UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Visión Electrónica
Más que un estado sólido
https://doi.org/10.14483/issn.2248-4728

VISIÓN ELECTRONICA
A Research Vision

Architecture for ports scan using mobile agents

Arquitectura para escaneo de puertos usando agentes móviles

July Andrea Rocha-Hernández[1], Roxana Andrea Jaramillo-Cobos[2], Luis Felipe Wanumen-Silva[3]

INFORMACIÓN DEL ARTICULO

Historia del articulo
Enviado: 01/11/2019
Recibido: 04/11/2019
Aceptado: 13/12/2019

Keywords:
Ports scanning,
Mobile agents,
Software Architecture,
Tahiti Server,
Aglets

Palabras clave:
Escaneo de puertos,
Agentes móviles,
Arquitectura de software,
Servidor Tahití,
Aglets.

ABSTRACT:

This article proposes a software architecture for port scanning using mobile agents (AM), in order to facilitate the monitoring of network traffic or detect vulnerabilities in the logical ports of the operating system. The exposed architecture allows: to execute resident processes in each one of the stations to be monitored; include communication mechanisms that provide a high degree of autonomy to the AM; and include route planning mechanisms that choose the optimal route for each agent assigned to perform tasks on a network host. Consequently, a high availability of information in distributed environments is guaranteed, as well as a decrease in network traffic, which allows other AMs in the system to be informed of the probability of an attack on a given port. In this case, given a local network made up of ten computers and AM on a Tahiti server, the architecture is validated by means of the Scan Mobile UD prototype - developed in Java using Aglets - which gives a vulnerability probability of 16, 43% for the proposed scenario.

RESUMEN

En este artículo se propone una arquitectura de software para el escaneo de puertos usando agentes móviles (AM), con el fin de facilitar el monitoreo de tráfico en la red o detectar vulnerabilidades de los puertos lógicos del sistema operativo. La arquitectura expuesta permite: ejecutar procesos residentes en cada una de las estaciones a ser monitoreadas; incluir mecanismos de comunicación que entregan un alto grado de autonomía a los AM e incluir mecanismos de planeación de rutas que eligen la ruta óptima para cada agente asignado en la ejecución de tareas en un host de la red. En consecuencia, se garantiza una alta disponibilidad de la información en ambientes distribuidos, así como una disminución del tráfico de red, lo cual permite informar a otros AM del sistema la probabilidad de ataque en un puerto determinado. En este caso, dada una red local compuesta por diez equipos de cómputo y AM sobre un servidor Tahití, la arquitectura es validada por medio del prototipo Scan Mobile UD desarrollado en Java haciendo uso de Aglets el cual arroja una probabilidad de vulnerabilidad del 16,43% para el escenario propuesto.

[1]BSc. (c) In TelematicsEngineering, Universidad Distrital Francisco José de Caldas, Colombia. Current Position: Pmo Analyst: TransUnion Colombia, Colombia. E-mail: julyandrea.rocha@transunion.com ORCID: https://orcid.org/0000-0001-5730-5284
[2]BSc.© In Telematics Engineering, Universidad Distrital Francisco José de Caldas, Colombia. Current position: Semi senior Developer: Experian, Colombia. E-mail: roxana.jaramillo@experian.com ORCID: https://orcid.org/0000-0003-1207-0092
[3]BSc. In Systems Engineering, Universidad Distrital Francisco José de Caldas, Colombia. MSc. In Computing Systems Engineering, Pontificia Universidad Javeriana, Colombia. Current position: Professor at Universidad Distrital Francisco José de Caldas, Colombia. E-mail: lwanumen@udistrital.edu.co ORCID: https://orcid.org/0000-0002-8877-5681

Cite this article as: J. A. Rocha-Hernández, R. A. Jaramillo-Cobos and L. F. Wanumen-Silva, "Architecture for ports scan using mobile agents", Visión Electrónica, vol. 2, no. 2, Special edition, 2019. https://doi.org/10.14483/issn.2248-4728

1. Introduction

Security measures in organizational information networks are part of the processes that must be constantly reviewed in order to deal with computer attacks. In this sense, it is necessary to identify, prevent and protect access to the network by making the logical ports safe which act as a door for the entry and exit of data (including unauthorized data) that can affect security systems, as well as the computers and devices arranged on the network.
According to the above, one of the problems that is still evident in the current architectures implemented in organizations for monitoring traffic or detecting vulnerabilities in their networks - specifically in the ports of the operating system (logical ports) - is that the Applications continue to run, typically, on a client-server architecture. If by means of a virus or some other technique the central application is violated and / or deactivated, the effect on the network is that it is completely unprotected.

On the other hand, scanning becomes tedious when it has to be done on a large number of ports for different machines and there is no decentralized architecture that ensures that the details of scanning on one machine are known by another station. This type of solutions would lead to a high availability of information in distributed environments in which, if one party fails, another can take control and maintain its operation, or report the failure for its due analysis and solution.
Therefore, planning architectures that allow executing resident processes in each of the stations to be monitored and subsequently collaboratively transporting this information through the network when a coordinator decides, guarantees to decrease network traffic, [1]. Obviously, this coordinator must implement the necessary mechanisms to guarantee that there will always be availability.
To mitigate to some extent, the problems described, it is proposed to investigate and evaluate an architecture that, assumed over AM supported by the advantages they offer, [2] as well as in multi-agent (PM) platforms [3], guarantees availability of information in a distributed environment through the development of software solutions.

From the above perspective, conceptually AMs are assumed as a program that can move through a network under its own control, being able to navigate through the underlying network and perform various tasks, in each node, independently, [4]. Likewise, AMs are seen as programs that can be sent from a computer and delivered to a remote computer for execution, so that upon arrival they present their credentials and access local services and data, [5]. Now, for a methodological approach consistent with the problem, the AMs present certain attributes such as: autonomy, social ability, reactivity, proactivity, mobility, continuous operation and adaptability, as indicated in [6], not in practice the presence of all these is mandatory.

That is to say: AMs have the main ability to run on several machines without the need for their code to be located on them. As the name implies, its code is mobile, and they can be run -without connection working on the network even if it is not working waiting for it to resume-. The main advantages identified in the use of AM lie in: the potential reduction of global communication costs through the migration of computing units to data; the possibility of distributing complex computations on different, possibly heterogeneous, hosts; in addition to efficiency capabilities, adaptation to the client, reduction of network traffic, management of large volumes of information and real-time communication, [7].

On the other hand, mobile agent systems are mainly based on two components: the agent and the agent platform. Here, an agent consists of code and the status information necessary to perform a task. The task is typically performed autonomously by the agent, perhaps with the cooperation of other agents. Mobility allows the agent to migrate or jump between different platforms. A mobile agent can therefore be defined as a guest who temporarily visits several agent platforms, which support their activities, [8].

Now, as for the platform, it must offer, apart from the creation support, at least three basic services: mobility, location and communication as detailed in [9]. For this, it is important that mechanisms exist that allow agents to organize themselves to carry out their work collaboratively. Communication between agents is essential in such circumstances. The AM, then, will be able to form working groups that allow them to work in coordination with a common goal.
On the other hand, Aglets-developed by the IBM Tokyo Research Laboratory, [10] - are Java objects that can move on the Internet from one host to another. In other words, an Aglet running on a host can suddenly stop the execution, or send it to a remote host and resume it there. When the Aglet moves, it brings with it its program code and its status (data), [11]; so an Aglet requires a Java host application: an Aglet host that runs on a computer before you can visit it, [12].
In short, a mobile agent system must be able to provide the runtime environment for AM. Among other

antecedents of mobile agents that have been developed are: Telescript, Concordial, Ara, Jade, Semoa, Tacoma, Tracy, Ajanta, and of course the Aglets.

Based on the concepts exposed, and their evolution, it can be concluded that a port scan is an inspection technique that tries to discover what services are being offered by a network or server. It consists of making connections or attempts to connect to different ports (TCP or UDP) on the victim, hoping to get a response from one or more of them and deduce which application or service is listening on a certain port, [13], [14].

In the given sense, the attack methodology is made up of six stages, namely: recognition, scanning, enumeration, exploitation, access maintenance and trace elimination [15], organized in such a way that the success of one guarantees a better scenario for the next stage, it is for this reason that it is extremely important to know all the tools, methods, and times available to perform each of them. The second stage, scanning, has four main objectives: to find active hosts, open ports, versions of applications and operating systems, and other vulnerabilities of the network that you want to attack.

It is for this reason that, in order to enrich the process of protecting logical ports, the use of mobile agent systems is proposed, since they are widely distributed and heterogeneous systems that allow agents to create, interpret, execute, transfer and terminate These systems involve two basic concepts: resource servers that can contain one or more agent systems and these in turn can contain one or more places, and agents that can be available in one or more places which have the capacity to move to access their resources, or contact other agents. The mobility of the agents, that is, their ability to migrate from one site to another, establishes the main difference that this approach has with respect to other existing alternatives for the development of distributed systems, such as those of remoteprocedurecall (RPC) or remote evaluation, for example [7].

Therefore, this article is structured as follows: in section two, the research and implementation methodology is established by describing each of the mechanisms of the architecture; in section three the architecture is verified but on a prototype; and in section four the architecture validation is carried out through the application of the SCAN UD prototype for logical port scanning; then the results are established; finally, the conclusions are indicated.

2. Methodology

The architecture is presented using Kruchten's 4 + 1 model [16], motivated by scenarios and developed iteratively. The implementation proposes four views: logic, development, processes and physics; and an additional view, called a stage, used to link to the others. The architecture validation is done by means of the Scan Mobile UD prototype following the processes: visualization of main interfaces; execution on a real network scenario; and evidence of port scan results at workstations to show attack probabilities.

3.Implementation of the proposed Architecture

Among the different platforms available to create, provide mobility and communication to AMs is the AGLETS framework, described in figure 1, which allows the development of various agents. It has a workspace that runs on an IBM server called Tahiti and uses the ATP protocol to pass messages between agents. This framework is one of the most used for the development of applications based on mobile agents.

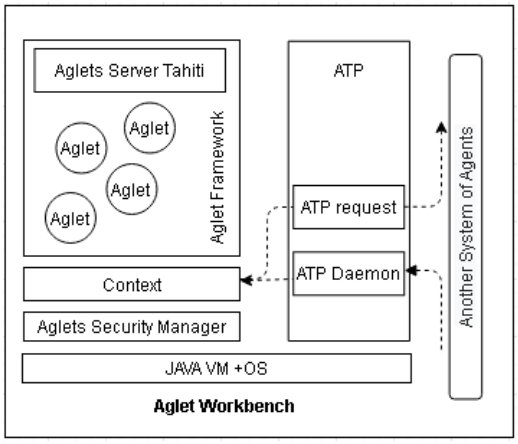

Figure 1. Description of the Aglets framework. Source: own.

On the framework the architecture is implemented, figure 2, through the development of several subsystems.

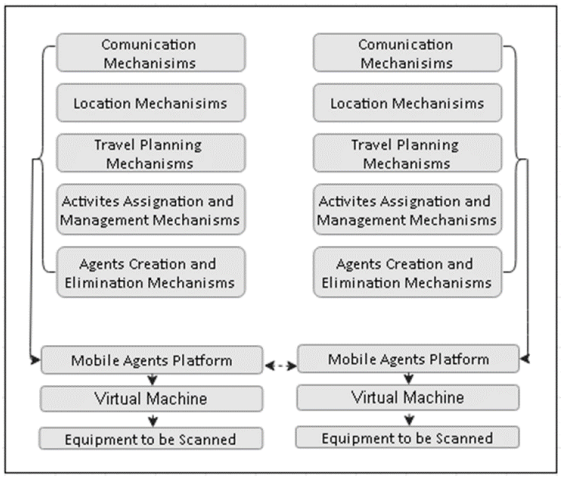

Figure 2. Proposed Architecture. Source: own.

3.1. Communication subsystem

Communication mechanisms are essential in any system, both mobile agents and agents in a multi-agent system, [17]. Among other things, the inclusion of these mechanisms in the architecture gives the system a high degree of autonomy to the agents, [18]. This subsystem allows other agents in the system to be informed of the probability of attack in a given port, it supports the idea of obtaining a working context for the Aglet, and that is, it defines the environment with which an agent relates to others. In short: if one agent has the same context as another agent, the two agents can communicate in the same workplace, [19].

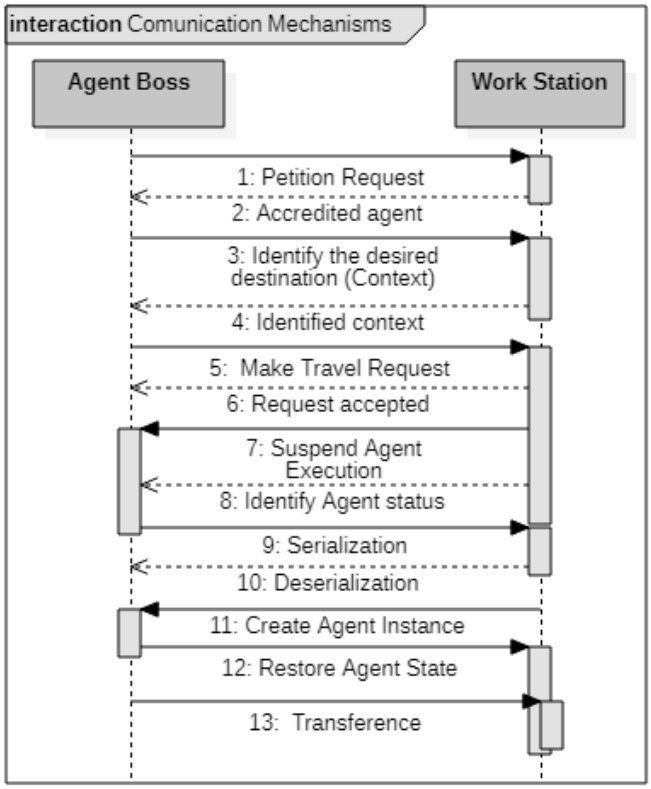

Figure 3. Description of transfer in Aglet's context approach. Source: own.

As agents transfer from one system to another, as shown in figure 3, the agent sets the desired destination and makes a travel request. If accepted, you receive authorization to make the transfer; for this the system suspends the execution of the agent and identifies the status and the parts that will be sent, the conversion of the agent's code and status is performed (serialization) and is coded according to the selected protocol, the recipient system accredits the client, [20]. The decoding of the agent is carried out and the conversion of the agent's code and state -distribution-, the system creates the agent instance, restores its state, continues its execution and proceeds with the transfer.

3.2. Location subsystem

This subsystem allows a coordinating agent from a group of port scanners to find their working agents and vice versa. The coordinator defines how an agent can be located on a remote machine in order to subsequently establish communication with these agents. Location mechanisms work well thanks to the development of standards such as FIPA, without which agents could not communicate, [21]. In the case of current technologies to model these mechanisms, it is advisable to follow a standard based on KQML or another that meets high levels of acceptance by the scientific community, [22], [23].

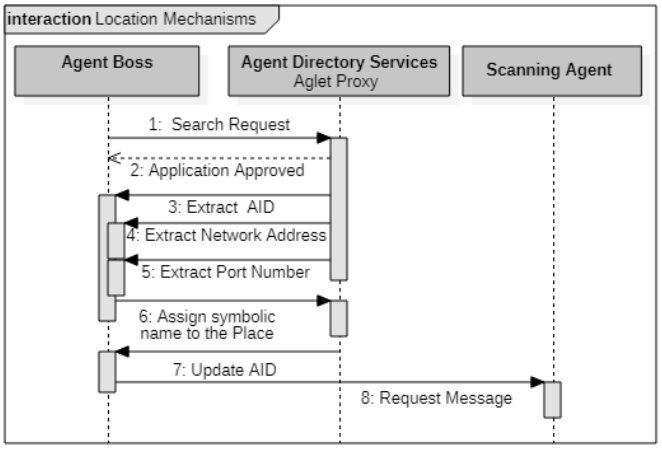

Figure 4. Description of the AgentDirectoryServices approach, used in the location subsystem. Source: own.

In figure 4 it can be seen that the location mechanism implements the functionality of a service directory, used by the coordinating agent / Boss to locate with the scanning agents / employees services with which it wishes to interact and invoke. This directory obtains minimum data such as the AID[6] (AgentIdentifier), network address, the target logical port, description of the place, which in turn contains the type of transport (protocol); the Aglet Proxy (Legal Representative) makes the communication, [24]. Finally, the agents are searched by simple matching of strings and attribute-value pairs.

4 FIPA: Foundation for Intelligent Physical Agents Multi-agent Systems and Autonomous Systems, dedicated to establishing a common and generic framework for both agents and multi-agent systems.
5KQML: Knowledge Query and Manipulation Language. It is a message format and a manipulation protocol to interact with intelligent systems. It can be used as a language through which an application program interacts with an intelligent system or more intelligent systems share knowledge for problem solving.
6 AID: Agent ID. Unique identification of each agent.

3.3. Trip planning subsystem

This subsystem allows each agent to plan the route they must choose to reach a remote machine and subsequently perform the scan on that machine. Internally, these mechanisms invoke the location and communication subsystems with remote agents to find out if there are already agents exercising the planned roles on the remote machine in order to make the best decision when sending agents and not incurring in sending agents to remote machines that run roles that are already running. It is important to note that the establishment of a specific architecture for each of the subsystems would greatly help in the best construction of an agent-based system, [25]; however, as noted in other agent society documents, the level of detail for the architectures depends on the size of the system.

On the other hand, the trip planning subsystem is based on Aglets itineraries, which incorporate the necessary functionality to carry out a travel plan, in which when an agent is dispatched, an itinerary is associated with said agent so that the agent knows where you must travel to reach the desired destination, [26]. To implement the description in figure 5, preset itineraries are created in the network topology configuration that is installed to the scanner. For this, an XML file was developed in which the user can configure the possible routes that an agent can follow to get from one computer to another; for this, the help of a network expert is required, because if these routes are made with poor planning, there would be unwarranted delays when moving an agent from one place to another, [27].

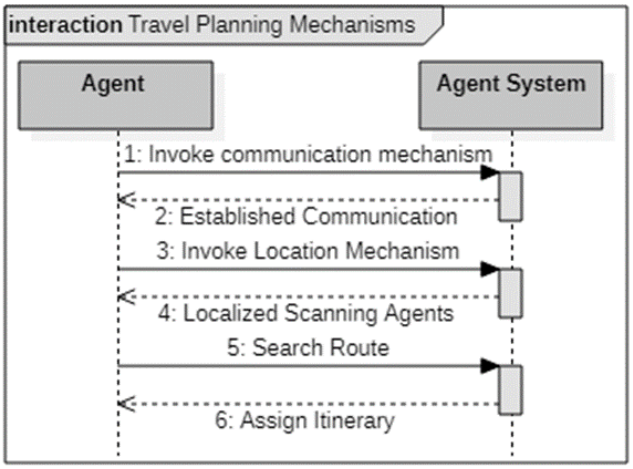

Figure 5. Description of the itinerary approach provided by Aglet used in the travel planning subsystem. Source: own

3.4. Activity allocation and management subsystem

This subsystem allows a mobile agent, once it has arrived at the destination, to communicate with the other mobile agents that are already hosted on that machine and take a role that allows it to be useful in the process of scanning ports on the established machine. At this point, agents may have to make decisions autonomously without the supervision of the remote general coordinator.

Activity allocation and management mechanisms are the key to the good performance of an agent-based mobile system, [28]. Even the literature indicates that the correct assignment of tasks, in the specific place and time, is the key to the good performance of a mobile agent system. To do this, the AID is used in order to assign activities to a specific agent. The AgentDirectoryService locates and verifies the associated AID, which is synchronized with the AgentBoss or Coordinator, which orders the activities to be carried out by the scanning agent, as shown in figure 6.

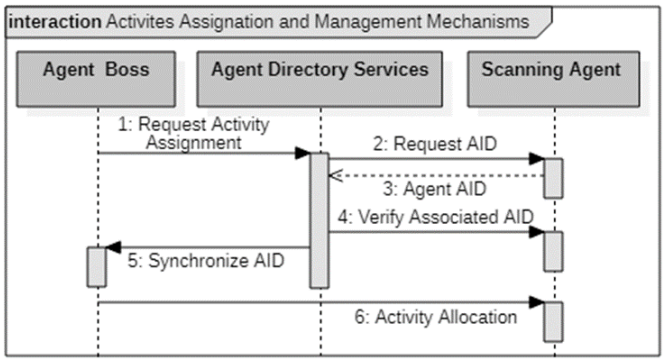

Figure 6. Description of the identifier approach provided by Aglets, used in the activity management and allocation subsystem. Source: own.

3.5. Agent creation and deletion subsystem

On some occasions, depending on the situations and the complexity of the port scan, the intervention of new agents that collaborate in the scan is required. On the other hand, creating many agents that do the same task overloads the system and redundancy problems arise that finally cause another series of problems on the network. Therefore, there are two cases: one in which the local coordinator detects that it must send new agents to help scan the remote ports; and another in which the agent that is scanning the port is able to invoke the creation of new agents to collaborate with said scan. In both, at the Aglet level, the Aglet Context is used, which provides the initialization and execution environment, in addition to the methods that an Aglet can invoke in its current context. It allows to create, recover and eliminate Aglets, among other processes, as shown in figure 7. In it, it can be seen that the only way

to create a suitable instance of an Aglet is within a context, access to which is obtained through its method getAgletContext, the Aglet is created, inserted into the current context, and invocation of onCreation () is started followed by run (). The method returns an identifier (AgletProxy) for the new Aglet as soon as the Aglet constructor method ends.

Agent creation and elimination mechanisms have commonly been developed at a low level with technologies such as CORBA[7] and RMI[8] [29]; however, new low-level mechanisms may be developed in the future to support the creation and removal of software entities remotely in mobile environments.

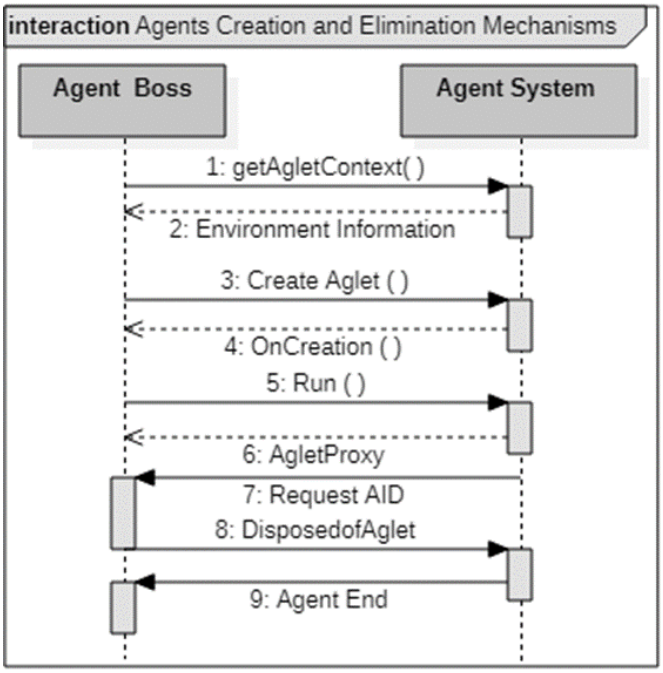

Figure 7. Use of the AgletContext functionalities used in the agent creation and removal subsystem. Source: own.

It is important to mention that the AgletProxy class is the one that allows the Aglet to communicate with the representatives of the other Aglets; it also allows dispatching an agent to another context, cloning an agent, deleting it and allowing it to be returned to the place where it was before dispatching it. All these capabilities of the AgletProxy class can be summarized in figure 8

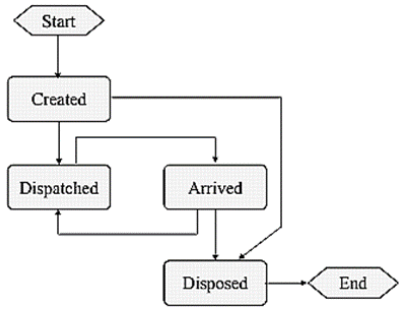

Figure 8. AgletProxy class capabilities used in the agent creation and removal subsystem. Source: own.

On the other hand, it cannot be forgotten that to achieve the implementation of the exposed capabilities, the Aglets platform uses the RMI functionalities [29] to allow communication between agents; use MASIF[9] for agent management; and implements the ACTP (AgentComunication Transfer Protocol[10]) to communicate between the various agents. As a summary, figure 9 shows the component diagram of the proposed architecture, [30].

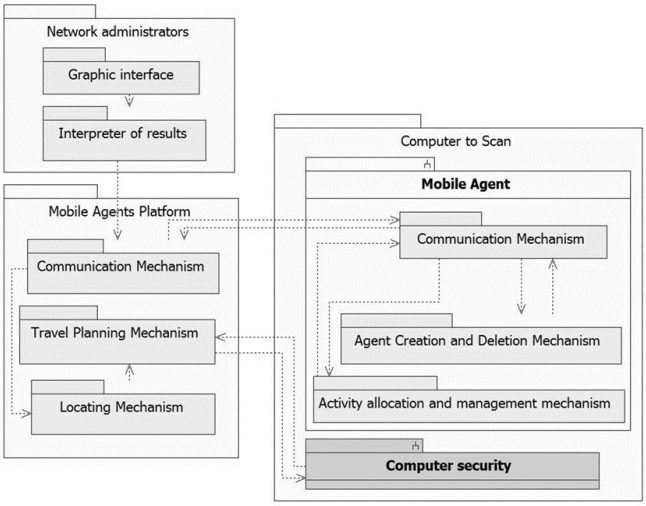

Figure 9. Component diagram of the proposed architecture. Source: own.

4. Evidence

The architecture validation was performed through the application of the SCAN UD prototype of mobile agents which supports the logical port scanning functionality. For the practical case, a local network consisting of ten computer equipment was defined as the execution scenario; as a mobile agent platform, a Tahiti server; a network administrator where the results interpreter is located together with the

7 CORBA: Common Object Request Broker Architecture. It enables the joint operation of software components written in different computer languages and running on different systems.
8 RMI: RemoteMethodInvocation. It is part of the standard Java runtime environment and provides a simple mechanism for server communication in Java-based distributed applications only.
9 MASIF: Mobile Agent System Interoperability Facilities. It is a standard for mobile agent systems that has been adopted as OMG technology.
10 ACTP: Protocolo de transferencia de comunicación del agente proporciona a cada agente un Daemon HTTP.

graphical user interface; and the rest acting as network nodes. In this context, the main objective is to perform a port scan on each host belonging to the network in order to show open logical doors that are linked to system vulnerabilities, this supported under the proposed architecture.

A screen in the user interface shows data such as: network IP, port, protocol, service currently running on the port and state-open or closed. The data obtained by the tool allows an exploration of network vulnerabilities and calculates attack probabilities. For the implementation of the Scan Mobile UD prototype, the deployment diagram of figure 10 was taken into account, and the relationship between each of the mechanisms explained in Table 1.

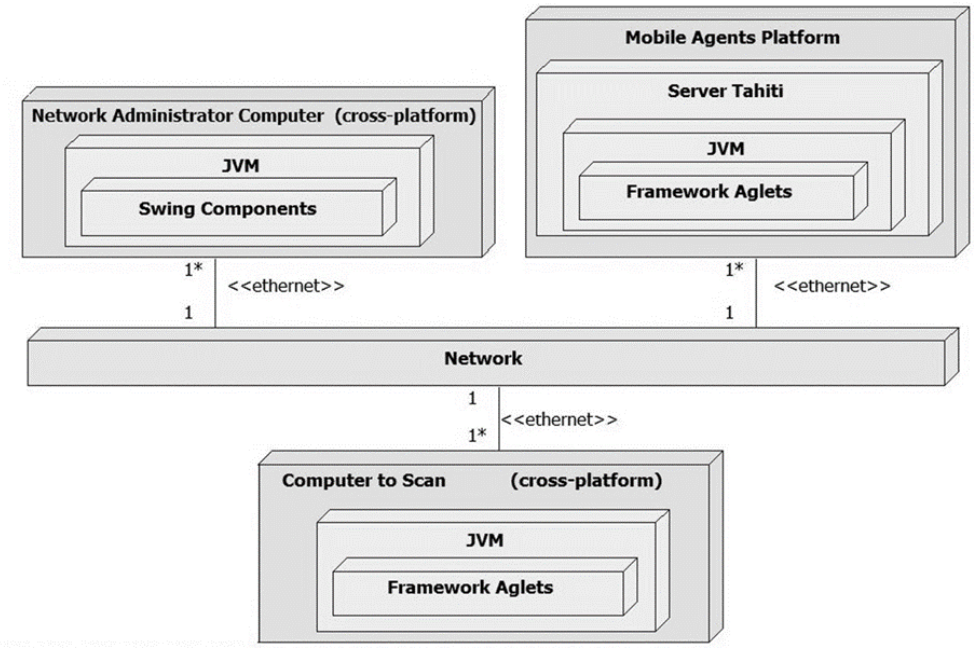

Figure 10. Deployment diagram of the proposed architecture. Source: own.

Mechanisms implemented	Features within the Scan Mobile UD prototype where the mechanism is applied
Mechanisms for creating and removing agents	It is applied when defining the range of ports that the user wants to scan in figure 11 or when he wants to do a general scan.
Communication mechanisms	When you request to start a scan, these mechanisms are activated.
Location mechanisms	They are activated when remote agents want to communicate with their coordinating agents or vice versa. These mechanisms are transparent to the end user, but they are also used when in the graphical interface of figure 12, the results are toaled to be displayed in the JTable.
Travel planning mechanisms	They are transparent mechanisms to the end user, but are activated when the agents plan their transfer to the local machines where the specific scan is going to be performed.
Activity allocationand management mechanisms	They are transparent mechanisms to the end user, but are activated on local machines
Mechanisms for creating and removing agents	They are mechanisms transparent to the end user that are activated both on local machines and on the machines that coordinate the scanning process.

Table 1. Relationship between the architectural mechanisms with the Scan Mobile UD graphical user interface. Source: own.

On the other hand, figure 11 shows the graphical interface that the user on the computer as a network administrator role perceives when they want to start a scan. It can be seen that a series of parameters is required before starting the process. For the test scenario, the system is instructed to run on all the IP's in the network along with all the ports that can be scanned.

Figure 11. Scan Mobile UD graphical interface to start the scan. Source: own

5. Results and discussion

Figure 12 shows the capture of results with the parameters of figure 11 where it can be seen that the mobile agents report: port, IP, application or service running, type of protocol and the status of the port. In turn, the number of agents employed in each scanned equipment is indicated.

With the obtained results, the probability of attack at the port level and at the general system level is calculated, applying the formulas (1) and (2) respectively, validated by the authors.

$$\%Probability\ of\ attack\ per\ port = \frac{\sum TCP\ type\ ports\ in\ open\ state\ *\ 100}{\sum TCP\ type\ ports} \quad (1)$$

$$\%Probability\ of\ system\ attack = \frac{\sum TCP\ ports\ in\ open\ state\ *\ 100}{\sum TCP\ type\ ports} \quad (2)$$

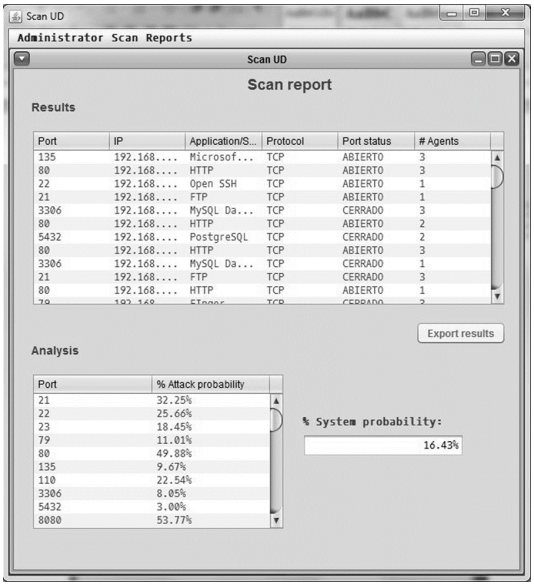

Figure 12. Scan Mobile UD results capture graphical interface. Source: own.

Port	Application / Service	Probable risk type	% Of attack
21	FTP	• Possibility of Buffer Overflow • Denial of Service (DoS) • Brute Force Attack • Access point	32,25
22	Open SSH	• Possibility of Buffer Overflow • Brute Force Attack • Access point	25,66
23	Telnet	• Possibility of sniffer • Possibility of Buffer Overflow • Denial of Service (DoS) • Brute Force Attack	18,45
79	Finger	• Collection of Information	11,01
80	HTTP	• CGI attack • Buffer Overflow • Denial of Service (From) • Collection of Information • Access point • Possibility of sniffer	49,88
110	POP3	• Buffer Overflow • Denial of Service (DoS) • Brute Force Attack • Access point	22,54
3306	MYSQL	• Injection of malicious code • Ransomware	8,05
5432	PostgreSQL	• Injection of malicious code • Ransomware	3,00
8080	HTTP	• CGI attack • Buffer Overflow • Denial of Service (From) • Collection of Information • Access point • Possibility of sniffer	53,77

Table 2. Sample results. Source: own.

Table 2, taken from the interface, shows a sample of ten records as a consequence of the scan carried out on the proposed network. A percentage of probability of attack per TCP type port is calculated for each node present in the network, which increases if it is in the open state. And a total percentage of 16.43% obtained from formula 2 indicating the risk of attack by the entire network. This management is done with direct proportionality since each open port represents an alternative that an attacker has to compromise the network.

When analyzing the results obtained for the case study, the objective is achieved since the Scan Mobile UD prototype developed on the proposed architecture demonstrated that it allows the scanning of ports through the use of mobile agents, obtaining as a final product the probability of attack on the system through port scanning

6. Conclusions

This proposed architecture based on mobile agents for port scanning was validated by means of a prototype developed in Java with the use of the Aglets framework.

In the prototype execution environment, as well as in the proposed scenario, the functionality of the agents is checked by scanning the different services in the ports that are the object of this study. In addition to providing

a quick and easy way to determine the different attacks to which a logical port may be exposed, it allows selecting and parameterizing each of the ports, their services on the network and the users responsible for controlling them. With these actions, you have the opportunity to act preventively before a vulnerability becomes a serious threat to the security of a network.

In summary, mobile agents are able to run on several machines without the need for their code to be found, thus allowing a flexible distributed computing architecture, making it easier to detect and monitor the network against vulnerabilities.

It is important to note that all of these results can be affected by the algorithm that mobile agents use to scan ports. Not only architecture is the key in this case, but also the way the communication mechanisms between agents are implemented. On the other hand, it is important to note that since the various mobile agent frameworks have their own questions that differentiate them from the others, it is possible that simply deciding to implement the model in another architecture positively or negatively affects the results shown in the present research.

As for the proposed case, the final result was a probability of system vulnerability of 16.43%, a figure that reveals to the network administrator the current risk status by scanning the ports of each of the computers.

In perspective, it is necessary to continue working on the generation of algorithms for port scanning in order to be able to carry out a more complete debugging of the proposed architecture, or other more sophisticated ones, with various scanning algorithms, for which the articles can be taken as background: And other applications [35-38][15, 31-34].

Acknowledgments

To the Research Group in Organizational Computing METIS of the Francisco José de Caldas District University, Faculty of Technology for their support and advice in the implementation of the architecture and the "Scan UD" prototype.

References

[1] A. H. Mohamed and K.H. Marzouk, "Mobile Agents for Wireless Network Security," *Journal of Computer Engineering (IOSR-JCE)*, vol. 18, no. 4, pp. 77-82, Agosto 2016. https://doi.org/10.9790/0661-1804017782

[2] R. Nicholas and M. Wooldridge, "A Roadmap of Agent Research and Development," Autonomous Agents and Multi-Agent Systems, vol. I, 1998.

[3] D. Samet, F. Barika Ktata, and K. Ghedira, "Securing Mobile Agents, Stationary Agents and Places in Mobile Agents Systems," in Agents and Multi-Agent Systems: Technologies and Applications., 2018, vol. 96.

[4] I. Satoh, "Mobile Agents for Telecommunication Aplications," in 5th International Workshop, Morroco, 2003, pp. 11-22. https://doi.org/10.1007/978-3-540-39646-8_2

[5] D. B. Lange, M. Oshima, G. Karjoth, and K. Kosaka, "Aglets: programación de agentes móviles en Java," vol. 1274, 2005.

[6] L. M. Camarinha-Matos and W. Vieira, "Intelligent mobile agents," 1998.

[7] E. Belloni and M. Campo, "Una Arquitectura de Software Para Soportar Agentes Móviles Inteligentes," in II Workshop de Investigadores en Ciencias de la Computación, 2000.

[8] J. M. Garcia and O. D. Jabonero, "Seguridad en Sistemas de Agentes Móviles," 1999.

[9] O. L. Roa, "Agentes de software:tecnologías, herramientas y aplicaciones," vol. 3, no. 1, 2004.

[10] A. Mishra, R. S. Chowhan, and A. Mathur, "Sniffer detection and load balancing using aglets in a cluster of heterogeneous distributed system environment," 2016 IEEE 7th Power India International Conference (PIICON), pp. 1-6, 2016. https://doi.org/10.1109/POWERI.2016.8077340

[11] D. B. Lange, "Documento técnico de la interfaz de programación de aplicaciones Java Aglet (J-AAPI)," 1996.

[12] O. Osunade, N. A. Azeez, A. Osofisan and M. Akinola, "Design and Implementation of a modified pull-all migration strategy in aglets mobile agent for effective network-load management," vol. 8, 2014.

"Utilizando Inteligencia Artificial para la detección de Escaneo de Puertos," in VI Jornada Nacional de Seguridad Informática ACIS 2006, Colombia, 2006.

[14] M. Anbar et al., "Investigating Study on Network Scanning Techniques," vol. 4, no. 9, 2013.

[15] C. Pérez, P. Arias, "Combinación Apropiada para Escaneo de Puertos Usando Algoritmos Genéticos". [Online]. https://www.academia.edu/28350009/Combinaci%C3%B3n_Apropiada_para_Escaneo_de_Puertos

[16] P. Kruchten. "Planos Arquitectónicos: El Modelo de "4+1" Vistas de la Arquitectura del Software". [Online]. http://cic.puj.edu.co/wiki/lib/exe/fetch.php?media=materias:modelo4_1.pdf

[17] A. C. Jimenez, J. P. Anzola, G. M. Tarazona, and S. J. Bolaños, "Modelo para la simulación de sistemas de multi-agentes robóticos en Python," *redes ing*, pp. 34-41, 2017.

[18] H. A. Diosa, "Propuesta para la conformación de grupo de trabajo en procesamiento basado en objetos distribuidos con CORBA y Objetos de Software Móviles Usando Java", agosto 2000.

[19] E. Zapata Granada and C. E. Gómez Montoya, "Arquitecturas de Software para Entornos Móviles," Revista de Investigaciones Universidad del Quindío, vol. 1, no. 25, pp. 20-27, 2014. https://doi.org/10.33975/riuq.vol25n1.145

[20] S. Franklin and A. Graesser, "Is it an Agent or just a Program? A Taxonomy for Autonomous Agents," University of Memphis, 1996. https://doi.org/10.1007/BFb0013570

[21] Siv Hilde Houmb, "Security Issues in FIPA Agents," paper in progress. [Online]. http://www.fipa.org

[22] The DARPA Knowledge Sharing Initiative and External Interfaces Working Group. "Specification of KQML Agent-Communication Language plus example agent policies and architectures", 1993. [Online]. http://www.cs.umbc.edu

[23] S. Jain, "KQML - From Scenario to Technology," *International Journal of Advanced Studies in Computer Science and Engineering IJASCSE*, vol. 7, no. 3, pp. 30-34, 2018.

[24] J. Xu and S. Wu, "Intrusion detection model of mobile agent based on Aglets," in International Conference on Computer Application and System Modeling ICCASM, 2010. https://doi.org/10.1109/ICCASM.2010.5620189

[25] M. Monroy Rios, "Arquitectura Basada En Agentes Inteligentes Para La Gestión de Configuración de Red," *Ciencias e Ingeniería al Día*, vol. 7, no. 1, pp. 39-48, 2012.

[26] V. K. Manupati, "Integration of process planning and scheduling using mobile-agent," *Computers & Industrial Engineering*, vol. 94, pp. 63-73, 2016. https://doi.org/10.1016/j.cie.2016.01.017

[27] F. Al-akashi, "Architecture of a Commercialized Search Engine Using Mobile Agents," *Artificial Intelligence Advances*, vol. 1, no. 1, pp. 44-51, 2019. https://doi.org/10.30564/aia.v1i1.688

[28] R. Pandey, "Aglet- A java based mobile agent and its security issues", May 1, 2018.

[29] F. Fernández Moya, "Introducción a la arquitectura corba para sistemas distribuídos," Grupo de Arquitectura y Redes de Computadores, 2001.

[30] J. L. Posada Yagüe, "Arquitectura de Procesos en sistemas Reactivos Distribuidos.: Departamento de informática de sistemas y computadores", 2000.

[31] A. F. Arboleda, C. E. Bedón, and S. A.

Donado, "Utilizando Inteligencia Artificial para la detección de Escaneos de Puertos," in VI Jornada Nacional de Seguridad Informática ACIS, 2006.

[32] N. Viet Hung, Q. Nguyen Van, L. Le Thi Trang, and N. Shone, "Using Deep Learning Model for Network Scanning Detection," in the 4th International Conference, 2018, pp. 117-121. https://doi.org/10.1145/3233347.3233379

[33] M. Alsaleh and P. van Oorschot, "Network scan detection with LQS: A lightweight, quick and stateful algorithm," in Proceedings of the 6th International Symposium on Information, Computer and Communications Security, 2011, pp. 102-113. https://doi.org/10.1145/1966913.1966928

[34] J. Choi, "Implementation of dendritic cell algorithm as an anomaly detection method for port scanning attack," International Conference on Information Technology Systems and Innovation (ICITSI), pp. 1-6, November 2015. https://doi.org/10.1109/ICITSI.2015.7437688

[35] M. Gupta, V. K. Solanki and V. K. Singh, "A Novel Framework to Use Association Rule Mining for classification of traffic accident severity", Ingeniería Solidaria, vol. 13, no. 21, pp. 37-44, 2017. https://doi.org/10.16925/in.v13i21.1726

[36] D. A. Riveros Hernández, D. H. Nausan García, D. S. García Miranda and J. I. Palacios Osma, "Development of a Virtual Environment for the Simulation of Electrical Maneuvers in Substations: A Case Study", Ingeniería Solidaria, vol. 13, no. 22, pp. 55-84, 2016. https://doi.org/10.16925/in.v13i22.1752

[37] E. Campero, "Productive Chains as a Source of Opportunities for Entrepreneurs in the Rural Sphere", Ingeniería Solidaria, vol. 11, no. 18, pp. 75-85, 2015. https://doi.org/10.16925/in.v11i18.993

[38] J. Arango Carrillo and J. Vargas Guativa, "Sistema de instrumentación virtual para la medición de radiación ionizante presente en el medio", Ingeniería Solidaria, vol. 8, no. 14, pp. 34-40, 2012.