Evaluation of an Online Parallel Anomaly Detection System

Shashank Shanbhag and Tilman Wolf
Department of Electrical and Computer Engineering
University of Massachusetts
Amherst, MA 01003–9284
Email: (sshanbha, wolf)@ecs.umass.edu

***Abstract*—The rapid and accurate detection of anomalies in network traffic has always been a challenging task, and is absolutely critical to the efficient operation of the network. The availability of numerous different detection algorithms makes it difficult to choose a suitable configuration. An algorithm may have a high detection rate for high rate attacks, but might behave unfavorably when faced with attacks with gradually increasing rates. This paper proposes an online parallel anomaly detection system that implements multiple anomaly detection algorithms in parallel to detect anomalies in real-time. The main idea is to aggregate the detection data from multiple algorithms to come up with a single anomaly metric. We evaluate this system with realistic attacks on the DETER testbed. Our results show improved true positive and false negative rates for both high intensity and slow-rise ramped floods. Furthermore, the system is able to detect attacks separated by as little as 15 seconds with a high true positive rate.**

I. Introduction

Anomalous behavior in networks pertains to traffic that departs from the normal observed behavior, and is thus tagged as unusual. Traffic anomalies could be the result of failure of network traffic equipment (e.g., aging hardware, faulty software, link failure), flash crowds (e.g., release of an operating system patch), or – more seriously – misuse or network attacks (e.g., Denial of Service attacks). The detection and classification of anomalies and to distinguish them from the normal traffic is a difficult task. A large number of systems have been proposed for anomaly detection. The focus of these systems ranges from statistical analysis of volume of a particular subset of traffic (e.g., SYN packets) or aggregate traffic (e.g., total number of packets, byte rate) on a link, to identification of different flows and the application of machine learning techniques toward anomaly detection.

Important metrics used to characterize the performance of such systems are the false positive rate, the false negative rate, and the true positive rate. The kind of anomalies the system is able to detect is another important measure of effectiveness of the system. Most of these systems perform well while targeting certain kinds of attacks, but are found lacking when targeted by attacks which work around the properties of the system. Also, most systems rely on post-processing where the anomaly occurrences are stored in logs or as traces to be later studied by the network operator. Thus, detection latency is an issue in such post-processing systems.

To counter this problem, we present a general Online Passive Anomaly Detection system that implements multiple anomaly detection algorithms to monitor a wide range of traffic classes in parallel. Our prototype of the detection system is based on the Online Passive Measurement Node [1]. The evaluation of the anomaly detection system is carried out on a testbed using realistic attack traffic. The system can monitor for anomalies in real-time negating the need to store complete traces and thus reducing the post-processing time.

The contributions of this paper are:

- A general normalization and aggregation process that provides a mechanism to interpret and combine the outputs from different anomaly detection algorithms generating an anomaly metric.
- An evaluation of the performance of the Online Parallel Anomaly Detection System prototype using a set of experiments carried out on the DETER testbed [2].

The remainder of this paper is organized as follows. Section II discusses related work which includes some anomaly detection algorithms that we use in our system. Section III describes the overall architecture of the Parallel Anomaly Detection System. Section IV presents our normalization and aggregation process. Evaluation results are presented in Section V. Section VI summarizes and concludes this paper.

II. Related Work

Anomaly detection in computer networks is a widely researched topic. A number of algorithms have been developed for this purpose, each differing in the manner in which information from the packet stream is processed. The Holt-Winter algorithm uses a time-series based forecasting model concentrating solely on volume data [3]. Lakhina et al. [4] identify anomalies in byte counts, packet counts and IP-flow counts by making use of the multiway subspace method. Siris and Papagalou [5] evaluate the Adaptive Threshold and Cumulative Sum algorithms in identifying SYN flooding attacks. Barford et al. [6] use signal analysis methods and more specifically wavelets to determine class and characteristics of the anomalies. Gu et al. [7] use a Maximum Entropy based Anomaly Detection and classification algorithm that divides the traffic into 1024 classes based on the Protocol and Ports. Chen et al. [8] make use of a multivariate statistical technique based on the Chi-square test statistic to detect

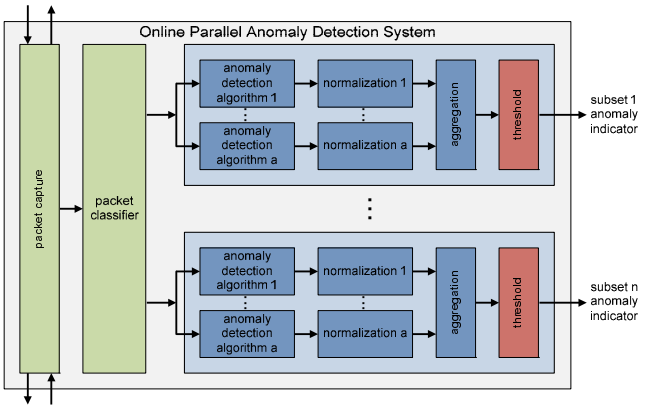

Fig. 1. Data Flow through System

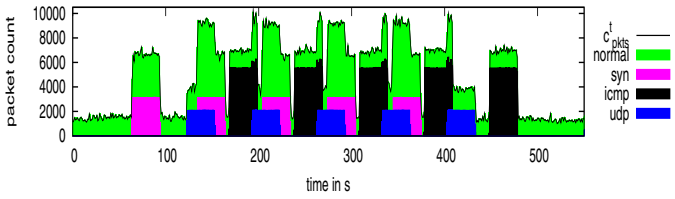

Fig. 2. Simultaneous attacks using multiple subsets of traffic. The system monitors each subset, and produces an aggregated anomaly score for each subset.

network anomalies. Lazarevic et al. [9] evaluate several outlier detection schemes for detecting anomalies. We employ some of these algorithms in our Parallel Anomaly Detection System. It is important to note that this work concentrates solely on volume based anomalies. Other anomaly types (e.g., timing attacks) are not considered.

III. Architecture

Online anomaly detection systems have a strict bound on the time taken to process packets in the network. This becomes especially important in high-rate networks. One could choose to implement a passive detection system which post-processes traces, but the huge amounts of data storage required coupled with the latency in detection can easily offset the advantages of such a system. One solution to the huge data storage problem can be to store summary information similar to Netflow data. However, the method restricts the scope to the characteristics of traffic represented by the summary information. An anomaly detection system should not only be good at detecting anomalies in traffic, but also at characterization. Characterization helps in pinpointing the exact subset of traffic that is causing the anomaly. Our system accomplishes this by making use of several anomaly detection algorithms in parallel and independent of each other while processing a particular subset of traffic. The metrics generated by each of these algorithms for that subset are further normalized and aggregated. Each of these algorithms also processes various subsets of traffic in parallel.

A. System Design

The Figure 1 shows the process in which parallelism is exploited by the system. Packet headers are initially captured from a monitored link. The Packet classifier extracts the required subset data from the headers. The data can be anything from aggregate byte counts to the more specific TCP-SYN packet counts to Port 80 per observation interval. For example, Figure 2 shows simultaneous attacks using different subsets of traffic (ICMP, TCP-SYN and UDP in this example). It is important to note that merely monitoring the aggregate packet count does not help in characterizing the anomaly as is evident from the Figure 2. Each anomaly detection algorithm processes packet counts pertaining to the subset its monitoring and produces a metric that is further normalized. The normalized metric of each algorithm is then aggregated with the normalized metrics from other algorithms. The aggregated metric for the subset is an anomaly score which indicates the severity of the attack using that subset. An anomaly is then triggered based on whether the anomaly score exceeds the threshold which is then reported to the operator.

B. Prototype Implementation

The anomaly detection algorithms implemented in our prototype system are:

- HW: Holt Winter Forecasting Model (e.g., [3]).
- ADAP: Adaptive Threshold Algorithm (e.g., [5]).
- AVG: Average over Window (e.g., [10]).
- EWMA: Exponential Weighted Moving Average (e.g., [11]).
- CUSUM: Cumulative Sum Algorithm (e.g., [12]).
- CHISQ: Chi-Square Algorithm (e.g., [8]).
- ENTR: Maximum Entropy based Algorithm (e.g., [7]).

Outputs of all algorithms are aggregated as described below.

IV. Normalization and Aggregation

Each algorithm outputs an anomaly metric that depends largely on the characteristics of the algorithm since each algorithm processes the same traffic data in any given observation interval. The challenge then is in devising a method to normalize each of these outputs so they can be effectively aggregated to produce a single anomaly metric. The normalization and aggregation modules are explained in detail after the notation is introduced. A more detailed discussion of normalization and aggregation is available as a technical report [13].

A. Notation

Our system processes traffic volume data to determine anomalies. Let c^t be the packet count during the observation interval $[t, t+\tau)$. Parameter τ is assumed to be a preset fixed time interval that decides the granularity of the data. The interval can range from 1 second to a few hours. Let s be the total number of different traffic subsets S_i, $1 \leq i \leq s$ and a be the number of anomaly detection algorithms A_j, $1 \leq j \leq a$. Then, during the observation interval $[t, t+\tau)$, the algorithms use a metric $m^t_{i,j}$=$c^t_i/p^t_{i,j}$, where $p^t_{i,j}$ denotes the prediction produced by each algorithm A_j based on the history of packet

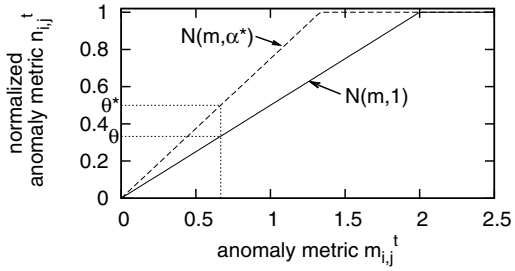

Fig. 3. Normalization Function $N(m, \alpha)$. α^* ensures that boundary between anomalies and no anomalies occurs at θ^*=0.5.

counts $c_i^{t_k}$ $(t_k < t)$ and c_i^t denotes the current packet count during the interval $[t, t + \tau)$ for subset S_i.

B. Normalization

The prediction $p_{i,j}^t$ depends on the characteristics of the algorithm j. Thus, normalization is necessary to ensure equal influence of each algorithm on the aggregate. We define a normalization function, N, that produces the normalized metric $n_{i,j}^t$, for algorithm j and subset i at time t. This function normalizes $m_{i,j}^t$ to the continuous interval $[0, 1]$, where 0 represents the condition of "no anomaly" and 1 represents "anomaly". The normalization function is as follows:

$$n_{i,j}^t = N\left(m_{i,j}^t, \alpha_j\right) = \min(1, max(0, 0.5 \cdot \alpha_j \cdot m)). \quad (1)$$

Parameter α_j determines the slope of the normalization function, and is adjusted such that the boundary between the anomalies and no anomalies falls exactly in the middle of the range $[0, 1]$ at θ^*=0.5 for $\alpha_j^* = \frac{n^*}{\theta_j}$ (refer Figure 3). This also ensures we do not have to use a system specific threshold, and can use θ^*=0.5 to compare the final aggregated value with (Other normalization functions we have evaluated are $N(m, \alpha)$=$\max(1 - e^{-\alpha \cdot \ln m}, 0)$, $N(m, s)$=$1 - e^{-|\alpha \cdot \ln m|}$, but they do not provide better performance (see [13])).

C. Aggregation and Anomaly Decision

Once the normalized anomaly metric $n_{i,j}^t$ is output by the normalization module for all algorithms, j, for a particular subset S_i, the aggregation module uses the aggregation function G to determine the final aggregated anomaly metric g_i^t across all a algorithms for subset i at time t. Some of the functions explored were arithmetic mean, geometric mean, median, minimum, and maximum, but the average of mean and minimum as proposed by Evangelista et al. in [14] has proven to be most effective:

$$g_i^t = G\left(n_{i,1}^t, \ldots, n_{i,a}^t\right) = \frac{1}{2}\left(\frac{1}{a}\sum_{j=1}^{a} n_{i,j}^t + \max_j n_{i,j}^t\right). \quad (2)$$

Finally, the aggregated anomaly metric g_i^t is compared to a threshold, θ^*, to make a binary decision as follows:

$$r_i^t = \begin{cases} 0, & \text{if } g_i^t < \theta^* \\ 1, & \text{if } g_i^t \geq \theta^*. \end{cases} \quad (3)$$

The result is then reported to the user.

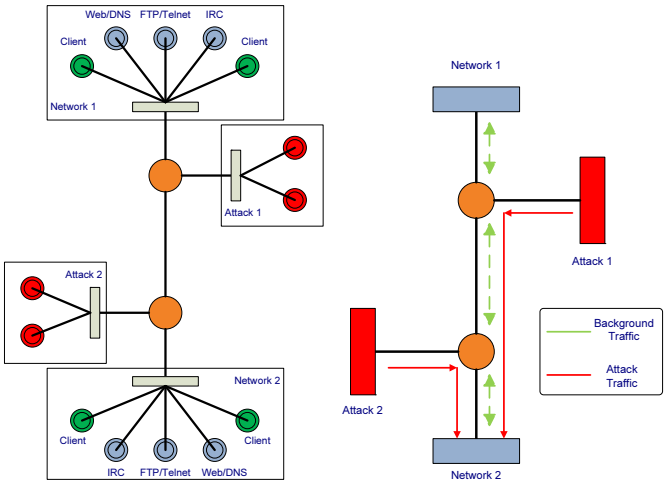

Fig. 4. Experimental Topology

V. Evaluation

To illustrate the effectiveness of our aggregation process, we evaluate a prototype system with realistic attack traffic on the DETER testbed [2].

A. Experimental Setup

The experimental topology shown in Figure 4 consists of four networks, two attack and two client, interconnected through two routers. The attack networks have two nodes each. Each client network has three server nodes and two client nodes. All links have a bandwidth of 100 Mbps and no delay. The six server nodes run five services: Web, FTP, IRC, Telnet and DNS. In each network, one node provides both Web and DNS services and another provides FTP and Telnet services. IRC service has a dedicated server node in each client network.

The setup uses two types of traffic generated in the system: background traffic and attack traffic. Every client node generates a mixture of traffic pertaining to the services provided. Clients in one network request services from server nodes in the other. Figure 4 illustrates the traffic flow in the network.

To generate realistic background traffic and attacks, the SEER traffic generation tool [15] is used. SEER supports automation and repeatable experiments on DETER testbed by use of a Perl-based scripting interface. The background traffic generated should accurately reflect the traffic in a typical network. Therefore, some of the traffic parameters and distributions as suggested by Mirkovic et al. [16] (refer Table I) have been used.

The attacks are synthetically generated using the Flooder tool provided with SEER, allowing us to control the characteristic of the attacks. The tool is capable of generating flooding attacks with custom protocol (TCP flags and ICMP codes included), packet rates and sizes. Besides, the experimenter can generate attacks with different rate characteristics (pulsed, constant rate, ramp or combinations). Figure 2 shows simultaneous ICMP, SYN and UDP flooding attacks together with the background traffic generated using the SEER tool.

TABLE I
BACKGROUND TRAFFIC DISTRIBUTIONS AND PARAMETERS. PARAMETER s IS THE SCALING FACTOR AND m IS THE MAXIMUM VALUE ALLOWED FOR A PARAMETER.

Traffic Type	Parameter	Distribution
Web	Request Interarrival Time(s)	Exp(1),s=1,m=15
	File Size(Bytes)	Pareto(1.04,1000),m=1000
DNS	Request Interarrival Time(s)	Exp(1),s=3,m=30
Telnet	Request Interarrival Time(s)	Exp(5),s=5,m=100
	Response Size(Bytes)	$\Gamma(40, 0.9)$
	Session Duration(s)	$\Gamma(30, 0.5)$
	Time between key strokes(s)	Exp(1),s=1,m=10
	Time between sessions(s)	Exp(2),s=1,m=10
FTP	Request Interarrival Time(s)	Exp(5),s=1,m=100
	File Size(Bytes)	Pareto(1.2,5000),m=5000
IRC	Request Interarrival Time(s)	Exp(1),s=5,m=100
	Message Size(Bytes)	$\Gamma(40, 0.9)$
Ping	Request Interarrival Time(s)	Exp(5),s=2,m=30

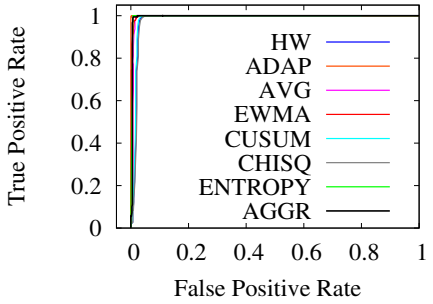

Fig. 6. ROC Curve for aggregation function for the Pulsed TCP-SYN Flood attack.

In all our experiments, the attack nodes generate floods that target port 80 in the Web/DNS server node in Network 2.

B. Results

Since our emphasis is on detecting different kinds of attacks rather than the subset used and on finding out whether the aggregate anomaly metric provides a better detection performance than any individual algorithm across different attack types, our experiments target TCP-SYN floods with different characteristics. It is important to note that the system is a general implementation capable of detecting attacks using multiple traffic subsets (as in Figure 2). The performance metrics we consider include True Positive Rate, False Positive Rate, and the False Negative Rate and the effect of different attacks on these metrics. We also examine the variation of the aggregate normalization parameter α_j with traffic rate.

1) Pulsed TCP-SYN Flood Attack: Our first experiment considers pulsed TCP-SYN floods of high intensity targeting port 80 on the Web/DNS server node in Network 2. The mean amplitude of the attack traffic is 2.5 times the mean traffic rate. The experiment is carried out over a duration of 600 seconds, and consists of 8 pulses of 20 second duration separated by 50 seconds. Figure 5 shows the variation of the aggregate with the onset of attacks. The aggregate function from Equation 2 is used as explained above. The results show that the anomaly detection system detects the attacks accurately.

To quantify the accuracy, Figure 6 compares the receiver operating characteristic (ROC) curves [17] of the aggregate with those of the individual algorithms. The aggregate shows an improvement of 4.31% (FNR=0.027) for the false negative rate over all algorithms. However, the false positive rate has degraded by 7.83% (FPR=0.094) which is still negligible considering the improved false negative rate. We attribute this increase to the greater time to stabilization for algorithms such as EWMA and Averaging algorithm, which tends to influence the classification at the start of the detection process on account of the use of the max function which is evident in Figures 7,8 and 9. The aggregate also has zero detection latency for this type of attack.

2) Slow Rise-time Ramp Attack: Next we examine the performance of the aggregate in case of a slow rise-time ramp flood. In case of such attacks, the imperative is on detection in the earliest of stages of the attack. In an experiment lasting 200 seconds, the port 80 on the Web/DNS server node in Network 2 is targeted with a slow rising ramped TCP-SYN flood of duration 100 seconds reaching a peak intensity of 500 TCP-SYN packets per second from 10 TCP-SYN packets per second. Figure 7 shows the variation of the aggregates with the onset of the attack.

Figure 10 shows the deteriorated performance for most algorithms. However, the aggregate continues to outperform most algorithms. For the slow-rise ramp attack, the aggregate shows an improvement of 3.42% (FNR=0.058) for the false negative rate over all algorithms. Again, the false positive rate performance marginally degrades by 11.95% (FPR=0.183). The attack detection latency for slow-rise ramped attack is 6 seconds, which is better than any other algorithm used.

3) Resolution: Resolution has been explored by Dainotti et al. [18] and is defined as the minimum distance between two distinct anomalous events such that the system is successful in detecting them as distinct anomalies. In two experiments that last for 1200 seconds, port 80 of Web/DNS server node in Network 2 is targeted with 5 TCP-SYN flood attack pulse pairs of 200pps and 400pps mean amplitude with increasing distances between the pulses. The bottom graphs in Figures 9 and 8 show the variation of aggregate. Table II summarizes the performance of the aggregate for different resolutions. The rates are calculated from the number of True Positives and False Negatives in the second pulse in the attack pair. For low intensity pulses, at a resolution of 35 seconds and above, the system has a higher true positive rate and a lower false negative rate. For distances as low as 5 seconds, the system detects the first instance (first few seconds in the second pulse) of an increase in rate but has a greater number of false negatives for the rest of the pulse. On the contrary, for high intensity pulses, there is a significant improvement in detection performance with true positive rate of 0.31 at a resolution of 5 seconds.

4) Variation of α_j with traffic rate: The aggregate normalization parameter, α_j^* dictates the performance of the anomaly detection as it determines the boundary between anomaly and no-anomaly which is set to $\theta^* = 0.5$ in our experiments. It is evident from Equation (1) that α_j depends entirely on the

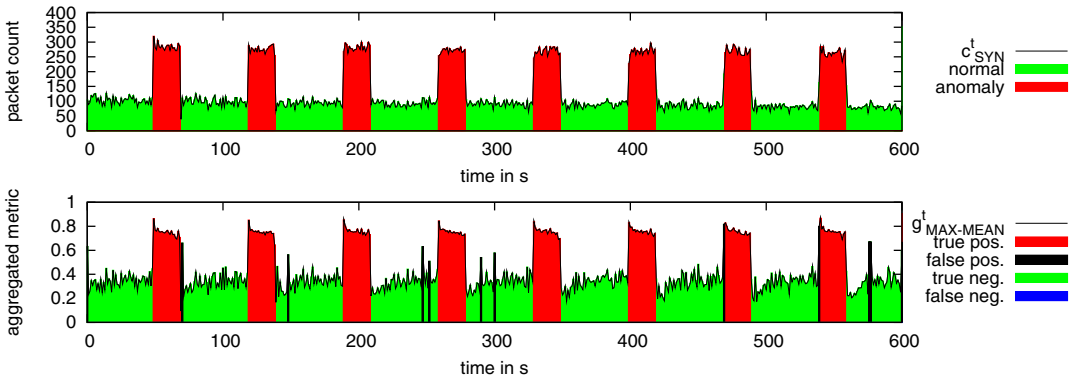

Fig. 5. Pulsed TCP-SYN Flood and Variation of the Aggregate.

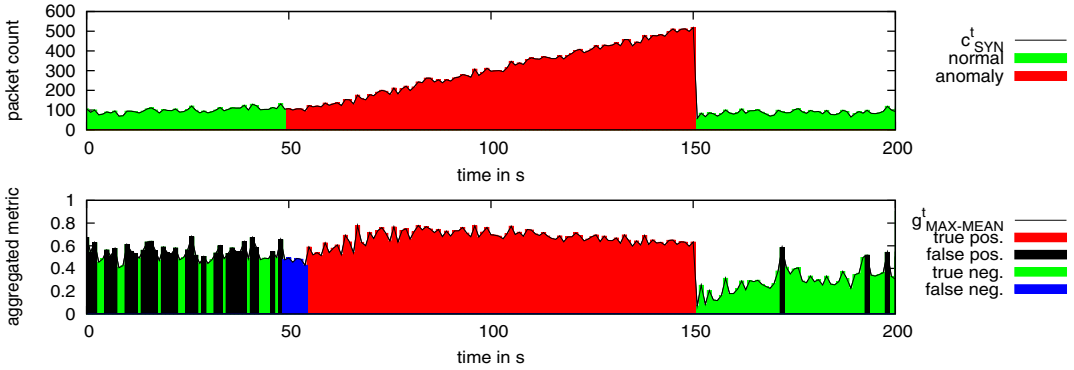

Fig. 7. Ramped TCP-SYN Flood and Variation of the Aggregate.

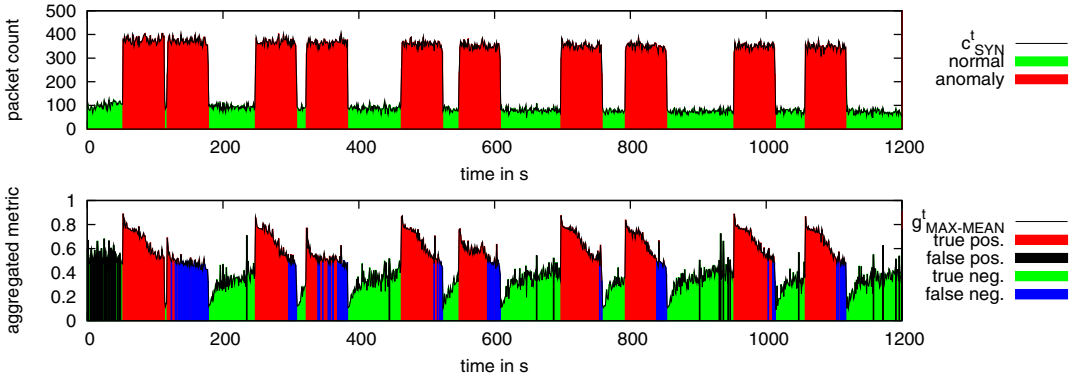

Fig. 8. Resolution of the Parallel ADS: High Intensity TCP-SYN Floods with increasing distances between attacks and Variation of the aggregate.

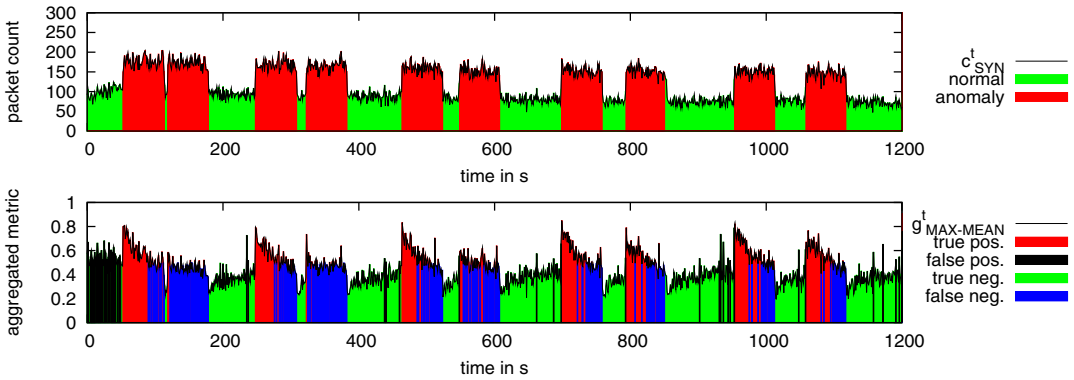

Fig. 9. Resolution of the Parallel ADS: Low Intensity TCP-SYN Floods with increasing distances between attacks and Variation of the aggregate.

TABLE II
TRUE POSITIVE RATES AND FALSE NEGATIVE RATES AT DIFFERENT RESOLUTIONS AND INTENSITIES USING *MAX-MEAN* AGGREGATE.

Experiment		Low Intensity (Fig 9)		High Intensity (Fig 8)	
No.	Resolution (s)	TPR	FNR	TPR	FNR
1	5	0.150000	0.850000	0.316667	0.683333
2	15	0.250000	0.750000	0.683333	0.316667
3	25	0.406780	0.593220	0.762712	0.237288
4	35	0.614035	0.385965	0.824561	0.175439
5	45	0.661017	0.338983	0.850000	0.150000

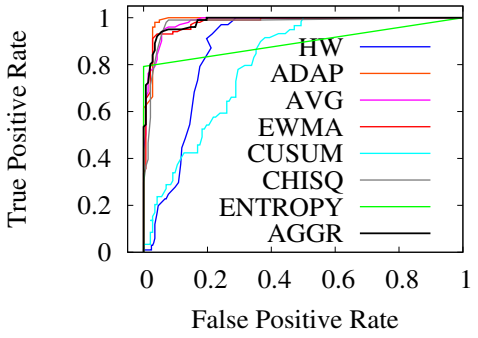

Fig. 10. ROC Curve for aggregation function for Ramped SYN Flood attack.

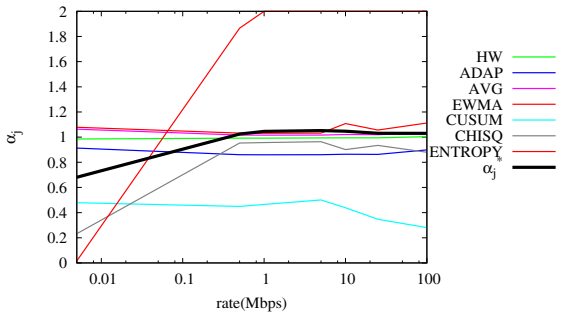

Fig. 11. Variation of α_j with traffic rate.

packet rate. The study of variation of α_j with packet rate can make it possible for network operators to set a α_j value fitting to the network, thus reducing the time spent in estimating the value of α_j. The packet rate is varied from 5kbps to 100Mbps over a series of experiments and the corresponding values of α_j plotted. From Figure 11, note that α_j^* stabilizes to a value between ≈ 1.03 and ≈ 1.05 between 1Mbps and 100Mbps.

VI. SUMMARY AND CONCLUSION

The paper proposed the design of an online parallel anomaly detection system that makes use of multiple algorithms in parallel to detect anomalies in real-time. The experiments show an improved true positive and false negative rate across different attack types. The paper also evaluates the resolution of the system: the system is able to detect attacks which are separated in time by as low as 15 seconds with a high true positive rate.

ACKNOWLEDGEMENTS

This material is based upon work supported by the National Science Foundation under Grant No. CNS-0325868.

REFERENCES

[1] T. Wolf, R. Ramaswamy, S. Bunga, and N. Yang, "An architecture for distributed real-time passive network measurement," in *Proc. of 14th Annual Meeting of the IEEE/ACM International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, Monterey, CA, Sep. 2006, pp. 335–344.

[2] T. Benzel, R. Braden, D. Kim, C. Neuman, A. Joseph, K. Sklower, R. Ostrenga, and S. Schwab, "Design, deployment, and use of the DETER testbed," in *DETER: Proceedings of the DETER Community Workshop on Cyber Security Experimentation and Test on DETER Community Workshop on Cyber Security Experimentation and Test 2007*, Boston, MA, 2007.

[3] J. D. Brutlag, "Aberrant behavior detection in time series for network monitoring," in *Proc. of the 14th Systems Administration Conference*, New Orleans, LA, Dec. 2000, pp. 139–146.

[4] A. Lakhina, M. Crovella, and C. Diot, "Characterization of network-wide anomalies in traffic flows," in *IMC '04: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, Taormina, Italy, Oct. 2004, pp. 201–206.

[5] V. A. Siris and F. Papagalou, "Application of anomaly detection algorithms for detecting SYN flooding attacks," in *Proc. of IEEE Global Communications Conference (GLOBECOM)*, Dallas, TX, Nov. 2004, pp. 2050–2054.

[6] P. Barford, J. Kline, D. Plonka, and A. Ron, "A signal analysis of network traffic anomalies," in *Proc. of the 2nd ACM SIGCOMM Workshop on Internet Measurement (IMW)*, Marseille, France, Nov. 2002, pp. 71–82.

[7] Y. Gu, A. McCallum, and D. Towsley, "Detecting anomalies in network traffic using maximum entropy estimation," in *Proc. of the 5th ACM SIGCOMM Conference on Internet Measurement (IMC '05)*, Berkeley, CA, Oct. 2005.

[8] N. Ye and Q. Chen, "An anomaly detection technique based on a chi-square statistic for detecting intrusions into information systems," *Quality and Reliability Engineering International*, vol. 17, no. 2, pp. 105–112, Mar. 2001.

[9] A. Lazarevic, L. Ertöz, V. Kumar, A. Ozgur, and J. Srivastava, "A comparative study of anomaly detection schemes in network intrusion detection," in *Proc. of the Third SIAM International Conference on Data Mining*, San Francisco, CA, May 2003.

[10] C. Schwarzer, "Prediction and adaptation in a traffic-aware packet filtering method," Master's thesis, École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland, Mar. 2006.

[11] S. Deshpande, M. Thottan, T. K. Ho, and B. Sikdar, "A statistical approach to anomaly detection in interdomain routing," in *Proc. of Third International Conferenc on Broadband Communications, Networks, and Systems (BROADNETS)*, San Jose, CA, Oct. 2006.

[12] H. Wang, D. Xhang, and K. G. Shin, "Change-point monitoring for the detection of dos attacks," *IEEE Transactions on Dependable Secure Computing*, vol. 1, no. 4, pp. 193–208, Oct. 2004.

[13] T. Wolf and S. Shanbhag, "Massively parallel anomaly detection system," University of Massachusetts, Amherst, MA, Tech. Rep. TR-08-CSE-08, Mar. 2008.

[14] P. F. Evangelista, M. J. Embrechts, and B. K. Szymanski, "Data fusion for outlier detection through pseudo-ROC curves and rank distributions," in *Proc. of International Joint Conference on Neural Networks (IJCNN)*, Vancouver, BC, Jul. 2006, pp. 2166–2173.

[15] S. Schwab, B. Wilson, C. Ko, and A. Hussain, "SEER: a security experimentation environment for DETER," in *DETER: Proceedings of the DETER Community Workshop on Cyber Security Experimentation and Test on DETER Community Workshop on Cyber Security Experimentation and Test 2007*, Boston, MA, 2007.

[16] J. Mirkovic, A. Hussain, B. Wilson, S. Fahmy, P. Reiher, R. Thomas, W.-M. Yao, and S. Schwab, "Towards user-centric metrics for denial-of-service measurement," in *ExpCS '07: Proceedings of the 2007 Workshop on Experimental Computer Science*, San Diego, CA, Jun. 2007.

[17] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, Jun. 2006.

[18] A. Dainotti, A. Pescape, and G. Ventre, "Wavelet-based detection of dos attacks," in *Proc. of IEEE Global Communications Conference (GLOBECOM)*, San Francisco, CA, Nov. 2006.