ALADIN: Active Learning of Anomalies to Detect Intrusion

Jack W. Stokes, John C. Platt
Microsoft Research
One Microsoft Way
Redmond, WA 98052
Email: {jstokes,jplatt}@microsoft.com

Joseph Kravis
Microsoft Network Security
Redmond, WA 98052 USA
Email: jkravis@microsoft.com

Michael Shilman
ChatterPop, Inc.
Email: shilman@chatterpop.net

March 4, 2008

Technical Report MSR-TR-2008-24

This page intentionally left blank.

Abstract

This paper proposes using *active learning* combined with *rare class discovery* and *uncertainty identification* to statistically train a network traffic classifier. For ingress traffic, a classifier can be trained for a network intrusion detection or prevention system (IDS/IPS) while a classifier trained on egress traffic can detect malware on a corporate network. Active learning selects "interesting traffic" to be shown to a security expert for labeling. Unlike previous statistical misuse or anomaly-detection-based approaches to training an IDS, active learning substantially reduces the number of labels required from an expert to reach an acceptable level of accuracy and coverage.

Our system defines "interesting traffic" in two ways, based on two goals for the system. The system is designed to *discover* new categories of traffic by showing examples of traffic for the analyst to label that do not fit a pre-existing model of a known category of traffic. The system is also designed to *accurately classify* known categories of traffic by requesting labels for examples which it cannot classify with high certainty. Combining these two goals overcomes many problems associated with earlier anomaly-detection based IDSs.

Once trained, the system can be run as a fixed classifier with no further learning. Alternatively, it can continue to learn by labeling data on a particular network. In either case, the classifier is efficient enough to run in real-time for an IPS.

We tested the system on the KDD-Cup-99 Network Intrusion Detection dataset, where the algorithm identifies more rare classes with approximately half the number of labels required by previous active learning based systems. We have also used the algorithm to find previously unknown malware on a large corporate network from a set of firewall logs.

1 Introduction

With the proliferation of on-line networks attacks and new forms of malware, computer security is an increasingly important area of research. Network intrusion detection involves identifying malicious activity by analyzing the transfer of packets across a computer network, while system and security events generated by a computer's operating system can be used to detect malware or unauthorized logons on individual computers (i.e. host intrusion detection). In the past, two styles of intrusion detection and prevention systems (IDS/IPS) have been proposed in the literature: misuse systems and anomaly detection. Misuse IDSs can either be rule-based where an expert analyst designs rules to match known traffic patterns [1] or statistically based where rules (i.e. classifier weights) are learned from the data itself [2]. Anomaly detection systems typically learn a statistical model of *normal* traffic and anomalies are identified as traffic patterns which do not fit the normal model [3, 4]. Hybrid models attempt to merge both styles of intrusion detection.

In the past, anomaly detection systems have performed poorly for a number of reasons. However, given the increasing complexity of the network traffic in an adversarial world, we believe that, in conjunction with a misuse system, anomaly detection must be employed in order to *discover* new network attacks masquerading as legitimate network transfers. One main problem of previously proposed anomaly detection algorithms is that given the huge number of different types of traffic on the network or in a computer, it is extremely difficult to design a *single* model to represent all *normal* traffic. Therefore, *anomalies must be identified within each separate class of traffic* [5]. Furthermore, we believe that misuse detection can also benefit from classifying known classes of normal traffic in addition to malicious traffic.

Another problem with standard anomaly detection systems is that rare events are often uninteresting. Typically, an IDS based on anomaly detection presents analysts with rare events, assuming that rare events are the result of intrusions. However, "rare" is not necessarily synonymous with "interesting." In our experience with the network packet logs, we have found many examples of normal traffic that have features that are unusual. We need a system to filter out these rare, but uninteresting items, in order to find the very rare, but interesting samples such as a zero-day worm.

Expert analysts have an amazing ability to identify strange traffic patterns within a cluster of traffic with similar characteristics. Training both rule-based or statistically-based systems requires a huge amount of time from expert analysts either designing rules for the former or labeling traffic for the latter. Employing security experts is difficult and expensive; thus in addition to achieving the highest possible detection rate, minimizing the amount of time required by the analysts is a primary objective in training an IDS/IPS. Active learning is a methodology for statistically based systems where the algorithm identifies the best samples to label thereby producing the highest classification accuracy with the least amount of human effort [6]. Active learning provides a tool to attempt to deal with the huge amounts of computer or network traffic in corporate computer networks. For example in October 2006, corporate proxy firewalls from Microsoft Corp. running Internet Security and Acceleration (ISA) Server logged up to 25 million entries of outbound network packets on a single day to a SQL database where an entry corresponds to a corporate computer sending data to another computer located outside of the corporate intranet. The ISA logs contained only 10% of the total amount of outbound, firewall traffic for any given day: analysts could be responsible for examining billions of events per month.

Recently, active learning has been used to minimize the number of samples (i.e. human effort) required for anomaly detection [7]. As the number of distinct classes of network traffic continues to increase, identifying anomalies within each class of traffic requires more and more labeled data. Again, active anomaly detection significantly reduces the effort required by analysts in order to discover the "rare" and interesting traffic which is most likely to be new instances of network attacks.

This paper proposes ALADIN: a synthesis of the active learning approach for classification and active anomaly detection. ALADIN stands for "Active Learning of Anomalies to Detect INtrusions" and is applicable to both host-based and network intrusion detection. Instead of presenting a firehose of anomalies to a security analyst, we first present a small number. The analyst then labels these into categories. We use these labels to update both a classifier and an anomaly detector. We repeat alternating labeling and learning: once an item gets labeled, items similar to it are no longer anomalies, and hence get classified correctly. This allows analysts to quickly filter out common categories of traffic and identify rare anomalies that are new security risks. The main contribution of this paper is a new framework that both quickly finds new classes of traffic using anomaly detection and also creates classifiers with high prediction accuracy.

Network intrusion detection systems try to detect inbound attacks from ingress network traffic. A related problem is searching for computers infected with malware which are transmitting data outside of a company by analyzing egress traffic. The ALADIN algorithm can also be used to design classifiers based on egress network traffic features. In this paper, we describe the design and implementation of a malware classification system based on ALADIN.

To be clear, the combination of classification and anomaly detection is only used during the *training* phase. For real-time (IPS) or off-line (IDS) detection, the classifier's weights are used as a statistical misuse system. Weights for a general purpose IDS/IPS can be generated by a relatively small number of expert security analysts working for the IDS manufacturer who label data for a large number of customers as shown in figure 1. As new samples are labeled, updated classification weights can be downloaded much like signatures for anti-virus engines are continuously updated today. Most small to medium size businesses should be able to use the general purpose weights with excellent performance. However the framework is extensible;

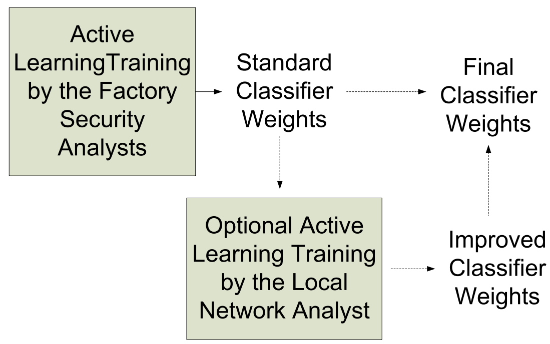

Figure 1: General Classifier Training with Optional End-User Adaptive Training.

for those companies which use the IDS/IPS and employ a small number of analysts who understand the specific nature of their corporate computing environment, the general purpose weights can further be adaptively trained based on any labeled samples provided by the end-user analysts. When new general purpose weights are available, samples previously labeled by the end-user can be used again to re-adapt the new general purpose weights.

We first review details of related work in active anomaly detection (section 2.1), in using active learning for an IDS (section 2.2), and in multi-mode and hybrid intrusion detection (section 2.3). We then present our proposed system for combining active learning, anomaly detection, and high classification accuracy in section 3. We describe our experiences with the use of the ALADIN algorithm for detecting malware on a corporate network from firewall logs for outbound network packets in section 4 and show experimental results for network intrusion detection in section 5 for the KDD-Cup 99 data set. We present conclusions and propose future research topics in 6.

2 Related Work

2.1 Active Anomaly Detection

Recent work in the machine learning community has investigated algorithms that combine active learning and anomaly detection. The machine learning algorithms discussed in this paper act on generic items, that can be in an event log. Each item is represented as a vector of features, $\vec{x}$, that describe a set of traffic (see section 4 for our definition of features). Features can be either continuous variables or a discretization of a categorical variable.

Pelleg and Moore [7] proposed a system for active anomaly detection. They view the goal of anomaly detection as the detection, with as few samples as possible, of all of the categories of data in an unlabeled data set. Thus, they share one of the goals of ALADIN: the equivalent task of discovering new types of network attacks or malware.

The key insight in Pelleg and Moore is that anomalies can be disregarded once they are labeled by an expert and accounted for by a model. Therefore, they maintain a model of input vectors $\vec{x}$: $P(\vec{x}|i)$, one model for each class label i. They measure anomalousness by how unlikely a data point is according to its class' model. In Pelleg and Moore, an unlabeled data point undergoes four steps:

1. The class label is estimated by finding the most likely class via

$$c = \arg\max_i P(i|\vec{x}) = \arg\max_i \frac{P(\vec{x}|i)P(i)}{\sum_j P(\vec{x}|j)P(j)} \tag{1}$$

   where $P(i|\vec{x})$ is the probability of a set of traffic described by feature $\vec{x}$ being in class i, $P(\vec{x}|i)$ is a model of traffic in class i where it is the probability of observing $\vec{x}$ given that we are certain that the traffic is in class i, and $P(i)$ is the prior probability of observing traffic in class i (without observing any features).

2. The anomaly score for an item is: $-\log P(\vec{x}|c)$.

3. Items with high anomaly scores are shown to an expert for labeling.

4. The models $P(\vec{x}|i)$ are updated with new labeled data. Repeat.

Pelleg and Moore propose several different ways of selecting items in step 3. The method they favor (called *interleave*) is to select a fixed number of outliers per class (where the outliers per class are selected to have the highest anomaly scores).

Note that Pelleg and Moore's work is optimized for finding anomalies as quickly as possible. The classifier used in step 1 of their algorithm, trained on the data labeled in step 3 may not provide the highest accuracy classification results.

Another active-learning-like algorithm for anomaly detection was presented in [21]. That algorithm used active learning as a method for creating diverse anomaly detectors: the number of labeled samples presented to an expert may become quite large.

2.2 Active Learning for Intrusion Detection

Another related work is that of Almgren and Jonsson [6], which uses active learning to make an accurate, statistical misuse intrusion detector. Almgren and Jonsson use a form of active learning [22], where an expert labels unlabeled items that lie near decision boundaries of a classifier. The items near the decision boundary are those that the classifier is most uncertain about: therefore, labeling them has the most value. In addition, the highly accurate Support Vector Machine (SVM) classifier only pays attention to items near the decision boundary: points far away are ignored. In [6], Almgren and Jonsson show that active learning provides much better results for classification accuracy in an IDS when compared to random sampling.

The Almgren algorithm operates in the following way:

1. A two-class radial basis function (RBF) SVM is evaluated on each data point $\vec{x}$:
$$z = \sum_j \alpha_j y_j K(\vec{x}_j, \vec{x}) + b \quad (2)$$
where z is the score for the item belonging to the positive class, y_j is the label for a set of traffic j, +1 if positive, -1 if negative, $(\vec{x}_j, y_j)$ are the labeled items, and α_j is the result of the SVM algorithm. $K(\vec{x}_j, \vec{x})$ is a measure of the similarity between a new input vector $\vec{x}$ and a stored memory vector $\vec{x}_j$: typically K is large when the two vectors are similar, and close to 0 if not.

2. The certainty score for an item is $|z|$.

3. Items with low certainty scores are shown to an expert for labeling.

4. The SVM is re-run with the new labeled data. Repeat.

Since Almgren's algorithm is optimized to accurately predict the label for the unlabeled samples, it will not typically find anomalies as quickly as Pelleg's anomaly detection algorithm. The paper also only presented a 2-class algorithm; we show that the performance is worse in multi-class problems in section 5.

2.3 Multi-Mode and Hybrid Intrusion Detection

Typical anomaly detection algorithms consider two modes: normal and anomalous. By modeling normal behavior correctly, the goal is to identify anomalous behavior which hopefully corresponds to detecting malicious activity. However, learning a single model representing all normal behavior is problematic and leads to high false positive rates where anomalous, *normal* behavior is incorrectly predicted to be malicious. As in the algorithm proposed in this paper, Valdes [5] also approaches anomaly detection for intrusion detection using multiple classes, or modes. Valdes analyzes multiple modes for sequences corresponding to the categorical destination TCP port.

Recently, Lane [23] proposed a hybrid IDS based on combining a statistical misuse IDS with an IDS based on anomaly detection. This algorithm uses semi-supervised learning with a partially observable Markov decision process (POMDP). Lane's algorithm is applied to host based time series of UNIX commands.

3 Proposed Active Learning Algorithm for High Classifier Accuracy and Rare-Class Discovery

It is possible to simply run both Pelleg's algorithm and Almgren's algorithm to create an intrusion detection system which both finds new intrusions and refines the rules for existing known categories. However, a security analyst would then be bombarded with labels: neither algorithm would cooperate or share labels.

Therefore, we propose the ALADIN algorithm: a single intrusion detection framework for both anomaly detection and classification. In ALADIN, the labels provided by the experts update both the classifier and the anomaly detector. The classifier finds uncertain items, which get labeled by an analyst. The anomaly detector consists of one model per class, where items that do not fit the corresponding model are considered anomalous. Anomalous items are also labeled by the analyst. The overall architecture of ALADIN is shown in figure 2.

3.1 Learning the Classifier

The first stage of the proposed algorithm is illustrated in the left hand side of figure 3. After the analyst is finished labeling one or more of the samples proposed by the active learning algorithm, illustrated by the larger dots in the figure, a multi-class, discriminative classifier is trained. In our work, we choose to use logistic regression [24] which learns models of the form

$$P(\text{class } i|\vec{x}) = 1/(1 + \exp(-\sum_j w_{ij}x_j + b_i)) \tag{3}$$

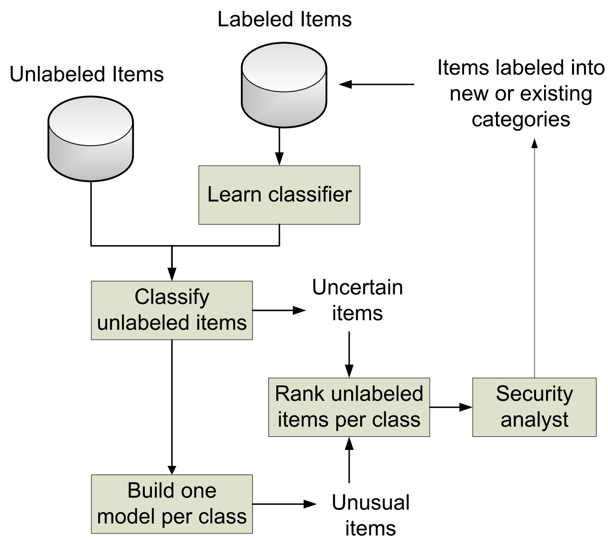

Figure 2: ALADIN algorithm for detecting malware from intrusion detection logs.

where x_j is the value of the jth feature in an item. Logistic regression chooses the parameters w_{ij} and b_i to minimize the cross-entropy loss function E [24]

$$E = -\sum_n \sum_{i=1}^{I} t_{in} \log P(i|\vec{x}_n) + (1 - t_{in}) \log(1 - P(i|\vec{x}_n)) \tag{4}$$

where t_{in} is 1 if the nth input vector ($\vec{x}_n$) is in class i and 0 otherwise, and I is the total number of classes.

Logistic regression is chosen so that the classifier can be quickly retrained for each labeling iteration, without forcing the analyst to wait for a lengthy training algorithm. Other classifiers, such as a linear SVM or RBF SVM [24] can also be used.

Note that for features x_j that are categorical (i.e., their domain is a discrete set), ALADIN uses the standard encoding [6], where each categorical feature is represented by N binary inputs to the logistic regression. Only one of these inputs has a value of one, corresponding to the feature category: the rest are zero. Continuous features are scaled so that the mean of the entire (label+unlabeled) data set is zero, and the variance is one.

In order to increase the accuracy of the logistic regression, ALADIN selects items for analyst labeling that have high uncertainty. That is, ALADIN chooses items with a low certainty score:

$$\min_{i,j \neq i} |P(i|\vec{x}_n) - P(j|\vec{x}_n)| \tag{5}$$

where $i = \arg\max_k(P(k|\vec{x}_n))$. Traffic with a low certainty score has the property that ALADIN has difficulty assigning the traffic to one class, because two classes are almost equally likely.

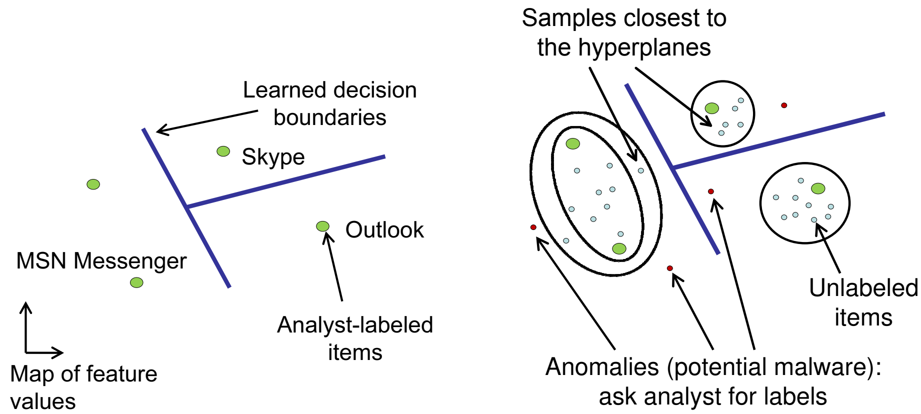

Figure 3: The left figure shows the result of training a classifier: decision boundaries in input space are hyperplanes. The right figure shows the selection of the anomalies by the per-class models.

3.2 Anomaly Detection

The anomaly detection of ALADIN is similar to that used in [7]. A model of all (labeled + unlabeled) data is built for each class. For each unlabeled item, the classifier is applied and the unlabeled item is temporarily assigned to its most likely class.

A model is then trained using all of the items assigned to a class. ALADIN uses naive Bayes for the model which assumes that all of the input features are statistically independent (given an item in a class). Thus, the probability of an item is the product of the probabilities of each feature. Equivalently, an anomaly score can be computed by taking the negative of the sum of the log of the probabilities (i.e. loglikelihood) for each feature:

$$-\log P(\vec{x}|\text{class } c) = -\sum_j \log P(x_j|\text{class } c) \tag{6}$$

where a large anomaly score indicates a low probability (anomalous) item. We build a model of each feature independently. Traffic tends to be anomalous if at least one feature is out of the ordinary; the more features that are strange, the higher the anomaly score. In addition, the more unusual a feature is, the higher the anomaly score. For anomaly detection, ALADIN presents unlabeled items to the analyst with large anomaly scores for each class.

The form of the feature probability $P(x_j|c)$ depends on the type of feature. If the feature is continuous, the probability is a Gaussian distribution, whose mean and variance is measured over all items assigned to class c. For a categorical feature, the probability is a histogram (a multinomial) estimated over all items assigned to class c.

In order to search for new malware or styles of intrusion, ALADIN selects a number of anomalies per class to show to a security analyst for labeling. These are the items that are far away from the centers of the clusters in the right side of figure 3: these items do not fit into the current model of a class, hence are potential new styles of intrusion or malware.

At every labeling iteration, ALADIN presents the same number of items to an analyst for labeling, evenly distributed between all known categories: half of the items are uncertain, the other half are anomalous. Thus if all samples were to be labeled in the order suggested by the algorithm, the analyst spends half of the time improving the classification performance, the other half finding potential new problems. It should be noted that anomalous items which lie near the decision boundaries may also be considered to be uncertain items. These items may belong to a different class and hence are labeled incorrectly, or may belong to a new, previously unlabeled class. However the opposite may not be true: anomalous items which are located far from the decision boundaries may turn out to be new categories which require a new class label but most likely do not belong to an existing class.

Instead of ranking based on (6) for anomaly detection, another potential algorithm could be to rank the anomalousness based on the output (posterior) probability, $P(c|\vec{x})$, resulting from the classification stage. This method does not tend to perform well since the output probability is a function of the classification boundaries and tends to be dominated by the distance to the nearest boundary. As a result, this type of algorithm may fail to accurately detect anomalies located far away from the boundaries.

Sometimes, the logistic regression identifies very few unlabeled data as belonging to a particular class (due to a severe lack of labeled data). In this case, there are fewer uncertain or anomalous items in the rare category compared to other categories. When this happens, ALADIN simply ranks all unlabeled data in order of the logistic regression output $P(c|\vec{x})$, and provides the most likely items for analyst labeling. This balances the number of labels per category and ensures that very rare categories do not get starved for labels during active learning.

The summary of one iteration of ALADIN's training phase is shown in figure 4.

---

1. Learn a classifier by minimizing the loss (4) on the labeled samples
2. Evaluate the classifier (3) and the certainty score (5) for the unlabeled samples
3. Assign all unlabeled samples to the most likely category
4. Compute the model parameters (mean, variance, or histogram) for every $P(\vec{x}_j|c)$
5. Compute the anomaly score (6) for all unlabeled samples
6. Select the next group of samples to be labeled choosing as follows, sweeping through the categories:
   (a) select the next, most anomalous unlabeled sample with the highest anomaly score (6) in each class
   (b) OR, select the sample with the smallest certainty score (5) for each class
   (c) if not enough samples for a class are found from 6a) or 6b) select the unlabeled sample with the second highest output probability $P(c|\vec{x}_j)$ corresponding to the desired class c
7. Repeat step 6 until the desired number of samples have been labeled for the iteration

---

Figure 4: One iteration of ALADIN training, as pseudo-code

3.3 Adaptive Training for the End-User Analyst

The framework can evaluate new samples in two modes as shown in figure 1: standalone where the generic classifier weights are provided by the third party which developed the system or in an adaptive mode where an end-user analyst provides additional labeled samples (uncertain or anomalies) using the active learning algorithm proposed above. To be clear, additional training by the end-user analyst is optional, and very good results can be obtained without further adaptive training.

3.4 Evaluating New Samples

After the weights for the linear classifier have been trained using logistic regression, new samples can be evaluated in an IDS or IPS. In both standalone and adaptive modes, all samples are evaluated using (3). Although the length of the vector can be over 100,000 dimensions, the input data is sparse and (3) can be evaluated very efficiently. As a result, evaluation can be real-time for an IPS or off-line as an IDS which analyzes large log files.

3.5 Relationship to the *Interleave* and *Mix-Ambig-Lowlik* Methods

In [7], Pelleg and Moore propose four algorithms: *lowlik*, *ambig*, *mix-ambig-lowlik*, and *interleave*. The first three algorithms find anomalies considering all of the data at once. *Lowlik* identifies anomalies as being the samples with the lowest likelihood among all of the samples, while *ambig* proposes anomalies which are most ambiguous (i.e. samples closest to the decision boundaries) again on a global scale. *Mix-ambig-lowlik* is a hybrid method which combines both of these strategies for all data samples. By contrast, the *interleave* method proposes samples with low likelihood for each individual class. Thus *interleave* is similar to step 6a) in ALADIN. Iterating between samples with low likelihood and high uncertainty in 6a) and 6b) is similar in nature to Pelleg's *mix-ambig-lowlik*, but differs in the following way; ALADIN combines selecting the two types of samples (i.e. anomalous, uncertain) on a per class basis, while *mix-ambig-lowlik* mixes the two types of samples across the entire data set. Just as Pelleg and Moore show that the *interleave* method outperforms the other methods by searching for rare-classes on a per class basis, ALADIN extends the notion to combine both strategies in the active learning scenario on a per class basis.

4 Malware Detection from Firewall Logs of Outbound Network Traffic

A prototype, large-scale malware detection system has been implemented based on ALADIN. Security analysts have used this system to analyze daily corporate network transmission logs from Microsoft Corp. generated by the proxy firewalls using Microsoft's ISA Server. ISA allows network packet metadata to be collected and stored in SQL databases. The metadata corresponding to the packets are the basis of the features used in the system and are a subset of those captured by the standard ISA logs [25].

The data from each of the separate SQL databases collected by the approximately twenty corporate network ISA servers is aggregated on a daily basis based on the Greenwich Mean Time (GMT). The resulting combined daily log file for the corporation is then preprocessed using the current available models created from any previously labeled data to predict a label for each of the unlabeled samples for a particular day.

4.1 Reviewing the Current Results

Scalability is an issue when dealing with extremely large data sets such as those produced by the corporate ISA logs. To combat this problem, the analyst first selects the top N (e.g. 1000) samples to label, where N is a user defined input, which have been ranked according to the proposed algorithm. As explained previously, approximately $N/2$ samples are used for anomaly detection and $N/2$ samples are used for improving classification accuracy. Analyzing the top N samples addresses two issues. First, once the samples have been selected initially, any subsequent labeling of data, updating of the models, and re-ranking of the results is only performed on this initial group of samples. This architecture allows a fast, interactive experience when the analyst labels new samples. Second, the analyst would be overwhelmed by being presented with the all of the samples to be labeled at once. By providing only the top N samples for labeling, the analyst can select a comfortable number of samples to analyze.

Another issue to explore is whether or not an automatic stopping criterion should be provided in the algorithm for presenting items from a particular class to the analyst for labeling. We believe that with the analyst in the loop, any labeled data can always be used to improve the algorithm. The algorithm will always offer new samples to label if the analyst is so inclined. If the algorithm automatically decides to hide a class for further review, it is possible that new malware whose features lie in the same space as a previously labeled class may be hidden with no chance for discovery. Furthermore using the current UI, the analyst can choose to label any item they desire in the top N list: they can choose to avoid labeling new items if a class seems to be well formed. Therefore, we do not suggest providing an automatic stopping criterion.

For a large number of classes, the analyst may be presented with only a single item from some classes and no items from other classes. Consider the case with $N = 1000$ but there are currently 2000 labeled classes, the question becomes which items should be presented to the analyst to label. In this case, we suggest providing the analyst with the N samples corresponding to the largest negative anomaly score for one iteration and the N samples with the smallest uncertainty score for the next iteration.

4.2 Reprocessing the Data

During a labeling session, any updated models are only applied to the unlabeled samples from the original top N samples. The analysts also have the option of reprocessing all of the unlabeled samples from the daily log files. In the current implementation, this requires an hour or so depending on the size of the file although much of this time is used to update the predicted label and anomaly score that are also stored in the daily ISA SQL table. If the analyst exits the tool, they return to analyzing the same top N samples ranked initially by the algorithm or during the most recent reprocessing step.

4.3 Implementation Details

The application is written in C# including all of the underlying machine learning algorithms. The UI consists of DataGridView on a WindowsForm which is used to display the rank, user label, predicted label, and the features. The DataGridView allows the analyst to pivot on any column (rank, label, feature) providing additional insight into the ranking process. The application is multi-threaded: labeling one or more samples causes the application to create an additional thread which is used to update the models based on the new class labels.

4.4 Malware Detection

We have applied ALADIN to several daily logs averaging 13 million captured events for each day. During testing of the tool, a new trojan ("5.exe") was discovered on the network which had not been previously identified by the corporate, rule-based misuse NIDS. In addition, other worms and trojans were also identified as anomalies during various iterations: these additional malware samples were also identified by the Network Security team's rule-based misuse NIDS but were waiting to be classified by analysts.

5 Network Intrusion Detection Experiments and Results

In this section, we analyze the components of ALADIN in the network intrusion detection setting to show that the combination of classification and anomaly detection are required to attain high accuracy quickly. We evaluate ALADIN's performance by comparing it to two different versions of Almgren's active learning intrusion detection system [6]. The first version simply drops the anomaly detection from ALADIN, while the second version uses RBF SVMs, to be more similar to [6]. Following [6], the SVM uses $C = 1$ and a Gaussian kernel with $\sigma^2 = 0.5$.

To compare the algorithms, we use the data from the 1999 KDD-Cup contest for network intrusion detection systems [26]. Although the data set is somewhat dated, we use the packet labels to conduct oracular experiments which are required to simulate how the algorithm performs with perfect knowledge of the class labels. These labels allow the comparison of the proposed active learning algorithm and the active learning algorithm presented in [6]. To be consistent with [6], all features from the 1999 KDD-Cup are used: no feature selection was performed which may give an unfair advantage to the proposed method.

Specifically, the first 100,000 samples from the file *kddcup.data_10_percent* are used, and the distribution of the data is shown in table 1. As the table shows, the data contains three common classes of traffic (*normal*, *smurf*, *neptune*) and seventeen rare classes. From a small number of labeled samples, ALADIN attempts to identify labels for all of the unlabeled samples. As a result, the algorithm may quickly identify all packets belonging to a single attack. For example, for the single Neptune SYN flood attack, the algorithm attempts to label all packets in the attack with the same label. If a variant of a particular attack occurs, a new label can be created representing the new attack example.

Class	Count	Class	Count	Class	Count	Class	Count
normal	56237	satan	539	pod	40	multihop	6
neptune	20482	portsweep	278	warezmaster	20	buffer_overflow	5
smurf	19104	nmap	231	land	17	phf	3
back	2002	teardrop	199	imap	12	loadmodule	2
ipsweep	760	guess_passwd	53	ftp_write	8	perl	2

Table 1: Distribution of process classes in the first 100,000 samples of *kddcup.data_10_percent*

To begin, 10 randomly chosen samples are labeled before running the first iteration of the algorithm for all experiments in this section. For all experiments, 100 samples are labeled for each iteration so that after 10 iterations, approximately 1% of the samples have been labeled. Choosing 100 samples to label per iteration corresponds to approximately 5-10 samples labeled per class depending on the number of previously labeled classes.

5.1 Anomaly Detection

In this first experiment, we seek to compare how quickly the algorithms discover classes in the data set. The results are illustrated in figure 5 for ALADIN and the two supervised algorithms using the KDD-Cup 99 data set. Three classes were sampled in the original 10 labeled samples. ALADIN quickly identifies six new classes in the first iteration and eighteen of the twenty classes after the fourth iteration of the algorithm. The other two algorithms labeled "Logistic Regression" and "SVM" are standard active learning algorithms without the additional anomaly detection stage; dropping anomaly detection from ALADIN degrades the detection performance of the system (as expected). Using an RBF SVM only identifies three classes in addition to the three in the original labeled data, while using the Logistic Regression identifies sixteen classes. The results clearly show that ALADIN's use of active anomaly detection significantly outperforms standard active learning. Typically, adding the second stage of anomaly detection requires only half the number of samples to be investigated and labeled by an analyst compared to the next best alternative.

5.2 Error Rates for Classification of Unlabeled Data

Next, we analyze the error rates of the three algorithms in figure 6 generated by comparing the predicted labels of the unlabeled data using the supervised classifier with the oracle labels from the data set. ALADIN performs extremely well with an error rate ranging from 5.9% at iteration two to 3.1% on iteration 10. The other two algorithms also converge to low error rates of 3-4% due to reliance on discriminative classifiers for prediction but exhibit high error probabilities in the initial iterations during which a sufficient number of samples have not been labeled to train accurate class estimators. These large spikes in error rate correspond to a single classifier mispredicting one of the common attacks (e.g. *smurf*,*neptune*) as *normal*.

In tables 2 and 3, we investigate the false positive (FP) and false negative (FN) rates for the seven most prevalent classes. In this multiclass setting,

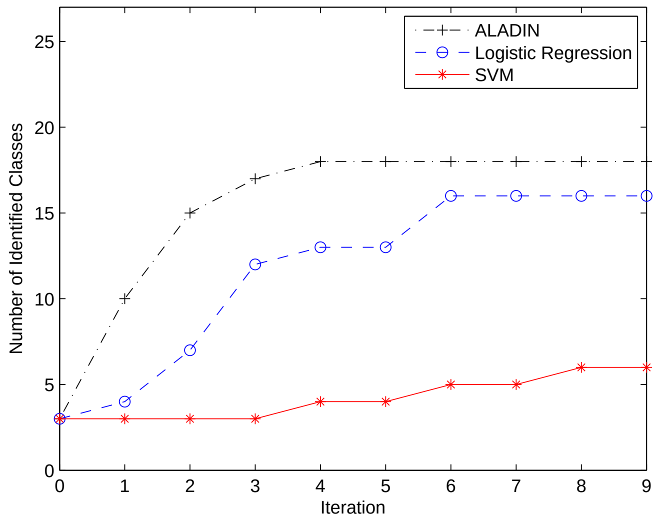

Figure 5: Number of identified classes using the proposed ALADIN algorithm and two versions of supervised active learning algorithm.

the FP and FN rates are calculated for each class relative to all of the other classes. Tables 2 and 3 provide the results after labeling 10 iterations, 1000 total samples, and 20 iterations, 2000 total samples, respectively. These tables show that the asymptotic error rate for ALADIN in figure 6 is mostly due to false negatives whereby the anomalous attack classes are incorrectly labeled as *normal*. After labeling 1000 samples, ALADIN misclassifies most or all of the samples in the *back* and *ipsweep* classes. The classifier learns the *ipsweep* class but still misclassifies most of the *back* class after labeling 2000 samples.

After analyzing the KDD-Cup 99 data set, we believe that the *normal* class is overly broad and that the anomaly detection is often identifying subclasses within the *normal* class for more granular labeling. Thus, *normal* may not be easily classified with a single logistic regression classifier. ALADIN may work even better in practice where analysts are able to further label normal traffic as belonging to a known class (e.g. MSN Messenger, Skype, etc.).

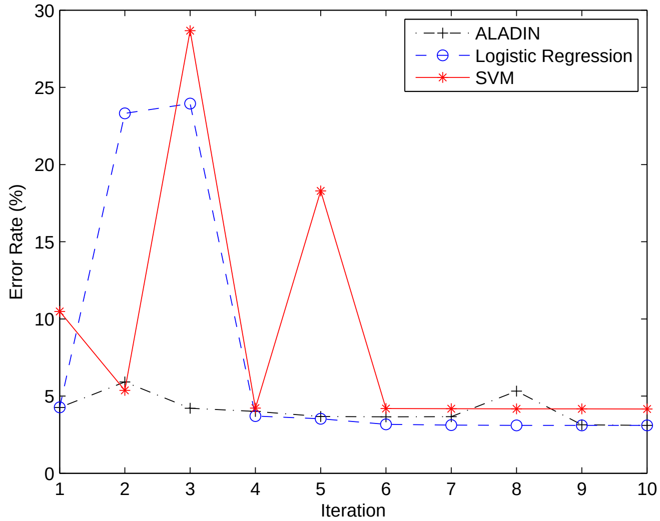

Figure 6: Error rates for the three anomaly detection algorithms using active learning.

6 Conclusions and Future Research

This paper presents ALADIN: an active learning framework for creating an intrusion detection or prevention system. The ALADIN framework can also be used to detect malware from outbound network traffic. In prior work, active learning was used to either improve the accuracy rate of a misuse classifier, or was used to detect new rare categories of traffic. ALADIN combines both of these styles of active learning into one hybrid framework: items provided to an analyst are used for both goals.

Experiments on the 1999 KDD-Cup data set show that ALADIN achieves both goals of high accuracy on known classes, and good detection of unknown rare categories. Because ALADIN is based on logistic regression and naive Bayes, it can scale to process very large log files. We have tested ALADIN on real corporate network firewall logs that contain 13 million events.

Although the current UI provides an effective method for examining anomalies, we are exploring new visualization techniques for efficiently analyzing the logs. In this paper, we have not done feature selection to understand which features improve ALADIN's ability to quickly identify anomalies and improve classification accuracy. In the future, we plan to investigate

True Label	Num Labeled Samples	TP Count	FP Count	Incorrectly Predicted Label	FN Count	FP Rate (%)	FN Rate (%)
normal	477	55745	3060	portsweep	9	6.59	0.0269
				guess_passwd	3		
				ipsweep	3		
neptune	47	20435	0		0	0.00	0.00
smurf	81	19018	0	normal	5	0.00	0.026
back	29	0	0	normal	1973	0.00	100
ipsweep	47	56	3	normal	657	0.003	92.14
satan	39	455	1	normal	45	0.001	9.000
portsweep	54	223	9	satan	1	0.009	0.446

Table 2: False Positive and False Negative Rates after 10 Iterations.

True Label	Num Labeled Samples	TP Count	FP Count	Incorrectly Predicted Label	FN Count	FP Rate (%)	FN Rate (%)
normal	923	55275	2306	back	39	5.11	0.071
neptune	137	20435	0		0	0.00	0.00
smurf	162	18942	0		0	0.00	0.00
back	111	7	39	normal	1884	0.00	99.6
ipsweep	157	593	0	normal	10	0.003	1.66
satan	111	389	0	normal	39	0.001	9.11
portsweep	98	180	0		0	0.00	0.00

Table 3: False Positive and False Negative Rates after 20 Iterations.

feature selection on the ISA data once we have a large number of labeled samples.

As with all algorithms which process extremely large databases, scalability is an issue that requires additional investigation. A tradeoff exists between fast response required for an interactive algorithm such as ALADIN, and processing more data in order to detect additional anomalies. In the current design, we focus on the most recent data in order to catch new malware outbreaks. However if older data is stored, the architecture can be extended so that a separate thread processes the previous data in a reverse time order (i.e. processing the most recent data first and processing the data backwards towards the original data). Several open questions include the following. Does predicting the labels of the older, unlabeled samples affect the performance of the models on the most recent unlabeled data due to the often dynamic nature of malware? How well does the effectiveness of the algorithm scale with large numbers of classes? After long periods of labeling data, a system may have thousands of classes depending on the level of analysis desired by the analysts. Processing scalability should not be an issue since the algorithm can easily be implemented to run in parallel on multiple processors. More investigation is needed to ensure the predictive accuracy and ability to discover anomalies does not degrade as the number of categories increases over time.

References

[1] M. Roesch, "Snort - lightweight intrusion detection for networks," in *13th USENIX Systems Administration Conference*, 1999, pp. 229–238.

[2] W. Lee, S. Stolfo, and K. Mok, "A data mining framework for building intrusion detection models," in *IEEE Symp. on Security and Privacy*, 1999, pp. 120–132.

[3] A. Ghosh, A. Schwartzbard, and M. Schatz, "Learning program behavior profiles for intrusion detection," in *Proc. 1st USENIX Workshop on Intrusion Detection and Network Monitoring*, 1999, pp. 51–62.

[4] K. Sequeira and M. Zaki, "ADMIT: anomaly-based data mining for intrusions," in *Proc. ACM SIGKDD Conf.*, 2002, pp. 386–395.

[5] A. Valdes, "Detecting novel scans through pattern anomaly detection," in *Proc. DISCEX*, 2003, pp. 140–151.

[6] M. Almgren and E. Jonsson, "Using active learning in intrusion detection," in *Proc. IEEE Computer Security Foundations Workshop*, 2004, pp. 88–98.

[7] D. Pelleg and A. Moore, "Active learning for anomaly and rare-category detection," in *Proc. Advances in Neural Information Processing Systems*, 2004, pp. 1073–1080.

[8] D. Endler, "Intrusion detection: Applying machine learning to Solaris audit data," in *Proc. ACSAC*, 1998, pp. 267–279.

[9] L. Portnoy, E. Eskin, and S. Stolfo, "Intrusion detection with unlabeled data using clustering," in *Proc. ACM Workshop on Data Mining Applied to Security*, 2001.

[10] I. Green, T. Raz, and M. Zviran, "Analysis of active intrusion prevention data for predicting hostile activity in computer networks," in *Communications of the ACM*, 2007, pp. 63–68.

[11] Z. Zhang and H. Shen, "Online training of svms for real-time intrusion detection," in *Proc. Int. Conf. on Advanced Information Networking and Application*, 2004, pp. 568–573.

[12] Y. Liao and V. R. Vermuri, "Use of k-nearest neighbor classifier for intrusion detection," in *Computers and Security*, 2002, pp. 439–448.

[13] Z. Liu, J. Almhana, V. Choulakian, and R. McGorman, "Online em algorithm for mixture with application to internet traffic modeling," in *Computational Statistics and Data Analysis*, no. 4, 2005, pp. 1052–1071.

[14] D. Gao, M. K. Reiter, and D. Song, "Behavioral distance for intrusion detection," in *Proc. RAID*, 2006, pp. 63–81.

[15] D. Denning, "An intrusion detection model," in *IEEE Trans. on Software Engineering*, 1987, pp. 222–232.

[16] Q. Yin, R. Zhang, and X. Li, "An new intrusion detection method based on linear prediction," in *Proc. Internation Conf. Computer Security*, 2004, pp. 160–165.

[17] Y. Gu, A. McCallum, and D. Towsley, "Detecting anomalies in network traffic using maximum entropy estimation," in *Internet Measument Conference*, 2005, pp. 345–350.

[18] A. Lazarevic, A. Ozgur, L. Ertoz, J. Srivastava, and V. Kumar, "A comparative study of anomaly detection schemes in network intrusion detection," in *Proceedings SIAM Conference on Data Mining*, 2003.

[19] Z. Anming and J. Chunfu, "Study on the applications of hidden markov models to computer intrusion detection," in *Proceedings World Conference on Intelligient Control*, 2004, pp. 4352–4356.

[20] X. Li and N. Ye, "Decision tree classifiers for computer intrusion detection," in *Parallel and Distrubuted Computing Practices*, 2001, pp. 179–190.

[21] N. Abe, B. Zadrozny, and J. Langford, "Outlier detection by active learning," in *Proc. ACM SIGKDD Conf.*, 2006, pp. 504–509.

[22] G. Schohn and D. Cohn, "Less is more: Active learning with support vector machines," in *Proc. Int'l. Conf. Machine Learning*, 2000, pp. 839–846.

[23] T. Lane, “Machine learning and data mining for computer security: Methods and applications,” in *Semi-supervised learning*, M. Maloof, Ed. Springer-Verlag, 2006.

[24] C. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[25] Microsoft, “Web proxy log fields,” Tech. Rep., 2004, http://www.microsoft.com/technet/isa/2004/help/FW_C_WebLogFields.mspx.

[26] S. Stolfo, “KDD cup 1999 dataset,” UCI KDD repository, Tech. Rep., 1999, http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.