ELSEVIER

Computer Networks 31 (1999) 805–822

COMPUTER NETWORKS

Towards a taxonomy of intrusion-detection systems

Hervé Debar *, Marc Dacier [1], Andreas Wespi [2]

IBM Research Division, Zurich Research Laboratory, Säumerstrasse 4, CH-8803 Rüschlikon, Switzerland

Abstract

Intrusion-detection systems aim at detecting attacks against computer systems and networks, or against information systems in general, as it is difficult to provide provably secure information systems and maintain them in such a secure state for their entire lifetime and for every utilization. Sometimes, legacy or operational constraints do not even allow a fully secure information system to be realized at all. Therefore, the task of intrusion-detection systems is to monitor the usage of such systems and to detect the apparition of insecure states. They detect attempts and active misuse by legitimate users of the information systems or external parties to abuse their privileges or exploit security vulnerabilities. In this paper, we introduce a taxonomy of intrusion-detection systems that highlights the various aspects of this area. This taxonomy defines families of intrusion-detection systems according to their properties. It is illustrated by numerous examples from past and current projects. © 1999 Elsevier Science B.V. All rights reserved.

Keywords: Security; Taxonomy; Intrusion-detection

1. Introduction

Since the seminal work by Denning in 1981 [10], many intrusion-detection prototypes have been created. Sobirey maintains a partial list of 59 of them [58]. Intrusion-detection systems have emerged in the field of computer security 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 both technically difficult and economically costly to build and maintain computer systems and networks that are not susceptible to attacks.

This paper introduces a taxonomy of intrusion-detection systems at a time when commercial tools are increasingly becoming available. Our taxonomy draws examples from research prototypes as well as commercial products to illustrate the most prominent 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, many of these products are now also available for Windows NT, which has been more widely deployed in organizations and has been subjected to enhanced scrutiny by the security and

* Corresponding author. E-mail: deb@zurich.ibm.com.
[1] E-mail: dac@zurich.ibm.com.
[2] E-mail: anw@zurich.ibm.com.

underground communities. An additional consideration is that the intrusion-detection commercial market has experienced considerable activity since WheelGroup corporation was acquired by Cisco Systems, followed by the cascade acquisition of Haystack Labs, Secure Networks and Trusted Information Systems by Network Associates.

This paper does not purport to be an exhaustive survey of intrusion-detection tools, techniques, projects, and products. Several surveys have already been published [2,13,18,37,39,40,42], 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 in 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 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 an information system being monitored by an intrusion-detection system. It can be a workstation, a network element, a server, a mainframe, a firewall, a web server, an enterprise network, etc.

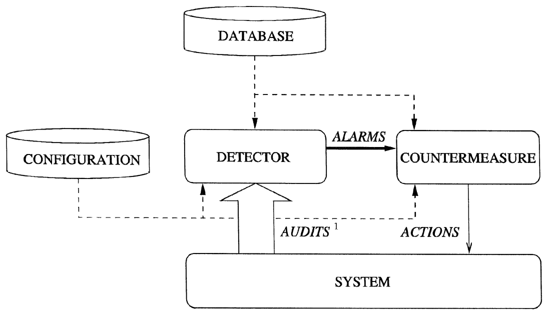
The term *audit* denotes information provided by a system concerning its inner workings and behavior. Examples of audits include but are not limited to C2 audit trail, accounting, and syslog in the UNIX world, Syslog in the MVS world, the event log in Windows NT, and incident tickets in X25 networks. 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 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 (a knowledge base of attacks, for example), configuration information about the current state of the system, and audit information describing the events that occur on the system. The role of the detector is to eliminate unnecessary information from the audit trail and present a synthetic view of the security-related actions taken by users. A decision is then made to evaluate the probability that these actions can be considered symptoms of an intrusion.



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

Fig. 1. Very simple intrusion-detection system.

2.3. Efficiency of intrusion-detection systems

The following three measures to evaluate the efficiency of an intrusion-detection system have been highlighted in Ref. [48]:

- 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 on top of 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 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.

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* distinguishes among intrusion-detection systems based on the kind of

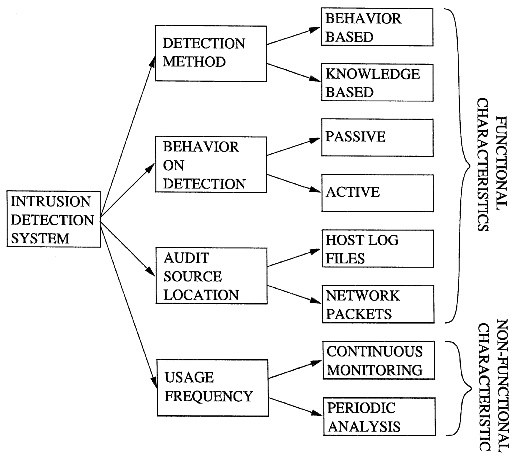


Fig. 2. Characteristics of intrusion-detection systems.

input information they analyze. This input information can be audit trails, system logs or network packets.

Usage frequency is an orthogonal concept. Certain intrusion-detection systems have real-time continuous monitoring capabilities, whereas others must be run periodically.

The three first axes are grouped in the category ‘‘functional characteristics’’ because they refer to the internal workings of the intrusion-detection engine, namely its input information, its reasoning mechanism, and its interaction with the information system. The fourth characteristic distinguishes RTID (Real-Time Intrusion Detection) from scanners used for security assessment. These scanners are sometimes attached to the intrusion-detection area, and we must differentiate discriminate between them and ‘‘real’’ intrusion-detection systems.

3.1. Knowledge-based versus behavior-based intrusion detection

There are two complementary trends in intrusion detection, (1) the search for evidence of attacks based on knowledge accumulated from known attacks, and (2) the search for deviations from a model of unusual behavior based on observations of a system during a known normal state. The first trend is often referred to as *misuse detection* [30,35] or *detection by appearance* [59]. The second trend is referred to as *anomaly detection* [30] or *detection by behavior* [59]. 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. Both terms are defined more precisely in the following subsections.

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 them. 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* requires that their knowledge of attacks be updated regularly.

Advantages of the knowledge-based approaches are that they have the potential for very low false alarm rates, and that the contextual analysis proposed by the intrusion-detection system is detailed, which makes 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 abreast 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. [3] Knowledge-based approaches also have to face the generalization issue. Knowledge about attacks is very focused 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 [38] 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 [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

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

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 [20]. Knowledge about the behavior of an attacker includes the attacker's goals, the actions required to reach these goals, and whether his usage of the system 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 the following limitations:

- *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 input. Sometimes the information required is not available in the audits. Also, there may be many ways to exploit a given vulnerability, which leads 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 poor.

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

3.1.1.2. Signature analysis. Signature analysis follows exactly the same knowledge-acquisition approach as expert systems, but the knowledge acquired 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 [23,28,66]. 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 all possible facets of the attacks as 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 tool has been ported.

3.1.1.3. Petri nets. To represent signatures of intrusions, IDIOT [35], a knowledge-based intrusion-detection system developed at Purdue University, uses Colored Petri Nets (CPN). The advantages of CPNs 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 CPNs, quite complex signatures can be written easily. However, matching a complex signature against the audit trail may become computationally very expensive.

Fig. 3 shows a simple example of a CPN that issues an alarm if the number of unsuccessful login attempts within one minute exceeds four. The transition, represented by a vertical bar, from state s1 to s2

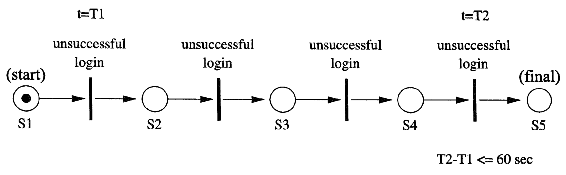


Fig. 3. Four failed login attempts within one minute.

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 s. 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, a technique proposed by Porras and Kemmerer [47], was implemented first in UNIX [27] 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. If 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 the 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 what is acceptable behavior. As a result, the behavior profile contains intrusive behavior, which is then not detected as anomalous.

3.1.2.1. Statistics. The most widely used tool to build behavior-based intrusion-detection systems is statistics [25,26,32]. 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 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 [31,32], 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. The following are two examples of approaches that have been taken in this area:

- Wisdom & Sense [64] 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 [12] 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 algorithms that learn about the relationship between input–output vectors and ''generalize'' them to obtain new input–output vectors in a reasonable way. Neural networks could theoretically be used in knowledge-based intrusion-detection tools to learn attack traces 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 [19,54]. 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 that use a neural network to predict the behavior of users [7]. 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 [60] is a technique developed during the SECURENET project [59]. This technique models 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. [17]. This technique builds 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 take unusual execution paths. 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 [8,9]. 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 [52], 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 if 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 [55]. [4]

With the arrival of intrusion-detection products, the countermeasure element has become increasingly preeminent. Tools such as RealSecure [28], NetRanger [66], and WebStalker [23] 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 to have been 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 [41] or on a separate machine [56], 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 to get host-based intrusion-detection systems to communicate [30]. 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 by exchanging a raw audit trail over the network *à la* Stalker [23], or by issuing alarms that come from a local analysis [57]. Both solutions incur costs; transferring audits has a potentially 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, 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 multihost environment, i.e. a network of workstations. DIDS [57] uses Haystack [56] running 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]), routers (NetRanger [66] or the newer documentation after Wheelgroup's acquisition by Cisco [6]), 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 sub-

[4] 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.

vert either the audit trail or the intrusion-detection system itself.

3.3.1.1. System sources. All operating systems have commands to obtain a snapshot of information on the processes currently active on the computer. In a UNIX environment, examples of such commands are *ps, pstat, vmstat, getrlimit*. These commands provide very precise information about 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 structured 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 equipment 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, 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. 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, which 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 as 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 not used for knowledge-based intrusion-detection, and rarely for behavior-based intrusion detection. The statistical and neural network modules of Hyperview [7] 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, which has prompted 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 [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 (e.g. if several machines report that an nfs server is down, these reports 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 [63], 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 [63]. The rationale for this security model sets 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 or 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;
- 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 by users of an information system. Work was conducted by several groups [21,43,49,62] 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 traffic at various network interfaces and at different layers of the network). This section describes experiments performed within the SECURENET project [59] 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 [33].

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, 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 dampened its interest to the intrusion-detection community. However, with the rise of SNMPv3, new projects are taking advantage of its features for intrusion-detection tools [33].

3.3.2.2. Network packets. As the popularity of network sniffers for gathering information has grown in the attacker 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 take place today over a network. 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 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:

- Detection of network-specific attacks. There are a number of network attacks, particularly denial-of-service, that cannot be detected in a timely fashion by searching for audit information on the host, but only by analyzing network traffic.

- 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.
- 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), it is not obvious where the sniffer should best be placed. Some tools are located on switches, other at gateways between the protected system and the outside world. The former yields better audit information but is also more costly. One has to realize, however, that switched networks are also much less vulnerable to sniffer attacks [5,50] and are actually recommended 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 [6,28,66], and several projects in the research community have taken this track as well [46,51,52,61]. A recent evaluation of these products by Ptacek and Newsham [50] shows that the sniffer approach, or at least the current implementations, has flaws that make it possible for a skilled attacker to evade detection. In particular, Ptacek and Newsham [50] 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 signatures.

Research is also being conducted in this area. After *IDES* and *NIDES*, SRI is now developing a prototype called *Emerald* [48] to deal with analysis of network traffic. Other network sniffers such as *Bro* [46] or *Network Flight Recorder* [51] have been developed as network data acquisition tools and therefore do not support intrusion detection *per se*.

3.4. 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.

These static tools assess the security level of the current configuration of the environment. Examples of these tools include *COPS* [14,16] and *Tiger* [53] for host environments, and Satan [15] and Ballista [55] (now called CyberCop Scanner [44] since the buyout of Secure Networks by Network Associates Inc.) 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 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* [34] or *ATP* [65]) 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 [34] 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 defined in Section 2.2.

Dynamic intrusion-detection tools monitor the actions that take place on the system. 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 implies no 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 contains more host-based intrusion-detection systems than network-based intrusion-detection systems. However, this is not the trend in intrusion detection, which is towards network information and protection of the infrastructure. There are more network-based intrusion-detection products [28,66] commercially available today than host-based ones [23,45], 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 [23,28,66] 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.
- Speed 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-detec-

Table 1
Panorama of intrusion-detection systems

IDS origin	IDS Name	Time	Ref	Knowledge-based IDS				Behavior-based IDS				HB	NB
		Frame		ES	SA	PN	STA	Stat	ES	NN	UII		
Univ. Namur	ASAX	1990–1997	[21]	X								X	
AT&T	ComputerWatch	1987–1990	[12]						X			X	
USAF	Haystack	1987–1990	[56]					X				X	
	DIDS	1989–1995	[57]	X				X				X	X
CS Télécom	Hyperview	1990–1995	[7]	X				X		X		X	
SRI	IDES	1983–1992	[11]	X				X				X	
	NIDES	1992–1995	[30]	X				X				X	
	Emerald	1996–	[48]		X			X					X
Purdue Univ.	IDIOT	1992–1997	[35]			X						X	
UC Davis	NSM	1989–1995	[24]		X			X					X
	GrIDS	1995–	[61]						X				X
LANL	W&S	1987–1990	[64]						X			X	
	Nadir	1990–	[29]	X					X			X	
Cisco/ WheelGroup	NetRanger	1995–	[6,66]		X								X
ISS	RealSecure	1995–	[28]		X								X
Securenet Consortium	SecureNet	1992–1996	[59]	X				X		X	X	X	
Network Associates Inc	Stalker	1995–	[23]		X							X	
	WebStalker CyberCop Server	1997–	[23,45]						X			X	
UCSB	STAT	1991–1992	[47]				X					X	
	USTAT	1992–1993	[27]				X					X	
Stanford Univ.	Swatch	1992–1993	[22]	X								X	
MCNC and NCSU	JiNao	1995–	[33]	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.

tion system has been studied [11,59] 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 capability 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) [62]. 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 interoperate. 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

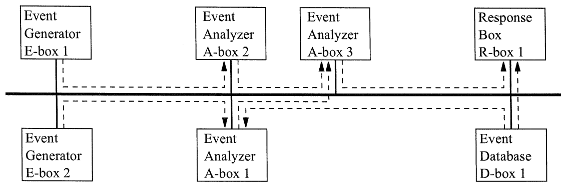


Fig. 4. CIDF description of an intrusion-detection system.

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.

Fig. 4 does not include the system being 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 on 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 semantics point of view, gidos currently represent either audit events that occurred in the system or an analysis of those audit events (henceforth referred to 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 to turn occurrences in that environment into CIDF gidos for use by other components. Fig. 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 Section 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 made 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 the 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.

Currently, the CIDF effort is giving birth to an IETF working group chartered to create standards in the intrusion-detection area. The current draft charter being discussed states that ‘‘the purpose of the Intrusion Detection Working Group is to define data formats and exchange procedures for sharing information of interest to intrusion-detection systems and their management infrastructure.’’ The output of the working group should include a requirements document, a common language specification, and a framework document. As the charter is still undergoing discussion, interested readers are referred to the CIDF mailing list (cidf@seclab.cs.ucdavis.edu) for up-to-date information.

6. Conclusion and future directions

Intrusion detection is currently attracting considerable interest from both the research community and commercial companies. Research prototypes continue to be created, 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 highlights the properties of 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 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] S.M. Bellovin, W.R. Cheswick, Network firewalls, IEEE Communications Magazine 32 (9) (1994) 50–57.

[2] J. Cannady, J. Harrell, A comparative analysis of current intrusion detection technologies, Proc. 4th Technology for Information Security Conf. (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] W.R. Cheswick, S.M. Bellovin, Firewalls and Internet Security – Repelling the Wily Hacker, Professional Computing Series, Addison-Wesley, Reading, MA, 1994.

[6] Cisco Systems Inc, NetRanger – enterprise-scale, real-time, network intrusion detection system, available from the company's website at http://www.cisco.com/warp/public/751/netranger/netra_ds.htm, 1998.

[7] H. Debar, M. Becker, D. Siboni, A neural network component for an intrusion detection system, Proc. 1992 IEEE Computer Society Symp. on Research in Security and Privacy Oakland, CA, May 1992, pp. 240–250.

[8] H. Debar, M. Dacier, A. Wespi, Fixed versus 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.

[9] H. Debar, M. Dacier, A. Wespi, Reference audit information generation for intrusion detection systems, in: R. Posch, G. Papp (Eds.), Proc. 14th International Information Security Conference IFIP SEC'98, Chapman and Hall, Vienna, Austria and Budapest, Hungaria, August 31–September 4, 1998.

[10] D. Denning, An intrusion-detection model, IEEE Transactions on Software Engineering 13 (2) (1987) 222–232.

[11] D.E. Denning, P.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.

[12] C. Dowell, P. Ramstedt, The ComputerWatch data reduction tool, Proc. 13th National Computer Security Conf., Washington, DC, October 1990, pp. 99–108.

[13] M. Esmaili, R. Safavi-Naini, J. 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.

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

[15] D. Farmer, W. 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] D. Farmer, E. Spafford, The cops security checker system, Proc. Summer USENIX Conf., Anaheim, CA, June 1990, pp. 165–170.

[17] S. Forrest, S.A. Hofmeyr, A. Somayaji, Computer immunology, Communications of the ACM 40 (10) (October 1997) 88–96.

[18] J. Frank, Artificial intelligence and intrusion detection: current and future directions, Proc. 17th Nat. Computer Security Conf. , Baltimore, MD, October 1994.

[19] P. Gallinari, S. Thiria, F. Fogelman-Soulie, Multilayer perceptrons and data analysis, Proc. IEEE Annual Int. Conf. on Neural Networks (ICNN88), Vol. I, San Diego, CA, July 1988, pp. 391–399.

[20] T. Garvey, T. Lunt, Model-based intrusion detection, Proc. 14th National Computer Security Conf., October 1991, pp. 372–385.

[21] N. Habra, B. Le Charlier, A. Mounji, I. Mathieu, Asax: software architecture and rule-based language for universal audit trail analysis, in: Y. Deswarte, G. Eizenberg, J.-J. Quisquater (Eds.), Proc. 2nd European Symp. on Research in Computer Security (ESORICS), Toulouse, Berlin, Lecture Notes in Computer Science, vol. 648, Springer, Berlin, November 1992.

[22] S.E. Hansen, E.T. Atkins, Automated system monitoring and notification with swatch, Proc. 7th Systems Administration Conf. (LISA'93), Monterey, CA, November 1993.

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

[24] L.T. Heberlein, G.V. Dias, K.N. Levitt, B. Mukherjee, J. Wood, D. Wolber, A network security monitor, Proc. Symp. on Research in Security and Privacy, IEEE Computer Society Press, Los Alamitos, CA, Oakland, CA, May 1990, pp. 296–304.

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

[26] P. Helman, G. Liepins, W. Richards, Foundations of intrusion detection, Proc. 5th Computer Security Foundations Workshop Franconic, NH, June 1992, pp. 114–120.

[27] K. Ilgun, Ustat: a real-time intrusion detection system for Unix, Proc. IEEE Symp. on Research in Security and Privacy Oakland, CA, May 1993, pp. 16–28.

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

[29] K. Jackson, D. DuBois, C. Stallings, An expert system application for network intrusion detection, Proc. 14th National Computer Security Conf., November 1991, pp. 215–225.

[30] R. Jagannathan, T. Lunt, D. Anderson, C. Dodd, F. Gilham, C. Jalali, H. Javitz, P. Neumann, A. Tamaru, A. 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.

[31] H. Javitz, A. Valdes, The SRI IDES statistical anomaly detector, Proc. IEEE Symp. on Research in Security and Privacy, May 1991, pp. 316–326.

[32] H.S. Javitz, A. Valdez, T.F. Lunt, A. Tamaru, M. Tyson, J. 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.

[33] Y.F. Jou, F. Gong, C. Sargor, S.F. Wu, W.R. 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.

[34] G.H. Kim, E.H. Spafford, The design and implementation of tripwire: A file system integrity checker, in: J. Stern (Ed.), 2nd ACM Conf. on Computer and Communications Security, ACM Press, COAST, Purdue, November 1994, pp. 18–29.

[35] S. Kumar, E. Spafford, A pattern matching model for misuse intrusion detection, Proc. 17th National Computer Security Conf. October 1994, pp. 11–21.

[36] C.E. Landwehr, A.R. Bull, J.P. McDermott, W.S. Choi, A taxonomy of computer program security flaws, ACM Computing Surveys 26 (3) (September 1994) 211–254.

[37] G. Liepins, H.S. Vaccaro, Anomaly detection: purpose and framework, Proc. 12th National Computer Security Conf., October 1989, pp. 495–504.

[38] T. Lunt, R. Jagannathan, A prototype real-time intrusion-detection expert system, Proc. Symp. on Security and Privacy, Oakland, CA, April 1988, pp. 59–66.

[39] T.F. Lunt, Automated audit trail analysis and intrusion detection: a survey, Proc. 11th National Computer Security Conf., Baltimore, MD, October 1988.

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

[41] T.F. Lunt, R. Jagannathan, R. Lee, S. Listgarten, D.L. Edwards, P.G. Neumann, H.S. Javitz, A. 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.

[42] N. McAuliffe, D. Wolcott, L. Schaefer, N. Kelem, B. Hubbard, T. Haley, Is your computer being misused? a survey of current intrusion detection system technology, Proc. 6th Annual Computer Security Applications Conf., Tucson, AZ, IEEE Computer Society Press, Los Alamitos, CA, December 1990, pp. 260–272.

[43] A. Mounji, languages and tools for rule-based distributed intrusion detection, Doctor of science, Facultés Universitaires Notre Dame de la Paix, Namur, Belgium, September 1997.

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

[45] Network Associates Inc., Cybercop server, available from the company's website at http://www.nai.com/products/security/cybercopsvr/index.asp, 1998.

[46] V. Paxson, Bro: a system for detecting network intruders in real-time, Proc. 7th USENIX Security Symp., San Antonio, TX, January 1998.

[47] P. Porras, R. Kemmerer, Penetration state transition analysis – a rule-based intrusion detection approach, Proc. 8th Annual Computer Security Applications Conf., November 1992, pp. 220–229.

[48] P.A. Porras, A. Valdes, Live traffic analysis of tcp/ip gateways, Proc. ISOC Symp. on Network and Distributed System Security (NDSS'98), San Diego, CA, March 1998 (Internet Society).

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

[50] T.H. Ptacek, T.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.

[51] M.J. Ranum, K. Landfield, M. Stolarchuk, M. Sienkiewicz, A. Lambeth, E. Wall, Implementing a generalized tool for network monitoring, Proc. 11th Systems Administration Conf. (LISA'97), San Diego, CA, October 1997.

[52] P. Rolin, L. Toutain, S. Gombault, Network security probe, CCS'94, Proc. 2nd ACM Conf. on Computer and Communication Security, November 1994, pp. 229–240.

[53] D.R. Safford, D.L. Schales, D.K. Hess, The tamu security package: an ongoing response to internet intruders in an academic environment, Proc. 4th USENIX Security Symp, Santa Clara, CA, October 1993, pp. 91–118.

[54] W.S. Sarle, Neural networks and statistical models, Proc.

19th Annual SAS Users Group Int. Conf., Cary, NC, April 1994, pp. 1538–1550.

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

[56] S. Smaha, Haystack: an intrusion detection system, 4th Aerospace Computer Security Applications Conf., October 1988, pp. 37–44.

[57] S.R. Snapp, J. Brentano, G.V. Dias, T.L. Goan, L.T. Heberlein, C.l. Ho, K.N. Levitt, B. Mukherjee, S.E. Smaha, T. Grance, D.M. Teal, D. Mansur, DIDS (distributed intrusion detection system) – motivation, architecture, and an early prototype, Proc. 14th National Computer Security Conf., Washington, DC, October 1991, pp. 167–176.

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

[59] P. Spirakis, S. Katsikas, D. Gritzalis, F. Allegre, J. Darzentas, C. Gigante, D. Karagiannis, P. Kess, H. Putkonen, T. Spyrou, SECURENET: a network-oriented intelligent intrusion prevention and detection system, Network Security Journal 1 (1) (1994).

[60] T. Spyrou, J. Darzentas, Intention modelling: approximating computer user intentions for detection and prediction of intrusions, in: S.K. Katsikas, D. Gritzalis (Eds.), Information Systems Security, Samos, Greece, May 1996, pp. 319–335.

[61] S. Staniford-Chen, S. Cheung, R. Crawford, M. Dilger, J. Frank, J. Hoagland, K. Levitt, C. Wee, R. Yip, D. Zerkle, GrIDS – a graph based intrusion detection system for large networks, Proc. 19th National Information Systems Security Conf., 1996.

[62] S. Staniford-Chen, B. Tung, P. Porras, C. Kahn, D. Schnackenberg, R. Feiertag, M. Stillman, The Common Intrusion Detection Framework-data Formats, Internet draft draft-ietf-cidf-data-formats-00.txt, March 1998.

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

[64] H.S. Vaccaro, G.E. Liepins, Detection of anomalous computer session activity, Proc. IEEE Symp. on Research in Security and Privacy, 1989, pp. 280–289.

[65] D. Vincenzetti, M. Cotrozzi, Atp – anti tampering program, Proc. 4th USENIX Security Symp., Santa Clara, CA, October 1993, pp. 79-9.

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

Hervé Debar is a research scientist in the global security analysis laboratory at the IBM Zurich Research Laboratory, where he works on system and network security (in particular intrusion detection) as well as system management. His interests include secure systems and artificial intelligence. Dr. Debar holds a Ph.D. from the University of Paris and is a telecommunications engineer from the Institut National des Télécommunications in Evry (France).

Marc Dacier is currently working at the IBM Zurich Research Laboratory in the Information Technology Solutions Department. He prepared his Ph.D. at LAAS-CNRS and then worked at Firstel as a security consultant. His research interests focus on penetration testing of computing systems and on security policies.

Andreas Wespi holds a M.Sc. in Computer Science from the University of Berne, Switzerland. He is currently working at the IBM Zurich Research Laboratory in the Information Technology Solutions Department. His research interests include network security as well as distributed and parallel computing.