Data Mining-based Intrusion Detectors:

An Overview of the Columbia IDS Project

Salvatore J. Stolfo*, Wenke Lee**, Philip K. Chan***, Wei Fan* and Eleazar Eskin*

*Columbia University

**Georgia Institute of Technology

***Florida Institute of Technology

September 9, 2001

Introduction

The field of Intrusion Detection has been an active area of research for some time. The goal of an Intrusion Detection System (IDS) is to provide another layer of defense against malicious (or otherwise unauthorized) uses of computer systems by sensing a misuse or a breach of a security policy and alerting operators to an ongoing (or, at least, recent) attack. (We make no distinction between external or internal attacks.) The field has made substantial strides since the earliest papers that have been published on the topic (eg., Denning's 1987 paper[1]). And today a growing market for IDSs is evidenced by the number of new companies offering various IDS products. (The IDS marketplace, however, is presently immature; many opportunities exist for new products.)

Even though the fields of security and cryptography have successfully produced a wide range of technologies that are now broadly used to secure computer systems, accounts in the news media appear almost on a weekly basis reporting that malicious users still succeed in attacking systems with sometimes devastating losses (in vital data, or services). Many fear that such losses often go undetected. (Attackers often announce their successes before anyone has noticed their deeds.) The reasons such attacks remain commonplace are due to lax security policies or procedures, error in configuring security systems, and, perhaps worse, insider attacks which obviously can operate behind security walls designed to protect systems from outsider attacks.

IDS systems have now taken on a key role in adding another layer of protection to our network systems and keeping administrators and operators informed when some misuse is or has taken place. This is good news. We can expect that the level of security and awareness of attempted intrusions will continue to be raised and ultimately only the most knowledgeable and well funded groups (perhaps nation states) will exist as the primary threat that organizations may have to consider. Generally speaking, most well run network systems can now be protected from the "casual" *script kiddie* attacks that were popularized, for example, in 2000 by the distributed denial of service (DDOS) attacks levied against popular websites.

IDSs typically operate within a managed network between a firewall and internal network elements. Other IDSs operate at the host level. In both cases, a stream of data (network packets in the case of network-based IDS) is inspected and "rules" are applied in order to determine whether some attack is underway. In the case of host-based intrusion detection, audit data of system processes (often system call data) are similarly analyzed to determine whether some condition of the system indicates a breach of security policy, or errant behavior of some process indicating an attack. In either case, the "models" used to determine the nature of an event are acquired by human experience or shared between people via security alert services, and are coded by humans in a rule-based fashion for deployment in an IDS detector. Much of the prior research in intrusion detection has focused on the languages in which humans express models of attacks, as well as the infrastructure to efficiently implement these rules.

We distinguish between IDSs that aim to detect attacks based upon a static state description or properties of some action or system, versus those that detect attacks based upon the change of system state over time. Virus detectors are examples of the former. Most commercial virus scanners find viruses by matching known "signature strings" embedded in the malicious payload of the virus, and that are distinct, and uniquely identifying over other known viruses. This technique works well for quickly detecting known viruses, but provides no protection against new viruses.

The earliest IDSs were developed with string matching rules looking for command sequences used by known attacks. These string-matching approaches have limited use and are easy to foil. Other attacks focus on communication protocols

[1] D.E. Denning, "An intrusion detection model", IEEE Transactions on Software Engineering", Vol. SE-13, No. 2, pp. 222--232, 1987.

(TCP/IP, for example) and seek to exploit vulnerabilities in specific protocol implementations. These more sophisticated attacks are dynamic in nature. Some current IDSs (Grids, STAT, etc.) that are based upon dynamic state changes provide a means for programmers to specify system state and changes of state as a means to detect illegal transitions indicative of known attacks.

This approach, however, relies upon human expertise, which is the current state of practice, and is a manual and error-prone development process involving the codification of incomplete "expert knowledge." This process is *reactive*. Models of an attack are hand-coded after an attack has been discovered and analyzed. Thus, systems are vulnerable to new attacks until a means of detecting and preventing them has been developed and widely deployed. IDSs are expensive and slow to develop. They are difficult to optimize according to local cost-benefit parameters, and ineffective in detecting novel attacks. New technologies are therefore needed to substantially improve the ability to generate detectors rapidly, with improved detection capabilities covering known attacks well and new attacks sufficiently well to provide better security at optimal cost.

Cost-Sensitive Data Mining-Based IDSs

A team of researchers at Columbia University conceived of a new approach to intrusion detection based upon data mining of audit sources. Rather than hand coding IDS rules, a much more effective approach is to provide to IDS designers automated tools and techniques to analyze (raw) audit data, extract evidence (features) of an attack and that then automatically compute rules that detect attacks. The human expert would only be required to "label" the audit data to identity the data pertaining to the "misuse". The task of labeling audit data can be partially automated by simulators, or by running an (newly discovered or known) exploit in a "sand box" laboratory environment separated from online operational systems.

Under DARPA support, the Columbia team developed a novel system for rapid development and deployment of effective and cost-sensitive IDSs. Our system automates *feature construction,* the critical step in building effective misuse and anomaly detection models, by analyzing the patterns of normal and intrusion activities computed from audit data. Detection models are constructed automatically using *cost-sensitive* machine learning algorithms to achieve optimal performance using given (and often site-specific) cost metrics. The system finds the clusters of attack signatures and normal profiles and accordingly constructs one *light-weight model* for each cluster to maximize the utility of each model. A dynamically configurable group of such light-weight models can be very effective and efficient, and resilient to attacks launched against the IDS itself. The detection models can be rapidly deployed through automatic conversion to *efficient real-time modules* of fielded IDSs.

The approach of "cost sensitive" data mining in particular was proposed to DARPA and funded. The notion of cost-sensitivity is new to the field of IDS and conceived by researchers at Columbia. For years, researchers in IDS have considered any misuse of a system equally as bad as any other. However, different attacks incur different costs for different victims, and the Columbia team has created a framework for modeling these costs. For example, some attacks may be annoying but incur no damage. Others may disable an entire network of mission-critical computers. Clearly, these two classes of attacks have varying *damage costs*. Likewise, some attacks may reveal themselves in a trivial fashion, and hence are very easy to detect. Such attacks may be revealed by simply inspecting header information in network packets, and thus are cheap to detect. Other attacks may be conducted over long periods of time and require an expensive archive of data held in memory for successful detection. In such cases, different attacks have different *operational costs.*

What is particularly important in this context is that from site to site and network to network, each facility must determine the relative costs and inherent value of the elements and services they wish to protect. Hence, a "one size fits all" strategy to field effective IDSs is simply not warranted. The Columbia team built a facility that generates models of attacks based upon site-specific cost models. Hence, each IDS installed at a site is cost-optimized to that site, automatically.

Over the course of several years of research and development, the Columbia team has built a data mining facility applied to intrusion detection that has demonstrated the utility of applying automated tools to

- gather data (observations of system behavior provided for example by commercially available products as well as systems developed at Columbia) in an RDBMs data warehouse,
- analyze data to identify useful (measurable) features to form the basis of a sensor,
- define the costs associated with the observable features and the services and systems being protected,
- automatically compute cost-effective models of known attacks and normal system behavior to build an effective detector for both known misuses and anomalous events, and
- deploy those models into a functioning detector (via a compiler).

This line of work has led to new developments that extend intrusion detection into another generation of systems which can better protect our critical infrastructure of complex computer networks.

Core Sensors for Audit Data

It is useful to think of the behavior of some system or object as the "Volume" and "Velocity" of certain primitive operations and their contents, performed by and with the objects of the system. To complete this metaphor, the hard part is determining what the objects and operations are, and what can be efficiently and accurately observed of those objects and their behaviors over time.

A data mining approach provides an environment in which this activity can be greatly improved and partially automated to develop IDSs in a more cost-effective manner. It also provides a huge advantage in developing a new class of models to detect new attacks before they have been discovered by human experts. This is the second important contribution of this work.

The core technologies developed at Columbia can be applied to any environment in which audit data can be gathered from some host or network using any suitable commercially available or open source system. For example, Solaris Unix (an OS from Sun Microsystems) includes BSM, a Basic Security Module that logs system process execution data, which may be one source of data that may be gathered. One can also utilize Network Flight Recorder, a commercial network monitoring system that includes a network packet sniffing engine with a high-level language for interpreting network packet data. In some cases we have built our own auditing facilities. We developed a set of "BAM's", Basic Auditing Modules, that provide the same utility for Windows platforms as Sun's BSM. Thus, our technology allows multiple platforms and alternative data sources to be conveniently gathered for subsequent analysis, all time-correlated, to provide alternative views of the operation of a complex network system.

One may also wish to inspect the behavior of a system's file subsystem. Toward this end, we developed a suite of " wrappers" for files and system processes which inspect and trace all accesses to and from a file or process. This too can be a useful source of audit data to model at the host level.

It is important that all of the distributed elements in a network are able to communicate and exchange data, especially with a centralized archive. We have thus developed an XML-based interchange language for all IDS elements to communicate. DARPA sponsored work to develop CIDF, the Common Intrusion Detection Framework, which has since been subsumed by work of the Internet Engineering Task Force. The IETF has recently proposed such a standard based upon XML, the Intrusion Detection Message Exchange Format (IDMEF). Our particular implementation conforms to the IDMEF standard and allows any IDS to be "plugged in" to our infrastructure with relative ease.

Feature Construction for Cost-sensitive Misuse Detection

Using raw system-level data (from either a host or a network), one can compute "features" that are the subject matter of models to detect various events that indicate attacks. (For example, the *frequency of service requests* by some source IP address is a *statistical feature* that may be extracted from network packet audit data. If the number of such requests made per second is very high, that event may indicate that a denial of service attack is underway.)

One component of the work at Columbia performed as part of a Ph.D. thesis was the development of data mining algorithms that propose useful feature sets for modeling attacks. The "MADAM/ID" system, which has now been widely distributed via the Internet, consists of two sets of data mining facilities. Raw audit data is analyzed by a variant of the well known

association rules algorithm in order to determine static features of attack data that do not appear in otherwise normal data. Sequential data (for example, network events that occur frequently together) can be determined by the frequent episodes algorithm. Both of these algorithms were developed and packaged as a utility to analyze audit data in order to determine a good set of features to use for generation of a misuse detection model.

Optimized Rule Models for Fast Run Time Evaluation

Once such a set of features have been determined, the next step is to apply a standard machine learning algorithm in order to generate specific models that distinguish normal activity from attacks, and possibly the class of attack. It is often very important to know not only that an attack has occurred, but what the attack is in order to asses its potential damage. Some attacks, for example *port scanning,* are used to gather information, but cause no harm. *Buffer overflow* attacks, however, can provide the attacker with administrator access to the victim's systems. Distinguishing among different types of attacks is thus made possible by computing a classifier over the data extracted and derived from the audit sources.

In the earliest work done at Columbia, a machine learning algorithm, RIPPER (a standard rule--based machine learning algorithm developed at ATT research), was used to compute rule-based models for detecting attacks. Aside from being efficient, RIPPER rules are easy for humans to read and understand, and even easier to compile into a deployable detector. Thus, one of the other core technologies developed by the Columbia team is a translator that converts a RIPPER rule set for detecting attacks into a format suitable for use by NFR.

The compilation process aims to generate rule sets that are optimized for real time performance. This is accomplished in two ways.

First, the Columbia team created a variant of RIPPER, a cost-sensitive variant of the standard rule learning algorithm, that takes into account the cost of features when generating rule models. Cost-sensitive machine learning has been an area of research for some time with many exemplar learning tasks drawn from medical diagnosis where the costs of features (laboratory tests, biopsies, etc.) clearly have widely varying costs. There are many ways to devise a cost-sensitive learning algorithm and Columbia researchers have experimented with and demonstrated several, including a variety of boosting methods. Boosting algorithms aim to reduce the misclassification error rate of a model by altering its training set. A specific algorithm, called AdaCost was developed to boost the performance of a model by minimizing the misclassification *cost* of a model over its training set. The cost model will bias the learning algorithm to compute a classifier with a number of cheaper feature tests rather than another that is perhaps more accurate but also more costly.

The second means of generating efficient rule models for real-time performance is by partitioning and scheduling feature computations intelligently. Certain features require much more computation than others, and should be avoided, if possible, to maintain system throughput without compromising system security. (IDSs contribute to system overheads which reduce computational resources devoted to providing services.) The cost models that declare the relative operational costs of features are used to derive estimates at compile time of the costs of the rule models that detect attacks at run time. Methods were explored in a Ph.D. thesis at Columbia to do this effectively. Recently a fully functional system has been completed that implements a strategy that schedules the rule model evaluation by partitioning "cheap" feature computations and expensive feature computations among two or more computers, a front end evaluator and a back end evaluator. The goal is to reduce the total overhead of running an IDS, and to minimize the IDS's response time in order to detect attacks as quickly as possible and avoid additional damage cost.

This mechanism provides an opportunity and means of repartitioning the computation at run time based upon observations about the run time performance of the operational IDS. (This latter work is the subject matter of the research activities of a former Columbia Ph.D. student, now at NC State and collaborator on the DARPA funded project.) This has become a crucial desideratum. It has recently been observed that some attackers have levied attacks against IDSs! By overwhelming an IDS with a large number of known attack signatures, the IDS becomes essentially ineffective. Thus, IDSs themselves need to be observed and protected. One means

of accomplishing this is to build in a cost model for IDSs (i.e. their operational costs are a key feature to model) which may vary dynamically at run time and thus vary the IDS computation to maintain throughputs. This is likely the only means of preventing an IDS from being overwhelmed and surviving a "Denial of Sensor" attack.

This completes the loop. From (labeled) audit data to feature extraction, and from model building to deployment into a functioning misuse detector. The entire set of technologies described thus far is part of an infrastructure we have come to call JUDGE.

Subsequent work has extended this framework in several ways. In one case, we needed a method to update deployed models rapidly, without the time consuming process of completely retraining a misuse detection model. Furthermore, misuse models only detect what is known, not what is new. Hence, the new data mining approach to intrusion detection is to extend and upgrade detectors to deal with the case of anomaly detection for new attacks. For this case, we developed a new class of models based upon probability theory. Experimentation showed that combining both misuse detection models with anomaly detection models creates a detector that senses known attacks and alerts operators to potential new attacks that were previously undetectable. First, we briefly describe updating an existing deployed model with a new model for a specific new attack that has been recently discovered. Then we describe approaches to anomaly detection.

Updating Misuse Detectors - Agility

Attackers are constantly developing new attacks that go undetected for long periods of time before being discovered by a human and incorporated into hand-crafted IDSs. Intrusion detection systems need to be agile. They need to efficiently and accurately adapt to incorporate new knowledge about previously unseen but newly discovered attacks. This task can be accomplished either by hand coding new attack patterns as rules and inserting into existing models which are then broadly distributed, or by some automatic means of learning about new attacks and incorporating this knowledge into existing deployed models.

To do the latter, one may add new labeled data to the archived training data used in developing the existing IDS model, and then re-computing the misuse detection model from scratch. This is expensive and slow.

The Columbia team built upon the earlier work on multiple model based systems (the JAM project that investigated fraud in e-commerce). The team developed algorithms for an "ensemble-based'" method to efficiently learn a light-weight model from audit data of new attack patterns that is then "attached" to an existing previously learned model by a decision rule system. Several configurations varying in the form of the underlying model and decision rules have been developed and tested under simulation.

The method first converts an existing IDS misuse model to a combined misuse and anomaly detection model. This is accomplished by injecting "artificial anomalies" into the training data used to create the first misuse detection model. The outcome of this DBA2, or Distribution-Based Artificial Anomaly, generation algorithm is a detector that can predict when a network connection is a known intrusion, an apparent "normal" connection, or an anomalous connection. In the latter case, a new rule is learned specifically for the newly discovered attack, which can be consulted by the detector during future analysis.

The training cost of this method is significantly less than re-training a monolithic model from both new and old training data. Empirical studies of simulations over audit data supplied by DARPA show that our ensemble-based method produces models that has accuracy comparable to a monolithic detector, but the model generation time is on the order of 150 times faster. This quick learning time provides an opportunity to deploy new models rapidly, which can be replaced later with an updated and better monolithic model as time and resources permit.

Anomaly Detection - New Attacks

As mentioned, many IDSs rely upon human expertise and the codification of expert knowledge about "known" attacks. These are malicious attacks that someone has noticed, analyzed, and reported to a large community of computer scientists in order to develop a "rule" that can detect that activity. (For example, an IDS developer may write the rule: if a server sees

more than 10 requests for a specific service within 10 milliseconds from the same IP address, then a denial of service attack is underway.) Thus, IDSs behave much like commercial virus detection systems. They are very good at detecting known attacks (if they have been updated), but entirely useless in detecting "new" attacks that have not yet been seen by expert observers. They may simply not have the required "signature rules" to detect an attack.

Data mining based intrusion detection provides an opportunity to learn a "generalized" model of an attack (or class of similar attacks) that may detect new, unseen attacks, which are variants of known exploits. Hence, we view intrusion detection as a "statistical pattern recognition" problem, rather than a "signature matching" problem. This new view provides an opportunity to develop a new class of models that may now be employed to protect our critical systems.

Viewed as a statistical pattern recognition problem, we need not only concern ourselves with computing effective misuse models for known attacks, but as well we may compute models of normal system behavior in order to detect "out of model anomalies." Thus, our data mining approach provides the only hope of computing anomaly detectors that cast a wider net of protection over complex systems. (To make the point clearer, imagine asking a system administrator to write down rules that describe normal system behavior for a network of more than 100 computers!)

For this purpose, the Columbia team has been investigating and developing various probabilistic modeling techniques for normal system operation that (a) learn continuously under the assumption of noisy data (adaptive learning) and (b) optimize models for the best possible performance (dynamic window modeling) specific to the system being protected. Similar to the case of cost-sensitive modeling for misuse detection, one system's model of normal must clearly be abnormal for other differing network environments. By first principles, one must therefore learn over time what *is* normal for a specific network. It should be evident that the "normal behavior" we wish to model is statistical in nature.

Adaptive Learning of Anomalies - Learning under noise

It has been observed that training a data mining-based IDS requires labeled data that is noise free (i.e. it is known that the data contains *only* normal events and *correctly* labeled intrusion data). This means that in order to compute a model of a known attack, one needs to present data of the attack, labeled as such, and data that is normal in order to compute a "rule" that distinguishes the two classes. Such data is hard to produce (although we have built a facility to generate such data in our laboratory using the SIMNET network data simulation and generator from MIT Lincoln Labs). And, when training a system to compute a model of normal behavior, one may not be certain that the system's operation being audited is entirely normal, i.e. attacks may appear in the data during training without the learning supervisor knowing that it is there. Hence, a statistical model trained in this fashion may model an attack as "normal!" An algorithm has been developed that provides a means of solving this problem by assuming the source of the data originates from two distributions.

We use a probabilistic modeling technique to model anomalies. We use a mixture model technique that assumes there are two cases for each datum. With some probability p, a datum is assumed to have been generated by a majority distribution (normal), while with probability (1-p), the datum is assumed to have originated from a minority distribution (anomaly). Under this probabilistic model, one can then test an unknown datum to determine the likelihood of which distribution it belongs to. Hence, whether it is normal or not.

Interestingly, this technique can be applied to any audit data source, at the network level or at the host level. And this approach can operate continuously in real-time environments.

As a system in use is audited, the real time data stream is first tested by the probabilistic model (after, of course, being tested by the misuse models). If that data is deemed "normal", then it may be used in real time. A model of system behavior is therefore continuously updated to hone the accuracy of the predicted likelihood of normalcy of unknown data.

Dynamic Window Sizes for Host-based IDS - building better models

In some cases, the audit data we are dealing with is sequential in nature. For example, for host-based intrusion detection, one may wish to model normal program or utility behavior as a function of the sequence of operating system calls typically used during normal operation.

Previous work on host-based IDS has shown the utility of modeling normal system call sequences. By gathering traces of system calls of some program or system utility under normal operation, one can model these normal traces in order to produce an anomaly detector. After training a model of normal sequences, one can detect abnormal execution patterns if they do not fit the model. This approach has been demonstrated for the s*endmail* facility of Unix-based platforms by Stephanie Forrest's group at UNM.

The earliest work on modeling sequences of system calls used a technique that is ad hoc and has met with limited success. There are several reasons for this; however, the most notable reason is the limited amount of information extracted from the system call sequences that have been analyzed. One of the most limiting aspects of this work has been the reliance upon *fixed window size* modeling. This means that models are built by extracting a static number of consecutive systems calls from a trace (the length of this static sequence is the "window size"), and simply storing those subsequences in what is called a "normal database." Researchers were quick to note that this was a trivial example of "n-gram" modeling, which is popular in a variety of fields, such as information retrieval, which aims to analyze and model text based data.

The work performed at Columbia by a Graduate Research Assistant extended this to probabilistic models over sequences of varying window size. Hence, rather than modeling sequences of fixed size, the approach developed at Columbia modeled normal system call sequences using a probabilistic-based model (called Sparse Markov Transducers, or SMT's) over dynamic window sizes passed over the training data. The approach extracts much more useful information from the system traces leading to better anomaly detectors with higher accuracy and lower error rate. The modeling technique has been released as an Open Source utility and is available from our group's website. Interestingly, the method has also been applied to sequence analysis problems in genomics.

Correlation of Multiple Sensors and Detectors

It is not sensible to assume that a single sensor and detector can accurately catch all possible attacks levied against a complex network system. In a distributed environment, one will likely develop multiple sensors, operating among a variety of different computational sites, each specialized for different purposes (e.g., a host based detector for one platform, such as Windows, interoperating with a network-level detector for TCP/IP connections). Thus, many detectors may broadcast alerts at different times about attacks whose behavior may be sensed and detected differently.

Note that there is little intellectual difference between a purposeful attack that damages or disables a network component (for example, halting a router), and a *fault* that disables a network component (for example, a failed power supply within the router). The field of network management has studied and developed a large body of technologies that help operators manage networks under various real time faults. (Indeed, it is sensible to assume that intrusion detection will ultimately become a component feature of standard network management products.) Faults need to be detected, and this is accomplished by observing system events among a large collection of distributed components. These observable events must be correlated to improve detection coverage while reducing the potential for an overwhelming number of alarms.

Event correlation, as applied to network management and distributed computing, deals with finding the source of a node or service failure, where such failures may cause cascading alarms that might mask the source of the problem. Given a set of symptoms, one must infer the underlying cause of the problem. This is an inherently complex task; in a complex network, nodes may mask the output of other nodes, and typical networks change rapidly as new network components are added or retired. It also involves outside knowledge about the network topology, configuration, and sources of events (e.g., the relationship among the different event sources).

Approaches to this problem involve correlating a model-based or rule-based approach to capture relationships among events. In all approaches, the objective is to reduce the amount of audit data and to identify the source event. The techniques that have been developed at Columbia, and more

generally in the area of network management, may of course be applied to the case of intrusion detection.

In intrusion detection, the amount of event information that is gathered by multiple sensors can overwhelm an analysis engine. Correlating events and using robust modeling techniques, such as a mixture model approach (that we have applied in the anomaly detection case), greatly reduces the amount of information that is needed to a point where it can become manageable.

Work is now underway to develop a probabilistic model of event correlation that seeks to correlate multiple sources of information from multiple sensors to improve overall intrusion detection rates. New and positive results are expected.

Data Mining for Virus Detection

As alluded to in the introduction of this document, data mining may also be applied to virus detection by analyzing and generalizing over static properties of programs, rather than the dynamic behavior of a system under attack. The team at Columbia gathered thousands of "published" viruses from various sources on the Internet (that are detectable by current signature-based commercial virus scanners) and applied data mining techniques to successfully generate models for detection.

Features are first extracted from virus programs using standard tools revealing the set of library calls used by the virus program, the binary code sequences of the program, and the set of "printable strings" embedded in the payload of the virus. A probabilistic model is then computed over training sets with viruses and normal programs using one of the simplest of modeling algorithms, the Naïve Bayes algorithm. The resultant experimental tests have demonstrated that this approach is indeed successful; it is able to detect viruses previously unseen by the model. Further development is under way to extend the set of features that are used in modeling virus programs, as well as in the particulars of the probabilistic model itself. For example, the dynamic window size technique using Sparse Markov Transducers will likely produce improved detection models that are also smaller and more compact than the models produced by the Naïve Bayes technique.

Generating Training Data

It should be evident that the data mining techniques that have been broadly described herein depend wholly upon the acquisition of appropriately labeled training data. (The adaptive learning technique aims to relieve the burden of demanding absolutely *clean* training data, but does not entirely eliminate the requirement.)

Many organizations gather and archive vast amounts of attack data and specific exploit scripts that implement attacks, but rarely do they share their data. The reasons are clearly due to the proprietary nature of the data (it may be quite valuable to competitors and would-be attackers), and also because of privacy concerns.

DARPA contracted MIT Lincoln Labs to produce a simulation and test environment for IDS research and the research community has greatly benefited from their efforts. (Indeed, we have been approved as a user of MIT's simulation code to generate our own network attack data.) Few other simulation environments exist, and it has proven nearly impossible to convince any organization to provide data gathered from live environments.

Since acquiring sufficient data for training is crucial in this line of work, the Columbia team has built its own simulation and test environment that allows any (CIDF/IDMEF-compliant) sensor or detector to be deployed and operated, along with a collection of exploit scripts with which to attack systems, with the ability to gather all data with a common clock time that is deposited into a data warehouse for inspection and use. This facility may also serve as the core of a *forensics* capability, whereby archived data stores may be analyzed for previous attack information that was previously undetectable or not noticed.

Technology Elements

By way of summary, the following component technologies have been developed at Columbia for an end-to-end data mining-based IDS:

1. A distributed, network-based infrastructure to insert and integrate components including
    1. auditing programs (BAMs, file wrappers)
    2. sensors (packet sniffers like NFR or Bro or any arbitrary CIDF/IDMEF compliant IDS with a sniffer),
    3. detectors with programmable models (rules for use by NFR that have been compiled from machine learning models for misuse detection as well as probabilistic anomaly detectors),

4. a data warehouse for storing and archiving gathered data (MySql)
5. data mining algorithms packaged as a set of utilities that propose useful features from audit data (Madam/ID),
6. machine learning programs that generate rules of known attacks or probabilistic models of normal or anomalous behavior (cost-sensitive RIPPER, AdaCost, SMT probabilistic modelers),
7. distributors to insert or update new models,
8. An infrastructure to deploy detectors and sensors, and to attack systems, gather labeled audit data, and evaluate performance,
9. An XML based interface language to handle communication between all components (based upon CIDF and IDMEF proposed standards),
10. Algorithms that compute cost-sensitive models that minimize training misclassification costs (AdaCost),
11. Programs that implement cost-sensitive models for misuse detection by partitioning model evaluations between front end and back end systems (Judge),
12. Ensemble-based algorithms to update models rapidly when new attacks are learned,
13. Algorithms to convert a misuse detector into a combined misuse and anomaly detector (DBA2),
14. Probabilistic modeling programs that model normal and anomalous behaviors (Mixture models),
15. Probabilistic modeling programs for sequential data sources (dynamic window size modeling via SMT's),
16. Tools that extract features from executables and email attachments for modeling of viruses and malicious code attachments transmitted in email messages.

BIBLIOGRAPHY

The following set of papers produced by the Columbia IDS project are available at the website http://www.cs.columbia.edu/ids.

Wenke Lee and Dong Xiang, Information-Theoretic Measures for Anomaly Detection, The 2001 IEEE Symposium on Security and Privacy, Oakland, CA, May 2001.

2. J. B. D. Cabrera, L. Lewis, X. Qin, Wenke Lee, Ravi Prasanth, B. Ravichandran, and Raman Mehra, Proactive Detection of Distributed Denial of Service Attacks Using MIB Traffic Variables - A Feasibility Study, The Seventh IFIP/IEEE International Symposium on Integrated Network Management (IM 2001), Seattle, WA, May 2001.
3. Wenke Lee, Rahul Nimbalkar, Kam Yee, Sunil Patil, Pragnesh Desai, Thuan Tran, and Sal Stolfo, A Data Mining and CIDF Based Approach for Detecting Novel and Distributed Intrusions In Proceedings of The Third International Workshop on Recent Advances in Intrusion Detection (RAID 2000), Lecture Notes in Computer Science No. 1907, Toulouse, France, October 2000.
4. Eleazar Eskin, Wenke Lee and Salvatore J. Stolfo. ``Modeling System Calls for Intrusion Detection with Dynamic Window Sizes." Proceedings of DISCEX II. June 2001.
5. Wenke Lee, Salvatore J. Stolfo, Philip K. Chan, Eleazar Eskin, Wei Fan, Matthew Miller, Shlomo Hershkop and Junxin Zhang. ``Real Time Data Mining-based Intrusion Detection.", Proceedings of DISCEX II. June 2001.
6. Matthew G. Schultz, Eleazar Eskin, and Salvatore J. Stolfo. ``Malicious Email Filter - A UNIX Mail Filter that Detects Malicious Windows Executables." Proceedings of USENIX Annual Technical Conference - FREENIX Track. Boston, MA: June 2001.
7. Matthew G. Schultz, Eleazar Eskin, Erez Zadok, and Salvatore J. Stolfo. ``Data Mining Methods for Detection of New Malicious Executables" Proceedings of IEEE Symposium on Security and Privacy. Oakland, CA: May 2001.
8. Leonid Portnoy. ``Intrusion Detection with Unlabeled Data using Clustering" Undergraduate Thesis. Columbia University: December,2000.
9. Eleazar Eskin, Matthew Miller, Zhi-Da Zhong, George Yi, Wei-Ang Lee, Sal Stolfo. ``Adaptive Model Generation for Intrusion Detection Systems" Workshop on Intrusion Detection and Prevention, 7th ACM Conference on Computer Security, Athens, GR: November, 2000.
10. Wenke Lee, Wei Fan, Matthew Miller, Sal Stolfo, and Erez Zadok. ``Toward Cost-Sensitive Modeling for Intrusion Detection and Response" Workshop on Intrusion Detection and Prevention, 7th ACM Conference on Computer Security, Athens, GR: November, 2000.
11. Eskin, Eleazar. ``Anomaly Detection over Noisy Data using Learned Probability Distributions" ICML00, Palo Alto, CA: July, 2000.

12. Wei Fan, Wenke Lee, Sal Stolfo, and Matthew Miller. ``A Multiple Model Cost-Sensitive Approach for Intrusion Detection'', Eleventh European Conference on Machine Learning (ECML '00) 2000.
13. Sal Stolfo, Wei Fan, Wenke Lee, Andreas Prodromidis, and Phil Chan. ``Cost-based Modeling for Fraud and Intrusion Detection: Results from the JAM Project'' In Proceedings of the 2000 DARPA Information Survivability Conference and Exposition (DISCEX '00), 2000.
14. Wenke Lee, Matthew Miller, Sal Stolfo, Kahil Jallad, Christoper Park, Erez Zadok, and Vijay Prabhakar. ``Toward Cost-Sensitive Modeling for Intrusion Detection'' Columbia University Computer Science Technical Report CUCS-002-00.
15. Matthew Miller. ``Learning Cost-Sensitive Classification Rules for Network Intrusion Detection using RIPPER'' Columbia University Computer Science Technical Report CUCS-035-1999.
16. Wenke Lee, Sal Stolfo, and Kui Mok. ``Mining in a Data-flow Environment: Experience in Network Intrusion Detection'' In Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '99), San Diego, CA, August, 1999.
17. Wenke Lee, Sal Stolfo, and Kui Mok. ``A Data Mining Framework for Building Intrusion Detection Models'' In Proceedings of the 1999 IEEE Symposium on Security and Privacy, Oakland, CA, May 1999.
18. Wenke Lee, Chris Park, and Sal Stolfo. ``Towards Automatic Intrusion Detection using NFR'' In Proceedings of the 1st USENIX Workshop on Intrusion Detection and Network Monitoring, April 1999.
19. Wenke Lee, Sal Stolfo, and Kui Mok. ``Mining Audit Data to Build Intrusion Detection Models'', In Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD '98), New York, NY, August 1998.
20. Wenke Lee and Sal Stolfo. ``Data Mining Approaches for Intrusion Detection'' In Proceedings of the Seventh USENIX Security Symposium (SECURITY '98), San Antonio, TX, January 1998.
21. Wenke Lee, Sal Stolfo, and Phil Chan. ``Learning Patterns from Unix Process Execution Traces for Intrusion Detection'', AAAI Workshop: AI Approaches to Fraud Detection and Risk Management, July 1997.

Journal Publications

Cost Complexity-based Pruning of Ensemble Classifiers, (with A. Prodromidis), *Journal on Distributed and Parallel KDD*, Special Issue on Knowledge and Information Systems, 2000.
2. A Framework for Constructing Features and Models for Intrusion Detection Systems, (with W. Lee), *ACM Transactions on Information and System Security*, TISSEC, Vol 3, No. 4, November, 2000.
3. Distributed Data Mining in Credit Card Fraud Detection, (with P. Chan, W. Fan, A. Prodromidis*), IEEE Intelligent Systems*, Vol. 14, No. 6, 1999.
4. Adaptive Intrusion Detection: a Data Mining Approach, (with W. Lee and K. Mok*), Artificial Intelligence Review,* Volume 14, No. 6, Kluwer Academic Publishers, 2000, pp. 533-567.
5. Wenke Lee, Wei Fan, Matthew Miller, Sal Stolfo, and Erez Zadok. ``Toward Cost-Sensitive Modeling for Intrusion Detection and Response'' Journal of Computer Security, (to appear), 2002.

Theses

1. Philip Chan, An Extensible Meta-Learning Approach for Scalable and Accurate Inductive Learning, 1996. (Asst. Professor Florida Institute of Technology)

2. Andreas Prodromidis, Efficiency and Scalability of Distributed Data Mining, Pruning and Bridging Multiple Models September 1999. (Director of Research iPrivacy LLC)

3. Wenke Lee, A Data Mining Framework for Constructing Features and Models for Intrusion Detection Systems, June, 1999. (Asst. Professor, Georgia Tech).

4. Dave (Wei) Fan, Cost-sensitive, Scaleable AdaptiveLlearning. May 2001, (Member of the research staff at IBM research)