum token balance of a host. Estimating the scan failure rate in such a manner is called the *suboptimal* method. When failure patterns of both legitimate and scan traffic are known and remain constant, CUSUM provides the optimal detector [7], where optimality means the shortest detection time, given a certain false positive upperbound.

Central to CUSUM is the concept of inter-failure time; the CUSUM detector rewards more tokens when the inter-failure time is large, and the opposite occurs when the inter-failure time is small. Equation 23 describes the net change of tokens per failure for a given inter-failure time, τ.

$$\Delta_{tokens} = (\lambda_1 - \lambda_0)\tau - ln(\lambda_1/\lambda_0) \tag{23}$$

Equations 24 and 25 provide the expected SST and `Escaped_Count` respectively.

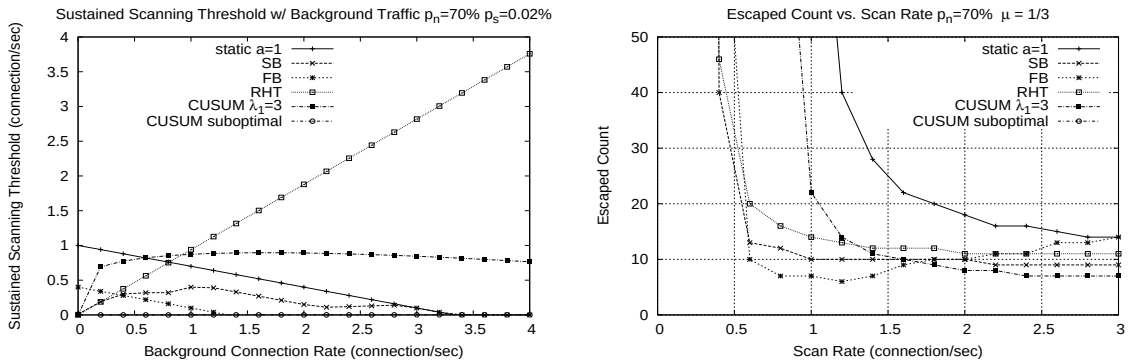
$$E[SST_{CUSUM}] = \left(\frac{\lambda_1 - \lambda_0}{\log(\lambda_1/\lambda_0)} - \mu(1 - p_n)\right) / (1 - p_s) \tag{24}$$

$$E[\texttt{Escaped\_Count}_{\texttt{CUSUM}}] = \frac{\texttt{E[Tokens]}}{\log(\lambda_1/\lambda_0)(\texttt{rs}(1 - \texttt{ps}) + \mu(1 - \texttt{p}_\texttt{n})) - (\lambda_1 - \lambda_0)} \tag{25}$$

If we assume the non-scan failure rate λ_0 is 1 failure/second and we estimate the scan rate λ_1 using Equation 22 such that $\lambda_1 = 1.003$. The same example where $\mu = 0.4$, $p_n = 70\%$, and $p_s = 0.02\%$ renders an SST of 0.882 scans per second. Note that Equation 23 dictates that the larger the difference between λ_1 and λ_0, the larger the number of tokens removed per failure, which results in faster detection. However, raising λ_1 will lead to a higher SST. As such, there is a trade-off between SST and detection speed for CUSUM as λ_1 varies.

6.2 Sustained Scanning Threshold Analysis

The Sustained Scanning Threshold (SST) is an important metric of success for a scan detector. Figure 2(a) shows the expected SST for all the schemes discussed thus far with varying background traffic rate, μ. In this plot, we use a non-scan success probability $p_n = 70\%$ and a token reward rate of $a = 1$ for both static-rate and FB. For CUSUM, we plotted two configuration settings: in one we estimate the scan failure rate λ_1 using the suboptimal method of Equation 22. In the other setting, λ_1 is set to 3. For both configurations the expected background failure rate, λ_0, is set to match the background failure rate. These settings are to demonstrate the tradeoff between SST and detection time for the CUSUM detector.



(a) Expected SST against background traffic rate ($p_n = 0.7, p_s = 0.02\%$)

(b) Expected `Escaped_Count` before detection

Figure 2: Sustained Scanning Thresholds and Escaped Counts

As shown in Figure 2, compared to the other schemes, both SB and FB render a substantially lower Sustained Scanning Threshold (SST) for a wide range of background traffic rates. The CUSUM suboptimal configuration yielded a near zero SST. However, it's corresponding `Escaped_Count` in Figure 2(b) is substantially larger than the others; in fact, the `Escaped_Count` for the suboptimal CUSUM was magnitudes larger, and is off the scale and therefore not visible in Figure 2(b). It is important to note that with the scan probability of success, p_s, used in calculation the basic reproduction number for the infected hosts would be slightly less than one. On average another host would not be infected. However, this long delay still permits a large number of scans to escape the network before the host is detected. As such, the CUSUM suboptimal detector is not a good choice in practice. The other CUSUM detector, when λ_1 is 3, a value larger than the suboptimal calculation, fared better in that its `Escaped_Count` is comparable to the other schemes once the scan rate surpasses the SST. With a larger difference between λ_0 and λ_1 the number of tokens consumed per failure, γ, is larger and less scans are permitted before detection.

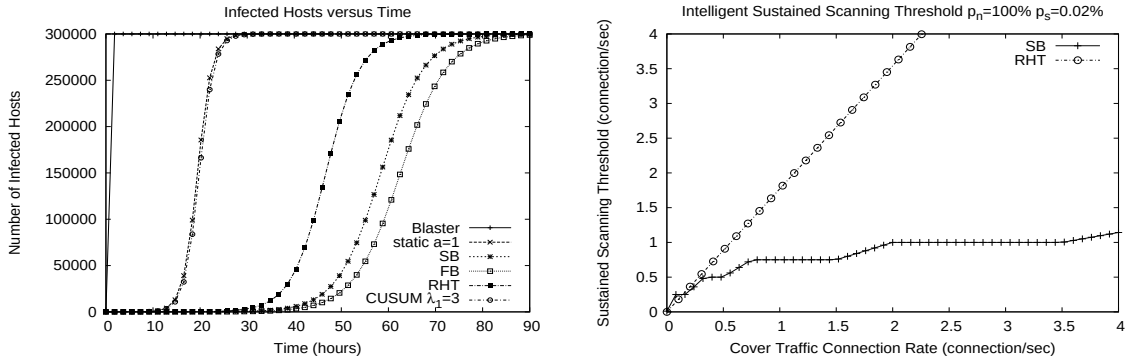
RHT yields a SST that is low for hosts whose traffic rates are below 0.5 connections/second, but goes up significantly when the background traffic rate increases. In our experience, normal desktop machines that run web and email clients tend to initiate connections in the neighborhood of 0.5 connections/second. These hosts are well suited for SB, FB, or RHT. For more active hosts whose traffic rates are over 0.5 connections/second, SB and FB are a better choice if slow scanners are concerned. Note that SB has consistently low SSTs across the different values of μ; as such, SB is better suited for hosts with diverse traffic patterns.

Assuming a constant background success probability, p_n, and a constant scan success probability, p_s, the static rate and FB schemes give rise to an expected SST value that is inversely proportional to the background traffic rate. This is because as the background traffic rate grows, the background failure rate also increases, which leaves less tokens

for scan traffic, thereby reducing the SST. For the CUSUM detector with a constant λ_1, as the background failure rate, λ_0, increases, the number of tokens rewarded first increases and then decreases as the difference between the two λs change. For RHT the expected value of SST grows linearly with the rate of the background traffic (as indicated by Equation 20). For SB, this increase is slower and it tapers off as the background traffic rate hits 1 connection/second. This disparity of behavior between RHT and SB is important. For RHT, it would require a significant amount of time to detect a scanner on a host with a near 100% success rate history (to drive down the ratio of success to failure). With SB, however, the way success indices are calculated allows consecutive failures to quickly reduce the token balance towards detection.

Figure 2(b) shows the expected `Escaped_Count` for each scheme with varying scan rates. For this plot, we assume a constant background traffic rate of 1/3 connection/second and success probability of 70%. For each scheme, when the scan rate surpasses its SST (where the `Escaped_Count` approaches infinity), the number of permitted scans decreases exponentially. Note that SB and FB exhibit a faster decrease in `Escaped_Count` than the other schemes. This fact is significant; one of the fundamental differences between our schemes and the others is that the former (both SB and FB) has a non-linear relationship between the traffic failure rate and the token balance; In SB and FB, the token balance decreases near exponentially to the increase of the connection failure rate when the failure rate is near the SSTs. As such, these schemes provide faster detection than others.

A graphical depiction of the epidemic growth of worms at the cusp of SST is shown in Figure 3(a). This plot includes the estimated, "untampered" growth of Blaster as a baseline comparison to the epidemic growth when scan detectors are in place. The points in this graph are generated using a constant μ of 0.4 connection/second, p_n of 70%, and the scheme configurations mentioned in the individual sections. As shown, compared with other detectors, SB and FB can deter worm growth significantly—reaching 15% of the susceptible population with SB or FB requires more than twice the amount of time as with CUSUM or the static rate detector.



(a) Epidemic Growth of Random Scanning Worms that Evade Detection ($p_s = 0.02\%, p_n = 0.7, \mu = 4/10$)

(b) Effect of Cover Traffic on SST $p_s = 0.02\%$

Figure 3: Permitted Worm Spread and Effect of Cover Traffic on the Schemes

Gaming of SST. A scanner can game RHT and SB by adding successful cover traffic. To be effective, the cover traffic needs to have a high success probability. Figure 3(b) plots the SSTs for SB and RHT versus the rate of cover traffic. For simplicity, we assume that all non-scan traffic is cover traffic that succeeds with 100% probability (i.e., p_n=1). As shown, SB's SST increases at a much slower rate that that of RHT's.

According to Equation 20, RHT's SST increases linearly to the cover traffic rate, μ, assuming p_n and p_s are constants. In contrast, SB's SST is upperbounded by $\frac{\omega}{1-p_s}$ where ω is the max to-be-tolerated failure rate (see Equation 13 when p_n is 1). As a result, the growth rate of SST for RHT is faster than that of SB. We note that it is possible to change the configuration of RHT such that its SST would increase at a lesser rate than that is plotted in Figure 3(b), but we reiterate that this will not change the fundamentally linear relationship between its SST to the traffic rate.

6.3 Error Rates

In this section, we present an empirical analysis of the various detectors, using both real and simulated network traffic.

Trace Data. The real traffic traces used in this study are collected at the boundary of a 1200-host network, primarily used for academic computing purposes. The network serves approximately 1500 users and has a variety of operating systems and applications. The traces include the headers of all inbound and outbound TCP packets entering and exiting the network. During the course of tracing, we recorded *Blaster* [14] and *Welchia* [15], both scanning worms that infected local hosts in our network.

For each attack recorded, we conducted post-mortem analysis to identify the set of infected hosts within the network. We further identified port-135 TCP_SYNs as outbound blaster or welchia scans from these infected hosts. It is important to note that hosts within our network were exclusively Windows clients that rarely (if ever) made any outbound port-135 connections. Once infected, however, the hosts made tens of thousands of port-135 connections. As such, the task of identifying Blaster and Welchia scans is fairly straightforward.

We use four traffic traces in this study. Trace I is a 25-day outbound trace, from August 6th to August 30th, 2003. This period contains the onset of Blaster infection in our network, which occurred on August 11th. We identified 103 Blaster infected hosts internally. We performed due-deligence to the best of our ability to ensure that in Trace I there is no outbound scans originated from our network other than those due to Blaster. Trace II is a 10-day outbound trace, from August 9th to the 18th, 2005, which contains no internal worm infection in so far as we can tell. We use Trace II to study false positives.

Traces III and IV are synthetic traffic that include simulated slow scans and traffic bursts. In both traces there are 1100 hosts, 100 of which perform scans at a rate of 1 every two seconds. 75% of the hosts generate background traffic that succeeds 70% of the time and follows a Poisson distribution with a rate of 0.1. The other 25% generate traffic that succeeds with a 30% probability and follows a Poisson distribution with a mean of 0.05. This mix of traffic is a simplification of the real traffic seen in our network; the majority of hosts make mostly successful connections at a higher rate while the other hosts tend to generate a slow stream of failures when contacting new destinations (recall we only account for first-contact failures). In both Trace III and IV the more active hosts (75% of hosts) have five-second traffic bursts. These bursts occur uniformly with 1% probability for each interval (the same interval for failure rate estimation) Trace III has relatively slow bursts at the rate of 0.5 connections/second while Trace IV has faster bursts at the rate of 2 connections/second to simulate traffic during business hours.

Simulation Results. The implementation of the various algorithms, in their token-based form, is straightforward; each host is allocated n tokens originally and our implementation keeps track of first-contact failures using a Previously-Contacted-History, much like what is done in RHT [6]. For the static-rate detector, we set the token reward rate, a, to 1 token per second, and each first-contact failure consumes a token. For SB, a is set based on the success index (see Table 2). For FB, we estimate the failure rate of the host periodically and use the value in Table 4 to determine the token penalties for each first-contact failure henceforth. For RHT and CUSUM, several configurations were simulated. In particular, one of the major sources of false positives for RHT is the somewhat archaic `ident` protocol. Since `ident` is rarely used and the servers running `ident` can be easily whitelisted, we present two sets of results for RHT, one includes `ident` servers and the other has them white-listed.

The CUSUM detector requires the detector to estimate the non-scan failure rate, λ_0, and the scan failure rate, λ_1. In our experiments, we set λ_0, to match the permitted failure rate/token reward rate of the static-rate scheme. We used two values for the scan failure rate, λ_1. The first one was 10 failures/second to match the observed Blaster failure rate[4]. This approximates the optimal situation when the worm scanning rate was known. The second one is using the "suboptimal" estimate (see Equation 22) when the scan rate is unknown. We used a scan success probability, p_s, of 0.02% for the suboptimal estimate.

Table 5 shows the results of our experiments on Trace I and II. We remind the readers that the false negative rate is defined as the portion of scanners that elude detection, while the false positive rate is the portion of legitimate hosts incorrectly labeled as scanners. The mean `Escaped_Count` is the average number of scans from infected hosts before the scanner is detected (averaged over all hosts infected with Blaster). The last column of the table lists the number of hosts running P2P clients identified as scanners by each detector. Peer-to-peer applications tend to fall outside the

[4]Our traces of Blaster and Welchia indicate that infected hosts generated failures at approximately 10 scans per second, with a success probability of 0.02%.

Detection Scheme	Parameter Settings	False Negative Rate	Mean Escaped Count	False Positive Rate		P2P Clients Detected	
Trace		I	I	I	II	I	II
Static Rate	$a = 1$	0%	13.09	0.35%	0.71%	17	5
RHT	normal	0%	10.194	27.216%	4.876%	36	6
	no Ident	0%	10.194	2.216%	4.876%	36	6
CUSUM	$\lambda_1 = 10$ (near optimal)	0%	13.356	0.62%	0.17%	9	5
$\lambda_0 = 1$	suboptimal λ_1	26.2%	8366.578	0%	0%	0	0
SB	$\sigma = 0.01, \omega = 4$	0%	10.35	0.798%	0.355%	32	7
FB	$a = 1, \beta = 5$ $x_1 = 4, x_2 = 2$	0%	10.12	0.62%	1.24%	32	9

Table 5: Experimental Results for Traces I & II

Detection Scheme	Parameter Settings	False Positive Rate		Mean Escaped Count	
Trace		III	IV	III	IV
Static Rate	a = 0.4	0%	48.5%	34.0	32.3
RHT	no ident	21.17%	23.1%	8.98	8.93
CUSUM	$\lambda_0 = 0.4, \lambda_1 = 0.5$	0%	0%	423	375.6
	optimal	1.5%	71.7%	8.21	8.07
SB	$\sigma = 0.01, \omega = 4$	1.4%	8.8%	13.53	13.39
FB	$a = 1, \beta = 5$ $x_1 = 4, x_2 = 2$	0.2%	29.6%	15.11	14.67

Table 6: Simulation Results for Traces III & IV with an inserted Scanning Worm ($r_s = 0.5$)

critical network operation, and are therefore often considered non-desirable activities. For this reason, we single out P2P hosts as a separate category.

As the results show, all but the "suboptimal" CUSUM were able to detect all the scanners. Most schemes rendered a similar detection efficiency, as indicated by the mean `Escaped_Count` . This is primarily due to the fact that Blaster is a fast scanning worm whose behavior is clearly different from normal traffic. In fact, in our experiments the Blaster traffic managed to deplete all the tokens before a single new token was awarded. For this reason, the Blaster trace is not ideal for studying detection efficiency. Rather, the experiments with Trace III and IV which include slower scanners have more descriptive results.

Note that CUSUM performs well when their estimated failure rates λ_0 and λ_1 are close to the real failure rates but otherwise gives poor results. CUSUM's "suboptimal" detector missed over 25% of the scanners in Trace I. The suboptimal estimation method, shown in Equation 22, renders a λ_1 that is far from the real scan rate of Blaster. This also contributed to the exceptionally large `Escaped_Count` . The CUSUM work is of theoretical interest because it shows that there exist an optimal detector if scan and traffic rates are known. In practice, however, these assumptions often do not hold.

Overall SB and FB performed well with Trace I and II. In addition, they yielded a lower false positive rate than RHT (even when the `ident` hosts are white-listed). This is because RHT only takes into account the accumulative number of failures rather than the failure rate. Consequently, RHT can't handle legitimate hosts that experience persistent but slow failures. In contrast, SB and FB consider both failure numbers and rates.

Table 6 shows the simulation results for Trace III and IV. Recall that these two traces include the simulated slow scanners that scan at a rate of 1 per every 2 seconds. To ensure that the scan rate is above the Sustained Scanning Thresholds (SSTs)—so that scans can be detected—a few parameters have to be tweaked. The new parameter settings are listed in Table 6. In particular, the token reward rate for static and suboptimal CUSUM is changed so that they render approximately the same SST's as the other schemes. We also included an optimal configuration of CUSUM where the background failure rate is $\mu(1 - p_n)$ and the mean scan failure rate is $r_s(1 - p_s)$.

For this set of simulations, all scanners are detected so we skip the false negative column. Trace III and IV include traffic bursts the purpose of which is to show the impact of background traffic perturbation . Recall that Trace III has near constant traffic with slower bursts, while Trace IV includes larger bursts which are representative of work day network traffic.

As shown, the static-rate scheme performed poorly when the background traffic becomes bursty—yielding a 48.5% false positive rate for the burstier traffic. This is consistent with our assessment in Section 5.3 where a sensitivity analysis showed that static-rate schemes are more sensitive to traffic perturbation. In addition, the optimal CUSUM detector also fared badly with Trace IV; it detected many bursty yet benign hosts (71.7% false positive rate). This is because the CUSUM detector relies on the assumption that the failure rates (both background and scan) remain constant, and therefore cannot accommodate changes due to traffic perturbations.

SB and FB performed well in the presence of bursty traffic. SB in particular maintained a low false positive rate and a low escaped count throughout. FB showed a larger increase in the false positive rate when traffic becomes burstier, but is still on par with RHT. When traffic is predictable (Trace III in Table 6), SB and FB's detection speed is only slightly worse than optimal (as indicated by the optimal CUSUM detector), and their false positive rates are extremely low. These results are consistent with the analytical models in Section 5.3.

7 Summary

In this paper, we presented two adaptive scan detectors, SB and FB, for the detection of local scanners. In the token-based framework, the success based scheme, SB, regulates the token reward rate in an effort to differentiate legitimate and malicious behaviors. FB, on the other hand, adjusts the token penalty based on the failure behavior of the host.

We demonstrate that SB and FB can quickly detect slow scanners without incurring notably more detection errors than other detectors. More importantly, we show that SB and FB are less susceptible to intelligent gaming and robust against traffic variations, which is particularly significant when slow scanners are concerned. We show that previously proposed scan detection schemes failed to achieve similar goals; the static rate scheme and the CUSUM detector have low error rates, but can only guarantee detection of relatively fast scanners. RHT can detect slow scanners, but is vulnerable to traffic manipulation.

The interesting properties of the adaptive detectors arise from their non-linear nature. This is in contrast to the previously proposed, largely linear detectors. One direct consequence of the non-linearity is that it permits the detector to correct its course continuously with respect to the detection hypothesis, in ways that have not been explored previously. Traditionally, detectors label a host scanner if the rate or the number of suspicious events exceeds a pre-determined threshold. The adaptive algorithms, in contrast, allows the detector to change its state based on both long-term and short-term statistics. This gives rise to faster detection and decreased sensitivity to short-term traffic perturbation. Another interesting property is the susceptibility to traffic manipulation. Although SB utilizes connection success statistics in ways that are similar to RHT, the growth of its SST is however sub-linear and consequently less susceptible to intelligent gaming.

The schemes discussed thus far utilize exclusively connection failure patterns to differentiate scanners from legitimate hosts. This is a reasonable assumption for scanning worms. Hit-list and topological worms, however, have different traffic patterns. Modern IDSes that acknowledges TCP_SYNs regardless if the destination address is allocated can also hamper the effectiveness of failure-based detectors. To detect spreading malware that do not necessarily embody similar failure patterns, a different class of defense would be needed. This would necessitate the use of a different type of suspicious event, would not however, in itself limit the efficacy of the adaptive schemes presented here.

References

[1] Heberlein, L.T., Dias, G.V., Levitt, K.N., Mukherjee, B., Wood, J., Wolber, D.: A network security monitor. In: Proc. IEEE Symposium on Research in Security and Privacy. (1990) 296–304

[2] Weaver, N., Staniford, S., Paxson, V.: Very fast containment of scanning worms. In: Proceedings of the 13th USENIX Security Symposium. (2004)

[3] Roesch, M.: Snort: Lightweight intrusion detection for networks. In: Proceedings of the 13th Conference on Computer and Communication Security (LISA-99). (1999) 229–238

[4] Paxson, V.: Bro: a system for detecting network intruders in real-time. Computer Networks **31(23-24)** (1999) 2435–2463

[5] Jung, J., Paxon, V., Berger, A.W., Balakrishman, H.: Fast portscan detection using sequential hypothesis testing. In: Proceedings of 2004 IEEE Symposium on Security and Privacy. (2004)

[6] Schechter, S., Jung, J., Berger, A.W.: Fast detection of scanning worm infections. In: Recent Advances In Intrusion Detection (RAID) 2004, France (2004)

[7] Ganesh, A., Gunawardena, D., Key, P., Massoulie, L., Scott, J.: Efficient quarantining of scanning worms: Optimal detection and coordination. In: Proceedings of IEEE INFOCOM 2006. (2006)

[8] Williamson, M.: Throttling viruses: Restricting propagation to defeat malicious mobile code. In: Proceedings of the 18th Annual Computer Security Applications Conference, Las Vegas, Nevada (2002)

[9] Chen, S., Tang, Y.: Slowing down internet worms. In: Proceedings of 24th International Conference on Distributed Computing Systems, Tokyo, Japan (2004)

[10] Wong, C., Bielski, S., Studer, A., Wang, C.: Empirical analysis of rate limiting mechanisms. In: Recent Advances In Intrusion Detection (RAID) 2005, Seattle (2005)

[11] Singh, S., Estan, C., Varghese, G., Savage, S.: Automated worm fingerprinting. Proceedings of the 6th ACM/USENIX Symposium on Operating System Design and Implementation (2004)

[12] Kim, H., Karp, B.: Autograph: Toward automated, distributed worm signature detection. In: Proceedings of the 13th USENIX Security Symposium, San Diego, California, USA (2004)

[13] Newsome, J., Karp, B., Song, D.: Polygraph: Automatic signature generation for polymorphic worms. In: Proc. IEEE Symposium on Research in Security and Privacy. (2005)

[14] Symantec: W32.Blaster.Worm. http://securityresponse.symantec.com /avcenter/venc/data/ w32.blaster.worm.html (2003)

[15] Symantec: W32.Welchia.Worm. http://securityresponse.symantec.com /avcenter/venc/data/ w32.welchia.worm.html (2003)

A Success Based Reward Rate Calculation

The index-to-reward-rate mapping for the Success Based scheme was designed such that a fixed error rate was achieved for a given token balance, success index, and connection rate.

If we assume that the background failures of the host follow a Poisson distribution with a rate parameter $\mu(1-p_n)$, Equation 26 provides the approximate calculation of the false positive probability, ϵ (this equation gives the probability that the token balance decreases for n connsecutive intervals, where n is the initial and maximum token balance.)

$$\left(1 - e^{-\mu(1-p_n)/a}\left(1 + \frac{\mu(1-p_n)}{a}\right)\right)^n = \epsilon \tag{26}$$

We make a further assumption that the connection rate of the background traffic for a legitimate host follows an exponential growth curve with respect to its success index such that $\mu = c_1 e^{c_2 \rho}$, where c_1 and c_2 are constants (this assumption stipulates that any host whose connection rate exhibits a faster than exponential growth relative to its success index is likely a scanner). Using the lower and upper bound of the token reward rate, we can calculate the expression of μ with respect to the success index ρ. We can then use linear approximation to solve Equation 26 for a in terms of ρ, σ, and ω, the solution is,

$$a \approx \frac{\sigma(1-\rho)e^{(10\ln(\omega/\sigma)\rho)/9}}{\epsilon^{1/n} - 0.146} \tag{27}$$

This provides the formula in Table 2.