Application of Artificial Intelligence Techniques in Network Management Security: An Expert System Application for the Detection of Abnormal Behaviour

T. Spyrou, J. Darzentas, K. Sfetsou

Research Laboratory of Samos
Department of Mathematics
University of the Aegean
Samos, Greece.
e-mail: tsp@aegean.gr

Abstract

This paper presents the conceptual and functional architectures as well as an implemented prototype of an intelligent system for the detection of the intentions of network users. The approach to designing and structuring such a system is based on task analysis and task modelling, treating user intentions as intended tasks which are then used to detect abnormal user behaviour at a high level. This system, UII, is an autonomous software module which is utilised within SECURENET system an integrated system for the network management protection. SECURENET is one of the major RACE research projects in the area of security of management in network environments. UII module plays a complementary role within the SECURENET system, trying to analyse and detect a range of malicious attacks with special characteristics. It is concerned with that type of attacks/intrusions which do not usually consist of illegal actions, but of a set of actions acceptable by the system which at a higher level may form non acceptable task or tasks. This form of intrusion is seen here as part of users' intentions about the use of the system, i.e. the tasks they intend to perform. Task modelling plays a major role in the approach adopted to detect the network user intentions. The approach utilises task analysis and task modelling from cognitive psychology for the elicitation and representation of task knowledge. It also uses functionally a semantic pattern matching in order to synthesise the potential intended task which may be carried out by the user, using a transformed input from audit trails. Reasoning mechanisms examine hypotheses about user intentions for generating recommendations with some explanation to the security officer. The conceptual approach for the design of the system is based on theories of action, while the development of the current version of the system which is based on the resulting architecture was carried out in the frame based environment of CLIPS.

1. Introduction

This paper presents the design and development of a prototype of an expert system application for the detection of certain types of abnormal behaviour in open networks and in particular in their management systems. Networks and in particular their management is open to various types of malicious attacks and intrusions which in most cases consist of few or many illegal acts, detection of which usually triggers appropriate countermeasures. The present work is concerned with the type of attacks/intrusions which do not usually consist of illegal actions, but of a set of actions acceptable by the system which at a higher level may form a non acceptable task or tasks. This form of intrusion is regarded as part of users' intentions about the use of the system, i.e. the tasks they intend to perform. Hence task modelling plays a major role in the approach adopted here to detect the network user intentions. The innovative aspect of this approach is this use of intended task models and it also poses the major difficulties in designing and developing such a system.

The difficulties are due primarily to the fact that the modelling refers to human activities which need to be described at higher levels with semantic description requirements as well as sophisticated relations and that has direct effect on the implementability of a truly operational architecture.

In contrast to other AI approaches [19,20,27] the present work utilises task analysis and task modelling from cognitive psychology for the elicitation and representation of task knowledge. It also uses functionally a semantic pattern matching in order to synthesise potential intended tasks which may be carried out by the user, using a transformed input from audit trails. Reasoning mechanisms examine hypotheses about user intentions for generating recommendations with some explanation to the security officer.

The UII (User Intention Identification) system is an autonomous software module which is utilised within the SECURENET [26,24,25] system: an integrated system for network management protection. SECURENET is one of the major RACE research projects in the area of *security of management* in network environments. UII module plays a complementary role within the SECURENET system, trying to analyse and detect a range of malicious attacks with special characteristics.

The conceptual approach for the design of the system is based on theories of action [1,23] and theories of intentions [10,9,6], while the development of the current version of the system, based on the resulting architecture, was carried out in the frame based environment of CLIPS [21,12].

Section 2 gives the theoretical background about tasks and the modelling approaches to study and represent this task knowledge. GOMS and TKS approaches are introduced and they are the basis for the further knowledge modelling which is described in the next section. The conceptual architecture of the present application is described in section 4 as an approach to intrusion detection and the functional components of the expert system are presented in detail. In section 5 implementation issues are discussed and finally a section with conclusions and discussion follows.

2. Knowledge about Tasks

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 implementation of UII. These are Cognitive Task Models (CTM) [3,4,2] and Task Knowledge Structures (TKS) [13,14,15,16]. Briefly, Cognitive Task Models (CTM) and Task Knowledge Structures (TKS) 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 [3,4,2] 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 [13,14,15,16] provide a rich representation of knowledge associated with task behaviours. Each TKS represents task goal, plans, procedural and declarative (general) knowledge associated with a task, along with objects associated with the task and the actions upon those objects. TKS as formalism has been extended via the Fuzzy Sets theory and used in representing knowledge for decisions as well as specific domain knowledge in Intelligent Systems development [11].

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.

2.1. GOMS- TKS

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 Gard, Moran & Newell [7,8,5]. GOMS is a formalism for representing routine cognitive skill. GOMS stands for Goals, Operators, Methods, and Selection rules. These four elements might be considered to be the basic structural components of any task knowledge that a person might recruit to perform a task. A *goal* is a symbolic structure that defines a state to be achieved, and determines a set of methods by which it may be accomplished. *Operators* are elementary perceptual, cognitive or motor acts, whose execution is necessary to change any aspect of the user's mental state or to affect the task environment. An operator is defined by specific effect (output) and by a specific duration. *Methods* are procedures for accomplishing a goal. A method is a sequence of goals and operators, with conditional tests on the contents of the user's immediate memory and on the state of the task environment.

The work of Kieras and Polson [18] 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.

Although neither Card *et al.* nor Kieras & Polson explicitly claim that people actually possess these task knowledge structures, Task Action Grammars [22] do claim this, but the detail of these knowledge structures and the theoretical or empirical evidence for their existence was weak. 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 [13,14] 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 the following: *Goal*: the state of affairs that a task can produce. *Plan or Goal Substructure:* particular and possible ordering of procedures undertaken to achieve a Goal. *Procedures:* actual procedures for implementing a Goal; it is possible to have alternative sets of procedures and different groupings and/or orderings for the same procedures. *Actions and Objects:* (lowest level) the constituents of Procedures.

These five elements of a TKS are organised as shown in Figure 1.

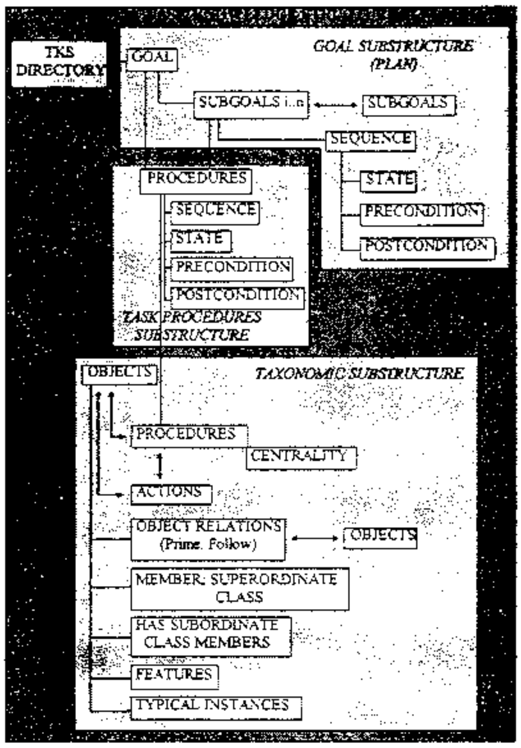

Figure 1 Task Knowledge Structure (TKS).

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.

The proposal in this work is to use TKSs as a 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 *user's* actions will be captured and these actions will be viewed as part of a wider general structure and not as isolated or non-related activities.

3. Modelling Intention-related Task Knowledge

Intention models model interactive behaviour of the user when he/she uses the network system and in particular for management purposes. These models have to cater for the following:

- The interface objects and their behaviour, modelling in this way elements of the interaction between the system and the user and the system characteristics which underlie the interactive behaviour.
- The knowledge structures of the jobs for which the system is used, have to be modelled considering the various task domains.
- The inferential, the perceptual, the planning and the attentional processes of the users.
- The consistency requirements of system use.
- The inter-system constraints.

Intention models can be thought of as a semi formal notation which are used to describe and represent the intention space around a system use. They can be considered as knowledge bases 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.

The intention models are not simply descriptions of user's actions but a representation of knowledge on the use of the system. Intrusion detection systems can utilise this knowledge in order to detect or predict abnormal or inappropriate behaviour or intentions. Intention models represent the intention spaces (task oriented) assembled by a set of intention nodes linked by special intention or behavioural relations.

The main structural components of these models 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 that one tasks to an intention. These relations depend on the generic characteristics of the task and varies from case to case depending on the particular user who is performing the task. Plans are the alternative means of task configuration in order to satisfy an intention.

Sub-tasks and sub-intentions are same as tasks and intentions in their nature. They are tasks and intentions themselves and describe phenomena at a lower level of description respectively. 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 intention models and based on these the whole construction of the intention models is assembled. Actions are performed upon objects and combination of conditioned actions upon objects are the procedures. Actions upon objects and procedures are the components of the intention models that are identified in the audit trails, i.e. correspond to users' activities.

Relations are the elementary linking components of intention models. 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.

Conceptually, in an intention model there are three areas which correspond to three sub-models. These are:

- the Task Knowledge sub-model,
- the Intention sub-model and
- the Relation sub-model

Task Knowledge sub-model is similar in its structure to the Task knowledge Structures (TKS) models as they have been expanded [16].

The Intention sub-model contains a categorisation of users' goals according to their behavioural characteristics.

The Relation sub-model is that part of the intention model that contains the behavioural relations between tasks and intentions of users or categories of users that perform these tasks, in other words a representation of reasoning about users behaviour when performing tasks.

This representation corresponds to a language (the intent specification language) based on which judgement and/or prediction of abnormal or inappropriate behaviour or intentions is carried out. This judgement and prediction are arrived at by elaborating the intentional and behavioural relations that correlate the various nodes of the intention sub-model to the Task knowledge structures sub-model.

4. Overview of the Architecture of UII

The User Intention Identification system is an autonomous module for the detection of anomalous behaviour by reasoning about the characterisation of the intentions of users. This module views the users of a system as using it in order to achieve certain goals by performing various tasks. This module plays a complementary role within the SECURENET system, trying to detect a range of malicious attacks with special characteristics. The major characteristics of the malicious attacks that this module aims at, are these 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 the 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.

There are four fundamental entities which are necessary for the conceptual understanding of the system:

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 who also performs tasks executing activities.

The ***system*** which is the second interacting 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 [17] and on the Johnson's TKSs.

The ***time framework*** which is a linear, interval based representation of time with well defined relations between intervals such as *between*, *overlaps* etc. and

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.

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

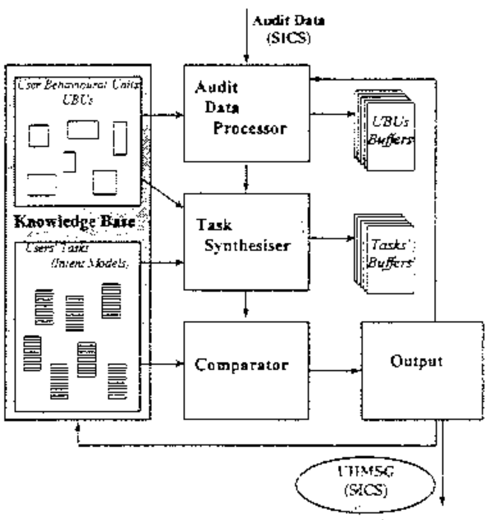

Figure 2. Architecture of the UII module

4.1. Task and Functionalities

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:

4.1.1. Audit Data Processor (ADP)

The input data for the UII is received/provided by the audit file. This audit file remains open during the process of identification of the tasks for possible execution by the UII. This file receives information about the user identity, the actions are executed in the net; this process is continuity. Every time that an identification has finished, a new audit is ready to be read by the UII. Sets of rules are triggered to determine the task or tasks to which this observed action can be attached.

4.1.2. Task Synthesiser

This function implements the first steps of the semantic interpretation of the behaviour of the observed network users. This interpretation is the basic function of the Intention Identification module. TKS entities identified by the ADP function are characterised 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. These relations are represented as pre-conditions or post-conditions, sequence relationships, association to wider tasks and goals etc.

The basic idea behind this function is that a first comparison has to be made in order to discard non valid hypotheses about the execution of tasks. This is necessary in order to let the comparator function work with valid tasks only. The examination of the TKS entities in this function is made primarily for the validation of an action as a part of a task.

4.1.3. 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 of the executed with 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 parts of a TKS. The TKS knowledge structure is used as the knowledge base for that function and knowledge about task execution is represented with that knowledge structure. There are three major aims for that function.

- It tries to combine the relations between the TKS entities in order to decide for the normality and validity in 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 and for the purposes of the demonstrator the basic role relations will be examined.

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 come from the previous two functions.

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

4.1.4. 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 this function selects the necessary information relevant to this suspected intrusion from the TKS based profile and offers it as a means of explanation.

5. Implementation

For the development of the Expert System application the CLIPS expert system shell 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 highly demands complex knowledge representation abilities as well as strong rule grouping handler mechanisms. This is the main reason for choosing CLIPS. Other required advantages of Clips are the speed, the high portability and the ability 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 constraint. 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 knowledge about the user-actions, the relations between actions, the relations between actions within a task, and the relations between tasks.

The knowledge about actions and tasks is in the form of TKS so the framed-based representation mechanism are 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 main modules that are defined in this program are: Main, Open_Files, Relationships, and Results. In the Main module the flow of the program is specified by defining the focus of the various modules according to a predefined priority. The module Open_Files is responsible for the initialisation of the knowledge base which is read and checked each time the programme starts. The communication between the Audit Data Processing function and the rest expert system is managed in this module by the establishment of an endless loop that inputs pre-filtered and pre-processed audit records.

The rest modules implement the reasoning mechanism based on which the synthesis of the executed tasks and the comparisons is performed.

Four of the frames/templates used here are presented in appendix 1. These are the frames that correspond to the representation of the tasks the users possibly execute and to the produced hypotheses which are the actual output of the UII system.

The module Relationships, is one of the central modules of the expert system. Here 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 the 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 task T.

The definitions given in Appendix 2 are necessary in order to define the relationships and the reasoning based on them.

In the Relationship module two set 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 is referred 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 3 gives a detailed description of the design of the relationships for a specific relation.

As an example a verbal description of these rules 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 all the preconditions of action α have been satisfied
THEN the value of the certainty of each task is increased by the supporting factor sf

(2) 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 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

(3) IF action α belongs to the set of tasks which are related to action α ($T\alpha_{rel_1}t$)
and there is not historical evidence of possible execution of these tasks
and the set of actions which are preconditions of α is the empty set
THEN the value of each of these tasks is increased by the supporting factor sf

These rules are implemented in Clips as presented in the sequel:

```
 (defrule rule_22_24
  (audit (already-used FALSE)(action ?act)(usr-code ?usr)(time ?time)
         (argum ?arg))
  (tasks (name-task ?taskx)(action ?act)(relation rell)
         (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)(task ?taskx)(rel rell)
          (satisfied-actions $?sat-actions ?act)(status PRESENT)
          (certainty (+ (* ?c (- 1 sf)) sf)))
   else
     (assert (satisfy (usr-code ?usr)(task ?taskx)(rel rell)
          (satisfied-actions $?sat-actions ?act)(status PRESENT)
          (certainty (* ?c cf))))
     (assert (certainty cf))))
(defrule rule_37
  (audit (already-used FALSE)(action ?act)(usr-code ?usr)
         (time ?time)(argum ?arg))
  (tasks (name-task ?taskx)(action ?act)(relation rell)(pre-actions))
  (not (b-satisfy (user ?usr)(task ?taskx)(satisfied-actions
         $?sat-actions)(certainty ?c)))
=>
  (assert (satisfy (usr-code ?usr)(task ?taskx)(rel rel1)
         (satisfied-actions ?act)(status PRESENT)
         (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. This module by reasoning about the current user's work 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. The 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 comes in 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 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

The conceptual and functional architectures as well as an implemented prototype of an intelligent system for the detection of the intentions of network users is presented. The approach to designing and structuring such a system is based on task analysis and task modelling, treating user intentions as intended tasks which are then used to detect abnormal user behaviour at a high level.

This system, UII, is an autonomous software module which is utilised within SECURENET system an integrated system for the network management protection. SECURENET is one of the major RACE research projects in the area of security of management in network environments. UII module plays a complementary role within the SECURENET system, trying to analyse and detect a range of malicious attacks with special characteristics.

The system is currently being evaluated in a real environment where the tasks are a number of network and system services such as routing, NIS, account management and crone daemons.

Acknowledgement: The work reported on in this paper was funded by the RACE - SECURENET II (R2113) project.

7. References

[1] Allen J.F., Towards a General Theory of Action and Time, *Artificial Intelligence* 23, pp. 123-154, 1984.

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

[3] Barnard P.J. and May J. Cognitive Modelling for User Requirements *In: Byerley P.F., Barnard P.J. & May J. (eds) Computers, Communication and Usability: Design issues, research and methods for integrated services*, Elsevier, Amsterdam, 1993.

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

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

[6] Bratman M. What is Intention. *In: Cohen R., Morgan J., and Pollack M. (Eds) Intentions in Communication*, pp. 15-32, MIT Press, 1990.

[7] Card,S..K., Moran, T. P.,& Newell, A. Computer text editing :An information- processing analysis of a routine cognitive skill. *Cognitive Psycology*, **12**, pp. 32-74, 1980.

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

[9] Cohen P.R., and Levesque H.J. Persistence, Intention and Communication. *In: Cohen R., Morgan J., and Pollack M. (Eds) Intentions in Communication*, pp. 33-70, MIT Press, 1990.

[10] Cohen P.R and Levesque H.J.,Intention Is Choice with Commitment *Artificial Intelligence*, 42, pp. 213-261, 1990.

[11] Darzentas, J., Loukopoulos, N., Darzentas, J., Spyrou, T., Implementing Knowledge Structures for Explanation and Learning, Deliverable no 6, B1 report no. 2, ESPRIT BRA 3160, 1990.

[12] Giarratano J. and Riley G. Expert Systems: Principles and Programming. PWS Publishing, Boston, MA., 2nd. edition, 1994.

[13] Johnson P., Johnson H., Waddington R. and Showls A. Task-related Knowledge Structures: Analysis, Modelling and Applications, in D. M. Jones & R. Winder (eds.), People and Computer IV: From Research to Implementation, Cambridge University Press, pp. 35-62, 1988.

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

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

[16] Johnson, H., Johnson, P., Theoretical Knowledge representations for Supporting Dialogues in Explanation and Learning, Deliverable no 6, B1 report no. 1, ESPRIT BRA 3160, 1990.

[17] Kautz H. A Circumscriptive Theory of Plan Recognition. In: Cohen R., Morgan J., and Pollack M. (Eds) Intentions in Communication, MIT Press, 1990.

[18] Kieras D., Polson G. P., An approach to the formal analysis of user complexity, *Intt. J. Man-Machine Studies*, **22**, pp.365-394, 1985.

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

[20] Lunt T. F. et. al, IDES final technical Report, SRI, 1992.

[21] NASA Johnson Space Center, Houston, TX, Clips Programmer's Guide, Version 6.0, JSC-25012, June 1993.

[22] Payne J. S, Green T. R. G, Task-Action Grammars: A Model of the Mental Representation of Task Languages, *Human-Computer Interaction*, **2**, pp.93-133, 1986.

[23] Pollack M. Plans as Complex Mental Attitudes. In: Cohen R., Morgan J., and Pollack M. (Eds) Intentions in Communication, MIT Press, 1990.

[24] SECURENET Project, Deliverable 3, Overall System Concept, Technical Report R2057.EXP.DR.L.030B1, RACE Programme, 1992

[25] SECURENET Project, Deliverable 5, System Development Plan, Technical Report R2057.EXP.DR.L.050B1, RACE Programme, 1992

[26] Spirakis P., Katsikas S., Gritzalis D., Allegre F., Darzentas J., Gigante C., Karagisnnis D., Kess P., Putkonen H. and Spyrou T. SECURENET: A Network-oriented Intelligent Intrusion Prevention and Detection System, *Network Security Journal,* **1**,**1**,Nov 1994

[27] Spyrou T, Darzentas J. Artificial Intelligence in Information Systems Security, In E. Kiountouzis, N. Alexandris (Eds.), GCS, 1995. In press.

Appendix 1

```
(deftemplate b-satisfy
 (slot user (default ?NONE))
 (slot task (default ?NONE))
 (slot time (default ?NONE))
 (slot certainty (default ?NONE)))
```

```
(deftemplate audit
 (slot time (default ?NONE))
 (slot usr-code (default ?NONE))
 (slot action (default ?NONE))
 (slot argum (default ?NONE))
 (slot already-used))
```

```
(deftemplate Hypothesis
 (slot user (default ?NONE))
 (slot h-code (default ?NONE))
 (slot h-time (default ?NONE))
 (slot h-certainty (default ?NONE))
 (multislot h-parameters ))
```

```
(deftemplate satisfy
 (slot usr-code)
 (slot task)
 (slot rel (default ?DERIVE))
 (multislot satisfied-actions)
 (slot status (default ?NONE))
 (slot certainty (default -1.0)))
```

Appendix 2

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 (history of user)
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'	$\{t \mid C_t = 0\}$ 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 3

Conditions	Results	Description
$T\alpha_{rel_1}t=\varnothing$	hypothesis 700100	There is no task related to action α
$T\alpha_{rel_1}t \subseteq T_t \wedge P_\alpha \neq \varnothing \wedge \forall\beta\in P_\alpha : (\alpha_{pre}\beta)$	$\forall t\in T\alpha_{rel_1}t \Rightarrow C_t\oplus=sf$	There are only running tasks related to action α and all their preconditions imposed by α are satisfied.
$T\alpha_{rel_1}t \subseteq T_t \wedge P_\alpha\neq\varnothing \wedge \forall t\in T\alpha_{rel_1}t ,\exists\beta\in P_{\alpha t}:(\alpha_{pre}\beta)'$	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.
$T\alpha_{rel_1}t \subseteq T_t (\exists t,t'\in T\alpha_{rel_1}t : \forall \beta\in P_{\alpha t}(\alpha_{pre}\beta), \exists \gamma\in P_{\alpha t'}(\alpha_{pre}\gamma)'$	$C_t\ominus=sf$ $C_{t'}\otimes=cf$	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.
$T\alpha_{rel_1}t \subseteq T_t \wedge P_\alpha=\varnothing$	$\forall t\in T\alpha_{rel_1}t\Rightarrow C_t\oplus=sf$	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
$T\alpha_{rel_1}t \subseteq T_t' \wedge P_\alpha\neq\varnothing$	hypothesis 700200	There are only non-running tasks related to action α and there are preconditions imposed by α
$T\alpha_{rel_1}t \subseteq T_t' \wedge P_\alpha=\varnothing$	$\forall t\in T\alpha_{rel_1}t\Rightarrow C_t\oplus=sf$	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.
$T\alpha_{rel_1}t \subseteq T_t \wedge T\alpha_{rel_1}t \subseteq T_t' \wedge P_{\alpha t}\neq \varnothing \wedge P_{\alpha t'}\neq\varnothing$	$(\forall t\in T\alpha_{rel_1}t \subseteq T_t,$ $\forall\beta\in P_{\alpha t}:(\alpha_{pre}\beta)\Rightarrow C_t\ominus=sf)$ $(\forall t'\in T\alpha_{rel_1}t' \subseteq T_t,$ $\exists\gamma\in P_{\alpha t'}:(\alpha_{pre}\gamma)'\Rightarrow C_t\otimes=cf)$	There are both running and non-running tasks related to action a. Both of them have preconditions. Thus, in the running tasks which all the preconditions have satisfied their certainty is increased by sf. Otherwise, if at least one of the preconditions has not satisfied, the certainty is descreased by cf.
$T\alpha_{rel_1}t \subseteq T_t \wedge T\alpha_{rel_1}t \subseteq T_t' \wedge P_{\alpha t}\neq\varnothing \wedge P_{\alpha t'}=\varnothing$	$\forall t\in T\alpha_{rel_1}t \subseteq T_t', \Rightarrow C_{t'}\oplus=sf$ $(\forall t_1\in T\alpha_{rel_1}t_1 \subseteq T_t:\forall\beta\in P_{\alpha t_1}$ $(\alpha_{pre}\beta)\Rightarrow C_{t_1}\oplus=sf)$ $(\forall t_2\in T\alpha_{rel_1}t_2 \subseteq T_t$ $:\exists\beta\in P_{\alpha t_2} (\alpha_{pre}\beta)' \Rightarrow$ $C_{t_2}\otimes=cf)$	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.
$T\alpha_{rel_1}t \subseteq T_t \wedge T\alpha_{rel_1}t \subseteq T_t' \wedge P_{\alpha t}=\varnothing \wedge P_{\alpha t'}\neq\varnothing$	$\forall t_1\in T\alpha_{rel_1}t_1 \subseteq T_t\Rightarrow C_{t_1}\oplus=sf$	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 only of running tasks will be increased by sf.
$T\alpha_{rel_1}t \subseteq T_t \wedge T\alpha_{rel_1}t \subseteq T_t' \wedge P_{\alpha t}=\varnothing \wedge P_{\alpha t'}=\varnothing$	$\forall t_1\in T\alpha_{rel_1}t_1 \subseteq T_t\Rightarrow C_{t_1}\oplus=sf$ $\forall t_2\in T\alpha_{rel_1}t_2 \subseteq T_t'\Rightarrow C_{t_2}'\oplus=sf$	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.