Using Artificial Anomalies to Detect Unknown and Known Network Intrusions

Wei Fan*
IBM T.J.Watson Research
Hawthorne, NY 10532
weifan@us.ibm.com

Matthew Miller, Salvatore J. Stolfo
Computer Science, Columbia University
New York, NY 10027
{mmiller,sal}@cs.columbia.edu

Wenke Lee
College of Computing, Georgia Tech
Atlanta, GA 30332
wenke@cc.gatech.edu

Philip K. Chan
Computer Science, Florida Tech
Melbourne, FL 32901
pkc@cs.fit.edu

Abstract

Intrusion detection systems (IDSs) must be capable of detecting new and unknown attacks, or anomalies. We study the problem of building detection models for both pure anomaly detection and combined misuse and anomaly detection (i.e., detection of both known and unknown intrusions). We propose an algorithm to generate artificial anomalies to coerce the inductive learner into discovering an accurate boundary between known classes (normal connections and known intrusions) and anomalies. Empirical studies show that our pure anomaly detection model trained using normal and artificial anomalies is capable of detecting more than 77% of all unknown intrusion classes with more than 50% accuracy per intrusion class. The combined misuse and anomaly detection models are as accurate as a pure misuse detection model in detecting known intrusions and are capable of detecting at least 50% of unknown intrusion classes with accuracy measurements between 75% and 100% per class.

1 Introduction

Classification and anomaly detection are two common data analysis tasks. *Anomaly detection* tracks events that are inconsistent with or deviate from events that are known or expected. For example, in intrusion detection anomaly detection systems flag observed activities that deviate significantly from established normal usage profiles. On the other hand, classification systems use patterns of well-known classes to match and identify known labels for unlabeled datasets. In intrusion detection, classification of known attacks is also called *misuse detection*.

Anomaly detection systems are not as well studied, explored, or applied as classification systems. Most of the leading commercial intrusion detection systems (IDSs) employ solely misuse detection techniques, which use patterns of "known" attacks to detect intrusions. However, as anecdotes of serious break-ins to major government, military, and commercial sites have shown, our adversaries, knowing that intrusion prevention and detection systems are installed in our networks, will always be attempting to develop and launch "new" attacks. Last year's Distributed Denial-of-Service (DDOS) attacks have caused major disruptions for services provided over the Internet.

In the generation of classification models, training data containing instances of known classes is often available for training (or human analysis) and the goal is simply to detect instances of these known classes. Anomaly detection, however, relies on data belonging to one single class (such as purely *normal* connection records) or limited instances of some known classes with the goal of detecting all unknown classes. It is difficult to use traditional inductive learning algorithms for such a task, as most are only good at distinguishing the boundaries among all given classes of data. In this paper, we explore the use of traditional inductive learning algorithms for anomaly detection by working from the dataset level. We present methods for generating artificial anomalies based on known classes to coerce an arbitrary machine learning algorithm to learn hypotheses that separate all known classes from unknown classes. We discuss the generation of anomaly detection models from pure normal data, and also discuss the generation of combined misuse and anomaly detection models from data that contains known classes. We apply the proposed approaches to network-based intrusions.

*This work was completed when the author was a PhD student at Columbia University

The rest of the paper is organized as follows. Section 2 discusses the motivation for artificial anomaly generation and the different methods. In Sections 3-6, we evaluate our methods using RIPPER [2], an inductive rule learner, trained and tested with the 1998 DARPA Intrusion Detection Evaluation dataset. Section 7 reviews related work in anomaly detection. Section 8 offers conclusive remarks and discusses avenues for future work.

A longer version of this paper can be found at `http://www.cs.columbia.edu/~wfan`

2 Artificial Anomaly Generation

A major difficulty in using machine learning methods for anomaly detection lies in making the learner discover boundaries between known and unknown classes. Since we begin without any examples of anomalies in our training data (by definition of the task of anomaly detection), a machine learning algorithm will only uncover boundaries that separate different known classes in the training data. This behavior is intended to prevent overfitting a model to the training data. Learners only generate hypotheses for the provided class labels in the training data. These hypotheses define decision boundaries that separate the given class labels. To achieve generalization and avoid overfitting, learning algorithms usually do not specify a boundary beyond what is necessary to separate known classes.

Some learners can generate a default classification for instances that are not covered by the learned hypothesis. The label of this default classification is often defined to be the most frequently occurring class of all uncovered instances in the training data. It is possible to modify this default prediction to be *anomaly*, signifying that any uncovered instance should be considered anomalous. It is also possible to tune the parameters of some learners to coerce them into learning more specific hypotheses. However, our experimentation with these methods does not yield a reasonable performance.

The failure of using more specific hypotheses and modifying a model's default prediction has motivated us to propose *artificial anomaly generation* for such a task. Artificial anomalies are injected into the training data to help the learner discover a boundary around the original data. All artificial anomalies are given the class label *anomaly*. Our approach to generating artificial anomalies focuses on "near misses," instances that are close to the known data, but are not in the training data. We assume the training data are representative; hence near misses can be safely assumed to be anomalous. Our artificial anomaly generation methods are independent of the learning algorithm, as the anomalies are merely added to the training data.

Input: D; Output: D'.

1. let F = set of all features of D.
2. let V_f = set of unique values of some feature $f \in F$.
3. let $D' = \emptyset$.
4. for each $f \in F$:
   - let $countV_{max}$ = the number of occurrences of the most frequently occurring value in V_f.
   - for each $v \in V_f$:
     - let $countV$ = the number of occurrences of v in D.
     - loop i : $countV < i \leq countV_{max}$:
       * let d = a randomly chosen datum $d \in D$.
       * let v_f = the value of feature f for d.
       * replace v_f with a randomly chosen value v' s.t. $v' \neq v \wedge v' \neq v_f$ to create d'.
       * $D' \leftarrow D' \cup \{d'\}$.
5. return D'.

Note: The algorithm can be modified to take a factor, n, and produce $n \times |D|$ artificial anomalies.

Figure 1. Distribution-Based Artificial Anomaly (DBA2) Generation Algorithm

2.1 Distribution-based Artificial Anomaly

Since we do not know where the exact decision boundary is between the known and anomalous instances, we assume that the boundary may be very close to the existing data. To generate artificial anomalies close to the known data, a useful heuristic is to randomly change the value of one feature of an example while leaving the other features unaltered.

Some regions of known data in the instance space may be sparsely populated. We compare sparse regions to small islands, and dense regions to large islands, in an ocean. To avoid overfitting, learning algorithms are usually biased towards discovering more general hypotheses. Since we only have known data, we want to prevent hypotheses from being *overly* general when predicting these known classes. That is, sparse regions may be grouped into dense regions to produce singularly large regions covered by overly general hypotheses. Using our analogy, small islands are unnecessarily grouped into large islands to form apparently larger islands. It is possible to produce artificial anomalies around the edges of these sparse regions and coerce the learning algorithm to discover the specific boundaries that distinguish these regions from the rest of the instance space. In other words, we want to generate data that will amplify these sparse regions.

Sparse regions are characterized by infrequent values of individual features. To amplify sparse regions, we proportionally generate more artificial anomalies around sparse regions depending on their sparsity using a proposed algorithm presented in detail in Figure 1. Assuming that the value v of some feature f is infrequently present in the dataset, we calculate the difference between the number of occurrences of v, $countV$, and the number of occurrences of the most frequently occurring value v_{max} of the given feature, $countV_{max}$. We then randomly sample $countV_{max} - countV$ data points from the training set. For each data point d in this sample, we replace the value of feature f, v_f, with any v' such that $v' \neq v \wedge v' \neq v_f$ to generate an artificial anomaly, d'. The learning algorithm used will then specifically cover all instances of the data with value v for feature f. This anomaly generation process is called *distribution-based* artificial anomaly generation or DBA2, as the distribution of a feature's values across the training data is used to selectively generate artificial anomalies.

2.2 Filtered Artificial Anomalies

In the above discussion, we assume that artificial anomalies do not intersect with any known data. We could always check for collision with known instances, but this is a very expensive process. Another approach is to filter artificial anomalies with hypotheses learned on the original data. We use the training set plus an initial generation of artificial anomalies to learn a model. We then evaluate this model over previously generated artificial anomalies and remove any anomalies classified as some known class. This process is repeated until the size of the the set of artificial anomalies remains relatively stable.

3 Experimental Setup

For generation of our models, we have chosen to use RIPPER [2], an inductive decision tree learner. RIPPER can learn both *unordered* and *ordered* rulesets. The use of these types of rulesets in IDSs has been discussed in our previous work [3]. In all reported results, unless clearly stated, we always use an *unordered* RIPPER ruleset and inject an amount of distribution-based artificial anomalies equal to the size of the training set (i.e., we use DBA2 with $n = 1$).

Our experiments use data distributed by the 1998 DARPA Intrusion Detection Evaluation Program, which was conducted by MIT Lincoln Lab (available from the UCI KDD repository as the 1999 KDD Cup Dataset). We use the same taxonomy for categorization of intrusions as was used by the DARPA evaluation. This taxonomy places intrusions into one of four categories: denial of service (DOS), probing (PRB), remotely gaining illegal remote access to a local account or service (R2L), and local user gaining illegal root access (U2R). The DARPA data were gathered from a simulated military network and includes a wide variety of intrusions injected into the network over a period of 7 weeks.

Table 1. Intrusions, Categories and Sampling

U2R	R2L	DOS	PRB
buffer_overflow	ftp_write	back	ipsweep
loadmodule	guess_passwd	land	nmap
multihop	imap	neptune	portsweep
perl	phf	pod	satan
rootkit	spy	smurf	
	warezclient	teardrop	
	warezmaster		

Table 2. Anomaly Detection Rate and False Alarm Rate of Pure Anomaly Detection

$\%a_{ttl}$	94.26
%far	2.02

(a)

	%a		%a
buffer_overflow	100.00√	ftp_write	50√
loadmodule	66.67√	guess_passwd	100.00√
multihop	57.14√	imap	83.33√
perl	-	phf	100.00√
rootkit	10.00	spy	-
		warezclient	64.25√
		warezmaster	80.00√
U2R	47.06√	R2L	66.67√
back	100.00√	ipsweep	-
land	75.00√	nmap	-
neptune	80.52√	portsweep	4.81
pod	9.62	satan	0.32
smurf	99.94√		
teardrop	-		
DOS	94.31√	PRB	1.34

√: significant or %a $\geq 50\%$

(b)

The data were then processed into connection records using MADAM ID [9]. A 10% sample was taken which maintained the same distribution of intrusions and normal connections as the original data (this sample is available as kddcup.data.10% from the UCI KDD repository). We used 80% of this sample as training data and left the remaining 20% unaltered to be used as test data for evaluation of learned models.

4 Pure Anomaly Detection

For pure anomaly detection, we learned a model using all available *normal* connections augmented by DBA2 anomalies generated from these normal connections. We refer to this collection as dataset_0. RIPPER learns a large number of rules for both *normal* and *anomaly* from this dataset.

4.1 Results

Table 2 shows the results of the pure anomaly detection model. We use detection rate and false alarm rate to evaluate performance. Anomaly detection rate, or percentage of occurrences of some unknown intrusion i that are detected as anomalies, is defined as $\%a_i = \frac{|A \cap W_i|}{|W_i|} \times 100\%$, where A is the set of all predicted anomalies and W_i is the set of occurrences of label i in the dataset. We omit the subscript i where the meaning is clear from the context. Similarly, we calculate cumulative anomaly detection rate over all unknown intrusions ($\%a_{ttl}$) and cumulative anomaly detection rate over different categories of unknown intrusions (such as $\%a_{u2r}$). Also, we measure the false alarm rate ($\%far$) of anomalous classifications. This is the percentage of predicted anomalies that are *normal* connections, and is defined as $\%far = \frac{|A \cap W_{normal}|}{|A|} \times 100\%$. If a measurement has a value of 0, we represent it with "$-$" to enhance readability of the presented tables.

The cumulative anomaly detection rate over all intrusions and false alarm rate are shown in Table 2(a). The anomaly detection model successfully detects 94.26% of all anomalous connections in the test data and has a false alarm rate of 2.02%. To examine the performance for specific intrusion classes and categories, the anomaly detection rate ($\%a$) for each class and category is shown in Table 2(b). The anomaly detection model is capable of detecting most intrusion classes, even though there are no intrusions at all in the training set. A total of 17 out of 22 intrusion classes (all non-null measurements) are detected as anomalies. For 13 out of 22 intrusions (all entries highlighted by $\surd$), the proposed method catches at least 50% of all occurrences. There are 3 intrusions (*guess_passwd, buffer_overflow* and *phf*) that our approach is capable of detecting perfectly (i.e., $\%a = 100\%$). These 3 intrusions belong to the more harmful U2R and R2L categories. The anomaly detection rates of all 4 categories of intrusions indicate that, in general, each category is successfully detected. In 3 out of 4 categories (U2R, R2L and DOS), the model detects more than 50% of all intrusion occurrences of that category, and it is important to note that these three categories are potentially more damaging than PRB.

5 Combined Misuse and Anomaly Detection

Pure anomaly detection might still have high false alarm rate; the boundaries implied by artificial anomalies can be sharpened by real intrusions. Separate modules for anomaly and misuse detection will not be as efficient as one single module that detects misuse and anomaly in the same time. All these have motivated us to explore the use of artificial anomalies for such as task.

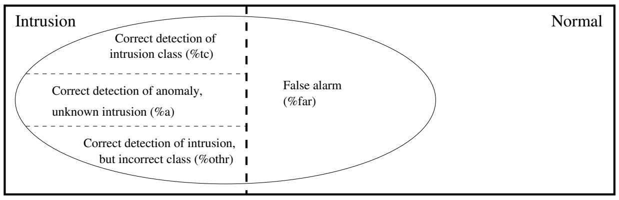


Figure 2. Relationship of Metrics

We learn a single ruleset for combined misuse and anomaly detection. The ruleset has rules to classify a connection to be *normal*, one of the known intrusion classes, or *anomaly*. In order to evaluate this combined approach, we group intrusions together into 13 small *clusters* (details can be found in the longer version of the paper). We create datasets (dataset$_i$, $1 \leq i \leq 12$)[1] by incrementally adding each cluster$_i$ into the *normal* dataset and re-generating artificial anomalies. This is to simulate the process of the invention of new intrusions and their incorporation into the training set. We learn models that contain misuse rules for the intrusions that are "known" in the training data, anomaly detection rules for unknown intrusions in left-out clusters, and rules that characterize normal behavior.

Each cluster contains intrusions that require similar features for effective detection. Clusters should not be confused with the attack categories in our taxonomy. One example of a cluster contains the following intrusions: *buffer_overflow, loadmodule, perl* and *rootkit*. These intrusions all attempt to gain unauthorized root access to a local machine and require features such as *root_shell* (whether a root shell is obtained) and *su_flag* (an indication of whether the `su root` command been used, in any of its derivations). Some clusters have completely disjoint feature sets, yet some intersect slightly. A model that is trained to detect intrusions from one cluster may have difficulties detecting intrusions from another cluster. For clusters with intersecting feature sets, we hope that a model learned using training instances of intrusions from some cluster may be used to detect intrusions of other clusters as anomalies.

Before explaining our results, we must define some frequently used terms. Any intrusion class that appears in the training data is a *known intrusion*. Similarly, any intrusion class not in the training set is an *unknown intrusion*, or *true anomaly*. *Predicted anomalies* include *true anomalies* and may also include instances of *known intrusions* and *normal*. We use *anomaly* to refer to *predicted anomaly* where our intention is clear from context.

5.1 Results

Our results for combined misuse and anomaly detection methods are shown in Figures 3-5 and Tables 3-4. Based on the outcome of detection, we calculate the following measurements: true class detection rate ($\%tc$), anomaly detection rate ($\%a$), and other class detection rate ($\%othr$). The relationship of these metrics is shown in the Venn diagram in Figure 2. The outside rectangle represents the set of data to be evaluated and the inside ellipse depicts the set of alarms generated by our learned detection models. True class detection rate measures the percentage of connection class i (*normal* or intrusions) being correctly predicted as its true class, and is defined as $\%tc = \frac{|P_i \cap W_i|}{|W_i|} \times 100\%$, where P_i is the set of predictions with label i. $\%a$ is defined as in Section 4.1, but it can be measured for both known and unknown intrusions or intrusion categories that are predicted as anomalies. Other class detection rate, or the rate of detection as another class of intrusion, is the percentage of occurrences of intrusion i that are detected as some class of intrusion other than its true label or anomaly, and is defined as $\%othr = \frac{\sum_{i' \neq i} |P_{i'} \cap W_i|}{|W_i|} \times 100\%$. Additionally, total detection rate is defined as $\%ttl = \%tc + \%a + \%othr$.

We examine our results from several perspectives. First, it is important to see how the proposed method influences the true class detection rate ($\%tc$) of known intrusions. Ideally, using artificial anomalies should allow anomaly detection to classify true anomalies without degrading the performance of detecting known intrusions with misuse rules. Second, we evaluate the effectiveness of detecting true anomalies. Third, we examine whether anomalous classification can compensate for low detection rates of known intrusions by misuse rules. Finally, we show the false alarm rates of anomaly detection in different test settings.

True Class Detection The true class detection rates of models learned with and without DBA2 are shown in Figure 3. The x-axis shows each dataset, ranging from dataset$_1$ to dataset$_{12}$ as explained in Section 5. We see that the curves for R2L, DOS and PRB are indistinguishable. The difference in U2R curves is reasonably small as well[2]. This observation shows that the proposed DBA method does not deteriorate the effectiveness of detecting particular categories of known intrusions. Next, we examine the efficacy of our approach in detecting anomalies.

True Anomaly Detection In analysis, we consider an anomaly detection model to be *significant* for a particular

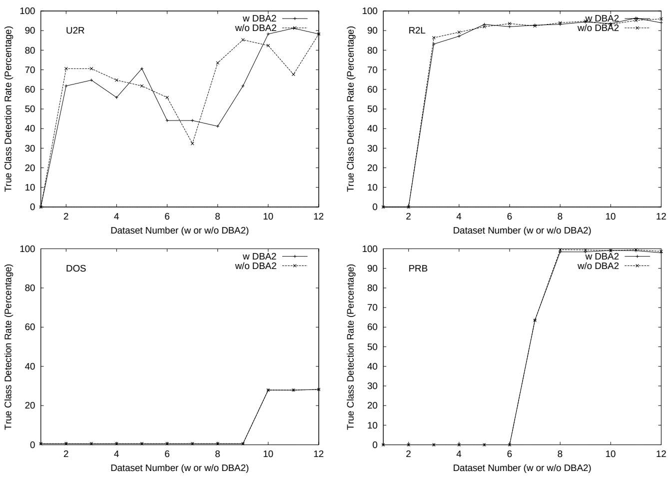


Figure 3. Comparison of True Class Detection Rate (%tc) of Datasets w. and w/o DBA2

Table 3. Percentage of Significant True Anomaly Detections

Dataset	0	1	2	3	4	5	6
Anomaly Types	22	21	17	14	13	12	11
Significant	13	15	14	10	8	9	7
%	59	71	82	71	62	75	64

Dataset	7	8	9	10	11	12
Anomaly Types	10	7	6	5	4	2
Significant	8	6	5	5	3	2
%	80	86	83	100	75	100

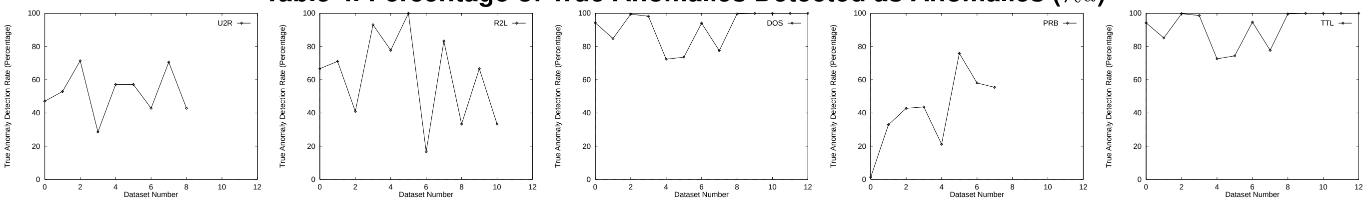
detection class if $\%a \geq 50.00\%$. Across 12 experiment settings, there are 122 truly anomalous cases (or 122 intrusions types in 12 experiment settings not included in the training data), and among them, 92 are significant; this is more than 75%. For each experiment setting, the percentage of cases that are significant is shown in Table 3; most are at least 70%.

Next, we study the effectiveness of anomaly detection on different categories of true anomalies. We measure anomaly detection rate ($\%a$) of true anomalies in each intrusion category and all true anomalies (TTL). The results are presented in Table 4. As shown in the upper rightmost curve and the last row of the table under "TTL," the true anomaly detection rate for all true anomalies remains relatively constant as we inject more clusters of intrusions. The curves for U2R, R2L and PRB categories are more bumpy than DOS because the anomaly detection model catches more in one category and fewer in the others.

[1] We leave out the 13^{th} cluster for testing.

[2] Note that there are only 34 U2R instances in the test data. Disagreements in just a few examples can make significant difference in $\%tc$.

Table 4. Percentage of True Anomalies Detected as Anomalies ($\%a$)



Dataset	0	1	2	3	4	5	6	7	8	9	10	11	12
U2R	47.06	52.94	71.43	28.57	57.14	57.14	42.86	70.59	42.86	na	na	na	na
R2L	66.67	71.08	40.96	93.10	77.78	100.00	16.67	83.33	33.33	66.67	33.33	0.00	0.00
DOS	94.31	84.82	99.53	98.22	72.37	73.61	94.08	77.52	99.65	99.96	99.98	99.95	100.00
PRB	1.34	32.89	42.79	43.64	21.15	75.92	58.07	55.44	na	na	na	na	na
TTL	94.26	84.78	99.46	98.18	72.33	73.57	94.02	77.46	99.65	99.96	99.97	99.95	100.00

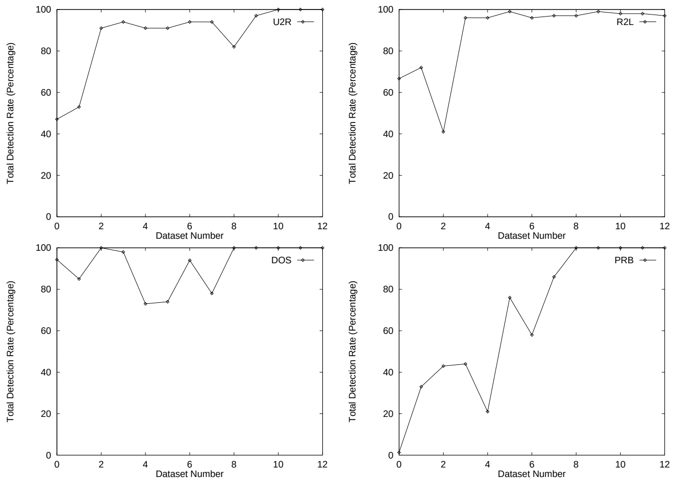


Figure 4. Total Detection Rate ($\%ttl$)

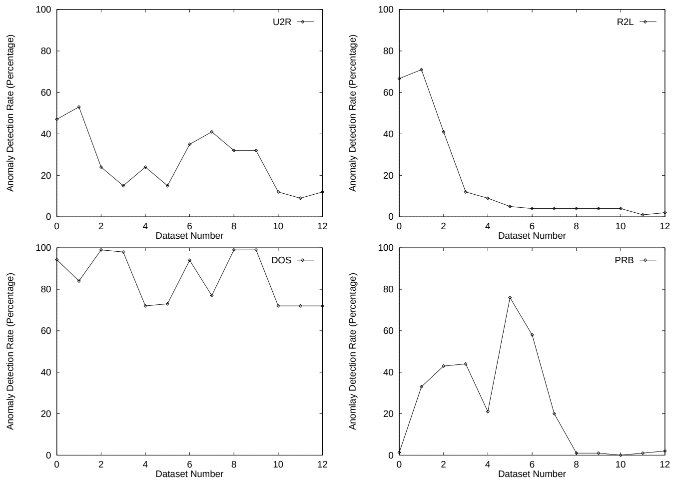


Figure 5. Percentage of Known Intrusions and True Anomalies Detected as Anomalies (%a)

Known Intrusions Detected As Anomalies It is interesting to determine if the proposed approach can prove effective in detecting un-classified known intrusions as anomalies. We consider an anomaly detection method to *significantly compensate* for misuse detection if either the anomaly detection increases the total rate of detection to nearly 100% ($\%tc + \%a \simeq 100\%, \%tc < 100\%$) or $\%a \geq 0.25 \times \%tc$ when $\%tc$ is very low. In 12 experiment settings, there are 88 cases that are candidates for compensation (i.e., $\%tc < 100\%$). Among them, 76 cases (or 86%) are significantly compensated by being detected as anomalies. In 5 of the remaining 12 cases, the intrusions are detected as some other intrusion, leaving no room for anomaly detection to provide any compensation. In Figure 4, we show the total percentage of detection ($\%ttl$) for all 4 categories of intrusions. As expected, there is a general trend of increase. Comparing Figure 4 with Figure 3, we can see a signifi-can higher total detection rate than true class detection rate, which is mostly attributed to known intrusions being detected as anomalies.

Overall Performance In the above discussion, we have covered the performance of true anomaly detection and misuse detection compensation. We now examine the combined overall performance of detecting both true anomalies *and* known intrusions. The results over all 4 intrusion categories are shown in Figure 5. As expected, there is a general trend of decrease in $\%a$ when the datasets are augmented with more clusters of intrusions. This is caused by the fact that we have learned misuse rules for more intrusions, leaving less room for these intrusions to be detected as anomalies. The shape of the anomaly detection rate curves is somewhat inversely related to their respective true class detection curves in Figure 3. This relationship is explained by the observation that $1 - \%tc$ is an indication of the amount of additional detection that anomaly detection can provide. These decreasing and inverse relationships apply throughout the U2R, R2L and DOS curves, and can be seen in the PRB curves for dataset$_{6-12}$. As we see in Figure 3, as more intrusion clusters are used to augment the *normal* data, the true class detection rates for both U2R and R2L increases and leave less room for anomaly detection to compensate. This explains the generally decreasing tendency of U2R and R2L $\%a$ curves. For DOS attacks, the true class detection rate only rises from 0 to 30% after dataset$_9$, and there is still sufficient room for compensation by anomaly detection — this explains the flatness of the DOS $\%a$ curve in Figure 5. For PRB, the rise in $\%tc$ takes place after dataset$_6$, which is also when we see the complimentary decrease in $\%a$ for dataset$_{6-12}$. The slight bumpiness of the U2R $\%a$ curve is due to the inverse bumpiness of U2R $\%tc$ curve in Figure 3. The slight bumpiness of the DOS and PRB curves are most likely caused by insufficient feature values available when learning a decision boundary for anomalies.

The false alarm rate of anomalous classifications are uniformly below 0.40% (details shown in the longer version of this paper). This confirms that nearly all detected anomalies are either true anomalies or known intrusions. Additionally, the $\%tc$ rates of *normal* connections (or *normal* correctly classified as *normal*) are over or near 99.00%. These observations show the utility of the anomaly detection approach in building highly precise models.

Effects of Cluster Ordering We performed two tests to verify that our results are not influenced by the order in which clusters are added to the training sets. One test is to reverse the cluster ordering as in the previous section, and the other one is a random ordering totally different from the original or second orderings. The results have confirmed that our results are indeed not influenced by cluster order.

6 Additional Issues

We have experimented with different amounts of injected artificial anomalies ($n = 1.5$ or 2). The general trend is that as we increase the injection amount, $\%tc$ of *normal* connections decreases slightly and $\%far$ increases slightly. When the amount of injected artificial anomalies increases, there are more artificial anomalies than normal connections in the training data and the learning algorithm tends to generate more *anomaly* rules. In general, however, the proposed algorithm is not sensitive to the amount of artificial anomalies in the training data.

We experimented with the use of other forms of RIPPER rulesets ($+freq$ and $given$, both of which are $ordered$ rulesets). $+freq$ rulesets classify connections in order of increasing frequency followed by *normal*, with a default classification of *anomaly*. For $given$, we used the following rule order: *normal, anomaly* and alphabetically ordered intrusion classes (essentially arbitrary). The results gathered for the use of a $+freq$ ruleset are very close to the detailed results given for our $unordered$ rulesets. It is interesting to observe that the $given$ rulesets are similar to $unordered$ rulesets at later datasets (when more than 3 clusters of intrusions are added to the normal data). However, in the first 2 datasets (dataset$_1$ and dataset$_2$), the anomaly detection is more likely to classify known intrusions as anomalies. This is due to the fact that anomaly rules appear before intrusion rules.

We experimented with the filtering method for DBA2 as proposed in Section 2.2. The generated artificial anomalies were filtered 3 times with a resulting reduction of between 1% and 3% each time. We did not see any significant improvement in performance using this method. We also experimented with filtering anomalies generated using the naive approach and observed that no artificial anomalies were removed at all. The main conclusion to be drawn from these filtering experiments is that most artificial anomalies are truly anomalous, and do not collide with known training data.

7 Related Work

SRI's IDES [6] measures abnormality of current system activity from the probability distributions of past activities. The activities they monitored are host events (e.g., CPU utilization and file accesses); in our work, we monitor network events. Forrest et al. [4] record frequent subsequences of system calls that are used in the execution of a program (e.g., `sendmail`). Absence of subsequences in the current execution of the same program from the stored sequences constitutes a potential anomaly. Lane and Brodley [8] used a similar approach but they focused on an incremental algorithm that updates the stored sequences and used data from UNIX shell commands. Lee [9], using a rule learning program, generated rules that predict the current system call based on a window of previous system calls. Abnormality is suspected when the predicted system call deviates from the actual system call. Ghosh and Schwartzbard [5] proposed using a neural network to learn a profile of normality. Similar to our approach, random behaviors are generated to represent abnormality for training purposes. Unlike our approach, each of their input features is a distance value from an exemplar sequence of BSM [14] events. This study is one of the first attempts in applying machine learning algorithms to network events for anomaly detection.

Algorithms for anomaly detection and misuse detection have traditionally been studied separately. In SRI's EMERALD [12], anomaly and misuse detection algorithms are encased in separate system components, though their output responses are correlated to generate alarms by the *resolver*. Ghosh and Schwartzbard [5] applied neural networks to both anomaly and misuse detection and compared their relative performance. One of our unique goals in this paper is to study the combination of anomaly and misuse detection in one model to improve overall performance.

We are not aware of closely related work in the generation of training data belonging to an unknown opposite class. Given unlabeled instances, Nigam et al. [13] assigned labels to them using a classifier trained from labeled data and put them in the training set for another round of training. In a skewed distribution scenario, Kubat and Matwin [7] attempted to remove majority instances too close to and too far from the decision boundary. Maxion and Tan [11] used conditional entropy to measure the regularity in the training set and have shown that it is easier to detect anomalies for data with high regularity. Lee and Xiang [10] also applied entropy to determine how hard it is to learn a model of normality and abnormality. Chang and Lippman [1] applied voice transformation techniques to add artificial training talkers to increase variabilities.

8 Conclusion and Future Work

Recent hacker activity has made evident the importance of network-based intrusion detection. Anomaly detection of unknown intrusions is an important and difficult area of IDS. In this paper, we studied the problems of using artificial anomalies to detect unknown and known network intrusions. We proposed a distribution-based anomaly generation algorithm that has proven effective in building anomaly and combined misuse and anomaly detection models that successfully detect known and unknown intrusions.

One assumption of DBA2 is that each dimension (i.e., feature) can be treated individually. In other words, we examine and generate anomalies "dimension by dimension." A possible variation of the algorithm could consider multiple dimensions concurrently or give each dimension a different weight depending on its importance.

References

[1] Eric Chang and Richard Lippmann. Using voice transformations to create additional training talkers for word spotting. In Tesauro et al, editor, *Advances in Neural Processing Systems 7*. MIT Press, 1995.

[2] William Cohen. Fast effective rule induction. In *Proceedings of Twelfth International Conference on Machine Learning (ICML-95)*, pages 115–123. Morgan Kaufman, 1995.

[3] Wei Fan, Wenke Lee, Salvatore Stolfo, and Matthew Miller. A multiple model approach for cost-sensitive intrusion detection. In *Proceedings of Eleventh European Conference on Machine Learning (ECML-00)*, Barcelona, Spain, May 2000.

[4] Stephanie Forrest, Steven A. Hofmeyr, Anil Somayaji, and Thomas A. Longstaff. A sense of self for UNIX processes. In *Proceedings of IEEE Symposium on Security and Privacy 1996*, 1996.

[5] Anup K. Ghosh and Aaron Schwartzbard. A study in using neural networks for anomaly and misuse detection. In *Proceedings of USENIX Security Symposium 1999*, 1999.

[6] Harold Javitz and Alfonso Valdes. The SRI IDES statistical anomaly detector. In *Proceedings of IEEE Symposium on Security and Privacy*, page 1991, 1991.

[7] Miroslav Kubat and Stan Matwin. Addressing the curse of imbalanaced training sets: One sided selection. In *Proceedings of Fourteenth International Conference on Machine Learning (ICML-97)*, pages 179–186. Morgan Kaufmann, 1997.

[8] Terrane Lane and Carla Brodley. Approaches to online learning and concept drift for user identification in computer security. In *Proceedings of Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98)*, pages 259–263, 1998.

[9] Wenke Lee. *A Data Mining Framework for Constructing Features and Models for Intrusion Detection Systems*. PhD thesis, Columbia University, June 1999.

[10] Wenke Lee and Dong Xiang. Information-theoretic measures for anomaly detection. In *The 2001 IEEE Symposium on Security and Privacy*, Oakland, CA, May 2001.

[11] Roy A Maxion and Kymie M.C. Tan. Benchmarking anomaly-based detection systems. In *International Conference on Dependable Systems and Networks*, pages 623–630, June 2000.

[12] Peter G. Neumann and Philip A. Porras. Experiments with EMERALD to date. In *Proceedings of 1999 USENIX Workshop on Intrusion Detection*, 1999.

[13] Kamal Nigam, Andrew McCallum, Sebastian Thrun, and Tom Mitchell. Learning to classify text from labeled and unlabeled documents. In *Proceedings of Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, 1998.

[14] SunSoft. *SunSHIELD Basic Security Module Guide*. SunSoft, Mountain View, CA, 1995.