Journal of Theoretical and Applied Information Technology
10th June 2016. Vol.88. No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

IDS IN CLOUD COMPUTING A NOVEL MULTI-AGENT SPECIFICATION METHOD

R.ROMADI, S.EDDAHMANI, B.BOUNABAT

Equip of Information Research and Indexing Documents Texts and Multimedia
ENSIAS, Rabat, Morocco
E-mail : rromadi@gmail.com, eddahmansisaid@gmail.com

ABSTRACT

Intrusion detection systems (IDS) are most efficient way of defending against network-based attacks aimed at computer systems. These systems are used in almost all large-scale IT infrastructures. As part of the migration to cloud services, the situation is even more complex because of the characteristics of cloud, everything is virtual. The number of virtual machines (VM) changes dynamically according to the resource requirement of the requested processing and can be of the order of thousands to tens of thousands. Each VM has an IDS adapted to its services (web server, mail server, ftp server, etc.) and to increase the performance we can use different types of IDS (signature-based IDS, anomaly-based IDS) in one machine. Due to their complex nature, IDS in a cloud environment are extremely difficult to specify and validate. In this paper, we propose a new formal model for the specification and the validation of such systems. This approach considers these Systems as a Multi-Agent System consisting of concurrent reactive agents that cooperate with each other to achieve the desired functionality. In addition, this approach uses formal synchronous specification and verification tools in order to specify and to verify the systems behaviors.

Keywords : *Cloud computing, IDS/IPS, Multi-Agents System*

1. INTRODUCTION

As defined by the NIST[1], the essential characteristics of cloud computing are: Pooling / roommate (multi -tenancy; the ability to respond to a very significant demand (massive scalability); elasticity or ability to simply adapt resources to needs; resource auto-activation (self-provisioning).

Thus the number of instances of VMs running simultaneously can be virtually enormous: thousands to tens of thousands of machines and secondly, it can change dynamically over time.

On the other hand, IDS can implement anomaly and/or signature-based intrusion detection[2]. A signature generally refers to a set of conditions that characterizes the direct manifestation of intrusion activities in terms of packet headers and payload content. Historically, the signature-based method has been the more common of the two methods when looking for suspicious or malicious activity on the network. This method relies on its database of attack signatures and when one or more of these signatures match what is observed in the live traffic, in the case of a IDS, an alarm is triggered and the event is logged for further investigation. Signature-based intrusion detection is only as good as its database, if a signature is not in the database, the IDS will not catch the attack. This is obviously a drawback when you consider that hackers spend a great deal of their time crafting attacks designed to fool signature-based systems. Anomaly-based intrusion detection, on the other hand, takes a more generalized approach when looking for and detecting threats to your network. A baseline of normal behavior is developed, and when an event falls outside that norm, it is flagged and logged. The behavior is a characterization of the state of the protected system, which is both reflective of the system health and sensitive to attacks. In this context, an anomaly-based method of intrusion detection has the potential to detect new or unknown attacks. Like the signature-based method, however, anomaly-based intrusion detection also relies on information that tells it what is normal and what isn't. This is called a profile, and it is key to an effective anomaly-based intrusion detection system.

There are advantages and disadvantages to each method[2] the best-fortified network uses the two methods together to provide the maximum defense for the network infrastructure.

The role of ids in protection against hacking is no longer in doubt . A good IDS that meets the required specifications is a great asset for any

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

organization. Attacks such as Dos/DDoS can cause significant financial losses.

Due to their complex nature, IDS in a cloud environment are extremely difficult to specify. In this paper, we propose a new formal model for the specification and the validation of such systems. This approach considers the intrusion detection System as a Multi-Agent System, i.e a distributed computing system consisting of several autonomous agents (Each IDS is represented by an agent) that coordinate their action in order to fulfill usually joint but also sometimes competitive tasks. Concurrency is further characterized by the need to express communication and synchronization among concurrent agents.

2. SPECIFICATION AND VERIFICATION TOOLS

Validation of an abstract specification of a system is an important aspect of system design. The problem here is how to determine if a reactive system is successful. Our approach for validation is to consider observable behavior as criteria to determine success.

To hit this target, the specified SYNCHARTS (SC) [3] behaviors are automatically translated to the synchronous language ESTEREL [4].

This section will describe all the specification and verification tools used in this work.

- **SYNCHARTS**

SC are introduced by Harel like a visual formalism that provides a way to represent state diagrams with notions like hierarchy, concurency, broadcast communication and temporized state. A SC can be seen like one or several automata which are labeled by ?event[condition]/!action. SC is said to be synchronous because the system reacts to events by instantly updating its internal state and producing actions, the actions produced can trigger in the same instant other transitions, this is named chain reaction causing a set of transitions, the system is always in a waiting state until the condition for a transition is true.

- **ESTEREL**

It's a language, with precisely defined mathematical semantics, for programming the class of input-driven deterministic systems. The software environment of ESTEREL provides high-quality tools, including an editor, compiler, simulator, debugger and verifier.

- **Real-Time Temporal Logic**

Temporal logic has been widely used for the specification and verification of concurrent systems. However, these temporal logics only allow qualitative reasoning about time. Several extensions have been proposed for expressing and reasoning about real-time systems. These include Real-Time Temporal Logic (RTTL), which is based on linear time temporal logic, and allows in addition the expression of quantitative real-time properties (e.g. exact delays or event deadlines).

Example of RTTL Formula

$s_1 \wedge t = T \rightarrow \Diamond (s_2 \wedge t \leq T + 5)$ - If s_1 is true now and the clock reads T ticks, then within T + 5 clock ticks, s_2 must become true. Thus, once s_1 becomes true, s_2 must become true no more than 5 ticks later. This formula can be also written as follows: $s_1 \rightarrow \Diamond_{[0,5]} s_2$ or $s_1 \rightarrow \Diamond_{<=5} s_2$

The formula $s_1 \leftrightarrow s_3$ indicates that events s_1, s_3 are simultaneous. If C(w) is a RTTL formula defining a temporal constraint on an event w, then $w \models C(w)$ means that w satisfies the formula C(w).

3. DAGT BASED HIERARCHICAL STRUCTURE OF IDS

In this paper, the agents are classed as either deliberative or reactive. Deliberative agents derive from the deliberative thinking paradigm: the agents possess an internal symbolic, reasoning model and they engage in planning and negotiation in order to achieve coordination with other agents. Reactive agents don't have any internal symbolic models of their environment, and they act using a stimulus/response type of behavior by responding to the present state of the environment in which they are embedded.

We consider that an IDS can be modeled as a distributed computing system consisting of autonomous Agent.

3.1 Internal Organization of IDS

An IDS is defined by a set of agents, connected to each other by communication interfaces. Thus, its basic structure rests on a two levels tree (fig. 1)

Journal of Theoretical and Applied Information Technology
10th June 2016. Vol.88. No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

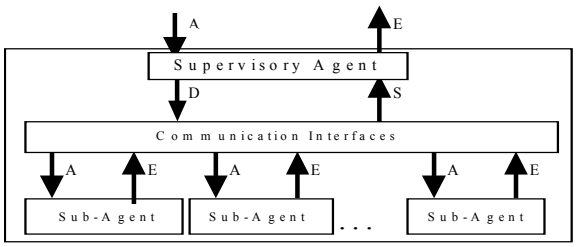

Fig.1. The Internal Organization Of A Reactive System Consists In A Tree That Is Made Up In Parallel Of A Supervisor (Supervisory Agent), Of Two Or Several Sub-Agents Components, And Two Communication Interfaces Between The Supervisor And The Sub-Agents.

Such system interacts with its environment by the means
of:

- Actions exerted by this environment.
- External States emitted to the environment.

Supervisory and Sub-Agents Levels.

The supervisory agent (SDAgt: Supervisory Decisional Agent) is a DAgt controlling the component sub-agents, in order to achieve a goal or to solve a given problem.
This agent will manage the sequences of activation and the definition of the controlled sub-agents objectives. This management depends on:

- The actions exerted by the environment,
- The events generated by the sub-agents activities,
- The temporal constraints specific to any reactive system.

Sub-Agent is a DAgt that can do basic operations required in a step of a given task. For example, it can check memory occupancy rate by a given process, the integrity of a system file, the contents of a log file, the status of a port (open, closed, or filtered), etc. Each Sub-Agent typically wraps calls to a single service or resource, implementing the appropriate error handling and retry logic (subject to a timeout constraint). If the steps in the workflow being run by the SDAgt utilize several services and resources across different steps, each step might reference a different Sub-Agent.

In addition, an IDS can be summarized with a simple SDAgt directly connected to the controlled process. Each sub-agent can be considered as a reactive system. Thus, its internal structure is composed by its own SDAgt, communication interfaces and sub-agents. A sub-agent objectives are to carry out sequences of tasks in response to any temporal constrained action exerted on him by the higher level.

Communication Interfaces. The communication interfaces are of two types: decisional interface (Top/Down) and signaling interfaces (Bottom/Up).

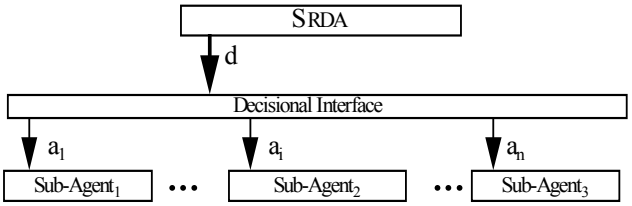

Fig.2. Decisional Interface That Translates A Decision (D) Generated By The Sdagt Into Several Actions (A_i), Each One Of Them Is Intended For A Sub-Agent Of The Lower Level.

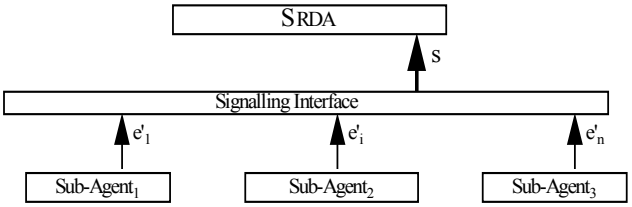

***Fig.3.** Signaling Interface That Synchronizes The External States (E'_i), Sent By Each Sub-Agent, And Emits One Signaling (S) Intended For The Sdagt.*

3.2 Temporal Properties

Through the notion of an action horizon (Ha) of a decision, the time during which the decision remains valid, the DAgt-based specification of an IDS ensures that the elements will have time periods coherent with the decision made by the agent, and coherent with the time periods of decisions made at lower levels of the hierarchy. The higher an agent is in the hierarchy, the greater the action horizon (Fig. 4).

Journal of Theoretical and Applied Information Technology
10th June 2016. Vol.88. No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

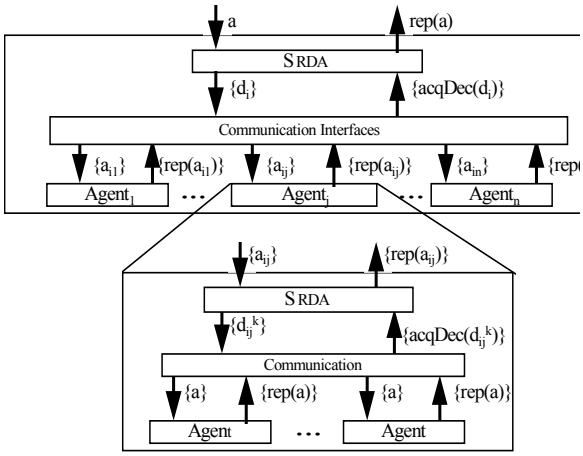

Fig.4. Flow Of Information Inside A SMA Formed By ARDC Agents. The Top-Down Flow Consists In Actions (A, A_{ij}) And Their Associated Decisions (D_i, D_{ij}^k). The Bottom-Up Flow Consists In External States (Rep (A), Rep(A_{ij})) And Their Associated Signaling (Acqdec(D_i), Acqdec(D_{ij}^k)).

The temporal constraints must be checked on each hierarchical level. The recursive character of this structure makes it possible to generalize the results obtained for only one hierarchical level. Thus, we can prove by deduction and according to notations of fig. 4:

$$d_{ij}^k \models C(d_{ij}^k) \Rightarrow a \models C(a) \quad (1)$$

Such system interacts with its environment by the means of:
- Actions exerted by this environment.
- Alarms emitted to the environment.

The use of decisional agent in the modeling, design, and implementation allows us to meet the requirements mentioned previously:

- Flexible. Agent architectures are more flexible, modular and robust than, for example, object-oriented ones. They tend to be open and dynamic as their components can be added, modified or removed at any time.
- Pro-activeness. Intelligent agents are able to exhibit goal-directed behavior by taking the initiative in order to satisfy their design objectives:
- Reactivity. Agents are crucial when operating in an unpredictable environment containing a large number of data sources scattered over multiples sources. If an agent queries an information source and finds no answers to its query, it would then try alternate sources of information until it could come up with a reasonable number of answers.
- Learning. Another important characteristic of autonomous behavior is the ability to enhance future performance as a result of past experiences. Machine learning techniques allow an agent to learn new methods or refine existing ones to meet specific needs.

Communication and cooperation. Intelligent agents are capable of interacting with other agents (and humans) in order to o achieve a common goal.

4. FORMAL DESCRIPTION OF DECISIONAL AGENT

The proposed model of agent consists in putting forward decisional models allowing the representation of objects according to their behavioral aspects and their degree of intelligence.

Definitions. A Decisional Agent (DAgt) is 9-tuple noted < A, D, S, E', O, O', act, dec, sig > where :

- A: Set of actions exerted on the agent. Each action, undergone by an object, represents a possible operation to be carried out on this object in order to achieve a specific goal.
- D: Set of decisions generated by the agent. Each decision is a solution concerning process behavior in the future; each decision is characterized by its action horizon : Ha, the time during which this decision remains valid.
- S: Set of Signaling received by the agent. Each Signaling received by an object, reflects at any given time the state of the controlled tools used to achieve a specific goal.
- E': Set of external states delivered by the agent. Each one represents the object state emitted to the environment.
- E: Set of agent's internal states. Each one indicates the current state of the agent.
- O: Set of agent's internal objectives. Each decision is elaborated in order to achieve an internal objective according to the current external objective and the actual internal state.

Journal of Theoretical and Applied Information Technology
10th June 2016. Vol.88. No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

- O': Set of agent's external objectives which can be achieved. These objectives represent the agent's interpreting of each action.

From a dynamic point of view, the sets above indicate the received events (A, S), the emitted events (D, E') and the internal events (E, O, O').

Decisional Functions. act, dec, and sig are three decisional functions that define the behavior of a DAgt.

act : $A \longrightarrow O'$
$a \longrightarrow o'$ with,

$\forall a \in A, \exists! o' \in O' / o' = act(a) \Rightarrow a \leftrightarrow o'$ (2)

(1) means that the occurrence of an action *a* implies instantaneously the occurrence of its associated external objective *o'* by the function *act*.

dec : $O' \times E \longrightarrow D \times O$
$(o', e) \longrightarrow (d, o)$ with,

$dec(o', e) = (d, o) \Rightarrow [o' \wedge e \leftrightarrow d \wedge o]$ (3)

(2) means that depending of the current external objective *o'* and as soon as the agent is in an appropriate internal state *e*, corresponding decision *d* an internal objective *o*, by the function *dec*, are instantaneously produced.

sig : $O' \times O \times S \longrightarrow E \times E'$
$(o', o, s) \longrightarrow (e, e')$ with,

$sig(o', o, s) = (e, e') \Rightarrow [o' \wedge o \wedge s \leftrightarrow e \wedge e']$ (4)

(3) means that that depending of the current external objective *o'* and the expected internal objective *o*, and as soon as the receipt of a signaling *s*, its associated external state *e'* is instantaneously emitted and the new agent internal state becomes *e*.

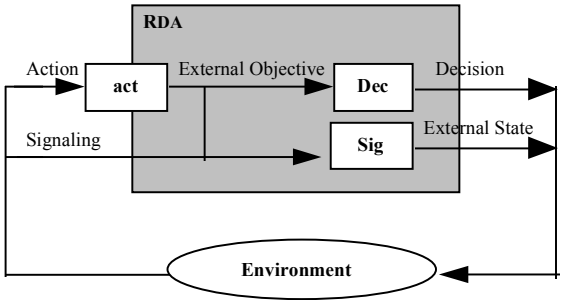

Fig.5. *According To The Formal Definitions Above, Figure.5. Shows The Internal Structure Of A Dagt. Act Interprets An Action As An External Objective, That It Used By Dec And Sig To Generate Agent Appropriate Responses.*

Internal Architecture of a DAgt. This section presents a set of SynCharts which describe the external objective of a DAgt.

External Objectives Manager. A Decisional Agent has an External Objective Manager. It consists in a SynCharts model of the function *act* described above (Fig. 6).

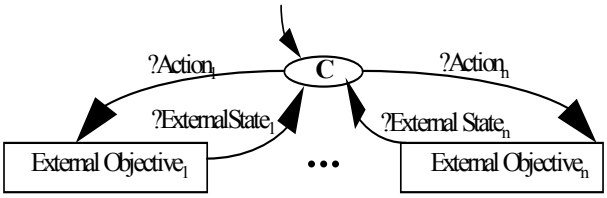

Fig.6. *This Shows A Figure Consisting Of A Synchart Model Of External Objectives Manager. Each State Represents An External Objective Whose Activation Is Started By The Reception Of A Specific Action (?Action), And Terminated By The Emission Of The Acknowledgment External State (!Externalobjective).*

In addition, each operating mode of the agent (normal mode, diagnostics modes, etc.) can be considered as an external objective to be reached. The objectives manager has to maintain the same objective or to change it, according to the occurred fault or failure.

External Objectives Modeling. An external objective is composed by many others SC states corresponding to the associated internal states and internal objectives that are deducted by the functions dec and sig definitions (Fig. 7). The specified SynCharts behaviors are automatically translated to the synchronous language ESTEREL [4]. It's a language, with precisely defined mathematical semantics, for programming the class of input-driven deterministic systems. The software environment of ESTEREL provides high-quality tools, including an editor, compiler, simulator, debugger and verifier.

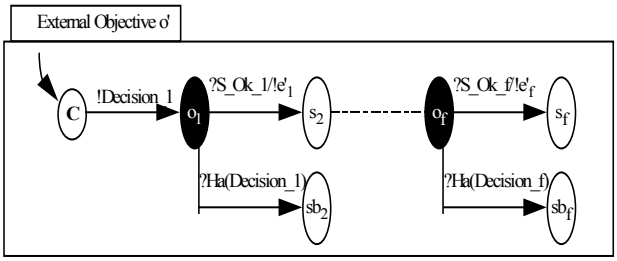

Fig.7. *This Figure Shows The General Syncharts Model Of An External Objective.*

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

The transition (*Internal state* → *Internal objective*) is made by a decision emission (*!Decision*), and the transition (*internal objective* → *Internal state*) is made by a signaling receipt (*?S_OK*), and eventually an external state emission (*!e'*). Internal state C corresponds to the default initial state of a SC model. Internal state and Internal objective are indicated respectively by e_i et o_i. In case of an action horizon exceeding without receiving any acknowledgment signaling, the agent's internal state changes from e_i to eb_i (*breakdown state*).

5. TEMPORAL CONSTRAINTS OF AN DAGT

Decision Temporal Constraints. Each decision is characterized by its action horizon, Ha: the time during which this decision remains valid. So, an occurrence of a decision requires the occurrence of its corresponding acknowledgment signaling, in a delay that doesn't exceed its action horizon.
This defines the following function, acqDec:

$$\begin{aligned} \text{acqDec} &: D \longrightarrow S \times IN \\ d &\longrightarrow (s, Ha) = \text{acqDec}(d), \end{aligned}$$

with

$$\text{acqDec}(d) = (s, Ha) \Rightarrow [\, d \rightarrow \Diamond_{<=Ha}\, s \,] \qquad (5)$$

In the following sections and for any decision d:

- acqDec(d) indicates the acknowledgment signaling of d,
- Ha(d) is the action horizon of d,
- C(d) points out the constraint $[d \rightarrow \Diamond_{<=Ha(d)}$ acqDec(d)]

The temporal property that a DAgt must verify :

$$\forall\, d \in D,\ d \Vdash C(d) \qquad (6)$$

External Objective Temporal Constraints. Each external objective o' is characterized by an acknowledgment specific external state e', that indicates the good ending of o'. this defines a function acq :

$$\begin{aligned} \text{acq} &: O' \longrightarrow E' \\ o' &\longrightarrow e' = \text{acq}(o'), \quad \text{with} \end{aligned}$$

$$\forall\, o' \in O',\ \exists!\, e' \in E' \,/\, e' = \text{acq}(o') \qquad (7)$$

Dynamically, the event acq(o') comes as early as the receipt of the acknowledgment of the last decision generated by o'.

Another function called durMAx is introduced in order to associate to each external objective o' the longest duration of its operations execution.

$$\begin{aligned} \text{durMax} &: O' \longrightarrow IN \\ o' &\longrightarrow \sum_{i=1}^{\text{card}(D_{o'})} Ha(d_i), \text{ where } d_i \in D(o') \end{aligned}$$

By combining the two functions acq and **durMAx**, we can obtain the following constraint:

$$\forall\, o' \in O',\ o' \rightarrow \Diamond <= \text{durMax}(o')\ \text{acq}(o') \qquad (8)$$

i.e. after an occurrence of an external objective o', the agent must generate the corresponding acknowledgment, in a delay that does not exceed durMax(o').

Action Temporal Constraints. Another function rep is introduced in order to define the acknowledgment of an action received by the agent.

$$\begin{aligned} \text{rep} &: A \longrightarrow E' \\ a &\longrightarrow e' = \text{acq}(\text{act}(a)) \end{aligned}$$

C(a) indicates the constraint $[a \rightarrow \Diamond_{<=durMax(act(a))}$ rep(a)], the temporal property that a DAgt must verify is :

$$\forall\, a \in A,\ a \Vdash C(a) \qquad (9)$$

The following assertion can be proved by deduction

$$\forall\, a \in A,\ [\forall\, d \in D(\text{act}(a)),\ d \Vdash C(d)] \Rightarrow a \Vdash C(a) \qquad (10)$$

6. CONCLUSION

The contribution of this paper is to give a new formal approach to deal with specification and formal verification of a monitoring system composed by several intrusion detection systems. The originality is to consider each component of this system as a Reactive Decisional Agent, and to bring together several formal synchronous modeling and validation tools. With its top-down process and its principles of decomposition, this method allows to get a model which is more easily understandable by the user. The SYNCHARTS models are used here in order to describe the reactive agent behaviors. These behaviors will be checked in a qualitative (respectively quantitative) way by the synchronous language ESTEREL (respectively by Real Time Temporal Logic deduction). The

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

mechanism of action horizon, the time during which an agent decision remains valid, is moreover useful to specify temporal performances.

The resulting model can be useful for every application in which it is necessary to include one or several reactive components.

REFERENCES

[1] P. Mell and T. Grance, "The NIST definition of cloud computing," September 2011.

[2] Guide to Intrusion Detection and Prevention Systems (IDPS), NIST special 800-94, by Karen Scarfone PeterMel.

[3] D.Harel, M.Politi, Modeling Reactive Systems with Statecharts : The STATEMATE Approach, Mc Graw-Hill, ISBN 0-07-026205-5

[4] F.Boussinot, and R. de Simone : The ESTEREL language. Proceeding of IEEE, 79(9) : 12931304

[5] Neda Afzali Seresht, Reza Azmi MAIS-IDS : « A distributed intrusion detection system using multi-agent AIS approch » *Engineering Applications of Artificial Intelligence*, *Volume 35*, *October 2014*, *Pages 286-298*

[6] Sapna S. Kaushik, Dr. Prof.P.R.Deshmukh, "Detection of Attacks in an Intrusion Detection System", International Journal of Computer Science and Information Technologies, Vol. 2 (3) , 2011, pp. 982-986

[7] F. Abdoli and M. Kahani, "Ontology-based Distributed Intrusion Detection System", Proceedings of the 14th International CSI Computer Conference (CSICC'09), pp.65-70

[8] Dayong Ye, Quan Bai, Minjie Zhang, "OntologyBased Knowledge Representation for a P2P Multi-Agent Distributed Intrusion Detection System", IFIP International Conference on Network and Parallel Computing, 2008, pp.111- 118

[9] Yu Lasheng , and MUTIMUKWE Chantal, "Agent Based Distributed Intrusion Detection System (ABDIDS)", Proceedings of the Second Symposium International Computer Science and Computational Technology(ISCSCT '09), pp. 134-138