

USAID: Unifying Signature-Based and Anomaly-Based Intrusion Detection

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

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

² Institute for Infocomm Research,
21 Heng Mui Keng Terrace, Singapore 119613

zhwei.li@pmail.ntu.edu.sg, asadas@ntu.edu.sg, jyzhou@i2r.a-star.edu.sg

Abstract. Most intrusion detection techniques suffer from either an inability to detect unknown intrusions, or unacceptably high false alarm rates. However, there lacks a general basis to analyze and find solutions to these problems. In this paper, we propose such a theoretical basis for intrusion detection, which makes it possible to systematically express and analyze the detection performance metrics such as the detection rate and false alarm rate in a quantified manner. Most importantly, the insights gained from the basis lead to the proposal for a new intrusion detection technique – USAID. USAID attempts to exploit the advantages of both techniques, and overcome their respective shortcomings. The experimental results show that USAID can achieve uniform level of efficiency to detect both known (99.78%) and new intrusions (98.18%), with a significantly reduced false alarm rate (1.45%). Most significantly, the performance of USAID is superior to all the participants in KDD’99 if the anomalies detected by USAID can be categorized correctly.

1 Introduction

In general, there exist two approaches for intrusion detection: signature-based (a.k.a. misuse detection) and anomaly-based intrusion detection. In principle, signature-based intrusion detection (SID) works reliably on well-known intrusions, but it is *incapable of detecting new intrusions*. In addition, it is also limited by several practical problems: *signature updating bottleneck*, *intrusion variation detection* (Rubin et al.[11]), and *too many false alarms* (Julisch[5]).

Consequently, anomaly-based intrusion detection (AID) has become the research focus as it is a useful alternative to SID, being capable of detecting novel intrusions. Besides that it can not identify intrusions, AID suffers from *higher false alarm rate*, the *difficulty in determining whether the anomalies are caused by intrusions* (Li et al.[8]), *concept drifting problem*, and *mimicry attacks* (Wagner et al.[12]). Most importantly, the *computational cost of intrusion detection must be reduced considerably* before it can be usefully employed in practical systems. This is more so as the size of the audit trails keep on growing.

In this paper, we try to find solutions to these problems. One possible solution may lie in the fact that only partial knowledge is used in each approach of intrusion detection. For example, SID only uses the knowledge about well-known intrusions (e.g., Snort[10]). For AID, it is the knowledge about the normal behaviors of the computing resources (e.g., LERAD[9], MADAM ID[7]). In our research, we try to use all the available knowledge about the behaviors (*intrusive and normal*) in the historical data to detect intrusions.

Our main contributions in this paper are two-fold. First, we propose a theoretical basis for intrusion detection. The basis then allows us to systematically analyze the detection performance to identify root causes for the problems in SID and AID. Secondly, the insights gained from the analysis naturally leads us to a new technique, named as USAID, which **Unifies SID and AID**.

The remaining parts are organized as follows. the nomenclature is introduced in section 2. Then, the existing intrusion detection approaches are analyzed systematically in section 3. Section 4 describes USAID. Experimental results showing the effectiveness of USAID and related work are presented in section 5 and section 6 respectively. Lastly, we conclude the paper and layout the future work.

2 Theoretical Basis for Intrusion Detection

In our theoretical basis for intrusion detection, we build behavior models like SID and AID, which use a *feature vector* $FV = \{F_1, F_2, \dots, F_m\}$, where F_i is a feature in the feature set. In general, a feature F_i can be categorized into *nominal*, *discrete* or *continuous* one. For example, ‘name’ is nominal, ‘TCP port number’ is discrete and ‘SYN rate within 2 seconds’ is continuous. A feature vector can contain any number of nominal, discrete, and/or continuous features. Besides that, we assume that there are *training audit trails*, which are constituted with labeled **normal** audit trails and **intrusive** audit trails.

2.1 Definitions

For any feature F , there is a meaningful domain $Dom(F)$ called the feature space. Any value occurring in the audit trails is a *feature value* v_F . With respect to the training audit tails, a feature value v_F will be labeled as *normal*, *suspicious* or *anomalous*. Specifically, if it only occurs in the normal audit trails, it is *normal*. If it only occurs in the anomalous audit trails, for example, in the intrusion signatures, it is *anomalous*. Otherwise, it is labeled as ‘*suspicious*’. We will refer to ‘normal’, ‘suspicious’, or ‘anomalous’ as the **NSA label** of the feature value v_F , denoted as $L(v_F) \in \{‘N’, ‘S’, ‘A’\}$ using the first letter of the label.

Feature Ranges. For any discrete/continuous feature F , the interval between v_F^1 and v_F^2 is defined as a *feature range* $R_F = [v_F^1, v_F^2]$. For the sake of uniformity, each nominal feature value is also referred to as a feature range. Thus, $R_F \subseteq Dom(F)$. The concept of **NSA labels** can be extended to the feature range:

$$\begin{aligned}
L(R_F) &= 'N' \Leftrightarrow \forall v_F (v_F \in R_F \wedge L(v_F) = 'N') \\
L(R_F) &= 'A' \Leftrightarrow \forall v_F (v_F \in R_F \wedge L(v_F) = 'A') \\
L(R_F) &= 'S' \Leftrightarrow \exists v_F^1 \exists v_F^2 (v_F^1 \in R_F \wedge v_F^2 \in R_F \wedge L(v_F^1) = 'N' \wedge L(v_F^2) = 'A')
\end{aligned}$$

Where, all the feature values occur in the training audit trails.

Next, for the feature F , we can collect a series of feature ranges from $Dom(F)$: $\{R_F^1, R_F^2, \dots, R_F^m\}$, such that $R_F^i \neq R_F^j$ and $L(R_F^i) \neq L(R_F^{j+1})$ ($1 \leq i, j \leq m$, and $i \neq j$). Furthermore, using the following rules, we can partition the feature space $Dom(F)$ into three feature subspaces: $N(F)$, $S(F)$ and $A(F)$.

$$\begin{aligned}
N(F) &= \{R_F^j \mid 1 \leq j \leq m, L(R_F^j) = 'N'\} \\
S(F) &= \{R_F^j \mid 1 \leq j \leq m, L(R_F^j) = 'S'\} \\
A(F) &= \{R_F^j \mid 1 \leq j \leq m, L(R_F^j) = 'A'\}
\end{aligned}$$

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

Definition 1 (compound feature). *A compound feature F_{12} is an ordered pair $\{F_1, F_2\}$, and $\Omega(F_{12})$ is a subset of the cartesian product of $\Omega(F_1)$ and $\Omega(F_2)$, such that each element in $\Omega(F_{12})$ actually represents at least one element in the audit trails. For the sake of expression, $F_{12} = F_1 \times F_2$.*

Intuitively, similar to its component features, a feature range of F_{12} has an NSA label with respect to its representative feature instance(s), and the compound feature space can also be partitioned into three feature subspaces, i.e., $\Omega(F_{12}) = N(F_{12}) \cup S(F_{12}) \cup A(F_{12})$. Note that the suspicious compound feature ranges can potentially shrink with respect to component feature ranges as the combinations of two 'suspicious' feature ranges may be 'normal' or 'anomalous'.

In summary, the compound feature built from two atomic features shows similar behaviours 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 $F_{1\dots n}$. In USAID, each compound feature range of $F_{1\dots n}$ is defined as a **behavior signature** in the behavior models for intrusion detection.

2.2 An Illustrative Example

Let us assume that $FV = \{F_1, F_2, F_3\}$. The example instances of the feature vector are listed below in Table 1. The feature ranges for every feature are listed as follows. For saving space, we will often use 'N' and 'I' respectively to denote the statuses 'normal' and 'intrusion'.

Table 1. The instances of the feature vector

INDEX	F_1	F_2	F_3	STATUS
1	TCP	1	0.01	normal
2	ICMP	2	0.04	normal
3	NETBIOS	6	0.10	intrusion ₃
4	TCP	4	0.08	intrusion ₁
5	UDP	5	0.06	intrusion ₂
6	UDP	8	0.14	intrusion ₄
7	NETBIOS	6	0.10	normal
8	UDP	7	0.02	normal
9	UDP	8	0.14	intrusion ₂

- For F_1 ,
 - $R_{F_1}^1 = TCP, N \& I_1 \Rightarrow 'S'$
 - $R_{F_1}^2 = ICMP, N \Rightarrow 'N'$
 - $R_{F_1}^3 = UDP, N \& I_2 \& I_4 \Rightarrow 'S'$
 - $R_{F_1}^4 = NETBIOS, N \& I_3 \Rightarrow 'S'$
- For F_2 ,
 - $R_{F_2}^1 = [1, 2], N \Rightarrow 'N'$;
 - $R_{F_2}^2 = [3, 5], I_1 \& I_2 \Rightarrow 'A'$;
 - $R_{F_2}^3 = [6, 6], N \& I_3 \Rightarrow 'S'$;
 - $R_{F_2}^4 = [7, 7], N \Rightarrow 'N'$;
 - $R_{F_2}^5 = [8, 8], I_2 \& I_4 \Rightarrow 'A'$;
- For F_3 ,
 - $R_{F_3}^1 = [0.01, 0.05], N \Rightarrow 'N'$;
 - $R_{F_3}^2 = (0.05, 0.09], I_1 \& I_2 \Rightarrow 'A'$;
 - $R_{F_3}^3 = (0.09, 0.14], N \& I_3 \& I_4 \Rightarrow 'S'$;
- $F_{23} = F_2 \times F_3$
 - $R_{F_{23}}^1 = R_{F_2}^1 \times R_{F_3}^1, N \Rightarrow 'N'$;
 - $R_{F_{23}}^2 = R_{F_2}^2 \times R_{F_3}^2, I_1 \& I_2 \Rightarrow 'A'$;
 - $R_{F_{23}}^3 = R_{F_2}^3 \times R_{F_3}^3, N \& I_3 \Rightarrow 'S'$;
 - $R_{F_{23}}^4 = R_{F_2}^4 \times R_{F_3}^1, N \Rightarrow 'N'$;
 - $R_{F_{23}}^5 = R_{F_2}^5 \times R_{F_3}^3, I_2 \& I_4 \Rightarrow 'A'$;
- $F_{123} = F_1 \times F_{23}$
 - $R_{F_{123}}^1 = R_{F_1}^1 \times R_{F_{23}}^1, N \Rightarrow 'N'$;
 - $R_{F_{123}}^2 = R_{F_1}^2 \times R_{F_{23}}^1, N \Rightarrow 'N'$;
 - $R_{F_{123}}^3 = R_{F_1}^1 \times R_{F_{23}}^2, I_1 \Rightarrow 'A'$;
 - $R_{F_{123}}^4 = R_{F_1}^3 \times R_{F_{23}}^2, I_2 \Rightarrow 'A'$;
 - $R_{F_{123}}^5 = R_{F_1}^4 \times R_{F_{23}}^3, N \& I_3 \Rightarrow 'S'$;
 - $R_{F_{123}}^6 = R_{F_1}^3 \times R_{F_{23}}^4, N \Rightarrow 'N'$;
 - $R_{F_{123}}^7 = R_{F_1}^3 \times R_{F_{23}}^5, I_2 \& I_4 \Rightarrow 'A'$;

The feature subspaces are: $N(F_{123}) = \{R_{F_{123}}^1, R_{F_{123}}^2, R_{F_{123}}^6\}$, $S(F_{123}) = \{R_{F_{123}}^5\}$, and $A(F_{123}) = \{R_{F_{123}}^3, R_{F_{123}}^4, R_{F_{123}}^7\}$.

3 Theoretical Analysis

In our analysis, we will focus on a compound feature F since the feature vector for intrusion detection can be compounded into a higher-order compound feature. In the ideal scenario, which assumes hypothetical complete knowledge, the three feature subspaces are determined without any misclassification: $N_i(F)$, $A_i(F)$ and $S_i(F)$. Similarly, in the real scenario where we possess only partial knowledge of them, three feature subspaces are $N_r(F)$, $A_r(F)$ and $S_r(F)$.

Due to the quality of the training audit trails (i.e., incompleteness and incorrect labels), there are two defects in the behavior model: (1) *model inaccuracy* and (2) *model incompleteness*. We use the following subsets of feature ranges to quantify the inaccuracy: $NA_1(F)$, $NS_1(F)$, $SN_1(F)$, $SA_1(F)$, $AN_1(F)$ and $AS_1(F)$. Every subset name is defined as: the first letter is the real feature subspace, the second letter represents the ideal one, and the subscript '1' indicates that it is caused by 'model inaccuracy'. As the behavior model is incomplete, there are some unknown feature ranges: $N_f(F)$, $A_f(F)$ and $S_f(F)$. The influence of incompleteness in the behavior model is quantified as: $NS_2(F)$ and $AS_2(F)$, where the subscript '2' indicates that it is caused by 'model inaccuracy'. Then,

$$\begin{aligned}
 N_r(F) - NS(F) + N_f(F) - NA(F) + AN(F) &= N_a(F) \\
 A_r(F) - AS(F) + A_f(F) - AN(F) + NA(F) &= A_a(F) \\
 S_r(F) + NS(F) + AS(F) + S_f(F) &= S_a(F)
 \end{aligned}$$

where, '+' and '-' denote set union and difference operations respectively.

3.1 Performance Analysis

In this subsection, we quantify and analyze the detection performance based on the detection results and the principles laid out so far. The detection performance is mainly represented by the detection rate and false alarm rate in the detection phase. In our discussion, we will assume that a fraction α of the feature ranges labeled as 'suspicious' will be detected as 'anomalous'.

Signature-Based Intrusion Detection. In SID, only $A_r(F)$ is known (a.k.a. the intrusion signature base). The behaviors which do not match $A_r(F)$ are regarded as 'normal' behaviors. Therefore, its detection performance is,

$$\begin{aligned}
 DR &= \frac{|A_r(F)| - |AS_1(F) + AS_2(F)| * (1 - \alpha) - |AN_1(F)|}{|A_i(F)| + |AS_1(F) + AS_2(F)| * \alpha - |NA_1(F)| + |S_f(F)| * \alpha} \\
 FAR &= \frac{|AS_1(F) + AS_2(F)| * (1 - \alpha) + |AN_1(F)|}{|A_r(F)|}
 \end{aligned}$$

Where, $|\dots|$ represents the size of a set of feature ranges.

Considering that $A_a(F)$ includes all the intrusions and their variations, the incapability to detect new intrusions as well as intrusion variations and the signature updating problem are due to the limited size and quality of $A_r(F)$. $AS_1(F)$, $AN_1(F)$ and $AS_2(F)$ lead to too many false alarms in practice.

Anomaly-Based Intrusion Detection. In AID, $N_r(F_i)$ is known beforehand in the normal run of a process, and the behaviors that violate $N_r(F_i)$ are regarded as 'anomalous', $S_r(F_i) = \Phi$. Therefore, its detection performance is,

$$\begin{aligned}
 DR &= \frac{|A_i(F)| + |S_f(F)| * \alpha}{|A_i(F)| + |S_f(F)| * \alpha + |NS_1(F) + NS_2(F)| * \alpha + |NA_1(F)|} \\
 FAR &= \frac{|N_f(F)| + |S_f(F)| * (1 - \alpha)}{|A_i(F)| + |N_f(F)| + |S_f(F)|}
 \end{aligned}$$

Obviously, the higher false alarm rate is largely rooted in $N_f(F)$ and $S_f(F)$. Mimicry attacks try to utilize $NS_1(F)$ and $NS_2(F)$. Concept drifting problem is to enlarge $NS_1(F)$, $NS_2(F)$ and $NA_1(F)$. Conversely, anomaly context identification tries to shrink the above sets as well as $S_f(F)$.

Conclusively, the quality issue in the behavior models leads to the problems in SID and AID. In our research, we try to utilize all available knowledge instead of the partial knowledge used in SID (i.e., $A_r(F)$) and AID (i.e., $N_r(F)$).

Unifying Signature-Based and Anomaly-Based Intrusion Detection. In USAID, all three *real* feature subspaces are known in advance: $N_r(F)$, $S_r(F)$

and $A_r(F)$. Except the known feature ranges in all three feature subspaces, other feature ranges are detected as ‘suspicious’ in USAID. As before, we can deduce the detection performance of USAID as follows.

$$DR = 1 - \frac{|NS_1(F) + NS_2(F)| * \alpha + |NA_1(F)| + |SA_1(F)| * (1 - \alpha)}{|A_i(F)| + |S_i(F)| * \alpha}$$

$$FAR = \frac{|N_f(F)| + |S_f(F) + AS_1(F) + AS_2(F)| * (1 - \alpha) + |SN_1(F)| * \alpha + |AN_1(F)|}{|N_f(F)| + |A_r(F)| + |A_f(F)| + |S_f(F)| + |S_r(F)| * \alpha}$$

It is clear that USAID will achieve similar detection rate as AID. Like AID, if the behavior model is accurate and complete, $DR = 100\%$ and $FAR = 0$. On the other hand, other than detecting anomalies, USAID can identify the intrusions in $A_r(F)$ as in SID. In summary, even though it is still limited by the quality issue of the behavior model, USAID provides advantages of both SID and AID.

4 A Novel Intrusion Detection Technique: USAID

In this section, we apply USAID for intrusion detection. The architecture of USAID consists of three modules as indicated in Figure 1. The first module extracts the three feature subspaces of every feature. Module 2 is to construct the signature base. The last module incorporates the detection mechanism.

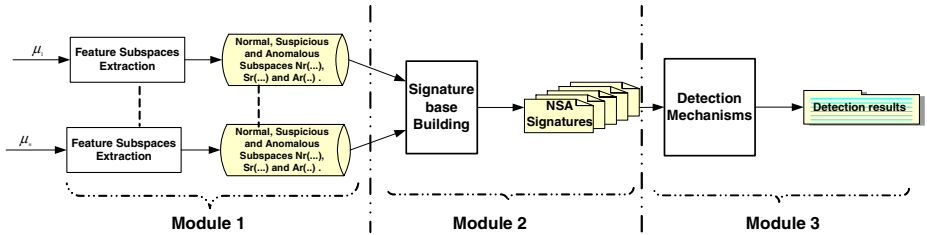


Fig. 1. A general framework for USAID

4.1 Feature Subspaces Extraction

Step 1: Feature Value Collection. In the labeled training audit trails, the feature values are collected for every feature. The statuses (*normal and/or intrusions*) of every feature value are also collected into its status list. Based on its status list, every feature value is assigned an NSA label. Note that this step is applied to nominal, discrete and continuous features, but the following two steps are only applicable to discrete and continuous features.

Step 2: Feature Value Clustering. The objective of this step is to form initial feature ranges for every feature by clustering the neighboring feature values. For a discrete feature, two feature values x_1 and x_2 are *neighboring* if

$|x_1 - x_2| = 1$ ¹. For a continuous feature, two feature values x_1 and x_2 are *neighboring* if $|x_1 - x_2| \leq \delta$. If several neighboring feature values have the same NSA label, they will be combined to form an *initial feature range*. As a special case, if, for a feature value, its neighbors have different NSA labels from itself, it forms an initial feature range itself. Every initial feature range thus formed inherits the NSA label of the feature values falling within it.

Step 3: Feature Range Generalization. Under most scenarios, the initial feature ranges will not cover all of the feature space. Any outside feature subspace is named as an *uncovered subspace*. Comparing to the neighboring definition of feature values, two feature ranges are *neighboring* if there is no other feature range(s) between them. Then, an uncovered subspace between two neighboring feature ranges is processed as follows: if the two feature ranges have the same NSA label, a new feature range will be formed to cover the two feature ranges as well as the uncovered subspace; otherwise, the uncovered space is divided equally and allocated to these two defined feature ranges. The NSA labels of the initial feature ranges will be inherited by the newly extended or combined feature ranges. Ultimately, all the *known* feature space of every feature will be covered by well-defined feature ranges.

4.2 Building Behavior Signatures

Initially, the NSA signature base (i.e., the behavior models) is empty. The following procedures are performed recursively on each feature instance. First, we construct a signature by replacing the feature values in the instance with respective feature ranges. Then, the signature is inserted into the NSA signature base with the status of the feature instance. Finally, based on the accumulated status list, every signature is assigned an NSA label.

4.3 Detection Mechanisms Via Signatures

We first extract a feature instance from the test audit trails at a time, and the feature value of every feature is replaced by its corresponding feature range to construct a temporary signature. We then try to search for the temporary signature in the NSA signature base, and if found, the current instance is assigned the same status list as that of the stored signature. Otherwise, the detection result is ‘anomaly’.

5 Experiments on USAID

We have chosen a typical dataset from KDD CUP 1999 contest, in which every record is an instance of a specific feature vector (Table 2). The dataset meets the requirements of USAID: *labeled audit trails* and *a feature vector*. The sizes

¹ If the distance is larger than 1, there is an uncovered space $|x_1 - x_2 - 1|$.

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

of the dataset are as follows: *training dataset: 4898431 instances, test dataset: 311029 instances*. For a detailed description of the datasets, please refer to [1]. In addition, as the precision for continuous features in our experimental datasets is 0.01, we set the neighboring threshold $\delta = 0.01$.

5.1 Experimental Results

First, let us talk about the dataset quality. In the training dataset, there are several illegal records (e.g. instance 4817100). In the test dataset, we found several instances (whose indices are 136489 and 136497) with illegal combination between ‘TCP’ protocol type and ‘ICMP’ service. Therefore, they are discarded in our experiments. Moreover, there is an intrusion ‘spy’ that is not documented properly (in instances 1381226, 1381227).

Feature Ranges of Every Feature. In our experiments, the average numbers of normal, suspicious and anomalous feature ranges of all features in the feature vector are $\bar{N} : \bar{S} : \bar{A} = 23 : 18 : 12$. Given the relatively large number of ‘suspicious’ feature ranges, it is clear that only one feature from our selected feature vector is not enough for intrusion detection.

The NSA Signature Base. In it, the numbers of the normal, suspicious and anomalous signatures are $N : S : A = 60371 : 58 : 2779$. Even though some suspicious signatures still exist, the detection capability has been improved much in comparison to any single feature. The existence of suspicious signatures indicates that the features in our experiments are not enough to detect all known intrusions. Note that the high ratio of N and A, namely $N : A = 60371 : 2779 \approx 21.7$ is quite significant since it indicates that the detection speed of SID will be faster. In addition, the total of possible signatures due to feature ranges of all 41 features are 8.38×10^{33} . In contrast, our signature base is compact enough. We further sense that searching future intrusion signatures via negative selection algorithm [4] from 8.38×10^{33} signatures is a mission impossible.

Signature Variations of a Behavior. At the same time, we observed that most intrusions cause more than one signature. For example, for portsweep, the number of signatures is 941; ipsweep, 72; satan, 389. The observation indicates that the intrusion variations do exist to a significant extent. In the NSA signature base, some signatures are shared by several intrusions, such as portsweep and nep-tune. Even though the number of shared signatures is small (Figure 2.(A)), this

	Normal	probe	DOS	U2R	R2L
normal	---	47	16	8	0
probe	47	33	10	2	0
DOS	16	10	0	0	0
U2R	8	2	0	0	0
R2L	0	0	0	0	0

(A) Statistics of Shared Signatures.

	0	1	2	3	4	A
0	57101	423	53	0	0	3015
1	2	2341	7	0	4	1818
2	5143	14	215142	0	0	11167
3	8	0	0	0	0	69
4	7817	2	0	0	0	6939

(B) Detection Results of USAID.

Fig. 2. Statistics of Shared Signatures and Detection Results

phenomenon shows that, under some scenarios (e.g., inadequate no. of features), it is difficult to identify some intrusions correctly. If the response strategies for the intrusions with overlapped signatures are much different, then generating responses to such intrusions may lead to disastrous results.

Shared Signatures among Behaviors. Also in Figure 2.(A), we enumerate the numbers of signatures shared between different intrusion categories. The normal category will share many signatures with other intrusion categories. This is the main source of false alarms or false negatives in intrusion detection. In this table, the signatures of R2L intrusions are not shared with other intrusion categories, whereas only ‘probe’ intrusions have shared signatures with each other category. One possible reason for this phenomenon lies in the proportions of signatures in every category, that is, $normal : probe : DOS : U2R : R2L = 60432 : 1661 : 1255 : 65 : 42$. In addition, the major principles of ‘probe’ intrusions are similar to each other [6], that’s why the shared ‘probe’ signatures are significant in Figure 2.(A). The strategies behind probe and DOS are similar, but it is different from the one behind U2R and R2L. Therefore, probe and DOS can be classified in one class, and U2R and R2L in another class [6]. The differences between these two classes explain why there are few signatures shared by them.

Detection Results from the Test Dataset. We eliminate two new intrusions, namely, snmpgetattack and mailbomb, from the detection results. This is because the information in the feature vector is not enough to detect these two intrusions.

Detection Performance. Quantitatively, the false alarm rate is 1.45%, the detection rate for known intrusions 99.78%, the detection rate for most new intrusions 98.18%. We also evaluate the USAID performance in comparison with the participants of KDD’99 Classifier Learning Contest[1], in which every entry will be assigned a detection cost, and an average cost per entry is calculated for comparison. The lower the average cost per entry is, the higher rank the classifier.

The detection results of USAID are summarized in Figure 2.(B), in which the first 5 columns constitute the confusion matrix, and the last column includes the numbers of detected anomalies. Its horizontal dimension is the predicted class of every test example, and the vertical dimension is its actual class. In the performance comparisons, the detected anomalies will be processed in two ways. First, these anomalies are classified correctly to their actual intrusion

categories. For example, the number of correctly predicated entries of ‘probe’ is $2341+1818=4159$. In such case, the performance of USAID is scored 0.1355, which is much better than the 1st rank of KDD’99, 0.2331. Secondly, under the worst scenario, these anomalies are classified incorrectly into the intrusion categories with highest cost. For instance, as $\text{cost}(\text{R2L,probe})=4$ is highest in row ‘R2L’, the anomalies detected from actual ‘R2L’ will be detected as category ‘normal’. The performance in the worst scenario is scored 0.3283, which is ranked 19th among all the participants. Note that almost half of the R2L intrusions, which are detected poorly in KDD’99, are detected as anomalies in USAID. In summary, USAID is expected to achieve better performance than all the participants of KDD’99 if the detected anomalies are categorized correctly.

6 Related Work

In USAID, two intrusion detection approaches are unified and their respective problems can be solved partially. The research work in [2] also shows the effectiveness of this combination, in which an algorithm (*similar to the negative selection algorithm in [4]*) is proposed to generate the artificial anomalies. An intrusion detection system is then built on the synthetic datasets. Actually, it only relies on the partial knowledge as well, and it lacks flexibility to fine-tune the model online. Other obvious advantages of USAID over [2] are the mechanisms for intrusion identification and anomaly context identification

Since the output of an intrusion detection technique can be considered to be a compound feature in our general feature vector, USAID is a multiple classifier ensembler [3]. In this aspect, USAID is similar to the research work in [3], in which one classifier is used to detect known intrusions, and another classifier tries to classify the new intrusions. However, [3] depends on the assumption that the first classifier can detect known intrusions accurately. That’s not true since there are significant intrusion variations as shown in our experiments.

7 Conclusions and Future Work

In this paper, we proposed a theoretical basis for intrusion detection, in which we unified signature-based and anomaly-based intrusion detection and systematically analyzed the hard problems faced by the researchers on intrusion detection. Our experimental results have also shown that the detection performance of USAID are encouraging. Specifically, most *new and known* intrusions are detected in USAID, and the false alarm rate is 1.451%. In our future work, we will continue research on our theoretical basis for intrusion detection.

References

1. C. Elkan. Results of the kdd'99 classifier learning contest. <http://www.cs.ucsd.edu/users/elkan/clresults.html>, 1999.
2. W. Fan, M. Miller, S. Stolfo, W. Lee, and P. Chan. Using artificial anomalies to detect unknown and known network intrusions. In *Proceedings of First IEEE International Conference on Data Mining (ICDM'01)*, pages 123–130, 2001.
3. W. Fan and S. Stolfo. Ensemble-based adaptive intrusion detection. In *Proceedings of SIAM International Conference on Data Mining 2002 (SDM2002)*, 2002.
4. S. Hofmeyr and S. Forrest. Architecture for an artificial immune system. *Evolutionary Computation*, 8(4):443–473, 2000.
5. K. Julisch. Clustering intrusion detection alarms to support root cause analysis. *ACM Transaction on Information and System Security*, 6(4):443–471, 2003.
6. K. Kendall. A database of computer attacks for the evaluation of intrusion detection systems. Master thesis, Massachusetts Institute of Technology, June 1999.
7. W. Lee and S. 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.
8. Z. Li and A. Das. Visualizing and identifying intrusion context from system calls trace. In *Proceedings of 20th Annual Computer Security Applications Conference*. IEEE Computer Society, Dec. 2004.
9. M. Mahoney and P. Chan. Learning Nonstationary Models of Normal Network Traffic for Detecting Novel Attacks. In *SIGKDD 2002*, July 23-26 2002.
10. M. Roesch. Snort - lightweight intrusion detection for networks. In *Proceedings of USENIX LISA*, 1999.
11. S. Rubin, S. Jha, and B. Miller. Automatic generation and analysis of nids attacks. In *Proceedings of 20th Annual Computer Security Applications Conference*, Tucson, AZ, USA, Dec. 2004. IEEE Computer Society.
12. D. Wagner and P. Soto. Mimicry attacks on host-based intrusion detection systems. In *Proceedings of the 9th ACM conference on Computer and communications security*, pages 255–264, 2002.