MASTER IN COMPUTER SCIENCE

Towards "Intelligent" Security Audit Trail Analysis Tools

Libion, Fabian

Award date:
1991

Awarding institution:
University of Namur

Link to publication

Facultés
Universitaires
N. D. de la Paix
Namur

Institut d'Informatique

Année académique 1990-1991

Towards "Intelligent" Security Audit Trail Analysis Tools

Fabian LIBION

Promoteur :

Professeur J. Ramaekers

Mémoire présente en vue
de l'obtention du titre
Licencié et Maître
en Informatique

ABSTRACT.

Computer system security violations and misuses are inherent part of the increasing evolution and utilization of the information technology. Trusted computer evaluation criteria define manners to counter and limit these threats to the good functioning of a computer system environment, threats that can sometimes have considerable consequences. One possible counter-measure, proposed in this document, is to log the activities of the users on a system and produce **security audit trails** that permit after-the-fact analysis of these activities to detect possible security breaches going from manipulation errors to voluntary attacks.

These security audit trails may be analysed by two types of evaluators : **reduction tools** that are passive means of analysis and **automatic analysis tools** that use artificial intelligence techniques to make the evaluation more active or "intelligent", and in some case to react in real-time against the discovered attack.

Des mauvaises utilisations et des violations de sécurité de systèmes d'ordinateurs sont des conséquences inhérentes à la croissance de l'évolution et de l'utilisation de l'informatique. Des critères d'évaluation de "systèmes informatiques sûrs" définissent des moyens afin de contrer et de limiter ces menaces au bon fonctionnement des systèmes, menaces qui parfois ont des conséquences considérables. Une contre-mesure envisageable proposée dans ce document est d'enregistrer l'activité des utilisateurs d'un système et ainsi de produire des "audit trails" concernant la sécurité qui permettront une analyse postérieure de cette activité afin de découvrir d'éventuelles violations allant des erreurs de manipulation aux attaques volontaires.

Ces audit trails concernant la sécurité peuvent, selon les cas, être analysés par deux types d'évaluateurs : les **outils réducteurs** qui analysent de manière passive et les **outils d'analyses automatiques** qui utilisent des techniques d'intelligence artificielle pour faire une évaluation plus active ou "intelligente", et dans certains cas pour réagir en temps réel à l'attaque en question.

ACKNOWLEDGMENTS.

First of all, I wish to sincerely thank the people who have played an important part before and during the realization of this thesis :

- Armand Falaise and Isabelle Mathieu, from SIEMENS-NIXDORF INFORMATIONSSYSTEM Namur, Winfried Weiss and Christian Wolf, from Siemens AG Munich, and Baudouin Lecharlier, professor at the FUNDP Namur, for their helpful guidance during my study about security audit trails;

- Jean Ramaekers, professor at the FUNDP of Namur, promoter of this thesis, for his beneficial comments on the first draft of this document and for the complete freedom of action he gave me;

- all the readers of this document (drafts and last version) and for all their comments and critics they gave me;

- I want also to thank all the people who by their contacts gave me (sometimes involuntarily) help and assistance.

TABLE OF CONTENTS.

LIST OF FIGURES.

LIST OF TABLES.

INTRODUCTION.

This document has the objective to explain to the reader that an "intelligent" help can be provided to the security officers who have the task to evaluate very large amounts of data contained in **security audit trail.** The term "intelligent" has the same meaning, the same **advantages** and **restrictions** than those of the *artificial intelligence* field, artificial intelligence that will be used to develop techniques of audit trail evaluation and analysis.

To achieve this purpose, chapter 1 first presents security breaches that may occur in any computer system, and insists on the particular security violations that are computer **break-ins**. Chapter 1 then proposes and presents some trusted computer evaluation criteria books that try to define and provide **counter-measures** to limit these security threats.

Chapter 2 describes theoretically and through several examples one particular manner to counter security breaches and attacks, that is the generation of **security audit trail.**

These audit trails have to be analysed or evaluated. Chapter 3 exposes a passive way of analysis and the kind of malicious activities that can be detected by reduction tools. Examples of these **reductions tools** are presented.

Trying to resolve the disadvantages of these passive tools, chapter 4 develops "intelligent" methods - used by **automatic analysis tools** - such as behaviour pattern recognition, statistical components and methods learning system behaviours, that make feasible active evaluations of security audit trails. Some already-existing tools and prototypes are detailed. In parallel, methods using artificial intelligence techniques (knowledge base, expert systems, learning methods) are developed.

Chapter 5 is the concrete part of this document, commenting the elaboration of a "learning normal system behaviour" component realized during the period of training in SIEMENS-NIXDORF INFORMATIONSSYSTEM (in Namur) and in SIEMENS AG (in Munich). This prototype analyses security audit trails based on the **expected behaviours** of the users.

In chapter 6, a look at the future evolutions of these automatic analysis tools is envisaged through a series of criteria, and particularly through the evolution of the artificial intelligence field. One example of this evolution is illustrated by the use of neural networks to evaluate security audit trails.

In several chapter of this thesis some audit mechanisms and tools are described and compared. They are considered at the end of the year 1990, and some evolutions may be brought to them since this period.

In the following text, all underlined terms make reference to the glossary. They are not the only ones present in the glossary but are supposed to be relative to the present particular subject of study and could therefore have an unknown, imprecise or even different meaning for the reader.

CHAPTER 1

SECURITY THREATS AND COUNTERMEASURES.

As computer science rapidly evolves, its utilization becomes more and more frequent and common. The computing and storage powers of Information Technologies (IT) are unceasingly a challenge. The information is computed more and more quickly and stored on smaller and smaller data supports. The information is also more and more (logically) centralized. Another facet of IT is the great dependence on the information that computer users (individuals, organizations, societies, ...) want to "automatically" process.

The increasing progresses of the technology and the dependence on the information must put in everyone's mind that security in IT is certainly not to be neglected. Natural catastrophes, accidental circumstances, breakdowns, errors of manipulation, misusing, thefts, ... are threats that may destroy computer systems, the information processed and all the organization which depends directly or not on them, if this one has neglected some elementary security measures.

A threat is a function of the operational environment of the system and the sensitivity of the data being processed in the system. The basic threats to which a computer system is exposed are :

- loss of **confidentiality** : means that some unauthorized accesses to information may occur and that information is not only disclosed to those users who are authorised to have access to it;

- loss of **integrity** : means that some unauthorized modifications of information may be perpetrated, and that information is not only modified by those users who have right to do so;

- loss of **availability** : means that some unauthorized weakening of functionality are possible, and that information and resources can not sometimes be accessed by authorized users when needed.

A lot of computer security measures are elaborated to counter the great number of various threats. Computer security is a very large field that consists of three groups of safeguards :

1) the **physical security** to protect and control the access to the computer environment including :

   - geographical situation free from sources of natural disasters (flooding, fire, ...),
   - building architecture,
   - air-conditioning equipment,
   - data communication,
   - ...

2) the **operational and procedural security** that is concerned with putting constraints on employees in data processing organizations :

   - separation of responsibility of people in sensitive jobs,
   - limitation of the access to only concerned, trusted personal members,
   - restricted access to data supports and devices,
   - remote storage of back-up supplies and copies of important data files and programs,
   - ...

3) the **internal computer security** that uses control and protection mechanisms within the hardware, software, and data communication circuits of the computer system :

- identification and authentication of the users,
- access control mechanisms to protect files, programs, ...
- cryptography capabilities,
- monitoring and auditing,
- restart and recovery capabilities,
- ...

These safeguards are needed for the **deterrence, prevention, detection** and **recovery**.

There are so many parameters concerning computer security that a great problem is the impossibility to reach a fully secure environment.

In this document, our attention will be focused on particular threats that are security violations. The following section (1.1) will introduce some different kinds of breaches to familiarize the reader with the **various measures** that exist to counter malicious actions. Security evaluation criteria are presented as countermeasures in section 1.2 of this chapter.

1.1. Security Break-ins.

As the information - sometimes of high interest and importance - is stored in smaller centralized areas, it could be attractive for malicious people to try to get or destroy information, just only by game or for malicious uses. Another type of misuses may be accidental. A system flaw may give unexpected access to a well meaning user. Typing errors could produce unintentional malicious result. A new user may even be unaware of certain aspects of the system policy. But we make no attempt to divine intent or malevolence.

Without going into details, several kinds of attacks can be distinguished and various malicious people may be involved in security breaches as described in the two next sections.

1.1.1. Kinds of attacks.

Four kinds of misuses to the security of a computing system can be distinguished : **interruption**, **interception**, **modification**, and **fabrication** as shown in figure 1.1.

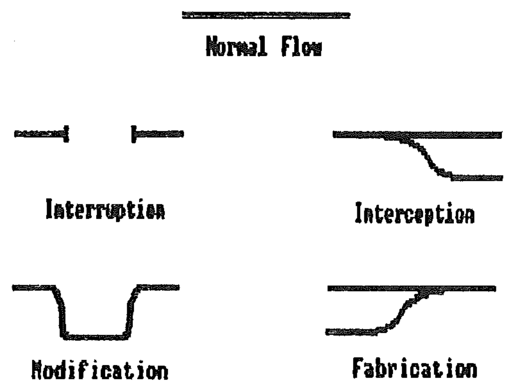

Figure 1.1 *Four classes of System Security Failures.*

a) In an *interruption*, an asset of the system becomes lost or unavailable or unusable. Examples are malicious destruction of a hardware device, erasure of a program, leakage of classified data, destruction of accounts, or failure of an operating system file manager so that it cannot find a particular disk file, ...

b) An *interception* means that some unauthorized party has access to an asset. The outside party can be a person, a program, or a computing system. While a loss may be discovered fairly quickly, a silent interceptor may leave no traces by which the interception can be readily detected. Interceptions may be perpetrated by "tourists" who trespass the system and break as hobby, or by people looking at the system, browsing passively or actively (scavenging with goal in mind), collecting information by aggregation (accumulation of information) or by inference (distillation of information). Examples of this type of failure are illicit copying of general softwares or specific sensitive data files, or wiretapping to obtain data in a network.

c) If an unauthorized party not only accesses but tampers with an asset, the failure becomes a *modification*. Modification may be performed on data (data diddling or false data entry), on programs (time bombs[1], logic bombs[2],Trojan horses[3], viruses[4]), or on system behaviours (access rights, password files, accounting, ownership). It is even possible for hardware to be modified. For examples, someone might modify the values in a data base, alter a program so that it performs an additional computation, or modify data being transmitted.

d) Finally, an unauthorized party might *fabricate* counterfeit objects for a computing system. The intruder may wish to add spurious transactions to a network communication system, or add records to an existing data base.

1.1.2. The involved people.

People may pose a threat because of accidental acts and deliberate acts. "The possibility of inadvertent mistakes by people is fully recognised by designers but the dangers from malicious and deliberate acts receive less attention, except by designers of highly sensitive systems" [Lane 85]. "Deliberate threats can be subdivided into three areas :

a) threats that do not require the subverter to be a user of the system, such as physical threats and data communication line threats;

[1] A time bomb is a program activated by the computer's clock to initiate a fraud, disruption or other "perverse activity".
[2] A logic bomb is similar to a time bomb but is stirred into action by a combination of events rather than time.
[3] Troyan Horses are illicit parts of innocent programs with effects similar to bombs but not activated by a specified set of circumstance.
[4] Viruses reproduce themselves within a disc and move from one system to another, but have a variety of ultimate effects.

b) threats that may allow the subverter to become an illegal user, in particular, threats to obtain user codes and passwords;

c) and threats that require the infiltrator to be a user, whether he is legitimate or not." [Walker 77]

In the following of this document, we will not pay attention to physical threats that are physical manners to destroy or get information (such as to steal magnetic tapes, ...), and/or to demolish or rob some components of hardware. We will rather concentrate our attention to other methods of attacks perpetrated on the data and software of a system. These attacks can be perpetrated by exploiting computer **vulnerabilities**, that are weaknesses in the system, to cause loss or harm. The two major kinds of security violations in computer systems that are interesting for automatic intrusion detection, are the **external** and **internal** intrusions of the system security.

External intrusions are those perpetrated by people who are not authorized to use the computer system and who try to by-pass the control mechanisms to have access to the system and to become, once inside this system, illegal users acting on a legal user account.

Internal intrusions occur when users are authorized to use the computer system, but not the data, programs, or resources, and try to get more privilege than they already have. They can act as **clandestine** users evading supervising mechanisms (auditing, monitoring, ...) and access controls; or as **masqueraders** operating under other userid and passwords, or acting in such a way that it is not possible to know "who is doing what". Another kind of internal breaches can be perpetrated by **misfeasors** who are also authorized to use the computer system and resources but misuse their privileges.

1.2. Countermeasures.

There are many possible countermeasures imagined to fight computer abuses and misuses. Some of these methods are listed below :

- terminal access controls, passwords, terminal locks and identification and authorisation procedures;
- encryption that transforms or codes data so that it is unintelligible to the outside observer, and the value of an interception and the possibility of a modification or a fabrication are almost nullified;
- threat monitoring that collects and analyses information on security system operations in real-time providing protection responses such as job cancellation, advising the operator, etc;
- security auditing[5] meaning the logging of information about requests of protected resources, and the subsequent post facto analysis in order to detect security violations, ...
- many other security methods are used such as back-up copies, physical access controls, redundancy, ...

1.2.1. Presentation of trusted system criteria.

As the security in IT is considered with more and more importance, several groups of work have been created (Department of Defence, ...), trying to establish rules, standards, norms and criteria concerning security. As guide-lines for security in computer systems, several documents describe trusted evaluation criteria that classify computer systems into hierarchical divisions of increased security protections. This section principally

[5] Security auditing will be developed in details in chapter 2 and further.

presents the "Department of Defense (DoD) Trusted Computer System Evaluation Criteria"[6] [DoD 83] that will be taken as reference along this document. But other books also exist such as :

- "IT-Security Criteria" [ITSC 89] in Germany;
- "Information Technology Security Evaluation Criteria" [ITSEC 90] that is the harmonised criteria of France, Germany, the Netherlands and the United Kingdom;
- "NATO Trusted Computer System Evaluation Criteria" [NTCSEC 85] that is the equivalent criteria of the Department of Defense ones for the NATO.

These books are presented in the next section. They all make references to the Orange Book first published in 1983 that is now presented. Some criteria concerning security audit trail and some other concepts are also described for a good understanding of the following chapters.

The Orange Book.

"The criteria defined in the [DoD 83] constitute a uniform set of basic requirements and evaluation classes for assessing the effectiveness of security controls built into Automatic Data Processing (ADP) systems". "The criteria were developed with three objectives in mind : (a) to provide users with a yardstick with which to assess the degree of trust that can be placed in computer systems for the secure processing of classified or other sensitive information; (b) to provide guidance to manufacturers as to what to build into their new, widely-available trusted commercial products in order to satisfy trust requirements for sensitive applications; and (c) to provide a basis for specifying security requirements in acquisition specifications." [DoD 83]

[6] Also known as the *Orange Book* because of the orange colour of its cover.

The DoD defines in the Orange Book six fundamental requirements derived from the "basic statement of objective : four deal with what needs to be provided to control access to information; and two deal with how one can obtain credible assurances that this is accomplished in a trusted computer system." [DoD 83]

Policy

Requirement 1 - *Security Policy* - There must be an explicit and well-defined <u>security policy</u> enforced by the system.

Requirement 2 - *Marking* - Every object must be associated with a "label" that indicates the security level of the object. The association, which is also known as "marking" the object, must be done so that the label is available for comparison each time an access to the object is requested.

Accountability

Requirement 3 - *Identification* - Every subject must be uniquely and convincingly identified. Identification is necessary so that subject/object access request can be checked.

Requirement 4 - *Accountability* - The system must maintain complete, secure records of actions that affect security. Such actions include introduction of new users to the system, assignment or change of the security level of a subject or an object, and denied access attempts.

Assurance

Requirement 5 - *Assurance* - The computing system must contain mechanisms that enforce security, and it must be possible to evaluate the effectiveness of these mechanisms.

Requirement 6 - *Continuous protection* - The mechanisms that implement security must be protected against unauthorized change.

All these requirements have to be respected by a system with more or less severity depending on the class it wishes to belong. The classes are based on divisions subdivided themselves in categories. There are four basic divisions A, B, C and D, where A is the division with the most comprehensive degree of security. Within divisions there are additional distinctions, denoted with numbers, where the higher numbers indicate tighter security requirements. The complete set of classes is D, C1, C2, B1, B2, B3 and A1. The descriptions of these levels are listed below. Within these descriptions, text in quote marks is directly issued of the Orange Book [DoD 83]. Table 1.1 summarises the requirements of the different levels by indicating for each criteria (in row) whether there are no requirement (in column) or whether there are (or not) additional new requirements.

Class D : Minimal Protection.

Class D is applied to systems that have been evaluated to verify their property to higher levels but have failed. No security characteristics are needed for this class.

Class C1 : Discretionary Security Protection.

A Trusted Computing Base (TCB) satisfying the class C1, provides an environment where users and data are separated. Users must be allowed to discretionarily protect their own data in order to limit access to other users or group of users (discretionary access control). A data owner may also decide whether and when the controls apply or not. This class is intended for an environment where cooperating users process data at the same level of sensitivity.

Class C2 : Controlled Access Protection.

A class C2 system also provides discretionary access protection with a finer granularity of control. The actions of the users must be individually recognisable so that it is possible to know who is doing what. This can be realized through login procedures, audit trail capabilities which must be capable of tracking each individual's access (or attempted access) to each object, and resource isolation.

Class B1 : Labelled Security Protection.

In addition to the level C2 requirements, a class B1 system includes <u>mandatory (nondiscretionary) access control</u>. A <u>security level</u> must be assigned to each subject, while only controlled objects must be individually labelled by security level (the protection mechanisms do not need to control all objects at a B1 level). The access control decisions must be based on these labels to model hierarchical levels and non-hierarchical categories. Moreover an informal model of the <u>security policy</u> must be available.

Class B2 : Structured Protection.

For a B2 level TCB, access control policies (discretionary and mandatory) must be enforced to all subjects and objects of the system, including devices. "The TCB must be structured into protection-critical and non-protection-critical elements." A <u>formal security policy model</u> must be present, and analysis of <u>covert channels</u> is required. "The system is relatively resistant to penetration".

Class B3 : Security Domains.

In class B3 TCB, subject/object <u>domains</u> are required, with a capability to implement access protection for each object, indicating allowed subjects, kind of access allowed for each, and disallowed subjects. A full <u>reference monitor concept</u> must be provided, so that every access is checked. Requirements are also made for the system audit facility that must be able to identify when a violation of security is imminent. "The system is highly resistant to penetration".

Class A1 : Verified Design.

The requirements of a class A1 TCB are equivalent as the one at a B3 level. "The distinguishing feature of systems in this class is the analysis derived from formal design specification and verification techniques and the resulting high degree of assurance that the TCB is correctly implemented". The more important criteria of the class A1 are : (1) a formal model of the protection system and a proof of its consistency and adequacy, (2) formal top-level specification of the protection system, (3) a demonstration that the top-level specification corresponds to the model, (4) an implementation informally shown to be consistent with the specification, and (5) formal analysis of covert channels.

Table 1.1 *Trusted Computer Evaluation Criteria (from the Orange Book [DoD 83]).*

	Requirement						
Criteria	D	C1	C2	B1	B2	B3	A1
Security Policy							
Discretionary Access Control	■	⊖	⊖	»	»	⊖	»
Object Reuse	■	■	⊖	»	»	»	»
Labels	■	■	■	⊖	⊖	»	»
Label Integrity	■	■	■	⊖	»	»	»
Exportation of Labelled Information	■	■	■	⊖	»	»	»
Labelling Human-Readable Output	■	■	■	⊖	»	»	»
Mandatory Access Control	■	■	■	⊖	⊖	»	»
Subject Sensitivity Labels	■	■	■	■	⊖	»	»
Device Labels	■	■	■	■	⊖	»	»
Accountability							
Identification and Authentication	■	⊖	⊖	⊖	»	»	»
Audit	■	■	⊖	⊖	⊖	⊖	»
Trusted Path	■	■	■	■	⊖	⊖	»
Assurance							
System Architecture	■	⊖	⊖	⊖	⊖	⊖	»
System Integrity	■	⊖	»	»	»	»	»
Security Testing	■	⊖	⊖	⊖	⊖	⊖	⊖
Design Specification and Verification	■	■	■	⊖	⊖	⊖	⊖
Covert Channel Analysis	■	■	■	■	⊖	⊖	⊖
Trusted Facility Management	■	■	■	■	⊖	⊖	»
Configuration Management	■	■	■	■	⊖	»	⊖
Trusted Recovery	■	■	■	■	■	⊖	»
Trusted Distribution	■	■	■	■	■	■	⊖
Documentation							
Security Features User's Guide	■	⊖	»	»	»	»	»
Trusted Facility Manual	■	⊖	⊖	⊖	⊖	⊖	»
Test Documentation	■	⊖	»	»	⊖	»	⊖
Design Documentation	■	⊖	»	⊖	⊖	⊖	⊖

Legend : ■ : no requirement;
⊖ : additional requirement;
» : same requirement as previous class

1.2.2. What are the principal differences with the other security evaluation criteria books?

The criteria proposed by the Orange Book are widely known and accepted as a basis for the security evaluation of computer systems. Some other books are now briefly presented to show other manners of evaluating the trustworthiness of IT systems. These two books are the "IT-Security Criteria" [ITSC 89] for Germany, and the "Information Technology Security Evaluation Criteria" [ITSEC 90] that is the harmonised criteria of France, Germany, the Netherlands and the United Kingdom. The [NTCSEC 85] is equivalent to the Orange Book but some details and then is not described here.

The "IT-Security Criteria".

The classes of functionality listed in the "IT-Security Criteria" [ITSC 89] are merely intended as guide-lines, to provide the users with assistance in the selection of a system, and the manufacturers with help in the design and categorisation of their systems. One system may conform with the requirements of several classes at the same time. All the classes that a system is conform with are listed in a certificate. These classes of functionality are :

Functionality class F1, F2, F3, F4 and F5 that are respectively derived from the functionality of the Orange Book Class C1, C2, B1, B2, and B3/A1.

"Functionality class F6 is for systems with high integrity requirements for data and programs. Such requirements are significant, e.g. for database systems.

Functionality class F7 sets high requirements for the availability of a complete system or special functions of a system. Such requirements are significant for process control systems for example.

Functionality class F8 sets high requirements with regard to the safeguarding of data integrity during data communication.

Functionality class F9 is intended for systems with high demands on the confidentiality of data during data communication. An example candidate for this class is a cryptographic device.

Functionality class F10 is intended for networks with high demands on the confidentiality and integrity of the information to be communicated. For example, this can be the case when sensitive information has to be communicated via insecure (for example : public) networks." [ITSC 89]

In addition of these functionality classes, assurance criteria and assurance levels are defined to be able to assess the assurance of the security functions of a system or a single component. As examples, some assurance criteria are : quality of the security policy, quality of the software development process, quality of the user related documentation, ...

Eight assurance levels are defined for the rating of the assurance. These assurance levels are :

Q0 : inadequate assurance,
Q1 : tested,
Q2 : methodically tested,
Q3 : methodically tested and partially analyzed,
Q4 : informally analyzed,
Q5 : semi-formally analyzed,
Q6 : formally analyzed,
Q7 : formally verified.

Higher assurance levels than Q7 are conceivable, but cannot be achieved with today's technology. Such higher levels should be - if the coming technology permits - the use of formally verified tools (compilers, linkers, etc.) and a formally verified hardware and firmware design.

The "Information Technology Security Evaluation Criteria".

The criteria of the "Information Technology Security Evaluation Criteria" [ITSEC 90] permit specification of arbitrary security **functionality**, and define seven evaluation levels representing increasing confidence in the **correctness** of a Target of Evaluation (TOE). "Once the TOE has been assessed for correctness in accordance with the correctness criteria for the target evaluation level, its **effectiveness** in meeting its security target is considered, at a level of rigour appropriate for that evaluation level. Thus these criteria can be applied to cover a wider range of possible systems and products than the [DoD 83]." [ITSEC 90]

a) The *functionality* of a TOE consists of its security functions taken as a whole. Ten functionality classes (F1 to F10) have been defined as part of the [ITSEC 90]. These functionality are identical to those of the [ITSC 89] that are predefined classes. But where the security functions to be specified are similar to the predefined classes, the statement of functions may be derived from those classes. Where the statement will wholly include one or more predefined classes, these may be referenced. Moreover as technical innovation in the field of IT security is rapidly evolving, TOEs can and will offer increasingly sophisticated functions in the future. It is then envisaged that new predefined classes defined as new groups of functions become sufficiently common to make such classes worthwhile.

b) Seven evaluation levels (E0 to E6) are defined for the evaluation of the correctness of a TOE. The correctness is the certification that the TOE accurately reflects the stated security target for a system or product. These levels are cumulative, it means that what is required at a lower level is also required at a higher level, this higher level including supplementary special features.

Level E0 represents inadequate assurance.

At level E1 a security target and an informal description of the security architecture of the TOE shall be provided. The TOE satisfies its security target.

Level E2 requires an informal description of the detailed design.

For level E3 the detail design and the source code corresponding to the security functions shall be provided.

A formal model of the security policy is required for level E4. The architectural and detailed design shall use a rigourous approach and notation. Vulnerability analysis is based upon this rigourous approach.

A close correspondence between the detailed design and the source code is asked at level E5. A vulnerability analysis shall be performed using the source code.

Level E6 requires a formal description of the security architecture, consistent with the formal model of the security policy.

c) Evaluation of effectiveness determines whether the confidence level established by evaluation of correctness remains valid for correctness and effectiveness in combination, having regard for the proposed use of the TOE in the context of its intended environment. Assessment of effectiveness is only performed after confidence in correctness has been established.

It is possible to produce a table (see table 1.2) showing the intended correspondence between the criteria of the [ITSEC 90] and those ones of the [DoD 83].

Table 1.2 *Correspondence between the criteria of the [ITSEC 90] and those ones of the [DoD 83] (from [ITSEC 90]).*

[ITSEC 90] Criteria		[DoD 83] Class
E0	--->	D
F1, E2	--->	C1
F2, E2	--->	C2
F3, E3	--->	B1
F4, E4	--->	B2
F5, E5	--->	B3
F5, E6	--->	A1

A product successfully evaluated against the criteria in the table 1.2 for a predefined functionality class F1 to F5 at an evaluation level not lower than given in the table 1.2 should fulfil the requirements of the equivalent [DoD 83] class shown in the table. The converse relationship, however, cannot be directly assumed, due to the wider confidence requirements found in these criteria.

CHAPTER 2

SECURITY AND AUDIT TRAIL.

There exist numerous of preventions and countermeasures possibilities to counter security violations. This chapter focuses the attention on a method that principally tries to prevent and detect security breaches. This method is the security **auditing**.

Trusted computers that claim to respect the C2 to A1 level requirements of the DoD Trusted Computer System Evaluation Criteria [DoD 83] have to provide an audit log mechanism.

2.1. What is "SECURITY AUDITING" ?

Security Auditing is a countermeasure directed towards prevention or detection of subversive or accidental programming or operational practice. A security audit trail is a collection of information that permits to retrace the activity of a computer system for security purpose. This information is collected by a mechanism which logs **security-relevant events** so that they can be monitored at a later time. Audit trails are logged into one or more files composed of audit records, each record representing an event.

2.1.1. Auditable events.

An *event* represents a subject acting on an object with a certain result. The *subject* is an active entity wanting to receive or act on information. The *object* is "a passive entity which contains or receives information" [DoD 83]. The action made by the subject on the object is characterized by a result indicating the success or the failure of the action. This is illustrated by the figure 2.1.

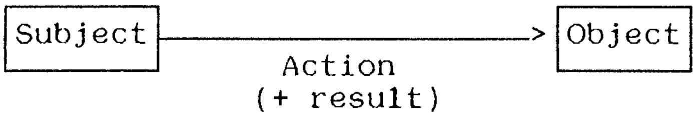

Figure 2.1 *An event : a subject acting on an object with a certain result.*

The auditable events required by the C2 Criteria class of the DoD are :

- "Use of identification and authentication mechanisms
- Introduction of objects into a user's address space
- Deletion of objects from a user's address space
- Actions taken by computer operators and system administrators and/or system security officers
- All <u>security-relevant events</u>." [NCSC 87/1]

In addition to the C2 audit requirements, the B1 Criteria class requires as auditable events :

- "Any override of human-readable output markings[1] (including overwrite of <u>sensitivity label</u> markings and the turning off of labelling capabilities) on paged, hard-copy output devices

- Change of designation (<u>single-level</u> to/from <u>multi-level</u>) of any communication <u>channel</u> or I/O device

- Change of <u>sensitivity level(s)</u> associated with a single-level communication channel or I/O device

- Change of range designation of any multi-level communication channel or I/O device." [NCSC 87/1]

In addition to the B1 audit requirements, the B2 Criteria class requires as auditable events :

- "Events that may exercise <u>covert storage channels</u>." [NCSC 87/1]

In addition to the B2 audit requirements, the B3 Criteria class requires as auditable events :

- "Events that may indicate an imminent violation of the system's security policy (e.g., exercise <u>covert timing channels</u>)." [NCSC 87/1]

No new auditable event is required at the above classes (B3 and A1).

---

1 From the level B1 of the [DoD 83], the TCB shall mark the begining and the end of all human-readable, paged, hardcopy output with human-readable sensitivity labels that properly represent the sensitivity of the output. The TCB shall, by default, mark the top and bottom of each page of human readable, paged, hardcopy output with human-readable sensitivity labels that properly represent the overall sensitivity of the output or that properly represent the sensitivity of the information on the page. The TCB shall, by default and in an appropriate manner, mark other forms of human-readable sensitivity labels that properly represent the sensitivity of the output. The override of these markings defaults shall be auditable by the TCB.

2.1.2. Content of an event.

The composition of an event depends from one audit generator to another but the DoD requires that the audit events contain at minimum the following information :

- At the C2 class : "

  . Date and time of the event
  . The unique identifier on whose behalf the subject generating the event was operating
  . Type of event
  . Success or failure of the event
  . Origin of the request (e.g., terminal ID) for identification / authentication events
  . Name of the object introduced, accessed, or deleted from a user's address space
  . Description of modifications made by the system administrator to the user / system security databases." [NCSC 87/1]

- At the B1 class, the security level of the object is to be added at the list of the C2 class. It is also recommended, but not required, to supply the <u>subject security level</u>.

- No new information is required at the above classes (B2, B3 and A1).

2.2. Purpose of the Audit Trail.

The collection of events (audit trail) is to be analysed by a security officer - more or less assisted by sophisticated tools - to **detect** possible anomalies that could occur in the system. The anomalies could have the form of :

- attempt of unauthorized users to by-pass the protection mechanisms,
- illegal destruction of any information,
- illegal access to sensitive data,
- illegal privileges acquired by legal users,
- wrong use of their privileges by users, ...

The audit trail must also act as a **deterrent** against perpetrators. For that goal, it is important that the system users understand that the audit mechanism exists and what impact it has on them. Without this user's understanding, user deterrence and user assurance goals of the audit mechanism cannot be achieved.

2.3. How auditing ?

Audit trails are generated by any mechanism that must be able to record the occurrence of security-relevant events.

2.3.1. The selection of the events.

Recording all security-relevant events into the audit files may produce an unmanageable amount of data involving problems of disk space and difficulties to find anomalies. The events must be selectively recorded to reduce the amount of data and allow efficient analysis. Some events, such as some login failures and use of privilege are always auditable. Other events, such as successful or unsuccessful attempts to gain access to sensitive files, can be selected for auditing. The TCSEC do not require the recording of any audit data, only the ability to record such data. The logging mechanism has to pre-select events that are recorded into the audit trail while the other events are discarded. This pre-selection avoid the audit files to be flooded with events that have no security signification or interest. The resulting files are called **collection files**. Moreover the security officer can apply post-selections to the collection files making easier the evaluation of the audit trail by focusing attention to some kind of events, users, objects, actions or result, ... The resulting files are called **reduction files**.

The selection can be based on the following criteria, singly or in combination :

- "selection based on type of auditable event,
- selection based on user identity,
- selection based on object identity,
- selection based on object properties (usually security level)" [NCSC 87/1].

2.3.2. The logging mechanisms.

The enabling of all events could result in a significant system degradation in processor speed if the audit mechanism is not efficiently implemented. Just the act of determining whether an event mediated by the TCB is subject to audit and whether the event has been marked for logging, could take significant processor resources.

The most used implementation solution for the logging of audit records is to initiate a special **system call** immediately after any of the events in the list of auditable actions. This type of events are called *micro-events*.

Some audit mechanisms add to this solution the possibility for **trusted application processes** (e.g., login) to write directly to the audit device. This enables login, for instance, to write a login "audit record" to the audit trail rather than letting a login on the system be represented as a collection of system calls (producing a set of micro-events) required to complete the login procedure. This solution has the advantage of reducing the amount of audit data written to the audit trail and to make the trail more meaningful. Such events are called *macro-events*.

Sometimes, a subsystem encounters inconsistencies or problems that make the writing of an informative audit record desirable. That is why some audit mechanisms make feasible for **privileged subsystems** to directly insert audit information into the trail.

2.3.3. Compaction of the data.

Another way to resolve the enormous volume of the data is to compact the information. The compaction of the audit data has two advantages. First the information is encoded to reduce the disk space required by collection and reduction files, and second the coding restricts read accesses to the audit information. But this protection is not sufficient.

2.3.4. Audit trail protection.

At the C2 class, "the TCB shall be able to ... protect from modification or unauthorized access or destruction an audit trail of accesses to the object it protects. The audit trail data shall be protected by the TCB so that read access to it is limited to those who are authorized for audit data." [NTCSEC 85]

Audit trail software (generator, reduction tools, ...), as well as the audit trail itself, should be protected and should be subject to strict access controls to be protected against modifications and unauthorized deletions or readings to permit detection and after-the-fact investigations of security violations. The log containing the audit trail data will be writable only by the TCB and by processes with the appropriate privileges, and be readable only by security privileged people. Additionally to the normal file system protection mechanisms, some other hardware and software protections may be used as :

- the writing of the audit data on write-once media such as optical disks,
- the use of devices with very small buffers that have a maximum protection,
- the restriction of physical accesses to the logging devices,
- the encryption of the audit data, ...

"At a minimum, the data on the audit trail should be considered to be sensitive, and the audit trail itself shall be considered to be as sensitive as the most sensitive data contained in the system." [NCSC 87/2]

2.4. Security rôles.

The only people who are authorized to read audit data, are those people who have **auditor**'s privileges. The *auditor* is an authorized individual with administrative duties, whose duties includes :

- the selection of the events to be audited on the system,

- the modification of the audit flags which enable the recording of those events,

- the analysis of the audit trail.

In some systems, the auditor's tasks may be dedicated to different trusted persons (at least two) to avoid the monopolization and the concentration of the auditing privileges in the hands of a single person. This measure is also important so that embezzlement cannot be perpetrated without collusion. Possible lasting damage caused by errors made by incompetent employees can be minimized, since each "trusted" individual would have only a limited rôle in the entire system.

2.5. Examples of audit trail mechanisms.

The next sections presents some audit trail mechanisms in terms of :

- the logging mechanisms : indicates how the auditable events are recorded into the audit trail and how the needed information is collected from.

- the solutions to the data volume : shows some mechanisms used to reduce the volume and the number of the audit records.

- the audit data protections : explains how the audit mechanism and the audit trail are protected against unauthorized accesses.

- the security rôles : presents which person(s) are responsible of the audit trail management.

At the end of this chapter, table 2.1 summarizes the principal characteristics of the audit trail generators presented in the next sections.

2.5.1. SAT.

The SAT (Security Audit Trail) working on a BS2000 operating system of SIEMENS NIXDORF INFORMATIONSSYSTEM, is a subsystem that supports the logging of <u>security relevant</u> pre-selected events in protected files, which can be analysed by means of the SATUT post-selection evaluator tool producing reduction files and/or result reports. The provided functionality covers the requirements of C2 class of the Orange Book.

The logging mechanism :

The audit events, which represent a subject acting (with a result) on an object (see figure 2.1.), are produced by calls to the SAT subsystem with the appropriated audit information. Furthermore, the data base systems are able to write security relevant information into the audit trail.

Solution to the data volume :

To face the problem of the data volume produced by the logging of the events, SAT compacts the data contained in the audit files.

Furthermore, the audit mechanism allows the pre-selection of auditable events. A security event is logged or not, according to the values of the "auditability" attributes of the subject (userid), the event-type and the object, and the result of the operation (i.e., success, failure or both). Note that some events considered as high security relevant are not switchable. This means that their attributes cannot be changed to be not recorded into the audit trail.

Last but not least, the post-selection is a function provided by SATUT which allows the auditor to select, from the SAT files, the part of information really interesting for his further analysis. But this function can be used for the building of an output file that replaces the input files and contains only the records really needed for the current security analysis.

Audit data protection :

"All accesses to the SAT-files are logged by an automatic setting of their audit attributes. Further more, Open/Close/Exchange of SAT files are considered as no switchable events and are always logged.

Security rôles :

To ensure trusted rôles against possibly threat being perpetrated by system security administrator or someone involved with the system security, SAT separates the auditor's functions from those of the system administrator. The "separation of the privileges" and "least privilege" principles are maintained by two main responsible rôles distributed to perform security tasks : "

- The **security officer**, who is responsible for :
  . the availability of the SAT functions,
  . the pre-selection of the SAT events to be logged in the collection files,
  . management of privileges including the auditor privilege.
- The **auditor**, who is responsible for :
  . the management of the SAT files, including switching of collection files,
  . the post-selection of the SAT events logged in the collection files, by using the SAT evaluator tool."

Some users can take part in the auditing : "the audit attribute of a file-object or a PLAM-library can be set by the file owner only when he is authorized. This authorization is granted by the group-administrator". [SAT 90/1]

2.5.2. RACF.

RACF (Resource Access Control Facility) is a program product running under a IBM MVS system and satisfying the C2 class of the NTCSEC. RACF provides functionalities as :

. identification and verification of users,

. user's authorization to access protected resources,

. control of the means of access to resources,

. logging and reporting various attempts of unauthorized access to protected resources.

The part in which we are interested is the logging part which provides a way to see who is using what resource.

The logging mechanism :

RACF writes security log records for detected unauthorized attempts to enter the system and, optionally, for detected authorized or unauthorized attempts of access to RACF-protected resources. Other auditable events are : identification and authentication of users, deletion of resources, and actions taken by privileged users.

Solution to the data volume :

RACF logging options allow a reduction I/O activity by restricting logging to specified access levels to individual resources.

Audit data protection :

The main protection of the audit data is provided by the RACF access control mechanisms.

Security rôles :

The security administrator has the overall responsibility for RACF implementation but RACF allows to choose between centralized and decentralized control through its ability to delegate responsibilities. RACF allows different users to perform different security tasks, such as auditing and security administration. For example, RACF can optionally notify a specified user of an unauthorized access attempt to a resource. [RACF 87]

2.5.3. SMP.

The SMP (Security Module Package) is a product of SECUREWARE Inc. that is designed to enhance and upgrade the UNIX operating system, its commands, and its utilities to the C2 class of trust as defined by the Orange Book. Audit trails produced by this subsystem can be used to detect penetrations of the system and the misuses of resources.

The logging mechanism :

Each audit record generated by the subsystem contains the user identification with the process effective and real user and group-ids. So the users can be hold strictly accountable for their actions.

The audit subsystem uses system call and utility usage to classify user actions into event types. The audit records are generated from three sources:

. The kernel audit mechanism : a large percentage of the audit records stored in the audit trail are generated by the kernel audit mechanism. This portion of the audit subsystem generates records in response to user process system calls that map to security-related events.

- The trusted application processes : To reduce the amount of audit data written to the audit trail and to make the trail more meaningful, some trusted applications (login, su[2], ...) are permitted to write directly to the audit device.

- The privileged subsystems : sometimes a subsystem encounters inconsistencies or problems that make the writing of an informative audit record desirable.

Solution to the data volume :

The audit data are compressed into packed record format that is stored in an audit compaction file by the audit compaction daemon[3].

Pre-selection options for record generation can be used to fine tune the audit trail. The user actions are classified into event types that can be used for selective audit generation and reduction.

The fact that trusted application processes have direct access to the audit trail by writing macro events rather than sequence of micro events, reduces the number of recorded events.

Audit data protection :

The audit data is maintained by an audit device driver as a set of audit collection files that can only be accessed by processes which have special privileges. This limits the access to the audit device only by trusted utilities such as the audit daemon and the audit administrator interfaces.

---

[2] su is the application that evokes super-user privileges.
[3] A daemon is a UNIX process running in background.

Security rôles :

The SMP audit subsystem is administered by the Audit Administrator who adjusts the audit subsystem parameters according to the performance and security requirements of the site. He is responsible for deciding how much information is recorded, how reliable it is recorded and for maintaining the information once it is collected. [SMP 88]

2.5.4. AIX.

AIX is a version of the UNIX operating system that runs on the IBM RT PC hardware. This UNIX version contains security features designed to satisfy both the class C2 security requirements of the Orange Book and some additional security requirements.

The logging mechanism :

Each event is described by an entry in an event table. The figure 2.2 illustrates the concept of an event table. Each entry is either a base event or an administrative event. A *base event* is either a system call name (e.g., fork) or an event in a trusted process (e.g., login-ok) or a non-system-call event in the kernel (e.g., RPC). An *administrative event* (e.g., login or object-create) is a convenient macro for an auditor that is defined by a set of base events and/or previously defined administrative events.

```
# Example of Event table :
# Base events : system calls
    access
    fork
    ...
    write
# Base Events : trusted processes
    adduser-fail
    adduser-ok

# Administrative Events
    login : login-fail, login-ok, logout-ok,
            su-fail, su-ok
    file-create : creat, open, openx
    object-create : file-create, ipc-create, pipe
    ...
```

Figure 2.2 *Event Table Entries.*

Solution to the data volume :

AIX includes audit trail file compression to face the produced amount of audit data.

The possibility to define two types of events (base events and administrative events) can reduce the volume of audit trails while selecting one of the two types for auditing.

In a same way, it is possible to define two classes of users to help the prefilter (i.e., selective collection) of the audit trail log file records. The two user classes are : the *general class* containing, for instance, unprivileged (ordinary) users, and the *special class* that may contain privileged (administrative) users.

Another way provided by AIX to avoid the flooding of the audit trail, is the possibility to turn off, or temporarily suspend then resume "normal" auditing (e.g., system call events) in a privileged process, and cut only a few selected records.

Audit data protection :

In addition to the compaction of the audit files, the audit subsystem may record user and system events on a write-once media.

Security rôles :

The two classes of users are defined by the auditor. He also associates with each class its own set of audit events. The auditor may create new administrative events by editing the event table.

The super-user has the privilege to control the auditing subsystem (enable and disable the auditing, specify the audit trail file, ...). He is the only one who can invoke the audit print command to generate audit reports. [AIX 88]

2.5.5. VAX/VMM.

The VAX/VMM security kernel is implemented as a virtual machine monitor (VMM) for the VAX architecture. The auditing facility for the VAX security kernel was designed and implemented to meet the requirements for an A1 class (NCSC) secure system. It includes the ability to monitor and signal impending security violations, to take defensive actions against attempted security violations and, the function in which we are interested : to create, maintain, and protect an audit trail.

The logging mechanism :

The virtual machine monitor supports several virtual machines simultaneously. These virtual machines are considered as untrustworthy subjects. The users are also subjects. All subjects are assigned an access class and can reference VAX security kernel objects when mandatory and discretionary controls allow. The VAX security kernel always audits as long as there are subjects on the system.

The VAX security kernel is constituted of layers, the Audit Trail layer being at a low level as shown by the figure 2.3. The higher layers audit accesses to the objects that they manage (the virtual printer layer manages the printers, ...). They audit security-relevant events by means of a call down to the Audit Trail layer.

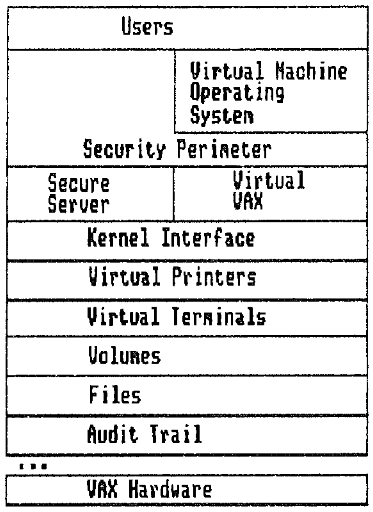

Figure 2.3 *VAX/VMM Layers.*

But not all layers manage objects. A parcel of the actions taken by computer operators and system administrators and/or system security officers is the use of commands calling such layers. Such users also have to be audited and that is the reason why audit events can also be produced at the command level. This auditing is performed after the command completes so that the audit record may contain the final resut of the command.

Solution to the data volume :

The auditing facility allows selectively audit based upon user's identity and/or object security level.

Counters and thresholds can be assigned to subject or object. The counters are updated depending on the events occurring in the system. When an audit request is sent by higher layers to the Audit Trail layer, this last one decides whether or not to collect the events into audit buffers based on the value of the audit counters and thresholds. However, the Audit Trail layer does provide a means by which higher layers may declare especially significant events that should be audited regardless of any audit thresholds.

Audit data protection :

Normal users and staff members are prohibited from performing functions delegated to the auditor. All of the mechanisms within the auditing facility are part of the TCB, and are protected by hardware from modification outside of the supplied software mechanisms.

A user with enough privilege can inspect an audit log, but can never overwrite its contents. The file system safeguards the contents of these files by allowing only read access for all opens initiated to them and requiring the auditor privilege to delete such files only using the supplied software mechanism.

In addition, all subjects, even the auditor, are always audited.

Security rôles :

The control over the audit trail can only be performed by the auditor. The auditor privilege allows the setting of audit and alarm thresholds as mentioned before and the archiving of audit logs.

2.6. Conclusion.

The principal goal of the security audit trail is to be **analysed** to deter, prevent and discover security anomalies or breaches. As the activity of each active user on the system generates audit records, these records are to be evaluated.

The Orange Book gives some criteria to realize efficient security audit trail mechanisms, to log the right information in the records, ... but it does not give any information about a way to analyse the audit information. Because of the volume of the data, this can not be entirely handled by a human operator reading the endless lists of audit records. Thus, a tool must be provided to analyse or help the security officer evaluating the audit trail.

There are several types of tools for this purpose :

- **reduction tools** or passive tools, that are presented in chapter 3,
- and **automatic analysis tools** or "more intelligent" tools, that are described in chapter 4.

Table 2.1 : *Summary of audit trail generators.*

	OB[4] Criteria	Logging Mechanisms	Selection / Data Volume	Audit Data Protection	Security Rôles
SAT V1.0	C2	- calls to the SAT subsystem - direct writing by data base systems	-pre-selection -post-selection -compaction -direct writing	-compaction -file protection -auditable commands	- security officer - auditor
RACF V1.9	B1		-selection based on resources, users	-RACF access controls	- security administrator - auditor - users
SMP V1	C2	-kernel audit mechanism -trusted applications -privileged subsystems	-pre-selection -direct writing -post-selection -compaction	-access limitations -compaction	- audit administrator
AIX	C2	-base event : system calls or event in a trusted process -administrative event: set of base events or administrative events	-base and administrative events -compaction -classes of users	-compaction -write-once media	-super-user -auditor
VAX/ VMM	A1	calls to the audit trail layer by the : - higher layers - command calls.	-selection based on users/objects security levels -counter and threshold system	-auditing of privileges people -software/ hardware protections	- auditor

[4] O.B. is the abreviation for Orange Book (see section 1.2.1 of chapter 1).

CHAPTER 3

PASSIVE OR REDUCTION TOOLS.

A reduction tool is a tool used by the security officer to reduce the amount of audit records generated and to only keep those ones that are really needed and interesting for further analysis.

An audit reduction tool is passive in that it is a combination of query language and data base manager and only selects records by applying filters. These filters are based on the contents of the fields of the audit records and combined with logical operators (AND, OR, NOT). The tool helps the security officer to accomplish his tasks, but **the evaluation of the audit trail is still made by the officer.**

There are some limitations involved by the tools of this category :

- the necessity of dividing (by selection of records) the audit trail to evaluate it, because of the volume. By such division, some information can be lost and anomalies be undetected;

- the high interconnection between the events makes difficult, and most of the time not possible, a deep evaluation of the audit trail by human people;

- the few knowledge about known intrusion scenarios and the impossibility to analyse all of them involves that other methods need to be used to detect **known** and **unknown** anomalies.

3.1. Functions provided to help the evaluation of audit trail.

The reduction tools are often supplied with the audit trail generator mechanisms and provide the security officer with several functions to analyse the generated audit data :

- selection of the records based on simple criteria or composed with "and", "or", "not", and brackets,
- sorting of the records based on one or several fields,
- report writer that makes a list of the anomalies,
- report generator that lists the selected records in a format readable for human operators,
- some tools provide alarm functions,
- ...

3.2. Kind of activities evaluated with reduction tools.

Here follow some examples of activities which should be evaluated with reduction/analysis tools. These activities are described as simple post-selection criteria that could be implemented for the evaluation of collection files[1]. These criteria are simple behaviours that are unusual or suspicious. There is no relation between the audit records.

[1] This study has been made for the evaluation of SAT collection files (under a BS2000 operating system) but can be easily extended to any other system.

- Detection of users acting (reading, writing) on objects of other users. A difference can be made between non-protected and protected objects[2]. For examples :

  . execution of other user's protected programs;
  . reading, copying of protected files;
  . attempts to act on protected files with incorrect passwords;
  . deletion of files that are not in the current userid[3];
  . modification (writing) of such files; ...

Particularly, all accesses to sensitive and privileged userid's (for instance the privileged userid $TSOS on BS2000) must be supervised.

- Detection of users attempting to modify files attributes, system parameters, passwords, ... (with a failure result).

- Certain commands and programs can not be executed by users who have not enough privileges. But certain commands or programs are not execution-protected but are not usually used by normal users. These commands and programs may be particularly supervised to detect misuses.

- A difference is to be made with those commands that can not be used by non-privilege users (audit commands, ...) and for which all accesses must be denied. This will result on a sequence of failed actions.

- Certain erroneous commands should always alarm the security officer. These actions will have a failure result. For example: someone attempting to modify parameters for the selection of audit events that are not switchable, ...

[2] An objection to this criteria could be that if the access control mechanisms work correctly, the intruder will not have access to protected objects. But the detection of such activity may reinforce the idea that an intrusion attempt is occurring and may allow the detection of who is doing what.
[3] On a BS2000 system the userid corresponds to a single directory owned by a user (or group of users).

- A database will contain information (static profiles) about the entire system like :

  . workdays and non-working days for the firm and for all users (days of week, ...)
  . official holiday and users' holiday
  . on- and off-hours, off-shift, lunchtime, ...

For example, if a user is connected on the system during one of his non-working days or during off-hours, he can be specially supervised.

- The result of the action (success or failure) is very important. If a sequence of failed actions is occurring, an anomaly is almost certain.

3.3. Examples of reduction tools.

The following examples are reduction tools provided by the logging mechanisms presented in chapter 2.

3.3.1. SAT.

The post-selection is that function provided by the SAT-Evaluator (SATUT) which allows the auditor to select, from the SAT files, the part of information really interesting for his further analysis. As shown in figure 3.1, the use of the SATUT has two main purposes :

a) The building of an output file (reduction file) that replaces the input files (collection files) and contains only the records really needed for the current evaluation.

b) The selection from the input file, of the records satisfying a post-selection condition, and the possibility to issue them on listings or files. In this case, the reduction file is simply a mean to store the results of the analysis and not to replace the input files.

The SAT Evaluator provides the principal functions :

- a selection function, to select from the input file(s) the records satisfying some post-selection conditions and to return the count of those record occurrences,
- an output function, which permits to show the selected records on listings,
- and a function to save the selected records in a reduction file.

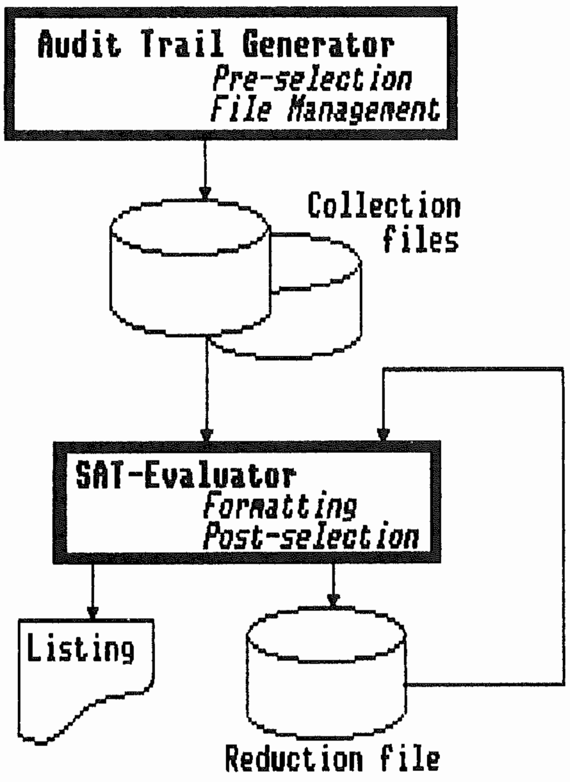

Figure 3.1 *SAT-Evaluator (From design of SAT project (SIEMENS)).*

The post-selection criteria are filters applied to the audit records. These filters are combined with logical operations AND, OR, NOT. The selected events may also be sorted based on the content of fields. [SAT 90/1]

In addition, alarm functions are provided in the BS2000. Alarm functions are a functionality that are no more passive but help the security responsible to detect anomalies. Security Alarms are real-time notifications to the auditor that attempts to violate system security policy may be under way.

3.3.2. RACF.

The RACF report writer is able to generate reports that:

- list the contents of RACF logging records,
- describe attempts to have access to a particular RACF-protected resource,
- list the warning messages that RACF issues during the "grace" period[4],
- list resources activity by resource owner,
- list the resource owners,
- describe users and groups activity,
- summarize system and resources uses.

These reports contain the user identity, the number and type of successful accesses, and the number and type of unauthorized accesses.

The user can request a summary report as well as many detailed summary reports.

Moreover, to provide relevant information to the security analyst, RACF maintains statistical information such as the date, time, and number of times that a user enters a

[4] The "grace" period is the period of time just before RACF control is enforced.

system and the number of time a specific resource was used by any user. In data set and general resource profiles, RACF optionally records the following statistics:

- the date the resource was last referred,
- the number of times the resource was used under each RACF authority (such as READ or UPDATE),
- the number of times that a specific user or group used the resource,
- the date the profile was last updated.

For user profiles, RACF optionally records statistics such as:

- the date and time of the last RACINIT[5] for a particular user,
- the number of RACINIT macros issued for a particular group,
- the date and time of the last RACINIT for a user to a particular group.

RACF provides a functionality that is no more passive and help in real-time in the detection of anomaly : it is the **alarm function.** For detected attempted security violations on the system, RACF sends messages to the security console. RACF immediately reports events such as detected unauthorized attempts to enter the system and, optionally, detected unauthorized attempts to have access to RACF-protected resources or modify the contents of the RACF data base. Optionally, RACF also notifies a specified user of an unauthorized attempt of access to a resource.

RACF is basically an open system in that resources not defined to RACF are not protected. Other systems are closed systems in which resources not declared as to be protected by the systems, are nevertheless protected. [RACF 87]

[5] RACINIT is a macro used during the login phase.

3.3.3. SMP.

There is a data reduction / analysis utility that is designed to facilitate the examination of audit trails from previous audit sessions or from the current one. The reduction utility is able to identify all of the compaction files needed to reduce an audit session.

To analyse an audit trail, the reduce utility supports a wide range of post-selection criteria that helps the administrator to target specific events, users, or objects. [SMP 88]

3.3.4. AIX.

The audit print command is only invoked by the super-user. It reads the audit trail and prints a report. It is possible to print all records as well as audit records that satisfy to a filter. The selectable options include user name (login, real, effective), event name (base or administrative), node name, and time (after or before) and can be combined with an "AND". [AIX 88]

3.3.5. VAX/VMM.

Security alarms are provided by the VAX security kernel in addition to the "normal" analysis or reduction tool. The security alarms are :

- Protection-Violation alarm : that is triggered by repeated attempts by a subject to violate privileges, mandatory access controls, or discretionary access controls,

- Covert-Channel alarm : is triggered when a subject repeatedly uses one of the system's known, audited covert channels,

- Login-Failure alarm : that is triggered by repeatedly failed attempts to log in to the system.

The "normal" audit reduction utilities provide help, through its commands, for security relevant decision. The auditor is able to select events based on some qualifiers as :

- before a date and time,
- after a date and time,
- produced by an individual (subject),
- containing a given string, ... [VAX/VMM 90]

3.4. Conclusion.

Looking at the few examples previously presented, it is already remarkable that some mechanisms base particularly their attention on the logging of the audit records and only provide tools with minimum analysis operations : record selections, translation into readable format, ... (see SMP, AIX, ...). Other mechanisms provide analysis tools with functionalities that really aid the security officer in evaluating audit trails. These functionalities are the sorting of the records, the re-evaluation of audit files, alarm functions, ... (see SAT, RACF, VAX/VMM, ...).

Some elaborated tools as automatic analysis tools, presented in the following chapter, are not provided with generator mechanisms. They are able to evaluate audit files coming from various mechanisms. They fill, like this, the lack of help brought by some audit trail generator mechanisms.

CHAPTER 4

AUTOMATIC ANALYSIS TOOLS.

As its name indicates, an automatic analysis tool or AAT can **automatically** analyse audit trails or help and guide the security officer evaluating audit trails. Such tool generally analyses : events or series of events that are suspicious or abnormal, user's profiles, behaviours of users, statistical data, ... to find anomalies.

An AAT can bring several solutions to resolve the limitations of reduction tools :

- As the evaluation is made entirely or partially by the tool with predefined rules, it is theoretically not necessary to divide the audit trail by selection of events (or only to reduce the evaluation time). All the security-relevant information contained in the audit trail can then be fully used to detect anomalies.

- The high interconnection between the events is no more an obstacle. Methods like **behaviour pattern recognition** describes some (un)expected behaviours with well-defined rules for the evaluation.

- Unknown anomalies and the few knowledge about known intrusion scenarios can be resolved by using other methods : **statistical analysis methods**, **methods that learn (ab)normal profiles** of user's behaviours, some other methods using **neural networks**, ...

All these methods will be introduced in the next sections[1].

[1] Methods using neural networks will be introduced in chapter 6.

One way of analysis is to evaluate the audit trail after it is recorded, it is called *off-line analysis*. Anomalies, intrusions and break-ins can only be discovered after they occur. This technique both detects and deters unauthorized actions. It does not prevent such actions.

Another type of analysis evaluates the events as they occur. It is thus possible for the tool to prevent unauthorized actions by analysing the events in real-time. This is called *on-line analysis*. The tool may then take an action against the detected anomaly as sending a message to the security officer, ... Therefore the security officer or the system itself can immediately perform the action to prevent from the discovered intruder performing his misuse.

4.1. Methods used by AATs.

To be a strong and efficient intrusion-detection system, an AAT should incorporate **several** approaches of evaluation, each of them detecting specific and particular types of anomaly, even whether some anomalies can be indiscriminately detected by several methods.

The following sections describe, more in details, the analysis methods : behaviour pattern recognition, statistical analysis and learning normal system behaviour. First, a little definition and/or description of the method is presented. Then one or more examples are presented to illustrate the described components implementing the method. Third, artificial intelligence techniques are developed to illustrate a possible solution to the realization of the presented analysis method. After the presentation of these methods of analysis, examples of existing tools or prototypes shows the possible interactions between the different methods, the reactions that can be taken, and the configurability of these tools.

4.1.1. Behaviour pattern recognition.

First of all there are analysis methods detecting single events or series of events, in the audit trail, that by oneself may be considered as abnormal and that considered in isolation are anomalous enough to raise concern. This is called **behaviour pattern recognition.**

4.1.1.1. What is a behaviour pattern?

A *behaviour pattern analysis component* is a mechanism capable of evaluating activities of users, systems, and objects (files, programs, ...) and recognizing **described** suspicious behaviours. A *behaviour pattern* is a description of the activity of a user, object or system. A *behaviour* can be represented in the audit trail by a single event as well as several consecutive (or not) events. The user, system or object behaviour is analysed without reference to whether it matches past or expected behaviour patterns. The mechanism is intended to detect those events that considered in **isolation**, are suspicious.

The description of such suspicious behaviours is based on knowledge of past intrusions and reported attacks, known system vulnerabilities, and the installation-specific security policy, as well as our intuitions.

This mechanism represents a superficial level of analysis. It operates with a very **narrow** view of the data and is, in some sense, **static** in its interpretation. It is *narrow* in that it generally involves only a small number of data items in its analysis; and it is *static* in that it does not make use of any statistical information. In effect, it is intended to detect those audit log entries that are, in isolation of any other information, anomalous enough to raise concern.

The security officer has to create some **simple or more elaborated criteria** to describe suspicious and abnormal behaviours.

The first thing a behaviour analysis component has to do is the selection or the detection of event records based on **simple criteria** (i.e., depending on the content of the fields of **ONE** event record). These simple criteria are **filters** and can be combined using "**and**", "**or**", "**not**", and **brackets**. The difference with the reduction tools is that variables may be used so that the content of the fields of the record representing the event may be compared.

For instance, a record representing an event where a user is accessing a file that belongs to him is described as : find a record containing an EVENT-TYPE field with the value 'access to file' and a USERID field whose value is stored in a variable USER and an OWNER field whose value is stored in a variable OWNER. Furthermore, the variable USER must be equal to the variable OWNER.

More elaborated criteria allows the tool to detect simple scenarios of attack. Thus it must be able to recognize not only single records but a sequence of records or a pattern. The pattern could be described as: "a record satisfying selection criterion A, followed by a record satisfying selection criterion B and finished by a record satisfying selection criterion C" [PDAT 89/2], each selection criterion being a **step** in the description of the scenario.

Variables may be used to make relations between the fields of the different records. For instance, three consecutive unsuccessful logins are considered as abnormal if they correspond to the same user. Three variables are to be defined, each containing the userid of each unsuccessful login record. An anomaly is detected if the three variables corresponding to three consecutive unsuccessful logins are identical.

In general it is possible to define behaviour criteria whose step can either be selection criteria (filter) or behaviour criteria that have been defined beforehand. Moreover, any step can be defined by a logical conjunction or disjunction of such previously defined criteria.

4.1.1.2. Some examples.

The **Protocol Data Analysis Tool (PDAT)** is a prototype, realized by SIEMENS and running on a SINIX system, that includes behavioural analysis. At present it is assumed that records are analysed off-line. This is possible to do this on a computer system different from the one that is being audited. It is planed to include functionalities which allow on-line analysis, statistical analysis and normal user's behaviour that can be learned by the system.

The PDAT was initially designed for the auditing of secure computer systems but it is flexible enough to be used for test analysing, diagnosis, optimization, validation and operational control. The tool is very flexible to be configurable with different manufacturers generated audit data (with different formats). [PDAT 88/1, PDAT 88/2, PDAT 89/1, PDAT 89/2, PDAT 89/3]

Here follow some criteria that, defined with the PDAT, recognize behaviour patterns in audit trails.

a) The first example illustrates a simple criteria (or filter) looking for events that describe unsuccessful accesses to protected files. For that goal, the security officer has to define criteria filtering records containing a event-type field with a value corresponding to 'file access' and a result field (result of the action) with a value representing an unsuccessful result, let's say F for Failed result. In our example, access to file will be represented by the events :

FRD for File Read Data,
FMD for File Modify Data,
FCL for Close File, ...

The PDAT-format selection criteria are defined as:

abbrev	type	pattern
a0	evt	FRD
a1	evt	FMD
...		
a10	evt	FCL
a11	res	F

where the first column is a name or abbreviation given to a criterion, the second column is the type of the record field (here evt for event-type, res for result) and the third column is the pattern that has to match with the field value. Then these criteria have to be assembled with the logical operators :

((a0 v a1 v ... v a10) & a11)[2].

Notice that wildcards as * for any sequence of characters and ? for any character, may be used. Supposing that all events describing file access, and only those one, begin with the letter F, it is then possible to define only two criteria equivalent to the above one :

abbrev	type	pattern
a0	evt	F*
a1	res	F

b) The second example introduces the concept of variables to compare the field contents of an event record. The filter is now : "looking for unsuccessful accesses to protected files and the file does not belong to the acting user".

[2] v is for logical OR and & for logical AND.

The same PDAT-format selection criteria are defined as before but two variables are introduced :

user_acting	that contains the value of the userid field of the filtered event records, and
user_target	that contains the value of the owner of the accessed file in the filtered records.

Then a relation is defined between these two variables indicating that the acting user is to be different of the owner of the file :

user_acting NOT EQUAL user_target.

c) The last example illustrates a behaviour pattern that recognizes a sequence of events. The behaviour pattern is defined to look for more than three consecutive unsuccessful logins for the same user.

The PDAT-format selection criteria are defined as:

abbrev	type	pattern
a0	evt	UCK
a1	res	F

where UCK represent an event that check whether the userid and the password correspond together when a user logs in. For unsuccessful logins, the result is to be a failed result. Then let's name the relation (a0 & a1) that describes one unsuccessful login : unsuccessful_login.

Now the behaviour is described as three steps, each one being an unsuccessful login :

first_step	unsuccessful_login
second_step	unsuccessful_login
last_step	unsuccessful_login
end_behaviour	successful_login

The end_behaviour step describes a successful login that permits to only consider consecutive unsuccessful logins. Otherwise the tool will detect all users making more than three errors while logging in the system. And who has never made more than such three errors ? Note that the steps can be different criteria.

We have now to define three variables to impose that the three unsuccessful_login criteria are for the same user :

```
USER1 that contains the value of the userid field
      of the first step,
USER2 that contains the value of the userid field
      of the second step,
USER3 that contains the value of the userid field
      of the third step,
```

Moreover a fourth variable (USER4) also associates the end_behaviour that has to refer the same user.

These variables are logically assembled :

```
USER1  EQUAL  USER2
       AND
USER1  EQUAL  USER3
       AND
USER1  EQUAL  USER4
```

Some other examples of elaborated behaviour criteria are presented in appendix A.

4.1.1.3. Artificial Intelligence techniques for behaviour recognition.

The architectures of two approaches making use of artificial intelligence (AI) techniques are now developed to resolve the implementation of a behaviour pattern recognition analysis component. The first approach utilizes simple knowledge base system while the second implements an Expert System (ES).

The figure 4.1 proposes the first architecture implementing a behaviour pattern recognition component with a simple knowledge base system as AI technique.

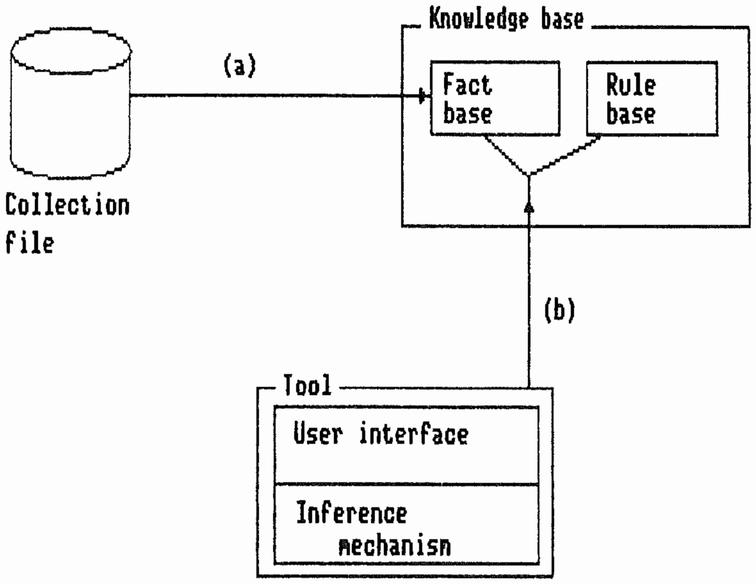

Figure 4.1 *Architecture for behaviour pattern recognition.*

In the architecture proposed in the figure 4.1 the event records of the collection file are translated into the fact base (a). In a first time, the user can directly ask questions to the knowledge base by asserting new rules and new facts (b). In a second time, the questions can be asked automatically by the tool by applying the more interesting rules found in the first time. These rules may describe simple criteria or sequences of events that are to be found.

This is a simple way to use AI techniques. This approach allows to search and create new behaviour patterns due to the interaction between the tool and the security operator.

Another (more elaborated) solution is to use an expert system. First of all the utility of the creation of an expert system is introduced by presenting what is an expert system and how helpful could be an expert system in analysing behaviour patterns.

An expert system is a computer program that emulates the behaviour of a human expert within a specific domain of knowledge. The major characteristics of an expert system are :

- **the ability to perform at the level of an expert,**
- **the representation of domain-specific knowledge in the manner in which the expert thinks** : the expert knowledge is separated from the general-reasoning mechanism,
- **the incorporation of explanatory mechanisms and ways of handling uncertainty** : the E.S. is more tolerant of errors and imperfect knowledge than conventional programming,
- and typically, **a prediction for problems that can be symbolically represented** : the knowledge may be processed in a manner that is not procedural.

"Expert systems are helpful in situations where an expert is unavailable, where expertise is scarce or expensive, or where the decision maker is placed under time and pressure constraints". [Parsaye 88]

The main components of an expert system are :

- the knowledge base where facts and rules are stored :
  - . facts can be used to refer to either permanent or temporary knowledge and are stored in a fact base,
  - . rules combined with facts can be used to arrive at conclusions which can be used as new facts; the rules are stored in a rule base.
- the inference mechanism that processes the knowledge,
- and the user interface that permits communication with the user.

Other components like the following ones can be added :

- a knowledge acquisition facility that can interact with the expert to extend the knowledge of the expert system,

- an explanation facility that provides to the user a justification of the found solution,

- a self-training facility that can assert new facts developed by the expert system into the knowledge base. [Martin 88]

As mentioned in its definition, an expert system has the knowledge of an expert human for a specific domain of expertise. And what is the security officer, if not an expert? And what is audit trail analysis, if not a specific domain of expertise?

Given the architecture of an expert system and the separation of the knowledge base and the reasoning mechanism, the "data" that are facts and rules stored in the knowledge base, are easy to examine, change and extend. A characteristic of expert system is that there is no necessary programming for how the system uses the knowledge that is presented to it. The possibility to assert rules or facts without having to implement an algorithm is a great advantage because of the necessity of updating continually the knowledge of the analysis tool. Moreover any facts or rules can be changed at any time, and in most cases the change can be made independently of one other.

All attack and intrusion scenarios can not be known and introduced in the knowledge of the tool. If a new attack scenario is discovered or a specific object or user is to be focused, it must be possible to modify, retract or assert facts and rules as rapidly as possible.

Another advantage of expert system is that it can justify the solution it has found and it can explain the reasoning it made to find this solution. This could be helpful in analysing audit trail because of the possible uncertainty of the type of an anomaly. A new user or somebody performing unusual tasks or making manipulation errors can have a behaviour that is considered as suspicious by the system. The security officer can thus analyse the justification provided by the expert system about the anomaly.

Expert systems can also deal with uncertain or incomplete information. And in some cases, what is more uncertain than an intrusion attempt? Imperfections in knowledge take a variety of forms : a piece of information being missing, or a less extreme possibility is that the information is likely rather than absolutely certain, or that it is vague or imprecise. It must be possible to express to the expert system that a rule is not certain at 100%, but that is true in, i.e., 75% of the all cases. One may know how likely or probable some proposition is, without knowing whether the proposition is true or false.

It must be also possible that the expert system expresses the solution it founds with an uncertainty, depending on the information hold in the knowledge base.

Now the different components of the proposed expert system are presented. In a first time, a rule-base system encodes the knowledge in the form of individualized reasoning rules. These rules have a simple form like if-then or antecedent-consequent forms. The rule-based component is in general simple to managed because of the independence of the rules. Small additions are allowed to be made incrementally without the necessity of understanding the entire control structure.

We can imagine a rule base composed of rules like :

```
IF   (behaviour pattern A)
THEN (activate the research of behaviour pattern B)

IF   (behaviour pattern A AND behaviour pattern B)
THEN (any behaviour pattern C is abnormal)
```

or in a more general form :

```
IF   (behaviour pattern)
THEN (actions).
```

to describe scenarios of attack that are sequences or occurrences of events or behaviours.

It is also possible to describe more elaborated scenarios by using variables to make comparisons between rules. In fact, such a system could be implemented by several different rules that are interconnected together and make reference to other rules and results of researches.

Moreover it is possible to use **static and dynamic tables** as other sources of information. These tables are links that permit interconnections between rules, results and the state of the system.

A *static table* connected to the tool will contain information about the system and its environment. It is information which **does not depend on a specific session.** The contents of the static table are rarely changed and can only be changed by the operator of the tool. As example, a static table may contain the normal location of the users (terminal), the full user-name, the address, the normal working hours of all users, the public holidays, ...

The *dynamic table* will be helpful to maintain a general knowledge of the system **depending on a specific session.** It is impossible to store all information about the state of the system at a certain point of time in every single record so the dynamic information can be stored in a table to make reference to other records or events. For example, events may only record the relative path of a file while the dynamic table contains the current directory where the user is working in. The dynamic table may contain other information as the current location where the user logs in, the task sequence number (TSN), the login userid, the active processes, the opened files or objects, the active connections, ... This dynamic table is updated by the tool depending on the contents of the analysed records. The records have to be worked through in sequential manner (sorted by time) since information which might modify the internal table, is provided via the audit records.

Both static and dynamic tables contain known information about the global state of the system while the information held in the audit trail is the one to be analysed.

The figure 4.2 describes an architecture where the tool is an expert system that guides the research of anomaly into the audit trail thanks to the rules stored in the rule base and those ones eventually provided by the user.

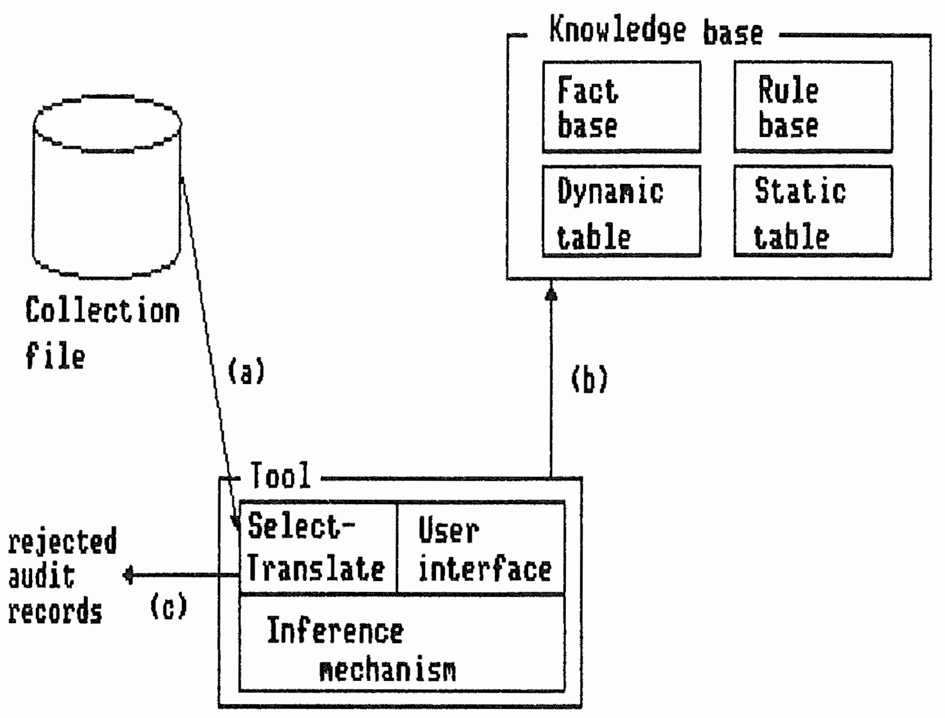

Figure 4.2 *Architecture for Expert System method.*

In the architecture proposed by the figure 4.2, the knowledge base is composed of :

. a rule base describing :
- behaviour pattern rules,
- actions to be taken,
- rules added by a user to analyse a specific session, ...

. a fact base representing :
- facts that come from the audit trail,
- new facts added for intermediate analysis, ...

. a dynamic table, and

. a static table.

To avoid the problem of flooding the knowledge base with unnecessary audit records, the event records of the collection file are read in by the tool one at a time (a). A selection is made depending on the contents of the knowledge base. The selected records are translated in facts and logged into the knowledge base (b). Reading the event records, the tool updates the knowledge base by analysing them. The tool can :

a) ignore the event because of :

. no utility for the current security analysis,

. the security administrator has not selected this type of event (not useful for a certain period of time or not relevant for this user, etc.)

Notice that this kind of selection can be done by the pre- and post-selection criteria provided by existing audit generator mechanisms, but some events may be necessary to update the knowledge base, thus they need to be logged in to be analysed, but do not need to be translated into the fact base. In figure 4.2, this is represented by the paths (a) and (c).

b) translate an event record and logs it in the fact base for further analysis. In figure 4.2, this is represented by the paths (a) and (b).

c) update the knowledge base (profiles, dynamic and static tables, activates new rules, adds new facts, ...) and ignore (see a.) or logs (see b.) it. This is represented in the figure 4.2 by the paths (a) and (b) for the update and (a) and (c) for the reject of the event record.

Then, depending on the contents of the knowledge base, the tool can take actions such as alerting the security officer, shutting down a task or the system, ...

The user can act on the selection of event records. He can also assert new facts and new rules in the knowledge base. He can interact with the inference mechanism and supply heuristic.

If we consider the last architecture (figure 4.2) and add more useful components relating to expert systems as described in section 4.1.1.3, we obtain the architecture shown by the figure 4.3.

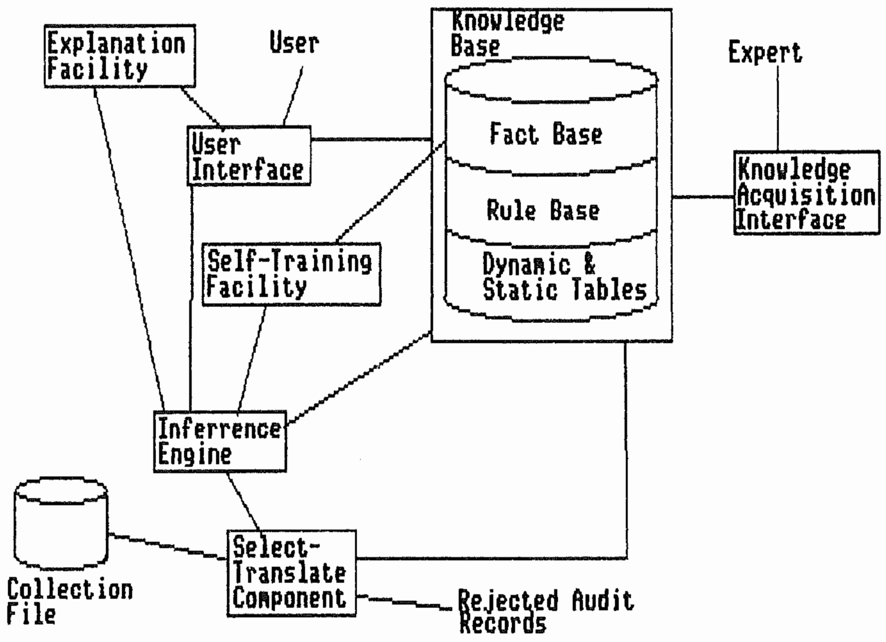

Figure 4.3 *Architecture and expert system components.*

4.1.2. Statistical methods.

4.1.2.1. What is statistical method?

The behaviour pattern recognition analysis method allows the recognition of well-defined scenarios of attack. But as previously mentioned this is not sufficient because of the continually-renovated imagination of the intruders. Rather than describing abnormal activities to be found, statistical methods can be used to determine normal system behaviours. These normal behaviours are summarized with some statistical values in terms of mean, variance, standard deviation, ... so that anomalies can be detected by comparing these computed values and the measured ones. It is, for instance, possible to analyse the number of time selection criteria or behaviour criteria occur, then this number can be compared with a certain threshold value.

By using statistical methods, it is also possible to analyse the content of records satisfying certain criteria. Statistical measures can therefore be computed based on the content of fields of the audit records[3].

4.1.2.2. Some examples.

As an example analysing the occurrence of event records, the rate of unsuccessful login attempts among all login attempts can be compared with a certain value. If this rate suddenly increases considerably, a break-in attempt is almost certain by guessing passwords. Note that this approach can detect a high rate of failed logins involved by erroneous passwords for one userid, but it can also detect a high rate of login failures based on somebody attempting to enter the system by applying one possible password to all userid.

The second example illustrates the analysis of the contents of records satisfying certain criteria. "One might be interested in the login time of the users. Thus one might activate the selection criterion selecting all successful login attempts. The statistical method would be to find the average login time and the variance of it. A great discrepancy between the actual login time and computed average login time may indicate that an illegal user has logged in under a legal user name." [PDAT 89/2]

4.1.2.3. Artificial Intelligence Techniques for statistical methods.

Artificial intelligence techniques are not required for the statistical analysis of audit trails. Some statistical values relating to behaviour patterns may be computed in relation with AI techniques presented in the previous architectures of figures 4.1 and 4.2.

[3] Note that statistical methods are especially used by the "learning normal system behaviour" approach that is described later in this chapter.

4.1.3. "Learning normal system behaviour" method.

4.1.3.1. What is "learning normal system behaviour" method?

A component that learns normal system behaviours uses statistical analysis methods. The idea is to measure particular features of the audit trail, and compare these measurements with historical activity. It is possible to determine historical activities for users, but also for the system in general and for files or any objects (programs, resources, ...). Historical activity can also be defined for subject-object relations. This means that a certain behaviour is to be maintained for a specific user or group of users having access to a specific object or class of objects.

The statistical analysis component maintains statistical knowledge about subjects consisting of profiles (representing the historical activity). A *profile* is a description of a subject's normal (expected) behaviour with respect to a set of intrusion-detection measures as this is described in the next section. Profiles require a minimum amount of storage while keeping only statistics such as frequency tables, means, and covariances. Thus a user's work-profile might consist of a description derived from his job description, augmented with a continually updated summary of the user's individual historical activity.

The statistical knowledge is updated by adaptively learning subject's behaviour patterns. As subjects alter their behaviours, their corresponding profiles change. Usually, the statistical analysis component will keep a knowledge of past user behaviour and compare it to the actual behaviour. The knowledge about past behaviour should be kept in such a way that the information should be weighted to differentiate old behaviour and more recent one. "Behaviour of a system a year ago is not as useful in trying to establish normal system behaviour as the behaviour of the past 24 hours" [PDAT 89/2].

It is possible to determine whether observed behaviour as reported in the audit data is normal with respect to past or acceptable behaviour characterized by specific **intrusion-detection measures.** The IDES[4] analysis tool proposes two types of measures :

a) "*Continuous measure* (also referred as ordinal) is a count of some numerically quantifiable aspect of observed behaviour" [IDES 90/1].

   It is for example a "connected time" measure that records the duration of a user session , "CPU and I/O time" recording the amount of processing time and I/O activities, ...

b) "*Discrete measure* (also known as categorical) is a function of observed behaviour over a finite set of categories. Its score value is determined by its frequency relative to other categories.

   . *binary categorical measure* has a score function that does not count the number of times that each category of behaviour occurs, but only whether or not the category was invoked.

   . *linear categorical measure* has a score function that counts the number of times each category of behaviour occurs." [IDES 90/1]

It is for instance, the "time of login" that records the login time each time a user logs on, the "location of login" that records the physical location when a user logs on, ...

The measures are applied to individual sessions (i.e., from login to logout), to days, to hours, or to other time periods, depending on the measure and on the type of subject.

[4] The Intrusion-Detection Expert System (IDES) is presented in setion 4.2.

The measures are to be compared with an historic. This comparison is performed by **models.** A *model* is the function which compares the profile values with the measured values. If a current value does not correspond to a defined profile, a violation is produced. A *violation* is a way to signalize an anomaly. A warning count (rating) linked to a user or group of users is incremented depending on the seriousness of the anomaly.

More precisely, when one has a metric for a random variable x and n observations named x_1 to x_n, the statistical model of x is going to determine whether a new observation x_{n+1} is normal or not with respect to the previous observations.

Depending on the measure, different models may be used :

- The Operational Model :

  The new observation x_{n+1} is compared with fixed limits. The previous observation of x are not used and the limits are determined by experience or intuition.

  For example : an event counter for the number of password failures during a brief period, where more than 10, suggests an attempted break-in.

- The Mean and Standard Deviation Model :

  The new observation is compared with the prior observations x_1 to x_n from which the mean and the standard deviation are calculated. The new observation x_{n+1} is defined to be abnormal if it falls outside a confidence interval that is a standard deviation from the mean for some parameter d :

$$\text{mean} \pm d * \text{stddev}$$

where sum = $x_1 + \ldots + x_n$
sumsquare = $x_1^2 + \ldots + x_n^2$
mean = sum / n
stddev = sqrt (sumsquare / (n-1) - mean^2)

Deviation : the basic average is maintained to determine deviation from the mean. To save space, a close approximation can be computed with old average and the new value. The length of history needs to be recorded to ensure proper weighting of the new value into the average.

- The Markov Process Model :

  The Markov Process Model applies only to event counters. Each distinct event type (audit record) is regarded as a state variable. A state transition matrix is used to characterize the transition frequencies between states (rather than just the frequencies of the individual states taken separately). New observations are considered as abnormal if its probability (as determined by the previous state and the transition matrix) is too low. "This model might be useful for looking at transitions between certain commands where command sequences were important." [IDES 87/1]

- The Time Series Model :

  This model uses an interval timer together with an event counter or resource measures. It takes into account the order and interarrival times of the observations $x_1, \ldots, x_n$, as well as their values. A new observation is considered as abnormal if its probability of occurring at that time is too low.

- Other models like exponential model, multivariate model, ... can be used. [IDES 90, MIDAS 88]

4.1.3.2. Some examples.

Some profiles of "normal behaviour" are listed in appendix B. Appendix B1, B2, B3 and B4 respectively presents profiles for users, system, objects, and subject-object relations.

As a practical part of this thesis, a prototype has been realized implementing a learning normal system behaviour component. This prototype, illustrating the use of profiles as previously described, is presented in chapter 5.

There exist other methods that make other uses of profiles to learn system behaviours. Two tools : TIM and Keystroke dynamics illustrate these other learning approaches.

a) TIM.

A time-based inductive learning approach has the potential of detecting masqueraders or misfeasors based on deviations from the known sequential patterns of a user. TIM (Time-based Inductive Machine) discovers temporal patterns in the form of sequential decision rules generalized from input data called episodes. This set of hypotheses is maintained and modified dynamically.

TIM was originally developed as a general-purpose tool to discover temporal patterns from observations of a temporal process, where the patterns represent highly repetitive activities and can be used for prediction with high accuracy.

TIM always attempts to find better hypotheses as long as it receives input data (sequence of events). TIM attempts to generate rules that accurately predict the occurrence of a specific type of events.

TIM can be applied to security auditing. It can be used to generate rules that predict the occurrences of certain specified events.

Auditing events are processed as soon as they are received. Security event rules generated via TIM, describe the behaviour patterns of either a user or a group of users based on past security audit history. Furthermore, each rule describes a sequential pattern that predicts the next possible events with satisfactory accuracy.

An example of a specified rule produced in TIM follows :

```
E1-E2-E3 --> (E4=95% ; E5=5%)
```

where E1, E2, E3, E4 and E5 are security events in the form of rules. The rule indicates that if E1 is followed by E2, and E2 is followed by E3, then there is 95% chance that E4 will follow and 5% chance that E5 will follow.

More advanced rules can be generated. For instance,

```
E1- * --> (E2=100%)
```

where * matches any single event. Any number of *'s in a rule is allowed.

Assuming that sequential rules are to be discovered from the following events where each letter represents an event :

```
A-B-C-S-T-S-T-A-B-C-A-B-C
```

the following rules may be generated :

```
R1 : A-B --> (C = 100%)
R2 : C   --> (S =  50% ; A = 50%)
R3 : S   --> (T = 100%)
R4 : T   --> (A =  50% ; S = 50%)
```

R2 and R4 may be deleted from the rule base eventually, because they may lead to wildly divergent predictions and are not "good" hypotheses.

R1 and R3 may remain in the user profile, because they seemed to cover or explain more events more accurately, and could be used for deviation detection in the future.

Deviation detections and detections of unrecognized activities can determine unusual activity. For example :

```
R1 : A-B --> (C=100%)
```

and then the sequence of events A-B-D will be considered a violation of the pattern.

The sequence of events A-C-F will be considered as unrecognized activities.

Such activities can be presented to security management for further examination. These activities could also be used to generate new rules. [TIM 90]

b) Keystroke dynamics.

Some systems use a keystroke dynamics mechanism. This is an extreme example of the use of artificial intelligence in automatic analysis tools. A keystroke dynamics is based on the "fit of sender" concept from the days of the telegraph when Morse Code operators could identify a sender by listening to the incoming signals. Some tools use over 110 typing characteristics, including intervals, rhythm, an analog of pressure, and error characteristics.

4.1.3.3. Artificial Intelligence Techniques to learn normal behaviour.

The use of AI techniques are illustrated by the prototype presented in chapter 5. It is the implementation of a knowledge base (fact base and rule base) in the PROLOG language.

4.2. Combination of the analysis methods.

The conclusion that an intrusion attempt is occurring is to be made very carefully to avoid false alarms. It is not possible and of no use to detect all anomalies that possibly may occur on the system. It is not possible in that misfeasors always imagine new attacks against a given security system mechanism. The goal of an AAT is not to detect 100% of the anomalies occurring in the system. It must detect the greater number of breaches as possible avoiding false alarms. If a too big number of false alarms warns the security officer, this one will no longer have trust in the AAT. Not all anomalies have to be discovered because all of them do not cause damages to a given system. A second reason is that an AAT acts in a great part as a deterrent against ill-disposed people.

To be a strong intrusion-detection system that detects a great number of "attacks" with a small number of false alarms, an AAT should incorporate several different approaches of analysis. For instance a statistical user profile approach that detects some unknown "attacks", augmented with a behaviour pattern recognition method that characterizes well-known intrusions, promise to be an effective combination. The different components may operate in parallel, sharing the same source of audit records and producing similar anomaly reports, but with the internal processing of the two systems done in isolation. At some point a resolver may be introduced to combine and further post-process the outputs of the components as shown by the figure 4.4.

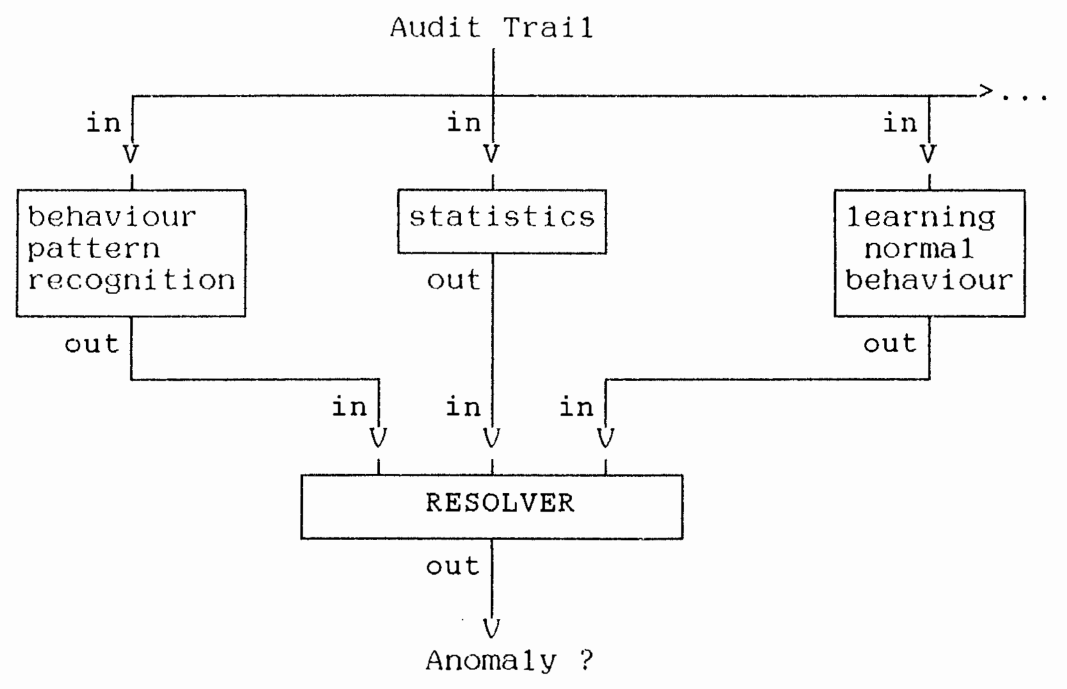

Figure 4.4 *Combination of the different analysis components.*

The PDAT, presented is section 4.1.1.2, plans to incorporate in the future several analysis methods as behavioural pattern recognition, statistical and learning normal system behaviour methods. Two other tools : IDES and MIDAS already combine several methods of anomaly detection.

a) IDES

The idea of the Intrusion-Detection Expert System (IDES) is to use innovative statistical algorithms for anomaly detection, as well as an expert system that encodes known intrusion scenarios. IDES is designed to operate in real time to detect intrusions as they occur. IDES can analyse audit trails of other target machines that are entirely responsible of the logging mechanism.

The IDES system is based on two approaches:

- intrusions can be detected by flagging departures from historically established norms of behaviour for individual users, and

- known intrusion scenarios, known system vulnerabilities, and other violations of a system's intended security policy are best detected through the use of an expert system rule base.

The IDES statistical anomaly detector (IDES/STAT) maintains a statistical knowledge base about subjects, consisting of profiles. A profile is a description of a subject's normal (i.e.: expected) behaviour with respect to a set of intrusion-detection measures[5]. Examples of measures used by IDES are :

. "I/O usage", "CPU usage" that indicate the amount of I/O usage and CPU usage by a particular user,

. "physical location of use" that records the number of times a user is connected to the target system from different locations, ...

The IDES/STAT adaptively learns subject's behaviour patterns: as subjects alter their behaviours, their corresponding profiles change. The IDES/STAT is driven by the arrival of audit records and by examining them as they arrive from the target system. It is able to determine through a variety of statistical algorithms whether the observed activity is abnormal with respect to the profiles.

[5] See section 4.1.3.1.

The expert system component of IDES evaluates user's behaviour on the basis of a set of rules. These rules describe suspicious behaviours based on knowledge of past intrusions, known system vulnerabilities, and the installation-specific security policy. The IDES expert system component operates in parallel with the statistical component. It is planned to introduce a resolver to combine the outputs of the two components. [IDES 87/1] [IDES 87/2] [IDES 88] [IDES 90/1] [IDES 90/2]

b) MIDAS

The Multics Intrusion Detection and Alerting System (MIDAS) approach to detect intrusions is developed as an intrusion-detection system (expert system) that encodes a-priori rules that define an intrusion. An activity is evaluated by comparing norms derived from past activity aggregation to on-going actions. The current activity is then determined as abnormal if it is outside some standard deviation.

MIDAS monitors at the user command line level and logs all commands used. Midas is implemented on a stand-alone symbolic LISP machine. It uses a home-grown expert system shell. Its rules are elaborated in LISP, and statistical user profiles are maintained in LISP structures.

Midas uses three types of heuristic rules :

- **immediate attack heuristics** are intended to detect those events that considered in isolation, are suspicious. These rules make no use of information of past or expected user behaviours.

- **user anomaly heuristics** use statistical user's profiles to detect when a user's behaviour departs from a pattern established by observing past behaviours. User's profiles are updated at the completion of a user session. The profiles contain a list of the user's usual commands, the usual access times and location for the user, and the expected typing rate for the user. MIDAS also profiles the observed behaviours of remote systems.

- **system-wide state heuristics** defined system-wide profiles maintained to characterize what is normal for the system globally.

MIDAS combines different intrusion indicators to decide whether an intrusion is occurring. A login time unusual for a given user, for example, is not alone sufficient to raise an alarm; but if combined with other anomalous data, MIDAS might decide an intrusion was in progress.

In its current implementation, audit data are dumped to tape and then fed into MIDAS. A real-time capability is planned for a later implementation phase. [MIDAS 88]

4.3. Actions taken by AAT.

The comparison between the measurements and the historical activity records does not directly reveal misuse, but rather indicates a degree of concern or a warning count. The system, objects and users can be given a suspicion rating. This rating starts at zero and is incremented with each observation of suspicious or improper behaviour. The incremented amount is dependent upon the seriousness of the observed mis-behaviour. When the rating of a particular user, object or system exceeds a pre-set threshold value, an anomaly report is generated. It could be useful that a textual justification of the conclusion was generated.

As an anomaly is detected by the AAT, actions is to be performed. These actions may alert the security responsible by :

- sending of message to the security console,
- writing a message into a file,
- sounding an alarm, ...

so that he can analyse the seriousness of the warning and take the best counter-measure.

Another type of actions may directly respond to an "attack" if the anomaly can be detected with accuracy. These automatic actions may be, for instance :

- to shut down the system, a specific terminal or session;

- to close down a certain service;

- to strengthen the protection mechanisms by limiting access to objects;

- to modify a knowledge or data base;

- to modify the logging selection of events in the audit trail as the activation or the cancelling of pre- or post-selection criteria : for example, if the activity of a user is not totally recorded into audit files and if a certain kind of anomaly occurs for this user, then it should be possible to automatically modify the (pre)selections and therefore logs all events in the trail for this user;

- as previously mentioned in expert systems, actions may also activate new rules (searching for more specific intrusions, ...) and then dynamically change elaborated criteria;

- ...

Automatic (re)actions only have sense with on-line analysis, trying to thwart the misfeasor or the intruder. The actions performed as part of off-line analysis act principally as mending, and as detection of the perpetrator identity.

The great - alredy-mentioned - problem of automatic actions is the false alarms. One possibility reducing false alarms is the use of **violation rate** associated with each user. When an anomaly is detected for a given user, a warning is not directly sent but a rate is incremented with a value depending on the seriousness of the anomaly. It is only when the violation rate of the user reaches a threshold that an action is to be performed. With such method, different intrusion indicators have to occur to decide whether an intrusion is performed. A login time unusual for a given user, for example, is not alone sufficient to raise an alarm; but if combined with other anomalous data, the AAT might decide an intrusion was in progress.

4.4. Configurability.

An AAT has to be configurable enough to analyse systems in heterogeneous environments, while security threats can have vastly differing significance for different systems. For some public databases systems, unauthorized access to information represents little or no threat, while for some banks, military command and control systems, it represents a severe attack . Some other systems have to be protected against external intrusions while other systems have to be safeguarded against breaches of internal users.

Any AAT must provide a way for a security officer of a specific system to create his own criteria of research and define his own normal or expected behaviours. Moreover, in some system environments (banks, ...), the expected behaviours of the users are supposed to be the same during all the time. In other environments (research and development centers, ...), the activity of the users is modified depending of the work they do, so that in some cases, the profiles describing normal user behaviours have to be flexible and adaptable.

4.5. Summary of the presented AATs.

The table 4.1 summarizes and compares the AATs (PDAT, IDES, MIDAS and TIM) presented in the previous sections in terms of the performed activities, the main components (analysis methods) being part of the AAT, the main mode of analysis (off/on-line), some particularities of the tool or the methods employed to evaluate audit trails, and, finally, what is planned in the future for these AATs.

Table 4.1 *Summary of AATs.*

	Activities	Components	Mode	Particularity	Future
PDAT	.auditing of secure computer systems .possibility of : -test analysing diagnosis -optimization -validation -operational control	.Behaviour pattern analysis : - filters - behaviours - variables	off-line	.other target systems .maintains -dynamic and static tables	.Statistical analysis .Learning normal system behaviour .on-line analysis
IDES	.evaluation of audit trail	.Statistical algorithm (user's profile) .Expert system (known intrusion scenarios) The two components analyse audit trail in isolation	on-line	.other target systems .maintains profiles of subjects	.Neural network component .Resolver to combine the analyse of the 2 components and post-process them
MIDAS	.logging of events .detection of intrusions	.Intrusion detection system (expert system) that encodes a-priori rules .Use of past activity knowledge (statistical user's profiles) .Includes an explanation facility	off-line	.logs at the user command line level .combines different intrusion indicators to decide whether an intrusion is occurring	.on-line analysis
TIM	.general purpose tool to discover temporal patterns from observations of temporal process .can be used to detect masqueraders or misfeasors (security auditing)	.Time based inductive learning approach	on-line	.try to predict occurrence of certain specified events	

4.6. Conclusion.

At the present, AATs are a great help for people who are responsible of analysing security audit trails. In some advanced cases, the security officer can be helped to take a decision about the reaction to a specific breach. AATs are not able to realize automatically and totally the task of evaluating audit trails. A human operator has to decide whether a suspicious activity is effectively an anomaly. It is of a great importance that AATs must have several approaches of evaluation whose outputs have to be analysed by a human individual or by a specialized component (resolver).

CHAPTER 5

PROTOTYPING-REALIZATION OF A COMPONENT FOR THE RECOGNITION OF (AB)NORMAL BEHAVIOURS.

This chapter presents a prototype realized to show the feasibility of a "learning normal system behaviour" component using statistical analysis methods as described in section 4.1.3 of chapter 4.

The prototype implements a "learning normal system behaviour" component using statistical methods and profiles to characterize normal users, objects and systems behaviours. It is implemented in PROLOG language and is running on a SINIX machine independently of the system generating the audit trail : in this case, the SAT[1] (Security Audit Trail) of a BS2000 operating system.

5.1. The components of the prototype.

The most important components being part of the prototype are now discussed : they are the input to be analysed (the audit trail), the profiles that retain the historical knowledge, and the rules that actually maintain the profiles and evaluate the audit trail.

5.1.1. The input.

The input of the prototype is a collection file (audit trail) created by any audit trail generation mechanism. It is under the responsibility of this mechanism to provide the file to be analysed in a given prototype-readable format. So, a utility first translates the SAT collection file to be analysed in a sequence of PROLOG lists (the prototype-readable format), each list

[1] See section 2.5.1 of chapter 2.

corresponding to an audit record. The SAT event records are composed of **fixed and variable parts**. The *fixed part* is composed of fields containing information that exists for all records (userid, timestamp, ...). The *variable part* contains only the information that has a meaning for the type of the logged event. For example, it does not make any sense to keep a field containing a filename for a login event because no file is directly concerned. The variable part of the record is also referred as the **information part**.

The prototype-readable format of an audit record is :

```
( [ field_name_1 (information_1), ...,

    field_name_J (information_J),

    field_name_J+1 (information_J+1), ...,

    field_name_N (information_N) ] ) .
```

where the field_name_i are the names of the fields and information_i are the corresponding values of the information (with i going from 1 to N). The fields 1 to J are the fixed part while the fields J+1 to N are the variable part of the record. Here follows a concrete example of two audit record in a PROLOG list format[2] :

```
([userid(Peter),
  tsn(opiu),                        fixed
  date(01-01-91),                   part
  time(09h30m15s03),
  event-type(login),
  result(success),

  station(terminal_0012) ]).        variable
                                    part
([userid(Peter),
  tsn(opiu),
  date(01-01-91),                   fixed
  time(09h30m21s08),                part
  event-type(file access),
  result(success),

  file-name(newdata),               variable
  access(input-output) ]).          part
```

[2] These formats are not exactly those used by the prototype, but are simplified to be understandable.

The six first elements of each event list are the fixed part : the user identification, the identification of the task[3] on which the user is logged, the date and time of the action, the type and the result of the action. In the first record example illustrating a login action, the variable part contains the location where the user has logged in, and in the second record example illustrating a file access action, the variable part contains the name of the file and the type of file access.

5.1.2. The profiles.

The profiles are the "memory" of the analysis component. The prototype learns about the behaviours of the users (also of objects and systems), it retains the "normal" activity and updates this knowledge when the behaviour changes slowly. In other words, the profiles contain the historical past activities of the analysed subjects so that normal or expected behaviours can be recognized. Appendix B gives some typical examples of profiles that are interesting to implement in secure systems.

5.1.2.1. The profile format.

A profile is a PROLOG fact of the form :

```
profile(variable_name,                      ]  1rst part

        event_type_pattern_list,            ]
        result,                             ]  2nd part
        user_pattern,                       ]
        information_pattern_list,           ]

        variable_type,                      ]
        threshold,                          ]  3th part
        value).                             ]
```

3 tsn is for task sequence number.

The profiles are composed of three parts :

a) the first part contains one element :

- **variable_name** that is the name of the profile.

b) the second part is composed of elements describing the event records that will be used to update the profile :

- **event_type_pattern_list** that is a list of patterns that have to match with the event field of the current analysed record. The elements of this list are logically combined with an OR operator. It means that at least one element of the list has to match the event field of the current record. Wildcards such as *, ? may be used.

- **result** that is the pattern that has to match the result field of the analysed record. Wildcards may be used.

- **user_pattern** that is the pattern that has to match the user field of the current record. This pattern may contain wildcards.

- **information_pattern_list** that is a list of patterns where all patterns mentioned have to match the corresponding fields in the variable part of the current record. The elements of this list are logically combined with an AND operator.

c) the third part contains the actual knowledge of the profile :

- **variable_type** that is the name of the type of the profile. The updating and the checking of the profile depend directly of this parameter. The variable type determines the type of measure and the model used.

- **threshold** that is a value or a list of values (depending on the variable_type) that parameterize the checking of the profile. For instance, for the operational model, the threshold parameter contains a threshold value (a fixed limit). In the mean and standard deviation model, the threshold parameter contains the value of a parameter d[4] used to determine the confidence interval.

- **value** that is a value or a list of values describing the profile. Particularly, using the operational model, the value parameter is the current value of the profile. Using the mean and standard deviation model, the value parameter is a list of values describing the history of a behaviour. The values of this list are used to compute all parameters needed (sum, sumsquare, mean, standard deviation, ...) to determine the normality of the analysed measure. The number of elements in the list may changed depending on the wanted length of the history.

As a first example, a profile for the user "user_1" concerning his failed logins may be created to check whether the number of failed logins for this user does not exceed a certain threshold on a daily base.

The profile is :

```
profile(failed_login,                % variable name
     [evt('JBE'), evt('JDE')],       % list of event patterns
     res('F'),                       % result pattern
     [userid('USER_1  ')],           % list of one user pattern
     [],                             % empty list of information pattern
     test_threshold,                 % type of treatment
     10,                             % threshold value
     7).                             % current value of the profile
```

The list of event patterns refers to the event field with a value of JBE (Job Batch Enable) or a value of JDE (Job Dialogue Enable) that correspond to a login event.

The result pattern refers to the result field with a value 'F' (failed result).

[4] See section 4.1.3.1 of chapter 4.

The list of user patterns contains only one user referring to the userid field of the record.

The list of information patterns is empty for this profile meaning that the variable part of the record is not interesting for this profile.

All records satisfying those four description parameters will be selected for the profile updating and checking.

Other examples of profiles are provided in appendix C.

5.1.2.2. The variable types.

The variable type parameter of a profile is that parameter that is the name of the abstract data type that defines a particular type of measure[5] and statistical model[6]. So, this variable determines the treatment (initialization, updating, checking) imposed to the profile. The implemented models are the operational model and the mean and standard deviation model. Other models like exponential model, multivariate model, markov process model, time series model may be used but are not yet implemented.

There are three implemented variable types at the moment (see appendix C) :

a) The **test-threshold** type uses the operational model.

   The profiles corresponding to this type counts the number of time a described event record pattern occurs. The pattern is described in the second part of the profile. Then the measured number of occurrence is compared with a fixed threshold value.

[5] See section 4.1.3.1 of chapter 4.
[6] See section 4.1.3.1 of chapter 4.

b) The **mean-of-time-for-session** type uses the mean and standard deviation model with continuous measure.

The profile corresponding to this variable type identifies the event record pattern described in the second part of the profile. In the implemented case, these records are login and logout events. Then the elapsed time between the two events is computed. The measured value is stored in the historical knowledge of the profile (list of values in the second parameter of the third part of the profile). Finally this measured period of time is analysed to verify its property inside a confidence interval computed on the base of the historical knowledge.

Note that not only the elapsed time of period between login and logout may be calculated, it is easy to imagine other events to be checked in the same way (open and close of a file, set-up of a communication via the network, ...).

c) The **mean-of-records** type uses the mean and standard deviation model with discrete measure (linear categorical).

The profile corresponding to this variable type finds all file access events (in our example, only read access are taken into account) that match with the second part of the profile. Each time a record is encountered, a variable is incremented. At the end of the session, the total number of those event records is stored in the historical knowledge of the profile. As in the previous variable type, this number is checked with a confidence interval.

In this case again, other event types may easily be checked. It is for examples the number of time a specific or a certain type of command is asked by the user, ...

5.1.3. The rules.

The treatments applied to the profiles will be decomposed to be performed. That is why rules are now described to identify the different processings applied to profiles.

First, the profiles are to be **initialized** each time it is required by the type of the profile (each day, each week, each time a specific event occurs, ...).

The audit file is analysed sequentially. When a record is evaluated, all profiles matching this record are **updated.** Then these profiles are **checked** to see whether they are normal. If not, an **action** is to be executed.

5.1.3.1. Initializing rules.

Initializing rules initializes the profiles depending on the type of the variable. A profile may be initialized when a certain type of audit record is analysed, or when a period of time is elapsed (hour, day, week, ...).

> The profiles of test-threshold type are initialized depending on the period of time the analysis is required. It is possible to evaluate event patterns in a period of one hour, or in one day. It is also possible to check the occurrence of the events during a user's session (identified by a login and a logout event). The initialization is to reset to 0 the parameter value that will contain at the end of the period the number of occurrences of the analysed events.
>
> The mean-of-time-for-session profiles must be initialized every time a new session begins. The initialization consists of storing the timestamp of the login record into a variable. This variable will be consulted at the end of the checked period (the end of the session) to compute the elapsed time of the session.

The mean-of-records variable types are to be initialized at each new user's session. This may also be extended to a period of time. A variable is reset to 0. This variable will contain at the end of the session the number of file accesses.

5.1.3.2. Updating rules.

Updating rules update the profiles. The value of a profile is updated depending on the type of the variable. The value may be updated depending whether an event occurs or whether a period of time is elapsed.

The test-threshold profiles are updated each time an event record matches with the second part of the profile. A variable is incremented by one at each occurrence.

The mean-of-time-for-session type profiles are only updated at the end of a session, when the logout time is known. The difference between logout and login time is computed and is inserted into the historical knowledge.

With the profiles of mean-of-records type, each time a file access (read access) or any other described event occurs, a variable is incremented. At the end of the analysed period, the computed value is inserted into the historical knowledge of the profile.

To update historical knowledge, the new measured value is added at the end of the list of value and the first element of the list is removed, acting as a sliding window as shown in figure 5.1 This methods allows to remove old values that do not correspond anymore to current behaviours.

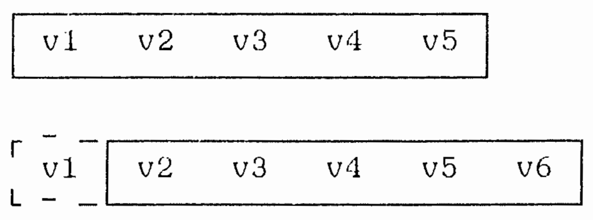

Figure 5.1 *Update of the historical knowledge as a sliding window.*

The first box of the figure 5.1 represents the knowledge before a new value is added. It is composed of 5 values v1 to v5. The second box of the figure 5.1 shows that the value v1 is removed out of the knowledge and the new value v6 is added at the end of the list.

5.1.3.3. Checking rules.

Checking rules test the profiles to determine whether the current behaviour is normal or not. The measured value is compared with the profile.

> The value of the test-threshold profile is checked each time a described event occurs. The profile value is compared with the threshold value. An action is taken if the value exceeds the threshold (upper threshold). In some case, it is possible to imagine a lower limit or the combination of the two thresholds.
>
> With the mean-of-time-for-session type of profile, the measured value (session period of time) is compared with a confidence interval at the end of the session. The interval is parameterized by the threshold value of the profile that contains the parameter d[7]. If the measured value is outside this interval, an action is to be performed.
>
> The mean-of-records profiles are checked in the same manner as for the mean-of-time-for session.

[7] See section 4.1.3.1 of chapter 4.

In the case of the checking of historical knowledge, several methods may be imagined to be used, depending on the system environment and/or on the profile it-self.

The first method is to update the historical knowledge before it is checked, even whether the measured value goes outside the confidence interval and does not correspond to the expected behaviour. The second possibility is to check the measured value, and only store it in the historic if it corresponds to the expected behaviour (if it falls inside the confidence interval). The first method would be used in environments where the user's behaviours may change suddenly. The modification is therefore notified and the profile is updated. The disadvantage is if the changed behaviour is due to a illegal user act on a legal identity, the profile of the legal user will be changed. The second method is the one used in the prototype. It only allows slow changes of behaviours.

5.1.3.4. Acting rules.

These rules are activated when an anomaly is detected by the checking rules. Their actions may be as described in section 4.3 :

. the printing of a message,

. the updating of a violation count,

. the activation of new rules (other profiles to check or in another manner, ...),

. modification of the selection criteria (pre-selection, ...),

. etc,

depending on the type of the detected anomaly.

5.2. Limitations of the prototype.

As a prototype, it is not intended to fulfil all functionalities required. Some limitations are to be imposed for a first implementation :

- All audit records are translated and analysed. There is no selection made by a translate_select component as described in section 4.1.1.3 of chapter 4.

- Only two models : operational model and mean and standard deviation model, are used to implement the three variable types (see section 5.1.2.2). Other models like the markov process model, the multivariate model, the time series model may be implemented.

- The detection of an anomaly involves a message to be displayed. No other action is performed as the update of a violation rate, the activation of new rules concerning the updating or the checking of the profiles, ...

- The analysing prototype is only a learning normal system behaviour component and is not coupled with other components like a statistical one or a behaviour pattern recognition one. It is possible to imagine a resolver analysing the output of the different analysing components - sharing the same input file but processing it in isolation - and producing a single output based on the analysis of the other components.

- No periodic anomaly rules are implemented. These rules are those fired when a period of time is elapsed. Only rules fired by the occurrence of events are implemented.

5.3. Experience learned and performance aspects.

By the realization of this prototype and by the study of the PDAT[8], an experience about the performance aspects can be learned. The prototype has been implemented in PROLOG language that is a fully interesting for its implementation speed capabilities in prototype realization. The PDAT is also a prototype initially realized in PROLOG and in which some modifications such as conversion of some PROLOG routines in C language, ... have been made to improve the quickness of the analysis. A speed ratio from 1 to 5 has been gained.

The audit trail generation takes from about 1%[9] to 5% of the total CPU time. Moreover about 5% of the CPU time are necessary again for the analysis of the generated audit trail depending strongly of the analysis methods used, the number of components, the number and content of the audit records (number of fields, micro or macro events, ...), and some other parameters.

These two complementary analysis components are not yet ready for ON-LINE analysis but works are seriously engaged in this way.

[8] See section 4.1.1.2 of chapter 4.
[9] The SAT of BS2000 (SIEMENS) needs the following rate of CPU time :
- when SAT is not installed : 0,02%
- when SAT is present but inactive : 0,03%
- when SAT is active : 1,21%

CHAPTER 6

FUTURE OF AUTOMATIC ANALYSIS TOOLS.

AATs are quite young (about 10 years old) and it is easy to imagine great improvements in the next 10 years.

6.1. Spreading and expansion of AATs.

One expected evolution of AATs can be that they should be used not only for security purposes, where the tool analyses the security audit trails and searches for intrusions, but they should be general and configurable enough to evaluate audit trails (not only security audit trails) and to analyse other fields of IT with adequate criteria. These fields should be :

- detection of bad functioning of some components in the system (devices, resources, ...);
- analysis of performance measures of critical components and resources of the system;
- ...

In an other hand, some evolutions in IT are waited such as :

- the evolution of the AI techniques,
- the increasing rapidity and storage capacity which computers work with,
- the less computational costs of expert systems,

- the possibility for human people to explain their wishes and interact with more "user-friendly" interfaces and languages (e.g., no algorithm, "natural" languages, ...),

- the new techniques developing learning machines as neural networks, ...

These should allow security responsible people to have means for detecting more anomalies in a computer system and making very difficult to common users to perpetrate ill-disposed actions.

The high computing speed and parallel computing may also be decisive in the development of AATs, making possible more simultaneity for analysis methods to evaluate the same audit data and thus reducing the risks of false alarms. Rapidity and few false alarms make more plausible automatic reactions by the tool itself in on-line systems.

6.2. Artificial intelligence evolution and its impact on AAT.

Artificial intelligence techniques are in full expansion and for example expert systems evaluating audit trails that have been introduced in chapter 4. Another new field of AI is neural networks. They are presented in the next section to show their utility in analysing security audit information.

6.2.1. AATs using neural networks.

Neural networks are a relatively new field on the vast domain of the artificial intelligence. Artificial neural systems (other name for neural networks) can be used to detect computer abuses by analysing audit trail files. For the reader who already has heard something about neural networks but who is not accustomed with their utilizations and the terms used in relation with this field, the following section defines and describes, without going into too much details, what are neural networks. The next section mentions some uses of neural networks in analysing security audit trails.

6.2.1.1. What are neural networks ?

Neural networks are one of the three forms taken by neurocomputing. The two other forms are : *neurocomputing hardware* (coprocessor boards), and *neural chips.* The term usually refers to software simulations of neurocomputing.

"A neural network is a massively parallel, information processing architecture composed of many simple processing elements interconnected to achieve certain collective computational capabilities" [Illingworth 89].

Neural networks are characterized as a network because they are composed of interconnected processing elements (or neurons) as well as the neuronal structure of the brain. The numerous processing elements of a neural network are interconnected by informal channels. Each neuron may have many input signals while only one output signal may be generated. The output signal of an element is the input signal of one or more other elements as shown by figure 6.1. Those input signals that are not output signals from another processing element are then input signals from the outside world. In that way the neurons are disposed in layers. The *input layer* receives data from the outside world. The *middle or hidden layers* are created by the neurocomputing software itself, and are where the "learning" takes place. "Learning is accomplished through trial and error, just as it might be with a human, using a feedback mechanism technique called *back propagation*" [Rochester 90]. This basic structure of neural networks makes them massively parallel computing devices.

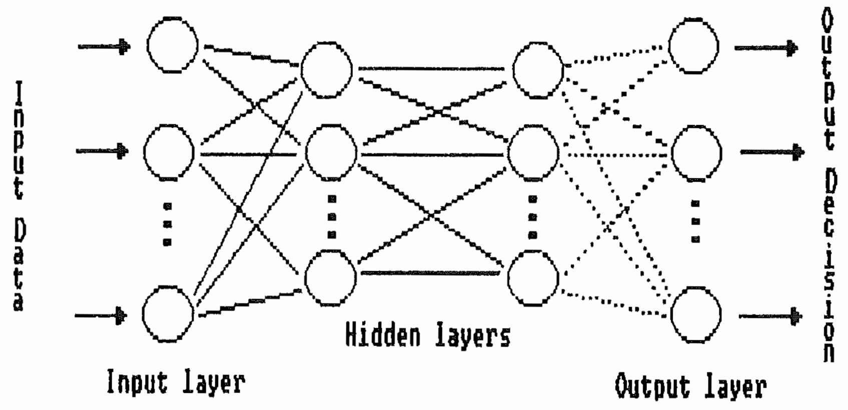

Figure 6.1 *Neural networks layers (from [Rochester 90]).*

If the sum of the inputs to a given processing element exceeds a predetermined threshold, the processing element "fires" and sends a signal to the other connected processing elements. Notice that the interconnections between processing elements are not "simple" connections. In fact, each input has a weighted value that determines the strength of the interconnection to the fire or not-fire decision of the following processing element.

The unique outstanding feature of a neural network is its ability to auto- or self-adapt as a dynamic system that can modify its own responses.

The many advantages of neural networks include : "

. Self-organization,

. No formal programming required,

. Ability to adapt and learn,

. Fault tolerance or graceful degradation[1],

[1] Graceful degradation : Information learned or stored in network memory (as in biological systems) can be very tolerant to failures in hardware. Some neural networks have lost as much as 20 percent of stored memory and still produced reasonable output results. [Illingworth 89]

. Pattern Recognition,

. Intuition, prediction, and statistical pattern reconstruction." [Illingworth 89]

The limitations of neural networks include : "

. Not good if precise answers are required,

. Cannot count (see forests, not trees),

. As yet cannot do things that conventional computers do effortlessly,

. Cannot justify answers,

. Offline learning or training is difficult and sometimes very tedious." [Illingworth 89]

6.2.1.2. Type of detected abuses.

Several uses of neural networks may be considered in analysing security audit trails. The first possibility is to consider the neural network just as any other analysis component (as are learning behaviour and pattern recognition methods). It is then coupled with the other existing components as shown in figure 6.2. As already explained in section 4.2, a resolver can analyse the different outputs.

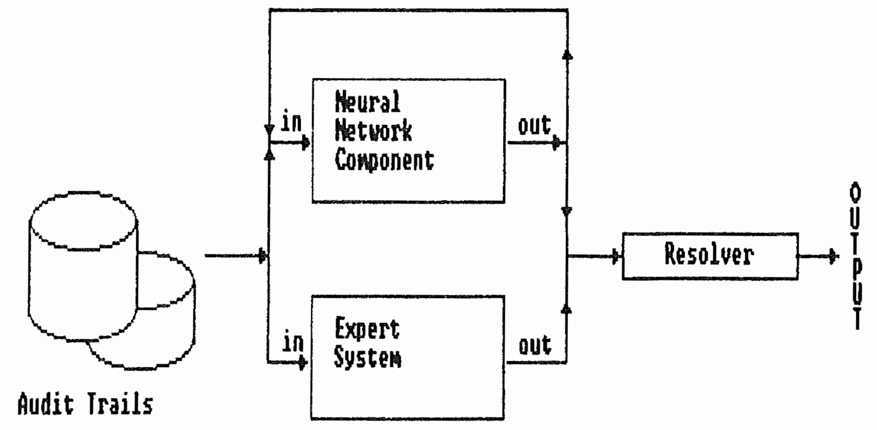

Figure 6.2 *A neural network as a analysis component.*

A second possibility can be the use of a neural network to learn and track the system-normal state, coupled with an expert system for in-depth intrusion analysis as figure 6.3 shows. The neural network is used as a real-time background monitor to adaptatively model the system and users normalcy. The purpose of the neural net is to learn the normal system activity and adapt to gradual changes. Rapid changes would trigger invocation of an expert system. A problem with expert system is that they are computationally expensive. With the architecture of the figure 6.3, the expert system can reside on the host computer and be invoked only when necessary.

The neural network would provide an efficient and elegant front-end status monitor which is also general enough to recognize unknown viruses and possible malicious user behaviour patterns.

This approach produces the following advantages :

- adaptative modelling of the users and the system,
- ability to deal with unknown viruses or intrusions,
- determination of when to use the more computationally expensive expert system.

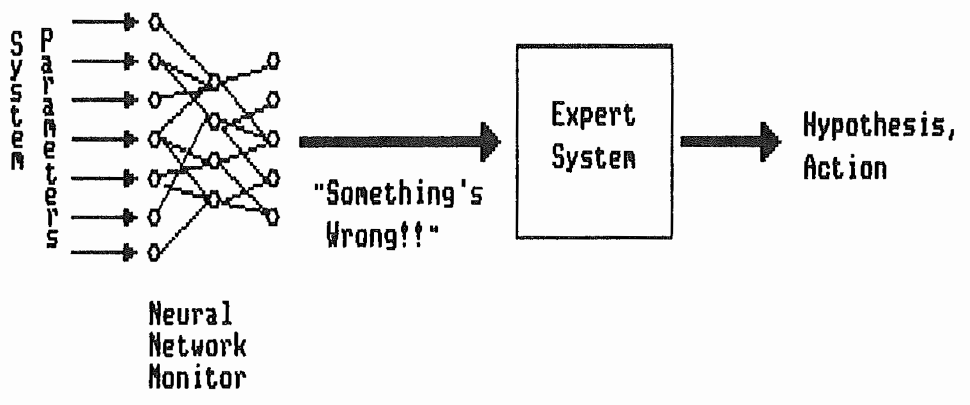

Figure 6.3 *A neural network as a background monitor (from [Fox 90]).*

Neural networks can be helpful in an intrusion-detection application by adaptively modeling the normal state for the users and the system, and take action when any abnormality is noted. For instance, the network would be trained by introducing samples of existing viruses to the system. Their patterns would be learned and associated with a human-prescribed antidote in each case. The next time the pattern appears in the system, the neural network monitor would trigger (or suggest) the defence.

The advantage of a neural network is that if a new virus appears which the computer has not been vaccinated against, the network should still be able to recognize it as suspicious activity and notify the operator. At the same time, it would be able to learn the new pattern for future use. [Fox 90]

6.2.2. Self-learning criteria.

As presented in chapter 4, AATs can learn patterns or behaviours to discover abnormal uses of a computer system. It is also possible to imagine that AATs should be able in the future to learn about research criteria. Self-learning criteria would provide a way for AATs to know what criteria are to be applied in such or such case. For instance, when an ill-disposed user is suspected or recognized to be a misfeasor (internal intruder[2]), it is not necessary to apply research criteria concerning external intruders. Knowing the elements on which criteria can be applied, the AAT would be able to combine or create the relevant questions needed to detect or confirm possible misuses.

[2] See chapter 1.

6.3. Evolution of the break-ins ?

It must be also taken into account that if all these evolutions are beneficial for intrusion-detections, they are also profitable for the criminality. The evolution of the technology is also a good weapon for misfeasors. The actual extent of computer crime is considered by most experts as the fastest growing illegal activities. And this, although misuses of computers are difficult to measure.

The incidence of computer crime will increase because of the increasing number of computers and the automation of business activities. The size of losses in significant cases will increase because of the concentration of information assets in fragile forms subject to manipulation via computer.

The principal reason for both the growth and lack of accurate measurement of computer crime is the difficulty in detecting a well-executed misuse. That is why AATs have future in IT. They contribute and help to avoid the danger that misuses are not detected. Moreover, when an attack is discovered, they contribute to search who?, what?, when?, how? making possible the find of ways to counter the violation.

The always-growing and changing imagination of misusers makes pertinent the use of **adaptable** (learning) and even **anticipating** AATs to reduce the danger that crimes could be perpetrated without being known.

IN CONCLUSION ...

Security in information technology can be compared with home security. If the entrance of a house has no door, every body, friends as well as enemies, can easily enter. To mark physically the limits of the house, a door can close the home. But this is not sufficient, an intruder may easily enter by pushing the door open. Then a lock can be used to allow access to only people being in possession of the right key. But they still probably exist some people who are able to force the lock and so to break open the door. At this point there is a significant step : the access is limited to few people. More secure locks can therefore prevent for any amateurish criminal. Alarm systems and locking of the windows would strengthen the home security, making more difficult and more expensive the penetration of the house (as well as for the inhabitants as disadvantage).

With security, we can always go one step forward in the protection of an environment. But we have to cope with the imagination of the intruders who can always by-pass this "last" step. The advantage of this perseverance is the reduction of the number of misfeasors at each step due to the expansiveness of the intrusion.

As for the entrance of a house, trusted system criteria evaluation books search and propose solutions of security measures to limit, and in some cases to avoid, computer system misuses to be perpetrated. These measures have to be evaluated and taken depending on the environment the system is dedicated.

Security audit trail generations and analysis are only **ONE** of the numerous solutions that contribute to make systems safer and more secure for their users. Security audit trail, as any other method, is certainly not **THE** solution, and the above example illustrates well two important principles of security that are particularly true for audit trail. First, it confirms that a complete security is not possible. What is feasible is to postpone the means used by criminals so that the access is more and more difficult until it is more expensive to break the environment than collecting the loot. But at this point, the game is not yet won : some "players" may still be interested only by the fact that an intrusion is accomplished than by what it could be profitable.

The second principle is that, as shown by the more and more complicated means employed to make the door secure, some disadvantages appears for the individual who has to enter (into the house or into the computer system). More a system is secure more there exist inconveniences for "normal" or "legitimate" users.

The major disadvantage of audit trail for the legitimate users is *privacy*. In U.S.A., keystroke dynamics are forbidden to protect typists against employers who would use this mechanisms to control the speed at which they work. The same problem is encountered with audit trail. An employer may analyse the logged information to know the performances of his employees. For instance it is possible with this type of data to know how much times a compiler is used by a user, or how many errors he has made, etc.

As already said security audit trail generation and analysis is certainly not **THE** solution to security problems. Nevertheless, this type of security mechanism has a great advantage that is its flexibility to counter new attacks. While the imagination of intruders is endless to by-pass the security mechanisms, AATs allow the auditor to adapt his analysis criteria to counter the new attack immediately after this one is discovered. This method is comparable to cryptography where once a key is discovered, the same mechanism can still be employed just by changing the secret key. For audit trail, the same means are always valid just by adapting analysis criteria.

By using technical safeguards like cryptography, advanced management controls, codes of conduct, audit trail, ... stimulated in part by strong criminal statutes, we can continue to limit the risks inherent in the use of computer technology to an acceptable level.

REFERENCES.

[AIX 88] M. S. Hecht, A. Johri, R. Aditham, T. J. Wei, "Experience Adding C2 Security Features to UNIX", IBM Systems Integration Division, 1988.

[Bharath 89] Bharath R., "Prolog : Sophisticated Applications in Artificial Intelligence", Windcrest Books, 1989.

[Bratko 86] Bratko I., "Prolog, Programming for Artificial Intelligence", Addison-Wesley Publishing Company, 1986.

[Brown 87] R. Leonard Brown, "Guidelines for Audit Log Mechanisms in Secure Computer Systems", The Aerospace Corporation, May 1987.

[DoD 83] Departement of Defence (DoD) Computer Security Center, "Trusted Computer System Evaluation Criteria" (Orange Book), DoD CSC-STD-001-83, December 1983. (Notice that a more recent version of 1985 exists).

[Fox 90] K.L. Fox, Henning, Simonian, Reed, "A Neural Network Approach Towards Intrusion Detection", Government Information Systems Division, Melbourne, July 2, 1990.

[IBM 89] US Marketing & Services. "Multi-Level Security for IBM MVS/Enterprise System Architecture Overview", IBM Programming Announcement, October 24, 1989

[IDES 87/1] D.E. Denning, "An Intrusion-Detection Model", IEEE Transaction on Software Engineering, Vol. SE-13 No. 2, February 1987.

[IDES 87/2] Denning, Edwards, Jagannathan, Teresa, Lunt, Peter, Neumann, "A prototype IDES : A Real-Time Intrusion-Detection Expert System", SRI Project ECU 7508, Menlo Park, August 1987.

[IDES 88] T.F. Lunt and R. Jagannathan. A prototype real-time intrusion-detection system. In Proceedings of the 1988 IEEE Symposium on Security and Privacy, April 1988.

[IDES 90/1] T.F. Lunt, Ann Tamaru, Fred Gilham, ... "A Real-Time Intrusion-Detection Expert System (IDES)", SRI International, SRI project 6784, May 1990.

[IDES 90/2] Teresa, Tanam, Gilham, Jagannathan, Neumann, Jalali, "IDES : A Progress Report", Proceedings of the Sixth Annual Computer Security Applications Conference, Tucson, Arizona, December 1990.

[Illingworth 89] William T. Illingworth, "Beginners Guide to Neural Networks", IEEE AES Magazine, September 1989.

[ITSC 89] ZSI - Zentralstelle für Sicherheit in der Informatinstechnik, "Criteria for the Evaluation of Trustworthiness of Information Technology Systems", ISBN 3-88784-200-6, GISA - German Information Security Agency, Köln, 1989.

[ITSEC 90] "Information Technology Security Evaluation Criteria", Der Bundesminister des Innern, Bonn, Mai 1990.

[Hsio 79] David K. Hsiao, Douglas S., Kerr, "Computer Security". ACADEMIC PRESS New York San Francisco London 1979.

[Lane 85] V.P. Lane, "Security of Computer Based Information Systems", Macmillan Education Ltd, 1985.

[Lunt 88] T.F. Lunt, "Automated Audit Trail Analysis and Intrusion Detection : a survey", Proceedings of the 11th National Computer Security Conference, Baltimore, MD, October 1988.

[Martin 88] J. Martin, S. Oxman, "Building Expert Systems : A Tutorial", Prentice Hall Englewood Cliffs, 1988.

[MIDAS 88] MM. Sebring, Shellhouse, Whitehurst, "Expert System in Intrusion Detection : A case study", Proceedings of the 11th National Computer Security Conference, Baltimore, MD, October 1988.

[MITRE 87] Picciotto J., "The Design of an Effective Auditing Subsystem," Proceeding of the 1987 IEEE Symposium on Security and Privacy, Oakland, California, pp. 13-22 (April 1987)

[NCSC 87/1] "Guidlines for Auditlog Mechanisms in Secure Computer Systems", The Aerospace Corporation, May 1987.

[NCSC 87/2] "A Guide to Understanding AUDIT in Trusted Systems", National Computer Security Center, July 1987.

[NTCSEC 85] NATO Security Committee, "NATO Trusted Computer System Evaluation Criteria", NATO AC/35-WP/145, September 1985.

[Parker 76] Donn B. Parker, "Crime by Computer", Ed. Charles Scriber's sons, New York, 1976.

[Parker 84] Donn B. Parker and Susan H. Nycum, "Computer Crime", Communications of the ACM, volume 27 number 4, pp. 313-315, april 1984.

[Parsaye 88] Parsaye K., Chignell M., "Expert Systems for experts", John Wiley & Sons, Inc. 1988.

[PDAT 88/1] A. Baur, W. Weiss, "Proposal for an Audit Analysis Tool (AAT)", ZT ZTI SOF 42 Mch P, Sept. 7th, 1988.

[PDAT 88/2] A. Baur, W. Weiss, "Audit Analysis Tool for systems with high demands regarding security and access control", ZFE* F2 SOF 42 Mch P, Nov. 30th 1988.

[PDAT 89/1] A. Baur, W. Weiss, "Specification of a Demoversion of the Audit Analysis Tool", ZFE F2 SOF 42 Mch P, 14.4.89.

[PDAT 89/2] A. Baur, W. Weiss, "Analysis of protocol data using AI techniques", ZFE F2 SOF 42 Mch P, 7.11.89.

[PDAT 89/3] A. Baur, W. Weiss, "Analyse of Audit Protocol Data using Methods from Artificial Intelligence", ZFE F2 SOF 42 Mch P, 1989.

[Pritvhard 79] J.A.T. Pritchard, "Security in On-Line Systems", The National Computing Centre Limited, 1979.

[RACF 87] "RACF General Information Manual", International Business Machines Corporation, May 1987.

[Rochester 90] Jack B. Rochester, "New Business uses for neurocomputing", I/S Analyser, vol. 28, No 2, February 1990.

[Rolston 88] Rolston D.W., "Principles of Artificial Intelligence and Expert Systems Development", McGraw - Hill, Inc., 1988.

[Ross 89] Ross P., "Advanced Prolog, Techniques and Examples", Addison-Wesley Publishing Company, 1989.

[SAT 90/1] SAT Team (SWN46), "Security Audit Trail (SAT) - EIS", Siemens Nixdorf Software S.A., Namur, 13.11.90.

[SAT 90/2] "Developers Handbook V10.0", Siemens-Datenverarbeitung, München, December 12th, 1990.

[Saunders 85] Saunders Michael, "Protecting Your Buisiness Secret", Gower Publishing Company Limited, 1985.

[Shea 86] Terence J. Shea, "NATO Trusted Computer System Evaluation Criteria", Nato Security Committee, September 1986.

[SMF 87] H.J. Van Aalderen, "The internal auditors' use of SMF", Conference Proceeding of 2nd European Conference on Computer Audit, Control & Security, November 1987.

[SMP 88] "C2 Trusted Facility Manual Version 1 - Security Module Package", Secureware Inc., August 12, 1988.

[Tassel 72] Dennis van Tassel. "Computer Security Management", Prentice-Hall, Englewood Cliffs New Jersey, 1972.

[TIM 90] Teng, Chen, Lu, "Adaptative Real-Time Anomaly Detection Using Inductively Generated Sequential Patterns", Applied Intelligent Systems Group Digital Equipment Corporation Marlboro, 1990.

[UNIX 88] Tony Heywood, "Secure UNIX: Implications for Application Software and End-users", Uniplex Ltd., 1988.

[VAX/VMS 85] DEC, "Guide to VAX/VMS System Security", Digital Equipment Corporation, Maynard, Massachusetts, July 1985.

[VAX/VMM 90] Kenneth F. Seiden, Jeffrey P. Melanson, "The Auditing Facility for a VMM Security Kernel", Digital Equipment Corporation, IEEE, 1990.

[Walsh 83] Stuart Walsh, "Software security", Policy, volume 25 no 3, pp. 9-10, april 1983.

[Walker 77] Bruce J. Walker, Ian F. Blake, "Computer Security and Protection Structures", Dowden, Hutchinson and Ross Stroudsburg (Pa.), 1977.

APPENDIX.

Appendix A : Advanced behavioural post-selection criteria.

A.1. More elaborated behaviour patterns.

Specific command sequences or behaviour patterns can be found interconnecting records together and possibly using variables to compare the fields content of the records.

Examples of elaborated behaviour patterns could be :

. One suspicious command immediately followed by a STATUS[1] command will enforced the suspicion.

. Detection of users logged on several physical terminals. Although it could be consider as normal for a user to be connected to several logical terminals. If this is not necessary an attempt of intrusion, it could be dangerous to leave more than one connected terminal without observation. Two variables are to be defined: one getting the value of the userid and another one getting the information about the terminal. The description of the behaviour is : find at least 2 records where the userid fields have the same content and where the terminal fields are not equal.

. More than three consecutive unsuccessful login (failed result) attempts for the same user-id within five minutes is considered as a penetration attempt.

[1] In BS2000, the STATUS command gives some informations about the current state of the system (logged users, current command, consumed CPU time, ...) and allows to follow the execution of a command, program, ...

. The detection of sequences of repetitive actions can help in the detection of an anomaly (loop in a procedure, ...) or an intrusion attempt (procedure that tries series of passwords on a userid or that tries one possible password on several userids, ...).

. Two or more login attempts within half an hour from geographically separated areas using the same user-id is suspicious.

. The detection of two attempts by the same user in such a short period of time that is doubtful that they could have been generated by a human at a conventional computer keyboard. This rule is designed to detect the type of attack in which a penetrator uses a computer to repeatedly generate userid / password pairs.

. Well-known intrusion scenarios can be considered as sequences of commands. For examples and practical scenarios, more knowledge and experience are needed. The best way is to talk with people involved with intrusions (people who test the security of the system, security administrators, ...)

A.2. Other behaviours.

Behaviours can also be determined for files or programs, not only for users, to detect, for instance, viruses or unusual object uses.

- the starting of a database is done every day at 6.30 am. If it starts once at 9.00 pm, illegal accesses will certainly follow.

Appendix B : Post-selection criteria for advanced statistical analysis.

Some profiles are now described for users, system, objects and subject-object relation. These profiles may be used to detect changes in the behaviours. These profiles have been selected for a BS2000 operating system but can easily be used to any other system.

B.1. Profiles for users.

- **time of login** : records the time in the day at which a user logs in, using the mean and standard deviation model.
  Each 24-hours period is divided into 3 categories : day period, night period, and graveyard period. Each user's login is classified into one of these three categories. Since a user's login behaviour may vary considerably during a work week, login occurrences may be represented by an array of event counters parameterized by day of week and time of day.

- **last login** : is an interval timer measuring time since last login, using the operational model. It would paticularly be useful for detecting a break-in on a "dead" account.

- **physical location** : records the number of times a user is connected on the system from different locations (terminal, remote host,...), using the mean and standard deviation model.

- **connect time** : measures the length of a user session on the system. Each session is referred as a task and has a unique task number. Several sessions for the same user can be active at the same time. This profile uses the mean and standard deviation model.

- **CPU time** : records the amount of processing time consumed in a user session on the system, using the mean and standard deviation model.

- **number of userids read** : records how many userids (directories) have been read by the user, using the mean and standard deviation model.

- **number of userids modified** : records how many other userids (directories) have been modified by a user. This profile uses the mean and standard deviation model.

- **number of files created/deleted/read or modified** : records how many files have been created, deleted, read, or modified by a user, using the mean and standard deviation model.

- **I/O activity** : indicates the amount of input and output activity in a user session (disk access, ...), using the mean and standard deviation model.

- **protection violation** : number of file access protection violations in a user session, using the operational model.

- **different commands used** : records how many different commands were invoked by the user. This can indicate whether a user always uses the same commands or uses a large amount of various of them. This profile uses the mean and standard deviation model.

- **system errors** : records the number of system-related errors invoked by the user, such as permission denied, disk usage overflow, ..., using the operational model.

- **system errors by type** : records the number of times each type of error occurred, using the operational model.

- **level of audit record activity** : indicates the amount of activity that occurred for the user. The value recorded is the number of audit records received for a particular user. This profile uses the mean and standard deviation model.

- **hourly audit record activity** : records the number of audit records received for each hours, categorized by the hours of the day, using the mean and standard deviation model.

- **day of use** : records how many audit records are produced by the user on a daily basis, using the mean and standard deviation model.

- **use of printers** : records the number of times a user prints out on a certain printer. If a user directs his printer output to some location other than where he normally sends output, he may be attempting to leak sensitive data. This profile uses the mean and standard deviation model.

- **use of networking functions**, ...

B.2. Profiles for system.

This type of profiles can determine whether a session has no anomaly in comparison with prior sessions.

- **number of users** : records the average number of users for a session, using the mean and standard deviation model.

- **login failures** : an inordinately large number of login failures system-wide might be indicative of an attempt to break the system. Records the number of bad login attempts made on the system, and encompasses both local and remote attempts into the system. This profile uses operational model.

- **system activity** : records the number of audit records received by the log mechanism. It is the accumulation of all audit records produced by all subjects on the system (note that it depends on the pre-selection criteria). This profile uses the mean and standard deviation model.

- **CPU usage** : records the amount of CPU time used on the system. It is the accumulation of CPU time used by all subjects on the system. This profile uses the mean and standard deviation model.

- **I/O usage** : records the amount of I/O used on the system. It is the accumulation of I/O calls used by all subjects on the system, using the mean and standard deviation model.

- **hourly bad login attempts** : records the number of bad login attempts made on the system during each hours. This measure is intended to track that times during the day that bad login attempts are usually common/uncommon. This profile uses the mean and standard deviation model.

- **system errors** : records the number of errors occurring on the system. It is the accumulation of all system errors invoked on the system. This profile uses the operational model.

- **system errors by type** : records the number of errors of different types made on the system, categorized by error types, using the operational model.

- **hourly system errors** : records the number of errors that occurred on the system during each hours, categorized by the hours of the day. It is intended to track the times during the day that system errors are likely to occur. This profile uses the operational model.

- **network activity** : records the number of audit records related to network activity produced on the system, using the mean and standard deviation model. Other measures can be applied concerning network activities.

- **origin of connection** : records for each origin of connection the number of times it logs in, using the mean and standard deviation model. This could determine unusual terminal, remote connection, ...

B.3. Profiles for objects.

a) **programs** :

. **typical times of days used** : some programs may only be used in the morning (starting of a database, ...) or during the night (batch programs and procedures). This profile uses the mean and standard deviation model.

. **execution frequency** : measures the number of times a program is executed during some time of period, using the mean and standard deviation.

. **typical data volume and rate** or any other resource used by the program, using the mean and standard deviation model.

. **typical duration** (CPU use) of the average time of a compilation or execution of a program can be stored. For instance, a normal interval of time for a compiler is between 0 and 1 second for failures and between 5 and 20 seconds in case of successful compiling. More could be a sign of presence of virus, ... This profile uses the mean and standard deviation model.

. **execution denied** : measure the number of attempts to execute a certain program, using the operational model.

. **program resource exhaustion** : count the number of times a program terminates in an abnormal manner during a day because of inadequate resources or because 'break' command invoked by a user. This profile uses the operational model.

. **the files used** : for example, a C-compiler only read from *.c and *.lib files, and write to *.obj files. Other read or write accesses could be a sign of virus. This profile uses the mean and standard deviation model.

. **seasonal or periodic use** : some program may only have a meaning if started after a period of time (program calculating the salaries), ... This profile uses the mean and standard deviation model.

b) **files** : Some profiles used for programs may also be used for files or other objects.

. **typical time of days used.**

. **access frequency.**

. **access denied.**

. **seasonal or periodic use**, ...

c) commands :

. **frequency of a command in a session** : ratio of one command versus another, ratio of one command versus total activity, using the mean and standard deviation model.

. **time between actions** : last use of a command, last action of any type, using the mean and standard deviation model, ...

B.4. Profiles for subject-object relations.

It is not possible to determine a profile for each action that each user can perform on each object. Thus classes of subjects can be determined as well as classes of objects. In the above criteria, when a user or an object is referred, it could also be a group of users or objects.

A possibility is to create profiles for subject-object only for privileged users or restricted objects in that way that if another user than the expected one access to a restricted object, his behaviour will probably be different and detected.

- **directories read/modify/create/delete frequency** : indicates the frequency that a directory is read/modified/created/deleted by a class of users, using the mean and standard deviation model.

- **file usage** : records the number of times a user or a group of users use a file or class of files within the system during a day (or some other period), using the mean and standard deviation.

- **read/write/create/delete frequency** for a file or group of files, for a user or a group of users, using the mean and standard deviation model.

- **program usage** : indicates the number of times each program was used by the user, using the mean and standard deviation model.

- it could be also interesting to know the **average time of use** for each of those programs, using the mean and standard deviation model.

- **average use** : indicates the user's average time of use for a program, a database, ..., using the mean and standard deviation model.

- **read/write/delete/create fails** : event counters that measure the number of access violation per day, using the operational model.

- **execution frequency** of programs using the mean and standard deviation model.

- etc.

This approach can be used for users, objects, systems that have homogeneous behaviours but can not be applied (or not easily) to non-homogeneous behaviours. A user that logs every day from 9.00 am to 17.00 pm, at the same location, has an homogeneous login behaviour. It is easy to detect an anomaly. But some users may log on several locations at any time of the day. It is therefore more difficult to describe such behaviours with profiles.

Appendix C : Examples of implemented profiles.

```
% The structure of a profile is
%            name of the variable,
%            list of event type patterns,
%                 ! the elements are connected with
%                   logical OR
%            result,
%            user name pattern,
%            list of information patterns,
%                 ! the elements are connected with
%                   logical AND
%            type of the variable,
%            threshold,
%            value.
%

%
% measures password failures for the entire system using
% the operational model
%
profile(failed_password,
     [evt('UCK')],
     res('F'),
     userid('*'),
     [],
     test_threshold,
     10,
     0).

%
% measures the number of failed login for the userid TSOS
% using the operational model
%
profile(failed_logon_and_password_for_TSOS,
     [evt('JDE'),evt('JBE'),evt('UCK')],
     res('F'),
     userid('TSOS    '),
     [],
     test_threshold,
     8,
     0).
```

```
%
% measures password failures for the userid TSOS using
% the operational model
%
profile(failed_password_for_TSOS,
     [evt('UCK')],
     res('F'),
     userid('TSOS    '),
     [],
     test_threshold,
     3,
     0).

%
% measures the number of failed access to any file ...
% using the operational model
%
profile(access_denied_for_file,
     [evt('FMD'), evt('FRD')],
     res('F'),
     userid('*'),
     [],
     test_threshold,
     4,
     0).

%
% measures how many logons are generated for the entire
% system
%
profile(how_many_logons,
     [evt('JDE'),evt('JBE')],
     res('*'),
     userid('*'),
     [],
     test_threshold,
     20,
     0).

%
% measures the number of audit records generated by SAT
%
profile(how_many_audit_records,
     [evt('*')],
     res('*'),
     userid('*'),
     [],
     test_threshold,
     300000,
     0).
```

```
%
% measures the number of events generated for the userid
% TSOS with a failed result.
%
profile(all_failed_events_for_TSOS,
     [evt('*')],
     res('F'),
     userid('TSOS    '),
     [],
     test_threshold,
     30,
     0).

%
% measures the length of a session for the userid TSOS
% using the mean and standard deviation model. The
% threshold value contains the parameter d. The value
% contains the history concerning the user past
% behaviour.
%
profile(length_of_session_for_TSOS,
     [evt('JDE'),evt('JED')],
     res('S'),
     userid('TSOS    '),
     [],
     mean_of_time_for_session,
     1.0,
     [001, 010, 018, 012, 003, 008, 015, 005, 052, 023]).
     % list of HHMM where HH is hours and MM minutes.

%
% measures how many records are read by the userid RIM in
% a session, using the mean and standard deviation model.
% The threshold value contains the parameter d. The value
% contains the history concerning the user past
% behaviour. This profile is useful to detect attempts to
% obtain sensitive data by inference and agregation
% (e.g., by obtaining vast amounts of related data).
%
profile(record_read,
     [evt('FRD'), evt('LRE'), evt('JED')],
     res('S'),
     userid('RIM     '),
     [],
     mean_of_read_records,
     1.0,
     [15, 10, 1, 16, 12, 23, 19, 30, 14, 10]).
```

```
%
% measures how many records are read by the userid TSOS
% in a session, using the mean and standard deviation
% model. The threshold value contains the parameter d.
% The value contains the history concerning the user past
% behaviour. This profile is useful to detect attempts to
% obtain sensitive data by inference and agregation
% (e.g., by obtaining vast amounts of related data).
%
profile(record_read,
     [evt('FRD'), evt('LRE'), evt('JED')],
     res('S'),
     userid('TSOS    '),
     [],
     mean_of_read_records,
     1.0,
     [15, 10, 7, 8, 12, 13, 19, 12, 14, 10]).
```

GLOSSARY.

Audit Trail : A set of records that collectively provide documentary evidence of processing used to aid in tracing from original transactions forward to related records and reports, and/or backwards from records and reports to their component source transactions. [NTCSEC 86]

Channel : An information transfer path within a system. May also refer to the mechanism by which the path is effected. [NTCSEC 86]

Collection File : A file collecting the security relevant events generated to make an audit trail. The security relevant events are selected based on pre-selection filters.

Covert Channel : A communication channel that allows a process to transfer information in a manner that violates the system's security policy. [NCSC 87/1]

Covert Storage Channel : A covert channel that involves the direct or indirect writing of a storage location by one process and the direct or indirect reading of the storage location by another process. Covert storage Channels typically involve a finite resource (e.g., sectors on a disk) that is shared by two subjects at different security levels. [NCSC 87/1]

Covert Timing Channel : A covert channel in which one process signals information to another by modulating its own use of system resources (e.g., CPU time) in such a way that this manipulation affects the real response time observed by the second process. [NCSC 87/1]

Discretionary Access Control : A means of restricting access to objects based on the identity of subjects and/or groups to which they belong. The controls are discretionary in the sense that a subject with a certain access permission is capable of passing that permission (perhaps indirectly) on to any other subject.

Domain : The set of objects that a subject has the ability to access.

Event : Operation of a subject to an object with a certain result.

Formal Security Policy Model : A mathematically precise statement of a security policy. To be adequately precise, such a model must represent the initial state of a system, the way in which the system progresses from one state to another, and a definition of a "secure" state of the system. To be acceptable as a basis for a TCB, the model must be supported by formal proof that if the initial state of the system satisfies the definition of a "secure" state and if all assumptions required by the model hold, then all future states of the system will be secure.

Mandatory Access Control : A means of restricting access to objects based on the sensitivity (as represented by label) of the information contained in the objects and the formal authorization (i.e., clearance) of subjects to access information of such sensitivity.

Multi Level Device : A device that is used in a manner that permits it to simultaneously process data of two or more security levels without risk of compromise. [NTCSEC 86]

NATO Trusted Computer System Evaluation Criteria : These criteria are the equivalent ones of the Department of Defence (DoD) approved by the NATO.

Object : A passive entity which contains or receives information.

Pre-Selection : Selection, by authorized personnel, of the auditable events that are to be recorded on the audit trail.

Post-Selection : Selection, by authorized personnel, of specified events that have been recorded on the audit trail.

Reference Monitor Concept : An access control concept that refers to an abstract machine that mediates all accesses to objects by subjects.

Reduction File : File created by application of post-selection criteria on collection files to reduce the amount of events to be analysed.

Security Level : The combination of a hierarchical classification and a set of non-hierarchical categories that represents the sensitivity of information. [NCSC 87/1]

Security Policy : A number of requirements and rules which specify how security relevant information is to be handled and processed.

Security Relevant Event : An event which can cause violation of the security policy. [NCSC 87/1]

Security Target : A specification of the security required of a Target of Evaluation (TOE), used as a baseline for evaluation. The security target will specify functions of the TOE. It may also specify the security objectives, the threats to those objectives and the particular security mechanisms that will be employed.

Sensitive Information : Information that must be protected because its unauthorized disclosure, alteration, loss, or destruction will at least cause perceivable damage to someone or something.

Sensitivity Label : A piece of information that represents the security level of an object and that describes the sensitivity (e.g., classification) of the data in the object. Sensitivity labels are used by the TCB as the basis for mandatory access control decisions. [NTCSEC 86]

Single Level Device : A device that is used to process data of a single security level at any one time. [NTCSEC 86]

Subject : An active entity, generally in the form of a person, process or device that causes information to flow among objects or changes the system state.

Subject Security Level : A subject's security level is equal to the security level of the objects to which it has both read and write access. A subject's security level must always be less than or equal to the clearance of the user the subject is associated with. [NTCSEC 86]

Target of Evaluation : An IT system or product which is subject to security evaluation.

Trusted Computing Base : The totality of protection mechanisms within a computer system - including hardware, firmware, and software - which, in combination, enforce the security policy.

ACRONYMS.

AAT	:	Automatic Analysis Tool.
DoD	:	Department of Defense.
ES	:	Expert System.
IDES	:	Intrusion-Detection Expert System.
IT	:	Information Technology.
MIDAS	:	Multics Intrusion Detection and Alerting System.
NCSC	:	National Computer Security Centre.
NTCSEC	:	NATO Trusted Computer System Evaluation Criteria.
OB	:	Orange Book [DoD 83].
PDAT	:	Protocol Data Analysis Tool.
RACF	:	Resource Access Control Facility.
SAT	:	Security Audit Trail.
SMP	:	Security Module Package.
TCB	:	Trusted Computing Base.
TIM	:	Time-based Inductive Machine.
TOE	:	Target of Evaluation.

INDEX.