Fates: A Granular Approach to Real-Time Anomaly Detection

Jeff Janies, Chin-Tser Huang
Department of Computer Science and Engineering
University of South Carolina
janies@cse.sc.edu, huangct@cse.sc.edu

Abstract— **Anomaly-based intrusion detection systems have the ability of detecting novel attacks, but in real-time detection, they face the challenges of producing many false alarms and failing to contend 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 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's 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's ability to distinguish scanning behavior from benign traffic in real time.**

Index Terms—**Network-based Intrusion Detection System, Anomaly-based Detection**

I. Introduction

An anomaly-based Network-based Intrusion Detection System (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's "normal" network traffic and report deviations from this norm. Anomaly-based detection has attracted major research interest, since it has the ability to detect novel attack strategies often missed by signature-based methods such as Bro [16] or Snort [21]. 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 positive and false negative alarms. 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 1000Mbps (for a complete discussion of this, please refer to [23]).

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. Fates 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's type, number of occurrences, source, and destination. The resulting charge is applied to the individual hosts' threat assessment, providing pinpointed analysis of anomalous activities.

II. Related Works

Most current real-time, anomaly-based NIDSs utilize entropy analysis and signal detection techniques. In [25], [5], [1] and [22] two entropy approaches and one signal detection approach are discussed respectively. Zachary et al. [25] 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 [2]. 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. [5] uses 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. [1] and Thareja [22] 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's "health" in a single numerical value does not provide granularity. For example, in the presence of scanning activity, the scan is detectable, but to find the source of the scanning, 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. 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. [9] propose a Threshold Random Walk (TRW) scheme, which assesses the health of the network based on a probabilistic analysis of a packet's 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's 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. [23] 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's 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 [9] and [23] 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 [19]. However, since the thresholds in both [9] and [23] are static and global, these systems are unable to adequately represent a network of diverse traffic needs.

This research extends the approaches discussed in [9] and [23] by incorporating dynamic, individual thresholds for each monitored host. As a result, the simple calculations used to assess the charge for a host provides a method by which to assess individual host's health with little regard to other hosts in the network. Moreover, in doing so, we are able to keep the processing load economical and thus meet the high speed requirements of modern networks.

III. 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. Similar mechanisms are used in [6], [7], [9], [11], [12], [13], and [23].

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. [8] to aid in distinguishing between flash crowds and DDoS attacks. The Fates system utilizes rudimentary knowledge of the network topology, i.e. 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.

The measuring unit, Lachesis, utilizes the granular view in internal-to-external monitoring. This is achieved with an internal hosts monitor component (IHM), which uses connection classification in order to assess the overall "health" of a specific monitored host.

A. *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 are 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. The IP_Packet_Table is an approximation cache indexed by a hash of the packet's 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 [10], a malicious user can use these packets for scanning purposes. In the case of a connectionless protocol, 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. Since connectionless protocols use only a "best effort" approach for packet delivery, there should be no duplicate packets of this type in a short amount of time (i.e. the source does not retransmit if a packet is lost).

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's 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's I/OCache, and then subtracts one from the I/OCache entry's 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's resulting value. The formula for calculating this charge is shown in Table I. 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, i.e. not scanning behavior. If the value of entry is greater than zero, the state is set equal to the entry's 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. As will be seen in our experimental results, this multiplier serves its purpose quite well. Note that 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.

TABLE I. FORMULAS FOR PACKET CHARGE

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

In the case of a UDP packet, the packet's payload is of importance because there is no connection information associated with 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's count value by one, and sets the TTL of the entry to 255. If the source of packet is a monitored host, the IHM component then assesses the host a charge. As Table I shows, the charge is simply two times the count value minus one. Note that an arbitrary non-duplicate packet would result in no charge.

In the case of any other protocol, Fates skips the packet. It is arguable that ICMP [18] should be processed. However, since this packet type is connectionless and used for control messages, 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. 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 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 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's activity. A well-behaved host's threshold will plateau, but a scanning host's activity constantly causes the host's 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.

B. 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:

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

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 this time step. This is similar to TCP roundtrip time (RTT) estimation as discussed in [17], which provides an efficient way to calculate a weighted average of readings. The formula encompasses both an implied time-to-live for charges against a host and a contextual analysis of a network host's status at present. In practice, the value of α should range between 0.5 and 0.75.

With this averaging, the IHM component can compare a host's 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's 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 [7]. However, the IHM component currently limits this analytical tool to providing a method to interpret network information for tuning a threshold, as discussed next.

C. *Threshold Adjustment*

As previously stated the IHM component is quick to raise a host's threshold but lowering the threshold requires further analysis of both current state of the host's activities and its previous activity. IHM attempts to find equilibrium for each host's activity. Quickly redeeming charges possess 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 the next section, 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's threshold adjustment, the threshold will remain the same until being exceeded by a host's score. Once a host's score exceeds the host's threshold, the value of the host's 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:

$$T = T_{current} - (T_{min} - S) / 2$$

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.

IV. EXPERIMENTATION

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, and a World of Warcraft (WOW) [24] traffic set. 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. Due to page limit we will only show the results of Slammer simulation and USC traffic.

A. *Slammer*

The Slammer worm [13] was one of the most infectious worms to plague computer networks. Within three hours of its introduction, the worm had infected almost all susceptible computers running an unpatched version of Microsoft SQL Server (see [13]). In an effort to test Fates, we developed a simulation package that attempts to simulate the slammer worm's 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 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 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, Slammer is not only easily detected, but the well-behaved node's threshold remains static throughout the monitoring time.

In Figure 1, 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's charges, the threshold constantly increases in a linear fashion throughout the duration of the experiment.

The trend of the worm to increase a host's threshold at a steady rate is a factor of its propagation method as opposed to the time associated with the propagation. As Figure 2 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 exhibited. Although the increase is not linear as in the previous example, we observe a steady increase in the threshold. Another feature that 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 mitigated by continued effort of the host to propagate duplicate malicious packets.

B. *USC Traffic*

Next, we test the Fates system's capabilities with regard to TCP/IP scanning methods in a real network environment. The University of South Carolina's Department of Computer Science and Engineering is gracious enough to allow for managed data collection from their subnet. This network

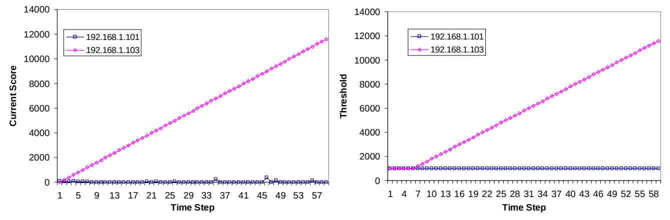

Figure 1. Slammer Simulation (with a propagation delay of 1 second)

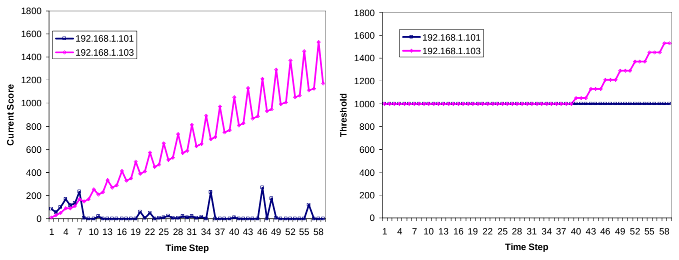

Figure 2. Slammer Simulation (with a propagation delay of 3 seconds)

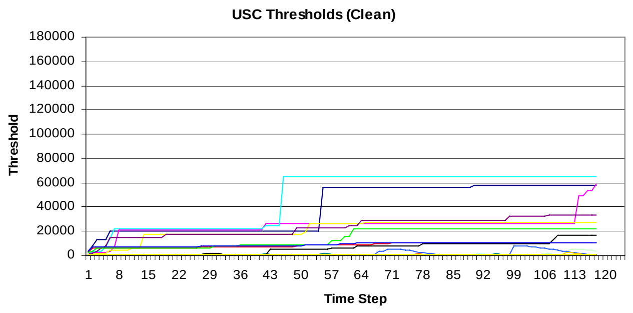

Figure 3. USC Traffic Threshold Analysis (clean)

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 [14] 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.

In order to examine the detection capabilities of the system the Fates system, we validate results against Snort [21], a widely utilized and respected NIDS system. The Snort system utilizes a rule-based analysis of network traffic and is completely configurable. Our aim is to detect everything that Snort detects for comparison purpose.

Prior to testing scanning behavior, we establish a baseline of normal network activity as shown in Figure 3. This base line reflects the normal activity of the network in absence of scanning. There are 37 entries in Figure 3, 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 4, 5, 6, and 7 describe the resulting thresholds present in the network. The first of these scans is the half-open scan. As is seen in Figure 4, 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 5, this behavior presented itself very clearly also with a steady increase in the threshold. Even though the 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 6. 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 a FIN scan but the second lowest host that is presumably benign. After further analysis, we determined the behavior to be a RST scan of port 22, SSH, which was an actual attack underway in the network! Figure 7 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.

V. 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 [23]. The system allows for dynamic, self-healing thresholds that allow for both forgiveness of misconfiguration and scaling to current network conditions. Furthermore, the Fates system uses simple calculations, unlike the entropy-based systems, such as [5], [22], and [25]. As a result, the functionality of the Fates system is appropriate for real-time detection.

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

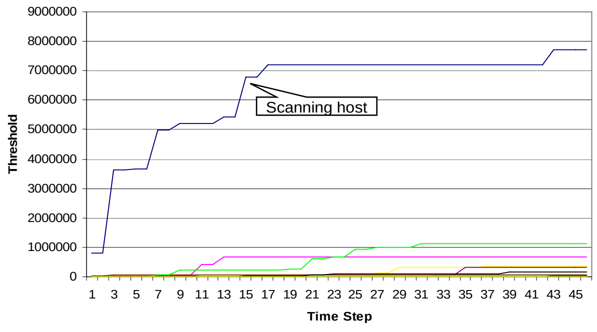

Figure 4. USC Traffic Threshold Analysis (half-open scan)

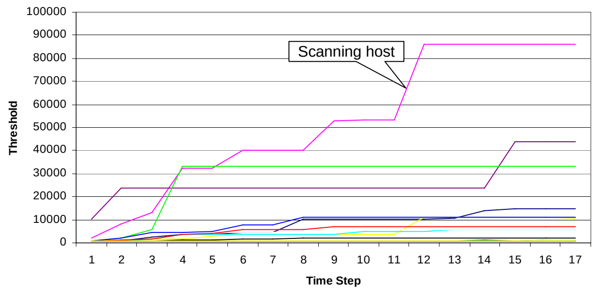

Figure 5. USC Traffic Threshold Analysis (ACK scan)

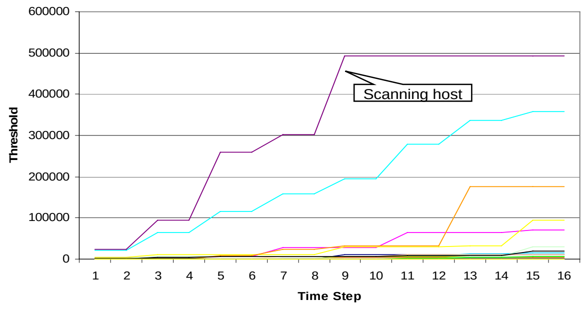

Figure 6. USC Traffic Threshold Analysis (FIN scan)

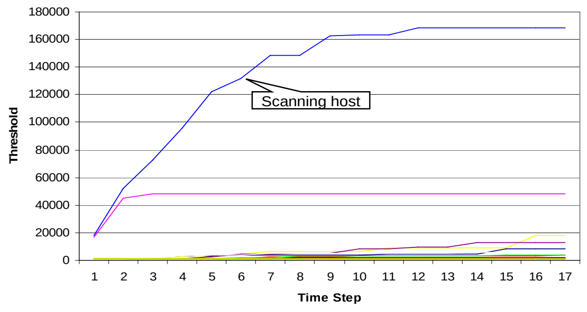

Figure 7. USC Traffic Threshold Analysis (RST scan)

require post processing 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. 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.

ACKNOWLEDGMENT

This work was supported by an AFRL/DARPA grant (FA8750-04-2-0260).

REFERENCES

[1] P. Barford, J. Kline, D. Plonka, A. Ron, "A Signal Analysis of Network Traffic Anomalies". In Proceedings of 2nd ACM Internet Measurement Conference, 2002.

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

[3] D. Denning. "An intrusion detection model". In Proceedings of the 1986 IEEE Symposium on Security and Privacy, pp 119–131, 1986.

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

[5] L. Feinstein et al., "Statistical Approaches to DDoS Attack Detection and Response". Proc. DARPA Information Survivability Conf. and Exposition, vol. 1, 2003, IEEE CS Press, pp. 303-314.

[6] H. Hajji, B. Far, J. Cheng. "Detection of Network Faults and Performance Problems". In Proceedings of IC2001, 2001.

[7] A. Hussain, J. Heidemann, C. Papadopoulos. "Framework for Classifying Denial of Service Attacks". In Proceedings of ACM SIGCOMM, 2003.

[8] J. Jung, B. Krishnamurthy, M. Rabinovich. "Flash Crowds and Denial of Service Attacks: Characterization and Implications for CDNs and Web Sites". In Proceedings of Int'l World Wide Web Conference, ACM Press, 2002, pp. 252-262.

[9] J. Jung, V. Paxson, A. Berger, H. Balakrishnan. "Fast Portscan Detection Using Sequential Hypothesis Testing". In Proceedings of 2004 IEEE Symposium on Security and Privacy p. 211, 2004.

[10] D. Kewley, J. Lowry, R. Fink, M. Dean. " Dynamic Approaches to Thwart Adversary Intelligence Gathering". Available at http://www.bbn.com/docs/whitepapers/ DISCEX_DYNAT.pdf.

[11] A. Lakhila, M. Crovella, C. Diot. "Mining Anomalies Using Traffic Feature Distributions". In Proccedings of ACM SIGCOMM 2005, 2005.

[12] W. Leland, M. Taqqu, W. Willinger, D. Wilson. "On the Self-Similar Nature of Ethernet Traffic". IEEE/ACM Transactions on Networking, Vol. 2, No. 1, pp. 1-15, 1994.

[13] D. Moore, V. Paxson, S. Savage, S. Staniford, N. Weaver, "Inside the Slammer Worm". Available at http://www-cse.ucsd.edu/~savage/papers/IEEESP03.pdf.

[14] NMap. http://www.insecure.org

[15] R. Pang, V. Yegneswaran, P. Barford, V. Paxson, L. Peterson, "Characteristics of Internet Background Radiation". In Proceedings of 4th ACM Internet Measurement Conference, 2004.

[16] V. Paxson. "Bro: A System for Detecting Network Intruders in Real-Time". Available at http://bro-ids.org/publications.html.

[17] V. Paxson. "RFC 2988 - Computing TCP's Retransmission Timer", Available at http://www.faqs.org/rfcs/rfc2988.html.

[18] J. Postel. "RFC 792 Internet Control Message Protocol". Available at http://www.faqs.org/rfcs/rfc792.html.

[19] T. Ptacek. "Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection". http://secinf.net/info/ids/idspaper/idspaper.html.

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

[21] Snort, http://www.snort.org

[22] S. Thareja: "A Real Time Network Traffic Wavelet Analysis". Master Thesis, Department of Computer Science and Engineering, University of South Carolina, 2005.

[23] N. Weaver, S. Staniford, V. Paxson. "Very Fast Containment of Scanning Worms". In Proceedings of the 13th USENIX Security Symposium, pages 29-44, 2004.

[24] World of Warcraft Community. http://www.worldofwarcraft.com.

[25] J. Zachary, J. McEachen D. Ettlich. "Conversation Exchange Dynamics for Real-Time Network Monitoring and Anomaly Detection". In Proceedings of IWIA, pp.59-70, 2004.