Continuous Time Bayesian Networks for Host Level Network Intrusion Detection

Jing Xu and Christian R. Shelton

Department of Computer Science and Engineering
University of California, Riverside
Riverside, CA 92521, USA
{jingxu,cshelton}@cs.ucr.edu

Abstract. We consider the problem of detecting host-level attacks in network traffic using unsupervised learning. We model the normal behavior of a host's traffic from its signature logs, and flag suspicious traces differing from this norm. In particular, we use continuous time Bayesian networks learned from historic non-attack data and flag future event sequences whose likelihood under this normal model is below a threshold. Our method differs from previous approaches in explicitly modeling temporal dependencies in the network traffic. Our model is therefore more sensitive to subtle variations in the sequences of network events. We present two simple extensions that allow for instantaneous events that do not result in state changes, and simultaneous transitions of two variables. Our approach does not require expensive labeling or prior exposure to the attack type. We illustrate the power of our method in detecting attacks with comparisons to other methods on real network traces.

Key words: Unsupervised Machine Learning, CTBNs, Host Based IDS

1 Introduction

Network attacks on computers have become a fact of life for network administrators. Detecting the attacks quickly is important in limiting their scope and destruction. Humans cannot analyze each and every connection or packet to determine if it might be part of an attack, so there is a need for an automated system for performing such checks. In this paper, we look at the problem of detecting such attacks at the host level. Instead of constructing a detector for the network as a whole, we construct a method that can be employed at each computer separately to determine whether a particular host has been compromised. While we lose global information, we gain speed, individual tuning, and robustness. A network under attack may not be able to aggregate information to a central detector.

We focus on an automatic approach to detection. While human intervention can improve security, it comes at the cost of increased time and effort. We would like a method that can adapt to the role and usage patterns of the host, while still detecting attacks automatically.

We approach this problem from the point-of-view of anomaly detection. Attacks vary greatly and new types of attacks are invented frequently. Therefore, a supervised learning method that attempts to distinguish good from bad network traffic based on historic labeled data is necessarily limited in its scope. Furthermore, the acquisition of such labeled data is time-consuming and error-prone. By contrast, gathering normal or good network traffic is relatively simple. It is often possible to designate times when we can be reasonably certain that no attacks occurred and use all of the data during that span. For an attack to be successful, it must differ in some way from normal network traffic. Our goal is to detect such abnormalities.

Our approach differs from previous approaches in a number of key ways. It is adaptive and constructed at the host level. It does not treat the packets or connections as *i.i.d.* sequences, but respects the ordering of the network traffic. Finally, it does not model the traffic features as normal or exponential distributions. Many features of network traffic are distinctly non-Gaussian and often multi-modal.

A network flow for a given host machine is a sequence of continuous and asynchronous events. We employ a generative probabilistic model to describe such dynamic processes evolving over continuous time. Furthermore, these events form a complex structured system, where statistical dependencies relate network activities like packets and connections. We apply continuous time Bayesian networks (CTBNs) [1] to the challenge of reasoning about these structured stochastic network processes. CTBNs have been successful in other applications [2, 3], but have not previously been used for network traffic modeling.

In Section 2 we relate our method to prior work on this problem. In Section 3 we briefly describe continuous time Bayesian networks. In Section 4 we describe our CTBN model and its use. In Section 5 we describe our experimental results.

2 Related Work

As a signature-based detection algorithm, we share many of the assumptions of [4]. In particular, we also assume that we do not have access to the internals of the machines on the networks (which rules out methods like [5] [6] [7] or [8]). However, we differ in that our approach does not rely on preset values, require human intervention and interpretation, nor assume that we have access to network-wide traffic information. Network-wide data and human intervention have advantages, but they can also lead to difficulties (data collation in the face of an attack and increased human effort), so we chose to leave them out of our solution.

Many learning, or adaptive, methods have been proposed for network data. Some of these (for example, [9] and [10]) approach the problem as a classification task which requires labeled data. [11] profiles the statistical characteristics of anomalies by using random projection techniques (sketches) to reduce the data dimensionality and a multi-resolution non-Gaussian marginal distribution to extract anomalies at different aggregation levels. The goal of such papers is usually not to detect attacks but rather to classify non-attacks by traffic type; if applied to attack detection, they would risk missing new types of attacks. Furthermore, they frequently treat each network activity separately, instead of considering their temporal context.

[12] has a nice summary of adaptive (or statistical) methods that look at anomaly detection (instead of classification). They use an entropy-based method for the entire

network traffic. Many of the other methods (such as [13]) use either statistical tests or subspace methods that assume the features of the connections or packets are distributed normally. [14] model the language features like n-grams and words from connection payloads. [15] also uses unsupervised methods, but they concentrate on clustering traffic across a whole network. Similarly, [16] build an anomaly detector based on Markov models, but it is for the network traffic patterns as a whole and does not function at the host level.

[10] is very similar in statistical flavor to our work. They also fit a distribution (in their case, a histogram modeled as a Dirichlet distribution) to network data. However, they model flow-level statistics, whereas we work at the level of individual connections. Additionally, they are attempting network-wide clustering of flows instead of anomaly detection. [17], like our approach, model traffic with graphical models, in particular, Naive Bayes networks. But their goal is to categorize network traffic instead of detecting attacks. [18] present a Bayesian approach to the detecting problem as an event classification task while we only care about whether the host is under attack during an interval. In addition, they also use system monitoring states to build their model, which we do not employ.

[19] is very similar to our work. It is one of the few papers to attempt to find attacks at the host level. They employ nearest neighbor, a Mahalanobis distance approach, and a density-based local outliers method, each using 23 features of the connections. Although their methods make the standard *i.i.d.* assumption about the data (and therefore miss the temporal context of the connection) and use 23 features (compared to our few features), we compare our results to theirs in Section 5, as the closest prior work. We also compare our work with [20]. They present an adaptive detector whose threshold is time-varying. It is similar to our work in that they also rely on model-based algorithms. But besides the usage of network signature data, they look at host internal states like CPU loads which are not available to us.

While there has been a great variety of previous work, our work is novel in that it detects anomalies at the host level using only the timing features of network activities. We do not consider each connection (or packet) in isolation, but rather in a complex context. We capture the statistical dynamic dependencies between packets and connections to find sequences of network traffic that are anomalous *as a group*. CTBNs are a natural modeling method for network traffic, although they have not been previously used for this task.

3 Continuous Time Bayesian Networks

We begin by briefly reviewing the definition of Markov processes and continuous time Bayesian networks (CTBNs).

3.1 Homogeneous Markov Process

A finite-state, continuous-time, homogeneous Markov process X_t is described by an initial distribution P_X^0 and, given a state space $Val(X) = \{x_1, ..., x_n\}$, an $n \times n$ matrix of transition intensities:

$$\mathbf{Q_X} = \begin{bmatrix} -q_{x_1} & q_{x_1x_2} & \cdots & q_{x_1x_n} \\ q_{x_2x_1} & -q_{x_2} & \cdots & q_{x_2x_n} \\ \vdots & \vdots & \ddots & \vdots \\ q_{x_nx_1} & q_{x_nx_2} & \cdots & -q_{x_n} \end{bmatrix}.$$

$q_{x_ix_j}$ is the intensity (or rate) of transition from state x_i to state x_j and $q_{x_i} = \sum_{j \neq i} q_{x_ix_j}$.

The transient behavior of X_t can be described by follows. Variable X stays in state x for time exponentially distributed with parameter q_x. The probability density function f for X_t remaining at x is $f(q_x, t) = q_x \exp(-q_x t)$ for $t \geq 0$. The expected time of transitioning is $1/q_x$. Upon transitioning, X shifts to state x' with probability $\theta_{xx'} = q_{xx'}/q_x$.

The distribution over the state of the process X at some future time t, $P_x(t)$, can be computed directly from Q_X. If P_X^0 is the distribution over X at time 0 (represented as a vector), then, letting exp be the matrix exponential,

$$P_X(t) = P_X^0 \exp(Q_X \cdot t).$$

3.2 Continuous Time Bayesian Networks

[1] extended this framework to continuous time Bayesian networks, which model the joint dynamics of several local variables by allowing the transition model of each local variable X to be a Markov process whose parametrization depends on some subset of other variables U as follows.

Definition 1. A conditional Markov process X *is an inhomogeneous Markov process whose intensity matrix varies as a function of the current values of a set of discrete conditioning variables* U*. It is parametrized using a conditional intensity matrix (CIM) —* $Q_{X|U}$ *— a set of homogeneous intensity matrices* $Q_{X|u}$*, one for each instantiation of values* u *to* U*.*

Definition 2. A continuous time Bayesian network $\mathcal{N}$ *over* $\boldsymbol{X}$ *consists of two components: an initial distribution* $P_{\boldsymbol{X}}^0$*, specified as a Bayesian network* $\mathcal{B}$ *over* $\boldsymbol{X}$*, and a continuous transition model, specified using a directed (possibly cyclic) graph* $\mathcal{G}$ *whose nodes are* $X \in \boldsymbol{X}$*;* $\boldsymbol{U}_X$ *denotes the parents of* X *in* $\mathcal{G}$*. Each variable* $X \in \boldsymbol{X}$ *is associated with a conditional intensity matrix,* $\boldsymbol{Q}_{X|\boldsymbol{U}_X}$*.*

The dynamics of a CTBN are qualitatively defined by a graph. The instantaneous evolution of a variable depends only on the current value of its parents in the graph. The quantitative description of a variable's dynamics are given by a set of intensity matrices, one for each value of its parents.

[21] presented an algorithm based on expectation maximization (EM) to learn parameters of CTBNs from incomplete data. Incomplete data are sets of partially observed trajectories $D = \{\sigma[1], ..., \sigma[w]\}$ that describe the behavior of variables in the CTBNs. A partially observed trajectory $\sigma \in D$ can be specified as a sequence of subsystems S_i, each with an associated duration. The transitions inside the subsystems are wholly

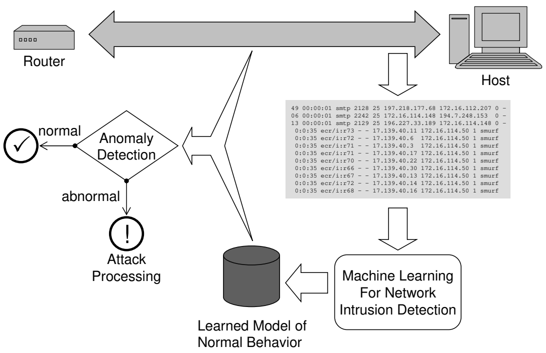

Fig. 1. The architecture of the attack detection system. The model is learned from normal traces. Anomaly detection compares the model to current network traffic. The output is the predicted labels for each time window.

unobserved. A simplest example of incomplete data is when a variable is hidden. In this case, the observed subsystems are the set of states consistent with the observed variables.

4 Approach

Figure 4 shows an overview of our host-based network attack detection system. In our work, we only focus on a single computer (a host) on the network. We use unsupervised learning to build a model of the normal behavior of this host-based on its TCP packet header traces, without looking into its system behavioral states like resource usages and file accesses. Those activities that differ from this norm are flagged as possible intrusion attempts.

4.1 CTBN Model for Network Traffic

The sequence and timing of events are very important in network traffic flow. It matters not just how many connections were initiated in the past minute, but also their timing: if they were evenly spaced the trace is probably normal, but if they all came in a quick burst it is more suspecious. Similarly, the sequence is important. If the connections were made to sequentially increasing ports it is more likely to be a scanning virus, whereas the same set of ports in random order is more likely to be normal traffic. These are merely simple examples. We would like to detect more complex patterns.

While time-sliced models (like dynamic Bayesian networks [22]) can capture some of these aspects, they require the specification of a time-slice width. This sampling rate must be fast enough to catch these distinctions (essentially fast enough that only one event happens between samples) and yet still long enough to make the algorithm efficient. For network traffic, with long delays between bursts of activity, this is impractical.

PORT	DESCRIPTION	PORT	DESCRIPTION
80	World Wide Wed HTTP	80	World Wide Web HTTP
139	NETBIOS Session Service	8080	HTTP Alternate
443	HTTP protocol over TLS/SSL	443	HTTP protocol over TLS/SSL
445	Microsoft-DS	113	Authentication Service
1863	MSNP	5101	Talarian TCP
2678	Gadget Gate 2 Way	995	pop3 protocol over TLS/SSL
1170	AT+C License Manager	51730	*unknown*
110	Post Office Protocol - Version 3	59822	*unknown*

Fig. 2. Ranking of the most frequent ports on LBNL dataset (left) and WIDE dataset (right).

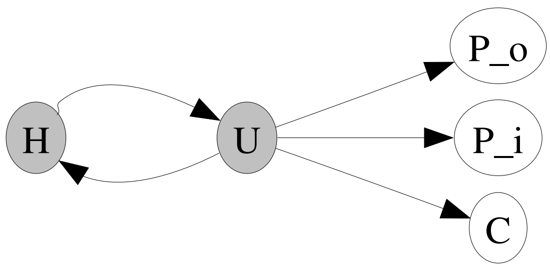

Fig. 3. CTBN port level submodel; the whole model contains 9 of such submodels.

As described in Section 3, CTBNs are well suited for describing such structured stochastic processes with finitely many states that evolve over continuous time. By using a CTBN to model the network traffic activities, we can capture the complex context of network behaviors in a meaningful and hierarchical way.

A typical machine in the network may have diverse activities with various service types (*e.g.* HTTP, mail server). Destination port numbers describe the type of service a particular network activity belongs to. Some worms usually propagate malicious traffic towards certain well known ports to affect the quality of the services who own the contact ports. By looking at traffic associated with different ports we are more sensitive to subtle variations that do not appear if we aggregate trace information across ports. Figure 2 shows the most popular ports ranked by their frequencies in the network traffic on the datasets we use (described in more depth later). These services are, to some extent, independent of each other. We therefore model each port's traffic with its own CTBN submodel.

Inside our port-level submodel, we have the fully observed node C — the number of concurrent connections active on the host — and nodes packet-in P_i and packet-out P_o — the transmission of a packet to or from the host. P_i and P_o have no intrinsic state: the transmission of a packet is an essentially instantaneous event. Therefore they have events (or "transitions") without having state. We discuss this further in the next subsection.

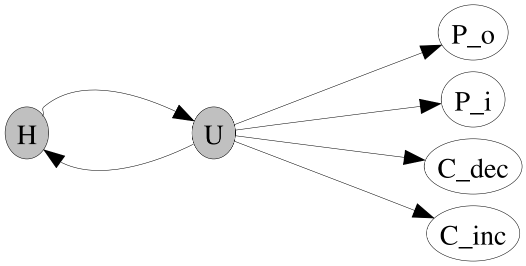

Fig. 4. The equivalent CTBN port level submodel

To allow complex duration distributions for the states of variables, we introduce another two nodes H and U. U is a partially observed variable which we call the mode whose state indicates whether the model can next send a packet, receive a packet, start a new connection, or terminate a connection. It therefore has four states, and we limit the conditional rates of the observed variables (C, P_o, and P_i) to be zero if the current mode differs from the variable's activity type. Therefore, U is observed when an event in C, P_o and or P_i occurs, but is unobserved between events.

H is a hidden variable that models the internal state and intrinsic features of the host machine. For the experiments we show later, we let H be a binary process. While the arrival times for events across an entire CTBN are distributed exponentially (as the entire system is Markovian), once certain variables are unobserved (like H), the marginal distribution over the remaining variables is no longer Markovian. These nodes make up the structure shown in Figure 3. The results in this paper are for binary hidden variables. We feel this model represents a good balance between descriptive power and tractability.

4.2 Adding Toggle Variables to CTBNs

As mentioned above, the variables P_o and P_i do not have state, but rather only events. To describe such transitionless events, we set P_o and P_i to be "toggle variables." That is, they have two indistinguishable states. As a binary variable, they have two parameters for each parent instantiation (the rate of leaving state 0 and the rate of leaving state 1) and we require these two parameters to be the same. A packet event, therefore, consists of flipping the state of the corresponding packet variable.

The concurrent connection count variable, C, also poses a slight modeling problem for CTBNs. As originally presented, CTBNs can only deal with finite-domain variables. Although most of the operating systems do have a limit on the number of concurrent connections, this number can potentially be extremely large. Our traffic examples do exhibit a wide range of concurrent connection counts. We tried quantizing C into fixed bins, but the results were fairly poor. We instead note that C can only increase or decrease by one at any given event (the beginning or ending time of a connection). Furthermore, we make the assumption that the arrival of a new connection and the termination of an existing connection are both independent of the number of other connections. This implies that the intensity with which *some* connection starts (stops) is same as any other connections. C is thus a random walk constrained to the non-negative integers.

Let Q_{inc} be the intensity for the arrival of a new connection, and Q_{dec} be the intensity for the termination of a new connection. Let $Q_{change} = Q_{dec} + Q_{inc}$. The resulting intensity matrix has the form:

$$\mathbf{Q_{C|m}} = \begin{pmatrix} & \ddots & & & & & \\ 0 & Q_{dec} & -Q_{change} & Q_{inc} & 0 & \cdots & \\ \cdots & 0 & Q_{dec} & -Q_{change} & Q_{inc} & 0 & \cdots \\ & \cdots & 0 & Q_{dec} & -Q_{change} & Q_{inc} & 0 \\ & & & & & \ddots & \end{pmatrix} .$$

Note that the only free parameters in the above matrix are Q_{inc} and Q_{dec}. Therefore, this model is the same as one in which we replace C with two toggle variables C_{inc} and C_{dec}. C_{inc} and C_{dec} operate like P_o and P_i above: their exact state does not matter, it is the transition between states that indicates an event of interest. The model structure is shown in figure 4.

4.3 An Extended CTBN Model

CTBN models assume that no two variables change at exactly the same instant. However, we might like U and H to change at the same time. As they both represent abstract concepts, there is no physical reason why they should not change simultaneously (they represent different abstract attributes about the machine's internal state).

The model in Figure 4 has 36 independent parameters.[1] Allowing U and H to change simultaneously requires introducing 24 new parameters: for each of the 8 states for U and H there are 3 new transition possibilities.

Equivalently, we can use the CTBN model shown in Figure 5 where H now has 8 states. However, this diagram does not demonstrate all of the structure. The toggle variables (P_o, P_i, C_{inc}, and C_{dec}) are each allowed to change only for 2 of the states of H (the two that corresponded to U from the previous model having the correct mode) and they are required to have the same rate for both of these states.

We have also considered other extensions to the model. For instance, it might be natural to allow the packet rate to have a linear dependence on the number of connections. However, in practice, this extension produced worse results. It seems that the rate is more of a function of the network bandwidth and the currently transmitting application can generate packets than the number of concurrent connections on a port.

4.4 Parameter Estimation

A CTBN is a homogeneous Markov process over the joint state space of its constituent variables. Our partially observed trajectory (H is unobserved, U is partially observed)

[1] Each toggle has only 1 because it can only change for one setting of U. An intensity matrix for U has 12 independent parameters for each of the two values of H and similarly H has 2 independent parameters for each of the four values of U.

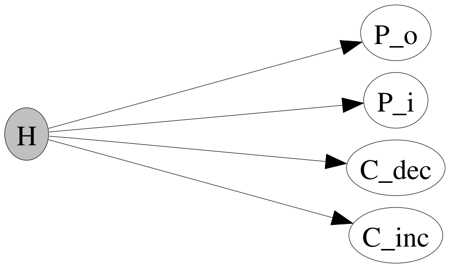

Fig. 5. The extended CTBN port level submodel

specifies a sequence of subsystems of this process, each with an associated duration. We use the expectation maximization (EM) algorithm to estimate the parameters [21]. In short, this algorithm estimates the posterior distribution over the hidden variables' trajectories. The sufficient statistics (summaries of the data necessary for learning) depend on the value of the hidden variables. EM calculates their expected value (using the posterior distribution) and then treats these expected sufficient statistics (ESS) as the true ones for selecting the parameters that maximize the likelihood of the data. This repeats until convergence.

The ESS for any variable X in a CTBN are $\bar{T}_{X|\mathbf{U}}[x|\mathbf{u}]$, the expected amount of time X stays at state x given its parent instantiation $\mathbf{u}$, and $\bar{M}_{X|\mathbf{U}}[x, x'|\mathbf{u}]$, the expected number of transitions from state x to x' given X's parent instantiation $\mathbf{u}$.

For our new types of variables (toggle variables) P_i, P_o, C_{inc} and C_{dec}, we need to derive different sufficient statistics. They are quite similar so we just take P_i as an example. Let $\mathbf{U_{P_i}}$ be its parent U's instantiation when event P_i can happen. The ESS we need are $\bar{M}_{P_i}|\mathbf{U_{P_i}}$: the expected number of times P_i changes conditioned on it parent instantiation $\mathbf{U_{P_i}}$. Since event P_i can only occur when $\mathbf{U} = \mathbf{U_{p_i}}$, the ESS $\bar{M}_{P_i}|\mathbf{U_{P_i}}$ is just M_{P_i}, the total number of times P_i changes. We also need the total expected amount of time that packet-in occurred while $\mathbf{U} = \mathbf{U_{p_i}}$.

In EM, we use the ESS as if they were the true sufficient statistics to maximize the likelihood with respect to the parameters. For a "regular" CTBN variable X (such as our hidden variable H and M), the following equation performs the maximization.

$$Q_X|u(x, x') = \frac{\bar{M}[x, x'|u]}{\bar{T}[x|u]}$$

For our new toggle variable, i.e. P_i, the maximization is

$$Q_{P_i|\mathbf{u}} = \frac{M_{P_i}}{T[U = \mathbf{u}]}$$

The above sufficient statistics can be calculated using the exact inference algorithm of [21].

4.5 Online Testing using Forward Pass Likelihood Calculation

Once the CTBN model has been fit to historic data, we detect attacks by computing the likelihood of a window of the data under the model. If the likelihood falls below a threshold, we flag the window as anomalous. Otherwise, we mark it as normal.

In our experiments, we fix the window to be of a fixed time length, T_w. Therefore, if the window of interest starts at time T, we wish to calculate $p(Y(T, T+T_w) \mid Y(0,T))$ where $Y(s,t)$ represents the joint trajectory of C_{inc}, C_{dec}, P_i, and P_o from s to t. This involves integrating out the trajectories of U and H and can be calculated in an on-line fashion using the standard forward passing inference algorithm [1].

5 Experiment Results

To demonstrate the effectiveness of our methods, we compare our CTBN models to three existing methods in the literature on two different real-world network traffic datasets and with three different attack types.

5.1 Data Sets

We verify our approaches on two publicly available real-traffic trace repositories: the MAWI working group backbone traffic [23] and LBNL/ICSI internal enterprise traffic [24].

The MAWI backbone traffic is part of the WIDE project which collects raw daily packet header traces since 2001. It collects the network traffic through the inter-Pacific tunnel between Japan and the USA. The dataset uses `tcpdump` and some IP anonymizing tools to record 15-minute traces everyday, and consists mostly of traffic from or to some Japanese universities. In our experiment, we use the traces from January 1st to 4th of 2008, with 36,592,148 connections over a total time of one hour.

The LBNL traces are recorded from a medium-sized site, with emphasis on characterizing internal enterprise traffic. Publicly released in some anonymized forms, the LBNL data collects more than 100 hours network traces from thousands of internal hosts. From what is publicly released, we take one hour traces from January 7th, 2005 (the latest date available), with 3,665,018 total connections.

5.2 Experiment Setup

For each of the datasets, we pick the ten most active hosts. To create a relatively abundant dataset based on the original packet traces, we constructed a training-testing set pair for each IP address. We use half of the traces as a training set to learn the CTBN model for a given IP (host). The other traces we save for testing. Since all the traces contain only normal traffic, we use trace-driven attack simulators to inject abnormal activities into the test traces to form our testing set. The three types of attack simulators are an IP Scanner, W32.Mydoom, and Slammer [25]. We select a point somewhere in the first half of the test trace and insert worm traffic for a duration equal to α times the length of the full testing trace. The shorter α is, the harder it is to detect the anomaly.

We also scaled back the rates of the worms. When running at full speed, a worm is easy to detect for any method. When it slows down (and thus blends into the background traffic better), it becomes more difficult to detect. We let β be the scaling rate (*e.g.* 0.1 indicates a worm running at one-tenth its normal speed).

For each algorithm (our CTBN algorithm and those below), we compute its score on consecutive non-overlapping windows of $T_w = 50$ seconds in the testing set. If the score exceeds a threshold, we declare the window a positive example of a threat. We evaluate the algorithm by comparing to the true answer: a window is a positive example if at least one worm connection exists in the window.

5.3 Other Methods

We compare against the nearest neighbor algorithms used in [19]. Not all of the features in [19] are available. The features available in our datasets are shown in Figure 6

packets flowing from source to destination
packets flowing from destination to source
connections by the same source in the last 5 seconds
connections to the same destination in the last 5 seconds
different services from the same source in the last 5 seconds
different services to the same destination in the last 5 seconds
connections by the same source in the last 100 connections
connections to the same destination in the last 100 connections
connections with the same port by the same source in the last 100 connections
connections with the same port to the same destination in the last 100 connections

Fig. 6. Features for nearest neighbor approach of [19].

Notice that these features are associated with each connection record. To apply the nearest neighbor method to our window based testing framework, we first calculate the nearest distance of each connection inside the window to the training set, and assign the maximum among them as the score for the window.

We also compare with a connection counting method. As most worms aggregate many connections in a short time, this method captures this particular anomaly well. We score a window by the number of initiated connections in the window.

Finally, we employ the adaptive naive Bayes approach of [20], which shows promising results on similar problems. We also follow the feature selection presented in this paper, while not all of them are available in our dataset. To train the Naive Bayes network parameters, we use five available features: the number of new connections in the previous three windows and the entropy of the number of distinct destination IPs and ports in the current window. All the features are discretized into six evenly spaced bins. These features are not exactly the same as those from the nearest neighbor. Each method was tuned by the authors to work as well as possible, so we try to follow each of their methodologies as best as possible to give a fair comparison; this includes the selection of features.

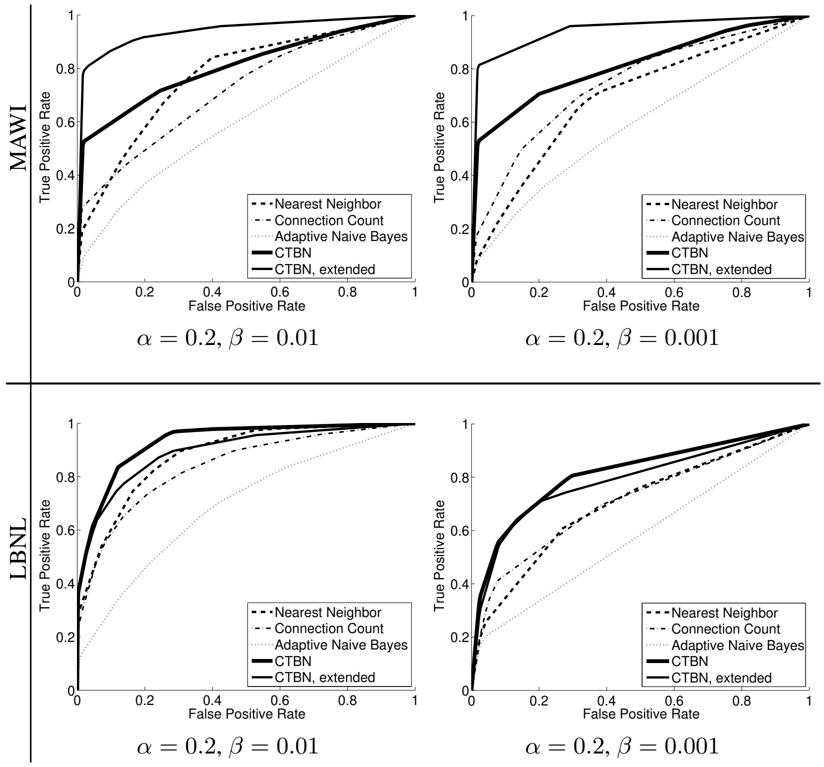

Fig. 7. ROC curves of testing results on IP scanning attack. Top: MAWI. Bottom: LBNL.

5.4 ROC Curves

Figure 7 compares the ROC curve for our method to those of the other methods for the IP scanning attack. The curves show the overall performance on the 10 hosts we chose for each dataset. α represents the fraction of time during which the attack is present and β represents the speed of the attack. The curves demonstrate that as the attack becomes more subtle (β is smaller), our method performs relatively better compared with other methods. Figures 8 and 9 show the same curves but for the Mydoom and Slammer attacks.

Each point on the curve corresponds to a different threshold of the algorithm. Because attacks are relatively rare, compared to normal traffic, we are most interested in the region of the ROC curve with small false positive rates.

We note that our method out performs the other algorithms consistently for the MAWI dataset. For the LBNL dataset, with the Mydoom and Slammer attacks, some of the other methods have better performance, in particular a simple connection counting method performs the best. We are uncertain of the exact reason, but suspect it may be due to differences in the traffic type (in particular, the LBNL data comes from enterprise traffic). The addition of a hidden variable to our model allows us to model

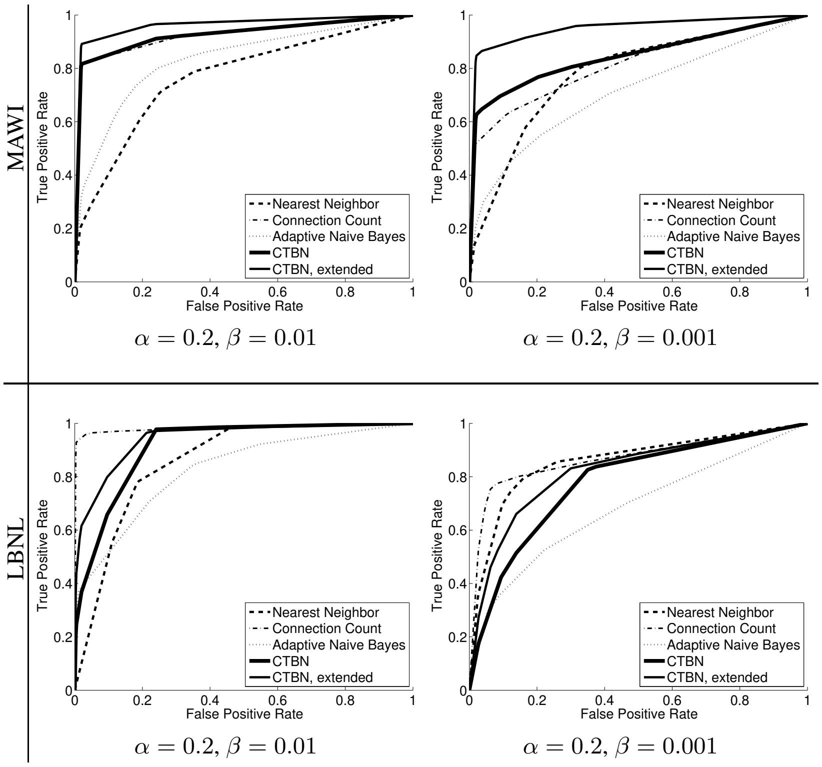

Fig. 8. ROC curves of testing results on Mydoom attack. Top: MAWI. Bottom: LBNL.

non-exponential durations. However, the complexity of such a phase-type distribution depends on the number of states in the hidden variable. If there are not enough states to model the true duration distribution, the algorithm may end up with an exponential model (at least for some events). Exponential models do not disfavor (in terms of likelihood) many quick transitions, compared to heavier tailed distributions. This might lead to worse performance in exactly the same situations where a connection-count method would work well. We hope to add more hidden states in the future to overcome this limitation.

6 Conclusions

We developed a new method for detecting network intrusions in the host level. Rather than treating packets and connections as *i.i.d* sequences, we model the network traffic by a generative probabilistic model that respects the ordering of events. The only features we used were packet and connection timings. We do not require labeled data, thus vastly improving the automation of the detection.

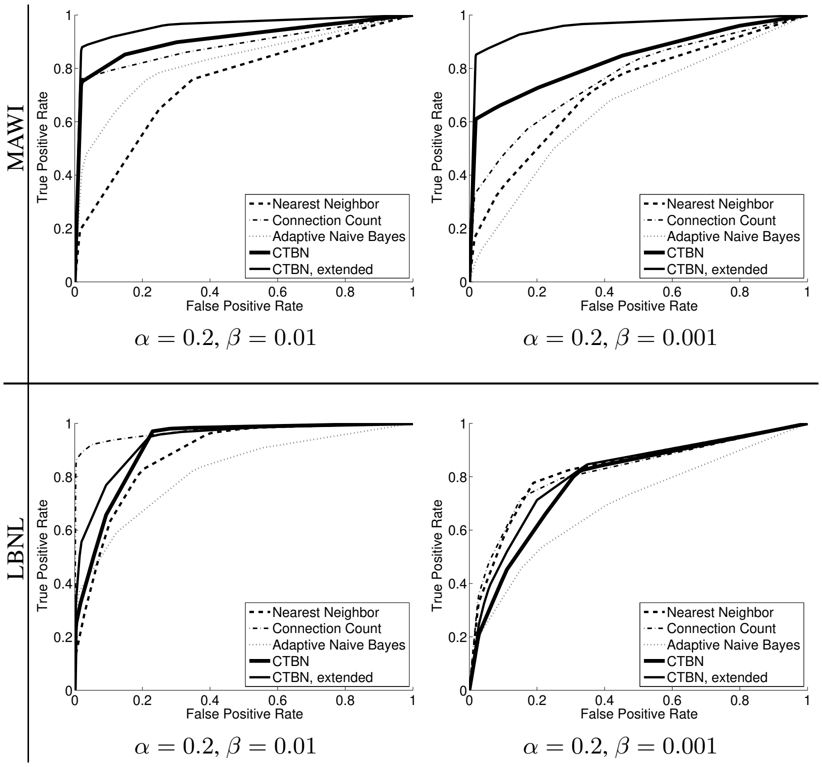

Fig. 9. ROC curves of testing results on Slammer attack. Top: MAWI. Bottom: LBNL.

Acknowledgments

This research was supported through the grant "Continuous Time Models for Malicious Network Trafic Detection" from Intel Research and UC MICRO.

References

1. Nodelman, U., Shelton, C.R., Koller, D.: Continuous time Bayesian networks. In: UAI. (2002) 378–387
2. Ng, B., Pfeffer, A., Dearden, R.: Continuous time particle filtering. In: AAAI. (2005) 1360–1365
3. Gopalratnam, K., Kautz, H., Weld, D.S.: Extending continuous time Bayesian networks. In: AAAI. (2005) 981–986
4. Karagiannis, T., Papagiannaki, K., Faloutsos, M.: BLINC: Multilevel traffic classification in the dark. In: ACM SIGCOMM. (2005)
5. Malan, D.J., Smith, M.D.: Host-based detection of worms through peer to peer cooperation. In: WORM. (2005)
6. Cha, B.: Host anomaly detection performance analysis based on system call of neuro-fuzzy using soundex algorithm and n-gram technique. In: Systems Communications (ICW). (2005)

7. Qin, X., Lee, W.: Attack plan recognition and prediction using causal networks. In: Annual Computer Security Application Conference. (2004) 370–379
8. Eskin, E., Arnold, A., Prerau, M., Portnoy, L., Stolfo, S.: A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data. In Barbara, D., Jajodia, S., eds.: Applications of Data Mining in Computer Security. Kluwer (2002)
9. Zuev, D., Moore, A.: Internet traffic classification using Bayesian analysis techniques. In: ACM SIGMETRICS. (2005)
10. Soule, A., Salamatian, L., Taft, N., Emilion, R., Papagiannali, K.: Flow classification by histogram. In: ACM SIGMETRICS. (2004)
11. Dewaele, G., Fukuda, K., Borgnat, P.: Extracting hidden anomalies using sketch and non Gaussian multiresulotion statistical detection procedures. In: ACM SIGCOMM. (2007)
12. Lakhina, A., Crovella, M., Diot, C.: Mining anomalies using traffic feature distributions. In: ACM SIGCOMM. (2005) 21–26
13. Ye, N., Emran, S.M., Chen, Q., Vilbert, S.: Multivariate statistical analysis of audit trails for host-based intrusion detection. IEEE Transactions of Computers **51**(7) (2002) 810–820
14. Rieck, K., Laskov, P.: Language models for detection of unknown attacks in network traffic. In: Journal in Computer Virology. (2007)
15. Xu, K., Zhang, Z.L., Bhattacharyya, S.: Profiling internet backbone traffic: Behavior models and applications. In: ACM SIGCOMM. (2005)
16. Soule, A., Salamatian, K., Taft, N.: Combining filtering and statistical methods for anomaly detection. In: Internet Measurement Conference. (2005) 331–344
17. W.Moore, A., Zuev, D.: Internet traffic classification using bayesian analysis techniques. In: ACM SIGMETRICS. (2005)
18. Kruegel, C., Mutz, D., Robertson, W., Valeur, F.: Bayesian event classification for intrusion detection. In: Annual Computer Security Applications Conference. (2003)
19. Lazarevic, A., Ertoz, L., Kumar, V., Ozgur, A., Srivastava, J.: A compare study of anomaly detection schemes in network intrusion detection. In: SDM. (2003)
20. Agosta, J.M., Duik-Wasser, C., Chandrashekar, J., Livadas, C.: An adaptive anomaly detector for worm detection. In: Proceedings of the Second Workshop on Tackling Computer Systems Problems with Machine Learning Techniques. (2007)
21. Nodelman, U., Shelton, C.R., Koller, D.: Expectation maximization and complex duration distributions for continuous time Bayesian networks. In: UAI. (2005) 421–430
22. Dean, T., Kanazawa, K.: A model for reasoning about persistence and causation. Computational Intelligence **5**(3) (1989) 142–150
23. MAWI: MAWI working group traffic archive http://mawi.nezu.wide.ad.jp/mawi/.
24. LBNL: LBNL/ICSI enterprise tracing project http://www.icir.org/enterprise-tracing/Overview.html.
25. NLANR: National laboratory for applied network research (2006) http://www.nlanr.net.