Toward Cost-Sensitive Modeling for Intrusion Detection and Response

Wenke Lee
College of Computing
Georgia Institute of Technology
801 Atlantic Drive Georgia, GA 30332-0280
wenke@cc.gatech.edu

Wei Fan
IBM T.J.Watson Research Center
Hawthorne, NY 10532
weifan@us.ibm.com

Matthew Miller and Salvatore J. Stolfo
Computer Science Department
Columbia University
1214 Amsterdam Avenue
New York, NY 10027
{mmiller, sal}@cs.columbia.edu

Erez Zadok
Computer Science Department
State University of New York at Stony Brook
Stony Brook, NY 11794-4400
ezk@cs.sunysb.edu

Abstract

Intrusion detection systems (IDSs) must maximize the realization of security goals while minimizing costs. In this paper, we study the problem of building cost-sensitive intrusion detection models. We examine the major cost factors associated with an IDS, which include development cost, operational cost, damage cost due to successful intrusions, and the cost of manual and automated response to intrusions. These cost factors can be qualified according to a defined attack taxonomy and site-specific security policies and priorities. We define cost models to formulate the total expected cost of an IDS, and present cost-sensitive machine learning techniques that can produce detection models that are optimized for user-defined cost metrics. Empirical experiments show that our cost-sensitive modeling and deployment techniques are effective in reducing the overall cost of intrusion detection.

1 Introduction

Accompanying our growing dependency on network-based computer systems is an increased importance on protecting our information systems. Intrusion detection (ID), the process of identifying and responding to malicious activity targeted at computing and networking resources [2], is a critical component of infrastructure protection mechanisms.

A natural tendency in developing an intrusion detection system (IDS) is trying to maximize its *technical effectiveness*. This often translates into IDS vendors attempting to use brute force to correctly detect a larger spectrum of intrusions than their competitors. However, the goal of catching all attacks has proved to be a major technical challenge. After more than two decades of research and development efforts, the leading IDSs still have marginal detection rates and high false alarm rates, especially in the face of *stealthy* or *novel* intrusions [1, 17]. This goal is also impractical for IDS deployment, as the constraints on time (i.e., processing speed) and resources (both human and computer) may become overwhelmingly restrictive. An IDS usually performs *passive monitoring* of network or system activities rather than *active filtering* (as is the case with Firewalls). It is essential for an IDS to keep up with the throughput of the data stream that it monitors so that intrusions can be detected in a timely manner. A real-time IDS can thus become vulnerable to *overload attacks* [22]. In such an attack, the attacker first directs a huge amount of malicious traffic at the IDS (or some machine it is monitoring) to the point that it can no longer track all data necessary to detect every intrusion. The attacker can then successfully execute the intended intrusion, which the IDS will fail to detect. Similarly, an incident response team can be overloaded by intrusion reports and may be forced to raise detection and response thresholds [7], resulting in real attacks being ignored. In such a situation, focusing limited resources on the most damaging intrusions is a more beneficial and effective approach.

A very important but often neglected facet of intrusion detection is its *cost-effectiveness*, or *cost-benefit* trade-off. An educated decision to deploy a security mechanism such as an IDS is often motivated by the needs of security risk management [5, 10, 21]. The objective of an IDS is therefore to provide protection to the information assets that are at risk and have value to an organization. An IDS needs to be cost-effective in that it should cost no more than the expected level of loss from intrusions. This requires that an IDS consider the trade-off among cost factors, which at the minimum should include development cost, the cost of damage caused by an intrusion, the cost of manual or automatic response to an intrusion, and the operational cost, which measures constraints on time and computing resources. For example, an intrusion which has a higher response cost than damage cost should usually not be acted upon beyond simple logging.

Currently these cost factors are, for the most part, ignored as unwanted complexities in the development process of IDSs. This is caused by the fact that achieving a reasonable degree of technical effectiveness is already a challenging task, given the complexities of today's network environments and the manual effort of knowledge-engineering approaches (e.g., encoding expert rules). Some IDSs do try to minimize operational cost. For example, the Bro [22] scripting language for specifying intrusion detection rules does not support `for`-loops because iteration through a large number of connections is considered time consuming. However, we do not know of any IDS that considers any other cost factors. These cost factors are not sufficiently considered in the deployment of IDSs either because many organizations are not educated about the cost-benefits of security systems, and analyzing site-specific cost factors is very difficult. Therefore, we believe that the security community as a whole must study the cost-effective aspects of IDSs in greater detail to help make intrusion detection a more successful technology.

We have developed a data mining framework for building intrusion detection models in an effort to automate the process of IDS development and lower its development cost. The framework uses data mining algorithms to compute activity patterns and extract predictive features, and then applies machine learning algorithms to generate detection rules [14, 15]. Results from the 1998 DARPA Intrusion Detection Evaluation showed that our ID model was one of the best performing of all the participating systems, most of which were knowledge-engineered [17].

In this paper, we examine the relevant cost factors, cost models, and cost metrics related to IDSs, and report the results of our current research in extending our data mining framework to build cost-sensitive models for intrusion detection. We propose to use cost-sensitive machine learning techniques that can automatically construct detection models optimized for overall cost metrics instead of mere statistical accuracy.

We do not suggest that accuracy be ignored, but rather that cost factors be *included* in the process of developing and evaluating IDSs. Our contributions are not the specific cost models and cost metrics described, but rather the principles of cost analysis and modeling for intrusion detection.

2 Cost Factors and Metrics

In order to build cost-sensitive ID models, we must first understand the relevant cost factors and the metrics used to define them. Borrowing ideas from the related fields of credit card and cellular phone fraud detection, we identify the following major cost factors related to intrusion detection: damage cost, response cost, and operational cost. Damage cost (DCost) characterizes the amount of damage to a target resource by an attack when intrusion detection is unavailable or ineffective. Response cost (RCost) is the cost of acting upon an alarm or log entry that indicates a potential intrusion. Operational cost (OpCost) is the cost of processing the stream of events being monitored by an IDS and analyzing the activities using intrusion detection models. We will discuss these factors in greater detail in Section 2.2.

Cost-sensitive models can only be constructed and evaluated when cost metrics are given. The issues involved in the measurement of cost factors have been studied by the computer risk analysis and security assessment communities. The literature suggests that attempts to fully quantify all factors involved in cost modeling usually generate misleading results because not all factors can be reduced to discrete dollars (or some other common unit of measurement) and probabilities [3, 6, 9, 10, 13]. It is recommended that qualitative analysis be used to measure the *relative magnitudes* of cost factors. It should also be noted that cost metrics are often site-specific because each organization has its own security policies, information assets, and risk factors [21].

2.1 Attack Taxonomy

An attack taxonomy is essential in producing meaningful cost metrics. The taxonomy groups intrusions into different types so that cost measurement can be performed for categories of similar attacks. Intrusions can be categorized and analyzed from different perspectives. Lindqvist and Jonsson introduced the concept of the *dimension* of an intrusion and used several dimensions to classify intrusions [16]. The *intrusion results* dimension categorizes attacks according to their effects (e.g., whether or not denial-of-service is accom-

Table 1: An Attack Taxonomy for DARPA Data

Main Category (by results)	Description	Sub-Category (by techniques)	Description	Cost
1. ROOT	illegal root access is obtained.	1.1 local	by first logging in as a legitimate user on a local system, e.g., *buffer overflow* on local system programs such as *eject*.	DCost=100 RCost=40
		1.2 remote	from a remote host, e.g., buffer overflow of some daemon running suid root.	DCost=100 RCost=60
2. R2L	illegal user access is obtained from outside.	2.1 single	a single event, e.g., guessing passwords.	DCost=50 RCost=20
		2.2 multiple	multiple events, hosts, or days, e.g., the *multihop* attack.	DCost=50 RCost=40
3. DOS	Denial-of-Service of target is accomplished.	3.1 crashing	using a single malicious event (or a few packets) to crash a system, e.g., the *teardrop* attack.	DCost=30 RCost=10
		3.2 consumption	using a large number of events to exhaust network bandwidth or system resources, e.g., *synflood*.	DCost=30 RCost=15
4. PROBE	information about the target is gathered.	4.1 simple	many of probes within a short period of time, e.g., fast *port-scanning*.	DCost=2 RCost=5
		4.2 stealth	probe events are distributed sparsely across a long time windows, e.g. slow *port-scanning*.	DCost=2 RCost=7

plished). It can therefore be used to assess the damage cost and response cost. The *intrusion techniques* dimension categorizes attacks based on their methods (e.g., resource or bandwidth consumption). It therefore affects the operational cost and the response cost. Also, the *intrusion target* dimension categorizes attacks according to the resource being targeted and affects both damage and response costs.

Our attack taxonomy is illustrated in Table 1, and categorizes intrusions that occur in the DARPA Intrusion Detection Evaluation dataset, which was collected in a simulated military environment by MIT Lincoln Lab [17]. In this dataset, each event to be monitored is a network connection, and the resources being attacked are mainly the network services (e.g., *http*, *smtp*, etc.) and system programs on a particular host in the network. We use the taxonomy described in Table 1 to first categorize the intrusions occurring in the dataset into ROOT, DOS, R2L, and PROBE, based on their intrusion results. Then within each of these categories, the attacks are further partitioned by the techniques used to execute the intrusion. The ordering of sub-categories is of increasing complexity of the attack method. Attacks of each sub-category can be further partitioned according to the attack targets. For simplicity, the *intrusion target* dimension is not shown. This table also shows the damage cost (DCost) and response cost (RCost) of each intrusion category. The operation cost (OpCost) is not shown because it is measured in different units. These cost factors will be discussed later in the paper.

2.2 Cost Factors

When measuring cost factors, we only consider individual attacks detectable by IDSs. For example, a coordinated attack, or an *attack scenario*, that involves port-scanning a network, gaining user-level access to the network illegally, and finally acquiring root access, would normally be detected and responded to by an IDS as three separate attacks because most IDSs are designed to respond quickly to events occurring in real-time. It is therefore reasonable to measure the attacks individually. Researchers are actively studying algorithms and systems for detecting complex attack scenarios. As part of our future work, we will study the cost-sensitive aspects of detection for attack scenarios.

2.2.1 Damage Cost

There are several factors that determine the damage cost of an attack. Northcutt uses *criticality* and *lethality* to quantify the damage that may be incurred by some intrusive behavior [21].

Criticality measures the importance, or value, of the target of an attack. This measure can be evaluated according to a resource's functional role in an organization or its relative cost of *replacement*, *unavailability*, and *disclosure* [10]. Similar to Northcutt's analysis, we assign 5 points for firewalls, routers, or DNS servers, 4 points for mail or Web servers, 2 points for UNIX workstations, and 1 point for Windows or DOS workstations. Lethality measures the degree of damage that could potentially be caused by some attack. For example, a more lethal attack that helped an intruder gain root access would have a higher damage cost than if the attack gave the intruder local user access. Other damage may include the discovery of knowledge about network infrastructure or preventing the offering of some critical service. For each main attack category in Table 1, we define a relative lethality scale and use it as the *base damage cost*, or $base_D$. When assigning damage cost according to the criticality of the target, we can use the *intrusion target* dimension. Using these metrics, we can define the damage cost of an attack targeted at some resource as $criticality \times base_D$. For example, a DOS attack targeted at a firewall has $DCost = 150$, while the same attack targeted at a Unix workstation has $DCost = 60$.

In addition to criticality and lethality, we define the *progress* of an attack to be a measure of how successfully an attack is in achieving its goals. For example, a Denial-of-Service (DOS) attack via resource or bandwidth consumption (e.g. SYN flooding) may not incur damage cost until it has progressed to the point where the performance of the resource under attack is starting to suffer. The progress measure can be used as an estimate of the percentage of the maximum damage cost that should be accounted for. That is, the actual cost is $progress \times criticality \times base_D$. However, in deciding whether or not to respond to an attack, it is necessary to compare the maximum possible damage cost with the response cost. This requires that we assume a worst-case scenario in which $progress = 1.0$.

2.2.2 Response Cost

Response cost depends primarily on the type of response mechanisms being used. This is usually determined by an IDS's capabilities, site-specific policies, attack type, and the target resource [5]. Responses may be either automated or manual, and manual responses will clearly have a higher response cost.

Responses to intrusions that may be automated include the following: termination of the offending connection or session (either killing a process or resetting a network connection), rebooting the targeted system, recording the session for evidence gathering purposes and further investigation, or implementation of a packet-filtering rule [2, 21]. In addition to these responses, a notification may be sent to the administrator of the offending machine via e-mail in case that machine was itself compromised. A more advanced response which has not been successfully employed to date could involve the coordination of response mechanisms in disparate locations to halt intrusive behavior closer to its source.

Additional manual responses to an intrusion may involve further investigation (perhaps to eliminate action against false positives), identification, containment, eradication, and recovery [21]. The cost of manual response includes the labor cost of the response team, the user of the target, and any other personnel that participate in response. It also includes any downtime needed for repairing and patching the targeted system to prevent future damage.

We estimate the relative complexities of typical responses to each attack type in Table 1 in order to define the relative *base response cost*, or $base_R$. Again, we can take into account the criticality of the attack target when measuring response cost. That is, the cost is $criticality \times base_R$. In addition, attacks using simpler techniques (i.e., sub-categories $x.1$ in our taxonomy) generally have lower response costs than more complex attacks (i.e., sub-categories $x.2$), which require more complex mechanisms for effective response.

2.2.3 Operational Cost

The main cost inherent in the operation of an IDS is the amount of time and computing resources needed to extract and test features from the raw data stream that is being monitored[1]. We associate OpCost with time because a real-time IDS must detect an attack while it is in progress and generate an alarm as quickly as possible so that damage can be minimized. A slower IDS which uses features with higher computational costs should therefore be penalized. Even if a computing resource has a "sunken cost" (e.g., a dedicated IDS box has been purchased in a single payment), we still assign some cost to the expenditure of its resources as they are used. If a resource is used by one task, it may not be used by another task at the same time. The cost of computing resources is therefore an important factor in prioritization and decision making.

Some features cost more to gather than others. However, costlier features are often more informative for detecting intrusions. For example, features that examine events across a larger time window have more

[1]For simplicity, we omit the discussion of personnel cost involved in administering and maintaining an IDS.

information available and are often used for "correlation analysis [2]" in order to detect extended or coordinated attacks such as slow host or network scans [5]. Computation of these features is costly because of their need to store and analyze larger amounts of data.

Based on our extensive experience in extracting and constructing predictive features from network audit data [14], we classify features into four relative levels, based on their computational costs:

- Level 1 features can be computed from the first packet, e.g., the *service*.
- Level 2 features can be computed at any point during the life of the connection, e.g., the *connection state* (*SYN_WAIT, CONNECTED, FIN_WAIT,* etc.).
- Level 3 features can be computed at the end of the connection, using only information about the connection being examined, e.g., the *total number of bytes sent from source to destination*.
- Level 4 features can be computed at the end of the connection, but require access to data of potentially many other prior connections. These are the temporal and statistical features and are the most costly to compute. The computation of these features may require values of the lower level (i.e., levels 1, 2, and 3) features.

We can assign relative magnitudes to these features according to their computational costs. For example, level 1 features may cost 1, level 2 features may cost 5, level 3 features may cost 10, and level 4 features may cost 100. These estimations have been verified empirically using a prototype system for evaluating our ID models in real-time that has been built in coordination with Network Flight Recorder [20].

3 Cost Models

A cost model formulates the total expected cost of intrusion detection. It considers the trade-off among all relevant cost factors and provides the basis for making appropriate cost-sensitive detection decisions. We first examine the cost trade-off associated with each possible outcome of observing some event e, which may represent a network connection, a user's session on a system, or some logical grouping of activities being monitored. In our discussion, we say that $e = (a, p, r)$ is an event described by the attack type a (which can be *normal* for a truly normal event), the progress p of the attack, and the target resource r. The detection outcome of e is one of the following: false negative (FN), false positive (FP), true positive (TP), true negative (TN), or misclassified hit. The costs associated with these outcomes are known as *consequential costs* (CCost), as they are incurred as a consequence of prediction, and are outlined in Table 2.

FN Cost is the cost of not detecting an attack, and is always incurred by systems that do not install IDSs. When an IDS falsely decides that a connection is not an attack and does not respond to the attack, the attack will succeed, and the target resource will be damaged. The FN Cost is therefore defined as the damage cost associated with event e, or DCost(e).

TP Cost is incurred in the event of a correctly classified attack, and involves the cost of detecting the attack and possibly responding to it. To determine whether response will be taken, $RCost$ and $DCost$ must be considered. If the damage done by the attack to resource r is less than RCost, then ignoring the attack actually reduces the overall cost. Therefore, if RCost(e) $>$ DCost(e), the intrusion is not responded to beyond simply logging its occurrence, and the loss is DCost(e). If RCost(e) $\leq$ DCost(e), then the intrusion is acted upon and the loss is limited to RCost(e). In reality, however, by the time an attack is detected and response ensues, some damage may have incurred. To account for this, TP cost may be defined as RCost(e) $+ \epsilon_1$DCost(e), where $\epsilon_1 \in [0, 1]$ is a function of the progress p of the attack.

FP Cost is incurred when an event is incorrectly classified as an attack, i.e., when $e = (normal, p, r)$ is misidentified as $e' = (a, p', r)$ for some attack a. If RCost(e') $\leq$ DCost(e'), a response will ensue and the response cost, RCost(e'), must be accounted for as well. Also, since normal activities may be disrupted due to unnecessary response, false alarms should be penalized. For our discussion, we use PCost(e) to represent the penalty cost of treating a legitimate event e as an intrusion. For example, if e is aborted, PCost(e) can be the damage cost of a DOS attack on resource r, because a legitimate user may be denied access to r.

TN Cost is always 0, as it is incurred when an IDS correctly decides that an event is normal. We therefore bear no cost that is dependent on the outcome of the decision.

Table 2: Model for Consequential Cost

Outcome	Consequential Cost *CCost(e)*	Condition
Miss (False Negative, *FN*)	DCost(e)	
False Alarm (False Positive, *FP*)	RCost(e') $+$ PCost(e) 0	if DCost(e') $\geq$ RCost(e') if DCost(e') $<$ RCost(e')
Hit (True Positive, *TP*)	RCost(e) $+ \epsilon_1$DCost(e), $0 \leq \epsilon_1 \leq 1$ DCost(e)	if DCost(e) $\geq$ RCost(e) if DCost(e) $<$ RCost(e)
Normal (True Negative, *TN*)	0	
Misclassified Hit	RCost(e') $+ \epsilon_2$DCost(e), $0 \leq \epsilon_2 \leq 1$ DCost(e)	if DCost(e') $\geq$ RCost(e') if DCost(e') $<$ RCost(e')

Misclassified Hit Cost is incurred when the wrong type of attack is identified, i.e., an event $e = (a, p, r)$ is misidentified as $e' = (a', p', r)$. If RCost(e') $\leq$ DCost(e'), a response will ensue and RCost(e') needs to be accounted for. Since the response taken is effective against attack type a' rather than a, some damage cost of ϵ_2DCost(e) will be incurred due to the true attack. Here $\epsilon_2 \in [0, 1]$ is a function of the progress p and the effect of the response intended for a' on a.

We can now define the cost model for an IDS. When evaluating an IDS over some labeled test set E, where each event, $e \in E$, has a label of *normal* or one of the intrusions, we define the cumulative cost of the IDS as follows:

$$CumulativeCost(E) = \sum_{e \in E} (CCost(e) + OpCost(e)) \qquad (1)$$

where CCost(e), the consequential cost of the prediction by the IDS on e, is defined in Table 2.

It may not always be possible to fold damage and response costs into the same measurement unit, instead, each should be analyzed in its own relative scale. We must, however, compare and then combine the two so that we can compute CCost(e) for use in the calculation of CumulativeCost in Equation 1.

One way is to decide first under what conditions to not respond to particular intrusions. For example, assuming that probing attacks should not be responded to and that the damage cost for probing is 2, then the response cost for probing must be greater, say, 20. Similarly, if the attack type with the lowest damage cost should not be ignored, then the corresponding lowest response cost should be a smaller value. Once a starting value is defined, remaining values can be computed according to the relative scales discussed in

Section 2.2.

OpCost(e) in Equation 1 can be computed as the sum of the computational costs of all the features used during rule checking. Since OpCost(e) and CCost(e) use two different measurement units and there is no possibility of comparing the two, as with damage cost and response cost, we can use Equation 1 at a conceptual level. That is, when evaluating IDSs, we can consider both the cumulative OpCost and cumulative CCost, but actual comparisons are performed separately using the two costs. This inconvenience cannot be overcome easily unless all cost factors are represented using a common measurement unit, or there is a reference or comparison relation for all the factors. Site-specific policies can be used to determine how to uniformly measure these factors.

4 Cost-Sensitive Modeling

Similar to risk analysis [3], cost-sensitive modeling for intrusion detection must be performed periodically because cost metrics must take into account changes in information assets and security policies. It is therefore important to develop tools that can automatically produce cost-sensitive models for given cost metrics.

We have developed and evaluated the use of many machine learning methods for reducing the CumulativeCost of intrusion detection [12, 18]. Because of space constraints, in this section and Section 5 we describe and evaluate the particular methods which have proven most effective in building cost-sensitive models.

4.1 Reducing Operational Cost

In order to reduce OpCost, ID models need to use low cost features as often as possible while still maintaining a desired level of accuracy. Our approach is to build multiple ID models, each of which uses different sets of features at different cost levels. Low cost models are always evaluated first by the IDS, and high cost models are used only when the low cost models cannot make a prediction with sufficient accuracy.

We implement this multiple-model approach using RIPPER [8], a rule induction algorithm. However, other machine learning algorithms or knowledge-engineering methods may be used as well.

Given a training set in which each event is labeled as either *normal* or some intrusion, RIPPER builds an *ordered* or *unordered* ruleset. Each rule in the ruleset uses the most discriminating feature values for classifying a data item into one of the classes. A rule consists of conjunctions of feature value comparisons, and if the rule evaluates to *true*, then a prediction is made. An example rule for predicting *teardrop* is "**if** $number\_bad\_fragments \geq 2$ **and** $protocol = udp$ **then** $teardrop$." Before discussing the details of our approach, it is necessary to outline the advantages and disadvantages of ordered and un-ordered rulesets.

Ordered Rulesets: An ordered ruleset has the form **if** r_1 **then** i_1 **elseif** r_2 **then** $i_2, \ldots,$ **else** $default$, where r_n is a rule (its conditions) and i_n is the class label predicted by that rule. Before learning, RIPPER first orders the classes by one of the following heuristics: $+freq$, which orders by increasing frequency in the training data; $-freq$, by decreasing frequency; $given$, which is a user-defined ordering; mdl, which uses the minimal description length to guess an optimal ordering [19] . After arranging the classes, RIPPER finds rules to separate $class_1$ from classes $class_2, \ldots, class_n$, then rules to separate $class_2$ from classes $class_3, \ldots, class_n$, and so on. The final class, $class_n$, will become the default class. The end result is that rules for a single class will always be grouped together, but rules for $class_i$ are possibly simplified, because they can assume that the class of the example is one of $class_i, \ldots, class_n$. If an example is covered by rules from two or more classes, this conflict is resolved in favor of the class that comes first in the ordering.

An ordered ruleset is usually succinct and efficient. Evaluation of an entire ordered ruleset does not require each rule to be tested, but proceeds from the top of the ruleset to the bottom until any rule evaluates to $true$. The features used by each rule can be computed one by one as evaluation proceeds. The operational cost to evaluate an ordered ruleset for a given event is the total cost of computing unique features until a prediction is made. For intrusion detection, a $-freq$ ruleset is usually lowest in operational cost and accurately classifies normal events. This is because the first rules of the ruleset identify normal, which is usually the most frequently occurring class. On the contrary, a $+freq$ ruleset would most likely be higher in operational cost but more accurate in classifying intrusions because the ruleset partitions intrusions from normal events early in its evaluation, and *normal* is the final default classification. Depending on the class ordering, the performances of $given$ and mdl will lie between those of $-freq$ and $+freq$.

Un-ordered Rulesets: An un-ordered ruleset has at least one rule for each class and there are usually many rules for frequently occurring classes. There is also a default class which is used for prediction when none of these rules are satisfied. Unlike ordered rulesets, all rules are evaluated during prediction and conflicts are broken by using the most accurate rule. Un-ordered rulesets, in general, contain more rules and are less efficient in execution than $-freq$ and $+freq$ ordered rulesets, but there are usually several rules of high precision for the most frequent class, resulting in accurate classification of normal events.

With the advantages and disadvantages of ordered and un-ordered rulesets in mind, we propose the following multiple ruleset approach:

- We first generate multiple training sets T_1, T_2, T_3, T_4 using different feature subsets. T_1 uses only cost 1 features; T_2 uses features of costs 1 and 5; T_3 uses features of costs 1, 5, and 10; and T_4 uses all available features of costs 1, 5, 10, and 100.
- Rulesets R_1, R_2, R_3, R_4 are learned using their respective training sets. R_4 is learned as either $+freq$ or $-freq$ ruleset for efficiency, as it may contain the most costly features. R_1, R_2, R_3 are learned as either $-freq$ or un-ordered rulesets, as they will contain accurate rules for classifying normal events and we filter normal as early as possible to reduce operational cost. $Given$ and mdl might be used, but their performance would not be better.
- A precision measurement p_r is computed for *every rule*, r, except for the rules in R_4[2].
- A threshold value τ_i is obtained for every class, and determines the tolerable precision required for a prediction to be made in execution.

In real-time execution, the feature computation and rule evaluation proceed as follows:

[2]Precision describes how accurate a prediction is. If P is the set of predictions with label i and W is the set of instances with label i in the data set, by definition, $p = \frac{|P \cap W|}{|P|}$.

- R_1 is evaluated and a prediction i is made by some rule r.
- If $p_r \geq \tau_i$, the prediction i is final. In this case, no more features are computed and the system examines the next event. Otherwise, additional features required by R_2 are computed and R_2 is evaluated.
- This process continues until a final prediction is made. The evaluation of R_4 always produces a final prediction because R_4 uses all features.

The precision and threshold values used by the multiple model approach can be obtained during model training from the training set, or can be computed using a separate hold-out validation set. The precision of a rule can be obtained easily from the positive and negative counts of a rule: $\frac{p}{p+n}$. Threshold values are set to the precisions of the rules in a single ruleset using all features (R_4) for each class in the chosen dataset, as we do not want to make less precise classifications in R_1, R_2, R_3 than would be made using R_4.

4.1.1 Real-Time Implementation

We have implemented a system that is capable of evaluating a set of cost-sensitive models in real-time. This system uses a sensor for extracting light-weight, or "primitive," features (usually levels 1 and 2) from raw network traffic data. Higher level feature computation and model evaluation are offloaded to a separate entity, "Judge." The motivation for offloading this computation and evaluation is to avoid overburdening the sensor (in this case a packet sniffing engine), which must be able to monitor very high-bandwidth networks. Judge uses models computed using the techniques described in the previous section.

Higher level features are computed and models are evaluated by Judge at different points in a connection as more primitive features become available. The sensor informs Judge of new feature values and updates to feature values that are maintained by Judge throughout a connection's existence, as they become available. Such notifications happen whenever there is a change in the connection's state (e.g., a connection has progressed from SYN_WAIT to CONNECTED). Sensors also update certain feature values whenever an "exception" event is observed. Exceptions are occurrences which should immediately update the value of a specific feature. For example, if two fragmented packets come in and the offsets for defragmentation are correct, the *bad_frag_offset* feature must be updated immediately.

Upon each state change and exception event, Judge compares the set of primitive features that are available for the given connection, F_c, with the set of primitive features used by all higher level features in each ID model, F_{R_i}. If for any i, $F_{R_i} \subseteq F_c$, then higher level features requiring the primitive features in F_c are computed and the model R_i is evaluated. The logic for determining when a prediction is made is the same as is described in the previous section.

Thus far, we have implemented this system using NFR as the sensor, however, the protocol for communication between the sensor and Judge allows any sensor which extracts features from a data stream to be used. We are in the process of adapting a number of host-based sensors to serve as Judge clients.

4.2 Reducing Consequential Cost

A traditional IDS that does not consider the trade-off between RCost and DCost will attempt to respond to every intrusion that it detects. As a result, the consequential cost for FP, TP, and misclassified hits will always include some response cost. We use a cost-sensitive decision module to determine whether response should ensue based on whether DCost is greater than RCost.

The decision module takes as input an intrusion report generated by the detection module. The report contains the name of the predicted intrusion and the name of the target, which are then used to look up the pre-determined DCost and RCost. If DCost $\geq$ RCost, the decision module invokes a separate module to initiate a response; otherwise, it simply logs the intrusion report.

The functionality of the decision module can also be implemented using a data re-labeling technique such as MetaCost [11], which re-labels intrusions with DCost $<$ RCost to *normal* so that the generated model will not contain rules for predicting these intrusions at all. We have experimented with such a mechanism [12] and have found that models trained on these datasets are much smaller and incur less operational cost. However, reducing consequential cost using a post-detection decision module eliminates the time consuming need to re-train models when cost factors change.

5 Experiments

Our experiments use data that was distributed by the 1998 DARPA Intrusion Detection Evaluation Program. The data was gathered from a military network with a wide variety of intrusions injected into the network over a period of 7 weeks. The details of our data mining framework for data pre-processing and feature extraction is described in our previous work [15]. We used 80% of the data for training the detection models. The training set was also used to calculate the precision of each rule and the threshold value for each class label. The remaining 20% were used as a test set for evaluation of the cost-sensitive models.

5.1 Measurements

We measure expected operational and consequential costs in our experiments. The expected average operational cost per event over the entire test set is defined as $\frac{\sum_{e \in S} OpCost(e)}{|S|}$. In all of our reported results, $OpCost(e)$ is computed as the sum of the feature computation costs of all unique features used by all rules evaluated until a prediction is made for event e. If any level 3 features (of cost 100) are used at all, the cost is counted only once. This is done because a natural optimization of rule evaluation is to compute all statistical and temporal features in one iteration through the event database.

For each event in the test set, its CCost is computed as follows: the outcome of the prediction (i.e., FP, TP, FN, TN, or misclassified hit) is used to determine the corresponding conditional cost expression in Table 2; the relevant RCost, DCost, and PCost are then used to compute the appropriate CCost. The CCost for all events in the test set are then summed to measure total CCost as reported in Section 5.2. In all experiments, we set $\epsilon_1 = 0$ and $\epsilon_2 = 1$ in the cost model of Table 2. Setting $\epsilon_1 = 0$ corresponds to the optimistic belief that the correct response will be successful in preventing damage. Setting $\epsilon_2 = 1$ corresponds to the pessimistic belief that an incorrect response does not prevent the intended damage at all.

Table 3: Average OpCost Per Connection

	-	$\pm\pm\pm-$	$----$	+	$\pm\pm\pm+$	$---+$
OpCost	128.70	48.43	42.29	222.73	48.42	47.37
%rdc	N/A	56.68%	67.14%	N/A	78.26%	78.73%

Table 4: CCost Comparison

Model Format		$-$	$\pm\pm\pm-$	$----$	$+$	$\pm\pm\pm+$	$---+$
Cost Sensitive	CCost	25776	25146	25226	24746	24646	24786
	%rdc	87.8%	92.3%	91.7%	95.1%	95.8%	94.8%
Cost Insensitive	CCost	28255	27584	27704	27226	27105	27258
	%rdc	71.4%	75.1%	74.3%	77.6%	78.5%	77.4%
%err		0.193%	0.165%	0.151%	0.085%	0.122%	0.104%

5.2 Results

In all discussion of our results, we use $+$, $-$ and $\pm$ to represent $+freq$, $-freq$ and *un-ordered* rulesets, respectively. A multiple model approach is denoted as a sequence of these symbols. For example, $----$ represents a multiple model where all rulesets are $-freq$.

Table 3 shows the average operational cost per event for a single classifier approach (R_4 learned as $-$ or $+$) and the respective multiple model approaches ($\pm\pm\pm-$, $----$ or $\pm\pm\pm+$, $---+$). The first row below each method is the average OpCost per event and the second row is the reduction ($\%rdc$) by the multiple model over the respective single model, $\frac{Single-Multiple}{Single} \times 100\%$. As clearly shown in the table, there is always a significant reduction by the multiple model approach. In all 4 configurations, the reduction is more than 57% and $---+$ has a reduction in operational cost by as much as 79%. This significant reduction is due to the fact that R_1, R_2, R_3 are very accurate in filtering $normal$ events and a majority of events in real network environments (and consequently our test set) are $normal$. Our multiple model approach computes more costly features only when they are needed.

CCost measurements are shown in Table 4. The $Maximal$ loss is the total cost incurred when always predicting $normal$, or $\sum DCost_i$. This value is 38256 for our test set. The $Minimal$ loss is the cost of correctly predicting all connections and responding to an intrusion only when $DCost(i) \geq RCost(i)$. This value is 24046 and it is calculated as $\sum_{DCost(i)<RCost(i)} DCost(i) + \sum_{DCost(j)\geq RCost(j)} RCost(j)$. A reasonable method will have a CCost measurement between $Maximal$ and $Minimal$ losses. We define reduction as $\%rdc = \frac{Maximal-CCost}{Maximal-Minimal} \times 100\%$ to compare different models. As a comparison, we show the results of both "cost sensitive" and "cost insensitive" methods. A cost sensitive method only initiates a response if $DCost \geq RCost$, and corresponds to the cost model in Table 2. A cost insensitive method, on the other hand, responds to every predicted intrusion and is representative of current brute-force approaches to intrusion detection. The last row of the table shows the error rate ($\%err$) of each model.

As shown in Table 4, the cost sensitive methods have significantly lower CCost than the respective cost insensitive methods for both single and multiple models. The reason is that a cost sensitive model will only respond to an intrusion if its response cost is lower than its damage cost. The error rates for all 6 models are very low ($< 0.2\%$) and very similar, indicating that all models are very accurate. However, there is no strong correlation between error rate and CCost, as a more accurate model may not necessarily have detected more costly intrusions. There is little variation in the total CCost of single and multiple models in both cost-sensitive and cost-insensitive settings, showing that the multiple model approach, while decreasing OpCost, has little effect on CCost. Taking both OpCost and CCost into account (Tables 3 and 4), the highest performing model is $---+$.

It is important to note that all results shown are specific to the distribution of intrusions in the test data set. We can not presume that any distribution may be typical of all network environments.

6 Related Work

Several researchers and experts have pointed out the importance of using intrusion detection (and computer security in general) as a means of risk management [10, 5, 21]. Our work in cost-sensitive modeling for IDSs has benefited from their insightful analysis and extensive real-world experiences.

Researchers have begun to develop principles and theories for intrusion detection. Axelsson [4] pointed out that the established fields of detection and estimation theory are similar to intrusion detection. For example, the subject of an anomaly detection model corresponds to the "signal source" in detection and estimation theory, the auditing mechanism corresponds to "signal transmission," the audit data corresponds to the "observation space," and in both cases, the task is to derive detection rules. Therefore, the results from detection and estimation, which have been found applicable to a wide range of problems, may be used in the IDS domain. One of the key findings by Axelsson is that a detection model should be optimized for some utility function, which need not represent statistical accuracy, but instead could involve some definition of cost. This finding validates the motivation of our research described in this paper.

As discussed throughout this paper, our work draws from research in computer security assessment and intrusion taxonomies. In particular, Glaseman et al. discussed a model for evaluating the total expected cost in using a security system s as $\mathrm{C}(s) = \mathrm{O}(s) + \mathrm{D}(s)$, where $\mathrm{O}(s)$ is the operational cost of s and $\mathrm{D}(s)$ is the expected loss [13]. $\mathrm{D}(s)$ is calculated by summing the products of exposed value and the probability of safeguard failure over all possible threats. This model is similar to our cost model for IDSs, as defined in Equation 1. However, our definition of consequential cost allows cost-based optimization strategies to be explored because it includes the response cost and models its relationship with damage cost.

Credit card and cellular phone fraud detection are closely related to intrusion detection because they also deal with detecting abnormal behavior. Both of these applications are motivated by cost-saving, and therefore, use cost-sensitive modeling techniques. In credit card fraud detection, for example, the cost factors include operation cost, the personnel cost of investigating a potentially fraudulent transaction (known as challenge cost), and loss (damage cost). If the dollar amount of a suspected transaction is lower than the challenge cost, the transaction is authorized and the credit card company will take the potential loss. Since the cost factors in fraud detection can be folded into dollar amounts, the cost-sensitive analysis and modeling tasks are much more simple than in intrusion detection.

Cost-sensitive modeling is an active research area in data mining and machine learning because of the demand from application domains such as medical diagnosis and fraud and intrusion detection. Several techniques have been proposed for building models optimized for given cost metrics. In our research we study the principles behind these general techniques and develop new approaches according to the cost models specific to IDSs.

7 Conclusion

It is very important to establish the cost-effectiveness of intrusion detection because the ultimate goal of an IDS is to protect the information assets that are at risk and are most valuable to an organization. In this paper, we have examined cost factors that are relevant to intrusion detection, which include development cost, operational cost, damage cost, and response cost. We have shown that it is necessary to use an attack taxonomy along with organization-specific security policies and priorities to measure these cost factors. We studied the trade-off relationships among these factors and defined consequential cost to be the cost associated with the predictions of an IDS. The total expected cost of an IDS is the sum of the operational and consequential costs. The cost-benefit of an IDS is manifested in its abilities to reduce this total expected cost. We presented a multiple model machine learning approach for reducing operational cost and a post-detection decision module for reducing consequential cost. Empirical evaluation using the DARPA Intrusion Evaluation dataset shows that our approaches are indeed effective.

As pointed out by Dorothy Denning, cost analysis (and risk assessment in general) is not an exact science because precise measurement of relevant factors is often impossible [10]. Cost-benefit analysis and modeling, however informal or incomplete, is often very helpful for an organization to determine appropriate protection mechanisms. The study of cost-sensitive modeling for intrusion detection is both challenging and extremely important. Our main contributions to this study are in the development of a framework for analyzing cost factors and building cost-sensitive models. In doing so, we offer a better understanding of the development and deployment of cost-effective IDSs.

7.1 Future Work

It is possible to improve the accuracy of ID models trained using our multiple model approach by using the maximum precision of all models for a given class as the threshold for firing a prediction, rather than the precision of the final and most costly model. The motivation for this approach is that learning algorithms do not necessarily produce more accurate models as more features are used in training. In fact, models using richer feature sets may even be too complex and less accurate for certain classes than less costly models. We would expect to see slightly higher accuracy using this approach. However, since model evaluation may proceed farther into the chain of more costly models, there would be a penalty in operational cost.

One limitation of our current modeling techniques is that when cost metrics change, it is necessary to reconstruct new cost-sensitive models. For future work, we will study methods for building dynamic models that do not require re-training. These techniques will help reduce the cost of re-learning models due to changes in intra-site cost metrics and deployment at diverse sites with inherently different cost models.

We will study cost-sensitive analysis and modeling techniques for detection of complex attack scenarios. We will also study how to incorporate *uncertainty* of cost analysis due to incomplete or imprecise estimation, especially in the case of *anomaly detection* systems, in the process of cost-sensitive modeling. And we will perform rigorous studies and experiments in a real-word environment to further refine our cost analysis and modeling approaches.

Acknowledgment

This research is supported in part by grants from DARPA (F30602-00-1-0603).

References

[1] J. Allen, A. Christie, W. Fithen, J. McHugh, J. Pickel, and E. Stoner. State of the practice of intrusion detection technologies. Technical Report CMU/SEI-99-TR-028, CMU/SEI, 2000.

[2] E. Amoroso. *Intrusion Detection: An Introduction to Internet Surveillance, Correlation, Traps, Trace Back, and Response*. Intrusion.Net Books, 1999.

[3] A. M. Anderson. Comparing risk analysis methodologies. In D. T. Lindsay and W. L. Price, editors, *Information Security*. Elsevier Science Publishers, 1991.

[4] S. Axelsson. A preliminary attempt to apply detection and estimation theory to intrusion detection. Technical report, Department of Computer Engineering, Chalmers University of Technology, Goteborg, Sweden, 2000.

[5] R. Bace. *Intrusion Detection*. Macmillan Technical Publishing, 2000.

[6] R. P. Campbell and G. A. Sands. A modular approach to computer security risk management. In *AFIPS Conference Proceedings*. AFIPS Press, 1979.

[7] F. Cohen. *Protection and Security on the Information Superhighway*. John Wiley & Sons, 1995.

[8] W. W. Cohen. Fast effective rule induction. In *Machine Learning: the 12th International Conference*, Lake Taho, CA, 1995. Morgan Kaufmann.

[9] DARCOM. *Engineering Design Handbook: Army Weapon Systems Analysis, Part Two (DARCOM-P 706-102)*. US Army Materiel Development And Readiness Command, 1979.

[10] D. Denning. *Information Warfare and Security*. Addison Wesley, 1999.

[11] P. Domingos. Metacost: A general method for making classifiers cost-sensitive. In *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD-99)*, August 1999.

[12] W. Fan, W. Lee, M. Miller, and S. J. Stolfo. Anomaly detection: An incremental approach. submitted for publication, 2000.

[13] S. Glaseman, R. Turn, and R. S. Gaines. Problem areas in computer security assessment. In *Proceedings of the National Computer Conference*, 1977.

[14] W. Lee. *A Data Mining Framework for Constructing Features and Models for Intrusion Detection Systems*. PhD thesis, Columbia University, June 1999.

[15] W. Lee, S. J. Stolfo, and K. W. Mok. A data mining framework for building intrusion detection models. In *Proceedings of the 1999 IEEE Symposium on Security and Privacy*, May 1999.

[16] U. Lindqvist and E. Jonsson. How to systematically classify computer security intrusions. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, Oakland CA, May 1997.

[17] R. Lippmann, D. Fried, I. Graf, J. Haines, K. Kendall, D. McClung, D. Weber, S. Webster, D. Wyschogrod, R. Cunninghan, and M. Zissman. Evaluating intrusion detection systems: The 1998 darpa off-line intrusion detection evaluation. In *Proceedings of the 2000 DARPA Information Survivability Conference and Exposition*, January 2000.

[18] Matthew Miller. Learning cost-sensitive classification rules for network intrusion detection using ripper. Technical report, Computer Science Department, Columbia University, June 1999.

[19] T. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[20] Network Flight Recorder Inc. Network flight recorder. http://www.nfr.com, 1997.

[21] S. Northcutt. *Intrusion Detection: An Analyst's Handbook*. New Riders, 1999.

[22] V. Paxson. Bro: A system for detecting network intruders in real-time. In *Proceedings of the 7th USENIX Security Symposium*, San Antonio, TX, 1998.