Hindawi Publishing Corporation
EURASIP Journal on Advances in Signal Processing
Volume 2009, Article ID 589413, 13 pages
doi:10.1155/2009/589413

Research Article

An Adaptive Approach to Granular Real-Time Anomaly Detection

Chin-Tser Huang and Jeff Janies

Department of Computer Science and Engineering, University of South Carolina, Columbia, SC 29208, USA

Correspondence should be addressed to Chin-Tser Huang, huangct@cse.sc.edu

Received 29 April 2008; Accepted 23 December 2008

Recommended by Polly Huang

Anomaly-based intrusion detection systems have the ability to detect novel attacks, but when applied in real-time detection, they face the challenges of producing many false alarms and failing to match with the high speed of modern networks due to their computationally demanding algorithms. In this paper, we present Fates, an anomaly-based NIDS designed to alleviate the two challenges. Fates views the monitored network as a collection of individual hosts instead of as a single autonomous entity and uses dynamic, individual threshold for each monitored host, such that it can differentiate between characteristics of individual hosts and can independently assess their threat to the network. Each packet to and from a monitored host is analyzed with an adaptive and efficient charging scheme that considers the packet type, number of occurrences, source, and destination. The resulting charge is applied to the individual hosts threat assessment, providing pinpointed analysis of anomalous activities. We use various datasets to validate Fates ability to distinguish scanning behavior from benign traffic in real time.

1. Introduction

Network-based intrusion detection systems (NIDSs) provide protection to an entire network of computers. These systems detect misuse by examining the packets coming into and out of the monitored network, and they are primarily one of two types: signature-based and anomaly-based. A signature-based NIDS, such as Bro [1] or Snort [2], examines network traffic in an effort to match the patterns of the traffic, or rules, to preestablished patterns of malicious activity. Such systems provide excellent detection capabilities against the known attacks, but require constant update to provide protection from new attack strategies. An anomaly-based NIDS works on the assumption that malicious network traffic is distinguishable from normal network traffic, as discussed in [3]. These systems attempt to quantify the protected network "normal" network traffic and reports deviations from this norm.

Anomaly-based detection has attracted major research interest, since it has the ability to detect novel attack strategies that are often missed by signature-based methods. By understanding and defining what is "normal" in a network, deviations from this norm indicate activities that require further investigation. This method of detection maintains the same level of sensitivity in the presence of novel and classic attack strategies.

Although the capabilities of anomaly-based detection are consistent, this method presents two unique challenges. First, since network traffic can vary wildly and certain traffic patterns are unpredictable, anomaly-based NIDSs run the risk of producing many false positives and false negatives. A false positive is when an NIDS flags benign (though possibly odd) traffic as malicious. Conversely, a false negative is when an NIDS flags malicious traffic as being benign. Second, since modeling the behavior of a network is complex, many proposed systems use computationally demanding algorithms for analysis. Although such algorithms provide the most promise for detection of malicious activity, they run the risk of being too slow to be effective in modern networks, which already achieve speeds of 1000 Mbps (for a complete discussion of this, please refer to [4]).

The system presented here is an anomaly-based NIDS, Fates, which attempts to alleviate the challenges discussed above while maintaining the advantage of detecting novel attacks. The proposed method views the network as a collection of individual hosts as opposed to an autonomous entity. By making such a fundamental view change, Fates has the ability to differentiate between characteristics of

individual hosts and independently assess their threat to the network. Packets to and from a monitored host are analyzed with an adaptive and efficient charging scheme that considers the packet type, number of occurrences, source, and destination. The resulting charge is applied to the individual hosts threat assessment, providing pinpointed analysis of anomalous activities.

2. Related Works

Most current real-time, anomaly-based NIDSs utilize entropy analysis and signal detection techniques. In [5–8], two entropy approaches and one signal detection approach are discussed, respectively. Zachary et al. [5] use an entropy analysis that is tunable to large-scale networks. In the presence of robust scanning, this approach proves to be effective. For instance, in a deployment demonstration this approach detected the beginning of a Code-Red attack [9]. The early warning of this attack allowed the administrators to minimize the impact of the attack, but the exact nature of the attack was unknown until administrators conducted further investigation of network activities. Similar to this approach, Feinstein et al. [7] use a chi-squared approach to distinguish DDoS attacks from other attack strategies, and properly notifies the administrator of the existence of the attack. Alternately, Barford et al. [8] and Thareja [6] propose a distributed signal detection approach to characterize network anomalies. In this approach, normal network traffic is viewed as noise. Using wavelet analysis, the method removes this "noise" in an effort to expose the underlying anomalous activity that would otherwise be indistinguishable. Both of the above approaches are scalable to large-scale networks because they generalize the monitored network to a single entity with a quantifiable "health." It is the aim of these approaches to gain a global perspective by viewing the network in a broad sense.

However, the effectiveness of the approaches specified above may be limited by the following two reasons. First, quantifying a network "health" in a single numerical value does not provide host level granularity. Practical information is lost. For example, in the presence of scanning activity, the scan is detectable, but to find the source of the scanning within a network of hosts, further analysis is required. Second, excluding parallel-computing, real-time processing is not possible in "fast" networks due to the amount of processing load required. Though it is demonstrated that certain approaches maintain a constant level of protection, even under network saturation, it is immature to assume that this level of protection would be sustainable in faster networks, as the system begins to drop packets. Combined, these two reasons suggest that a granular approach with lightweight computation loads is an appropriate next step in advancing anomaly-based intrusion detection to a feasible, economic solution to modern network security.

In an effort to provide both the granularity and the economy of operations that are required in modern networks, Jung et al. [10] propose a threshold random walk (TRW) scheme which assesses the health of the network based on a probabilistic analysis of a packet likelihood of successful delivery. In this approach, a single packet does not result in the labeling of a host as benign or malicious, but analyses of subsequent packets originating from the host add to the assessment to provide an adequate view of the host current state. The system maintains a likelihood ratio for each host until the value falls below a lower threshold (indicating a benign host) or increases above an upper threshold (indicating scanning behavior). This approach has the advantage of being lightweight while providing the ability to distinguish between scanning and benign behavior.

Weaver et al. [4] propose an approximation cache approach which incorporates a simplified TRW scheme. In this approach, the system subdivides a network into autonomous regions. The system examines all hosts within a region in accordance to the host connection history with other hosts. The health of a host is represented by a single integer value, which indicates the number of unacknowledged connection attempts that a host makes. If this value exceeds a predefined threshold, the system disallows any new connection attempts.

Although both [4, 10] utilize a granular view of the network, they both fail to capitalize on its ability to distinguish between varying traffic needs. For instance, it is obvious that a web server and a standard workstation computer would have different network traffic loads and, therefore, a network administrator should not generalize them to have similar traffic patterns, as discussed in [11]. However, since the thresholds in both [4, 10] are static and global, these systems are unable to adequately represent a network of diverse traffic needs.

This research extends the approaches discussed in [4, 10] by incorporating dynamic, individual thresholds for each monitored host. As a result, the simple calculations used to assess the charge for a host provide a method to assess individual host health with little regard to other hosts in the network. In contrast to the static threshold approach, our adaptive approach results in fewer false positives for a benign host with a high-traffic profile, and results in fewer false negatives for a malicious host with a low-traffic profile. Moreover, with the simple calculations used in our approach, we are able to keep the processing load economical and thus meet the high-speed requirements of modern networks.

It is worthwhile to note that there have been many NIDSs proposed in academia and industry, for example, NID [12], Cisco security intrusion detection/prevention system [13], and D-WARD [14]. These systems provide comprehensive monitoring of network traffic, but they generate results that either incur too much overhead, or require further analysis, or need to be matched against some signatures or normal models, and so are not along the same line of our objective of designing an economical real-time anomaly detection system.

3. Overview of Fates System

The Fates of Greek mythology are three goddesses: Clotho the Weaver, Lachesis the Apportioner, and Atropos the Cutter of Thread. They determine the life of mortals by examining the world as a woven tapestry. With each person representing

a thread used in the tapestry, the three goddesses see the tapestry as a collection of individual threads. Likewise, Fates examines the network as a collection of individual entities and does so using three subsystems: a sniffer (Clotho), a measuring unit (Lachesis), and an alarm unit (Atropos). The sniffer, Clotho, is a passive listener that records packets as they enter and leave the network. In a standard network topology, the Fates system would reside on a host positioned between the firewall and the rest of the monitored subnet and maintains a direct link to the firewall. Since the firewall is the only means by which the subnet may communicate with the outside world and the firewall will forward a copy of all traffic passing through it to the Fates system, the sniffer can monitor all incoming and outgoing messages. Similar mechanisms are used in [4, 10, 14–19].

From the perspective of anomaly-based detection, a granular view is the most appropriate for modeling a subnet behavior, as discussed by Weaver et al. [4]. An ideal granular view would be activity analysis of individual host in the monitored network. Since modern computer networks support a variety of systems with unique traffic demands, individual assessments of a host health with regards to only its normal network activity is of greater importance than a comparison with other hosts network activity. For instance, it is obvious that a web server and standard workstation computer would have different network traffic loads. Therefore, an NIDS should not generalize them to have similar traffic patterns, since this would result in an inaccurate analysis, as discussed in [11]. On the other hand, a group of workstations in the same lab should have similar traffic patterns, and therefore, an NIDS can group them together without loss of generality.

In order to appropriately model traffic and support this differentiation between hosts, the Fates system utilizes prior knowledge of the network topology and event management to initialize the system. This is similar to an approach discussed by Jung et al. [20] to aid in distinguishing between flash crowds and DDoS attacks. The Fates system utilizes rudimentary knowledge of the network topology, that is, the IP addresses present in the network. Fates regards each IP address or range of addresses as a separate unit with its own threshold and scoring. By doing so, Fates provides the ability to differentiate between various traffic needs for a variety of hosts that may be present on a subnet.

The Fates system can support any number of protected hosts and any degree of granularity. However, we observe that a reduction in granularity results in less pinpoint accuracy in detection. Thus, there is no claim of perfect scalability to large networks. For example, if an entire/24 subnet is represented as a single entity, Fates aggregates all readings together as one entity. Therefore, Fates reports misbehavior within the subnet but does not specify the IP address of the specific culprit. However, if Fates represents each of the addresses in the subnet with an individual entry, Fates indicates the exact health of the host on an individual level.

The measuring unit, Lachesis, utilizes the granular view in one of two types of detection: external-to-internal monitoring and internal-to-external monitoring. Fates accomplishes both detection types with a two-tier system that consists of an *external scan detection* component (ESD) and an *internal hosts monitor* (IHM) component. The ESD monitors the number of failed connection attempts to a monitored subnet from an outside source. If the number of failed connection attempts from a given source exceeds a predefined threshold, Fates blacklists the external host for a predefined interval of time. The IHM component uses connection classification in order to assess the overall "health" and adjust the dynamic, individual threshold of a specific monitored host. If the dynamic, individual threshold of a monitored host increases continuously for a period of time, then the alarm unit, Atropos, will raise an alarm on that host. Note that there is no given threshold in the alarm unit for raising an alarm because there are different communication needs and traffic patterns in every different network, even the same network at different time; it is up to the network administrator to apply his/her knowledge and experience about the specific network to set an appropriate boundary for raising an alarm. Figure 1 presents the component selection in a state diagram. When Fates processes a packet, it first determines if the source and destination are monitored hosts. If Fates is monitoring either address, the IHM component processes the packet; otherwise, the ESD component processes the packet. After the process completes, the system returns to ready state and awaits the arrival of another packet.

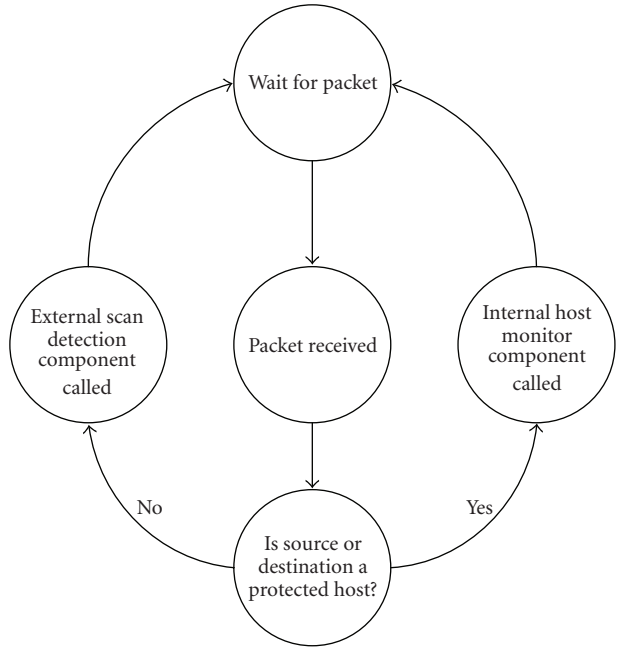

Figure 1: Component selection state diagram.

We give an overview of the Fates system temporal view of the network before discussing each component in more detail. Similar to other NIDS, Fates views its operational time as a series of consecutive intervals of time known as time steps. The time steps used in NIDS are of fixed size and are either overlapping or discrete, as described in [21]. Overlapping time steps provides easier detection of long

patterns of intrusive activity, but has the drawback of the possibility of processing the same packet multiple times. The Fates system uses discrete time steps in order to avoid the skew caused by repeated processing of the same packet, but maintains information about previous readings to analyze the current activity of a system by aggregating readings from previous time steps so that the strength of overlapping time steps is preserved. Fates accomplishes this by associating time-to-live (TTL) values with all readings that are pertinent to interpreting the health of a given host. These TTL values are greater than the size of a time step, but small enough not to cause a great number of false alarms.

The size of time step affects the response time and sensitivity of an NIDS, as discussed in [11, 21]. If the duration of an attack is greater than the size of the time step, the NIDS is at risk of completely missing the attack. However, if the size of the window is too large, the NIDS is at risk of both untimely reporting and greater false alarm rate. For instance, if an NIDS time step size is ten minutes and a Slammer-like intrusion occurs [19] (which has the potential to infect an entire/16 in a matter of seconds), then the worm could infect the entire subnet several times over before the first alert is generated. Moreover, too large a window may aggregate too many readings, resulting in the possibility of an increase in false positives and false negatives. On one hand, if no intrusive activity is present in the traffic, the conglomeration of all readings runs the risk of appearing similar to the signature of an attack, thus resulting in a false positive. On the other hand, if an intrusion is present in the readings for a given time step, the attack may be masked if it concurs with an overwhelming load of benign traffic, thus resulting in a false negative.

Fates does not completely solve the problem of choosing an inappropriate time step size as faced by other such NIDS systems, but Fates does provide the advantage of remembering previous readings. Therefore, smaller time step sizes (less than 30 seconds) are less problematic and preferable if extremely small settings (such as one second) are avoided. Through experimentation, we observe that a time step size of ten seconds provides Fates with adequate detection of rudimentary scanning techniques, such as common scanning worm activity and network mapping tool sets.

3.1. External Scan Detection (ESD) Component. The ESD is the simpler of the two components and is the first line of defense against intrusive activities, and as such, it is paramount that the component incurs low cost in both processing time and memory consumption. Therefore, this component employs a caching system. Furthermore, the ESD only examines the IP header information in a rudimentary fashion.

The ESD mechanism attempts to track the misses of incoming connections. Misses are packets whose source and destination are unmonitored addresses. Since Fates maintains a list of internal addresses, a determination of the existence of a source and destination in the network is simply a lookup. Furthermore, since Fates monitors traffic between the firewall and the rest of the subnet, it is easily inferred that any packet seen by the system should originate from or be destined to a host within the network. Therefore, the existence of a packet that is neither from nor directed to a host in the network is indicative of some kind of miss.

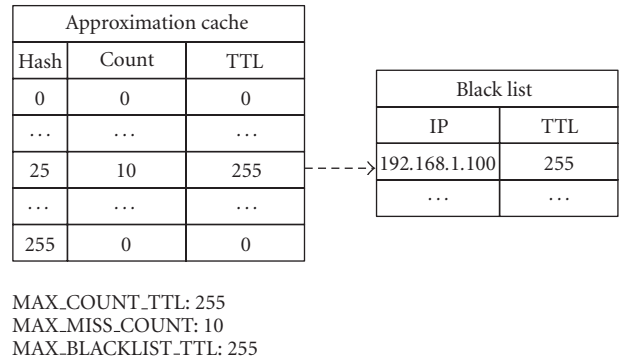

Figure 2: Structures used by the ESD Component.. 192.168.1.100 hashes to entry 25 in the approximation cache.. At the time of the hit, the count at that location was nine. Therefore, the count was incremented and the TTL is set to MAX_COUNT_TTL. Since the count is no longer less than MAX_MISS_COUNT, 192.168.1.100 is added to the black list with a TTL set to MAX_BLACKLIST_TTL.

It is important to note that a single miss is not indicative of scanning. Sometimes hosts misdirect packets to a nonexistent address with no malicious intent. Such cases can arise both from misconfigurations of devices and broken links [22]. Therefore, disregarding the occasional miss is a reasonable reaction. It is large amounts of misses that indicate a problem, such as greater than 50% of a host traffic results in misses, as discussed in [10]. One possible method of disseminating this information would be to track the number of misses for all external hosts that have contacted the monitored subnet. However, keeping track of the number of misses for all addresses is an ill-posed solution to this problem. An adversary can easily overwhelm the system by spoofing a large amount of packets with different source IP addresses. As a result, the system quickly depletes all available storage and, thus, fails to log additional connections or completely fails, as discussed by Weaver et al. [4].

In order to overcome this hurdle, Fates employs the use of an approximation cache, which is similar to the approach used in [4]. An approximation cache is a hash table with no collision avoidance. Simply put, Fates views all entries that hash to the same location as one entry. This approach has the advantages of constant size and quick lookup, but does not provide precision in value of each entry. This tradeoff of precision for speed and constant space is amiable, since both are primary concerns in the development of this component.

As illustrated in Figure 2, the ESD component consists of an outsider count table and a blacklist. When the ESD component processes a packet, it hashes the packets source IP address to a location in the outsider count table, increments the value by one, and sets the TTL of the entry to MAX_COUNT_TTL. If the value exceeds the MAX_MISS_COUNT threshold, Fates adds the offending IP to the blacklist with a TTL value set to the MAX_BLACKLIST_TTL. The MAX_MISS_COUNT,

MAX_COUNT_TTL, and MAX_BLACKLIST_TTL are all user-defined values. The MAX_MISS_COUNT should be set high enough to allow for occasional misses, but low enough to flag offending hosts in a timely manner. When selecting this value, the user should take into consideration the size of the outsider count table. A smaller cache would result in larger count values, since the smaller cache results in a greater frequency of hits to individual locations. Both the MAX_COUNT_TTL and the MAX_BLACKLIST_TTL values should be high enough to dissuade any adversary from attempting to thwart the system, and as such, the maximum value of the field, 255, is an appropriate setting. However, in the case of high-traffic subnets with many constantly changing links, it may be advantageous to allow for a lower value for both in order to compensate for the frequency of misses.

At the expiration of each time step, the TTL values of both the blacklist and outsider count table entries are decremented by one. If an element of the outsider count table TTL reaches zero, Fates resets the count of the element to zero. If a blacklisted element TTL reaches zero, Fates removes the element from the blacklist.

3.2. Internal Hosts Monitoring (IHM) Component. The IHM component is the monitor of all user-specified internal host of the network. This component utilizes both the a priori IP address information provided at initialization and current connection state information to produce an analysis of individual hosts in the network. Prior to active monitoring of the network, the measuring unit acquires a list of active IP addresses (or range of addresses) in the monitored subnet and the minimum thresholds of the host (or range of hosts). The minimum threshold is the lowest sustainable threshold that Fates allows the host to have and uses the minimum threshold to adjust the current threshold of the host.

The IHM component utilizes two structures to represent the monitored hosts and monitor the traffic of the network: the IP_List and IP_Packet_Table. IP_List is a binary search tree in which each element represents a monitored host. An element of the IP_List contains an IP address (or IP range), the current threat score (initialized to 0), the average threat score (also initialized to 0), and a hash table of nodes that is currently in communication with this monitored host (I/OCache). I/OCache is an approximation cache of integers with each integer representing the state of communication between the monitored host and any host whose IP address hashes to that location. In addition to the IP_List structure, Fates stores information on the IP packets previously seen in the IP_Packet_Table. The IP_Packet_Table is an approximation cache indexed by a hash of the packet payload and contains both a time-to-live and occurrence counter for each entry.

When IHM processes an IP packet, it first determines if the upper-layer protocol is connection-oriented, such as TCP/IP, or connectionless, such as UDP. In the case of a connection-oriented protocol, the state of the connection is of primary concern. Since scanning behavior tends to exploit weaknesses in existing protocol structures, there is very little that can be taken for granted. For example, in the TCP/IP protocol a packet with an ACK bit set should only exist in an established connection. However, as is demonstrated by [23], a malicious user can use these packets for scanning purposes. In the case of a connectionless protocol, there is no connection state information to rely on. Instead, the number of packets with duplicate payloads is of importance. The main assertion of such a practice is that scanning behavior will present itself in only a finite amount of possible packet payloads. Most connectionless protocols use only a "best effort" approach for packet delivery, so there should be no duplicate packets of this type in a short amount of time because the source does not retransmit a lost packet. In some cases of UDP applications, retransmission policy is applied and duplicate packets are retransmitted (e.g., [24]), but this happens only when no acknowledgment from the destination is received, which can be distinguished from scanning by checking the destination status.

Table 1: Formulas for packet charge.

Packet type	Formula
TCP	Charge = $2*(state-1)$
UDP	Charge = $2*(count-1)$

In the case of a TCP packet, the IHM component determines whether the packet is destined to or originated from a monitored host and the packet type. This information is used to modify a given host I/OCache entries. If the destination of the packet is a monitored host, the IHM component first finds from the IP_List the element corresponding to the destination address, uses the source IP address to index into the element I/OCache entry, and then subtracts one from the I/OCache entry current value (conversely, if the source of the packet is a monitored host, add one to the corresponding I/OCache entry). The IHM component then assesses a charge for the packet using the entry resulting value. The formula for calculating this charge is shown in Table 1. If the value of the entry is less than or equal to zero, the state is set equal to zero and the host is not assessed a charge because the host is receiving more communications than it is transmitting, that is, not scanning behavior. If the value of entry is greater than zero, the state is set equal to the entry value. The reason for the multiplication of the state information by two is to provide a quick jump in charges in the presence of persistent unacknowledged outgoing messaging. If the anomalous scanning behavior is viewed as the signal of interest, then the multiplication will effectively strengthen the signal and make it easier to detect. We choose two as the multiplier because it is the smallest integer larger than one (involving floating point will make the calculation less efficient). As will be seen in our experimental results, this multiplier serves its purpose quite well, so there is no need to use a larger multiplier. In a standard three-way handshake and packet transmission (the destination transmits an ACK for each message received), the monitored host receives a net charge of zero. We note that some receivers may implement a delayed acknowledgment mechanism [25, 26], in which the receiver sends an acknowledgment only

for every second received message for performance reason. However, our experimental results show that even this delayed acknowledgment mechanism is still distinguishable from scanning behavior with our approach and the chosen multiplier.

In the case of a UDP packet, the packet payload is of importance because there is no connection information associated with the protocol. When the IHM component processes a UDP packet, it uses the payload of the packet to index the IP_Packet_Table, increments the entry count value by one, and sets the TTL of the entry to 255. If the source of the packet is a monitored host, the IHM component then assesses the host a charge. As Table 1 shows, the charge is simply two times the count value minus one. Note that an arbitrary nonduplicate packet would result in no charge.

In the case of any other protocol, Fates skips the packet. Though this may be inappropriate in certain settings, the design of Fates is for standard practice. It is arguable that ICMP [27] should be processed. However, since this packet type is connectionless and is used for control and testing purposes, there is a risk of skew in processing. For instance, ping, a widely used mechanism for determining connectivity of a host, sends echo request messages to a user-specified destination. In many cases, these packets are identical with regard to payload, and therefore, result in the IHM component immediately flagging any host issuing a ping request as malicious. Therefore, the ambiguity of circumstance necessitates the absence of this protocol from analysis.

At the expiration of the current time step, the IHM component assesses the health of all monitored hosts and prepares for the next time step. First, the IHM component calculates the cumulative charge for all packets for each host seen during the current time step, resulting in a threat score for the host. The IHM component compares the threat score to the current threshold of the host. If the threat score exceeds the current threshold, the IHM sets the threshold equal to the threat score and makes a note of the change in a log file. If the threat is less than the threshold, the IHM component compares the threshold with the minimum threshold. If the values are equal, the IHM component takes no action. In all other cases, the IHM component uses a threshold adjustment scheme. Note that a threshold is easily increased but further analysis is required to determine if the threshold should be lowered. The principle idea is that the component attempts to ascertain an appropriate upper bound of a host activity. A well-behaved host threshold will plateau, but a scanning host activity constantly causes the host threshold to increase. After the IHM component adjusts the thresholds of each host, it then prepares for the next time step by resetting the threat score to zero, decreasing the TTL of each entry in the I/OCache by one, and decreasing the TTL of all elements in the IP_Packet_Table by one. If the TTL of an entry in the IP_Packet_Table is equal to zero, the IHM component sets the count of the entry to zero.

3.3. Aggregation of Readings. In order to address the issue of decreasing threshold, the IHM component uses the weighted average of previous readings to understand the current state of the host. The averaging method used is as follows:

$$S = (1 - \alpha)S_{\text{current}} + \alpha S_{\text{new}}, \tag{1}$$

where S is the weighted average score, α is a preset value for the decay of old readings, S_{current} is the previous weighted average score, and S_{new} is the threat of the host in the current time step. This is similar to TCP roundtrip time (RTT) estimation as discussed in [28], which provides an efficient way to calculate a weighted average of readings. The IHM component gives older readings less weight than new value, and thus, while past readings still affect the result, their impact lessens over time. Therefore, the formula encompasses both an implied time-to-live for charges against a host and a contextual analysis of a network host status at present. In practice, the value of α should range between 0.5 and 0.75 because any value less than 0.5 places too much emphasis on previous readings and rarely allows the threshold to be redeemed, and any value greater than 0.75 places too much emphasis on the current readings and runs the risk of prematurely lowering a threshold.

With this averaging, the IHM component can compare a host current threat level to its previous activity, assess the duration of anomalous activity, and scale changes to thresholds. With simple comparisons, the weighted average provides an analytical tool for assessing the speed at which a host activity is changing. This is useful in assessing cases of flash crowd and DoS attack, where network activity from one or many hosts increases rapidly, as discussed in [16]. The duration of the activity is a key component in determining the "X factor" that initiates the malicious activity. As time progresses, sustained rates of activity cause the weighted average to approach the current score, at first very quickly, then slowly until the values are equal. Analysis of the resulting curve allows for accurate interpretation of the time interval in which the malicious activity in question really began, which greatly aids in forensic analysis. However, such an analysis is beyond the scope of this paper, and the IHM component is focused on providing a method to interpret network information for tuning a threshold, as discussed next.

3.4. Threshold Adjustment. As previously stated, the IHM component is quick to raise a host threshold but lowering the threshold requires further analysis of both current state of the host activities and its previous activity. IHM attempts to find equilibrium for each host activity. Quickly redeeming charges possesses two important risks. First, it provides no stable ground on which to base assessments about the health of a host. If the threshold is not allowed to plateau, the system provides no solid ground upon which an administrator can make decisions. Second, allowing the threshold to drop quickly could cause the masking of malicious activity. As will be seen in Section 4, certain normal network activities cause dramatic changes in the threshold, but the system quickly returns to normal, while scanning activities cause lasting and continual changes to the thresholds, resulting in obvious distinctions from normal host behavior.

In the IHM component threshold adjustment, the threshold will remain the same until being exceeded by a host score. Once a host score exceeds the host threshold, the value of the host threshold will increase to the score that exceeded it. For every time step afterward, if the weighted average score of the host is lower than the minimum threshold, then the threshold value decreases by half of the difference between the minimum threshold and the weighted average score until it reaches the minimum threshold value. The formula for this threshold adjustment is as follows:

$$T = T_{\text{current}} - \frac{(T_{\min} - S)}{2}, \tag{2}$$

where T_{current} is the current threshold value, $T_{\min}$ is the initial threshold value of the host, and S is the current score of the host. After experimentation with the values of S, it was found that this formula has a redemptive quality for a previously ill-behaved host but requires an adequate number of time steps before the threshold returns to its minimum value.

4. Experimentation

We have implemented a version of the Fates system in C++ programming language. It utilizes both the libpcap and the pthread libraries. Libpcap is a library that facilitates quick and easy setup of network sockets for packet capture and provides several functions for analysis of TCPdump files. The pthread library provides functions for parallel processing using forks.

We test the Fates system on several different datasets in order to understand how the system functions under environments with different characteristics. The datasets presented here are the Slammer simulation package, the University of South Carolina (USC), Department of Computer Science and Engineering subnet traffic, a World of Warcraft (WOW) traffic set, and a dataset consisting of file sharing traffic. Some of the experiments have been briefly discussed in our previous paper [29]. Each of these datasets has unique characteristics and provides an adequate cross-section of network activity to test Fates various components. The Slammer simulation tests the UDP charging scheme. The USC subnet traffic tests the TCP/IP charging scheme. The WOW traffic set tests the false positive rate of the system when presented with traffic that exhibits packet loss due to congestion at an endpoint. The file sharing dataset presents benign traffic closely resembling malicious scanning behavior and is used to determine the Fates capabilities of distinguishing between the two.

Before we get into the details and results of each experiment, we discuss the proper selection of the user-defined values of Fates. These values are all dependent on the size of the network and the services the hosts use. For example, in a network consisting of only one hundred workstation computers with static IP addresses, there is little need for a large IP_Packet_Table size, outsider count table size, or I/O approximation cache size. However, if the subnet consists of several web servers, each of these items would require larger values, since the traffic load for the network is large.

We discuss the configuration of Fates suitable for a subnet consisting of less than 100 workstation computers and no web, SMTP, or DHCP servers. Through experimentation, we observe that a time step size of ten seconds provides adequate detection of rudimentary scanning techniques, such as common scanning worm activity and network mapping tool sets. For the ESD component, this time step size necessitates a larger outsider count table size, since ten seconds worth of network traffic represents a large number of packets, each of which has the possibility of affecting the outsider count table. An outsider count table size of 255 is appropriate; it provides enough entries to represent the outside world, but not enough to hide scanning behavior. For the IHM component, the I/OCache for each element does not need to be very large, say 64 entries, because the threshold adjustment will allow for a slight forgiveness of the small size, since Fates attempts to establish equilibrium with the thresholds. The size of the IP_Packet_Table and the TTL associated with its entries directly correlate to the bandwidth consumption of the network. In the case of a network that maintains constant low rates of bandwidth consumption (less than or equal to 10 Mbps), a size of 255 entries with a TTL of three time steps is enough to give an adequate representation of the network activity. However, if the network bandwidth consumption is greater than 10 Mbps, the size of the table should be doubled. The value of α used in the weighted averaging of a host cumulative charge is set to 0.5, and the initial threshold value of a host, $T_{\min}$, is set to 1000.

4.1. Slammer. The Slammer worm [19] was one of the most infectious worms to plague computer networks. Though per capita, it was far less nefarious than the Morris worm [30], its rate of infection and simplicity of design is something to be both admired and feared. Within three hours of its introduction, the worm had infected almost all susceptible computers running an unpatched version of Microsoft SQL Server (see [19]).

In an effort to test Fates, we developed a simulation package that provides a variety of configuration options and allows for fast generation of a multitude of datasets. This simulation package attempts to simulate the slammer worm infectious nature and provides a good alternative to real-world data by both being completely modifiable and lacking the legal entanglements normally associated with the capture of real-world data.

The simulation package functions as a packet generator and TCPdump merger. It takes for input a TCPdump of background, or presumed benign, traffic for input, and merges the data contained within with simulated results from a slammer infection. Therefore, the resulting file contains malicious traffic hidden within the benign. The file maintains all of the advantages of TCPdump files and is otherwise indistinguishable from an actual capture log. We ran this simulation against the Fates system in an effort to test the UDP charging scheme. The simulated data consisted of two IP addresses 192.168.1.101 and 192.168.1.103 that were monitored for 10 minutes (this time is a bit excessive

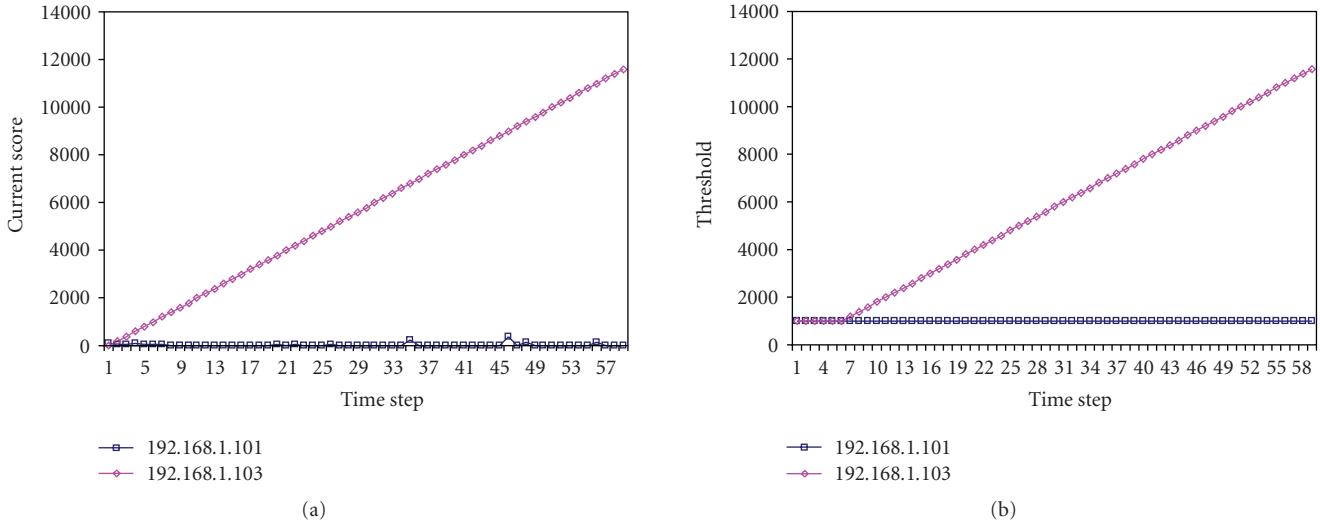

Figure 3: Slammer simulation (with a propagation delay of 1 second).

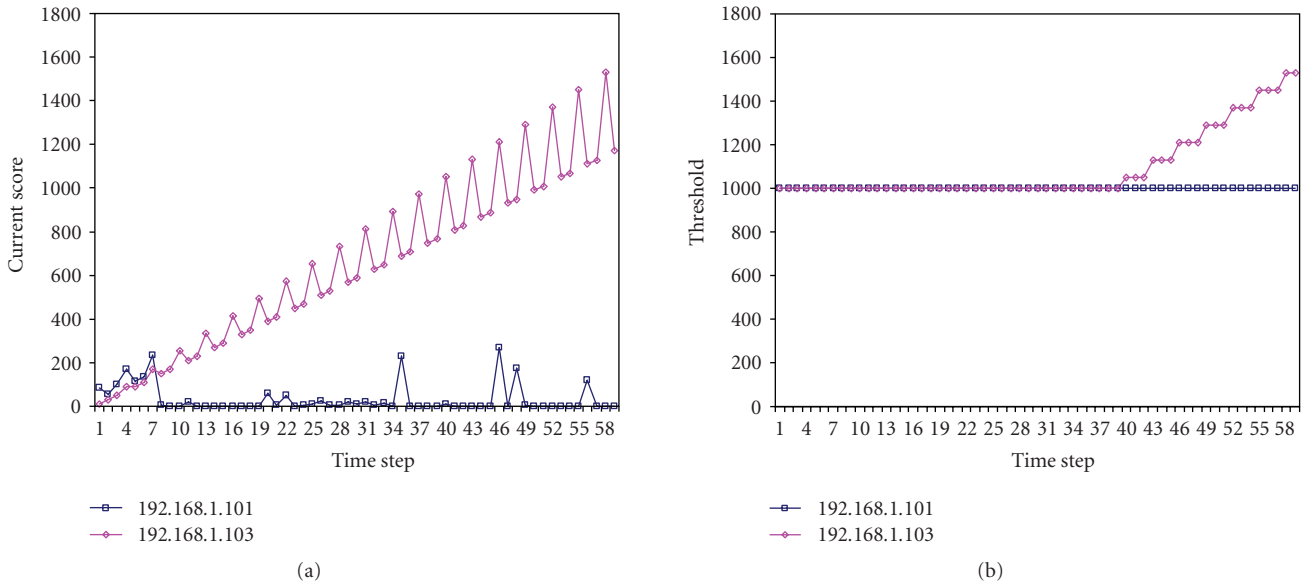

Figure 4: Slammer simulation (with a propagation delay of 3 seconds).

since the worm was actually detected in only 30 seconds). 192.168.1.103 is an infected host that is attempting to propagate the slammer infection, and 192.168.1.101 is a host that is running 20 minutes worth of web traffic, a video stream, and an SSH connection. For the purposes of this simulation, the rate at which the worm propagates is one second. This rate is far slower than the actual Slammer worm, which only aids in hiding the signature of the worm. However, as can easily be seen in the graph provided below, not only Slammer is easily detected, but the well-behaved node threshold remains static throughout the monitoring time.

In Figure 3, the first graph plots charges assessed for each host by the Fates system, and the second graph is the plot of the threshold at every time step. As can be seen, the additive nature of the algorithm does not result in any form of reduction in charges or the threshold for the infected host. However, this additive charging also results in no increase in the charges and threshold of the well-behaved host that is running web traffic. Because of the infected host charges, the threshold constantly increases in a linear fashion throughout the duration of the experiment.

The trend of the worm to increase a host threshold at a steady rate is a factor of its propagation method as opposed to the time associated with the propagation. As Figure 4 demonstrates, if the delay between propagation attempts is limited to three seconds (a value far lower than even TCP/IP worm propagations), the same trend in behavior is still exhibited. Although the increase is not linear as in the previous example, we observe a steady increase in

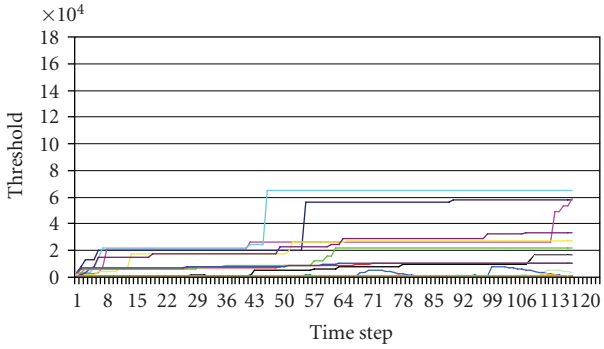

Figure 5: USC traffic threshold analysis (clean).

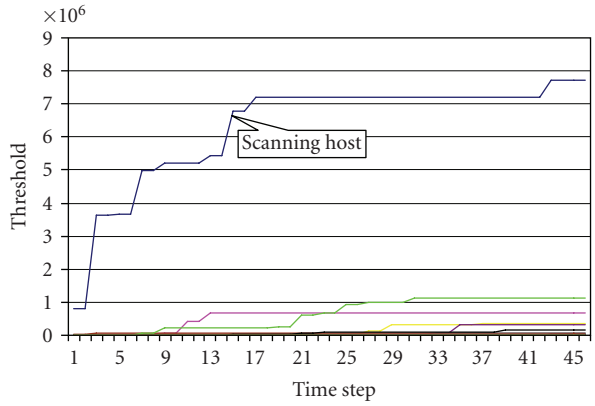

Figure 6: USC traffic threshold analysis (half-open scan).

the threshold. Another feature which is apparent in this experiment is a series of peaks in the cumulative charges of the infected host. This is a direct result of the duration between successive attempts at propagation. The lulls result in a steady decrease in current charge for a malicious packet, but this decrease is soon reversed by the continued effort of the host to propagate duplicate malicious packets.

4.2. USC Traffic. Next, we test the Fates system capabilities with regard to TCP/IP scanning methods in a real network environment. The University of South Carolina Department of Computer Science and Engineering is gracious enough to allow for managed data collection from their subnet. This network consists of eight/24 subnets divided over administrative offices (containing an SMTP server), research labs, and open public labs. There are approximately one thousand hosts on the network which is divided into 37 monitored ranges. The subnets are at most half populated, and the variety of the traffic present on the systems gives a diverse sensing environment.

In order to test the ability of the system to detect scanning behavior in the presence of real-world network traffic, we employed standard scanning techniques supplied in Nmap [31] network mapping software that probes for available ports on a host (or range of hosts). In standard operation, Nmap first attempts to ping all hosts in the subnet. If a host in the subnet responds, Nmap runs a user-specified scanning technique on all active ports for a host. If there is no response from the ping, Nmap attempts to locate hosts by scanning port 80 for all possible hosts in the target range. If the scan of port 80 locates hosts, Nmap runs the user-specified scanning technique on all ports of the active hosts. This method of host discovery provides the advantage of time because it limits the number of hosts that it scans to only those that truly exist.

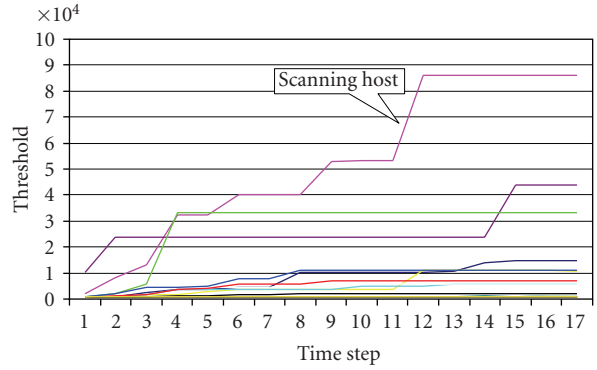

Figure 7: USC traffic threshold analysis (ACK scan).

In order to examine the detection capabilities of the Fates system, we validate results against Snort [2], a widely utilized and respected NIDS system. The Snort system utilizes a rule-based analysis of network traffic and is completely configurable. Snort has a default setting, for which it flags a source IP address that has sent connection requests to 5 different IP addresses within 60 seconds or has sent to 20 different ports of the same IP address within 60 seconds. Our aim is to detect everything that this default Snort setting detects for comparison purpose.

Prior to testing scanning behavior, we establish a baseline of normal network activity as shown in Figure 5. This baseline reflects the normal activity of the network in absence of scanning. There are 37 entries in Figure 5, representing the 37-monitored ranges. As is seen in this figure, all entries reach equilibrium and plateau very quickly. Also, note that the modifications in the threshold of benign activity result in sharp jumps as opposed to the steady increases in the Slammer simulation. The presence of these sharp jumps and plateaus indicates that the system is adjusting to a current and steady bandwidth demand, and not to consistent missing behavior. Therefore, these sharp jumps indicate normal operation, and thus are distinguishable from native scanning behaviors.

After a satisfactory establishment of normal network traffic modeling, we introduced several scans into the network. Figures 6, 7, 8, and 9 describe the resulting thresholds present in the network. The first of these scans is the half-open scan. As is seen in Figure 6, a steady increase in the threshold is present. At time step 16, the threshold plateaus. This is a result of steady connections to active ports, as opposed to connection attempts to closed ports. However, the scanning activity presents itself very clearly as compared to the benign traffic that surrounds it. Next, we ran an ACK scan. As is seen in Figure 7, this behavior presented itself very clearly also with a steady increase in the threshold. Even though the

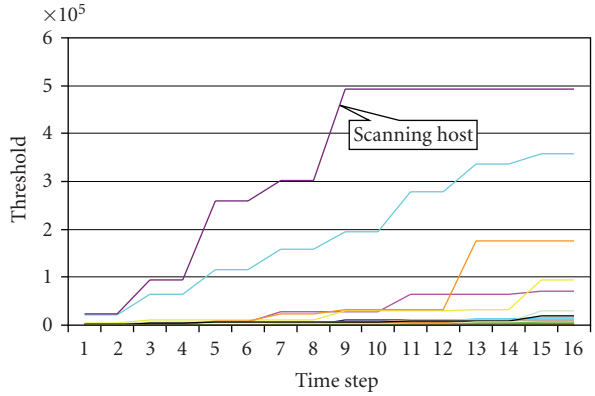

Figure 8: USC traffic threshold analysis (FIN scan).

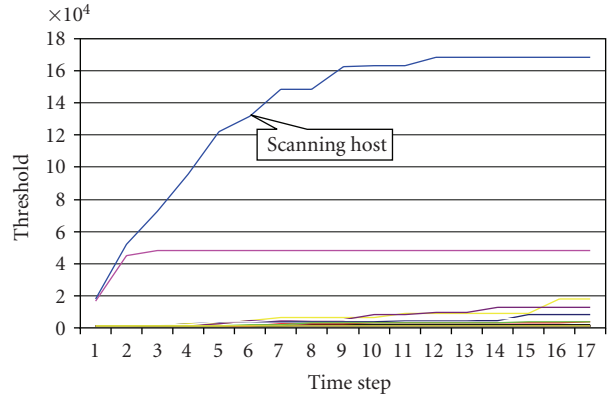

Figure 9: USC traffic threshold analysis (RST scan).

increase is not as much as is seen in the half-open scan, the increase is observable and distinct from the benign traffic. Then, we ran a FIN scan, which is demonstrated in Figure 8. Once again, the scanning entity presented itself in a steady increase. However, the most interesting part of this graph is not the sharply increasing threshold of the host conducting an FIN scan but the second lowest host that is presumably benign. After further analysis, we determined the behavior to be an RST scan of port 22, SSH, which was an actual attack underway in the network. Figure 9 is a representation of the behavior.

In all of the above examples, the magnitude of benign traffic does not obscure the scanning behavior. Instead, it provides comparative information that makes the steady increase in the threshold obvious to the user. From these examples, we can derive the conclusion that for Fates, standard scanning behavior is distinguishable from benign activity.

4.3. World of Warcraft Traffic. World of Warcraft (WOW) [32] is a massively multiplayer online role playing game (MMORPG) that utilizes several servers to provide a large world feel to the game. In the online world, each server represents a localized area, which can be as large as a continent or as small as a city. When a user enters an area, the client registers with the server that handles the specific area. All users in the area utilize the same server, thus allowing for ease of communication with one another and collaboration in game play. With a total number of approximately 1.5 million users, there is a large probability of packet loss due to congestion, since simultaneous use can cause a great amount of congestion on the servers. According to the investigation results in [33], the packet loss rate at the server could be as high as 27%, and 3% of the sessions had a loss rate 5%. Such packet loss rate is enough to induce noticeable latency [34].

This traffic is a perfect example of abnormal but benign traffic, since such activity has high rates of TCP/IP traffic, where packets are lost due to congestion. Since the Fates system utilizes the two-way communication between hosts to assess a host state, this could result in false positives. However, since TCP/IP ensures the delivery of data, the loss of a packet does not hinder communication. Our conjecture is that with a low number of lost packets in a connection, the thresholds will not demonstrate the same behavior as is seen in scanning. We will demonstrate this by examining WOW traffic.

The data used for the test is represented in a TCPdump file containing approximately 20 minutes of various web traffics, including video streaming, HTTP traffic, and WOW traffic. The host at address 192.168.2.19 is running the WOW traffic that is of concern. Each of the other hosts is producing low levels of HTTP traffic, which cause no false alarms. As Figure 10 indicates, there are no spikes that result in premature adjustment of the threshold. Even more interesting, in time step 120 the user experienced a massive lag. This is because of the user transferring to an already taxed WOW server. Even this did not cause a jump in the threshold because the host still maintained a connection in which two-way communication (though bottlenecked) persisted.

According to the above analysis, it is clear that the loss of a moderate number of packets in a connection does not result in an unwarranted increase in a host threshold. In addition, the spikes in network traffic resulting from transition to new WOW servers fail to exceed even the initial threshold. Though it is arguable that a connection to several servers, each of which resulting in unacknowledged messages and packet retransmissions, would result in a problematic circumstance in which a host would have a threshold increase, the readings obtained indicate that this loss of packets occurs in a bursty fashion. Therefore, the threshold increase would not present itself as a steady increase in the threshold. Instead, a series of sharp jumps in the threshold (similar to those found in the clean USC data) result. Thus, we can conclude that for Fates, this type of abnormal but benign traffic is distinguishable from scanning behavior.

4.4. File Sharing Traffic. The last type of traffic we examine is traffic that closely resembles scanning behavior and thus is problematic. A classic example of this is peer-to-peer networks. Peer-to-peer programs present a unique set of challenges in NIDS research. The programs used to establish such networks often use scanning to locate peers, and most detection systems fail to distinguish between this form of

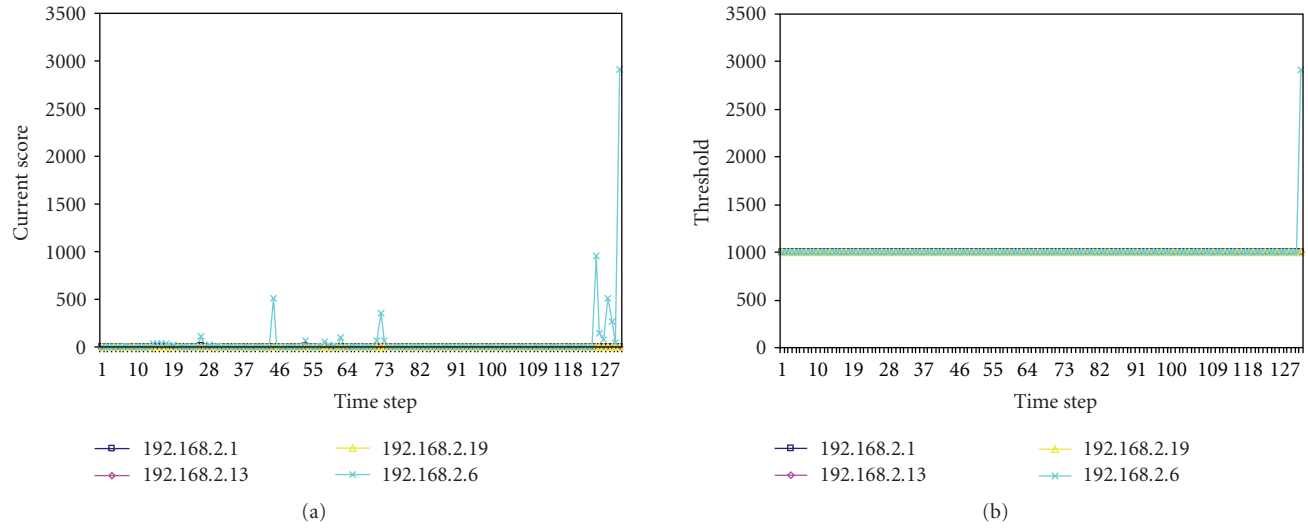

Figure 10: WOW traffic analysis.

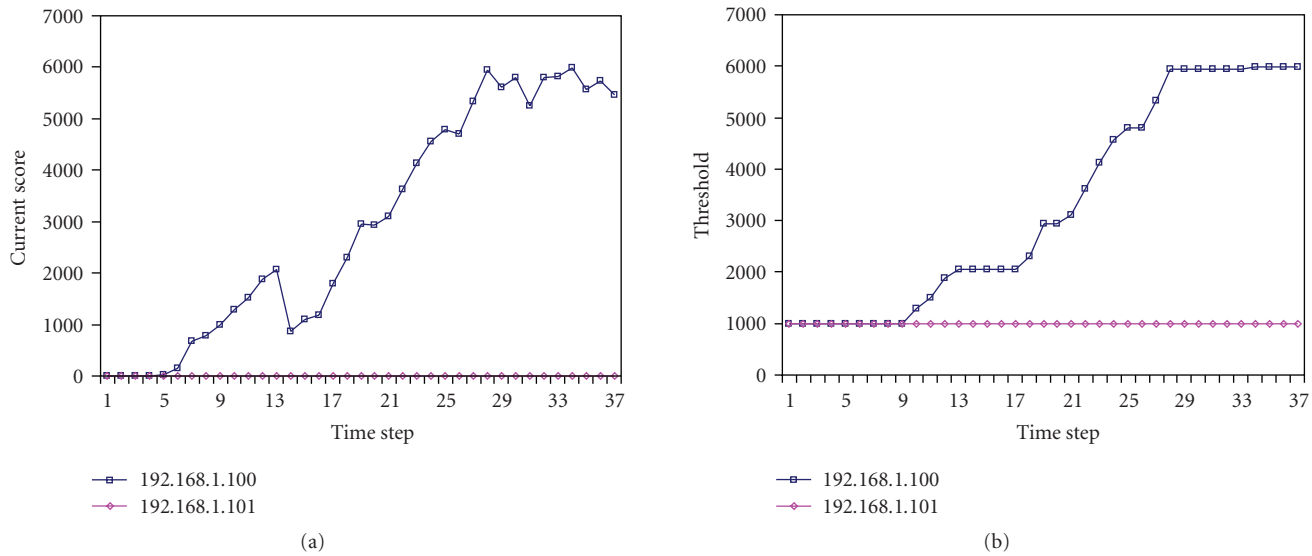

Figure 11: Emule traffic analysis.

benign scanning and malicious scanning. Note that in this context, "benign" is limited to meaning that the user is not attempting to compromise hosts, though many peer-to-peer networks allow file transfer of copyrighted material, which is illegal.

To compare Fates with this form of traffic, we used a TCPdump of a host running an Emule [35] client. Emule is a widely used program that provides a variety of services and connection capabilities ranging from IRC chat to file transfer. The program primarily functions as a client, which will connect to a preestablished list of servers. The servers function as a centralized service coordinator for all peers in the network. However, these servers are not static and the list constantly changes. Thus, clients often attempt connections to servers that have since moved or are completely shutdown. Because of this, the client behavior often resembles scanning in refined address space (i.e., the scanning is limited to the list of servers that it currently possesses). To further compound the problematic nature of the traffic produced by the client, the users have the ability to interrupt transfer attempts, often resulting in an abnormally large volume of reset connections and failed connection attempts compared to normal TCP/IP traffic.

As illustrated in Figure 11, the traffic recorded is quite similar to the scanning behavior exhibited in the USC dataset. This is a result of the client attempts to connect to several unavailable servers and several failed connection attempts to peers. These failed connection attempts are a direct result of a user denial of a file transfer. The momentary dips in the host score are the result of connections that were established and productive in transferring data (actual file transfers). However, the existence of such connections does

not prevent the failed connection attempts and interrupted transfers from increasing the threshold.

Since Fates only considers the success of connections, the benign file transfer and scanning produced by Emule results in an incorrect labeling of the host. However, the only way to distinguish between this behavior and truly malicious scanning behavior would be the analysis of the context intended by the user. Such an analysis falls outside the scope of this work, and therefore, this distinction remains an open question.

5. Concluding Remarks

The Fates system exploits the advantages of a granular view by allowing for precision detection of network activity while also maintaining an economy similar to [4]. The system allows for dynamic, self-healing thresholds that allow for both forgiveness of misconfiguration and scaling to current network conditions. However, this scaling does not hinder the system in detecting rudimentary scanning behavior in monitored hosts. Furthermore, the Fates system uses simple calculations, unlike the entropy-based systems, such as [5–7]. As a result, the functionality of the Fates system is appropriate for real-time detection.

As is seen in Section 4, Fates provides easy analysis of the current state of a network with regard to scanning behavior. Furthermore, Fates does not falter in the presence of lost acknowledgments. Instead, it tolerates occasional packet losses without instantaneous flagging of the host as malicious.

There are still open issues under investigation. First, the issue of scalability is unresolved. Fates is not intended for deployment across a diverse/8 network. As such, it is intended to be a lightweight approach that better serves a small- to medium-sized business environment. Second, the output of the Fates system is comma-delineated text files, which both require postprocessing and are resource consuming. One possible solution would be to incorporate this system into an already existing system such as Snort, which has its own established reporting mechanisms. However, in order to do this, a rate of change analysis is necessary to automate flagging. Although these issues provide for further avenues of investigation, the fact remains that the Fates system both adequately interprets current network conditions and distinguishes between benign traffic and basic scanning behavior in a user notable manner.

References

[1] V. Paxson, "Bro: a system for detecting network intruders in real-time," in *Proceedings of the 7th Annual USENIX Security Symposium*, San Antonio, Tex, USA, January 1998.

[2] "Snort: The Open Source Network Intrusion Detection System," http://www.snort.org/.

[3] D. Denning, "An intrusion detection model," in *Proceedings of IEEE Symposium on Security and Privacy*, pp. 119–131, Oakland, Calif, USA, April 1986.

[4] N. Weaver, S. Staniford, and V. Paxson, "Very fast containment of scanning worms," in *Proceedings of the 13th Conference on USENIX Security Symposium*, pp. 29–44, San Diego, Calif, USA, August 2004.

[5] J. Zachary, J. McEachen, and D. Ettlich, "Conversation exchange dynamics for real-time network monitoring and anomaly detection," in *Proceedings of the 2nd IEEE Information Assurance Workshop*, pp. 59–70, Charlotte, NC, USA, April 2004.

[6] S. Thareja, *A real time network traffic wavelet analysis*, M.S. thesis, Department of Computer Science and Engineering, University of South Carolina, Columbia, SC, USA, 2005.

[7] L. Feinstein, D. Schnackenberg, R. Balupari, and D. Kindred, "Statistical approaches to DDoS attack detection and response," in *Proceedings of the 3rd DARPA Information Survivability Conference and Exposition (DISCEX '03)*, vol. 1, pp. 303–314, Washington, DC, USA, April 2003.

[8] P. Barford, J. Kline, D. Plonka, and A. Ron, "A signal analysis of network traffic anomalies," in *Proceedings of the 2nd Internet Measurement Workshop (IMW '02)*, pp. 71–82, Marseille, France, November 2002.

[9] R. Danyliw and A. Householder, "CERT® Advisory CA-2001-19 "Code Red" Worm Exploiting Buffer Overflow in IIS Indexing Service DLL," January 2002, http://www.cert.org/advisories/CA-2001-19.html.

[10] J. Jung, V. Paxson, A. W. Berger, and H. Balakrishnan, "Fast portscan detection using sequential hypothesis testing," in *Proceedings of IEEE Symposium on Security and Privacy*, pp. 211–225, Berkeley, Calif, USA, May 2004.

[11] T. H. Ptacek, "Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection," January 1998, http://insecure.org/stf/secnet_ids/secnet_ids.html.

[12] "Computer Security Technology Center, Lawrence Livermore National Laboratory, Network Intrusion Detector Overview," http://www.doecirc.energy.gov/ciac/ConferenceProceedings/DOECompSec97/nid.pdf.

[13] "Cisco Secure Intrusion Detection System," http://www.cisco.com/univercd/cc/td/doc/product/iaabu/csids/index.htm.

[14] J. Mirkovic, G. Prier, and P. Reiher, "Attacking DDoS at the source," in *Proceedings of the 10th IEEE International Conference on Network Protocols (ICNP '02)*, pp. 312–321, Paris, France, November 2002.

[15] H. Hajji, B. Far, and J. Cheng, "Detection of network faults and performance problems," in *Proceedings of Internet Computing (IC '01)*, Las Vegas, Nev, USA, June 2001.

[16] A. Hussain, J. Heidemann, and C. Papadopoulos, "A framework for classifying denial of service attacks," in *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pp. 99–110, Karlsruhe, Germany, August 2003.

[17] A. Lakhila, M. Crovella, and C. Diot, "Mining anomalies using traffic feature distributions," in *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pp. 217–228, Philadelphia, Pa, USA, August 2005.

[18] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson, "On the self-similar nature of Ethernet traffic," *IEEE/ACM Transactions on Networking*, vol. 2, no. 1, pp. 1–15, 1994.

[19] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver, "Inside the slammer worm," *IEEE Security & Privacy*, vol. 1, no. 4, pp. 33–39, 2003.

[20] J. Jung, B. Krishnamurthy, and M. Rabinovich, "Flash crowds and denial of service attacks: characterization and implications for CDNs and web sites," in *Proceedings of the 11th International World Wide Web Conference (WWW '02)*, pp. 252–262, ACM Press, Honolulu, Hawaii, USA, May 2002.

[21] K. Tan, K. Killourhy, and R. Maxion, "Undermining an anomaly-based intrusion detection system using common exploits," in *Proceedings of the 5th International Symposium on Recent Advances in Intrusion Detection (RAID '02)*, pp. 54–73, Zurich, Switzerland, October 2002.

[22] R. Pang, V. Yegneswaran, P. Barford, V. Paxson, and L. Peterson, "Characteristics of internet background radiation," in *Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement (IMC '04)*, pp. 27–40, Taormina, Italy, October 2004.

[23] D. Kewley, J. Lowry, R. Fink, and M. Dean, "Dynamic approaches to thwart adversary intelligence gathering," http://www.bbn.com/resources/pdf/DISCEX_DYNAT.pdf.

[24] P. Mockapetris, "Domain names—implementation and specification," RFC 1035, November 1987.

[25] R. Braden, "Requirements for Internet Hosts—Communication Layers," RFC 1122, October 1989.

[26] M. Allman, "On the generation and use of TCP acknowledgments," *Computer Communication Review*, vol. 28, no. 5, pp. 4–21, 1998.

[27] J. Postel, "Internet Control Message Protocol," RFC 792, September 1981.

[28] V. Paxson and M. Allman, "Computing TCP's Retransmission Timer," RFC 2988, November 2000.

[29] J. Janies and C.-T. Huang, "Fates: a granular approach to real-time anomaly detection," in *Proceedings of the 16th International Conference on Computer Communications and Networks (ICCCN '07)*, pp. 605–610, Honolulu, Hawaii, USA, August 2007.

[30] Ricochet Team Server Security, "Internet Worms: Self-Spreading Malicious Programs," http://www.mcafee.com/us/local_content/white_papers/wp_ricochetbriefworms.pdf.

[31] NMap, http://www.insecure.org.

[32] World of Warcraft Communicty, http://www.worldofwarcraft.com.

[33] K.-T. Chen, P. Huang, G.-S. Wang, C.-Y. Huango, and C.-L. Lei, "On the sensitivity of online game playing time to network QoS," in *Proceedings of the 25th IEEE International Conference on Computer Communications (INFOCOM '06)*, pp. 1–12, Barcelona, Spain, April 2006.

[34] K.-T. Chen, C.-Y. Huang, P. Huang, and C.-L. Lei, "An empirical evaluation of TCP performance in online games," in *Proceedings of the International Conference on Advances in Computer Entertainment Technology (SIGCHI '06)*, Hollywood, Calif, USA, June 2006.

[35] Emule, http://www.emule-project.net/home/perl/general.cgi?l=1.