Automated Anomaly Detector Adaptation using Adaptive Threshold Tuning

MUHAMMAD QASIM ALI and EHAB AL-SHAER, University of North Carolina Charlotte (UNCC)
HASSAN KHAN, National University of Sciences and Technology (NUST), Pakistan
SYED ALI KHAYAM, PLUMgrid Inc

Real-time network- and host-based Anomaly Detection Systems (ADSs) transform a continuous stream of input data into meaningful and quantifiable anomaly scores. These scores are subsequently compared to a fixed detection threshold and classified as either benign or malicious. We argue that a real-time ADS' input changes considerably over time and a fixed threshold value cannot guarantee good anomaly detection accuracy for such a time-varying input. In this article, we propose a simple and generic technique to adaptively tune the detection threshold of any ADS that works on threshold method. To this end, we first perform statistical and information-theoretic analysis of network- and host-based ADSs' anomaly scores to reveal a consistent time correlation structure during benign activity periods. We model the observed correlation structure using Markov chains, which are in turn used in a stochastic target tracking framework to adapt an ADS' detection threshold in accordance with real-time measurements. We also use statistical techniques to make the proposed algorithm resilient to sporadic changes and evasion attacks. In order to evaluate the proposed approach, we incorporate the proposed adaptive thresholding module into multiple ADSs and evaluate those ADSs over comprehensive and independently collected network and host attack datasets. We show that, while reducing the need of human threshold configuration, the proposed technique provides considerable and consistent accuracy improvements for all evaluated ADSs.

Categories and Subject Descriptors: C.2.0 [**Computer-Communication Networks**]: General—*Security and protection (e.g., firewalls)*; D.4.6 [**Operating Systems**]: Security and Protection—*Invasive software*; K.6.5 [**Management of Computing and Information Systems**]: Security and Protection—*Unauthorized access*

General Terms: Algorithms, Security

Additional Key Words and Phrases: Adaptive thresholding, anomaly detection, intrusion detection, anomaly scores

ACM Reference Format:
Ali, M. Q., Al-Shaer, E., Khan, H., and Khayam, S. A. 2013. Automated anomaly detector adaptation using adaptive threshold tuning. *ACM Trans. Inf. Syst. Secur.* 15, 4, Article 17 (April 2013), 30 pages.
DOI: http://dx.doi.org/10.1145/2445566.2445569

1. INTRODUCTION

The last decades have witnessed an unprecedented increase in the volume and sophistication of network attacks. The exponential growth of zero-day attacks has

Part of this work appeared in the *Proceedings of the ACM Conference on Computer and Communication Security (CCS)*, 2009 [Ali et al. 2009].
Authors' addresses: M. Q. Ali (Corresponding author) and E. Al-Shaer, Department of Software and Information Systems, University of North Carolina Charlotte, 9201 University City Blvd. Charlotte, NC 28223; email: mohdqasimali@gmail.com; H. Khan and S. A. Khayam, School of Electrical Engineering and Computer Science, University of Sciences and Technology, Pakistan.

DOI: http://dx.doi.org/10.1145/2445566.2445569

emphasized the need of defense mechanisms that can accurately detect previously-unseen attacks in real-time with little (if any) human intervention. Consequently, significant research effort has been targeted towards real-time network- and host-based anomaly detection [Agosta et al. 2007; Forrest et al. 1996; Twycross and Williamson 2003]. In contrast to traditional signature-based intrusion detection systems, which detect known attack vectors, an Anomaly Detection System (ADS) detects deviations from normal behavior to detect zero-day attacks.

Most real-time ADSs require human intervention (in terms of intuition and experience) to set rigid (time and behavior invariant) anomaly detection thresholds, and therefore do not have the ability to adapt with varying traffic and host characteristics. In order to determine operating threshold for an ADS, Receiver Operating Characteristics (ROC) curves are generated by applying a range of classification thresholds to an ADS' anomaly scores and then plotting the detection rate versus the false alarm rate for each threshold value. For real-time operation, the best ROC operating point is chosen to threshold anomaly scores. In practice, such rigid thresholding results in a loss of automation and accuracy [Ashfaq et al. 2008; Lippmann et al. 2000] and consequently hinders wide spread deployment of ADSs [Gartner Report]. To automate this process, some commercial network-based ADS products adjust their thresholds in accordance with the input traffic characteristics [Arbor PeakFlow; Cisco Anomaly Guard] and some recent ADSs provide methods to calculate optimum thresholds for their specific algorithms [Ide and Kashima 2004; Jung et al. 2004; Lakhina et al. 2004]. However, a *generic* threshold adaptation technique, that can be plugged into existing ADS is largely unexplored.

In this article, we challenge the conventional wisdom that ROC curves generated using fixed thresholds can identify the best achievable accuracy for an ADS. We argue that the input characteristics of a real-time ADS change continuously with time and setting a rigid (time and behavior invariant) classification threshold limits the accuracy that the ADS can potentially achieve. Moreover, threshold determination using ROC curves also introduces undesirable human intervention in an ADS' operation. To mitigate these limitations, we propose a technique to adaptively tune the detection threshold of a real-time ADS in accordance with varying host and network behavior.

The basic premise of the proposed threshold adaptation technique is that: If we can accurately predict the expected values of future anomaly scores under benign conditions, the classification threshold can be adapted as a function of the predicted score. Thus threshold adaptation requires a tracking algorithm that can accurately predict future anomaly scores. Additional constraints on a practical threshold adaptation technique are: (1) it should be able to automatically learn the temporal behavior of anomaly scores under benign conditions; (2) it should remain stable in presence of sporadic short-term changes in the input (traffic bursts); (3) it should enable an ADS to achieve good accuracy points on the ROC plane; (4) it should be generic so that it can be readily incorporated into an existing ADS working on threshold manner, irrespective of the detection principles and the host/traffic features used by the ADS; and (5) it should have low run-time complexity.

In view of these constraints, we first reveal some generic statistical properties of an ADS' anomaly scores. Specifically, we use autocorrelation and conditional entropy analyses to show that real-time anomaly scores under benign conditions exhibit a decaying temporal dependence structure. Future scores can hence be accurately predicted using a few previous scores. Since a random process with decaying temporal dependence can be accurately modeled using Markov chains, we show that a low-complexity Markov-based target tracking algorithm can accurately predict future anomaly scores. In order to provide resilience against an evasive attacker, we use a low-pass filter to remove noisy short-term spikes from the observed anomaly scores

and normalize the prediction error by standard deviation of the anomaly scores over a time period. These enhancements enable the predictor to avoid learning sporadic short-term changes in the input behavior, while gradually learning long-term anomaly score trends. We show that the proposed Markovian algorithm provides significantly better prediction accuracy than the conventional Kalman filter and Holt-Winters predictors. Anomaly scores predicted using the proposed algorithm are used to define the classification thresholds for subsequent real-time measurements.

We demonstrate accuracy improvements (detection and false alarm rate) that can be achieved using the proposed technique by incorporating the adaptive thresholding module into eight prominent and diverse network- and host-based Anomaly Detection Systems (ADSs). Adaptive ADSs are evaluated on seven labeled and independently-collected datasets. All the datasets used in this work have been made publicly available except the payload-based network dataset due to privacy and nondisclosure agreement. We show that significant and consistent improvements over the best ROC operating points can be achieved by the adaptive ADSs. We also show that the proposed adaptive thresholding technique has the ability to deal with an evasive attacker who may attempt to destabilize the prediction by introducing sporadic oscillatory changes in traffic behavior. Finally, we show that adaptive thresholding has negligible complexity relative to the original anomaly detection algorithms' complexity.

The rest of the article is organized as follows. Section 2 describes the background and related work in this area. Motivation and core methodology of the proposed work is discussed in Section 3. Section 4 briefly describes the datasets and anomaly detectors used in this work (the details of datasets and detectors can be found in Appendices A and B, respectively). In Section 5, we perform statistical analysis on ADS' scores. Based on the statistical analysis, in Section 6 the adaptive thresholding algorithm is proposed along with its stability and evasion robustness. The accuracy and complexity evaluation of proposed approach are provided in Section 7. Finally, Section 8 summarizes key conclusions of this work.

2. BACKGROUND AND RELATED WORK

In this section, we first provide necessary background of anomaly-based intrusion detection systems. We then explore the related work in adaptive thresholding domain. We now provide the background of ADSs.

2.1. Background

Intrusion Detection Systems (IDS) are generally categorized into two broad classes - misuse detectors (also known as signature detection) and anomaly detectors. Misuse detection requires a signature beforehand and it is widely used in today's anti-virus programs. While signature based approaches provide 100% detection rates with no false alarms for known threats (threats for which signature has been extracted), these approaches fail to detect zero-day attacks. The rapidly growing rate of zero-day attacks add an additional complexity layer to this problem and makes it difficult to extract and disseminate threat signatures in a timely manner. On the other hand, an anomaly detector models the normal/benign behavior and tests the deviation of an unknown instance from that model. If the deviation from normal behavior exceeds a particular threshold, the unknown instance is marked as anomalous or malicious. The operation of an anomaly detection system is shown in Figure 1. Figure 1 shows the increase in traffic on port 80 due to the CodeRed worm [Moore et al. 2002]. If the number of unique hosts with traffic on port 80 are plotted and an appropriate threshold value is selected (800+ hosts in this case), one can clearly see the anomalies.

Anomaly detection can be further classified into network- and host-based anomaly detectors depending upon whether the detectors are operating on network traffic

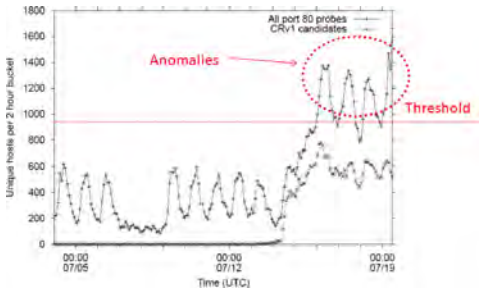


Fig. 1. Anomaly detection system operation - CodeRed significantly increased traffic on port 80 which can be thresholded to detect an anomaly [Moore et al. 2002].

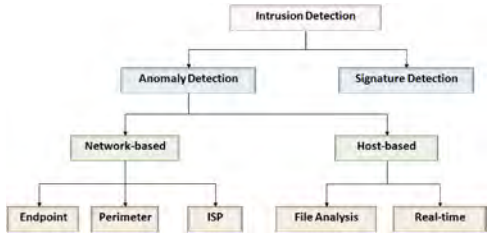


Fig. 2. A high-level taxonomy of existing intrusion detection systems.

or host characteristics. The network based anomaly detectors operate on a variety of traffic features and similarly host based anomaly detectors either statically/dynamically analyze an executable file or analyze the behavior of a host to detect an anomaly. An example of IDS classification is shown in Figure 2.

Accuracy of intrusion detection systems is usually evaluated on two competing factors, that is, detection (True Positive, TP) rate and false alarm (False Positive, FP) rate. Detection rate is calculated by observing the true alarms generated by the detector, that is, the detector classified a malicious event as a malicious one and the false alarm rate is calculated by observing the false positives generated by the detector, that is, the detector classified a benign event as a malicious one.

Generally anomaly detection systems have a tune-able parameter (or set of parameters) to adjust the detection rate and false alarm rate known as *threshold*. Tuning the threshold to different values result in some detection rate against a false alarm rate. The threshold value is varied to obtain different detection rates against different false alarm rates. The detection rates are then plotted on *y-axis* and false alarm rates are plotted on *x-axis* and the line joining these points yield a curve referred as Receiver Operating Characteristics (ROC) curve. ROC curves are used as a de-facto standard to visualize the accuracy evaluation of anomaly detectors. The steeper the curve (more area under the curve), the better the accuracy of the detector. Our work mainly focuses on the anomaly detectors that are operating on threshold method.

2.2. Related Work

Some recent network anomaly detectors provide methods to calculate thresholds for their specific algorithms [Ide and Kashima 2004; Jung et al. 2004; Lakhina et al. 2004]. For example, Ide and Kashima [2004], use ADS specific feature such as a measure associated with the number of data points in a given critical boundary and a measure of the effective size of each eigen cluster for threshold calculation. Similarly, Jung et al. [2004] models a random walk for excursion probabilities and the excursion probability is derived from two possible sets of probabilities that are specific to their detection principle. Lakhina et al. [2004] also use principal components for threshold calculation, which are features of the proposed detector. Since most of these studies do not cater for the time-varying behavior of the input, therefore these approaches fail to provide acceptable performance under varying traffic conditions [Ashfaq et al. 2008].

Some commercial ADS products also adjust their thresholds in accordance with the input traffic characteristics [Arbor PeakFlow; Cisco Anomaly Guard]. Since these products use proprietary algorithms, we can not substantiate the accuracy gains achieved

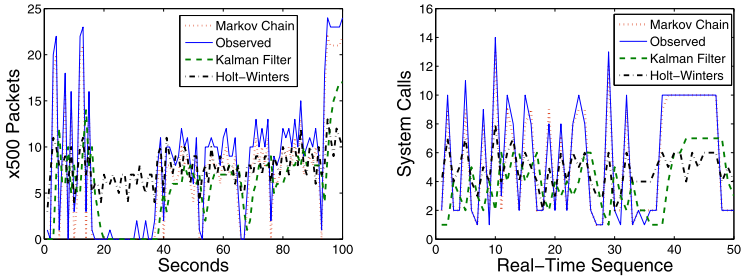
by these algorithms. Therefore, in this section, we only discuss the approaches proposed by the research community.

A model-tuning algorithm has been proposed in Yu et al. [2007], which automatically tunes the detection model of an IDS in real-time and achieves upto 20% performance improvement. However, this model is dependent on feedback by system operator thus requiring human intervention. An extension of Yu et al. [2007] has been proposed in Yu et al. [2008], which uses a prediction filter to adaptively select the most suspicious predictions so that the system operator could verify these predictions. Evaluations show that out of 304 predictions, 100 predictions were false alarms. This approach does minimize the load on system operator, however it still requires human intervention.

A performance trade-off model for ADS has been proposed in Cardenas et al. [2006], which explores Intrusion Detection Operating Characteristics (IDOC) curve as a formal framework for evaluating performance trade-off for ADSs. IDOC models the evaluation problem as multicriteria optimization problem that includes expected cost and sensitivity metrics. The IDOC curve combines all the relevant parameters that affect the practical performance of an IDS. Yet another evaluation model has been proposed [Ryu and Rhee 2008] for anomaly detectors under an inspection constraint, that is, an information security officer's limitation in incident inspection. The work discusses the analysis to help system administrator in understanding the operational, managerial and evaluation aspects of an IDS. While both of these approaches provide alternate measures to evaluate the ADSs other than ROCs, these approaches do not provide a mechanism to completely automate the ADSs process.

The most pertinent prior work in this area is Agosta et al. [2007] and Gu et al. [2006]. An ADS has been proposed in Agosta et al. [2007] which adjusts its threshold according to the variations observed in the input. However, a practical threshold adaptation technique instead of being devised for a specific anomaly detection algorithm should seamlessly operate with and provide accuracy improvements for existing algorithm operating in threshold manner. Gu et al. [2006] have proposed a set of information-theoretic metrics that can quantitatively measure the effectiveness of an IDS in terms of feature representation capability, classification information loss and overall intrusion detection capability. The proposed framework provides practical guidelines for IDS fine-tuning (static and real-time), IDS evaluation and IDS design. However, it uses the approach of calculating intrusion detection capability and other features of detectors to define a threshold at given time. On the other hand, we model the anomaly scores observed in real time by the anomaly detector.

The counterpart techniques to our work are concept-drift data stream techniques [Aggarwal et al. 2006; Cretu-Ciocarlie et al. 2009; Kolter and Maloof 2005; Masud et al. 2010, 2011]. A novel class detection algorithm along-with classification is proposed in Masud et al. [2011]. It can use any base learner with little or no modification to create and save the decision boundary that is used for classification later. Another novel class detection approach was proposed in Masud et al. [2010]. It uses adaptive threshold and the Gini Coefficient for outlier detection as part of novel class detection. A framework for concept drift evaluation and ensemble methods for handling concept drift was proposed in Bifet et al. [2009]. In Cretu-Ciocarlie et al. [2009], a self-calibration based automated framework is presented, which is agnostic to anomaly detection sensors. It uses the ensemble method but with unsupervised learning. Moreover, it builds micromodels for the adaptation purpose that are then used for anomaly detection by voting method. Chen et al. [2008] mine concepts offline from a stream and build high-quality models for each. It finds the most likely current model and classify using it. Model-averaging- and simple-voting/based framework is presented in Gao et al. [2007]. All these approaches deal with the run-time behavior change in data and updates the



(a) LBNL background traffic (b) UNM System Calls Sequence

Fig. 3. Prediction accuracy of Markov Chain on varying input data characteristics of an ADS; since packet rates in (a) are divided into equal-sized bins of 500, 0 (on y-axis) represents packets rates between 0 and 499 packets/sec.

underlying classification/baseline model. However, our approach adapts the boundary of the classification model. Most of the ensemble based approaches rely on multiclass phenomenon, however, not all the intrusion detectors build multiclass model. Therefore, concept-drift approaches may require modifications to the intrusion detectors in order to be used as base learner. However, adaptive thresholding requires no modification in intrusion detectors and treats them as black-box. Later, we will show that computational requirements are much less for adaptive thresholding than concept-drift techniques. Concept drift approaches and adative thresholding techniques can be deployed in parallel with minor modification(s).

3. MOTIVATION AND METHODOLOGY

In this section, we discuss and emphasize the rationale and high-level methodology of our work.

3.1. Motivation

An ADS' accuracy is traditionally characterized on an ROC plane by applying different classification thresholds to the ADS' anomaly scores. Points on the ROC plane are obtained by computing the average detection rate (y-axis) versus the average false alarm rate (x-axis) for each threshold value. Steepness and height of the ROC curve quantifies the accuracy that can be achieved by the ADS. While ROC evaluation is useful to get a sense for the average-case accuracy of an ADS, methods to achieve the best ROC accuracy points for arbitrary (traffic or OS) data are not well investigated.

In practice, an ADS will have to somehow learn a good classification threshold for an arbitrary benign behavior in real-time. To make matters worse, raw data that are input to an ADS typically show considerable variations. Traffic characteristics vary across organizations and network deployment points, and due to diurnal and other usage patterns. As an example, consider the LBNL background traffic rates shown in Figure 3(a) (solid line). It can be observed that the traffic rates change from approximately 500 pkts/sec to 10,000 pkts/sec within a few seconds. Similarly, host-based anomaly detection metrics are a function of user behavior, applications being used, operating system, hardware, etc. For instance, consider the benign system calls executed shown in Figure 3(b). It can be seen that system calls erratically vary from 1 to 15 in the benign traces. It can be intuitively argued that a fixed (time- and

(a) Kullback-Leibler Detector on UNM Dataset

(b) Maximum Entropy Detector on LBNL Dataset

Fig. 4. Less oscillatory variations in anomaly score with respect to variations in input (traffic or host) characteristics; thus anomaly scores are relatively easier to predict.

behavior-invariant) classification threshold cannot possibly achieve good accuracy for such time-varying input.

Finally, as input data characteristics vary, determination of a fixed threshold requires repeated manual intervention. In a typical operational scenario, a system/network administrator is responsible for adjusting the sensitivity of a network-based anomaly detector when the number of false alarms (i.e., traffic classified as malicious but which is in fact benign) increases. Similarly, host-based ADSs expect a user to adjust its sensitivity to cater for his/her security and behavioral requirements. Such repeated manual input renders an ADS less automated and more prone to configuration errors. Moreover, in a real-time system it is difficult to ascertain if a manually configured threshold is yielding good accuracy.

3.2. General Methodology

We argue that an effective ADS should automatically detect varying input data patterns and adjust its classification threshold accordingly. If accurate, such an *adaptive thresholding* mechanism will enable an ADS to achieve good operational points on the ROC plane. As a by-product, adaptive thresholding will also reduce the need for human threshold tuning, thereby making an ADS more automated.

We observed that significant variations in input data characteristics are difficult to track. Instead, it is much easier to track the anomaly score of an ADS before application of the thresholding function. As an example, Figure 4 shows variations at the input and output of a host and a network ADS. It can be observed from Figure 4(a) that, while the input (system calls) vary frequently from 1 to 160, not much variation is observed in the output (anomaly scores). Similarly, Figure 4(b) shows significant variations observed in traffic characteristics, but such erratic variations are not reflected in the anomaly scores. Hence, anomaly scores are easier to track because they reduce the high-dimensional input data to a relatively small set of scores. In addition to the complexity advantage, since these scores are coherent with input data characteristics, and as these scores comprise the domain of the thresholding function, it is intuitively likely that adaptively tracking anomaly scores yields better accuracy than direct tracking of input data. This thesis leads us to the following rationale for the adaptive thresholding technique proposed in this article: If we can accurately predict the expected values of future anomaly scores under benign conditions, the classification threshold can be adapted as a function of the predicted score.

Tracking an ADS' anomaly scores requires a robust model of scores observed under normal conditions. To develop such a model, in the later sections, we evaluate some pertinent statistical properties of anomaly scores.

4. DATASETS AND ANOMALY-BASED INTRUSION DETECTION SYSTEMS

We now briefly discuss the network and host datasets and ADSs used for evaluation. Details of these datasets and ADSs are provided in Appendix A and Appendix B, respectively.

4.1. Datasets

In order to show efficacy of proposed approach over both network- and host-based ADSs, we use four network-based datasets (containing network traffic) and three host-based datasets (containing system calls/API sequences). All the datasets that were used in this study were independently collected, labeled and are publicly available except one. Furthermore, our selection of datasets ensures the diversity. For the network traffic data, we have used datasets that employed different collection points in a network. On one hand, we use datasets from a gateway and intermediate router, and on the other hand we, use session level traffic collected at multiple end hosts. For gateway router-based dataset, we use LBNL dataset [LBNL Dataset] and ISP dataset. LBNL dataset was collected at the gateway router of Lawrence Berkeley National Laboratory (LBNL), USA. The attack traffic comprises of incoming portscans targeted towards the internal hosts which have low-rate as compared to the background (benign) traffic. ISP dataset was collected at the gateway of a leading Internet Service Provider (ISP) in Pakistan in 2012 for this work; thus it represents recent, real and large network of different users. Complete packets including payload were captured. Attack traffic comprises of brute force password, SQL injection, privilege escalation, uploading script file, starting reverse shell. We also used an intermediate router-based dataset which handles traffic from multiple research labs [NUST Dataset] representing multiple users. The attack traffic comprises of portscans and flooding attacks. The dataset was also recently collected on a large network in 2009 and was first used in Ali et al. [2010]. Finally, we use an Endpoint dataset, which contains session level traffic collected at multiple hosts which were infected using different types of portscan malware [WisNet ADS].

In order to evaluate host-based ADSs, we use Linux system calls (synthetic and empirical traces) and Windows API sequence calls. The synthetic trace from UNM [UNM Dataset] was generated by enumerating potential sources of variation for normal sendmail operations. We used intrusion traces of sscp, decode and forwarding loops provided at [UNM Dataset]. Another dataset we used was collected at MIT Lincoln Lab [MIT Dataset]. The MIT LL dataset comprises of system calls of Solaris hosts. From MIT LL dataset, we used data from the first week (starting at 03/01/1999) as benign ensembles and for labeled attacks, second week's data (starting 03/08/1999) was used. Finally, we used an API call logs dataset that was generated using 416 malware (trojan, virus and worm) and 100 benign *Win*32 executable [Nexgin Dataset]. A commercial API call tracer was used to log the API calls made by each process by inserting a kernel-mode hook for running processes on Microsoft Windows XP.

4.2. Anomaly Detectors

Before describing the real-time ADSs used in this work, we reiterate that a practical threshold adaptation algorithm should not be specific to a particular ADS. Therefore, while we have selected some ADSs for proof-of-concept and performance benchmarking, all the analysis and characterization that will be provided in the

following sections are generic and should hold across ADSs working on threshold principle.

We also emphasize that the ADSs used in this work are quite diverse in their underlying detection features and principles. For instance, the network ADSs used in this work include simple rule modelling systems like PHAD [Mahoney and Chan 2001] to complex self-learning systems like the Maximum-Entropy [Gu et al. 2005], Sequential Hypothesis Testing [Jung et al. 2004] and PAYL [Wang and Stolfo 2004] traffic anomaly detectors. Similarly, on the host side, we include a simple anomalous sequence detector [Forrest et al. 1996], a machine learning based detector [Kang et al. 2005] and sequence alignment based detectors [Gao et al. 2005; Sung et al. 2004]. This diversity is introduced to show that any real-time ADS, regardless of its functionality, can adapt to the variations in its input in order to provide acceptable performance without the considerable need of manual configuration.

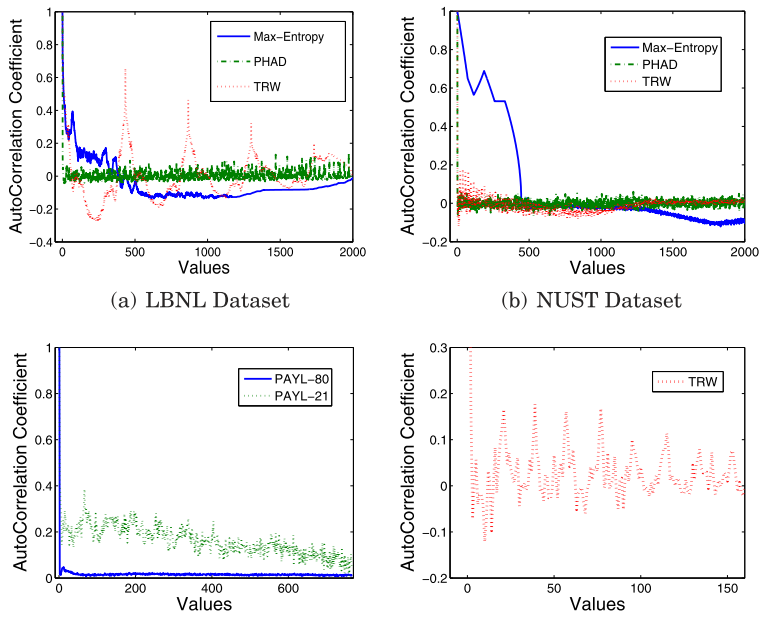
All the ADSs used in this work, were trained using 50% of the total data; the remaining 50% is used for testing. The samples between training and testing set do not overlap. For performance benchmarking, we change the anomaly detection thresholds of all detectors to generate ROC curve. All other parameters of the ADSs are the same as reported in the original papers. We now briefly discuss the ADSs used in this work; readers are referred to the original papers for further details of these algorithms.

Maximum Entropy detector [Gu et al. 2005] estimates the benign traffic distribution using maximum entropy estimation and compares it with real-time window distribution of traffic using the Kullback-Leibler (K-L) divergence measure. An alarm is raised if a packet class' K-L divergence exceeds a threshold. The TRW algorithm [Jung et al. 2004] detects incoming portscans by testing the hypothesis that the probability of a connection attempt being a success should be much higher for a benign host than for a scanner. Packet Header Anomaly Detector (PHAD) [Mahoney and Chan 2001] learns the normal range of values for 33 fields in packet headers. An anomaly score is assigned to each packet header field in the testing phase and the fields scores are summed to obtain a packet's aggregate anomaly score. Finally, Payload-based detector (PAYL) [Wang and Stolfo 2004] models the normal application payload of network traffic. It builds a normal profile by computing byte frequency distribution and their standard deviation of the application payload flowing to a single host and port. This normal profile is compared with run-time profile using Mahalanobis distance. An alarm is raised if the distance increases more than a threshold value.

The Sequence Time Delay Embedding (STIDE) [Forrest et al. 1996] builds a benign profile for a process using all unique contiguous sequences of system calls of predetermined and fixed length. The Support Vector Machines detector (SVMs) [Kang et al. 2005] considers the bag of system calls without taking into account the sequence or order of system calls. SAVE [Sung et al. 2004] finds the correlation between variants of a malware by aligning the sequence of API calls and then uses three similarity functions to determine whether the unknown file is a variant of a known malware. Bdist [Gao et al. 2005] detects variants of a known malware by finding the semantic similarity/difference between two replicas of a malware using the evolutionary distance when the input to the processes is same.

5. PRELIMINARY INVESTIGATION OF ANOMALY SCORES

In this section, we evaluate statistical properties of an ADS' anomaly scores that can be used to automatically model and adapt its classification threshold. Based on these statistical properties, the subsequent section proposes an adaptive thresholding algorithm that can accurately track the changing behavior of real-time traffic and/or OS behavior.



(a) LBNL Dataset (b) NUST Dataset

(c) ISP Dataset (d) TRW on NUST Dataset (Zoom)

Fig. 5. Decaying trend of Autocorrelation coefficient of Network ADSs anomaly scores exhibit certain level of temporal dependence; Maximum-Entropy ADS' score is plotted for port 80.

5.1. Temporal Dependence in Anomaly Scores

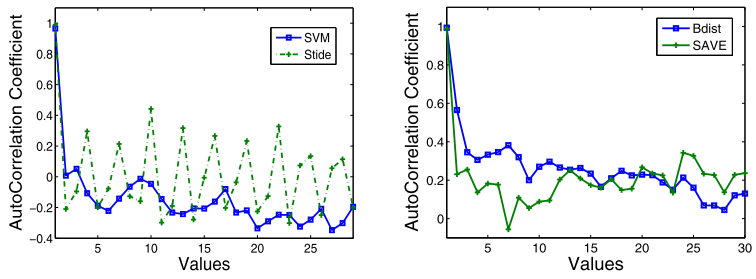
We analyzed a number of statistical properties of benign traffic and OS scores. One relevant property that provided us interesting insights into anomaly scores was the analysis of their temporal dependence. It can be intuitively argued that, as long as an ADS' input data are produced by a benign source, the anomaly scores observed at the output of an ADS should exhibit a certain level of temporal dependence. In case of an anomaly, perturbations in this dependence structure are flagged as anomalies. Therefore, the level of temporal dependence can serve as an important metric for modeling anomaly scores.

Autocorrelation measures the on-average temporal dependence between the random variables in a stochastic process at different points in time. For a given lag k, the autocorrelation function of a stochastic process X_n (where n is the time index) is defined as:

$$\rho[k] = \frac{E\{X_0 X_k\} - E\{X_0\}E\{X_k\}}{\sigma_{X_0}\sigma_{X_k}}, \tag{1}$$

where $E\{.\}$ represents the expectation operation and σ_{X_k} is the standard deviation of the random variable at time lag k. The value of the autocorrelation function lies in the range $[-1, 1]$, where $\rho[k] = 1$ means perfect correlation at lag k (which is obviously true for $k = 0$) and $\rho[k] = 0$ means no correlation at lag k.

Figures 5 and 6 show the autocorrelation function plotted against the lag for the network- and host-based ADS' scores, respectively. For all the ADSs, a high level of temporal dependence can be easily observed at small lags. This correlation decays in time and eventually drops down to a negligible value. However, the correlation decay is not consistent for all ADSs.



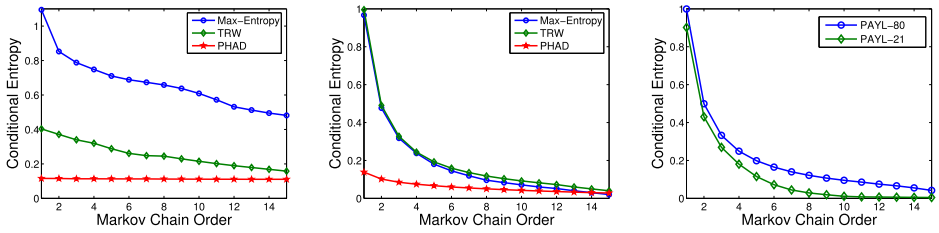
(a) UNM Dataset (b) Nexgin Dataset

Fig. 6. Decaying trend of Autocorrelation coefficient of Host ADSs anomaly scores exhibit certain level of temporal dependence.

Figure 5(a), (b) and (c) show the correlation decay for network-based ADSs on LBNL, NUST and ISP datasets, respectively. However Figure 5(d) shows the zoomed version of TRW on NUST from Figure 5(b). It can be observed that PHAD shows the steepest correlation decay on LBNL and NUST datasets; that is, one anomaly score by PHAD is correlated only with few previous scores. This can be explained by noting that PHAD's anomaly detection algorithm is not dependent on traffic rate/volume or frequency. Rather, PHAD classifies every packet on its own without borrowing any information from preceding packets. Maximum-Entropy detector shows the slowest correlation decay on both LBNL and NUST datasets because its anomaly detection algorithm operates on a port usage distribution. Port frequencies do not change significantly over time (as was also shown in Lakhina et al. [2005],) and consequently each anomaly score of the Maximum-Entropy detector is inherently correlated with prior scores. Similar trends are observed for the TRW ADS. Interestingly, when TRW is evaluated at large lags on LBNL dataset, it shows somewhat periodic correlation structure in which the correlation exhibits a high-to-low repetitive trend (instead of showing a steep decaying trend). However, on NUST dataset it drops immediately like PHAD but a periodic trend can still be observed on a smaller scale when looking closer as shown in Figure 5(d), which is a zoomed version of Figure 5(b). Similarly, decaying trend can be observed for PAYL on ISP dataset. Since PAYL also classifies every packet on its own like PHAD, the decay is steep. Decaying trend of PAYL on ISP dataset is shown for port 80 and 21, denoted by PAYL-80 and PAYL-21 respectively.

Figures 6(a) and (b) show the correlation trend for host-based ADSs on UNM and Nexgin dataset, respectively. It can be clearly seen in Figure 6(a) that STIDE and SVM both show the exponential decay on UNM dataset because both ADSs operate on chunks of system calls without borrowing much information from prior calls sequences. A very similar exponentially decaying trend is shown for Bdist and SAVE on Nexgin dataset in Figure 6(b). Decaying correlation trend is a universal property arising as a consequence of the anomaly scores' Markovian nature. In other words, irrespective of the ADS, changes in background traffic trends/OS features induce considerable decorrelation in anomaly scores over time. For example, at a given time, traffic belonging to similar sessions or system calls belonging to similar applications is observed. However, with time new network connections/applications are launched thus changing the behavior of the input to ADSs that induces decorrelation in the anomaly scores.

Based on the decaying correlation structure present in anomaly scores, we argue that a stochastic model of a few previous scores can be used to accurately predict future scores. However, to constrain the complexity of threshold adaptation, we must answer the following question: How many past anomaly scores are needed to accurately predict the next score? Interestingly, the following section shows that the answer to



(a) LBNL Dataset (b) NUST Dataset (c) ISP Dataset

Fig. 7. Decaying trend of Conditional entropy over higher markov chain order shows network ADSs future anomaly scores can be predicted using few previous scores.

this question also yields an accurate stochastic model of anomaly scores, which can in turn be used for threshold prediction.

5.2. Modeling Temporal Dependence in Anomaly Scores

It is well known that a decaying temporal dependence structure can be accurately modeled using Markov chains [Merhav et al. 1989]. Therefore, the question posed above can be rephrased as: What is the order of the Markov chain model that should be used to predict the next anomaly score? To determine the Markovian order, we use the conditional entropy based measure proposed in Merhav et al. [1989].

Conditional entropy, $H(B|A)$, of two discrete random variables A and B characterizes the information remaining in B when A is already known. Phrased differently, conditional entropy is "information about B *not* given by A." If A and B are highly correlated, most of the information about B is communicated through A and $H(B|A)$ is small. On the other hand, if p_A and p_B (which respectively represent the probability mass functions of A ad B) are quite different then $H(B|A)$ assumes a high value.[1]

To identify the order of correlation presence in the ADS scoring random process, we define a Markov chain based stochastic model as follows: Let the score at discrete time instance n represent the realization of a random variable derived from a stochastic process X_n. This process is a Markov chain if it satisfies the Markov property, which is defined as:

$$\Pr\left\{\mathrm{X}_n = j \,\middle|\, \mathrm{X}_{n-1} = i, \mathrm{X}_{n-2} = i_{n-2}, \ldots, \mathrm{X}_0 = i_0\right\} = \Pr\left\{\mathrm{X}_n = j \,\middle|\, \mathrm{X}_{n-1} = i\right\} = p_{j|i}.$$

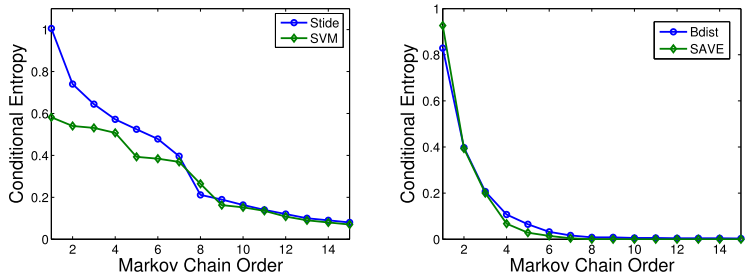
In other words, the probability of choosing a next state is only dependent on the current state of the Markov chain.

In the present context, we can define a Markov chain model X_n for an ADS' scores by dividing all possible values of the score in multiple nonoverlapping bins. Each bin then represents a state of the Markov chain, while the set of all bin indices ψ is its state space. Based on this state representation, we can define a 1st order Markov chain, $X_n^{(1)}$, in which each bin represents a state of the random process. The transition probability matrix of the 1st order Markov chain $P^{(1)}$ can be computed by counting the number of times state i is followed by state j. The resulting $|\psi^{(1)}|$ histograms can be normalized to obtain the state-wise transition probability mass functions as the rows of $P^{(1)}$.

We can find the conditional probability of the 1st order Markov chain as:

$$H^{(1)} = -\sum_{i\in\psi^{(1)}} \pi_i^{(1)} \sum_{j\in\psi^{(1)}} p_{j|i}^{(1)} \log_2\left(p_{j|i}^{(1)}\right), \tag{2}$$

[1]In the limiting cases, $H(B|A) = 0$ when $A = B$, while $H(B|A) = H(B)$ when A and B are independent.



(a) UNM Dataset (b) Nexgin Dataset

Fig. 8. Decaying trend of Conditional entropy over higher markov chain order shows host ADSs future anomaly scores can be predicted using few previous scores.

where $\pi_i^{(1)}$ is the average probability of being in state i which is computed by counting the total number of times each state is visited and then normalizing this frequency histogram.

The measure $H^{(1)}$ defines how much average information is remaining in anomaly score X_n when it is predicted using score X_{n-1}. If the present score is not highly correlated with scores before X_{n-1}, $H^{(1)}$ will be relatively large implying that information about X_n *not* provided by X_{n-1} is high. In such a case, generalizing the above discussion, we can define a higher lth order Markov chain, $X_n^{(l)}$, in which each state is an l-tuple $< i_0, i_1, \ldots, i_{l-1} >$ representing the values taken by the random process in the last l time instances. Aggregating multiple time instances in a single state allows us to satisfy the Markov property, and hence a transition probability matrix $P^{(l)}$ can be computed by counting the number of times $< i_0, i_1, \ldots, i_{l-1} >$ is followed by state $< i_1, \ldots, i_{l-1}, i_l >$. The conditional entropy of $X_n^{(l)}$ defined on $\psi^{(l)}$ can then be computed using the same method as (2). It is easy to observe that $H^{(1)} \geq H^{(2)} \geq \cdots \geq H^{(l)}$, as each older anomaly score can either be independent of or provide some information about the present score. The number of previous scores required to accurately predict the next score can then be determined by plotting $H^{(l)}$ as the function of the Markov chain order, $l = 1, 2, \ldots$. The order at which the conditional entropy saturates defines the total number of previous scores which have conveyed as much information of the present score as possible.

Figures 7(a), (b), and (c) show the conditional entropy of the network-based ADSs on LBNL, NUST and ISP dataset respectively. It can be observed that the Maximum Entropy detector shows an exponentially decaying trend on LBNL dataset until order four, after which the decay become somewhat linear. However, on NUST dataset the exponential decay is continuous. Therefore, for the Maximum Entropy detector, four previous values are sufficient to predict the next score. Similarly, TRW's scores can be predicted using approximately six prior scores on LBNL and four prior scores on NUST dataset. As expected and explained earlier, PHAD exhibits very low conditional entropy decay which is hardly visible on the y-axis scale of Figure 7(a), however a slight decay can be seen in Figure 7(b). Hence, PHAD's scores can be predicted using only one prior score. PAYL shows a continuous exponential decay on ISP dataset for port 80 and 21. Although PAYL also classify each packet individually like PHAD, it shows some dependence since it implements the online incremental algorithm, which takes into account the previous packets for updating the normal profile. Hence, PAYL's scores can be predicted using four prior scores. Similarly, host-based detectors also show the same decaying trend in Figure 8. STIDE shows the most steepest decay followed by SVM in

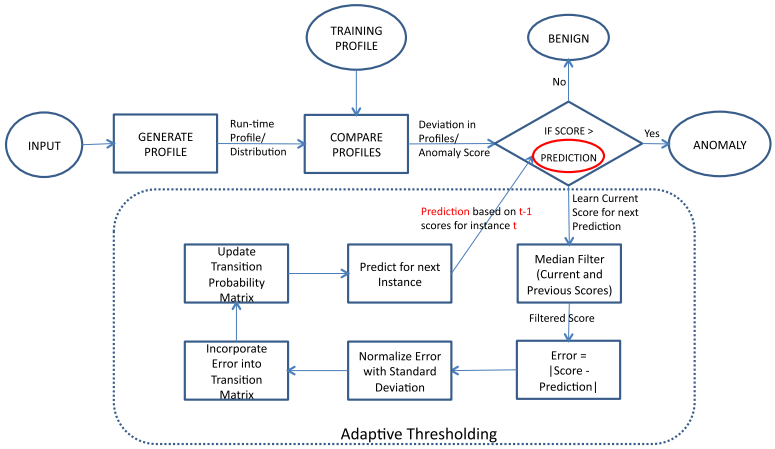


Fig. 9. Input is used to generate run-time profile which is then compared with training to produce anomaly score, which in turn is compared with the prediction (threshold) based on $t-1$ scores for instance t.

Figure 8(a). Therefore, we predict scores for STIDE and SVM detectors using eight and nine prior scores, respectively. Figure 8(b) shows almost the same exponentially decaying trend for both Bdist and SAVE. Therefore, the score for Bdist and SAVE can also be predicted using three previous scores.

At this point, in addition to having a generic method for ascertaining the number of previous scores required for future score prediction, we also know that an ADS' scores can be accurately modeled using high-order discrete Markov chains. In the following section, we use this stochastic model to develop a tracking algorithm that can accurately adapt an ADS' classification threshold.

6. ADAPTIVE THRESHOLDING ALGORITHM

Based on the results of the last section, we now propose a simple and generic Markovian anomaly score predictor. This Markovian predictor is in essence a variant of the stochastic target tracking algorithm proposed in [Hollinger et al. 2008]. At this point, it is important to reiterate that our rationale for anomaly score prediction is that the predicted scores can be used to threshold future scores in accordance with varying input characteristics. The rest of the section describes the algorithm and compares its accuracy with two well-known stochastic prediction algorithms.

6.1. Algorithm

Let us subdivide an ADS' anomaly score into k equal-sized bins, where k is determined as a by-product of the conditional entropy analysis of the last section. Specifically, the Markovian order at which the decaying conditional entropy saturates is chosen as the value of k. The size of each bin is then calculated by taking the difference of the minimum and the maximum anomaly score and dividing that difference by k. The sizes of the first and the last bin are kept flexible to accommodate any previously-unseen anomaly scores that may be observed during real-time operation.

Let $P^{(n)}$ denote the $k \times k$ transition probability matrix of the Markov chain predictor at time n, where $p_{i|j}^{(n)}$ represents an entry at the ith row and jth column of $P^{(n)}$. Also, let $r^{(n)}$ be the actual value of an ADS score observed at time instance n and let $\hat{r}^{(n)}$ be the

Markovian prediction from the last time instance. Then, the algorithm for adaptive thresholding operates as follows:

$$r^{(n)} = \text{medfilt}\{r^{(n-T)}, r^{(n-T+1)}, \ldots, r^{(n)}\}, \tag{3}$$

$$\varepsilon^{(n)} = |r^{(n)} - \hat{r}^{(n)}|, \tag{4}$$

$$\beta^{(n)} = \frac{\varepsilon^{(n)}}{\sigma_{r^{(n)}}}, \tag{5}$$

$$\tilde{p}^{(n+1)}_{r^{(n)}|r^{(n)}} = \beta^{(n)} \times p^{(n)}_{r^{(n)}|r^{(n)}}, \tag{6}$$

$$p^{(n+1)}_{j|r^{(n)}} = \frac{\tilde{p}^{(n+1)}_{j|r^{(n)}}}{\sum_{i=1}^{k} \tilde{p}^{(n+1)}_{i|r^{(n)}}}, \forall j = 1, \ldots, k, \text{ and}$$

$$\hat{r}^{(n+1)} = \max_{j=1,\ldots,k} p^{(n+1)}_{j|r^{(n)}}. \tag{7}$$

Equation (3) applies a Tth order median filter on the observed values in order to remove short-term noise from them. Filtered values are then used for the prediction. Equation (4) calculates the prediction error $\varepsilon^{(n)}$ between the predicted and the observed filtered score. The error is normalized by $\sigma_{r^{(n)}}$, which is the standard deviation of the row of $P^{(n)}$ corresponding to $r^{(n)}$. Then, at each time step n, Eq. (6) feeds the normalized error $\beta^{(n)}$ back into $P^{(n)}$ in order to adapt and learn the varying traffic or host patterns. Using this error feedback, the weight of a value observed $r^{(n)}$ is increased proportionally using the parameter β as the weight. Thus higher error means that the probabilities of these states increase proportionally and the predicted values for the next time instance will likely drift away from the current state. The updated row of the transition probability matrix is re-normalized to obtain an updated probability mass function for state $r^{(n)}$. Finally, Eq. (7) predicts the the next anomaly score, $\hat{r}^{(n+1)}$, as the state having the highest probability in the updated transition probability matrix. This predicted anomaly score is used as the adaptive threshold for time instance $n+1$.

Figure 9 shows a high-level working of the proposed adaptive thresholding algorithm with existing ADSs, which work in a threshold manner. It can be seen that it treats ADS as a black-box and takes anomaly scores produced by the ADS as input. Based on the input, it predicts the anomaly score for next time interval which is then used as a threshold. ADSs take traffic or host data (at run-time) as input which is used to generate a run-time profile/distribution. This run-time profile/distribution is then compared with training profile/dsitribution generated earlier and difference between the two is calculated, which is anomaly score. Different ADSs use different methodologies to generate and compare these distributions. In general, this anomaly score is compared with a threshold to classify the input as anomalous (if it is greater than threshold) or benign (if it is less than threshold). However, adaptive thresholding sets the threshold which is predicted based on $t-1$ earlier instances for the current instance t. Therefore, the threshold will be predicted for every instance using the previous instances' score. Since traditional ADSs work in threshold method, adaptive thresholding can be easily plugged in as shown in Figure 9.

6.2. Prediction Accuracy of the Adaptive Thresholding Algorithm

For accuracy benchmarking, we compare the prediction accuracy of the proposed algorithm with two well-known predictors, namely Kalman filter and Holt-Winters [Trees 2001]. Figures 3(a) and (b) show the accuracies of the three predictors (Markovian,

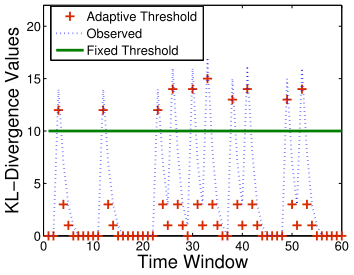


Fig. 10. Fixed threshold failed to raise an alarm for K-L divergence (Maximum-Entropy), however adaptive thresholding adjusted and raised an alarm in anomalous window.

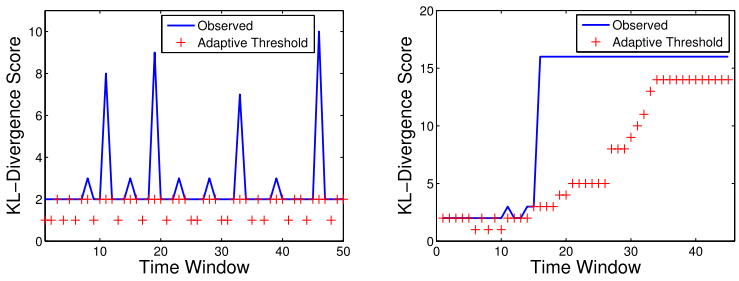
Kalman, Holt-Winters) in tracking the input (traffic and system call) trends observed in the LBNL and UNM datasets. It can be seen in Figure 3 that, while all three predictors can follow real-time measurements, the Markovian predictor tracks these measurements much more accurately than the Kalman filter and Holt-Winters predictors.

As a proof-of-concept example of the accuracy advantages that can be provided by adaptive thresholding, Figure 10 shows the Markovian-predicted thresholds in an anomalous LBNL time window for the Maximum-Entropy detector; the threshold in this case is the K-L divergence of a packet class that has been perturbed by the attack. Moreover K-L divergence values (*y-axis*) represents the equal sized bins, each containing 2 values, that is, values [0,1] belongs to bin 0. Please note that for the purpose of observing prediction accuracy, we disabled Eq. (3) by setting $T = 0$ and Eq. (5). It can be clearly seen that the Markovian predictor estimates the highly-varying K-L divergence values with remarkable accuracy.

It is also interesting to note that in Figure 10, selecting a fixed threshold might allow a few anomalies to go undetected, especially anomalies that do not cause significant perturbations in the actual network traffic. For instance, in the 60 second output shown in Figure 10, only 10 of these values cross the fixed threshold. In this experiment, the Maximum-Entropy algorithm was detecting an anomaly if 12 or more values in a 60 second window exceed the fixed threshold. Hence, this anomaly will not be detected by a fixed threshold. Adaptive thresholding, on the other hand, accurately predicts the K-L divergence in the next window and the observed (perturbed) divergence exceeds this threshold more than 20 times in a 60 second window, thereby allowing the Maximum-Entropy detector to flag the anomaly. Finally, it can be observe from Figure 10, that for a few seconds the K-L values drop to 0. These low values give a crafty attacker the leverage to introduce malicious traffic that does not exceed the fixed threshold of $[0, 10]$. However, an adaptive threshold immediately learns the change and sets the threshold to 0, thus ensuring that no room is available for such a mimicry attack.

6.3. Stability of the Adaptive Thresholding Algorithm

We have demonstrated that the proposed predictor provides considerably better estimates than the Kalman Filter and Holt-Winters predictors. In addition to efficiently learning the temporal behavior of anomaly scores, we also want the proposed predictor to exhibit stable behavior when sporadic changes in traffic/system calls are observed. Previous studies on FTP, TELNET, WWW, NNTP, and SMTP traffic have shown that significant traffic variance (burstiness) is present on a wide range of time scales [Crovella and Bestavros 1997]. Due to these changes in input characteristics, abrupt changes in anomaly scores are observed and their subsequent anomaly score predictions oscillate. While these changes are sporadic and short-term in nature, they



(a) Stability on NUST Dataset (b) Evasion on NUST Dataset

Fig. 11. Adaptive thresholding does not learn short term oscillations, however it gradually learns long term changes thereby providing stability and evasion robustness.

can destabilize a predictor and consequently degrade its prediction accuracy. These short-term changes should be treated as noise as these changes do not reflect a permanent or long-term deviations in input behavior.

In order to make the predictor insensitive to these sporadic and short-term perturbations, we apply a median filter (Eq. (3)) on the observed anomaly scores before predicting the next score. A median filter stores T previous values of the input and at each step outputs the median of the stored values. This results in removal of high-frequency spikes from the input data. It should be noted that the value of T represents a crude upper bound on the maximum duration of an anomalous short-term spike. We set T equal to the order of Markov chain model used for prediction. If the change persists longer than T observations, then it is treated as a long-term change in the input's behavior.

Figure 10 shows a prediction example for the Maximum Entropy detector without the median filter (Eq. (3)). It can be seen from Figure 10 that when there is short-term variance in input, the corresponding predicted threshold value also varies. On the other hand, when median filter is used, short-term spikes in anomaly scores are ignored which results in a more stable prediction algorithm. Figure (11)(a) shows the anomaly scores observed in real-time for the Maximum Entropy detector on the NUST dataset. It can be observed that by using the median filter the sporadic short-term bursts were filtered outand the magnitude of oscillations in the predicted threshold value is reduced significantly.

6.4. Robustness against Evasion Attacks

Evasion attacks can be mounted on the proposed adaptive thresholding algorithms by an attacker who can intelligently craft network traffic or system calls that change ADS' input behavior. This will allow an attacker to launch an attack without being detected by changing the ADS' impression of benign traffic to match more closely that of the attack; hence, the attack would be masked by the ADS' expectation of benign traffic and would be guaranteed to fall below the threshold. In order to provide resilience against such an evasion attack, the adaptive thresholding algorithm should not immediately adapt to the input. Therefore, robustness against evasion is in terms of learning time. Since the threshold is not fixed and it adapts, the concern here is how quickly it adapts and stabilizes. If it adapts on a few samples (i.e., in short period of time), then a crafty attacker can change the normal behavior by crafting few samples in a short time and then launch an actual attack to go undetected. However, the adaptive thresholding algorithm is designed to learn slowly based on the difference in observed and predicted score with respect to standard deviation. This standard

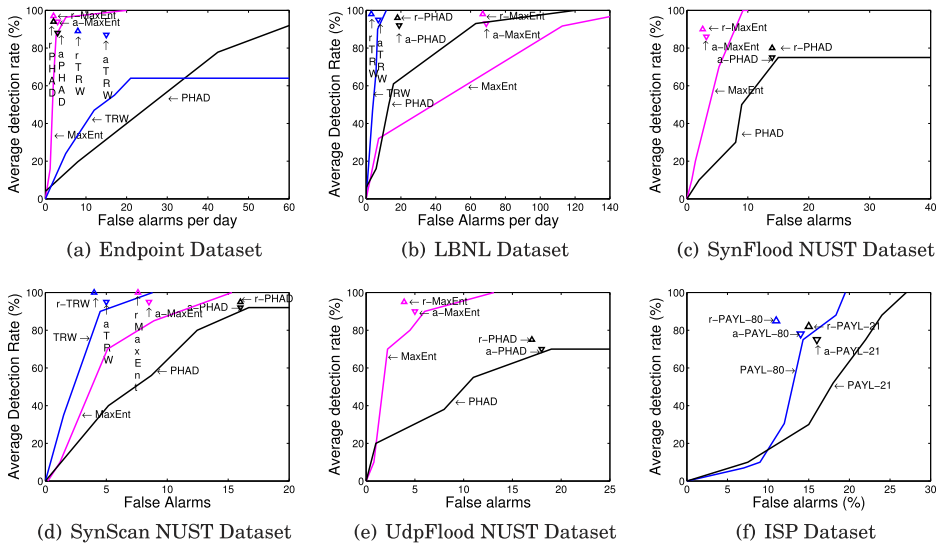
deviation makes it learn slowly, and consequently an attacker has to wait longer in order to go undetected by making the ADS learn the new trend.

For example, if an attacker wants to launch a Denial of Service (DoS) attack, he can send benign packets that slightly follow the characteristics of the attack as determined by the ADS (such as a specific distribution for Maximum Entropy detector). The ADS would see the change in traffic characteristics and provided that this traffic does not exceed the threshold, the ADS would adaptively increase its threshold to manage the changed nature of the benign traffic. After some time, the attacker can once again manipulate the benign traffic characteristics to be closer to the desired attack traffic. Steadily but surely, the ADS will adapt to this and once the threshold for the DoS attack detection is sufficiently changed, the attacker can eventually launch the DoS attack and remain below the threshold level.

In order to circumvent such attacks, the proposed adaptive thresholding algorithm normalizes error (difference between observed ADS score and predicted score) with respect to the standard deviation of the error, which takes longer to learn the change in the input behavior. Equation (5) in the algorithm allows the adaptive thresholding algorithm to take longer to learn such changes. Comparison of Figure 11(b) with Figure 10 shows the evasion robustness capability of the proposed thresholding algorithm. In our experiment, since robustness against evasion is a characteristic of adaptive thresholding, for proof of concept we used Maximum Entropy detector which builds profile based on destination port and protocol. It calculates the KL-divergence for normal and run-time profile thus comparing it with a threshold. To increase the divergence we launched TCP SYN (100, 1000 pkts/sec) on various destination IP and ports. It can be seen in Figure 11(b) that the behavior of the traffic was changed which is denoted with a solid line. As soon as the behavior was changed using high rate instances of TCP SYN portscans, low rate targeted UDP flood (0.1, 1, 10 pkts/sec) attack was launched on two remote servers to bypass the detection. It can be observed in Figure 11(b) that algorithm gradually adapts to the behavioral change in the input thus raising an alarm for UDPFlood attack. This allows the system to raise alarms before becoming a victim to evasion attacks and triggering suitable policy to take care of such an evasion attack. In order to go evasive, an attacker has to change the behavior of traffic for a long time until the adaptive thresholding adapts the observed behavior and then launch an actual attack to bypass detection. However, we believe that changing the behavior for a long time will make it more detectable.

7. EVALUATION

The proposed adaptive thresholding algorithm is *generic* because it does not rely on the detection features and principles of the underlying ADSs and can be easily plugged into *any* existing ADS working on the threshold principle. We now evaluate the proposed thresholding algorithm by incorporating it into eight ADSs. We expect the adaptive thresholding algorithm will choose considerably better threshold points on an ROC plane as compared to a fixed threshold value. We present results of our evaluation for two versions of our adaptive thresholding algorithm: (1) only adaptive thresholding module that is without stability and evasion robustness (these results are labeled as a-ADS name); and (2) results with stability and evasion robustness modules (labeled as r-ADS name). In addition to the performance improvements achieved over existing ADSs, the accuracy comparison with the two counter-part techniques for concept-drifting, that is, ECSMiner [Masud et al. 2011] and Multi Class Miner (MCM) [Masud et al. 2010], is shown. Finally, we also show that the additional complexity introduced by the adaptive thresholding is negligible as compared to the complexity of ADS without adaptive thresholding module.



(a) Endpoint Dataset (b) LBNL Dataset (c) SynFlood NUST Dataset

(d) SynScan NUST Dataset (e) UdpFlood NUST Dataset (f) ISP Dataset

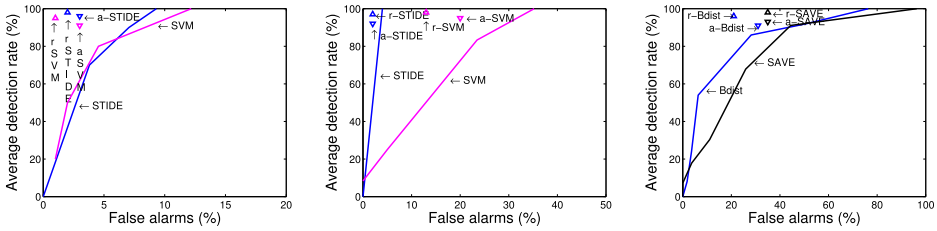
Fig. 12. Accuracy improvements achieved by adaptive thresholding over network-based ADSs.

7.1. Accuracy Evaluation

Before discussing the accuracy improvements achieved, we first explain the implementation of proposed algorithm with existing detectors used in our work. For Maximum Entropy, we plugged the proposed detector to predict the next time instance KL-divergence value (since KL value is used as a tuning parameter). Since PHAD calculates a packet score based on the header field values, we use the adaptive thresholding module to learn and predict the future packet score values. TRW operates on the likelihood ratio to classify a source as a scanner. We used our algorithm to predict the next likelihood ratio value. More specifically, we applied it on the upper bound in the TRW algorithm, which leads to the hypothesis that the source is a scanner. PAYL works on computing the Mahalanobis distance between run-time profile and normal profile, therefore, adaptive thresholding is plugged in to predict the next packet's distance from the normal profile.

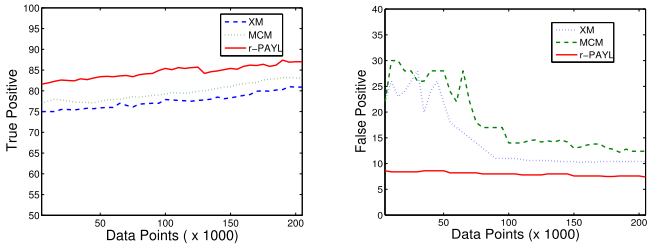
Similarly for STIDE, the proposed algorithm is used to predict the percentage mismatch between the sequences from benign profile and the sequences from malicious profile. In case of SVM, the proposed algorithm is used to determine the sum of kernels for the next set of frequency of system calls to the detector. Finally for SAVE and Bdist, the combined correlation value and the evolutionary distance threshold value is predicted, respectively. Both of these values are predicted whenever an API call is added to the existing sequences.

Figure 12 shows the ROC-based accuracy comparison of the Maximum Entropy, TRW, PHAD and PAYL detectors with and without adaptive thresholding. For Endpoint dataset, Figure 12(a) shows that a-PHAD considerably reduces PHAD's false alarms by learning legitimate changes in the network traffic behavior. Similarly, r-PHAD further improves accuracy of a-PHAD by providing a 10% reduction in false alarms and a 4% improvement in detection rate. Qualitatively similar results can be observed for a-TRW and r-TRW. We do not get considerable improvements in case of a-MaxEnt detector on the Endpoint dataset because the original algorithm provides high accuracy. However, marginal improvements can be observed in case of r-MaxEnt.



(a) UNM Dataset (b) MIT-LL Dataset (c) Nexgin Dataset

Fig. 13. Accuracy improvements achieved by adaptive thresholding over host-based ADSs.



(a) Detection on ISP Dataset (b) False Positive on ISP Dataset

Fig. 14. Accuracy comparison of XM, MCM and Adaptive Thresholding on ISP Dataset.

Figure 12(b) shows the accuracy comparison on LBNL dataset. It can be observed that MaxEnt failed to maintain performance gains as compared to the Endpoint dataset because of the large number of false alarms introduced by the erratic traffic variations at the edge router. In case of a-MaxEnt, considerable improvements in detection accuracy and reduction in false alarms are observed due to the ability of algorithm to learn the traffic variations. Similarly, r-MaxEnt further reduces false alarms by another 2% and provides slight improvement in detection accuracy. Qualitatively similar results can be observed for a-PHAD and r-PHAD. Marginal improvements were observed for a- and r-TRW due to TRW's existing high detection rate.

From NUST dataset we report results for SynFlood, SynScan and UDPFlood attacks. It can be observed from Figure 12(c), Figure 12(d), and Figure 12(e) that consistent and considerable accuracy improvements are achieved by adaptive ADSs over their original counterparts. It should be noted that the results for TRW on SynFlood and UDPFlood attacks are not provided since TRW uses reply packets for its detection and reply packets for SynFlood and UDPFlood do not exist. Another interesting observation is that the r-PHAD provides performance improvements as compared to a-PHAD because r-PHAD did not learn the traffic variations immediately and thus improves the detection accuracy by 7%.

Figure 12(f) shows the accuracy comparison of PAYL detector on ISP dataset for port 80 and 21 denoted by PAYL-80 and PAYL-21 respectively. It can be seen that consistent improvements are seen for adaptive thresholding over the original algorithm. It is shown that r-PAYL improves the detection rate in both the cases, however, considerable reduction in false alarm was only observed in case of port 80.

Figure 13 shows the ROC-based comparison of STIDE, SVM, Bdist and SAVE detectors with and without adaptive thresholding versions. It can be observed that the a-STIDE and r-STIDE provide up to 30% improvement in detection accuracy and an upto 50% reduction in false alarm rates on UNM dataset. Qualitatively similar improvements are achieved using a-SVM and r-SVM on both UNM and MIT LL dataset.

SAVE failed to provide acceptable accuracy on Nexgin dataset, however, r-SAVE provided up to 15% improvement in detection rate and considerable reduction in false alarms. Finally, some improvement in detection rate were observed for r-Bdist over the original and a-Bdist.

Figure 14 shows the accuracy comparison of adaptive thresholding with the two recently proposed counter-part techniques, that is, ECSMiner (XM) [Masud et al. 2011] and Multi Class Miner (MCM) [Masud et al. 2010]. PAYL detector was used as a base learner for both the techniques. We used the same configuration parameters as in Masud et al. [2011]. Number of pseudopoints per classifier was set to 50. Ensemble size of 6 was used. Chunk size was kept to 2000. We ignored the novel class detection accuracy since that is out of the scope of our work. However, we only focused on the attack detection accuracy and false positives on benign traffic. ISP dataset for port 80 was used. It can be clearly observed that r-PAYL outperforms both the techniques since stochastic angle brings in value in dealing with sporadic behaviors as compared to deterministic mechanism.

7.2. Complexity Evaluation and Implementation

We measured the algorithm run-time complexity and memory requirements using the `hprof` tool [Heap Profiler] on a 2.2GHz dual-core Intel machine. Complexity was measured by running the algorithms on the endpoint, ISP and UNM dataset. The complexity for legacy (a) and robust (r) adaptive thresholding modules on the endpoint's traffic data was approximately 60 and 80 milliseconds respectively, which was two and five orders of magnitude less than the time taken for execution of the Maximum-Entropy and TRW/PHAD anomaly detection algorithms, respectively. For example, complexity observed for Maximum entropy was 22.79 seconds. However, when employed with robust adaptive thresholding technique, the complexity turned out to be 22.87 seconds (i.e., 80 milliseconds overhead). Similarly, the data memory requirements of adaptive thresholding algorithm were also negligible (a few hundred KBs, approx 400) with respect to the memory consumed by the anomaly detection algorithms; three orders of magnitude for Maximum-Entropy and two orders of magnitude for TRW and PHAD. Similar complexity results (approximately two orders of magnitude in run-time and an order of magnitude in data memory) were observed for host-based detectors. PAYL detector complexity was observed to be 21.31 seconds. However, it was 21.39 seconds when employed with adaptive thresholding. Therefore, adaptive thresholding introduced an overhead of 80 milliseconds on a 2.2GHz machine. XM and MCM observed the complexity of 28.35 and 26.67 seconds, respectively. Similarly, much higher memory requirement was seen for them, that is, few MBs as compared to adaptive thresholding which requires few KBs.

Implementation of the adaptive thresholding with existing ADSs is fairly simple. ADSs quantifies the difference in normal and run-time profile. This quantifiable measure is compared against a fixed threshold in the classifier. Adaptive thresholding can be implemented in two ways: (1) replace the fixed threshold number with a variable whose value can be set by the adaptive thresholding module, (2) replace the fixed threshold number with a method call that returns the new adaptive threshold number. The benfit of the second approach is that the run-time observed value can be sent to adaptive thresholding as an argument for learning in order to predict the new threshold which is returned.

7.3. Limitations

— Adaptive thresholding can only be applied to the real-time detectors that work in a threshold manner, that is, detectors that compare learned profile with the run-time profile using a quantifiable measure against a threshold for classification. Therefore,

it can not be applied to most of the machine learning or clustering based detectors like file analysis (machine learning) and rule-based traffic classification algorithms.
— Some anomalies may go undetected if the attack traffic gradually manipulates the behavior, that is, low-rate attacks that generate very less traffic as compared to normal traffic. A simple countermeasure to this problem is to set the appropriate median filter window length. The larger the window size, the slower is the adaptation rate, that is, it tends to act like fixed threshold. In our experiments, we tested the approach on low rate attacks as well in NUST dataset, that is, as low as 0.1 pkts/sec.

8. CONCLUSION

We showed that ROC-based accuracy evaluation using fixed thresholds is not necessarily representative of the actual accuracy that an ADS can potentially achieve. Moreover, we presented the extension of our previous algorithm [Ali et al. 2009] and demonstrated further performance improvements on three important metrics: accuracy, stability and evasion robustness. We proposed a generic threshold tuning algorithm that: (1) allows an ADS to achieve high accuracy points on the ROC plane; (2) can be readily introduced into existing ADSs; (3) reduces the need for human threshold configuration in an ADS; and (4) has very low run-time complexity and memory requirements.

APPENDIX

A. DATASETS USED FOR EVALUATION

In this section, we describe the network traffic and system call/API sequence datasets used for evaluation in this work.

A.1. Network Traffic Datasets

We use four traffic datasets that have been independently-collected at different deployment points. All the datasets are labeled and publicly available at [LBNL Dataset], [WisNet ADS], and [NUST Dataset] except ISP dataset.

A.1.1. LBNL Dataset. This dataset was collected at two international network locations at the Lawrence Berkeley National Laboratory (LBNL), USA. The main applications in internal and external traffic were Web (HTTP), Email and Name Services. Some other applications like Windows Services, Network File Services and Backup were also being used by internal hosts. Malicious traffic mostly comprises failed incoming TCP SYN requests targeted towards LBNL hosts; see Pang et al. [2005] for details. Some pertinent statistics of the LBNL dataset are given in Table I. Note that the attack rate is significantly lower than the background traffic rate. Thus these attacks can be considered low rate relative to the background traffic rate. We filtered local traffic from the dataset.

A.1.2. Endpoint Dataset. This dataset comprises session-level traffic collected at 13 network endpoints. The users of these endpoints included home users, research students, and technical/administrative staff. The endpoints were running different types of applications, including peer-to-peer file sharing software, online multimedia applications, network games, SQL/SAS clients, etc. Statistics of the highest and lowest benign traffic rate endpoints are shown in Table II. Attack traffic in this dataset mostly comprises outgoing portscans; see Ashfaq et al. [2008] for details. Attack traffic was generated using the following malware: `Zotob.G`, `Forbot-FU`, `Sdbot-AFR`, `Dloader-NY`, `SoBig.E@mm`, `MyDoom.A@mm`, `Blaster`, `Rbot-AQJ`, and `RBOT.CCC` [Symantec Security]. Table III shows statistics of the highest and lowest scan rate worms. For completeness, we also

Table I. Background Traffic Information for the LBNL Dataset

Date	LBNL Hosts	Remote Hosts	Avg. Background Pkt Rate(/sec)	Avg. Attack Pkt Rate(/sec)
10/4/04	4,767	4,342	8.47	0.41
12/15/04	5,761	10,478	3.5	0.061
12/16/04	5,210	7,138	243.83	72

Table II. Background Traffic Information for the Highest and Lowest Rate Endpoints

Endpoint Type	Duration (months)	Total Sessions	Avg. Session Rate(/sec)
Home	2	444,345	5.28
University	9	60,979	0.19

Table III. Endpoint Attack Traffic for Two High- and Two Low-Rate Worms

Malware	Avg. Scan Rate(/sec)	Port(s) Used
`Dloader-NY`	46.84	TCP 135,139
`Forbot-FU`	32.53	TCP 445
`MyDoom-A`	0.14	TCP 3127 – 3198
`Rbot-AQJ`	0.68	TCP 139,769

simulated three additional worms, namely `Witty` (worm with fixed source port 4000), `CodeRedv2` (worm with fixed destination port 80) and a low-rate TCP worm (with a fixed and unusual source port 2200). `Witty` and `CodeRedv2` were simulated using the scan rates, pseudocode and parameters given in Shannon and Moore [2004] and Symantec Security.

A.1.3. NUST Traffic Dataset. Traffic was collected at a router at National University of Sciences and Technology, Islamabad, Pakistan (NUST), which handled traffic from three distinct research labs. These labs had approximately 50 hosts in total. Ports were mirrored on the router to receive the traffic (inbound, outbound and internally routed). The dataset was collected in 2009 and was first used in Ali et al. [2010]. Thus, it is recent, large, and real traffic dataset.

For attack traffic, port scans (TCP SYN), DoS (TCP SYN) and fraggle (UDP flood) were launched simultaneously from three end hosts in a research lab. The DoS attacks were launched on two servers under our administration with public IP addresses. Each attack was launched for a period of five minutes with spoofed IP address. For every attack type, three low-rate (0.1, 1, 10 pkts/sec) and two high-rate (100, 1000 pkts/sec) instances were launched. The attack characteristics for each attack are shown in Table IV. The normal traffic was captured in six periods, each one of over three hours. During traffic capturing, different applications were hosted on the machines including file transfer, web browsing, instant messaging, real-time video streaming, etc.

A.1.4. ISP Dataset. ISP dataset was collected at a leading Internet Service Provider (ISP) of Pakistan in 2012. Under the nondisclosure agreement, we can not make the dataset publicly available. The dataset was collected at the gateway router. Full packets were captured along with the payload. It was collected by port mirroring at a gateway switch. Tcpdump tool [Tcpdump tool] was used to log the traffic destined for port 80 and 21 of the hosted servers. For port 80, the traffic was collected for 3 hours. On average 274.37 pkts/sec were observed with a rate of 1.69 mb/sec. Total volume of the traffic recorded was 2.41 GB. However, there was not much activity on port 21 (ftp server) as compared to the port 80, 1GB of traffic was collected in 14 hours.

Table IV. Background Traffic Information During Attacks

Attack Name	Attack Characteristics	Attack Rate (pkts/sec)	Background Traffic Rate (pkts/sec)	
			Wing Router	
			μ	σ
TCP-SYN portscans	Random dest IP addr Fixed src IP addr Two distinct attacks: First scan on port 80, Second scan on port 135	0.1	2462.9	474.4
		1	3002.6	398.0
		10	3325.2	397.7
		100	6100.0	2492.4
		1000	3084.7	247.4
TCP-SYN flood (DoS)	Spoofed src IP addr Two remote servers attacked Attacked ports: 143, 22, 138, 137, 21	0.1	2240.1	216.7
		1	2699.1	328.8
		10	4409.8	1666.2
		100	3964.1	1670.4
		1000	3000.9	238.0
UDP flood fraggle	Spoofed src IP addr Two remote servers attacked Attacked ports: 22, 80, 135, 143	0.1	2025.8	506.4
		1	2479.1	291.0
		10	4028.4	1893.1
		100	6565.7	3006.9
		1000	2883.7	260.8

Attack traffic was generated using tools like [Sqlninja tool], [Netsparker tool], and [FTP Brute Forcer]. Multiple instances of different attacks like brute force password, SQL injection, privilege escalation, uploading script file, starting reverse shell, were generated and mixed in the traffic after labeling. The experimentation was done in the environment provided by ISP. The dataset is recent, large, and realistic, representing variety of users of an ISP.

A.2. Host-Based Datasets

For host-based experiments, we choose widely-used and publicly available datasets from UNM [UNM Dataset] and MIT Lincoln Lab (LL) [MIT Dataset]. We also use a more recent dataset from Nexgin [Nexgin Dataset]. Before we provide a description of these datasets, we highlight the problems with the UNM and MIT LL datasets, which has also been discussed in length in Merhav et al. [2010]. Both of these datsets are fairly outdated and even the operating systems used to collect these datasets have changed (all the machines involved are Solaris version 2.5.1 hosts, which are ancient nowadays). We also note that most of the attacks in these datasets are outdated and contemporary attacks are much more complex. However, due to the lack of availability of other labeled data sets, most existing researches use these datasets.

A.2.1. UNM System Calls Sequences. The University of New Mexico (UNM) dataset provides system call traces for various processes. Forrest et al. [1996] argue that monitoring the behavior of a process might not cover the full spectrum of normal behavior as some processes behave in quite a varied manner. Therefore, they artificially generate system call sequences. We used the synthetic `sendmail` traces for our experiments. These traces were generated by enumerating potential sources of variation for normal `sendmail` operations. Trace files contained process IDs and their respective system calls. For intrusive traces, we use 3 traces of *sscp* intrusions, 2 traces of decode intrusions, and 5 traces of forwarding loop traces provided by UNM on its website. Table V shows the intrusions and their instances used; see [UNM Dataset] for details.

A.2.2. MIT Lincoln Lab System Call Sequences. From the MIT Lincoln Lab dataset [MIT Dataset], we used Solaris BSM audit data, which provides system calls of a Solaris

Table V. UNM Intrusion Instances

sscp	decode	forwarding loops
3	2	5

Table VI. MIT-LL Intrusion Instances

httptunnel	land	ps	eject	ftp-write	secret	mailbomb
2	2	2	2	2	1	2

host as BSM logs. The BSM file contains information of process IDs, system calls, user IDs, machine name, description, and date/time. A separate network traffic analysis data file is also provided which indicates inbound network connections to the system. Table VI lists the attack instances in this dataset; see [MIT Dataset] for details. We cross-indexed the system call trace file with network traffic data file using the arguments to the `exec` system call. Furthermore, a tolerance of one second was chosen as suggested by Kang et al. [2005] in order to match the majority of connection attempts. We used benign data from the first week (03/01/1999) and labeled attacks from second week (03/08/1999).

A.2.3. Nexgin Dataset. For the ADSs that operate on Windows API call sequences, we used the Nexgin dataset [Nexgin Dataset]. The Nexgin dataset was collected by executing benign and malicious executables on a Microsoft Windows XP machine and a commercial API call tracer was used to log the API calls made by the processes using a kernel-mode hook. To reduce the complexity of run-time tracing, the authors short-listed 237 core API calls from six different functional categories such as socket, memory management, processes, and threads etc. The authors have collected system call logs of 100 benign programs, 117 trojans, 165 viruses, and 134 worms. The malware samples provided in this dataset uses proper nomenclature for malware which makes it easier to identify the family and variants of different malware.

B. ANOMALY DETECTION SYSTEMS

In this appendix, we briefly describe the anomaly detection algorithms used for accuracy evaluation. Before describing the real-time ADSs used in this work, we reiterate that a practical threshold adaptation algorithm should not be specific to a particular ADS. Therefore, while we have selected some ADSs for proof-of-concept and performance benchmarking, all the analyses and characterizations that will be provided in subsequent sections are generic and should hold across different real-time ADSs. We also emphasize that the ADSs used in this work are quite diverse in their underlying detection features and principles.

All the ADSs used in this work were trained using 50% of the total data; the remaining 50% is used for testing. For performance benchmarking, we change the anomaly classification thresholds of all detectors to generate ROC curves. All other parameters of the ADSs are the same as reported in Forrest et al. [1996], Gu et al. [2005], Jung et al. [2004], Wang and Stolfo [2004], Kang et al. [2005], Mahoney and Chan [2001], Sung et al. [2004], and Gao et al. [2005].

To employ adaptive thresholding with the ADSs, 15th order markov chain is used since most of the ADSs show low conditional entropy on this order. Although few detectors could operate on lower order but to keep it simple we used the same order. Median filter was applied on the 15 values in order to remove the sporadic changes. Therefore, the transition probability matrix is of the order 15×15. For each ADS, the minimum and maximum anomaly score were calculated on training data. These anomaly scores were mapped to 15 equal sized bins. First and last bin were kept flexible to accommodate unseen anomaly scores, that is, greater than maximum or less than minimum.

Therefore, r^n in Eq. (3) is the bin representing the anomaly score. Other parameters of the algorithm, Eqs. (4) and (5), are calculated based on the bin observed. However, the transition probability matrix is updated and predicts the next bin using Eqs. (6) and (7). We now briefly discuss the ADSs used in this work; readers are referred to the original papers for details of these algorithms.

B.1. Network ADSs

In this section, we briefly discuss the network-based anomaly detection systems.

Maximum Entropy Anomaly Detector [*Gu et al.* 2005]. It estimates the benign traffic distribution using maximum entropy estimation. Traffic was divided into 2348 packet classes. Empirical distribution was calculated for the training traffic. TADM toolkit [TADM toolkit] was then used to estimate parameter for maximum entropy model. This parameter estimation alongwith distribution was used to generate baseline distribution. Packet class distributions observed in real-time windows were then compared with the baseline distribution using the Kullback-Leibler (K-L) divergence measure. If probability distribution of a particular class in baseline distribution is 0, it would result in infinite K-L divergence. To avoid this, a large value (greater than threshold) was used for K-L divergence. An alarm was raised if a packet class' K-L divergence exceeds a threshold, k, more than h times in the last W windows of t seconds each. ROCs were generated by varying k. However, the number of windows W was set to 60 and each window t was of size 1 second. The value of h was set to 30. To this end, adaptive thresholding was applied on a packet class' K-L divergence to predict the threshold every t seconds based on previously observed K-L divergence scores for that class.

Threshold Random Walk (TRW) Algorithm [*Jung et al.* 2004]. TRW detects incoming portscans by noting that the probability of a connection attempt being a success should be much higher for a benign host than for a scanner. To leverage this observation, TRW uses sequential hypothesis testing (i.e., a likelihood ratio test to classify whether or not a remote host is a scanner). This likelihood ratio is compared with two thresholds; upper and lower bound, η_1 and η_0 respectively. If likelihood ratio is greater than η_1, the remote host is classified as scanner and if it is less than η_0, it is classified as benign. However, if the likelihood ratio is neither above η_1 and nor below η_0, it waits for more samples until the likelihood ratio hits one of these bounds. It put the limits on the bounds such as $\eta_1 \leq \frac{P_D}{P_F}$ where P_D and P_F are detection and false positive probabilities respectively. It reports that it is not a priori clear how to pick these thresholds. However, the probabilities P_D and P_F can be replaced with user chosen β and α respectively. To this end, ROCs were generated by varying the values of these bounds. However, adaptive thresholding was applied on the upper bound η_1 of the likelihood ratio, that is, if the actual ratio exceeds the predicted ratio (set as upper bound), the host will be classified as scanner. Lower bound η_0 was kept fixed. Adaptive thresholding was used to predict the likelihood ratio for every instance based on previos instances, which in turn was set as upper bound.

Packet Header Anomaly Detection (PHAD) [*Mahoney and Chan* 2001]. PHAD learns the normal range of values for all 33 fields in the Ethernet, IP, TCP, UDP and ICMP headers. An anomaly score is assigned to each packet header field in the testing phase and the fields' scores are summed to obtain a packet's aggregate anomaly score. We evaluate PHAD-C32 [Mahoney and Chan 2001] using the following packet header fields: source IP, destination IP, source port, destination port, protocol type and TCP flags. The top n values are thresholded as anomalous. ROCs were generated by varying the value of n, however, adaptive thresholding was set on the packet's aggregate anomaly score. Therefore, the value of n was not fixed and made adaptive.

Payload-Based Anomaly Detection (PAYL) [*Wang and Stolfo* 2004]. PAYL computes the byte frequency distribution and their standard deviation of the application payload flowing to a host and port using n-gram model. Here n is set to 1 since it is done for number of occurrences of each byte in the payload. This is done for all the observed payload lengths separately. To learn the normal profile on the fly, incremental learning is used for updating the distribution. Since it is done for all the payload lengths observed, it yields large number of models. Clustering is done to reduce the number of models by comparing the neighboring models (length bins $i - 1$ and $i + 1$) using Manhattan distance. If the distance was less than a threshold t (which was set to 0.5), the models were merged using incremental learning approach. Since the traffic was not sanitized and may contain attacks, unsupervised learning approach was implemented to remove the noise. Learned model was applied to training data in order to identify the outliers and the training was done again after removing the outliers. Outliers were identified using detection principle of algorithm that is, calculating mahalanobis distance between trained model and observed packet. If the distance was more than a threshold d (which was set to 256), packets were classified as outliers. During detection phase, there is a possibility that the observed length was not observed during training. In that case the run-time distribution was compared to the neighboring model having closest length. To avoid the infinite distance in-case of zero standard deviation (which may happen if a byte never appeared in training or appeared with exact same frequency in each sample), smoothing factor of 0.001 was added to the standard deviation. In order to generate ROC, mahalanobis distance threshold d was changed to observe different detection/false positive rate. However, adaptive thresholding was applied on the mahalanobis distance and d was set adaptively based on the previous samples observed of the same length.

B.2. Host ADSs

In this section, we briefly discuss the host-based anomaly detection systems.

Sequence Time Delay Embedding (STIDE) [*Forrest et al.* 1996]. STIDE detects anomalous system calls of an intrusive process. STIDE first builds a normal profile for a process by sliding a window of size $k + 1$ across the trace of system calls and record which calls follow which within the sliding window. Once we have a database of patterns, we check new trace against it by sliding the new trace with window size $k + 1$ against the database. If the sequence of system calls differs from that recorded in the normal database, the trace generates mismatches. We record the number of mismatches as a percentage of the total possible number of mismatches. For our tests we used $k = 6$. A threshold value is applied on percentage mismatches to classify anomalous behavior. The adaptive thresholding algorithm is then applied to predict the anomaly score (percentage mismatches) for the next sequence of system calls. For example, output from the each lprcp intrusion trace was used to predict the output for the next lprcp intrusion trace.

Support Vector Machines (SVMs) Using Bags of System Calls Approach [*Kang et al.* 2005]. Kang et al. [2005] propose a bag of system calls representation for detection of intrusive system call sequences. During conversion to the bag of system call representation, frequency of each system call in the bag is preserved and the ordering information between system calls is lost. Each normal process execution instance is used to create only one feature that is defined as an ordered list of the frequency of all the system calls in that execution sequence. The unknown instance is then compared with the normal instances to generate classifier score using SVM. Similar to STIDE, the adaptive thresholding algorithm then uses output from one intrusion trace to predict the anomaly score (SVM classifier score) for the next intrusion trace.

Behavioral Distance (Bdist) [*Gao et al.* 2005]. The Behavioral Distance approach works on the hypothesis that there should be semantic similarity between two replicas of a process when the input to the processes is same. In order to determine the behavioral change between replicas of a process, the authors correlate a particular system call phrase (subsequence of system calls that frequently appear together in program executions) emitted by one replica with a system call phrase emitted by the other replica. The correlation is calculated by using a distance measure called evolutionary distance, which is the sum of the costs of substitutions, deletions, insertions and misalignments. The distance measure is calculated using different alignments and the alignment that resulted in smallest behavioral distance was selected. For phrase extraction and reduction, the same algorithms that were used by Gao et al. [2005] (i.e, TEIRESIAS) were used. For our experiments, we used the Nexgin dataset to identify variants of a malware. To this end, threshold was applied to behavioral distance score of every API call phrase. Please note that although Gao et al. [2005] only provide evaluation results on Linux system calls, they have explicitly stated in their article that this approach has been proposed for both Linux system calls and Windows API calls and we use it on Windows API calls. For adaptive thresholding, the output of one variant of malware was used to predict the next anomaly score threshold value (distance score threshold). For example, the threshold output from w32.Mydoom.A was used to predict the threshold for another polymorphic version of W32.Mydoom.A.

Static Analyzer for Vicious Executables (SAVE) [*Sung et al.* 2004]. SAVE detects obfuscated (or polymorphic) and mutated (or metamorphic) malware. SAVE works on the simple premise that all versions of the same malware share a common core signature that is a combination of a sequence of API calls of the malware. A sequence of API calls is extracted from a malware sample. This sequence provides a basis for detecting variants and mutants of the same malware. To compare an unknown sequence with the extracted API sequence, the sequences are first aligned and then the Cosine measure, extended Jaccard measure, and the Pearson correlation measures are used collectively to determine whether the unknown sample is malicious or not. For our experiments, we take the mean value of the Cosine measure's threshold value, the extended Jackard measure's threshold values and the Pearson correlation measure's threshold value (as in the original paper). In order to use adaptive thresholding algorithm, similar to Bdist, the threshold output from one polymorphic sample is used to predict the threshold value for the next sample.

REFERENCES

Aggarwal, C. C., Han, J., Wang, J., and Yu, P. S. 2006. A framework for on-demand classification of evolving data streams. *IEEE Trans. Knowl. Data Eng. 18,* 5, 577–589.

Agosta, J. M., Wasser, C. D., Chandrashekar, J., and Livadas, C. 2007. An adaptive anomaly detector for worm detection. In *Proceedings of the 2nd USENIX Workshop on Tackling Computer Systems Problems with Machine Learning Techniques*. USENIX Association, Berkeley, CA, 3:1–3:6.

Ali, M. Q., Khan, H., Sajjad, A., and Khayam, S. A. 2009. On achieving good operating points on an ROC plane using stochastic anomaly score prediction. In *Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS'09)*. ACM, New York, 314–323.

Ali, S., Haq, I., Rizvi, S., Rasheed, N., Sarfraz, U., Khayam, S. A., and Mirza, F. 2010. On mitigating sampling-induced accuracy loss in traffic anomaly detection systems. *ACM SIGCOMM Comput. Commun. Rev. 40*, 4–16.

Arbor PeakFlow. Arbor networks' peakflow product. http://www.arbornetworks.com/peakflowsp.

Ashfaq, A. B., Joseph, M., Mumtaz, A., Ali, M. Q., Sajjad, A., and Khayam, S. A. 2008. A comparative evaluation of anomaly detectors under portscan attacks. In *Proceedings of the 11th International Symposium on Recent Advances in Intrusion Detection (RAID'08)*. Springer-Verlag, Berlin, 351–371.

Bifet, A., Holmes, G., Pfahringer, B., Kirkby, R., and Gavaldà, R. 2009. New ensemble methods for evolving data streams. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'09)*. ACM, 139–148.

Cardenas, A. A., Baras, J. S., and Seamon, K. 2006. A framework for the evaluation of intrusion detection systems. In *Proceedings of the IEEE Symposium on Security and Privacy (SP'06)*. IEEE.

Chen, S., Wang, H., Zhou, S., and Yu, P. S. 2008. Stop chasing trends: Discovering high order models in evolving data. In *Proceedings of the 2008 IEEE 24th International Conference on Data Engineering (ICDE'08)*. IEEE Computer Society, Los Alamitos, CA, 923–932.

Cisco Anomaly Guard. Cisco anomaly guard module homepage. www.cisco.com/en/US/products/ps6235/.

Cretu-Ciocarlie, G. F., Stavrou, A., Locasto, M. E., and Stolfo, S. J. 2009. Adaptive anomaly detection via self-calibration and dynamic updating. In *Proceedings of the 12th International Symposium on Recent Advances in Intrusion Detection (RAID'09)*. Springer-Verlag, Berlin, 41–60.

Crovella, M. E., and Bestavros, A. 1997. Self-similarity in world wide web traffic: Evidence and possible causes. *IEEE/ACM Trans. Netw. 5*, 835–846.

Forrest, S., Hofmeyr, S. A., Somayaji, A., and Longstaff, T. A. 1996. A sense of self for unix processes. In *Proceedings of the 1996 IEEE Symposium on Security and Privacy (SP'96)*. IEEE Computer Society, Los Alamitos, CA, 120–128.

FTP Brute Forcer. Ssh2ftpcrack ftp/ssh brute forcer. `http://packetstormsecurity.org/files/98155/SSH2FTPCrack-FTP-SSH-Brute-Forcer.html`.

Gao, D., Reiter, M. K., and Song, D. 2005. Behavioral distance for intrusion detection. In *Proceedings of the 8th International Symposium on Recent Advances in Intrusion Detection (RAID)*. 63–81.

Gao, J., Fan, W., and Han, J. 2007. On appropriate assumptions to mine data streams: Analysis and practice. In *Proceedings of the 7th IEEE International Conference on Data Mining (ICDM'07)*. IEEE Computer Society, Los Alamitos, CA, 143–152.

Gartner Report. 2003. Gartner information security hype cycle declares Intrusion detection systems a market failure money slated for intrusion detection should be invested in firewalls. http://www.gartner.com/about/press_releases/pr11june2003c.jsp.

Gu, G., Fogla, P., Dagon, D., Lee, W., and Skoric, B. 2006. Towards an information-theoretic framework for analyzing intrusion detection systems. In *Proceedings of the 11th European Symposium on Research in Computer Security (ESORICS'06)*.

Gu, Y., McCullum, A., and Towsley, D. 2005. Detecting anomalies in network traffic using maximum entropy estimation. In *Proceedings of the 5th ACM SIGCOMM Conference on Internet Measurement (IMC'05)*. USENIX Association, Berkeley, CA, 32–32.

Heap Profiler. HPROF: A heap/CPU profiling tool in j2se5.0. `http://docs.oracle.com/javase/7/docs/technotes/samples/hprof.html`.

Hollinger, G., Djugash, J., and Singh, S. 2008. Tracking a moving target in cluttered environments with ranging radios: Extended results. Tech. rep. CMU-RI-TR-08-07, Robotics Institute, Carnegie Mellon University.

Ide, T. and Kashima, H. 2004. Eigenspace-based anomaly detection in computer systems. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'04)*. ACM, New York, 440–449.

Jung, J., Paxson, V., Berger, A. W., and Balakrishnan, H. 2004. Fast portscan detection using sequential hypothesis testing. In *Proceedings of the IEEE Symposium on Security and Privacy (SP'04)*. IEEE Computer Society, Los Alamitos, CA.

Kang, D. K., Fuller, D., and Honavar, V. 2005. Learning classifiers for misuse and anomaly detection using a bag of system calls representation. In *Proceedings of 6th IEEE Systems Man and Cybernetics Information Assurance Workshop (IAW'05)*.

Kolter, J. Z., and Maloof, M. A. 2005. Using additive expert ensembles to cope with concept drift. In *Proceedings of the 22nd International Conference on Machine Learning (ICML'05)*. ACM, New York, 449–456.

Lakhina, A., Crovella, M., and Diot, C. 2004. Diagnosing network-wide traffic anomalies. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM'04)*. ACM, New York, 219–230.

Lakhina, A., Crovella, M., and Diot, C. 2005. Mining anomalies using traffic feature distributions. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM'05)*. ACM, New York, 217–228.

LBNL Dataset. LBNL/ICSI enterprise tracing project. http://www.icir.org/enterprise-tracing/Overview.html.

Lippmann, R. P., Haines, J. W., Fried, D. J., Korba, J., and Das, K. 2000. The 1999 DARPA offline intrusion detection evaluation. *Comput. Netw. 34*, 579–595.

Mahoney, M. V. and Chan, P. K. 2001. PHAD: Packet header anomaly detection for indentifying hostile network traffic. Tech. rep. CS-2001-4, Florida Tech.

Masud, M. M., Chen, Q., Khan, L., Aggarwal, C., Gao, J., Han, J., and Thuraisingham, B. 2010. Addressing concept-evolution in concept-drifting data streams. In *Proceedings of the IEEE International Conference on Data Mining (ICDM'10)* IEEE Computer Society, Los Alamitos, CA, 929–934.

Masud, M. M., Gao, J., Khan, L., Han, J., and Thuraisingham, B. M. 2011. Classification and novel class detection in concept-drifting data streams under time constraints. *IEEE Trans. Knowl. Data Eng. 23,* 6, 859–874.

Merhav, M., Gutman, M., and Ziv, J. 1989. On the estimation of the order of a markov chain and universal data compression. *IEEE Trans. Inf. Theory 35,* 5, 1014–1019.

Merhav, M., Gutman, M., and Ziv, J. 2010. Detecting intrusions through system call sequence and argument analysis. *IEEE Trans. Depend. Secure Comput. 7,* 4.

MIT Dataset. MIT lincoln laboratory, information systems technology. http://www.ll.mit.edu/mission/communications/ist/corpora/ideval/data/index.html.

Moore, D., Shannon, C., and Claffy, K. 2002. Code-red: A case study on the spread and victims of an internet worm. In *Code-Red: A Case Study on the Spread and Victims of an Internet Worm*, ACM, New York.

Netsparker tool. Netsparker, web application security scanner. `http://www.mavitunasecurity.com/netsparker/`.

Nexgin Dataset. Nexgin rc dataset. http://www.nexginrc.org/Datasets/Default.aspx.

NUST Dataset. NUST traffic datasets. http://wisnet.seecs.nust.edu.pk/projects/nes/datasets.html.

Pang, R., Allman, M., Bennett, M., Lee, J., Paxson, V., and Tierney, B. 2005. A first look at modern enterprise traffic. In *Proceedings of the 5th ACM SIGCOMM Conference on Internet Measurement (IMC'05)*. USENIX Association, Berkeley, CA, 2–2.

Ryu, Y. U. and Rhee, H. S. 2008. Evaluation of intrusion detection systems under a resource constraint. *ACM Trans. Inf. Syst. Secur. 11*, 20:1–20:24.

Shannon, C. and Moore, D. 2004. The spread of the witty worm. In *Proceedings of IEEE Security and Privacy (SP'04) 2*, 46–50.

Sqlninja tool. Sqlninja, a SQL server injection and takeover tool. http://sqlninja.sourceforge.net/.

Sung, A. H., Xu, J., Chavez, P., and Mukkamala, S. 2004. Static analyzer of vicious executables (SAVE). In *Proceedings of the 20th Annual Computer Security Applications Conference (ACSAC'04)*. IEEE Computer Society, Los Alamitos, CA, 326–334.

Symantec Security. Symantec security response. http://securityresponse.symantec.com/avcenter.

TADM toolkit. Tadm, toolkit for advanced discriminative modeling. `http://tadm.sourceforge.net`.

Tcpdump tool. Tcpdump/libpcap public repository. http://www.tcpdump.org/.

Trees, H. L. V. 2001. *Detection, Estimation and Modulation Theory: Part I* 1st Ed. Wiley-Interscience.

Twycross, J. and Williamson, M. M. 2003. Implementing and testing a virus throttle. In *Proceedings of the 12th Conference on USENIX Security Symposium*. USENIX Association, Berkeley, CA, 20–20.

UNM Dataset. Computer immune systems, datasets. http://www.cs.unm.edu/~immsec/data/synth-sm.html.

Wang, K. and Stolfo, S. J. 2004. Anomalous payload-based network intrusion detection. In *Proceedings of the 7th International Symposium on Recent Advances in Intrusion Detection (RAID)*. 203–222.

WisNet ADS. Wisnet ADS comparison homepage. http://wisnet.niit.edu.pk/projects/adeval.

Yu, Z., Tsai, J. J. P., and Weigert, T. 2007. An automatically tuning intrusion detection system. *IEEE Trans. Syst., Man, and Cybernet. 37*, 373–384.

Yu, Z., Tsai, J. J. P., and Weigert, T. 2008. An adaptive automatically tuning intrusion detection system. *ACM Trans. Autonom. Adaptive Syst. 3*, 10:1–10:25.

Received February 2011; revised September 2011, March 2012, October 2012; accepted January 2013