MODERN INTRUSION DETECTION, DATA MINING, AND DEGREES OF ATTACK GUILT

Steven Noel
Center for Secure Information Systems
George Mason University, Fairfax VA 22030-4444, USA
snoel@gmu.edu

Duminda Wijesekera
Center for Secure Information Systems
George Mason University, Fairfax VA 22030-4444, USA
dwijesek@gmu.edu

Charles Youman
Center for Secure Information Systems
George Mason University, Fairfax VA 22030-4444, USA
charles.youman@att.net

Abstract

This chapter examines the state of modern intrusion detection, with a particular emphasis on the emerging approach of data mining. The discussion parallels two important aspects of intrusion detection: general detection strategy (misuse detection versus anomaly detection) and data source (individual hosts versus network traffic). Misuse detection attempts to match known patterns of intrusion, while anomaly detection searches for deviations from normal behavior. Between the two approaches, only anomaly detection has the ability to detect unknown attacks. A particularly promising approach to anomaly detection combines association mining with other forms of machine learning such as classification. Moreover, the data source that an intrusion detection system employs significantly impacts the types of attacks it can detect. There is a tradeoff in the level of detailed information available versus data volume. We introduce a novel way of characterizing intrusion detection activities: degree of attack guilt. It is useful for qualifying the degree of confidence associated with detection events, providing a framework in which we analyze detection quality versus cost.

Keywords: Information security, Intrusion detection, data mining

1. Introduction

The goal of intrusion detection is to discover intrusions into a computer or network, by observing various network activities or attributes. Here intrusion refers to any set of actions that threatens the integrity, availability, or confidentiality of a network resource.

Given the explosive growth of the Internet and the increased availability of tools for attacking networks, intrusion detection becomes a critical component of network administration. While such detection usually includes some form of manual analysis, we focus on software systems for automating the analysis.

One useful method of classification for intrusion detection systems is according to general strategy for detection. There are two categories under this classification: misuse detection and anomaly detection.

Misuse detection finds intrusions by looking for activity corresponding to known techniques for intrusion. This generally involves the monitoring of network traffic in search of direct matches to known patterns of attack (called signatures). This is essentially a rule-based approach. A disadvantage of this approach is that it can only detect intrusions that follow pre-defined patterns.

In anomaly detection, the system defines the expected behavior of the network (or profile) in advance. Any significant deviations from this expected behavior are then reported as possible attacks. Such deviations are not necessarily actual attacks. They may simply be new network behavior that needs to be added to the profile. The primary advantage of anomaly-based detection is the ability to detect novel attacks for which signatures have not been defined.

Another useful method of classification for intrusion detection systems is according to data source. The two general categories are host-based detection and network-based detection.

For host-based intrusion detection, the data source is collected from an individual host on the network. Host-based detection systems directly monitor the host data files and operating system processes that will potentially be targets of attack. They can, therefore, determine exactly which host resources are the targets of a particular attack.

For network-based intrusion detection, the data source is traffic across the network. This involves placing a set of traffic sensors within the network. The sensors typically perform local analysis and detection and report suspicious events to a central location. Since such monitors

perform only the intrusion detection function, they are usually much easier to harden against attack and to hide from the attackers.

We propose another way of characterizing intrusion detection activities, through degree of attack guilt. That is, it is interesting to understand how well a system can correctly separate genuine attacks from normal activity in terms of attack guilt. This is not a classification of detection systems, but rather of network activities. Intrusion degree of guilt is useful for qualifying the degree of confidence associated with detection events, providing a framework for analyzing detection quality versus cost.

This chapter examines the state of modern intrusion detection. Section 2 discusses the state of the art with respect to generally strategy for detection. Section 3 then considers intrusion detection systems in terms of their data sources. In Section 4, we introduce degree of attack guilt as a way of characterizing intrusion detection activities, providing a framework in which we analyze detection quality versus cost. Section 5 has our concluding comments.

2. Detection Strategies

The current generation of commercial intrusion detection systems is largely network-based, and employs misuse detection. As such, current tools completely lack the ability to detect attacks that do not fit a pre-defined signature. Several other researchers have reached this conclusion (Corporation, 2000; Jackson, 1999; LaPadula, 1999; LaPadula, 2000; Allen et al., 2000; Axelsson, 1999; Axelsson, 2000b; Kvarnstrom, 1999).

Given the shortcomings of misuse detection in commercial systems, an important research focus is anomaly detection, rather than mere extensions of misuse detection. Research is also needed in systems that combine the two approaches. A critical issue for anomaly detection is the need to reduce false alarms, since any activity outside a known profile raises an alarm. Indeed, false alarm rate is the limiting factor in the performance of current intrusion detection systems (Axelsson, 2000a; Lundin and Jonsson, 1999).

Increased network speeds, switched networks, and the application of encryption have prompted a trend toward host-based detection. Another interesting new approach is distributed intrusion detection, in which host-based systems monitor a number of hosts on the network and transfer the monitored information to a central site.

Overall, intrusion detection technology is immature and rapidly evolving. In the commercial realm, new vendors appear frequently but are

often absorbed by others. On the research front, a variety of approaches are being investigated. However, an overall theoretical framework is still lacking.

2.1. Misuse Detection

Misuse detection searches for known patterns of attack. This is the strategy employed by the current generation of commercial intrusion detection systems. A disadvantage of this strategy is that it can only detect intrusions that follow pre-defined patterns.

The major approaches that have been proposed for misuse detection are expert systems, signature analysis, state-transition analysis, and data mining. Approaches have also been proposed involving colored Petri nets and case-based reasoning.

Misuse detection searches for known patterns of attack. This is the strategy employed by the current generation of commercial intrusion detection systems. A disadvantage of this strategy is that it can only detect intrusions that follow pre-defined patterns.

The major approaches that have been proposed for misuse detection are expert systems, signature analysis, state-transition analysis, and data mining. Approaches have also been proposed involving colored Petri nets and case-based reasoning.

2.1.1 Expert Systems. The expert system approach to misuse detection uses a set of rules to describe attacks. Audit events are translated into facts carrying their semantic significance in the expert system. An inference engine then draws conclusions using these rules and facts.

Examples of misuse detection systems using expert systems are IDES (Intrusion Detection Expert System) (Denning, 1987; Lunt, 1989; Lunt et al., 1992; Javitz and Valdes, 1991), ComputerWatch (Dowell and Ramstedt, 1990), NIDX (Network Intrusion Detection Expert System) (Bauer and Koblentz, 1988), P-BEST (Production- Based Expert System Toolset) (Lindqvist and Porras, 1999), and ISOA (Information Security Officer's Assistant) (Winkler and Landry, 1992; Winkler, 1990).

IDES (developed at SRI) uses an expert system that encodes known intrusion scenarios, known system vulnerabilities, and site-specific security policies. It addresses external attacks from unauthorized users, authorized users who masquerade as other users, and authorized users who abuse their privileges by evading access controls.

ComputerWatch (developed at AT&T) takes an expert system approach to summarize security sensitive events and apply rules to detect anomalous behavior. It checks users' actions according to a set of rules

that describe proper usage policy, and flags any action that does not fit the acceptable patterns.

NIDX (developed at Bell Communication Research) is a knowledge-based prototype intrusion detection expert system for Unix System V. It combines knowledge of the target system, history profiles of users' past activities, and intrusion detection heuristics. The result is a knowledge-based system capable of detecting specific violations that occur on the target system. A unique feature of NIDX is that it includes facts describing the target system and heuristics embodied in rules that detect particular violations from the target system audit trail. NIDX is thus operating system dependent.

P-BEST (developed at SRI) is a rule-based, forward-chaining expert system that has been applied to signature-based intrusion detection for many years. The main idea is to specify the characteristics of a malicious behavior and then monitor the stream of events generated by system activity, hoping to recognize an intrusion signature.

P-BEST is a general-purpose programmable expert system shell, sporting a rule definition language that is simple enough to be used by non-experts. The system was first deployed in the MIDAS ID system at the National Computer Security Center. Later, P-BEST was chosen as the rule-based inference engine of NIDES, a successor to the IDES prototype. The P-BEST expert system shell is also used in EMERALD's expert, a generic signature-analysis engine (Porras and Neumann, 1997).

ISOA (developed at Planning Research Corporation) is a real time security monitor that supports automated as well as interactive audit trail analysis. It contains a statistical analysis module and an expert system. For the events not constituting direct violations of security policy, their expected behavior is compared against profiles that specify thresholds and the reliability factor for the events. Deviations are identified by statistical checks of expected versus actual behavior. For events that cannot be monitored by examining the thresholds, the expert system component can specify the possible relationships and implied meaning of the events.

2.1.2 Signature Analysis. Signature analysis transforms the semantic description of attacks into information that can be found in the audit trail in a straightforward way. Examples of such information include the sequences of audit events that attacks generate, or patterns of data that can be sought in the audit trail.

Systems that use signature analysis include Haystack (Smaha, 1988), NetRanger (NetRanger, 1999), RealSecure (Real-Secure, 1999), and MuSig (Misuse Signatures) (Lin et al., 1998).

Haystack is a misuse detection system that helps Air Force security officers detect misuse of Unisys mainframes. Working on the reduced audit trails data, it performs misuse detection based on behavioral constraints imposed by official security policies and on models of typical user behavior.

NetRanger (developed at Cisco) is composed of two modules: sensors and directors. Sensors are network security monitors that analyze the network traffic on a network segment and the logging information produced by Cisco routers to detect network-based attacks. Directors are responsible for the management of a group of sensors and can be structured hierarchically to manage large networks.

RealSecure (developed at Internet Security Systems) is composed of three modules: network engines, system agents, and managers. The network engines are network monitors equipped with attack signatures that are matched against the traffic on a network link. The system agents are host-based intrusion detection systems that monitor security sensitive log files on a host. These modules report their finds to the central manager, which displays the information to the user and provides functionalities for remote administration system agents and network engines.

MuSig (developed at George Mason University's Center for Secure Information Systems) applies a high-level language for abstract signatures. It attempts to overcome certain limitations of traditional misuse detection systems, including the limited expressiveness of signatures expressed in low-level language, and fixed monitoring algorithms for misuse that have difficulty adapting to a changing operating environment or security objectives. Through its high-level language, MuSig can represent misuses in a simple form with high expressiveness.

2.1.3 State-Transition Analysis. State-transition analysis describes attacks with a set of goals and transitions based on state-transition diagrams. Any event that triggers an attack state will be considered an intrusion. Examples of systems applying state transition analysis are USTAT (Unix State Transition Analysis Tool) (Porras and Kemmerer, 1992; Ilgun, 1992) and NetSTAT (Network-based State Transition Analysis Tool) (Vigna and Kemmerer, 1998).

USTAT (developed at UC Santa Barbara) is a real-time intrusion detection system for Unix. The original design was STAT (State Transition Analysis Tool) (Porras, 1992). STAT employs rule-based analysis of the audit trails of multi-user computer systems. In STAT, an intrusion is identified as a sequence of state changes that lead the computer system from some initial state to a target compromised state. USTAT makes use of the audit trails that are collected by the C2 Basic Security Module

of SunOS. It keeps track of only those critical actions that must occur for the successful completion of the penetration. This approach differs from other rule-based penetration identification tools that pattern match sequences of audit records.

NetStat (developed at UCSB) performs real-time network-based intrusion detection by extending the state transition analysis technique (first introduced in STAT) to the networked environment. The system works on complex networks composed of several sub-networks. Using state transition diagrams to represent network attacks entails a number of advantages, including the ability to automatically determine the data to be collected to support intrusion analysis. This enables a lightweight and scalable implementation of the network probes.

2.1.4 Data Mining. Data mining refers to a process of nontrivial extraction of implicit, previously unknown, and potentially useful information from databases. Example misuse detection systems that use data mining include JAM (Java Agents for Metalearning) (W. Lee and Mok, 1998; Lee and Stolfo, 1998; Lee et al., 1999; Lee et al., 2000; Lee, 1999) MADAM ID (Mining Audit Data for Automated Models for Intrusion Detection) (Lee et al., 2000), and Automated Discovery of Concise Predictive Rules for Intrusion Detection (Lee, 1999).

JAM (developed at Columbia University) uses data mining techniques to discover patterns of intrusions. It then applies a meta-learning classifier to learn the signature of attacks. The association rules algorithm determines relationships between fields in the audit trail records, and the frequent episodes algorithm models sequential patterns of audit events. Features are then extracted from both algorithms and used to compute models of intrusion behavior. The classifiers build the signature of attacks. So essentially, data mining in JAM builds a misuse detection model.

JAM generates classifiers using a rule learning program on training data of system usage. After training, resulting classification rules is used to recognize anomalies and detect known intrusions. The system has been tested with data from Sendmail-based attacks, and with network attacks using TCP dump data. MADAM ID uses data mining to develop rules for misuse detection. The motivation is that current systems require extensive manual effort to develop rules for misuse detection.

MADAM ID applies data mining to audit data to compute models that accurately capture behavioral patterns of intrusions and normal activities. While MADAM ID performed well in the 1998 DARPA evaluation of intrusion detection systems (Lippmann et al., 2000), it is ineffective in detecting attacks that have not already been specified.

Researchers at Iowa State University report on Automated Discovery of Concise Predictive Rules for Intrusion Detection (Helmer et al., 1999). This system performs data mining to provide global, temporal views of intrusions on a distributed system. The rules detect intrusions against privileged programs (such as Sendmail) using feature vectors to describe the system calls executed by each process. A genetic algorithm selects feature subsets to reduce the number of observed features while maintaining or improving learning accuracy. This is another example of data mining being used to develop rules for misuse detection.

2.1.5 Other Approaches. The colored Petri nets approach is a graphical language for design, specification, simulation and verification of systems. It is particularly well-suited for systems in which communication, synchronization and resource sharing are important (Jensen, 1997).

The only known example of an intrusion detection system that uses colored Petri nets is IDIOT (Intrusion Detection in Our Time) (Crosbie et al., 1996), developed at Purdue University. In particular, IDIOT applies colored Petri nets to represent attack signatures. Advantages of colored Petri nets include their generality, their conceptual simplicity, and their ability to be represented as graphs. However, matching a complex signature against the audit trail can become computationally expensive.

Another approach to misuse detection involves case-based reasoning. Case-based reasoning is a problem solving methodology in which previous problem solving situations are used in solving new problems. It is most suitable when it is difficult or impossible to break down the knowledge into a set of rules, and only records of prior cases exist.

The only known application of case-based reasoning to intrusion detection is AUTOGUARD (Esmaili et al., 1996; Esmaili et al., 1997), although it is not clear if the system has been fully implemented.

2.2. Anomaly Detection

A significant disadvantage of misuse detection is the inability to detect attacks for which signatures have not been defined. A detection strategy that addresses this shortcoming is anomaly detection.

In anomaly detection, the system defines the expected network behavior (known as the profile) in advance. Any significant deviations from the profile are then reported as possible attacks. Such deviations are not necessarily actual attacks. They may simply be new network behavior that needs to be added to the profile. Anomaly detection systems have emerged that have very promising performance against novel attacks.

The major approaches to anomaly detection include statistical methods, expert systems, and data mining. Approaches have also been proposed involving neural networks and computer immunology.

2.2.1 Statistical Methods. Statistical methods measure the user and system behavior by a number of variables sampled over time, and build profiles based on the variables of normal behavior. The actual variables are then compared against the profiles, and deviations are considered abnormal.

Example systems employing statistical methods for anomaly detection are IDES (Intrusion Detection Expert System) (Denning, 1987; Lunt, 1989; Javitz and Valdes, 1991; Lunt et al., 1992) NIDES (Next- Generation Intrusion Detection Expert System) (Anderson et al., 1995a; Anderson et al., 1995b), and Event Monitoring Enabling Responses to Anomalous Live Disturbances (EMERALD) (Porras and Neumann, 1997; Neumann and Porras, 1999).

Section 2.1.1 describes the IDES system in the context of misuse detection. IDES also employs statistical anomaly detection. In particular, it uses audit data to characterize user activity and detect deviations from normal user behavior. Information extracted from audit data includes user login, logout, program execution, directory modification, file access, system calls, session location change, and network activity. The NIDES system extends IDES by integrating its response logic with the results produced by the anomaly detection subsystem.

EMERALD (developed at SRI) aims to detect intrusions in large networks, and focuses on the scalability of the system. It is a hybrid of misuse detection and anomaly detection that contains an expert system PBEST and a statistical anomaly detector. It allows hierarchical composition of decentralized service monitors that apply the statistical analysis to network data.

In other work, Cabrera et al.(Cabrera et al., 2000) examine the application of statistical traffic modeling for detecting novel attacks against networks. They show that network activity models efficiently detect denial of service and probe attacks by monitoring the network traffic volume. For application models, they use the KolmogorovSmirnov test to demonstrate that attacks using telnet connections in the DARPA dataset (Lippmann et al., 2000) are statistically different from normal telnet connections.

2.2.2 Expert Systems. For anomaly detection, expert systems describe users' normal behavior by a set of rules. Examples of expert systems applied to anomaly detection include ComputerWatch (Dowell

and Ramstedt, 1990) and Wisdom & Sense (Liepins and Vaccaro, 1992; Vaccaro and Liepins, 1989; Liepins and Vaccaro, 1989).

ComputerWatch (developed at AT&T) uses an expert system approach to summarize security sensitive events and apply rules to detect anomalous behavior. It checks users' actions according to a set of rules that describe proper usage policy, and flags any action that does not fit the acceptable patterns.

Wisdom & Sense (developed at the Los Alamos National Lab) detects statistical anomalies in users' behavior. It first builds a set of rules that statistically describe behavior based on recordings of user activities over a given period of time. Subsequent activity is then compared against these rules to detect inconsistent behavior. The rule base is rebuilt regularly to accommodate new usage patterns.

Terran Lane of Purdue University studied machine learning techniques for anomaly detection (Lane, 2000). The project profiles Unix user command line data. It shows that anomaly detection is effective at user differentiation under some conditions, but that alone it is insufficient for high-confidence defense. This approach assumes the false alarms generated by an intrusion detection sensor will be filtered out by a higher-level decision maker. The results are also tentative because they do not include any data known to be hostile.

2.2.3 Data Mining. Data mining attempts to extract implicit, previously unknown, and potentially useful information from data. Applications of data mining to anomaly detection include ADAM (Audit Data Analysis and Mining) (Wu, 2001a; Barbara et al., 2001; Barbara et al., 1999), IDDM (Intrusion Detection using Data Mining) (Abraham, 2001), and eBayes (Valdes and Skinner, 2000).

ADAM (developed at George Mason University Center for Secure Information Systems) uses a combination of association rules mining and classification to discover attacks in TCP dump data. Section 2.1.4 discusses the JAM system, which also combines association mining and classification. But there are two significant differences between ADAM and JAM. First, ADAM builds a repository of *normal* frequent itemsets that hold during attack-free periods. It does so by mining data that is known to be free of attacks. Then, ADAM runs a sliding-window algorithm that finds frequent itemsets in the most recent set of TCP connections, and compares them with those stored in the normal itemset repository, discarding those that are deemed normal. With the rest, ADAM uses a classifier that has been previously trained to classify the suspicious connections as a known type of attack, an unknown type, or

a false alarm. The system performs especially well with denial of service and probe attacks.

The ADAM system is able to detect network intrusions in real time with a very low false alarm rate. One of the most significant advantages of ADAM is the ability to detect novel attacks, without depending on attack training data, through this through a novel application of the pseudo-Bayes estimator (Barbara et al., 2001). In the 1999 DARPA Intrusion Detection Evaluation (Lippmann et al., 2000), ADAM ranked 3rd overall. Among the top 3, ADAM is the only system employing anomaly detection (Wu, 2001a). Not also that the DARPA evaluation has no criterion that singles out performance on new attacks. In short, there is no known system that is more effective at detecting unknown attacks than ADAM.

Abraham of the Defense Science and Technology Organization in Australia recently reported on IDDM. The system characterizes change between network data descriptions at different times, and produces alarms when detecting large deviations between descriptions. However, IDDM has problems achieving real-time operation. In particular, results are produced only after sufficient amounts of data are collected and analyzed.

The eBayes system is a newly developed component for the statistical anomaly detector of EMERALD. Defining a session as temporally contiguous bursts of TCP/IP traffic from a given IP, it applies Bayesian inference on observed and derived variables of the session, to obtain a belief for the session over the states of hypotheses. Hypotheses can be either normal events or attacks. Given a nave Bayes model, training data, and a set of hypotheses, a conditional probability table is built for the hypotheses and variables, and is adjusted for the current observations. By adding a dummy state of hypothesis and a new conditional probability table row initialized by a uniform distribution, eBayes can dynamically generate the new hypothesis that helps it detect new attacks. But eBayes may be computationally expensive as the number of hypothesis states increases.

Kohavi et al.(Kohavi et al., 1997) study different approaches for handling unknowns and zero counts when estimating probabilities for naive Bayes classifiers, and propose a new variant of the Laplace estimator that shows better performance. The method works well if some cells of a row contain zeros. However, if a row is composed of all zeros, each cell of the row will have same conditional probability.

Data mining techniques have also been explored to detect new malicious executables (Schultz et al., 2001).

2.2.4 Other Approaches. Neural networks are used to learn users' normal behavior and predict the expected behavior of users. Ghosh and Schwartzbard (Ghosh and Schwartzbard, 1999) propose applying a neural network to learn a profile of normality.

Somayaji, Hofmeyr, and Forrest of the University of New Mexico have proposed a method of detecting intrusions that is based on the human immune system (Forrest et al., 1996; Somayaji et al., 1997). The technique first collects a set of reference audits representing the appropriate behavior of the service, and extracts a reference table containing all the known *good* sequences of system calls. These patterns are then used for live monitoring to check whether the sequences generated are listed in the table or not. If they do not, an alarm is generated. Although the immune system approach is interesting and intuitively appealing, so far it has proven to be difficult to apply (Engelhardt, 1997).

Wespi et al. (Wespi et al., 2000) propose a technique to build a table of variable length patterns based on Teiresias algorithm. They claim that, compared to a fixed length approach, the variable length pattern model uses fewer patterns to describe the normal process behavior and achieves better detection. Some research has exploited new techniques to model system and users' normal behavior. Lee et al. (Lee and Xiang, 2001) propose several information theoretic measures to describe the characteristics of an audit data set, and suggest the appropriate anomaly detection models. Each proposed measure could describe different regularities in the dataset.

Wagner et al. (Wagner and Dean, 2001) propose static analysis to automatically derive a model of application behavior. Assuming the system call traces of a program's execution are consistent with the program's source code, the approach first computes a model of expected application behavior, built statically from program source code. At run time, it then monitors the program and checks its system call trace for compliance to the model. Since the proposed models need to include every possible path of system call trace of a program's normal execution, the approach may be not feasible. The run time overhead is high.

3. Data Sources

A useful classification for intrusion detection systems is according to their data source. To a large extent, the data source determines the types of intrusions that can be detected. The two general categories are host-based detection and network-based detection.

For host-based systems, the data source is collected from an individual host on the network. In particular, these systems employ their

host's operating system audit trail as the main source of input. Because host-based systems directly monitor the host data files and operating system processes, they can determine exactly which host resources are the targets of a particular attack.

Given the rapid development of computer networks, some traditional single-host intrusion detection systems have been modified to monitor a number of hosts on a network. They transfer the monitored information from multiple monitored hosts to a central site for processing. These are termed distributed intrusion detection systems. Example distributed systems are IDES (Denning, 1987; Lunt, 1989; Lunt et al., 1992), NSTAT (Kemmerer, 1997), and AAFID (Spafford and Zamboni, 2000).

Network-based intrusion detection employs network traffic as the main source of input. This involves placing a set of traffic sensors within the network. The sensors typically perform local analysis and detection and report suspicious events to a central location. These sensors are generally easier to harden against attack and to hide from attackers, since they perform only the intrusion detection function.

Recent developments in network oriented intrusion detection have moved the focus from network traffic to the computational infrastructure (the hosts and their operating systems) and the communication infrastructure (the network and its protocols). They use the network as just a source of security-relevant information.

Examples of this trend are NSM (Network Security Monitor) (Heberlein et al., 1992), DIDS (Distributed Intrusion Detection System) (Snapp et al., 1991), the JiNao system (Wu et al., 1999), and NADIR (Network Anomaly Detection and Intrusion Reporter) (Hochberg et al., 1993).

Network-based intrusion detection systems have been widened to address large, complex network environments. Examples include GrIDS (Graphbased Intrusion Detection System) (Staniford-Chen et al., 1996), EMERALD (Porras and Neumann, 1997), NetStat (Vigna and Kemmerer, 1998), CARDS (Coordinated Attack Response and Detection System) (Yang et al., 2000), NetRanger (NetRanger, 1999), and RealSecure (Real-Secure, 1999).

As an internal research and development effort, the MITRE Corporation has studied the use of data mining on intrusion detection alarms to reduce the false alarm rate (Staniford-Chen et al., 1996; Clifton and Gengo, 2000). These reports include some interesting statistics on the level of alarms generated by sensors at MITRE (over 1,000,000 alarms per week).

IBM's emergency response service provides real-time intrusion detection (RTID) services through the Internet for a variety of clients. The

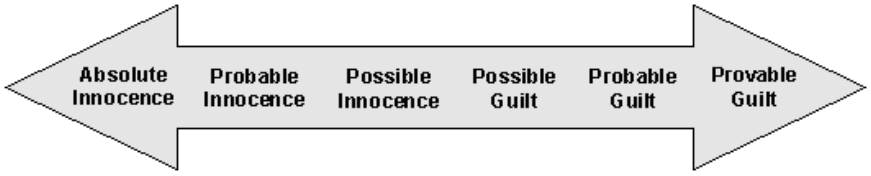

Figure 1. General Degrees of Attack Guilt

emergency response service needs to analyze and respond to thousands of alerts per day. Data mining techniques were used to analyze a database of RTID alerts. They developed profiles of normal alerts and of their clients. Several different types of clients were discovered, each with different alert behaviors and thus different monitoring needs (Manganaris et al., 2000).

4. Degrees of Attack Guilt

In this section, we introduce an interesting way of characterizing intrusion detection activities: degree of attack guilt. This characteristic focuses on how well an intrusion detection system can correctly separate genuine attacks from normal activity. Unlike detection strategy and data source, this characteristic does not apply to the classification of detection systems themselves. Rather, it applies to the classification of network activities, though still with respect to intrusion detection.

In essence, degree of attack guilt is a generalization of detection accuracy. That is, detection accuracy considers whether positive or negative detections are true or false. Degrees of attack guilt consider a continuous spectrum, rather than simply positive or negative detections. This captures something about the degree of confidence of detections, and provides a framework for discussing the costs of improving confidence.

Degree of attack guilt spans the spectrum from absolute innocence to provable guilt, as shown in Figure 1. Provable guilt means that there is no question that the behavior is malicious or unauthorized. Absolute innocence refers to normal, authorized behavior that shows no sign of attack guilt. Actually, absolute innocence is impossible to prove. For example, a user may be involved in activity that is, strictly speaking, authorized and non-malicious. But that same behavior may be part of some subsequent malicious activity.

There is really a continuous spectrum of guilt between the two extremes of absolute innocence and provable guilt. But for general discussion, it is convenient to define a discreet set of degrees of known guilt.

The complete lack of knowledge of guilt falls in the center of the scale, though we do not include it in Figure 1. Moving further from the center of the scale corresponds to increased knowledge of the nature of the behavior, towards either guilt or innocence. We introduce the categories *possible* and *probable* corresponding to increasing levels of known guilt or innocence.

Because degree of attack guilt concerns the confidence associated with detection events, it provides a framework for analyzing cost versus confidence. For example, there is additional cost associated with moving an activity from possible to provable guilt. In intrusion detection, it is often important to understand these types of quality-to-cost tradeoffs.

Moreover, different levels of monitoring may be warranted for particular desired degrees of guilt. That is, different operating environments may have varying needs of detection. For example, highly secure systems may want to investigate all activity that cannot be considered absolutely innocent. Other environments may allow activities to proceed as long as they are probably innocent.

The remainder of this section applies the idea of degree of attack guilt to the various forms of intrusion detection. Just as for Section 2, the discussion is organized by the general detection strategy, either misuse detection or anomaly detection.

4.1. Misuse Detection

This section discusses degrees of attack guilt for misuse detection. Section 2.1 reviews specific systems that perform misuse detection, and is organized by the predominant approach that a system takes. Here the discussion is more general, though the organization is still based on the detection approach. In particular, the various approaches in Section 2.1 are coalesced to two general types: knowledge-based methods and machine-learning methods.

4.1.1 Knowledge-Based Methods. In knowledge-based methods for misuse detection, network or host events are checked against predefined rules or patterns of attack. The goal is to employ representations of known attacks that are general enough to handle actual occurrences of the attacks. Examples of knowledge-based methods are expert systems, signature analysis, and state-transition analysis.

Figure 2 shows degrees of attack guilt for knowledge-based misuse detection. These approaches search for instances of known attacks, by attempting to match with pre-determined attack representations. The search begins as all intrusion detection approaches begin, with a com-

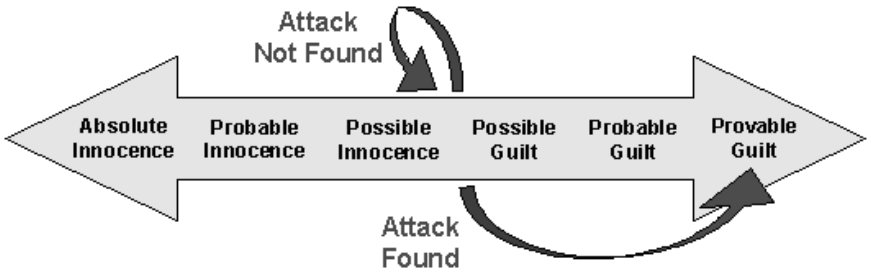

Figure 2. Degrees of Attack Guilt for Knowledge-based Misuse Detection

plete lack of knowledge of any attacker guilt This corresponds to the center of the attack-guilt scale.

But when a particular activity is matched with a known attack, the system gains the knowledge of the attack. For knowledge-based misuse detection, the activity matches a rule or pattern of known guilt. This gained knowledge corresponds to moving to the right of the known-guilt scale. To the extent that the system has a correct attack representation and matching scheme, this method would be expected to provide *provable guilt*.

Such provable guilt is expensive, because it is very time consuming to design general representations for detecting possible variations of specific attacks. Also, a different representation must be created for each type of attack, and there are many possible types of attacks.

For activities that are not matched to a known attack, no real guilt information is provided. All that has been proven is that such activities do not match known attacks. In particular, new attacks or even sufficiently different variations of known attacks are missed. This complete lack of attack knowledge corresponds to remaining in the center of the attack-guilt scale, with no additional cost incurred.

4.1.2 Machine-Learning Methods. In machine-learning methods for misuse detection, patterns and classes of attacks are discovered rather than being pre-defined. Such methods exploit any regularities or strong associations inherent in data such as network traffic. The goal is still to create general representations of attacks, just as for knowledge-based methods. The difference is that the representations are automatically induced, avoiding the costly design of representations in knowledge-based approaches.

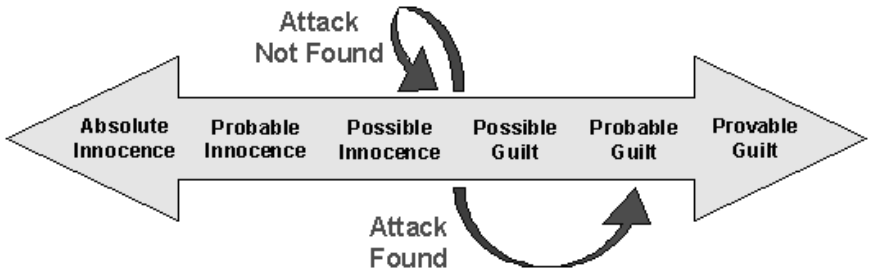

Figure 3. Degrees of Attack Guilt for Machine-learning Misuse Detection

Examples of machine-learning methods are data mining and classification. In facts, these two methods are usually combined in intrusion detection systems. Data mining discovers strong associations among data elements, which often correspond to attack patterns. Classifiers then induce classes of attacks, based on data mining results and other attributes from training data with known attacks.

Figure 3 shows degrees of attack guilt for machine-learning misuse detection. These approaches attempt to classify according to known attacks, by comparing them with previously learned attack representations. Before classification, there is a complete lack of knowledge of any attacker guilt, corresponding to the center of the attack-guilt scale.

During classification, a particular activity may be classified as a known attack. Just as for knowledge-based misuse detection, this corresponds to moving to the right of the known-guilt scale. But because the classification process is inherently a statistical summary that is prone to error, machine-learning methods can move us only as far as *probable guilt*.

Here we see the tradeoff in quality (degree of known guilt) versus cost. It is less expensive to automatically induce attack representations, but the resulting classifications are less reliable (not as provably guilt) compared to handcrafted representations.

In machine-learning approaches to misuse detection, classifiers are only trained for known attacks, not for normal events. That is, the intrusion detection system has attack signatures that have been learned, but no model for normal activity. As such, activity not classified as a known attack really corresponds to remaining in the center of the attack guilt scale (lack of guilt knowledge). In particular, misuse detection based on machine-learning methods has no ability to detect unknown attacks, and will misclassify them as normal behavior.

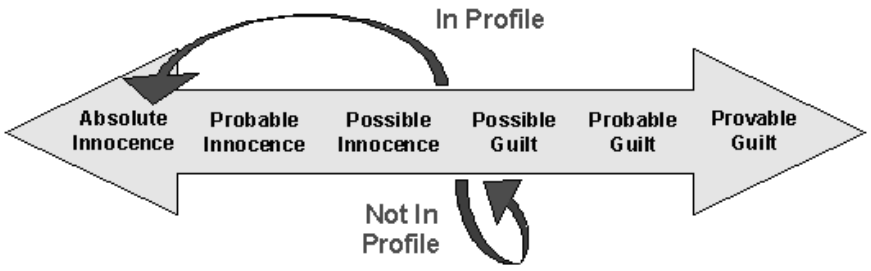

Figure 4. Degrees of Attack Guilt for Knowledge-based Anomaly Detection

4.2. Anomaly Detection

This section discusses degrees of attack guilt for anomaly detection. Section 2.2 reviews specific systems that perform anomaly detection, and is organized by a system's predominant approach. In this section, the discussion is more general, though it still organizes the material based on the approach to detection. In particular, the various approaches in Section 2.2 are coalesced to three general types: knowledge-based methods, statistical methods, and machine- learning methods.

4.2.1 Knowledge-Based Methods. In knowledge-based methods for anomaly detection, network or host activity is checked against pre-defined rules or patterns of normal behavior. The goal is to employ a representation of normal behavior (a profile), from which anomalous behavior is identified as possible attacks. An example knowledge- based method is the expert systems approach.

Figure 4 shows degrees of attack guilt for knowledge-based anomaly detection. These approaches search for instances anomalous behavior, by comparing activities to a pre-determined profile. The search begins with a complete lack of knowledge of any attacker guilt, corresponding to the center of the attack-guilt scale.

When a particular activity is found in the profile, the system gains the knowledge that the activity is not an attack, corresponding to moving to the left of the known-guilt scale. Assuming that that the system has a correct representation in the profile and a correct matching scheme, this method would be expected to prove *absolute innocence*, at least with respect to current activities.

Obtaining such knowledge of absolute innocence is expensive, because it is involves the time consuming crafting of rules to represent normal be-

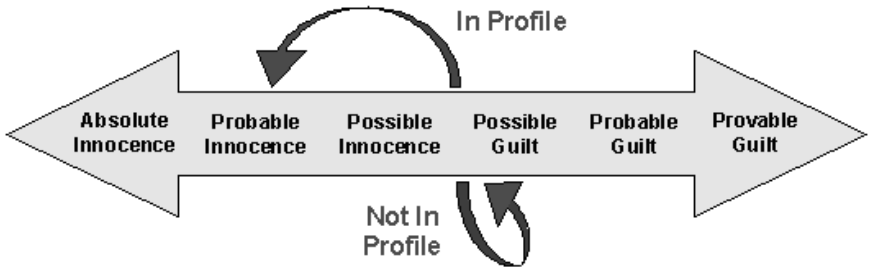

Figure 5. Degrees of Attack Guilt for Statistical Anomaly Detection

havior. This is particularly true when a comprehensive profile of normal behavior is needed that contains many rules.

For activities that are not found in the profile, no real guilt information is provided. All that has been proven is that such activities match known normal behavior. In particular, activities not in the profile may simply represent normal behavior that needs to be added to the profile. This lack of knowledge corresponds to remaining in the center of the attack-guilt scale, with no additional cost incurred.

4.2.2 Statistical Methods. In statistical methods for anomaly detection, profiles of normal behavior are generated by statistical criteria rather than being handcrafted. The goal is still to create representations of normal behavior, just as for knowledge-based methods of anomaly detection. The difference is that the representations are generated automatically as parameters of pre-determined statistical models. This automatic generation avoids the more costly handcrafting of profiles found in knowledge- based approaches.

Figure 5 shows degrees of attack guilt for statistical anomaly detection. This class of approaches attempts to statistically model normal behavior, and compare new behavior to the statistical profile. Before the comparison, there is a complete lack of knowledge of any attacker guilt, corresponding to the center of the attack-guilt scale.

When new behavior is compared to the statistical profile, a particular activity may be associated with normal behavior. Just as for knowledge-based anomaly detection, this corresponds to moving to the right of the known-guilt scale. But because the profile model is statistical, it is prone to error. Thus statistical methods of anomaly detection can move us only as far as *probable innocence*.

Again we see the tradeoff in quality versus cost. It is less expensive to automatically generate statistical profiles, but the resulting models are less reliable compared to profiles with handcrafted rules.

In statistical approaches to anomaly detection, statistical models are only built for normal behavior, not actual attacks. Thus activity not found in the profile really corresponds to remaining in the center of the attack guilt scale, i.e. lack of guilt knowledge. That is, statistical anomaly detection can identify normal behavior not in the profile as a possible attack, leading to higher false-alarm rates.

4.2.3 Machine-Learning Methods. A more recent direction in intrusion detection is the application of machine-learning methods to anomaly detection. This approach is particularly promising for the detection of novel attacks. Example machine-learning methods for anomaly detection include data mining and classification. We consider this approach in more depth with respect to degree of attack guilt.

In machine-learning methods for anomaly detection, profiles of normal behavior are automatically discovered. This automatic discovery avoids the costly handcrafting of profiles (via rules) found in knowledge-based approaches. It also avoids the labor-intensive process of carefully designing the applicable models needed for statistical approaches.

An example machine-learning approach to anomaly detection is the application of association mining to network audit data. This discovers rules that capture the predominant associations among traffic senders and receivers. Assuming that the mining is done initially during known attack-free times, the discovered rules then form the profile against which subsequently mined rules are compared.

A current limitation of this approach is that it has difficulty detecting very stealthy attacks. In other words, it detects attacks that involve a relatively large number of events within a period of time. This is because it sends an alarm only when the number of occurrences of an unexpected rule exceeds a threshold. This limitation is not unique to this approach; most of the anomaly detection models have the same problem (Ning, 2001).

A primary focus of research in anomaly detection is in detecting novel attacks while maintaining sufficiently low numbers of false alarms. This is particularly challenging because anomaly detection in general is prone to higher false-alarm rates. It is hard to define abnormal deviations as attacks if they cannot predictably be distinguished from variations of normal behavior.

An approach to this problem is to employ classifiers that are trained to learn the difference between normal and abnormal deviations from user

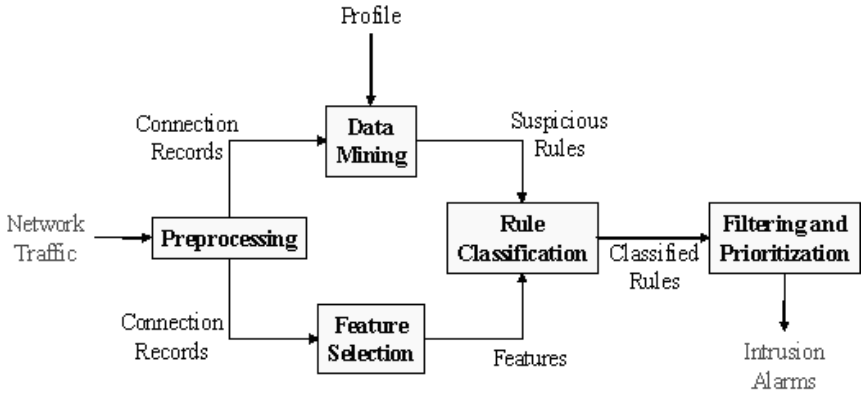

Figure 6. Combining data mining and classification for anomaly detection

profiles. These classifiers sift true intrusions from normal deviations, greatly reducing false alarms.

Figure 6 shows an architecture combining data mining and classification for anomaly detection. An initial training phase builds the profile of normal user behavior, by mining rules from network traffic known to be attack free. Also during training, rule classifiers learn attack classes through network traffic that has been labeled with known attacks. In the case of TCP/IP traffic, a preprocessing module extracts information from packet headers and builds records of TCP connections, which are subsequently mined and classified.

In the detection phase, a dynamic on-line mining algorithm produces suspicious rules. Suspicious rules are those that exceed a particular support level and are missing from the profile of normal behavior. To achieve real-time performance, an incremental mining method can find suspicious rules within a sliding time window, so that datasets need only be scanned once (Wu, 2001b).

In principle, anomaly detection has the ability to detect novel attacks. But in practice this is far from easy, since detecting attacks for which no knowledge is available may seem an ill-posed problem. But attacks can be seen as violations of normal activity whose behavior can be estimated from the existing information.

A pseudo-Bayesian approach can enhance the ability of classifiers to detect new attacks while keeping the false-alarm rate low (Barbara et al., 2001). Pseudo-Bayes estimators estimate the prior and posterior probabilities of new attacks. A naive Bayes classifier can then classify instances as normal, known attacks, or new attacks. One advantage of pseudo-

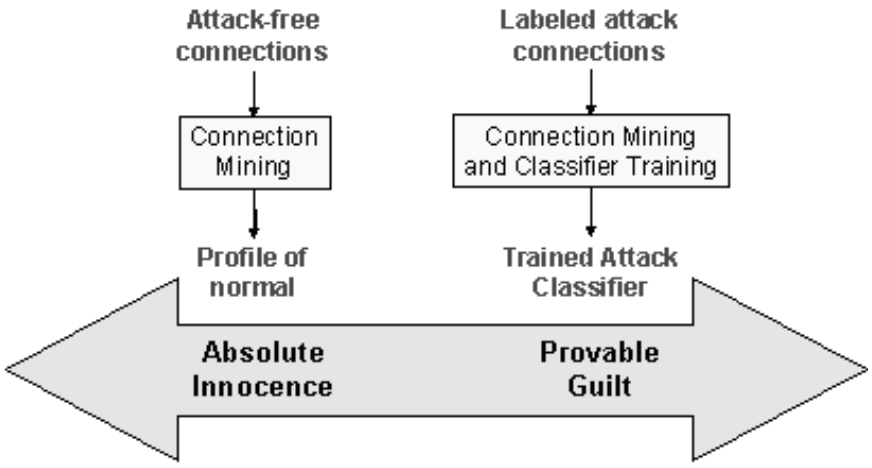

Figure 7. Degrees of Guilt for the Training Phase

Bayes estimators is that no knowledge about new attacks is needed, since the estimated probabilities of new attacks are derived from the information of normal instances and known attacks.

In the training phase, the degrees of guilt are perfectly known, as illustrated in Figure 7 Network connections known to be guilt free are mined, and the discovered connection rules form the profile of normal behavior. Network connections with known attack labels are then mined for connection rules, which are used to train the attack classifiers.

At the beginning of the detection phase, the degrees of guilt for monitored network connections are completely unknown. This corresponds to the center of the known-guilt scale. The connection association mining process then discards mined connection rules that are already in the profile, and retains those not in the profile for further classification. This corresponds to moving from completely unknown guilt to innocence beyond suspicion for rules already in the profile, or to possible guilt for rules not in the profile. This process is illustrated in Figure 8.

Next, mined connection rules not found in the profile are submitted to the rule classification engine. This corresponds to moving from possible guilt to either probable or possible innocence (for rules in the profile), or to probable or possible guilt (for rules not in the profile). This is shown in Figure 9.

The potential ambiguity in degrees of guilt in Figure 9 arises because of the nature of the attack classifiers. The classifiers undergo supervised training, in which class decision surfaces are induced that minimize some

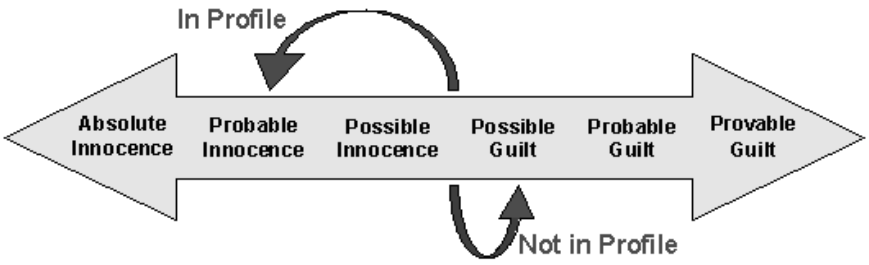

Figure 8. Degrees of Guilt for Connection Mining during Detection

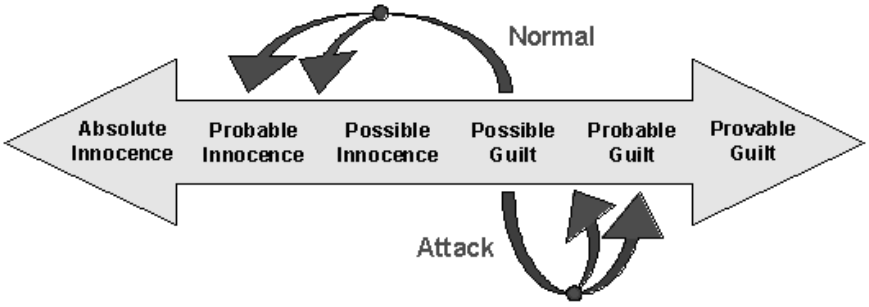

Figure 9. Degrees of Guilt for Attack Classification during Detection

statistical measure of misclassification based on training data. Because of the statistical nature of the classifiers and their dependence on the quality of training data, a question of classification confidence arises.

Figure 10 illustrates possible confidence scenarios for the attack classifiers. For connection rules relatively far from the decision surface, the probability of misclassification is small. For these rules, we could thus say that either innocence or guilt (respectively) is very probable. But connection rules relatively close to the decision surface have a higher chance of being misclassified. Innocence or guilt is then less probable.

One possible classifier is the naive Bayes classifier with pseudo-Bayes estimators. This classifier includes a class for unknown attacks, which is missing from the supervised classifiers. This class handles connection rules that cannot be classified either normal or known attacks. The Bayes classifier avoids the misclassification of unknown attacks (as either normal or known attacks) that plague supervised classifiers.

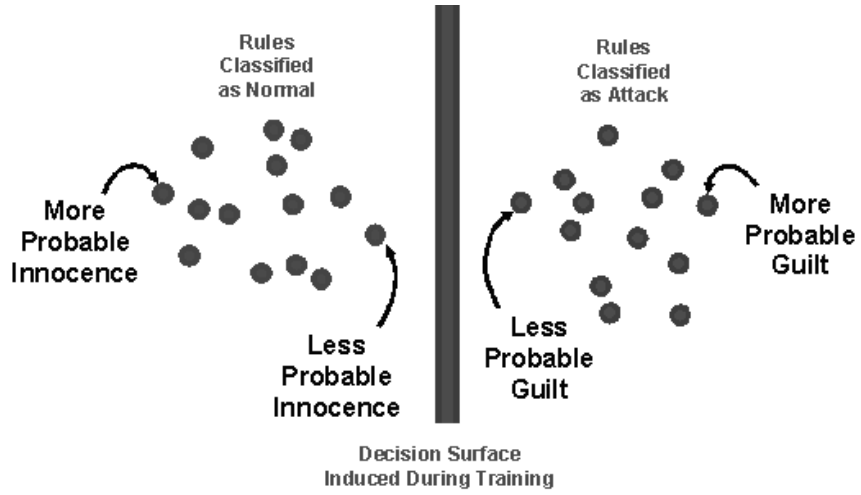

Figure 10. Degrees of Guilt with respect to Attack Classification Confidence

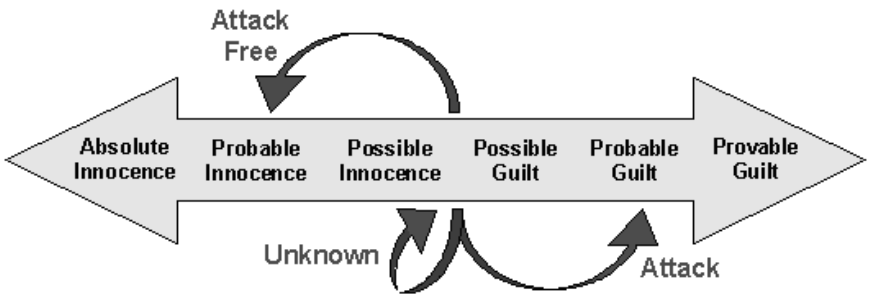

Figure 11. Degrees of Guilt for Overall Machine-learning Approach

Connection rules classified as unknown remain in the center of the degree-of-guilt scale. That is, such a rule is known to belong to neither the normal class nor a known attack class. At this point we have no knowledge of whether it is an unknown attack or anomalous attack-free behavior. But at least is has not been misclassified as normal or known attack, and can be further investigated to determine its true nature.

Figure 11 shows the overall degrees of guilt for this machine-learning approach to anomaly detection. Initially, there is no knowledge of the state of network intrusion. This complete lack of knowledge corresponds to the center of the guilt scale. Connection rules either found in the

profile or classified as normal generate no alarms. This corresponds to moving from the center of the scale to probable innocence. Connection rules classified as known attacks correspond to moving from the center of the scale to probable guilt. Rules classified as unknown remain in the center of the scale, pending further investigation.

5. Conclusion

In this chapter, we examine the state of modern intrusion detection. The discussion follows two well-known criteria for categorizing intrusion detection systems: detection strategy and data source. The general detection strategies are misuse detection and anomaly detection, and data source categories are host-based and network-based. We introduce degree of attack guilt as an interesting way of characterizing intrusion detection activities. It provides a framework in which we analyze detection quality versus cost.

Intrusion detection systems have been an area of active research for over 15 years. Current commercial intrusion detection systems employ misuse detection. As such, they completely lack the ability to detect new attacks. The absence of this capability is a recognized gap in current systems.

Given the shortcomings of current commercial systems, an important research focus is anomaly detection through machine learning, particularly through data mining. A critical issue for anomaly detection is the need to reduce false alarms, since any activity outside a known profile raises an alarm. Research prototypes combining data mining and classification have shown great promise in this area.

Acknowledgments

This work was partially supported by the National Science Foundation under the grant CCR-0113515. We thank Sushil Jajodia and Ningning Wu for sharing their thoughts during various stages of this chapter.

References

Abraham, T. (2001). Iddm: Intrusion detection using data mining techniques. Technical Report DSTO-GD-0286, DSTO Electronics and Surveillance Research Laboratory.

Allen, J., Christie, A., Fithen, W., McHugh, J., Pickel, J., and Stoner, E. (2000). State of the practice of intrusion detection technologies. Technical Report CMU/SEI-99-TR-028, Software Engineering Institute, CMU, Pittsburgh, PA.

Anderson, D., Lunt, T. F., Javitz, H., Tamaru, A., and Valdes, A. (1995a). Detecting unusual program behavior using the statistical component of the next-generation

intrusion detection expert system (nides). Technical Report SRI-CSL-95-06, SRI International, Menlo Park, CA.

Anderson, D., Lunt, T. F., Javitz, H., Tamaru, A., and Valdes, A. (1995b). Detecting unusual program behavior using the statistical component of the next-generation intrusion detection expert system (nides). Technical Report SRI-CSL-95-06, Computer Science Laboratory, SRI International, Menlo Park, CA.

Axelsson, S. (1999). Research in intrusion-detection systems: A survey. Technical Report TR: 98-17, Department of Computer Engineering, Chalmers University of Technology, Goteborg, Sweden.

Axelsson, S. (2000a). The base-rate fallacy and the difficulty of intrusion detection. *ACM Transactions on Information and System Security*, 3(1):186–205.

Axelsson, S. (2000b). Intrusion detection systems: A survey and taxonomy. Technical report, Department of Computer Engineering, Chalmers University of Technology, Goteborg, Sweden.

Barbara, D., Jajodia, S., Wu, N., and Speegle, B. (1999). Mining unexpected rules in network audit trails. Technical report, George Mason University.

Barbara, D., Wu, N., and Jajodia, S. (2001). Detecting novel network intrusions using bayes estimators. In *First SIAM Conference on Data Mining*, Chicago, IL. Society for Industrial and Applied Mathematics.

Bauer, D. S. and Koblentz, M. E. (1988). Nidx-an expert system for real-time. In *Computer Networking Symposium*.

Cabrera, J. B. D., Ravichandran, B., and Mehra, R. K. (2000). Statistical traffic modeling for network intrusion detection. In *8th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, San Francisco, CA.

Clifton, C. and Gengo, G. (2000). Developing custom intrusion detection filters using data mining. In *21st Century Military Communications Conference*, volume 1, pages 440–443. IEEE Computer Society.

Corporation, C. T. (2000). Best of breed appendices. 0017-UU-TE-000712.

Crosbie, M., Dole, B., Ellis, T., Krsul, I., and Spafford, E. (1996). *IDIOT Users Guide*. Purdue University, West Lafayette, IN. TR-96-050.

Denning, D. E. (1987). An intrusion-detection model. *IEEE Transactions on Software Engineering*, 13:222–232.

Dowell, C. and Ramstedt, P. (1990). The computerwatch data reduction tool. In *13th National Computer Security Conference*, Washington, DC.

Engelhardt, D. (1997). Directions for intrusion detection and response: A survey. Technical Report DSTO-GD-0155, DSTO Electronics and Surveillance Research Laboratory.

Esmaili, M., Balachandran, B., Safavi-Naini, R., and Pieprzyk, J. (1996). Case-based reasoning for intrusion detection. In *12th Annual Computer Security Applications Conference*, San Diego, CA.

Esmaili, M., Safavi-Naini, R., and Balachandran, B. M. (1997). Autoguard: A continuous case-based intrusion detection system. In *Twentieth Australasian Computer Science Conference*.

Forrest, S., Hofmeyr, S., Somayaji, A., and Longstaff, T. (1996). A sense of self for unix processes. In *IEEE Symposium on Security and Privacy*, pages 120–128, Oakland, CA. IEEE Computer Society.

Ghosh, A. K. and Schwartzbard, A. (1999). A study in using neural networks for anomaly and misuse detection. In *Usenix Security Symposium*, Washington, DC.

Heberlein, L. T., Mukherjee, B., and Levitt, K. N. (1992). Internet security monitor: An intrusion detection system for large-scale networks. In *15th National Computer Security Conference*, Baltimore, MD.

Helmer, G., Wong, J., Honavar, V., and Miller, L. (1999). Automated discovery of concise predictive rules for intrusion detection. Technical Report TR 99-01, Department of Computer Science, Iowa State University, Ames, IA.

Hochberg, J., Jackson, K., Stallings, C., McClary, J., DuBois, D., and Ford, J. (1993). Nadir: An automated system for detecting network intrusions and misuse. *Computers and Security*, 12(3):248–253.

Ilgun, K. (1992). *USTAT A Real-time Intrusion Detection System for UNIX*. Master of science, University of California Santa Barbara.

Jackson, K. A. (1999). Intrusion detection system (ids) product survey. Technical Report LA-UR-99-3883, Los Alamos National Laboratory, Los Alamos, NM.

Javitz, H. S. and Valdes, A. (1991). The sri ides statistical anomaly detector. In *IEEE Symposium on Research in Security and Privacy*, Oakland, CA.

Jensen, K. (1997). A brief introduction to coloured petri nets. Technical report, presented at Tools and Algorithms for the Construction and Analysis of Systems (TACAS) Workshop, Enschede, The Netherlands.

Kemmerer, R. A. (1997). Nstat: A model-based real-time network intrusion detection system. Technical Report TR 1997-18, University of California Santa Barbara Department of Computer Science.

Kohavi, R., Beckeer, B., and Sommerfield, D. (1997). Improving simple bayes. In *European Conference on Machine Learning*, Prague, Czech Republic.

Kvarnstrom, H. (1999). A survey of commercial tools for intrusion detection. Technical Report TR 99-8, Department of Computer Engineering, Chalmers University of Technology, Goteborg, Sweden.

Lane, T. D. (2000). *Machine Learning Techniques for the Computer Security Domain of Anomaly Detection*. Doctor of philosophy, Purdue University.

LaPadula, L. J. (1999). State of the art in anomaly detection and reaction. Technical Report MP 99B0000020, The MITRE Corporation, Bedford, MA.

LaPadula, L. J. (2000). Compendium of anomaly detection and reaction tools and projects. Technical Report MP 99B0000018R1, The MITRE Corporation, Bedford, MA.

Lee, W. (1999). A data mining framework for constructing features and models for intrusion detection systems. Technical report, Graduate School of Arts and Sciences, Columbia University.

Lee, W., Stolfo, S., and Mok, K. (2000). Adaptive intrusion detection: a data mining approach. *Artificial Intelligence Review*, 14:533–567.

Lee, W. and Stolfo, S. J. (1998). Data mining approaches for intrusion detection. In *Proceedings of the 7th USENIX Security Symposium*, San Antonio, TX.

Lee, W., Stolfo, S. J., and Mok, K. W. (1999). A data mining framework for building intrusion detection models. In *IEEE Symposium on Security and Privacy*.

Lee, W. and Xiang, D. (2001). Information-theoretic measures for anomaly detection. In *IEEE Symposium on Security and Privacy*, pages 130–143, Oakland, CA. IEEE Computer Society.

Liepins, G. and Vaccaro, H. (1989). Anomaly detection purpose and framework. In *12th National Computer Security Conference*, pages 495–504, Baltimore, MD. NIST and NSA.

Liepins, G. E. and Vaccaro, H. S. (1992). Intrusion detection: It's role and validation. *Computers and Security*, pages 347–355.

Lin, J.-L., Wang, X. S., and Jajodia, S. (1998). Abstraction-based misuse detection: High-level specifications and adaptable strategies. In *11th IEEE Computer Security Foundations Workshop*.

Lindqvist, U. and Porras, P. A. (1999). Detecting computer and network misuse through the production-based expert system toolset (p-best). In *IEEE Symposium on Security and Privacy*.

Lippmann, R. P., Fried, D. J., Graf, I., J. W. Haines, K. R. K., D., McClung, D. Weber, S. E. W., Wyschogrod, D., Cunningham, R. K., , M., and Zissman, A. (2000). Evaluating intrusion detection systems: the 1998 darpa off-line intrusion detection evaluation. In *DARPA Information Survivability Conference and Exposition*.

Lundin, E. and Jonsson, E. (1999). Some practical and fundamental problems with anomaly detection. In *Proceedings of the Nordic Workshop on Secure Computer Systems*.

Lunt, T., Tamaru, A., Gilham, F., Jagannathan, R., Jalali, C., Neumann, P. G., Javitz, H. S., Valdes, A., and Garvey, T. D. (1992). A real time intrusion detection expert system (ides). Technical report, SRI.

Lunt, T. F. (1989). Real-time intrusion detection. In *presented at COMPCON: Thirty-Fourth IEEE Computer Society International Conference: Intellectual Leverage*.

Manganaris, S., Christensen, M., Zerkle, D., and Hermiz, K. (2000). A data mining analysis of rtid alarms. *Computer Networks*, 34(No. 4):571–577.

NetRanger (1999). *NetRanger*. at www.cisco.com/univercd/cc/td/doc/product/iaabu/netrangr.

Neumann, P. G. and Porras, P. A. (1999). Experience with emerald to date. In *First Usenix Workshop on Intrusion Detection and Network Monitoring*, Santa Clara, CA.

Ning, P. (2001). *Abstraction-based Intrusion Detection in Distributed Environments*. Doctor of philosophy, George Mason University.

Porras, P. (1992). *STAT: A State Transition Analysis for Intrusion Detection*. Master of science, University of California Santa Barbara.

Porras, P. A. and Kemmerer, R. A. (1992). Penetration state transition analysis: A rule-based intrusion detection approach. In *Eighth Annual Computer Security Applications Conference*.

Porras, P. A. and Neumann, P. G. (1997). Emerald: Event monitoring enabling responses to anomalous live disturbances. In *Proceedings of the 20th National Information Systems Security Conference*, Baltimore, MD.

Real-Secure (1999). *RealSecure*. http://www.iss.net/customer_care/resource_center/product_lit/, Internet Security Systems.

Schultz, M. G., Eskin, E., Zadok, E., and Stolfo, S. J. (2001). Data mining methods for detection of new malicious executables. In *IEEE Symposium on Security and Privacy*, Oakland, CA. IEEE Computer Society.

Smaha, S. E. (1988). Haystack: An intrusion detection system. In *Fourth Aerospace Computer Security Applications Conference*.

Snapp, S., Brentano, J., Dias, G., Goan, T., Grance, T., Heberlein, L., Ho, C.-L., Levitt, K. N., Mukherjee, B., Mansur, D. L., Pon, K. L., and Smaha, S. E. (1991).

A system for distributed intrusion detection. In *Compcon Spring*, pages 170–176. IEEE Computer Society.

Somayaji, A., Hofmeyr, S., and Forrest, S. (1997). Principles of a computer immune system. In *New Security Paradigms Workshop*, Langdale, Cumbria UK.

Spafford, E. H. and Zamboni, D. (2000). Intrusion detection using autonomous agents. *Computer Networks*, 34(4):547–570.

Staniford-Chen, S., Cheung, S., Crawford, R., Dilger, M., Frank, J., Hoagland, J., Levitt, K., Wee, C., Yip, R., and Zerkle, D. (1996). Grids-a graph based intrusion detection system for large networks. In *19th National Information Systems Security Conference*, pages 361–370, Baltimore, MD. NIST and NSA.

Vaccaro, H. and Liepins, G. (1989). Detection of anomalous computer session activity. In *IEEE Symposium on Security and Privacy*. IEEE Computer Society.

Valdes, A. and Skinner, K. (2000). Adaptive, model-based monitoring for cyber attack detection. In *Recent Advances in Intrusion Detection*, pages 80–93, Toulouse, France. Springer-Verlag.

Vigna, G. and Kemmerer, R. A. (1998). Netstat: A network-based intrusion detection approach. In *Proceedings of the International Conference on Knowledge and Data Mining*, New York, NY.

W. Lee, S. J. S. and Mok, K. W. (1998). Mining audit data to build intrusion detection models. In *Proceedings of the International Conference on Knowledge and Data Mining*, New York, NY.

Wagner, D. and Dean, R. (2001). Intrusion detection via static analysis. In *IEEE Symposium on Security and Privacy*. IEEE Computer Society.

Wespi, A., Dacier, M., and Debara, H. (2000). Intrusion detection using variable-length audit trail patterns. In *Recent Advances in Intrusion Detection*, pages 110–129, Toulouse, FR. Springer-Verlag.

Winkler, J. R. (1990). A unix prototype for intrusion and anomaly detection in secure networks. In *13th National Computer Security Conference*, Washington, DC.

Winkler, J. R. and Landry, L. C. (1992). Intrusion and anomaly detection, isoa update. In *15th National Computer Security Conference*, Baltimore, MD.

Wu, N. (2001a). *Audit Data Analysis and Mining*. PhD thesis, George Mason University, Department of Information and Software Engineering. Fairfax, VA.

Wu, N. (2001b). Research statement.

Wu, S. F., Chang, H., Jou, F., Wang, F., Gong, F., Sargor, C., Qu, D., and Cleaveland, R. (1999). Jinao: Design and implementation of a scalable intrusion detection system for the ospf routing protocol.

Yang, J., Ning, P., Wang, X. S., and Jajodia, S. (2000). Cards: A distributed system for detecting coordinated attacks. In *IFIP TC11 16th Annual Working Conference on Information Security*, pages 171–180. Kluwer.