An Introduction to Intrusion-Detection Systems

Hervé Debar
IBM Research, Zurich Research Laboratory,
Säumerstrasse 4, CH–8803 Rüschlikon, Switzerland
deb@zurich.ibm.com

Abstract

Intrusion-detection systems aim at detecting attacks against computer systems and networks or, in general, against information systems. Indeed, it is difficult to provide provably secure information systems and to maintain them in such a secure state during their lifetime and utilization. Sometimes, legacy or operational constraints do not even allow the definition of a fully secure information system. Therefore, intrusion–detection systems have the task of monitoring the usage of such systems to detect any apparition of insecure states. They detect attempts and active misuse either by legitimate users of the information systems or by external parties to abuse their privileges or exploit security vulnerabilities. This paper is the first in a two-part series; it introduces the concepts used in intrusion–detection systems around a taxonomy.

1 Introduction

Since the seminal work by Denning in 1981 [11], many intrusion-detection prototypes have been created. Sobirey maintains a partial list of 59 of them [52]. Intrusion-detection systems have emerged in the computer security area because of the difficulty of ensuring that an information system will be free of security flaws. Indeed, a taxonomy of security flaws by Landwehr *et al.* [36] shows that computer systems suffer from security vulnerabilities regardless of their purpose, manufacturer, or origin, and that it is technically difficult as well as economically costly (in terms of both building and maintaining such a system) to ensure that computer systems and networks are not susceptible to attacks.

2 Description of a generic intrusion-detection system

2.1 Terminology

The term *system*(a.k.a. *target system*) is used here to denote the information system being monitored by the intrusion-detection system. It can be a workstation,a network element,a server,a mainframe,a firewall,a web server, etc.

The term *audit* denotes information provided by a system concerning its inner workings and behavior. Examples of audits include, but are not limited to, C2 audit trail, accounting, and syslog in the UNIX world, Syslog in the MVS world, event log in Windows NT, and incident tickets in X25 networks. A description of some of these audits is given in Section 5.

2.2 Description

An intrusion-detection system acquires information about an information system to perform a diagnosis on the security status of the latter. The goal is to discover breaches of security, attempted breaches, or open vulnerabilities that could lead to potential breaches. A typical intrusion-detection system is shown in Figure 1.

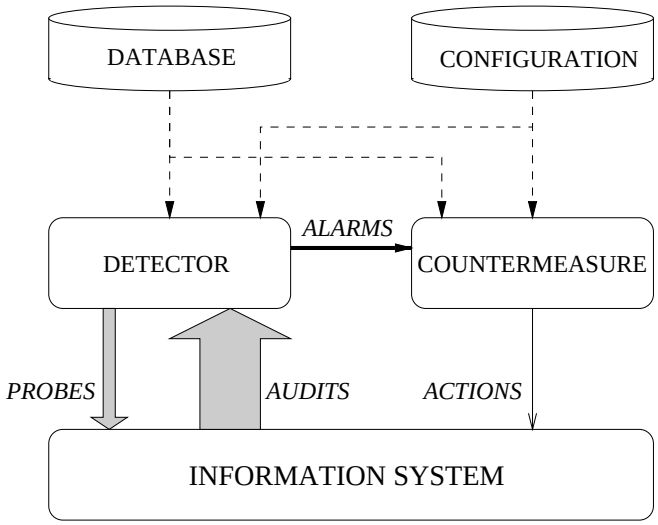

Figure 1: Very simple intrusion-detection system.

An intrusion-detection system can be described at a very macroscopic level as a detector that processes information coming from the system to be protected (Fig. 1). This detector can also launch probes to trigger the audit process, such as requesting version numbers for applications. It uses three kinds of information: long-term information related to the technique used to detect intrusions (a knowledge base of attacks, for example), configuration information about the current state of the system, and audit information describing the events that are happening on the system.

The role of the detector is to eliminate unneeded information from the audit trail. It then presents either a synthetic view of the security-related actions taken during normal usage of the system, or a synthetic view of the current security state of the system. A decision is then taken to evaluate the probability that these actions or this state can be considered as symptoms of an intrusion or vulnerabilities. A countermeasure component can then take corrective action to either prevent the actions from being executed or change the state of the system back to a secure state.

2.3 Efficiency of intrusion-detection systems

To evaluate the efficiency of an intrusion-detection system, Porras and Valdes [42] have proposed the following three parameters:

Accuracy. Accuracy deals with the proper detection of attacks and the absence of false alarms. Inaccuracy occurs when an intrusion-detection system flags a legitimate action in the environment as anomalous or intrusive.

Performance. The performance of an intrusion-detection system is the rate at which audit events are processed. If the performance of the intrusion-detection system is poor, then real-time detection is not possible.

Completeness. Completeness is the property of an intrusion-detection system to detect all attacks. Incompleteness occurs when the intrusion-detection system fails to detect an attack. This measure is much more difficult to evaluate than the others because it is impossible to have a global knowledge about attacks or abuses of privileges.

Let us introduce two additional properties:

Fault tolerance. An intrusion-detection system should itself be resistant to attacks, especially denial-of-service-type attacks, and should be designed with this goal in mind. This is particularly important because most intrusion-detection systems run above commercially available operating systems or hardware, which are known to be vulnerable to attacks.

Timeliness. An intrusion-detection system has to perform and propagate its analysis as quickly as possible to enable the security officer to react before much damage has been done, and also to prevent the attacker from subverting the audit source or the intrusion-detection system itself. This implies more than the measure of performance because it not only encompasses the intrinsic processing speed of the intrusion-detection system, but also the time required to propagate the information and react to it.

3 Taxonomy elements

We introduce three concepts to classify intrusion-detection systems, which are summarized in Fig. 2.

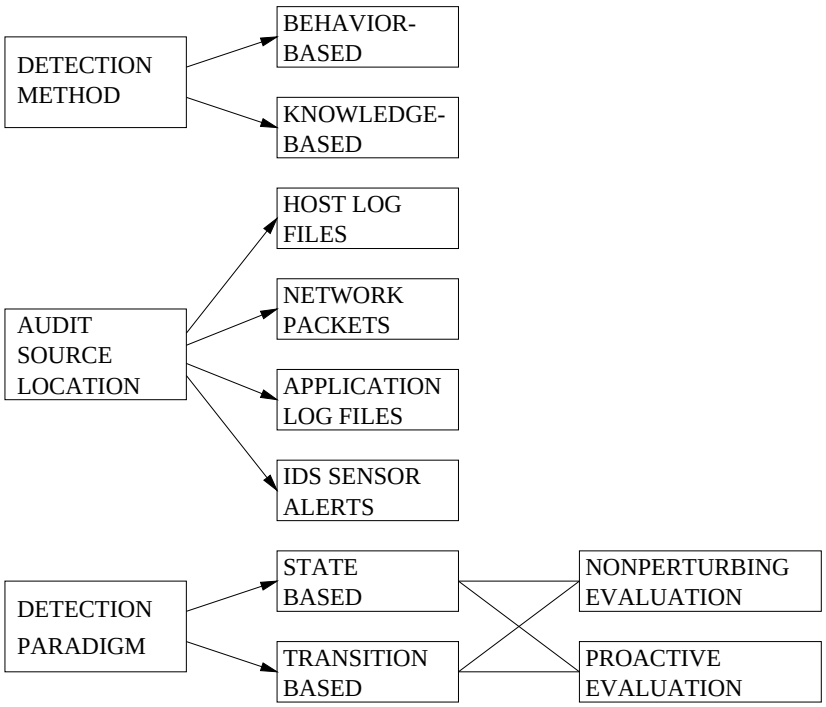

Figure 2: Characteristics of intrusion-detection systems.

The *detection method* describes the characteristics of the analyzer. When the intrusion-detection system uses information about the normal behavior of the system it monitors, we qualify it as behavior-based. When the intrusion-detection system uses information about the attacks, we qualify it as knowledge-based.

The *behavior on detection* describes the response of the intrusion-detection system to attacks. When it actively reacts to the attack by taking either corrective (closing holes) or pro-active (logging out possible attackers, closing down services) actions, then the intrusion-detection system is said to be active. If the intrusion-detection system merely generates alarms (such as paging), it is said to be passive.

The *audit source location* discriminates intrusion-detection systems based on the kind of input information they analyze. This input information can be audit trails (a.k.a. system logs) on a host, network packets, application logs, or intrusion-detection alerts generated by other intrusion-detection systems.

The *detection paradigm* describes the detection mechanism used by the intrusion-detection system. Intrusion-detection systems can evaluate states (secure or insecure) or transitions (from secure to insecure).

In addition, this evaluation can be performed in a nonobtrusive way or by actively stimulating the system to obtain a response.

4 Knowledge-based versus behavior-based intrusion detection

There are two complementary trends in intrusion detection: (1) to use the knowledge accumulated about attacks and look for evidence of the exploitation of these attacks, and (2) to build a reference model of the usual behavior of the information system being monitored and look for deviations from the observed usage. The first trend is often referred to as *misuse detection* [29, 34], but also as detection by appearance [53]. The second trend is referred to as anomaly detection [29] or detection by behavior [53]. In this paper, we use the term *knowledge-based* intrusion detection for the first trend, as we feel it describes the technique being used more accurately. The second trend is characterized by the term *behavior-based* intrusion detection. Both terms are more extensively defined hereafter.

4.1 Knowledge-based intrusion detection

Knowledge-based intrusion-detection techniques apply the knowledge accumulated about specific attacks and system vulnerabilities. The intrusion-detection system contains information about these vulnerabilities and looks for *attempts* to exploit them. When such an attempt is detected, an alarm is raised. In other words, any action that is not explicitly recognized as an attack is considered acceptable. Therefore, the *accuracy* of knowledge-based intrusion-detection systems is considered good. However, their *completeness* depends on the regular update of knowledge about attacks.

Advantages of the knowledge-based approaches are that they have, in theory, very low false-alarm rates, and that the contextual analysis proposed by the intrusion-detection system is detailed, making it easier for the security officer using this intrusion-detection system to understand the problem and to take preventive or corrective action.

Drawbacks include the difficulty of gathering the required information on the known attacks and keeping it up to date with new vulnerabilities and environments. Maintenance of the knowledge base of the intrusion-detection system requires careful analysis of each vulnerability and is therefore a time-consuming task. Knowledge-based approaches also have to face the generalization issue. Knowledge about attacks strongly depends on the operating system, version, platform, and application. The resulting intrusion-detection system is therefore closely tied to a given environment. Also, detection of insider attacks involving an abuse of privileges is deemed more difficult because no vulnerability is actually exploited by the attacker.

4.1.1 Expert systems

Expert systems [37] are primarily used by knowledge-based intrusion-detection techniques. The expert system contains a set of rules that describe attacks. Audit events are then translated into facts carrying their semantic signification in the expert system, and the inference engine draws conclusions using these rules and facts. *This method increases the abstraction level of the audit data by attaching a semantic to it.*

Rule-based languages [21] are a natural tool for modeling the knowledge that experts have collected about attacks. This approach allows a systematic browsing of the audit trail in search of evidence of attempts to exploit known vulnerabilities. They are also used to verify the proper application of the security policy of an organization.

Model-based reasoning, which also uses expert systems but has additional properties, has been introduced by Garvey and Lunt [19]. The knowledge about the behavior of an attacker is described by the attacker's goals, the actions he takes to reach these goals, and his usage of the system which sometimes reveals a certain level of paranoia. The tool then scans the audits for evidence of these actions and transitions.

This approach of using rule-based languages has shown limitations in the following areas:

Knowledge engineering (related to the *completeness* issue):
It is difficult to extract knowledge about attacks. It is even more difficult to translate this knowledge into production rules using audits as inputs. Sometimes the information required is not available in

the audits. Also, there may be many ways to exploit a given vulnerability, thus leading to as many rules.

Processing speed (related to the *performance* issue):
Use of an expert system shell requires that all audits be imported into the shell as facts, and only then can reasoning take place. Even though some expert system tools allow compilation of rules, the overall performance of the tool often remains low.

Owing to the processing speed issue, expert system shells are used only in prototypes, as commercial products have chosen more efficient approaches.

4.1.2 Signature analysis

Signature analysis follows exactly the same knowledge-acquisition approach as expert systems do, but the knowledge is exploited in a different way. The semantic description of the attacks is transformed into information that can be found in the audit trail in a straightforward way. For example, attack scenarios might be translated into the sequences of audit events they generate or into patterns of data that can be sought in the audit trail generated by the system. *This method decreases the semantic level of the attack descriptions.*

This technique allows a very efficient implementation and is therefore applied in commercial intrusion-detection products [23, 28, 59]. The main drawback of this technique—common to all knowledge-based approaches—is the need for frequent updates to keep up with the stream of new vulnerabilities discovered. This situation is aggravated by the requirement to represent all possible facets of the attacks by signatures. This leads to an attack being represented by a number of signatures, at least one for each operating system to which the intrusion-detection system has been ported.

4.2 Behavior-based intrusion detection

Behavior-based intrusion-detection techniques assume that an intrusion can be detected by observing a deviation from the normal or expected behavior of the system or the users. The model of normal or valid behavior is extracted from reference information collected by various means. The intrusion-detection system later compares this model with the current activity. When a deviation is observed, an alarm is generated. In other words, anything that does not correspond to a previously learned behavior is considered intrusive. Therefore, the intrusion-detection system might be *complete*, but its *accuracy* is a difficult issue.

Advantages of behavior-based approaches are that they can detect attempts to exploit new and unforeseen vulnerabilities. They can even contribute to the (partially) automatic discovery of these new attacks. They are less dependent on operating-system-specific mechanisms. They also help detect "abuse-of-privilege"-type attacks that do not actually involve exploiting any security vulnerability.

The high false-alarm rate is generally cited as the main drawback of behavior-based techniques because not the entire scope of the behavior of an information system may be covered during the learning phase. Also, behavior can change over time, introducing the need for periodic on-line retraining of the behavior profile, resulting either in unavailability of the intrusion-detection system or in additional false alarms. The information system can undergo attacks at the same time the intrusion-detection system is learning the behavior. As a result, the behavior profile will contain intrusive behavior, which is not detected as anomalous.

4.2.1 Statistics

The most widely used tool to build behavior-based intrusion-detection systems is statistics [25, 26, 31]. The user or system behavior is measured by a number of variables sampled over time. Examples of these variables include the login and logout time of each session, the resource duration, and the amount of processor-memory-disk resources consumed during the session. The time sampling period ranges from very short (a few minutes) to long (one month or more).

The original model keeps averages of all these variables and detects whether thresholds are exceeded based on the standard deviation of the variable. Actually, this model is too simple to represent the data

faithfully. Even comparing the variables of individual users with aggregated group statistics fails to yield much improvement. Therefore, a more complex model has been developed [30, 31] that compares profiles of long-term and short-term user activities. The profiles are regularly updated as the behavior of users evolves. This statistical model is now used in a number of intrusion-detection systems and prototypes.

4.2.2 Expert systems

Expert systems have also been used for behavior-based intrusion detection. Examples of two approaches that have been pursued in this area are

- Wisdom & Sense [57], an intrusion-detection system that detects statistical anomalies in the behavior of users. The tool first builds a set of rules that statistically describe the behavior of the users based on recordings of their activities over a given period of time. Current activity is then matched against these rules to detect inconsistent behavior. The rule base is rebuilt regularly to accommodate new usage patterns.
- AT&T's ComputerWatch [13], a tool delivered with AT&T's UNIX/MLS multilevel security operating system. This tool checks the actions of users according to a set of rules that describe proper usage policy, and flags any action that does not fit the acceptable patterns.

This approach is useful for policy-based usage profiles, but is less efficient than the statistical approach for processing large amounts of audit information.

4.2.3 Neural networks

Neural networks are algorithmic techniques used to first learn the relationship between the two sets of information, and then "generalize" to obtain new input-output pairs in a reasonable way. Neural networks could theoretically be used in knowledge-based intrusion-detection systems to identify the attacks and seek them in the audit stream. However, as there is currently no reliable way to understand what triggered the association, the neural network cannot explain the reasoning that led to the identification of the attack.

In intrusion detection, neural networks have mainly been used to learn the behavior of actors in the system (e.g. users, daemons). Some equivalence between neural network models and statistics has been demonstrated [18, 48]. The advantage of using neural networks rather than statistics lies in having a simple way to express nonlinear relationships between variables, and in learning/retraining the neural network automatically. Experiments have been performed using a neural network to predict the behavior of users [8]. These experiments have shown that the behavior of UNIX root users is extremely predictable (owing to the very regular activity generated by automatic system actions, daemons, etc.), that the behavior of most users is also predictable, and that there is only a very small fraction of users whose behavior is unpredictable. However, neural networks are still a computationally intensive technique, and are not widely used by the intrusion-detection community.

4.2.4 Computer immunology

Computer immunology has been described by Forrest *et al.* [17]. This technique attempts to build a model of normal behavior of the UNIX network services, rather than of the behavior of users. This model consists of short sequences of system calls made by the processes. Attacks that exploit flaws in the code are likely to go through execution paths not usually taken. The tool first collects a set of reference audits, which represents 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; if not, the intrusion-detection system generates an alarm.

This technique has a potentially very low false-alarm rate if the reference table is sufficiently exhaustive. Extensions to reach that goal are currently being developed [9, 10]. One drawback, however, is that this technique does not protect against configuration errors in a service, i.e. when attacks use legitimate actions of the service to gain unauthorized access.

5 Host-based versus network-based intrusion detection

Host-based intrusion detection was the first area explored in intrusion detection. When the first intrusion-detection systems were designed, the target environment was a mainframe computer, and all users were local to the system considered. This greatly simplified the intrusion-detection task, as interaction from outside was rare. The intrusion-detection system analyzed the audit information provided by the mainframe, either locally [38] or on a separate machine [50], and reported security-suspicious events.

As the focus of computing shifted from mainframe environments to distributed networks of workstations, several prototypes of intrusion-detection systems were developed to accommodate network issues. Here the first step was to get host-based intrusion-detection systems to communicate [29]. In a distributed environment, users hop from one machine to another, possibly changing identities during their moves, and launch their attacks on several systems. Therefore, the local intrusion-detection system on the workstation has to exchange information with its peers. This exchange of information takes place at several levels, either by exchanging a raw audit trail over the network as Stalker [23] does, or by issuing alarms based on a local analysis [51]. Both solutions incur costs: transferring audits can potentially have a huge impact on network bandwidth, whereas processing them locally affects the workstation's performance.

With the widespread use of the Internet, intrusion-detection systems have become focused on attacks to the network itself. Network attacks (DNS spoofing, TCP hijacking, port scanning, ping of death, etc.) cannot be detected by examining the host audit trail, or at least not easily. Therefore, specific tools have been developed that sniff network packets in real time, searching for such network attacks. In addition, a number of classical attacks against servers can also be detected by parsing the payload of the packet and looking for suspicious commands. Moreover, often these tools are attractive for system administrators because a small number of them can be installed at strategic points in the network to cover most of the current attacks.

Hybrid approaches have also been developed that use both network-based and host-based intrusion-detection tools in a multi-host environment. *DIDS* [51] uses *Haystack* [50], which runs on each host to detect local attacks, and *NSM* [24] to monitor the network. Both components report to the *DIDS Director*, where the final analysis is done.

As a side effect, more specialized intrusion-detection tools have emerged that monitor the most critical elements of an organization's presence on the Internet. These products monitor firewalls (*NetStalker* [23]), web servers (*WebStalker* [23]) or routers (*NetRanger* [7, 59]), looking for evidence of attacks in the highly specific context of these network elements.

5.1 Host-based information sources

Host audit sources are the only way to gather information on the activities of the users of a given machine. On the other hand, they are also vulnerable to alterations in the case of a successful attack. This creates an important real-time constraint on host-based intrusion-detection systems, which have to process the audit trail and generate alarms before an attacker taking over the machine can subvert either the audit trail or the intrusion-detection system itself.

5.1.1 Accounting

Accounting is one of the oldest sources of information on system behavior. It provides information on the consumption of shared resources, such as processor time, memory, disk or network usage, and applications launched, by the users of the system. Accounting is found almost everywhere, in network equipment, in mainframes as well as in UNIX workstations. This omnipresence has led some designers of intrusion-detection prototypes to try to use it as an audit source.

In the UNIX environment, accounting is a universal source of information. The format of the accounting record is the same on all UNIXes, the information is compressed to gain disk space, and the overhead introduced by the recording process is very small. It is well integrated in modern operating systems, and easy to set up and exploit.

However, accounting information also has a number of drawbacks, which make it untrustworthy for security purposes. In particular, the information identifying the command launched as well as the time stamps are too imprecise to allow efficient detection of attacks.

5.1.2 Syslog

Syslog is an audit service provided to applications by the operating system (UNIX and others). This service receives a text string from the application, prefixes it with a time stamp and the name of the system on which the application runs, and then archives it, either locally or remotely.

Syslog is known to have security vulnerabilities, as Syslog daemons on several UNIX operating systems have been the subject of CERT documents [5] showing that buffer overflows in the syslog daemon can be exploited to execute arbitrary code.

Syslog is very easy to use, and that has incited many application developers to use it as their audit trail. A number of applications and network services use it, such as *login, sendmail, nfs, http*, including security-related tools such as *sudo, klaxon,* or *TCP wrappers*. Therefore, a few intrusion-detection tools have been developed that use information provided by the syslog daemon, for example Swatch [22]. Although syslog is a lightweight audit source that does not generate a large amount of audit data per machine, a large network can generate a large number of messages, very few of which are security-relevant. Swatch [22] reduces the burden of the system administrator by correlating messages (reports from several machines that an nfs server is down would be aggregated into one) and highlighting security-related ones.

5.1.3 C2 security audit

The security audit records all potentially security-significant events on the system. As the US government has required that all computer systems it purchases be certified at the C2 level of the TCSEC [56], all operating-system vendors competing in this area have had to include an "accountability" feature. This translates into security audit trails such as SUN's BSM and Shield packages, or AIX audit.

All these security audit trails have the same basic principle: They record the crossing of instructions executed by the processor in the user space and instructions executed in the Trusted Computing Base (TCB) space [56]. This security model postulates that the TCB is trusted, that actions in the user space cannot harm the security of the system, and that security-related actions that can impact the system only take place when users request services from the TCB.

In the UNIX environment, the TCB is basically the kernel. Therefore, the audit system records the execution of system calls by all user-launched processes. Compared with a full system-call trace, the audit trail provides limited abstraction: context switches, memory allocation, internal semaphores, and consecutive file reads do not appear in the trail. On the other hand, there is always a straightforward mapping of audit events to system calls.

The UNIX security audit record contains a vast amount of information about the events. It includes detailed user and group identification (from the login identity to the one under which the system call is executed), the parameters of the system call execution (file names including path, command line arguments, etc.), the return code from the execution, and the error code.

The main advantages of the security audit are

- a strong identification of the user, its login identity, its real (current) identity, its effective (set-user-id bit) identity, its real and effective (set-group-id bit) group identities;
- a repartition of audit events into classes to facilitate the configuration of the audit system;
- a fine-grained parameterization of the information gathered according to user, class, audit event, and failure or success of the system call, and
- a shutdown of the machine if the audit system encounters an error status (usually a running out of disk space).

The main drawbacks of the security audit are

- a heavy use of system resources when detailed monitoring is requested. Processor performance could potentially be reduced by as much as 20%, and requirements for local disk space storage and archiving are high;
- a possible denial-of-service attack by filling the audit file system;

- difficulty to set up the audit service owing to the number of parameters involved. Standard configurations delivered by vendors minimize the performance hit by recording only classes of rare events (administrative actions, logins, and logouts). The auditing requirements of an intrusion-detection tool demand more detailed information, such as file accesses, processes executed;
- difficulty to exploit the information obtained owing to its size and complexity. This is compounded by the heterogeneity of audit-system interfaces and audit record formats in the various operating systems, and
- the parameterization of the audit system involving subjects (users) and actions (system calls or events), and only very rarely objects (on which the action is performed). Important objects should be monitored by an intrusion-detection tool, and this is done primarily by scanning the entire trail.

The C2 security audit is the primary source of audit information for the majority of host-based intrusion-detection prototypes and tools because it currently is the only reliable mechanism for gathering detailed information on the actions taken on an information system. Work has been conducted by several groups [21, 39, 43, 55] to define which information should be included in the security audit trail as well as a common format for audit trail records, but this is an ongoing research effort.

5.2 Network-based information sources

5.2.1 SNMP information

The Simple Network Management Protocol (SNMP) Management Information Base (MIB) is a repository of information used for network management purposes. It contains configuration information (routing tables, addresses, names) and performance/accounting data (counters to measure the traffic at various network interfaces and at different layers of the network). This section describes experiments performed within the SECURENET project [53] to use SNMP V1 common MIB for Ethernet and TCP/IP. Other projects also aim at using SNMPv2 and SNMPv3 for security and intrusion detection [32].

The SECURENET project explored whether the counters maintained in this MIB are usable as input information for a behavior-based intrusion-detection system. The starting point was to examine the counters at the interface level because this was the only place where one can differentiate between information sent over the wire and information transmitted inside the operating system via the loop-back interface. The prototype collected increments on the number of bytes and packets transmitted and received at each interface every five minutes. The outcome of a very simple average/standard deviation analysis of this data was not satisfactory, as the standard deviation was larger than the average for almost all sets collected during daytime activity.

MIB counters at higher levels of the network do not contain much more information. On the IP, TCP and UDP layers, the counters exhibited similar behavior but, owing to the larger number of counters at these layers, we did not compute all possible correlations. The ICMP counters show more consistency with respect to their statistical modeling, but ICMP attacks [4] have not been tried to validate this approach.

This study shows that SNMP MIBs are a potentially interesting candidate as an audit source for intrusion-detection systems. The demise of SNMPv2 owing to a lack of consensus on the security features has certainly lowered the interest of the intrusion-detection community in it. However, with the deployment of SNMPv3, new projects are exploiting its features for intrusion-detection tools [32].

5.2.2 Network packets

As the popularity of network sniffers for gathering information has grown in the attackers community, they are nowadays also regarded as an efficient means for gathering information about the events that occur on the network architecture. This is consistent with the trend of moving from a centralized to a distributed computing model, and the pace of change has even increased with the widespread diversification of the Internet. Most accesses to sensitive computers today take place over a network, and therefore capturing the packets before they enter the server is probably the most efficient way to monitor this server.

It is also consistent with the occurrence of denial-of-service attacks. As companies put valuable information on the Internet, and even depend on it as a source of revenue, the prospect of simply shutting down

a web site creates an effective threat to the organization running it. Most of these denial-of-service attacks originate from the network and must be detected at the network level, as a host-based intrusion-detection system does not have the capability to acquire this kind of audit information.

There is an inherent duality in network sniffers, which is also apparent in the firewall world with its differences between application-level gateways and filtering routers [3]. If the analysis is carried out at a low level by performing pattern matching, signature analysis, or some other kind of analysis of the raw content of the TCP or IP packet, then the intrusion-detection system can perform its analysis quickly. This is a stateless approach that does not take session information into account because the latter could span several network packets. If the intrusion-detection system acts as an application gateway and analyzes each packet with respect to the application or protocol being followed, then the analysis is more thorough, but also much more costly. This is a stateful analysis. Note that this analysis of the higher levels of the protocol also depends on the particular machine being protected, as implementations of the protocols are not identical from one network stack to another.

This approach addresses several issues:

- The detection of network-specific attacks. There are a number of network attacks, particularly denial-of-service attacks that cannot be detected in a timely fashion by searching for audit information on the host, but only by analyzing network traffic.
- The impact of auditing on the host performance. Information is entirely collected on a separate machine, with no knowledge of the rest of the network. Therefore, installation of such tools is facilitated because they do not impact the entire environment in terms of configuration and performance.
- The heterogeneous audit trail formats. The current de facto standardization towards TCP/IP facilitates the acquisition, formatting, and cross-platform analysis of the audit information.
- Certain tools analyze the payload of the packet, which allows attacks against hosts to be detected by signature analysis. However, an efficient analysis requires knowledge of the type of machine or application for which the packet is intended.

But it also has a number of drawbacks:

- It is more difficult to identify the culprit when an intrusion has been discovered. There is no reliable link between the information contained in the packets and the identity of the user who actually submitted the commands on the host.
- With switched networks (switched-Ethernet, switched Token-Ring, ATM), selecting an appropriate location for the sniffer is not straightforward. Some tools are located on switches, others at gateways between the protected system and the outside world. The former gives better audit information but also incurs a higher cost. Note, however, that switched networks are also much less vulnerable to sniffer attacks [6, 44] and actually are a recommended solution to improve the security of a network.
- Encryption prevents the analysis of the payload of the packets, and therefore hides a considerable amount of important information from these tools. Also, even without encryption, it is possible to obfuscate the contents of the packet to evade detection [44] if the signatures are not sufficiently comprehensive.
- Systematic scanning, for example at the firewall, is difficult because it might create bottlenecks. This will only worsen as the bandwidth to access the Internet is increased at sensitive sites (e.g. banks, electronic-commerce web sites).
- Finally, these tools are inherently vulnerable to denial-of-service attacks if they rely on a commercial operating system to acquire network information. Just as the network stacks of these commercial operating systems are vulnerable to attacks, so is the intrusion-detection system.

Network packets currently are the source of information used by several recent commercial products [7, 28, 59], and several projects in the research community also pursue this track [41, 45, 46, 54]. A recent evaluation of these products by Ptacek and Newsham [44] shows that the sniffer approach, or at least the current implementations, have flaws that make it possible for a skilled attacker to evade detection. In particular, they show that IP fragmentation is not handled well, and that the use of wildcards and control sequences in protocols such as `http` makes it possible to evade detection by signature. Also in this area is research being conducted. After *IDES* and *NIDES*, SRI is now developing a prototype called *Emerald* [42] to deal with the analysis of network traffic. Other network sniffers such as *Bro* [41] or *Network Flight Recorder* [45] have been developed as network data-acquisition tools and therefore do not support intrusion-detection *per se*.

5.3 Application log files

As the trend towards application servers becomes more pronounced and the notion of operating system fades, application log files gain greater importance as a data source for intrusion detection.

Compared with system audits or network packets, the use of application log files has three advantages:

Accuracy. C2 audit data or network packets require processing before the intrusion-detection system can understand which information has actually been received by the application. This processing is based on interpretations of protocol specifications or API specifications, and it is probable that the interpretation of the application developer differs from that of the intrusion-detection system developer. By obtaining the information directly from the log, the accuracy of the information is almost guaranteed.

Completeness. C2 audit data or network packets require reassembly of several audit calls or several network packets, potentially on multiple hosts, to rebuild the session at the application level, which can be very difficult to achieve. Moreover, not even a simple reassembly, such as matching incoming http request and outgoing response to determine the success of a request, is done by current tools. The application log, on the other hand, contains all relevant information, even if the application is distributed over a cluster of machines (e.g web server or database server). In addition, the application can provide internal data that does not show up in audit trails or network packets.

Performance. By letting the application select which information is relevant for security purposes, the overhead induced by the collection mechanism is greatly reduced when compared with security audit trails.

There are two drawbacks in using application log files for intrusion detection:

Race condition. Attacks are only detected when the application log is written. If the attack can prevent the writing of the application log (this may be the case in many denial-of-service attacks), then the information needed by the intrusion-detection system is not there.

Low-level attacks. There are a number of attacks (again, particularly denial-of-service attacks) that target the lower levels of the system software, such as network drivers. As these attacks do not exercise application code, they may not appear in the application logs, or only the consequence of the attack (such as reboot) is visible.

An example of such a tool is *WebWatcher* [1], which monitors web server logs in real time and provides much more detailed information about web server attacks than its network-based counterparts do. A similar approach could be envisioned for database servers.

6 Detection paradigm

6.1 State-based versus transition-based intrusion-detection systems

There are essentially two paradigms in intrusion-detection engines. The first class of intrusion-detection engines works on states, the second on transitions between states. Figure 3 explains the paradigm using

terms defined in dependability [2]. NORMAL represents the desired state of the system. ERROR1 and ERROR2 represent two steps in a chain that will eventually lead to a FAILURE state. The goal of the intrusion-detection system is to detect that the system has left the NORMAL state before it reaches the FAILURE state. In the context of intrusion detection, ERROR1 could mean that a vulnerable version of an application is installed on the system. If this vulnerable application is used by the attacker to recover the `passwd` file, then the system would be in the ERROR2 state. The final FAILURE state could represent the fact that an external attacker logs in with administrative privileges.

To achieve their goal, intrusion-detection tools can do several things:

- Recognize the NORMAL state. This is only moderately useful because it provides no information on where the system is in the chain towards the FAILURE state. It is also extremely difficult to characterize normality, as discussed in the context of behavior-based intrusion detection earlier.
- Recognize some ERROR state on the way. It is usually possible to characterize ERROR states in a way that allows detection.
- Recognize some TRANSITION leading to an ERROR state.

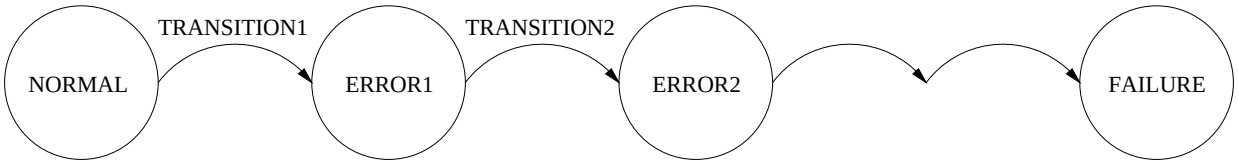

Figure 3: State versus transition.

Example of state-based intrusion-detection systems include security assessment tools such as *COPS* [14, 16] and *Tiger* [47] for host environments, and *Nessus* [12], *Satan* [15] and *Ballista* [49] for network-based tools. These tools identify ERROR states when a vulnerable application or a configuration error is present. They query the system for application versions or configurations, and signal an ERROR state when the version or configuration matches information in their database of vulnerabilities. *Tripwire* [33] maintains signatures of important files and looks for changes. Such changes could indicate that the system is now in an ERROR state (e.g. because the attacker has changed one of the original files with a trojan horse for easy penetration). Yet another system is *Wisdom & Sense* [57], which keeps statistical rules of appropriate behavior and indicates when these rules are not satisfied (e.g. because the person behind the terminal is masquerading as an authorized user, but does not behave exactly as expected). None of these tools will detect the TRANSITION when the system goes from one state to another.

Examples of transition-based intrusion-detection systems include *NetRanger* [7] or *RealSecure* [28] for network-based tools, and *USTAT* [27] for host-based tools. These tools watch for specific events that are known to trigger TRANSITION from one state to another (e.g. attacks, malicious actions, malicious packets). *USTAT* has the capability to combine several transitions, while *RealSecure* and *NetRanger* look for individual transitions.

6.2 Nonperturbing versus pro-active analysis of state or transition

The state or transition analysis can be carried in two ways, either by a nonperturbing observation of the system or by a pro-active attempt to evaluate the state or the transition that will modify the state of the system afterwards.

6.2.1 Nonperturbing analysis

In a nonperturbing observation, the vulnerability-assessment side requests application versions or banners, and compares them with a table of known vulnerabilities. If the version of the application is in the table,

then the system is tagged as being in the VULNERABLE state, otherwise it will be tagged as being in the SECURE state. This kind of probe attempts to minimize the impact on the system while reliably investigating its state or transition.

COPS or *SATAN* are examples of non-perturbing, state-analysis tools. By requesting version information from applications and analyzing the configuration file, they determine whether the system is in a SECURE state.

Examples of non-perturbing transition-analysis tools include log or network traffic analysis tools, such as *RealSecure*, *NetRanger*, or *WebWatcher* [1]. These tool acquire information transparently, either by capturing network packets or by receiving log entries, but do not significantly impact the environment on which they are running.

6.2.2 Pro-active analysis

Another class of tools performs a pro-active analysis by explicitly triggering events on the environment to determine states or create transitions.

Nessus or *Ballista* are examples of pro-active, state-analysis tools. They actively exploit vulnerabilities to determine the state of the system. These attempts are almost indistinguishable from undesired ones, and may trigger a transition from an ERROR state to the next in the chain.

The *Sniffer Detector* [20] and *AntiSniff* [35] aim at detecting passive sniffers that may have been installed on a network to capture sensitive data. If a sniffer is installed, the system would be in an ERROR state. *AntiSniff* pro-actively triggers side effects in the sniffer, which would detect the ERROR state if these side effects are observed. The *Sniffer Detector* pro-actively introduces baits in the system and looks for evidence of usage of these baits. If such evidence is found, it shows the TRANSITION from the ERROR state with the sniffer installed to the next ERROR state in which the sniffer owner is making use of the information collected.

7 Additional properties

7.1 Continuous monitoring versus periodic analysis

Continuous versus periodic intrusion detection applies to the way the tool performs its analysis. A dynamic intrusion-detection tool performs a continuous, real-time analysis by acquiring information about the actions taken on the environment immediately after they happen. A static intrusion-detection tool periodically takes a snapshot of the environment and analyzes this snapshot, looking for vulnerable software, configuration errors, and so on.

Static tools assess the security level of the current configuration of the environment. Examples include *COPS* [14, 16] and *Tiger* [47] for host environments, and *Satan* [15] and *Ballista*[1] [49] for networks. In the same category are virus detectors, which scan the disks looking for patterns identifying known viruses. These checks include verifying the version of the applications installed to ensure that the latest security patches have been applied, checking for weak passwords, verifying the contents of special files in users' home directories, and verifying the configuration of open network services. This analysis provides an instant snapshot of the state of the system, but is only valid at that precise moment.

These tools are well known and widely used by system administrators, but they are not sufficient to ensure high security. First of all, security patches are not necessarily available on legacy systems, which cannot be upgraded without losing their operational requirements. Moreover, running these security-assessment tools is often a lengthy process, particularly in a networked environment where every system has to be checked individually. Therefore, the security exposure between two consecutive runs might be significant, approximately an entire day or so, and it has been shown that active exploitation of vulnerabilities over the Internet can take less than one day.

[1] Now *CyberCop Scanner* [40] since the acquisition of Secure Networks by Network Associates Inc.

These tools as well as others specifically developed for that purpose, such as *Tripwire* [33] and *ATP* [58], can be used to detect the traces of an intrusion. Such traces can be the replacement of a given application by an older, vulnerable one, which would be signaled by *COPS* [14] and *Tiger* [47] to the system administrator as a potential intrusion. *Tripwire* [33] extends this principle by computing the signature of a large set of system files, and comparing it with a database of reference signatures kept in a safe place, therefore rendering the change detection process systematic. An alarm by a Tripwire-like system signals an intrusion in a behavior-based way, i.e., that some file in the system is not what it used to be.

Dynamic intrusion-detection tools monitor the actions that take place on the system. The monitoring occurs either in real time or in batch, i.e. reviewing audit files or network packets accumulated over a given period of time. Dynamic monitoring implies real-time analysis and allows a constant assessment of the security of the system. It is, however, a costly process, both for transporting the audits and for processing them.

7.2 Protection of the intrusion-detection system

When an intrusion-detection system is deployed, it becomes the natural primary target of hostile attacks, with the aim of disabling the detection feature and allowing an attacker to operate without being detected. Disabling the intrusion-detection system can happen in the following ways:

Denial-of-service attacks. Denial-of-service attacks are a powerful and relatively easy way of temporarily disabling the intrusion-detection system. The attack can take place against the detector, by forcing it to process more information than it can handle (for example by saturating a network link). This usually has the effect of delaying detection of the attack or, in the worst case, of confusing the detector enough so that it misses some critical element of the attack. A second possibility is to saturate the reaction capability of the operator handling the intrusion-detection system. When the operator is presented with too many alarms, he can easily miss the important one indicating penetration, even if it is present on the screen.

Evasion of the detection. Several techniques have been developed to evade detection of an attack by intrusion-detection systems. Network-based tools, the most popular tools today, particularly suffer from these attacks involving hand-crafted network packets:

1. Attack by IP fragmentation. Intrusion-detection systems have difficulties reassembling IP packets. Therefore, splitting an attack artificially into multiple packets creates a mismatch between the data in the packet and the signature, thus hiding the attack.
2. Attack via the TTL (Time To Live). By altering the TTL of IP packets, it is possible to make the intrusion-detection system see packets that will not arrive at the target of the attack. By inserting fake data into the communication stream, an attacker can interleave the attack with bogus information, thus hiding the attack from the intrusion detection system while the target correctly reconstructs this attack data and reacts to it.

A good description of various attack techniques and how intrusion-detection systems react to them is given by Ptacek and Newsham [44]. Intrusion-detection systems are beginning to protect themselves from these attacks, but little information is released by vendors as to the effectiveness of these protection measures.

In addition, it is often difficult to assert the configuration of an intrusion-detection system, as in most cases there is no easy way to check the configuration and the proper detection of the attacks.

8 Conclusions and future directions

Intrusion detection currently attracts considerable interest from both the research community and commercial companies. Research prototypes continue to appear, and commercial products based on early research are now available. In this paper, I have given an overview of the current state of the art of intrusion detection, based on a proposed taxonomy illustrated with examples of past and current projects. The taxonomy clearly

highlights the properties of these intrusion-detection systems, covering both past and current developments adequately.

Information sources for these tools are either a C2 audit trail, syslog, or network packets. Whereas system sources were widely used in the early stages of research, the current focus of research prototypes as well as products is on protecting the infrastructure rather than the end-user station, and this paradigm has led to the use of network sniffers that analyze packets. As shown, quite a number of research issues concerning the efficiency of both network and host audit sources, the formatting and existence of a common audit trail format, and even the contents of the audit trail itself, still await an answer.

There are also a number of unsolved issues concerning the analysis of the audit trail. Signature analysis is clearly in the commercial domain now, but has been shown to be insufficient for detecting all attacks. Therefore, work is still in progress to experiment with new approaches to both knowledge-based and behavior-based intrusion detection. The detection of abuse-of-privilege attacks (primarily insider attacks) is also the subject of ongoing work.

References

[1] Magnus Almgren, Hervé Debar, and Marc Dacier. A lightweight tool for detecting web server attacks. In Gene Tsudik and Avi Rubin, editors, *Proceedings of NDSS 2000 (Network and Distributed System Security Symposium)*, pages 157–170, San Diego, CA, February 2000. The Internet Society.

[2] T. Anderson, A. Avizienis, W.C. Carter, A. Costes, F. Cristian, Y. Koga, H. Kopetz, J.H. Lala, J.C. Laprie, J.F. Meyer, B. Randell, A.S. Robinson, L. Simonici, and U. Voges. *Dependability: Basic Concepts and Terminology*. Dependable Computing and Fault Tolerance. Springer Verlag, 1992.

[3] Steven M. Bellovin and William R. Cheswick. Network firewalls. *IEEE Communications MAGAZINE*, 32(9):50–57, September 1994.

[4] CERT Coordination Center. Denial-of-service attack via ping. Available by anonymous ftp from ftp.cert.org, December 1986.

[5] CERT Coordination Center. Syslog vulnerability - a workaround for sendmail. Available by anonymous ftp from ftp.cert.org, October 1995.

[6] William R. Cheswick and Steven M. Bellovin. *Firewalls and Internet Security – Repelling the Wily Hacker*. Professional Computing Series. Addison-Wesley, 1994. ISBN 0-201-63357-4.

[7] Cisco Systems Inc. *NetRanger* – Enterprise-scale, Real-time, Network Intrusion Detection System. Internet http://www.cisco.com/, 1998.

[8] Hervé Debar, Monique Becker, and Didier Siboni. A neural network component for an intrusion detection system. In *Proceedings of the 1992 IEEE Computer Society Symposium on Research in Security and Privacy*, pages 240–250, Oakland, CA, May 1992.

[9] Hervé Debar, Marc Dacier, Medhi Nassehi, and Andreas Wespi. Fixed vs. variable-length patterns for detecting suspicious process behavior. In Jean-Jacques Quisquater, Yves Deswarte, Catherine Meadows, and Dieter Gollmann, editors, *Computer Security - ESORICS 98, 5th European Symposium on Research in Computer Security*, volume 1485 of *LNCS*, pages 1–15, Louvain-la-Neuve, Belgium, September 1998. Springer Verlag.

[10] Hervé Debar, Marc Dacier, and Andreas Wespi. Reference Audit Information Generation for Intrusion Detection Systems. In Reinhard Posch and György Papp, editors, *Information Systems Security, Proceedings of the 14th International Information Security Conference IFIP SEC'98*, pages 405–417, Vienna, Austria and Budapest, Hungaria, August 31–September 4 1998.

[11] Dorothy Denning. An intrusion-detection model. *IEEE Transactions on Software Engineering*, 13(2):222–232, 1987.

[12] Renaud Deraison. The nessus project. http://www.nessus.org/documentation.html, 1999.

[13] Cheri Dowell and Paul Ramstedt. The ComputerWatch data reduction tool. In *Proceedings of the 13th National Computer Security Conference*, pages 99–108, Washington, DC, October 1990.

[14] Dan Farmer. Cops overview. Available from http://www.trouble.org/cops/overview.html, May 1993.

[15] Dan Farmer and Wietse Venema. Improving the security of your site by breaking into it. available at http://www.trouble.org/security/admin-guide-to-cracking.html, 1993. Internet white paper.

[16] Daniel Farmer and Eugene Spafford. The cops security checker system. In *Proceedings of Summer USENIX Conference*, pages 165–170, Anaheim, CA, June 1990.

[17] Stephanie Forrest, Steven A. Hofmeyr, and Anil Somayaji. Computer immunology. *Communications of the ACM*, 40(10):88–96, October 1997.

[18] Patrick Gallinari, Sylvie Thiria, and Francoise Fogelman-Soulie. Multilayer perceptrons and data analysis. In *Proceedings of the IEEE Annual International Conference on Neural Networks (ICNN88)*, volume I, pages 391–399, San Diego, CA, July 1988.

[19] Thomas Garvey and Teresa Lunt. Model-based intrusion detection. In *Proceedings of the 14th National Computer Security Conference*, pages 372–385, October 1991.

[20] Steṕhane Grundschober. Design and implementation of a sniffer detector. In *Proceedings of RAID 98, Workshop on Recent Advances in Intrusion Detection*, Louvain-la-Neuve, Belgium, September 1998.

[21] Naji Habra, Baudouin Le Charlier, Aziz Mounji, and Isabelle Mathieu. Asax: Software architecture and rule-based language for universal audit trail analysis. In Y. Deswarte, G. Eizenberg, and J.-J. Quisquater, editors, *Proceedings of the Second European Symposium on Research in Computer Security (ESORICS)*, volume 648 of *Lecture Notes in Computer Science*, Toulouse, France, November 1992. Springer-Verlag, Berlin Germany.

[22] Stephen E. Hansen and E. Todd Atkins. Automated system monitoring and notification with swatch. In *Proceedings of the seventh Systems Administration Conference (LISA '93)*, Monterey, CA, November 1993.

[23] Haystack Labs, Inc. Stalker. http://www.haystack.com/stalk.htm, 1997.

[24] L. Todd Heberlein, Gihan V. Dias, Karl N. Levitt, Biswanath Mukherjee, Jeff Wood, and David Wolber. A network security monitor. In *Proceedings of the 1990 IEEE Symposium on Research in Security and Privacy*, pages 296–304, Oakland, CA, May 1990. IEEE Computer Society Press, Los Alamitos, CA.

[25] Paul Helman and Gunar Liepins. Statistical foundations of audit trail analysis for the detection of computer misuse. *IEEE Transactions on Software Engineering*, 19(9):886–901, September 1993.

[26] Paul Helman, Gunar Liepins, and Wynette Richards. Foundations of intrusion detection. In *Proceedings of the Fifth Computer Security Foundations Workshop*, pages 114–120, Franconic, NH, June 1992.

[27] Koral Ilgun. Ustat: A real-time intrusion detection system for unix. In *Proceedings of the 1993 IEEE Symposium on Research in Security and Privacy*, pages 16–28, Oakland, CA, May 1993.

[28] Internet Security Systems, Inc. RealSecure. Internet http://www.iss.net/prod/rsds.html, 1997.

[29] R. Jagannathan, Teresa Lunt, Debra Anderson, Chris Dodd, Fred Gilham, Caveh Jalali, Hal Javitz, Peter Neumann, Ann Tamaru, and Alfonso Valdes. System design document: Next-generation intrusion detection expert system (NIDES). Technical Report A007/A008/A009/A011/A012/A014, SRI International, 333 Ravenswood Avenue, Menlo Park, CA 94025, March 1993.

[30] Harold Javitz and Alfonso Valdes. The SRI IDES statistical anomaly detector. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, pages 316–326, May 1991.

[31] Harold S. Javitz, Alfonso Valdez, Teresa F. Lunt, Ann Tamaru, Mabry Tyson, and John Lowrance. Next generation intrusion detection expert system (NIDES) - 1. statistical algorithms rationale - 2. rationale for proposed resolver. Technical Report A016–Rationales, SRI International, 333 Ravenswood Avenue, Menlo Park, CA, March 1993.

[32] Y. Frank Jou, Fengmin Gong, Chandru Sargor, Shyhtsun Felix Wu, and W. Rance Cleaveland. Architecture design of a scalable intrusion detection system for the emerging network infrastructure. Technical Report CDRL A005, MCNC Information Technologies Division, Research Triangle Park, N.C. 27709, April 1997.

[33] Gene H. Kim and Eugene H. Spafford. The design and implementation of tripwire: A file system integrity checker. In Jacques Stern, editor, *2nd ACM Conference on Computer and Communications Security*, pages 18–29, COAST, Purdue, November 1994. ACM Press.

[34] Sandeep Kumar and Eugene Spafford. A pattern matching model for misuse intrusion detection. In *Proceedings of the 17th National Computer Security Conference*, pages 11–21, October 1994.

[35] Inc. L0pht Heavy Industries. What is packet sniffing. http://www.l0pht.com/antisniff/overview.html, 1999.

[36] Carl E. Landwehr, Alan R. Bull, John P. McDermott, and William S. Choi. A taxonomy of computer program security flaws. *ACM Computing Surveys*, 26(3):211–254, September 1994.

[37] Teresa Lunt and R. Jagannathan. A prototype real-time intrusion-detection expert system. In *Proceedings of the 1988 Symposium on Security and Privacy*, pages 59–66, Oakland, CA, April 1988.

[38] Teresa F. Lunt, R. Jagannathan, Rosanna Lee, Sherry Listgarten, David L. Edwards, Peter G. Neumann, Harold S. Javitz, and Alfonso Valdes. IDES: The enhanced prototype – a real–time intrusion–detection expert system. Technical Report SRI-CSL-88-12, SRI International, 333 Ravenswood Avenue, Menlo Park, CA, October 1988.

[39] Abdelaziz Mounji. *Languages and Tools for Rule-Based Distributed Intrusion Detection*. Doctor of science, Facultés Universitaires Notre-Dame de la Paix, Namur (Belgium), September 1997.

[40] Network Associates Inc. Cybercop scanner. Available from the company's website at http://www.nai.com/products/security/ballista/default.asp, 1998.

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

[42] Phillip A. Porras and Alfonso Valdes. Live traffic analysis of tcp/ip gateways. In *Proceedings of the 1998 ISOC Symposium on Network and Distributed System Security (NDSS'98)*, San Diego, CA, March 1998. Internet Society.

[43] Katherine E. Price. Host-based misuse detection and conventional operating systems' audit data collection. Master of science thesis, Purdue University, Purdue, IN, December 1997.

[44] Thomas H. Ptacek and Timothy N. Newsham. Insertion, evasion, and denial of service: Eluding network intrusion detection. Technical report, Secure Networks, Inc., Suite 330, 1201 5th Street S.W, Calgary, Alberta, Canada, T2R-0Y6, January 1998.

[45] Marcus J. Ranum, Kent Landfield, Mike Stolarchuk, Mark Sienkiewicz, Andrew Lambeth, and Eric Wall. Implementing a generalized tool for network monitoring. In *Proceedings of the Eleventh Systems Administration Conference (LISA '97)*, San Diego, CA, October 1997.

[46] P. Rolin, L. Toutain, and S. Gombault. Network security probe. In *CCS'94, Proceedings of the 2nd ACM Conference on Computer and Communication Security*, pages 229–240, November 1994.

[47] David R. Safford, Douglas Lee Schales, and David K. Hess. The tamu security package: An ongoing response to internet intruders in an academic environment. In *Proceedings of the Fourth USENIX Security Symposium*, pages 91–118, Santa Clara, CA, October 1993.

[48] Warren S. Sarle. Neural networks and statistical models. In *Proceedings of the Nineteenth Annual SAS Users Group International Conference, April, 1994*, pages 1538–1550, Cary, NC, April 1994. SAS Institute.

[49] Secure Networks, Inc. Ballista security auditing system. Internet http://www.securenetworks.com/, 1997.

[50] Stephen Smaha. Haystack: An intrusion detection system. In *Fourth Aerospace Computer Security Applications Conference*, pages 37–44, October 1988.

[51] Steven R. Snapp, James Brentano, Gihan V. Dias, Terrance L. Goan, L. Todd Heberlein, Che lin Ho, Karl N. Levitt, Biswanath Mukherjee, Stephen E. Smaha, Tim Grance, Daniel M. Teal, and Doug Mansur. DIDS (distributed intrusion detection system) - motivation, architecture, and an early prototype. In *Proceedings of the 14th National Computer Security Conference*, pages 167–176, Washington, DC, October 1991.

[52] Michael Sobirey. Intrusion detection system bibliography. Internet: http://www-rnks.informatik.tu-cottbus.de/ sobirey/ids.html, March 1998.

[53] Paul Spirakis, Sokratis Katsikas, Dimitris Gritzalis, Francois Allegre, John Darzentas, Claude Gigante, Dimitris Karagiannis, P. Kess, Heiki Putkonen, and Thomas Spyrou. SECURENET: A network-oriented intelligent intrusion prevention and detection system. *Network Security Journal*, 1(1), November 1994.

[54] S. Staniford-Chen, S. Cheung, R. Crawford, M. Dilger, J. Frank, J. Hoagland, K. Levitt, C. Wee, R. Yip, and D. Zerkle. GrIDS – a graph-based intrusion detection system for large networks. In *Proceedings of the 19th National Information Systems Security Conference*, 1996.

[55] Stuart Staniford-Chen, Brian Tung, Phil Porras, Cliff Kahn, Dan Schnackenberg, Rich Feiertag, and Maureen Stillman. The common intrusion detection framework - data formats. Internet draft draft-ietf-cidf-data-formats-00.txt, March 1998. Work-in-progress.

[56] U.S. Department of Defense. Trusted computer systems evaluation criteria, August 1983.

[57] H. S. Vaccaro and G. E. Liepins. Detection of anomalous computer session activity. In *Proceedings of the 1989 IEEE Symposium on Research in Security and Privacy*, pages 280–289, 1989.

[58] David Vincenzetti and Massimo Cotrozzi. Atp – anti tampering program. In *Proceedings of the Fourth USENIX Security Symposium*, pages 79–89, Santa Clara, CA, October 1993.

[59] WheelGroup Corporation. Brochure of the netranger intrusion detection system. Internet http://www.wheelgroup.com/.