PROF PAUL SPRIRAKIS
UNIVERSITY OF PATRAS
DEPT. OF COMP. ENG. & INFORM
GREECE

SECURENET: A NETWORK-ORIENTED INTELLIGENT INTRUSION PREVENTION AND DETECTION SYSTEM *E2*

SECURENET: A network-oriented intelligent intrusion prevention and detection system

Paul Spirakis [1], Sokratis Katsikas [2,3], Dimitris Gritzalis [2,3],
Francois Allegre [4], Dimitris Androutsopoulos [5,6], John Darzentas [2], Claude Gigante [7],
Dimitris Karagiannis [8], Heikki Putkonen [9], Thomas Spyrou [2]

[1] University of Patras
Department of Computer Engineering & Informatics
Greece

[2] University of the Aegean
Department of Mathematics
Greece

[3] Technological Educational Institute of Athens
Department of Informatics
Greece

[4] FRANCE TELECOM
Centre Nationale d' Etudes des Telecommunications (CNET)
France

[5] EXPERTNET Ltd.
Computer Systems Security Research Group
Greece

[6] Lappeenranta University of Technology
Department of Information Technology
Finland

[7] DASSAULT Automatismes et Telecommunications
Security Department
France

[8] University of Vienna
Dept. of Knowledge Based Systems
Austria

[9] University of Oulu
Department of Information Processing Sciences
Finland

Contact person: Assoc. Prof. Dimitris Gritzalis, 1 Sevastias Str., Athens GR-12242, Greece.

This work was supported in part by the Commission of the European Communities, Directorate General XIII, RACE Programme under the SECURENET R2057 Contract.

SECURENET: A network-oriented intelligent intrusion prevention and detection system

Abstract: This paper provides detailed technical information on the development plan of SECURENET. SECURENET is an Intelligent Intrusion Detection System designed to operate in Integrated Broadband Communications (IBC) network environments. SECURENET employs advanced and previously relatively unexplored, in such contexts, technologies such as Neural Nets and Intent Specification Languages, in conjunction with Expert Systems to perform its tasks. The description of the system provided herein is based on both a technology and a functionality point of view.

I. Introduction

The development of the IBC framework on a European basis has brought several security issues to a critical point. One of the most important issues is the protection of an open computer network against various forms of malicious attacks. The SECURENET project aims at developing an integrated system capable of effectively defending open computer networks -in an IBC environment- against attacks perpetrated against the network or its hosts. In particular, the project aims at providing means for both preventing and detecting malicious software intrusions in a distributed environment.

Moreover, the SECURENET project aims at providing the appropriate means for guaranteeing communication capabilities of some level after a major intrusion, or whenever such a communication capability is considered to be of high importance. These capabilities are essential in order for vital messages on the proper countermeasures to be used against the intrusion to be able to reach all communicating parties in minimum time and certainly before a major damage has been inflicted.

Computer viruses, worms, tojan horses and logic bombs pose the most severe intrusion threat [1-2] against a computerised environment, especially if this environment is distributed and the ability to enforce physical access control is very limited. These structures are also collectively referred to as malicious software [3-4]. Their formal description can be given either through a computational approach [5-6] or through an abstract formalism [7]. Recent research results [8-9] provide additional formal approaches to these structures, but have not yet been sufficiently evaluated and tested.

Several traditional techniques, based on detection-by-appearance models and often referred to as *solid detection techniques*, have been proposed to prevent against malicious software attacks. Vaccines, matching patterns, simple checksums, software self-defence or fault-tolerance, change control and integrity shells - to name some- have been found to be efficient in only a few cases [4,10,11]. Only cryptographic checksums appear to be more promising, in the sense that they could be efficient in a larger number of cases [12]. This is primarily because cryptographic checksums can handle both known and unknown viruses, operate at medium speeds, have a low cost, are able to prevent as well as to detect attacks, and produce no false positives. The applicability of these techniques is further limited in the case of a distributed computation environment [9,13].

Alternatively, *sound detection techniques*, which limit the transitivity, the sharing, or the functionality of information flow or of the system objects, may be employed; however, such approaches have been demonstrated to be inapplicable in most real systems [4,6]. For these reasons, the employment of detection-by-behaviour models, utilising Artificial Intelligence techniques such as Expert Systems methodologies, Neural Network technology and Intent Specification Languages, has been recently proposed [8,14-16]. These models have been proposed as an appropriate means for modelling a user's behaviour, or as a means for classifying information, thus producing vital input for an intrusion detection system [3,15]. Indeed, many of the existing and emerging Intrusion Detection systems have adopted the idea of using Expert Systems, even in a distributed environment [15]. However, to the authors' knowledge, there is currently no system that incorporates all three classes of behavioural models, even though their combination has been proposed to be advantageous, particularly in the case

of computer networks [14,17].

Among Intrusion Detection Systems currently operational or under development, IDES [15] is a software system that uses innovative statistical algorithms for anomaly detection, as well as an Expert System that encodes known intrusion scenario. Wisdom and Sense [18] is also a software system that identifies transactions that are at variance with established patterns, based on historical data statistics. Time-based Inductive Learning [19] is a system learning by experience and creating rules for user behaviour over time. Anomaly detection is made by deviation detection and by detection of unrecognised activities. The Pattern-oriented Intrusion Detection Model [20] defines patterns of data flows and access context. It is able to ascertain specific intrusion patterns in audit trails. The Intrusion Detection Agent [21] is a rule-based pattern matcher software, utilising statistical profiles. The Network Security Monitor [22] is an intelligent network monitoring system utilising audit trails and statistical profiles. MIDAS [23] and HAYSTACK [24] are rule-based Expert Systems utilising audit trails to perform intrusion detection; they also have anomaly detection mechanisms. NADIR [25] is a prototype for network anomaly detection. It uses audit trails, as well as a rule-based Expert System. Finally, the Distributed Intrusion Detection System (DIDS) [26] is a prototype that monitors information on user behaviour in order to detect intrusions. It uses audit trails and a rule-based Expert System.

It is therefore evident, that although Expert Systems have been extensively used within the context of intrusion detection systems, they have not been complemented by either Neural Nets or Intent Specification Languages. Such a combined approach could lead to a result particularly useful in a distributed environment, where remote login and processing are routine operations, and where an intrusion effect (e.g. a worm spread) could be rapidly dispersed to several systems and locations if these are not properly protected, preferably by an unalterable detection mechanism such as a hardware-based neural net.

SECURENET aims at providing such an integrated environment, which will also be enriched with a few traditional techniques, such as cryptographic checksums and pattern matchers, which -although generally inefficient- are indeed efficient against the majority of currently known simple attacks.

In this paper, the architecture of SECURENET is described. At present, SECURENET has also established the Functional Specifications of its components [27], and has also implemented two pilot knowledge bases.

II. System Architecture

II.1 Tasks and functionalities

For the purposes of this work, the term *malicious attack* is taken to mean the carrying out of unauthorised activities. Unauthorised activities are classified into two broad categories, namely *viruses* and *intrusions*. A *virus attack* is based on appending code to an existing (executable) file of the system. The virus does not act (e.g. spread) on its own; rather the executable must be called first. An *intrusion attack* is based on stand-alone program(s), or commands entered directly by the keyboard. In many, the intruder's first goal is to gain access to hosts for which he has no access permission. Misuses of computer/network resources, i.e. unauthorised activities performed by an authorised user are also included here.

SECURENET must be able to perform three major tasks:

- detection of an attack
- classification of an attack in real time
- selection and application of appropriate countermeasures against the attack.

These are schematically shown in Figure 1.

The *detection* task includes all activities from the monitoring of information about the network and its components, to the step that concludes that the network is (or is not) under attack from malicious code. It comprises two main sub tasks, namely *monitoring* and *interpretation. Monitoring* is subdivided into two classes:

- monitoring of the *appearance* of files, i.e. performing cryptographic checksums to detect suspicious changes to executable files and/or computing useful indices from "snapshot" information, such as what processes are running at time t, or under which configuration a host is running.

- monitoring by *behaviour*, i.e. surveying the users and hosts activities and checking them for "normality".

It should be noted here that monitoring of the appearance of files may in fact prevent an attack, and also that the monitoring sub task includes the important activity of "interfacing with the Network Management System".

Interpretation consists of deciding whether an unusual behaviour or appearance is in fact the result of a malicious attack to the system.

The *classification* task consists of refining the knowledge that SECURENET has on an attack (recall that the detection task has only established that an attack is taking place). This classification is clearly "multifactor", since linear classification can hardly be precise enough if one considers that a virus may reside on the same executables as a 4096-type virus does, but replicate as the Icelandic and have "side activities" as Frodo. The purpose of the classification task is, of course, to allow accurately targeted countermeasures to be selected.

The *selection and application of appropriate countermeasures* task consists of utilising the results of the classification task in order to choose the most appropriate measures to be taken to counter this attack. It can then report its findings to the Security Officer, or, when SECURENET reaches a certain high reliability level, it can automatically apply countermeasures. Such countermeasures include the establishment of secure communication between network sites. This may be done via phone, fax, other data networks or by employing the method of Secure Distributed Computations -SDC [9].

Having established these secure communication paths, the Security Officer may distribute appropriate vaccines; deactivate paths; disable/forcibly logoff/restrict users; switch operation nodes, or allow a controlled evaluation of the attack to learn more about it. Vaccines for known attacks are selected from a suitable database that is part of SECURENET, whereas vaccines against innovative attacks have to be constructed on an ad-hoc basis.

Finally, a *simulation* task is also included in SECURENET. This task, which simulates the network functions, serves three main purposes, namely training the neural net component of the system; testing the actual IBC network for weaknesses by launching appropriately designed malicious attacks; and classifying actual attacks by utilising incomplete information available during an actual attack to predict useful indices pertaining to the future progress of the attack.

II.2 Appropriate technologies

Having described the tasks and functionalities of SECURENET, we now turn our attention towards selecting appropriate kinds of technologies that can be employed to successfully implement these tasks. These choices are schematically depicted in Figure 2, and will be discussed in some detail in the sequel.

For *detection by appearance* we use the method of cryptographic checksums. The reason for this choice is that this is the most secure (in terms of cost, speed, and efficiency against known malicious

attacks) among known integrity checking methods and also produces the smallest percentage of false positives. File scanners will also be employed as a secondary technique. For *detection by behaviour* we employ the technology of audit trails and profiling as well as the methodology of Intent Specification Languages (ISL) and of Cognitive Task Modelling. The *interpretation* of monitored information is done by employing both Neural Network technology and rule-based Expert Systems. The same technologies, complemented by symbolic manipulation techniques, are used for the implementation of the *classification* task. Decision-making will indicate the necessity for taking countermeasures when an attack is diagnosed with high probability.

Among the technologies selected for *countermeasures*, we list Self-correcting Secure Distributed Computations (SDC) [9], which is a technology not based on cryptography, but rather on self-stabilising techniques and algorithmic methods, and vaccines' selection/creation and distribution.

Among the above technologies, cryptographic checksums, file scanners and rule-based Expert Systems have been widely used in the past and are well understood. However, SECURENET is the first world-wide attempt to integrate these technologies with Neural Nets and User Intention Formal Models. The latter are relatively new and not much explored; this is why a more detailed discussion on them follows.

II.2.1 Neural Networks in Detection and Classification

As previously mentioned, malicious attacks can be classified as viruses and intrusions. While the former attempt to spread out rather passively, the latter do so quite actively, by looking for new blank files, hosts and systems. It is useful to point out here that once viruses start running their malicious code to perform a non-legitimate action, they turn towards intrusions. Therefore, any scheme aiming at classifying the spread of malicious attacks should take into account the fact that viruses, in the process of their spreading, increase the number of infected executable files, whereas intrusions increase irregular activities within the IBC network. Consequently, any such scheme should perform two distinct tasks, namely to attempt to detect irregular network activity (by comparing to meaningful patterns of "regular" activity) and to attempt to classify a virus spread mode (by comparing to known related spread schemes), when the existence of a virus has been established.

Therefore, an advanced alternative technique able to model as accurately as possible what the "regular" network activity really is, is needed. Such a technique should adhere to the following requirements:

- Adaptability, i.e. the ability to dynamically reconfigure the model according to formerly unencountered patterns of activity
- Generalisation, i.e. the ability to deal with incomplete or noisy patterns of activity
- Well integrated temporal aspects, i.e. the ability to process time-dependent patterns of activity

These requirements are met by neural networks computing. However, before concluding that neural nets are indeed a sound approach to use for the task at hand, one needs to explicitly define and model the concept of "regular" network activity; to show that the approach is indeed capable of learning the implicit underlying logic of the network activity; and also to show that the approach can take advantage and efficiently handle correlation between audit trails' measures.

At a high level, the concept of "regular" network activity can be given a comprehensive meaning as the existence of a cyclic use of its global resources to perform external or internal independent actions. At a low level, such a concept of "regular" activity is more closely related to the behaviour of the *actors* using the network. *Actors* can be users, applications (services provided by the network itself - 7th layer of the ISO reference model), or whatever uses resources to perform action (an intrusion procedure for example). The behaviour of actors may be modelled quite efficiently using the concept of *session*.

When a user is working, he usually attempts to accomplish a given task when submitting commands to the system. With time, he will take habits on doing things, picking one way he likes better rather than frequently changing his ways of doing things. Such a new habit will then be quickly integrated in his habitual work actions. The activity of a user is generally segmented into sequences, each one focusing on one part of the work.

Therefore, it seems reasonable to assume that user behaviour can be considered as a quasi-stationary process within a given sequence of actions. However, when the user is changing sequence, a transient state may occur, which can be considered as a stochastic process. A *session* can then be defined as a series of quasi-stationary sequences separated by non-stationary ones. Fortunately, when the user concentrates on one task, he attempts to follow the underlying logic required to achieve this task, with respect to his habits. This simply means that the sequences within a given session may be ordered, in which case the transitions between sessions lose part of their stochastic nature, thus turning overall user behaviour towards a more global quasi-stationary process.

For the neural network to be able to learn as accurately as possible what the "regular" network activity is, it is critical first to make sure that a certain regularity of the activity exists (which then may be shown through marked trends) and second, that these trends regularly appear in the audit trails. The snapshot of the network activity available to SECURENET is made up of audit records, each related to some performed action. The set of these records constitutes an audit trail, which may quite accurately be considered as a multivariate time series. Each audit record may be hierarchically organised to contain meaningful data samples, so that each field within the record can be viewed as a random variable distributed according to a probability distribution function law, that law reflecting a particular part of the influence of the action on current network activity. Such laws are difficult to establish based only on sampled values of the associated random variables; however, a neural net, by virtue of its learning and generalising capabilities can quite accurately approximate such a law of "regular" activity, given of course that some implicit logic governs the actions performed within the network.

Coming back to the structure of the audit record, it seems quite reasonable to assume that the random variables constituting its fields are correlated. Such correlations are certainly not explicitly expressed within the audit record itself and their use is linked with the underlying probability distribution functions (pdfs) governing the individual random variables within the record. Having established the fact that a neural network may in principle learn such laws, its ability to efficiently use correlations follows easily.

What remains in order to complete the discussion on the soundness of the neural net approach to intrusion detection and classification is the issue of temporal input. An efficient way to detect irregular activity is to predict the new state of activity based on an analysis of previous states and to compare - through distance analysis- it with the actual new state, rather than attempting to identify irregularities per se. Such an approach may be implemented by neural networks usually referred to in the literature as *predictive*. These are usually based on the multilayer perceptron and utilise the error back propagation learning algorithm; they are trained with inputs made up from the N most recent patterns selected through a shifting time window.

However, such networks have to be ruled out because of the following reason: N has to be fixed and has to remain constant during the life of the network, since any change would require complete retraining. Recall that the concept of "regular" network activity can be thought of as the existence of a cyclic use of the network's resources to perform external or internal intended actions. Fixing N implies that this cyclic process would have to be periodic, so that the width of the shifting time window can fit the cycle period, in order for it to be well analysed. Unfortunately, this is obviously an unreasonable assumption.

An alternative, which remedies the above problem, is to use a recurrent neural network, whereby part of the output at time t is fed back as input to be processed at time t+1. This creates an internal memory within the neural net, effectively functioning as the time window used in perceptrons. These networks

do meet the specified requirements, since

- they are naturally able to handle time series and to keep track of past events in their internal memory.

- they have a long term memory that stores the law of the "regular' network activity within the connections.

II.2.2 User Intent Specification Modelling

The component of the system responsible for detecting and/or predicting user intentions by utilising user behaviour information is based on *Cognitive Task Modelling* and *Task Knowledge Structures. Cognitive Task Models (CTM)* and *Task Knowledge Structures (TKS)* are two formalisms emanating from Applied Psychology and Cognitive Science. CTMs are the result of investigations on the nature of mental activities taking place when a human carries out a task. TKSs describe how the knowledge needed to carry out a task or series of tasks is organised, or structured.

The CTM approach [28],[29] to user modelling involves building "approximate" descriptions of the cognitive activity underlying task performance in human-computer interactions. This approach does not aim at simulating exactly what is going on in the user's head, but just to capture the salient features of the cognitive processing activity. These have been identified so far, as being

- the human mental processing configurations.

- the procedural knowledge used by the human mental processes.

- the properties of any memory records that are assumed to be accessed.

- a description of the way the whole mental mechanism is dynamically controlled and co-ordinated.

TKSs [30] provide a rich representation of knowledge associated with task behaviours. Each TKS represents task goals, task plans, procedural and declarative (general) knowledge associated with a task, along with objects associated with the task and the actions upon objects. TKS as a formalism has been extended via the Fuzzy Sets Theory and has been used in representing knowledge for decision making as well as specific domain knowledge in intelligent Systems development.

SECURENET uses information included in audit trails in the form of actions upon objects and specific procedures; all these are amongst the recognised components of TKSs. User intentions within this formalism simply means the TKS of the task intended by the user; this can be built step by step via the appropriate SECURENET component using audit trails. This process may produce (synthesise) either the model of the actual intended task or the most plausible one among the ones recognised within the class of users, or even the unknown ones, which will be looked upon as illicit until cleared.

The results of user intentions identification are utilised by the "Detection through User Intentions Identification (DUII)" component of SECURENET. DUII traces the network use via audit trails on a real time basis, in order to synthesise user behaviour in terms of intended tasks. This component is complementary to the other detecting components, in the sense that it is capable of detecting attacks that would otherwise go undetected, being composed of a number of authorised actions that, when combined, result in an illicit task. The functional components of DUII are

- the Audit Data Processor (ADP)
- the Task Synthesiser (TS)
- the Comparator
- the Decision Support Function (DSF).

Their role is to "translate" audit trails into network user intentions, making use of the structural

components, which consist mainly of the relevant knowledge base. The basic primitive units that the module recognises are the User Behavioural Units (UBU); these are synthesised by actions upon objects or simple procedures and actually correspond to minimum behaviourally meaningful templates. UBUs are also recognised TKS components and are represented as such in the common SECURENET Knowledge Base. Conditioned groupings of UBUs, involving conditioned repetitions, form the various authorised (and illicit) tasks, also represented in the KB in the form of TKSs. These are considered, for our purposes, as the Intention Models. These tasks are updated in the KB, by the security officer, according to security policies, to jobs performed within the system, as well as to any information given by SECURENET itself.

The main functional component of DUII is the *Comparator*, which classifies the synthesised tasks as authorised ones or as possible intrusions. For every new UBU, a new task Buffer (if one does not already exist) is created by the synthesiser for the tasks in the KB that are related to it. Deviations, similarities, differences, inconsistencies, etc. are translated into user intentions. This is done through the utilisation of an Intent Specification Language, a computational framework for reasoning about intentions.

The DSF communicates the Comparator's conclusions, as well as those of the overall DUII to the SECURENET security officer. It also communicates with the other similar functions of SECURENET that share the same countermeasure KB.

II.3 Building Blocks

Figure 3 depicts a first draft of SECURENET's building blocks, as well as of the information flow between them. The building blocks chosen follow from the selected technologies and are, naturally, subject to revision in the course of prototype implementation.

II.3.1 Network Interface

The function performed by this block is to supply the Independent Network Monitoring (INM) system -to be described in detail in the sequel- with necessary input for the latter to successfully carry out its functions. This input is provided either by IBC functions other than SECURENET or by SECURENET itself, in terms of *measures.*

The proposition for the IBC performance management levels [31-32] and its refinement in terms of monitored objects, which is shown in the following list, specifies several types of functions, several of which are useful to the INM; these are marked with an x:

Service level
monitoring performance related to customer
- measure quality of service
- measure workload
- measure waiting time

alarm monitoring

Network level
- traffic control monitoring
- queue monitoring

monitoring overall network performance

Element level
performance logs
monitoring records
alarms monitoring
- setting monitoring thresholds to trigger events

Measures are classified as *User Measures*, *User Network Activity Measures*, and *Remote Node Measures*. *User Measures* have been defined in [33] and are given also in Table 1 for the sake of completeness. *User Network Activity Measures* are described in Table 2. *Remote Node Measures* are described in Table 3. In all these tables, letters in parentheses indicate type of data; O stands for Ordinal, LC stands for Linear Categorical (which can be represented by an array), and BC stands for Binary Categorical (which can be represented by a single bit).

II.3.2 Independent Network Monitoring system (INM)

Seven types of active parts, called classes of *objects,* are defined; five of them belong explicitly to the INM, whereas the other two are parts of SECURENET with which INM communicates. Objects contain data as well as rules to handle these data and communicate using an asynchronous message passing scheme. This means that the sender of a message does not know who the receivers will be; it also allows easy reconfiguring and restructuring of the system in real time. The classes of objects explicitly belonging to the INM are

- protocol parsers
- short term profilers
- long term profilers
- convergent profilers and
- the INM co-ordinator

whereas the two additional classes of objects with which INM communicates are

- triggers and
- profile analysers

On small installations all classes of objects can reside in one machine, whereas on larger installations they can be distributed among several machines and on very large installations, each class of objects can be distributed among a few machines.

Protocol parsers are the data switchers of the system. They are hierarchically arranged, so that the Ethernet parser is at the first level. The Ethernet parser intercepts each Ethernet data packet, parses it (i.e. forms a message containing -as its fields- all useful information in the packet), takes the Ethernet header and trailer out of the packet, and sends a message to the Ethernet profiler. It also checks to which higher level protocol this packet belongs (here there is only IP, but there could also be other protocols, as DECNET for example) and sends another message to the IP parser; all information that could be used by higher levels is also included in this message. The IP parser works in a similar manner: After parsing the packet and getting the relevant IP information -like internet numbers of sender and receiver- the data packet is sent to the IP profiler. Another message is then sent to the appropriate protocol parser at the higher than IP level.

Short term profilers are short lived objects created when a new association (or tuple) of key data fields appears, for example when a new connection between two hosts in the network is made. There is one short term profiler for each connection on each level where there is a corresponding connection between programs in hosts. Short term profilers perform two tasks: First, they gather data used momentarily (for example the telnet profiler takes the user-id and password from a telnet session) and stores them for later use by another object; second, they gather statistical information (for example how long the session lasted or how many characters were transmitted). The life of the short term profiler ends when the corresponding connection between hosts ends. The accumulated data pertaining to that connection are sent to the corresponding statistical profiler. If there is no corresponding statistical profiler, one is created.

Long term profilers collect and store information that is relatively constant in the network. For example, an IP long term profiler should have statistics about the amount of data transfer between any

two host pairs in the network, what data are stored, etc. There is a long term profiler associated with each short term profiler that ever existed in the system, as well as additional long term profilers, such as a user profiler, which collects information from many other long term profilers.

Convergent profilers are high level profilers collecting data from various low level profilers; they concentrate on their own domain of interest and form their view of the network traffic profile. Low level profilers are divergent because all data packets go through the Ethernet profiler, which distributes them to IP and ICMP profilers according to their type. The IP profiler distributes its data packets to TCP, UDP, and other Layer 3 profilers, according to their type. Examples of convergent profilers are the user profiler, forming the profile of a user in the network, and the password profiler, saving legitimate user-id/password pairs in a database, for comparison purposes.

The *INM Co-ordinator,* whose range of action consists of all the monitoring and interpreting activities described herein, creates profilers and triggers when new ones are needed, and is responsible for co-ordinating the activities of the INM in general.

Triggers are objects that can be automatically created when a short term profiler is created. Triggers create possible alarms of well-known alarm situations, for example when someone attempts to get he /etc/passwd file via tftp.

Profile analysers are the objects supporting the interpretation part of detection. They operate on the long term and convergent profiler data. They perform three major tasks: checking short/long term profilers for suspicious events, processing/confirming/rejecting alerts raised by triggers, and providing information to the co-ordinator.

It is clear from the description of the INM components given previously that INM is built on a distributed architecture. There are several advantages to this option, as follows:

- It allows distribution of computing over several different platforms with an easy and sophisticated method.
- It allows adding resources (such as processing power and special hardware) to the SECURENET system without any modifications to the existing components.
- It provides for methods to handle SECURENET's own security.
- It facilitates the implementation of the final system, thus also facilitating the use of sophisticated and efficient methods for each part; furthermore it allows using methods and hardware not currently available for use.
- It facilitates the configuration, maintenance and use of SECURENET, since the user interface does not need to be implemented as part of another software. This allows independent development of the user interface.
- It allows higher internal security, since information concerning the internal design of the SECURENET's components, as well as information concerning methods used and solutions found does not need to be distributed.
- It allows the relocation of running objects/processes onto another processor, thus allowing balancing computer resources.
- It saves computing resources, since objects waiting for their trigger messages do not spend processing time by waiting actively.
- It ensures that every piece of information that needs to be processed will be processed, saving as much of the resources as possible in the process of doing so.

- It provides tools for handling fault tolerance, so that no messages are lost even though parts of the system may suddenly fail.

Its advantages notwithstanding, this distributed architecture does impose the need for increased and reliable communication between system components. These are achieved through a set of communications servers, based on the Linda distributed/parallel computing communication method, but heavily modified to fit the needs of SECURENET. The SECURENET version of Linda is called the *SECURENET Internal Communication Service (SICS).*

Objects access SICS via a small number of library routines. These routines are defined strictly by behaviour, thereby facilitating implementation on different systems. SICS does not make any assumptions on the underlying communication network; any reliable bit stream is valid.

We could have several communication servers, in order to distribute the processing needs and to isolate separate parts of the system. Each server can be another server's object. Several objects can be collected into a *domain.* A *domain* is a collection of a communication server, some objects related to each other, and a security class. A domain lives under specific hardware. Domains can communicate with each other only by using predefined paths and under strict supervision. This means that we can build several physical collections of hardware, so that they can only receive information from certain objects from domains in higher security levels and so can be used to create "workstations" for each role that will be needed, such as a security officer. Domains are actually needed for two reasons:

- They secure information and the processing platform, by locating information and processing at a certain security level, only to a hardware which is at the same level.

- They facilitate the definition, configuration, and auditing of communication between objects at different security levels.

II.4 User Interface

SECURENET discriminates three classes of users:

- System Manager, whose main responsibility is to guarantee continuous operation of the IBC network services and all systems attached to the network, for the benefit of the organisation in question.

- Security Manager, whose responsibility is the preservation of security of all IT services in the organisation.

- Security Controller/Security Guard, whose responsibility is the continuous monitoring of IT services.

The User Interface receives input basically from three sources:

- Administration type of data interchange between the user and SECURENET come via (1) Internal Controls, (2) Network Management Data Transfer, (3) Data Base , Knowledge Base and Neural Network Data Management and (4) Internal Message Handling.

- Data related to detection come from (1) the Network Description Function, (2) the Network Monitoring Function, (3) the Virus Detection Function, (4) the Intrusion detection Function and (5) the User Intention Scanning Function.

- Countermeasures' information comes from the Countermeasure Function.

and produces output to the user terminal, the printer, an ordinary mass storage media and a WORM.

The User Interface is realised through a number of displays. These can be grouped as follows:

- Status (of the various aspects of the network and of the connected systems, their users, etc.).
- Dynamic behaviour (of the various aspects of the network and of the connected systems, their users, etc.).
- Warnings of suspicious events.
- Alarms of suspicious events.
- Guidelines for various countermeasures.

SECURENET performs a priority based User Interface. This means that in the case of need to alarm the user because of (1) suspected or identified virus, (2) intrusion, (3) any suspicious event in the network or (4) any critical malfunction in the system itself, the alarm will get the highest priority. The second highest priority goes to all types of warnings. These events will supersede all other information except alarms. The third priority level is the information from the constant monitoring of the key parameters in the network. The lowest priority level from the user viewpoint includes, among others internal parameters, network topology and profiles.

Users have access to SECURENET via the User Interface subsystem, which receives and submits data from and to various functional subsystems.

III. Conclusions

This paper described the architecture of SECURENET and laid the basis of the future development of SECURENET. SECURENET, as now planned, is the first system to integrate alternative advanced technologies in order to provide effective and efficient detection of malicious attacks in IBC networks environments. It can adaptively learn normal network behaviour, and detect anomalous network activity in real-time, while the alternative detection approaches minimise false alarm and enhance security. It is also the first system able to systematically apply countermeasures to enhance network immunity.

Even though SECURENET has already established the functional specifications of its components, these were not presented herein due to space limitations. A complete and operational prototype has not been yet completed; however, part of it has been tackled, namely the beginning of the implementation of two crucial Knowledge/Data Bases, the ones on Viruses and Intrusions. The development platform underlying the content of this paper provides a solid basis for the successful completion of SECURENET within the next two years. This will coincide with the actual operation -in Europe- of IBC networks, where SECURENET will be tested and validated.

References

1. National Research Council, *Computers at Risk: Safe Computing in the Information Age*, National Academy Press, Washington DC, 1991.
2. P.E. Denning (Ed.), *Computers under attack: Intruders, Worms and Viruses*, Addison-Wesley, 1990.
3. D. Guinier, "Prophylaxis for Virus Propagation and General Computer Security Policy", *ACM SIGSAG Review*, Vol. 9, no. 2, pp.1-10, 1991.
4. H. Highland, *Computer Virus Handbook*, Elsevier, 1990.

5. F. Cohen, *Computer Viruses,* Ph.D. Dissertation, University of Southern California, 1985.

6. F. Cohen, "Computer Viruses: Theory and Experiments", *Computers and Security,* Vol. 6, no. 1, pp. 22-35, 1987.

7. L. Adleman, "An Abstract Theory of Computer Viruses", in L. Hoffman (Ed.), *Rogue Programmes: Viruses, Worms and Trojan Horses,* Van Nostrand-Rhineholt, pp. 307-323, 1990.

8. J. Kephart and S. White, "Directed Graph Epidemiological Models of Computer Viruses", in *Proceedings, 1991 IEEE Computer Society Symposium on Research in Security and Privacy,* IEEE Computer Society Press, pp. 343-359, 1991.

9. R. Ostrowski and M. Yung, "How to Withstand Mobile Virus Attacks", in *Proceedings, 10th ACM Symposium on Principles of Distributed Computing,* pp. 51-59, 1991.

10. F. Cohen, "Implications of Computer Viruses and Current Methods of Defence", in P.E. Denning (Ed.), *Computers Under Attack: Intruders, Worms and Viruses,* Addison-Wesley, pp. 381-406, 1990.

11. D. Gritzalis, *Information Systems Security,* Greek Computer Society Monograph Series, Athens, 1989.

12. G. Davida, Y. Desmedt and B. Matt, "Defending Systems Against Viruses Through Cryptographic Authentication", in *Proceedings, 1989 IEEE Computer Society Symposium on Research in Security and Privacy,* IEEE Computer Society Press, pp. 312-318, 1989.

13. P. Feldman and S. Micali, "An Optimal Algorithm for Synchronous Byzantine Agreement", *ACM Transactions on Computer Systems,* Vol. 31, no. 3, pp. 148-161, 1988.

14. H. Debar, M. Becker and D. Siboni, "A Neural Network Component for an Intrusion Detection System", in *Proceedings, 1992 IEEE Symposium on Research in Computer Security and Privacy,* IEEE Computer Society Press, to appear, 1992.

15. T.F. Lunt, A. Tamatu, F. Gilham, R. Jagannathan, C. Jalali, H. Javitz, A. Valdes and P. Neumann, *A Real-time Intrusion Detection Expert System,* Technical Report SRI-CSL-90-05, Stanford Research Institute, 1990.

16. J. Darzentas, J. Mamaletos, T. Spyrou and J. Darzentas, "Knowledge Retrieval to Support Explanation in Task Domains", *Knowledge Based Systems,* submitted, 1992.

17. R. Simonian, R. Henning, J. Reed and K. Fox, "A Neural Network Approach Towards Intrusion Detection", in *Proceedings, 13th National Computer Security Conference,* pp. 125-134, 1990.

18. H.S. Vaccaro and G.E. Liepins, "Anomaly Detection: Purpose and Framework", in *Proceedings, 12th National Computer Security Conference,* pp. 280-289, 1989.

19. H.S. Teng, K. Chen and S.C.-Y. Lu, "Adaptive Real-time Anomaly Detection Using Inductively Generated Sequential Patterns", in *Proceedings, 1990 IEEE Symposium on Research in Computer Security and Privacy,* IEEE Computer Society Press, pp. 278-284, 1990.

20. S.W. Shieh and V. Gligor, "A Pattern-Oriented Intrusion Detection Model and its Applications", in *Proceedings, 1991 IEEE Symposium on Research in Computer Security and Privacy,* IEEE Computer Society Press, pp. 327-342, 1991.

21. K. Brunnstein, S. Fischer-Hubner and M. Swimmer, *Computer Virus Catalogue,* Technical Report, Hamburg Virus Test Centre, University of Hamburg, 1992.

22. D. Mansur, "Network Security Monitor", in *Presentation Notes for IDES Workshop 3*, 1988.

23. M. Sebring, E. Shellhouse, M. Hanna and A. Whitehurst, "Expert Systems in Intrusion Detection: A Case Study", in *Proceedings, 11th National Computer Security Conference*, 1988.

24. S.E. Smaha, "Haystack: An Intrusion Detection System", in *Proceedings, 12th National Computer Security Conference*, pp. 37-44, 1988.

25. K.A. Jackson, D.H. Dubois and C.A. Stallings, "An Expert System Application for Network Intrusion Detection", in *Proceedings, 14th National Computer Security Conference*, pp. 215-225, 1991.

26. G. Dias, K. Levitt and B. Mukherjee, "Modelling Attacks on Computer Systems: Evaluating Vulnerabilities and Forming a Basis for Attack Detection", in *Proceedings, SRI Intrusion Detection Workshop 5*, pp. 296-304, 1990.

27. SECURENET RACE R2057 Project, *SECURENET System Development Plan*, Deliverable 5, 1992.

28. P.J. Barnard, M. Wilson and A. Maclean, "Approximate Modelling of Cognitive Activity With an Expert System: A Theory Based Strategy for Developing an Interactive Design Tool", *The Computer Journal*, Vol. 31, pp. 445-456, 1988.

29. P.J. Barnard, J. May and A.J.K. Green, *Report on the Design Capabilities and Potential of an Expert System Modeller Based Upon Cognitive Theory*, Technical Report, ESPRIT Basic Research Action 3066, AMODEUS Project Deliverable 13, 1991.

30. P. Johnson and H. Johnson, "Knowledge Analysis of Tasks: Task Analysis and Specification for Human-Computer Systems", in A. Dowton (Ed.), *Engineering the Human-Computer Interface*, McGraw Hill, London, 1991.

31. CEC DG XIII, *RACE Common Functional Specifications (CFS)*, Document 1, Issue B, RIC Association Internationale.

32. CEC DG XIII, *RACE Common Functional Specifications (CFS)*, Document 7, Issue B, RIC Association Internationale.

33. T. Lunt, *A Real-time Intrusion Detection Expert System*, Final Technical Report, SRI Project 6784, Stanford Research Institute, 28 February 1992.

Measure	Description
CPU Usage (O)	CPU Usage
I/O Usage (O)	I/O Usage
Location of Use (LC)	number of connections from each location
Mailer Usage (LC)	number of times each mailer was used
Editor Usage (LC)	number of times each editor was used
Compiler Usage (LC)	number of times each compiler was used
Shell Usage (LC)	number of times each shell was involved
Window Command Language (LC)	number of times each window command was used
Program Usage (LC)	number of times each program was used
System Call Usage (LC)	number of times each system call was used
Directory Usage (LC)	number of times each directory was accessed
Directory Usage (BC)	whether a directory was accessed
Commands Used (O)	number of different commands used
Directories Created (O)	number of directories created
Directories Deleted (O)	number of directories deleted
Directories Read (O)	number of directories read/accessed
Directories Modified (O)	number of directories modified
File Usage (LC)	number of times each file was accessed
File Usage (BC)	whether a file was accessed
Temporary File Usage (O)	number of temporary files accessed
Files Created (O)	number of files created
Files Deleted (O)	number of files deleted
Files Read (O)	number of files read/accessed
Files Modified (O)	number of files modified
User IDs Accessed (LC)	number of times user ID was changed
User IDs Accessed (BC)	whether another user ID was changed
System Errors (O)	number of system related errors
System Errors by Type (LC)	number of times each type of error occurred
Audit Record Activity (LC)	number of audit records for each hour
Hourly Activity (BC)	whether an audit record was received for each hour
Day of Use (LC)	number of audit records for each day
Day of Use (BC)	whether audit records were received for each day
Remote Network Activity (O)	amount of remote network activity
Network Activity by Type (LC)	amount of network activity of each type
Network Activity by Hosts (LC)	amount of network activity for each remote host
Local Network Activity (O)	amount of local network activity
Local Network Activity by Type (LC)	amount of local network activity of each type
Local Network Activity by Hosts (LC)	amount of local network activity for each host

Table 1: User Measures

Measure	Description
Target Activity (O)	Number of records generated
CPU Usage (O)	CPU time used
I/O Usage (O)	I/O used
Bad Login Attempts (O)	Number of bad login attempts
Hourly Bad Login Attempts (LC)	Number of bad login attempts for each hour
Errors (O)	Number of system errors
Errors by Type (O)	Number of errors of each type
Hourly System Errors (LC)	Number of errors for each hour
Network Activity (O)	Amount of network activity
Network Activity by Type (LC)	Amount of network activity of each type
Network Activity by Host (LC)	Amount of network activity for each remote host

Table 2: User Network Activity Measures

Measure	Description
User Accounts (LC)	Number of times each account is accessed
Activity Type (LC)	Amount of network activity of each type
Hourly Use (LC)	Amount of network activity each hour
Hourly Use by Type (LC)	Amount of network activity of each type for each hour
Bad Login Attempts (LC)	Number of bad login attempts

Table 3: Remote Node Measures

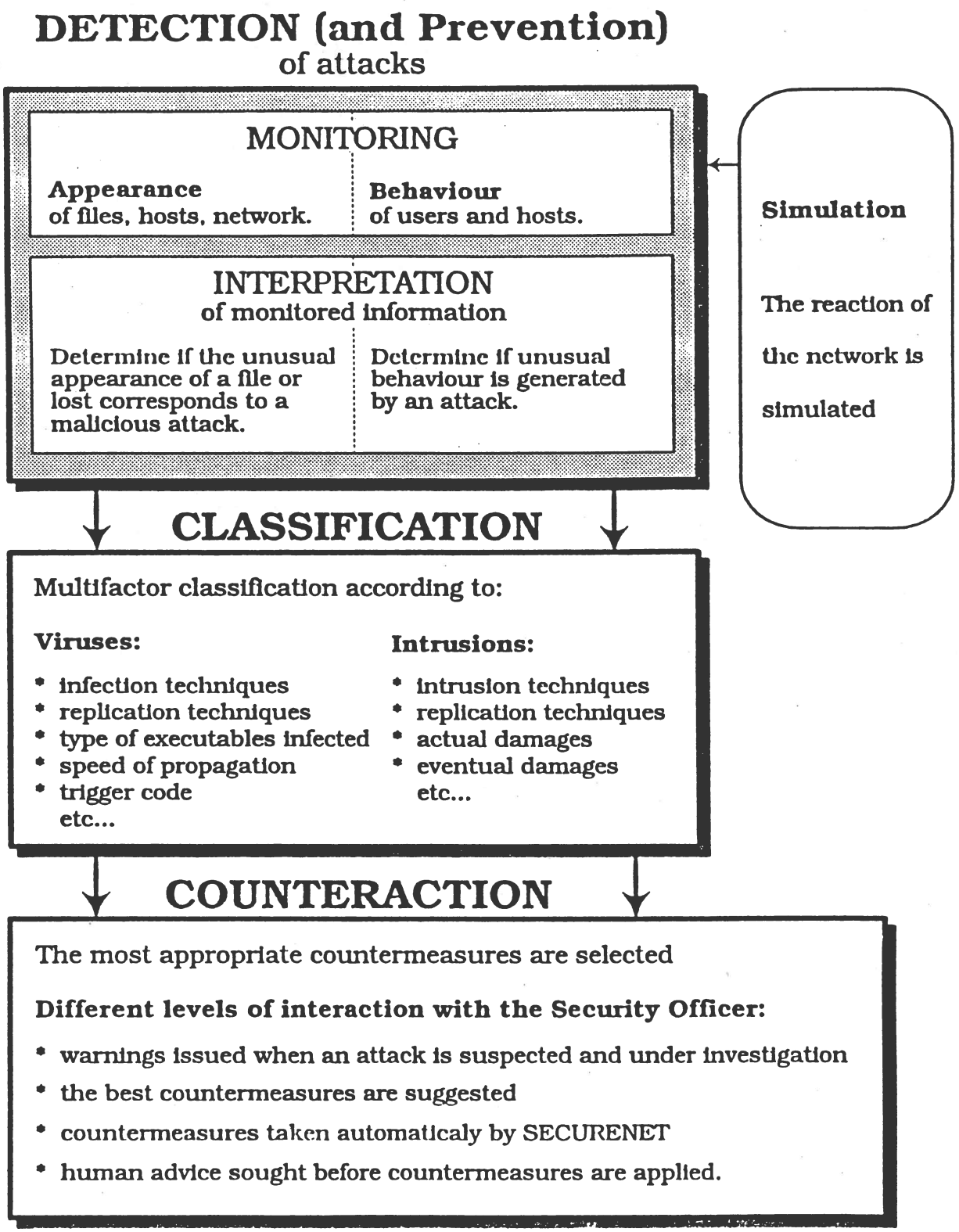

DETECTION (and Prevention)
of attacks
MONITORING
Appearance
of files, hosts, network.
Behaviour
of users and hosts.
INTERPRETATION
of monitored information
Determine if the unusual appearance of a file or lost corresponds to a malicious attack.
Determine if unusual behaviour is generated by an attack.
Simulation
The reaction of the network is simulated
CLASSIFICATION
Multifactor classification according to:
Viruses:
* infection techniques
* replication techniques
* type of executables infected
* speed of propagation
* trigger code
etc...
Intrusions:
* intrusion techniques
* replication techniques
* actual damages
* eventual damages
etc...
COUNTERACTION
The most appropriate countermeasures are selected
Different levels of interaction with the Security Officer:
* warnings issued when an attack is suspected and under investigation
* the best countermeasures are suggested
* countermeasures taken automaticaly by SECURENET
* human advice sought before countermeasures are applied.

Fig. 1

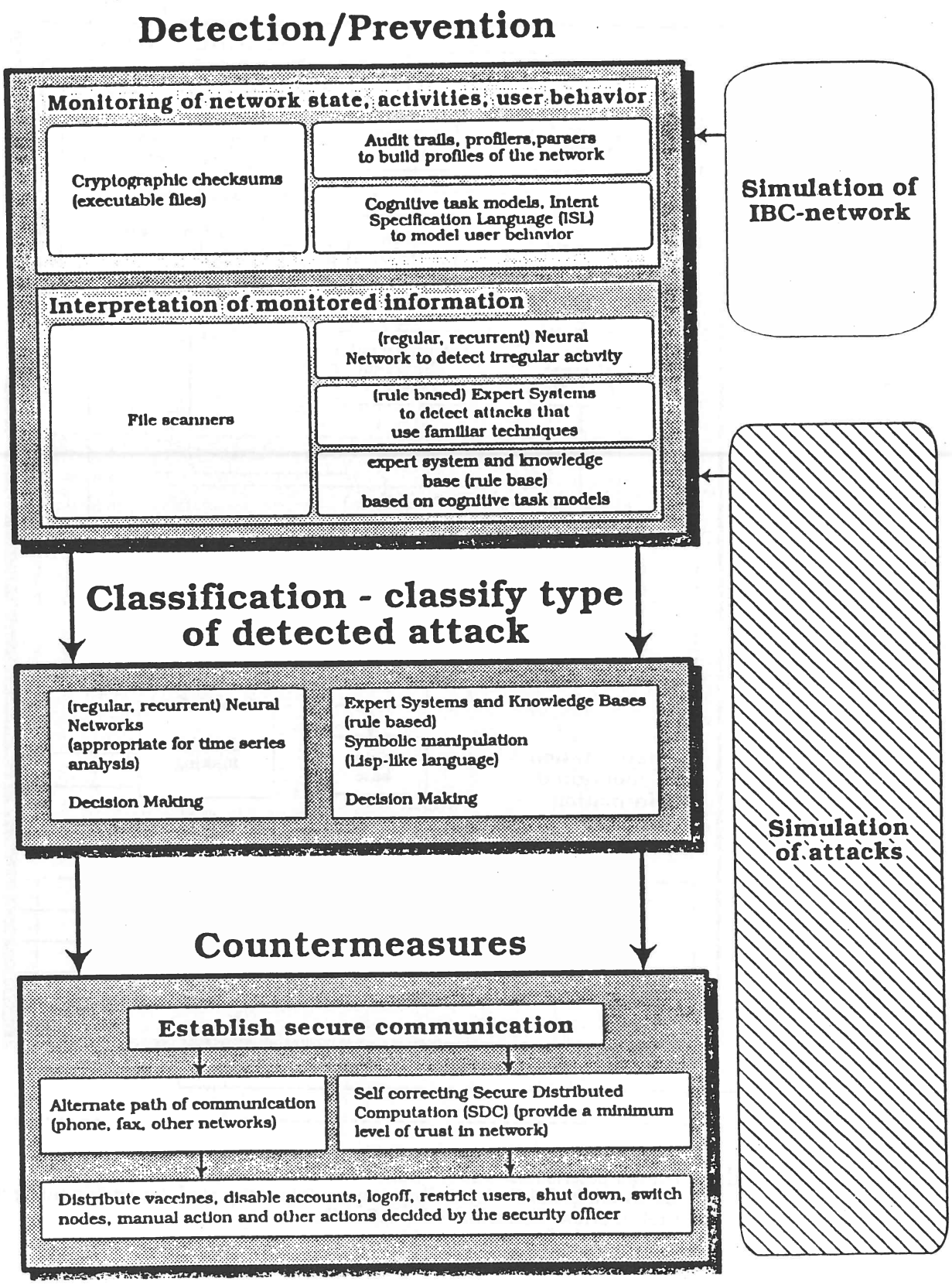
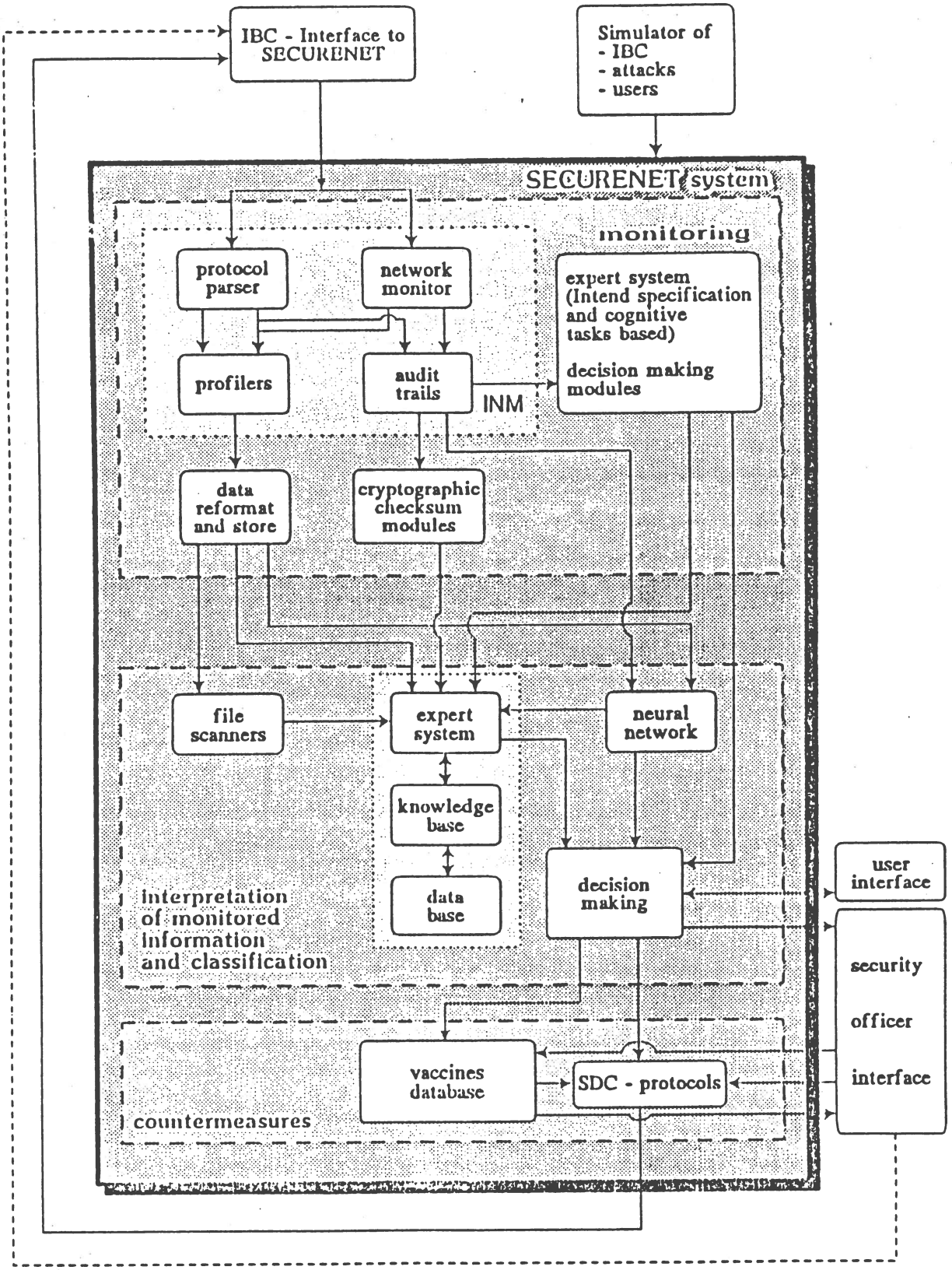

Detection/Prevention
Monitoring of network state, activities, user behavior
Cryptographic checksums (executable files)
Audit trails, profilers,parsers to build profiles of the network
Cognitive task models, Intent Specification Language (ISL) to model user behavior
Simulation of IBC-network
Interpretation of monitored information
File scanners
(regular, recurrent) Neural Network to detect irregular activity
(rule based) Expert Systems to detect attacks that use familiar techniques
expert system and knowledge base (rule base) based on cognitive task models
Classification - classify type of detected attack
(regular, recurrent) Neural Networks (appropriate for time series analysis)
Decision Making
Expert Systems and Knowledge Bases (rule based) Symbolic manipulation (Lisp-like language)
Decision Making
Simulation of attacks
Countermeasures
Establish secure communication
Alternate path of communication (phone, fax, other networks)
Self correcting Secure Distributed Computation (SDC) (provide a minimum level of trust in network)
Distribute vaccines, disable accounts, logoff, restrict users, shut down, switch nodes, manual action and other actions decided by the security officer

Fig. 2



Legend: a. Arrows describe information flow.
b. Boxes are building blocks. They may contain many modules implementing functions.
c. Broken arrow: a potential information flow.

Fig. 3