CLASS-W, a Grammar for Security System Development based on Environment Equipment Agents with Windows-Technology NT

Guadalupe Cota[1], Pedro Flores[1] and Joel Suárez[2]

[1] Departamento de Matemáticas, Universidad de Sonora, Hermosillo, Sonora, México, Código Postal 83000, lcota@hades.mat.uson.mx, pflores@hades.mat.uson.mx
[2] Centro de Investigación en Tecnologías de Información y Sistemas, Universidad Autónoma del Estado de Hidalgo, Pachuca de Soto, Hidalgo, México, CP 42000, jsuarez@reduaeh.uaeh.mx

Abstract. The issues regarding the development of security systems and "zero-day threats" are issues of concern to experts, individuals and public or private organizations, since new forms of intrusion used by malicious users or programs that are not recognized by protective tools constantly appear, and they are entered into the computer equipment to make them vulnerable. Taking this into account, and that no evidence was found on grammar for security systems development based on agents, this article describes an original proposal for the CLASS-W grammar, which provides mechanisms to coordinate joint work and feedback on operating system information about services, etc. For testing purposes, a system of agents has been developed to communicate and detect abnormal activity of ports and tasks that are loaded with the operating system.

Keywords: Grammar, Agent, Security, Windows.

1 Introduction

It is to be recognized that although there are software tools that attempt to resolve computer security problems, there is still the problem of the 'zero-day attack', the period which starts from the appearance of 'malware' and is not detected by the security tools database, which persists until the problem that is resolved. This vulnerability is exploited by viruses, worms, or users who seek to infiltrate into the computer systems to commit cyber crimes [1], [2] [3].

On the other hand, we must recognize that although various alternatives for solving this problem exists [4], [5], one of the most important related with it is Intrusion Detection Systems [6], that analyze network traffic and issue security alerts for detect possible "attacks", and nevertheless that these systems have been effective, they have serious drawbacks, both in its operation and in the complexity of their management, that, in addition to generating a large number of false alerts, and for analyzing the information traffic on the network, only based on records of the knowledge base they have, which is why they cannot avoid the damage that occur for unknown events, until the administrator detects them, finds a solution and updates the data, which can

Advances in Computer Science and Engineering.
Research in Computing Science 42, 2009, pp. 65-76

Received 16/03/09
Accepted 27/04/09
Final version 06/05/09

be done in an indefinite period of time, which can range from one day to months or even years [7].

In this work it is suggested to use a different approach that consists of using CLASS_W (Content-Language Windows Security System) grammar, applying the agent of communication language (ACL) [8], defined as the standard by the Foundation for Intelligent Physical Agents (FIPA) [9], and implementing security levels using knowledge base based on the rules pre-defined by the computer network administrator.

For purposes of organization, this document is divided into six sections: the first corresponds to this introduction; the second, in view of the extensive of the grammar, we only mention the most relevant grammatical aspects of CLASS-W; the third describes the functional diagram of agents, which is represented by logic modules and profiles or roles specified; the fourth describes the form knowledge representation used by agents, and the fifth, is an example of the use of grammar using the agent platform JADE (Java Agent Development Framework) [10] and ACL.

2 Grammatical Aspects

CLASS-W has been designed based on the theory related to Context Free Grammar, which will be referenced hereafter as ***CFG*** [11], [12] and will be denoted by G as a quadruple $G = (NT, T, P, \sigma)$, where:

- NT Not terminal symbols.
- T Terminal symbols.
- σ Initial symbol. ($\sigma \in N$).
- P Rules o grammaticals productions where:
  $A \rightarrow a$ and $A \in NT$ and $a \in (NT \vee T)^{+}$.

To make the syntax description of ***CFG***, the Normal Form Extended Backus-Naur (***EBNF***) [13] will be used. The basic notation that is shown in table 1 is to specify the operations of a general nature such as the specification of the initial symbol, terminal and not terminal symbols, etc.

Table 1. Basic notation in *EBNF* for CLASS-W

Notation	Description
|	'Or'
{ t }	0 or more elements t
[t]	Optionality on t
,	'And'
< t >	No Terminal symbol t
::=	Involvement
....	Values to define
N	Maximum number of items

The language of content that will be generated by CLASS-W is implemented on the label ***'content'*** within the framework of the basic format ACL (see Figure 1), where communicative acts ***'sender'***, ***'receiver'*** and ***'content'*** are mandatory elements and the rest is optional.

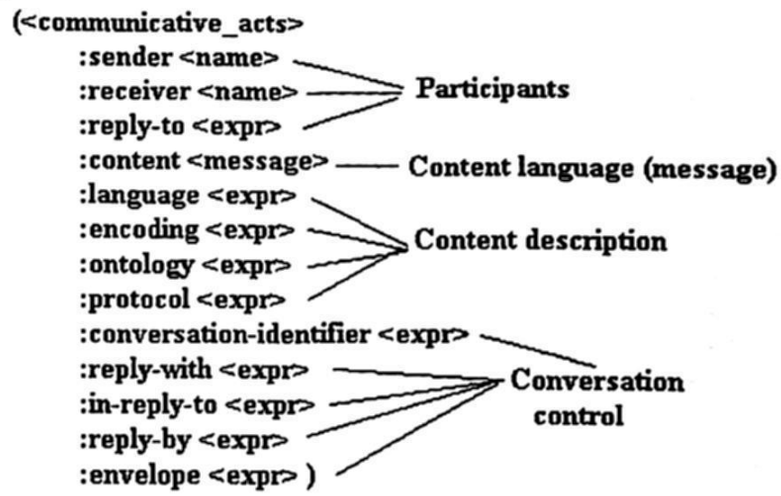

Figure 1. Basic format ACL.

The interpretations of the symbols of *CFG* that are extensive in CLASS-W are:

T Divided into a priori defined terminal symbols and terminal symbols which are referenced by the system security administrator to work on the security context of the operating system Windows-Technology NT, and the remaining will be stored in a database designed for this case (see Figure 3 and 5).

σ Refering to the scheme denoted by ACL ***'content'***, the initial symbol of CLASS-W is constituted by the non-terminal **<content>**

Below are specifications for the units of syntactic CLASS-W, which allow to write a program to control and define transactions sessions, instructions, alerts and safety record.

2.1 General Structure of a Program

```
<content>::= '(' < message > ')'
<message>::= '#'<key>'/' 'message' '/'
          <transmitter_agent> '/' <type_agent> '/'
          <receiver_agent> '/' <type_conversacion> '/'
          <type_transaction> '/'
```

<type_transaction> is the non-terminal symbol, which is used to control the messages that are generated on the events.

2.2 Examples Transactions

Sessions:

```
<session>::='initial_s'<initial_session>
    |'close_s' <close_session> |'estab_s'
    <established_session> |'reject_s' <reject_session>
    |'error_s' <error_session>
<initial_session>::= '/' <agent>
<close_session>::= '/' <agent>
<reject_session>::=  '/' <agent>
<established_session>::= '/'<agent>'/' 'ref' '/' <agent>
<error_session>::=  <string>
<string>::= {<letter>}+
<letter>::= 'a'| ... |'z'|'A' | ... |'Z' |'0' |...|'9'
```

In the context of agents, this conversation can be represented according to the rules of protocol 'request protocol' FIPA [9] (see Figure 2) which is used when an agent asks another for an action and the target agent accepts, rejects or informs on the state of the communication. In cases 1, 4 and 5, when there are errors, rejections or not responses, the issuer must set a reasonable time to reschedule the request.

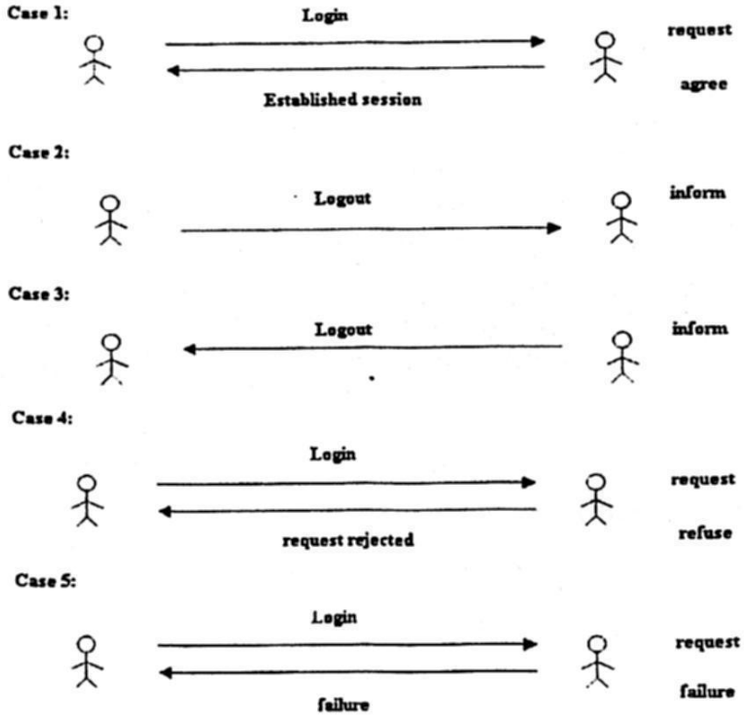

Figure 2. Session conversation represented according to the rules of request protocol FIPA.

Instructions:

```
<instruction> ::= 'stop'<stopr>|'active' <active>
    |'review' <review> |'enable' <enable>
```

```
|'disable'<disable> |'create' <create>
|'delete'<delete> |'modify'<modify>
|'update' <update> |'searchr' <search>
|'clasify' <clasify>
```

The instructions are related to actions that apply specifically on services, ports, applications, processes, policies, initial tasks, registry, network settings, etc, which are defined as 'topics' in the corresponding database.

Alerts:

```
<alert>::= 'alert' '/'<id_alert>'/'<agent>
    '/'<level_security>'/'<ontology_topics>'/'
    <inf_ontology> '/'<action>
<inf_ontology>::= <list_processes> |<list_ports>
    |<list_services>|<list_task_initials>
    |<list_polices>|<list_changes_network>
<result_alert>::=<agent>'/'<id_alert> '/'
    <date>'/'<time>
```

The alerts are controlled by coordinator agents, who define the actions to be taken in cases that are considered harmful and could stand spread through computers on the network controlled by the agents system.

Definitions of Security:

```
<registry_security>::= <id_message>'/'<agent>
    '/'<extent>'/'<ontology_topics> '/'<date>
    '/'<time>'/'<level_security>'/'
    <data_part_extern > '/'{',<action>}+
<level_security>::= 0 |1|2|3|....|N
<action>::= In database.
```

where **<level_security>** can take the values of [1 ,...., N]: {low, medium, medium high, high, ..., N} (see Figure 3) .

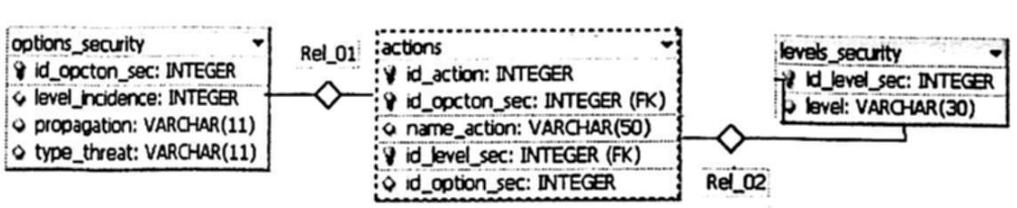

Figure 3. Relational schema for specifications of security options, levels and actions.

Interpreting the values as follows:

a. Type of communication [0-2]: {private, public, selective}
b. Incidence: {0-1, 2-10 and 10 onwards}
c. Propagation: {yes, no}
d. Type of threat: {internal, external}

3 Functional Diagram Agents

The overall operation of the system of agents is based on the detection of events that occur in their environment, collected through sensors, such as network cards and operating system resources [14] (see Figure 4), and is recorded on a blackboard [15] organized by topics, implemented in a database to feed the knowledge that is used in settings of security levels in the managed system.

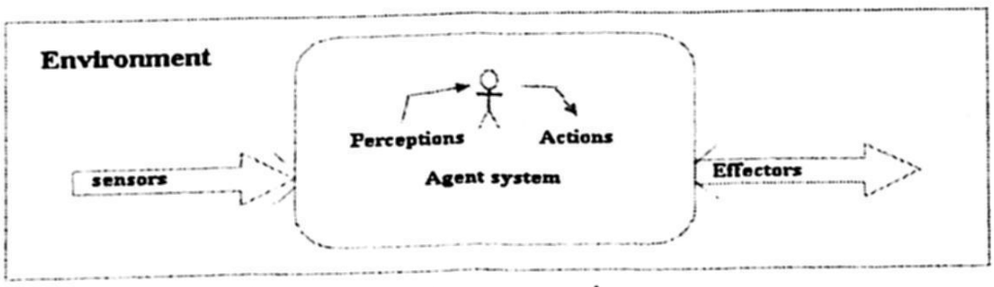

Figure 4. Agent system environment.

As one of the basic elements of CLASS-W is the blackboard, for which was created in database a group of tables (see Figure 5) and the grammar productions that are listed below:

```
<blackboard>::=<registry_blackboard>
    |<notify_blackboard> | <end_blackboard>
    |<wait_blackboard> | <operation_blackboard>
<registry_blackboard>::='registry_blackboard' '/'
    <agent> '/' <ontology_topics>
<notify_blackboard>::='notify_blackboard' '/'
    <ontology_topics>
<end_blackboard>::='end_blackboard' '/'<ontology_topics>
<operation_blackboard>::='id_blackboard' '/'<namber> '/'
    <agent> '/' <ontology_f>'/'<inf_aditional>'/'
    <type_aport> | 'id_blackboard' '/'
    <namber>'/'<agent> '/'<ontology_f>'/'<type_aport>
<type_aport>::='query' '/'<string>|'resp_yes' '/'<string>
    |'resp_no' '/'<string>|'resp_informative' '/'<string>
    |'resp_knowledge' '/'<string>|'proposal' '/'<string>
    |'intention' '/'<string>|'desire' '/'<string>
    |'data_aditional' '/'<string>
```

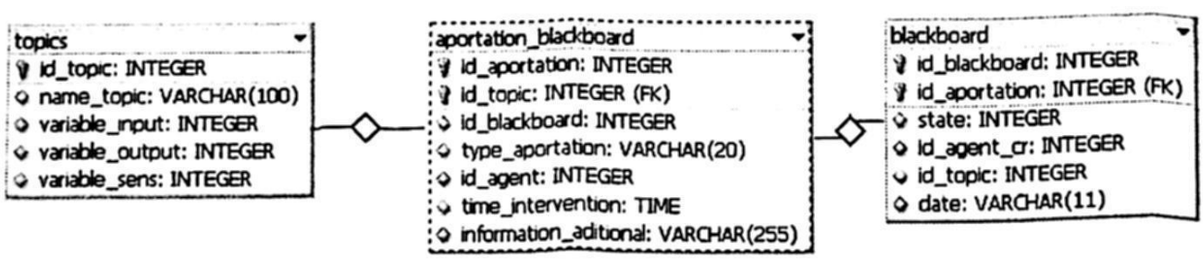

Figure 5. Relational schema for specifications to blackboard.

where the valued to be taken (attribute-table) can be:
(blackboard-state) [1-3]: {initiated, resolved, unfinished}
(blackboard-operation) [1-2]: {read, write}
(aportation_blackboard-type_aportation) [1-9]: {query, yes, no, response knowledge, information, proposal, intention, desire, additional data}
and
(topics-id_topic) [0 to N].
which will be provided by the administrator. The default values are: {processes, ports, services, startup tasks, policies, configuration, ..., N}.

3.1 Roles of Agents

In the hierarchy designed for the system of agents and implementation of CLASS-W, the entities of Administrator Multiagent System (AMS) and Directory Facilitator (DF) are mandatory elements in the scheme of FIPA [9] and JADE [10].

Below is a description of the type of agents that can be created under the specifications of CLASS-W, in addition to the already mentioned from the prior paragraph:

- 'Administrator'*.- Manages the overall operation of the agents system.
- 'Directory facilitator'*.- Manages records of agents and services.
- 'coordinator'.- Coordinates specialized work of multiagente system.
- 'network'.- Monitors performance of network.
- 'local'.- Responsible for monitoring internal function on equipment with different specification.

Note *: Mandatory elements in the scheme of FIPA agent.

4 Representation of Knowledge

As the logic is one of the main bases in the area of mathematics and computer science, and taking into account that the formalization of knowledge and the automation of the forms of reasoning are essential in many scientific development or technology areas, and has been very helpful, especially in the area of Artificial Intelligence [16]. For the implementation of W-CLASS, a section through the non-terminal <rule> is included, which allows representing knowledge using rules type Prolog in a functional scheme of system agents based on a cognitive profile that has the following characteristics [17]:

a) There is a mechanism to interact with each other to agents through a blackboard.
b) It implements a hierarchy of agents assigned to a role that defines their capabilities, limitations and assigned tasks to each of them.
c) It represents knowledge through the beliefs that an agent has over itself, relying on a knowledge base contained in a relational database.

d) It uses the knowledge recorded by the agents system to prevent or resolve problems that arise with the support in the analysis of a blackboard organized by topics and in the evaluation of rules contained in the logical modules, which has a relation to an instruction and an action or actions to be performed, when required.

The grammatical rules that correspond to this section are the following:

```
<programming_knowledge >::= 'insert_rule' '/' <rule>
    | 'evaluate_rule' '/' <rule> | 'search_rule' '/' <rule>
    | 'insert_condition '/' <condition>
    | 'evaluate_condition' '/' <condition>
    | 'search_condition' '/' <condition>
```

For the generation of rules for testing purposes, there is a module that has the following characteristics:

a. Translate a Prolog predicate to relational database including, in addition to information, the structure of predicates and relations.
b. Has been templates designed to generate Structured Query Language (SQL) [18], and interpret the validate of the information in the consultations undertaken.
c. Interact with the user through a visual interface that lets you enter and retrieve information from relational database designed for the module

5 Practical Examples Implemented on CLASS-W

A. The first example described in this section is built based on the behavior of port 135, which has been taken to this effect, because although it is used for services like Remote Procedure Call (RPC) and Message Queue Service (MQS), it has also been used by "worm" type programs to enter itself into systems with Windows environment [2], [3], [19]. In 2003, the worm W32/Lovsan.worm [20] and W32.Blaster.Worm [21, [22], have exploited the RPC vulnerability, which could allow remote code execution without authentication in affected system that received a specially crafted RPC request. In October 2008, the worm Conficker [23], is spreaded through networks at alarming rates via Microsoft Windows Server Service, exploiting RPC Handling Remote Code Execution Vulnerability and it was reported on March 2009 as Win32/Conficker.D.

The multiagent system that is created for the implementation of W-CLASS on JADE [10], in case analysis of blackboard on topic 'ports' is based in the following options:

- **'Definition Rules'**.- Generation of script for introduced in database through a logical module the rule ***'attackworm1'***, which is associated to port 135 and with a particular remote address network:
  Semantic predicates:
  `attackworm1(n_topic,n_port,n_connections,n_action).`

```
ontology_topic(n_topic).
port(n_port).
connections(n_connections).
action(naction).
```

Rules:

```
attackworm1(n_topic,n_port,n_connections,naction):-
ontology_topic(n_topic),port(n_port),
connections(n_connections), action(naction).
```

Facts:

```
ontology_topic("ports").port("135").
connections("1000"). action("stop").
action("review").
```

- **'Register of events'**.- The agents can register in blackboard events related with behavior of port 135 and address network associated.
- **'Evaluate rules'**.- The procedure for executing this consultation consists in:
  a. **'search rules'** previously defined through a logical module, which regulate use of the port that mentioned above
  b. **'Evaluate rules'** based on contributions that exist in the records of blackboard 'ports' with value 135:

     attackworm1("ports","135" , "1500", naction).

     If rule is true 'naction' returns values 'stop' and 'review', which refering to packets network of IP address registered in blackboard.
- **'Analyze blackboard'**.- Steps of the operation related to the topic 'ports':
  a. Search rules specified for the existing records on reported ports.
  b. Analyze state of 'alerts' on 'rules' specified by the administrator in the logic modules on the use of ports.
  c. View criteria to detect situations that may coincide with a pattern that is considered abnormal in relation to the use of port 135.
  d. Give instructions to the appropriate agents in order to execute certain actions previously defined and stipulated in the rules that have been set by the administrator in the logic modules, and these in turn recorded on the blackboard, their results are obtained from the same applications. An example of a message CLASS-W in order to send instructions after the evaluation rule ***'attackworm1'*** and review records of blackboard is:

     Messages sent for coordinator agent ('CAgent01'):

     "#0001/message/CAgent01/Coordinator/LAgent10/
     private/instruction/stop/packets_network/135/"
     "#0001/message/CAgent01/Coordinator/LAgent10/
     private/instruction/review/135/"

     Message sent for local agent ('LAgent10'):

     "#0001/message/CAgent01/Coordinator/LAgent10/
     private/response/agree/"

B. In order to present a second example based on references of virus and worms related with features that allow them to initiated automatically when the operating system is loaded, using the registry where are tasks of system startup.

The situation described in this section relates to the plan that can be implemented when a 'unknown' process has been registered in the task start and is considered to be a potential security problem. For purposes of illustration is used a current problem and some features of the Conficker Worm [23], which creates the files 'hloader_exe.exe' and 'hloader_dll.dll' in the system files folder ("c:\winnt", "c:\winnt\system32", "c:\windows\system"), inject this in the file 'explorer.exe' and recording with the following keys values:

```
HKEY_LOCAL_MACHINE/software/microsoft/windows/
currentversion/run/auto__hloader__key=c:\windows\system32
\hloader_exe.exe
HKEY_CURREN_USER/software/microsoft/windows/
currentversion/run/auto__hloader__key=c:\windows\system32
\hloader_exe.exe
```

Here is a scenario for the system of agents related to the topic 'startup_tasks', which can be implemented with CLASS-W:

a. Local agents have a registry of tasks authorized which will be loaded at system startup, and a list of processes that can run at any time.

b. Some local agents 'LAgent2', 'LAgent6' and 'LAgent10' have been found that the process 'hloader_exe.exe' is included as a startup task in the registry, and is not found on the list of authorized processes.

c. Local agents mentioned in the previous write on the blackboard, under the topic 'startup_tasks', its identification data and the process name 'hloader_exe.exe' and is classified as 'unkown task'

d. The coordinator agent 'CAgent01' revised the blackboard 'startup_tasks', and finds reported problem on 'hloader_exe.exe' and proceeds to do the following actions:

If there are rules associated with the process 'hloader_exe.exe' then the coordinator agent send instructions to the local agents which reported the problem. If not, the following instructions are sent to the local agents:

1. Check if the process 'hloader_exe.exe' is active, and if so, disable it until it receives a new instruction to confirms or revoke the action.
2. Remove strings located in the registry in startup tasks that are related to the file 'hloader_exe.exe'.
3. Check that the 'hloader_exe.exe' process is blocked and that it is not in execution.
4. Record a message in the administrator container with the following information: topic 'startup_tasks', process name 'hloader_exe.exe' which has been regarded as unknown, and 'preventive' instructions were sent to local agents who reported the problem.

When the administrator reviews the container, a decision should be made whether to create a rule with the actions that need to be implemented or deactivated the precautionary measures, and for the process named 'hloader_exe.exe' will be registered the respective indication on the blackboard 'instructions'.
Later, when the coordinator agent reviews the blackboard 'instructions', turns the respective actions.

A plan like the above can help to avoid the problem 'zero day', particularly relating to the unknown process that has been registered as startup tasks and can be a potential threat.

6 Conclusions

The security in computer networks is a topic that is emerging day by day due to the massive data management, though it has increased the productivity and wealth of all kinds of organisms or businesses, it has also been the means by which have led to significant financial losses, since as the countries make use of these type of services, the goal becomes more vulnerable to attacks based on the information. To counter and control this problem, it is necessary to adopt safety policies, based on computer technology, that provide the possibility of setting safety standards that allow to increase the level of confidence in the exchange of information through this medium.

Although the vulnerability of the systems cannot be eliminated completely, and therefore, the timely detection of security problems plays an important role, the "zero-day threats" are what most worry security experts.

Among the various alternatives that exist to control this problem, are those based on Intrusion Detection Systems that allow monitoring events that might be associated with malicious or harmful actions, but have the disadvantage of generating a large number of alerts false, and which do not prevent attacks that arise when these are unknown to the databases used to support the application.

With a different approach to the current, this presentation introduces the proposal to use a grammar such as CLASS-W as a tool to allow an expert, an administrator or developer of security systems, manage or develop system bases on agents that provide the possibility to harness and apply the experience taken in the area, in addition to using information obtained from the operating system based on Windows NT Technology, network resources, configuration, loaded applications start, the existence or absence of processes, use of ports, etc.

Finally, it should be added that although there are different types of grammars for management agents, we not found evidence of a grammar for development agent based systems in the area of security, and trust that the alternative is presented in this work, which may mark the way forward in developing such systems as the technology of agents has proven to be very efficient in other areas, and surpass other techniques or methodologies which are not achieved the same results.

References

1. McAfee Proven Security, Microsoft Word 0-Day Vulnerability III, http://vil.nai.com/vil/Content/v_vul27264.htm (2006).
2. Clark R.: How to cope with security radical changes in cyberspace, July 6, 2004, ID: 4198,
3. http://www.symantec.com/region/mx/enterprisesecurity/content/expert/LAM_4198.html. (in spanish).
4. Richard D. Pethia: Viruses and Worms: What Can We Do About Them?, Software Engineering Institute, Carnegie Mellon University, CERT® Coordination Center, Pittsburgh, Pennsylvania, USA, September 2003,
5. http://www.cert.org/congressional_testimony/Pethia-Testimony-9-10-2003/.
6. Bruce Schneier, SIMS: Solution, or Part of the Problem?, IEEE Security and Privacy, vol. 02, no. 5, pp. 88 (2004),
7. http://doi.ieeecomputersociety.org/10.1109/MSP.2004.83.
8. Cox, P. & Sheldon, T.: Windows 2000, Security Manual, pp. 18-24, 22-34, 44, 47, 54-81, 177, 410-424, 475, 675, 712-717. McGraw-Hill (2003). (in spanish).
9. The Universal.com.mx (Computer), is spreading the worm on YouTube, June 18, 2007, http://www.el-universal.com.mx/articulos/40642.html. (in spanish).
10. Snort (Intrusion Detection System), http://www.snort.org/about_snort/.
11. FIPA–Agent Communication Language Specifications,
12. http://www.fipa.org/repository/aclspecs.html.
13. FIPA – The Foundation for Intelligent Physicals Agents, http://www.fipa.org.
14. JADE – Java Agent Development Network, http://jade.tilab.com/.
15. Kelley Dean: Theory of Automata and Formal Languages, pp. 30, 105-170. Prentice Hall (1995). (in spanish).
16. Grune, Dick, Bal, Henri E., Jacobs, Ceriel J. H., and Langendoen, Koen G. Modern Compiler Design, pp. 548-596. McGraw-Hill/Interamericana de España, S.A.U, (2007). (in spanish).
17. V. Aho Alfred, Sethi Ravi & D. Ullman Jeffrey: Modern Compiler Design, pp. 25-82. Addison Wesley Longman, Pearson (1998). (in spanish).
18. Russell S. & Norving P.: Artificial Intelligence (A Modern Approach), pp. 36-45, 57, 110, 151, 157, 161, 163, 165, 166, 171, 196-201, 304, 419. Prentice-Hall (1995).
19. Carver N. and Lesser V.: The evolution of blackboard control architectures, pp. 19.23, Technical report, University of Massachusetts Amherst, October 1992. http://www.cs.siu.edu/~carver/ps-files/tr92-71.ps.gz.
20. Moore C. Robert. Logic and Representation. CSLI Lecture Notes nº 39. CSLI Publications, Stanford, California, 1995 (pp. 1-7, 11-14).
21. Pajares M. G. and Santos P. M.: Artificial Intelligence and Know Ingeniery, pp. 45-68, 95-106, 116-126. Alfaomega Grupo Editor, S.A. de C.V., México, D.F. (2006). (in spanish)
22. Groff James R, Weinbert Paul N.: SQL Reference Manual, pp. 95-276. McGrawHill (2002). (in spanish).
23. Cox, P. and Sheldon, T.: Windows 2000, Security Manual, pág. 18-24, 22-34, 44, 47, 475, 674, 712-717. McGraw-Hill (2003). (in spanish).
24. McAfee® Security AVERT (Anti-Virus Emergency Response Team),
25. http://www.nai.cl/es/security/virus/detail/v_100547.htm.
26. NOD32 Anti-Virus System, http://www.vsantivirus.com/gaobot-aa.htm.
27. Symantec Corporation,
28. http://www.symantec.com/region/mx/avcenter/data/la-w32.blaster.worm.html.
29. Conficker, http://conficker.com/.