Intention Modelling: Approximating Computer User Intentions for Detection and Prediction of Intrusions.

T. Spyrou and J. Darzentas
Research Laboratory of Samos, Department of Mathematics
University of the Aegean, GR 83 200 Karlovassi, tsp@aegean.gr

Abstract

This paper introduces and describes an innovative modelling approach which utilises models that are synthesised through approximate calculations of user actions and extensive representation of knowledge about how to perform these actions. The Intention modelling approach is based on theories of cognitive and task modelling as well as on theories of intention, rational action and plan recognition. Intention Models (IMs) have been used in the detection of malicious attacks which usually do not consist of illegal actions, but of a set of actions individually acceptable to the system which at a higher level may form non acceptable task(s). A first effort at implementing these models for a real application was for the creation of the UII system, a research prototype for the detection of anomalous behaviour of network users obtained by reasoning about the characterisation of their intentions. It was developed as an autonomous module within SECURENET, a European funded programme that aims at defending open computer systems, employing advanced techniques and methodologies.

Keywords

Intrusion detection, Intention modelling, Computer security.

1. INTRODUCTION

The enormous development and extensive usage of open computer systems and especially computer networks have brought several security issues to a critical point. A large number of innovative methods and methodologies have been used for the detection of anomalies in such systems. Two general categories of detection methodologies have been established up till now; those which rely on detection-by-appearance and those which use detection-by-

behaviour. A number of theoretical approaches (Denning, 1987; Dias et.al. 1990; Lunt, 1990; Javitz, 1991), and a number of systems under development or already developed systems (Debar et.al. 1992; Fox et.al. 1990; Jackson et.al. 1991; Lunt 1988; Lunt, 1993; Lunt et.al. 1992; Mansur, 1988; Sebring et.al. 1991; Shieh and Gligor, 1991; Smaha, 1988; Spirakis et.al. 1994; Vaccaro and Liepins, 1989) are recently representative of these methods and methodologies which use statistical algorithms, encoding of known intrusion scenaria, identification of transactions that are at variance with established patterns based on historical data, deviation detection, pattern-oriented detection, rule-based pattern matching, model based detection, and in general statistical or rule-based processing of audit data.

This paper introduces and describes an innovative methodology used in the detection of various malicious attacks. Conceptually, this approach utilises models that are synthesised through approximate calculations of user actions and extensive representation of knowledge about how to perform these actions. It makes predictions and characterisations of activities which compose tasks, by reasoning about representations of user's tasks, user's cognitive representations and task processing. This approach aims at defending open computer systems, employing advanced techniques and methodologies which have not yet been explored in such a context. This methodology is intended to be complementary to already developed methodologies, basically by detecting attacks which cannot be detected by the others, since these attacks could be composed of a number of authorised actions which when combined result in illicit tasks.

A main assumption here is that if the 'task' or 'tasks' the user wants, or intends to perform using a computer system could be 'identified', then reasoning about the normality of the intentions could be produced. Anything not expected or explained by the predefined knowledge could be considered out of the range of the normal user's behaviour. The idea is that the user should be observed at the overall task (or subtask) level and not just monitored on the basis of separate actions which are judged as permissible or not. Being able to judge the user on the basis of overall tasks (or subtasks) and not on the basis of individual or groups of events, should be more efficient for the cases of intelligent intrusion attempts. In other words, if these events are identified or associated to specific components of a task structure then although they may not be illicit at a specific stage of use, the part of the task structure they will eventually form (e.g. procedure) may very well be. In addition an event which is not allowed does not necessarily indicate an intrusion, most of the time it is an error, while an intention identified as not permissible can be readily recognised as an intrusion.

Task Knowledge Structures (TKS) and to a great extent Cognitive Task Modelling (CTM) (Johnson et. al. 1988; Barnard et. al. 1991) are employed here for the design and development of this approach and in particular in the representation of the cognitive and task knowledge structures which are required.

Also, current intention theories, theories of rational action and plan recognition (Allen, 1990; Bratman, 1990; Cohen and Levesque, 1990; Kautz, 1990) are explored and utilised as means for the operational and applicable definition of the intention models (IMs), the objective being to transform such theories and modelling techniques into a more applicable form considering the assumptions and simplifications originating from the intrusion detection related applications.

The next section gives a brief overview of CTM and TKS in the context of their use in intention modelling here. In section 3 an overview of the theories relevant to intention

modelling is given and section 4 introduces the theoretical construction of the IMs. Section 5 describes the architecture and the implementation efforts towards the development of a prototype based on these models and finally a discussion follows in section 6 on the usability of intention modelling as a tool for developing computer security products ending with a summary outlining future perspectives.

2. COGNITIVE TASK MODELS AND TASK KNOWLEDGE STRUCTURES

The aim of this section is to introduce and discuss the two modelling approaches which are used as the theoretical basis for the design and development of the IMs. These are Cognitive Task Models (CTM) (Barnard et.al. 1991; Barnard and May, 1993; Barnard et.al. 1988) and Task Knowledge Structures (TKS) (Johnson et.al. 1988; Johnson and Johnson, 1991; Johnson 1989). Briefly, CTMs and TKSs are two formalisms emanating from applied psychology and cognitive science in general. CTMs have investigated the nature of mental activities that take 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 to user modelling involves building approximate descriptions of the cognitive activity underlying task performance in human-computer interactions. This approach does not aim to simulate exactly what is going on in the user's head, but just to capture the salient features of their cognitive processing activity. These have been identified so far as the following four approximations which describe key attributes of human mental processing configurations. These approximations are the configurations themselves; the procedural knowledge used by the human mental processes; the properties of any memory records that are assumed to be accessed; and a description of the way the whole mental mechanism is dynamically controlled and co-ordinated.

TKSs provide a rich representation of knowledge associated with task behaviours. Each TKS represents task goal, task plans, procedural and declarative (general) knowledge associated with a task, along with objects associated with the task and the actions upon those objects. The theory of Task-related Knowledge Structures arose out of a need to model the knowledge people recruit when called upon to perform a task, for use in developing intelligent computer based systems. Thus a TKS is a summary representation of the different types of knowledge that are required to carry out a task or tasks.

This knowledge has been traditionally investigated by Task Analysis (TA) involving the collection of information about what people do when they carry out tasks and how people perform those tasks. Various approaches to Task analysis (TA) have been developed and probably the most often cited is the GOMS approach of Card, Moran and Newell (Bonnie et.al. 1994; Card et.al. 1983; Card et.al. 1980). GOMS is a formalism for representing routine cognitive skill.

The work of Kieras and Polson (1985) extended GOMS by arguing that production rules can be used to model goals, operations, methods and selection rules and that these production rules bear a close relationship to the way humans structure their knowledge of the task. The argument that the task knowledge structures are functionally equivalent to the knowledge structures that people possess and use when performing tasks was further elaborated with the work of Johnson et al (1991) and the following theoretical assumptions were made:

- Task knowledge is represented in conceptual or general knowledge structures in long term memory. All the knowledge a person possesses about a task is contained within the task knowledge structure and the TKS is activated in association with task performance.
- The structure of this knowledge is not an imposed structure but is a reflection of the structure found in tasks in the real world.
- A TKS includes knowledge about objects and their associated actions. These objects and associated actions differ in how representative they are of the TKS of which they are part. The main implication of this is that the procedures containing these representative objects and actions are more central to the TKS than other procedures.

As a theoretical construction TKS consists of *Goals, Plans or Goal Substructure, Procedures, Actions and Objects.* For the purposes of computationability, the TKS formalism also includes a Task Taxonomic Substructure. This is based on the objects to be found in the task as it is represented in the TKS and contains their characteristics: features, typicality, instances, centrality; related actions and relationships to other objects, etc.

Notable features of the formalism are the connection between the Taxonomic Substructure and the Procedures of the Goal Substructure (or Plan) via the Objects. In a computer system this allows for greater flexibility and efficiency in the process of locating desired knowledge. Other features are the subgoals relationship to other subgoals which may be both within the TKS in question or across TKSs, thus linking TKSs within a domain. A TKS is related to other TKSs by a number of possible relations such as *within role* and *between role* relations, as well as *further* relations which are the relations that exist between TKSs and the learning processes. All these relations make the TKSs dynamic rather than static structures.

In the work on intention modelling described here TKSs are used as an appropriate task knowledge representation formalism for the depiction of the task related issues regarding the *users* of the system. In this way valuable elements of the *users'* actions are captured and these actions are viewed as part of a wider general structure and not as isolated or non-related activities.

From the brief descriptions given above, it can be understood that CTMs is an approach to user modelling, TKSs a formalism for representing the knowledge a user recruits in order to carry out a task, and that both CTMs and TKSs are approaches to modelling users' task behaviour. As such, both provide useful methodologies for building an operational model of intentions for the purpose of intrusion detection.

3. INTENTION THEORIES

In this section an outline of the most recent work done in the area of intention modelling is introduced, in order to examine the relevant research and to gather all the necessary notions for the further definition of operational IMs that could be used for an estimation of *users' intentions* in computer security systems.

Current theories deal with questions such as 'what are intentions and plans' or 'what characterises the process by which an agent recognises the intentions and plans of another agent who is attempting to communicate with him' (Allen, 1990; Bratman, 1990; Cohen et.al. 1990). The objective here is the transformation of such theories and modelling techniques into a more applicable form bearing in mind a number of assumptions and simplifications originating from the nature of the specific group of applications under consideration.

sequences of events *e* himself, after which p holds. Here there is the requirement for the agent to think he is about to do something (event sequence *e'*) bringing about p.

From these definitions the extracted properties of INTEND are: i. intentions provide a 'screen of admissibility' for adopting other intentions; ii. agents 'track' the success of their attempts to achieve intentions; iii. if an agent intends to do a then he believes it is possible to do action a, sometimes he believes he will in fact do a and he does not believe he will never do a. and iv. agents need not intend all the expected side-effects of their intentions.

A complementary approach adopted by Kautz (1990) i.e. a theory of plan recognition has been considered as an aid to modelling intentions.

Plan recognition problems can be classified as cases of intended or keyhole recognition (Cohen et.al. 1982). While in the first case the agent activities are deliberately structured in order to make clear to the observer the agent's intentions, in the second case, which is the case of concern, it is not easy to identify the real intentions of the agent. The approach described here considers the second case. Also the observer is considered to have complete knowledge of the domain. In this theory the models of the observation statements are related to the models of conclusions and the mechanical transformation of the observed statements to a form from which reasonable conclusions could follow is described. For this description the observations have been considered as *descriptions of events*, the observer's knowledge is represented by a collection of a restricted-form first-order axioms, the *event hierarchy* and the results of the recognition process is a description of the *end events*. The event hierarchy represents the abstraction, specialisation and functional relationships between various kinds of events and recognition is considered as the problem of classifying the *end events* that generate a set of observed events.

In the next section the developed IMs are presented as an amalgam of the above outlined views of approaching intention modelling. The proposed framework has been developed as an operational model of user intention with an eye always to implementation issues.

4. THE INTENTION MODELS (IM)

The purpose of this section is to describe a theoretical approach to intention modelling and consequently discuss the implementability of the produced models. IMs model interactive behaviour of the user when he/she uses a computer system, modelling in this way essential characteristics which underlie interactive behaviour. They represent all those knowledge structures related to the jobs for which the system is used, in relation to the inferential, the perceptual, the planning, and the attentional processes of the users. The consistency requirements of the system use, the inter-system constraints, and the interface objects and their behaviour are also elements of this modelling scheme. The main constraining factors, assumptions and characteristics of these models can be summarised as follows:

- Intentions have to be considered in relation to plans, and intention modelling has to analyse plans as particular configurations of beliefs and intentions.
- The relation between observation of actions and conclusions about the intended plans must be defined as a means of plan recognition and consequently intention identification.
- The above plan recognition must be restricted only for the case of keyhole recognition, i.e. the observer cannot assume that the agent is deliberately structuring his activities in order to make his intentions clear.

- The observer has complete knowledge of the domain of concern.
- The relevant knowledge has to be structured as a summary representation of the different types of knowledge that are required to carry out a task or tasks.
- Task Analysis is a prerequisite stage in the investigation of what users do when they carry out tasks and how do they perform these tasks and thus a prerequisite for the formation of the information to be modelled.
- The abstraction, the specialisation and the functional relationships between various kinds (and levels) of agent actions have to be structured by a hierarchy, similar to the event hierarchy, which is viewed as a logical encoding of a semantic network.
- There must be an approximate description of the cognitive activity underlying user's task performance, which has to capture the important features of this processing activity.

Generally speaking, IMs can be thought of as a semi-formal notation and are used to describe and represent the intention space around a system use. They can be considered as modelling structures which include a representation of alternative use options, and representation of reasoning for using these options with respect to the underlying intentions of the users. IMs are not simply descriptions of user's actions but a representation of knowledge about and of the use of the system. IMs represent the intention space (task oriented) assembled by set of intention nodes linked by special intentions or behavioural relations.

In an intention modelling system there are the following fundamental entities.

The ***agent*** who is the active entity of the *interaction space* and who generates and/or owns the intentions, the beliefs and the plans and generally a whole structure regarding the tasks he can perform and also performs tasks executing activities.

The ***system*** which is the second entity of the *interaction space* that interacts with the agent.

The ***action structure*** which is a hierarchical/task based representation of the various kinds of actions performed in the *interaction space* and the relationships (functional and structural) of its components. The construction of the action structure is based on the Kautz's *Event Hierarchy* and on the Johnson's TKSs and is described in the sequel.

The ***time framework*** which is a linear, interval based representation of time with well defined relations between intervals such as *between*, *overlaps* etc. (Allen, 1983; Allen, 1984).

The ***observer*** who is the observing entity of the model who observes the interaction of the *agents* with the *system* (the interaction space) and performs *keyhole plan recognition* trying to identify and characterise agents' intentions.

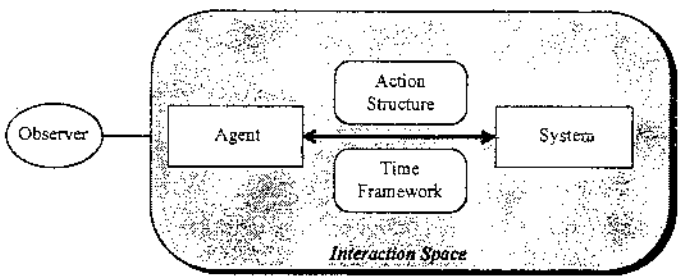

Figure 1 Structural components of the Intention Models (Ims).

Figure 1 shows a structural outline of the components of the IMs where the interaction of the agent with the system can be seen. The action structure and the time framework describe this

interaction. All these constitute the interaction space which is observed by the observer who is trying, through plan recognition techniques, to reason about intentions.

In order to apply these models in a computer system the roles of Agent and system have to be specifically defined. The *Agent* entity stands for every active entity, i.e. a user, a process, an intruder, or the computer system itself. The *System* entity is the corresponding interacting environment for every kind of agent entity. For example, when Agent is a single user then system is the actual computer system which this user is interacting with; when Agent is an intruder then the whole computer is the system and when the computer is the Agent then the whole environment is the system. In other words *system* could stand for any reacting entity that reacts when the corresponding Agent initiates the interaction.

The *Agent-System* interaction which takes place in the interaction space happens whenever there is a task (or tasks) that has to be performed in order for the Agent to achieve his/its goals. The intentions underlying the Agent's actions that are expressed by the revealing of a certain characteristic behaviour during this interaction have to be identified by the observer.

In the next subsection, the description of the Action structure entity of the models is described and the necessary knowledge is further specified.

4.1 Action Structure

As already mentioned this is the part of the model where the actions performed in the interaction space together with their hierarchical and task based structure are represented.

The main structural components of this structure are the Tasks; the Intentions; the subclasses of these tasks and intentions at various levels (i.e. Subtasks and Sub-intentions), the Plans; the Procedures; the Actions; the Objects; the Sequences; and finally the Relationships and Associations amongst the above.

Tasks are performed in order to achieve one or more goals i.e. in order to serve one or more intentions. There are more than one intention for every task but there are also more than one tasks that could satisfy an intention. There is a number of relations that determines the correspondence of a task to the corresponding intentions and relations that correlate more than one task to an intention. These relations depend on the generic characteristics of the task and vary from case to case depending on the particular user who is performing the task. Plans are the alternative means of task conjunction formed in order to satisfy an intention.

Sub-tasks and sub intentions are also same to tasks and intentions in their nature. They are tasks themselves and just describe phenomena at a lower level of description. Usually tasks are composed of more than one sub-tasks which correspond to relevant sub-intentions.

Actions and objects are the elementary components of these IMs and on the basis of these whole IMs are assembled. Actions are performed upon objects and combinations of conditioned actions upon objects make up procedures. Actions upon objects and procedures are the components of the IMs that are identified at the lowest level of description, i.e. correspond to users' activities.

Relations are the elementary linking components of IMs. There are two categories of relations according to their connection with intentions. The relations between actions, objects and procedures which determine the way they are combined to build sub-tasks and tasks and the relations which associate tasks and subtasks with intentions and sub-intentions.

For applicability reasons the following knowledge is required:

- Knowledge about the static characteristics of the interaction space i.e. the characteristics of the *Agent* and the *system*. The possible contents of the relevant knowledge base depend on the type of *Agent* and *system* that will be considered and contains a description of their important aspects. The degree of detail and the range of the contents of this knowledge base is subject to the aims and considerations of the *Observer*.
- Knowledge about the dynamic characteristic of the interaction space. This knowledge is structured in a Task based form and it contains knowledge about the tasks carried out in the system along with all their task structure. The relevant knowledge base will contain actions upon objects of the system, procedures, subtasks and tasks, subgoals and goals, and the goal substructure (plans). Also the Taxonomic substructure as this is defined in the TKS theory, has to be represented in this knowledge base.
- Knowledge that represents the cognitive activity underlying task performance in the interaction space. Theoretically this is the knowledge necessary for the transformation of the real world situation in an abstract way where a number of relationships could be applied. This knowledge is utilised in order to model user cognition resulting in the mapping of this modelling back to the specific situation. Three basic types of knowledge are required: knowledge that maps descriptions of users' actions, tasks and goals, onto identifiers that represent entities of the IMs; knowledge that operates on these entities to characterise properties of human information processing activity; and knowledge that maps entities in the IMs to characteristics of human behaviour. In this way a hierarchical goal formation is created which in relation to the action specification and the action execution represented in the dynamic characteristic of the interaction space mentioned above constitutes the core knowledge of the IMs.

This knowledge is implemented through a set of relationships within and between the entities of the IMs and this implementation is described in detail in the next section

5. TOWARDS IMPLEMENTATION (UII ARCHITECTURE)

The first effort at implementing these IMs for a real application was for the creation of UII (User Intention Identification) system, a system for the detection of anomalous behaviour by reasoning about the characterisation of the intentions of users. This system views the users of a system as using it in order to achieve certain goals by performing various tasks. This system was developed as an autonomous module within SECURENET (Spirakis et.al. 1994), a European Union funded project which tries to protect open networks by detecting malicious attacks primarily against their management. The major characteristics of the malicious attacks that UII module aims to prevent, are those cases where malicious tasks are composed by legal events. Since the basic actions for these tasks are allowable they cannot be detected by simple matching mechanisms. The examination of the whole rationality behind the execution of these basic actions has to be considered in relation to the general goals these actions are trying to fulfil (when composed to form tasks). Reasoning about the deviations observed in the execution of actions within a task in relation to the normal task execution (under the general goal-oriented constraints) offers an indication of the suspiciousness of the observed behaviour.

In other words the UII system aims to detect a certain type of malicious attacks by

characterising the normality of the behaviour of the users. Based on the knowledge representations described above the system represents expected normal behaviour and compares it against the current, observed behaviour. Deviations correspond to abnormality and a certainty factor is calculated as an indication of the importance of this abnormality.

As a system, UII receives as input the action that the user is executing, the time of execution and the user identity and gives as output an indication of the suspiciousness of the observed behaviour and an evaluation of executed tasks.

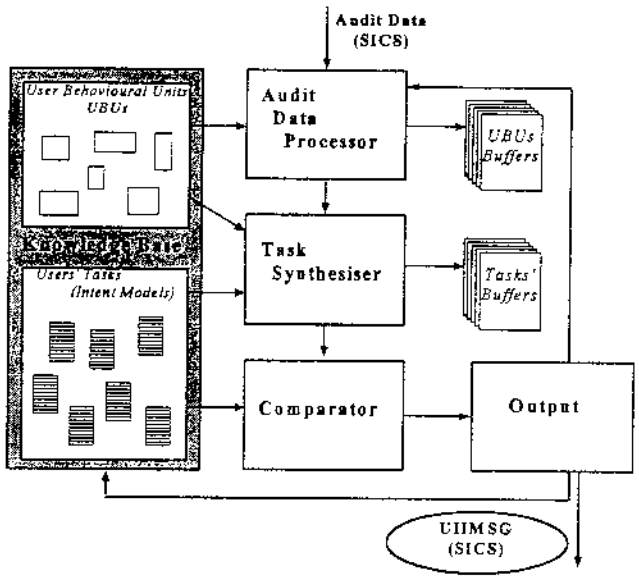

Figure 2 Architecture of the UII module.

Figure 2 gives an outline of the architecture of the User Intention Identification Module and its components which are discussed in the sequel.

The main objective of the User Intention Identification Module is the semantic interpretation of the collected data that correspond to the network users' actions. According to this the Intention Identification module must be able to perform four major tasks.

- Pre-processing of the input data.
- Task Synthesis.
- Behaviour Comparison.
- Output.

There are four functions corresponding to these four tasks:

Audit Data Processor (ADP)

The input data for the UII module is provided by an auditing system. This auditing system produces a sophisticated shared list of audit trails. This list remains open for the pre-processing of the audit data. Information about user identity and his/her activity executed in the computer system are the constituents of this audit data. The audit data processor plays the role of a filtering mechanism, reducing the amount of information that has to be processed further by the other functions of the module.

Task Synthesiser

This function implements the first steps of the semantic interpretation of the observed behaviour of the computer users. This interpretation is the basic function of the Intention

Identification module. TKS entities processed by the ADP function are characterised here as task components. Sets of rules are triggered to determine the task(s) to which these TKS entities can be attached. These rules represent the relations of the observed actions within and between the possible tasks executed by the observed entity and these relations are represented as pre-conditions or post-conditions, sequence relationships, and associations to wider tasks and goals etc.

The basic idea behind this function is that a first comparison has to be made in order to provide evidence about the possible tasks the user might be performing. This is necessary in order to let the comparator function work with the relations between these potentially performed tasks and their related goals. The examination of the TKS entities in this function is made primarily through the validation of an action as a part of a task.

Comparator

This is the main inference mechanism of the module. It makes all the semantic interpretation of the observed network-user behaviour and produces the intrusion hypotheses. The basic idea behind this function is that by reasoning about deviations from the normal task execution and by reasoning about the similarities between the executed and the allowable tasks, estimations regarding the suspiciousness of the performed activity could be made. These hypotheses are formed in the output function.

The reasoning mechanism of this function is realised with sets of rules that represent the relationships between the various elements of the IMs (TKS entities). The TKS knowledge structure is used as the knowledge base for the comparator function i.e. the knowledge about task execution. There are three major aims for this function.

- It tries to combine the relations between the TKS entities in order to decide about the normality and validity of a Task execution.
- It tries to combine inter-task relations and associations in relation to relevant goals in order to decide whether a combination of task execution is suspicious, and finally
- It tries to combine the goal substructure of the various TKS structures of tasks executed with within and between role relations in the tasks execution. This is the most complex aim of the comparator function.

The basic operation of this function is performed by a recursive process which elaborates the TKS structure representation in relation to the asserted facts that correspond to observed behaviour and derive from the previous two functions.

The output of this function is the creation of a hypothesis that classifies a suspicious situation observed and provides the data for a representative description of the reasoning that produced that hypothesis.

Output

This function plays the role of the explanation generation part of an expert system. Every time that a hypothesis is generated by the comparator function the output function selects the necessary information relevant to this suspected intrusion from the TKS based profile and offers it as a means of explanation.

For the development of the Expert System application the CLIPS expert system shell (NASA, 1993; Giarratano, 1994) has been used. CLIPS is an expert system tool which provides a complete environment for the construction of rule-based and object-based expert

systems. It facilitates a wide variety of knowledge representation techniques and supports three different programming paradigms: rule-based, object-oriented and procedural. Rules are grouped into rule sets composing modules. These modules are triggered and fired according to firing strategies by a focusing mechanism. This can help in performing sets of actions in given situations and also allows complex systems to be modelled as modular components.

The specific application presented here demands complex knowledge representation abilities as well as strong rule grouping handler mechanisms. This is the main reason for choosing CLIPS. Other requirements met by Clips are speed, high portability and the capability for easy integration with external systems. CLIPS also uses templates which can have more than one multifield slots and the function arguments and the slot values can be checked by static and dynamic constraints. Additionally CLIPS supports modular design and development (modules) which allows a knowledge base to be partitioned and a set of constructs to be grouped together restricting the access of the constructs by other modules and providing execution control. The knowledge domain in this problem consists of the knowledge bases described in the IMs. i.e. knowledge about the user-actions and about the relations between actions within tasks and relations between tasks and goals.

The knowledge about actions and tasks is in the form of TKS so the framed-based representation mechanism is used. TKS's substructures are represented as separate frames and relevant instances are asserted for every specific task and specific user. The relations between actions within tasks are represented as slot values in a separate frame. There is also sets of rules that are triggered in order to combine and utilise these relations.

It was very important for the program to respond with a good speed when it is running, because the audit mechanism produces a large number of input records. For this reason the Audit Data Processing function was written in C, a fast algorithmic language. This function plays the role of a prefiltering mechanism and the role of an intermediate buffer that adapts the audit records production rate to the expert system processing possibilities.

For the same reason, the use of modules along with control facts was adopted to tune the system's speed. This combination is very helpful because it allows the utilisation of both the control mechanisms and the modular design. In addition the module construct can be used to control the execution of rules. Each module has its own agenda instead of just a global one. Execution can then be controlled by selecting which module's agenda is selected for rule firing and execution.

The module *Relationships*, is one of the central modules of the expert system. In this module the relations between the structural components of a TKS are elaborated to produce the Task synthesis and consequently after the comparisons the final hypotheses about user behaviour. The design and implementation of the relations and the relationship module are further described in the sequel.

5.1 The design and implementation of relations

There are two basic types of relations. Relations that when satisfied, support the evidence that a task is executed (*supporting*) and relations that when satisfied are against the evidence that a task is executed (*contrasting*). The negation of a supporting relation may be a contrasting one depending on the actual relation. In their nature these relations correspond to pre-conditions, post-conditions, sequential and existential relationships within the task structure.

A relation (rel_i) is defined as an ordered triad:

$$rel_i : (act_a, act_b, T).$$

which relates actions act_a and act_b within a task execution. When such relations are satisfied the value of the certainty of the evidence that the task is executed is altered (increased/decreased for supporting/contrasting relations). For example, if rel_i is a precondition it shows that act_a is executed when the task T is performed under the constrain that act_b has already been executed within the task T. The definitions given in Appendix 1 are necessary in order to define the relationships and the reasoning based on them.

In the Relationship module two sets of rules are represented; the first set of rules refers to the relations which have been defined between the actions within tasks and between tasks. The other set refers to the rational, based on which the hypotheses are produced.

Whenever the first set of rules is fired the certainty of execution for one or more tasks is effected. In essence, when there is evidence that a task is executed because of the satisfaction of a supporting relationship then the certainty factor regarding the specific task for the specific user is modified according to the supporting factor. On the other hand when a contrasting relation is satisfied then the evidence that the task is executed is small and accordingly the certainty factor is lowered according to the contrasting factor. The second set of rules produces hypotheses that correspond to information about the type of the problem that has been detected by the UII.

Appendix 2 gives a detailed description of the design of the relationships for a specific relation. As an example a verbal and a CLIPS-like description of a rule follows:

(1) IF action α belongs to the set of tasks which are related to action α ($T\alpha_{rel_1}t$)
and there is historical evidence of possible execution for these tasks
and the set of actions which are preconditions of α is not the empty set
and in some of these tasks the preconditions of action α have been satisfied
and in the rest of the tasks, at least one precondition of α has not been satisfied

THEN the value of the certainty of tasks that have all their preconditions satisfied is increased by the supporting factor sf
and the value of tasks that have not all their preconditions satisfied is decreased by the contrasting factor cf

```
(defrule rule_22_24
 (audit (already-used FALSE)(action ?act)(usr-code ?usr)(time ?time)(argum ?arg))
 (tasks (name-task ?taskx)(action ?act)(relation rel1)
          (pre-actions $?actions&: (< 0 (length ?actions)))))
 ?b<-satisfy (user ?usr)(task ?taskx)(satisfied-actions $?sat-actions)(certainty ?c))
=>
 (bind ?count 0) (bind ?lengthx (length $?actions)) (bind ?loop ?lengthx)
 (while (< 0 ?loop) do
  (bind ?actx (nth ?loop $?actions)) (bind ?result (member ?actx $?sat-actions))
  (if (eq ?result FALSE) then (break))
  (bind ?loop (- ?loop 1)) (bind ?count (+ 1 ?count)))
 (if (eq ?count ?lengthx)  then
  (assert (satisfy (usr-code ?usr) (satisfied-actions $?sat-actions ?act)
     (status PRESENT) (rel rel1) (task ?taskx)(certainty (+ (* ?c (- 1 sf)) sf)))))
```

5.2 Hypotheses

The last module of the expert system, *Results,* concentrates on the templates which show for which tasks there is currently evidence of possible execution by a specific user. By reasoning about the current user's work this module produces hypotheses about the suspiciousness of

this work. The hypotheses that are generated by the system have a code number which is composed by two parts in the form: XXYYYY. XX is a number that indicates which module produced that hypothesis (always 70 for the UII module) and YYYY represents the actual hypothesis number. Some hypotheses produced by the UII module are described below:

The hypothesis 700100 is produced when the executed action is not contained in any structure of tasks in the Knowledge Base. The additional information with this hypothesis is the active history of the user which doesn't fit with the present action.

The hypothesis 700200 is produced when there is strong evidence that the executed action makes a major contrast with the already executed tasks. The additional information with this hypothesis is the active history of the user which doesn't fit with the present action.

The hypothesis 700300 is produced when during the execution of a specific task there is a crucial deviation in this execution. The UII module has found strong evidence that the user is performing a specific task but during this task performance there are a number of actions that don't fit any other task and they form a deviation in the execution of that task. Taking into account both the importance (in terms of security) of this task and the category of the specific user, the overall system has to proceed with the application of the proper countermeasures.

The hypothesis 700400 is produced when there is strong evidence from the history of the user that during the execution of a number of tasks there exist crucial strategy or goal conflicts. These conflicts are identified after the comparison of the current history of the user with the knowledge in the knowledge base about the tasks/goals relations.

These hypotheses are being validated in runs with real-world systems, and it is expected that other instances of these hypothesis types will be elaborated.

6. SUMMARY AND DISCUSSION

In this paper a theoretical approach to intention modelling was introduced and consequently the implementability of the produced models was discussed through the description of an implemented intrusion detection system based on these models. IMs model interactive behaviour of the user when he/she uses a computer system in order to perform certain tasks, modelling in this way essential characteristics which underlie the interactive behaviour. They represent all those knowledge structures related to the jobs for which the system is used, in relation to the inferential, the perceptual, the planning, and the attentional processes of the users. The consistency requirements of the system use, the inter-system constraints, and the interface objects and their behaviour are also elements of this modelling scheme. The development of these models is based on current theories of intentions, theories of rational action and plan recognition and on special forms of cognitive and task modelling structure.

The implemented prototype (UII) is an autonomous module build within SECURENET, a EU funded project which tries to protect open networks by detecting malicious attacks made primarily against their management. The major characteristics of the malicious attacks that UII module aims at are those cases where malicious tasks are composed by legal events. Since the basic actions for these tasks are allowable they cannot be detected by simple matching mechanisms. The examination of the whole rationality behind the execution of these basic actions has to be considered in relation to the general goals these actions are trying to fulfil (when composed to form tasks). Reasoning about the deviations observed in the execution of actions within a task in relation to the normal task execution (under the general goal-oriented

constraints) offers an indication of the suspiciousness of the observed behaviour.

The module is now in its testing phase running in a close to real environment i.e. a real environment where only a small number of tasks are monitored. This is because the necessary knowledge elicitation process has been performed only for this subset of tasks. An extensive knowledge elicitation phase has been planned in order to cover the total of the task performed in the target computer system.

7. REFERENCES

Allen J.F. (1983) Maintaining Knowledge about Temporal Intervals. *Communications of the ACM*, **26**, 832-843.

Allen J.F. (1984) Towards a General Theory of Action and Time. *Artificial Intelligence* **23**, 123-154.

Allen J.F. (1990) Two Views of Intention: Comments on Bratman and on Cohen and Levesque, in *Intentions in Communication* (ed. R. Cohen, J. Morgan, and M. Pollack), 71-76, MIT Press.

Barnard P.J., May J. and Green A.J.K. (1991) Report on the Design Capabilities and Potential of an Expert System Modeller Based Upon Cognitive Theory. *ESPRIT Basic Research Action 3066, Amodeus Project D13.*

Barnard P.J. and May J. (1993) Cognitive Modelling for User Requirements in *Computers, Communication and Usability: Design issues, research and methods for integrated services* (ed. Byerley P.F., Barnard P.J. and May J.), Elsevier, Amsterdam.

Barnard P.J., Wilson M. and MacLean A. (1988) Approximate modelling of cognitive activity with an Expert System: A Theory Based Strategy for Developing an Interactive Design Tool. *The Computer Journal*, **31**, 445-456.

Bonnie E. J. Alonso H. V. and Allen N. (1994) Towards real-time GOMS: a model of expert behaviour in a highly interactive task. *Behaviour and Information Technology*, **13**, 255-267.

Bratman M. (1990) What is Intention in *Intentions in Communication* (ed. Cohen R., Morgan J., and Pollack M.), 15-32, MIT Press.

Card S..K., Moran T. P. and Newell A. (1980) Computer text editing: An information-processing analysis of a routine cognitive skill *Cognitive Psycology*, **12**, 32-74.

Card S.K., Moran T.P. and Newell A. (1983) *The Psychology of Human-Computer Interaction*, Hillsdale, Lawrence Erlbaum Associates, N.J.

Cohen P.R., Perrault R. and Allen J. (1982) Beyond question-answering in *Strategies for Natural Language Processing* (ed. W.Lehnert and M. Ringle), Erlbaum Associates, Hillsdale, N.J.

Cohen P.R., and Levesque H.J. (1990) Persistence, Intention and Communication in *Intentions in Communication* (ed. Cohen R., Morgan J., and Pollack M.), MIT Press, 33-70.

Cohen P.R and Levesque H.J. (1990) Intention Is Choice with Commitment. *Artificial Intelligence*, **42**, 213-261.

Debar H., Becker M., and Siboni D. (1992) A Neural Network Component for an Intrusion Detection System, in *Proceedings, IEEE Symposium on Research in Computer Security and Privacy.*

Denning D.E. (1987) An Intrusion-Detection Model, *IEEE Transactions on Softwre Engineering*, **13**.

Dias G., Leviyy K. and Mukherjee B. (1990) Modelling Attacks on Computer Systems: Evaluating Vulnerabilities and Forming a Basis for Attack Detection, in *Proceedings,*

SRI Intrusion Detection Workshop 5, 296-304.

Fox K., Henning R. and Reed J. (1990) A Neural network approach towards intrusion detection, in Proc. of the 1990 Symposium on Research in Security and Privacy, 125-134.

Giarratano J. and Riley G. (1994) *Expert Systems: Principles and Programming.* PWS Publishing, Boston, MA., 2nd. edition.

Jackson K.A., Gubois D.H., and Stallings C.A. An Expert System Application for Network Intrusion Detection in *Proceedings, 14th Nat. Computer Security Conference*, 215-225.

Javitz H. and Valdes A. (1991) The SRI IDES Statistical Anomaly Detector, in *Proceedings, 1991 IEEE Symposium on Security and Privacy*, Oakland, California.

Johnson P., Johnson H., Waddington R. and Shools A. (1988) Task-related Knowledge Structures: Analysis, Modelling and Applications, in *People and Computer IV: From Research to Implementation*, (ed. D. M. Jones and R. Winder), Cambridge Univ. Press, 35-62.

Johnson P. and Johnson H. (1991) Knowledge analysis of Tasks: Task Analysis and Specification for Human-computer Systems, in *Engineering the Human-computer Interface* (ed. A. Downton), London, McGraw Hill.

Johnson P. (1989) Supporting System Design by Analyzing Current Task Knowledge. in *Task Analysis for Human-Computer Interaction*, (ed.) D. Diaper, Ellis Horwood, 160-185.

Kautz H. (1990) A Circumscriptive Theory of Plan Recognition, in *Intentions in Communication*, (ed. Cohen R., Morgan J., and Pollack M.) MIT Press.

Kieras D and Polson P.G. (1985) An approach to the Formal Analysis of User Complexity, *International Journal of Man-Machine Studies*, **22**, 365-394.

Lunt T. F. (1988) Automated Audit Trail Analysis and Intrusion Detection: A Survey, in *Proceedings of the 11th National Computer Security Conference*, Baltimore, MD.

Lunt T.F. (1993) A Survey of Intrusion Detection Techniques, *Computers and Security*, **12**, 405-418.

Lunt T. F. et. al. (1992) *IDES final technical Report*, SRI.

Lunt T. F. (1990) Using Statistics to Track Intruders, in *Proceedings of the Joint Statistical Meetings of the American Statistical Association.*

Mansur D. (1988) Network Security Monitor, in *Presentation Notes for the IDES Workshop 3.*

NASA (1993), *Clips Programmer's Guide, Version 6.0*, JSC-25012, NASA Johnson Space Center, Houston, TX.

Sebring M., Shellhouse E., Hanna M. and Whitehurst A. (1991) Expert Systems in Intrusion Detection: A Case Study, in *Proceedings, 11th Nat. Computer Security Conference*, 215-225.

Shieh S. W. and Gligor V. D. (1991) A Pattern-Oriented Intrusion-Detection Model and Its Applications, in *Proceedings of the 1991 IEEE Symposium on Research in Security and Privacy*, USA.

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

Spirakis P., Katsikas S., Gritzalis D., Allegre F., Darzentas J., Gigante C., Karagiannis D., Kess P., Putkonen H. and Spyrou T. (1994) SECURENET: A Network-oriented Intelligent Intrusion Prevention and Detection System *Network Security Journal*, vol. **1**, no **1**, Nov. 1994. (Also in IFIP SEC94, Proceedings of the 10th International Conference on Information Security, The Netherlands 1994)

Vaccaro H. S. and Liepins G. E. (1989) Detection of Anomalous Computer Session Activity, in *Proceedings of the 1989 IEEE Symposium on Security and Privacy.*

Appendix 1.

sf	$sf \in (0, 1]$ a factor that supports the evidence that a task t is executed
cf	$cf \in [0, 1)$ a factor that contrasts the evidence that a task t is executed
C_t	$C_t \in [0, 1]$ certainty value corresponding to the evidence that task t is executed
$x \oplus = y$	$x := x(1-y)+y$
$x \otimes = y$	$x := xy$
α, β, γ	actions
A_{kb}	the set of all the possible actions in the domain
A_u	$\{\alpha \in A_{kb} \mid \alpha$ has been executed by u$\}$ the set of all the possible actions in the domain which have been executed by user u
t	any task in the domain
$\alpha_{rel_i} t$	action α is related to task t via relation rel_i
T_{kb}	the set of all the possible tasks in the domain.
$T\alpha_{rel_i} t$	$\{t \in T_{kb} \mid \alpha_{rel_i} t\}$, the subset of tasks which are related to action α
$\alpha_{pre}\beta$	β is a precondition of α
P_α	$\{\beta \mid \alpha_{pre}\beta\}$ Set of actions that are precondition of α
$P_{\alpha t}$	Set of actions that are precondition of α within a task t
$(\alpha_{pre}\beta)$	$\exists\beta: \alpha_{pre}\beta$ and the fact that β is a precondition of α is satisfied
$(\alpha_{pre}\beta)'$	$\exists\beta: \alpha_{pre}\beta$ and the fact that β is a precondition of α is not satisfied
T_t	$\{t \mid C_t > 0\}$ subset of tasks for which there is historical evidence of possible execution
T_t'	$\{t, C_t = 0\}$ subset of tasks for which there isn't historical evidence of possible execution
$T_{t(\alpha_{pre}\beta)}$	Set of running tasks for which β is a precondition of α and $(\alpha_{pre}\beta)$
$T_{t(\alpha_{pre}\beta)'}$	Set of running tasks for which β is a precondition of α and $(\alpha_{pre}\beta)'$

Appendix 2.

Conditions	Results	Description
$T\alpha_{rel_i} t = \varnothing$	hypothesis 700100	There is no task related to action α
$T\alpha_{rel_i} t \subseteq T_t \wedge P_\alpha \neq \varnothing \wedge \forall\beta \in P_\alpha : (\alpha_{pre}\beta)$	$\forall t \in T\alpha_{rel_i} t \Rightarrow C_t \oplus = sf$	There are only running tasks related to action α and all their preconditions imposed by α are satisfied.
$T\alpha_{rel_i} t \subseteq T_t \wedge P_\alpha \neq \varnothing \wedge \forall t \in T\alpha_{rel_i} t, \exists\beta \in P_{\alpha t} : (\alpha_{pre}\beta)'$	hypothesis 700200	There are only running tasks related to action α and for every one of these tasks there is at least one precondition imposed by α which is not satisfied.
$T\alpha_{rel_i} t \subseteq T_t (\exists t, t' \in T\alpha_{rel_i} t : \forall \beta \in P_{\alpha t} (\alpha_{pre}\beta), \exists \gamma \in P_{\alpha t'} (\alpha_{pre}\gamma)')$	$C_t \oplus = sf$ $C_{t'} \otimes = cf$	There are only running tasks related to action α (and they are more than one). In some of these tasks there is at least one precondition imposed by α which is not satisfied and for the rest tasks all the preconditions imposed by α are satisfied. As a result the certainty of the first category of tasks is decreased by cf and the second category of tasks is increased by sf.
$T\alpha_{rel_i} t \subseteq T_t \wedge P_\alpha = \varnothing$	$\forall t \in T\alpha_{rel_i} t \Rightarrow C_t \oplus = sf$	There are only running tasks related to action α and there is not any precondition imposed by α. As a result the certainty value of the tasks is increased by sf
$T\alpha_{rel_i} t \subseteq T_t' \wedge P_\alpha \neq \varnothing$	hypothesis 700200	There are only non-running tasks related to action α and there are preconditions imposed by α
$T\alpha_{rel_i} t \subseteq T_t' \wedge P_\alpha = \varnothing$	$\forall t \in T\alpha_{rel_i} t \Rightarrow C_t \oplus = sf$	There are only non-running tasks related to action α and there are not preconditions imposed by α. As a result the certainty value of the tasks is increased by sf.
$T\alpha_{rel_i} t \subseteq T_t \wedge T\alpha_{rel_j} t \subseteq T_t' \wedge P_{\alpha t} \neq \varnothing \wedge P_{\alpha t'} \neq \varnothing$	$(\forall t \in T\alpha_{rel_i} t \subseteq T_t, \forall \beta \in P_{\alpha t} : (\alpha_{pre}\beta) \Rightarrow C_t \oplus = sf)$ $(\forall t' \in T\alpha_{rel_j} t' \subseteq T_t', \exists \gamma \in P_{\alpha t'} : (\alpha_{pre}\gamma)' \Rightarrow C_{t'} \otimes = cf)$	There are both running and non-running tasks related to action a. Both of them have preconditions. Thus, in the cases which all the preconditions have satisfied their certainty is increased by sf. Otherwise, if at least one of the preconditions has not satisfied, at these cases the certainty is decreased by cf.
$T\alpha_{rel_i} t \subseteq T_t \wedge T\alpha_{rel_j} t \subseteq T_t' \wedge P_{\alpha t} \neq \varnothing \wedge P_{\alpha t'} = \varnothing$	$\forall t \in T\alpha_{rel_i} t \subseteq T_t', \Rightarrow C_t \oplus = sf$ $(\forall t_1 \in T\alpha_{rel_j} t_1 \subseteq T_t' : \forall \beta \in P_{\alpha t_1} (\alpha_{pre}\beta) \Rightarrow C_{t_1} \oplus = sf)$ $(\forall t_2 \in T\alpha_{rel_i} t_2 \subseteq T_t : \exists \beta \in P_{\alpha t_2} (\alpha_{pre}\beta)' \Rightarrow C_{t_2} \otimes = cf)$	There are both running and non-running tasks related to action α. In the first category there are precondition. In contrast to the first category, the second one has not preconditions. Thus, the certainty of non-running tasks is increased by sf. The second category of tasks is complicated. It should be checked if the preconditions have satisfied or not. So if all of them have satisfied the certainty is increased by sf. Otherwise, it is decreased by cf.
$T\alpha_{rel_i} t \subseteq T_t \wedge T\alpha_{rel_j} t \subseteq T_t' \wedge P_{\alpha t} = \varnothing \wedge P_{\alpha t'} \neq \varnothing$	$\forall t_1 \in T\alpha_{rel_i} t_1 \subseteq T_t \Rightarrow C_{t_1} \oplus = sf$ $\forall t_2 \in T\alpha_{rel_j} t_2 \subseteq T_t'$ $\Rightarrow C_{t_2} \otimes = cf$	There are both running and non-running tasks related to α and in the first category of tasks the action α has not preconditions but in the second category there are preconditions. Thus, the certainty of running tasks is increased by sf but the certainty of non-running tasks is decreased by cf.
$T\alpha_{rel_i} t \subseteq T_t \wedge T\alpha_{rel_j} t \subseteq T_t' \wedge P_{\alpha t} = \varnothing \wedge P_{\alpha t'} = \varnothing$	$\forall t_1 \in T\alpha_{rel_i} t_1 \subseteq T_t \Rightarrow C_{t_1} \oplus = sf$ $\forall t_2 \in T\alpha_{rel_j} t_2 \subseteq T_t' \Rightarrow C_{t_2} \oplus = sf$	There are both running and non-running tasks related to action α and there are not preconditions (in none of them) imposed by α. As a result, the certainty value of both cases is increased by sf.