The Utility of Partial Knowledge in Behavior Models: an Evaluation for Intrusion Detection

Zhuowei Li and Amitabha Das
(Corresponding author: Zhuowei Li)

School of Computer Engineering, Nanyang Technological University
50 Nanyang Avenue, Singapore, 639798 (Email: zhwei.li@pmail.ntu.edu.sg)

(Received June 9, 2005; revised and accepted July 4, 2005)

Abstract

To enlarge the detection capability of an incomplete behavior model, model generalization is necessary to make every behavior signature identify more behavior instances. In this paper, based on a general intrusion detection framework, M out of N features in a behavior signature are utilized to detect the behaviors ($M \leq N$) instead of using all N features. This is because M of N features in a signature can generalize the behavior model to incorporate unknown behaviors, which are useful to detect novel intrusions outside the known behavior model. However, the preliminary experimental results show that all features of any signature should be fully utilized for intrusion detection instead of M features in it. This is because the M of N features scheme will make the behavior identification capability of the behavior model lost by detecting most behaviors as 'anomalies' or 'alarms'.

Keywords: Intrusion detection, machine learning, model evaluation, model generalization, security

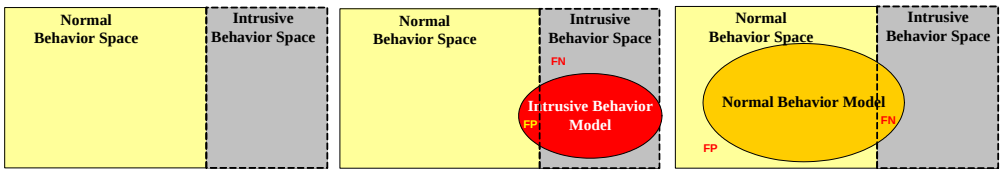
1 Introduction

In general, there exist two approaches for detecting intrusions into computer systems and networked information systems [6]: signature-based intrusion detection (a.k.a. misuse detection), where an intrusion is detected if its behavior matches existing intrusion signatures, and anomaly-based intrusion detection, where 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, signature-based intrusion detection is based on knowledge in intrusive behavior space, and anomaly-based intrusion detection is based on knowledge in normal behavior space [3]. Perfect detection of intrusions can be achieved only if we have a complete model of anyone of the two behavior spaces, because what is not bad is good and vice versa ideally.

However, it is difficult to model any such behavior space completely and correctly in reality, and Figure 1 (b) and (c) illustrate real behavior models for SID (i.e., intrusive behavior model) and for AID (i.e., normal behavior model) [3]. As the figure indicates, there exist model errors in the behavior model for SID techniques as well as AID ones. For example, a part of intrusive behavior model in SID falls into normal behavior space. At the same time, the intrusive behavior model cannot cover all intrusive behavior space, and the normal behavior model cannot cover all normal behavior space either.

To make up for the incompleteness in the existing audit trails, most existing intrusion detection techniques try to infer the unknown behaviors via *model generalization* [4, 5, 12, 14, 15, 17]. However, model generalization may lead to model errors (Figure 1), and model generalization cannot solve the incompleteness problem completely under most scenarios because it infers part but not all of unknown behaviors. In this paper, as a part of effort to analyze the problems in intrusion detection, we evaluate the usefulness of the model generalization led to by M of N features in a signature with respect to its influence on the detection performance of the behavior model. For example, suppose that there exist one signature '$height \in (156cm, 189cm], weight = (45kg, 75kg], and\ Nationality = Singapore$'. If all N(=3) features are utilized, the instance '$height = 174cm, weight = 65kg, and\ Nationality = China$' is not identified by the signature. But if only any M(=2) features are utilized, the signature will identify the instance.

Under our proposed framework, the evaluation will be done as follows. First, the behavior signatures in the existing audit trails are extracted with all features. Secondly, with specified M, the behavior model is generalized in the detection phase. In addition, an average detection cost of any instance in the test audit trails is defined to quantify the detection performance. As a special case,



(a) Behavior spaces (b) Intrusive behavior model (c) Normal behavior model

Figure 1: Behavior spaces and models

for the behavior model using all features, its average cost serves as a baseline in evaluating the usefulness of the scheme for intrusion detection. However, our preliminary experimental results show the failure of the M of N features scheme for intrusion detection. In other words, for intrusion detection, all features should be utilized to identify an behavior using signatures in the behavior model(s).

In summary, our main contributions in this paper are as follows. (1) New concepts, namely 'feature range', 'compound feature' and 'behavior signature', are introduced to build a general intrusion detection framework. (2) Model generalization led to by M of N features in a signature is discussed, and then an evaluation methodology on it is designed. In addition, an average detection cost function is defined to quantify the detection performance for intrusion detection.

The remaining parts of this paper are organized as follows. Section 2 talks about the related work. Section 3 describes the formal intrusion detection framework in detail. In Section 4, an intrusion detection methodology using the M of N features scheme is introduced, and an evaluation methodology is also designed. Experiments in Section 5 are done to reveal the useless of the scheme on intrusion detection. Lastly, we draw conclusions and layout the future work in Section 6.

2 Related Work

Our research work in this paper is generally related to model generalization in the behavior model. As indicated above, for most intrusion detection techniques, the behavior model is trained from an existing audit trails [19]. If there are new instances in the detection phase, these intrusion detection techniques should determine whether they are anomalous/intrusive using the part of behavior model generalized from the existing audit trails, which is achieved by so-called *model generalization*.

First, the intrusion signatures in SID techniques can be generalized to cover more behavior space, i.e., the intrusive behavior model in Figure 1 is extended. In [1], using a fitness function which depends on false positive rate and detection rate, the generalized signatures (represented by a *finite state transducer*) is optimized by the evolution programming. In general, the model generalization on intrusion signatures can solve the intrusion variations detection problem partially.

Secondly, the normal behavior model of AID techniques can be generalized as well to detect novel instances, and it can be done in several ways. Based on a distance metric and a distance threshold [6, 7, 13, 16, 18], 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 [9, 10, 12, 14, 18], 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. Among these two styles, the existing audit trails are modeled inexactly to accommodate more resource behaviors in the profiles, and thus to achieve the model generalization.

However, most of past works only gave the whole efficiency of an intrusion detection technique with such model generalization. Even though the model generalization is critical for intrusion detection, there is not any evaluation about whether the model generalization within these intrusion detection techniques is useful to enhance their efficiency. Most significantly, even though it is critical for the behavior model, there are no answer to the following question: *what extent of the behavior model should be generalized for intrusion detection?* This is due to the difficulty of these techniques in splitting the efficiency led to only by model generalization. Fortunately, the difficulty can be solved in our proposed framework. Our objective in this paper is to do such evaluation, and then to find its implications on intrusion detection.

3 A Formal Intrusion Detection Framework

Summarized from the principles of signature-based and anomaly-based intrusion detection [19], any intrusion detection system builds the models of the resources/processes using a set of features, or a *feature vector* $FV = \{\mu_1, \mu_2, \ldots, \mu_n\}$, where μ_i is a feature in the feature set. Every feature in the feature vector can be one property at the current timestamp (e.g., *the fields in the current packet*), or one context-sensitive property before the current timestamp (e.g., *the system-call events in stide [8]*). Thus, the feature vector is general enough to represent most (or even all) inputs of intrusion detection systems.

In general, a feature μ_i in the feature vector can be categorized into *nominal*, *discrete* or *continuous* one. Discrete and continuous features are ordinal feature types,

such that the feature values can be ordered according to their values, and a distance can be determined between any two values according to some distance metric. On the other hand, nominal features are not quantifiable and hence do not possess any order amongst them. Discrete features are isolated points in a continuous spectrum of values. While the number of values for a continuous feature can be infinitely many, the number of discrete values is often few or finite. A feature vector for intrusion detection can contain any number of nominal, discrete, and/or continuous features.

Example 3.1 *Let us assume that a feature vector consists of three features μ_1, μ_2 and μ_3, thus, $FV = \{\mu_1, \mu_2, \mu_3\}$. Specifically, within the context of network intrusion detection, μ_1 is the 'service', μ_2 is the 'number of urgent packets', and μ_3 is the 'SYN error rate' in a network session. The example instances of the feature vector are listed below in Table 1, and it will be utilized to illustrate the following definitions and concepts in our framework. Note that the 'status' for every instance in this table gives the condition of the process when collecting the instance. Obviously, in $FV = \{\mu_1, \mu_2, \mu_3\}$, μ_1 is a nominal feature, μ_2 is a discrete feature, and μ_3 is a continuous feature.*

Table 1: The example instances

μ_1	μ_2	μ_3	STATUS
TCP	1	0.01	normal
ICMP	2	0.04	normal
HTTP	6	0.10	intrusion$_3$
TCP	4	0.08	intrusion$_1$
UDP	5	0.06	intrusion$_2$
UDP	8	0.14	intrusion$_4$
HTTP	6	0.10	normal
UDP	7	0.02	normal
UDP	8	0.14	intrusion$_2$

3.1 Definitions and Concepts

For any feature μ in the audit trails, its meaningful domain will be called its *value space* or *feature space*, denoted as $\Omega(\mu)$. Any value of the feature μ in the audit trails is defined as a *feature value* $\nu_j (j \geq 0)$ in its value space. Based on whether it is found in normal audit trails or in the audit trails left by intrusions, a feature value is labeled as *N: normal*, *S: suspicious*, or *A: anomalous*. More specifically, if a feature value occurs only in the normal audit trails, it is a *normal* feature value. If it occurs only in the anomalous audit trails, for example, the intrusion signatures in SID, it is labeled as an *anomalous* feature value. Otherwise, i.e., if it occurs in both normal and anomalous audit trails, it is labeled as a *suspicious* feature value. For brevity, we will refer to the normal, suspicious, or anomalous labels as the *NSA label* of the feature value ν_j, denoted as $\Lambda(\nu_j)$.

Example 3.2 *Take the same scenario as example 3.1. For the feature μ_2 in Table 1, $\Omega(\mu_2) = [1, 8]$, $\Lambda(1) = 'N'$, and $\Lambda(4) = 'A'$. For the feature μ_1 in the same table, $\Lambda(ICMP) = 'N'$, and $\Lambda(TCP) = 'S'$.*

3.1.1 Feature Range

Definition 3.1 (feature range) *For a feature μ, a feature range $\tau(\nu_1, \nu_2)$ is the range between any two feature values ν_1 and ν_2 in its feature space, in which all of its feature values fall between ν_1 and ν_2.*

For a discrete or continuous feature, the feature range $\tau(\nu_1, \nu_2)$ includes all the feature values falling between ν_1 and ν_2, i.e., $\tau(\nu_1, \nu_2) = [\nu_1, \nu_2]$. For a nominal feature, every feature value is independent. Thus, each nominal feature value is referred to as a feature range in this paper, i.e., $\nu_1 = \nu_2$, and $\tau(\nu_1, \nu_2) = [\nu_1] = [\nu_2]$. In addition, if a feature value ν_j is within the bounds of a feature range, we say that it falls within the feature range, denoted as $\nu_j \in \tau(\nu_1, \nu_2)$. For convenience of later description, two additional notations are given: $lower(\tau(\nu_1, \nu_2)) = \nu_1$, and $upper(\tau(\nu_1, \nu_2)) = \nu_2$.

Example 3.3 *Within the context of network intrusion detection in Table 1, for the nominal feature μ_1, [TCP] is one of its feature ranges. For the discrete feature μ_2, $[1, 5]$ is one of its feature ranges. For the continuous feature μ_3, $[0.09, 0.12]$ is one of its feature ranges. Furthermore, $2 \in [1, 5]$, $0.10 \in [0.09, 0.12]$, and $TCP \in [TCP]$.*

Similarly, the concept of NSA labels (i.e., *N: normal, S: suspicious and A: anomalous*) can be extended to the feature ranges as follows. For the feature range $\tau(\nu_1, \nu_2)$,

$$\begin{aligned}
\Lambda(\tau(\nu_1, \nu_2)) = 'N' &\Leftrightarrow \forall\nu(\nu \in \tau(\nu_1, \nu_2)) \wedge (\Lambda(\nu) = 'N') \\
\Lambda(\tau(\nu_1, \nu_2)) = 'A' &\Leftrightarrow \forall\nu(\nu \in \tau(\nu_1, \nu_2)) \wedge (\Lambda(\nu) = 'A') \\
\Lambda(\tau(\nu_1, \nu_2)) = 'S' &\Leftrightarrow \exists\nu\exists\nu'(\nu \in \tau(\nu_1, \nu_2) \wedge \Lambda(\nu) = 'A') \\
&\quad \wedge(\nu' \in \tau(\nu_1, \nu_2) \wedge \Lambda(\nu') = 'N')
\end{aligned}$$

With respect to an user-defined strategy, the feature space $\Omega(\mu)$ can be split into a set of feature ranges $\{\tau_\mu^1, \tau_\mu^2, \ldots, \tau_\mu^m\}$, such that the neighboring feature ranges (such as τ_μ^j and τ_μ^{j+1}) have different NSA labels, and there is no common feature value ν, which falls into τ_μ^j and τ_μ^k at the same time $(j \neq k)$. The feature ranges for the three features in Table 1 are illustrated in Figure 2, in which their NSA labels are indicated by their colors of feature values or feature ranges. Specifically, 'white' indicates 'normal', 'gray' for 'suspicious' and 'black' for 'anomalous'. Furthermore, the circles represent the feature values in the audit trails, and the squares represent the feature ranges outputted from the specified splitting strategy.

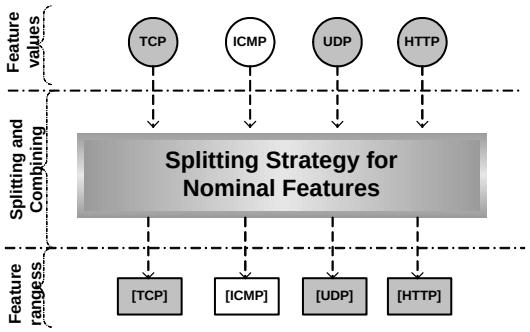
Finally, by grouping together the feature ranges with identical NSA labels, we can partition the feature space into three feature subspaces: *normal, suspicious and anomalous*. We will denote the normal feature subspace of a feature μ as $N(\mu)$, in which all its feature ranges

come from the normal audit trails. Similarly, the suspicious and anomalous feature subspaces of μ are denoted as $S(\mu)$ and $A(\mu)$ respectively. Thus we have,

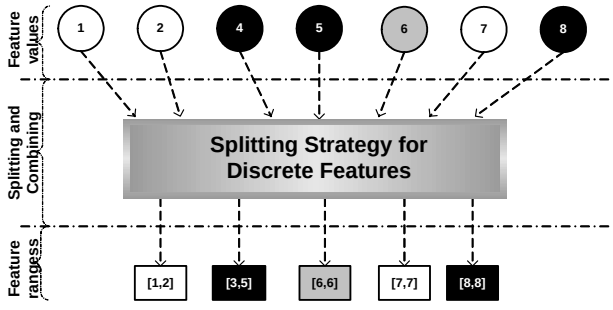
$$\begin{aligned} N(\mu) &= \{\tau_\mu^j |\ 1 \le j \le m,\ \Lambda(\tau_\mu^j) = \text{`}N'\} \\ S(\mu) &= \{\tau_\mu^j |\ 1 \le j \le m,\ \Lambda(\tau_\mu^j) = \text{`}S'\} \\ A(\mu) &= \{\tau_\mu^j |\ 1 \le j \le m,\ \Lambda(\tau_\mu^j) = \text{`}A'\} \end{aligned}$$

Where, obviously, $\Omega(\mu) = N(\mu) \cup S(\mu) \cup A(\mu)$.

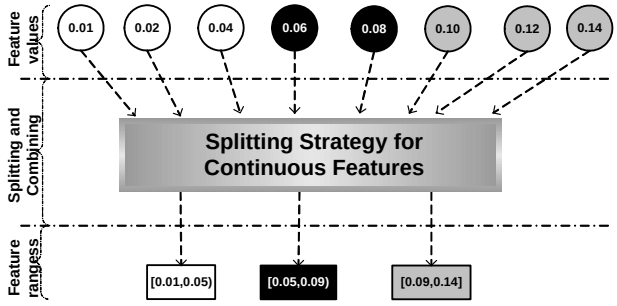
Example 3.4 *As indicated in Figure 2, the following feature ranges can be deduced from the example instances in Table 1.*



(a) Feature ranges of nominal features



(b) Feature ranges of discrete features.



(c) Feature ranges of continuous features

Figure 2: Feature ranges

- *For* μ_1,
  $\tau_{\mu_1}^1 = [TCP], N\&I_1 \Rightarrow \Lambda(\tau_{\mu_1}^1) = \text{`}S'$;
  $\tau_{\mu_1}^2 = [ICMP], N \Rightarrow \Lambda(\tau_{\mu_1}^2) = \text{`}N'$;
  $\tau_{\mu_1}^3 = [UDP], N\&I_2\&I_4 \Rightarrow \Lambda(\tau_{\mu_1}^3) = \text{`}S'$;
  $\tau_{\mu_1}^4 = [HTTP], N\&I_3 \Rightarrow \Lambda(\tau_{\mu_1}^4) = \text{`}S'$;
- *For* μ_2,
  $\tau_{\mu_2}^1 = [1,2], N \Rightarrow \Lambda(\tau_{\mu_2}^1) = \text{`}N'$;
  $\tau_{\mu_2}^2 = [3,5], I_1\&I_2 \Rightarrow \Lambda(\tau_{\mu_2}^2) = \text{`}A'$;
  $\tau_{\mu_2}^3 = [6,6], N\&I_3 \Rightarrow \Lambda(\tau_{\mu_2}^3) = \text{`}S'$;
  $\tau_{\mu_2}^4 = [7,7], N \Rightarrow \Lambda(\tau_{\mu_2}^4) = \text{`}N'$;
  $\tau_{\mu_2}^5 = [8,8], I_2\&I_4 \Rightarrow \Lambda(\tau_{\mu_2}^5) = \text{`}A'$;
- *For* μ_3,
  $\tau_{\mu_3}^1 = [0.01, 0.05), N \Rightarrow \Lambda(\tau_{\mu_3}^1) = \text{`}N'$;
  $\tau_{\mu_3}^2 = [0.05, 0.09), I_1\&I_2 \Rightarrow \Lambda(\tau_{\mu_3}^2) = \text{`}A'$;
  $\tau_{\mu_3}^3 = [0.09, 0.14], N\&I_3\&I_4 \Rightarrow \Lambda(\tau_{\mu_3}^3) = \text{`}S'$;

3.1.2 Compound Feature

The concept of NSA labeling and subspace partitioning can easily be extended to more than one feature. For convenience, we will refer to a single feature as an *atomic* feature and a combination of multiple features as a *compound* feature. We formally define a compound feature as follows:

Definition 3.2 (compound feature) *For any two features μ_1 and μ_2 in a specified feature vector, the feature space $\Omega(\mu_{12})$ of the compound feature μ_{12} is defined as a subset of the cartesian product of $\Omega(\mu_1)$ and $\Omega(\mu_2)$, such that each element in this set actually occurs in the audit trails of the relevant resource. In other words, if the ordered pair $(a, b) \in \Omega(\mu_{12})$, then any compound feature value (a, b) must have occurred in some audit trails of the corresponding resource.*

$$\begin{aligned} \Omega(\mu_{12}) = \{(a,b) | a \in \Omega(\mu_1), b \in \Omega(\mu_2), \\ (a,b) \textit{ is in audit trails}\} \end{aligned}$$

Intuitively, similar to the atomic features, the feature ranges of the new compound feature μ_{12} can also be labeled as normal, suspicious and anomalous ones, i.e., they have the NSA labels. Furthermore, just like the atomic feature space, a compound feature space can be partitioned into three feature subspaces i.e., $\Omega(\mu_{12}) = N(\mu_{12}) \cup S(\mu_{12}) \cup A(\mu_{12})$. To determine the membership of a given feature range $\tau_{\mu_{12}}^i$ to one of the three subspaces, the following intuitive rules are applied:

$$\begin{aligned} \tau_{\mu_{12}}^i \in N(\mu_{12}) &\Leftrightarrow \forall (a,b) \in \tau_{\mu_{12}}^i (a \in N(\mu_1) \vee b \in N(\mu_2) \\ &\quad \vee (a \in S(\mu_1) \wedge b \in S(\mu_2) \wedge \Lambda((a,b)) = \text{`}N')) \\ \tau_{\mu_{12}}^i \in A(\mu_{12}) &\Leftrightarrow \forall (a,b) \in \tau_{\mu_{12}}^i (a \in A(\mu_1) \vee b \in A(\mu_2) \\ &\quad \vee (a \in S(\mu_1) \wedge b \in S(\mu_2) \wedge \Lambda((a,b)) = \text{`}A')) \\ \tau_{\mu_{12}}^i \in S(\mu_{12}) &\Leftrightarrow \forall (a,b) \in \tau_{\mu_{12}}^i (a \in S(\mu_1) \wedge b \in S(\mu_2) \\ &\quad \wedge \Lambda((a,b)) = \text{`}S')) \end{aligned}$$

It is worth noting that the suspicious feature ranges in $\Omega(\mu_{12})$ can potentially shrink with respect to the original feature ranges in the component features as the combinations of the 'suspicious' feature ranges may be 'normal' or 'anomalous'.

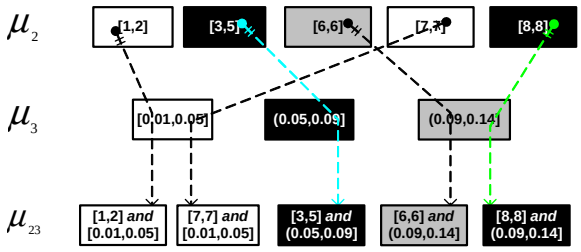
Since the compound feature built from two atomic features shows the same property as any atomic feature in the feature vector, it can be treated like an atomic feature to build higher order compound features. Using this recursive procedure, the feature vector FV for intrusion detection can be converted into an equivalent n-order

compound feature $\mu_{1...n}$ with normal $N(\mu_{1...n})$, suspicious $S(\mu_{1...n})$ and anomalous subspaces $A(\mu_{1...n})$.

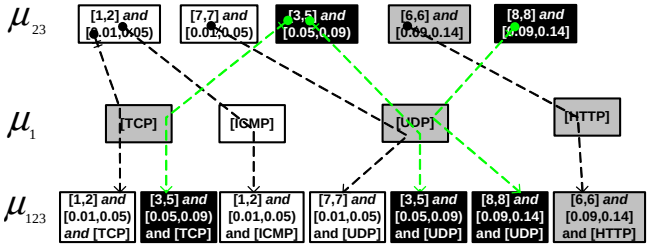
Example 3.5 *The features in Table 1 can be compounded as follows.*

- $\mu_{23} = \mu_2 \times \mu_3$
  $\tau^1_{\mu_{23}} = \tau^1_{\mu_2} \times \tau^1_{\mu_3}, N \Rightarrow \Lambda(\tau^1_{\mu_{23}}) = `N';$
  $\tau^2_{\mu_{23}} = \tau^2_{\mu_2} \times \tau^2_{\mu_3}, I_1 \& I_2 \Rightarrow \Lambda(\tau^2_{\mu_{23}}) = `A';$
  $\tau^3_{\mu_{23}} = \tau^3_{\mu_2} \times \tau^3_{\mu_3}, N \& I_3 \Rightarrow \Lambda(\tau^3_{\mu_{23}}) = `S';$
  $\tau^4_{\mu_{23}} = \tau^4_{\mu_2} \times \tau^1_{\mu_3}, N \Rightarrow \Lambda(\tau^4_{\mu_{23}}) = `N';$
  $\tau^5_{\mu_{23}} = \tau^5_{\mu_2} \times \tau^3_{\mu_3}, I_2 \& I_4 \Rightarrow \Lambda(\tau^5_{\mu_{23}}) = `A';$
- $\mu_{123} = \mu_1 \times \mu_{23}$
  $\tau^1_{\mu_{123}} = \tau^1_{\mu_1} \times \tau^1_{\mu_{23}}, N \Rightarrow \Lambda(\tau^1_{\mu_{123}}) = `N';$
  $\tau^2_{\mu_{123}} = \tau^2_{\mu_1} \times \tau^1_{\mu_{23}}, N \Rightarrow \Lambda(\tau^2_{\mu_{123}}) = `N';$
  $\tau^3_{\mu_{123}} = \tau^1_{\mu_1} \times \tau^2_{\mu_{23}}, I_1 \Rightarrow \Lambda(\tau^3_{\mu_{123}}) = `A';$
  $\tau^4_{\mu_{123}} = \tau^3_{\mu_1} \times \tau^2_{\mu_{23}}, I_2 \Rightarrow \Lambda(\tau^4_{\mu_{123}}) = `A';$
  $\tau^5_{\mu_{123}} = \tau^4_{\mu_1} \times \tau^3_{\mu_{23}}, N \& I_3 \Rightarrow \Lambda(\tau^5_{\mu_{123}}) = `S';$
  $\tau^6_{\mu_{123}} = \tau^3_{\mu_1} \times \tau^4_{\mu_{23}}, N \Rightarrow \Lambda(\tau^6_{\mu_{123}}) = `N';$
  $\tau^7_{\mu_{123}} = \tau^3_{\mu_1} \times \tau^5_{\mu_{23}}, I_2 \& I_4 \Rightarrow \Lambda(\tau^7_{\mu_{123}}) = `A';$

At the same time, the compounding operations can be illustrated as in Figure 3.



(a) Compounding between two atomic features



(b) Compounding between a compound feature and an atomic feature

Figure 3: compounding operations between feature ranges

Finally, the feature subspaces determined by the instances in Table 1 are:

$$\begin{aligned} N(\mu_{123}) &= \{\tau^1_{\mu_{123}}, \tau^2_{\mu_{123}}, \tau^6_{\mu_{123}}\} \\ S(\mu_{123}) &= \{\tau^5_{\mu_{123}}\} \\ A(\mu_{123}) &= \{\tau^3_{\mu_{123}}, \tau^4_{\mu_{123}}, \tau^7_{\mu_{123}}\} \end{aligned}$$

As a result, the formal foundations of intrusion detection can be built on a n-order compound feature without regard to the size of the feature vector n. In our subsequent discussion about the performance of intrusion detection techniques, we will stick to a single (compound) feature.

3.1.3 Behavior Signature

To do intrusion detection, every instance of the feature vector will be evaluated whether it matches the existing behavior model. In our framework, the behavior model of the resource is constituted by behavior signatures, which are defined as follows.

Definition 3.3 (behavior signature) *Assuming that there exists a feature vector $FV = \{\mu_1, \mu_2, \ldots, \mu_n\}$, the feature ranges of every feature are determined beforehand. A behavior signature is a feature range of the compound feature $\mu_{1...n}$ with its NSA label. In other words, the behavior signature is the combination of feature ranges of all features in the feature vector labeled by its statuses in the existing audit trails.*

As indicated in the above definition, every behavior signature represents a state of the resource at a specified time or point. In a special case with $n = 1$, the behavior signature can be a feature value in the existing audit trails. For example, the TCP port '139' is ever determined as 'anomalous' due to the worm 'Nimda'. Most importantly, to label an instance in the detection phase, the behavior signature possesses the same NSA label as its corresponding feature range of $\mu_{1...n}$.

Example 3.6 *Take the same scenario in Table 1, one behavior signature of the feature vector is $([ICMP], [1, 2], [0.01, 0.05))$. Obviously, the total number of possible behavior signatures in this example is $4 \times 5 \times 3 = 60$.*

For brevity, the term '*behavior signature*' will be simplified as '*signature*' within the context of this paper.

4 Intrusion Detection via Signatures

All the signatures collected from the training audit trails constitute the behavior model for intrusion detection. To achieve it, an splitting strategy is designed as follows to build the feature ranges for every feature. For nominal features, the splitting strategy do nothing except building one feature range for every feature value. For every discrete/continuous feature, an initial feature range is built for every feature value. Two initial feature ranges are *neighboring* if there are no feature values between them in the audit trails.

Specific for every discrete feature, the unknown feature subrange between any two neighboring initial feature ranges is split and combined into these two initial feature ranges as follows. If the size of the unknown feature subrange is an odd number n, $(n-1)/2$ of it will combine into every side, and the left 1 is assigned to one side randomly. If the size is an even number n, $n/2$ of it will combine into every side. In contrast, specific for every continuous feature, the unknown feature subrange

between any two neighboring initial feature range will be split equally and combined into both sides.

In the following step, if two neighboring feature ranges have the same NSA label, they will be combined into a single feature range by expanding its range size but with the NSA label. This can economize the storage space for the ultimate behavior models.

4.1 Detecting Behaviors Using M of N Features in a Signature

In our evaluation methodology, an instance in the test audit trails will be detected as follows. Utilizing the feature ranges of every feature, a temporal signature will be formed for the instance. If it matches any signature in the behavior model with M among N features, the status list of the signature will be inserted into the status list of the temporal signature. Obviously, the status list of the temporal signature is empty initially. After comparing with all signatures in the behavior model, the detection results for the instance is aggregated into its status list.

According to the status list and the nature of the instance, the following average cost for every instance in the test audit trails is calculated to quantify the detection performance. For a normal behavior, it will be detected as an anomaly if the status list include other status(es) other than 'normal'. Otherwise, it is detected as 'normal'. For a intrusive behavior, it will be detected as the same intrusion if the status list is identical to the status of the behavior, and it will be detected as normal if the status list only include the 'normal' status. Otherwise, it will be detected an an 'anomaly'.

4.2 To Measure 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). With respect to the detection results, every instance in the test audit trails will be assigned a *cost* value as the detection performance of the behavior model to it [11]. Specifically, if the *normal* instance is detected as '*normal*', the cost is 0, otherwise, the cost is 3. Simultaneously, if the intrusive instance is identified as its original intrusion label, the cost is 0. If the intrusive instance is detected as an anomaly, the cost is 1. If the intrusive instance is detected as 'normal', the cost is 3.

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

- $\#_{(N,N)}(M)$: the number of *normal* instances detected as 'normal';
- $\#_{(N,A)}(M)$: the number of *normal* instances, but detected as 'anomalies';
- $\#_{(N,*)}(M)$: the number of *normal* instances in the test audit trails;
- $\#_{(I,I)}(M)$: the number of *intrusive* instances detected as their original intrusions;
- $\#_{(I,A)}(M)$: the number of *intrusive* instances detected as 'anomaly';
- $\#_{(I,N)}(M)$: the number of *intrusive* instances detected as 'normal';
- $\#_{(I,*)}(M)$: the number of *intrusive* instances in the test audit trails.

where, it is obvious,

$$\begin{aligned} \#_{(N,*)}(M) &= \#_{(N,N)}(M) + \#_{(N,A)}(M) \\ \#_{(I,*)}(M) &= \#_{(I,I)}(M) + \#_{(I,A)}(M) + \#_{(I,N)}(M) \\ T &= \#_{(N,*)}(M) + \#_{(I,*)}(M) \end{aligned}$$

In the detection results, actually, $\#_{(N,A)}(M)$ is the size of false positives, and $\#_{(I,N)}(M)$ is the size of false negatives.

With respect to specific M, the average cost of every instance in the test audit trails is defined as:

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

From above equation, with the increase of $cost(M)$, the detection performance with the parameter M becomes worse. Obviously, the average cost at $M = N$ is the baseline for the detection performance. If $cost(M) > cost(N)$, the efficiency for intrusion detection has been degraded by such M of N scheme. Otherwise, it is useful for intrusion detection. An efficient intrusion detection technique will cause smaller average cost for every instance.

5 Experiments

We have chosen a typical dataset for network intrusion detection from KDD CUP 1999 contest, in which every record is an instance of a specific feature vector collected from the audit trails. This is because the dataset meets the requirements of our formal framework: labeled audit trails and 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 detailed description of the datasets, please refer to '*http://www-cse.ucsd.edu/users/elkan/clresults.html*'.

5.1 Evaluating the M of N Scheme

In our experimental evaluations, $N = 41$ and the parameter M is variable from 41 to 30. The behavior model is first built from the training audit trails. Then, every instance in the test audit trails is detected with respect to specific M, and the detection performance is quantified by the average cost of every instance within the detection results.

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)	serror_rate, srv_serror_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_serror_rate, dst_host_srv_serror_rate, dst_host_rerror_rate, dst_host_srv_rerror_rate

5.1.1 Experimental Results

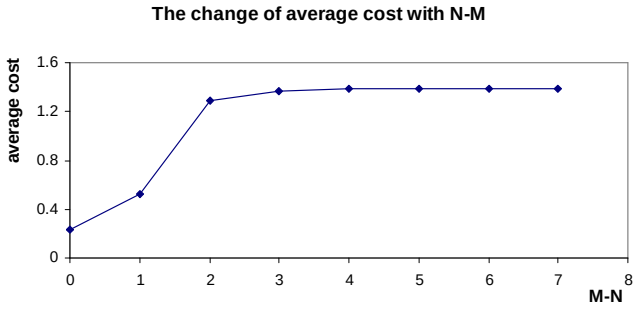


Figure 4: The influence of M of N features scheme

The detection performance baseline with $M = 41$ is $cost(41) = 0.228$. For the sake of comparison, we use $N - M$ as the horizontal axis in Figure 4, in which the influence of M of N features scheme on intrusion detection is illustrated. It is obvious that the average cost of every instance is decreased with the increase of $N - M$, i.e. with the decrease of M. In other words, even though the M of N features scheme can generalize the behavior model, it will degrade the detection performance for intrusion detection. Therefore, the scheme cannot enhance the detection performance in intrusion detection.

5.1.2 The Statistics about the Detection Results

In Table 3, the detection results are detailed with respect to varying M. In general, an efficient intrusion detection technique should identify most (normal and intrusive) behaviors, and the identification ability is indicated by the numbers in $\#_{(N,N)}(M)$ and $\#_{(I,I)}(M)$. However, in Table 3, these two numbers are decreased with the decrease of M. In other words, with the decrease of the parameter M, the identification capability is degraded, and most normal and intrusive behaviors will be detected as 'anomalies'.

As an extreme case, all the behaviors will be detected as 'anomalies'. From another perspective, with the increase of M, the behavior model becomes more random. This case will also occur if the behavior model is empty. In other words, the behavior model with more generalization caused by the M of N features scheme is almost no use for intrusion detection. In Table 3, when $M \leq 36$, almost all normal behaviors are detected as false positives (i.e., false alarms). This phenomenon will deteriorate the higher false alarm rate, which already problematic to make intrusion detection inefficient due to base-rate fallacy [2]. In summary, the M of N feature scheme will largely degrade the detection performance for intrusion detection.

6 Conclusions and Future Work

In this paper, we first present a formal intrusion detection framework based on a general framework. Using the framework, the M of N feature scheme is evaluated with respect to the detection performance for intrusion detection. To achieve it, we also propose a average cost function to quantify the detection performance for intrusion detection. The experimental results show that, even though the M of N scheme can generalize the behavior model to cover more unknown behaviors, it will degrade the detection performance for intrusion detection by triggering more false alarms. Ultimately, with the increase of M, the behavior model becomes so random that it is equal to an empty behavior model. More specifically, the identification ability of every signature will be lost with the decrease of M, i.e., with more generalization in the behavior model. The conclusion is critical for intrusion detection since all the features in a signature should be used to identify a (normal/intrusive) behavior, which does not follow our intuition.

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*, pp. 12–21, 2002.

Table 3: The detection results with respect to M

M	$\#_{(N,N)}(M)$	$\#_{(N,A)}(M)$	$\#_{(I,I)}(M)$	$\#_{(I,A)}(M)$	$\#_{(I,N)}(M)$	$cost(M)$
41	57102	3491	215835	21635	12966	0.228293825
40	53739	6854	134578	102487	13371	0.524587739
39	8067	52526	10225	238353	1858	1.290892489
38	2571	58022	39	249817	580	1.368435098
37	26	60567	3	250342	91	1.389953991
36	2	60591	0	250354	82	1.390137254
35	1	60592	0	250418	18	1.389735362
34	0	60593	0	250429	7	1.389674275
33	0	60593	0	250436	0	1.389629263
32	0	60593	0	250436	0	1.389629263
31	0	60593	0	250436	0	1.389629263
30	0	60593	0	250436	0	1.389629263

[2] S. Axelsson, "The base-rate fallacy and its implications for the difficulty of intrusion detection," in *Proceedings of the 6th ACM conference on Computer and communications security*, pp. 1–7, 1999.

[3] S. N. Chari and P. Cheng, "BlueBox: A policy-driven, host-based intrusion detection system," *ACM Transaction on Infomation and System Security*, vol. 6, no. 2, pp. 173–200, May 2003.

[4] D. Dasgupta and F. González, "An immunity-based technique to characterize intrusions in computer networks," *IEEE Transactions on Evol. Comput.*, vol. 6, no. 3, pp. 1081–1088, 2002.

[5] H. Debar, M. Dacier, and A. Wespi, "A revised taxonomy for intrusion detection systems," *Annales des Telecommunications*, vol. 55, no. 7–8, pp. 361–378, 2000.

[6] D. E. Denning, "An intrusion detection model," *IEEE Transaction on Software Engineering*, vol. SE-13, no. 2, pp. 222–232, Feb. 1987.

[7] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, and S. Stolfo, "A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data," D. Barbara and S. Jajodia (editors), in em Applications of Data Mining in Computer Security, Kluwer, 2002.

[8] S. A. Hofmeyr, S. Forrest, and A. Somayaji, "Intrusion detection using sequences of system calls," *Journal of Computer Security*, vol. 6, no. 3, pp. 151–180, 1998.

[9] H. Javits and A. Valdes, *The NIDES Statistical Component: Description and Justification*, SRI Anual Report A010, SRI International, Computer Science Laboratory, March 1993.

[10] W. Ju and Y. Vardi, "A hybrid high-order markov chain model for computer intrusion detection," *Journal of Computational and Graphical Statistics*, vol. 10, no. 2, pp. 277–295, 2001.

[11] 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.

[12] W. Lee and S. J. Stolfo, "A framework for contructing features and models for intrusion detection systems," *ACM Transactions on Information and System Security*, vol. 3, no. 4, pp. 227–261, Nov. 2000.

[13] Y. Liao and V. R. Vemuri, "Using text categorization techniques for intrusion detection," in *Usenix: Security 2002*, pp. 51–59, Aug. 2002.

[14] M. V. Mahoney and P. K. Chan, "Learning nonstationary models of normal network traffic for detecting novel attacks," in *SIGKDD 2002*, pp. 23–26, July 23-26 2002.

[15] C. Manikopoulos and S. Papavassiliou, "Network intrusion and fault detection: A statistical anomaly approach," *IEEE Communications Magazine*, vol. 40, no. 10, pp. 76–82, Oct. 2002.

[16] M. Ramadas, S. Ostermann, and B. Tjaden, "Detecting anomalous network traffic with self-organizing maps," in *Sixth International Symposium on Recent Advances in Intrusion Detection (RAID 2003)*, Pittsburgh, Pennsylvania, pp. 36–54, Sep. 8-10 2003.

[17] R. Sekar, M. Bendre, D. Dhurjati, and P. Bollineni, "A fast automaton-based method for detecting anomalous program behaviors," in *Proceedings of the 2000 IEEE Symposium on Security and Privacy*, pp. 144–155, 2000.

[18] K. Wang and S. J. Stolfo, "Anomalyous payload-based network intrusion detection," in *Proceedings of RAID*, pp. 203–222, 2004.

[19] Wikipedia, Def. of 'Intrusion-Detection System', http://en.wikipedia.org/w/wiki.phtml?title=Intrusion-detection_system.

Zhuowei Li is a Ph.D student in School of Computer Engineering, Nanyang Technological University, Singapore. He has gotten his M.Sc. degree from Institute of Computing Technology, Chinese Academy of Sciences, in 2001. For the time being, he is doing some research on Computer and Network Security (Intrusion Detection in particular), System Behavior Modeling, Model Evaluation, and the applications of Machine Learning techniques into Network Security domain.

Amitabha Das obtained his B.Tech. (Hons.) in Electronics and Electrical Communication Engineering from Indian Institute of Technology, Kharagpur, and his Ph.D. in Computer Engineering from University of California, Santa Barbara. He has been with the School of Computer Engineering in Nanyang Technological University, Singapore since 1992, where he is currently an associate professor. He is a senior member of the IEEE. His research interests include Wireless Networks, Network Security and Industrial Application of Machine Learning techniques.