# Towards an Information-Theoretic Framework for Analyzing Intrusion Detection Systems

Guofei Gu[1], Prahlad Fogla[1], David Dagon[1], Wenke Lee[1] and Boris Skoric[2]

[1] Georgia Institute of Technology, U.S.A.
{guofei,prahlad,dagon,wenke}@cc.gatech.edu
[2] Philips Research Laboratories, Netherlands
boris.skoric@philips.com

**Abstract.** IDS research still needs to strengthen mathematical foundations and theoretic guidelines. In this paper, we build a formal framework, based on information theory, for analyzing and quantifying the effectiveness of an IDS. We firstly present a formal IDS model, then analyze it following an information-theoretic approach. Thus, we propose a set of information-theoretic metrics that can quantitatively measure the effectiveness of an IDS in terms of feature representation capability, classification information loss, and overall intrusion detection capability. We establish a link to relate these metrics, and prove a fundamental upper bound on the intrusion detection capability of an IDS. Our framework is a *practical* theory which is data trace driven and evaluation oriented in this area. In addition to grounding IDS research on a mathematical theory for formal study, this framework provides practical guidelines for IDS fine-tuning, evaluation and design, that is, the provided set of metrics greatly facilitates a static/dynamic fine-tuning of an IDS to achieve optimal operation and a fine-grained means to evaluate IDS performance and improve IDS design. We conduct experiments to demonstrate the utility of our framework in practice.

## 1 Introduction

As an essential component of the defense-in-depth strategy, intrusion detection systems (IDSs) have achieved more and more attention in both academic and industry. A number of IDSs have been developed in the last two decades [8]. Research work in the IDS field mainly focuses on how to construct a new detector based on some new idea so that the IDS can detect certain attacks with reasonable accuracy (in terms of false positives and false negatives). These are important topics, of course. However, very little work has been conducted on the fundamental theory. As a result, unlike cryptography, which now has a solid mathematical ground based on probability theory and the random oracle model, the IDS community still lacks a mathematical foundation that can be used to reason about the effectiveness of an IDS formally and practically. It is definitely necessary to base IDS research on a solid mathematical background [19] that can lead to a better understanding, evaluation, and design of an IDS. Such a theoretic framework should be mathematically sound, and useful in analyzing and quantifying the effectiveness of an IDS in both theory and practice.

In this paper, we investigate a novel theoretic framework for IDS research based on information theory. The basic observation is that, the intrusion detection process is actually a series of data processing and transformation procedures. This motivates us to use information theory, which is successfully applied in the area of communication (which is also a data/signal processing and transformation process), to study the efficiency of the intrusion detection process. We significantly extend our previous work on an information-theoretic measure of the intrusion detection capability [10], which only treats the IDS as a black box and measures the overall performance. In this paper, we further look into the basic components and architecture of an IDS, and apply information-theoretic analysis on the detailed intrusion detection procedure. Specifically, we make the following contributions in this paper:

1. We present **a formal model of an IDS** in Section 2 using an eight-tuple representation containing four data structures and four algorithms, which are used in three procedures, i.e., feature selection, profiling, and detection. This IDS model unifies signature-based and anomaly-based IDSs, thus, we can reason about all these IDSs using the same analytical approach. We also show how existing realistic IDSs such as PAYL [31] and Snort [26] fit into our model.

2. We perform **a fine-grained information-theoretic analysis on the IDS model** in Section 4. The detection procedure can be considered as a Markov transition chain in which two algorithms, i.e., a data reduction and representation algorithm $\mathcal{R}$ and a classification algorithm $\mathcal{C}$, are sequentially applied. This establishes a connection between intrusion detection and information theory. Further, we present **a series of information-theoretic metrics that can quantitatively measure the effectiveness of an IDS and its components**. We define the measures of feature representation capability ($C_R$), classification information loss ($L_C$), as well as the overall intrusion detection capability ($C_{ID}$, [10]). We establish a link among these metrics, and prove a fundamental upper bound of $C_{ID}$. The task of the IDS is to faithfully reflect the ground truth about intrusion information in observed data. If we assume the original ground truth information is 1 (normalized), when the data reduction and representation algorithm $\mathcal{R}$ is applied, this information is reduced to $C_R$. After the classification algorithm $\mathcal{C}$ is performed, there is further $L_C$ amount of information loss. The end result is $C_{ID}$, the overall capability of the IDS. We also discuss how the metrics can be used in a robust way to tolerate uncertainties and possible estimation errors of parameters in practice.

3. This framework provides **practical guidelines for fine-tuning, evaluation and design of IDSs**. With the help of $C_{ID}$, one can select the optimal operating point (where $C_{ID}$ is maximized) for an IDS, and we provide a concrete example for dynamically fine-tuning PAYL [31]. With the whole set of metrics, we provide a fine-grained analysis and quantification on the effectiveness of an IDS and its components. This yields a guideline for IDS design improvement, in particular, whether and how the feature representation or classification algorithm is (the bottleneck) to be improved. Experiments are conducted to show the utility of our framework in Section 5.

Note that in this paper we are not dealing with other important IDS performance issues, such as resilience to stress [25] and ability to resist attacks directed at the IDS [24, 23]. These are different research topics beyond the scope of this paper. Also we are not trying to address cost related issues in IDS analysis because cost factor is subjective, but we are building an objective theoretic framework. Finally, we need to point out that although our technique/framework may be applicable to other domains (e.g., to analyze a general classifier), we focus on the intrusion detection (specifically network-based intrusion detection) field.

## 2 Modeling an Intrusion Detection System

In order to formally reason and analyze an IDS, we firstly present a formal model of the IDS. Briefly, an IDS is represented as an eight-tuple $(\mathbb{D}, \Sigma, \mathbb{F}, \mathbb{K}, \mathcal{S}, \mathcal{R}, \mathcal{P}, \mathcal{C})$, in which the first four items are data structures, and the last four are algorithms. Note that whenever we analyze and evaluate any IDS, we cannot talk about it without dealing with its data source. After all, our IDS model and framework are *data trace driven* and *evaluation oriented*.

$\mathbb{D}$: the data source that an IDS will examine and analyze. Essentially this is a stream of consecutive data units. Since each IDS has its own unit of analysis, e.g., packet level or flow level for a network-based IDS (NIDS), without loss of generality, we define $\mathbb{D} = (D_1, D_2, ...)$ where $D_i$ is an analysis unit of data for the target IDS and $D_i \in \{d_1, d_2, ...\}$, $d_j$ is the possible data unit. For example, an NIDS uses network traffic (packet stream), so the data source is a packet stream $\boldsymbol{P} = (P_1, P_2, ...)$. For a host-based IDS (HIDS) using system call sequence, the data source is a system call stream $\boldsymbol{C} = (C_1, C_2, ...)$. In this paper, we mainly take network data as our example, and packet as our data unit.

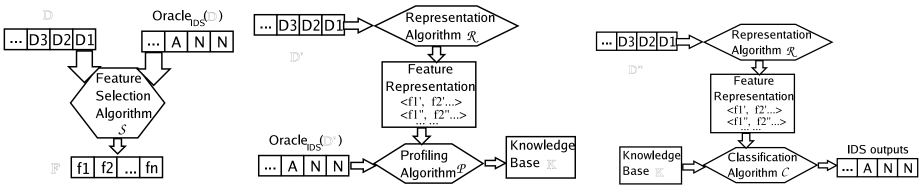
$\Sigma$: a finite set of data states indicating whether the data unit $D_i$ is normal or anomalous (or further what type of intrusion). For convenience, we define an IDS oracle $Oracle_{IDS}$ which accepts any query with data unit $D_i$, and outputs an indication whether the unit is normal or anomalous. The IDS oracle knows the ground truth so it will always tell the truth[3]. Then for every data unit $D_i$, its state is $Oracle_{IDS}(D_i)$. The space of this state set is finite. For anomaly detection, $\Sigma = \{Normal, Anomalous\}$, or simply $\Sigma = \{N, A\}$, or $\Sigma = \{0, 1\}$ where 0 denotes normal and 1 denotes anomalous. For misuse detection, we can let $\Sigma = \{Normal, AttackType_1, AttackType_2, ...\}$, or $\Sigma = \{N, A_1, A_2, ...\}$.

$\mathbb{F}$: a feature vector contains a finite number of features, formally $F =< f_1, f_2, ..., f_n >$. Every feature is a meaningful attribute of a data unit. For example, $f_1$ could be the protocol type (TCP, UDP, ICMP, etc.), $f_2$ could be the port number. Each feature has its own meaningful domain (called feature space) which is a set of discrete or continuous values (either numerical or nominal). The full range of $\mathbb{F}$ is the product of the ranges of all the features. We denote it as $Range(F) = f_1 \times f_2 ... \times f_n$.

$\mathbb{K}$: the knowledge base about the profiles of normal/anomalous data. This knowledge base consists of profiling model (stored in some data structures) of normal and/or attack information. The detailed structure of $\mathbb{K}$ is possibly different for every IDS. It could be a tree, a Markov model, a Petri net, a rule set, a signature base, etc. For a signature-based NIDS, $\mathbb{K}$ is its rule set which contains only the attack profiling model (i.e., intrusion signatures). For an anomaly NIDS, $\mathbb{K}$ is mainly the profile of the normal traffic. Any activity that deviates the normal profile is considered as anomaly.

[3] In *real evaluation* of any IDS, since we should always know the ground truth of data, we are acting as the IDS oracle in these cases.

$\mathcal{S}$: feature selection algorithm. Given some $\mathbb{D}$ and the corresponding states $Oracle_{IDS}(D)$ (note sometimes only partial or even no such state information available), this algorithm should return several features $f_i$ for the IDS to use. Although there is some preliminary effort to automatically generate worm signature [14, 22] for misuse IDSs as part of their features, generally speaking $\mathcal{S}$ still highly depends on domain knowledge and is normally conducted manually. The automatic selection or generation of features for both anomaly and misuse IDSs remains a grand challenge. The quality of features is one of the most important factors that will affect the effectiveness of an IDS.



D
Oracle_IDS(D)
... D3 D2 D1
... A N N
Feature Selection Algorithm S
F f1 f2 ... fn
... D3 D2 D1
Representation Algorithm R
D'
Feature Representation <f1', f2'...> <f1'', f2''...>
Oracle_IDS(D')
... A N N
Profiling Algorithm P
Knowledge Base K
... D3 D2 D1
Representation Algorithm R
D''
Feature Representation <f1', f2'...> <f1'', f2''...>
Knowledge Base K
Classification Algorithm C
IDS outputs
... A N N

(a) Feature selection procedure (b) Profiling/training procedure (c) Detection procedure

**Fig. 1.** Three IDS procedures

$\mathcal{R}$: data reduction and representation algorithm. When processing data, the IDS will firstly reduce the data and represent it in the feature space. This is a mapping/transition function, mapping the given data to a proper feature vector representation, namely $\mathcal{R} : \mathbb{D} \to \mathbb{F}$.

$\mathcal{P}$: profiling algorithm, which is the procedure of generating the profiling knowledge base $\mathbb{K}$. Given all the feature vector representations of data and their corresponding states, this algorithm will return the profiling knowledge base $\mathbb{K}$.

$\mathcal{C}$: classification algorithm. It is a mapping/transition function that maps the feature vector representation of given data to some states (it will also use the profiling base $\mathbb{K}$ in classification decision). Formally, $\mathcal{C} : \mathbb{F} \to \Sigma$.

Most IDSs work in three steps.

1. Feature selection procedure (Fig.1(a)). When we are developing an IDS, this is one of the first steps. Once the proper feature set is defined, it will be used in the following procedures. Normally, the feature selection procedure is conducted once, only during development.

2. Profiling procedure (Fig.1(b), sometimes also called training procedure[4]). We will run $\mathcal{P}$ (also involving $\mathcal{R}$) on a sufficiently large amount of training data and get the profiling knowledge base $\mathbb{K}$. Normally this procedure is performed once, only during development/training. In some situation, this procedure can be performed dynamically/periodically to update $\mathbb{K}$.

3. Detection procedure (Fig.1(c)). In this procedure, the IDS is used to detect intrusions in the data stream. This is the most important and frequently used procedure. We will perform an information-theoretic analysis on this procedure in Section 4.

[4] Some unsupervised learning based approach may skip this step.

Our IDS model unifies anomaly detection and misuse detection. In Appendix 9, we examine two representative IDSs, i.e., PAYL(Payload Anomaly Detection [31]) and Snort [26], to show how real world IDSs fit into our model.

## 3 Information Theory Background

Prior to introducing our information-theoretic framework for IDSs, we will first review a few basic concepts in information theory [6] to assist readers to follow our analysis.

***Entropy:*** The entropy (or self-information) $H(X)$ of a discrete random variable $X$ is defined as $H(X) = -\sum_x p(x)\log p(x)$. This definition, also known as Shannon entropy, measures the uncertainty of $X$. A smaller value of $H(X)$ indicates that $X$ is less uncertain (more regular). The definition of entropy can also be easily extended to the case of jointly distributed random variables.

***Conditional entropy:*** The conditional entropy $H(X|Y)$ is defined as $H(X|Y) = -\sum_y \sum_x p(x,y)\log p(x|y)$. It is the amount of information uncertainty of $X$ after $Y$ is seen. One can show that $H(X|Y)=0$ if and only if the value of $X$ is completely determined by the value of $Y$. Conversely, $H(X|Y)=H(X)$ if and only if $X$ and $Y$ are completely independent. The conditional entropy $H(X|Y)$ has the following property: $0 \le H(X|Y) \le H(X)$.

***Mutual information:*** Assume two random variables $X$ and $Y$ with a joint probability mass function $p(x,y)$ and marginal probability mass functions $p(x)$ and $p(y)$. The mutual information $I(X;Y)$ is defined as $I(X;Y) = \sum_x \sum_y p(x,y)\log \frac{p(x,y)}{p(x)p(y)}$. It is the amount of *reduction* of uncertainty in $X$ after $Y$ is known, $H(X|Y)$ being the *remaining* uncertainty. It tells us the amount of information shared between two random variables $X$ and $Y$. $I(X;Y)=0$ if and only if $X$ and $Y$ are independent. Obviously, $I(X;Y)=I(Y;X)$.

There is a nice relationship between entropy, conditional entropy and mutual information, i.e., $I(X;Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)$. That is, $I(X;Y)$ corresponds to the intersection of the information in $X$ with the information in $Y$. Clearly, $0 \le I(X;Y) \le H(X)$.

## 4 An Information-Theoretic Framework for Analyzing IDSs

The detection procedure (Fig.1(c)) of an IDS is the most important process for us to analyze. For simplicity, we will assume an anomaly NIDS with $\Sigma = \{N, A\}$ in all the following analysis (the analysis can be extended to an IDS with more than two states).

We firstly introduce three random variables $X^o, Z^o, Y$. $X^o$ represents all possible input data units to the IDS. It can take value in $\{d_1, d_2, ...\}$ with some probability. $\boldsymbol{X^o}$ is the data stream $\mathbb{D} = (D_1, D_2, ...)$. $Z^o$ (taking value in $Range(F)$ with some probability) is the intermediate representation of the data unit using the given feature set (performing $\mathcal{R}$). $\boldsymbol{Z^o}$ is the feature representation stream $(Z_1^o, Z_2^o, ...)$ where $Z_i^o = \mathcal{R}(D_i)$. $Y$ (taking value in $\Sigma$ with some probability) is the output alert of the IDS (the classification result of the IDS). $\boldsymbol{Y}$ is the alert stream $(Y_1, Y_2, ...)$ where $Y_i = \mathcal{C}(\mathcal{R}(D_i))$. Note that here we assume there is always an IDS output (decision) corresponding to each input. Although a real IDS only needs to output alerts when there is an intrusion, this does not affect our analysis.

Thus, the detection process is the Markov chain of $X^o \to Z^o \to Y$ (data→representation→alert) as shown in Fig.2(a), which we refer to as the original model. The input data are processed in sequence through two algorithms, $\mathcal{R}$ and $\mathcal{C}$. The mapping from $X^o$ to $Z^o$ is the result of $\mathcal{R}$. The mapping from $Z^o$ to $Y$ is the result of $\mathcal{C}$.

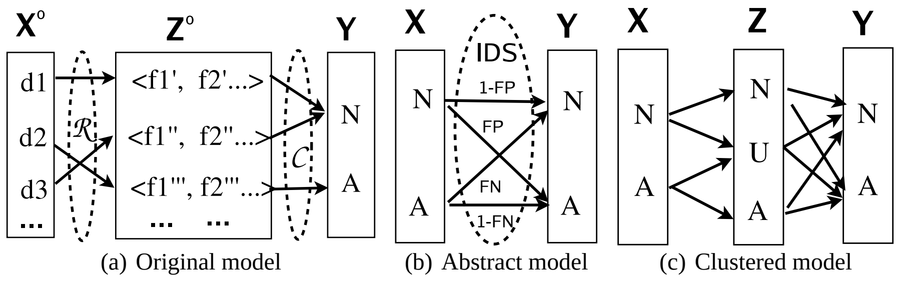


**Fig. 2.** Intrusion Detection Procedure: An Information-Theoretic View

The simple observation of this Markov chain data processing procedure motivates us to use information theory to analyze the process. Intuitively, we can roughly consider $\mathcal{R}$ as an encoding algorithm that uses feature vector to encode the original data unit. And then, $\mathcal{C}$, as a decoding algorithm, decodes the feature representation to an output of the IDS. We should point out that although $\mathcal{R}$ and $\mathcal{C}$ resemble encoding and decoding procedures, they are not exactly the strict encoding and decoding schemes. In information theory, either encoding or decoding needs an encoding/decoding table containing all possible codewords for all possible source codes, so it can ensure a perfect encoding and decoding (without error or ambiguity). In the case of intrusion detection, we cannot enumerate all the possible input data units (source codes) and feature representations (code words), nor can we afford to store such a huge encoding/decoding table. As a result, both $\mathcal{R}$ and $\mathcal{C}$ algorithms can only work roughly correct, i.e., these algorithms may not guarantee errorless information transmission. We can analyze and quantify the effectiveness of this information transmission using information-theoretic metrics.

It is still a little hard to practically measure the effectiveness of the intrusion detection process based on the original model in Fig.2(a), because this model involves too many states in $X^o$ and $Z^o$. We can hardly enumerate all the states and practically measure the transition probabilities. However, we notice that the purpose of an IDS is *not* to identify the original input data unit, but to identify the *state* of the data unit. That is, we are interested in only limited states of the data, i.e., $\Sigma$. We can group the input data to their states. This greatly simplifies the original model and our practical analysis. Similar idea can also be applied to the feature representation. Thus, we will introduce two simplified models step by step in the next paragraphs.

### 4.1 Abstract Model Analysis

First, we introduce a new random variable $X$ to replace $X^o$ in our analysis. $X$ takes values in $\Sigma$, which represents the state of all possible input data unit to the IDS, with certain probabilities. $\boldsymbol{X}$ is the state stream $(X_1, X_2, ...)$ where $X_i = Oracle_{IDS}(D_i)$.

As the first step of our simplification, we ignore the intermediate feature representation process (that is, we ignore $Z^o$, and only consider $X, Y$). We treat an IDS as a black box and thus, introduce our first simplified model, i.e., the abstract model in Fig.2(b), as firstly shown in [10]. In this abstract model, $\Sigma = \{N, A\}$. We can denote transition probabilities between $X$ and $Y$ using false positive rate ($FP$, $P(Y = A|X = N)$, denoted as $\alpha$) and false negative rate ($FN$,$P(Y = N|X = A)$, denoted as $\beta$). Thus, we have an abstract model of intrusion detection with a very simple Markov transition matrix between $X$ and $Y$. The capability of an IDS to classify the input events correctly (i.e., faithfully reflect the "truth" about the input) can be measured using (normalized) mutual information, which captures the reduction of original uncertainty (intrusion or normal) given that the IDS alerts are known.

**Definition 1.** *Intrusion detection capability $C_{ID}$ is defined as the normalized mutual information between $X$ and $Y$ [10], i.e., $C_{ID} = \frac{I(X;Y)}{H(X)}$.*

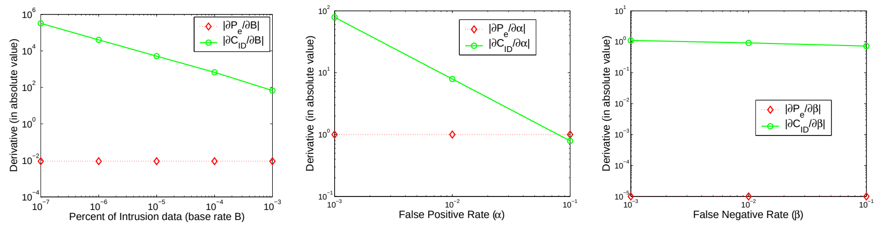
We can easily derive $C_{ID} = \frac{I(X;Y)}{H(X)} = \frac{H(X)-H(X|Y)}{H(X)} = 1 - \frac{H(X|Y)}{H(X)}$. Since $0 \leq H(X|Y) \leq H(X)$, we get $0 \leq C_{ID} \leq 1$.

Intuitively, $C_{ID}$ is interpreted as how much (normalized) ground truth information an IDS can identify. For example, $C_{ID} = 0.8$ means that the IDS identifies 0.8 bit of ground truth information assuming the original ground truth contains information 1. It indicates how well an IDS can distinguish normal from anomaly and distinguish anomaly from normal. In other words, it is an objective trade-off between $FP$ and $FN$.

$C_{ID}$ has several nice properties [10]: (1) it naturally takes into account all the important aspects of detection capability (if we expand the equation of $C_{ID}$), i.e., false positive rate, false negative rate, positive predictive value ($PPV$, or Bayes detection rate [3]), negative predictive value ($NPV$), and base rate (the probability of intrusion $P(X = A)$, denoted as $B$); (2) it objectively provides an *intrinsic* measure of intrusion detection capability; (3) $C_{ID}$ yields a series of related information-theoretic metrics, which will be discussed soon. This gives a fine-grained measure of the basic architecture and components of an IDS; (4) it is very sensitive to IDS operation parameters such as $\alpha, \beta$, which can demonstrate the effect of the subtle changes of an IDS.

[10] has showed that $C_{ID}$ is more sensitive than some existing metrics ($PPV, NPV$), however, comparison with the probability of error $P_e = Pr(Y \neq X)$ (which is another metric to define how $Y$ is different from $X$) is missing. Now we demonstrate that $C_{ID}$ is also more sensitive to operation parameters than $P_e$ in reasonable situations in which the base rate is very low [3]. For $\Sigma = \{N, A\}$, we can derive $P_e = B\beta + (1 - B)\alpha$. Similarly we can express the formula of $C_{ID}$ using $B, \alpha, \beta$. Since both $P_e$ and $C_{ID}$ have the same scale (value range [0,1]), it is fair to compare their sensitivities. To compare the sensitivities of $C_{ID}$ and $P_e$, we perform a differential analysis of $B, \alpha, \beta$ to study the effect of changing these parameters on $P_e$ and $C_{ID}$. For most IDSs and their operation environments, base rate and false positive rate are very low [3] so we can assume $B \ll 1$ and $\alpha \ll 1$.

Fig.3 shows the partial differential analysis (in absolute value) on different metrics. We only need to care about the absolute value of the derivatives. A larger derivative value shows more sensitivity to changes. For all the cases in Fig.3(a)(b)(c), a change in $B, \alpha$ or $\beta$ results in very tiny change in $P_e$. Only when $\alpha > 0.1$, the derivative of $P_e$ on $\alpha$ begins to be greater than that of $C_{ID}$. But for real world IDSs, it is very unlikely to have a false positive higher than 10%. (For example, it is quite reasonable to have more than one million packets per day in an enterprise network. If a packet level IDS has a false positive rate of 10%, this will generate more than 100,000 false positives per day!) Clearly, from Fig.3 we can see that $C_{ID}$ is more sensitive to changes in $B$, $\alpha$, $\beta$ than $P_e$ (several orders of magnitude more sensitive).



(a) Partial differential on $B$ ($\alpha = 0.001, \beta = 0.01$) (b) Partial differential on $\alpha$ ($B = 0.00001, \beta = 0.01$) (c) Partial differential on $\beta$ ($B = 0.00001, \alpha = 0.001$)

**Fig. 3.** Partial differential analysis (in absolute value). In every situation $C_{ID}$ has the highest sensitivity compared to $P_e$, except in (b) when $\alpha > 0.1$ (which is unlikely, in practice every IDS should have a much smaller false positive rate than 10%). For realistic situations, its derivative is always higher (several orders of magnitude) than $P_e$.

### 4.2 Clustered Model Analysis

In the previous abstract model, we have clustered numerous states in $X^o$ into a smaller set of states in $X$. As a next step, we reconsider the intermediate feature representation $Z^o$. Using similar simplification techniques, we will cluster numerous states in $Z^o$ into a smaller set. Specifically, we cluster the feature representation vectors to only three states, $\{N, U, A\}$. We can imagine that the IDS Oracle is labeling each feature representation vector $F_i$, denoted as $L(F_i) \in \{N, U, A\}$. State $N$ means the feature representation vector is from and only from the data unit which is normal. If the feature vector is from and only from data unit which is anomaly, then this is labeled as $A$. Those feature vectors that can be from both normal and anomalous data have the state $U$ (means *undistinguishable*). Formally,

$$\begin{aligned}
L(F_i) = N &\Leftrightarrow \forall D_j, \mathcal{R}(D_j) = F_i, Oracle_{IDS}(D_j) = N \\
L(F_i) = A &\Leftrightarrow \forall D_j, \mathcal{R}(D_j) = F_i, Oracle_{IDS}(D_j) = A \\
L(F_i) = U &\Leftrightarrow \exists D_1 \neq D_2, \mathcal{R}(D_1) = F_i, \mathcal{R}(D_2) = F_i, Oracle_{IDS}(D_1) = N, Oracle_{IDS}(D_2) = A
\end{aligned}$$

We use a new random variable $Z$ to replace $Z^o$. $Z$ denotes the clustered feature representation state, and $Z \in \{N, U, A\}$. Thus, we can slightly change the original transition model (Fig.2(a)) to a new one (Fig.2(c)).

In this clustered model, we can perform a fine-grained information-theoretic analysis on the intrusion detection procedure. Instead of viewing the IDS as a black box in the abstract model (Fig.2(b)) , we will analyze and reason about the basic architecture and components of an IDS. Here we have three random variables $X, Z, Y$, which form a Markov chain in the order $X \rightarrow Z \rightarrow Y$.

In this detailed model, we first consider the transition from $X$ to $Z$. Here we can define a metric which measures the capability of feature representation. The definition is also the normalized mutual information, similar to the definition of $C_{ID}$.

**Definition 2.** *We define the feature representation capability $C_R$ as the normalized mutual information between $X$ and $Z$, i.e., $C_R = \frac{I(X;Z)}{H(X)}$.*

Clearly $C_R$ is also a measure of the capability of $\mathcal{R}$. Similar to $C_{ID}$, $0 \leq C_R \leq 1$.

A larger $C_R$ means a better feature representation capability. If $C_R = 1$, then we say the IDS has an ideal feature representation capability. Intuitively this is saying that there is no information loss during the first transition from $X$ to $Z$.

If there are some feature vectors with state $U$, it is hard to distinguish whether they are from normal or anomalous data only given the feature vectors (note that for $\mathcal{C}$, the feature representation vector is the only input). Intuitively, when transition from $X$ to $Z$, we lose the "information". Information-theoretically, we will have a smaller $C_R$. Ideally, if the feature set has a perfect feature representation capability, we will have no feature vector with state $U$, which also means $P(Z = U|X = x) = 0$ for $\forall x \in \Sigma$. In this case, we get the identical distribution of $X$ and $Z$, so we get $C_R = 1$. Then the model is much simplified as well as the abstract model in Fig.2(b).

Now let us consider the transition from $Z$ to $Y$. In order to measure how good $\mathcal{C}$ is, we expect that there will be less information loss after the classification algorithm. Note here we do not simply use the normalized mutual information between $Z$ and $Y$ because actually in this transition, from $Z$ to $Y$, we still need to involve $X$, otherwise we cannot know how good $Y$ is (the classification result) according to $X$. Let us consider a new random variable $Y^X$ which is the joint probability distribution of $X$ and $Y$. Then the mutual information difference between $(Y^X, Z)$ (i.e., $I(X, Y; Z)$) and $(Y, Z)$ (i.e., $I(Y, Z)$) is the proper measure of classification information loss of $\mathcal{C}$. We will soon see this definition also yields another nice property stated in Theorem 1.

**Definition 3.** *We define the classification information loss $L_C$ as the normalized information loss between $I(X, Y; Z)$ and $I(Y; Z)$, i.e., $L_C = \frac{I(X,Y;Z)-I(Y;Z)}{H(X)}$.*

Because of the chain rule for information process, $I(X; Z|Y) = I(X, Y; Z) - I(Y; Z)$, we can also write $L_C$ as $L_C = \frac{I(X;Z|Y)}{H(X)}$.

Since $I(X; Z|Y) = H(X|Y) - H(X|Y, Z) \leq H(X|Y) \leq H(X)$, we know that $0 \leq L_C \leq 1$.

We always expect that $\mathcal{C}$ does a good job so as to have less information loss. Thus, a smaller $L_C$ means a better classification algorithm. If $L_C = 0$, then we say the IDS has an ideal classification algorithm $\mathcal{C}$ (so ideal classification information loss).

Now we have two new metrics $C_R$ and $L_C$ which can measure the feature representation capability and the classification information loss. The following theorem provides a nice relationship between these two metrics and $C_{ID}$.

**Theorem 1.** *The intrusion detection capability $C_{ID}$ is equal to the feature representation capability $C_R$ minus the classification information loss $L_C$, i.e., $C_{ID} = C_R - L_C$.*

*Proof.* Since $X, Z, Y$ form a Markov chain in the order $X \to Z \to Y$, the conditional distribution of Y depends only on Z and is conditionally independent on X. We can get $I(X;Y|Z) = 0$ in this case (because $X$ and $Y$ are conditionally independent given $Z$).

Using the chain rule, we can expand mutual information in two different ways.

$$\begin{aligned} I(X;Z,Y) &= I(X;Y) + I(X;Z|Y) \\ &= I(X;Z) + I(X;Y|Z) \end{aligned}$$

Applying the fact that $I(X;Y|Z) = 0$, we can get $I(X;Y) = I(X;Z) - I(X;Z|Y)$. Divided by $H(X)$, we get $C_{ID} = C_R - L_C$. $\square$

We already know $C_{ID}$ is the fraction of ground truth information identified by the IDS. If we assume the original ground truth information is 1 (normalized), when $\mathcal{R}$ is applied, this information will be reduced to $C_R$. After $\mathcal{C}$ is performed, we will further lose $L_C$ amount of information due to the classification algorithm. So finally we can get $C_{ID}$ amount of information. If both $C_R$ and $L_C$ are ideal, then the IDS has an ideal intrusion detection capability ($C_{ID} = 1$).

From this theorem, clearly we have $C_R \geq L_C$ because $C_{ID} \geq 0$. Also we can obtain the following corollary easily.

**Corollary 1.** *For an IDS, the intrusion detection capability is no more than its feature representation capability,i.e., $C_{ID} \leq C_R$.*

This establishes an upper bound of $C_{ID}$ for an IDS. For any given IDS, $C_{ID}$ can never exceed $C_R$. Once the feature set and $\mathcal{R}$ are given, the upper bound of $C_{ID}$ is also established no matter how good $\mathcal{C}$ is.

## 4.3 Implication and Discussion

**Implication for Fine-Tuning, Fine-Grained Evaluation and Improvement of IDSs** Now we can perform a fine-grained evaluation of IDSs using a set of information-theoretic metrics, $C_R, L_C$, as well as $C_{ID}$. We can compare different IDSs, not only in terms of the overall performance, but also the performance of their specific components.

The overall measure, $C_{ID}$, is surely very useful. We can fine-tune an IDS to some configuration that maximizes the $C_{ID}$ so that we have an optimal operation point. Section 5 will show a concrete example to demonstrate the static and dynamic fine-tuning of an IDS based on $C_{ID}$.

$C_R$ can help us evaluate whether the features in use have a good representation capability or not, independent of the classification algorithm. An ideal feature set should have no information loss during the process, i.e., there should be no “undistinguishable” feature representation vector. Once there exist some, they will definitely be classified to one output category although they are actually from two different input category (normal and anomaly). Here we lose the information, and the lost information will never come back or be complemented by further process (classification algorithm).

If we do find some undistinguishable state (conflicts), we need to further reconsider/reselect the features (refine $\mathbb{F}$). For example, we can *carefully* add more features so that the existing undistinguishable state will become distinguishable. (The original distinguishable states are still distinguishable). Thus, we can improve $C_R$ and avoid information loss in the first process ($X \rightarrow Z$). Note that only simply adding more features does not guarantee increasing accuracy (decreasing $L_C$) in the testing data, which is known as “overfitting” problem in machine learning literature, because the change of the feature set may also affect the accuracy of the classification algorithm. As a result, when adding more features, we increase the upper bound of $C_{ID}$ (i.e., $C_R$), but we still need to do some possible adjustment/change on classification algorithm to make sure that $L_C$ does not increase, so that we can improve the final $C_{ID}$.

In most cases when we compare two different IDSs, they can have different feature sets and different classification algorithms. With our framework, we can tell their fine-grained performance difference. For example, the reason why one IDS is less capable (lower $C_{ID}$) than another one can be mainly because its poor feature representation capability or classification information loss. Knowing the exact reason will point out future improvement direction (bottleneck) of IDS design. We have shown this using an example in experiment 4 of Section 5.

In practical evaluation, $C_{ID}$ and $C_R$ are easy to measure because the distribution, transition probabilities from $X$ to $Y$ in Fig.2(b) and the ones from $X$ to $Z$ in Fig.2(c) are easy to obtain in evaluation data. We may not need to directly calculate $L_C$, but simply apply Theorem 1 to compute $L_C = C_R - C_{ID}$.

**Implication for IDS Design**

***Feature*** $\mathbb{F}$ ***and algorithm*** $\mathcal{R}$ ***requirement:*** Feature selection is very important for any IDS. $C_R$ is the first quantitative measure of its representation capability. If features are not carefully selected, the information will be lost when $\mathcal{R}$ is applied. Once $C_R$ becomes lower, $C_{ID}$ will also decrease no matter how good $L_C$ is.

An IDS will not have a good representation capability if different types of data are represented in the same feature vector. It will misclassify some events because in the first transition process $(X \rightarrow Z)$, these different type of events cannot be distinguished from each other in terms of the feature vector representation (e.g., for Snort, some normal packets may match the same rule of some attack; for PAYL, the frequency vector of byte sequence for some attacks may be within the range of normal profile).

A lower feature representation capability $C_R$ normally implies two possible reasons, either features are not well selected or $\mathcal{R}$ is not well designed. So we are left with two possible ways to improve $C_R$. (1) Re-select the feature set or at least carefully add more features (this implies a better feature selection algorithm $\mathcal{S}$). For example, a context-aware Bro [27] is better than the one without considering context because it essentially adds new features (about the context). (2) Well implemented data reduction and representation algorithm $\mathcal{R}$ will also improve $C_R$ than poorly designed $\mathcal{R}$. For instance, in network intrusion detection, when using full assembling, protocol parsing $\mathcal{R}$, an IDS may achieve better $C_R$. Traffic normalization [11] is another good example of a better $\mathcal{R}$.

***Knowledge base*** $\mathbb{K}$ ***requirement:*** Since knowledge base $\mathbb{K}$ is used in the procedure of $\mathcal{C}$, an accurate (complete and general) $\mathbb{K}$ is an important factor to improve the performance of classification algorithm $\mathcal{C}$, so as to improve $L_C$. For a signature based IDS, it is important to make sure the signature set is accurate and covering as many as possible known attacks. This directly affects the quality of $\mathbb{K}$, and $\mathcal{C}$. For anomaly detection, an exact profiling of large amount of normal data is the key to improve the quality of $\mathbb{K}$ and $\mathcal{C}$.

***Realtime requirement on algorithms:*** The four algorithms in an IDS have different realtime requirement. $\mathcal{S}, \mathcal{P}$ are off-line algorithms, so there is fewer runtime speed requirement. However, for algorithm $\mathcal{R}$ and $\mathcal{C}$, they are mostly used online, so they should be efficiently implemented.

Finally, we should note that most of the implications are not surprising facts. They can be used as a sanity test for the correctness of any IDS model and theory. Our IDS model and information-theoretic framework nicely confirm them.

**Prior and Transition Probabilities**

***Static situation:*** When evaluating IDSs, we should always have the data set with detailed ground truth knowledge. Thus, from the evaluation data we can easily find out the base rate (fraction of intrusion) and measure all the transition probabilities ($\alpha, \beta$, etc.) in Fig.2 (b) and (c).

***Error bound and confidence:*** Machine learning researchers have given some bounds with certain confidence on the estimation of true error based on an observed error over a sample of data [21]. Given an observed sample error $e_s$, with approximately $N\%$ (e.g. 99%) probability, the true error $e_t$ will lie in the interval $e_s \pm z_N \sqrt{\frac{e_s(1-e_s)}{n}}$, where $n$ is the number of records in sample data, $z_N$ is a constant related to the confidence interval $N\%$ we want to reach. For example, if we want approximately 99% confidence intervals then $z_N = 2.58$. Since the possible difference between testing data and real data is a general problem for every data-centric evaluation research, we are not trying to solve this problem in this paper. In practice, we can assume the transition probabilities are relatively stable (such as $\alpha, \beta$) with reasonable high confidence, if the testing data is a representative sample of the real situation.

***Base rate estimation:*** In the real world, the base rate may vary in different situations. Here we give a heuristic approach to estimate the base rate. Once we have the estimated FP ($\alpha$), FN ($\beta$), we can approximately estimate the base rate in real traffic as follows. All we need is an alert rate ($r_a$) of the IDS (the fraction of generated alerts over total data). As we know this alert rate can be computed as $r_a = B(1-\beta)+(1-B)\alpha$. So we can approximately estimated the base rate as $B = \frac{r_a-\alpha}{1-\beta-\alpha}$, which provides us a good estimation of the real base rate. It is easy to prove that this is an unbiased estimator for $B$. In the next section, we will show how to use this estimation to dynamically fine-tune an IDS to keep it working on optimal operation points.

***Towards a robust consideration:*** Our framework can also be easily analyzed with a robust consideration. For a robust evaluation with uncertain parameters in real world, we consider the real $B, \alpha, \beta$ can deviate from our estimation to some certain degree (a range). Thus, we release the assumptions in all above sections. Now instead of calculating $C_{ID}, C_R, L_C$ with a static setting of $B, \alpha, \beta$, we use a range of these parameters (to tolerate largest possible estimation error bound), and among all possible results, we take the worst values (stands for the worst cases with all possible situation of

$B, \alpha, \beta$ as we expect) as the final resulting metric. By doing this, we are actually finding the best performing IDS against the worst situation (with the worst possible estimation error bound), instead of finding the best performing IDS on average (this is similar to the idea in [5]). Thus, we can make sure that this final measure (say, $C_{ID}$) is robust in sense that it is the low bound in all cases of possible estimated range of parameters. This will help if one is really concerned about the (large) estimation errors and uncertainties of the parameters in practice. The IDS is guaranteed to be better than this robust metric given the largest possible estimation error bound.

## 5 Experiments

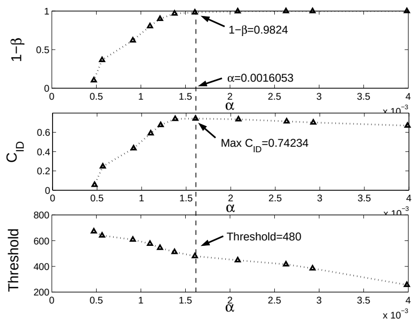
In this section, we describe the experiments we conducted to show how our information-theoretic framework is useful for fine-tuning an IDS in the real world, and we also show how a fine-grained measurement of IDSs is helpful for improving IDS design.

### 5.1 Dynamically Fine-Tuning an IDS

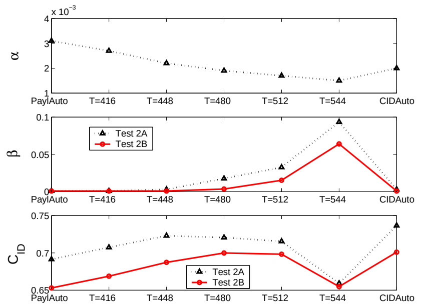
Fine-tuning an IDS is an important and non-trivial task, especially for anomaly-based IDSs. We can use $C_{ID}$ as a yardstick to find an operation point yielding the best trade-off between $FP$ and $FN$ (best is in terms of the intrinsic ability of the IDS to classify input data). Specifically, we firstly change the threshold of the anomaly IDS so that we can achieve different $FP$ and $FN$ pairs, and create an ROC curve. Then, we can calculate corresponding $C_{ID}$ for every point in the ROC. We select the point with the highest $C_{ID}$, and the threshold corresponding to this point provides the optimal threshold for use.

To demonstrate this, we select an anomaly IDS, PAYL [31], as our example. PAYL requires a threshold for determining whether the observed byte frequencies vary significantly from a trained model. For example, a threshold of 256 allows each character in an observed payload to vary within one standard deviation of the model. We collected a HTTP trace at a web server from our campus backbone network. Since PAYL only handles the HTTP requests from client to the server, we filter out all the outgoing HTTP responses. The trace data set only consists of incoming HTTP requests, approximately 7.5 million packets. We also filtered the trace set through to remove known attacks, and equally split the trace into three sets: training set, testing set 1, and testing set 2. We injected numerous HTTP attacks into the testing set, using tools such as Nikto [29].

In our first experiment, we train PAYL on the training set, and test it on testing set 1. The purpose is to choose an optimal threshold as the static operation point for PAYL in our testing environment. The base rate in testing set 1 is $B = 0.00081699$. The result is shown in Fig.4(a). We see that for the testing trace, as the threshold drops, $C_{ID}$ reaches a peak and then drops, while the ROC curve (shown in the top graph) continues to slowly increase. The maximum point of $C_{ID}$ corresponds to $<\alpha = 0.0016053, 1-\beta = 0.9824>$, and the corresponding threshold is 480. This tells us that PAYL works optimal in sense of intrusion detection capability at this threshold



(a) Experiment 1: Static tuning PAYL



(b) Experiment 2: Dynamic tuning PAYL using different schemes

**Fig. 4.** Fine-tuning PAYL in static and dynamic situations. In experiment 1, $C_{ID}$ can tell the optimal operation point, while a pure ROC curve cannot. In experiment 2, $C_{ID}Auto$ outperforms other schemes in terms of achieving optimal operation points dynamically in different base rate situations.

in our testing data. Without our information-theoretic guideline $C_{ID}$, it is not clear how to choose an optimal operation point from the ROC curve.

Experiment 1 finds optimal threshold in testing set 1. If the base rate in testing set 1 is representative to the real situation, then it is perfect. However, in real world situation, the base rate may vary from time to time. If we fix the operation point at a certain threshold, then in other testing data, we may not always achieve optimal $C_{ID}$ when the base rate varies. To address this problem, we introduce a new dynamic fine-tuning scheme that can be adaptive to any real situation. In previous section, we have discussed an unbiased estimation of base rate, i.e., $B = \frac{r_a - \alpha}{1-\beta-\alpha}$. If we divide the time series into many intervals, at each interval $n$, we estimate $B_n$, and then choose optimal operation point at this base rate to maximize $C_{ID}$. By dynamically fine-tuning the IDS, we ensure the IDS always operates on optimal points. Thus, we get a self-adaptive, self-tuning version of the IDS, which is very useful in practice.

We conduct a second experiment to investigate the effectiveness of dynamic fine-tuning. In experiment 2, we use the same training set in experiment 1. For the testing set, we inject different amount of attacks into testing set 2, and generate two new testing set $2A$ and $2B$. They contain the same normal data but different amount of attacks, so their base rates are different from testing set 1. Specifically, $B_{2A} = 0.00077256$, which is only slightly different from testing set 1, $B_{2B} = 0.00044488$, which is almost half of that in testing set 1. We modified PAYL to deploy a dynamic fine-tuning using $C_{ID}$ as the guideline. And we denote this scheme as $C_{ID}Auto$ scheme. We compare the results to the cases when we fix the threshold at some certain values from 416 to 544. We also compare with the original automatic threshold adjusting scheme provided by PAYL (denoted as $PaylAuto$). This scheme is to adjust the threshold in testing to control the alarm rate below certain value (0.001 in PAYL's setting). Once the alarm rate is stable low for some time, then the threshold is fixed during the rest of the testing.

The results of experiment 2 are shown in Fig.4(b). $C_{ID}Auto$ outperforms all other schemes in all cases, i.e., it outputs the highest $C_{ID}$ in both two testing sets $2A$ and $2B$. Fixing threshold at 480 as in experiment 1 still achieves satisfied result but not the optimal one because the base rate varies. Fixing threshold at 480 scheme gives the second highest $C_{ID}$ in test $2B$, less than $C_{ID}Auto$. Fixing threshold at 448 achieves the second highest $C_{ID}$ in test $2A$, still less than $C_{ID}Auto$. In both testing sets, the $PaylAuto$ scheme runs with a final stable threshold at 376, and this scheme has the highest $FP$ and lowest $C_{ID}$ (which is not good).

Our experiments clearly demonstrate the usefulness of $C_{ID}$ in dynamic fine-tuning of IDSs.

## 5.2 Fine-Grained Evaluation and Design Improvement of IDSs

In this section, we will show how our framework can be used for a fine-grained evaluation of IDSs and how we can improve the design. As a motivating example, we will use several machine learning based IDSs in this experiment, because they have a clear architecture and we can easily manipulate them as we want to change features or classification algorithms. Thus, it is much easier for readers to understand the usefulness of our framework.

In [17], Lee and Stolfo proposed three categories of features to be used by IDSs, i.e., 9 basic features of individual TCP connections, 13 content features within a connection suggested by domain knowledge, and 19 traffic features computed using a two-second time window. Using these total 41 features, they processed data set from 1998 DARPA Intrusion Detection Evaluation program [18]. The processed data are available known as KDD cup 1999 data set [1]. Every connection record is labeled as N (normal) or A (anomalous). The training set has 494,020 connection records. The testing data set has 311,029 records, among which 250,436 are anomalous. Obviously we can see that the distribution (of normal and anomalous data) in the testing data is not good because the base rate is so high (about 0.8) and obviously not a reasonable operation environment. *Only for the purpose of providing a reasonable base rate*, we artificially duplicate the normal connections 4000 times, so that the base rate $B = 250436/(4000 * 60593 + 250436) \approx 0.001$ is more reasonable. Note that our duplicating normal data does not affect other parameters such as $\alpha, \beta$.

We have noticed the critique [20] on the DARPA data set and the limit of the KDD data set. However, since our purpose is *not* to design a new IDS nor to conduct an official testing evaluation, we merely take them as a (public available) platform to demonstrate our framework. In this sense, we think the data are still valid to achieve our goal. We also plan to conduct more experiments using more real world IDSs on real world data to demonstrate our framework in the future.

In experiment 3, we use all 41 features (denoted as feature set 1). For the classification algorithm $\mathcal{C}$, we choose three different machine learning algorithms, i.e., decision tree (specifically, we use C4.5 [21]), Naive Bayes classifier, and SVM (Support Vector Machine [30]). All of them have been successfully applied to intrusion detection [2, 13]. Since they are standard machine learning classification algorithms that are well documented in [21, 30], we skip the details of these algorithms in this paper. The result of experiment 3 is shown in Table 1.

Form Table 1, we can see that in the transition $X \to Z$, these 41 feature set does *not* provide an ideal feature representation capability (i.e., $C_R = 0.9644 < 1$). Specifically, we measure the transition probabilities $P(Z = U|X = N) \approx 0.12$ and

**Table 1.** Experiment 3: a fine-grained evaluation

| IDS | $\alpha$ | $\beta$ | $C_{ID}$ | $C_R$ | $L_C$ |
|---|---|---|---|---|---|
| Feature set 1 with C4.5 | 0.017609 | 0.089676 | 0.4258 | 0.9644 | 0.5386 |
| Feature set 1 with Naive Bayes | 0.025713 | 0.099802 | 0.3756 | 0.9644 | 0.5888 |
| Feature set 1 with SVM | 0.0036473 | 0.12397 | 0.5642 | 0.9644 | 0.4002 |

$P(Z = U|X = A) \approx 0.030319$. When further analyzing the state $U$ in detail, we surprisingly find that all of the $U$ states are caused by `snmpgetattack` (one kind of R2L attack), i.e., 7,593 (out of total 7,741) `snmpgetattack` connection records have the same feature representations with some normal data. In other words, only from these feature representation, one cannot distinguish these `snmpgetattack` from normal traffic. So we already have information loss at the data reduction and representation process. For the classification process, SVM has the lowest classification information loss ($L_C = 0.4002$, much lower than other two algorithms). Thus, SVM finally outputs $C_{ID} = 0.5642$, which means on average, SVM can achieve slightly more than half of the original ground truth 'information'. The other two algorithms get less than half.

Experiment 3 clearly shows that the feature set in use still has room to improve because the feature representation capability is not ideal (a simple possible solution to improve $C_R$ is to add one more feature which can distinguish these `snmpgetattack` from normal traffic, e.g., SNMP protocol or not). We are the first to provide a quantitative measure of such capability. We also quantitatively compare the classification information loss of different machine learning algorithms such as SVM, decision tree and Naive Bayes. The result shows SVM has the least classification information loss in the data set.

In practice, more likely we will have IDSs with different feature sets and different classification algorithms. In these cases, we can first compare them using the overall intrusion detection capability ($C_{ID}$). Moreover, we can further (fine-grained) compare their feature representation capability and classification information loss. It will help us understand why an IDS is better, i.e., mainly due to its $C_R$ or $L_C$. This not only helps us evaluate IDSs (especially when the IDSs have similar overall $C_{ID}$), but also indicates the direction for further improvement and tuning of IDS design. To demonstrate this point, we conduct experiment 4. We choose two different feature sets and two different classification algorithms to form two IDSs: one uses feature set 2 (including 9 basic features and 13 content features) and C4.5 classification algorithm, the other uses feature set 3 (including 9 basic features and 19 traffic features) and Naive Bayes classifier. For feature set 2, we get the transition probabilities $P(Z = U|X = N) \approx 0.2021$, $P(Z = U|X = A) \approx 0.1892$ in testing set, and $C_R = 0.8092$. For feature set 3, we get the transition probabilities $P(Z = U|X = N) \approx 0.12$, $P(Z = U|X = A) \approx 0.030319$, and $C_R = 0.9644$.

The experiment result is shown in Table 2. We can see that the two IDSs have similar $C_{ID}$ (IDS1 is slightly better). But by further exploring the components of these IDSs, we find IDS1 has a much worse $C_R$ but a better $L_C$. On the contrary, IDS2 has a better $C_R$ but the classification algorithm is very poor (causing larger classification information loss). This fine-grained analysis indicates the bottleneck and

**Table 2.** Experiment 4. The fine-grained analysis indicates the improvement direction for each IDS.

| IDS | $\alpha$ | $\beta$ | $C_{ID}$ | $C_R$ | $L_C$ |
|---|---|---|---|---|---|
| IDS1(with feature set 2 and C4.5) | 0.023699 | 0.079437 | 0.4002 | 0.8092 | 0.4090 |
| IDS2(with feature set 3 and Naive Bayes) | 0.022577 | 0.10329 | 0.3875 | 0.9644 | 0.5769 |
| IDS1(after improving feature set) | 0.017609 | 0.089676 | 0.4258 | 0.9644 | 0.5386 |
| IDS2(after improving classification algorithm) | 0.017576 | 0.090374 | 0.4255 | 0.9644 | 0.5389 |

further improvement direction for IDS2 is mainly on classification algorithm, while for IDS1 is primarily a better feature set (since its $C_R$ is too low compared to that of IDS2). Following this direction, we do another experiment with indicated improvements. When IDS1 improves its feature set (classification algorithm unchanged) by simply adding more traffic features to become feature set $1$, we can get a better $C_{ID} = 0.4258$, which is higher than the original 0.4002. By improving IDS2's classification algorithm (use c4.5 to substitute Naive Bayes in our experiment, feature set unchanged), we can improve the $C_{ID}$ from 0.3875 to 0.4255.

The above example clearly demonstrates that fine-grained analysis can indicate our further (component) improvement direction for the design of IDSs.

## 6 Related Work

Intrusion detection is a field of active research for more than two decades. However, there is still little work on fundamental (theoretical) research, and there is still a huge gap between theory and practice.

For theoretical studies of intrusion detection, in 1987, Denning [9] was the first to systematically introduce an intrusion detection model, and also proposed several statistical models to build normal profiles. Later, Helman and Liepins [12] studied some statistical foundations of audit trail analysis for intrusion detection. Axelsson [4] pointed out that results from detection and estimation theory may be used in the IDS research. However, it is unclear how these similarities can benefit IDS evaluation and design. Song *et al.* [28] used ACL2 theorem prover for the analysis of IDSs that employ declarative rules for attack recognition by proving the specifications satisfy the policy with various assumptions. This approach is only useful for a certain type of IDSs, i.e., specification-based intrusion detection. In contrast, our framework is general to all types of IDSs. Recently, Di Crescenzo *et al.* [7] proposed a theory for IDSs based on both complexity-theoretic notions and well-known notions in cryptography (such as computational indistinguishability). Cardenas *et al.* [5] proposed a framework for IDS evaluation (not analysis) by viewing it as a multi-criteria optimization problem and gave two approaches: expected cost, and a new trade-off curve (IDOC) considering both the detection rate and the Bayes detection rate. Different from these existing work, our framework is an objective (without taking subject cost factors), natural and fine-grained approach with information-theoretic grounding. Besides, we established a clear and detailed IDS model, and provided an entire framework to analyze components inside an IDS and improve the design of IDSs.

Information-theoretic metrics have been widely applied in many fields. For instance, in the machine learning area, there are well-known algorithms (such as C4.5 [21]) that use information gain as a criterion to select features. [15] proposed to use entropy as a measure of distributions of packet features (IP addresses and ports) to identify and classify anomaly network traffic volumes. Lee *et al.* [16] applied information theoretic measurement to describe the characteristics of audit data set, suggested the appropriate anomaly detection model, and explained the performance of the models. This paper is a significant improvement and extension on our previous work [10], in which $C_{ID}$ was firstly proposed as a measure of the overall intrusion detection capability by viewing the whole IDS as a black box. An overall measure of the IDS is useful, but it cannot measure the performance of each component of the IDS. In this paper, we looked into the detailed processes within an IDS and performed a white box information-theoretic analysis on the components of the IDS. Thus, we built a complete framework. In addition, we demonstrated fine-tuning an IDS in both static and dynamic cases. We also showed how to use our framework to evaluate IDSs in a fine-grained way and improve the design of IDSs with experiments.

## 7 Conclusion and Future Work

In the paper, we established a formal framework for analyzing IDSs based on information theory. As a *practical* theory, it is data trace driven and evaluation oriented. Within our framework, the analysis of anomaly based and signature based IDSs can be unified. In addition to providing a better understanding of IDSs grounded on information theory, the framework also facilitates a static/dynamic fine-tuning of an IDS to achieve optimal operation, a better or finer-grained means to evaluate IDS performance and improve IDS design. Our framework provided intrusion detection research a solid mathematic basis and opened the door for the study of many open problems.

This paper is only a preliminary start in the field. There are many topics for possible future work. One is to use more information theory, e.g., channel capacity models, to further study the effect of multiple processes/layers/sensors of the IDS architecture. Thus, we can analyze and improve *both internal and external* designs of the IDSs by extending our current framework. We will also further study robust ways of applying the framework.

## 8 Acknowledgements

This work is supported in part by NSF grant CCR-0133629 and Army Research Office contract W911NF0510139. The contents of this work are solely the responsibility of the authors and do not necessarily represent the official views of NSF and the U.S. Army.

## References

1. Kdd cup 1999 data. Available at http://kdd.ics.uci.edu/databases/kddcup99/, 2006.
2. Nahla Ben Amor, Salem Benferhat, and Zied Elouedi. Naive bayes vs decision trees in intrusion detection systems. In *SAC '04*, 2004.

3. S. Axelsson. The base-rate fallacy and its implications for the difficulty of intrusion detection. In *Proceedings of ACM CCS'1999*, November 1999.
4. Stefan Axelsson. A preliminary attempt to apply detection and estimation theory to intrusion detection. Technical Report 00-4, Dept. of Computer Engineering, Chalmers Univerity of Technology, Sweden, March 2000.
5. Alvaro Cardenas, Karl Seamon, and John Baras. A Framework for the Evaluation of Intrusion Detection Systems. In *Proceedings of the 2006 IEEE Symposium on Security and Privacy*, Oakland, California, May 2006.
6. Thomas Cover and Joy Thomas. *Elements of Information Theory*. John Wiley, 1991.
7. Giovanni Di Crescenzo, Abhrajit Ghosh, and Rajesh Talpade. Towards a theory of intrusion detection. In *ESORICS'05*, 2005.
8. Herve' Debar, Marc Dacier, and Andreas Wespi. Towards a taxonomy of intrusion-detection systems. *Computer Networks*, 31(8):805–822, 1999.
9. D. Denning. An intrusion-detection model. *IEEE Transactions on Software Engineering*, 13(2), Feb 1987.
10. Guofei Gu, Prahlad Fogla, David Dagon, Wenke Lee, and Boris Skoric. Measuring intrusion detection capability: An information-theoretic approach. In *Proceedings of ACM Symposium on InformAction, Computer and Communications Security (ASIACCS'06)*, March 2006.
11. Mark Handley, Vern Paxson, and Christian Kreibich. Network intrusion detection: Evasion, traffic normalization, and end-to-end protocol semantics. In *Proc. USENIX Security Symposium 2001*, 2001.
12. P. Helman and G. Liepins. Statistical foundations of audit trail analysis for the detection of computer misuse. *IEEE Transactions on Software Engineering*, 19(9), September 1993.
13. Wenjie Hu, Yihua Liao, and V. Rao Vemuri. Robust support vector machines for anomaly detection in computer security. In *Proc. 2003 International Conference on Machine Learning and Applications (ICMLA'03)*, 2003.
14. Hyang-Ah Kim and Brad Karp. Autograph: Toward automated, distributed worm signature detection. In *USENIX Security Symposium*, pages 271–286, 2004.
15. Anukool Lakhina, Mark Crovella, and Christophe Diot. Mining anomalies using traffic feature distributions. In *SIGCOMM '05*, 2005.
16. W. Lee and D. Xiang. Information-theoretic measures for anomaly detection. In *Proceedings of the 2001 IEEE Symposium on Security and Privacy*, May 2001.
17. Wenke Lee and Salvatore J. Stolfo. A framework for constructing features and models for intrusion detection systems. *ACM Transactions on Information and System Security (TISSEC)*, 3(4):p.227–261, 2000.
18. Massachusetts Institute of Technology Lincoln Laboratory. 1998 darpa intrusion detection evaluation data set overview. http://www.ll.mit.edu/IST/ideval/, 2005.
19. Teresa F. Lunt. Panel:foundations for intrusion detection. In *Proc. 13th Computer Security Foundations Workshop (CSFW 2000)*, 2000.
20. John McHugh. Testing intrusion detection systems: A critique of the 1998 and 1999 darpa off-line intrusion detection system evaluation as performed by lincoln laboratory. *ACM Transactions on Information and System Security*, 3(4), November 2000.
21. Tom Mitchell. *Machine Learning*. McGraw-Hill, 1997.
22. James Newsome, Brad Karp, and Dawn Song. Polygraph: Automatically generating signatures for polymorphic worms. In *IEEE S&P '05*, 2005.
23. Vern Paxson. Bro: A system for detecting network intruders in real-time. *Computer Networks*, 31(23-24):2435–2463, December 1999.
24. T. H. Ptacek and T. N. Newsham. Insertion, evasion, and denial of service: Eluding network intrusion detection. Technical report, Secure Networks Inc., January 1998.

25. Nicholas J. Puketza, Kui Zhang, Mandy Chung, Biswanath Mukherjee, and Ronald A. Olsson. A methodology for testing intrusion detection systems. *IEEE Transactions on Software Engineering*, 22(10):719–729, 1996.
26. M. Roesch. Snort - lightweight intrusion detection for networks. In *Proceedings of USENIX LISA'99*, 1999.
27. Robin Sommer and Vern Paxson. Enhancing byte-level network intrusion detection signatures with context. In *CCS '03*, 2003.
28. Tao Song, Calvin Ko, Jim Alves-Foss, Cui Zhang, and Karl N. Levitt. Formal reasoning about intrusion detection systems. In *Proceedings of RAID'2004*, September 2004.
29. Sullo. Nikto, 2006. Available at http://www.cirt.net/code/nikto.shtml.
30. V.N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995.
31. Ke Wang and Salvatore J. Stolfo. Anomalous payload-based network intrusion detection. In *Proceedings of RAID'2004*, September 2004.

## 9 Appendix: How Real World IDSs Fit into Our Model

Table 3 shows how PAYL and Snort[5] fit into our model.

**Table 3.** Modeling PAYL and Snort

| | PAYL model |
|---|---|
| $\mathbb{D}$ | Packet sequence $(P_1, P_2, \ldots)$ |
| $\Sigma$ | $\{N, A\}$, only indicates normal or anomalous. |
| $\mathbb{F}$ | A character frequency vector $< fre_0, fre_1, \ldots, fre_{255} >$, here $fre_i$ is the frequency of char $i$ in the payload of the packet. |
| $\mathbb{K}$ | For each specific observed length $i$ of each port $j$, $M_{ij}$ stores the mean and standard deviation of the frequency for each distinct byte. |
| $\mathcal{S}$ | Manually examines exploits and finds out the importance of byte frequency. |
| $\mathcal{R}$ | Scans each incoming payload of the packet, computes its byte value distribution. |
| $\mathcal{P}$ | Runs $\mathcal{R}$ on large normal data set to generate normal profile ($\mathbb{K}$) of the frequency. |
| $\mathcal{C}$ | For each new payload distribution given by $\mathcal{R}$, compares against model $M_{ij}$. If their Mahalanobis distance significantly deviates from the normal threshold, flags the packet as anomalous and generates an alert. |

| | Snort model |
|---|---|
| $\mathbb{D}$ | Packet sequence $(P_1, P_2, \ldots)$ |
| $\Sigma$ | $\{N, A_1, A_2, \ldots\}$. $N$ is the normal state. $A_i$ is the type of attack that can be detected by Snort, e.g., `WEB-IIS .asp HTTP header buffer overflow attempt`. Currently Snort can detect over three thousand attacks. |
| $\mathbb{F}$ | Feature vector such as <`srcIP, dstIP, dstPort`, `payload containing '\|3A\|' or not`, `payload containing '\|00\|' or not`, ...>. Many of the features are boolean values to indicate whether the payload contains some substring (part of the intrusion signatures) or not. |
| $\mathbb{K}$ | The rule set of Snort. |
| $\mathcal{S}$ | Manually examines exploits and finds out most common strings in intrusions. |
| $\mathcal{R}$ | Grinders the packet, preprocesses (defragmentation, assembling, etc.). If possible, uses string matching to explore feature space according to the rule set. Due to the implementation of Snort, $\mathcal{R}$ does not need to represent the packet into the whole feature space. Instead, $\mathcal{R}$ can stop when the packet is represented to some subset of features (matching a certain rule). |
| $\mathcal{P}$ | Manually extracts signatures of intrusions and generates the rule set. |
| $\mathcal{C}$ | Although $\mathcal{C}$ is not clearly separated from $\mathcal{R}$ in the implementation of Snort, we can consider it as a simple exact matching in knowledge base $\mathbb{K}$ (as 1-nearest neighbor searching in the rule set for the exact rule). |

[5] There is no clear separation between $\mathcal{R}$ and $\mathcal{C}$ in the implementation of Snort. But we can still model the whole process into two algorithms. $\mathbb{K}$ is used in the whole process.