25

Data Mining in Security

Objectives:

- Data mining offers a convenient way to monitor the large computer networks, thus providing the security. The data mining system does this by developing a profile of the typical activities of each user in the network.
- To improve the accuracy, data mining programs are used to analyze audit data and extract features that can distinguish normal activities from intrusions; we use artificial anomalies along with the normal and/or intrusion data to produce more effective misuse and anomaly detection models.
- To improve efficiency, the computational costs of features are analyzed and a multiple-model cost-based approach is used to produce detection models with low cost and high accuracy.
- To improve usability, adaptive learning algorithms are used to facilitate model construction and incremental updates; unsupervised anomaly detection algorithms are used to reduce the reliance on labeled data.
- Security of network systems is becoming increasingly important; intrusion detection systems (IDSs) have thus become a critical technology.
- IDSs models generalize from both known attacks and normal behavior in order to detect unknown attacks.
- Data mining-based IDSs (especially anomaly detection systems) have higher false positive rates than traditional handcrafted signature-based methods, making them unusable in real environments.
- The anomaly detection algorithms explore the use of information-theoretic measures, i.e., entropy, conditional entropy, relative entropy, information gain, and information cost to capture intrinsic characteristics of normal data and use such measures to guide the process of building and evaluating anomaly detection models.

Abstract. Data mining can be used to improve efficiency, quality of data, marketing and sales, and has many more benefits. Furthermore, even in the case of security problems, we have addressed the case where data mining tools could be used to

S. Sumathi and S.N. Sivanandam: *Data Mining in Security*, Studies in Computational Intelligence (SCI) **29**, 629–648 (2006)
www.springerlink.com

detect abnormal behavior and intrusions in the system. Data mining also has many applications in detecting fraudulent behavior. While all of these applications of data mining can benefit humans, there is also a dangerous side to mining, since it could be a serious threat to the security and privacy of individuals. An overview on data mining in security and real-time data mining-based intrusion detection system case study is illustrated in this chapter.

Security of network systems is becoming increasingly important, as more and more sensitive information is being stored and manipulated online. Intrusion detection systems (IDSs) have thus become a critical technology to help protect these systems.

Most IDSs are based on handcrafted signatures that are developed by manual encoding of expert knowledge. These systems match activity on the system being monitored to known signatures of attacks. The major problem with this approach is that these IDSs fail to generalize to detect new attacks or attacks without known signatures. Recently, there has been an increased interest in data mining-based approaches to building detection models for IDSs. These models generalize from both known attacks and normal behavior in order to detect unknown attacks. Domain experts can also generate them in a quicker and more automated method than manually encoded models that require difficult analysis of audit data. Several effective data mining techniques for detecting intrusions have been developed, many of which perform close to or better than systems engineered by domain experts.

This chapter discusses several problems inherent in developing and deploying real-time data mining-based IDS and presents an overview of the research, which addresses these problems.

25.1 Data Mining in Security Systems

In June 1994, a computer expert in St. Petersburg, Russia, Vladmir Leonidovich Levin, penetrated the Citibank electronic funds transfer network. Over the course of five months, he funneled 10 million dollars into accounts in California, Israel, Germany, Finland, the Netherlands and Switzerland. He was eventually apprehended, and most of the money was recovered, but the incident revealed the vulnerability of large databases to computer hackers.

Incidents such as the one described above make the security of a company's computer system a serious issue in today's business world. System administrators and security officers monitor these computer networks, often comprising thousands of computers and terabytes of storage space. Their job is daunting; especially since a security violation on one workstation could become a multimillion-dollar incident. The Computer Emergency Response Team, an organization of computer security professionals, estimates that only five percent of companies whose security has been compromised are even aware that they have been infiltrated. Although the raw information needed to detect an intrusion is often available in the audit data recorded by each computer, there is far too much of it generated each day for the system administrators and security officers to inspect it. Even if they tried, the vast majority of the audit record would be completely mundane and innocuous actions.

Data mining offers a convenient way to monitor these large computer networks. By detecting anomalous activities in the logs of computers, a data mining system could flag suspicious events for later inspection by system administrators, allowing them to avoid checking all the normal daily activities. The data mining system does this by developing a profile of the typical activities of each user in the network. Deviations from the expected pattern could be harmful or abusive behavior and would therefore be flagged. The system would have to be flexible enough to compensate for normal deviations from expected behavior like users learning new programs or doing new tasks. One study done at the Purdue University found that a data mining system was able to identify a profiled user 99% of the time and differentiated between a profiled user and another user with almost 94% accuracy.

25.2 Real Time Data Mining-Based Intrusion Detection Systems – Case Study

In this section, we present an overview of the research in real-time data mining-based intrusion detection systems (IDSs) taken from W. Lee (1998). North Carolina State University, Raleigh, NC. We focus on issues related to deploying a data mining-based IDS in a real-time environment. We describe the approaches to address the three types of issues: accuracy, efficiency, and usability. To improve the accuracy, data mining programs are used to analyze audit data and extract features that can distinguish normal activities from intrusions; we use artificial anomalies along with the normal and/or intrusion data to produce more effective misuse and anomaly detection models. To improve efficiency, the computational costs of features are analyzed and a multiple-model cost-based approach is used to produce detection models with low cost and high accuracy. We also present a distributed architecture for evaluating cost-sensitive models in real time. To improve usability, adaptive learning algorithms are used to facilitate model construction and incremental updates; unsupervised anomaly detection algorithms are used to reduce the reliance on labeled data. We also present architecture consisting of sensors, detectors, a data warehouse, and model generation components. This architecture facilitates sharing and storage of audit data and the distribution of new and update models. This architecture also improves the efficiency and scalability of the IDS.

However, successful data mining techniques are themselves not enough to create deployable IDSs. Despite the promise of better detection performance and generalization ability of data mining-based IDSs, there are some inherent difficulties in the implementation and deployment of these systems. We can group these difficulties into three general categories: accuracy (i.e., detection performance), efficiency, and usability. Typically, data mining-based IDSs (especially anomaly detection systems) have higher false positive rates than traditional handcrafted signature-based methods, making them unusable in

real environments. Also, these systems tend to be inefficient (i.e., computationally expensive) during both training and evaluation. This prevents them from being able to process audit data and detect intrusions in real time. Finally, these systems require large amounts of training data and are significantly more complex than traditional systems. In order to be able to deploy real-time data mining-based IDSs, these issues must be addressed.

In this section, we discuss several problems inherent in developing and deploying a real-time data mining-based IDS and present an overview of the research, which addresses these problems. These problems are independent of the actual learning algorithms or models used by an IDS and must be overcome in order to implement data mining methods in a deployable system. An effective data mining-based IDS must address each of these three groups of issues. Although there are tradeoffs between these groups, each can generally be handled separately. We present the key design elements and group them into which general issues they address.

25.2.1 Accuracy

Crucial to the design and implementation of effective data mining-based IDS is defining specifically how detection performance, or accuracy, of these systems is measured. Because of the difference in nature between a data mining-based system and a typical IDS the evaluation metrics must take into account factors that are not important for traditional IDSs.

At the most basic level, accuracy measures how well an IDS detects attacks. There are several key components of an accuracy measurement. One important component is detection rate, which is the percentage of attacks that a system detects. Another component is the false positive rate, which is the percentage of normal data that the system falsely determines to be intrusive. These quantities are typically measured by testing the system on a set of data (normal and intrusions) that are not seen during the training of the system in order to simulate an actual deployment.

There is an inherent tradeoff between detection rate and false positive rate. One way to represent this tradeoff is by plotting the detection rate versus false positive rate on a curve under different parameter values creating an ROC curve. [Receiver Operating Characteristics (ROC) graphs are used in many detection problems because they depict the tradeoffs between detection rate and false positive rate.] A method to compare accuracy between two IDSs is to examine their ROC curves.

In practice, only the small portion of an ROC curve corresponding to acceptably low false positives is of interest, as in a deployable system, only a low false positive rate can be tolerated. Handcrafted methods typically have a fixed detection threshold they perform at a constant detection rate across different false positive rates. In an ROC curve, we can assume that their curve is a straight line at each detection level. Data mining-based systems have the advantage of potentially being able to detect new attacks that handcrafted

methods tends to miss. Data mining-based IDSs are only useful if their detection rate is higher than a handcrafted method's detection rate with an acceptably low false positive rate. Given this framework, the goal is to develop a data mining-based IDS that is capable of outperforming handcrafted signature-based systems at the tolerated false positive rate.

We have developed and applied a number of algorithm-independent techniques to improve the performance of data mining-based IDSs. In this section, we focus on a few particular techniques that have been proven to be empirically successful. We first present a generic framework for extracting features from audit data, which help the discriminate attacks from the normal data. These features can then be used by any detection model building algorithm. We then describe a method for generating artificial anomalies in order to decrease the false positive rate of anomaly detection algorithms. The research has shown that by generating artificial anomalies, we can improve the accuracy of these ID models. Finally, we present a method of combining anomaly and misuse (or signature) detection models. Typically misuse and anomaly detection models are trained and used in complete isolation from each other. The research has shown that by combining the two types of models, we can improve the overall detection rate of the system without compromising the benefits of either detection method.

25.2.2 Feature Extraction for IDS

Two basic premises of intrusion detection are that system activities are observable, e.g., via auditing, and there is distinct evidence that can distinguish normal and intrusive activities. We call the evidence extracted from raw audit data *features*, and use these features for building and evaluating intrusion detection models. Feature extraction (or construction) is the process of determining what evidence that can be taken from raw audit data is most useful for analysis. Feature extraction is thus a critical step in building an IDS. That is, having a set of features whose values in normal audit records differ significantly from the values in intrusion records is essential for having good detection performance.

We have developed a set of data mining algorithms for selecting and constructing the features from audit data. First the raw (binary) audit data is processed and summarized into discrete records containing a number of basic features such as in the case of network traffic: timestamp, duration, source and destination IP addresses and ports, and error condition flags. Specialized data mining programs are then applied to these records to compute the frequent patterns describing correlations among the features and frequently co-occurring events across many records. A pattern is typically from $A, B \rightarrow C, D[\textit{confidence, support}]$, which translates to events A and B are followed by events C and D with a certain confidence and occur with a certain frequency in the data (the patterns' support). The consistent patterns are of normal activities and the "unique" patterns associated with an intrusion are

then identified and analyzed to construct additional features for connection records. It can be shown that the constructed features can indeed clearly separate intrusion from normal ones. Using this approach, the constructed features are more grounded on empirical data, and thus more objective than an expert knowledge. Results from the 1998 DARPA Intrusion Detection Evaluation showed that an IDS model constructed using these algorithms was one of the best performing of all the participating systems.

As an example, let us consider the SYN flood attack. When launching this attack, an attacker uses many spoofed source addresses to open many connections, which never become completely established (i.e., only the first SYN packet is sent, and connection remains in the "S0" state) to some port on a victim host (e.g., *http*). We compared the patterns from the 1998 DARPA data set that contain SYN flood attacks with the patterns from a "baseline" normal data set (of the same network), by first encoding the patterns into numbers and then computing "difference" scores. The following pattern, a frequent episode, has the highest "intrusion-only" (i.e., unique for the intrusion) score:

"(flag = S0, service = http, dst_host = victim), (flag = S0, service = http, dst_host = victim) → (flag = S0, service = http, dst_host = victim) [0.93, 0.03, 2]." This means that 93% of the time, after two http connections with S0 flag are made to host victim, within 2 seconds from the first of these two, the third similar connection is made; and this pattern occurs in 3% of the data. Accordingly, the feature construction algorithm parses the pattern features: "a count of connections to the same dst_host in the past 2 seconds," and among these connections, "the percentage of those that have the same service, and the percentage of those that have the S0 flag." For the two "percentage" features, the normal connection records have values close to 0, but the connection records belonging to syn_flood have values above 80%. Once these discriminative features are constructed, it is easy to generate the detection rules via either manual (i.e., hand-coding) or automated (i.e., machine learning) techniques. For example, we use RIPPER, W.W. Cohen et al., in 1995, an inductive rule learner, to compute a detection rule for syn_flood using these extracted features: if for the past 2 seconds, the count of connections to the same dst_host is greater than 4; and the percentage of those that have the same service is greater than 75%; and the percentage of those that have the "S0" flag is greater than 75%, then there is a syn_flood attack.

25.2.3 Artificial Anomaly Generation

A major difficulty in using machine learning methods for anomaly detection lies in making the learner discover boundaries between known and unknown classes. Since there are no examples of anomalies in the training data (by definition of anomaly), a machine-learning algorithm will only uncover boundaries that separate different known classes in training data. A machine-learning algorithm will not specify a boundary between the known data and unseen data (anomalies). We present the technique of *artificial anomaly generation*

to enable the traditional learners to detect anomalies. Artificial anomalies are injected into the training data to help the learner discover a boundary around the original data. All artificial anomalies are given the class label *anomaly*. The approach to generating artificial anomalies focuses on "near misses," instances that are close to the known data, but are not in the training data. We assume the training data are representative, hence near misses can be safely assumed to be anomalous.

Since we do not know where the exact decision boundary is between the known and anomalous instances, we assume that the boundary may be very close to the existing data. To generate the artificial anomalies close to the known data, a useful heuristic is to randomly change the value of one feature of an example to a value that does not occur in the data while leaving the other features unaltered.

Some regions of known data in the instance space may be sparsely populated. We can think of the sparse regions as small islands and dense regions as large islands in an ocean. To avoid overfitting, learning algorithms are usually biased toward discovering more general models. Since we only have known data, we want to prevent models from being overly general when predicting these known classes. That is, we want to avoid the situation where sparse regions may be grouped into dense regions to produce singularly large regions covered by overly general models. It is possible to produce artificial anomalies around the edges of these sparse regions and coerce the learning algorithm to discover the specific boundaries that distinguish the regions from the rest of the instance space. In other words, we want to generate data that will amplify these sparse regions.

Sparse regions are characterized by infrequent values of individual features. To amplify sparse regions, we proportionally generate more artificial anomalies around sparse regions depending on their sparsity using *Distribution Based Artificial Anomaly* algorithm (DBA2) in W. Fan, PhD thesis, 2001. The algorithm uses the frequency distribution of each feature's values to proportionally generate a sufficient amount of anomalies.

25.2.4 Combined Misuse and Anomaly Detection

Traditionally, anomaly detection and misuse detection are considered separate problems. Anomaly detection algorithms typically train over normal data while misuse algorithms typically train over labeled normal and intrusion data. Intuitively a hybrid approach should perform better; in addition, it has the obvious efficiency advantages in both model training and deployment than using two different models. We use the artificial anomaly generation method to create a single model that is both a misuse and anomaly detection method. This allows us to use traditional supervised inductive learning methods for both anomaly detection and misuse detection at the same time. We train a single model from a set of data that contains both normal records and records corresponding to intrusions. In addition, we also generate artificial anomalies

using the DBA2 algorithm and train the algorithm over the combined data set. The learned model can detect anomalies and intrusions concurrently.

We learn a single rule set for combined misuse and anomaly detection. The rule set has rules to classify a connection to be normal, one of the known intrusion classes, or anomaly. In order to evaluate this combined approach, we group intrusions together into a number of small clusters. We create multiple data sets by incrementally adding each cluster into data set and regenerating artificial anomalies. This is to simulate the real-world process of developing and discovering new intrusions and incorporating them into the training set. We learn models that contain misuse rules for the intrusions that are known in the training data, anomaly detection rules for unknown intrusions in left-out clusters, and rules that characterize normal behavior.

We have seen that the detection rates of known intrusions classified by misuse rules in models learned with and without artificial anomalies are indistinguishable or completely identical. This observation shows that the proposed DBA2 method does not diminish the effectiveness of misuse rules in detecting particular categories of known intrusions.

25.2.5 Efficiency

In typical applications of data mining to intrusion detection, detection models are produced off-line because the learning algorithms must process tremendous amounts of archived audit data. These models can naturally be used for off-line intrusion detection (i.e., analyzing audit data off-line after intrusion detection should happen in real time, as intrusions take place, to minimize security compromises). In this section, we discuss the approaches to make data mining-based ID models work efficiently for real-time intrusion detection.

In contrast to off-line IDSs, a key objective of real-time IDS is to detect intrusions as early as possible. Therefore, the efficiency of the detection model is a very important consideration. Because the data mining-based models are computed using off-line data, they implicitly assume that when an event is being inspected (i.e., classified using an ID model), all activities related to the event have completed so that all features have meaningful values available for model checking. As a consequence, if we use these models in real time without any modification, then an event is not inspected until complete information about that event has arrived and been summarized, and all temporal and statistical features are computed. This scheme can fail miserably under real-time constraints. When the volume of an event stream is high, the amount of time taken to process the event records within the past n seconds and calculate statistical features is also very high. Many subsequent events may have terminated (and thus completed with attacks actions) when the "current" event is finally inspected by model. That is, the detection of intrusions is severely delayed. Unfortunately, DoS attacks, which typically generate a large amount of traffic in a very short period time, are often used by intruders to first overload

an IDS and use the detection delay as a window of opportunity to quickly perform their malicious intent. For example, they can even seize control of the host on which the IDS lives, thus eliminating the effectiveness of intrusion detection altogether.

It is necessary to examine the time delay associated with computing each feature in order to speed up model evaluation. The time delay of a feature includes not only the time spent for its computation, but also the time spent waiting for its readiness (i.e., when it can be computed). For example, in the case of network auditing, *the total duration* of a network connection can only be computed after the last packet of the connection has arrived, whereas the *destination host* of a connection can be obtained by checking the header of the first packet.

From the perspective of cost analysis, the efficiency of an intrusion detection model is its *computational cost,* which is the sum of the time delay of the features used in the model. Based on the feature construction approaches discussed in previous section, we can categorize features used for network intrusion detection into 4 cost levels:

- Level 1 features can be computed from this packet, e.g., the *service*.
- Level 2 features can be computed at any point during the life of the connection state (*SYN_WAIT*, *CONNECTED*, *FIN_WAIT*, etc.).
- Level 3 features can be computed at the end of the connection, using only the information about the connection to be examined, e.g., the total number of bytes sent *from source to the 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 temporal and statistical features and are the most costly to compute.

In order to conveniently estimate the cost of a rule, we assign a cost of 1 to the level 1 features, 5 to the level 2 features, 10 to level 3, and 100 to level 4. These cost assignments are very close to the actual measurements we have obtained via extensive real-time experiments. However, we have found that the cost of computing the level 4 features is linearly dependent on the amount of connections being monitored by the IDS within the time window used for computation, as they require iteration of the complete set of recent connections.

In this section, we discuss approaches to reduce computational cost and improve the efficiency of the real-time intrusion detection models. Next, we describe the techniques for *cost-sensitive modeling* and real-time system for implementing *distributed feature computation* for evaluating multiple cost-sensitive models.

25.2.6 Cost-Sensitive Modeling

In order to reduce the computational cost of IDS, detection rules need to use low-cost features as often as possible while maintaining a desired accuracy

level. We propose a multiple rule set approach in which each rule set uses the features from different cost levels. Low-cost rules are always evaluated first by IDS, and high-cost rules are used only when low-cost rules cannot predict with sufficient accuracy.

In the domain of network intrusion detection, we use four different levels of costs to compute features, as discussed in the pervious method. Features of costs 1, 5, and 10 are computed individually and features of cost 100 can be computed in a single lookup of all the connections in the past n seconds. With the above costs and goals in mind, we use the following multiple rule set approach:

- We first generate multiple training sets T_{1-4} using different feature subsets. T_1 uses only cost 1 features. T_2 uses the features of costs 1 and 5, and so forth, upto T_4, which uses all available features.
- Rule sets R_{1-4} are learned using their respective training sets. R_4 is learned as an ordered rule set [An ordered rule set is in the form of "**If** condition$_1$ **then** action$_1$ **else if** condition$_2$ **then** action$_2$....**else** action$_n$". The rules are checked sequentially. We typically place rule for the most prevalent class, i.e., *normal,* as the first rule.] for its efficiency, as it may contain the mostly costly features. R_{1-3} are learned as unordered rule sets [An unordered rule set is in the form of "**If** condition$_1$ **then** action$_1$; **if** condition$_2$ **then** action$_2$;.....**If** condition$_n$ **then** action$_n$"the rules can be checked in parallel], as they will contain accurate rules for classifying *normal* connections.
- A precision measurement p_r [Precision describes how accurate a prediction is. Precision is defined as $p = |P \cap W||P|$, where P is the set of predictions with label i, and W is the set of all instances with label i in the data set] is computed for *every rule r*, expect for rules in R_4.
- A threshold value τ_I is obtained for every single class, which determines the tolerable precision required in order for a classification to be made by any rule set except for R_4.

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

- All cost 1 features used in R_1 are computed for the connection being examined. R_1 is then evaluated and prediction time i is made.
- If $p_r > \tau_{I,}$ the prediction i is fired. In this case, no more features are computed and the system examines the next connection. Otherwise, additional features required by R_2 are computed and R_2 is evaluated in the same manner as R_1.
- Evaluation continues with R_3, followed by R_4, until the prediction is made.
- When R_4 (an ordered rule set) is reached, features are computed as needed while evaluation proceeds from the top of the rule set to the bottom. The evaluation of R_4 does not require any firing condition and will always generate a prediction.

In the experiments, they used data from the 1998 DARPA evaluation. The detailed experimental set up and results can be found in W. Lee et al., in 2000. In summary, the multiple model approach can reduce the computational cost by as much as 97% without compromising predictive accuracy, where the cost for inspecting a connection is the total computational cost of all unique features used before a prediction is made. If multiple features of cost 100 are used, the cost is counted only once since they can all be calculated in a single iteration through the table of recent connections.

25.2.7 Distributed Feature Computation

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 from raw network traffic data to produce connection records, and then offloads model evaluation and higher level feature computation to a separate entity, called JUDGE. The motivation for offloading this computation and evaluation is that it is quite costly and we do not wish to overburden the sensor (in this case a packet-sniffing engine).

JUDGE uses models that have been learned using the techniques described previously. That is, there exists a sequence of models, each of which uses increasingly more costly features than the previous model. Models are evaluated and higher level features are computed at different points in a connection by JUDGE as more primitive features become available.

The sensor informs JUDGE of new feature values, or updates to feature values that are maintained throughout the connection's life, whenever there is a change in the connection's state (e.g., a connection has gone from SYN_WAIT to CONNECTED). Sensors also update certain feature values whenever there is an "exception" event. Exceptions are certain 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 computes the set of features that are available for the given connection. If the set of features is a proper subset of set of light-weighted features, (the level 1 and 2 features described earlier) used by one of the ID models, then higher level features are computed and that model is evaluated. The logic for determining when a prediction is made is the same as described before.

Once a prediction is made by JUDGE a complete connection record, with the label, is inserted into a data warehouse as described in the system architecture outlined previously. We have currently implemented this system using NFR's Network Flight Recorder as the sensor, although the protocol for communication between the sensor and JUDGE would allow any sensor, which extracts features from a data stream to be used.

Usability

A data mining-based IDS is significantly more complex than a traditional system. The main cause for this is that data mining systems require large sets of data from which to train. The hope to reduce the complexity of data mining systems has led to many active research areas.

First, management of both training and historical data sets is a difficult task, especially if the system handles many different kinds of data. Second, once new data has been analyzed, models need to be updated. It is impractical to update models by retraining over all available data, as retraining can take weeks, or even months, and updated models are required immediately to ensure the protection of the systems. Some mechanism is needed to adapt a model to incorporate new information. Third, many data mining-based IDSs are difficult to deploy because they need a large set of clean (i.e., not noisy) labeled training data. Typically the attacks within the data must either be manually labeled for training signature detection models, or removed for training anomaly detection models. Manually cleaning training data is expensive, especially in the context of large networks. In order to reduce the cost of deploying a system, we must be able to minimize the amount of clean data that is required by the data mining process.

We present an approach to each of these problems. We use the technique *adaptive learning,* which is a generic mechanism for adding new information to a model with out retraining. We employ *unsupervised anomaly detection*, which is a new class of intrusion detection algorithms that do not rely on labeled data. In the next section, we present a system architecture, which automates model and data management.

Adaptive Learning

We propose to use ensembles of classification models ($R_{1,\ldots,}R_4$ described earlier is an example of an ensemble of classification models) as a general and algorithm-independent method to adapt the existing models in order to detect newly established patterns. The goal is to improve the efficiency of both learning and deployment. In reality, when a new type of intrusion is discovered, it is very desirable to be able to quickly adjust an existing detection system to detect the new attack, even if the adjustment is temporary and may not detect the new attack.

At the same time, after we have at least some method of defense, we can look for possibly better ways to detect the attack, which involves recomputing the detection model and may take much longer period of time to compute. When a better model is computed, we may choose to replace the temporary model. For such purposes, we seek a "plug-in" method, i.e., we efficiently generate a simple model that is only good at detecting the new intrusion, and plug or attach it to the existing models to enable detection of new intrusions. Essentially, we efficiently generate a lightweight classifier (i.e., classification

- **If** (H_1 (x) = *normal*) V (H_1 (x) = *anomaly*) **then**
  - **if** H_2 (x) = *normal*
    then *output* ← H_1 (x) (*normal or anomaly*)
  - **else** *output* ← *new_intrusion*
- **else** *output* ← H_1**(x)**

Fig. 25.1. Ensemble-based Adaptive Learning Configuration

model) for the new pattern. The existing main detection model remains the same. When the old model detects an anomaly, this data record is sent to the new classifier for further classification. The final prediction is the function of both the old classifier and the new classifier. Computing the new classifier is significantly faster than generating a monolithic model for all established patterns and anomalies.

In one such configuration, given an existing classifier H_1, an additional classifier, H_2, is trained from data containing normal records and records corresponding to the new intrusion. We refer to H_1 as the existing IDS model, while H_2 refers to a new model trained specifically for a new or recently discovered attack. The decision rules in Fig. 25.1 are evaluated to compute the final outcome. This method is independent of the actual model building algorithm. Each classifier can be anything from a decision tree, a rule-based learner, to a neural network, etc.

We have experimented with different configurations to test the effectiveness of this approach. The cost of training of the proposed method (as measured in cost-sensitive modeling) is almost 150 times less expensive than learning a monolithic classifier trained from all available data, and the accuracy of both essentially equivalent.

Unsupervised Learning

Traditional model building algorithms typically require a large amount of labeled data in order to create effective detection models. One major difficulty in deploying a data mining-based IDS is the need for labeling system audit data for the use by these algorithms. For misuse detection systems, the data needs to be accurately labeled as either normal or attack. For anomaly detection system, the data must be verified to ensure it is completely normal, which requires the same effort. Since models (and data) are specific to the environment on which the training data was gathered, this cost of labeling the data must be incurred for each deployment of the system.

Ideally, we would like to build detection models from the collected data without needing to manually label it. In this case, the deployment cost would greatly be decreased because the data would not need to be labeled. In order to build these detection models, we need a new class of model building algorithm. These model-building algorithms can take as input unlabeled data and create

a detection model. We call these algorithms unsupervised anomaly detection algorithms.

In this section, we present the problem of unsupervised anomaly detection and relate it to the problem of outlier detection in statistics. We present an overview of two unsupervised anomaly detection algorithms that have been applied to intrusion detection.

These algorithms can be also referred to as anomaly detection over noisy data. The reason the algorithm must be able to handle noise in the data is that we do not want to manually verify that the audit data collected is absolutely clean (i.e., contains no intrusions).

Unsupervised anomaly detection algorithms are motivated by two major assumptions about the data, which are reasonable for intrusion detection. The first assumption is that anomalies are very rare. This corresponds to the fact that normal use of the system greatly outnumbers the occurrence of intrusions. This means that the attacks compose relatively small proportion of the total data. The second assumption is that the anomalies are quantitatively different from the normal elements. In intrusion detection this corresponds to the fact that attacks are drastically different from the normal usage.

Since anomalies are very rare and quantitatively different from the normal data, they stand out as outliers in the data set. Thus, we can cast the problem of detecting the attacks into an outlier detection problem. Outlier detection is the focus of much literature in the field of statistics.

In intrusion detection, intuitively, if the ratios of attacks are different to normal data is small enough, then because the attacks are different, the attacks stand out against the background of normal data. We can thus detect the attack within the data set.

We have performed experiments with two types of unsupervised anomaly detection algorithms, each for a different type of data. We applied probabilistic-based unsupervised anomaly detection algorithm to build detection models over the system calls and a clustering-based unsupervised anomaly detection algorithm for network traffic.

The probabilistic approaches to detect outliers are by estimating the likelihood of each element in the data. We partition the data into two sets, normal elements and anomalous elements. Using a probability-modeling algorithm over the data, we compute the most likely partition of the data.

The clustering approach detects the outliers by clustering the data. The intuition is that the normal data will cluster together because there is a lot of it. Because anomalous data and normal data are very different from each other, they do not cluster together. Since there is very little anomalous data relative normal data, after clustering, the anomalous data will be in the small clusters. The algorithm first clusters the data and then labels the smallest clusters as anomalies.

25.2.8 System Architecture

The overall system architecture is designed to support a data mining-based IDS with the properties described throughout this section. As shown in Fig. 25.2, the architecture consists of sensors, detectors, a data warehouse, and a model generation component. This architecture is capable of supporting data gathering, sharing, and analysis, and also data archiving and models generation and distribution.

The system is designed to be independent of the sensor data format and model representation. A piece of sensor data can contain an arbitrary number of features. Each feature can be continuous or discrete, numerical, or symbolic. In this framework, a model can be anything from a neural network, to a set of rules, to a probabilistic model. To deal with this heterogeneity, an XML encoding is used so each component can easily exchange data and/or models.

The design was influenced by the work in standardizing the message formats and protocols for IDS communication and collaboration: the Common Intrusion Detection Framework (CIDF, funded by DARPA) and the more recent Intrusion Detection Message Exchange Format (IDMEF by the Intrusion Detection Working Group of IETF, the Internet Engineering Task Force). Using CIDF or IDMEF, IDSs can securely exchange attack information, encoded in the standard formats, to collaboratively detect distributed intrusions. In the architecture, data and model exchanged between the components are

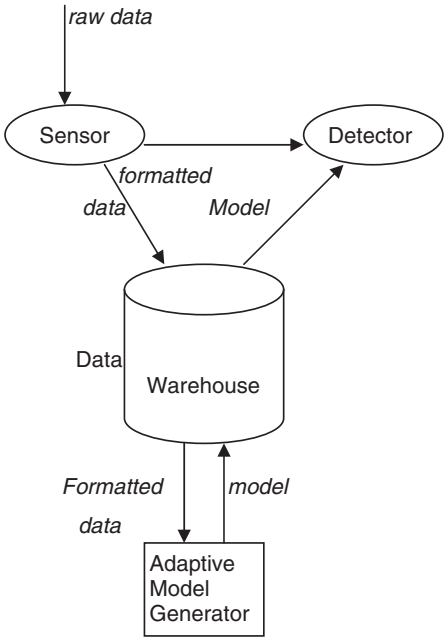

Fig. 25.2. The Architecture of Data Mining-based IDS

encoded in our standard message format, which can be trivially mapped to either CIDF or IDMEF formats. The key advantage of the architecture is its high performance and scalability. That is, all components can reside in the same local network, in which case, the work load is distributed among the components; or the components can be in different networks, in which case, they can also participate in the collaboration with other IDSs in the Internet. In the following sections we describe the components depicted in Fig. 25.2 in more detail.

Sensor

Sensors observe raw data on a monitored system and compute feature for use in model evaluation. Sensors insulate the rest of the IDS from the specific low-level properties of the target system being monitored. This is done by having all of the sensors implement a Basic Auditing Module (BAM) framework. In a BAM, features are computed from the raw data and encoded in XML.

Detectors

Detectors take processed data from sensors and use a detection model to evaluate the data and determine if it is an attack. The detectors also send back the result to the data warehouse for further analysis and report.

There can be several (or multiple layers of) detectors monitoring the same system. For example, work loads can be distributed to different detectors to analyze events in parallel. There can also be a "back-end" detector, which employs very sophisticated models for correlation or trend analysis, and several "front-end" detectors that perform quick and simple intrusion detection. The front-end detectors keep up with high-speed and high-volume traffic and must pass data to the back-end detector to perform more thorough and time-consuming analysis.

Data warehouse

The data warehouse serves as a centralized storage for data and models. One advantage of a centralized repository for the data is that different components can manipulate the same piece of data asynchronously with the existence of a data base, such as off-line training and manually labeling. The same type of components, such as multiple sensors, can manipulate data concurrently. Relational database features support "stored procedure calls," which enable easy implementation of complicated calculation, such as efficient data sampling carried out automatically on the server.

Arbitrary amount of sensor data can also be retrieved by a single SQL query. Distribution of detection models can be configured to push or pull.

The data warehouse also facilitates the integration of data from multiple sensors. By correlating data/results from different IDSs or data collected over a longer period of time, the detection of complicated and large scale attacks becomes possible.

Model Generator

The main purpose of the model generator is to facilitate the rapid development and distribution of new intrusion detection models. In this architecture, an attack detected first as an anomaly may have its exemplary data processed by the model generator, which in turn, using the archived normal and intrusion data sets from the data warehouse, automatically generates a model that can detect the new intrusion and distributes it to the detectors (or any other IDSs that may use these models). Especially useful are unsupervised anomaly detection algorithms because they can operate on unlabeled data that can be directly collected by the sensors.

We have successfully completed a prototype implementation of a data mining and CIDF-based IDS. In this system, a data mining engine equipped with feature extraction programs and machine learning programs serves as the model generator for several detectors. It receives audit data for anomalous events from a detector, computes patterns from the data, compares them with historical normal patterns to identify the "unique" intrusion patterns, and constructs features accordingly. Machine learning algorithms are then applied to compute the detection model, which is encoded as GIDO and sent to all the detectors. Much of the design and implementation efforts had been on extending the common intrusion detection models (CISL). In preliminary experiments the generator is able to produce and distribute new effective models upon receiving audit data.

Related Work

The research encompasses many areas of intrusion detection, data mining, and machine learning. In this section, we briefly compare our approaches with related efforts.

In terms of feature construction for detection models, DC-1 (Detector Constructor, T. Fawcett et al., 1997) first invokes a sequence of operation for constructing features (indicators) before constructing a cellular phone fraud detector (a classifier). We are faced with a more difficult problem here because there is no standard record format for connection or session records (we had to invent our own). We also need to construct temporal and statistical features not just for individual records, but also different connections and services. That is, we are modeling different logical entities that take on different roles and whose behavior is recorded in great detail. Extracting these from a vast and overwhelming stream of data adds considerable complexity to the problem.

The work most similar to unsupervised model generation is a technique developed at SRI in the Emerald system. Emerald uses the historical records to build normal detection models and compares the distributions of new instances to historical distributions. Discrepancies between the distributions signify an intrusion. One problem with this approach is that intrusions present in the historical distributions may cause the system to not detect similar intrusions in unseen data.

Related to automatic model generation is adaptive intrusion detection. Teng et al. (1990) perform adaptive real-time anomaly detection using inductively generated sequential patterns. Also relevant is Sobirey's work on adaptive intrusion detection using an expert system to collect data from audit sources.

Many different approaches to building anomaly detection models have been proposed. Stephanie Forrest presents an approach for modeling normal sequences using look-ahead pairs and contiguous sequences. Helman and Bhangoo present a statistical method to determine sequences, which occur more frequently in intrusion data as opposed normal data. Lee et al. use a prediction model trained by a decision tree applied over the normal data. Ghosh and Schwarzbard use the neural networks to model normal data. Lane and Brodley examine unlabeled data for anomaly detection by looking at user profiles and comparing the activity during an intrusion to the activity under normal use.

Cost-sensitive modeling is an active research area in data mining and machine learning communities because of the demand from application domains such as medical diagnosis and fraud and intrusion detection. Several techniques have been proposed for building 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. In intrusion data representation, related work is the IETF Intrusion Detection Exchange Format project and CIDF effort.

25.3 Summary

Although only a few companies currently have data mining systems checking their audit records, the number is expected to dramatically increase in the near future as companies desperately try to ensure that their computer systems are secure from intrusion.

In this section, we have outlined the breadth of the research efforts to address important and challenging issues of accuracy, efficiency, and usability of real-time IDSs.

We have implemented feature extraction and construction algorithms for labeled audit data (i.e., when both normal and intrusion data sets are given). We are implementing algorithms for unlabeled data (which can be purely normal or possibly containing unknown intrusions).

We have developed several anomaly detection algorithms. In particular, we have completed the implementation of and extensive experimentation with "artificial anomaly generation" approaches. We are exploring the use of information-theoretic measures, i.e., entropy, conditional entropy, relative entropy, information gain, and information cost to capture intrinsic characteristics of normal data and use such measures to guide the process of building and evaluating anomaly detection models. We are also developing efficient approaches that use statistics on packet header values for network anomaly detection.

We studied the computational costs of features and models and have implemented a multiple model-based approach for building models that incurs minimal computational cost while maintaining accuracy. We have also developed a real-time system, "Judge," for evaluating models learned using this method.

We are developing adaptive learning algorithms to facilitate model construction and incremental updates. We are also developing unsupervised anomaly detection algorithms to reduce the reliance on labeled training data. We have completed the design and specification of our system architecture with sensor, detector, data warehouse, and modeler components. A prototype system has been implemented and we will continue to build on and experiment with this system.

We are developing algorithms for data mining over the output of multiple sensors. This is strongly motivated by the fact that single sensors do not typically observe entire attack scenarios. By combining the information from multiple sensors we hope to improve detection accuracy.

The ultimate goal of the research is to not only to demonstrate the advantages of our approaches but also to provide useful architectures, algorithms, and tool sets to the community to build better IDSs in less time and with greater ease. Toward this end, we are integrating the feature construction and unsupervised anomaly detection algorithms into the model generator, and building detectors that are equipped with misuse and anomaly detection algorithms. We are deploying the prototype IDS on real-world networks in order to improve the techniques.

A serious limitation of the current approaches (as well as with most existing IDSs) is that we only do intrusion detection at the network or system level. However, with the advent and rapid growth of e-commerce (or e-business) and e-government (or digital government) applications, there is an urgent need to do intrusion and fraud detection at the application level. This is because many attacks may focus on applications that have number effect on the underlying network or system activities. We have previously successfully developed data mining approaches for credit card fraud detection. We plan to start research efforts on IDSs for e-commerce and e-government applications in the near future. We anticipate that we will be able to extend our current approaches to develop application-level IDSs because the system architecture and many of our data mining algorithms are generic

(i.e., data format independent). For example, we can develop (and deploy) a sensor for a specific application, and extend the correlation algorithms, with application domain knowledge, in the detectors to combine evidence from the application and the underlying system in order to detect intrusion and frauds. The relevant reports detailing the results described in this case study can be found at http://www.csc.ncsu.edu/faculty/lee/project/id.html, http://www.cs.columbia.edu/ids, and http://www.cs.fit.edu/∼pkc/id.

Thus this section has given an overview on data mining in security and real-time data mining-based intrusion detection system case study.

Review Questions

1. How is data mining used in security systems?
2. With a case study explain data mining intrusion detection systems (IDSs).
3. Draw the architecture of data mining based IDS and explain.