RZ 3030 (# 93076) 06/01/98
Computer Science/Mathematics 20 pages

Research Report

Towards a Taxonomy of Intrusion–Detection Systems

Hervé Debar, Marc Dacier and Andreas Wespi

IBM Research Division
Zurich Research Laboratory
8803 Rüschlikon
Switzerland

Research Division
Almaden · Austin · Beijing · Haifa · T.J. Watson · Tokyo · Zurich

Towards a Taxonomy of Intrusion–Detection Systems

Hervé Debar, Marc Dacier and Andreas Wespi

IBM Research Division, Zurich Research Laboratory, 8803 Rüschlikon, Switzerland

Abstract

Intrusion–detection systems aim at detecting attacks against computer systems and networks. As it is difficult to provide provably secure information systems, intrusion–detection systems have the task of monitoring the usage of such systems. They detect attempts by users of the information systems or external parties to abuse their privileges or exploit security flaws. In this paper, we introduce a taxonomy that highlights the various aspects of this area. This taxonomy divides families of intrusion–detection systems according to their properties. It is illustrated by numerous examples from past and current projects.

1 Introduction

Since the seminal work by Denning in 1981 [9], many intrusion–detection prototypes have been created. Sobirey maintains a partial list of 59 of them [55]. 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.* [35] shows that computer systems suffer from security vulnerabilities regardless of their purpose, manufacturer, or origin, and that it is both technically hard and economically costly (both in terms of building and maintaining such a system) to ensure that computer systems and networks are not susceptible to attacks.

This paper introduces a taxonomy of intrusion–detection systems at a time when commercial tools are increasingly becoming available. The taxonomy draws examples from both the research prototypes and commercial products to illustrate the most preeminent features of intrusion–detection systems. The paper focuses on the TCPIP/UNIX world, for which the largest number of prototypes and tools have been developed. However, a number of these products are now also available for Windows NT, which has been more widely deployed in the enterprise world and has been subjected to enhanced scrutiny by the security and underground communities.

This paper does not attempt to be a complete survey of existing intrusion–detection tools, techniques, projects, and products. Several surveys have indeed been published [2, 12, 17, 36, 38, 39, 41], but the growth of the intrusion–detection field has been such that many new projects have appeared in the meantime. Therefore, we shall present an updated image of the intrusion–detection field, organized around a proposed taxonomy for intrusion–detection systems, and illustrated with examples from past and current tools.

The paper is organized as follows: Section 2 describes the architecture of a generic intrusion–detection system, Section 3 presents the taxonomy that we use to describe and classify intrusion–detection systems and examples of techniques and information sources, Section 4 illustrates the concepts described with a summary of existing tools and prototypes, and Section 5 describes the reusability issue of intrusion–detection systems and components.

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.

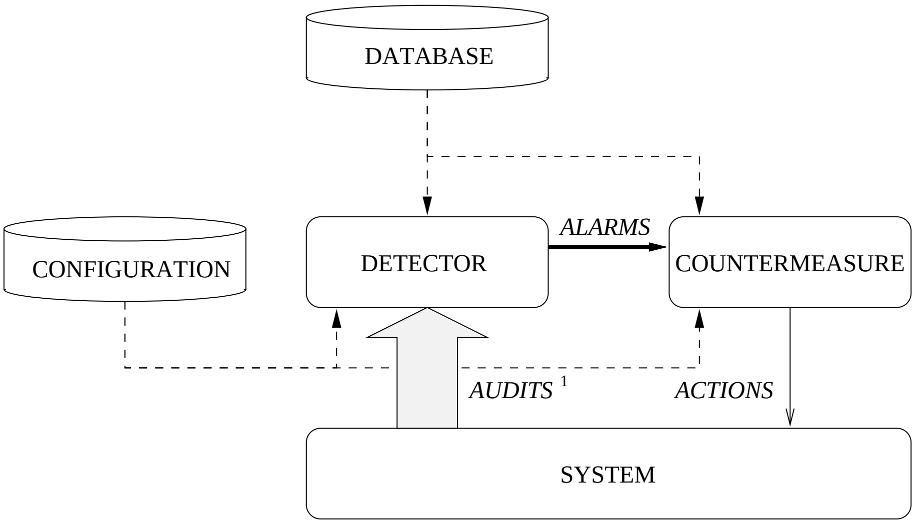
The term *audit* denotes information provided by a system concerning its inner workings and behavior. Examples of audits include but are not limited to C2 audit trail, accounting, and syslog in the UNIX world, syslog in the MVS world, the event log in Windows NT, and incident tickets in X25 networks. A description of some of these audits is given in Section 3.3.

The term *component* refers to a box inside an intrusion–detection system. There are many kinds of components, an overview of which is given in Section 3.1.

2.2 Description

An intrusion–detection system dynamically monitors the actions that are taken in a given environment, and decides whether these actions are symptomatic of an attack or constitute a legitimate use of the environment. Therefore, with respect to this definition, we do not consider well–known tools such as Cops or Satan to be intrusion–detection systems; we consider them configuration analyzers, even though some of their functionalities can be used to detect intrusions.

An intrusion–detection system can be described at a very macroscopic level as a detector that processes information coming from the system that is to be protected (Fig. 1). This detector uses three kinds of information: long–term information related to the technique used to detect intrusions



[1]The arrow thickness represents the amount of information flowing from one component to the other

Figure 1: Very simple intrusion–detection system.

(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 and present a synthetic view of the security–related actions taken by the users. A decision is then taken to evaluate the probability that these actions can be considered symptoms of an intrusion.

2.3 Efficiency of intrusion–detection systems

The following three measures to evaluate the efficiency of an intrusion–detection system have been highlighted in [45]:

Accuracy. Inaccuracy occurs when an intrusion–detection system flags as anomalous or intrusive a legitimate action in the environment.

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. 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.

In addition, we introduce two additional properties:

Fault tolerance. An intrusion–detection system should itself be resistant to attacks, particularly denial of service, 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

There are a number of concepts which we use to classify intrusion–detection systems, presented in Fig. 2.

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 proactive (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 (including paging, etc.), 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, system logs or network packets.

The *usage frequency* is an orthogonal concept. Certain intrusion–detection systems have real–time continuous monitoring capabilities, whereas others have to be run periodically.

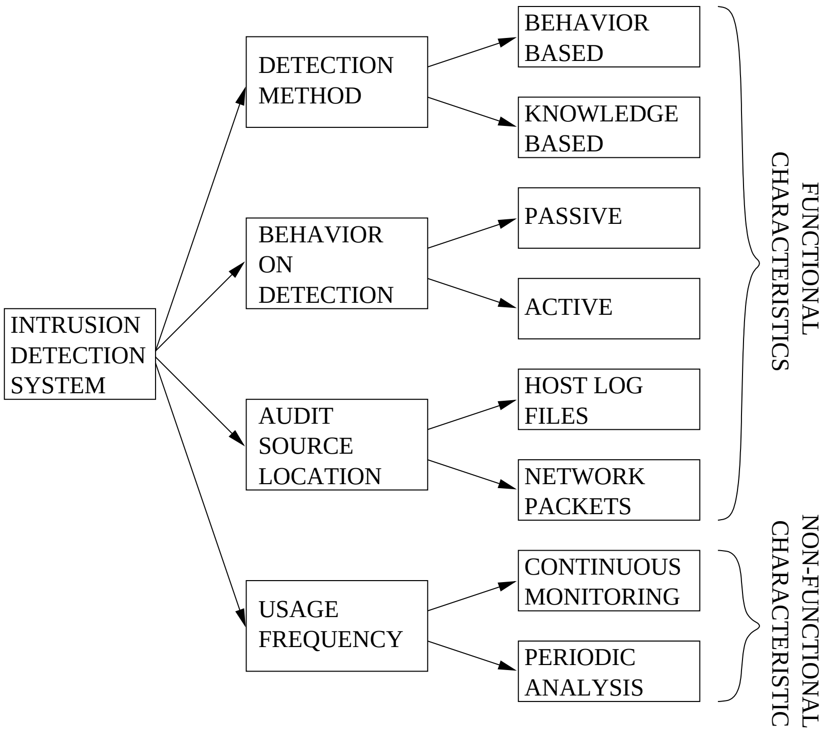


Figure 2: Characteristics of intrusion–detection systems.

3.1 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 [56]. The second trend is referred to as anomaly detection [29] or detection by behavior [56]. In this paper, we use the term *knowledge–based* intrusion detection for the first trend, which we feel describes more precisely the technique being used. The second trend is characterized by the term *behavior–based* intrusion detection.

3.1.1 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 these vulnerabilities. When such an attempt is detected, an alarm is triggered. 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 the potential for very low false alarm rates, and the contextual analysis proposed by the intrusion–detection system is detailed, making it easier for the security officer using this intrusion–detection system 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[1]. Knowledge–based approaches also have to face the generalization issue. Knowledge about attacks is very focused, dependent on the operating system, version, platform, and application. The resulting intrusion–detection tool 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.

3.1.1.1 Expert systems

Expert systems [37] are used primarily 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 [20] 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 for verifying the proper application of the security policy of an organization.

Also using expert systems but having additional properties, model–based reasoning 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 revealing 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.

[1] For internal use, we are maintaining a vulnerability database to which we weekly add five or six new vulnerabilities and multiple attacks!

3.1.1.2 Signature analysis

Signature analysis follows exactly the same knowledge–acquisition approach as expert systems, 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 attacks description.*

This technique allows a very efficient implementation and is therefore applied in commercial intrusion–detection products [22, 27, 63]. The main drawback of this technique — like all knowledge–based approaches — is the need for frequent updates to keep up with the stream of new vulnerabilities discovered, this situation being aggravated by the requirement to represent by signatures all possible facets of the attacks. This leads to an attack being represented by a number of signatures, at least one for each operating system to which the intrusion–detection tool has been ported.

3.1.1.3 Petri nets

To represent signatures of intrusions, IDIOT [34], a knowledge–based intrusion–detection system developed at Purdue University, uses Colored Petri Nets (CPN). The advantages of CPN are their generality, their conceptual simplicity, and their graphical representability. System administrators are assisted in writing their own signatures of attacks and integrating them in IDIOT. Owing to the generality of CPN, quite complex signatures can be written easily. However, matching a complex signature against the audit trail may become computationally very expensive.

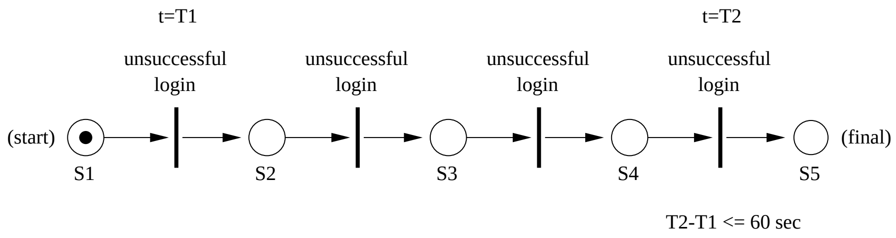


Figure 3: Four failed login attempts within one minute.

Figure 3 shows a simple example of a Colored Petri Net that issues an alarm if the number of unsuccessful login attempts exceeds four within one minute. The transition, represented by a vertical bar, from state s1 to s2 can occur if there is a token in state s1 and an unsuccessful login attempt. The time of the first unsuccessful login attempt is stored in the token variable T1. The transition from state s4 to state s5 can happen if there is a token in s4, an unsuccessful login attempt, and the time difference between this and the first unsuccessful login attempt is less than 60 seconds. Reaching the final state s5 corresponds to a matched signature, and may therefore result in an alarm being issued.

3.1.1.4 State–transition analysis

State–transition analysis is a technique proposed by R. Kemmerer [44], implemented first under UNIX [26], and later in other environments. The technique is conceptually identical to model–based reasoning; it describes the attacks with a set of goals and transitions, but represents them as state–transition diagrams.

3.1.2 Behavior–based intrusion detection

Behavior–based intrusion–detection techniques assume that an intrusion can be detected by observing a deviation from 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 privileges" types of 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 the entire scope of the behavior of an information system may not 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 contains intrusive behavior, which is not detected as anomalous.

3.1.2.1 Statistics

The most widely used tool to build behavior–based intrusion–detection systems is statistics [24, 25, 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 ($\sim$ one month).

The original model keeps averages of all these variables and detects whether thresholds are exceeded based on the standard deviation of the variable. This model is actually too simple to represent the data faithfully. Even comparing the variables of individual users with aggregated group statistics does not yield much improvement. Therefore, a more complex model has been developed [30, 31], which 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 tools and prototypes.

3.1.2.2 Expert systems

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

- Wisdom & Sense [61] is an intrusion–detection tool 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 [11] is 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.

3.1.2.3 Neural networks

Neural networks are algorithmic techniques used to learn the relationship between the two sets of information, and then to "generalize" to obtain new input–output pairs in a reasonable way.

Neural networks could theoretically be used in knowledge–based intrusion–detection tools to learn of 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 propose a reasoning or an explanation of the attack.

Therefore, the main use of neural networks for intrusion detection is 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, 51]. Therefore, the advantage of using neural networks over statistics resides 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 [6]. 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 a very small fraction of users whose behavior is unpredictable. Neural networks are still a computationally intensive technique, and are not widely used in the intrusion–detection community.

3.1.2.4 User Intention Identification

User Intention Identification [57] is a technique developed during the SECURENET project [56]. The goal of this technique is to model the normal behavior of users by the set of high–level tasks they have to perform on the system. These tasks are then refined into actions, which in turn are related to the audit events observed on the system. The analyzer keeps a set of tasks that each user can perform. Whenever an action occurs that does not fit the task pattern, an alarm is issued. To our knowledge, this technique has only been used in the SECURENET project.

3.1.2.5 Computer Immunology

Computer immunology has been described by Forrest *et al.* [16]. This technique attempts to build a model of normal behavior of the UNIX network services, rather than that 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 that are 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 exhaustive enough. Extensions to reach that goal are currently being developed [7, 8]. 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.

3.2 Passive versus active intrusion detection

Most intrusion–detection tools are passive, meaning that when an attack is detected, an alarm is generated, but no countermeasure is actively applied to thwart the attack. This made sense in a research context, where such tools might possibly generate a large number of false alarms, having a negative impact on the availability of the system. We are aware of only one tool with early countermeasure capability, NetProbe [49], which monitors a network for undesired connections and terminates them on the spot.

A number of intrusion–detection tools based on periodic analysis have had some active capability added when a security issue was detected in the configuration of the system. These tools generate scripts both to suppress the vulnerability (by changing the permissions on a file system, for example) and to restore the system to its previous state. Hence the application of a countermeasure is made safer by the capability of reverting quickly to a former state in the event of an abnormality. An example of this category of tools is Secure Network's Ballista[2] [52].

With the arrival of intrusion–detection products, the countermeasure element has become increasingly preeminent. Tools such as RealSecure [27], NetRanger [63], and WebStalker [22] now include the capability of cutting connections that carry attacks, blocking traffic from the hosts from which attacks originate, or reconfiguring other equipment such as firewalls or routers. Such proactive security strategies are gaining momentum as intrusion–detection products are becoming more reliable.

3.3 Host–based versus network–based intrusion detection

Host–based intrusion detection is the first area explored in intrusion detection. When the first intrusion–detection tools were designed, the target environment was a mainframe computer, and all users were local to the system considered. This simplified greatly the intrusion–detection task, as interaction from outside was rare. The intrusion–detection tool analyzed the audit information provided by the mainframe, either locally [40] or on a separate machine [53], 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. The first research in this area was getting 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 launching 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 exchanging a raw audit trail over the network *à la* Stalker [22], or issuing alarms coming from a local analysis [54]. Both solutions incur costs; transferring audits has potentially 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 these 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, these tools are often 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 [54] uses Haystack [53], which runs on each host to detect local attacks, and NSM [23] 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, which monitor the most critical elements of an organization's presence on the Internet. These products monitor firewalls (NetStalker [22]), web servers (WebStalker [22]) or routers (NetRanger [63]), looking for evidence of attacks in the very specific context of these network elements.

3.3.1 Host–based information sources

Host audit sources are the only way to gather information about 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.

[2] Ballista is not an intrusion–detection system, but a vulnerability search tool similar to Satan. It analyzes the network to detect vulnerabilities in its configuration, but does not perform real-time monitoring.

3.3.1.1 System sources

All operating systems have commands to obtain a snapshot of information on what is happening. In a UNIX environment, examples of such commands are *ps, pstat, vmstat, getrlimit*. These commands provide very precise and focused information about the events because they examine the kernel memory directly. However, they are very difficult to use for continuous audit collection in intrusion–detection tools, because they do not offer a structural way of collecting and storing the audit information.

3.3.1.2 Accounting

Accounting is one of the oldest sources of information on system behavior. It provides information on the consumption of shared resources by the users of the system. Resources are, for example, processor time, memory, disk or network usage, and applications launched. Accounting is found everywhere, from network equipments to mainframes to 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 has also a number of drawbacks, which make it untrustworthy for security purposes. By default, accounting files are sometimes located in the same disk partition as the */tmp* directory. Users then simply have to fill the partition up to 90%, and accounting stops. Although this is easily fixed, more important drawbacks include:

- Lack of parameterization. Accounting is either on or off, but cannot be activated for selected users only.
- Lack of precise time stamp. The date included in the accounting record is precise to the second, and that does not allow the sorting and resequencing of actions. As commands in the accounting file are logged in the order in which they terminate, this lack of precision does not allow one to obtain the list of commands in the order in which they were actually submitted. Command sequencing might be important information for some intrusion–detection techniques.
- Lack of precise command identification. Only the first 8 characters of the name of the command submitted by the user are stored in the accounting record. Important path information (to fully identify the command) and command line arguments are lost. This would render the detection of Trojan horses well as the use of knowledge–based intrusion–detection techniques impossible.
- Absence of system daemon activity. Accounting keeps information only about binary executables that terminate. In this case, continuously running executables such as system daemons (e.g. sendmail) are never audited.
- Delay of obtaining information. The accounting record is written when the application terminates. Therefore, intrusion detection can only perform damage control as the intrusion would already have been carried out.

Owing to these drawbacks, accounting is never used for knowledge–based intrusion–detection, and rarely for behavior–based intrusion detection. The statistical and neural network modules of Hyperview [6] made use of accounting information as a complement to security audit but not as a substitute for it.

3.3.1.3 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 not known for its security, as syslog daemons on several UNIX operating systems have been the subject of CERT documents [4] showing the exploitation of buffer overflows in the syslog daemon 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*, and this also includes 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, an example of this approach being Swatch [21]. 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 [21] reduces the burden of the system administrator by correlating messages (several machines reporting that an nfs server is down would be aggregated into one) and highlighting security–related ones.

3.3.1.4 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 [60], 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 audits 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 [60]. The rationale for this security model set forth 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 processes launched by the users. Compared with a full system call trace, the audit trail provides a 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 great deal 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–grain parameterization of the information gathered according to user, class, audit event, or failure or success of the system call;
- 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;

- difficulties 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, e.g about file accesses, processes executed;

- difficulties 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;

- 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 an overwhelming number of host–based intrusion–detection prototypes and tools because it is currently the only reliable mechanism for gathering detailed information on the actions taken on an information system. Work has conducted by several groups [20, 42, 46, 59] to define what should be in the security audit trail as well as a common format for audit trail records, but this is an ongoing research effort.

3.3.2 Network–based information sources

3.3.2.1 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 [56] to use SNMP V1 common MIB for Ethernet and TCP/IP. Other projects also target the use of SNMPv2 and v3 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 that are 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, and no correlation was observed between the two interfaces.

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 we have not tried ICMP attacks [3] 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 rise of SNMPv3, new projects are taking advantage of its features for intrusion–detection tools [32].

3.3.2.2 Network packets

As the popularity of network sniffers for gathering information has grown in the attackers community, it is also regarded today 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 [1]. 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, but does not take into account session information, which could span over 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. Moreover, this analysis of the higher levels of the protocol is also dependent on the particular machine being protected, as implementations of the protocols are not identical from one network stack to another.

This approach addresses several problems:

- The detection of network–specific attacks. There are a number of network attacks, particularly denial–of–service, which 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 collected entirely on a separate machine, with no knowledge of the rest of the network. Therefore, installation of such tools is facilitated because both in terms of configuration and performance they do not impact the entire environment.
- 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 the detection of attacks against hosts 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 is discovered. There is no reliable link between 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), placing the sniffer is not obvious. Some tools are located on switches, other at gateways between the protected system and the outside world. The former gives better audit information but also has a higher cost. One has to realize, however, that switched networks are also much less vulnerable to sniffer attacks [5, 47] and are actually a recommended solution to improve the security of a network.

- Encryption makes it impossible to analyze the payload of the packets, and therefore to hide a considerable amount of important information on these tools. Also, it is possible, even without encryption, to obfuscate the contents of the packet to evade detection 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. As the network stacks of these commercial operating systems are vulnerable to attacks, so is the intrusion–detection system.

Network packets are now the source of information used by several recent commercial products [27, 63], and several projects in the research community have taken this track as well [43, 48, 49, 58]. A recent evaluation of these products by Ptacek and Newsham [47] 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, Ptacek and Newsham [47] show that IP fragmentation is not handled well, and the use of wildcards and control sequences in protocols such as http makes it possible to evade detection by signatures.

Research is also being conducted in this area. After *IDES* and *NIDES*, SRI is now developing a prototype called *Emerald* [45] to deal with analysis of network traffic. Other network sniffers such as *Bro* [43] or *Network Flight Recorder* [48] have been developed as network data acquisition tools and therefore do not support intrusion–detection *per se,* but they make possible the use of network information by acting as a CIDF E–box.

3.4 Static versus dynamic intrusion detection

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

These static tools assess the security level of the current configuration of the environment. Examples of these tools include *COPS* [13, 15] and *Tiger* [50] for host environments, and Satan [14] and Ballista [52] 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, or 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. Then, 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 for a day or so, for it has been shown that active exploitation of vulnerabilities over the Internet can take less than one day.

Such tools, as well as others specifically developed for that purpose (e.g. *Tripwire* [33] or *ATP* [62]) 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 or Tiger 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. However, these checks are periodic, and in this sense they do not fulfill the timeliness and performance requirements of intrusion–detection systems. Therefore, we do not consider them in the scope of this paper as being full–fledged intrusion–detection systems, as mentioned in Section 2.2.

Dynamic intrusion–detection tools monitor the actions that take place on the system. This monitoring takes place either in real time or in batch, 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.

4 Intrusion–detection tools

Table 1 presents a selection of intrusion–detection tools that we have encountered and shows a taxonomy of their components. The selection merely illustrates the notions described in this paper, and does not imply any judgment of the quality of the tool, product or method on our part. Also, the number of tools and prototypes being developed throughout the world is such that an exhaustive list is difficult to compile, and we shall continue to add entries to this table.

Table 1: Panorama of intrusion–detection systems

IDS origin	IDS Name	Time Frame	Ref	Knowledge–based IDS				Behavior–based IDS				HB	NB
				ES	SA	PN	STA	Stat	ES	NN	UII		
Univ. Namur	ASAX	1990–1997	[20]	X								X	
AT&T	ComputerWatch	1987–1990	[11]						X			X	
USAF	Haystack	1987–1990	[53]					X				X	
	DIDS	1989–1995	[54]	X				X				X	X
CS Télécom	Hyperview	1990–1995	[6]	X				X		X		X	
SRI	IDES	1983–1992	[10]	X				X				X	
	NIDES	1992–1995	[29]	X				X				X	
	Emerald	1996–	[45]		X			X					X
Purdue Univ.	IDIOT	1992–1997	[34]			X						X	
UCDavis	NSM	1989–1995	[23]		X			X					X
	GrIDS	1995–	[58]						X				X
LANL	W&S	1987–1990	[61]						X			X	
	Nadir	1990–	[28]	X					X			X	
WheelGroup	NetRanger	1995–	[63]		X								X
ISS	RealSecure	1995–	[27]		X								X
Securenet Cons	SecureNet	1992–1996	[56]	X				X		X	X	X	
Haystack labs	Stalker	1995–	[22]		X							X	
	WebStalker	1997–	[22]						X			X	
UCSB	STAT	1991–1992	[44]				X					X	
	USTAT	1992–1993	[26]				X					X	
Stanford Univ.	Swatch	1992–1993	[21]	X								X	
MCNC and NCSU	JiNao	1995–	[32]	X				X					X

Abbreviations used: ES: expert system; SA: signature analysis; PN: Petri net; STA: state transition analysis; Stat: statistics; NN: neural network; UII: user intent; HB: host–based, and NB: network–based.

Table 1 contains more host–based intrusion–detection systems than network–based intrusion–detection systems. However, this does not represent the trend in intrusion detection, which is towards network information and protection of the infrastructure. There are more network–based intrusion–detection products [27, 63] commercially available today than host–based ones [22], as well as recent projects still under development. The main reasons for this are probably the ease of installing a network–based tool (no user workstation manipulation required), the performance degradation experienced by systems when an audit is started, and the difficulty and cost of managing a large–scale host audit infrastructure.

Table 1 also shows that, even though many techniques have been explored for intrusion detection, most commercial products available today implement one and only one technique, and that the majority of the recent ones [22, 27, 63] use signatures, for two reasons:

- The knowledge–based approach is easier to implement than the behavior–based one. In fact, the cost in terms of false alarms of the behavior–based techniques has hitherto made them inappropriate for commercial intrusion detection.
- The speed factor is essential in processing the audits and preempts the expressiveness of the technique. Therefore, signatures are used instead of rules.

In addition, the collaborative approach of correlating several analyzers to improve the intrusion–detection system has been studied [10, 56] and is retained as part of the ongoing work in CIDF, but has not yet been incorporated in the commercial world.

5 The reusability issue

One of the greatest challenges faced by intrusion–detection products and prototypes is the possibility to reuse existing components in an environment different from the original one. This is due mainly to incompatible audit and alarm formats.

A working group has been created under the auspices of the Defence Advanced Research Projects Agency (DARPA) to develop a common intrusion–detection framework (CIDF) [59]. This work aims primarily at coordinating the many projects funded by DARPA that are concerned with intrusion detection, and ensuring that the tools developed are able to inter–operate. The CIDF description of an intrusion–detection system (Fig. 4) is more detailed than the one above and defines all the possible *roles* of components that can comprise an intrusion–detection system. The interfaces of each of these component roles are then defined, so that any CIDF–compliant box can be integrated into a larger tool. The CIDF group is currently in the process of joining the Internet Engineering Task Force to make their work a standard in the Internet world.

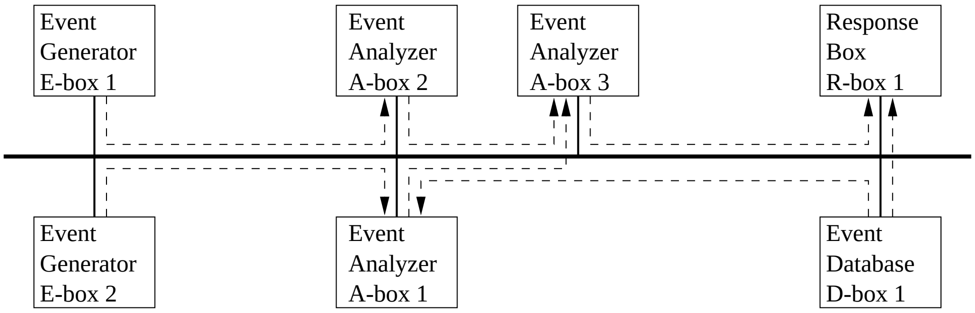


Figure 4: CIDF description of an intrusion–detection system.

Figure 4 does not include the system that is monitored. Obviously, the boxes run on hardware of some kind, most likely the system that produces the events in the case of the event box, or either the monitored system or a specific hardware in the case of the other boxes.

CIDF defines four kinds of components for an intrusion–detection system and very specific roles for each of them. All these components deal with *gidos* (generalized intrusion–detection objects), which are represented via a standard common format (s–expressions). Gido streams are represented as dashed arrows in Fig. 4. Gidos carry information that is moved around in the intrusion–detection system. From a semantic point of view, gidos currently represent either audit events that occurred in the system or an analysis of those audit events (referred to henceforth as alarms).

- *Event boxes (E–boxes)* generate audit events that are processed by the intrusion–detection system. E–boxes typically run on the system that generates the audit events, where they collect the audit events and make them available to other components of the intrusion–detection system. E–boxes produce audit events but do not consume them. Their task is to sample the particular environment for which they are specialized, and turn occurrences in that environment into CIDF gidos for use by other components. Figure 4 shows two event–generator boxes delivering audit events to two analyzers.

- *Analysis boxes (A–boxes)* process (similar to the detector component) events from the E–boxes to create alarms. Analyzers take in gidos and analyze their significance (policy violations, anomalies, intrusions). Their conclusions are turned out as alarms. In Fig. 4, two of the three A–boxes receive audits from E–boxes, whereas the third one aggregates information and passes it to the countermeasures.

- *Database boxes (D–boxes)* store events for later retrieval. D–boxes are gidos archivers. They receive events sent by E–boxes or A–boxes, store them for long–term keeping, and provide a retrieval and query service. For example, a D–box would store the audit and alarm streams described in Fig. 2. Configuration and database are private to each A–box and must be maintained independently. In Fig. 4, the D–box provides gidos to one of the analyzers and to the countermeasures.

- *Response boxes (R–boxes)* (sometimes also called countermeasure boxes) apply countermeasures to the system according to the alarms generated. They are the active arm of the intrusion–detection system; they enforce the decisions taken in response to attacks. In Fig. 4, an R–box takes its input from the third A–box.

CIDF is work–in–progress. The most important contribution of CIDF is to define interfaces by which the different kinds of boxes can communicate, thus introducing reusability of components in intrusion detection. It is a fact that as of today, a large number of research prototypes and products have been developed, but these heterogeneous developments do not allow the reusability of techniques or tools in different environments.

6 Conclusion and future directions

Intrusion detection is currently attracting 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, we 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. This taxonomy clearly highlights the properties of these intrusion–detection systems and covers the past and current developments adequately.

Information sources for these tools are currently 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 to protect the infrastructure, rather than the end–user station, and this paradigm has introduced the usage of network sniffers that analyze packets. As shown, there are still 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.

There are also a number of open research issues concerning the analysis of the audit trail. Signature analysis is clearly in the commercial domain now, but it has been shown to be insufficient to detect 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] Steven M. Bellovin and William R. Cheswick. Network firewalls. *IEEE Communications MAGAZINE*, 32(9):50–57, September 1994.

[2] James Cannady and Jay Harrell. A comparative analysis of current intrusion detection technologies. In *Proceedings of the fourth Technology for Information Security Conference'96 (TISC'96)*, Houston, TX, May 1996.

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

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

[5] 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.

[6] 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.

[7] Hervé Debar, Marc Dacier, and Andreas Wespi. Fixed vs. variable-length patterns for detecting suspicious process behavior. Technical Report RZ 3012, IBM Zurich Research Laboratory, Säumerstrasse 4, CH-8803 Rüschlikon, Switzerland, April 1998. Submitted to Esorics'98.

[8] Hervé Debar, Marc Dacier, and Andreas Wespi. Reference audit information generation for intrusion detection systems. In Reinhard Posch, editor, *Information Systems Security*, Vienna, Austria and Budapest, Hungaria, August 31–September 4 1998. Chapman & Hall.

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

[10] Dorothy E. Denning and Peter G. Neumann. Requirements and model for IDES - a real-time intrusion detection expert system. Technical report, Computer Science Laboratory, SRI International, Menlo Park, CA, 1985.

[11] 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.

[12] Mansour Esmaili, Rei Safavi-Naini, and Josef Pieprzyk. Computer intrusion detection: A comparative survey. Technical Report 95-07, Center for Computer Security Research, University of Wollongong, Wollongong, NSW 2522, Australia, May 1995.

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

[14] 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.

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

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

[17] Jeremy Frank. Artificial intelligence and intrusion detection: Current and future directions. In *Proceedings of the 17th National Computer Security Conference*, Baltimore, MD, October 1994.

[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] 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)*, number 648 in Lecture Notes in Computer Science, Toulouse, France, November 1992. Springer-Verlag, Berlin Germany.

[21] 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.

[22] Haystack Labs, Inc. Stalker. Available from the company's website at http://www.haystack.com/stalk.htm, 1997.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

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

[28] Kathleen Jackson, David DuBois, and Cathy Stallings. An expert system application for network intrusion detection. In *Proceedings of the 14th National Computer Security Conference*, pages 215–225, November 1991.

[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, NC 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] 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.

[36] Gunar Liepins and H. S. Vaccaro. Anomaly detection: Purpose and framework. In *Proceedings of the 12th National Computer Security Conference*, pages 495–504, October 1989.

[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. Automated audit trail analysis and intrusion detection: A survey. In *Proceedings of the 11th National Computer Security Conference*, Baltimore, MD, October 1988.

[39] Teresa F. Lunt. A survey of intrusion detection techniques. *Computers & Security*, 12(4):405–418, June 1993.

[40] 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.

[41] Noelle McAuliffe, Dawn Wolcott, Lorrayne Schaefer, Nancy Kelem, Brian Hubbard, and Theresa Haley. Is your computer being misused? a survey of current intrusion detection system technology. In *Proceedings of the Sixth Annual Computer Security Applications Conference*, pages 260–72, Tucson, AZ, December 1990. IEEE Computer Society Press, Los Alamitos, CA.

[42] 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.

[43] 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.

[44] Phillip Porras and Richard Kemmerer. Penetration state transition analysis – a rule-based intrusion detection approach. In *Proceedings of the Eighth Annual Computer Security Applications Conference*, pages 220–229. IEEE Computer Society press, IEEE Computer Society press, November 1992.

[45] 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.

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

[47] 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.

[48] 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.

[49] 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.

[50] 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.

[51] 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, SAS Institute.

[52] Secure Networks, Inc. Ballista security auditing system. Available from the company's website at http://www.securenetworks.com/ballista/ballista.html, 1997.

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

[54] 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.

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

[56] 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.

[57] Thomas Spyrou and John Darzentas. Intention modelling: Approximating computer user intentions for detection and prediction of intrusions. In S.K. Katsikas and D. Gritzalis, editors, *Information Systems Security*, pages 319–335, Samos, Greece, May 1996. Chapman & Hall.

[58] 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.

[59] 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.

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

[61] 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.

[62] 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.

[63] WheelGroup Corporation. Brochure of the netranger intrusion detection system. Available from the company's website at http://www.wheelgroup.com/netrangr/netranger_broch.html.