Employing Mobile Agents to Secure Local Networks

Panagiotis Karagiannis, Konstantinos Paparrizos, Nikolaos Samaras, Angelo Sifaleras
Department of Applied Informatics, University of Macedonia
156 Egnatia St, 54006, Thessaloniki, Greece
{pkar,paparriz,samaras,sifalera}@uom.gr

Abstract

As a consequence of the increasing availability of commercial and open source tools, maintaining network security is deemed to be a rather challenging task for the Artificial Intelligence community. Traditional, purely centralized network configuration systems exhibit traffic-load problems. Furthermore, such systems are proved to be inflexible in consistently enforcing security policies in large networks. On the other hand, purely distributed systems are difficult to organize and maintain. This paper proposes a hybrid scheme based on mobile agents, which automates the task of network monitoring and enforcing security policies. A Policy Manager agent is responsible for managing security policies and responding to network attacks. The Policy Manager and the mobile agents are enhanced with reasoning capabilities in the form of a rule-based system. Such a system has been developed and tested with encouraging results.

1. Introduction

Since the advent of Computer Science as a discipline, computer security issues have been persistently present. The traditional approach to network security management includes a group of specially privileged users, who have permission to configure all the applications used to secure a network. Super users can also define the way non-privileged users can use the network's resources. Nevertheless, complex network security related tasks are most likely to arise even on relatively small networks [1]. Furthermore, the number and heterogeneity of the infrastructures that need to be managed dictate the necessity to automate security management activities [2], [3], [16].

Artificial Intelligence (AI) and more specifically Intelligent Mobile Agents (IMAs), its most flexible offspring within a networked environment, can provide a concrete and reliable automated model for configuring and monitoring a network's security aspects [4]. The approach presented in this paper does not ignore traditional methods and techniques. On the contrary the presented scheme incorporates the ability to use a variety of already existing tools if such a demand occurs. Agents, a critical component in our implementation, decide when to use applications that enhance security by evaluating their impact in any present situation [5]. It has been proved recently [6] that any functionality implemented using technology based on mobile agents can also be achieved by conventional methods. Therefore, using IMAs for network management should be perceived not as substitute technology, but merely as yet another useful framework for creating distributed applications that can assist the task of managing computer networks.

This paper is organized as follows. In section 2 we present a detailed description of the general scheme and design characteristics of the proposed system. The features of the platform the agents run on are presented in section 3. This particular platform was implemented specifically for managing network security. Section 4 addresses the components and attributes of the agents that were developed. Finally, section 5 concludes the paper and poses future directions.

2. Scheme Design and Architecture

Mobile agents, due to their capability to migrate autonomously from node to node while performing various computations, can be perceived as representing or assisting a user or an application within a network [7], [8], [9], [16]. Each agent has a state at each point of its execution, which depending on the implementation, can consist of beliefs, ongoing interactions, commitments, desires and goals [10], [11], [17]. These data structures must be preserved when a mobile agent relocates. There are two kinds of events that can cause agent migration. Specifically, migration can occur at specific parts of an agent's code. This characteristic is known as weak mobility. Alternatively, agents can be implemented in a way that supports dynamic relocation in case they sense the appearance of particular events. Should this approach, known as strong mobility, has been adopted there must be additional programming effort to preserve the agent's execution state [6], [12].

Despite the fact that the notion of weak mobility, which we have used to model the agents in our implementation, seems to reduce flexibility, it has been shown [7] that it suffices to solve virtually all problems agents are called to face. The implicit requirement that arises at this point is that the agent's status, be it structural or execution state, has not only to be preserved but efficiently resumed as well, after migration. This is critical because mobile agents are likely to be under ongoing interaction with processes remote to the current execution platform. If the mobile agent needs to migrate all these communications should be terminated and resumed if necessary while the agent executes at another node. The mental state of an intelligent agent should also be updated before each transfer. If a terminated procedure results in a change to the agent's beliefs or commitments, then this change should be reflected to the data structure that represents the agent's mental state upon arrival to the next host. The presence of an efficient implementation of the aforementioned procedure, in spite of the minor implementation difficulties previously depicted, comprises a critical factor for establishing mobile agents as key components, as far as network configuration and management is concerned.

The ability to move their code and state to remote servers permits mobile agents to perform complex tasks at remote sites, dynamically adapting themselves to the constantly changing patterns of network environments and intrusion attempts [18]. Furthermore, by using mobile agents we avoid data transfers which would have been necessary if the traditional client-server approach had been used to model the basis of our scheme [12], [16]. Thus, the number of network connections required for data transfers is reduced. This reduction also applies to the volume of data being transferred, creating an analogous impact in bandwidth requirements.

A mobile agent, assisted however by an intermediary such as an execution environment or platform, allows the very same program code to run on different platforms and operating systems. On this basis, incompatibility issues due to code portability are eliminated. The point to which code incompatibility is being resolved or what kind of functionalities are available for a mobile agent in each host depends heavily on the programming language used to implement the system and the interactions the platform allows with the operating system [12].

The proposed platform functions essentially as an execution environment. There is no ability to create agents that run on it using an integrated visual environment. The purpose behind this approach is that the structure of the agents that run on the system must be hidden from third parties. An agent that runs on the specific platform can only be implemented if the way that interacts with the platform is known. This is similar to knowing specific implementation details that pertain to the platform. Furthermore, any agent built using different specifications is neglected by the platform, even if the authentication procedure, which is described in a following section, is successful. In this way our framework protects itself from agents build by legitimate users with malicious intentions that can access the nodes we protect. The proposed scheme recognizes only agents created by it categorizing as alien all other kinds of mobile agents. An agent categorized as alien cannot be transferred or executed.

Despite the minimal implementation features incorporated, the main characteristics of a mobile agent platform that allows efficient execution of mobile code are present [12], [13], [14]. The platform provides a single entry point for agents and supports agent and platform naming service, agent authentication, protection of host resources, agent communication, on demand class loading and remote agent control. Our framework, also defines a central decision point for securing the network. A single agent, namely the Policy Manager agent, has been assigned the task to evaluate and resolve all possible situations caused by attempts to compromise one or more hosts.

The Policy Manager builds a centralized knowledge base which is further used for reasoning purposes. Thus we avoid problems experienced in distributed knowledge systems [20].

This approach is different than the one followed by a number of existing intrusion detection systems such as AAFID [23] and EMERALD [24]. In these two systems there is no central controller or coordinator. Their architecture distributes all decision points thus avoiding a single point of failure. We have approached this issue in the following way. The location of the Policy Manager has been made completely transparent. Mobile agents perceive the Policy Manager agent as a local component in any of the hosts they execute on. All mobile agent platforms find the Policy Manager proxy through any of the multiple lookup services that can run within our network. If an intruder aimed to attack the node on which the policy Manager resides, he should have to attack all the nodes present in the network.

As an implementation language Sun's Java emerged as the optimal solution. Java apart from its advantages as described in [15] provides a very efficient infrastructure for code mobility, implementing naming services, multithreading, process management and handling security considerations. Furthermore Java renders the agent execution environment machine independent [16].

3. The Agent Platform

As already stated, the platform the agents run on has been developed without neglecting the key attributes of a generic agent platform [15], [19]. The term 'generic' is used to describe the core characteristics inherent to most mobile agent platforms. Namely, these characteristics include Agents, Agent Servers, Entry Points and Proxies [15].

In order to establish communication between agent platforms a lookup service is provided. The lookup service is independent from the agent platforms but it is critical for the execution of mobile agents. In case platforms cannot locate a lookup service, mobile agents cannot either be received or sent because they are unable to obtain the proxies of the mobile platforms included in their itinerary. Local or situated agents, nevertheless, can continue their execution as they are not affected. The lookup service functions as a yellow page service. Its entries are used by clients (agents) or Agent Servers to locate any component they can remotely use. Each agent platform upon start up registers its own proxy component with the lookup service. This proxy can be

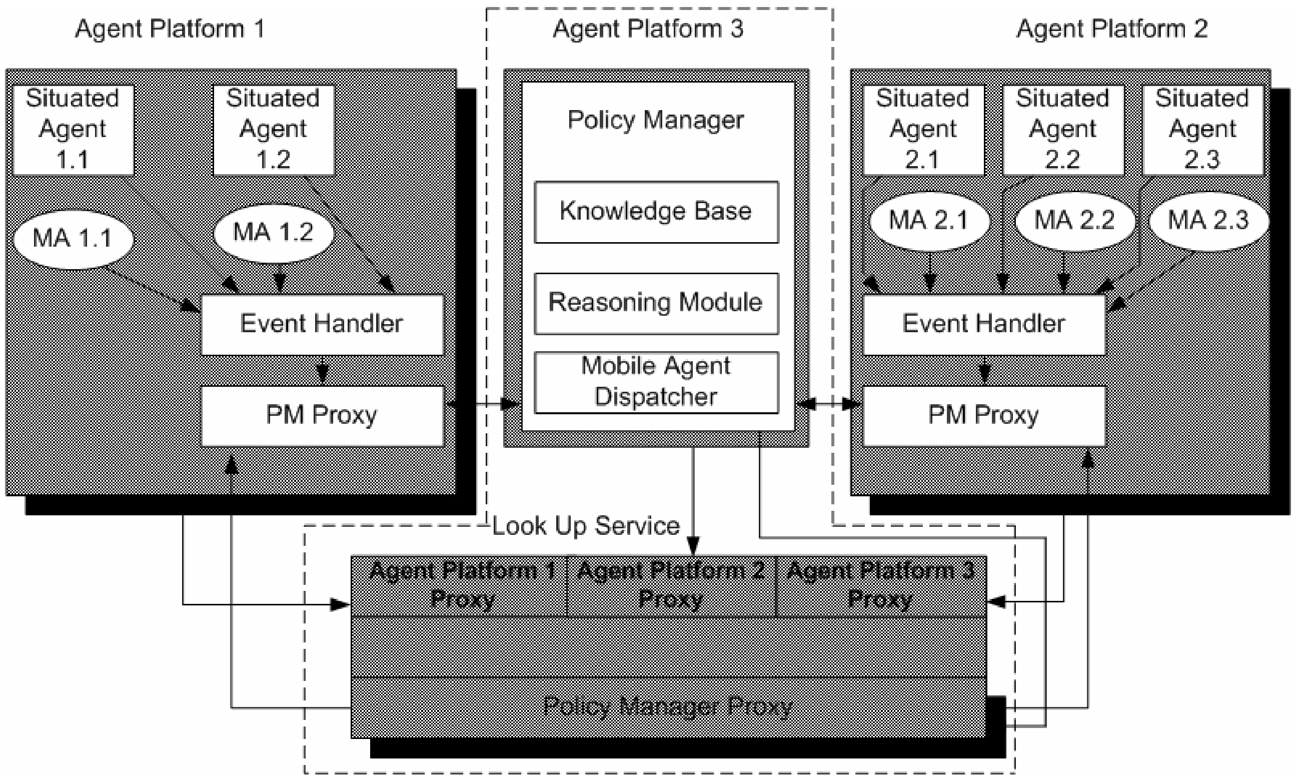

Figure 1: A snapshot of the proposed scheme.

downloaded at the client site. The local proxy component transparently executes the client's request at the remote site (Figure 1). As far as the client is concerned the proxy is a local component and its computations are perceived as taking place locally. In reality the proxy interacts with the remote host, executes all the requested tasks at the remote host and returns the computational results, if this is necessary. In our implementation a platform's proxy can only have one method which is responsible for running the mobile agent. As described the client perceives that the proxy executes the agent whereas the proxy sends the agent to the specified host. In this way messaging between the Policy Manager and mobile agents becomes easier to implement. The remote host creates a new thread to execute the incoming agent. Therefore all mobile agents must implement the Serializable interface. Each proxy component in the lookup service may have multiple methods but in our implementation of the platform's proxy we have included only the one which executes the agent. Proxies with additional methods such as the policy Manager's proxy can also be accepted by the lookup service. The Policy Manager's proxy is used by the agents in order to send messages back to the Policy Manager.

The lookup service also tracks availability of the agent platforms in the network. During the registration process, a host can define the period which the service it provides will be available for. There are also the possibilities to register forever or renew a registration which is about to terminate. In this case the entry that is created is removed only when the host is terminated or the machine is down. The former is a built in functionality of the agent platform. The method that terminates a host automatically removes its entry in the lookup service if there is one. The latter is a task the lookup service is responsible for. An entry is immediately removed when the lookup service cannot detect the presence of the machine that the registration request came from. The lookup service accepts registration requests only from known hosts. The network's firewalls therefore should be preconfigured to prevent IP spoofing.

There is no constraint on the machine of the network that can host the lookup service. In fact there is the possibility to run multiple lookup services. In that case a platform can register itself to all lookup services. It is up to agents to decide which lookup service to use. The procedure is as follows: When an agent wants to migrate to a specific host first locates all available lookup services. Subsequently the agent searches the entries of each lookup service in order to see if there is a platform running on that host. If such an entry is found the platform's proxy is obtained from the lookup service and the agent migrates as previously described. If multiple entries of different platforms in a single host are found, it is up to the agent to decide which one is going to be used.

Platforms obtain the available lookup services in a similar way. They register to one, by default. The choice is being done evaluating locality criteria. In order to preserve bandwidth, all platforms aim to register with a lookup service that runs in their sub-network. In case a lookup service is absent, platforms that execute in this sub-network pick a service that exists in the closest sub-network. This is decided after consulting a network topology list maintained by the Policy Manager. Any change in the network topology can be easily reflected on the list. Therefore even if the topology changes dynamically, agent platforms are always aware of their location within the network.

The Policy Manager is the only component in the whole scheme that can create a mobile agent. All remaining hosts can only execute, send or terminate a mobile agent. Termination can take place in two cases. The first case occurs if the Policy Manager sends a relevant signal and the alternative happens if the agent itself carries a termination command in its code.

Agents are implemented through an Agent Interface. All subsequently implemented mobile or stationary agents must implement this interface. The necessary runtime environment for agents to execute is inherent within the platform. It invokes certain methods on the agents and provides services to them such as migration, serialization and communication. The ability to extend the platform's behavior either by granting access to additional components or by defining extension mechanisms is not provided for security reasons.

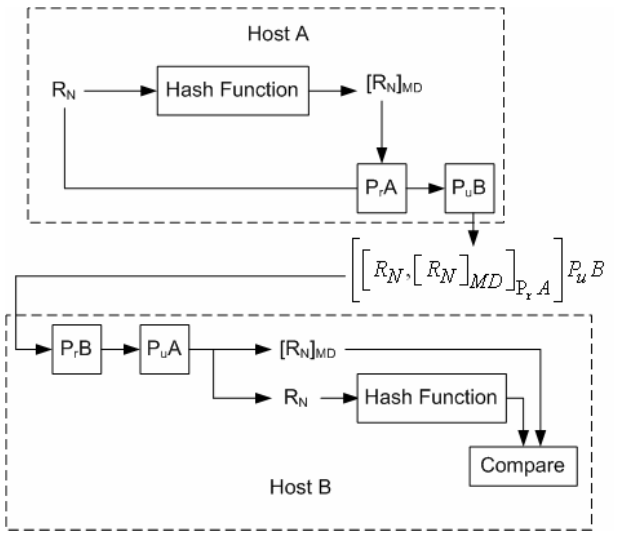

Figure 2: The authentication procedure. R_N is a random number and PrA, PuA, PrB, PuB are the private and public keys of the Host A and Host B respectively.

Java does not provide the ability to save the execution stack of a thread and transport it with the class byte-code and data. Consequently, when an agent migrates, execution at the destination will not continue from the next code statement. A public method will be called instead that will allow the agent to continue processing on its own thread. We have specified a single "entry point". Agents save necessary state information to member variables allowing the entry point method to proceed executing the agent without losing data obtained at previous points of its itinerary. Proxy support for a mobile agent platform is defined as the ability of a mobile agent to leave, upon migration, a proxy component that will forward messages or method invocations using a location transparent manner. In platforms that support this functionality, a utility is provided that based on the mobile agent's class, can create the corresponding proxy. Using mobile agent proxies is complex and in case an agent creates a proxy at each host it visits the framework becomes complex to manage. Message forwarding also increases traffic load. Thus, we chose a different implementation. When mobile agents need to have one of their methods remotely invoked, register a predefined proxy with a lookup service. The component that needs to invoke a method specified in the proxy can not only locate the agent but use the method in the desired host as well. In order to achieve this, an agent has to remove its registration upon migration and register again at the new host. Only methods included in the proxy can be remotely invoked.

All messages and class transfers between agents or platforms, whether they require plain socket connections or Java RMI, are implemented using the Secure Socket Layer (SSL) protocol. Agents are authenticated using digital signatures as shown in Figure 2. Authentication verifies that any mobile agent received comes from a platform that runs in a trusted network and is therefore considered secure. The code is encrypted with the receiver's public key and signed with the sender's private key. Encryption using the receiver's public key comes last in order to avoid man in the middle attacks.

4. Software Agents

The agents developed fall into two main categories. These are situated agents and mobile agents. All agents collaborate to achieve the same goal; enhance the security of their network. The Policy Manager agent runs on the Primary host and is responsible for responding to network attacks by dispatching mobile agents to the attacked hosts. Response is defined by deciding the course of action in order to overcome security threats. Mobile agents are sent carrying the proper instructions to reconfigure the jeopardized hosts. Only the Policy Manager can dispatch mobile agents and is also capable of terminating them. Furthermore, the Policy Manager can trace a mobile agent's current position and modify its itinerary.

Situated agents perform a variety of tasks in order to detect if the host they run on is attacked or compromised. Such system inspections can detect changes in the file system, multiple unsuccessful login attempts, increased network traffic originating from a specific port, unknown processes and unexpected entries in the system's logs. Should situated agents obtain unexpected results after a system poll, the Policy Manager receives a message which is evaluated considering at the same time the current network configuration and security policy. Every platform has an Event Handler component which is responsible for listening to requests from all active agents that currently execute on the specific platform. If the Event Handler receives a messaging request it invokes the corresponding method of the Policy Manager's proxy that resides on the platform, thus forwarding the message.

The network's security policy resides in the on the Policy Manager. The first action that the Policy Manager performs when a new agent platform is detected is to send a mobile agent carrying the specifications of the current security policy to this platform. This results to an initial configuration of the host the platform runs on. All initial specifications are entered manually but the Policy Manager can alter the file by adding or subtracting entries.

If the Policy Manager decides that action should be taken it creates and dispatches a suitable mobile agent to alter the configuration of the attacked host. Upon success a report is produced and the incident is inserted into the Policy Manager's logs. The report contains the IP the attack originated from, the ports that experienced this traffic and the action that recovered the port's traffic to usual levels. This is not only for creating a history record. The Policy Manager when a request for action from a remote host is received consults this file to trace similar problems and respond accordingly.

The Policy Manager can also remotely invoke situated agents in order to obtain knowledge pertaining to the current state of a host. Furthermore it can instruct a mobile agent that will visit this host to perform the same task. The difference is as follows: In the first case the Policy Manager has detected that a third party has been trying to access a specific resource in neighboring hosts and wants to have a more clear idea of what is happening on the hosts that have not provided a relevant alarm. The second case involves situations where a mobile agent has been created to perform a security countermeasure and needs additional information the requesting host has not provided.

We have abstained from using the Agent Transfer Protocol [21] for agent migration. This approach makes agent transport heavyweight, since classes that will not be used are transferred by default. Alternatively additional agent's classes are obtained, upon request, from the Primary host's code-base server when agent executes on an agent platform. This also gives to the Policy Manager the flexibility to construct different types of agents choosing the actions they can perform from a wide pool of available methods tailoring an agent's capabilities to handle the issues of an emerging situation. Again, if upon execution on a remote Agent Server there is a lack of ability to perform a necessary action the agent can request additional classes to be transferred. This approach makes agent transfer lightweight as it transfers, or asks for only those classes that are needed during the execution.

Access policies for server resources are strictly defined for each mobile agent, upon creation. These policies specify the remote system resources a mobile agent can assume control of, such as files, devices, network interfaces and local processes. These privileges cannot be altered afterwards. A mobile agent cannot request classes that will change the type of resources it has been called to reconfigure. Since the Policy Manager is not aware of the group of users the remote Agent Server belongs to such a countermeasure protects processes or file ownerships the agent cannot identify. For example, if the Agent Server has super-user privileges a mobile agent would be able to terminate any running process. Although overall control is extremely helpful in dedicated machines, such as web servers, where only few known processes should be running problems may arise in workstations where multiple users can be logged in simultaneously.

Although mobile agents are not as flexible as the Policy Manager in terms of making decisions that can have serious impact, they differ from situated agents. Mobile agents possess mental state. The Policy Manager may decide to provide limited knowledge to a mobile agent setting at the same time a range of options for the actions the agent can perform. It is subsequently up to the mobile agent to decide, choosing from a pool of possible actions, which is the best course of action. For, instance if an alarm for network traffic on a previously inactive port is received the Policy Manager dispatches a mobile agent who knows the networks policy for the specific port. The agent at first is assigned to find where the traffic comes from. Secondly, depending on who creates the traffic the agent may choose to close the port temporarily or permanently. Furthermore, the agent may decide to close the port only for specific hosts or sub-networks.

The system demonstrated a very solid and accurate response confronting attacks aiming at ports that are assigned to known services such as ftp, telnet and ssh. Furthermore all hosts remained intact while being under a sendmail DoS attack as described in [22].

5. Conclusions and Future Work

In this paper we propose a hybrid scheme based on mobile agents, which automates the task of network monitoring and enforcing security policies. The scheme consists of a Policy Manager agent, responsible for managing security policies and responding to network attacks. This task is assisted by a variety of mobile agents running on a mobile agent platform, which help applying configurations locally. Moreover, situated agents operating on the network nodes continuously monitor network traffic and use of their host resources. In the previous sections we illustrated our approach by describing the implementation issues of the mobile agent platform, the mobile agents and the stationary agents.

During the testing procedure, there was no point at which the system's processes caused critical consumption in host resources.

The mobile agents that have been developed are customized to handle each emerging situation. Thus by circumventing unnecessary code transfers, we have managed to keep network traffic in reasonable levels. Furthermore, we have provided methods to protect the resources a mobile agent can configure. We have also built a concrete and robust knowledge base, avoiding inconsistencies that occur in distributed knowledge systems. The prototype system that has been implemented has successfully resolved various types of known network attacks.

Despite the ongoing effort to secure computer networks intruders devise increasingly sophisticated attacks that surpass current security countermeasures. A goal of utmost importance is to provide to our framework enhanced reasoning capabilities in order to easily detect novel network attacks. This would be assisted by building mobile agents able to find, install and use software, absent from a system that would help dealing with new security threats.

6. References

[1] P.C. Hyland, R. Sandhu, "Management of Network Security Applications", Proceedings of the 21st National Information Systems Security Conference, 1998.

[2] J.W. Coyne, N.C. Kluksdahl,. "Mainstreaming Automated Information Systems Security Engineering (A Case Study in Security Run Amok)", Proceedings of the 2nd ACM Conference on Computer and Communications security, 1994, pp. 251-257.

[3] R. Boutaba., S. Znaty, "An Architectural Approach for Integrated Networks and Systems Management", ACM Computer Communication Review, Vol.25, No. 5, 1995, pp. 13-39.

[4] G. Premkumar, P. Venkataram, "Artificial Intelligence approaches to network management: Recent advances and a survey", Computer Communications, Vol.20, 1997, pp. 1313-1322.

[5] M. Woodridge, "Agent Based Software Engineering", IEE Proc. Software Engineering Vol. 144 (1), 1997, pp. 26–37.

[6] G.P. Picco, "Mobile Agents: an introduction, Microprocessors and Microsystems", Vol. 25, 2001, pp. 65-74.

[7] A. Tripathi,, T. Ahmed,, N. M. Karnik, "Experiences and future challenges in mobile agent programming", Microprocessors and Microsystems Vol. 25, 2001, pp. 121-129.

[8] C. Raibulet, C. Demartini, "Mobile agent technology for the management of distributed systems – a case study", Computer Networks Vol. 34, 2000, pp. 823-830.

[9] D. Zhang, W. Zorn, "Developing network management applications in an application-oriented way using mobile agents", Computer Networks and ISDN Systems Vol. 30, 1998, pp. 1551-1557.

[10] M. Wooldridge, P. Ciancarini, "Agent-oriented software engineering", Handbook of Software Engineering and Knowledge Engineering, 1999.

[11] Y. Shoham, "Agent-oriented programming", Artificial Intelligence Vol. 60, 1993, pp. 51-92.

[12] A. Tripathi, N. Karnik,, T. Ahmed, T. D. Singh, A. Prakash, V. Kakani, M.K. Vora, M. Pathak, "Design of the Ajanta system for mobile agent programming", The Journal of Systems and Software Vol. 62, 2002, pp. 123-140.

[13] L. Chunlin, L. Zhengding, L. Layuan, "Design and Implementation of a hybrid Agent Platform", Programming and Computer Software Vol. 29(1), 2003, pp. 28-42.

[14] J. Riekki,, J. Huhtinen,, P. Ala-Siuru,, P. Alahuhta, J. Kaartinen, J. Roning, "Genie of the net, an agent platform for managing services on behalf of the user", Computer Communications Vol. 26, 2003, pp. 1188–1198.

[15] M. K. Perdikeas, F. G. Chatzipapadopoulos, I. S. Venieris, G. Marino, "Mobile agent standards and available platforms", Computer Networks Vol. 31, 1999, pp. 1999-2016.

[16] M. Dalmeijer, D.K. Hammer, A.T.M. Aerts, "Mobile Software Agents", Computers in Industry Vol. 41, 2000, pp. 251-260.

[17] R.W. Collier, C.F.B. Rooney, R. P. S. O'Donoghue, G.M.P. O'Hare, "Mobile BDI Agents", Proceedings of the 11th Irish Conference on Artificial Intelligence & Cognitive Science, Galway, Ireland, 2000.

[18] K. Boudaoud, C. McCathieNevile, "An intelligent Agent-based model for security management", Proceedings of Seventh International Symposium on Computers and Communications (ISCC), Taormina, Italy, 2002, pp. 877-882.

[19] D. Horvat, D. Cvetkovic, V. Milutinovic, P. Kocovic, V. Kovacevic, "Mobile Agents and Java Mobile Agents Toolkits", Proceedings of the 33rd Hawaii International Conference on System Sciences, 2000.

[20] W.Z. Ras, A. Dardzinska, "Ontology-based distributed autonomous knowledge systems", Information Systems, Vol. 29 (1), 2004, pp. 47-58.

[21] D. B . Lange, Y. Aridor, "Agent Transfer Protocol – ATP/0.1", Technical Report, IBM Tokyo Research Laboratory,1997.

[22] S. Jha, M. Hassan, "Building agents for rule-based intrusion detection system', Computer Communications, Vol. 25, 2002, pp 1366-1373.

[23] J.S. Balasubramaniyan, J.O. Garcia-Fernandez, D. Isacoff, E. Spafford, D. Zamboni, "An architecture for intrusion detection using autonomous agents", Technical report, Purdue University, West Lafayette, Coast Laboratory, Indianapolis, 1998.

[24] P.P. Porras, P.G. Neumann, "Emerald: event monitoring enabling responses to anomalous live disturbances", Proceedings of the 20th National Information Systems Security Conference, National Institute of Standards and Technology, 1997, pp. 353-363.