2013-12

Network Anomaly Detection: A Survey and Comparative Analysis of Stochastic and Determinis...

Network Anomaly Detection: A Survey and Comparative Analysis of Stochastic and Deterministic Methods*

Jing Wang, † Daniel Rossell,‡ Christos G. Cassandras,§ and Ioannis Ch. Paschalidis §

***Abstract*— We present five methods to the problem of network anomaly detection. These methods cover most of the common techniques in the anomaly detection field, including Statistical Hypothesis Tests (SHT), Support Vector Machines (SVM) and clustering analysis. We evaluate all methods in a simulated network that consists of nominal data, three flow-level anomalies and one packet-level attack. Through analyzing the results, we point out the advantages and disadvantages of each method and conclude that combining the results of the individual methods can yield improved anomaly detection results.**

I. Introduction

A network anomaly is any potentially malicious traffic that has implications for the security of the network. It is of particular importance to the prevention of zero-day attacks, i.e., attacks not previously seen, and malicious data exfiltration. These are key areas of concern for both government and corporate entities.

From the perspective of methodology, network anomaly detection methods can be classified as stochastic and deterministic. Stochastic methods fit reference data to a probabilistic model and evaluate the fitness of the new traffic with respect to this model [7], [11], [12], [22], [29]. The evaluation can be done using Statistical Hypothesis Testing (SHT) [5], [13], [16], [19]. Deterministic methods, on the other hand, try to partition the feature space into "normal" and "abnormal" regions through a deterministic decision boundary. The boundary can be determined using methods like Support Vector Machine (SVM), particularly 1-class SVM [9], [21], [25], and clustering analysis [1], [6].

From the perspective of data, network anomaly methods can be either packet-based [7], [17], flow-based [2], [18] or window-based [12]. Packet-based methods evaluate the raw packets directly while both flow-based and window-based methods aggregate the packets first. Flow-based methods evalulate each flow individually, which is defined as a collection of packets with similar properties. Flows are considered as a good tradeoff between cost of collection and level of detail [26]. Window-based methods group consecutive packets or flows based on a sliding window.

* Research partially supported by the ARO under grants W911NF-11-1-0227 and 61789-MA-MUR, by the NSF under grants EFRI-0735974, CNS-1239021 and IIS-1237022, by the AFOSR under grant FA9550-12-1-0113, and by the ONR under grants N00014-09-1-1051 and N00014-10-1-0952.

† Division of Systems Eng., Boston University, email: wangjing@bu.edu.

‡ Department of Electrical and Computer Eng., Boston University, email: aggie0642@googlemail.com.

§ Department of Electrical and Computer Eng., and Division of Systems Eng., Boston University, 8 Saint Mary's St., Boston, MA 02215, e-mails: cgc@bu.edu and yannisp@bu.edu.

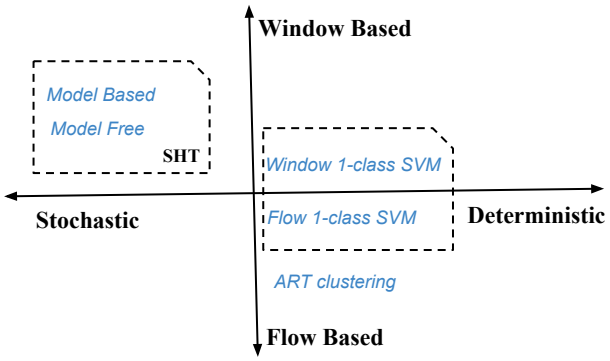

Fig. 1. Relationships among the five evaluated methods.

The main goal of this paper is to discuss the advantages and distadvantages of each category of methods for different applications. This paper presents five methods, covering most of the categories discussed above, to the problem of host-based network anomaly detection and provides a comparative analysis. The first four methods are revisions of authors' previous work with collaborators and the last method is new.

The first two methods are based on SHT, utilizing results from Large Deviations Theory (LDT) [4] to compare current network traffic to probability laws governing nominal network traffic. The two methods fit traffic, which is a sequence of flows, with probabilistic models under i.i.d. and Markovian assumptions, respectively. We refer to these two methods as *model-free* and *model-based* methods.

The next two methods are based on an 1-class SVM. In the first of the two methods, individual data transmissions from a given source are examined independently from neighboring transmissions, producing a flow-by-flow detector. In the other, sequences of flows within a time window are considered together to construct a window-based detector. These two methods will be called *flow 1-class SVM* and *window 1-class SVM* method, respectively.

Finally, we also present a clustering method based on Adaptive Resonance Theory (ART) [3], which is a machine learning technique originating in the biology field. This algorithm, named *ART clustering* [23], partitions network traffic into clusters based on the unique features of the network flows.

The relationships among these methods are depicted in Figure 1. The *flow 1-class SVM* and the *ART clustering* method are flow-based and capable of identifying individual network flows that are anomalous. By contrast, the remaining methods are window-based, under which the flows are grouped into a window based on their start time and only

suspicious windows of time can be identified as anomalous. *Model-free* and *model-based* methods are stochastic and the rest methods are deterministic.

A challenging problem in the evaluation of anomaly detection methods is the lack of test data with ground truth, due to the limited availability of such data. The most widely used labeled dataset, DAPRA intrusion detection dataset [14], was collected 14 years ago. Since then, the network condition has changed significantly. In order to address this problem, we developed software to generate labeled data, including a flow-level anomaly data generator (SADIT [28]) and a packet-level botnet attack data generator (IMALSE [27]). We evaluate all of our methodologies on a simulated network and compare their performance under three flow-level anomalies and one Distributed Denial of Service (DDoS) attack.

The rest of the paper is organized as follows. Section II describes the representation of network traffic data. Section III provides a mathematical description of the methods used to identify anomalies. Section IV provides an in-depth explanation of the simulated network and the anomalies. Section V presents the results of the five methods on the simulated network data. Finally, Section VI provides concluding remarks.

II. Network Traffic Representation

Let $\mathcal{S} = \{\mathbf{s}^1, \ldots, \mathbf{s}^{|\mathcal{S}|}\}$ denote the collection of all packets on the server which is monitored, where each element of $\mathcal{S}$ is one packet. We focus on host-based anomaly detection, in which case we only care about the user IP address, namely the destination IP addresses for the outgoing packets and the source IP addresses for the incoming packets. Denote the user IP address in a packet $\mathbf{s}^i$ as $\mathbf{x}^i$, whose format will be discussed later. The size of $\mathbf{s}^i$ is $b^i \in [0, \infty)$ in bytes and the start time of transmission is $t^i \in [0, \infty)$ in seconds. Using this convention, the packet $\mathbf{s}^i$ can be represented as $(\mathbf{x}^i, b^i, t^i)$ for all $i = 1, \ldots, |\mathcal{S}|$.

Due to the vast number of packets, we consolidate this representation of network traffic by grouping series of packets into flows. We compile a sequence of packets $\mathbf{s}^1 = (\mathbf{x}^1, b^1, t^1), \ldots, \mathbf{s}^n = (\mathbf{x}^n, b^n, t^n)$ with $t^1 < \cdots < t^n$ into a *flow* $\mathbf{f} = (\mathbf{x}, b, d_t, t)$ if $\mathbf{x} = \mathbf{x}^1 = \cdots = \mathbf{x}^n$ and $t^i - t^{i-1} < \delta_F$ for $i = 2, \ldots, n$ and some prescribed $\delta_F \in (0, \infty)$. Here, the size b is simply the sum of the sizes of the packets that comprise the flow. The value $d_t = t^n - t^1$ denotes the flow duration. The value $t = t^1$ denotes the start time of the first packet of the flow. In this way, we can translate the large collection of traffic packets $\mathcal{S}$ into a relatively small collection of flows $\mathcal{F} = \{\mathbf{f}^1, \ldots, \mathbf{f}^{|\mathcal{F}|}\}$.

In some applications in which large numbers of users frequently access the server under surveillance, it may be infeasible to characterize network behavior for each user. Different methods deal with this dilemma differently.

For both statistical and SVM methods, we first distill the "user space" into something more manageable while enabling us to characterize network behavior of user groups instead of just individual users. For simplicity of notation, we only consider IPv4 addresses. If $\mathbf{x}^i = (x_1^i, x_2^i, x_3^i, x_4^i) \in \{0, 1, \ldots, 255\}^4$ and $\mathbf{x}^j = (x_1^j, x_2^j, x_3^j, x_4^j) \in \{0, 1, \ldots, 255\}^4$ are two IPv4 addresses, the *distance* between them is defined as: $d(\mathbf{x}^i, \mathbf{x}^j) = \sum_{k=1,\ldots,4} 256^{4-k} |x_k^i - x_k^j|$. This metric can be easily extended to IPv6 addresses if needed. Suppose $\mathcal{X}$ is the set of unique IP addresses in $\mathcal{F}$. We apply typical K-means clustering on $\mathcal{X}$ [8], [15]. For each $\mathbf{x} \in \mathcal{X}$, we thus obtain a cluster label $k(\mathbf{x})$. Suppose the cluster center for cluster k is $\bar{\mathbf{x}}^k$, then the distance of $\mathbf{x}$ to the corresponding cluster center is $d_a(\mathbf{x}) = d(\mathbf{x}, \bar{\mathbf{x}}^{k(\mathbf{x})})$. Using user clusters, we can produce our final representation of a flow as:

$$\mathbf{f} = (k(\mathbf{x}), d_a(\mathbf{x}), b, d_t, t). \tag{1}$$

For the *ART clustering* method, distilling the user space beforehand is not required. However, instead of using the IP address directly, we use a compact representation. Let $n_f(\mathbf{x})$ be the number of flows transmitted between the user with IP address $\mathbf{x}$ and the server. Define $d_b(\mathbf{x}) = d(\mathbf{x}, \mathbf{x}^*), \forall \mathbf{x} \in \mathcal{X}$, where $\mathbf{x}^*$ is the IP address of the server we are monitoring; then the alternative flow representation we use is:

$$\mathbf{f} = (n_f(\mathbf{x}), d_b(\mathbf{x}), b, d_t, t). \tag{2}$$

III. Anomaly Detection Methods

A. Statistical Methods

Let h be the interval between the start points of two consecutive time windows and w_s be an appropriate window size; then the total number of windows is $n_w = \lceil (t^{|\mathcal{F}|} - t^1 - w_s)/h \rceil$. We say flow $\mathbf{f}^i = (k(\mathbf{x}^i), d_a(\mathbf{x}^i), b^i, d_t^i, t^i)$ belongs to window j if $t^1 + (j-1)h \leq t^i < t^1 + (j-1)h + w_s, \ \forall j = 1, \ldots, n_w$.

Let $\mathbf{g}^i = (k(\mathbf{x}^i), d_a(\mathbf{x}^i), b^i, d_t^i)$ be the flow attributes in $\mathbf{f}^i$ without the start time t^i and $\mathcal{G}_j = \{\mathbf{g}^1, \mathbf{g}^2, \ldots, \mathbf{g}^{|\mathcal{G}_j|}\}$ be the flows in window j. Let $\mathcal{G}_{ref}$ be the set of all flows used as reference. The window-based methods will compare $\mathcal{G}_j$ with $\mathcal{G}_{ref}$ for all $j = 1, \ldots n_w$. Both statistical methods we will present in this section fall into this category and can work in supervised as well as unsupervised modes. In supervised mode, $\mathcal{G}_j$ is generated by removing suspicious flows from a small fragment of data through human inspection. In unsupervised mode, we assume that the anomales are short-lived thus $\mathcal{G}_j$ can be chosen as a large set of nework traffic.

Since the approach introduced in what follows applies to all windows as well as to nominal flows, we use $\mathcal{G} = \{\mathbf{g}^1, \ldots, \mathbf{g}^{|\mathcal{G}|}\}$ to refer to $\mathcal{G}_{ref}$ and $\mathcal{G}_j, \forall j = 1, \ldots n_w$. Suppose the range of $d_a(\mathbf{x}^i), \forall i = 1, \ldots, |\mathcal{G}|$ is $[d_a^{min}, d_a^{max}]$. We can then define a discrete alphabet $\Sigma_{d_a} = \left\{ d_a^{min} + (m + 1/2) \times (d_a^{max} - d_a^{min})/|\Sigma_{d_a}| \right\}_{m=0,\ldots,|\Sigma_{d_a}|-1}$ for $d_a(\mathbf{x}^i)$, where $|\Sigma_{d_a}|$ is called quantization level. Σ_b and Σ_{d_t} can be defined similarly for b^i and d_t^i. We then quantize $d_a(\mathbf{x}^i)$, b^i and d_t^i in $\mathbf{g}^i$ to the closest symbol in the discrete alphabet set Σ_{d_a} and Σ_b and Σ_{d_t}, respectively. Suppose the total number of user clusters is K. Then we can denote the quantized flow sequence $\mathcal{G} = \{\mathbf{g}^1, \ldots, \mathbf{g}^{|\mathcal{G}|}\}$ as $\Sigma(\mathcal{G}) = \{\boldsymbol{\sigma}(\mathbf{g}^1), \ldots, \boldsymbol{\sigma}(\mathbf{g}^{|\mathcal{G}|})\}$, where $\Sigma = \{0, \ldots, K-1\} \times \Sigma_{d_a} \times \Sigma_b \times \Sigma_{d_t}$ is the

discrete alphabet for quantization where each symbol in Σ corresponds to a flow state.

1) Model-free Method: In cases in which all flows emanating from the server under surveillance are i.i.d., we construct the *empirical measure* of flow sequence $\mathcal{G} = \{\mathbf{g}^1, \ldots, \mathbf{g}^{|\mathcal{G}|}\}$ as the frequency distribution vector

$$\mathcal{E}^{\mathcal{G}}(\boldsymbol{\rho}) = \frac{1}{|\mathcal{G}|} \sum_{i=1}^{|\mathcal{G}|} \mathbf{1}\{\boldsymbol{\sigma}(\mathbf{g}^i) = \boldsymbol{\rho}\}, \tag{3}$$

where $\mathbf{1}\{\cdot\}$ denotes the indicator function and $\boldsymbol{\sigma}(\mathbf{g}^i)$ denotes the flow state in Σ that $\mathbf{g}^i$ gets mapped to. We will denote the probability vector derived from the empirical measure of the form in (3) as $\boldsymbol{\mathcal{E}}^{\mathcal{G}} = \left\{\mathcal{E}^{\mathcal{G}}(\boldsymbol{\sigma}^1), \ldots, \mathcal{E}^{\mathcal{G}}(\boldsymbol{\sigma}^{|\Sigma|})\right\}$.

Let $\boldsymbol{\mu}$ denote the probability vector calculated from the reference flows $\mathcal{G}_{ref}$. That is, $\mu(\sigma)$ is the reference marginal probability of flow state σ. Using Sanov's theorem [16], [4], we construct a metric to compare empirical measures of the form in (3) to $\boldsymbol{\mu}$, thus a metric of the "normality" of a sequence of flows. For every probability vector $\boldsymbol{\nu}$ with support Σ, let $H(\boldsymbol{\nu}|\boldsymbol{\mu}) = \sum_{\boldsymbol{\sigma}\in\Sigma} \nu(\boldsymbol{\sigma}) \log\left(\nu(\boldsymbol{\sigma})/\mu(\boldsymbol{\sigma})\right)$ be the relative entropy of $\boldsymbol{\nu}$ with respect to $\boldsymbol{\mu}$. Allowing $\eta = -\frac{1}{n}\log\epsilon$, where ϵ is a tolerable false alarm rate, then the *model-free* anomaly detector is:

$$\mathcal{I}(\mathcal{G}) = \mathbf{1}\{I_1(\boldsymbol{\mathcal{E}}^{\mathcal{G}}) \geq \eta\}, \tag{4}$$

where $I_1(\boldsymbol{\mathcal{E}}^{\mathcal{G}}) = H(\boldsymbol{\mathcal{E}}^{\mathcal{G}}|\boldsymbol{\mu})$. It was shown in [19] that (4) is asymptotically Neyman-Pearson optimal.

2) Model-based Method: As an alternative to the i.i.d. assumption on the sequence of flows under the *model-free* method, we now turn to the case in which the sequence of flows adheres to a first-order Markov chain. The notion of empirical measure on the sequence $\mathcal{G} = \{\mathbf{g}^1, \ldots, \mathbf{g}^{|\mathcal{G}|}\}$ must now be adapted to consider subsequent pairs of flow states. We assume no knowledge of an initial flow state $\boldsymbol{\sigma}(\mathbf{g}^1)$ and define the empirical measure on $\mathcal{G}$, under the Markovian assumption, as the frequency distribution on the possible flow state transitions,

$$\mathcal{E}_B^{\mathcal{G}}(\boldsymbol{\sigma}^i, \boldsymbol{\sigma}^j) = \frac{1}{|\mathcal{G}|} \sum_{l=2}^{|\mathcal{G}|} \mathbf{1}\left\{\boldsymbol{\sigma}(\mathbf{g}^{l-1}) = \boldsymbol{\sigma}^i, \boldsymbol{\sigma}(\mathbf{g}^l) = \boldsymbol{\sigma}^j\right\}, \tag{5}$$

where $\mathbf{1}\{\cdot\}$ denotes the indicator function and $\boldsymbol{\sigma}(\mathbf{g}^l)$ denotes the flow state in Σ which $\mathbf{g}^l$ gets mapped to. We will denote probability matrices formed by the empirical measure in (5) as $\boldsymbol{\mathcal{E}}_B^{\mathcal{G}} = \left\{\mathcal{E}_B^{\mathcal{G}}(\boldsymbol{\sigma}^i, \boldsymbol{\sigma}^j)\right\}_{i,j=1,\ldots,|\Sigma|}$.

In the following, we will refer to matrices of the form $\mathbf{Q} = \{q(\boldsymbol{\sigma}^i, \boldsymbol{\sigma}^j)\}_{i,j=1,\ldots,|\Sigma|}$ as probability matrices with support $\Sigma \times \Sigma$. By design, the empirical measures of the form (5) are probability matrices with support $\Sigma \times \Sigma$. Each probability matrix, under the Markovian assumption, is associated with a transition probability matrix of the form $\{q(\boldsymbol{\sigma}^j|\boldsymbol{\sigma}^i)\}_{i,j=1,\ldots,|\Sigma|}$ where $q(\boldsymbol{\sigma}^j|\boldsymbol{\sigma}^i) = q(\boldsymbol{\sigma}^i, \boldsymbol{\sigma}^j)/q(\boldsymbol{\sigma}^i)$. Here $q(\boldsymbol{\sigma}^i) = \sum_{j=1}^{|\Sigma|} q(\boldsymbol{\sigma}^i, \boldsymbol{\sigma}^j)$ denotes the marginal probability of flow state $\boldsymbol{\sigma}^i$ in $\mathbf{Q}$.

Let $\boldsymbol{\Pi} = \{\pi(\boldsymbol{\sigma}^i, \boldsymbol{\sigma}^j)\}_{i,j=1,\ldots,|\Sigma|}$ denote, under the Markovian assumption, the true probability matrix of sequences of flows. As in the i.i.d. case, we compute $\boldsymbol{\Pi}$ via (5) from $\mathcal{G}_{ref}$. Following a similar procedure as in the i.i.d. case, we use an analog of the Sanov's Theorem for the Markovian case, which appears in [4], as the basis for our model-based stochastic anomaly detector. For every shift invariant probability matrix $\mathbf{Q}$ with support $\Sigma \times \Sigma$, let

$$H_B(\mathbf{Q}|\boldsymbol{\Pi}) = \sum_{i,j=1}^{|\Sigma|} q(\boldsymbol{\sigma}^i, \boldsymbol{\sigma}^j) \log \frac{q(\boldsymbol{\sigma}^j|\boldsymbol{\sigma}^i)}{\pi(\boldsymbol{\sigma}^j|\boldsymbol{\sigma}^i)}.$$

be the relative entropy of $\mathbf{Q}$ with respect to $\boldsymbol{\Pi}$. Then in the *model-based* method, the indicator of anomaly for $\mathcal{G}$ is:

$$\mathcal{I}_B(\mathcal{G}) = \mathbf{1}\{I_2(\boldsymbol{\mathcal{E}}_B^{\mathcal{G}}) \geq \eta\} \tag{6}$$

where $I_2(\boldsymbol{\mathcal{E}}_B^{\mathcal{G}}) = H_B(\boldsymbol{\mathcal{E}}_B^{\mathcal{G}}|\boldsymbol{\Pi})$ and $\eta = \frac{1}{n}\log\epsilon$ with ϵ be an allowable false alarm rate. Again, the *model-based* detector has been proved in [19] to be asymptotically Neyman-Pearson optimal.

B. 1-class SVM

We turn now to deterministic methods based on the construction of a decision boundary. We focus on one popular technique named 1-class SVM [16], [24]. The premise behind 1-class SVM is to find a hyperplane that separates the majority of the data $\mathcal{Z} = \{\mathbf{z}^1, \ldots, \mathbf{z}^{|\mathcal{Z}|}\}$ from the outliers by solving a Quadratic Programming (QP) Problem [19], [24]. The hyperplane can be generalized to a nonlinear boundary by mapping the inputs into high-dimensional spaces with a kernel function $K(\cdot,\cdot)$ [9]. There is a tunable parameter ν effectively tuning the number of outliers.

1) Flow 1-class SVM: We consider a set of flows $\mathcal{G} = \{\mathbf{g}^1, \ldots, \mathbf{g}^{|\mathcal{G}|}\}$ that need to be evaluated. According to (1), each flow has the format of $\mathbf{g} = (k(\mathbf{x}), d_a(\mathbf{x}), b, d_t)$, which has already provided a rather compact representation of network traffic. The only additional process required is to remove the label of the cluster each user belongs to. The new data are: $\mathcal{Z} = \{\mathbf{z}^1 = (d_a(\mathbf{x}^1), b^1, d_t^1), \ldots, \mathbf{z}^{|\mathcal{Z}|} = (d_a(\mathbf{x}^{|\mathcal{Z}|}), b^{|\mathcal{Z}|}, d_t^{|\mathcal{Z}|})$. The reasoning for this is that, since we are measuring departures from nominal users, the actual cluster a user belongs to is less important than the distance between the user the cluster center. Besides, as a categorical attribute, cluster labels make 1-class SVM method more unstable in practice. Besides, we choose the radial basis function $K(\mathbf{u}, \mathbf{v}) = \exp\left(-\gamma(\mathbf{u}-\mathbf{v})^T(\mathbf{u}-\mathbf{v})\right)$ as the kernel function [16].

2) Window 1-class SVM: We combine the techniques described in Section III-A and the 1-class SVM into a window-based 1-class SVM method. For each window j with flows $\mathcal{G}_j$, we can get the *model-free* empirical measure $\boldsymbol{\mathcal{E}}^{\mathcal{G}_j}$ and the *model-based* empirical measure $\boldsymbol{\mathcal{E}}_B^{\mathcal{G}_j}$. Let the feature vector for window j be $Y^j = \left\{\boldsymbol{\mathcal{E}}^{\mathcal{G}_j}, \boldsymbol{\mathcal{E}}_B^{\mathcal{G}_j}, |\mathcal{G}_j|\right\}$. Let $\mathcal{Y} = \{Y^1, \ldots, Y^{|\mathcal{Y}|}\}$ be a time series consisting of the features for all windows, then an 1-class SVM can be used to evaluate $\mathcal{Y}$, resulting in a window-based anomaly detector. Note that since the dimension of feature Y^j is usually very large, it often helps to apply Principal Component Analysis (PCA) [20] to reduce the dimensionality first.

C. ART Clustering

In this section, we present a clustering algorithm based on ART theory [3] and apply it to network anomaly detection. The algorithm first organizes inputs into clusters based on a customized distance metric. Then, a dynamic learning approach is used to update clusters or to create new clusters.

Assume a set of flows $\mathcal{F} = \{\mathbf{f}^1, \ldots, \mathbf{f}^{|\mathcal{F}|}\}$ with form in (2). Similar to the statistical methods, we define $\mathbf{g}^i = (n_f(\mathbf{x}), d_b(\mathbf{x}), b^i, d_t^i)$ to be the attributes in $\mathbf{f}^i$ without the start time t^i and $\mathcal{G}$ is the counterpart of $\mathcal{F}$. Suppose g^{ij} is the jth attribute of flow $\mathbf{g}^i$ for all $i = 1, \ldots, |\mathcal{G}|$ and $j = 1, 2, 3, 4$. Defining $f_{min}(j, \mathcal{G})$ and $f_{max}(j, \mathcal{G})$ to be the minimum and maximum of the set $\{g^{ij} : \forall i = 1, \ldots, |\mathcal{G}|\}$, we can normalize $\mathcal{G}$ according to $\hat{g}^{ij} = \frac{g^{ij} - f_{min}(j,\mathcal{G}))}{f_{max}(j,\mathcal{G}) - f_{min}(j,\mathcal{G})}$ for all $i = 1, \ldots, |\mathcal{G}|$, and $j = 1, 2, 3, 4$. In this section we assume that the data in $\mathcal{G}$ has already been normalized.

Define the distance metric

$$D(\mathbf{p}, \mathbf{q}) = \sum_{j=1}^{m} \left(\frac{p_j - q_j}{1 - v_j} \right)^2 \tag{7}$$

for two m-dimensional vectors $\mathbf{p} = (p_1, \ldots, p_m)$ and $\mathbf{q} = (q_1, \ldots, q_m)$, where $\mathbf{v} = (v_1, \ldots, v_m)$ is a set of parameters $v_j \in [0, 1)$ that controls the *vigilance* in dimension j. Let $\mathcal{T}_k$ be the set of all flows in cluster k. Letting $\mathbf{c}_k = (n_f^k, d^k, b^k, d_t^k)$ represent the center of cluster k and c_k^j be its j component. Let $\mathcal{C}$ be the set of all cluster centers. For every $\mathbf{c} \in \mathcal{C}$ and a prescribed r,

$$D(\mathbf{g}, \mathbf{c}) \leq r, \mathbf{g} \in \mathbb{R}^m \tag{8}$$

defines an ellipsoid in $\mathbb{R}^m$. A higher vigilance in one dimension means the ellipsoid is more shallow in this direction.

The *ART clustering* algorithm is shown in Algorithm 1. Initially $\mathcal{C}$ is empty. For each flow $\mathbf{g}^i \in \mathcal{G}$, we calculate the set $\mathcal{D}$ which consists of all clusters whose ellipsoid defined by (8) contains $\mathbf{g}^i$. Suppose $E(\mathbf{g}, \mathbf{c})$ is the Euclidean distance between $\mathbf{g}$ and $\mathbf{c}$. If $\mathcal{D}$ is not empty, $\mathbf{g}^i$ is assigned to the cluster whose center has the smallest Euclidean distance with $\mathbf{g}^i$ and the corresponding cluster center is updated; otherwise a new cluster is created. Suppose that flow $\mathbf{g}^i$ will be assigned to cluster k, let $c_k^{j'}$ and c_k^j be the jth component of the center of cluster k before and after the assignment, then

$$c_k^{j'} = (p \times c_k^j + g^{ij})/(p + 1), \forall j = 1, \ldots, m, \tag{9}$$

where p is the number of flows in cluster k before the assignment. Because of the adaptive update (9), some assignments may become unreasonable after update as some flows may become closer to other cluster centers. As a result, the algorithm processes flows in $\mathcal{G}$ again until an equilibrium is reached.

Once a stable equilibrium is reached, small outlying clusters are identified as anomalous based on the rule

$$\mathcal{I}_A(\mathcal{T}_k) = \mathbf{1}\{|\mathcal{T}_k| < \tau \times |\mathcal{G}|/|\mathcal{C}|\} \tag{10}$$

where $\mathcal{I}_A(\mathcal{T}_k)$ is an indicator of anomaly for $\mathcal{T}_k$, $\tau \in [0, 1]$ is a prescribed detection threshold, $|\mathcal{C}|$ and $|\mathcal{G}|$ are the total number of clusters and flows, respectively. τ determines how small a cluster must be to be considered as anomalous, thus it influences the number of alarms. We will discuss the relationship of τ and the false alarm rate further in Section V.

Algorithm 1 ART clustering Algorithm

Require: Flow Data $\mathcal{G} = \{\mathbf{g}^1, \ldots, \mathbf{g}^{|\mathcal{G}|}\}$
  $\mathcal{C} = \mathcal{C}_{last} = \{\}$
  while $\mathcal{C} \neq \mathcal{C}_{last}$ **do**
    $\mathcal{C}_{last} = \mathcal{C}$
    for $i = 1 \rightarrow |\mathcal{G}|$ **do**
      $\mathcal{D} = \{\mathbf{c} \in \mathcal{C} : D(\mathbf{g}^i, \mathbf{c})) < r\}$
      if $|\mathcal{D}| = 0$ **then**
        $\mathcal{C} = \{\mathcal{C}, \mathbf{g}^i\}$
      else
        $\mathbf{c}_o = \arg\min_{\mathbf{c} \in \mathcal{C}} E(\mathbf{g}^i, \mathbf{c})$
        $\mathcal{T}_k = \{\mathcal{T}_k, \mathbf{g}^i\}$, k is the index of $\mathbf{c}_o$ in $\mathcal{C}$.
        Recalculate cluster center of $\mathcal{T}_k$ using (9)
      end if
    end for
  end while

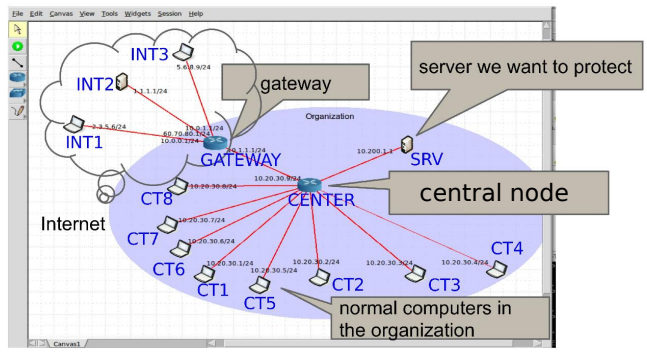

Fig. 2. Simulation setting.

IV. Network Simulation

The lack of annotated data is a common problem in the network anomaly detection community. As a result, we developed two open source software packages to provide flow-level and packet-level validation datasets, respectively. SADIT [28] is a software package containing all the algorithms we described above. It also provides an annotated flow record generator powered by the *fs* [26] simulator. IMALSE [27] uses the NS3 simulator [10] for the network simulation and generates packet-level annotated data. Simulation at the packet-level takes more computation resources but can mimic certain attacks, like botnet-based attacks, in a more realistic way. We validate our algorithms with the help of these two software packages. The packets generated by IMALSE, which is of *pcap* format [27], are transformed into flow records first. Then the flows generated by SADIT and IMASLE are tested independently with each algorithm.

The simulated network is partitioned into an internal network with a hub and spoke topology that connects to the Internet via a gateway (Fig. 2). The internal network consists of 8 normal users (*CT1-CT8*) and 1 server (*SRV*) with some sensitive information. We monitor the traffic on the server.

A. Flow-level Anomalies

First, we generate a dataset with flow-level anomalies. The size and the transmission of the nominal flows for user i is assumed to follow a Gaussian distribution $N(m_i, \sigma_i^2)$ and Poisson process with arrival rate λ_i, respectively. We investigate three most common types of flow-level anomalies.

The first one mimics the scenario according to which a network intruder or unauthorized user downloads restricted data. A previously unseen user who has a large IP distance to the rest of the users starts transmission for a short period. The second one is a user i with suspicious flow size distribution characterized by a mean m_i^a higher than a typical value m_i. Usually flows with substantially large flow size are associated with the situation when some users try to download large files from the server, which can happen when the attacker tries to download the sensitive information packed into a large file. The last one is a user increasing its flow transmission rate to an unusual value λ_i^a, which could be indicative of the user finding an important directory on the server and downloading, repeatedly, sensitive files within that directory.

B. Packet-level Anomalies

A second anomalous dataset is created using the tool IMALSE [27]. The nominal traffic is generated using the *on-off* application in NS3 [10], [27] in which the user sends packets for t_{on} seconds and the interval between two consecutive transmission is t_{off}. The traffic is a Poisson process, which means the *on* time and *off* times are exponentially distributed with parameter λ_{on} and λ_{off}, respectively.

We assume there is a botnet in the network. There is a botmaster controlling the bot network and a Command and Control (C&C) server issuing control commands to the bots. In our simulation, both the botmaster and C&C server are the machine *INT2* in the Internet, and *CT1-CT5* in the internal network have been infected as bots. We investigate a DDoS Ping flood attack in which each bot sends a lot of ping packets to the server *SRV* upon the request of the C&C server, aiming to exhaust the bandwidth of *SRV*. The attack is simulated at the packet-level and the data are then transformed into flow records using techniques described in Section II. With appropriate δ_F, the t_{on} becomes the flow duration of nominal flows and the t_{off} determines the flow transmission rate of nominal flows. The initiating stage of the attack is similar to the first case in the previous section. During the attack, both the flow transmission rate and the flow size of the bots may be affected. First, the flow transmission rate is increased as the bots ping *SRV* more frequently. Second, the ping packets have different sizes from normal network traffic. Also, consecutive ping packets may be combined together if they are sent over a short time interval. The resulting flows may be very large in size if these combinations are common or very small otherwise, depending on the attack pattern.

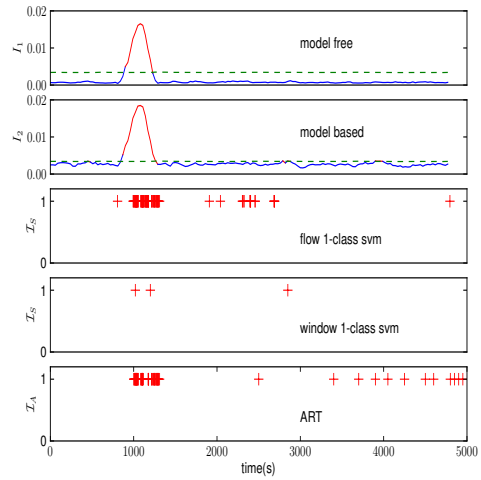

Fig. 3. The results of five methods in the atypical user case.

V. Results

A. Flow-level Anomalies

1) Atypical User: Figure 3 shows the response of all methods described above when there is an atypical user trying to access the server between 1000s and 1300s. For window-based methods, the interval between the starting point of two consecutive time windows is $h = 30s$ and the window size is chosen as $w_s = 200s$, so there is overlap between two consecutive time windows. We also distill the user space by using K-Means clustering with 3 clusters. The quantization levels for flow size, distance to cluster and flow duration are 3, 2, 1, respectively, thus $|\Sigma| = 18$. The x-axis in all graphs corresponds to time (s) and the total simulation time is 5000s. The first two graphs depict the entropy metric in (4) and (6) of the *model-free* and *model-based* methods, respectively. For both graphs, the green dashed line is the threshold when the false alarm rate is $\epsilon = 0.01$. The interval during which the entropy curve is above the threshold line (the red part) is the interval the method reports as abnormal. The x coordinates of the red points with a '+' marker correspond to the start point of the flow or the window the method reports as abnormal. The parameter ν for the *flow 1-class SVM* and *window 1-class SVM* is 0.002 and 0.1, respectively. The threshold τ for *ART clustering* is 0.05.

We can observe from Figure 3 that stochastic methods, including our *model-free* and *model-based* methods, tend to produce more stable results in the sense that they generate fewer false alarms. At the same time, the *flow 1-class SVM* and *ART clustering* methods, both of which are flow-based, can provide higher identification resolution in the sense that they can identify the suspicious flows, which is beyond the capabilities of the stochastic methods. In the *window 1-class SVM* method, we can tune the window size to adjust the tradeoff of resolution and stability. However, the window size in the *model-free* and *model-based* methods has to be reasonably large since the optimality of the decision rule (4) and (6) relies on the assumption of a large flow number in each window.

This observation indicates that these methods are complementary to each other. One way to combine them is

to use stochastic methods and window-based deterministic methods to get a rough interval of an anomaly. Then, only the flows that are both identified as suspicious by flow-based deterministic methods and belong to the interval need to be further evaluated. The first subfigure in Figure 4 shows the Receiver Operating Characteristic (ROC) curve of the *ART clustering* method, which is a flow-based method, and the combination of the *ART clustering* and the *model-free* method. The ROC curve has been substantially improved after combining the two methods. The second subfigure in Figure 4 shows the relationship between the threshold τ defined in (10) and the false alarm rate. The x-axis is the false alarm rate and y-axis corresponds to the threshold. As we can see, the false alarm rate increases when the threshold increases and they are almost linearly related to each other.

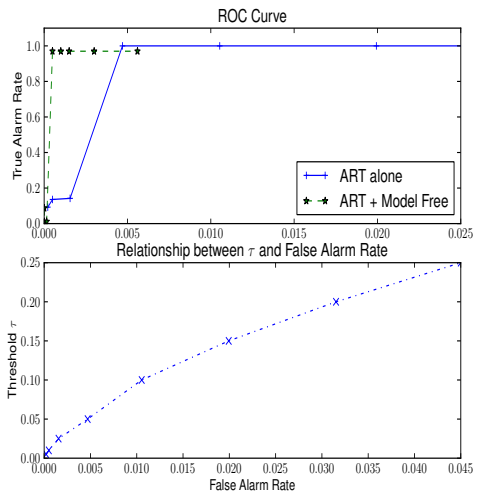

Fig. 4. ROC curve and relationship of τ and the false alarm rate for ART.

2) Large File Download: Figure 5 is the output of all methods in the case where a user doubles its mean flow size between 1000s and 1300s. Again, the first two graphs show the entropy curve and threshold line of the *model-free* and *model-based* methods. The total simulation time is 5000s. The common window parameters h and w_s are the same as in the previous case. The false alarm rate is $\epsilon = 0.01$ for both *model-free* and *model-based* methods. The parameter ν for *flow 1-class SVM* and *window 1-class SVM* is 0.0015 and 0.1, respectively. $\tau = 0.01$ for *ART clustering*.

3) Large Access Rate: Figure 6 shows the response of *model-free*, *model-based*, *window 1-class SVM* and *ART clustering* methods when a user suspiciously increases its access rate to 6 times of its normal value during 1000s and 1300s. The total simulation time is 2000s. The parameters for the algorithms are the same in the atypical user case.

Note that *flow 1-class SVM* cannot work for this type of anomaly since it is purely temporal-based. The flow itself does not change but its frequency does. There is no way to identify the frequency change by just observing the individual flows with representation in (1). *ART clustering* works fairly well for this case because the attacker will have larger $n_f(\mathbf{x})$ as it transmits more flows. Interestingly, the *model-based* and *model-free* methods can work very well

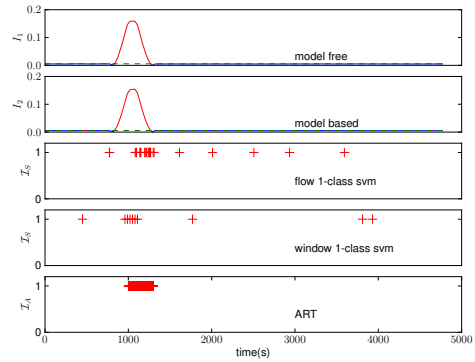

Fig. 5. The results of five methods in the large file download case.

since the portion of traffic originating from the attacker changes, influencing the empirical measure defined in (3) and (5). The two methods will not be effective in the very rare case when all users increase their rate by the same ratio synchronously.

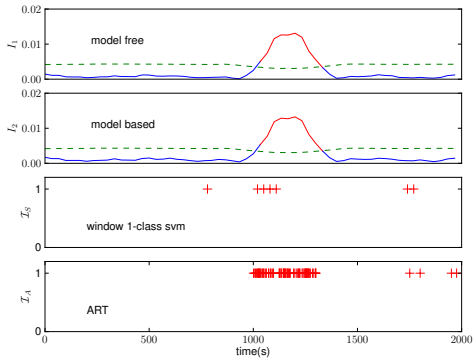

Fig. 6. The results of five methods in the large access rate case.

B. DDoS Attack

Figure 7 shows the response of *model-free*, *model-based*, *window 1-class SVM* and *flow 1-class SVM* methods when there is a DDoS attack targeting *SRV* between 500s and 600s. The total simulation time is 900s. For window-based methods, the interval between consecutive time windows is $h = 10s$ and the window size is $w_s = 100s$. The false alarm rate for the *model-free* and *model-based* method is $\epsilon = 0.01$ and $\nu = 0.05$ for *window SVM*.

Since the nominal traffic in IMALSE is generated based on an i.i.d assumption, it is hard for the *model-based* method to capture a Markov model. Yet, the *model-based* method still detects the start and the end of the attack, during which the transitional behavior changes the most. *Model-free* and *window 1-class SVM* are more stable while the *flow 1-class SVM* method provides higher resolution.

The *ART clustering* method is also not suited to detect these type of attacks because the unsupervised learning model is based on the assumption that malicious network traffic represents a small percentage of total network traffic. A DDoS attack generates a large number of packets and without some prior knowledge of good or bad network traffic,

the *ART clustering* algorithm cannot distinguish between the nominal and abnormal flows. It is also the reason for the relatively unsatisfactory performance of the *flow 1-class SVM* method. However, *window 1-class SVM* is not affected by this because despite the large number of abnormal flows, the number of abnormal windows is still very small.

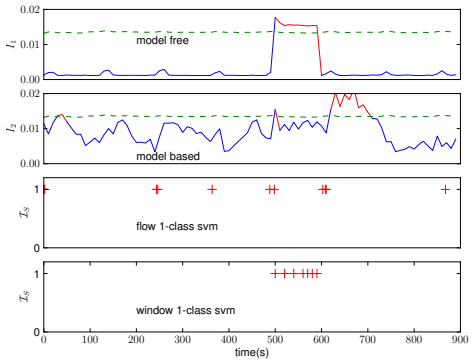

Fig. 7. The results for DDoS attack.

VI. Conclusion

We presented five complementary approaches, based on SHT, SVM and clustering, that cover the common techniques for host-based network anomaly detection. We developed two open source software packages to provide flow-level and packet-level validation datasets, respectively. With the help of these software packages, we evaluated all methods on a simulated network mimicking typical networks in organizations. We consider three flow-level anomalies and one packet-level DDoS attack.

Through analyzing the results, we summarize the advantages and disadvantages of each method. In general, deterministic and flow-based methods, such as *flow 1-class SVM* and *ART clustering*, are more likely to have unstable results with higher false alarm rates but they can identify abnormal flows, namely they have better resolution. Stochastic and window-based methods, such as our *model-free* and *model-based* methods, could yield more stable results and detect temporal anomalies better, but they have relatively poor resolution as they are not able to explicitly detect the anomalous network flows. In addition, deterministic and window-based methods, like *window 1-class SVM* offer parameters to adjust the tradeoff of resolution and stability. This observation suggests that combining the results of all, instead of just using one method, can yield better overall performance.

References

[1] M. R. Anderberg. Cluster analysis for applications. Technical report, DTIC Document, 1973.

[2] G. Androulidakis and S. Papavassiliou. Improving network anomaly detection via selective flow-based sampling. *Communications, IET*, 2(3):399–409, 2008.

[3] G. A. Carpenter and S. Grossberg. ART 2: self-organization of stable category recognition codes for analog input patterns. *Applied Optics*, 26(23):4919–4930, 1987.

[4] A. Dembo and O. Zeitouni. *Large deviations techniques and applications*, volume 38. Springer, 2009.

[5] A. B. Frakt, W. C. Karl, and A. S. Willsky. A multiscale hypothesis testing approach to anomaly detection and localization from noisy tomographic data. *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society*, 7(6):825–37, Jan. 1998.

[6] G. Gu, R. Perdisci, J. Zhang, W. Lee, et al. Botminer: clustering analysis of network traffic for protocol-and structure-independent botnet detection. In *Proceedings of the 17th conference on Security symposium*, pages 139–154, 2008.

[7] I. Hareesh, S. Prasanna, M. Vijayalakshmi, and S. M. Shalinie. Anomaly detection system based on analysis of packet header and payload histograms. In *Recent Trends in Information Technology (ICRTIT), 2011 International Conference on*, pages 412–416. IEEE, 2011.

[8] J. A. Hartigan and M. A. Wong. Algorithm AS 136: A K-Means Clustering Algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):pp. 100–108, 1979.

[9] T. Hastie, R. Tibshirani, J. Friedman, et al. The elements of statistical learning: data mining, inference, and prediction, 2001.

[10] T. R. Henderson, M. Lacage, G. F. Riley, C. Dowell, and J. B. Kopena. Network simulations with the ns-3 simulator. *SIGCOMM demonstration*, 2008.

[11] A. Lakhina, M. Crovella, and C. Diot. Mining anomalies using traffic feature distributions. In *ACM SIGCOMM Computer Communication Review*, volume 35, pages 217–228. ACM, 2005.

[12] W. Lee. Information-theoretic measures for anomaly detection. *Proceedings 2001 IEEE Symposium on Security and Privacy. S&P 2001*, pages 130–143, 2001.

[13] E. L. Lehmann and J. P. Romano. *Testing statistical hypotheses*. Springer, 2005.

[14] R. Lippmann, J. Haines, D. Fried, J. Korba, and K. Das. The 1999 DARPA off-line intrusion detection evaluation. *Computer networks*, 34, 2000.

[15] S. P. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28:129–137, 1982.

[16] R. Locke, J. Wang, and I. Paschalidis. Anomaly detection techniques for data exfiltration attempts. Technical Report 2012-JA-0001, Center for Information & Systems Engineering, Boston University, 8 Saint Mary's Street, Brookline, MA, June 2012.

[17] M. V. Mahoney and P. K. Chan. PHAD : Packet Header Anomaly Detection for Identifying Hostile Network Traffic. (1998):1–17, 2001.

[18] C. Manikopoulo. Flow-based Statistical Aggregation Schemes for Network Anomaly Detection. *2006 IEEE International Conference on Networking, Sensing and Control*, pages 786–791, 2006.

[19] I. Paschalidis and G. Smaragdakis. Spatio-temporal network anomaly detection by assessing deviations of empirical measures. *Networking, IEEE/ACM ...*, 2009.

[20] K. Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

[21] R. Perdisci, G. Gu, and W. Lee. Using an Ensemble of One-Class SVM Classifiers to Harden Payload-based Anomaly Detection Systems. *Sixth International Conference on Data Mining (ICDM'06)*, pages 488–498, Dec. 2006.

[22] I. Perona, I. Albisua, and O. Arbelaitz. Histogram based payload processing for unsupervised anomaly detection systems in network intrusion. *Proc. of the 14th ...*, 2010.

[23] D. Rossell. An ART Network Anomaly Detection Tool. `http://people.bu.edu/drossell/network.html`, 2012.

[24] B. Schölkopf, J. C. Platt, J. C. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Estimating the Support of a High-Dimensional Distribution, July 2001.

[25] T. Shon and J. Moon. A hybrid machine learning approach to network anomaly detection. *Information Sciences*, 177(18):3799–3821, 2007.

[26] J. Sommers, R. Bowden, B. Eriksson, P. Barford, M. Roughan, and N. Duffield. Efficient network-wide flow record generation, 2011.

[27] J. Wang. IMALSE: Integrated MALware Simulator and Emulator. `http://people.bu.edu/wangjing/open-source/imalse/html/index.html`, 2012.

[28] J. Wang. SADIT: Systematic Anomaly Detection of Internet Traffic. `http://people.bu.edu/wangjing/open-source/sadit/html/index.html`, 2012.

[29] X. Zhang, Z. Zhu, and P. Fan. Intrusion detection based on the second-order stochastic model. *Journal of Electronics (China)*, 24(5):679–685, Sept. 2007.