An Automatic Portscan Detection System with Adaptive Threshold Setting

Sang Kon Kim, Seung Ho Lee, and Seung Woo Seo

Abstract: **For the purpose of compromising hosts, attackers including infected hosts initially perform a portscan using IP addresses in order to find vulnerable hosts. Considerable research related to portscan detection has been done and many algorithms have been proposed and implemented in the network intrusion detection system (NIDS). In order to distinguish portscanners from remote hosts, most portscan detection algorithms use a fixed threshold that is manually managed by the network manager. Because the threshold is a constant, even though the network environment or the characteristics of traffic can change, many false positives and false negatives are generated by NIDS. This reduces the efficiency of NIDS and imposes a high processing burden on a network management system (NMS).**

In this paper, in order to address this problem, we propose an automatic portscan detection system using an fast increase slow decrease (FISD) scheme, that will automatically and adaptively set the threshold based on statistical data for traffic during prior time periods. In particular, we focus on reducing false positives rather than false negatives, while the threshold is adaptively set within a range between minimum and maximum values. We also propose a new portscan detection algorithm, rate of increase in the number of failed connection request (RINF), which is much more suitable for our system and shows better performance than other existing algorithms. In terms of the implementation, we compare our scheme with other two simple threshold estimation methods for an adaptive threshold setting scheme. Also, we compare our detection algorithm with other three existing approaches for portscan detection using a real traffic trace. In summary, we show that FISD results in less false positives than other schemes and RINF can fast and accurately detect portscanners. We also show that the proposed system, including our scheme and algorithm, provides good performance in terms of the rate of false positives.

Index Terms: **Adaptive threshold setting, automatic portscan detection, false negative, false positive.**

I. INTRODUCTION

With the global expansion of the Internet, various kinds of malicious code have emerged and the damage caused by network attacks has dramatically increased [1]. The most harmful network attacks are the fast, self-propagating malicious code known as worms [2], [3], because they are capable of propagating worldwide very quickly within a short period of time and causing enormous damage to global networks [4], [5]. Malicious code exploits vulnerabilities of transmission control protocol (TCP) or user datagram protocol (UDP) service ports to compromise victim hosts. If they are compromised, then the malicious code is installed [5], [6]. Malicious code begins with an attacker's reconnaissance phase, in which a large set of IP addresses is probed to find vulnerable new victims. Because this scanning uses specific TCP or UDP ports on target hosts, it is now called a portscan. If the target host replies to the probe packet, the attacker attempts to compromise and infect the target host. Thus, portscan detection is becoming increasingly important in order to effectively contain those malicious codes at the early stage of their propagation.

Most portscan detection algorithms attempt to detect abnormal behavior of portscanners by using the anomaly detection method, which detects an unknown attack, instead of the misuse detection method, which detects a known attack. Those are based on a threshold in order to distinguish portscanners from remote hosts. This threshold is usually managed manually by the network manager in the network intrusion detection systems (NIDSs). However, if the threshold is fixed, many false positives and false negatives are generated in the real network which reduces the efficiency of NIDS. This is the reason why there is presently an increasing need for an automatic portscan detection system that is able to adaptively set the threshold.

In reality, there are a number of difficulties when we attempt to design an automatic system for portscan detection. The first difficulty is that there is no true information about any remote host. The system needs to be periodically operated and must adaptively set the threshold depending on statistical data for traffic during prior time periods. In order to obtain statistical data about portscanners or normal hosts, we must know which remote host is a normal host and which one is a portscanner. But it is impossible to know whether a remote host is a portscanner or not in real time because we do not have enough information about all of remote hosts. Therefore, we can not obtain statistical data about true portscanners or true normal hosts from the traffic during the time period. This is the fundamental and well known problem in designing an automatic portscan detection system.

The second difficulty is that there is no clear definition of a portscanner. Due to the first difficulty mentioned, we must decide whether a remote host is a portscanner or not, based on the inbound traffic destined to the monitored network during a time period. Because we can only monitor a part of the connection request packets that are sent from a remote host, it is very hard to accurately estimate the entire activity. In addition, some remote hosts show ambiguous portscan activity, such that their connection request packets are sent to a few IP addresses and some of them succeed, whereas others fail. The means to treat them is

Manuscript received March 27, 2008; approved for publication by Jong Kim, Division III Editor, October 19, 2009.

This work was supported by the IT R&D program of MKE/IITA. [2008-F-034-02, Development of Security-Quality Guarantee Technology in Resilient Networks].

The authors are with the School of Electrical Engineering and Computer Science & Institute of New Media and Communications, Seoul National University, Korea, email: {sangkon.kim, shlee}@cnslab.snu.ac.kr, sseo@snu.ac.kr.

closely related to the policy of the managed network and it is beyond the scope of our research.

The third difficulty is that of applying a control system model to an automatic portscan detection system. For example, in the case of a heating system, in order to reach the target temperature, a heater is controlled, depending on the difference between the target temperature and the current temperature. Because the heater exerts a direct influence on the temperature, we can control the temperature of the object by controlling the heater. However, in the case of a portscan detection system, there is no actuator (like a heater) that can directly exert an influence on the system output. The system output is much more strongly related to the inbound traffic than parameters of the system, such as a threshold. That is, the portscan detection system is not effectively controlled to reach a target state by controlling any system parameter such as threshold and a control system model is not applicable to the portscan detection system.

The last difficulty is that of achieving better system performance using less system resources. What kinds of different log data do we process? How many time periods of log data do we correlate? Depending on these factors, the amounts of required resources can be fixed. In order for the portscan detection system to operate in real time, we must consider an ever decreasing amount of log data to reduce the required resources. On the other hand, if we want to achieve better system performance, we must consider more log data. Therefore, the system performance that is directly related to the accuracy of the estimated threshold and the required resources of the system are in a trade-off.

Because of the difficulties mentioned above, research about an automatic portscan detection system makes slow progress and any proper system has not appeared yet. Therefore, most portscan detection systems are obliged to use the fixed threshold and most current NIDS suffer from many false positives or false negatives. In this paper, we propose an automatic portscan detection system which can adaptively set the threshold in order to reduce false positives. Both false positives and false negatives are key factors when we analyze the performance of NIDS. But false positives have a much greater effect on the performance of NIDS than false negatives, because the portscan is not a real attack behavior which compromises the destination host. In addition, when the NIDS operates in real-time, false positives cause the aftermath of those that the network manager has to do and the resources of network security monitor (NSM) are expended, while any aftermath and resource consumption are not needed by false negative because the NIDS can not be aware of the existence of false negatives. Therefore, we focus on reducing false positives, rather than false negatives.

There are three main contributions of our research work, which are as follows:

- We propose an automatic portscan detection system that is discrete-time. Our system can adaptively set the threshold that will be used in the subsequent time period based on the results of analysis of the traffic. We aim to generalize our system to enable applying the existing schemes and algorithms to ours.
- We propose an adaptive threshold setting scheme, fast increase slow decrease (FISD), in order to reduce false positives. In addition, as we emphasize false positives, we compare our scheme with the other two schemes, the moving average scheme (MA) and the exponentially weighted moving average scheme (EWMA) using the implementation result. We show that our scheme is better than the others in terms of the rate of false positives.
- We propose a portscan detection algorithm, rate of increase in the number of failed connection request (RINF) that is suitable for our system as well as fast and accurate. We compare the performance of our algorithm with the other three well known algorithms (Snort, Bro, and threshold random walk (TRW)) and show that our algorithm is better than the others in terms of the rate of detections, false positives, and false negatives.

This paper is organized as follows. In Section II, we present related work in portscan detection. Section III, Section IV, and Section V provide detailed descriptions of the proposed automated portscan detection system, the proposed adaptive threshold setting scheme, FISD and the proposed portscan detection algorithm, RINF, respectively. We explain the implementation issues and evaluate the performance of ours in Section VI. Finally, in Section VII, the conclusion is given, including future work.

II. RELATED WORK

In this section, we briefly describe the prior portscan detection approaches closely related to our work. In particular, we focus on detecting the abnormal behavior of the connection requests that are sent from network based attackers.

Most portscan detection approaches have generally aimed to detect a simple form of abnormal behavior, N events within a time interval of T seconds, after NSM [7] used such detection rules. Currently, Snort [8], [9], an open source NIDS, is representative of this kind of portscan detection approach. It also has a simple rule, such as N connection attempts in T seconds, to detect a remote host as a portscanner. Because Snort is simple and does not need to know the state of the connection, it is much faster and more efficient in terms of system resources than other approaches. Therefore, it is very commonly used as a NIDS in the stub network and we choose a similar approach. However, it has the disadvantage that many false positives and false negatives are generated.

Bro [10], [11], another open source NIDS, uses failed connection requests as the distinguishing feature in order to detect portscanners. It is based on the assumption that portscanners will often send connection requests to inactive IP addresses, because portscanners have little information about the monitored network system. Except for the overhead arising from the fact that Bro has to know the state of the connection that is a success or failure, its detection rule has a simple counter, N. If the number of failed connection requests reaches N, then Bro detects the remote host as a portscanner. In Bro NIDS, the default value of N is 100, but we use $N = 20$, because this value is used in the TRW approach and it results in easy performance comparison.

The TRW [12] is based on the state of connection in a manner similar to Bro. Bro uses only the connections that are fail-

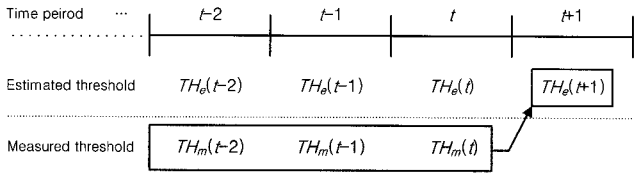

Fig. 1. Discrete time model of our system.

ures as significant events, but TRW uses all of the connections and aims to detect portscanners promptly. As each connection attempt, event Y, is observed, TRW calculates the likelihood ratio, $\Lambda(Y)$, using the sequential hypothesis testing in order to decide whether the remote host is a portscanner or a benign host. If $\Lambda(Y) \geq \eta_1$, an upper threshold, then the remote host is a portscanner. If $\Lambda(Y) \leq \eta_0$, a lower threshold, then the remote host is benign. Otherwise, the method is to wait for the subsequent observation and update $\Lambda(Y)$. Applying this rule, TRW detects portscanners and aims to reduce the number of observations until it reaches a decision.

The graph based intrusion detection system (GrIDS) [13], [14], detects portscanners by building graphs of activity in which the nodes represented hosts and the edges represented the network traffic between hosts.

Staniford *et al.* [15] and Leckie *et al.* [16] propose probabilistic approaches using the likelihood of a source being a scanner. At first, these approaches assign a priori probabilities to IP's existing within the network and then portscanners are detected by comparing the access probabilities of source IP's targeting this network with the priori probabilities.

Time based access pattern sequential (TAPS) hypothesis testing [17], [18] is proposed and implemented in order to detect portscans on the backbone.

Zang *et al.* [19] demonstrates that sampling distorts various traffic features and degrades the performance of three selected portscan detection algorithms in terms of success detection ratio and false positives.

III. PROPOSED AUTOMATIC PORTSCAN DETECTION SYSTEM

With the aim of designing an automatic portscan detection system to detect the portscan behavior of fast, self-propagating malicious codes, we investigate both a proper system model and the behavioral differences between a normal remote host and a remote portscanner. We aim to design a discrete-time system that is suitable for updating the threshold periodically, as shown in Fig.1. If we can have a set of true scanners during each time period, the threshold of the subsequent time period can be adaptively estimated and set, based on the statistical analysis of the inbound traffic that is sent from portscanners in prior time periods. Unfortunately, because of the first difficulty in designing the system that is mentioned in the Introduction, without an "oracle" that can provide us with true information about any remote host, we can not discover a "ground truth" set of remote hosts that are true portscanners.

As the second best solution, we take a "best effort" approach to obtain a set of portscanners that is distinguished from the all

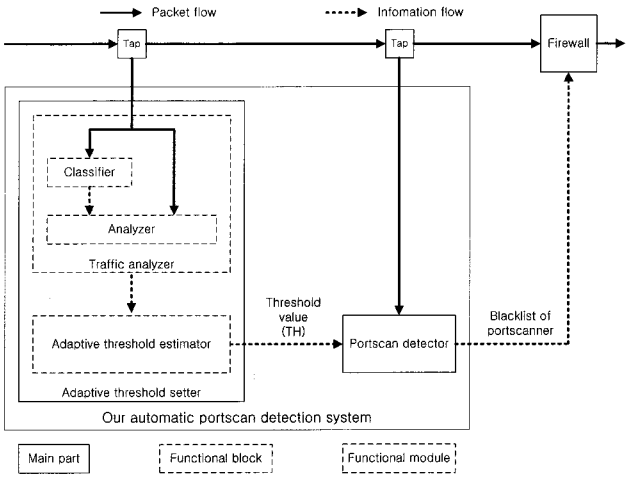

Fig. 2. Architecture of our automatic portscan detection system.

of remote hosts within a time period. In order to do this we need a criterion to distinguish portscanners from remote hosts. Firstly, we define a "best effort" set of portscanners detected based only on the threshold used during the current time period. In this case, the detected portscanners are totally dependent on the threshold that is used during the time period. In addition, when the same inbound traffic is used as the input of the system during different time periods, two different "best effort" sets of portscanners can be obtained, because both of the thresholds can be different. That is, even though the input of the system is the same, the output is varying depending on the threshold. Therefore, if we use this system model, both the "best effort" set of portscanners and the performance of the system will be unreliable. Due to the unreliability of the system, our first attempt, defining a "best effort" set of portscanners detected during the current time period is not applicable to the system. In order to make the system become independent from the threshold, we need a common criterion for the "best effort" set of portscanners and the same criterion must be used during all time periods. In this case, even though the same inbound traffic is used as the input of the system during different time periods, the same "best effort" set of portscanners will be obtained, due to the use of common criterion. These are explained in detail in what follows, and in this paper, we call the "best effort" set of portscanners as potential portscanners.

For the purpose of the independence from the threshold that is mentioned above, we separate the traffic analysis that is the basis of our system and the prompt portscan detection in our system. The architecture of our automatic portscan detection system is shown in Fig. 2, which consists of two main parts.

- An adaptive threshold setter whose role is to adaptively set the subsequent threshold based on statistical analysis of inbound traffic during prior time periods.
- A portscan detector which has the role of promptly detecting portscanners among the remote hosts by using the threshold set by the adaptive threshold setter.

Note that the adaptive threshold estimator is implemented in the former part based on the result of the traffic analyzer, and in the latter part, portscanners are rapidly and accurately detected and reported to the firewall in order to block the portscan traffic.

A. *Adaptive Threshold Setter*

The adaptive threshold setter is a fundamental part of our system. It provides the potential portscanners existing during the current time period and the estimated threshold that is used for the portscan detector during the subsequent time period. There are three main roles of this part as follows.

- Distinguish the potential portscanners from the remote hosts that exist during the current time period, according to the predefined criterion.
- Calculate and analyze statistical data for the inbound traffic that is sent from the potential portscanners distinguished previously.
- Adaptively estimate the threshold of the subsequent time period by using the statistical results of analysis of the traffic during prior time periods.

In order to perform these three roles, this part consists of two functional blocks, a traffic analyzer and an adaptive threshold estimator, as shown in Fig. 2. The first two roles are processed in the traffic analyzer. The last is performed in the adaptive threshold estimator.

A.1 Traffic Analyzer

Depending on its role, this functional block is divided into two functional modules, a classifier and an analyzer. A classifier aims to discover potential portscanners while an analyzer focuses on obtaining statistical data that is needed by the adaptive threshold estimator.

A.1.a Classifier. A classifier plays a core role in our system, because it enables us to obtain the potential portscanners that provide the basis of other functions in our system and exerts a significant influence on system performance. In fact, according to the predefined criterion, potential portscanners are distinguished from remote hosts by this classifier of the traffic analyzer. Therefore, the predefined criterion is very important for our system and it is closely related to the policy of the managed network.

In our system, to define a criterion of the potential portscan in the classifier, we provide two categories to discover the abnormal behavior of portscanners. When a remote host is fixed, one is the number of failed connection requests (NFCR) that are destined to the distinct destination IPs in the local network. The other is the percentage of inactive local hosts (P_ILH), which is the ratio of the number of distinct local hosts that received the failed connection request to the number of distinct local hosts that received the connection request [6]. NFCR and P_ILH are good distinguishing features of the abnormal behavior of portscanners. However, when only one of two categories is used as the criterion, this is not enough to define the abnormal behavior of portscanner precisely. When only NFCR is used as the criterion, it is the same as the Bro. If only P_ILH is used, a large number of false positives are generated and the detailed description is in the Section V. Therefore, we use both categories to define the criterion for the classifier as follows:

- **Condition 1**: NFCR $\geq \alpha$
- **Condition 2**: $P\_ILH \geq \beta$

where α and β are configurable parameters. The network manager has to properly set these two values, but there is no need to repeatedly set these values. In this paper, α and β are set to 5 and 80%, respectively and the detailed description is given in Section VI. When a remote host is fixed, if both Condition 1 and Condition 2 are satisfied, then the remote host is classified as a potential portscanner. The operations of the classifier are shown in Fig. 3.

During the time period, when a connection request is received, using the information about the source IP (SIP), the destination IP (DIP) and the state of the request, the classifier must simultaneously perform two procedures: One is to count NFCR and the other is to calculate P_ILH. These two procedures are performed as shown in Fig. 3(a).

At the end of a time period, all of the hosts that are registered in the list of the remote hosts are judged by the classifier according to the NFCR and P_ILH values that are obtained. Based on the α and β parameters, potential portscanners are distinguished from registered remote hosts as shown Fig. 3(b). Because this is the basis of further procedures, we adapt both factors to the criterion of our classifier in order to increase the accuracy and reliability of the results.

If we assume N_R number of remote hosts sent at least a connection request packet to the managed network and each of those tried to connect to averagely N_L number of distinct local hosts during a time period. The required memory will be proportional to $(N_R + N_L)$ and the required process power will be proportional to $(N_R^2 N_L^2)$ because most system resources are spent by IP storage and IP comparison procedures. Therefore, the classifier is much slower than other portscan detection algorithms due to the complexity of the procedure. Thereby, the classifier can not operate in real time and it is not applicable to the area of prompt portscan detection. That is, this is not applicable to the portscan detector of our system. As the size of the managed network becomes ever larger, the required amount of processing power, time delay and memory resources rapidly increase. Therefore, our automatic portscan detection system is not applicable to a large scale network but rather an enterprise network.

A.1.b Analyzer. The role of the analyzer is to provide the required statistical results of analysis of traffic to the subsequent functional block, an adaptive threshold estimator. Because the analyzer must examine all the connection requests during a time period, it has a similar structure to the classifier, as shown in Fig. 4. That is, during a time period, when a connection request is received, the distinguishing feature that is required in the portscan detector must be calculated, based on the SIP of the received packet. And, at the end of the time period, the statistical result must be calculated and analyzed using potential portscanners that are provided by the classifier. In practice, the statistical result consists of the average and standard deviation of potential portscanners.

The distinguishing features of four portscan detection algorithms are briefly described as follows.

- **Snort**: Calculate the number of connection requests within a fixed time interval when a connection request is received.
- **Bro**: Count the failed connection requests when a connection request is received and fails.
- **TRW**: Calculate the likelihood ratio when a connection request is received.

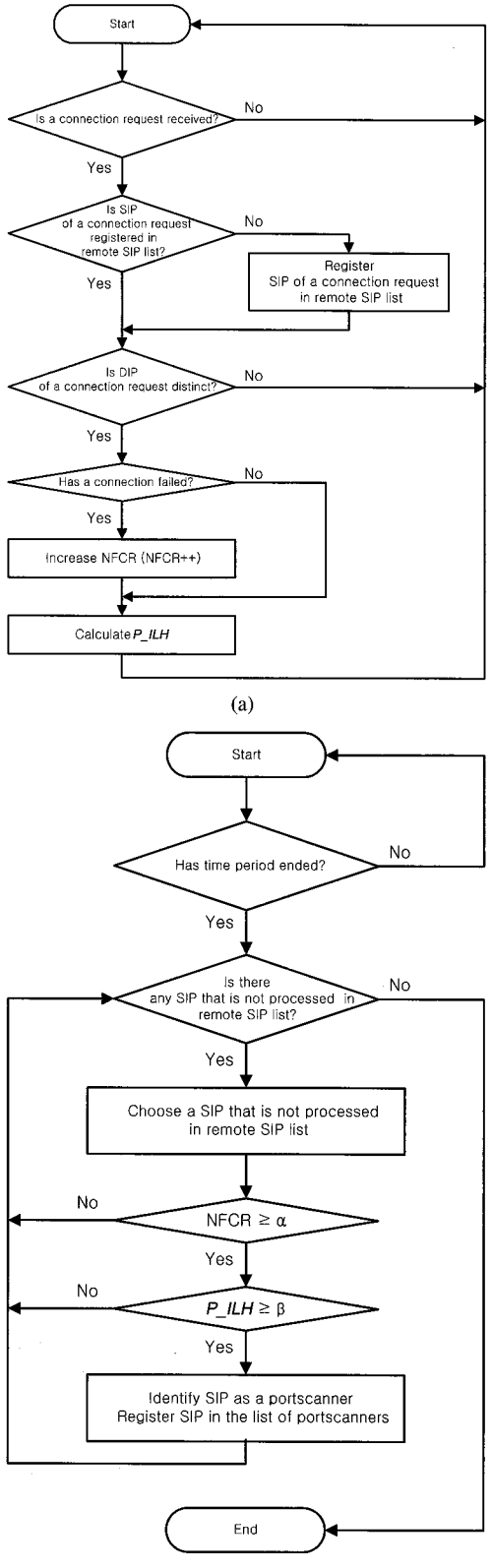

Fig. 3. Functional flowchart of a classifier: (a) NFCR counter and P_ILH calculator and (b) portscanner classification.

- **RINF**: Calculate the RINF value when a connection request is received and fails.

The detailed description of the distinguishing features that are used in each of four portscan detection algorithms is given in Section V.

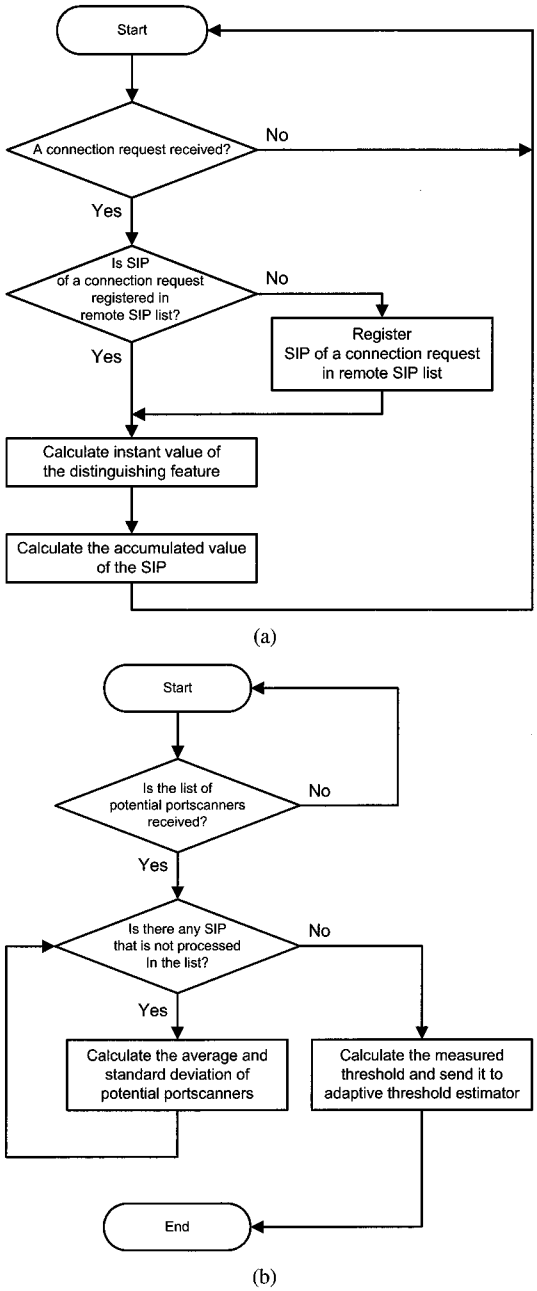

Fig. 4. Functional flowchart of an analyzer: (a) Distinguishing feature calculation and (b) statistical data calculation.

During a time period, the procedure used to calculate the required distinguishing feature is performed as shown in Fig. 4(a). Depending on the portscan detection algorithm, the type of the accumulated value is determined. For example, Bro needs a simple count value, while Snort and RINF need the average of the immediate values.

At the end of time period, after the potential portscanners are provided by a classifier, the average and standard deviation values of the potential portscanners are calculated. Now, we must define the measured threshold in the current time period, by using statistical result. In this paper, when the average value is represented as $m(t)$ and the standard deviation value as $\sigma(t)$ of potential scanners in terms of a distinguishing feature, the mea-

sured value is defined as follows:

$$TH_m(t) = m(t) - k\sigma(t) \quad (1)$$

where k is a configurable constant value and set by the network manager. The measured threshold manifests that it is calculated based on the statistical results of analysis of the traffic and we will treat this measured threshold as a reference. In other words, at the end of time period, we can obtain the measured threshold, $TH_m(t)$ and use this value to estimate the threshold of the subsequent time period. In order to do this, this value is sent to a subsequent functional block, the adaptive threshold estimator. This is shown in Fig. 4(b).

A.2 Adaptive Threshold Estimator

The role of this function block is to estimate the threshold of the subsequent time period using the measured threshold that is provided by the traffic analyzer. The measured thresholds of prior time periods are used to estimate the subsequent threshold. Because of the generalized processing flow in our system, existing adaptive estimation schemes such as MA and exponentially EWMA etc. can be applied to this functional block. We compare these two schemes with our scheme. Both the detailed description of our scheme, FISD, and the extended explanation of MA and EWMA are given in Section IV.

B. Portscan Detector

The role of this part is to detect a portscan using the estimated threshold. It must examine all of the inbound packets and operate in real time. In our system, any one of the portscan detection algorithms can be used for this part. Snort, Bro, TRW, and RINF are considered as prospective candidates in this paper and each of them is evaluated in term of adaptability to our system. RINF is used for this part when we perform comparative analysis of the adaptive threshold setting schemes.

IV. FAST INCREASE SLOW DECREASE

Using the measured thresholds in prior time periods, we can estimate the threshold of the subsequent time period. According to our system model that is shown in Fig. 1, the subsequent threshold is estimated based on the trend in the change of prior measured thresholds. Therefore, when the measured thresholds are continuously changing within a narrow range, there is no great difference between the estimated threshold and the measured thresholds. Thus, the detection results of the portscan detector are approximately the same as the potential portscanners classified by the traffic analyzer of the adaptive threshold setter, because the subsequent threshold that is estimated, is close to the measured threshold. However, when there is a great deal of variation, this results in a significant difference between the estimated value and the measured value of the threshold, and as a result, many false positives and false negatives will be generated. Because the estimation process is based on prior thresholds and information about inbound traffic during a subsequent time period is unknown, it is impossible to estimate the subsequent threshold accurately. Under these circumstances, we propose an FISD scheme in order to reduce false positives as efficiently as possible, even when there is a great deal of variation. Firstly, we briefly introduce MA and EWMA schemes, then, we describe our FISD scheme in detail.

A. MA and EWMA

If $TH_m(t)$ is the measured threshold during time period t, and $TH_e(t)$ is the estimated threshold during time period t, then the estimated threshold can be calculated over the prior time period using MA as follows:

$$TH_e(t+1) = \frac{\sum_{i=0}^{\min\{n,t\}-1} TH_m(t-i)}{\min\{n,t\}} \quad (2)$$

where n is a number of time periods for which an average of the thresholds is taken and $\min\{a,b\}$ is a, if $a \leq b$ or b, if $a > b$.

In the EWMA scheme, the estimated value is calculated as follows:

$$TH_e(t+1) = \mu TH_e(t) + (1-\mu)TH_m(t) \quad (3)$$

where μ is the EWMA factor.

Because MA and EWMA are simple and do not need a great deal of processing power, these are popularly used in many applications. However, during rapid variations of the measured thresholds, performance is greatly degraded. Because of this, we propose an adaptive threshold setting scheme, FISD, to reduce false positives.

B. FISD

Generally, if the threshold is increased, false positives are reduced. On the other hand false negatives are increased. By contrast, if the threshold is decreased, false positives are increased, while false negatives are decreased. By using the previously mentioned relation between threshold and false positives, we design an FISD scheme that increases the threshold rapidly, but decreases it slowly, based on the measured thresholds in order to decrease false positives as possible as we can. We define the notations for our scheme as follows:

- $TH_{\min}$: The minimum value of the threshold;
- $TH_{\max}$: The maximum value of the threshold;
- N_{step}: The number of discrete steps between $TH_{\min}$ and $TH_{\max}$;
- $TH_m(t)$: The measured threshold at time period t;
- $TH_e(t)$: The estimated threshold at time period t.

In order to avoid setting the threshold too high or low and guaranteeing the proper performance of the system, FISD imposes the limitation that the threshold must be set to a value between the minimum and maximum value. Further, we use a discrete threshold, and the size of a step, ΔTH, can be configured by the following

$$\Delta TH = \frac{TH_{\max} - TH_{\min}}{N_{\text{step}} - 1}. \quad (4)$$

At the end of time period t, the measured threshold, $TH_m(t)$ is provided by the traffic analyzer functional block. The functional flowchart of FISD is shown in Fig. 5.

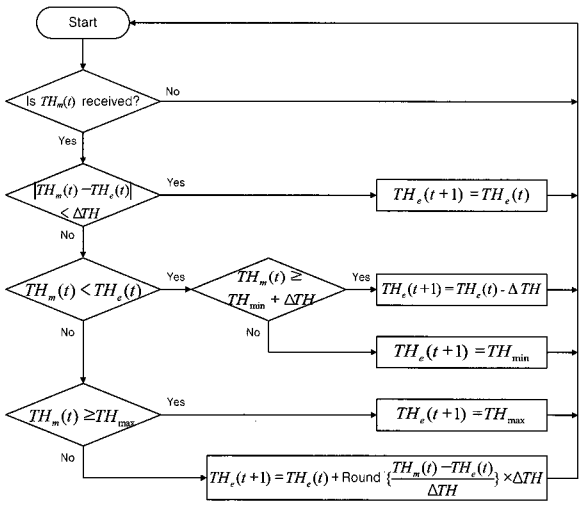

Fig. 5. Functional flowchart of FISD.

As the name of our scheme, FISD, implies, the estimated threshold can be rapidly increased but slowly decreased, based on the measured value. Our scheme is simple and its processing overhead is quite low, but it can effectively reduce false positives. In the case for which false negatives are more serious than the false positive, a fast decrease slow increase (FDSI) scheme is needed and can be easily derived from FISD.

V. RATE OF INCREASE IN THE NUMBER OF FAILED CONNECTION REQUEST

In our automatic portscan detection system, the threshold of a subsequent time period will be estimated by an adaptive threshold setting scheme, based on statistical analysis of potential portscanners. The distinguishing feature is determined by which portscan detection algorithm is used. That is, when one of Snort, Bro, and TRW is applied to the portscan detector of our system, statistical results of analysis for traffic must be calculated in the form of the distinguishing feature that is used in the applied algorithm. According to our analysis result of real traffic trace, the standard deviation of distinguishing feature of potential portscanners is too high in Snort and Bro cases. It is quite natural because there are many different types of portscanners in potential portscanners when we observe potential portscanners in terms of distinguishing feature. In this case, there is a problem in defining the measured value using (1). Therefore, it is very hard to apply one of three existing algorithms to our system. The difficulties are explained as follows.

Snort uses the number of connection attempts within a fixed time interval as the distinguishing feature and in practice the number of connection requests that are sent from a fixed remote host are counted within the moving time window with the same size as the fixed time interval. At a glance, this factor seems similar to the rate of connection request increase but in practice this is implemented using a simple counter with a time window. In this case, the statistical results are the average and standard deviation of counter values of potential portscanners. If the standard deviation is too large, the measured threshold that is calculated by (1) can be a negative value and it is not accessible. Even though we provide a boundary for the threshold, by using a minimum and maximum value, the measured value is not reliable, thus, the system can not properly operate. When a 3 seconds time window is used and the peak counter value is measured for a remote host in our traffic analysis, the average and standard deviation of counter values of potential scanners are 2.26 and 4.68, respectively.

In Bro case, the number of failed connection requests is used as the distinguishing feature and it is simply implemented by a counter similar to Snort. According to our traffic analysis, the average and standard deviation of NFCR of potential scanners are 811.87 and 1572.27, respectively. It is verified that the standard deviation of the statistical result is too large. Thereby, it has the same problem as Snort.

TRW based on the sequential hypothesis test, uses the likelihood ratio as the distinguishing feature and its threshold is related, not to the inbound traffic, but to the target values of the detection probability and the false positive probability. The probability that a benign host sends a connection request that is a success, θ_0 and the probability that a portscanner sends a connection request that is a success, θ_1, are both closely related to the inbound traffic. θ_0 and θ_1 can be updated by statistical results of analysis for traffic, but this needs a huge amount of system resources. Moreover, it uses two thresholds to decide whether a remote host is a portscanner or benign. Therefore, it is a very complicated and difficult matter to apply the case for which both thresholds are adaptively set to our system.

As mentioned previously, the existing portscan detection algorithms are not suitable for our system even though we tried to generalize our system. Therefore, we propose a new portscan detection algorithm. NFCR and P_ILH are good distinguishing features for detecting portscanners but it is impractical to use both procedures at once. We aim to develop a practical portscan detection algorithm that is both applicable to our system and fast and accurate.

We assume that a fast, self-propagating malicious code does not have any information about the managed network and performs portscans at a high rate. Therefore, portscanners generate failed connection requests at a much higher rate than benign hosts. Our algorithm, RINF, uses two distinguishing features. The main feature is the RINF. The complementary feature is the NFCR.

When a connection request is received and fails, RINF has to be calculated by using its time information. Therefore, if we process each of failed connection requests, it can make too high RINF and needs a large amount of system resources. In order to solve this problem, we design RINF to be a discrete time-based algorithm. In this paper, we use a 1 milli-seconds (ms) time slot so that all the failed connection requests within a time slot are processed at once. When p number of failed connection requests are received from a fixed remote host within the current time slot t_c, and the last failed connection request was received within the prior time slot t_p, we define the instant rate of increase in the number of failed connection requests, instant_RINF as follows:

$$\text{instant\_RINF}(t_c) = \log_{10}\left(\frac{p}{t_c - t_p}\right). \tag{5}$$

In order to reduce the standard deviation, we impose a restric-

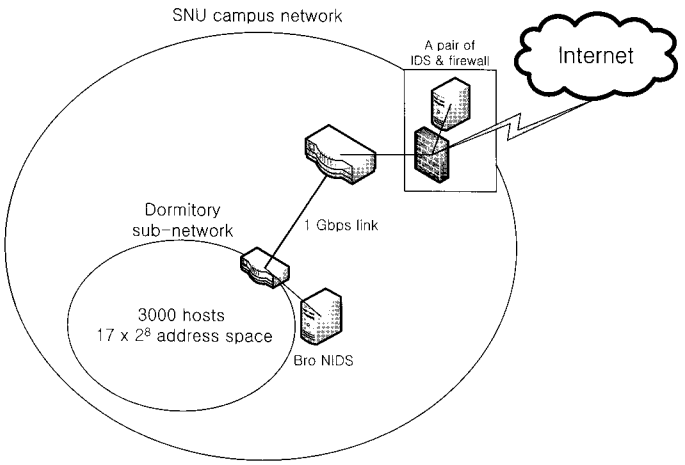

Fig. 6. Network environment of the dormitory in SNU.

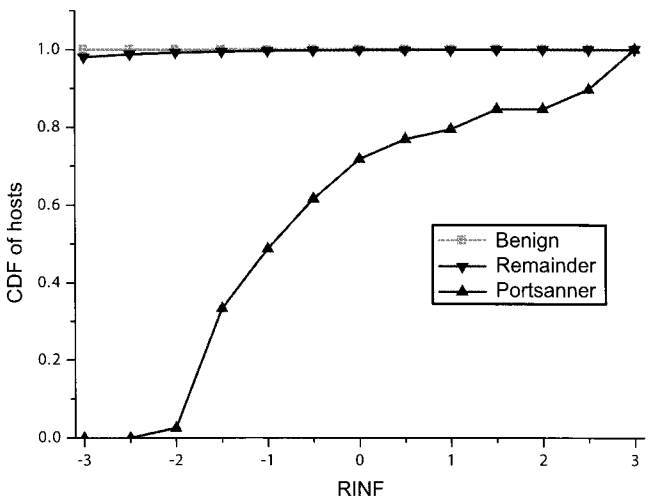

Fig. 7. RINF-CDF of portscanner, benign, and remainder.

tion on instant_RINF, as follows:

$$\text{instant\_RINF} = \begin{cases} -3, & \text{if } y < -3\,; \\ y, & \text{if } -3 \le y \le 3\,; \\ 3, & \text{if } y > 3. \end{cases} \tag{6}$$

We define RINF as the arithmetic average of the instant_RINF values that are generated within the fixed time window, T_w. If there are q number of time slots within the time window with a failed connection request, the RINF is expressed as follows:

$$\text{RINF} = \left(\frac{\sum_{i=2}^{q} \text{instant\_RINF}(i)}{q-1} \right) \tag{7}$$

where, $i = 1, 2, \cdots, q$ is a number representing the time slots having a failed connection request. In this paper, we set the time window T_w to 5 minutes. The RINF is a value within the range -3 to 3.

A. Data Analysis

In order to verify the performance of our algorithm and system, we gathered log data from a dormitory at Seoul National University (SNU) and used it for analysis. In the verification, we show the difference in the portscan behavior between portscanners and other hosts from the RINF perspective.

As shown in Fig. 6, this dormitory at SNU has about 3,000 hosts and an address space of 17×2^8 with a 1 Gbps link. The verification is based on analytical results for real-world log data. We focus in detail on analyzing connection requests and state information sent from remote hosts. Bro NIDS provides state information about each connection request. We installed Bro NIDS at the front gateway router of the dormitory network. For Bro NIDS, we used release 0.9, which is a stable version. We gathered a set of log data during a period of 160 minutes, as follows.

- Log data : November 9, 2006 12:50–15:30 (160 min)

To verify the abnormal behavior of portscanners and show the RINF value is a good and precise distinguishing feature for the detection of portscanners, we analyze a set of log data. Firstly, we classify the remote hosts as portscanners, benign hosts and remainders, according to criteria like a potential portscanner. We define strict criteria for portscanner and benign hosts, because a remote host is examined and classified by using all log data. If a remote host's NFCR counter is less than 5 and its P_ILH is less than 10%, then, this host is classified as a benign host. If a remote host's NFCR counter is greater than 20 and its P_ILH is greater than 80%, then, this host is classified as a portscanner. If a remote host does not satisfy any of two criteria, it is classified as a remainder. The result of classification is summarized as follows.

- The number of total remote hosts: 22,534.
- The number of portscanners: 39.
- The number of benign hosts: 9,163.
- The number of remainders: 13,332.

At this moment, in order to show that the abnormal behavior of portscanner is not properly defined by only P_ILH, we analyze a set of remainders in terms of P_ILH. As an analysis result, P_ILH of approximately 95% of remainders have larger P_ILH than 80% and a large number of false positives will be generated. Therefore, both NFCR and P_ILH are used as the criterion for classification.

Our results of analysis are shown in Fig. 7, which shows the CDF of each subset graph to respect to the RINF. That is, as the RINF value increases, the CDF of each subset increases. It is clear that, in the RINF, there is a significant difference between portscanners and other hosts, including benign hosts and remainders. According to our results of analysis, the average and standard deviations of RINF of portscanners are 1.82 and 2.26, respectively. Even though the standard deviation is greater than the average, the result calculated by (1) is a value which can be set, in RINF. In this case, a negative threshold value can be obtained, and it can be set in the RINF, whereas, it is impossible to set it in other algorithms. Therefore, RINF is suitable for our system.

B. RINF

We named our algorithm RINF using a main distinguishing feature, but in fact NFCR is also used. When a remote host sends

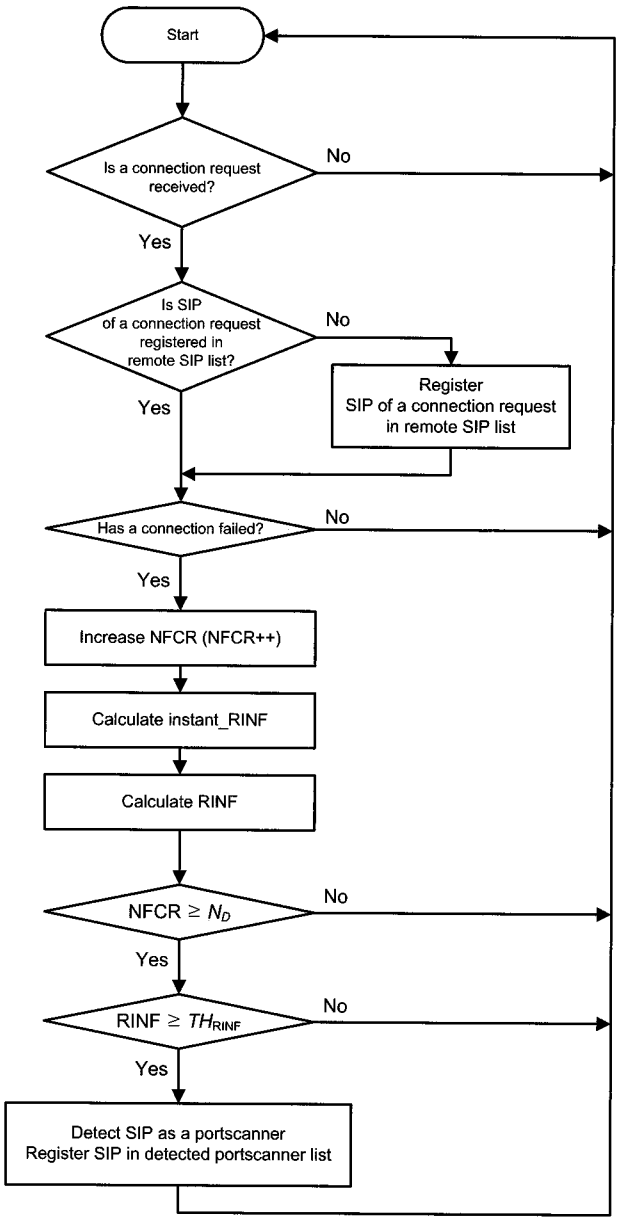

Fig. 8. Functional flowchart of RINF.

a small number of failed connection requests within a short time interval, then, there are no more failed connection requests for a long time, the remote host will be detected as a portscanner, if we use only RINF for portscan detection. However, this violates our assumption that a portscanner will generate many NFCRs with a high rate. Therefore, in order to reduce this kind of false positive, NFCR compensates for the weak point of RINF. That is, both RINF and NFCR are used to detect portscanners in our algorithm. The two conditions are shown as follows:

- **Condition 3**: NFCR $\geq N_D$
- **Condition 4**: RINF $\geq TH_{\text{RINF}}$

where N_D is a threshold of NFCR and TH_{RINF} is a threshold of RINF. Only when a remote host sends more failed connection requests than N_D to the managed network and the host's RINF is greater than TH_{RINF}, is this detected as a portscanner. The operation of our algorithm is shown in Fig. 8

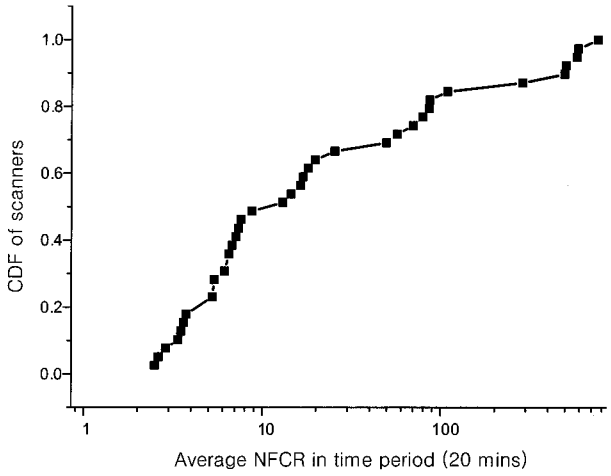

Fig. 9. CDF of portscanners vs. average NFCR.

Since we verify the difference between portscanners based on the RINF values using real-world log data in our portscan detection algorithm, we define (7) as a numerical distinguishing feature for detecting portscanners. Based on Conditions 3 and 4, we eventually detect portscanners in real time.

VI. IMPLEMENTATION AND PERFORMANCE EVALUATION

In this section, we describe the means to set the value of the parameters in our automatic portscan detection system, in order to achieve proper performance. If we implement other portscan detection algorithms in our system, the performance of our system and other algorithms can be reasonably compared and evaluated. However, because of the difficulties of applying other algorithms to our system, which are mentioned in Section V, Snort, Bro, and TRW were not implemented in our system. Therefore, these portscan detection algorithms are independent of our system and compared to our algorithm. Firstly, we observe the system.

Our system is a discrete-time system, so we must decide the size of the time period T_p. Considering the network environment, the network manager can set T_p by the minute, hour or day. In this paper, we set T_p to 20 minutes, so the 160 minute log data is separated into eight time periods. In order to define criterion for a potential portscanner for the classifier, we set α and β in Conditions 1 and 2 to 5 and 80%, respectively. Our analysis result of portscanners is shown in Fig. 9, which shows the CDF of portscanners to respect to the average NFCR during a time period. In order for potential scanners to include about 80% of portscanners, we set α to 5. In the analyzer, the k parameter of (1) is set to 1.45.

In our FISD scheme, three parameters, TH_{min}, TH_{max}, and N_{step} are configurable. TH_{min} and TH_{max} are closely related to the probability that a portscanner is detected by the system, when the scan rate of a portscanner is fixed. Therefore, we approximately estimate the detectable scan rate of remote portscanners, based on the network environment, TH_{min} and TH_{max}. Let us suppose that a remote host sends portscan packets to random IP addresses at a constant scan rate s during a time

Table 1. Comparative analysis result of portscan detection algorithms.

	Snort (7/3sec)	Bro ($N = 20$)	TRW	RINF $= -2$		
				$N_D = 3$	$N_D = 4$	$N_D = 5$
Detections	14	29	37	39	35	33
False positives	223	125	68	38	9	4
False negatives	25	10	2	0	4	6
$E[N\|H_1]$	33.27	25.67	4.31	5.85	8.14	10.51

period, and we monitor a network with a size of 2^{32-x} IP addresses, denoted as $/x$ network. The probability that a portscan packet sent from a remote portscanner is destined to the monitored network, P_{hit}, is given as $P_{\text{hit}} = 2^{-x}$. The network manager can obtain the probability that an inbound connection request packet will fail, P_f, by statistical analysis of traffic. Therefore, the probability that a portscan packet sent from a remote portscanner is destined to the monitored network and fails, P, is given as $P = P_{\text{hit}}P_f$. The average rate of the failed connection request (FCR) that is sent from a portscanner is destined to the monitored network, $E(\lambda)$, is given as $E(\lambda) = sP$. Here, if we assume that during the time period, there is at least one RINF that is larger than $E(\lambda)$, the rate of FCR is the same as for RINF so the threshold, $TH \leq E(\lambda) = sP$. We can obtain the estimated scan rate of a remote portscanner that can be detected by our system as follows:

$$\hat{S} \geq \frac{TH}{P}. \tag{8}$$

In order for this estimated scan rate of a portscanner to be accurate, the following conditions are derived:

The probability that j number of NFCRs is received in a time period is

$$\binom{sT_p}{j} P^j (1-P)^{(sT_p - j)}.$$

In this case, when all of the time intervals between failed connection requests are the same, RINF is the minimum;

$$\min(\text{RINF}) = \frac{j-1}{T_p}.$$

In order to ensure a true assumption;

$$\min(\text{RINF}) = \frac{j-1}{T_p} \geq E(\lambda) = sP.$$

If this equation is satisfied;

$$\max(\text{RINF}) \geq E(\lambda) = sP.$$

Therefore,

$$\Pr\{\max(\text{RINF}) \geq E(\lambda)\} \geq \Pr\{\frac{j-1}{T_p} \geq E(\lambda)\}.$$

And,

$$\begin{aligned}\Pr\left\{\frac{j-1}{T_p} \geq E(\lambda)\right\} &= \Pr\{j \geq sT_pP + 1\} \\ &= \sum_{j=sT_pP+1}^{sT_p} \binom{sT_p}{j} P^j (1-P)^{sTp-j} \\ &= 1 - \sum_{j=0}^{sT_pP} \binom{sT_p}{j} P^j (1-P)^{sTp-j}.\end{aligned}$$

Finally, we obtain following equation;

$$\begin{aligned}&\Pr\{\max(\text{RINF}) \geq E(\lambda)\} \\ &\quad \geq 1 - \sum_{j=0}^{sT_pP} \binom{sT_p}{j} P^j (1-P)^{sTp-j}.\end{aligned} \tag{9}$$

According to our assumption, we can approximately estimate the scan rate of a portscanner using (8). Then, we can examine the probability that the estimated scan rate of a portscanner is detected within a time period using (9).

We evaluate the performance of adaptive threshold setting schemes, including MA, EWMA, and FISD, in terms of detections, false positives, and false negatives, during each time period, using real log data. In addition, the performance of the portscan detection algorithms, Snort, Bro, TRW, and RINF is evaluated, in terms of detections, false positives, false negatives and the average number of connection requests that are destined to distinct local hosts before a remote host is flagged as a portscanner within all the log data ($E[N|H_1]$).

A. Comparison of Adaptive Threshold Setting Schemes

For the comparative analysis of MA, EWMA, and FISD, RINF is used for the portscan detector of our system and two comparison results are provided using two traffic traces gathered from two sites, SNU and DJ University.

In the case of SNU, we set the initial threshold to the minimum value, -2.30, in order to deal with rapid variations. The measured threshold of each time period is calculated by (1) with $k = 1.45$ and used for threshold estimation. The parameter n is set to 2 for MA. The parameter of EWMA, μ is set to 0.1 in order to give weight to the most recent value. For the FISD, TH_{min}, TH_{max}, and N_{step} are set to -2.30, -1.30, and 40, respectively.

Fig. 10 shows the results of comparison of detections, false positives and false negatives. In Figs. 10(a) and 10(b), except for the second time period, there is no difference in the number of detections and false negatives. The number of false positives of our FISD scheme is the smallest and it shows better performance than MA and EWMA in Fig. 10(c).

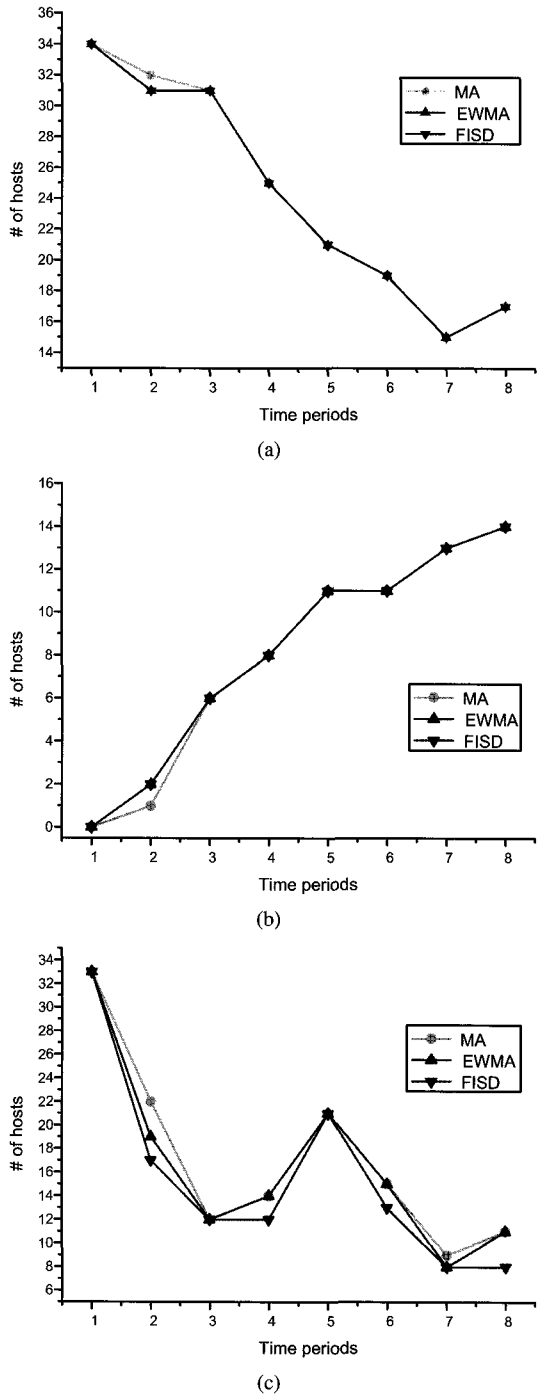

Fig. 10. Comparison result of adaptive threshold setting schemes using SNU traffic: (a) Detections, (b) false negative, and (c) false positive.

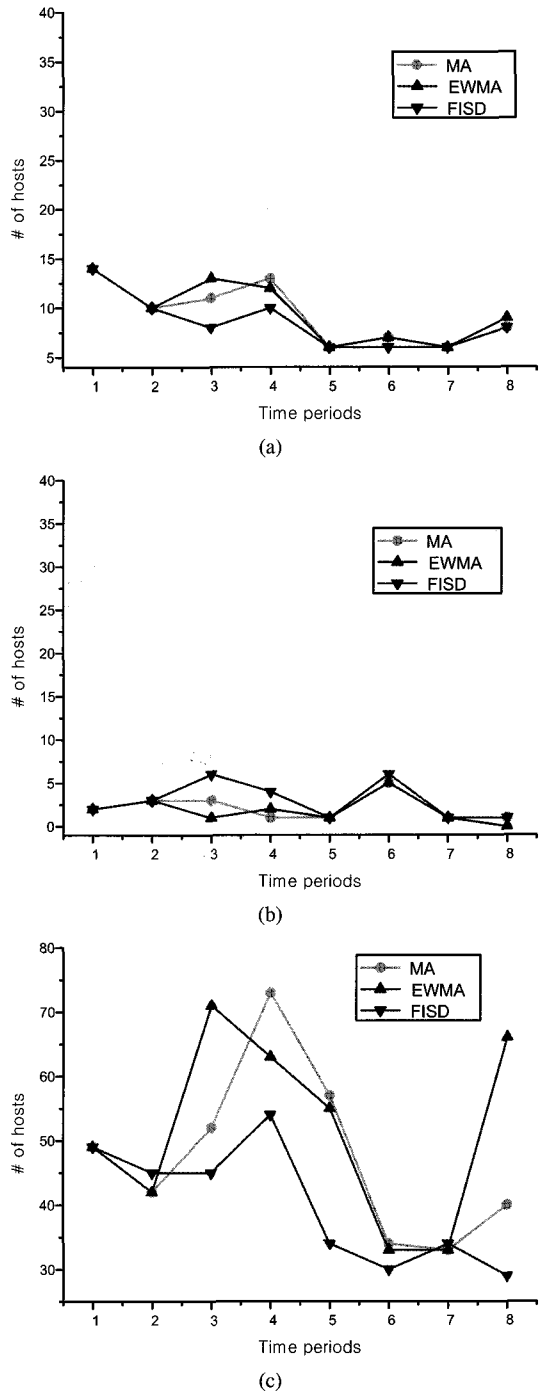

Fig. 11. Comparison result of adaptive threshold setting schemes using DJ University traffic: (a) Detections, (b) false negative, and (c) false positive.

In the case of DJ University, we set the initial threshold to -1.09 and k is set to 1.15. The parameter n is set to 2 for MA. The parameter of EWMA, μ is set to 0.4. For the FISD, $TH_{\min}$, $TH_{\max}$, and N_{step} are set to -1.09, -0.41, and 40, respectively.

Fig. 11 shows the results of comparison of detections, false positives and false negatives. In Figs. 11(a) and 11(b), there is no big difference in the number of detections and false negatives but the difference is slightly larger than SNU. in Fig. 11(c), the performance of our FISD scheme is better than MA and EWMA.

We verify that FISD can reduce the incidence of false positives without serious degradation of detection and false negatives using real log data using the two implementation results. In addition, we show that our system can adaptively set the threshold based on the analysis results of traffic.

B. Comparison of Portscan Detection Algorithms

For comparison of Snort, Bro, TRW, and RINF, the values of parameters are chosen as follows. Snort's threshold $= 7/3$ sec in order to obtain best performance. In the case of Bro,

$N = 20$ for the detection result obtained by Bro NIDS. $\eta_1 = 99$, $\eta_0 = 0.01$, $P[Y = 0|H_0] = \theta_0 = 0.8$, $P[Y = 1|H_0] = 0.2$, $P[Y = 0|H_1] = \theta_1 = 0.2$ and $P[Y = 1|H_1] = 0.8$ are used for TRW. For RINF, $N_D = 3$ and $TH_{\text{RINF}} = -2$ in Conditions 3 and 4. The log data gathered from SNU is used for the analysis and the results are shown in Table 1.

It is shown that RINF is much better than Snort and Bro in all respects. In terms of detections and false negatives, RINF and TRW are almost the same but RINF is slightly better than TRW. In terms of false positives, RINF is relatively better than TRW. However, RINF is slightly worse than TRW, in terms of $E(N|H_1)$. If N_D of RINF is increased, false positives are greatly reduced without serious degradation of detections and false negatives, but $E(N|H_1)$ is somewhat increased. We thus verify that the RINF is a fast and accurate portscan detection algorithm using the results of analysis.

VII. CONCLUSION

In this paper, we proposed an automatic portscan detection system with adaptive threshold setting. First, we designed a practical discrete-time system to be sequentially implemented. In addition, we proposed an adaptive threshold setting scheme that is capable of reducing false positives, while adaptively setting the threshold based on the results of analysis of traffic. Moreover, we proposed a portscan detection algorithm that is suitable for the proposed system. Note that we provided the log analysis to verify the theory of our portscan detection algorithm, and described the design and implementation of our automatic portscan detection system, adaptive threshold setting scheme and portscan detection algorithm. We evaluated the performance of our system, scheme and algorithm. According to the performance evaluation, the proposed system automatically and adaptively sets the threshold. The proposed scheme effectively reduces false positives. The proposed algorithm can fast and accurately detect portscanners.

Our research was completely focused on portscan detection, which is the reconnaissance phase of a fast, self-propagating network attack. In our future work, we will include an automatic specific attack detection system in order to detect and contain new attacks, using an anomaly detection method. In addition, we will include an automatic pattern generation system, in order to support the misuse detection method, for a known attack. Then, our system will be a fully automated network defense system.

REFERENCES

[1] N. Weaver, V. Paxson, S. Staniford, and R. Cunningham, "Large scale malicious code: A research agenda," University of California, Berkeley, Tech. Rep. 2003

[2] CERT, CERT/CC, "CERT advisory CA-2001-19 code red worm," July 2001. [Online]. Available: http://www.cert.org/advisories/CA-2001-19.html

[3] CERT, CERT/CC, "CERT advisory CA-2001-26 nimda worm," Sept. 2001. [Online]. Available: http://www.cert.org/advisories/CA-2001-26.html

[4] CERT, CERT/CC, "advisories." [Online]. Available: http://www.cert.org/advisories/

[5] S. Staniford, V. Paxson, and N. Weaver, "How to own the Internet in your spare time," in *Proc. 11th USENIX Security Symposium*, Aug. 2002.

[6] Z. Chen, L. Gao, and K. Kwiat, "Modeling the spread of active worms," in *Proc. IEEE INFOCOM*, Mar. 2003.

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

[8] M. Roesch, "Snort: Lightweight intrusion detection for networks," in *Proc. 13th Conf. Sys. Admin.*, Berkeley, CA, Nov. 1999, pp. 229–238.

[9] [Online]. Available: http://www.snort.org

[10] V. Paxson, "Bro: A system for detecting network intruders in real-time," in *Proc. Comput. Netw.*, Amsterdam, Netherlands, 1999, pp. 2435–2463.

[11] [Online]. Available: http://www.icir.org/vern/bro-info.html

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

[13] S. Cheung, R. Crawford, M. Dilger, J. Frank, J. Hoagland, K. Levitt, J. Rowe, S. Staniford-Chen, R. Yip, and D. Zerkle, "The design of GrIDS: A graph-based intrusion detection system," U. C. Davis Computer Science Department, Tech. Rep. CSE-99-2, 1999.

[14] S. Staniford-Chen, S. Cheung, R. Crawford, M. Dilger, J. Frank, J. Hoagland, K. Levitt, C. Wee, R. Yip, and D. Zerkle, "GrIDS-a graph-based intrusion detection system for large networks," in *Proc. 19th National Inf. Sys. Security Conf.*, 1996.

[15] S. Staniford, J. A. Hoagland, and J. M. McAlerney, "Practical automated detection of stealthy portscans," in *Proc. 7th ACM Conf. Comput. and Commun. Security*, 2000.

[16] C. Leckie and R. Kotagiri, "A probabilistic approach to detecting network scans," in *Proc. Network Operations and Management Symposium*, 2002.

[17] A. Sridharan, T. Ye, and S. Bhattacharyya, "Connectionless port scan detection on the backbone," in *Proc. 25th IEEE IPCCC*, 2006.

[18] A. Sridharan and T. Ye, "Tracking port scanners on the IP backbone," in *Proc. Workshop on Large Scale Attack Defense with ACM Sigcomm*, 2007.

[19] J, Mai, A. Sridharan, C.-N. Chuah, H. Zang, and T. Ye, "Impact of packet sampling on portscan detection," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 12, pp. 2285–2298, Dec. 2006.

Sang Kon Kim received the B.S. and M.S. degrees from Hong-Ik University, Seoul, Korea, both in Electrical Engineering and the Ph.D. degree in Electrical and Computer Engineering from Seoul National University, Seoul, Korea. He is currently a part-time Lecturer of Korea University. His current interesting research area is network and computer security for wired and wireless networks, especially with an emphasis on how to detect Network Intrusion based on the attacker's abnormal behavior.

Seung Ho Lee received the B.S. degree in electrical engineering from Seoul National University, Seoul, Korea, in 2007. He is currently pursuing his Ph.D. degree at the school of Electrical Engineering and Computer Science, Seoul National University, Seoul, Korea. His research areas include network security and resource allocation. His most recent research has focused on the areas of network design, resource provisioning, and security in virtual networks.

Seung Woo Seo received the B.S. and M.S. degrees from Seoul National University, Seoul, Korea, both in Electrical Engineering and the Ph.D. degree in Electrical and Computer Engineering from Pennsylvania State University, University Park in USA. He was on the Faculty of the Department of Computer Science and Engineering, Pennsylvania State University, and was a Member of the Research Staff in the Department of Electrical Engineering in Princeton University, Princeton, NJ. In 1996, he joined the Faculty of Seoul National University, where he is currently a Professor in the School of Electrical Engineering. He has served as a Chair or a Committee Member in various international conferences and workshops including INFOCOM, GLOBECOM, PIMRC, VTC, MobiSec, Vitae, etc. He also served for five years as a Director of the Information Security Center in Seoul National University. His research areas include computer & network security, future mobile & wireless networks, and system optimization.