

Model Generalization and Its Implications on Intrusion Detection

Zhuowei Li^{1,2}, Amitabha Das¹, and Jianying Zhou²

¹ School of Computer Engineering, Nanyang Technological University
50 Nanyang Avenue, Singapore 639798

zhwei.li@mail.ntu.edu.sg, asadas@ntu.edu.sg

² Institute for Infocomm Research, 21 Heng Mui Keng Terrace, Singapore 119613
jyzhou@i2r.a-star.edu.sg

Abstract. To make up for the incompleteness of the known behaviors of a computing resource, model generalization is utilized to infer more behaviors in the behavior model besides the known behaviors. In principle, model generalization can improve the detection rate but may also degrade the detection performance. Therefore, the relation between model generalization and detection performance is critical for intrusion detection. However, most of past research only evaluates the overall efficiency of an intrusion detection technique via detection rate and false alarm/positive rate, rather than the usefulness of model generalization for intrusion detection. In this paper, we try to do such evaluation, and then to find the implications of model generalization on intrusion detection. Within our proposed methodology, model generalization can be achieved in three levels. In this paper, we evaluate the first level model generalization. The experimental results show that the first level model generalization is useful mostly to enhance the detection performance of intrusion detection. However, its implications for intrusion detection are different with respect to different detection techniques. Our studies show that in general, though it is useful to generalize the normal behavior model so that more normal behaviors can be identified as such, the same is not advisable for the intrusive behavior model. Therefore, the intrusion signatures should be built compactly without first level generalization.

Keywords: Security, Machine Learning, Intrusion Detection, Generalization, Intrusion, Security Infrastructure.

1 Introduction

Intrusion detection has become a very important defense mechanism in the face of increasing vulnerabilities exposed in today's computer systems and the Internet (Debar et al.[3]), where authentication, cryptography, and access control mechanisms routinely prove inadequate in preventing new and increasingly numerous and disastrous attacks. In general, there exist two approaches for detecting intrusions (Denning[4]): signature-based intrusion detection (SID, a.k.a. misuse detection), where an intrusion is detected if its behavior matches existing intrusion signatures, and anomaly-based intrusion detection (AID), where

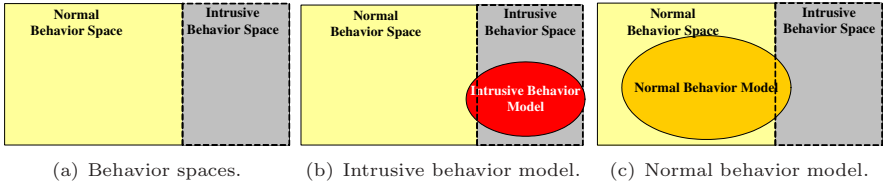


Fig. 1. Behavior Spaces and Models.

an intrusion is detected if the resource behavior deviates from known normal behaviors significantly. From another aspect, there are two behavior spaces in a computing resource for intrusion detection (Figure 1.a): *normal behavior space* and *intrusive behavior space*, and they are complementary to each other. Conceptually, SID is based on knowledge in intrusive behavior space, and AID is based on knowledge in normal behavior space (Chari et al.[2]). Perfect detection of intrusions can be achieved only if we have a complete model of any one of the two behavior spaces, because what is not bad is good and vice versa ideally.

Motivations: However, it is difficult to model such behavior spaces completely and correctly in reality. Figure 1 (b) and (c) illustrate real behavior models for signature-based (i.e., *intrusive behavior model*) and for anomaly-based intrusion detection (i.e., *normal behavior model*) (Chari and Cheng[2]). As the figure indicates, there exist model errors in the behavior models for SID techniques as well as AID ones. These model errors are called the *inaccuracy* in the behavior models. For example, a part of intrusive behavior model falls into normal behavior space. In addition, the intrusive behavior model cannot cover all intrusive behavior space, and the normal behavior model cannot cover all normal behavior space either. This is referred to as the *incompleteness* in the behavior models. In summary, there are two quality factors in every behavior models, namely *inaccuracy* and *incompleteness*. To build a practical intrusion detection system, it is critical to know the precise influence of these two quality factors on its performance.

On the other hand, the behavior models for intrusion detection are generally built from known knowledge of the protected resource. Thus, the quality of the behavior model is determined by the known knowledge as well as the ‘*model building*’ technique. In reality, it is hard to guarantee the completeness of known knowledge under most scenarios (e.g., with ‘concept drifting’ problem, Li et al.[10]) as we do not know the future behaviors of a computing resource. To make up for the incompleteness, most existing ‘*model building*’ techniques try to infer the unknown behaviors via *model generalization* (Debar et al.[3], Lee et al.[9], Chan et al.[13]). However, model generalization may lead to model inaccuracy (Figure 1), and model generalization cannot solve the incompleteness problem completely under most scenarios where it infers part but not all of unknown behaviors. Thus, the quality factors in the behavior models are completely determined by model generalization. In order to get the precise influence

of the two quality factors, it is necessary to evaluate model generalization and its implications for intrusion detection.

Related Work: First, the intrusion signatures can be generalized to cover more intrusive behavior space, i.e., the intrusive behavior model in Figure 1 is extended. Anchor et al. [1] have done the only work on this to our knowledge. Using the evolution programming with a fitness function which depends on false positive rate and detection rate, they optimized the degree of generalization for intrusion signatures (represented by a *finite state transducer*). In general, the model generalization on intrusion signatures can solve the intrusion variations detection problem partially. Secondly, the normal behavior model of anomaly-based intrusion detection can be generalized as well, and it can be done in several ways. Based on a distance metric and a threshold (Denning[4], Eskin[5], Liao[12], Wang[16]), the instances in the existing audit trails are clustered unsupervisedly, and the new instances are labeled by the existing instances in their clusters. In statistical methods for intrusion detection (MADAM ID[9], LERAD[13], NIDES[7] etc.), the (statistical) resource usage profiles are mined from the existing audit trails. The novel instances are detected according to whether they fall into these profiles. In these two approaches, the existing audit trails are modeled inexactly to accommodate more resource behaviors in the profiles, and thus to achieve the model generalization.

Most of past research only credited the overall efficiency of an intrusion detection technique to such model generalization, and there is hardly any evaluation of the effect of the model generalization. This is partially due to the difficulty of pinpointing the contribution of model generalization to the overall efficiency. Fortunately, our methodology not only overcomes the problem, it also allows one to adjust the extent of model generalization.

Our Contributions: First, a new intrusion detection methodology with promising properties is introduced briefly. Then, an evaluation framework for model generalization is designed using our methodology, and an average detection cost function is defined to quantify the detection performance for intrusion detection. Finally, the experiments are done on a typical dataset to get the exact implications of model generalization for intrusion detection.

The remaining parts of this paper are organized as follows. Section 2 describes a formal intrusion detection methodology in brief. In section 3, model generalization is introduced and quantified, and its evaluation framework is designed. Experiments in section 4 reveals the implications of model generalization on intrusion detection. Lastly, we draw conclusions and lay out the future work on model generalization in section 5.

2 Retrospection: An Intrusion Detection Methodology

Any intrusion detection system builds the behavior models of the resources using a set of features, or a *feature vector* $FV = \{F_1, F_2, \dots, F_n\}$, where F_i denotes one of its features. Every feature in the feature vector can be one of these types: A

feature associated with an instant of time (e.g., *the fields in the current packet*), or with a time interval (e.g., *the number of SYN packets within 2 seconds*), or with the context of a current event (e.g., *the system-call events in stide [6], the state events in STAT [15]*). The context is defined over the timeline preceding the point in time when the event in question happens. In general, a feature F_i in the feature vector can be categorized into *nominal*, *discrete* or *continuous* one. A feature vector for intrusion detection can contain any number of nominal, discrete, and/or continuous features. In addition, a feature can also be as complex as a compound feature (see Section 2.3).

In this methodology [11], we assume that there is a training audit trail, which consists of normal and intrusion audit trails. We also assume that the training audit trails represent the known (or past) knowledge about the computing resource. Then, the instances of the feature vector are collected from the training audit trails as $\{I_{FV}^1, I_{FV}^2, I_{FV}^3, \dots\}$. For each instance I_{FV}^i , there is a **status** that indicates the label of audit trails where it is collected. For example, if an instance I_{FV}^i is produced by an intrusion ‘Nimda’, its status is ‘Nimda’.

2.1 Basic Concepts and Notations

For a specific feature F , several of its concepts are defined as follows.

- Its feature space $Dom(F)$ is the defining domain w.r.t. a computing resource.
- Any value in $Dom(F)$ is defined as a feature value v_F , and $v_F \in Dom(F)$. In general, there are many feature values in the feature space $Dom(F)$.
- A feature range R_F is an interval between any two feature values v_F^1 and v_F^2 in its feature space, which includes all feature values falling between v_F^1 and v_F^2 . For a discrete or continuous feature, $R_F = [v_F^1, v_F^2]$. For a nominal feature, every feature value is independent. Thus, each nominal feature value is referred to as a feature range so that for a nominal feature F , $R_F = [v_F^i] = [v_F^i, v_F^i]$. If a feature value v_F^j is within the bounds of a feature range, we say that it falls within it, denoted as $v_F^j \in R_F$. The concept of *feature range* is used to treat uniformly every (nominal, discrete, or continuous) feature. For the range R_F , we further define $upper(R_F) = v_F^1$ and $lower(R_F) = v_F^2$.

Notations. We describe here the notations used in the subsequent expressions. Note that in order to avoid cluttering the expressions, we have dropped the subscript of F . If $F \in FV$,

- $v(I_{FV}^i, F)$ is the feature value of the feature F in the instance I_{FV}^i .
- $I(v_F, F)$ is the set of instances whose values of F are equal to v_F .

$$I(v_F, F) = \{I_{FV}^k | v_F = v(I_{FV}^k, F)\}$$

- $I(R_F, F)$ is the set of instances whose values of F fall in the feature range R_F .

$$I(R_F, F) = \{I_{FV}^k | v(I_{FV}^k, F) \in R_F\}$$

2.2 NSA Label

Definition 1 (NSA label of a feature value). *If a feature value v_F occurs only in the normal audit trails, it is normal. If it occurs only in the intrusive audit trails, for example, intrusion signatures, it is labeled as anomalous. Otherwise, i.e., if it occurs in both normal and intrusive audit trails, it is labeled as suspicious. For brevity, we will refer to the normal, suspicious, or anomalous label as the NSA label of the feature value v_F , denoted as $L(v_F) \in \{‘N’, ‘S’, ‘A’\}$.*

It is worth noting that a feature value is either normal or anomalous in a specific instance, but its NSA label is collected from all related instances. We will further extend the concept of **NSA label** to feature ranges of a feature.

NSA Labels of Feature Ranges. From $Dom(F)$, we can collect a set of mutually exclusive feature ranges $\{R_F^1, R_F^2, \dots\}$, such that **(1)** there is no common feature value v_F , which falls in R_F^j and R_F^k at the same time ($j \neq k$), and **(2)** $I(R_F^i, F) \neq \emptyset$ ($i \geq 1$). Then, the concept of NSA labels can be extended to these feature ranges as follows. For the feature range R_F ,

$$\begin{aligned} L(R_F) = ‘N’ &\Leftrightarrow \forall i(v(I_{FV}^i, F) \in R_F \rightarrow L(v(I_{FV}^i, F)) = ‘N’) \\ L(R_F) = ‘A’ &\Leftrightarrow \forall i(v(I_{FV}^i, F) \in R_F \rightarrow L(v(I_{FV}^i, F)) = ‘A’) \\ L(R_F) = ‘S’ &\Leftrightarrow \exists i \exists j(v(I_{FV}^i, F) \in R_F \wedge L(v(I_{FV}^i, F)) = ‘A’) \\ &\quad \wedge (v(I_{FV}^j, F) \in R_F \wedge L(v(I_{FV}^j, F)) = ‘N’) \end{aligned}$$

Feature Subspaces. Based on NSA labels, we can partition the feature space $Dom(F)$ into *three* feature subspaces: *normal*, *suspicious* and *anomalous*, denoted as $N(F)$, $S(F)$ and $A(F)$, respectively. Thus we have,

$$\begin{aligned} N(F) &= \{R_F^j \mid j \geq 1, L(R_F^j) = ‘N’\} \\ S(F) &= \{R_F^j \mid j \geq 1, L(R_F^j) = ‘S’\} \\ A(F) &= \{R_F^j \mid j \geq 1, L(R_F^j) = ‘A’\} \end{aligned}$$

We also define $\Omega(F) = N(F) \cup S(F) \cup A(F)$, so that it is a collection of all feature ranges found in the training audit trails.

2.3 Compound Feature

Definition 2 (compound feature). *A compound feature F_{12} is an ordered pair $\{F_1, F_2\}$, such that $\Omega(F_{12})$ is a subset of the cartesian product of $\Omega(F_1)$ and $\Omega(F_2)$. It also requires that each element in $\Omega(F_{12})$ represents at least one feature vector instance in the training audit trails. Mathematically,*

$$\Omega(F_{12}) = \{(R_{F_1}^a, R_{F_2}^b) \mid R_{F_1}^a \in \Omega(F_1), R_{F_2}^b \in \Omega(F_2), I((R_{F_1}^a, R_{F_2}^b), F_{12}) \neq \emptyset\}$$

Based on the definition of *cartesian product*, for any instance recognized by a compound feature range $(R_{F_1}^a, R_{F_2}^b)$, $I_{FV}^i \in I((R_{F_1}^a, R_{F_2}^b), F_{12})$, it will be recognized by feature ranges $R_{F_1}^a$ and $R_{F_2}^b$ as well (i.e., $I_{FV}^i \in I(R_{F_1}^a, F_1)$ and $I_{FV}^i \in I(R_{F_2}^b, F_2)$), and vice versa. Therefore, $I((R_{F_1}^a, R_{F_2}^b), F_{12}) = I(R_{F_1}^a, F_1) \wedge I(R_{F_2}^b, F_2)$. To avoid any ambiguity, we will refer to a single feature as an *atomic* feature. When a compound feature is formed from two component features, the ranges of the compound feature that are generated as a result of the process of composition are also mutually exclusive just like the feature ranges of the component features. This property is expressed in the following theorem, the proof of which is given in the appendix.

Theorem 1. *The feature ranges of a compound feature are mutually exclusive, i.e., for two different feature ranges, $(R_{F_1}^a, R_{F_2}^b)$ and $(R_{F_1}^c, R_{F_2}^d)$, there is no such instance I_{FV}^i so that $I_{FV}^i \in I((R_{F_1}^a, R_{F_2}^b), F_{12})$ and $I_{FV}^i \in I((R_{F_1}^c, R_{F_2}^d), F_{12})$.*

A compound feature space can be partitioned into three feature subspaces like an atomic feature, i.e., $\Omega(F_{12}) = N(F_{12}) \cup S(F_{12}) \cup A(F_{12})$.

Compounding More Features. In summary, the compound feature built from two atomic features shows the same properties as any of its component atomic features. Therefore, we can treat the compound feature as an atomic one to build higher order compound features. Using this recursive procedure, the feature vector FV for intrusion detection can be converted into an equivalent n -order compound feature $F_{1\dots n}$ with normal $N(F_{1\dots n})$, suspicious $S(F_{1\dots n})$ and anomalous $A(F_{1\dots n})$ subspaces.

2.4 Behavior Signatures

Definition 3 (behavior signature). *Assuming that there exists a feature vector $FV = \{F_1, F_2, \dots, F_n\}$, and that the feature ranges of every feature are determined beforehand. A behavior signature Sig_{FV}^i is a feature range of the compound feature $F_{1\dots n}$ with its NSA label. In other words, the behavior signature is the combination of feature ranges of all the features in the feature vector.*

As indicated in the above definition, every behavior signature¹ represents a state of the resource at a specified time point. According to NSA labels of signatures, the behavior model can be split into three parts: normal, suspicious, and intrusion behavior models. In AID, only the normal behavior model is utilized, but SID identifies intrusions based on the intrusive behavior model. However, the best scenario is to do intrusion detection using the complete behavior models.

Detecting Instances via Signatures. Suppose that $FV = \{F_1, \dots, F_n\}$, and the signatures are $Sig_{FV}^1, Sig_{FV}^2, \dots, Sig_{FV}^m$, where $Sig_{FV}^i = (R_{F_1}^{i_1}, R_{F_2}^{i_2}, \dots, R_{F_n}^{i_n})$, and the set of instances recognized by Sig_{FV}^i is $I(Sig_{FV}^i, F_{1\dots n})$. According to the definition of the compound feature $F_{1\dots n}$

$$I(Sig_{FV}^i, F_{1\dots n}) = I(R_{F_1}^{i_1}, F_1) \wedge I(R_{F_2}^{i_2}, F_2) \wedge \dots \wedge I(R_{F_n}^{i_n}, F_n)$$

¹ For brevity, ‘behavior signature’ will be simplified as ‘signature’ in this paper.

For every instance of the computing resource, for example I_{FV}^j ,

$$I_{FV}^j \in I(\text{Sig}_{FV}^i, F_{1\dots n}) \\ \Leftrightarrow v(I_{FV}^j, F_1) \in R_{F_1}^{i_1}, v(I_{FV}^j, F_2) \in R_{F_2}^{i_2}, \dots, v(I_{FV}^j, F_n) \in R_{F_n}^{i_n}$$

In order to detect I_{FV}^j , all of its feature values must fall in their corresponding feature ranges in the signature. Then, the relation between the NSA label of a signature and the instance is utilized in intrusion detection. In the behavior model, if the NSA label of the signature is ‘N’, the instance is detected as ‘normal’. If the NSA label of the signature is ‘S’, the instance is detected as ‘anomalous’. Otherwise, i.e., if its NSA label is ‘A’, the instance will be labeled by the respective intrusion(s). Significantly, if there is no signature in the behavior model that recognizes the instance, the instance will be detected as ‘anomalous’ because we lack knowledge about the ‘new’ instance in the behavior model.

Note that the methodology thus introduced has a number of very desirable properties that make it attractive for intrusion detection, such as

(1)**Incremental:** it is possible to append a new feature in the feature vector, and it is also possible to append a new feature range to an existing feature dynamically;

(2)**Distributed:** as mentioned earlier, a feature can also be an output of a meta-intrusion detection system. Thus, different (atomic or compound) features can be deployed distributively and/or hierarchically;

(3)**Privacy protection:** The information in a feature has been replaced by the index of its corresponding feature range, and thus, the privacy in the instance has been protected;

(4)**Uniqueness:** signatures in the behavior models are mutually exclusive, thus there is no ambiguity in detecting an instance;

(5)**Quantified behavior models/spaces:** the behavior spaces/models can be quantified as the numbers of signatures in them. For this reason, we can define intrusion detection and analyze existing problems in a quantified manner;

(6)**Adjustable quality factors:** This methodology can describe the known knowledge precisely and extend the knowledge via a configurable degree of model generalization.

In the following section, we will use the *adjustable quality factors* to evaluate the usefulness of model generalization for intrusion detection.

3 Model Generalization

In general, *model generalization can improve the detection rate by identifying more novel behaviors (e.g., normal behaviors) but may also degrade the detection performance by mis-identifying novel behaviors due to generalization errors (Valdes et al.[14])*. Therefore, it is necessary to articulate the relation between model generalization and the detection performance. In our methodology, the task of model generalization is to make every signature in the behavior models identify more instances, and it can be achieved in three levels described below.

Note that, without model generalization, the known behaviors in the training audit trails can be represented in the behavior models precisely.

- L1: **[Generalizing feature ranges]**. In the *feature range collection strategies*, the feature ranges can be generalized to cover more unknown parts of feature space other than the known feature values.
- L2: **[Generalizing the behavior model]**. At the same time, using the *crossover* operations (or other generalization operations), especially on two signatures with the same NSA label, some signatures that are not in the training audit trails are augmented into the behavior model. Thus, the behavior model is generalized as well. To some degree, the effect of this generalization is to ignore several features in building signatures.
- L3: **[Generalizing signatures during detection]**. With a *distance metric*, the model generalization introduces some signatures that are not in the training audit trails but within a distance threshold from existing signatures.

Figure 2 illustrates the three levels for model generalization using a simple but representative scenario, in which there are two features and 9 instances (*the points A to I in the figure*) initially. The light squares with label ‘L1’, which include initial instances, represent signatures due to the L1 generalization. The squares with ‘L2’ represent signatures introduced by L2 generalization. Lastly, the dark squares with label ‘L3’ represent signatures introduced by L3 generalization. Note that we have not illustrated all possible signatures from L2 and L3 model generalization in Figure 2.

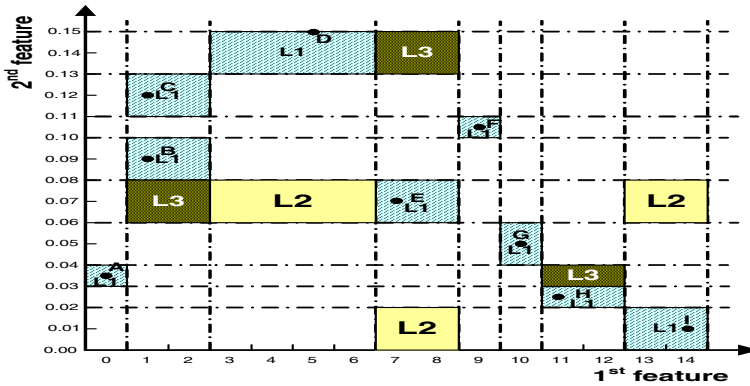


Fig. 2. Three levels for model generalization.

It is obvious that L1 generalization is the basis for L2 and L3 generalization. Thus, we will first discuss the model generalization introduced by collection strategies (*i.e., L1 model generalization*). In other words, we will evaluate the relation between model generalization introduced by collection strategies and the detection performance for intrusion detection.

For every feature, to achieve L1 generalization, we assume that the unknown parts in the feature space have the same NSA label as its nearest feature values.

Algorithm 1 Expanding the initial feature ranges for a discrete feature.

Require: (1) Initial feature ranges $R_{F_d}^i$ and $R_{F_d}^{i+1}$; (2) v_i and v_{i+1} ; and (3) λ_d .

```

1: FUNCTION: Expand( $R_{F_d}^i, R_{F_d}^{i+1}, v_i, v_{i+1}, \lambda_d$ )
2:  $expand_i = 0; expand_{i+1} = 0;$ 
3:  $gap = v_{i+1} - v_i - 1;$ 
4: if  $gap = 0$  then return;
5: if  $gap \geq 2\lambda_d$  then
6:    $expand_i = \lambda_d; expand_{i+1} = \lambda_d;$ 
7: else
8:   if  $gap \% 2 = 0$  then
9:      $expand_i = gap/2; expand_{i+1} = gap/2;$ 
10:  else
11:     $expand_i = (gap - 1)/2; expand_{i+1} = (gap - 1)/2;$ 
12:    Randomly select a value in  $\{0,1\}$ , and assign it to  $Rnd$ ;
13:     $expand_i + = Rnd; expand_{i+1} + = 1 - Rnd;$ 
14:  end if
15: end if
16: Increasing the upper bound of  $R_{F_d}^i$  by  $expand_i$ ;
17: Decreasing the lower bound of  $R_{F_d}^{i+1}$  by  $expand_{i+1}$ ;

```

Obviously, we indicate that there is a distance concept in L1 model generalization. However, because there is no distance concept for nominal features, L1 model generalization is not applicable to them actually. Therefore, we will only discuss the L1 model generalization on discrete/continuous features in this paper, and regard every feature value of a nominal feature as a feature range.

3.1 Model Generalization by Collection Strategies

Before applying the collection strategies, an initial feature range is formed for every feature value, and both its upper and lower bounds are equal to the feature value. With these initial feature ranges, the signatures can represent the behaviors in the training audit trails correctly and precisely. The model generalization is performed to cover more unknown feature space.

A Collection Strategy for Discrete Features. For a discrete feature F_d , other than feature values in all instances, there are some feature values that do not occur in the training audit trails. The collection strategy is to extend the initial feature ranges to embrace these non-occurring feature values. For example, $\Omega(F_d) = [0, 20]$, and the feature values in the training audit trails are $\{0, 2, 3, 11, 13, 18, 19, 20\}$. Thus, the non-occurring feature values are $\{[1, 1], [4, 10], [12, 12], [14, 17]\}$, and the task of the collection strategy is to infer NSA labels for all of them. To achieve it, the unknown feature values will be labeled by its nearest known feature values, where the concept of ‘nearest’ is represented by a generalization parameter λ_d .

Given the generalization parameter λ_d , the collection strategy is designed as follows. Suppose that existing feature values are v_1, v_2, \dots, v_m in an incremental order. The initial feature ranges $R_{F_d}^i$ and $R_{F_d}^{i+1}$ will be expanded by the unknown feature values between v_i and v_{i+1} ($1 \leq i \leq m - 1$). Algorithms 1 and 3 show the pseudo codes for the collection strategy of discrete features.

Algorithm 2 Expanding the initial feature ranges for a continuous feature.

Require: (1) Initial feature ranges $R_{F_c}^i$ and $R_{F_c}^{i+1}$; (2) v_i and v_{i+1} ; and (3) λ_c .

- 1: FUNCTION: **Expand**($R_{F_c}^i, R_{F_c}^{i+1}, v_i, v_{i+1}, \lambda_c$)
- 2: $expand = 0$;
- 3: **if** $\lambda_c > 0.5$ **then** $L = 0.5$;
- 4: $gap = v_{i+1} - v_i$;
- 5: $expand = gap \times \lambda_c$;
- 6: Increasing the upper bound of $R_{F_c}^i$ by $expand$;
- 7: Decreasing the lower bound of $R_{F_c}^{i+1}$ by $expand$;

Algorithm 3 The Splitting Strategy for a Discrete or Continuous Feature F .

Require: (1) Initial feature ranges R_F^1, R_F^2 , and R_F^l ; (2) v_1, \dots, v_l ; and (3) λ .
 {If the feature F is discrete, λ represents λ_d , otherwise, λ_c .}

- 1: **for** $i = 1$ to $l - 1$ **do**
- 2: **Expand**($R_F^i, R_F^{i+1}, v_i, v_{i+1}, \lambda$);
- 3: **end for**
- 4: $i=1$;
- 5: **while** $i = l$ **do**
- 6: **if** No additional feature subspace between R_F^i and R_F^{i+1} , and $L(R_F^i) = L(R_F^{i+1})$ **then**
- 7: Merging R_F^{i+1} into R_F^i ;
- 8: Deleting R_F^{i+1} ; {The index of every following feature range is decreased by 1}
- 9: $l = l - 1$;
- 10: **else**
- 11: $i = i + 1$;
- 12: **end if**
- 13: **end while**

A Collection Strategy for Continuous Features. Similarly, for a continuous feature F_c , there are feature values in the training audit trails, v_1, v_2, \dots, v_l , in an incremental order, and the initial feature ranges are $R_{F_c}^1, R_{F_c}^2, \dots, R_{F_c}^l$, where, for every feature range $R_{F_c}^i$, $lower(R_{F_c}^i) = upper(R_{F_c}^i) = v_i$. However, unlike the collection strategy for discrete features, there always exist non-occurring feature space between any two consecutive initial feature ranges, and the objective of its collection strategy is to split these non-occurring feature space. For example, $\Omega(F_c) = [0, 1.0]$, and the feature values in the training audit trails are $\{0, 0.02, 0.03, 0.12, 0.23, 0.40, 1.0\}$. Thus, the non-occurring feature space are $\{[0,0.02], [0.02,0.03], [0.03,0.12], [0.12,0.23], [0.23,0.40], [0.40,1.0]\}$, and the task of its collection strategy is to label them with NSA label(s). Similar to discrete features, there is another generalization parameter λ_c to infer the NSA labels of unknown feature parts in the feature space of the continuous feature F_c .

Given the generalization parameter λ_c , the collection strategy is done as follows. For every feature value v_i ($1 \leq i \leq l - 1$) in the existing audit trails, the initial feature space $R_{F_c}^i$ and $R_{F_c}^{i+1}$ will be expanded by non-occurring feature ranges between v_i and v_{i+1} . Algorithms 2 and 3 show the pseudo codes for the collection strategy of continuous features.

Ultimately, for every pair of parameters $\langle \lambda_d, \lambda_c \rangle$, the feature ranges for all features in the feature vector will become determined. Then, the signatures are collected from all instances in the training audit trails. In the detection phase, the instances in the test audit trails are utilized to measure the performance

of the behavior model with parameter pair $\langle \lambda_d, \lambda_c \rangle$. In the following section, an average cost function for every instance of the detection performance is designed.

3.2 Measuring the Detection Performance

The two main objectives of intrusion detection are (1) to detect the intrusions correctly (as anomalies), and (2) to identify the behaviors correctly (i.e., normal behaviors or its original intrusions). Considering these two objectives, we calculate the average detection cost (Lee et al.[8]) of an instance in the test audit trails as in Table 1. If the behavior is identified correctly, the cost is 0. Otherwise, we can give some penalty for the detection result. In our cost scheme, we assume that the detection of an intrusion as an anomaly is useful but it is less useful than intrusion identification. Lastly, the average cost of every instance in the test audit trails is utilized to quantify the detection performance of the behavior model.

Suppose that there are T instances in the test audit trails. Based on the detection results, several statistics are further defined as follows.

Table 1. Detection Results and Costs.

INDEX	NOTATIONS	ORIGINAL CLASS	DETECTION RESULTS	COST
1	$\#_{(N,N)}(\lambda_d, \lambda_c)$	normal	normal	0
2	$\#_{(N,A)}(\lambda_d, \lambda_c)$	normal	anomaly	3
3	$\#_{(I,I)}(\lambda_d, \lambda_c)$	intrusion	original intrusion	0
4	$\#_{(I,A)}(\lambda_d, \lambda_c)$	intrusion	anomaly	1
5	$\#_{(I,N)}(\lambda_d, \lambda_c)$	intrusion	normal	3

With respect to generalization parameters $\langle \lambda_d, \lambda_c \rangle$, the average cost of every instance in the test audit trails is defined as:

$$cost(\lambda_d, \lambda_c) = (\#_{(N,A)}(\lambda_d, \lambda_c) \times 3 + \#_{(I,N)}(\lambda_d, \lambda_c) \times 3 + \#_{(I,A)}(\lambda_d, \lambda_c) \times 1) \times \frac{1}{T} \quad (1)$$

In our framework, the average cost at $\lambda_d = \lambda_c = 0$ is the detection performance baseline as there are no model errors. In practice, the usefulness of model generalization with $\langle \lambda_d, \lambda_c \rangle$ is reflected in $cost(\lambda_d, \lambda_c)$. If $cost(\lambda_d, \lambda_c) > cost(0, 0)$, the efficiency for intrusion detection has been degraded by such model generalization. Otherwise, the model generalization is useful for intrusion detection.

4 Experiments

We have chosen a typical dataset for network intrusion detection from KDD CUP 1999 contest. This is because the dataset meets the requirements of our formal framework: *labeled audit trails and a intrusion-specific feature vector*. The specifications of the dataset are listed as follows: *training-4898431 records, test-311029 records*. Table 2 lists all the features used in our experiments. For a

Table 2. Features in the Connection Records.

TYPES (41)	FEATURES
nominal (9)	protocol_type, service, flag, land, logged_in, root_shell, su_attempted, is_hot_login, is_guest_login
discrete (15)	duration, src_bytes, dst_bytes, wrong_fragments, urgent, hot, num_failed_logins, num_compromised, num_root, num_file_creations, num_shells, num_access_files, num_outbound_cmds, count, srv_count
continuous (17)	error_rate, srv_error_rate, rerror_rate, srv_rerror_rate, same_srv_rate, diff_srv_rate, srv_diff_host_rate, dst_host_count, dst_host_srv_count, dst_host_same_srv_rate, dst_host_diff_srv_rate, dst_host_same_src_port_rate, dst_host_srv_diff_host_rate, dst_host_error_rate, dst_host_srv_error_rate, dst_host_rerror_rate, dst_host_srv_rerror_rate

detailed description of the datasets, especially for the intrusions in the datasets, please refer to ‘<http://www-cse.ucsd.edu/users/elkan/clresults.html>’.

4.1 Evaluating L1 Model Generalization

In our experimental evaluations, the model parameters $\langle \lambda_d, \lambda_c \rangle$ is adjusted to simulate different degrees of model generalization. Within our following experimental layout, the model generalization parameters are assigned from several specific values: $\lambda_d \in [0, 19]$, and $\lambda_c \in \{0, 0.1, 0.2, 0.3, 0.4, 0.5\}$.

Experimental Results. To increase the significance of model generalization in the behavior model, only part of the training audit trails (i.e., the first 500000 instances, around 10%) is utilized in our experiments. In our experimental results, the detection performance baseline with $\lambda_d = \lambda_c = 0$ is $cost(0, 0) = 0.805$. Thus, for a specific parameter pair $\langle \lambda_d, \lambda_c \rangle$, the usefulness of model generalization for intrusion detection is determined by the relation between $cost(\lambda_d, \lambda_c)$ and $cost(0, 0) (=0.805)$.

Figure 3(a) illustrates the influence of L1 model generalization on detection performance. It is obvious that the average cost of every instance is decreased with the increase of λ_d . However, it is slightly strange that, for every value of λ_d , $cost(\lambda_d, 0) = cost(\lambda_d, 0.1) = cost(\lambda_d, 0.2) = cost(\lambda_d, 0.3) = cost(\lambda_d, 0.4)$, but $cost(\lambda_d, 0.5)$ is dropped. Further study about the training audit trails and our proposed collection strategy for continuous features shows that the phenomenon is caused by the following reasons. First, the precision of continuous features in the training audit trails is 0.01, thus, there are no non-occurring feature spaces between v_i and v_{i-1} if $v_i - v_{i-1} = 0.01$. Secondly, almost all feature values of continuous features within the precision occur in the training audit trails. Therefore, the influence of model generalization on continuous features is very small. Thirdly, in our designed evaluation methodology, the neighboring feature ranges with same NSA labels will be combined into a novel feature range only when $\lambda_c = 0.5$ (Algorithm 3). Thus, the combination of neighboring feature ranges with same NSA labels is the only reason for the average cost dropping when $\lambda_c = 0.5$. In view of the above, we will only consider the model generalization with $\lambda_c = 0$ and $\lambda_c = 0.5$ in the following discussion.

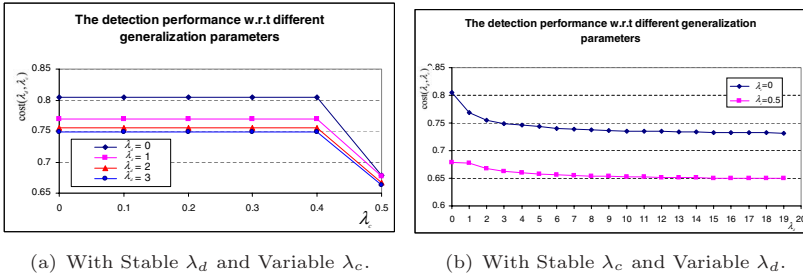


Fig. 3. Evaluation of L1 Model Generalization.

Overall Evaluation for L1 Model Generalization Figure 3(b) illustrates the overall influence of model generalization on intrusion detection. When $\lambda_c = 0$ (corresponding to the upper curve), there is no generalization for continuous features, i.e., there is only generalization for discrete features. It is obvious that the model generalization for discrete features is useful to enhance the efficiency of intrusion detection, indicated by the lower cost after more generous model generalization. The difference between the two curves implies the usefulness of model generalization for continuous features (when $\lambda_c = 0.5$). In other words, the combination of the neighboring feature ranges with the same NSA labels is useful for intrusion detection. Therefore, it can be concluded that model generalization can enhance the detection performance for intrusion detection. We provide the statistics of the detection results in the appendix.

Next, the influence of model generalization on normal behavior model and intrusive behavior model will be analyzed by our experimental results respectively. To achieve it, the average cost for normal instances is calculated by assuming that $\#_{(I,I)}(\lambda_d, \lambda_c) = \#_{(I,A)}(\lambda_d, \lambda_c) = \#_{(I,N)}(\lambda_d, \lambda_c) = 0$. Similarly, $\#_{(N,N)}(\lambda_d, \lambda_c) = \#_{(N,A)}(\lambda_d, \lambda_c) = 0$ for calculating the average cost for intrusive instances. Figure 4 gives the detection performance for normal and intrusive instances in the test audit trails.

L1 Model Generalization of the Normal Behavior Model. In Figure 4(a), it is obvious that the average cost for every normal instance is dropping with the increase of model generalization parameters (i.e., λ_d and λ_c). Thus, it is useful to generalize the behavior model, and then to enhance the detection performance for the normal behaviors. At the same time, a normal behavior is identified if it matches a signature in the normal behavior model. Thus, the enhanced detection performance is achieved by the generalization of the normal behavior model. In other words, it is profitable and necessary to generalize the normal behavior model, and then to identify more normal behaviors.

L1 Model Generalization of the Intrusive Behavior Model. The difference between Figure 4(b) and Figure 4(a) is so significant that we can reach a

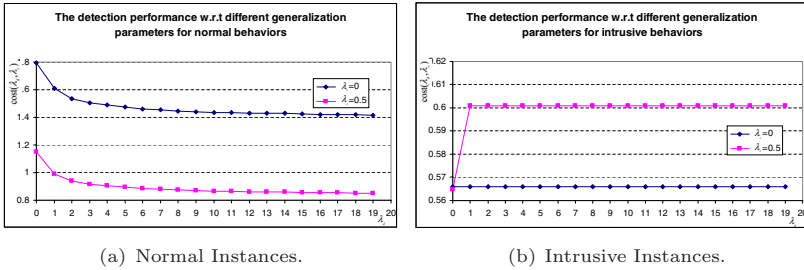


Fig. 4. Detection Performance for Normal and Intrusive Instances.

contrary conclusion to the one for normal behavior model. That is, the model generalization in intrusive behavior model is not advisable for intrusion detection. Even worse, it will aggravate the detection performance. In the experimental results, $\#_{(I,N)}(0,0) = 0$ but $\#_{(I,N)}(i,0.5) \geq 4814$ ($1 \leq i \leq 19$), indicate that the generalization of the normal behavior model will accommodate more intrusive signatures. Thus, the average cost of every instance when $\lambda_c = 0.5$ is increased considerably in Figure 4(b) at $\lambda_d = 1$.

In summary, the generalization of the normal behavior model is necessary to cover more normal signatures, but it is limited by the intrusion signatures in the generalized normal behavior model due to the generalization errors. On the other hand, the intrusion signatures should be built compactly as model generalization applied to the intrusive behavior model cannot enhance (or even can aggravate) the detection performance.

5 Conclusions and Future Work

In this paper, we designed a formal framework to evaluate the usefulness of model generalization and its implications on intrusion detection. As the first step, we only evaluated the L1 model generalization for intrusion detection for the time being. The preliminary results are identified as follows (which are consistent with the observations in [14]). With respect to L1 model generalization, the normal behavior model can be generalized more generously to accommodate more unknown normal behaviors to enhance its detection efficiency, but the intrusive behavior model should be generalized a little stingily to describe the attacks more correctly as the generalized intrusive behavior model cannot identify more intrusions. In general, the formal framework can be applied to the datasets in other fields to articulate the usefulness of model generalization as well.

Due to the time and space limits, our experiments only gave the preliminary results for the evaluation about L1 model generalization on intrusion detection. The future work is to evaluate L2 & L3 model generalization. Actually, the abstraction attack scenario in STAT family sensors (Kemmerer and Vigna[15]) is achieved by L2 & L3 model generalization. Considering that STAT is successful

in detecting new intrusions, it can be envisioned that L2 & L3 model generalization may give different pictures compared with L1 model generalization.

References

1. K.P. Anchor, J.B. Zydallis, G.H. Gunsch, and G.B. Lamont. Extending the computer defense immune system: Network intrusion detection with a multiobjective evolutionary programming approach. In *ICARIS 2002: 1st International Conference on Artificial Immune Systems Conference Proceedings*, 2002.
2. S.N. Chari and P. Cheng. BlueBox: A Policy-Driven, Host-based Intrusion Detection System. *ACM Transaction on Information and System Security*, 6(2):173–200, May 2003.
3. H. Debar, M. Dacier, and A. Wespi. A revised taxonomy for intrusion detection systems. *Annales des Telecommunications*, 55(7–8):361–378, 2000.
4. D.E. Denning. An intrusion detection model. *IEEE Transaction on Software Engineering*, SE-13(2):222–232, February 1987.
5. E. Eskin, A. Arnold, M. Prerau, L. Portnoy, and S. Stolfo. A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data. In *D. Barbara and S. Jajodia (editors), Applications of Data Mining in Computer Security*, Kluwer, 2002.
6. S.A. Hofmeyr, S. Forrest, and A. Somayaji. Intrusion detection using sequences of system calls. *Journal of Computer Security*, 6(3):151–180, 1998.
7. H. Javits and A. Valdes. The NIDES statistical component: Description and justification. SRI Annual Report A010, SRI International, Computer Science Laboratory, March 1993.
8. W. Lee, M. Miller, and S. Stolfo. Toward cost-sensitive modeling for intrusion detection. Technical Report No. CUCS-002-00, Computer Science, Columbia University, 2000.
9. W. Lee and S.J. Stolfo. A framework for constructing features and models for intrusion detection systems. *ACM Transactions on Information and System Security*, 3(4):227–261, Nov. 2000.
10. Zhuowei Li and Amitabha Das. Analyzing and Improving the Performance of a Class of Anomaly-based Intrusion Detectors. In *CoRR cs.CR/0410068*, 2004.
11. Zhuowei Li, Amitabha Das, and Jianying Zhou. Unifying Signature-based and Anomaly-based Intrusion Detection. In *Proceedings of the Ninth Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD-05)*, Hanoi, Vietnam, May. 2005. Lecture Notes in Artificial Intelligence.
12. Y. Liao and V.R. Vemuri. Using text categorization techniques for intrusion detection. In *Usenix: Security 2002*, Aug. 2002.
13. M.V. Mahoney and P.K. Chan. Learning Nonstationary Models of Normal Network Traffic for Detecting Novel Attacks. In *SIGKDD 2002*, July 23–26 2002.
14. Alfonso Valdes and Keith Skinner. Adaptive, model-based monitoring for cyber attack detection. In *Recent Advances in Intrusion Detection (RAID 2000)*, number 1907 in Lecture Notes in Computer Science, pages 80–92, Toulouse, France, October 2000. Springer-Verlag.
15. G. Vigna and R.A. Kemmerer. NetSTAT: A Network-based Intrusion Detection System. *Journal of Computer Security*, 7(1):37–71, 1999.
16. K. Wang and S.J. Stolfo. Anomalous payload-based network intrusion detection. In *Proceedings of RAID*, 2004.

A Proof of Theorem 1

Proof. We prove this by contradiction. If there exists an instance I_{FV}^i so that $I_{FV}^i \in I((R_{F_1}^a, R_{F_2}^b), F_{12})$ and $I_{FV}^i \in I((R_{F_1}^c, R_{F_2}^d), F_{12})$.

$$\begin{aligned} & I_{FV}^i \in I((R_{F_1}^a, R_{F_2}^b), F_{12}) \\ \Leftrightarrow & I_{FV}^i \in I(R_{F_1}^a, F_1) \wedge I(R_{F_2}^b, F_2) \\ \Leftrightarrow & v(I_{FV}^i, F_1) \in R_{F_1}^a, v(I_{FV}^i, F_2) \in R_{F_2}^b \end{aligned} \tag{2}$$

Similarly,

$$\begin{aligned} & I_{FV}^i \in I((R_{F_1}^c, R_{F_2}^d), F_{12}) \\ \Leftrightarrow & v(I_{FV}^i, F_1) \in R_{F_1}^c, v(I_{FV}^i, F_2) \in R_{F_2}^d \end{aligned} \tag{3}$$

Recall that there is no common feature value v_F , which falls into R_F^j and R_F^k simultaneously ($j \neq k$). From (1) and (2), we can get $R_{F_1}^a = R_{F_1}^c$ and $R_{F_2}^b = R_{F_2}^d$. Thus, $(R_{F_1}^a, R_{F_2}^b) = (R_{F_1}^c, R_{F_2}^d)$. This contradicts the assumption that the two feature ranges are different.

B Detection Results

Table 3. The Statistics of Detection Results.

$\lambda_c \rightarrow$	$\lambda_c = 0$					$\lambda_c = 0.5$				
$\lambda_d \downarrow$	$\#(N,N)$	$\#(N,A)$	$\#(I,I)$	$\#(I,A)$	$\#(I,N)$	$\#(N,N)$	$\#(N,A)$	$\#(I,I)$	$\#(I,A)$	$\#(I,N)$
0	24367	36226	108712	141724	0	37369	23224	109555	140599	282
1	28097	32496	108712	141724	0	40550	20043	109555	136067	4814
2	29557	31036	108712	141723	1	41610	18983	109555	136067	4814
3	30213	30380	108712	141723	1	42067	18526	109555	136065	4816
4	30527	30066	108712	141723	1	42357	18236	109555	136065	4816
5	30787	29806	108712	141723	1	42550	18043	109555	136065	4816
6	31080	29513	108712	141723	1	42709	17884	109555	136064	4817
7	31193	29400	108712	141723	1	42838	17755	109555	136064	4817
8	31427	29166	108712	141723	1	42944	17649	109555	136063	4818
9	31487	29106	108712	141723	1	43004	17589	109555	136063	4818
10	31601	28992	108712	141723	1	43085	17508	109555	136063	4818
11	31647	28946	108712	141723	1	43134	17459	109555	136062	4819
12	31676	28917	108712	141723	1	43184	17409	109555	136062	4819
13	31718	28875	108712	141723	1	43228	17365	109555	136062	4819
14	31755	28838	108712	141723	1	43267	17326	109555	136062	4819
15	31829	28764	108712	141723	1	43313	17280	109555	136062	4819
16	31879	28714	108712	141723	1	43346	17247	109555	136062	4819
17	31889	28704	108712	141723	1	43369	17224	109555	136062	4819
18	31920	28673	108712	141723	1	43400	17193	109555	136062	4819
19	31986	28607	108712	141723	1	43439	17154	109555	136062	4819