

Ensemble-based Adaptive Intrusion Detection

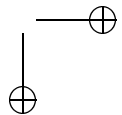
Wei Fan^{*}, and *Salvatore J. Stolfo*[†]

Abstract

Intrusion detection is an essential component of computer security mechanisms. Intrusion detection systems (IDSs) need to efficiently and accurately adapt to incorporate new knowledge of previously unseen *classes* of attacks (different from incremental learning) that are constantly invented to prevent any further damage as early as possible. Learning a completely new detection model from both known attacks and new unknown attacks is usually very slow due to the complexity of the problem and large size of the dataset. There isn't much previous research to address this issue. In this paper, we propose an "ensemble-based" method to efficiently learn a light weight model from audit data of new attack patterns that is then "attached" to an existing previously learned model by a decision rule system. Our method solves the problem of fast training and efficient model deployment that prevents the damage of new types of intrusions at its earliest stage. Several configurations varying in the form of the underlying model and decision rules are explored. The training cost of this method is significantly less than re-training a monolithic model from both new and old training data. Empirical studies show the ensemble-based method has comparable accuracy as the monolithic detector, but the model generation time is 150 times faster. This quick learning time provides an opportunity to deploy new models rapidly to thwart damage of both new and old classes of attacks, which can be replaced later with an updated and better monolithic model as time and resources permit.

^{*} IBM T.J.Watson Research, email address weifan@us.ibm.com. This work was completed when the author was a PhD student at Columbia University. The Intrusion Detection Project was funded by DARPA

[†] Columbia University, Computer Science Department, email address sal@cs.columbia.edu



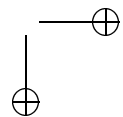
1 Introduction

As the Internet plays an increasingly important role in our society, e.g., the infrastructure for E-Commerce and Digital Government, criminals and enemies have begun devising and launching sophisticated attacks motivated by financial, political, and even military objectives. We must ensure the security, i.e., confidentiality, integrity, and availability, of our network infrastructures. *Intrusion detection* is the process of identifying and responding to malicious activity aimed at compromising computer and network security [2]. It is a critical component of the defense-in-depth security mechanisms, which also include: security policy, vulnerability scanning and patching, access control and authentication, encryption, program wrappers, firewalls, and intrusion tolerance.

Intrusion detection systems need to efficiently and accurately adapt to incorporate new knowledge of previously unseen classes of attacks (different from incremental learning as discussed below) that seem to be constantly invented. This task can be accomplished either by hand coding new attack patterns and inserting into existing models which are then broadly distributed, or by some automatic means of learning about new classes of attacks that are then incorporated into existing deployed models. The approach we have taken to IDS is to employ data mining methods that analyze datasets including attacks and normal usage patterns to learn models of attacks as well as anomalies. The obvious benefit of automated intrusion detection is the efficiency of both training and deployment. 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, one important problem for intrusion detection is the detection of these new types of attacks. Last year’s Distributed Denial-of-Service (DDOS) attacks have caused major disruptions for services provided over the Internet. Once knowledge about new attacks are collected, they need to be quickly incorporated into existing detection systems to prevent any further damage of the new attacks as early as possible. However, re-training a model for both existing and new attacks are usually very slow in nature due to the complexity of the learning problem and large size of the dataset. By the time a new detection model is ready, the new type of attack may have already caused significant damage. There isn’t much previous research to address this issue. In this paper, we explore an ensemble-based method to solve the efficiency problem.

We develop an ensemble-based method that efficiently learns a model from data about new attacks only, and then attach or “plug-in” this new attack detection model into existing model that only detects old attacks. The combined model detects both existing and new attacks. The training time of the new attack detection model is significantly less than re-generating a completely new single model for all known attacks from scratch. The method is straightforward to implement and verify.

There are two major steps to generate the adaptive ensemble. The original training set usually contains *attack* data and *normal* connection data to model specific known attacks. First, we demonstrate a method to convert this “two class”



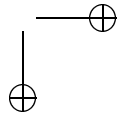
problem into a new one to detect “three classes” of connections: *attack* (a known attack or misuse pattern), *normal* (the connection appears to be entirely a normal connection) and unknown *anomaly* (a connection is not classified as either a known attack or normal, hence its status is unknown). We refer to the “old model” trained from old attack data as H_1 . H_1 classifies a connection event to be one of three classes, *normal*, *attack* or *anomaly*.

After the new attack data is collected, a new model H_2 is trained to classify the new type of attack only. Since the training of H_2 only uses relatively small training set without any of the old data about existing attacks, it is very efficient. Given both the old detector, H_1 , and the new detector, H_2 , at the second step, we show a simple rule-based approach to plug in the new light weight model, H_2 , of a newly discovered attack into H_1 . At detection time, all connection events are filtered by H_1 first; when H_1 predicts an *anomalous connection*, the connection event will be consulted by H_2 to further decide if it is the new attack or just indeed true anomalies. Thus two models are integrated, with the intent that the model update is fast and easily deployed. Predictions by this ensemble-based method are produced under several configurations and decision rules. In reality, H_2 can be plugged into H_1 on the fly at anytime without disrupting the detection by H_1 . Usually, new attack data is not collected at one time, but through a period of time; H_2 can always be re-trained and replaced at anytime when more new attack data are intercepted.

We have evaluated the proposed ensemble-based adaptive learning method on DARPA 1998 Intrusion Detection Evaluation Dataset. We have found that the method can reduce the training time by as much as 150 times without sacrificing any prediction accuracy. The significant time saved by using the ensemble greatly reduces the amount damage caused by new attacks.

Difference from previous work Previous work on ensemble of classifiers mainly focus on improving accuracy and scalability. In [9, 11], Dietterich and Fan review various existing methods on how to generate and combine base classifiers to form an ensemble. Yet, there is very little known research on using ensembles to solve “adaptive learning problems,” where *a new class of data* need to be recognized by the existing model and training efficiency is strictly demanded. In this paper, we explore a hierarchical approach to attach a new attack detection model to the existing model in order to be able to classify both existing and new classes of intrusions. In most of the previously proposed “flat” combining methods, all the base classifiers’ outputs are combined and used. In our “hierarchical” approach, the first classifier is always called; only when its prediction belongs to a set of classes, the second classifier will be used and its prediction will be considered. The problem of adaptive learning differs significantly from incremental learning. The assumption of incremental learning is that class labels are fixed, but data points about these given classes are assimilated one at a time. However, in our case, a new or a few completely new classes of data become available and the model is required to be able to classify them without compromising its ability to classify existing classes.

This paper is organized as follows. Section 2 briefly reviews the algorithm that converts the two-class problem into a three-class problem. In Section 3, we





Curved line is the border of training space.
 The area enclosed in the curved line is the training space.
 × are artificial anomalies (more at sparse regions).

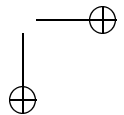
Figure 1. *Artificial Anomalies*

propose four configurations of ensembles. Sections 4 and 5 discuss the experiment. Section 6 reviews related work. The paper concludes in Section 7.

2 Distribution-based Artificial Anomaly Generation

The Distribution-based Artificial Anomaly or DBA2 algorithm is fully described in [12]. The algorithm forces a typical machine learning algorithm to generate a “three” class detector, *attack*, *normal* and *anomaly*, from training data initially only labelled with *attacks* and *normal* data. This is accomplished by adding artificially generated anomalous data to the original training set. The DBA2 algorithm solves the problem in the first step to compute an adaptive ensemble.

The typical machine learning algorithm analyzes features in a labelled training dataset to output a classifier that distinguishes one class of data from another. The outcome is a set of feature tests that assigns an unlabelled datum to one of a number of given classes. If a datum with a completely new class is given to the classifier, the classifier will have no other choice and the predicted class will be one of the previously known classes, which is obviously wrong. For example, if a classifier trained to distinguish between “table” and “chair” is given a datum that actually describes a “wardrobe,” the classifier will mistakenly declare it as a table or a chair. Ideally, the correct answer should be “don’t know” or *anomaly*. To solve this problem, the DBA2 algorithm works at the dataset level. The instance space is completely defined by the Cartesian product of feature values of all features. The training data or the data of all known attacks and normal connections are only a subspace of the instance space. We call this subspace “training space.” The empty space (the instance space minus the training space) contains data points that are not in the training data and thus for which we have no ground truth. Many of them are potentially “anomalies.” In order for the typical machine learner to distinguish the training data points from anomalies, it needs to find the border between the training space and the empty space. Unlabelled data that falls within the training space will be given a label, such as normal or attack; those outside will be predicted as anomalies.



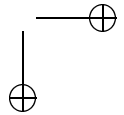
Input: D ; Output: D' .

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' .

Figure 2. *Distribution-Based Artificial Anomaly (DBA2) Generation Algorithm*

To find out where this border may be, the DBA2 algorithm uses the heuristic of “near misses,” or instances of data that are “close” to labelled data, but that are not present in the original training data. Near misses or artificial anomalies are computed from the training set. An example artificial anomaly is a datum that has the same feature values equal to the corresponding feature values of a datum in the training set “except for one differing feature value.” This heuristic has been employed in early work in AI on different contexts and we apply the same concepts here as well. The instance space has both sparse and dense regions. Sparse regions are characterized by infrequent feature values. We compare sparse regions to small islands, and dense regions to large islands. Sparse regions tend to be grouped into dense regions to produce singularly larger regions by inductive learning programs. The DBA2 algorithm highlights sparse regions by computing more artificial anomalies whose number is proportional to the area’s sparsity. In Figure 1, we illustrate the ideas of training space, border of training space and artificial anomalies. When the original training data is augmented with artificial anomalies to delineate the border between *known* and *unknown*, a classifier trained from this augmented dataset will classify an instance to either belong to a known class or the unknown class called “anomaly”. For example, the exemplar classifier that only distinguishes “table” and “chair” will classify an instance of “wardrobe” to be “don’t know”.

The DBA2 algorithm is summarized in Figure 2. It uses the frequency of features to discover sparse regions in the training space and compensate their sparsity with proportional number of artificial anomalies. 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 to train on the augmented dataset will then specifically cover all instances of the data with value v for feature f . For example, a rule learner may produce the following rule to cover all instances with value v , **if** $f = v \wedge \dots$ **then** *known*. 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.

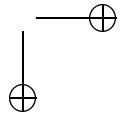
In [12], the method is applied to both pure anomaly detection and combined misuse detection and anomaly detection problems. It has been shown that the majority of anomalies can be successfully detected. A classifier trained from a dataset augmented in this fashion with artificial anomalies predicts all known classes of attacks, normal connections as well as anomalies. Knowing how to convert a two-class (*normal* and *attack*) problem into a three-class problem (*normal*, *attack* and *anomaly*), we next discuss the ensemble-based approach to adapt existing model to detect the new attacks.

3 Ensemble-based Adaptive Learning

The idea of ensemble-based adaptive learning is illustrated in Figure 3. H_1 is trained to classify a connection event to be one of three classes: *attack*, *anomaly* and *normal*. Assuming that H_1 is accurate in classifying known attacks, the new attacks are very likely to be classified as either *normal* or *anomaly*. When H_1 's prediction is either *anomaly* or *normal*, the connection event will be given to H_2 to classify if it is an instance of *new_intrusion*. H_2 will predict a default class if it doesn't classify the connection to be the new attack. If that is the case, the final prediction will be the original prediction of H_1 , which is either *anomaly* or *normal*. For simplicity, we limit the scope of our discussion to the situation where only one new class of intrusion is most recently discovered and knowledge of this new intrusion needs to be efficiently incorporated into an existing detection model. When more than one class of new attacks are gathered, H_2 can always be trained from more than one classes of data at the same time.

In order to classify three classes, the first classifier, H_1 , is trained from all available data at the time of training plus artificial anomalies generated by DBA2 from this data. There are different configurations on how H_2 is trained when data of an entirely new class is collected, and how the decisions of H_1 and H_2 are combined to produce a final prediction. Although, our methods are applicable to a wide variety of domains, we base the discussion on intrusion detection. Thus, in the following discussion we refer to the class label *new_intrusion* as the new class of attack recently discovered. H_2 has been trained to detect *new_intrusion*'s.

Configuration I: H_2 is trained from *new_intrusion* and *normal* data. The decision rules in Figure 4 are evaluated to compute the final prediction. If H_1 classifies the connection event to be either *normal* or *anomaly*, this event will be sent to H_2 .



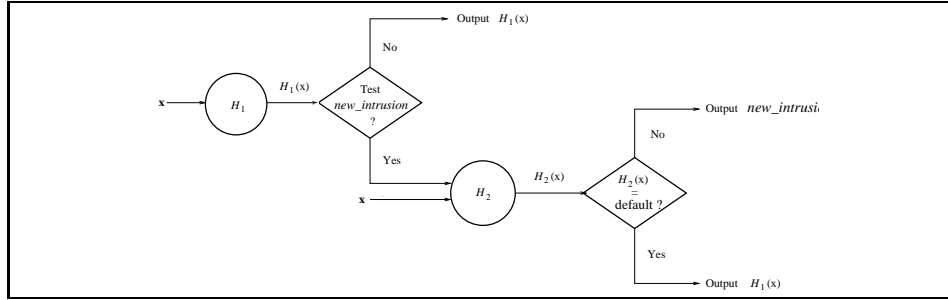


Figure 3. *Ensemble-based Adaptive Learning*

If H_2 classifies it to be *new_intrusion*, the final prediction will be *new_intrusion*. Otherwise, the final prediction will be the one predicted by H_1 .

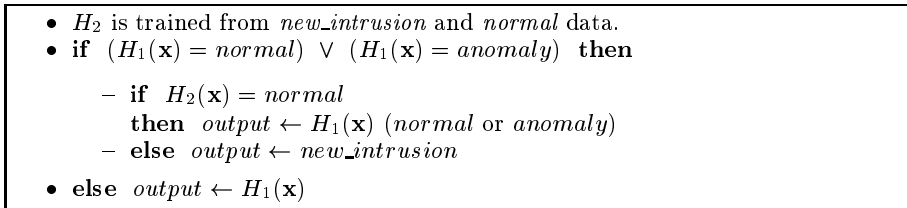


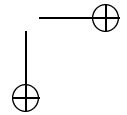
Figure 4. *Ensemble-based Adaptive Learning Configuration I*

H_1 is computed to separate known attacks from *normal* and *anomaly*; most *new_intrusion* connections will be classified as either *anomaly* or *normal*. Since H_2 is computed from *new_intrusion* and *normal* data, and *new_intrusion* is the minority class, H_2 is accurate in classifying the new attack, but classify true *normal* and *anomaly* connection events as *normal*. When H_2 predicts *normal*, we simply emit the prediction by H_1 , which is either *normal* or *anomaly*.

Configuration II: As shown in Figure 5, Configuration II is a slight variation of Configuration I. Only when the prediction by H_1 is *anomaly*, the connection data record will be given to H_2 for classification. If H_1 has a high precision¹ for predicting *normal* and high recall rate² for predicting *anomaly* (in which case most new attacks will be classified as *anomaly*), the performance of Configurations I and II will be very close. Otherwise, the Configuration I will have a higher recall rate for *new_intrusion*. Nonetheless, Configuration II will have a much higher throughput

¹Precision rate measures the percentage of predicted class i that are actually class i , and is defined as $\textit{precision} = \frac{|P_i \cap W_i|}{|P_i|} \times 100\%$ where P_i is the set of predictions with class i and W_i is the set of instances with class i

²Recall rate measures the percentage of instances of class i being correctly classified, and is defined as $\textit{recall} = \frac{|P_i \cap W_i|}{|W_i|} \times 100\%$.



rate since a majority of connections are *normal* and Configuration I spends extra time to examine every predicted *normal* connection by H_1 .

- H_2 is trained from *new_intrusion* and *normal* data.
- **if** $H_1(\mathbf{x}) = \textit{anomaly}$ **then**
 - **if** $H_2(\mathbf{x}) = \textit{normal}$ **then** $\textit{output} \leftarrow \textit{anomaly}$
 - **else** $\textit{output} \leftarrow \textit{new_intrusion}$
- **else** $\textit{output} \leftarrow H_1(\mathbf{x})$

Figure 5. *Ensemble-based Adaptive Learning Configuration II*

Configuration III: The combination rules are shown in Figure 6. The second classifier is trained from *new_intrusion* and artificial anomalies computed from data of *new_intrusion* only. When the prediction by H_1 is *anomaly*, it will be given to H_2 to classify. If H_2 predicts *new_intrusion*, the final prediction will be the new attack. Otherwise, it will be the prediction by H_1 . Most instances of *new_intrusion* will be classified as *anomaly* by H_1 and the second classifier H_2 will identify *new_intrusion* from all other anomalies.

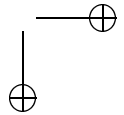
- H_2 is trained from *new_intrusion* and artificial anomalies computed from *new_intrusion*.
- **if** $H_1(\mathbf{x}) = \textit{anomaly}$ **then**
 - **if** $H_2(\mathbf{x}) = \textit{anomaly}$ **then** $\textit{outcome} \leftarrow \textit{anomaly}$
 - **else** $\textit{output} \leftarrow \textit{new_intrusion}$
- **else** $\textit{output} \leftarrow H_1(\mathbf{x})$

Figure 6. *Ensemble-based Adaptive Learning Configuration III*

Configuration IV: As show in Figure 7, Configuration IV is similar to Configuration III, but all of the connections predicted as either *normal* or *anomaly* will be given to H_2 for classification.

- H_2 is trained from *new_intrusion* and artificial anomalies computed from *new_intrusion*.
- **if** $(H_1(\mathbf{x}) = \textit{anomaly}) \vee (H_1(\mathbf{x}) = \textit{normal})$ **then**
 - **if** $H_2(\mathbf{x}) = \textit{anomaly}$ **then** $\textit{outcome} \leftarrow \textit{anomaly}$
 - **else** $\textit{output} \leftarrow \textit{new_intrusion}$
- **else** $\textit{output} \leftarrow H_1(\mathbf{x})$

Figure 7. *Ensemble-based Adaptive Learning Configuration IV*



Efficiency versus Accuracy In all four configurations, the predicted anomalies by H_1 will always be sent to H_2 to identify possible members of the class *new_intrusion*. If the precision of *normal* classification or recall rate of *anomaly* detection is not perfect, sending predicted *normal* records to H_2 will increase the recall rate of *new_intrusion*. However, at the same time it will reduce the classification throughput rate in a real time environment since most connections in reality are *normal* anyway. In practice, we can dynamically decide whether to test a connection that is predicted *normal* based on system performance, i.e., there is clearly a tradeoff between accuracy and efficiency.

Next, we discuss the empirical studies using DARPA evaluation dataset and RIPPER on the above approaches.

4 Experiment Setup

We first discuss the choice of learning algorithm, then discuss the dataset and a simulated real-world scenario.

Learning Algorithm We have chosen to use RIPPER [8], an inductive decision rule learner, for these experiments. One advantage to use rule learners is the comprehensibility of rules by domain experts. RIPPER generates two types of rules, either *ordered* or *un-ordered*.

An ordered ruleset has the form “**if** r_1 **then** i_1 **elseif** r_2 **then** i_2, \dots , **else default**”. Before learning rules from a dataset, RIPPER first heuristically orders the classes by one of the following methods: *+freq*, increasing frequency; *-freq*, decreasing frequency; *given*, a user-defined ordering; *mll*, minimal description length heuristics to guess an optimal ordering. After arranging the classes, RIPPER finds rules to separate $class_1$ from classes $class_2, \dots, class_n$, then rules to separate $class_2$ from classes $class_3, \dots, class_n$, and so on. The final class $class_n$ will become the default class. The end result is that rules for a single class will always be grouped together, but rules for $class_i$ are possibly simplified, because they can assume that the class of the example is one of $class_i, \dots, class_n$. If an example is covered by rules from two or more classes, this conflict is resolved in favor of the class that comes first in the ordering. An ordered ruleset is usually succinct and efficient, however, may not be very accurate since it biases towards classes early in the order. That is, it favors $class_i$ over $class_{i+1}$. A *+freq* ruleset will not have any *normal* rules at all; it is likely for a *normal* connection to be predicted as some attacks.

An un-ordered ruleset has at least one rule for each class and there are usually many rules for frequently occurring classes. There is also a default class which is used for prediction when none of these rules are satisfied. Unlike ordered rulesets, all rules are evaluated during prediction and conflicts are broken by using the most accurate rule. It is possible to convert any un-ordered ruleset into a logically equivalent ordered one. To do so, we order all rules by decreasing precision and alter them to ordered form. For example, consider the rules **if** $A \wedge B$ **then** i_1 (0.99); **if** C **then** i_2 : (0.98) : \dots . We then change them into ordered form **if** $A \wedge B$ **then** i_1 **elseif** C **then** i_2 **elseif** \dots . This process will not affect the logic because if an early rule (with high precision) is satisfied, there is no

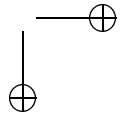


Table 1. *Intrusions, categories and sampling*

U2R		R2L		DOS		PRB	
buffer_overflow	1	ftp_write	4	back	1	ipsweep	1
loadmodule	2	guess_passwd	1	land	1	nmap	1
multihop	6	imap	2	neptune	$\frac{1}{20}$	portsweep	1
perl	6	phf	3	pod	1	satan	1
rootkit	2	spy	8	smurf	$\frac{1}{20}$		
		warezclient	1	teardrop	1		
		warezmaster	1				

utility in checking later rules with lower precision. Un-ordered rulesets, in general, contain more rules and are usually more accurate for multi-class problem like intrusion detection. If there are k classes in the dataset, learning an un-ordered ruleset is about k times more expensive than learning an ordered ruleset. Our previous work [12] has shown that un-ordered rulesets are more accurate for anomaly detection, which is the first step of the proposed ensemble approach; we choose to use un-ordered ruleset in this study as well.

Dataset The 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. The data were then processed into connection records using MADAM ID [17]. 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. For infrequent intrusions in the training data, the records for those connections were repeatedly injected to prevent the learning algorithm from neglecting them as statistically insignificant and not generating any rules for them. For overwhelming intrusions in the training data, only 1 out of 20 records were sampled. This is an ad hoc approach, but it produces reasonable results. Table 1 shows the category (U2R, R2L, DOS, PRB) and sampling rate of each intrusion.

Simulated Scenario The purpose of ensemble-based adaptive learning is to significantly increase learning efficiency, but at the same time, preserve accuracy. We assume that the monolithic classifiers trained from the same training set from scratch are potentially more accurate models than the ensembles. To verify the effectiveness, we compare their training efficiency and testing accuracy.

We simulate a real-world scenario to study the approach. The detection model

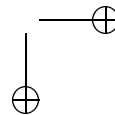
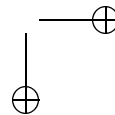


Table 2. Three unique orders to introduce simulated new intrusions

1	back	smurf	perl
2	buffer_overflow	spy	land
3	loadmodule	teardrop	phf
4	perl	pod	multihop
5	rootkit	phf	back
6	ftp_write	neptune	imap
7	warezclient	multihop	warezmaster
8	warezmaster	nmap	smurf
9	guess_passwd	ipsweep	neptune
10	imap	satan	guess_passwd
11	land	portsweep	spy
12	portsweep	land	satan
13	satan	imap	rootkit
14	ipsweep	guess_passwd	ipsweep
15	nmap	warezmaster	teardrop
16	multihop	warezclient	ftp_write
17	neptune	ftp_write	loadmodule
18	phf	rootkit	pod
19	pod	perl	warezclient
20	teardrop	loadmodule	buffer_overflow
21	spy	buffer_overflow	portsweep
22	smurf	back	nmap

is trained from data of a few intrusion classes and later some new intrusion types and data are available for learning. There are $n + 1$ connection classes in the evaluation dataset, including n different types of intrusions and *normal*; the data of connection class ℓ is denoted as D_ℓ . The full set of connection classes is $F = \{\ell_1, \dots, \ell_{n+1}\}$. We form n initial training sets S_i ($i = 0, \dots, n - 1$) and keep track of connection classes excluded from S_i in a set of labels L_i ; $S_0 = D_{normal}$ and $L_0 = F - \{normal\}$, $S_i = S_{i-1} \cup D_\ell$ and $L_i = L_{i-1} - \{\ell\}$, where ℓ is any label *s.t.* $\ell \in L_{i-1}$. The excluded intrusions $D_{\ell'}$ (where ℓ' is any class *s.t.* $\ell' \in L_i$) are taken individually as *new_intrusions* to train H_2 for the ensemble-based approach. As a comparison, we compute a monolithic classifier from $S_i \cup D_{\ell'}$ plus computed artificial anomalies. For each S_i , there are $n - i$ different tests. For n initial training sets, there are $\sum_0^{n-1} (n - i) = \frac{n(n-1)}{2}$ unique tests.

For n intrusion types, there are totally $n!$ different ways to order how the intrusions are introduced to produce initial training set and there are as many as $n!n(n - 1)/2$ learning tasks. In the 1998 DARPA intrusion detection evaluation dataset, there are 22 intrusion types, which allowed us to run 231 ($= \frac{22 \times 21}{2}$) comparisons for each order. To avoid 21! (or 1.124×10^{21}) different orders and 2.60×10^{23} comparisons, we have chosen only the three unique orders shown in Table 2. The first and last two orders are completely random. The second order is a completely reverse order of the first one; this is meant to test if the accuracy to detect some new intrusions are significantly influenced by those intrusions already in the training set.



5 Experiment Results

We first study the accuracy of the ensemble-based methods as compared to the single classifier, then the efficiency to train the ensembles.

Accuracy We compare the ensemble-based method with its respective monolithic classifier on both predictive accuracy and training efficiency. For *new_intrusion*, there are 4 predictive outcomes: *new_intrusion*, other intrusions that are not *new_intrusion*, *anomaly* and *normal*. We calculate *precision*, *recall*, *anomaly detection*, *other* and *false negative* (or *non-detection*) rates to measure all 4 possible outcomes. Precision shows how much we can trust the prediction by the detector when it flags *new_intrusion*. Recall rate tells us if we trust the prediction by the detector, the percentage of *new_intrusion* we can correctly detect. Anomaly detection rate measures the percentage of *new_intrusion* being detected as *anomaly*, and is defined as $\%anomaly = \frac{|D_{new_intrusion} \cap A|}{|D_{new_intrusion}|} \times 100\%$, where A is the set of predicted anomalies and $D_{new_intrusion}$ is the set of *new_intrusion* data. Other detection rate measures the percentage of *new_intrusion* being classified as some other types of intrusions. Anomaly detection rate and other detection rate describe the percentage of true *new_intrusions* if not detected correctly but at least flagged as “abnormal.” *False negative* or *non-detection* measures the overall performance.

Since we have 231 comparisons (explained in Section 4) for each order and five different measurements, and each comparison has results for both the ensemble and monolithic classifier, there are a total of 2310 ($=231 \times 2 \times 5$) measured rates for each order or 6930 for all three orders. It would not be useful to report every detailed result. Instead, we report the average of the measured rates for the ensemble and monolithic classifier, and their differences as shown in Table 3.

In Table 3, we show averaged *precision*, *recall*, *anomaly*, *other* and *false negative* rates for the ensemble-based method (e) and monolithic classifier (m), and their differences ($e-m$) for all intrusion types over all 3 orders. The results of four configurations are shown in each row marked I to IV. Since the monolithic classifier is the basis of comparison, their results are shown in single columns.

Configuration I has the best overall performance; it has significantly the highest precision and recall rates, which result in its lowest non-detection rate. Its precision rate is 20% lower than that of the monolithic classifier, which contributes to its slightly higher recall rate. Its non-detection rate is slightly lower than that of the monolithic classifier due to its slightly higher recall rate. The second best performer is Configuration II. Its precision rate is slightly lower and the recall rate is 40% lower than Configuration I; these result in its much higher non-detection rate. The big difference of recall and non-detection rates between I and II is apparently due to whether we use H_2 to examine predicted *normal*'s by H_1 . Since the anomaly detection may not detect all anomalies, or the recall rate of anomaly detection is not 100%, it is useful to re-examine predicted normals of H_1 by H_2 .

The performance of III and IV is worse than that of Configurations I and II; the precision rates are about 50% less, and recall rates are slightly lower as well. The reason is that the data size for each intrusion type is very small by itself; a few types of intrusions have only 30 to 50 examples in the training data. It is hard to produce effective artificial anomalies from such a small sample. The recall rate

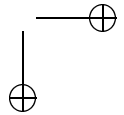


Table 3. Accuracy of monolithic classifier and four configurations on all three different orders

	Precision(%)			Recall(%)			Anomaly(%)		
	<i>e</i>	<i>m</i>	<i>e-m</i>	<i>e</i>	<i>m</i>	<i>e-m</i>	<i>e</i>	<i>m</i>	<i>e-m</i>
I	58.6	76.6	-18.0	90.2	86.8	+3.47	2.5	8.2	-5.6
II	51.0		-25.6	50.2		-36.6	2.5		-5.6
III	22.2		-54.4	46.1		-40.7	6.7		-1.5
IV	17.0		-59.6	86.0		-0.8	6.7		-1.5

	Other(%)			False Negative(%)		
	<i>e</i>	<i>m</i>	<i>e-m</i>	<i>e</i>	<i>m</i>	<i>e-m</i>
I	4.3	0.7	+3.6	3.0	4.4	-1.4
II	4.3		+3.6	43.0		+38.6
III	4.3		+3.6	43.0		+38.6
IV	4.3		+3.6	3.0		-1.4

e: ensemble-based method

m: monolithic classifier

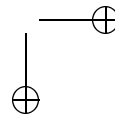
e - m: the difference of ensemble-based method and monolithic classifier

of Configuration IV is higher than that of Configuration III since we use H_2 to re-examine predicted normals by H_1 .

The anomaly detection rates of I and II, and also III and IV are the same; this is because H_2 is the same and the test set is also the same. The other detection rates of all four configurations are the same since it uses the same classifier (either H_1 or the monolithic classifier) when flagging *new_intrusion* as other types of known intrusions.

Training Efficiency We measured the training time by running learning tasks in a separate machine that had no other jobs running at the same time, and all files were on the local hard disk. The training time included the time to generate artificial anomalies (for the monolithic classifier and Configurations III and IV). The ensemble classifier approach finished in between 3 to 4 hours, but the monolithic model ran about 3 weeks or 504 hours to generate the 22 un-ordered rulesets for each order. In other words, the monolithic classifier approach is about 150 times more expensive than the ensemble approach. This significant difference in learning comes from that fact that monolithic classifier is trained as an k -class problem using accurate un-ordered ruleset and its training set size is much bigger than the one to train H_2 as a 2-class problem.

Taking both accuracy and training efficiency into consideration, Configuration



1 is the best performer; it has a predictive accuracy as good as a monolithic classifier, but the training cost is 150 times less.

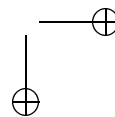
Additional Experiments As an interesting comparison, we also measured the efficiency to use monolithic classifier without artificial anomalies. This monolithic classifier doesn't provide as much utility as one trained with artificial anomalies, since it cannot detect anomalies and will always misclassify unknown intrusions as normal or one of the common intrusion attacks. We used the first order in Table 2 for the comparison. We have found that the ensemble-based approach is as accurate as the monolithic classifier, and the training cost is about 100 times less (instead of 150 times less). Training a monolithic classifier without artificial anomalies is faster, but still not as efficient as training an ensemble.

6 Related Work

Ensembles of classifiers can be generated in various ways. The major issues are how individual classifiers are trained, and how they are combined to form an ensemble.

In past research, individual classifiers have been trained by generating subsamples of the training set, manipulating the input features and output targets, and injecting randomness. A few of the most interesting works to subsample training sets are bagging [3], cross-validation committee [23] and boosting [26]. Bagging [3] works by generating n examples drawn randomly with replacement from the original training set with n instances. K -fold cross-validation [23] partitions the training set and forms a sample by using $k - 1$ partitions. Boosting [26], especially AdaBoost [13], maintains a distribution over the entire training set. The weighted training set is given to the learner to produce a hypothesis. The error rate is used to update the distributions. Manipulating input features generates multiple classifiers trained from disjoint feature subsets [6]. ECOC [10] transforms a K -class problem into L binary problems by partitioning K classes and associated training examples into two disjoint sets L times. A binary hypothesis is produced for each converted binary problem. Various methods of randomization have been introduced to produce individual classifiers such as initial parameterization [15], random selection of multiple choices [1], injection of noise in training samples [25], and random hypothesis sequence generation using the Markov Chain Monte Carlo method [19, 21, 7]. In this paper, the individual classifiers are trained from completely disjoint dataset that may not even share the same feature sets.

Individual classifiers are combined into ensembles by simple-voting, weighted-voting, gating networks, stacking and arbitration. As suggested by their names, simple-voting and weighted-voting combine the decisions by predicting the class with the highest vote. In gating network, the weight is computed as a function of both the input and the hypothesis [14]. Both stacking or combiner [27, 4] and arbitration [4] learn a separate classifier to combine the predictions of individual classifiers. Stacking or combiner [27, 4] learns a meta-level hypothesis on the correlations of predictions of individual hypotheses to the true label. Arbitration [4] learns a separate classifier from base-level training set whose weighted votes by individual classifiers are below a threshold value. Combiner has been implemented



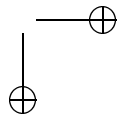
as a publicly available JAVA based meta-learning system JAM to provide scalable learning [24]. Various approaches to prune the ensemble to increase throughput and methods to combine a classifier learned from different dataset to increase accuracy have been explored in JAM. In the proposed adaptive learning method, the individual hypotheses are combined in a hierarchical way; data items are sent to the second hypothesis for classification only when necessary.

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. [22] 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 [16] attempted to remove majority instances too close to and too far from the decision boundary. Maxion and Tan [20] 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 [18] also applied entropy to determine how hard it is to learn a model of normality and abnormality. Chang and Lippman [5] applied voice transformation techniques to add artificial training talkers to increase variabilities.

7 Conclusion

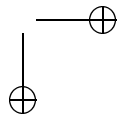
As our adversaries constantly invent and launch new types of attacks to services provided over the Internet, Intrusion Detection Systems need to efficiently and accurately incorporate knowledge of previously unseen attacks into existing detection model to prevent any further damage by new attacks as early and as much as possible. There isn't much previous research to address this problem. In this paper, we studied the problem of using supervised inductive learning techniques to efficiently and accurately adapt existing detection model to new class of intrusions. Based on a general anomaly detection framework, we use ensembles of classifiers to quickly learn and deploy models to detect newly discovered intrusions. We first show how to convert a detector that only detects normal and attacks into one that detect three classes of connections, normal, attack and anomaly. We then discuss a method to train a light weight model from new attack data only and use a simple decision rule system to attach this new model to existing detection model. Thus two models are integrated and the training time only comes from the computation of the second model on new intrusion only, which is significantly less than re-computing a new model from both new intrusion and all known intrusion data. We explore and analyze four different configurations to train the light weight model and respective decision rule systems.

Empirical evaluations on the 1998 DARPA intrusion detection evaluation dataset have shown that using ensemble-based adaptive learning methods to detect newly discovered intrusions is as accurate as learning monolithic models over all available data to detect intrusions, yet with significantly lower training cost. This quick learning provides an opportunity to deploy new models rapidly to thwart damage of both new and old attacks, which can be replaced later with an updated and better monolithic model as time and resources permit.

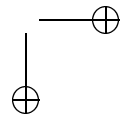


Bibliography

- [1] Kamal Ali and Michael Pazzani. Error reduction through learning multiple descriptions. *Machine Learning*, 24(3):173–202, 1996.
- [2] E Amoroso. *Intrusion Detection: An Introduction to Internet Surveillance, Correlation, Traps, Trace Back, and Response*. Intrusion Net Books, 1999, 1997.
- [3] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.
- [4] Philip Chan and Salvatore Stolfo. Experiments on multi-strategy learning by meta-learning. In *International Conference on Knowledge and Information Management*, pages 314–323, 1993.
- [5] 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.
- [6] K J Cherkauer. Human expert-level performance on a scientific image analysis by a system using combined artificial neural networks. In P Chan, editor, *AAAI Workshop on Integrating Multiple Learned Models*, pages 15–21. AAAI, 1996.
- [7] H Chipman, E George, and R McCulloch. Bayesian CART. Technical report, Statistics, University of Chicago, 1996.
- [8] William Cohen. Fast effective rule induction. In *Proceedings of Twelfth International Conference on Machine Learning (ICML-95)*, pages 115–123. Morgan Kaufman, 1995.
- [9] T Dietterich. Machine learning research: Four current directions. *AI Magazine*, 18(4), 1998.
- [10] T Dietterich and G Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995.
- [11] W Fan. *Cost-sensitive, Scalable and Adaptive Learning Using Ensemble-based Methods*. PhD thesis, Columbia University, Dec 2000.



- [12] Wei Fan, Matthew Miller, Salvatore J Stolfo, Wenke Lee, and Philip K Chan. Using artificial anomalies to detect unknown and known network intrusions. In *To appear Proceedings of First IEEE International Conference on Data Mining (ICDM'01)*, 2001.
- [13] Yoav Freund and Robert Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Computer and System Sciences*, 55(1):119–139, 1997.
- [14] Michael Jordan and R Jacobs. Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, 6(2):181–214, 1994.
- [15] J F Kolen and J B Pollack. Back propagation of sensitive to initial conditions. In *Advances in Neural Information Processing Systems*, volume 3, pages 313–321. Morgan Kaufmann, San Francisco, California, 1991.
- [16] M Kubat and S Matwin. Addressing the curse of imbalanced training sets: One sided selection. In *Proceedings of Fourteenth International Conference on Machine Learning (ICML-97)*, pages 179–186, 1997.
- [17] W Lee. *A Data Mining Framework for Constructing Features and Models for Intrusion Detection Systems*. PhD thesis, Columbia University, June 1999.
- [18] Wenke Lee and Dong Xiang. Information-theoretic measures for anomaly detection. In *The 2001 IEEE Symposium on Security and Privacy*, Oakland, CA, May 2001.
- [19] D MacKay. A practical Bayesian framework for back-propagation networks. *Neural Computation*, 4(3):448–472, 1992.
- [20] 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.
- [21] R Neal. Probabilistic inference using Markov chain: Monte Carlo methods. Technical report, Computer Science, University of Toronto, Toronto, Canada, 1993.
- [22] K Nigam, A McCallum, S Thrun, and T Mitchell. Learning to classify text from labeled and unlabeled documents. In *Proceedings of Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, 1998.
- [23] B Parmanto, P W Munro, and H R Doyle. Improving committee diagnosis with resampling techniques. In D S Touretzky, M C Mozer, and M E Hesselmo, editors, *Advances in Neural Information Processing Systems*, pages pp.882–888. MIT Press, 1996.
- [24] Andreas Prodromidis. *Management of Intelligent Learning Agents in Distributed Data Mining Systems*. PhD thesis, Columbia University, Oct 1999.



- [25] Y Raviv and N Intrator. Bootstrapping with noise an effective regularization technique. *Connection Science*, 8(3-4):355–372, 1996.
- [26] R Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–220, 1990.
- [27] David Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.

