Multi-agents Architecture for Distributed Intrusion Detection

Vinícius da Silva Thiago, Paulo Antonio Leal Rego, José Neuman de Souza
Master and Doctorate in Computer Science
Federal University of Ceará
Fortaleza - CE, Brazil
vsthiago@gmail.com, pauloalr@ufc.br, neuman@ufc.br

***Abstract*—The growing concern about information security in computer networks is responsible for constantly producing new ways to defend them. This work describes the proposal for an Intrusion Detection System architecture that uses agents and an ontology for sharing information. Mobile agents provide a convenient way to distribute the detection process, enabling peer to peer cooperation between network nodes. The ontology provides an organized way of storing and sharing knowledge. To evaluate the proposed solution, the architecture has been implemented using the Java programming language and Java Agent Development Framework.**

Keywords - detection; intrusion; agents

I. INTRODUCTION

The growing number of computer networks applications is responsible for the everyday great diversity and sophistication of attacks and intrusion methods, raising awareness about the safety of these networks. Intrusion detection is one of the key techniques to protect networks and is based on collecting and storing data for auditing systems and networks. According to [1], when detected, an intrusion should be reported to the security manager, and an automatic reply, in order to eliminate the causes and/or the effects of the intrusion, could be triggered.

An Intrusion Detection System (IDS) tries to detect and warn of intrusion attempts to a system or network, in which an intrusion is considered to be an unauthorized or unwanted activity [2]. A centralized IDS runs on a machine in the network in a way that it can collect data from each one of the nodes and then analyze it. However, centralization becomes a major weakness because if the machine crashes, intrusions will not be able to be detected, apart from the fact that the central analyzer can easily become a bottleneck [3].

The distributed detection architectures are more efficient and can solve the problems of centralized architectures. The more sources of information are used to ensure intrusion detection, the more accurate it becomes. The main problem faced by distributed architectures is how to collect and correlate information and then evaluate the security status of the monitored system.

The paper is organized as follows. Section II presents concepts concerning intrusion detection and Section III presents the problem to be treated. Related work is presented in Section IV. Section V describes the proposed architecture, while Section VI details its implementation. Section VII describes the experiment results. In Section VIII, the conclusion and future work are presented.

II. INTRUSION DETECTION

When dealing with intrusion detection, it should be assumed that users and programs activities are observable by auditing mechanisms and that normal activities and intrusions have different behaviors [4]. It is also worth considering that an attacker can try to compromise the IDS itself [5]. Thus, it is important for an IDS to be fault tolerant and/or able to detect problems in its own operation.

In general, the IDSs are composed of four components (sensors, analyzers, database and response units) and are responsible for activities such as monitoring the users and systems activities, auditing systems configuration, accessing data files, recognizing known attacks, identifying odd activities, auditing data manipulation, tagging normal activities, error correction and storing information concerning invaders [6].

Agent systems are composed by a collection of software agents that are autonomous and directed to a goal, located in an organizational context to cooperate through adaptable and flexible interactions and cognitive mechanisms to achieve goals that could not be achieved by a single agent [7]. Mobile agents are defined as processes that can navigate through large networks interacting with machines, gathering information and returning after having carried out the tasks defined by the user [8]. The agents are dynamically updatable, lightweight, have a specific operation and can be used as part of a flexible and dynamically configurable IDS [2].

III. PROBLEM CHARACTERIZATION

An IDS architecture should be simple and effective to provide security against different attacks. According to [9], it is an efficient solution to defend against intrusions cooperatively. It is also important for the IDS to be able to perform its function without compromising the normal operation of the network [10]. The problem then consists in how to build a distributed architecture that is robust to withstand attacks to the very structure of the IDS, enable data sharing between network nodes without creating an excessive overhead in traffic and avoid creating potential bottlenecks.

In the case of a peer-to-peer (P2P) IDS, each host can send detection requests to others without the weakness of a central controller. However, many systems like this only allow hosts on the network to get information from limited sources, such as directly connected neighbors, which can lead to inaccurate decision-making, especially in the case of attacks on multiple hosts [11]. The most important feature of

 ISBN: 978-1-61208-279-0

these types of attacks is that the activity level of the attack in each of the hosts may not be large enough to raise an alarm for the entire network. However, if a distributed IDS can collect and analyze information from multiple hosts, it may be possible to recognize the attack.

One way to implement a distributed IDS is through the use of mobile agents. A host can send mobile agents to others in order to collect relevant information from multiple hosts and to recognize an attack on multiple hosts. According to [12] and [13], the advantages of mobile agent technology include: reducing the overhead of the network, overcoming the problem of delay in the network, executing asynchronously and autonomously, fault tolerance, system scalability and operation in heterogeneous environments.

IV. Related Work

The work presented in [11] showed a proposal for a P2P IDS using mobile agents to achieve a lower processing overhead on the hosts, reduce the risk of a centralized architecture and get more accurate detections. The proposed architecture does not use ontologies and the direct contact between agents is the only way to exchange information about intrusions.

The work presented in [14] proposed the use of semantic techniques in IDS, using ontologies to extract semantic relations between intrusions in a distributed IDS. When IDSagents detect an attack or a suspicious condition, they send messages to the MasterAgent, which can extract the semantic relationships using the ontology and decide whether the activity represents or not an attack. Implemented in a hierarchical architecture, the MasterAgent has shown to be a point of failure, because if an attacker could prevent its operation, intrusions wouldn't be detected. The architecture is efficient in reducing the false positives and false negatives rates and has been implemented using Java Agent Development Framework (JADE).

The work presented in [12] showed the proposal for a distributed IDS using mobile agents and a data mining algorithm to classify network traffic behavior. It proposes the creation of signature detection agents and anomaly detection agents, the latter using data mining techniques. The article also proposed the creation of several classes of agents, in a detection structure similar to [11], with a different technique that classifies network connections according to the level of abnormality found and also proposes that, when detecting new attacks, the signatures are added to the signature detection agents. The architecture does not rely on an ontology. The authors used the JADE framework to implement the proposed IDS.

The work in [15] showed the proposal for a distributed IDS that uses agents and ontology. However, the author had not defined the internal operation of the agents, did not mentioned the use of mobile agents and proposed that the ontology should be left available in a web server, which eventually becomes a single point of system failure. The present paper shows a proposal to define the missing elements, detail the internal operation of the agents and, making some adjustments and changes, to implement an architecture that uses mobile agents very similar to that proposed by the author.

V. Proposed Architecture

This paper proposes a distributed architecture, based on the proposal found in [15] in which agents perform the task of detection by communication and collaboration, using a global ontology. The architecture is organized as a multi-agent detection system which consists of the following classes of agents: sensor, analyzer, manager, ontology, actuator and global ontology. The ontology and manager agents are mobile, while sensor, actuator and analyzer agents are fixed on network hosts and the global ontology agent is a fixed agent which is located on a single host.

The sensor agent captures raw network traffic, transforms it into a pre-defined format and lets it available for the analyzer. This one will analyze the data and apply the detection rules. If an intrusion is confirmed or suspected, two cases are possible: in the first case, a malicious activity is confirmed and the analyzer agent calls an actuator agent to perform the necessary actions; in the second case, the activity is classified as suspicious. In this case, an ontology agent can invoke a sharing of global ontology data by accessing information from the global ontology agent which, if not sufficient for the analyzer agent to decide on suspicions, will make it call a manager agent which will request information related to local suspicious activity from other analyzer agents located in other hosts of the IDS. The operation of the architecture can be seen in Figure 1.

The specification of an ontology separates the data model which defines the intrusion from the operation logic of the IDS, what allows different systems, with distinct operational logics, to share data with no previous agreement on semantics [14].

JADE allows the creation of P2P platforms and the implementation of mobile agents and is under the rules of the Lesser General Public License (LPGL). It is written in Java and offers a large amount of programming abstractions. The structure of the messages exchanged in the communication between agents is based on the Agent Communication Language (ACL) defined by Foundation for Intelligent Physical Agents (FIPA).

VI. Architecture Implementation

In order to observe the operation of the architecture, the proposed agents and ontology have been implemented using the JADE framework. In this section, the implementation of the agents and the ontology are described in details.

A. Sensor Agent Implementation

The sensor agent captures network traffic and saves it in a file in the following format: the Jpcap [16] method Packet.toString( ) is called and the result string is added to the date and time of the reception of the package, each line of the file representing a captured packet. Then it creates an analyzer agent using the JADE command "createNewAgent".

B. Analyzer Agent Implementation

The analyzer agent treats the captured packets one by one and extracts source and destination addresses, source and destination ports (if any), date and time of capturing. It stores the number of packets of a type (with the same features except for date and time) that have been analyzed, the time

 ISBN: 978-1-61208-279-0

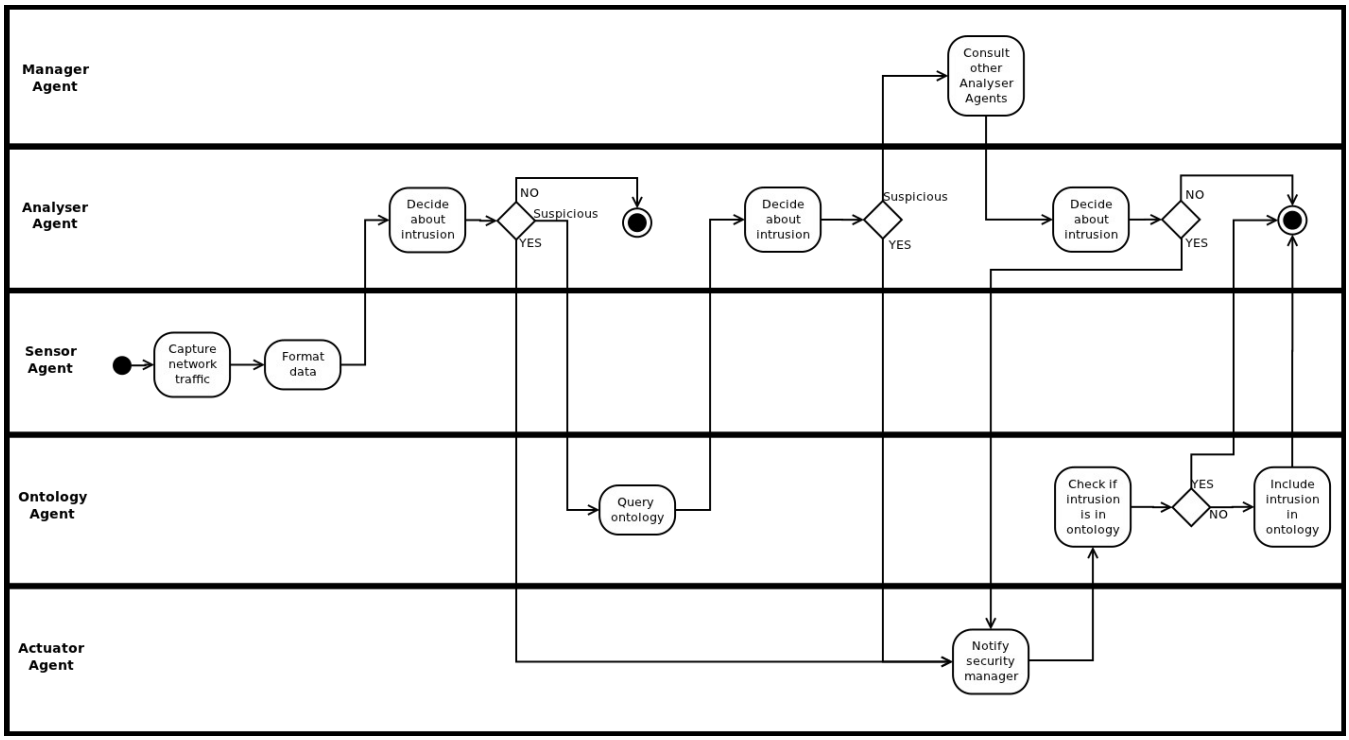

Figure 1. Agents interactions

interval between the last captured packet with these characteristics and if its origin is considered suspicious of attacks.

When implementing the architecture to detect attacks, the following method [15] has been used: for each service, the number of captured packets in the network was counted and two pre-defined limits (L_{min} e L_{max}) were used to decide the traffic's nature. If the observed number of packets was less than L_{min} then it is normal traffic, in this case, the agent temporarily stores the value. Otherwise, if the number exceeds L_{max} then it is a malicious activity. The third case is when the number of packets is greater than L_{min} and less than L_{max} , in which case the traffic is judged as suspect (probably malicious). Consequently, further analyzes should be performed, then it will be necessary collaboration with other agents for more information on the service.

The amount of time that a service has not been accessed until it is noted access in network traffic, has also been considered as a source of information for deciding about suspicious (a simplified solution of the proposed in [17]). The autor ignored in his analysis packages that are not IP protocol and outgoing traffic, in order to reduce the amount of false positives and increase processing speed. However, due to these considerations, some attacks went undetected.

Unlike [17], in the present implementation it has not been disregarded any packet or traffic, to make the system more generic and able to identify more types of attacks. Another difference is that it does not have a training period, executing classification of anomalies based only on the time interval. Thus, if a service goes without being accessed by a time interval greater than a specified value and be accessed, the access is considered suspicious. However, this criterion only classifies as anomalies or normal traffic, not directly identifying intrusions. This identification is performed by the distributed system, by interactions between agents, which, as it uses several sources of information, is expected to reduce the false positive rates.

In case of detecting a number of packets exceeding L_{max}, the analyzer agent creates an actuator agent, passing as arguments the characteristics of the packets that led to the detection. After creating the actuator agent, an ontology agent is created, being passed as arguments its purpose ("inclusion"), which is to include an attack in the ontology, as well as the information passed to create the actuator agent.

In case of detecting a number of packets greater than L_{min} and less than L_{max} , or detecting a time interval of capturing packets of the same type higher than the one set for generating suspicious, the analyzer agent marks the representation of such as suspicious packets and creates an ontology agent, passing as an argument its purpose ("query"), which is to query the global ontology, as well suspicious packets characteristics.

After analyzing each packet, the analyzer agent checks if it has received a message. Upon receiving a message from a native ontology agent stating that it was not possible to confirm a suspicion, it extracts the characteristics of the suspicion and uses them as parameters to create a manager agent. Upon receiving a message from a native ontology agent stating that a suspicion was confirmed, it extracts the characteristics and uses them as parameters to create an actuator agent. Upon receiving a message from a manager agent generated in another host to check the suspects list, it extracts the suspect's characteristics and compares them with its internal information, verifying if the information in the message corresponds to a packet coming from a host it also considers as suspected of generating attacks. If the suspicion

 ISBN: 978-1-61208-279-0

is confirmed, it creates a response message with the content "intrusion occurred". If it is not possible to confirm the suspicion, it creates a response message with the content "not detected". Upon receiving a message from a native manager agent stating that it was not possible to confirm certain suspicion with the others system hosts, it considers that it is not happening an intrusion. Upon receiving a message from a native manager agent stating that a suspicion was confirmed, it extracts the protocol, addresses and ports and creates an actuator agent, passing them as parameters. Then, an ontology agent is created, being passed as arguments its purpose ("inclusion") which is to include an attack in the ontology, as well as the information passed to create the actuator agent.

C. Ontology Agent Implementation

An ontology agent moves to the main container by calling the JADE command "doMove()" and checks the arguments passed in its creation. If the purpose is "query" the agent's goal is to query the global ontology to verify if a suspected intrusion corresponds to one stored in it. If the purpose is "intrusion" the agent's goal is to include the confirmed intrusion in the global ontology. It sends an ACL message to the global ontology agent, whose content is an instance of "suspicion" that consists of two instances of the "host" class (attacker and target) that identify the protocols and network addresses, and two strings that identify the ports numbers. Both hosts and ports match the arguments of the creation of the ontology agent. The message is written in the detection system defined ontology language. Classes "host" and "suspicion" correspond to statements of the global ontology. If the agent's purpose is querying, the message's performative is QUERY_IF, if it is to include an intrusion, the performative is INFORM.

If the message is sent to query, the ontology agent waits for the response message and when receives it, checks if it confirmed the suspicion or not. Then it migrates back to its native host, calling the command doMove( ) and creates an ACL message to the native analyzer agent. The message content is filled with the same parameters used to create the ontology agent, the performative varying to REQUEST if it confirmed an intrusion (because requests that the analyzer agent creates an actuator agent to generate an alarm) or to PROPOSE if it has not confirmed the suspicion (as proposes that the analyzer agent creates an manager agent to check the suspicion in other hosts).

D. Manager Agent Implementation

A manager agent sends a request to get informed about all active containers, creating a list. Thus, it uses the JADE command doMove( ) to move container by container. When migrating to a new host, it sends an ACL message to the local analyzer agent informing the characteristics of the suspicion that was passed as an argument in its creation, asking if it is present in its list of suspects. If the answer to the message is that the suspicion was not confirmed, it migrates to the next host of the list. If the answer is that the suspicion was confirmed, or it has traveled to all hosts on the list, it migrates back to its native container. If the intrusion is confirmed, it creates an ACL message for the analyzer agent, passing information about the protocol, source and destination addresses and ports which were considered as intrusion. Using this information, the local analyzer agent creates an actuator agent to generate the corresponding alarm. If the intrusion is not confirmed, it creates an ACL message to inform the local analyzer agent that it was not possible to confirm the suspicion.

E. Actuator Agent Implementation

The actuator agent extracts the information that has been passed in its creation and adds information regarding the date and time of when the alert is generated, saving this information in a file.

F. Global Ontology Agent Implementation

The global ontology agent is responsible for maintaining the information saved in the ontology and only receives messages written in the ontology language defined for the detection system. Upon receiving a message, it verifies its performative. If it is a message of type QUERY_IF, it is a message asking to perform a query on ontology information. It checks if the attacker's address shown in the query matches an source address in the ontology. After checking, it creates an ACL response message of type INFORM to inform if the detection was confirmed or not. As it only checks the attacker's address when querying the ontology knowledge, the global ontology agent is using the global ontology (in laboratory tests) as a way to represent knowledge about the attackers. If a message received by the global ontology agent is of the type INFORM, its content is a proposed information to be added in the ontology.

G. Detection Ontology Implementation

To define the detection ontology it was necessary to extend the JADE Ontology class. The vocabulary is composed of fifteen strings that represent elements and may be used to represent entities of knowledge that the ontology is intended to describe. The terms that comprise the vocabulary of the ontology are:

- HOST that defines network hosts;
- HOST_ADD that defines a host's address;
- HOST_TYPE_ADD that defines the protocol of the previous term;
- SUSPICION that defines the characteristics of a network stream considered as suspect of being an intrusion;
- SUSPICION_ATTACKER that represents the host suspected of being an attacker;
- SUSPICION_ATTACKER_PORT;
- SUSPICION_TARGET;
- SUSPICION_TARGET_PORT;
- SUSPICION_NUM_PACKETS that represents the amount of packets that were detected accessing a particular service and that generated the suspected intrusion;
- SUSPICION_INTERVAL that represents the time interval that a service has not being accessed;
- INTRUSION that represents an intrusion;
- INTRUSION_TARGET;
- INTRUSION_TARGET_PORT;
- INTRUSION_ATTACKER;
- INTRUSION_ATTACKER_PORT.

 ISBN: 978-1-61208-279-0

After defining the vocabulary, it was necessary to define the schemas that represent the concepts, predicates and agent actions. In the detection ontology, we defined schemas for the concept HOST, the predicate SUSPICION and agent action INTRUSION. Each added scheme is associated with a Java class, so that, when using the defined ontology, expressions indicating the terms need to be instances of these classes.

The concept HOST was associated with the class Host, as well as to primitive schemas HOST_ADD and HOST_TYPE_ADD, both of type BasicOntology.STRING. These associations imply that information passed to the ontology to represent a host must be an instance of Host, that implements the class Concept and has as attributes the strings *Add* and *TypeAdd* and the methods needed to access and assign values to them.

The predicate SUSPICION was associated with class Suspicion, as well as to concept schemas SUSPICION_ ATTACKER and SUSPICION_TARGET, both of type HOST; primitive schemes SUSPICION_ATTACKER_ PORT and SUSPICION_TARGET_ PORT, both of type BasicOntology.STRING; and primitive schemes SUSPICION_NUM_PACKETS and SUSPICION_ INTERVAL, both of type BasicOntology.INTEGER. These associations imply that information passed to the ontology to represent a suspicion must be an instance of Suspicion, which implements the class Predicate and has as attributes the Hosts *Attacker* and *Target*, the strings *AttackerPort* and *TargetPort*, the integers (long) *NumPacotes* and *Interval* as well as methods needed to access and assign values to them.

The agent action INTRUSION was associated to class Intrusion, to the concept schemes INTRUSION_TARGET and INTRUSION_ATTACKER, both of type HOST and to primitive schemas INTRUSION_TARGET_PORT and INTRUSION_ATTACKER_PORT, both of BasicOntology.STRING type. These associations imply that information passed to the ontology representing an intrusion should be an instance of the class Intrusion that implements the class AgentAction and has as attributes Hosts *Target* and *Attacker*, strings *AttackerPort* and *TargetPort* and methods needed to access and assign values to them.

VII. TESTS AND RESULTS

To evaluate the solution, a test lab was prepared. To perform attacks it was used Low Orbit Ion Cannon (LOIC), that performs simple denial of service attacks by sending a sequence of TCP or UDP requests to a target machine. The attacks initiate multiple connections to the same target host and continuously send a predefined string. The group Anonymous used LOIC to carry out attacks on several sites in recent years [18].

The hosts that took part in IDS will be called A, B and C. In the tests, C acted as main container (the container hosting global ontology). Attacks were carried out on the three hosts to see if they could identify the attacks, how the attacks were being detected, and false positives generated by the system. The objective of the tests was to determine whether the proposed architecture could be used in an IDS, although the detection criteria used were quite simple, which do not reflect the reality of the commercial systems currently used. As discussed earlier, the fact that the global ontology agent is located in a specific network node makes this node a weak point of the architecture, however, as seen in [19], JADE allows the main container to be replicated a few times creating redundant containers that take the main-container's place if it becomes unavailable.

There have been performed a total of eight sequences of attacks, in which LOIC was set to flood the target with TCP packets in the first four sequences and UDP packets in the last four. Table 1 summarizes the results.

A. First, Second, Third and Fourth Attack Sequences

In the first sequence, attacks with TCP packets were performed with an amount of packages that exceeded the L_{min} but not exceeded the L_{max} of the analyzer agents. Hosts A, B and C were attacked in that order. At the end of the attacks, the following results were obtained: A did not detect any attack, B detected the attack by means of its manager agents and C detected the attack through its ontology agents. These detections happened according to expectations, since the first host attacked (A) has detected suspicious activity and called its ontology agents, which resulted in no conclusion because there was no information about this attack in the ontology. Thus the manager agent was called, migrated to B and C and also found no information about the attack. Unable to conclude anything about the suspicious, it did not detect the attack. B has detected suspicious activity, called an ontology agent (which also resulted in no useful information) and then called a manager agent that, when migrating to A, received information that the attacker was already considered as a suspect, confirming an attack, calling the actuator agent that generated an alert. After that, the analyzer agent called an ontology agent to add information about the attacker to the global ontology. Host C, when detected suspicious activity, called an ontology agent, that, by consulting the global ontology, confirmed the suspicion of attack and called an actuator agent to generate the alarm.

In the second sequence, attacks were carried out in the same way as at the first, but at the end of the sequence it was performed another attack on A, flooding it with a number of packages exceeding the limit L_{max}. Once again, the results were exactly as expected: B and C generated alarms similar to those of previous sequence and host A has generated two alarms, both due to the detection of packets in excess of the L_{max} limit, an alarm with source on the attacker and an alarm with source on A, which corresponds to the responses of requests from the attacker.

In the third sequence, the attack to A was performed with an amount of packets which exceeded L_{max}, while attacks to B and C were performed with an amount of packets that exceeded L_{min} but did not exceeded L_{max}. At the end of the attacks, A generated two detections when its analyzer agent detected a number of packages exceeding L_{max} coming from the attacker and the generated responses to these requests. B and C detected the attacks through their ontology agents. Once A was attacked first, it has detected the attack, and its actuator agent has called an ontology agent to include the attacker in the global ontology. B and C, when detected suspicious activity, called ontology agents, that by consulting the global ontology, confirmed the suspicions and called actuator agents to generate alarms.

The fourth attack sequence was performed similarly to the first, but in this one it was waited, before initiating the

 ISBN: 978-1-61208-279-0

TABLE I. TESTS RESULTS

	Attack sequence	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*
Host A	correct detections	0	2	2	2	0	1	1	1
	false positives	0	0	0	12	0	0	0	6
Host B	correct detections	1	1	1	2	1	1	1	1
	false positives	0	0	0	13	0	0	0	9
Host C	correct detections	1	1	1	1	1	1	1	1
	false positives	0	0	0	10	0	0	0	8

attacks, a period of time greater than the interval set in the detection system from which the captured packages are to generate suspicious. Host A generated fourteen alerts: two from the packets coming from the attacker and their responses and twelve false alerts. Host B generated fifteen alerts, two from the packets coming from the attacker and the responses and thirteen false alerts. Host C generated eleven alerts, an alert from the packets coming from the attacker machine and ten false alerts.

B. Fifth, Sixth, Seventh and Eighth Sequences

Sequences five, six, seven and eight were performed in the same way as numbers one, two, three and four, but the LOIC program has been set to flood the targets with UDP packets. Sequence number five obtained the same results as number one. Sequences number six and seven obtained similar results to sequences two and three, but host A generated only one alert, concerning the detection of packets from the attacker machine in an amount that exceeded L_{max}. In the eighth sequence, host A generated seven alerts, an alert for the packets coming from the attacker and six false alerts. Host B generated ten warnings, one concerning the packets coming from the attacker and nine false alerts. Host C generated nine alerts, an alert concerning the packets coming from the attacker and eight false alerts.

VIII. CONCLUSION AND FUTURE WORK

The results of the laboratory tests confirmed that the proposed architecture can be used in intrusion detection processes. All the attacks have been identified by the system, and many have been identified by all hosts attacked. The detection method that considers as decision parameter the time interval the service has not been accessed showed to be able to detect attacks, however, led to the generation of a large number of false positives.

As opportunities for future work, it could be identified: the deployment of a more complex detection, with smarter agents, using statistical anomalies detection identified by managers agents and enabling the creation of attack signatures, which would be stored in the ontology alongside signatures already known; the development of more complex detection ontology, with more parameters to characterize the attacks; the study of the impact of the use of the proposed architecture in network traffic; and the implementation and testing of the architecture with a redundant and fault-tolerant main container.

REFERENCES

[1] R. Puttini, J. Percher, L. Mé, and R. De Sousa, “A fully distributed ids for manet”. In Computers and Communications, Proceedings. ISCC 2004. Ninth International Symposium on, IEEE, Vol. 1, Jun 2004, pp. 331–338, doi: 10.1109/ISCC.2004.1358426.

[2] O. Kachirski and R. Guha, “Effective intrusion detection using multiple sensors in wireless ad hoc networks”. In System Sciences. Proceedings of the 36th Annual Hawaii International Conference on, IEEE, Jan 2003, pp. 57-64, doi: 10.1109/HICSS.2003.1173873.

[3] I. Osman and H. Elshoush, “Alert correlation in collaborative intelligent intrusion detection systems-a survey”. Applied Soft Computing, Vol. 11(7), Oct 2011, pp. 4349-4365, doi: 10.1016/ j.asoc.2010/12/004.

[4] Y. Zhang, W. Lee, and Y. Huang, “Intrusion detection techniques for mobile wireless networks”. Wireless Networks, Vol. 9(5), 2003, pp 545–556, doi: 10.1023/A:1024600519144.

[5] Y. Huang and W. Lee, “A cooperative intrusion detection system for ad hoc networks”. In Proceedings of the 1st ACM workshop on Security of ad hoc and sensor networks, Oct. 2003, pp. 135–147, doi: 10.1145/986858.986877.

[6] D. Farid and M. Rahman, “Anomaly network intrusion detection based on improved self adaptive bayesian algorithm”. Journal of computers, Vol. 5(1), Jan 2010, pp. 23–31, doi: 10.4304/jcp.5.1.23-31

[7] T. Oren and L. Yilmaz, “Synergies of simulation, agents, and systems engineering”. Expert Systems with Applications, Vol. 39(1), 2012, pp. 81–88, doi: 10.1016/j.eswa.2011.06.038.

[8] R. Nakkeeran, T. Albert, and R. Ezumalai, “Agent based efficient anomaly intrusion detection system in adhoc networks”. IACSIT International Journal of Engineering and Technology, Vol. 2(1), Feb 2010, pp. 52-56.

[9] E. Ahmed, K. Samad, and W. Mahmood, “Cluster-based intrusion detection (cbid) architecture for mobile ad hoc networks”. In 5th Conference, AusCERT2006 Gold Coast, Australia, May 2006 Proceedings, <http://eprints.qut.edu.au/33277/> 04.12.13

[10] E. Ferreira, G. Carrijo, R. Oliveira, and N. Araujo, “Intrusion detection system with wavelet and neural artifical network approach for networks computers”. Latin America Transactions, IEEE (Revista IEEE America Latina), Vol. 9(5), Sep 2011, pp. 832–837, doi: 10.1109/TLA.2011.6030997.

[11] D. Ye, Q. Bai, M. Zhang, and Z. Ye, “P2P distributed intrusion detections by using mobile agents”. In Computer and Information Science. ICIS 08. Seventh IEEE/ACIS International Conference on, IEEE, May 2008, pp. 259–265, doi: 10.1109/ICIS.2008.21.

[12] I. Brahmi, S. Yahia, and P. Poncelet, “MAD-IDS: Novel intrusion detection system using mobile agents and data mining approaches”. Intelligence and Security Informatics, Jun 2010, pp. 73–76, doi: 10.1007/978-3-642-13601-6_9.

[13] W. Jansen, “Intrusion detection with mobile agents”. Computer Communications, Vol. 25(15), Sep 2002, pp. 1392–1401, doi: 10.1016/S0140-3664(02)00040-3.

[14] F. Abdoli and M. Kahani, “Ontology-based distributed intrusion detection system”. In Computer Conference, 2009. CSICC 2009. 14th International CSI, Oct. 2009, pp. 65–70, doi: 10.1109/CSICC.2009.5349372.

[15] A. Zaidi, “Recherche et détection des patterns d’attaques dans les réseaux IP à haut débits”. Tesis, Université d’Evry Val d’Essonne, Évry. 2011, 109 f.

[16] K. Fujii, (2007). Jpcap Tutorial. <http://www.eden.rutgers.edu /~muscarim/jpcap/tutorial/index.html>. 04.14.13.

[17] M. Mahoney, “Network traffic anomaly detection based on packet bytes”. In Proceedings of the 2003 ACM symposium on applied computing, Mar 2003, pp. 346–350, doi: 10.1145/952532.952601.

[18] A. Pras, et al. Attacks by ”anonymous” wikileaks proponents not anonymous. Report. University of Twente, Centre for Telematics and Information Technology (CTIT), Dec 2010, <http://doc.utwente.nl/75331/> 04.16.13

[19] F. Bellifemine, G. Caire, T. Trucco, G. Rimassa, and R. Mungenast, Jade administrator’s guide. <http://jade.tilab.com/doc/index.html> 04.12.13.

 ISBN: 978-1-61208-279-0