Predictive Network Anomaly Detection and Visualization

Mehmet Celenk, *Member, IEEE*, Thomas Conley, John Willis, *Student Member, IEEE*, and James Graham, *Student Member, IEEE*

***Abstract*—Various approaches have been developed for quantifying and displaying network traffic information for determining network status and in detecting anomalies. Although many of these methods are effective, they rely on the collection of long-term network statistics. Here, we present an approach that uses short-term observations of network features and their respective time averaged entropies. Acute changes are localized in network feature space using adaptive Wiener filtering and auto-regressive moving average modeling. The color-enhanced datagram is designed to allow a network engineer to quickly capture and visually comprehend at a glance the statistical characteristics of a network anomaly. First, average entropy for each feature is calculated for every second of observation. Then, the resultant short-term measurement is subjected to first- and second-order time averaging statistics. These measurements are the basis of a novel approach to anomaly estimation based on the well-known Fisher linear discriminant (FLD). Average port, high port, server ports, and peered ports are some of the network features used for stochastic clustering and filtering. We empirically determine that these network features obey Gaussian-like distributions. The proposed algorithm is tested on real-time network traffic data from Ohio University's main Internet connection. Experimentation has shown that the presented FLD-based scheme is accurate in identifying anomalies in network feature space, in localizing anomalies in network traffic flow, and in helping network engineers to prevent potential hazards. Furthermore, its performance is highly effective in providing a colorized visualization chart to network analysts in the presence of bursty network traffic.**

***Index Terms*—Auto-regressive moving average (ARMA) modeling, entropy, Fisher discriminant, network anomaly, Wiener filtering.**

I. Introduction

FIREWALLS and intrusion detection devices are the primary way of protecting today's modern enterprise networks from a host of network anomalies such as viruses, worms, scanners, and denial of service (DoS) from botnets. Their defense mechanism relies on detection of attacks after they have begun affecting the targeted network. Existing methods are able to identify specific packets which match a known pattern or originate from a specified location. However, these signature-based systems fail to detect unknown anomalies. An anomaly might be an old attack that has changed in some way to avoid detection, or it could be a completely new form of attack. To alleviate these shortcomings, significant research has been devoted to the task of identifying network anomalies using methods from statistical signal analysis and pattern recognition theory. Kwitt and Hoffman [1] and Shen *et al.* [2] dealt with anomaly detection using a robust principal component analysis (PCA) and metrics of aggregated network behavior. Additionally, the approach undertaken by Karasaridis *et al.* [3] deals primarily with the detection of botnets. The work described in [4] considers the detection of network intrusions in covariance space using pattern-recognition methods. Sang and Li [5] describe how far into the future one can predict network traffic by employing auto-regressive moving average (ARMA) as a model. Similarly, Cho *et al.* [6] present a method in which near-real-time network traffic data can be measured and filtered utilizing the Patricia tree and least recently used (LRU) replacement policy. In the work of Pang *et al.* [7], they examine known anomalies and possible ways for detecting them by filtering data to reduce load on the system. Feldman *et al.* [8] proposed a cascade-based approach that dealt with multifractal behavior in network traffic. The paper also describes a way of detecting network problems using their system. Yurcik and Li [9] and Plonka [10] demonstrate how NetFlows and FlowScans can be a more efficient way to monitor network traffic. Gong [11], [12] suggests methods in which the NetFlows can be used to detect worms and other types of intrusion into a network. Wavelet-based approaches have also been proposed for detecting anomalies and predicting ethernet traffic (see, for example, [13]).

Entropy is another well-known measure for quantifying the information of network traffic and has been extensively studied for anomaly detection and prevention [14], [15]. Significant research has also been devoted to the task of studying traffic structure and flows in conjunction with visual correlation of network alerts [16]. Most of the approaches are devised based on the long-term statistics of network traffic entropy [17]–[20]. One example is the work by Eimann, *et al.* [17] which discuses an entropy-based approach to detect network events. Harrington's work [19] is similar, but uses cross entropy and second-order distribution to detect changes in network behavior. Lall, *et al.* [18] employ the entropy of traffic distributions to aid in network monitoring, while Gu, *et al.* [14] utilize an entropy measure to detect anomalies in network traffic. More recently, Gianvecchio and Wang [20] introduced an entropy-based approach to detect the exploitation of covert timing channels in network traffic among the large amount of regular traffic. In Kim, *et al.* [21], the data in packet headers are examined using aggregate anal-

Manuscript received November 19, 2009; revised October 25, 2009; accepted December 15, 2009. Date of publication February 17, 2010; date of current version May 14, 2010. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Christian Cachin.

The authors are with School of Electrical Engineering and Computer Science, Stocker Center, Ohio University, Athens, OH 45701 USA (e-mail: celenk@ohio.edu; conleyt@ohio.edu; jw174304@ohio.edu; jg193404@ohio.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIFS.2010.2041808

ysis of correlation data and discrete wavelet transforms. Statistical data analysis of pattern recognition theory is also applied to the same problem with varying degrees of success [22]. A supervised statistical pattern recognition technique is proposed by Fu, *et al.* [23], which requires the complete statistics of network load and attack. Wagner and Plattner [24] have discussed a method based on changes in entropy content for IP addresses and ports but have not attempted to distinguish normal traffic from abnormal.

Other researchers have taken a variety of approaches. Thottan and Ji [25] apply signal processing techniques to the problem of network anomaly detection using statistical data analysis. The IP network anomaly detection is defined in a single class domain in conjunction with the types and sources of data available for analysis. They present a method based on sudden change detection on signals from multiple metrics, each of which has different statistical properties. In Hajji's work [26], the approach undertaken addresses the problem of change in characteristics of network traffic, and its relation with anomalies in local area networks. The emphasis is on fast detection for reducing potential impact of problems on network services' users by finite Gaussian mixture traffic model and a baseline of network normal operation as the asymptotic distribution of the difference between successive estimates of multivariate Gaussian model parameters with mean zero under normal operations, and sudden jumps in this mean in abnormal conditions. In [27], the researchers introduced a supervised anomaly detection method by concatenating the k-Means clustering and the ID3 decision tree learning. In their work, k-Means clustering is carried out first on training instances to determine k number of distinct clusters, representing regions of similar instances. An ID3 decision tree is then trained with the instances in each k-Means cluster so that the anomaly detection can be performed via a score matrix. Authors of [28] make use of the Tsallis (or nonextensive) entropy to deal with network traffic anomalies. They have demonstrated the effectiveness of this measure over the traditional Shannon entropy-based techniques by detecting more network anomalies and reducing false negatives. In turn, a flexibility improvement in the detection process is reported due to the finely tuned sensitivity of the anomaly detection system as opposed to the conventional entropy measure. Kim and Reddy [29] consider the time series analysis of different packet header data and propose simple and efficient mechanisms for collecting and analyzing aggregated data in real-time. They demonstrate that their proposed signal series have higher efficacy in detecting attacks than the analysis of traffic volume itself. Recently, Androulidakis, *et al.* [30] have proposed a method of anomaly detection and classification via opportunistic sampling which also makes use of port entropy.

None of the studies described above provides a method of predicting an attack before it occurs. In this work, we aim to predict network anomalies. We define an anomaly as any detected network behavior that is of interest to a network or security engineer such as worm outbreaks, botnet command and control traffic, misconfigured network devices, or DoS attacks. To this end, we statistically analyze network flow data and apply Weiner filtering to pass normal traffic. This, in turn, helps us identify the signal corresponding to network anomalies in the selected feature measurement, which characterizes the network flow in that dimension. By estimating the auto-correlation function of normal traffic, the ARMA predictor [31] is devised using the well-known Yule–Walker regression [32].

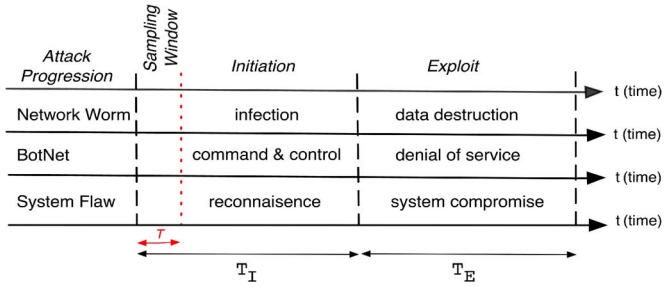


Fig. 1. Modeling of cyber attack progression in a network.

In the following sections, we describe the approach and the results achieved in our attempt to predict network traffic anomalies. Section II explores the structure of the proposed algorithm. The experimental test-bed and computer results are described in detail in Section III. Remaining discussion is devoted to conclusions, future work, and potential applications.

II. Overall Approach

In this work, we characterize a network by considering its N features (e.g., average port, traffic load, etc.), each of which is a continuous vector of dimension K; i.e., $\vec{X} = \{\vec{X}_i(t), i = 1, 2, \ldots, N\}$ while a cyber attack is formulated by a continuous random function $\eta(t)$ whose behavior on the network can be modeled as in Fig. 1. More rigorously, the attack function $\eta(t)$ may be written as a linear combination of two stochastic components; namely, $T_I(t)$ initiation and $T_E(t)$ exploit. Thus,

$$\eta(t) = T_I(t) + T_E(t). \tag{1}$$

Here, $T_I(t)$ is the initiation phase which proceeds the actual network attack and is identified by its anomalous traffic. For example, in Fig. 1 the initiation phase of a network worm is characterized by infectious traffic, a BotNet is initiated by command and control signals, and a network flaw (e.g., unpatched operating systems, open ports, infected servers, etc.) is first exploited using reconnaissance traffic. Exploit phases of these sample anomalies are also depicted in Fig. 1 and include data destruction by network worms, DoS by BotNets, or a system compromise. Notice that the anomaly progression time periods $T_I(t))$ and $(T_E(t)$ are not identical in length or behavior for the various attack types.

In this particular research, we have determined the initiation time $T_I(t)$ empirically by experimentally changing the window size starting from 3 s and gradually increasing by 1 s until an anomaly presents itself visually. Note that the length of time for the exploit $(T_E(t))$ depends on such varying factors as network configuration, computer defenses, and individual exploit characteristics. Similar studies have also been done which report measured values of the aforementioned timing parameters for different attack types and systems. For example, in [33] the author has elaborated on how to determine the timeframes of various phases of infection and exploitation using a probabilistic model for estimation of the times. Schneider [34] studied

the timing of propagation for various types of network worms and determined theoretical initiation times between 1.42 and 1.85 s for various kinds of worms. In [35], it is reported that the slammer worm was able to complete infection and replicate in 8.5 s. Note that the infection of a single machine may occur without generating any network traffic. Such events, considered instantaneous from a network perspective, may be part of the initial phase of a widespread network anomaly which is detected as the compromised machine begins to use network resources in an abnormal way.

Our research deals with varying time scales and with the periodic nature of events by changing the sampling rate and window size in the underlying statistical model. Other researchers have taken a similar timing approach to the analysis of the worm outbreaks. In particular, Rajab, *et al.* [36] attempts to determine the probability of detection based on the underlying statistics such as sampling rate and window size.

In our case, shorter initialization phases as well as periodic patterns in the network traffic are handled by a smaller window size, which is made feasible by increasing the sampling rate. A faster sampling rate allows the smaller window size without degrading the statistics, since the sample size of a window will remain high. This sampling scheme works for any single event as long as the window size is smaller than the initiation phase. In addition, any periodic event is detected if the window size is smaller than the period of the event in progress.

For example, a daily process will not be considered anomalous in statistics calculated for a window size of one week, since the multiple daily occurrences will be normal for the weekly statistics. However, the initiation of such an event will be detected as an anomaly in a window size of one hour because that window's statistics will vary from normal for that window size. Thus, observations from a network under attack will comprise not only normal network traffic $\vec{X}_i(t)$ but also anomalous traffic features $\eta(t)$, which is represented as $\vec{X}_i(t)+\eta(t)$. This modeling also explains the rationale behind predicting anomalies and is driven by the premise that if we sample the network traffic fast enough, the initiation phase would be sufficient to predict the onset of the actual attack (i.e., exploit). Referring to the model in Fig. 1, we can write

$$\begin{aligned}\vec{X}_i(t)+\eta(t)|_{t=nT} &= \vec{X}_i(nT)+\eta(nT)|_{\text{Dropping } T}\\ &= \vec{X}_i(n)+(\eta_I(n)+\eta_E(n))\\ &= \underbrace{\vec{X}_{i_I}(n)+\eta_I(n)}_{\text{valid in time interval } T_I}\\ &\quad + \underbrace{\vec{X}_{i_E}(n)+\eta_E(n)}_{\text{valid in time interval } T_E}. \end{aligned} \tag{2}$$

Here, n denotes the discrete time variable index and T is an infinitesimal sampling interval. Since T appears in all the terms, it is customarily ignored in discrete time signal representation. As a result, the discrete network feature vector can be expressed as

$$\begin{aligned}\vec{X}_i(n) = [&x_1(n)+(\eta_I(n)+\eta_E(n)),\\ &x_2(n)+(\eta_I(n)+\eta_E(n)),\ldots,\\ &x_K(n)+(\eta_I(n)+\eta_E(n))]'\end{aligned} \tag{3}$$

where the symbol $(')$ denotes matrix transposition. Fig. 2 shows a sample screenshot of the real-tme appearance of $\vec{X}_i(n)$ as displayed on a monitoring console currently used by network and security engineers. This snapshot of raw network traffic data is sampled under load from Ohio University network and shows source and destination IP, port, netflow statistics, and calculated suspicious activity ("EVIL") score. Since this display contains a significant amount of information, it is difficult for a network engineer to capture the overall picture of activity for even a short time period. In order to represent such large amounts of information in a compact form without loosing generality, we map its content into the pattern vector as defined in (3). As an example, a particular entry from Fig. 2 is shown below as the ith feature representing destination port

$$\begin{aligned}&(80:1741)(8000:335)(16427:218)(23076:183)\\ &\quad(6881:175)\ldots(6969:168)(24133:153)(9737:150)\\ &\quad(588:148)(\ldots).\end{aligned} \tag{4}$$

In this case, the corresponding ith feature pattern vector $\vec{X}_i(n) = [P1(n), P2(n), \ldots, P65535(n)]'$ becomes $\vec{X}_i(n) = [P80(n), P8000(n), P16427(n)\ldots]'$. Notice that since the total number of ports is 65 535, the maximum dimension (K) for the feature vector $\vec{X}_i(n)$ is 65 535, respectively. Here, each port utilization is a random variable varying by time. The total number (N) of distinct feature classes depends on the network attributes being analyzed. In general, if we consider all the possible network feature vectors and the discreet time variable, the stochastic process describing network behavior forms a hyper-space of dimension N^K varying by time n. Hence, our task is to find a suboptimal space that gives rise to an anomaly and to come up with an intelligible visual representation of this complex stochastic feature space and associated data flow. The basic information measure selected as the discriminatory factor for this purpose is the entropy (E) of a stochastic process. In general, for a continuous stochastic process $X(t)$ the entropy $E(t)$ is defined by [37]

$$E(t) = \int_x P_x(x;t)\cdot\log_2\frac{1}{P_x(x;t)}dx. \tag{5}$$

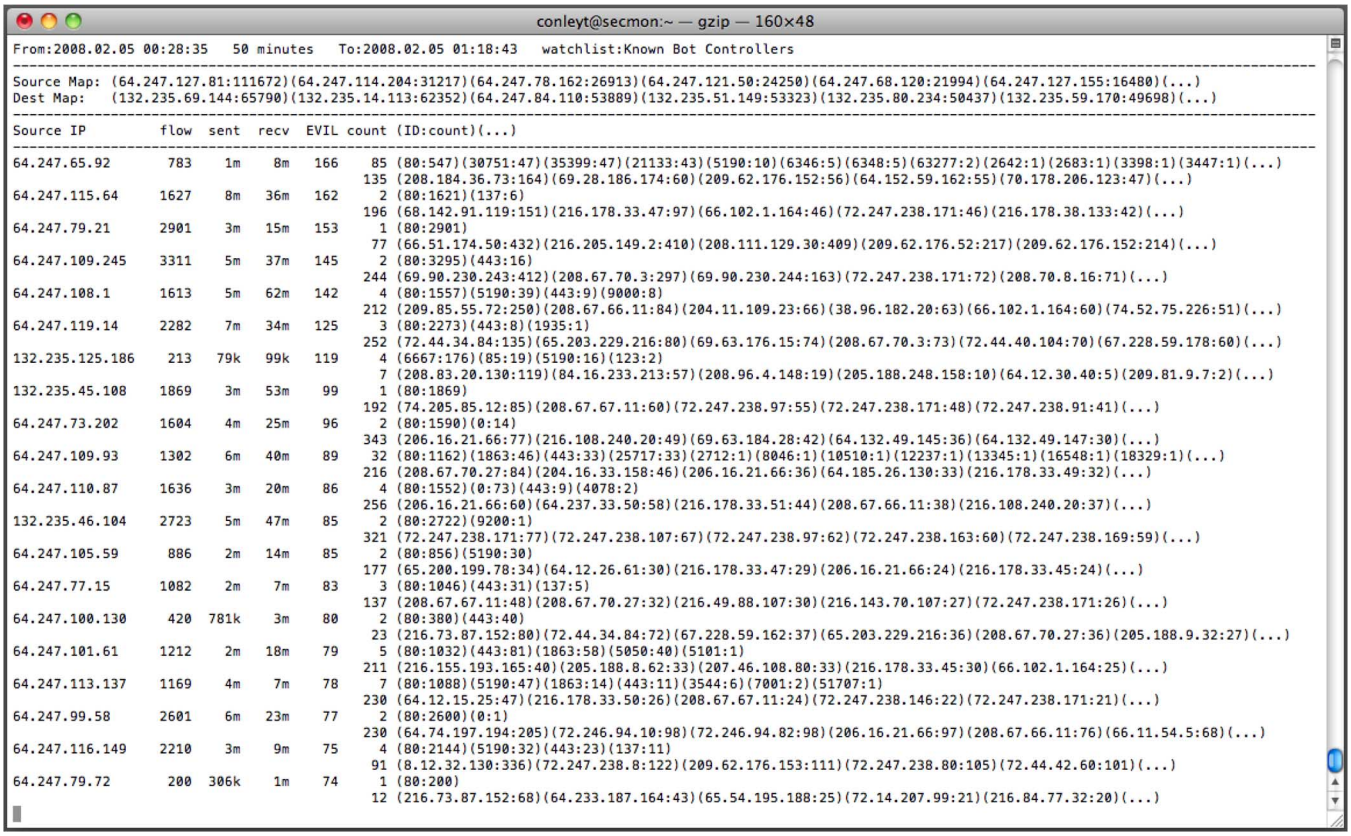
For a discrete-space and discrete-time stochastic process $X(n)$, entropy $E(n)$ is computed as

$$E(n) = \sum_x P_x(x,n)\cdot\log_2\frac{1}{P_x(x;n)}. \tag{6}$$

For every second, the number of observations of a particular feature is counted and linearly quantized. For discrete features, such as port id, no quantization is needed. From these data, we generate a feature space to be used in average entropy calculations over time for each $\vec{X}_i(n)$, as

$$\left\{\bar{E}(m,n), \bar{\mu}_E(m,n), \bar{\sigma}^2_E(m,n)\right\};\quad n=1,2,\ldots. \tag{7}$$

where m is the time averaging window size, $\bar{E}(m,n)$ is the time averaged entropy, $\bar{\mu}_E(m,n)$ is the time average of the mean en-



conleyt@secmon:~ — gzip — 160×48

```
From:2008.02.05 00:28:35   50 minutes   To:2008.02.05 01:18:43   watchlist:Known Bot Controllers
-------------------------------------------------------------------------------------------------------------------------------------------------------
Source Map: (64.247.127.81:111672)(64.247.114.204:31217)(64.247.78.162:26913)(64.247.121.50:24250)(64.247.68.120:21994)(64.247.127.155:16480)(...)
Dest Map:   (132.235.69.144:65790)(132.235.14.113:62352)(64.247.84.110:53889)(132.235.51.149:53323)(132.235.80.234:50437)(132.235.59.170:49698)(...)
-------------------------------------------------------------------------------------------------------------------------------------------------------
Source IP         flow  sent  recv  EVIL count (ID:count)(...)
-------------------------------------------------------------------------------------------------------------------------------------------------------
64.247.65.92       783    1m    8m   166    85 (80:547)(30751:47)(35399:47)(21133:43)(5190:10)(6346:5)(6348:5)(63277:2)(2642:1)(2683:1)(3398:1)(3447:1)(...)
                                          135 (208.184.36.73:164)(69.28.186.174:60)(209.62.176.152:56)(64.152.59.162:55)(70.178.206.123:47)(...)
64.247.115.64     1627    8m   36m   162     2 (80:1621)(137:6)
                                          196 (68.142.91.119:151)(216.178.33.47:97)(66.102.1.164:46)(72.247.238.171:46)(216.178.38.133:42)(...)
64.247.79.21      2901    3m   15m   153     1 (80:2901)
                                           77 (66.51.174.50:432)(216.205.149.2:410)(208.111.129.30:409)(209.62.176.52:217)(209.62.176.152:214)(...)
64.247.109.245    3311    5m   37m   145     2 (80:3295)(443:16)
                                          244 (69.90.230.243:412)(208.67.70.3:297)(69.90.230.244:163)(72.247.238.171:72)(208.70.8.16:71)(...)
64.247.108.1      1613    5m   62m   142     4 (80:1557)(5190:39)(443:9)(9000:8)
                                          212 (209.85.55.72:250)(208.67.66.11:84)(204.11.109.23:66)(38.96.182.20:63)(66.102.1.164:60)(74.52.75.226:51)(...)
64.247.119.14     2282    7m   34m   125     3 (80:2273)(443:8)(1935:1)
                                          252 (72.44.34.84:135)(65.203.229.216:80)(69.63.176.15:74)(208.67.70.3:73)(72.44.40.104:70)(67.228.59.178:60)(...)
132.235.125.186    213   79k   99k   119     4 (6667:176)(85:19)(5190:16)(123:2)
                                            7 (208.83.20.130:119)(84.16.233.213:57)(208.96.4.148:19)(205.188.248.158:10)(64.12.30.40:5)(209.81.9.7:2)(...)
132.235.45.108    1869    3m   53m    99     1 (80:1869)
                                          192 (74.205.85.12:85)(208.67.67.11:60)(72.247.238.97:55)(72.247.238.171:48)(72.247.238.91:41)(...)
64.247.73.202     1604    4m   25m    96     2 (80:1590)(0:14)
                                          343 (206.16.21.66:77)(216.108.240.20:49)(69.63.184.28:42)(64.132.49.145:36)(64.132.49.147:30)(...)
64.247.109.93     1302    6m   40m    89    32 (80:1162)(1863:46)(443:33)(25717:33)(2712:1)(8046:1)(10510:1)(12237:1)(13345:1)(16548:1)(18329:1)(...)
                                          216 (208.67.70.27:84)(204.16.33.158:46)(206.16.21.66:36)(64.185.26.130:33)(216.178.33.49:32)(...)
64.247.110.87     1636    3m   20m    86     4 (80:1552)(0:73)(443:9)(4078:2)
                                          256 (206.16.21.66:60)(64.237.33.50:58)(216.178.33.51:44)(208.67.66.11:38)(216.108.240.20:37)(...)
132.235.46.104    2723    5m   47m    85     2 (80:2722)(9200:1)
                                          321 (72.247.238.171:77)(72.247.238.107:67)(72.247.238.97:62)(72.247.238.163:60)(72.247.238.169:59)(...)
64.247.105.59      886    2m   14m    85     2 (80:856)(5190:30)
                                          177 (65.200.199.78:34)(64.12.26.61:30)(216.178.33.47:29)(206.16.21.66:24)(216.178.33.45:24)(...)
64.247.77.15      1082    2m    7m    83     3 (80:1046)(443:31)(137:5)
                                          137 (208.67.67.11:48)(208.67.70.27:32)(216.49.88.107:30)(216.143.70.107:27)(72.247.238.171:26)(...)
64.247.100.130     420  781k    3m    80     2 (80:380)(443:40)
                                           23 (216.73.87.152:80)(72.44.34.84:72)(67.228.59.162:37)(65.203.229.216:36)(208.67.70.27:36)(205.188.9.32:27)(...)
64.247.101.61     1212    2m   18m    79     5 (80:1032)(443:81)(1863:58)(5050:40)(5101:1)
                                          211 (216.155.193.165:40)(205.188.8.62:33)(207.46.108.80:33)(216.178.33.45:30)(66.102.1.164:25)(...)
64.247.113.137    1169    4m    7m    78     7 (80:1088)(5190:47)(1863:14)(443:11)(3544:6)(7001:2)(51707:1)
                                          230 (64.12.15.25:47)(216.178.33.50:26)(208.67.67.11:24)(72.247.238.146:22)(72.247.238.171:21)(...)
64.247.99.58      2601    6m   23m    77     2 (80:2600)(0:1)
                                          230 (64.74.197.194:205)(72.246.94.10:98)(72.246.94.82:98)(206.16.21.66:97)(208.67.66.11:76)(66.11.54.5:68)(...)
64.247.116.149    2210    3m    9m    75     4 (80:2144)(5190:32)(443:23)(137:11)
                                           91 (8.12.32.130:336)(72.247.238.8:122)(209.62.176.153:111)(72.247.238.80:105)(72.44.42.60:101)(...)
64.247.79.72       200  306k    1m    74     1 (80:200)
                                           12 (216.73.87.152:68)(64.233.187.164:43)(65.54.195.188:25)(72.14.207.99:21)(216.84.77.32:20)(...)
```

Fig. 2. Snapshot of textual display of raw traffic data from Ohio University network showing source and destination IP, port, netflow statistics, and calculated suspicious activity (called "EVIL") score.

tropy, and $\bar{\sigma}_E^2(m,n)$ is the variance average. They are calculated as

$$\bar{E}(m,n) = \frac{1}{m}\sum_{k=1}^{m} E(k,n) \tag{8}$$

$$\bar{\mu}_E(m,n) = \frac{1}{m}\sum_{k=1}^{m} \mu(k,n) \tag{9}$$

$$\bar{\sigma}_E^2(m,n) = \frac{1}{m}\sum_{k=1}^{m} \sigma^2(k,n) \tag{10}$$

where $E(k,n)$, $\mu(k,n)$, and $\sigma^2(k,n)$ are the entropy, mean, and variance of the feature x in time duration from k to n (see Fig. 3). The modified Fisher linear discriminant (FLD) function [31] is selected as a scatter measure for each network feature pattern class. For the purpose of discriminating anomaly patterns from regular traffic, we define the modified FLD performance index $J(m,n)$ as a function of the window size m and the discrete-time variable n. Hence, we have [38]

$$J(m,n) = \frac{\bar{\mu}_E(m,n)}{\bar{\sigma}_E^2(m,n)}. \tag{11}$$

Notice that the proper selection of window size m enables the prediction scheme to identify slow initiation attacks as well as rapidly progressing exploits. Using a larger m will be more effective in detecting changes in measured features that occur slowly as in attacks that are commonly referred to as "slow and low," whereas a smaller m helps a network security analyst in finding explosive attacks such as DoS.

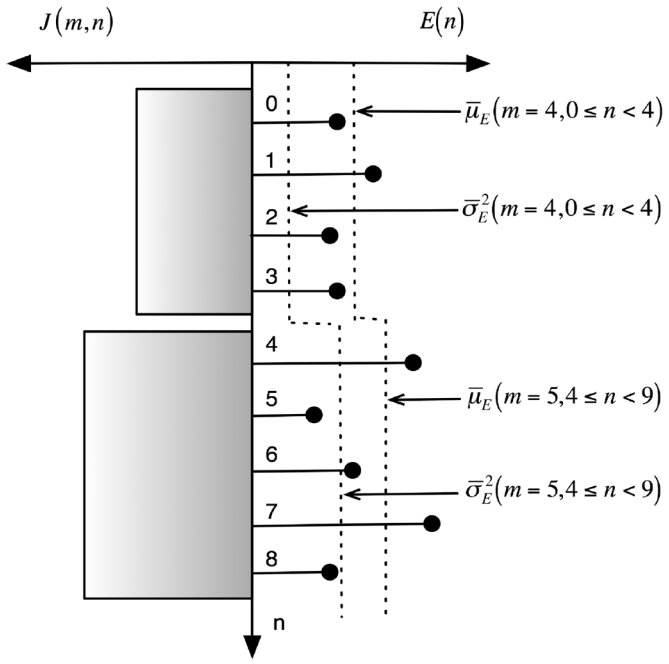


Fig. 3. Network entropy datagram in which the entropy $E(n)$ is denoted by impulses (right side) while the FLD measure $J(m,n)$ is represented by rectangular shapes (left side) and the mean and variance of entropy are indicated with dotted lines.

Fig. 3 is indeed a new graphical representation of $J(m,n)$ and the associated network attributes. This visualization technique is similar to the datagram concept of network communication as described in [39] and employed here to be an effective means

of one-dimensional traffic tracking and alerting. Section IV will explore the aforementioned visualization concept to illustrate the complex text-based information graph currently used by network engineers as shown in Fig. 2. The above performance index J assumes a large value when the entropy has high mean and low variance in varying-size time-window m due to unusual activity in the network. In experimentation, window size m is dynamically adjusted in such a way that the range in which the maximum value of J is reached. Hence, the maximum value of J indicates the occurrence of a possible anomaly. This corresponds to determining an m-dimensional polyhedra of varying sizes that show which network attributes are more likely currently involved in a cyber-attack in the given $N^K(m,n)$-dimensional time-varying network feature space. For simplicity, we first consider a single feature which describes the network behavior. In this case, solution to the anomaly detection problem becomes finding the solution to the second-order derivative of $J(m,n)$ for which the fourth-order derivative is negative, thus, indicating a maximum value of J. Mathematically, we have

$$\frac{\partial^2 J(m,n)}{\partial m \cdot \partial n} = 0. \tag{12}$$

The roots, $m = m_{\text{opt}}$ and $n = n_{\text{opt}}$, which make the fourth-order derivative of J with respect to m and n negative, are the desired time and duration parameters for detecting anomalies. Explicitly, this is defined by

$$\left.\frac{\partial^4 J(m,n)}{\partial^2 m \cdot \partial^2 n}\right|_{m=m_{\text{opt}},n=n_{\text{opt}}} < 0. \tag{13}$$

Solutions of (12) and (13) must be sought continuously since the performance index J is a function of the time variable n and window size m. To alleviate the high computational cost associated with the process of localizing n and m, we present an optimization procedure by rewriting (12) as

$$\frac{\partial}{\partial m}\left(\frac{\partial J(m,n)}{\partial n}\right) = 0. \tag{14}$$

We then search for a value of n that maximizes J; i.e.,

$$\frac{\partial J(m,n)}{\partial n} = 0 \tag{15}$$

which is subject to

$$\frac{\partial^2 J(m,n)}{\partial^2 n} < 0. \tag{16}$$

In the computer implementation, the performance index function J is first smoothed along the discrete time axis n by using an adaptive Gaussian filter. Then, (15) and (16) are approximated by the first- and second-order derivative templates of $(-1,+1)$ and $(+1,-2,+1)$, respectively. For optimal window size determination, which follows the function behavior closely, we present a prediction method based on Wiener filtering and ARMA modeling in Section III.

III. Optimal Window Size Determination

A. Adaptive Digital Anomaly Predictor (ADAP)

First, we consider a single component $x_i(t)$ of the K-dimensional traffic feature set as defined above. This measured input is

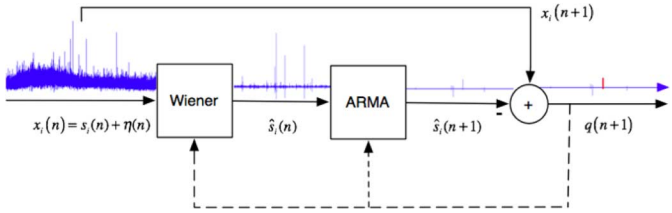


Fig. 4. Block diagram of ADAP.

modeled similar to (2) as a linear combination of normal traffic signal $s_i(t)$ and anomalous traffic noise $\eta(t)$ expressed mathematically by

$$x_i(t) = s_i(t) + \eta(t). \tag{17}$$

As was done earlier with continuous processes in Section II, we discretize (17) by letting $t = nT$ where T is the sampling interval which can be made as small as possible. Hence,

$$x_i(nT) = s_i(nT) + \eta(nT). \tag{18}$$

Since T is common for all the terms, it can be ignored without loosing generality. Thus, we have

$$x_i(n) = s_i(n) + \eta(n). \tag{19}$$

The approach undertaken herein is to extract the normal network flow $(s_i(n))$ and use it to predict anomalies as shown in Fig. 4. The Wiener filter removes the noise from the signal and outputs the estimate of the normal traffic flow. To achieve a balanced estimate, we adjust the window size and associated coefficients in the Wiener filter and the ARMA predictor thereby tracking signal variation more closely. The feedback control channel shown by a dashed line in Fig. 4 lets the algorithm adapt to a changing network signal waveform. This is made in accordance with the adaptable Wiener filter implementation method proposed in [40] as

$$\hat{s}_{i(n)} = \mu_{\hat{s}_i}(n) + \frac{\sigma^2_{\hat{s}_i}(n)}{\sigma^2_{\hat{s}_i}(n) + \sigma^2_{\eta}(n)} \cdot (x_i(n) - \mu_{\hat{s}_i}(n)) \tag{20}$$

where $\hat{s}_{i(n)}$ is the Wiener filter output in discrete time domain, $\mu_{\hat{s}_i}(n)$ is the mean value of the normal flow, and $\sigma^2_{\hat{s}_i}$, $\sigma^2_{\eta_i}$ are the respective variances of the measured traffic and anomaly computed in a window of size m. These signal signatures are calculated as

$$\mu_{\hat{s}_i}(n) = \frac{1}{(2m+1)} \sum_{k=n-m}^{n+m} x_i(k) \tag{21}$$

$$\hat{\sigma}^2_{\hat{s}_i}(n) = \begin{cases} \hat{\sigma}^2_{x_i}(n) - \hat{\sigma}^2_{\eta}(n), & if\ \left(\hat{\sigma}^2_{x_i}(n) - \hat{\sigma}^2_{\eta}(n)\right) > 0 \\ 0, & \text{otherwise} \end{cases} \tag{22}$$

$$\sigma^2_{x_i}(n) = \frac{1}{(2m+1)} \sum_{k=n-m}^{n+m} \left((x_i(k) - \mu_{\hat{s}_i}(k)\right)^2. \tag{23}$$

The estimated signal $\hat{s}_i(n)$ is then applied to the ARMA unit, which predicts the next value $\hat{s}_i(n+1)$ of $\hat{s}_i(n)$. This is similar to what has been done in [5] with the exception that they predict network traffic but ignore any anomaly that may exist at the time of measurement. Their prediction is based on the

assumptions of stationary Gaussian white noise with unit variance and the Gaussian nature of the network traffic without any empirical justification. On the other hand, in this research we have no restriction on network flow, nor do we have restriction on noise. Noise is considered to be the combination of network anomalies and white noise as described in [5]. ARMA starts the process of estimation by calculating the auto-correlation function for $\hat{s}_i(n)$ as in (27). The auto-correlation function is then used in the third-order predictor as in the following:

$$\begin{bmatrix} R_{\hat{s}_i}(n+1) \\ R_{\hat{s}_i}(n+2) \\ R_{\hat{s}_i}(n+3) \end{bmatrix} = \begin{bmatrix} R_{\hat{s}_i}(n+0) & R_{\hat{s}_i}(n+1) & R_{\hat{s}_i}(n+2) \\ R_{\hat{s}_i}(n-1) & R_{\hat{s}_i}(n+0) & R_{\hat{s}_i}(n+1) \\ R_{\hat{s}_i}(n-2) & R_{\hat{s}_i}(n-1) & R_{\hat{s}_i}(n+0) \end{bmatrix} \cdot \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} \tag{24}$$

$$\hat{s}_i(n+1) = \alpha_1 \cdot \hat{s}_i(n) + \alpha_2 \cdot \hat{s}_i(n-1) + \alpha_3 \cdot \hat{s}_i(n-2). \tag{25}$$

Here, α_i represents the predictor coefficient and $R_{\hat{s}_i}(i)$ denotes the auto-correlation function value at i. Let $P_\eta(\omega)$ be the power spectrum (i.e., the Fourier transform) of the auto-correlation function of the network anomalous signal $\eta(n)$ and $P_{\hat{s}_i}(\omega)$ be the power spectrum (i.e., the Fourier transform) of the auto-correlation of the normal network flow $\hat{s}_i(n)$. The measurement of these stochastic signal signatures is carried out using

$$R_\eta(n) = \frac{1}{m} \sum_{k=-\infty}^{\infty} \eta(k) \cdot \eta^*(k-n) \tag{26}$$

$$R_{\hat{s}_i}(n) = \frac{1}{m} \sum_{k=-\infty}^{\infty} \hat{s}_i(k) \cdot \hat{s}_i^*(k-n) \tag{27}$$

$$P_\eta(\omega) = \sum_{n=-\infty}^{\infty} R_\eta(n) \cdot e^{-j\omega n} \tag{28}$$

$$P_{\hat{s}_i}(\omega) = \sum_{n=-\infty}^{\infty} R_{\hat{s}_i}(n) \cdot e^{-j\omega n} \tag{29}$$

where (26) and (27) denote the estimated auto-correlation functions of the noise signal and the normal traffic signal in a window of size m while (28) and (29) represent the power spectrum of the noise signal and normal traffic signal, respectively. Since direct measurement of the power spectrum is complex and costly, it is desirable to predict it using the periodogram approach described in [40] and summarized by

$$P_\eta(\omega) = \frac{1}{m} |N(\omega)|^2 \tag{30}$$

$$P_{\hat{s}_i}(\omega) = \frac{1}{m} \left| \hat{S}_i(\omega) \right|^2 \tag{31}$$

where (30) and (31) correspond to the estimated power spectrums using periodograms of the noise $\eta(n)$ and the normal traffic signal $\hat{s}_i(n)$, and m is the number of measurements. Notice that the auto-correlation function is even; i.e., $R_{\hat{s}_i}(+i) = R_{\hat{s}_i}(-i)$, and has its maximum at the origin, that is, $R_{\hat{s}_i}(0) \geq R_{\hat{s}_i}(n)$ for all values of n. Using (24) and (25), the value of $\hat{s}_i(n+1)$ is predicted by the ARMA module and compared to measured signal, producing the output $q(n+1)$, of the ADAP function. Hence, we have

$$q(n+1) = x_i(n+1) - \hat{s}_i(n+1) \tag{32}$$

$$q(n+1) = [s_i(n+1) - \eta_i(n+1)] - \hat{s}_i(n+1) \tag{33}$$

$$q(n+1) = [s_i(n+1) - \hat{s}_i(n+1)] + \eta(n+1) \tag{34}$$

$$q(n+1) = \epsilon - \eta(n+1) \tag{35}$$

where ϵ represents the prediction or estimation error associated with the ARMA unit. Equations (32) through (35) enable us to come up with a decision predicate expressed as

1) if $\hat{s}_i(n+1) = x_i(n+1)$, then $q(n+1) = 0$; hence, no anomaly;
2) if $\hat{s}_i(n+1) = s_i(n+1)$, then $q(n+1) = \eta(n+1)$; hence, anomaly predicted;
3) if $\hat{s}_i(n+1) \neq s_i(n+1)$, then $q(n+1) = \eta(n+1)$; hence, anomaly plus error predicted;

for identifying any anomaly. This structured logic extends the work of [5] by directly predicting near-real-time network attacks.

B. ADAP Using Multiple Features

In this section, we extend the single ADAP functionality, to incorporate information from multiple features into the detection process. It is important to consider a diverse feature set in order to cover a wide range of complex anomaly signatures. Malicious activities are specifically designed to spread out their effect over multiple features in order to avoid detection by narrowly focused detectors. A cyber-attack, for instance, may try to reduce its visibility by using a common, well-known port and cause only a slight anomalous increase and no alarm. However, even the slightest increase in a single feature, when combined with anomalous activity in other features, can have a synergistic effect which makes it possible to predict that anomaly. By using a multiple feature space, the extended method being described is able to tap the additive power of the rich network traffic feature set. As a way of dealing with complex anomalies without loosing generality, we use the feature set $\vec{X} = [x_1, x_2, x_3, x_4]'$, where x_1 is "average port," x_2 is "high ports," x_3 is "server ports," x_4 is "peered ports," as a representative experiment. Since these individual characteristics may be correlated, this feature space can lead to a wrong conclusion, unless the underlying features are first uncorrelated and the resultant feature space has a Euclidean metric. This is achieved using the well-known Karhunen–Love (PCA or discrete Hotelling) transformation [31]. Let $C_{\vec{X}\vec{X}}$ be the cross-correlation matrix for $\vec{X}$ given by

$$C_{\vec{X}\vec{X}} = \{\vec{X} \cdot \vec{X}'\} = \begin{bmatrix} C_{x_1x_1} & C_{x_1x_2} & C_{x_1x_3} & C_{x_1x_4} \\ C_{x_2x_1} & C_{x_2x_2} & C_{x_2x_3} & C_{x_2x_4} \\ C_{x_3x_1} & C_{x_3x_2} & C_{x_3x_3} & C_{x_3x_4} \\ C_{x_4x_1} & C_{x_4x_2} & C_{x_4x_3} & C_{x_4x_4} \end{bmatrix} \tag{36}$$

where the diagonal elements are the auto-correlation functions of the features and off-diagonals are cross-correlations of respective feature pairs. Notice that $C_{\vec{X}\vec{X}}$ is a symmetric square matrix leading to a fourth-order characteristic equation and results in four corresponding eigenvectors. By normalizing these eigenvectors, we generate a four-dimensional space in which

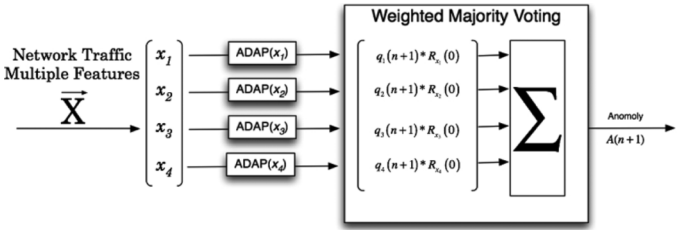


Fig. 5. Detecting network anomalies with ADAP and weighted majority voting.

the selected features are uncorrelated. Hence, the ADAP function can filter features individually by adaptively computing the mean and variances as the predictor output starts to deviate significantly from actual measured values. Here, we refer to the uncorrelated x_i's by y_i's, and the respective probability density function (p_y) of the uncorrelated version of $\vec{X}$, represented by $\vec{Y}$, via

$$p_{\vec{Y}} = p_{y_1} \cdot p_{y_2} \cdot p_{y_3} \cdot p_{y_4}. \tag{37}$$

By empirical data analysis, we have verified that each feature in $\vec{X}$ is normally distributed. Hence, we write

$$p_{\vec{Y}} = \prod_{i=1}^{4} \frac{1}{\sqrt{2\pi}\sigma_i} \cdot e^{-\frac{(y_i-\mu_i)^2}{2\sigma_i^2}} \tag{38}$$

where mean μ_i and σ_i^2 are the mean and variance of y_i. For the ease of implementation, we only consider auto-correlation for the diagonal elements of the correlation matrix of (36). As a result, the anomaly prediction decision is made in the $[R_{x_1x_1}, R_{x_2x_2}, R_{x_3x_3}, R_{x_4x_4}]$ space using a weighted majority-voting scheme as shown in Fig. 5. The ADAP function processes each single feature in parallel and provides a prediction output. We present the overall result $A(n+1)$ as a linear combination of the individual channels with weights corresponding to the maximum value of each feature's auto-correlation function. Thus,

$$\begin{aligned} A(n+1) = {} & q_1(n+1) \cdot R_{x_1}(0) + q_1(n+2) \cdot R_{x_2}(0) \\ & + q_1(n+3) \cdot R_{x_3}(0) + q_4(n+4) \cdot R_{x_4}(0). \end{aligned} \tag{39}$$

If it exceeds a predetermined empirical anomaly threshold, then the activity represented by the feature set $\vec{X}$ is deemed an anomaly. The system can then be directed to respond at the time instant of $n+1$. The anomaly detector's effectiveness is driven by a set of parameters chosen in accordance with their significance. By fine-tuning the set of features, the Wiener filter and auto-correlation window sizes and the majority voting thresholds, we find an optimal feature space in which characteristics are uncorrelated and yet still contain all the information associated with network traffic and attacks.

C. Normal Density Approximation for Network Traffic

There has been considerable research in the statistical analysis of network traffic data. However, previous work has assumed the data to be normally distributed without supporting this assumption [4]. In our observations, the Gaussian nature of the traffic is determined by the bell-shaped frequency histogram of the features used in this study. Additionally, we examine the periodogram graph for similarity with graphs of generated Gaussian data with a matching mean and variance. The periodogram is also an indicator of correlation [41], [42]. In addition to visual verification, the mean square error (MSE) between the measured and synthetically generated Gaussian-shaped density is computed using

$$\%\text{MSE} = \frac{(\text{measured data}(i) - \text{generated data}(i))^2}{(\text{mean}(\text{measured data}))^2} \tag{40}$$

where "measured data(i)" denotes the normalized histogram of the feature x_i, "generated data(i)" is the value of the respective normal density, and "mean(measured data)" is the mean value of the histograms all the features used. We have experimentally shown that the random Gaussian nature of these features does not adversely affect their ability to discriminate network anomalies. In fact, the feature with the lowest MSE, as shown in Table I, is "average port" which turns out to be the most discerning data feature. The average port is an average of all the port numbers seen on the network and should not change drastically under normal conditions. On the other hand, "peer factor" is the feature with the highest MSE in terms of Gaussian density. Peer factor is a measurement of connections to the same port on both the source and destination side and indicates an a priori agreement between the two peer computers. The probability of two computers picking the same port at random is $1/(65535)^2 = 2.33 \cdot 10^{-10}$, which is negligibly small.

TABLE I
MSE SORTED FEATURE SET USED FOR NORMAL DENSITY APPROXIMATION AND ANOMALY DETECTION

%MSE	Measured Feature	Description
0.014403	Average port	Avg. port number as indicator of usage
0.055449	High-ports	Percentage of port numbers $>$ 10000
0.105316	Total ports	Number of ports seen
0.105888	Flow records	Count of flow records
0.119697	Total bits	Bits per second load on network
0.137958	Destination bits	Destination bits per second load
0.148073	Source bits	Source bits per second load
0.183783	Packets per second	Total packets per second
0.194217	Dest. packets	Destination packets per second
0.229815	Server factor	Measure of typical port usage
0.257600	Mid-range ports	Percentage of ports $>$ 1024 and $<$ 10000
0.284241	Low-range ports	Percentage of ports $<$ 1025
0.301142	Source packets	Source packets per second
2.109031	Total bytes	Total bytes per second
2.244572	Destination bytes	Destination bytes per second
4.569049	Source bytes	Source bytes per second
39.264246	Peer factor	Measure of same port usage

IV. IMPLEMENTATION AND RESULTS

A. Network Data

Network flow data is loosely defined as a collection of information about the bidirectional transfer of packets between a source IP address and port and a destination IP address and port. In this research, we consider an 11-tuple record consisting of protocol, source and destination rates, source and destination load, source and destination bytes, source and destination port, and source and destination IP addresses. Monitoring devices at the Ohio University Internet backbone collect this information

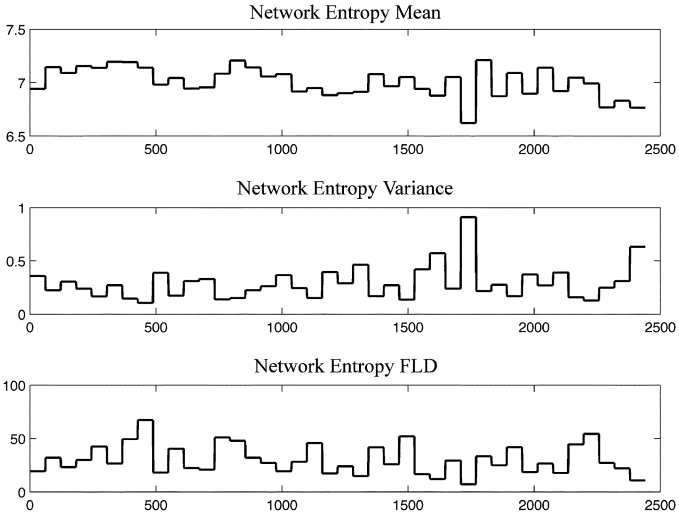


Fig. 6. Entropy mean, entropy variance, and entropy FLD, averaged over 60 s.

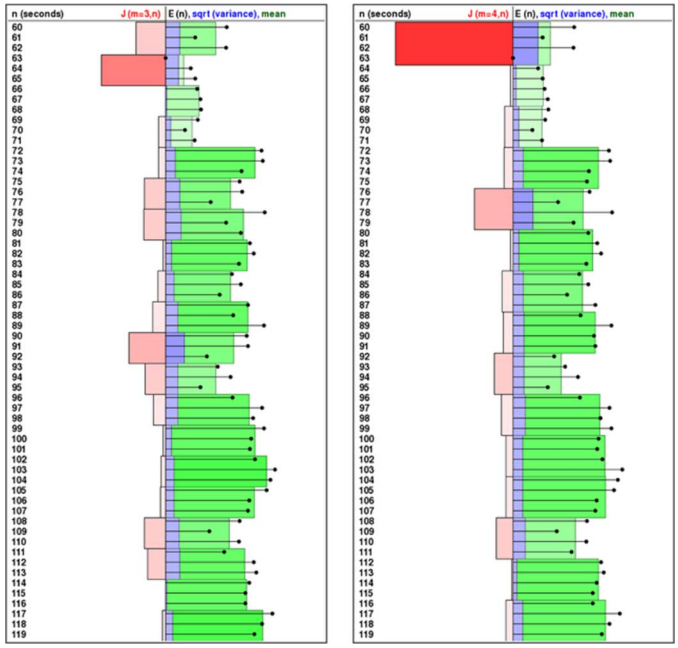


Fig. 7. Network entropy and FLD datagrams derived for destination port entropy over 60 s with sliding window sizes of $m = 3$ (left) and $m = 4$ (right).

in real-time and save it to disk for near-real-time statistical analysis and visualization. In order to process data as a time series, we collect all the records for the same time period into a single record representing one second in time. Cumulative statistics are gathered using a C++ program and the data are analyzed in MatLab functions based on [40, eqs. (9.44)–(9.46)]. Multiple experiments are run on various feature values and parameter settings in order to identify a system configuration which has the most discriminating power.

B. Data Processing

The approach, implemented in C++, first calculates per-second average entropy for each feature as defined in (6). Different window sizes (m) are applied to the entropy data stream in order to calculate the time averaged second-order statistics described in (8), (9), and (10). These calculations are used in the FLD (11) and shown on the left-hand side of the entropy graph. The resultant entropy graphs are subjected to FLD optimization as defined in (12) and (13) to determine the optimal window size. Fig. 6 shows one-dimensional plots of FLD in comparison with time-averaged mean and variance. Notice that this type of illustration is not as informative as the one that is developed in this research and subsequently explained.

C. Visualization and Detection Accuracy

The network entropy and the FLD graphs are designed to visually alert analysts about a network traffic anomaly. Formerly, engineers relied on a slow and detailed examination of the text-based display to make these determinations. With the proposed visualization, a network engineer can quickly capture, and comprehend at a glance, the statistical characteristics of the event while the software performs the first-pass examination of large amounts of network data without the need for human interaction. When an anomaly is visually indicated then a network engineer can decide whether the event warrants further labor-intensive investigation.

Current practice still involves security analysts when deciding what actions to take for any given anomaly, hence there are no automated responses built-in to the analysis tool. As engineers become comfortable with the results, then more automated interaction procedures may be incorporated.

The graphs themselves are formulated to create a visual distinction between two different (normal and anomaly) network traffic classes. The illustration of the mean is important to indicate the magnitude of the event and the variance shows the rate of change in events. High time-average mean and low variance supports the hypothesis stated that the FLD is an indicator of network anomalies. The importance of the FLD value J is visually emphasized by the height of the bar and the intensity of the color red while the mean and variance indicators are highlighted with blue and green of varying intensity, respectively.

The colorization of the graphs was also designed to aid in the rapid analysis of the stochastic event. An investigator, seeing brighter red knows that FLD is high, indicating a high level of interest in the anomaly. Seeing brighter blue tells the engineer that the information is highly variable and indicates a need for more detailed traffic analysis. The amount and brightness of green in the visualization tells the responder how much traffic information is being processed. Further enhancement of this colorization-based visualization is the subject of future research.

Changing window sizes can enhance the visibility of the anomaly as shown in Fig. 7, where a window size of 4 makes the anomaly more noticeable by increasing the size and brightness of the red FLD indicator. For example, an examination of Figs. 7 and 8 shows how network anomalies may be more or less visible, in a given time frame, depending on the system settings. The visibility of anomalies is made clearer by comparing visualizations of different data types over the same time period. For example, the anomaly that appears in Fig. 7 when examining the single feature "port" is nonevident in Fig. 8 where the total entropy of all features is considered for the same set of data. This illustration suggests that measuring the

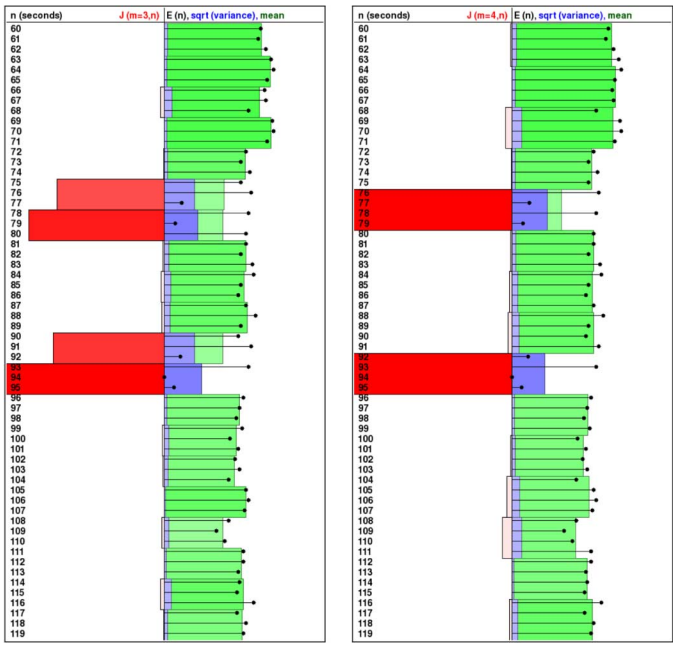


Fig. 8. Network entropy and FLD datagrams derived for total entropy over 60 s with sliding window sizes of $m = 3$ (left) and $m = 4$ (right).

entropy of multiple features in comparison with total entropy is essential for increasing detection accuracy (η_{accuracy}).

These figures also show that window size plays a big role in the efficacy of the proposed solution. Different window sizes are more or less effective at different times. In practice, several varying window sizes are run simultaneously and the optimal m is selected for any given moment in time. The voting scheme described in Fig. 5 assists in finding not only the optimal window size but also the optimal feature set to be used. For example, if we consider Fig. 7, the window size $m = 3$ is more informative at time 63 while window size 4 is more descriptive at time 76. Further, at time 90 and forward $m = 3$ appears to convey more valuable information. Although we show datagrams for varying window sizes (e.g., 3 and 4) for demonstration purposes, more widely varying widow sizes have been found to offer better results when converging on optimal settings as explained in Section IV. Use of multiple features and windows of varying sizes contributes to the event driven scalability and usability.

Detection accuracy is typically measured in terms of true positive and true negative results by the ratio of the number of correctly detected anomalies over the total number of anomalies which occurred during the observation period; i.e.,

$$\eta_{\text{accuracy}} = \frac{\#\text{ of correctly detected anomalies}}{\#\text{ number of actual anomalies}}. \qquad (41)$$

To further clarify, we define an anomaly as a pattern of traffic which varies from Gaussian distribution in the manner described in this research. Even expected, nonmalicious network events such as backups, maintenance outages, or scheduled downloads are identified as anomalies if the traffic patterns "vary from normal" for the given window size. These activities are also considered anomalies "by design," and as such, they are of interest to engineers and subject to justification by security analysts.

Hence, the definition of anomaly as a "variation from Gaussian density" also guides our analysis of false positives and false negatives. Since all anomalous traffic must be justified by network analysts, there is no false positive from that perspective. Further, since the definition of anomaly is purely statistical, the actual false positive and false negative error rates are determined using the underlying error rates of measured traffic when compared to Gaussian traffic. This normal traffic data is a statistical construct described in previous research by Celenk, *et al.* [22], [38] which serves as the ground truth for error rate calculations. The recording of an actual ground truth of traffic data for any significantly complex and unique network is not possible since the exact nature of normal and anomalous traffic is not known. The unknown nature of network traffic patterns is the underlying reason for applying a statistics-based approach to anomaly detection. This research shows that the measure of accuracy is inversely proportional to the mean squared error measurements of Table I; i.e., $\eta_{\text{accuracy}} \propto (1/\%\text{MSE})$. For example, by selecting "Average port" as a feature from Table I, the detection accuracy can reach a value of 0.69 or approximately 70% whereas the feature "Peer factor" from the same table yields almost 0.03 or 3% accuracy. Thus, a feature that has a small MSE yields results that have better accuracy because the data statistics are closer to a normal distribution. On the other hand, a feature with a high MSE results in a lesser accuracy and, therefore, it is not selected. The presentation, management and logging of all network anomalies is a matter for developers of the final software application. Since error rates are determined by the underlying statistics and not by individuals, all events are monitored by the system, although operators can manage the visibility of anomalies by changing the underlying configuration parameters such as window size. Furthermore, administrators can provide feedback during the initial configuration of the network.

D. Anomaly Prediction

This section describes the results of the normal density approximation study and the prediction algorithm results. Fig. 9 shows the periodogram, correlation results, and histogram of a selected feature set; namely x_1 = "Average port," x_2 = "High ports," x_2 = "Server factor," and x_2 = "Peered factor." Similarity between the measured feature values and generated Gaussian data for all features are clearly visible except for the attribute "Peered factor." The significant variation may be attributed to the highly stochastic nature of the traffic. In Fig. 10, we show the auto- and cross-correlation plots of the selected features in order to determine the level of correlation for this measured feature set. Notice that, while (x_1, x_2) and (x_1, x_3) possess high correlation, the remaining pairs do not. This implies that features x_1, x_2, and x_3 should be uncorrelated before they are used individually or collectively. Fig. 11 depicts 100 s of collected real-time data and illustrates the prediction results at different stages in the ADAP algorithm. In subplots (A) and (B), a dashed line shows the boundary of the Wiener and auto-correlation windows, respectively. It appears that the initiation time of the attack is 2 s starting from time 261 s through 263 s. Once the attack is detected at time 263 s, preventive measures are taken by the network analyst to prevent

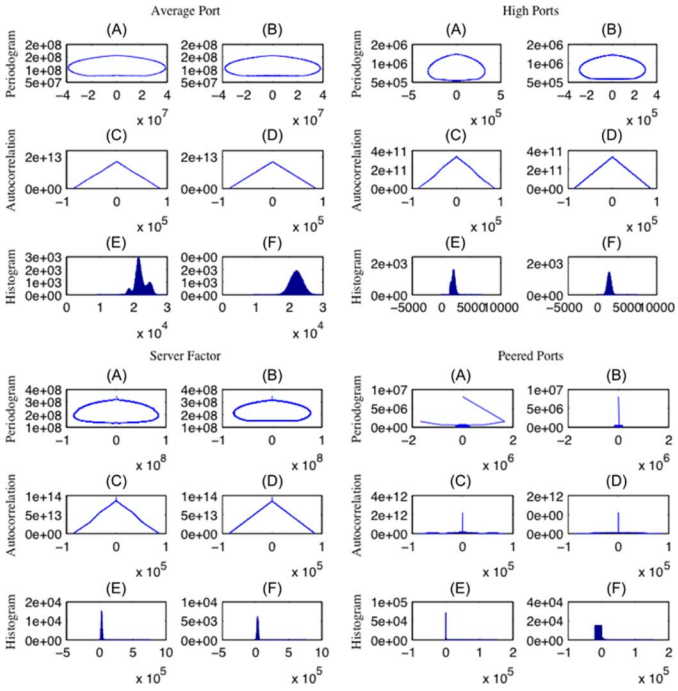


Fig. 9. Snapshots of (A) measured signal, (B) Wiener output, (C) ARMA prediction, (D) difference between Wiener input and output, (E) difference between measured signal and ARMA output, and (F) difference between Wiener and ARMA outputs.

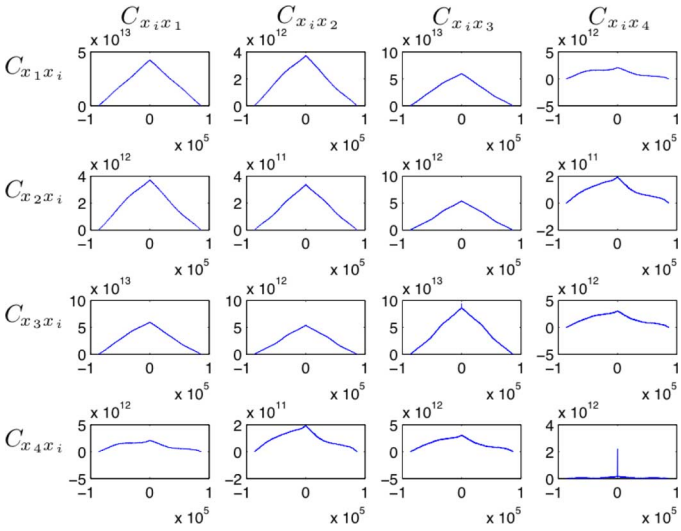


Fig. 10. Auto- and cross-correlation plots for features x_1 (average port), x_2 (high ports), x_3 (server factor), and x_4 (peered factor) showing varying degrees of correlation among them.

damage to the network. Part (C) represents the ARMA predictor signal, and (D) shows the difference between measured values and ARMA output. The solid vertical line indicates a predicted anomaly at time $n + 1$ (263 s). Notice that the peak in predictor measurements (E) and (F) occurs just before the maximum value of the actual anomaly in (A). This supports our conclusion that the algorithm predicts network anomalies. In Fig. 12, we demonstrate the affect of varying parameters such as window size (shaded area). The solid vertical lines indicate locations in time of predicted anomalies. Part (A) corresponds to smaller sized windows while part (B) represents

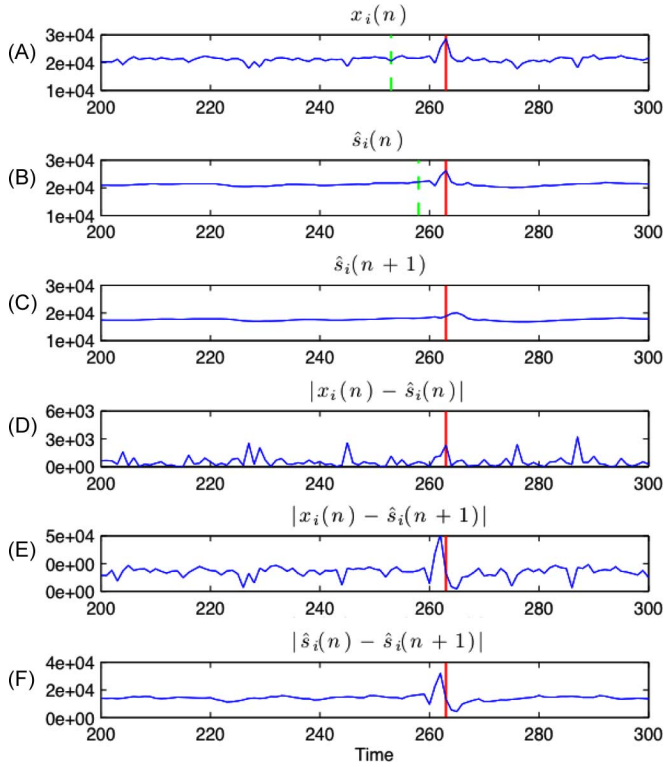


Fig. 11. Snapshots of (a) measured signal $x_i(n)$, (b) Wiener output $\hat{s}_i(n)$, (c) ARMA prediction $\hat{s}_i(n+1)$, (d) magnitude of the difference between Wiener input and output $|x_i(n) - \hat{s}_i(n)|$, (e) magnitude of the difference between measured signal and ARMA output $|x_i(n) - \hat{s}_i(n+1)|$ as anomaly measure, and (F) magnitude of the difference between Wiener and ARMA outputs $|\hat{s}_i(n) - \hat{s}_i(n+1)|$ as anomaly predictor.

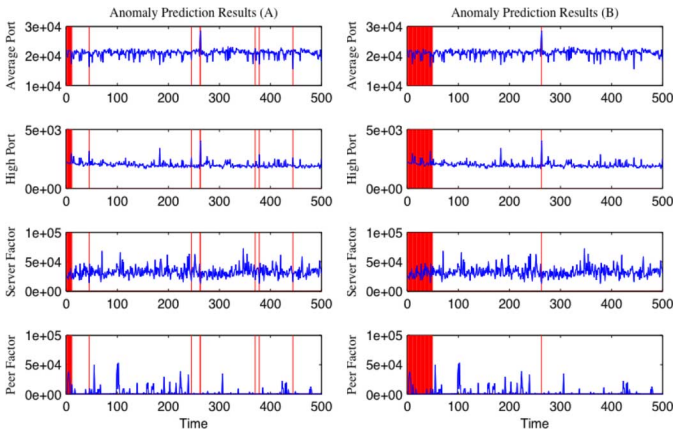


Fig. 12. Experimental results of sample runs on x_1, x_2, x_3, and, x_4, showing the effect of changing parameters. The solid vertical lines indicate locations in time of predicted anomalies. Part (A) corresponds to smaller sized windows while part (B) represents larger window sizes.

larger window sizes. The system predicts anomalies of various magnitudes by changing parameters. This robust performance is due to the fact that the ADAP unit has a feedback control, which allows it to adjust to the changing signal waveform.

V. Conclusions and Discussions

We have presented an approach that is based on relatively short-term observations of network features and their respective time averaged entropy values. This approach helps in de-

termining acute and long-term changes in the network feature space and presents system status in a visually compact information graph (called datagrams). First, average entropy for each feature is calculated for every second of observation. Then, the resultant short-term information measurement is subjected to first- and second-order time averaging statistics. The process then applies the modified FLD to the underlying time averaged network entropy to identify the exact time of the security incident or attack on the network. The colorized dual-entropy-type datagram visualization is devised to help network engineers interact with the staggering amounts of network data. The proposed method has been tested under load on real-time network traffic data from Ohio University's main Internet connection. Experimentation has shown that the presented algorithm is able to identify anomalies in selected network features including average port, high port, server port, and peered port. Although the present work is currently used to detect such diverse network attacks as BotNets, worm outbreaks, and DoS's, it really identifies any anomaly of interest to network and security analysts. We are proceeding to evaluate this research in conjunction with existing network defense systems. Furthermore, additional research is needed to implement the proposed method in a time-varying multidimensional feature space by considering the correlations of additional network features.

REFERENCES

[1] R. Kwitt and U. Hofmann, "Unsupervised anomaly detection in network traffic by means of robust PCA," in *Proc. Int. Conf. Computing in the Global Information Technology (ICCGI)*, 2007, pp. 37–37.

[2] G. Shen, D. Chen, and Z. Qin, "Anomaly detection based on aggregated network behavior metrics," in *Proc. Wireless Communications, Networking and Mobile Computing, 2007 (WiCom 2007)*, pp. 2210–2213.

[3] A. Karasaridis, B. Rexroad, and D. Hoeflin, "Wide-scale botnet detection and characterization," in *Proc. First Conf. First Workshop on Hot Topics in Understanding Botnets (HotBots'07)*, Berkeley, CA, 2007, p. 7, USENIX Association.

[4] S. Jin, D. S. Yeung, and X. Wang, "Network intrusion detection in covariance feature space," *Pattern Recognit.*, vol. 40, no. 8, pp. 2185–2197, 2007.

[5] A. Sang and S. Li, "A predictability analysis of network traffic," in *Proc. INFOCOM (1)*, 2000, pp. 342–351.

[6] K. Cho, R. Kaizaki, and A. Kato, "An aggregation technique for traffic monitoring," in *Symp. Applications and the Internet (SAINT) Workshops*, 2002, p. 74.

[7] R. Pang, V. Yegneswaran, P. Barford, V. Paxson, and L. Peterson, "Characteristics of internet background radiation," in *Proc. ACM Internet Measurement Conf.*, Oct. 2004, pp. 27–40.

[8] A. Feldmann, A. C. Gilbert, and W. Willinger, "Data networks as cascades: Investigating the multifractal nature of internet WAN traffic," in *Proc. SIGCOMM*, 1998, pp. 42–55.

[9] W. Yurcik and Y. Li, "Internet security visualization case study: Instrumenting a network for netflow security visualization tools," in *Proc. Annual Computer Security Applications Conf. (ACSAC 05)*, Tucson, AZ, Dec. 5–9, 2005.

[10] D. Plonka, "A network traffic flow reporting and visualization tool," in *Proc. 14th USENIX Conf. System Administration*, New Orleans, LA, 2000, pp. 305–318.

[11] Y. Gong, Security Focus Article: Detecting Worms and Abnormal Activities With NetFlows, Part 1 Aug. 2004 [Online]. Available: http://www.securityfocus.com/infocus/1796

[12] Y. Gong, Security Focus Article: Detecting Worms and Abnormal Activities With NetFlows, Part 2 Sep. 2004 [Online]. Available: http://www.securityfocus.com/infocus/1796

[13] V. Alarcon-Aquino and J. A. Barria, "Multiresolution FIR neural-network-based learning algorithm applied to network traffic prediction," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 36, no. 2, pp. 208–220, Mar. 2006.

[14] Y. Gu, A. McCallum, and D. Towsley, "Detecting anomalies in network traffic using maximum entropy estimation," in *Proc. 5th ACM SIGCOMM Conf. Internet Measurement (IMC '05)*, New York, 2005, pp. 1–6, ACM.

[15] A. Lakhina, M. Crovella, and C. Diot, Mining Anomalies Using Traffic Distributions CS Department, Boston University, Tech. Rep. 2005-002, Feb. 2005.

[16] S. Foresti, J. Agutter, Y. Livnat, S. Moon, and R. F. Erbacher, "Visual correlation of network alerts," *IEEE Comput. Graphics Applicat.*, vol. 26, no. 2, pp. 48–59, Mar./Apr. 2006.

[17] R. Eimann, U. Speidel, N. Brownlee, and J. Yang, Network Event Detection With T-Entropy Centre for Discrete Mathematics and Theoretical Computer Science, University of Auckland, New Zealand, Rep. CDMTCS-266, May 2005.

[18] A. Lall, V. Sekar, M. Ogihara, J. J. Xu, and H. Zhang, Data Streaming Algorithms for Estimating Entropy of Network Traffic Computer Science Department, University of Rochester, Tech. Rep. TR886, Nov. 2005.

[19] E. F. Harrington, "Measuring network change: Rényi cross entropy and the second order degree distribution," in *Proc. Passive and Active Measurement (PAM) Conf.*, Adelaide, Australia, Mar. 2006.

[20] S. Gianvecchio and H. Wang, "Detecting covert timing channels: An entropy-based approach," in *Proc. ACM Conf. Computer and Communications Security*, 2007, pp. 307–316.

[21] S. S. Kim, A. L. N. Reddy, and M. Vannucci, "Detecting traffic anomalies through aggregate analysis of packet header data," in *Networking*. New York: Springer, 2004, vol. 3042, pp. 1047–1059.

[22] M. Celenk, T. Conley, J. Graham, and J. Willis, "Anomaly prediction in network traffic using adaptive Wiener filtering and ARMA modeling," in *Proc. IEEE Int. Conf. Systems, Man, and Cybernetics*, Oct. 2008, pp. 3548–3553.

[23] X. Fu, B. Graham, R. Bettati, and W. Zhao, "On effectiveness of link padding for statistical traffic analysis attacks," in *Proc. 23rd IEEE Int. Conf. Distributed Computing Systems (ICDCS '03)*, Washington, DC, 2003, p. 340.

[24] A. Wagner and B. Plattner, "Entropy based worm and anomaly detection in fast IP networks," in *Proc. 14th IEEE Int. Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise*, Washington, DC, 2005, pp. 172–177.

[25] M. Thottan and C. Ji, "Anomaly detection in IP networks," *IEEE Trans. Signal Process.*, vol. 51, no. 8, pp. 2191–2204, Aug. 2003.

[26] H. Hajji, "Statistical analysis of network traffic for adaptive faults detection," *IEEE Trans. Neural Netw.*, vol. 16, no. 5, pp. 1053–1063, Sep. 2005.

[27] S. R. Gaddam and K. S. Balagani, "K-Means+ID3: A novel method for supervised anomaly detection by cascading K-Means clustering and ID3 decision tree learning methods," *IEEE Trans. Knowl. Data Eng.*, vol. 19, no. 3, pp. 345–354, Mar. 2007.

[28] A. Ziviani, M. L. Monsores, P. S. S. Rodrigues, and A. T. A. Gomes, "Network anomaly detection using nonextensive entropy," *IEEE Commun. Lett.*, vol. 11, no. 12, pp. 1034–1036, Dec. 2007.

[29] S. S. Kim and A. L. N. Reddy, "Statistical techniques for detecting traffic anomalies through packet header data," *IEEE/ACM Trans. Netw.*, vol. 16, no. 3, pp. 562–575, Jun. 2008.

[30] G. Androulidakis, V. Chatzigiannakis, and S. Papavassiliou, "Network anomaly detection and classification via opportunistic sampling," *IEEE Network*, vol. 23, no. 1, pp. 6–12, Jan./Feb. 2009.

[31] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*, 3rd ed. New York: Academic, Feb. 2006.

[32] B. Porat, *Digital Processing of Random Signals Theory and Methods*. Englewood Cliffs, NJ: Prentice-Hall, 1994.

[33] S. M. Asmoredjo, "A Probabilistic Model for Cyber Attacks and Protection," Master's thesis, Delft University of Technology, Delft, The Netherlands, Jun. 2005.

[34] A. Schneider, "Methods of Internet Worm Propagation," Master's thesis, Wake Forest University, Winston-Salem, NC, 2009, 27109.

[35] G. Travis, E. Balas, D. A. J. Ripley, and S. Wallace, Analysis of the "SQL Slammer" Worm and Its Effects on Indiana University and Related Institutions Feb. 2003 [Online]. Available: http://paintsquirrel.ucs.indiana.edu/pdf/SLAMMER.pdf, pp. 1–18

[36] M. Abu Rajab, F. Monrose, and A. Terzis, "Worm evolution tracking via timing analysis," in *Proc. 2005 ACM Workshop on Rapid Malcode (WORM '05)*, New York, 2005, pp. 52–59.

[37] B. C. Bag, S. K. Banik, and D. S. Ray, "The noise properties of stochastic processes and entropy production," *Phys. Rev. E*, vol. 64, p. 026110, 2001.

[38] M. Celenk, T. Conley, J. Willis, and J. Graham, "Anomaly detection and visualization using Fisher discriminant clustering of network entropy," in *Proc. IEEE Third Int. Conf. Digital Information Management (ICDIM 2008)*, Nov. 2008, pp. 216–220.
[39] W. Stallings, *Data and Computer Communications (7th Edition)*. Englewood Cliffs, NJ: Prentice-Hall, 2004.
[40] J. S. Lim, *Two-Dimensional Signal and Image Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1990.
[41] W. L. Crum, "Tests for serial correlation in regression analysis based on the periodogram of least-squares residuals," *J. Amer. Statist. Assoc.*, vol. 18, no. 143, pp. 889–899, Sep. 1923.
[42] J. Durbin, "The resemblance between the ordinate of the periodogram and the correlation coefficient," *Biometrika*, vol. 56, no. 1, pp. 1–15, Mar. 1969.

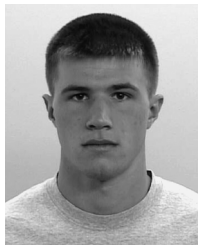
Mehmet Celenk (S'84–M'85) received the B.S. and M.S. degrees from Istanbul Technical University, in 1974 and 1976, in electrical and communications engineering, and the Ph.D. degree from Stevens Institute of Technology in electrical engineering and computer science, in 1983, where he was the Robert Crook Stanley Graduate Fellow in 1985.

He served on the Turkish Army from 1984 to 1985 as a lieutenant. He joined Ohio University in 1985, where he is currently an Associate Professor of Electrical Engineering and Computer Science. His research interests include image and video procession, computer vision, multimedia, data fusion, pattern recognition, and biomedical engineering. He has published over 200 articles in these fields and received over $0.5 M hypercube processor equipment grant from Symult and Ametek Co. He supervised two visiting scholars, one of whom was awarded $20 K from TUBITAK, and 31 M.S./Ph.D. theses.

Dr. Celenk received the distinguished service award from the Signal School in Ankara for his R&D work and his associate editorial services for the *Communications Journal*. He was the recipient of the 1988 Fritz & Dolores Russ Research Award of the College of Engineering and Technology of Ohio University. He was awarded the Avionics Academic Challenge Faculty Fellowship at Ohio University from 1988 to 1992. He has been an active reviewer for various professional societies, journals, publishers, and the NSF. He has been a review panelist for the New York State Office of Science, Technology & Academic Research (NYSTAR) in 2002–2007. He has been an Associate Editor of the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART A: SYSTEMS AND HUMANS, since 2005, a member of the Editorial Board of the *Journal of Recent Patents on Signal Processing*, since 2008, and served as session-chair/organizing-committee-member of ICDP 2009, ICSMC 2008, IASTED ICSIP 2008, VISAPP 2008, ICIP 2006, ICPR 2006, SSST 2005–1992, CGIP 2000, and CVPR 1999 conferences. He is a member of Eta Kappa Nu, and former member of SPIE, IS&T, ACM, ASEE, and OE.

Thomas Conley received the B.S. degree in environmental resources management from The Pennsylvania State University, in 1983. He is working toward the Master's degree in the School of Electrical Engineering and Computer Science at Ohio University, specializing in computer networking and information security.

He then joined the Peace Corps as a Fisheries Extension Officer and served in Kenya from 1983 to 1986. He is a Certified Information Systems Security Professional (CISSP). He has worked as a computer consultant for various government organizations including the U.S. State Department, the U.S. Department of Education, and the Office of Research and Development for the Central Intelligence Agency. In addition, he was a research and development engineer for the wireless industry before joining Ohio University as a Senior Network Security Analyst where he is responsible for numerous information security-related security monitoring, computer and network forensics, advising network architecture and engineering efforts, and developing custom technical solutions in support of information security. His research interests include data mining, statistical data analysis, and visualization for network traffic applications. He recently published two papers; one in IEEE ICDIM 2008 and the other in IEEE ICSMC 2008.



John Willis (S'10) received the B.S. degree in electrical and computer engineering from Ohio University, Athens, OH, in 2009, where he was a member of the men's varsity track team running the 400 meter dash. He is working toward the M.S. degree in electrical engineering and computer science from Ohio University, specializing in data fusion and image processing.

He started his research career when he was a junior student in his B.S. program as an integral member of a team seeking a potential application of pattern recognition research in network traffic. He participated in publishing papers and presenting at IEEE ICDIM 2008 and IEEE ICSMC 2008. In addition, he is working on a cost estimation project for General Electric's Jet Engine division.

James Graham (S'07) received the B.S. degree in electrical engineering from the University of Michigan, Ann Arbor, in 2001, and the M.S. degree in electrical engineering from Ohio University, Athens, OH, in 2007. He is currently working toward the Ph.D. degree in electrical engineering and computer science at Ohio University.

He worked in industry as a research contractor with Kelly Scientific at Dow Chemical for a few years. He specializes in image processing, intelligent machines and systems, in which he has contributed to several publications.