Automatic Construction of Anomaly Detectors from Graphical Models

Erik M. Ferragut
David M. Darmon
Craig A. Shue
Stephen Kelley
Cyberspace Sciences and Information Intelligence Research Group
Oak Ridge National Laboratory
{ferragutem, darmondm, shueca, kelleys}@ornl.gov

***Abstract*—Detection of rare or previously unseen attacks in cyber security presents a central challenge: how does one search for a sufficiently wide variety of types of anomalies and yet allow the process to scale to increasingly complex data? In particular, creating each anomaly detector manually and training each one separately presents untenable strains on both human and computer resources. In this paper we propose a systematic method for constructing a potentially very large number of complementary anomaly detectors from a single probabilistic model of the data. Only one model needs to be trained, but numerous detectors can then be implemented. This approach promises to scale better than manual methods to the complex heterogeneity of real-life data. As an example, we develop a Latent Dirichlet Allocation probability model of TCP connections entering Oak Ridge National Laboratory. We show that several detectors can be automatically constructed from the model and will provide anomaly detection at flow, sub-flow, and host (both server and client) levels. This demonstrates how the fundamental connection between anomaly detection and probabilistic modeling can be exploited to develop more robust operational solutions.**

I. Introduction

In recent years, much of society has become heavily dependent on cyberinfrastructure. An important issue raised by the use of this infrastructure is the threat of cyberattacks. These attacks can take on many forms: denial-of-service, man-in-the-middle, etc. In order to detect these attacks, one can assume that the behavior of malicious individuals is likely to be anomalous. An anomaly detector is an algorithm that takes in streaming data and flags the most unusual events as anomalous and tags them with a score indicating the degree of anomalousness. The anomaly detection approach is based on the hypothesis that atypical events are more likely to be of interest [1]. The concept of operations for anomaly detectors is typically that human operators will manually analyze the most anomalous events as time allows.

One issue with anomaly detection is that the notion of anomalousness is not generally well-defined. For example, should the system flag as anomalous a daily event that almost always occurs at 7:15 AM or 7:17 AM, but one day occurs at 7:16 AM? If the event time is thought of as a unimodal random variable (e.g., a normally distributed random variable), then 7:16 AM event, even though it had not previously been observed, would be rated *more* typical than 7:15 AM or 7:17 AM. On the other hand, if the times were treated as categorical variables, then the previously unobserved event would likely be flagged as anomalous. Ideally, the anomaly detection system will be guided by domain experts so that the more appropriate modeling approach is used.

A second issue with anomaly detection is the need to test for multiple types of anomalies. If the data are only comprised of event times, the number of reasonable questions is limited. However, real-life data are, more often than not, highly complex and heterogeneous. Given a set of variables, an anomaly detection question can be asked for each variable conditional on every subset of the other variables. For example, if the data are comprised of `EVENT TIME`, `LOCATION`, `EVENT LENGTH`, and `EVENT TYPE`, then one can ask (for example):

1) Is the `EVENT TIME` anomalous given the `LOCATION`?
2) Is the `EVENT TIME` anomalous given the `EVENT LENGTH`?
3) Is the `LOCATION` anomalous given the `EVENT TIME` and `EVENT TYPE`?
4) Is the `EVENT LENGTH` anomalous given all other variables?

Clearly a set of n variables will admit $n2^{n-1}$ such anomaly detection questions. This explosion of anomaly detectors, while apparently necessary for completeness has three main issues. First, training and applying all of these detectors is likely to be computationally intractable. Second, it is unlikely that there will be enough data to justify the complexity of these models. Third, even if computational tractability and data requirements are met, a very small false positive rate p will be required in order to keep the expected number of false positives, $pn2^{n-1}$ small enough to avoid overwhelming analysts.

In this paper, we show that mutiple appropriately defined anomaly detectors can be constructed from a corresponding well-designed graphical model of the data so that the detectors exhibit good coverage of anomalous events of interest

while minimizing computational requirements. In particular, only one graphical model needs to be trained, but an entire collection of anomaly detectors may then be derived.

Anomaly detectors can be constructed based on various methods of computational intelligence, including statistical analysis, machine learning, and probabilistic modeling. Statistical methods provide useful initial exploratory analyses. However, they are often difficult to apply to complex structured data, such as observations on a network. Machine learning offers a 'black box' answer: a model (known as a classifier) is trained on a data set and then future data can be classified using this model as either normal or anomalous. This approach, while effective, offers little intuition into the mechanism behind the data. In addition, machine learning methods, while general, most typically require the conversion of the data into numeric vectors, which obscures the structural properties of the data and forces the machine learning algorithm to rediscover it.

Probability models explicitly represent the relationships between variables of interest to produce a model, learn from data, and perform inference in the presence of uncertainty [2]. As a result, probability models explicitly incorporate the known structural information about the data and allow for modeling of complex structured data. Another advantage of this approach is that it offers insight into the mechanisms behind the generation of the data. It also separates the representation of the model structure from any learning or inference performed on the model, which allows structural questions (e.g., presence of dependencies) to be tested. Very importantly for real world applications, probabilistic modeling does not require data that has already been classified and thus is suitable for unsupervised learning.

In this paper we use the explicit relationships between variables as represented in a graphical model to guide the automated construction of anomaly detectors. Generally, the anomaly detectors will compute the individual conditional probabilities associated with a data record or collection (depending on the detector). The advantage of having a constellation of such detectors is that an event found to be anomalous can be further annotated in terms of which detector discovered its anomalousness. This provides a rich contextual background for human analysts by describing the cause of the anomaly, giving examples of what would have been more typical, and providing a probability score. Contrast this with a black-box algorithm where an event is either labeled as anomalous or normal without further explanation or context.

While previous work [3]–[8] has used probability modeling as the basis for anomaly detection in cyber security and other domains, this is the first paper to describe a general methodology for doing so. In many cases, previous probability models can be used to extend the completeness of their anomaly detection solutions essentially for free in the sense that no new training is required, but additional anomaly detectors can be constructed from the model as trained. Also, the particular usage of Latent Dirichlet Allocation (LDA) used in this work is new, although the application of LDA to cyber data analysis in other ways is not [9].

Section II provides a brief introduction to graphical models and introduces the important example of Latent Dirichlet Allocation, which will be the central example of this paper. Section III describes our methodology for constructing anomaly detectors from a graphical model in a complete and systematic way to produce anomaly detectors of three kinds. Section IV describes the application of LDA to real ORNL cyber data and enumerates the systematically constructed anomaly detectors in the LDA example. Section V shows the results of a selection of the resulting detectors.

II. Graphical Models and Latent Dirichlet Allocation

A graphical model[1] is a probability model that represents dependencies among random variables by the directed edges within a graph. In particular, vertices of the graph represent random variables. A variable x is directly dependent on the variable y if the graph contains the edge $x \rightarrow y$. The conditional probability distributions for each variable given the variables on which it directly depends is assumed to be known. If a variable does not directly depend on any other variables, then the conditional probability is in fact a marginal probability.

The graph of variables and dependencies together with the conditional distribution families of the variables comprise the structure of the graphical model. This structure is generally manually determined and then fixed. The parameterizations of the conditional distributions are learned from data. The parameters inform the knowledge discovery. A trained graphical model can be used to compute the probabilities of new observations, which is the key to anomaly detection. This is described in the next section.

Graphical models generalize many well known machine learning algorithms including hidden Markov models, naive Bayes, Bayesian belief networks, and conditional random fields. They have found successes and represent some of the best known solutions in voice processing, computer vision, natural language processing, and bioinformatics.

The remainder of this section describes a particular graphical model developed in the context of natural language processing. We will be applying (assigned new meaning to its variables, training it, and using it for inference) this model to the analysis of cyber data and to anomaly detection within that data.

In natural language processing, document classification and topic model training [10] have often employed probabilistic modeling approaches. One such model is Latent Dirichlet Allocation (LDA) [11]. LDA is a generative probabilistic model for a set of observations explained by unobserved topics. LDA can be used for and was originally developed to identify latent (i.e., unobserved) topics within a corpus of text documents based on the distribution of words within and across those documents.

A recent application of LDA to cyber security focused on text-mining within blogs [12]. Blog posts were considered as mixtures of topics, some of which were of security interest. This work successfully identified interesting topic areas within

[1]For simplicity of exposition, we discuss only directed graphical models.

the blog posts. However, the authors focused on blogs known to focus on cyber security threats. Thus, this work does not fall under the umbrella of anomaly detection. Recent work applied LDA to identifying exfiltration events in artificial network logs generated for a cyber security challenge [9]. LDA was applied to individual network connections, treating each connection as a document composed of several categorical attributes (the 'words' of the model). The application of LDA to the network events identified all exfiltration events present in the challenge.

A. Latent Dirichlet Allocation

In explaining Latent Dirichlet Allocation (LDA), we adopt the language of text analysis as it is the most natural terminology for describing LDA.

LDA is a probabilistic model for multinomial discrete data [11]. One of the keys to derivation of LDA is the "bag-of-words" assumption: we assume that the order of the words within a document can be neglected. This is the assumption of exchangeability of the words within a document. The exchangeability assumption makes the words independent and identically distributed, conditional on the parameters of the model. Using this independence, we create the model depicted in Figure 1. The model is represented using plate notation, a way of representing variables that repeat within a graphical model. Using plate notation, a rectangle is used to designate variables into subgraphs that repeat together, and the number within the rectangle designates the number of copies of the subgraph. Therefore, in Figure 1, there are M documents, N words within each document, and k topics across all the documents. The method for choosing the words for a document within a corpus is encoded in this figure. For each document d, the multinomial distribution of topics $\boldsymbol{\theta}$ is sampled from the Dirichlet prior $\boldsymbol{\alpha}$. Then a topic z is assigned to each of the N words within a document by sampling from $\boldsymbol{\theta}$. Based on the topic z, for each word w within a document, a particular instance word is chosen from a vocabulary of V words from the Dirichlet prior $\boldsymbol{\beta}$. With a large vocabulary size, there is the possibility that, given a new document not in a corpus used for training the model, words will appear that did not appear in the training corpus. This would result in assigning zero probability to that document. Thus $\boldsymbol{\beta}$ is smoothed using $\boldsymbol{\eta}$, conditioned on the data. This is done by drawing each row of $\boldsymbol{\beta}$ from $\boldsymbol{\eta}$, an exchangeable Dirichlet distribution.

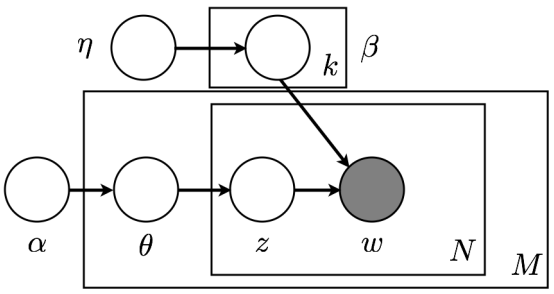

Fig. 1. A graphical model of the smoothed LDA.

B. Variational Approximation Algorithm

In order to make learning and inference using LDA feasible, we must deal with the posterior distribution $P(\boldsymbol{\theta}, \mathbf{z}|\mathbf{w}, \boldsymbol{\alpha}, \boldsymbol{\beta})$, but this distribution is intractable to compute. There are many approximate inference algorithms that may be used for LDA, which include Markov Chain Monte Carlo methods such as Gibbs sampling [13], [14] and variational algorithms [15]–[17]. Here, we describe the latter in more detail; its trained parameters will be used for anomaly detectors below.

The convexity-based variational algorithm works by using Jensen's inequality to determine an adjustable lower bound on the log likelihood of the model. The lower bound is found by dropping the edges between $\boldsymbol{\theta}$ and $\mathbf{z}$, removing the node $\mathbf{w}$ and adding free variational parameters to the model, as in Figure 2. This gives a simplified approximating distribution $Q(\boldsymbol{\theta}, \mathbf{z}|\boldsymbol{\gamma}, \boldsymbol{\phi})$ with variational parameters $\boldsymbol{\gamma}$ and $\boldsymbol{\phi}$. Now tightening the lower bound on the log likelihood corresponds to the optimization problem

$$(\boldsymbol{\gamma}^*, \boldsymbol{\phi}^*) = \underset{(\boldsymbol{\gamma}, \boldsymbol{\phi})}{\operatorname{argmin}} \, \mathrm{D}(Q(\boldsymbol{\theta}, \mathbf{z}|\boldsymbol{\gamma}, \boldsymbol{\phi}) || P(\boldsymbol{\theta}, \mathbf{z}|\mathbf{w}, \boldsymbol{\alpha}, \boldsymbol{\beta}))$$

where D is the Kullback-Leibler divergence between the variational distribution and the true posterior distribution.

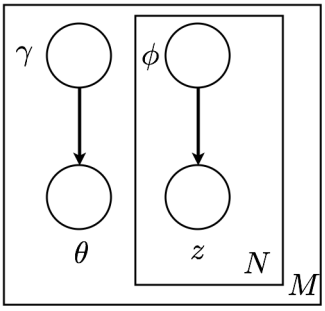

Fig. 2. A graphical model of the variational distribution used to approximate LDA.

III. Construction of Anomaly Detectors

In this section, we assume that we have a graphical model and a mapping from our data set features to the variables within the model. We identify three natural classes of anomaly detectors based, respectively, on dependency, plate, and index schemes.

Dependency. For every variable, we can consider its conditional distribution based on the variables on which it directly depends. These conditional distributions are precisely what was learned from the data. As a result, these distributions can easily be evaluated on new data. There will be one such anomaly detector for every variable represented by the new data record.

Plate. The plates in the graphical model represent replications of random variables of the same kind. This suggests the comparison of one such variable to all others of the same kind. These anomaly detectors require the ability to perform this comparison. If the parameter average of the distributions is computable and meaningful, then one might

consider a distribution similarity or distance measure between the excluded distribution and the average of the others. (We will use the Kullback-Liebler Divergence for this below.)

Index. In some cases, replicated plates, rather than being indexed by a single integer are more naturally multivariate. If a plate is indexed by $i_1, i_2, \ldots, i_k$, we can fix all indices but, say, i_j for some $j = 1, 2, \ldots, k$. Then we can compare the distribution for $i_j = t$ against the average across all $i_j \neq t$. This is a refinement of the Plate anomaly detector[2].

In each case, the required information for the anomaly detector is either directly in the trained graphical model or is easily computed from it (as, say, an average). Hence, there is no additional computation required. This means that a potentially large number of anomaly detectors can be derived from a single graphical model. In practice, many of these will be uninteresting to the operational analysts.

However, the probabilities produced by these anomaly detectors in the null hypothesis case that the data fit the model will vary considerably potentially making it difficult to compare across detectors. This is an issue when a large number of detectors are used in parallel. One way to address this issue is via synthesized thresholds where we use the graphical model to synthesize a large number of events and compute the anomaly detector scores for each. From these synthesized data, we can estimate thresholds for a pre-decided false positive rate. Each anomaly detector receives its own threshold and the resulting flagged anomalies are then at least filtered to meet the same minimal requirement of anomalousness.

IV. Methodology

This section describes the collection and preprocessing of the data, as well as the specifics of how LDA was used to model and learn from the data.

A. Data Collection and Preprocessing

Internet Protocol (IP) addresses and port information were collected for all connections crossing the Oak Ridge National Laboratory (ORNL) network perimeter. The logs contained a single record for every Transmission Control Protocol (TCP) connection and a record for each User Datagram Protocol (UDP) and Internet Control Message Protocol (ICMP) packet. The data were parsed to obtain TCP packets with SYN flags. These packets indicate the beginning of a directed communication between a client and server.

Further preprocessing was performed on the TCP packets. The 'to' port for each packet was classified as follows: a port below 1024 was labeled with the port number if the port was present in the Internet Assigned Number Authority (IANA) directory and as unknown otherwise[3]. All ports greater than 1024 were labeled unknown. The data were then partitioned into from-to pairs with the 'to' IP always within ORNL.

The data used during this study was collected from July 18 to July 23, 2010 and corresponds to approximately 74 million IP to-from pairs (approximately 3 million of which are unique) and 193 unique port labels.

B. Application of Latent Dirichlet Allocation

In analogy to text analysis, each IP address pair corresponds to a document, each 'to' port label corresponds to a word, and each behavior corresponds to a topic. The vocabulary of the corpus is the unique ports used by IP pairs. A document is composed of the count of ports used by a given IP address pair, resulting in a typical behavioral profile for each from-to IP address pair.

A variational expectation-maximization (EM) algorithm was used to learn the posterior parameters $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ of the LDA model. The model was trained using both five and ten topics. For all runs, an initial value of $\alpha_i = 0.1$ was used for the Dirichlet prior. The distribution of $\boldsymbol{\beta}$ was smoothed using k randomly selected documents. The EM algorithm was run with a convergence criterion of 1×10^{-6}. The total variational algorithm was run with a convergence criterion on the change in the log likelihood of the model of 1×10^{-6}.

C. Enumeration of Constructed Anomaly Detectors

We consider the three classes of natural anomaly detectors described in III. Detectors 1–4 are dependency-based, detectors 5–7 are plate-based, and detectors 8–9 are index-based. Below, we use the notation $M(X\ X_i)$ to denote the "mean" of the distributions of the variables X_j with $j \neq i$. Here, mean refers to a properly weighted average of the parameters of the distributions, if averaging is meaningful. Kullback-Liebler divergence is denoted by D.

1) Which words are least (and most) likely given the topic and word distributions? $P(w \mid z, \boldsymbol{\beta})$
2) Which word distributions are significantly different from the prior distribution? (These are represent significant deviations implied by the data and may be thought of especially cogent word distributions or likely meaningful topics.) $P(\boldsymbol{\beta} \mid \boldsymbol{\eta})$
3) Which topics are especially atypical within a document? $P(z \mid \boldsymbol{\theta})$
4) Which documents have topic distributions that are signficantly different from the prior distribution? $P(\boldsymbol{\theta} \mid \boldsymbol{\alpha})$
5) Which topics have the most distinctive word choice (given $\boldsymbol{\eta}$, which is common across word distributions)? $\mathrm{D}(\beta_i || M(\boldsymbol{\beta}\ \beta_i))$
6) Which document topic distributions are the most distinctive (given the topic prior $\boldsymbol{\alpha}$, which is common across documents)? $\mathrm{D}(\theta_i || M(\boldsymbol{\theta}\ \theta_i))$
7) Which word topics are most atypical within a document (given the topic distribution, $\boldsymbol{\theta}$, for that document)? $\mathrm{D}(z_i || M(z\ z_i))$
8) Given a destination IP address, which source IP addresses exhibit the most atypical behavior?
9) Given a source IP address, which destination IP addresses exhibit the most atypical behavior?

The most and least frequent words per topic, as described in detector 1, could be used to identify words (which in our cyber

[2]We could in fact consider allowing more than one index to vary, comparing one value against the average of all others. This produces exponentially many anomaly detectors in the dimensionality of the indexing scheme.

[3]See http://www.iana.org/assignments/port-numbers for a list of well-known ports.

data example are ports) as being anomalous within topics. Since topics are assigned to maximize likelihood, a rare word for a topic is in reality a word that is never deduced to come from that topic (assuming another topic uses that word with greater probability). As a result, using detector 1 for anomaly detection is not meaningful. However, it is useful to look at the most frequent words as these give an indication of the semantics of the topic.

Detector 2 is useful in the case where documents are selecting words approximately uniformly and there is not enough deviation from uniformity to force the word distribution to deviate from the prior distribution. This detector serves more as a diagnostic tool as it may indicate topics that are not meaningful. However, a more common convergence issue is for multiple topics to be essentially the same, which would not be picked up by this detector.

Detector 3 helps to detect anomalous connections within a flow on the basis of port usage. In an operational setting, this would detect anomalies at a sub-flow level. In practice, it may be wise to take into account other features in addition to port when searching for sub-flow anomalies.

Detector 4 is important to help distinguish various types of document anomalies. It is possible that a document fits a single topic or a topic distribution very strongly. However, it is also possible that a document does not appear to fit any topic distribution. In this case, it will likely be relatively close to the prior distribution; this detector will find such anomalous documents.

Detector 5 may be useful in a post hoc analysis of topic distributions as it may, for example, assist in clustering topics or validating topic assignments on labeled documents. In terms of anomaly detection it is not expected that a document that heavily uses a topic that is distinct from the otheres would indicate anomalousness in any meaningful sense.

Detector 6 is probably the first detector an operational analyst would devise on their own, since it measures the anomalousness of a flow. In contrast to the machine-focused detectors 8 and 9, this detector is focused on finding individual flows that stand out against the entire corpus of flows.

Detector 7, like detector 3, is focused on sub-flow anomaly detection. Rather than analyze the topics based on their distribution $\boldsymbol{\theta}$, it compares the topics across documents using the divergence between distributions. Nevertheless, detectors 3 and 7 appear to carry essentially the same information.

Detectors 8 and 9 are alternative versions of detector 6. In particular, detector 8 is focused on finding anomalous clients and detector 9 is focused on finding anomalous servers.

Of these 9 systematic anomaly detectors, we see that three (1, 2, and 5) are appropriate for diagnostic or post hoc analysis, 5 (3, 4, 6, 8, and 9) are appropriate for operational detectors, and one (7) is redundant. We have therefore constructed five interesting and meaningful operational anomaly detectors where a more traditional approach would have settled for one. Our detectors include detectors for anomalous flows, connections, clients, and servers. Operational anomaly detection requires the analysis of data at all of these scales. Using a graphical model as the basis for a set of anomaly detectors unifies the approach and multiplies the number of meaningful detectors, but does not increase the computational demands on training the model.

D. Leave-One-Out IP Pair Characterization

The trained LDA model was used to characterize the typical behavior of an individual IP address internal to ORNL. The γ_i values are approximately equal to the α_i values plus the expected number of ports allocated to each behavior of a particular IP pair. Thus, these values characterize the typical mixing of behaviors for a given IP pair. For a given 'to' IP address, there are n connections with associated $\boldsymbol{\gamma}$ values: $\boldsymbol{\gamma}_{i,1}, \boldsymbol{\gamma}_{i,2}, \ldots, \boldsymbol{\gamma}_{i,n}$. For each IP connection, the leave-one-out weighted average of the $\boldsymbol{\gamma}_{i,n}$ is

$$\boldsymbol{\gamma}_q = \frac{\sum_{j \neq i} N_j \boldsymbol{\gamma_{i,j}}}{\sum_{j \neq i} N_j}$$

where the weight, N_j, is the number of times the particular IP pair occurred in the data set. The Kullback-Leibler divergence is then computed between the held out $\boldsymbol{\gamma}_p$ and the associated weighted average $\boldsymbol{\gamma}_q$:

$$\begin{aligned} \mathrm{D}(\boldsymbol{\gamma_q}||\boldsymbol{\gamma_p}) = \log \frac{\Gamma(\gamma_{q,\mathrm{tot}})}{\Gamma(\gamma_{p,\mathrm{tot}})} + \sum_{i=1}^{n} \log \frac{\Gamma(\gamma_p(i))}{\Gamma(\gamma_q(i))} \\ + \sum_{i=1}^{n} [\gamma_q(i) - \gamma_p(i)][\Psi(\gamma_q(i)) - \Psi(\gamma_{q,\mathrm{tot}})] \end{aligned}$$

where

$$\gamma_{r,\mathrm{tot}} = \sum_{i=1}^{n} \gamma_r(i) \quad \text{for } r = p, q,$$

and $\Psi(\cdot)$ is the digamma function, the first derivative of $\log \Gamma(\cdot)$.

V. Results

Multiple methods of inference were performed on the data. The first method, discussed in V-A, investigates the port labels typical of a given behavior (i.e., detector 1). The second method, discussed in V-B, examines the behaviors of a given server (i.e., detector 5). The third method, discussed in V-C and V-D, analyzes the incoming IP connections to a particular server to identify anomalies (i.e., detector 9).

A. Typical Port Behaviors

This subsection explores the results from detector 1 as defined in Subsection IV-C. From the derivation of the LDA model in [11], it is known that

$$\boldsymbol{\beta} = \log P(\mathbf{w}|z = k).$$

Thus, $\boldsymbol{\beta}$ can be used to identify the most likely port labels within each behavior. Table I lists behaviors and their associated ports for a model trained with five behaviors. The model was trained using only IP addresses internal to ORNL that also acted as 'to' addresses.

By inspection of the top ports within a behavior, we can hypothesize as to the role that would lead to such behavior. The first behavior and second behavior, with large fractions

TABLE I
A TYPICAL DISTRIBUTION OF TCP PORTS FOR A MODEL TRAINED WITH FIVE BEHAVIORS.

Port	TCP Protocol	Fraction of Behavior
443	HTTPS	0.8011
993	IMAPS	0.1623
389	LDAP	0.0223
143	IMAP	0.0047
515	Printer	0.0028

Port	TCP Protocol	Fraction of Behavior
80	HTTP	0.9934
21	FTP	0.0059
389	LDAP	0.0003

Port	TCP Protocol	Fraction of Behavior
53	DNS	0.9325
706	SILC Server	0.0090
910	KINK	0.0060
651	IEEE-MMS	0.0057
419	Ariel	0.0054

Port	TCP Protocol	Fraction of Behavior
25	SMTP	0.9862
113	Authorization	0.0075
587	Secondary SMTP	0.0026

Port	TCP Protocol	Fraction of Behavior
UNK	N/A	0.9632
22	SSH	0.0365

of HTTPS or HTTP, most likely correspond to web servers. The third behavior, with a large fraction of DNS, most likely corresponds to DNS servers. The fourth behavior, with a large fraction of SMTP, most likely corresponds to a mail server. The classification of the fifth port is non-obvious: the large fraction of ports not in the IANA well-known list could indicate any number of roles behind this behavior. This behavior is further discussed in V-D

When the model was trained using ten behaviors, it was found that many of the behaviors were exact duplicates. Thus, ten topics were determined to be too many.

B. Ground Truth Validation

This subsection explores the results from detector 5 as defined in Subsection IV-C. A set of known ORNL IP addresses were used to validate the usefulness of the model from the previous section. The set contains four mail servers, four DNS servers, and one web server. It was found that one of the DNS servers was never accessed during the one week period in which the data were collected. For the remaining servers, γ was used to classify the mixture of behaviors characteristic of their IP address. The normalized γs are displayed in Figure 3.

From Figure 3, it is clear that the model properly identified useful behaviors for the given set of servers. Servers of the same type can be grouped by a similar mixture of behaviors.

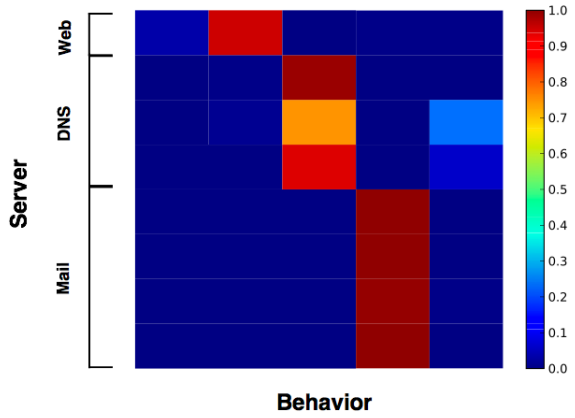

Fig. 3. The distribution of behaviors from Table I for known servers within ORNL.

Additionally, it is clear that the behaviors correspond to the labeling in V-A. It is also evident that the unknown fifth behavior is most typical of DNS servers.

C. Identification of Anomalous Events

This subsection is the first of two that explore the results from detector 9 as defined in Subsection IV-C; this one focuses on labeled servers. To validate the usefulness of the LDA model for anomaly detection, three of the known servers from the set in V-B were investigated manually. The leave-one-out Kullback-Liebler divergence was then computed for each of those servers. By ranking the from-to pairs by their divergence, it was possible to determine the pairs that varied most from the average behavior of the server and thus identify anomalous activity linked to a specific 'from' IP address.

The most anomalous connection (D =26844.15) on the web server was characterized by the usage of 389/tcp : LDAP. The three next most anomalous connections (D =11758.59, 10417.77, and 9758.78) were characterized by repeated access to 80/tcp : HTTP and 443/tcp : HTTPS, on the order of 10^3 and 10^4 times, respectively. Though this seems like typical behavior for a web server, the least anomalous connection (D =3.3998) was characterized by access to 80/tcp : HTTP and 443/tcp : HTTPS 888 and 24 times, respectively. Because γ for this server has a negligible representation of behaviors beyond the second behavior, and due to the large number of connections associated with this server, it is the purity of the behavior that is anomalous.

The most anomalous connection (D =285.67) for the mail server corresponded to attempts to access 80/tcp : HTTP. Using `nslookup`, the IP address for the 'from' connection was found to be a webcrawler. Similar behavior, as well as an `nslookup`, classified the eight next most anomalous connections as webcrawlers attempting to access the mail server through port 80/tcp. Typical behavior for this server ($D = 1.97 \times 10^{-3}$) was accessing port 25/tcp : SMTP on the order of 10^2 times.

The DNS server exhibited the strangest behavior. The top four most anomalous events (D = 2317.61) corresponded to

attempts at connections to ports greater than 1024. During the one week period, each IP address attempted to make 458 connections. The attempts at connections were synchronized: the first IP attempted to make a connection to the same port twice in a five second period, and then five seconds later the next IP address made the same double attempt at the same port, etc. These cycles of four IP addresses were consistently three to nine minutes apart. Using a geolookup on the IP addresses, all four were found to originate from the same city within a sensitive country.

The behavior exhibited by these four IP addresses is typical of port scanning [18]. During port scanning, an attacker sends TCP requests to ports on a target machine and, based on the responses, determines whether those ports are in use. One drawback for the attacker is that port scanning leaves an obvious fingerprint for port auditing tools. One way to circumvent auditing tools is to perform a stealthy port scan. For this type of port scan, the attacker tries to mask their behavior so that the auditing tool does not register it as malicious. One possible method for performing this type of port scan is to scan slowly, over the course of a day or week. The approach taken by these four IP addresses demonstrates an alternative method: spreading the port scanning over four different IP addresses. However, if this was the goal of these IP addresses, the stealthy port scan was poorly executed, since all four IP addresses attacked the *same* port in quick succession.

An alternative hypothesis to explain the four anomalies is that they were attempting to use port knocking, a cryptographic scheme originally used for authentication [19]. For example, if $A, B, C,$ and D are ports, then accessing the sequence $AABCB$ on a server might indicate that port D should be opened for the user. Although this technique was originally created for legitimate use, a malicious insider could use this technique to hide a port used for data exfiltration. It is possible that a synchronized port knocking scheme could be carried out by four IP addresses. However, the lag time between cycles of ports is not typical.

D. Characterization of Servers Exhibiting the Fifth Behavior

This subsection is the second of two to explore the results from detector 9 as defined in Subsection IV-C. The connections into ORNL were ranked according to γ for the fifth behavior from V-A. Recall that this behavior was characterized by ports not in the IANA well-known list. The IP addresses with the five largest values of γ for this behavior were examined. All five IP addresses accessed either one or two ports greater than 1024 for a total on the order of 10^6 times. They also accessed 22/tcp, 25/tcp, 80/tcp, and 443/tcp, most of which were accessed only one to ten times. 80/tcp was accessed on the order of 10^4 times for two of these IP addresses.

After consulting with the networking staff, it was determined that the internal servers involved in these connections use Splunk, a network and systems activity logging tool. This service tracks the activities on a user's computer and compiles them into a searchable repository. This software runs on all government-owned computers at ORNL. When users take these computers off-site, Splunk continues to report back on network behavior, but its attempts are blocked by ORNL's firewall.

VI. Conclusion

We developed a probabilistic graphical model for the network traffic through ORNL over a one week period. Using this model, a small number of port behaviors were found to characterize the majority of IP profiles. This model was then tested against a ground truth of known servers and found to properly identify the correct port behavior expected for a given server type. We have captured normal server behavior in a simple topic model. This model was applied to anomaly detection and successfully identified behavior indicative of stealthy port scanning or port knocking from four IP addresses from a sensitive country. The ability to identify this behavior without specifically looking for it exemplifies the advantages of probability modeling for anomaly detection schemes.

More importantly, we have demonstrated how the automatic construction of anomaly detectors from a graphical model provides a meaningful, efficient, and scalable way to augment operational detectors. The detectors are meaningful in that they detect anomalies that align with the graphical model, which is presumably designed to model the interesting properties of the data. The detector generation is efficient because it can be done systematically (with manual determination of usefulness). The training and application of the anomaly detectors is scalable since it follows as a computationally free by-product of training the graphical model. This work motivates the usage of graphical models for cyber security, since anomaly detection will follow for free from the systematic construction described in this paper.

References

[1] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: a survey," *ACM Computing Surveys*, vol. 41, no. 3, 2009.

[2] D. Koller and N. Friedman, *Probabilistic graphical models: principles and techniques*. Cambridge, MA: Massachusetts Institute of Technology, 2009.

[3] C. Siaterlis and B. Maglaris, "Towards multi-sensor data fusion for DoS detection," in *Proceedings of the ACM Symposium on Applied Computing*. ACM Press, 2004, pp. 439—446.

[4] A. A. Sebyala, T. Olukemi, and L. Sacks, "Active platform security through intrusion detection using naive Bayesian network for anomaly detection," in *Proceedings of the London Communications Symposium.*, 2002.

[5] A. Valdes and K. Skinner, "Adaptive, model-based monitoring for cyber attack detection," in *Proceedings of the 3rd International Workshop on Recent Advances in Intrusion Detection*. Springer-Verlag, 2000, pp. 80–92.

[6] A. Bronstein, J. Das, M. Duro, R. Friedrich, G. Kleyner, M. Mueller, S. Singhal, and I. Cohen, "Bayesian networks for detecting anomalies in Internet-based services," in *Proceedings of the International Symposium on Integrated Network Management*, 2001.

[7] D. Janakiram, V. Reddy, and A. Kumar, "Outlier detection in wireless sensor networks using Bayesian belief networks," in *Proceedings of the 1st International Conference on Communication System Software and Middleware*, 2006, pp. 1–6.

[8] K. Das and J. Schneider, "Detecting anomalous records in categorical datasets," in *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM Press, 2007.

[9] D. Robinson, "Statistical language analysis for automatic exfiltration event detection," *Sandia National Laboratories Report*, 2010.

[10] M. Steyvers and T. Griffiths, *Probabilistic topic models*, D. McNamara, T. Landauer, S. Dennis, and W. Kintsch, Eds. Mahwah, NJ: Erlbaum, 2005.
[11] D. Blei, A. Ng, and M. Jordan, "Latent dirichlet allocation," *Journal of Machine Learning Research*, vol. 50, no. 3, 2003.
[12] F. Tsai and K. Chan, *Blog data mining for cyber security threats*, L. Cao, P. Yu, C. Zhang, and H. Zhang, Eds. New York, NY: Springer Science, 2009.
[13] C. Andrieu, "An introduction to mcmc for machine learning," 2003.
[14] M. Chen, Q. Shao, and J. Ibrahim, *Monte Carlo methods in Bayesian computation*. Springer, 2001.
[15] M. J. Beal, "Variational algorithms for approximate bayesian inference," Tech. Rep., 2003.
[16] D. M. Blei, "Latent Dirichlet allocation in C," Web page, October 2010. [Online]. Available: http://www.cs.princeton.edu/~blei/lda-c
[17] M. Jordan, Z. Ghahramani, T. Jaakkola, and L. Saul, "An introduction to variational methods for graphical models," *Machine Learning*, vol. 37, no. 2, pp. 183–233, 1999.
[18] P. Mateti. (2010) Port scanning. [Online]. Available: http://www.cs.wright.edu/~pmateti/InternetSecurity/Lectures/Probing/index.html
[19] M. Krzywinski, "Port knocking from the inside out," *SysAdmin Magazine*, vol. 12, no. 6, pp. 12–17, 2003.