Evaluating the Effects of Model Generalization on Intrusion Detection Performance

Zhuowei Li[1,2], Amitabha Das[2] and Jianying Zhou[3]

[1] Indiana University, USA. `zholi@indiana.edu`
[2] Nanyang Technological University, Singapore. `asadas@ntu.edu.sg`
[3] Institute of Infocomm Research, Singapore. `jyzhou@i2r.a-star.edu.sg`

Abstract. An intrusion detection system usually infers the status of an unknown behavior from limited available ones via model generalization, but the generalization is not perfect. Most existing techniques use it blindly (or only based on specific datasets at least) without considering the difference among various application scenarios. For example, signature-based ones use signatures generated from specific occurrence environments, anomaly-based ones are usually evaluated by a specific dataset. To make matters worse, various techniques have been introduced recently to exploit too stingy or too generous generalization that causes intrusion detection invalid, for example, mimicry attacks, automatic signature variation generation etc. Therefore, a critical task in intrusion detection is to evaluate the effects of model generalization. In this paper, we try to meet the task. First, we divide model generalization into several levels, which are evaluated one by one to identify their significance on intrusion detection. Among our experimental results, the significance of different levels is much different. Under-generalization will sacrifice the detection performance, but over-generalization will not lead to any benefit. Moreover, model generalization is necessary to identify more behaviors in detection, but its implications for normal behaviors are different from those for intrusive ones.

1 Introduction

There exist two general approaches for detecting intrusions: *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 an intrusion is detected if the resource behavior deviates from normal behaviors significantly. From another aspect, there are two behavior spaces for intrusion detection (Figure 1): *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 [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. Figure 1 (a) and (b) illustrate the behavior models for SID (i.e., *intrusive behavior model*) and for AID (i.e., *normal behavior model*) in the real applications.

A critical problem. There are two quality factors within the behavior models: *inaccuracy* and *incompleteness*. For example, a part of the intrusive behavior

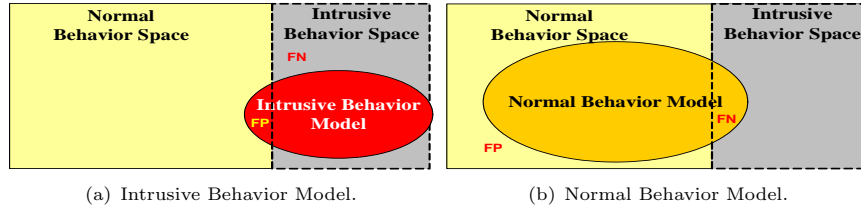(a) Intrusive Behavior Model.   (b) Normal Behavior Model.

Fig. 1. Behavior spaces and models.

model falling into the normal behavior space leads to the *inaccuracy*. Due to incompleteness, the intrusive behavior model cannot cover all intrusive behavior space, and the normal behavior model cannot cover all the normal behavior space either. In SID (Figure 1.a), model inaccuracy will lead to false positives (FP) and model incompleteness in it will lead to false negatives (FN). In contrast, model inaccuracy in the normal behavior model will lead to FNs and model incompleteness in it will cause FPs (Figure 1.b). To build a practical intrusion detection system, *it is critical to reduce the model inaccuracy and incompleteness, and thus to lower FPs and FNs in the detection phase.*

Past addressings. To make up for the incompleteness, most existing 'model building' techniques try to infer the unknown behaviors via *model generalization* (defined in Section 3), which is able to eliminate FNs in SID and to reduce FPs in AID. However, as indicated in Figure 1, it can also lead to more FPs in SID and more FNs in AID. In other words, *model generalization is two-edged for intrusion detection in principle* [9]. Various techniques have been introduced recently to exploit too stingy or too generous model generalization (*Section 2*), for example, mimicry attacks[11], mutate exploits[10], automatic signature variation generation[7] etc.

Evaluation. Thus, it is very useful to identify the utility of model generalization. We can envision at least four of its applications.

– Determine deployment conditions for an intrusion detection technique, as well as proper techniques to detect intrusions into a specific environment.
– Guide the development of an adaptive intrusion detection technique by adjusting the generalization extent.
– Alleviate concept drifting. Intrusion and application evolution patterns can determine the extent of generalization in an ad hoc deployment.
– Perform intrusion detection evaluation. According to different generalization extents, we can generate appropriate artificial datasets, which can identify the generic detection capability of a SID/AID technique.

Our contributions. We believe that our evaluation advances the research on intrusion detection in two perspectives. First, we design *a framework to evaluate the effect of model generalization*, in which model generalization is achieved at different levels according to the reasonableness of the underlying assumptions. Secondly, on a typical dataset, our experiments are performed to verify the evaluation framework, and to identify the utility of model generalization.

The remaining parts are organized as follows. Section 2 reviews the related work on model generalization. In section 3, an evaluation framework for model generalization is designed. As a case study, experiments in section 4 reveal the implications of model generalization on intrusion detection. Lastly, we draw conclusions and lay out the future work in section 5.

2 Related Work

To our knowledge, we are the first to evaluate model generalization for intrusion detection while there are two existing implicit applications of model generalization: extending behavior models and evade detection.

First, the intrusion signatures can be generalized to cover more intrusion variations. Anchor et al. [1] applied the evolutionary programming to optimize the generalization in an intrusion signature, and thus to detect more intrusion variants. Rubin et al. [8] presented a method to construct more robust signatures from existing intrusion signatures. Secondly, the normal behavior model of AID can be generalized as well. In [5, 12], existing audit trails are modeled *inexactly* to accommodate more behaviors, and thus to achieve model generalization.

Several work is proposed to utilize the false negatives introduced by model generalization. In AID techniques, mimicry attacks [11] are designed to misuse the generalization by mimicking its normal behaviors, and thus to avoid being detected. In SID techniques, model generalization is also exploited [10, 7] to generate intrusion variations, which cannot be detected either.

In summary, too generous generalization in AID will make mimicry attacks successful [11], while too stingy generalization in SID will make some attack variations undetectable [8, 10]. In our research, we try to identify the relations between the extent of generalization and detection performance.

3 An Evaluation Framework for Model Generalization

In this section, we proposed the evaluation framework for model generalization based on a theoretical basis for intrusion detection [6].

3.1 Theoretical Basis for Intrusion Detection

In a nutshell, the basis introduces three new concepts to formalize the process of intrusion detection: `feature range`, `NSA label` and `compound feature`. Every instance in a training audit trail can be represented as a feature range of a high-order compound feature, and every feature range has a NSA label, which is used to detect behaviors in test audit trails. In detail, the value of every feature in an instance can be replaced with a feature range, which is gotten by extending its value so that the extension does not conflict with other existing values. The feature ranges of all features are compounded using cartesian products to build a (training or test) behavior signature for intrusion detection.

In this framework, it is supposed that there is a training audit trail and a feature vector $FV = \{F_1, F_2, \ldots, F_n\}$. For every feature F_i, a series of feature ranges $R^1_{F_i}, R^2_{F_i}, \ldots, R^m_{F_i}$ is first mined from the training audit trails. Using feature ranges of all features, the behavior signatures $Sig_1, Sig_2 \ldots, Sig_l$ are

constructed for intrusion detection. In the detection phase, a test instance is formalized as a signature Sig_t, and it is detected in accordance with whether it matches any existing behavior signature.

3.2 Model Generalization

We first define model generalization within the context of intrusion detection.

Definition 1 (Model Generalization). *Suppose that there exists a set of behaviors associated with a resource.* **Model generalization** *is an operation that tries to identify a new behavior associated with the same resource based on the existing set of behavior instances.*

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 because of generalization errors [9]. This influence of model generalization on detection performance is generally determined by its underlying assumptions per se. In our evaluation, we first pinpoint three phases of our framework where we can use various assumptions to apply three levels of generalization, and then evaluate them one by one for model generalization. We also include a level without any generalization in which the behaviors in the training audit trails are represented precisely.

In the follow-up subsections, we describe the methods to evaluate the three levels of generalizations which moves the model from most specialized to most generalized as we move down the level (from L0 to L3).

3.3 L0 Without Generalization

Suppose that for a feature F, there exists a series of feature values, $v_1, v_2, \ldots, v_l$. Without generalization, every feature value v_i is regarded as a feature range with its upper and lower bounds equal to v_i. In this way, the instances in the training audit trails are represented *precisely* by the signatures generated from these feature ranges. Note that, for F, we have not inferred the NSA label of unknown feature subspace between any two feature values.

3.4 L1 Model Generalization

For every feature, to achieve L1 generalization, we assume that the unknown parts in its feature space have the same NSA label as its neighboring feature values. Obviously, inherent in this assumption is a concept of distance. Therefore, due to the lack of distance concept in nominal features, we will only discuss the L1 generalization on numerical (discrete and continuous) features, and regard every feature value of a nominal feature as a feature range. For convenience, we use two more notations on a feature range R_F^i: $Upp(R_F^i)$ is its upper bound and $Low(R_F^i)$ is its lower bound. With respect to a feature value v_i, an initial feature range R_F^i will be formed with $Upp(R_F^i) = Low(R_F^i) = v_i$.

L1 generalization is described in algorithm 1. In this generalization, one critical step is to split the unknown subspace $(v_i, v_{i+1}) = (Upp(R_F^i), Low(R_F^{i+1}))$ $(i + 1 \leq l)$, and allocate the two parts to existing neighboring ranges R_F^i and R_F^{i+1}. We use several strategies and evaluate them in our framework. These are:

Algorithm 1 L1 model generalization for a discrete/continuous feature F.

Require: (1) $R_F^1, R_F^2, \ldots R_F^l$. (2)$\varepsilon$ (ε_d for discrete features and ε_c for continuous features).
1: **for** $i = 1$ to $l - 1$ **do**
2:    Determine a splitting border S within $(Upp(R_F^i), Low(R_F^{i+1}))$;
3:    Split $(Upp(R_F^i), Low(R_F^{i+1}))$ into two parts $(Upp(R_F^i), S]$ and $(S, Low(R_F^{i+1}))$;
4:    $R_F^i = (Low(R_F^i), S]$; $R_F^{i+1} = (S, Upp(R_F^{i+1}))$;
5: **end for**
6: i=1;
7: **while** $i < l$ **do**
8:    **if** $Low(R_F^{i+1}) - Upp(R_F^i) \leq \varepsilon$, and $L(R_F^i) = L(R_F^{i+1})$ **then**
9:       Merge R_F^{i+1} into R_F^i; Delete R_F^{i+1}; $l = l - 1$;
10:    **else**
11:       $i = i + 1$;
12:    **end if**
13: **end while**

---

(1) no splitting (2) equal splitting, (3) frequency-based splitting, (4) intrusion-specific splitting. Note that, in Algorithm 1, the merging step for feature ranges (i.e., lines 6-12) is selective after the splitting step (i.e., lines 1-5). This step for merging range is also a generalization operation in L1 generalization.

1. **L1.1: No splitting**. If we do not conduct the merging step either, the L1.1 generalization actually becomes same as L0, i.e., no generalization.
2. **L1.2: Splitting it equally**. The unknown interval between v_i and v_{i+1} is split at the midpoint $S = \frac{v_i + v_{i+1}}{2}$. That is, $(v_i, S]$ is assigned the same NSA label as v_i, and (S, v_{i+1}) is assigned the same NSA label as v_{i+1}.
3. **L1.3: Frequency-based Splitting**. Let the frequency of v_i in the training audit trails be f_{v_i}. Then, the splitting point is $S = v_i + (v_{i+1} - v_i) * \frac{f_{v_i}}{f_{v_i} + f_{v_{i+1}}}$. $(v_i, S]$ is assigned as $L(v_i)$, and (S, v_{i+1}) is assigned as $L(v_{i+1})$.
4. **L1.4: Intrusion specific splitting**. Given a predefined generalization parameter G_{in} for intrusions. For a pair of neighboring values v_i and v_{i+1}, if $L(v_i) = $ N and $L(v_{i+1}) = $ A, $S = v_{i+1} - G_{in}$. If $L(v_i) = $ A and $L(v_{i+1}) = $ N, $S = v_i + G_{in}$. Otherwise, $S = \frac{v_i + v_{i+1}}{2}$. $(v_i, S]$ is assigned as $L(v_i)$, and (S, v_{i+1}) is assigned as $L(v_{i+1})$.

In addition, we also evaluate the merging step for every splitting strategy.

In the detection phase, every instance is formalized as Sig_t by replacing every value with its feature range. Finally, we evaluate whether Sig_t matches any signature in $\Omega(F_{1\ldots n})$. If matched, it is identified by that signature. Otherwise, Sig_t will further be evaluated by L2 generalization evaluation processes.

3.5 L2 Model Generalization
After the L1 model generalization, all the (nominal, discrete, and/or continuous) features are uniformly represented by a series of feature ranges. In L2 model generalization, we will utilize the relations between feature ranges rather than values, which are measured by the distance of two signatures. To this end, let us first define a distance function of two signatures in the behavior models.

Signature distance. Let $R(Sig_1, F_i)$ denote the feature range of F_i in a signature Sig_1. For any two signatures, Sig_1 and Sig_2, their distance is:

$$D(Sig_1, Sig_2) = \sum_{i=1}^{n} \delta(Sig_1, Sig_2, F_i)$$

Where, $\delta(Sig_1, Sig_2, F_i) = \begin{cases} 0, \text{ if } R(Sig_1, F_i) = R(Sig_2, F_i); \\ 1, \text{ otherwise.} \end{cases}$

Evaluating L2 generalization. L2 generalization is achieved by the following two generalization operations. `L2.1: grouping feature ranges`. If several feature ranges of a feature are interchangeable in $\Omega(F_{1\ldots n})$ without loss of signature distinguishability, they will be combined into a group. `L2.2: mutating feature ranges`. For a feature, its feature range in a signature can be mutated to any of its other feature ranges without loss of signature distinguishability.

Grouping feature ranges. For a feature F_i, if a feature range in $\Omega(F_{1\ldots n})$ is interchangeable with another feature range without loss of signature distinguishability (i.e. without changing its NSA label), their significance is equal to each other. We can group these feature ranges in constructing behavior models. As a special case, a feature range can form a group by itself. In this way, we can form a series of groups for F_i, $G_{F_i} = \{G_{F_i}^1, G_{F_i}^2, \ldots\}$ such that for any feature range $R_{F_i}^j$, there is a group $G_{F_i}^k$, $R_{F_i}^j \in G_{F_i}^k$. Finally, we achieve a grouping scheme for all features in the feature vector: $G_{FV} = \langle G_{F_1}, G_{F_2}, \ldots, G_{F_n} \rangle$.

For two signatures Sig_1 and Sig_2 in $\Omega(F_{1\ldots n})$, they are *equivalent* to each other with respect to G_{FV} based on the following rule.

$$Sig_1 \overset{G_{FV}}{=} Sig_2 \Leftrightarrow \exists i(\delta(Sig_1, Sig_2, F_i) = 1) \tag{1}$$
$$\wedge (\exists j \{R(Sig_1, F_i), R(Sig_2, F_i)\} \subset G_{F_i}^j)$$

For any two equivalent signatures, they are *compatible* if they have the same NSA label. Otherwise, they are *conflict* to each other in the behavior models.

The behavior models can be generalized by grouping feature ranges. For example, for signatures "$\langle a, 1, E \rangle$" and "$\langle b, 2, F \rangle$", if 'a' and 'b' are grouped, the behavior models can be enlarged by two additional signatures "$\langle b, 1, E \rangle$" and "$\langle a, 2, F \rangle$". Essentially, like in Genetic Algorithm [4] we are allowing *crossover* operation between signatures by interchanging the feature ranges in a group.

Algorithm 2 Evaluating a test signature via grouping.

Require: (1) $\Omega(F_{1\ldots n})$; (2) Sig_t; and (3) np_g.
1: Initialization, $StatusList = \emptyset$
2: **for** every signature $Sig_1 \in \Omega(F_{1\ldots n})$ **do**
3:     calculate $D(Sig_t, Sig_1)$
4:     **if** $D(Sig_t, Sig_1) \leq np_g$ **then**
5:         /* if $R(Sig_1, F_i) \neq R(Sig_t, F_i)$, $P = \{R(Sig_1, F_i), R(Sig_t, F_i)\}$ */
6:         Enumerate all feature range pairs $P_1, \ldots, P_k$ ($k \leq np_g$);
7:         **if** no conflicting signatures w.r.t. $P_1, P_2, \ldots, P_k$ **then**
8:             Append status(es) of Sig_1 into $StatusList$; /*Lemma 3*/
9:         **end if**
10:     **end if**
11: **end for**
12: determine the detection results based on $StatusList$;

Moreover, to measure the diversity in G_{FV}, the number of grouping points np_g is utilized in the detection phase. In other words, if the grouping scheme

does not exist, there are at least $n - np_g$ equivalent feature ranges between Sig_t and any signature Sig_i in the behavior models. The larger the parameter np_g is, the more diverse the group operation is. Given np_g and $\Omega(F_{1...n})$, a test instance is evaluated as in Algorithm 2.

If the output is an anomaly, we will evaluate Sig_t using mutation operation.

Mutating feature ranges. Neglecting some features will cause a signature to identify more behaviors. For example, suppose that there is a signature "*height $\in$ (156cm, 189cm], weight $\in$ (45kg, 75kg], and Nationality = USA*". If all three features are used, it cannot identify the instance '*height = 174cm, weight = 65kg, and Nationality = China*' will not be identified. But if '*Nationality*' is ignored, the signature will identify the instance. Essentially, ignoring features is equal to the mutation operation in Genetic Algorithms[4]. One condition of the mutation is that it should not lead to any contradiction in the existing signatures. For example, if we let F_1 and F_2 mutate, signatures "$\langle a, b, c, d \rangle$" in $N(F_{1...4})$ and "$\langle x, y, c, d \rangle$" in $A(F_{1...4})$ will contradict to each other.

Furthermore, we use a mutation point number np_m to measure the diversity of the mutation process. In the detection phase, given np_m and $\Omega(F_{1...n})$, the unidentified test signature Sig_t will be evaluated as in Algorithm 3.

Algorithm 3 Evaluating a test signature via mutation.

Require: (1) $\Omega(F_{1...n})$; (2) Sig_t; and (3) np_m.
1: Initialization, $StatusList = \emptyset$
2: **for** every signature $Sig_1 \in \Omega(F_{1...n})$ **do**
3:     calculate $D(Sig_t, Sig_1)$
4:     **if** $D(Sig_t, Sig_1) \leq np_m$ **then**
5:         /*if $R(Sig_1, F_i) \neq R(Sig_t, F_i)$, F_i will be mutated*/
6:         Enumerate all mutated features $F_{m_1}, \ldots, F_{m_k}$ ($k \leq np_m$);
7:         **if** no conflicting signatures w.r.t. $F_{m_1}, F_{m_2}, \ldots, F_{m_k}$ **then**
8:             Append the status(es) of Sig_1 into $StatusList$;
9:         **end if**
10:     **end if**
11: **end for**
12: Determine the detection results based on $StatusList$;

3.6 L3 Model Generalization

If the test signature Sig_t cannot be identified by L1 and L2 generalization, it will be identified by the signature(s) with the minimum distance to it.

Nearest signatures. We assume that the test signature has the same NSA label as its nearest signature(s) in the behavior models, which is measured by its minimum distance to all signatures in $\Omega_{F_{1...n}}$,

$$D_{min}(Sig_t, \Omega(F_{1...n})) = \min_{Sig_i \in \Omega(F_{1...n})} D(Sig_i, Sig_t)$$

3.7 Measuring the Detection Performance

We assign a cost scheme as in Table 1 to quantify the detection performance, and calculate the average detection cost of an instance in the test audit trails. If the behavior is identified correctly, the cost is 0. Otherwise, we can assign 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 identifying an intrusion. Specifically, suppose that there are T instances in the test audit trails. The number of false positives is $\#_{NA}$, and for false negatives, it is $\#_{IN}$. The average cost of a test instance is defined

INDEX	NOTATIONS	ORIGINAL CLASS	DETECTION RESULTS	COST
1	$\#_{NN}$	normal	normal	0
2	$\#_{NA}$	normal	anomaly	3
3	$\#_{II}$	intrusion	original intrusion	0
4	$\#_{IA}$	intrusion	anomaly	1
5	$\#_{IN}$	intrusion	normal	3

Table 1. Detection results and their costs.

as: $cost = \#_{NA} \times 3 + \#_{IN} \times 3 + \#_{IA} \times 1 \times \frac{1}{T}$. In addition, the average cost in absence of any generalization gives the reference baseline, $cost_{base}$, of the detection performance. In practice, the usefulness of model generalization is reflected in the relation between its average cost and $cost_{base}$. If $cost > cost_{base}$, its performance has been degraded by such model generalization. Otherwise, the model generalization can be assumed to be useful for intrusion detection.

4 Experiments: A Case Study

We have chosen a typical dataset from KDD CUP 1999 contest [3], which meets the requirements of our framework: *labeled audit trails* and *an intrusion-specific feature vector*, in which $\varepsilon_d = 1$ and $\varepsilon_c = 0.01$. In order to keep the computation within reasonable limits, we sample instances from the datasets: 10000 instances from the total 4898431 training instances and 500 instances from 311029 test instances randomly. For convincing, we give three pairs of such training and test samples. We have performed our experiments on larger samples, but the experimental results on our larger samples have the same characteristics to the results on the current samples.

4.1 Without Model Generalization

Table 2 lists the detection results when there is no generalization, and they are regarded as the baseline $cost_{base}$. Also in this table, the 2nd and 3rd columns give the numbers of normal and intrusive instances in every sample pair.

Sample	Norm.	Intru.	$\#_{NN}$	$\#_{NA}$	$\#_{II}$	$\#_{IA}$	$\#_{IN}$	$cost$
Pair 1	103	397	0	103	203	193	1	1.01
Pair 2	91	409	0	91	216	193	0	0.932
Pair 3	108	392	5	103	193	198	1	1.02

Table 2. L0: without model generalization.

Among the detection results, more than half of intrusive instances are identified correctly (denoted by $\#_{II}$), but, in comparison, almost all normal instances are detected incorrectly. To some extent, it indicates that the normal behaviors are of great variety, and more generalization is needed to infer their statuses.

L1	G_{in}	with the range merging step:						without the range merging step:					
		$\#_{NN}$	$\#_{NA}$	$\#_{II}$	$\#_{IA}$	$\#_{IN}$	$cost$	$\#_{NN}$	$\#_{NA}$	$\#_{II}$	$\#_{IA}$	$\#_{IN}$	$cost$
L1.4	0	35	68	280	115	2	0.65	6	97	278	118	1	0.824
L1.4	1	35	68	280	115	2	0.65	6	97	278	118	1	0.824
L1.4	2	35	68	281	114	2	0.648	6	97	278	118	1	0.824
L1.4	3	35	68	280	115	2	0.65	6	97	279	117	1	0.822
L1.4	4	35	68	281	114	2	0.648	6	97	279	117	1	0.822
L1.4	5	35	68	281	114	2	0.648	6	97	278	118	1	0.824
L1.4	10	35	68	280	115	2	0.65	6	97	279	117	1	0.822
L1.4	20	35	68	281	114	2	0.648	6	97	278	118	1	0.824

Table 3. L1.4 generalization on the 1st sample pair.

4.2 Evaluating L1 Model Generalization

Table 3 gives the detection performance on the 1st sample pair with L1.4 generalization, where $G_{in} \in \{0, 1, 2, 3, 4, 5, 10, 20\}$. Obviously, the value of G_{in} has no influence on the detection performance in all aspects. The same phenomenon is held in the other two sample pairs of our experiments as well. Thus, we let $G_{in} = 0$ in the following experiments.

The utility of the range merging step. In Table 3, the range merging step has contributed much to the performance enhancement by identifying more normal behaviors. Note that the range merging step has little effect on the identification ability for intrusive behaviors.

Table 4 gives the evaluation results on the four scenarios of L1 generalization. We analyze their utility for intrusion detection, and their difference.

L1	with the range merging step						without the range merging step					
	$\#_{NN}$	$\#_{NA}$	$\#_{II}$	$\#_{IA}$	$\#_{IN}$	$cost$	$\#_{NN}$	$\#_{NA}$	$\#_{II}$	$\#_{IA}$	$\#_{IN}$	$cost$
(**Pair 1**) Normal:Intrusion=103:397												
L1.1	6	97	241	155	1	0.898	0	103	203	193	1	1.01
L1.2	35	68	281	114	2	0.648	6	97	279	117	1	0.822
L1.3	35	68	280	115	2	0.65	6	97	278	118	1	0.824
L1.4	35	68	280	115	2	0.65	6	97	278	118	1	0.824
(**Pair 2**) Normal:Intrusion=91:409												
L1.1	3	88	263	144	2	0.828	0	91	216	193	0	0.932
L1.2	39	52	294	113	2	0.55	7	84	294	113	2	0.742
L1.3	39	52	292	115	2	0.554	7	84	294	113	2	0.742
L1.4	39	52	290	117	2	0.558	7	84	290	117	2	0.75
(**Pair 3**) Normal:Intrusion=108:392												
L1.1	9	99	234	154	4	0.926	5	103	193	198	1	1.02
L1.2	45	63	273	115	4	0.632	10	98	273	115	4	0.842
L1.3	45	63	273	115	4	0.632	10	98	273	115	4	0.842
L1.4	45	63	273	115	4	0.632	10	98	273	115	4	0.842

Table 4. L1 model generalization (L1.1$\sim$4, $G_{in} = 0$).

The utility of the unknown subspace splitting step. L1.1 generalization without the range merging step is L0, which has no generalization at all. Comparing the detection results in Table 4 and Table 2, it is apparent that the generalization led to by the unknown subspace splitting step is useful to identify more instances, and significantly so for intrusive behaviors.

The difference between L1.2/3/4. The new false negatives caused by L1 generalization is negligible in all three sample pairs (with 1, 2 and 3 additional ones). Overall, L1.2/3/4 have little difference on the detection results.

In summary, L1 generalization with L1.2/3/4 and range merging is useful but the detection results are not sensitive to the splitting strategies. Therefore, we arbitrarily select L1.4 with $G_{in} = 0$ in the following experiments.

4.3 Evaluating L2 Model Generalization

Figures 2 and 3 list the evaluation results highlighting the influence of grouping and mutation operations on intrusion detection. In both figures, we only illustrate $\#_{NA}$, $\#_{IA}$ and $\#_{IN}$ but the numbers of $\#_{NN}$ and $\#_{II}$ can be deduced with ease since the total of normal and intrusive behaviors remains constant.
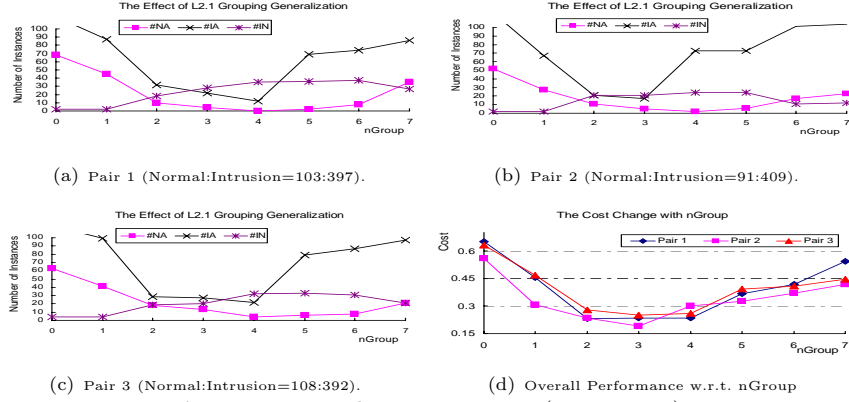

(a) Pair 1 (Normal:Intrusion=103:397).

(b) Pair 2 (Normal:Intrusion=91:409).

(c) Pair 3 (Normal:Intrusion=108:392).

(d) Overall Performance w.r.t. nGroup

Fig. 2. L2 generalization-grouping (nMutate=0).

L2.1 grouping generalization. As indicated in Figure 2, the grouping operation enhances intrusion detection, and the detection performance on the three samples have the same characteristics. Specifically, the overall detection performance improves because of a reduction in the detection cost. With the increase of $nGroup$, $\#_{NN}$ and $\#_{II}$ increase while $\#_{NA}$ and $\#_{IA}$ decrease, all of which are desirable. One negative aspect of grouping generalization is the increase of $\#_{IN}$ with the increase of $nGroup$.

Overall, the generalization from the grouping mechanism is useful for intrusion detection even though it will lead to a few more false negatives. We choose $nGroup = 3$ in the following experiments.

L2.2 mutation generalization. In Figure 3, the improvements caused by L2.2 mutation generalization is not that significant as L2.1 or L1 generalization. The decreased extent of false positives, $\#_{NA}$, is neutralized by the increased extent of false negatives, $\#_{IN}$. This fact is also reflected by the overall detection cost in subfigure 3.d, which is reduced only by a very small extent. The mutation operation will further worsen the negative aspects in grouping generalization.

In our case study, the L2.2 mutation generalization is useful but it is not that significant. We select $nMutate = 5$ in evaluating L3 model generalization.

4.4 Evaluating L3 Model Generalization

In evaluating L3 generalization (Table 5), $nGroup = 3$ and $nMutate = 5$. In the sample pair 1, all the normal behaviors are identified correctly, and most intrusions are also identified correctly (88.7%=352/397). In pair 1/2/3, most normal behaviors can be identified correctly with fewer false positives (i.e., $\#_{NA}$, which decreases with more generalization) after the model generalization (from L1 to
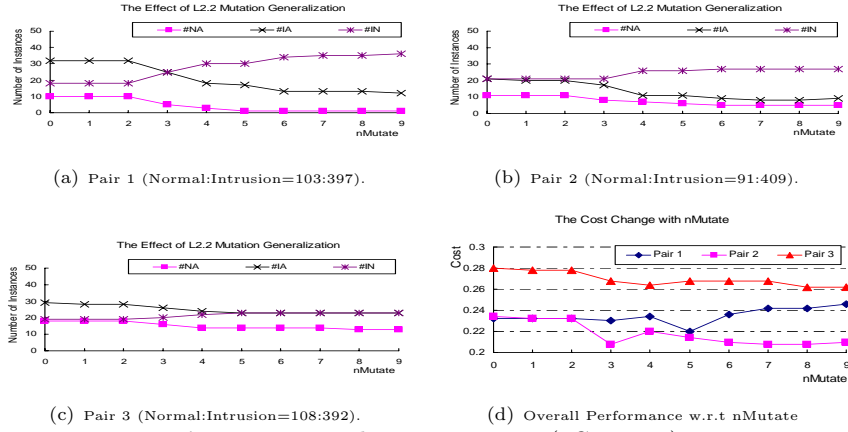
(a) Pair 1 (Normal:Intrusion=103:397).

(b) Pair 2 (Normal:Intrusion=91:409).

(c) Pair 3 (Normal:Intrusion=108:392).

(d) Overall Performance w.r.t nMutate

Fig. 3. L2 generalization-mutation (nGroup=3).

Sample	$\#_{NN}$	$\#_{NA}$	$\#_{II}$	$\#_{IA}$	$\#_{IN}$	cost
Pair 1	103	0	352	4	41	0.254
Pair 2	89	2	375	9	25	0.18
Pair 3	105	3	353	2	37	0.244

Table 5. L3 generalization (nGroup=3,nMutate=5).

L3). In contrast, even though more intrusive behaviors are identified correctly as well with more generalization, the false negatives (i.e., $\#_{IN}$) increase to a large extent (in comparison with Table 2).

4.5 The Implications of Model Generalization

In summary, model generalization is necessary for intrusion detection for identifying more behaviors correctly. The significance of every level of model generalization for intrusion detection is summarized in Table 6.

Levels	FP	FN	Utility
L0, L1.1	-	-	they act as an evaluation baseline to indicate whether model generalization is necessary for intrusion detection. We also found that most intrusions are identified even without generalization.
L1.2/3/4	↓	-	They improve the detection performance in our case study, significantly for intrusive behaviors. Most importantly, they lead to only a few more false negatives. Their difference are negligible.
Range Merging in L1	↓	-	It is very useful to infer the statuses for normal behaviors, but it contributes less in identifying intrusive behaviors. Another good point is that it does not lead to more false negatives.
L2.1	↓	↑	The identification capability is significantly lifted with decreasing anomalies. However, there is an optimal value for the number of grouping points, which should be determined in advance.
L2.2 /L3	↓	↑	The identification capability is slightly lifted with decreasing anomalies. But the increase of false negatives is so large that we should neglect the increase of identification capabilities.

Table 6. The significance of different levels of model generalization. The symbol '↓' represents 'decrease' and the symbol '↑' represents 'increase'. '-' denotes that it will not affect the parameter.

5 Conclusions and Future Work

In this paper, we designed a formal framework to evaluate the effect of various model generalization on intrusion detection in accordance with the reasonableness of its underlying assumptions. In a case study, we applied it to identify the implications of model generalization. We found that L1 generalization is generally useful to identify more 'novel' behaviors, especially for normal behaviors. L2.1 generalization will benefit intrusion detection by significantly improving the identification capability with slight increase of false negatives. The gains and losses from applying L2.2 and L3 generalization should be considered seriously under different application scenarios.

Even though our evaluation framework is generally applied to most scenarios for intrusion detection, it should be pointed out that our conclusions are only based on our case study on a typical dataset for intrusion detection. Our further work is to collect datasets to further evaluate the utility of model generalization in other areas, such as bioinformatics.

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 Infomation and System Security*, 6(2):173–200, May 2003.
3. The KDD CUP 1999 Contest Dataset. As of january, 2006. http://www-cse.ucsd.edu/users/elkan/clresults.html, 1999.
4. David E. Goldberg. *Genetic algorithms in search, optimization, and machine learning.* Addison-Wesley Pub. Co., 1989.
5. W. Lee and S.J. Stolfo. A framework for contructing features and models for intrusion detection systems. *ACM Transactions on Information and System Security*, 3(4):227–261, Nov. 2000.
6. Zhuowei Li, Amitabha Das, and Jianying Zhou. Theoretical basis for intrusion detection. In *Proceedings of 6th IEEE Information Assurance Workshop (IAW)*, West Point, NY, USA, June 2005. IEEE SMC Society.
7. Shai Rubin, Somesh Jha, and Barton P. Miller. Automatic generation and analysis of nids attacks. In *Proceedings of the 20th Annual Computer Security Applications Conference (ACSAC'04)*, pages 28–38, 2004.
8. Shai Rubin, Somesh Jha, and Barton P. Miller. Language-based generation and evaluation of nids signatures. In *Proceedings of S&P'05*, pages 3–17, 2005.
9. Alfonso Valdes and Keith Skinner. Adaptive, model-based monitoring for cyber attack detection. In *Proceedings of RAID'00*, pages 80–92, October 2000.
10. Giovanni Vigna, William Robertson, and Davide Balzarotti. Testing network-based intrusion detection signatures using mutant exploits. In *Proceedings of CCS'04*, pages 21–30, 2004.
11. David Wagner and Paolo Soto. Mimicry attacks on host-based intrusion detection systems. In *Proceedings of CCS'02*, pages 255–264, 2002.
12. K. Wang and S.J. Stolfo. Anomalyous payload-based network intrusion detection. In *Proceedings of RAID*, 2004.