Negotiation of network security policy by means of agents

Pablo Martin and Agustin Orfila and Javier Carbo

Abstract Nowadays many intranets are deployed without enforcing any network security policy and just relying on security technologies such as firewalls or antivirus. In addition, the number and type of network entities are no longer fixed. Typically, laptops, PDAs or mobile phones need to have access to network resources occasionally. Therefore, it is important to design flexible systems that allow an easy administration of connectivity without compromising security. This article shows how software agents may provide secure configurations to a computer network in a distributed, autonomous and dynamic manner. Thus, here we describe the system architecture of a prototype, the negotiation protocol it uses and how it works in a sample scenario.

1 Introduction

In the last decade the number of local networks, particularly small ones, has grown considerably motivated by wireless technology. Many of these networks are usually deployed in environments where users lack sufficient technical knowledge and, as a consequence, they are not being properly protected. Thus, security policies are often not defined and network firewalls and antivirus are the unique defensive measures. This situation motivates the creation of systems that assist in security administration. In this paper we propose an agent system that dynamically and autonomously agrees if network services are allowed, depending on the security requirements of the entities involved, the threat model for each scenario and the vulnerability level of the network elements. Thus, each agent imposes the security constraints for the entity it represents. Then, agents negotiate in order to fulfill their interconnection

Pablo Martin · Agustin Orfila · Javier Carbo
Universidad Carlos III de Madrid, Avda. de la Universidad 30, 28911 Leganes, Spain e-mail: 100029778@alumnos.uc3m.es, adiaz@inf.uc3m.es, jcarbo@inf.uc3m.es

goals. Different global configurations can be agreed and a network administrator is able to know the network policy negotiated at any moment.

The idea of using software agents for this purpose is relatively new. Traditionally, a network security policy determines what is allowed and what is not in a network [3]. It is centrally defined and centrally or distributely enforced [5]. Although this approach is more straightforward and rigorous, it demands a management and technical effort to define it and enforce it. This effort is not usually done in many medium and small networks what leaves them unprotected. In addition, current networks have become highly dynamic and heterogeneous due to mobile devices, what makes a centralized configuration harder to manage. These facts promote the exploration of the agent paradigm to provide a flexible, intelligent, dynamic and autonomous solution that improves the security administration of these networks. Related work include higher-level approaches like the one on policy-based governance by multiagent systems by Udupi and Singh [7, 8]. The multiagent architecture they propose is proactive (supporting policy monitoring, governance and enactment) and focused on virtual organizations (VO) while ours is reactive and focused on local area networks. In addition, Krügel and Toth [6] developed *Sparta* a system that allows to detect security policy violations and network intrusions in a heterogeneous, network environment. In order to fulfill this goal they use mobile agents for the task of correlating events in a fully decentralized manner. *Sparta* focuses in the detection of security policy violations but not in the policy definition itself.

Next section describes our proposed agent system architecture. This is followed by the exposition of the negotiation protocol between agents. Then, a sample scenario is explained in order to illustrate system functionality. Finally, the article ends up with the main conclusions.

2 System Architecture

In our proposal, each computer within the network has its own agent acting on behalf of it. In addition, there is an agent for each user of the network. When a user launches the execution of an agent, he determines the desired level of security by choosing a predefined security profile. Then the user informs his own agent about the operation he wants to perform and the agent carries out this task starting the negotiation with the other agents involved in the required task. Depending on the success of the negotiation, the operation will be permitted or blocked[1]. A schema of the agent system design developed is showed in Figure 1.

Roughly, the functionality of the system can be summarized as follows. The client agent negotiates to provide to the client the permission to perform the requested operation. Simultaneously, the rest of the agents involved in the operation impose restrictions due to their own security profile in order to ensure the security of the operation. This results in a complex communication scheme in which the agents

[1] Security parameters may change during the negotiation process

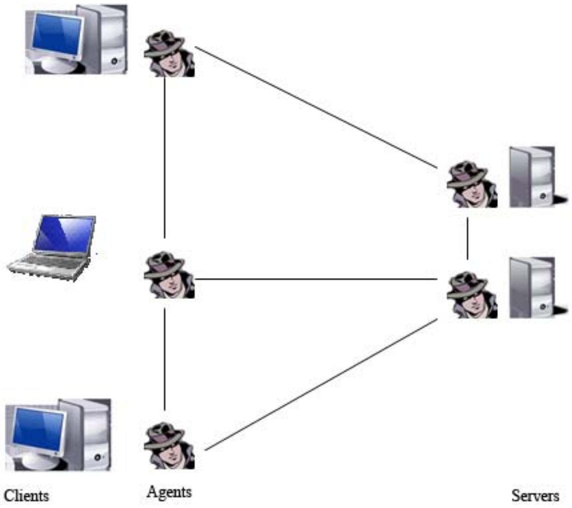

Fig. 1 Agent system architectural design. Every entity has an agent that negotiates on behalf of it

are able to change their configurations to achieve their requested operations. In order to provide such functionality, our system comprises of several roles of agents according to the services that they provide in the network. Table 1 shows these roles and their corresponding services in our prototype.

Table 1 Agents' roles and services

Agent Role	Service
Web server	Configures web server's security Negotiates security
Mail server	Configures mail server's security Negotiates security
Client	Configures client's security Negotiates security
Monitoring system	Communicates the alert level of the network
Firewall	Configures firewall's security Negotiates security
Vulnerability scanner	Provides vulnerability level reports from the requested agents
Logger	Monitors messages exchanged between agents

There are configuration variables that may change during the negotiation process. Some of these are: the allowed number of incoming/outgoing connections, trusted certificate ownership, open ports, maximum simultaneous connections from an authorized IP, remote administration allowed, automatic updates, etc. A set of pre-configured profiles or security levels for agents can be defined according to these configuration variables. These profiles allow non-expert users to configure agent's security according to their needs just choosing one of these three levels: low, medium and high.

- Low level corresponds to an agent that does not impose any constraints. It does not enforce the principle of least privilege[2]. The principle establishing that users, servers and applications may have only the necessary privileges required to perform their tasks. A network with all agents configured with this security profile is equivalent to a network without the agents.
- Medium level does not enforce the principle of least privilege but it restricts some services to improve security.
- High level offers every security mechanism available. The agents play exclusively their assigned role. It enforces the principle of least privilege.

These three levels must be present in every agent and determine the security configuration notwithstanding the fact that variables can also be changed individually for advanced users.

As shown in Table 1, our design corresponds to a purely communicative agent in accordance with the fact that agents have a perception of the environment that only comes from messages from other agents. The knowledge included in these messages is represented by an ontology which let agents understand their communicative intentions properly and act in consequence. The predicates/actions, concepts (elements), and relationships that form our ontology are shown in Figure 2.

3 Negotiation Protocol

Agents must negotiate their security configurations to provide the services requested by the user. This negotiation will be based on a message exchange with different contents expressed with terms, predicates and actions of the ontology. Since negotiation could terminate in decontrolled situations due to agents are not aware of the global state of the network, it is necessary to define a protocol that contains a set of authorized states in the network. Each service will have a set of valid configurations depending on the level of security assigned to the computer providing this service. A network is considered in a valid state if and only if all the services provided are performed with valid security configurations. During negotiation agents internally perform as a rule-based system. They check each rule and ask the rest of agents for the information they need. These rules allow changes in configuration

[2] The principle establishing that users, servers and applications may have only the necessary privileges required to perform their tasks [3]

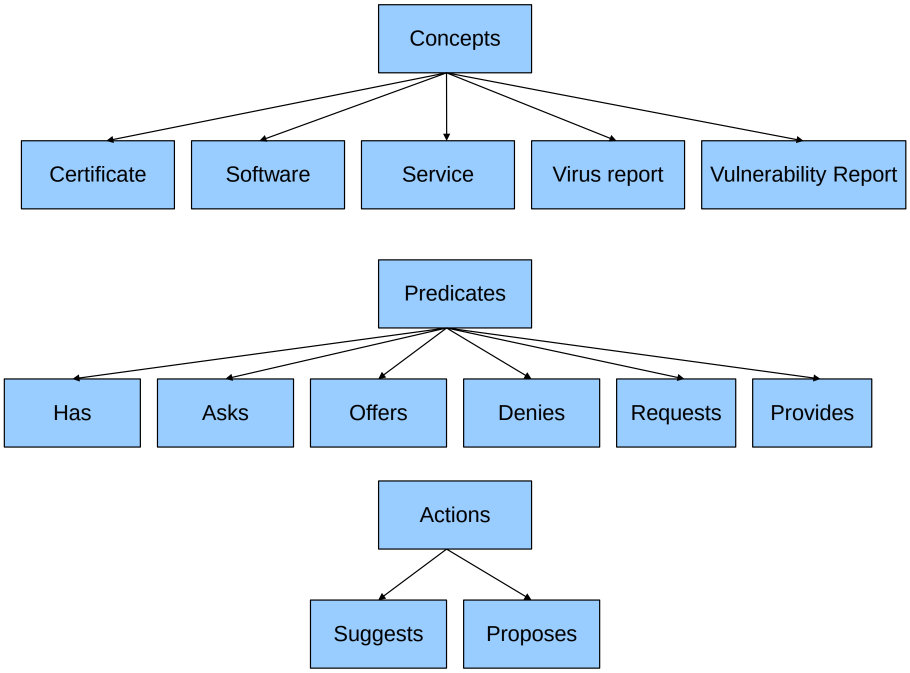

Fig. 2 Schema of the ontology used

and, therefore, agents will use them to change their settings in order to reach their goals. Messages fulfill FIPA standard [4]. The different phases of negotiation are described as follows:

1. The beginning of negotiation. An agent representing a client requests any service or tries to perform an action.
2. Feasible conditions checking. At this stage, messages about agent configurations are exchanged. Server may change its configuration temporarily (just while an operation is performed) and can suggest changes in the client's configuration.
3. Acceptance/denial of the operation. If phase 2 reaches a stage on the network that becomes the operation feasible, the server communicates it to the client. Otherwise, the server denies the connection and gives the client some information about the causes and possible alternatives.
4. Proposals. If the operation was denied, the client proposes some changes in his configuration or asks the server to look for alternatives. If the server accepts the changes proposed in the first case, then we go back to step 2. Otherwise, the server informs the client of another reason, if any, for the denial of the operation so that the client can generate new proposals. If no proposal is possible the request is denied.

Currently, our prototype implements the negotiation protocol for four scenarios. It has been developed in JAVA and the agents use JADE platform [2] to work.

4 A sample scenario

In order to illustrate the operation of the system, this section exposes one of the scenarios developed by the prototype. First, an analysis of the threats for the scenario at issue and an examination of the measures that can mitigate these threats are provided. Second, the different combinations that make a specific operation possible is shown. Finally, a diagram showing how messages are exchanged between the agents in the negotiation process is depicted.

4.1 Browsing from an internal client

This scenario illustrates the situation where a typical windows client in the intranet wants to connect to a web server that is not in the intranet. For this scenario, only three of the agent roles described in Table1 need to be considered: client, firewall and vulnerablility scanner. A simplified threat model for this operation is summarized in Table 2. As it is shown, the main threats are three. First, the client can download malware, such as virus, trojans or spyware, that infects the intranet. The main security mechanism to avoid this threat is to force the client to have an updated antivirus. Of couse it would be better not to allow the client to download software but this situation is too restrictive. Second, the client browser can have vulnerabilities that cause an infection just visiting certain web pages. In order to avoid this threat, the browser should be updated with the corresponding security pathches. A vulnerability scanner (like nessus [1]) can scan the client in order to know if the browser is vulnerable or not. Third, technologies like ActiveX may execute dangerous programs. Restricting this kind of controls is a good preventive measure in certain cases.

Table 2 Simplified threat model for the browsing scenario

Threat	Security mechanism
Download and execute malware	Update antivirus
Automatic exploitation of browser vulnerabilities	Update browser
Execution of dangerous programs through browser	Disallow ActiveX controls

Having in mind the preceeding analysis, we can elaborate a table with the security configuration combinations that make this operation possible depending on the desired level of security. This level is established by the administrator on the firewall through its corresponding agent. As can be seen in Table 3, an antivirus installed in the client is necessary for every possible combination. Depending on the level of security, the firewall agent would demand to have it updated more or less recently. In addition, browsing will be possible depending on the vulnerability level of the client browser and on the possiblity of restricting ActiveX execution.

Table 3 Allowed configurations for the browsing scenario. Dash represents any value.

Combination	Firewall	Client		Vulnerability Scanner
	Security level	Antivirus update	ActiveX allowed	Vulnerability level
1	Low	Last 24h	-	-
2	Medium or Low	Last 12h	-	Low or medium
3	Medium or Low	Last 12h	No	-
4	-	Last 8h	-	Low
5	-	Last 8h	No	Medium

Figure 3 shows a possible sequence of negociation steps for the case the level of security is set to high. First the client agent (CA) tries to connect to an external web page. The firewall agent (FA) does not have any rule for this operation so it begins checking the fourth configuration of Table 2 (because is the less restrictive regarding the security level). Thus, FA asks the CA if it has an updated antivirus. CA sends back the antivirus report that states it has not been updated in two days. As a consequence, FA blocks the connection and informs the client about the cause: having the antivirus out of date. Then, the CA proposes the FA to update it and the FA accepts it. The client downloads and install the update and the CA sends a hash of the update signed by the antivirus provider. At this moment, the FA proceed to verify the autenticity and integrity of the update. Once verified the FA asks the client to scan itself with the recently updated antivirus and ask him for a report. The client is not infected and the CA sends the report to FA. Accordingly, FA goes to the next step of the protocol and communicate to the CA the need of scanning the client for vulnerabilities. CA accepts and, as a consequence, the FA asks the vulnerability scanner agent (VSA) to scan the client. The result is a medium vulnerability level and it is reported to FA. As a consequence, the FA blocks the connection (security configuration number 4 is not fulfilled) and informs the CA about the cause. Then, the CA proposes to desactivate ActiveX execution and the FA accepts. The CA proceeds and communicates it to the FA. Finally, security configuration number 5 fulfills and, consequently, the FA accepts the connection and notifies it to the CA. At this moment the FA aggregate a rule in the firewall to allow this connection. Nevertheless, it is not a permanent rule because the conditions can change (for instance, the antivirus can become out of date)

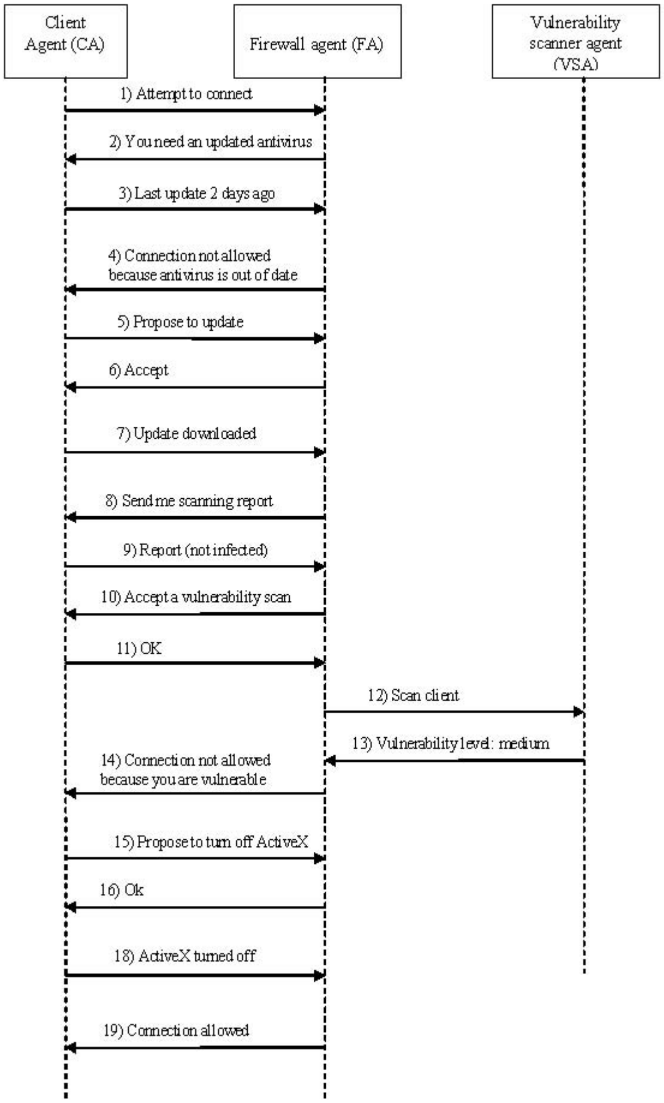
Client Agent (CA)
Firewall agent (FA)
Vulnerability scanner agent (VSA)
1) Attempt to connect
2) You need an updated antivirus
3) Last update 2 days ago
4) Connection not allowed because antivirus is out of date
5) Propose to update
6) Accept
7) Update downloaded
8) Send me scanning report
9) Report (not infected)
10) Accept a vulnerability scan
11) OK
12) Scan client
13) Vulnerability level: medium
14) Connection not allowed because you are vulnerable
15) Propose to turn off ActiveX
16) Ok
18) ActiveX turned off
19) Connection allowed

Fig. 3 Negotiation diagram for the browsing scenario when the security level is set to high

5 Conclusion

In this paper we have introduced the idea of negotiating the network security policy by means of software agents. This makes sense in those networks where no security policy has been defined in advance. Unfortunately this is a typical situation because administrators of small networks do not usually have the knowledge or the time to define and enforce one. Futhermore, the every day most deployed ad-hoc networks demand a flexible, distributed and autonomous admnistration. Our work focuses on developing an agent system that makes the process of security administration easier and dynamic. Thus, agents represent entities in the network and their goals are to protect the network while keeping an acceptable level of connectivity. The main problem faced was to offer a deterministic level of security that limits the intrinsic uncertainty of negotiation. However, the design of negotiation protocols for different scenarios according to the corresponding threat models has lead to deterministic solutions. The prototype we have implemented shows that a distributed conception of security based on agent paradigm is promising. Future work will involve the study of how agents can enforce what has been negotiated, the analysis of a threat model against the own agent system and a higher-level formalized architecture. In addition, more complex scenarios will be studied in detail.

References

1. Beale, J., Deraison, R., Meer, H., Temmingh, R., Walt, C.V.D.: Nessus Network Auditing. Syngress Publishing (2004)
2. Bellifemine, F., Caire, G., Poggi, A., Rimassa, G.: Jade: A software framework for developing multi-agent applications. lessons learned. Information and Software Technology **50**(1-2), 10–21 (2008)
3. Bishop, M.A.: The Art and Science of Computer Security. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (2002)
4. FIPA: FIPA ACL Message Structure Specification. FIPA (2001). URL `http://www.fipa.org/specs/fipa00061/`
5. Ioannidis, S., Keromytis, A.D., Bellovin, S.M., Smith, J.M.: Implementing a distributed firewall. In: CCS '00: Proceedings of the 7th ACM conference on Computer and communications security, pp. 190–199. ACM, New York, NY, USA (2000)
6. Krügel, C., Toth, T., Kirda, E.: Sparta, a mobile agent based instrusion detection system. In: Proceedings of the IFIP TC11 WG11.4 First Annual Working Conference on Network Security, pp. 187–200. Kluwer, B.V., Deventer, The Netherlands, The Netherlands (2001)
7. Udupi, Y.B., Singh, M.P.: Multiagent policy architecture for virtual business organizations. In: Proceedings of the IEEE International Conference on Services Computing, SCC '06, pp. 44–51. IEEE Computer Society, Washington, DC, USA (2006)
8. Udupi, Y.B., Singh, M.P.: Governance of cross-organizational service agreements: A policy-based approach. In: Proceedings of the 2007 IEEE International Conference on Services Computing, SCC 2007, pp. 36–43. IEEE Computer Society, Salt Lake City, Utah, USA (2007)