

A Design of local resource access control for mobile agent in PDA

Takuya Iizuka,[†], Angel Lau, *Nonmember* and Tatsuya Suda^{††}, *Member*

Summary

Personal Device Assistance (PDA) devices are becoming popular, and some such devices have extended their capability to run a mobile agent platform. This paper describes a new security mechanism for mobile agent platforms running on PDAs. Our security mechanism is based on PDAgentSecurityManager built upon a Java virtual machine environment and provides dynamic authentication and flexible access control. Our security mechanism was implemented and empirically evaluated using an example application.

Key words: *mobile agent, security, access control*

1. Introduction

In the project described in this paper, we consider a scenario where a large number of users carry a PDA supporting a mobile agent platform. We also assume that many retailers advertise in the Internet using mobile agent platforms and send information related to their products (e.g., digital discount coupons) that are tailored to their customers based on their personal user profiles. For such PDA applications, security (i.e., protecting user information) becomes a major concern. Malicious agents can invade PDA user privacy and damage data in a PDA. Unfortunately, techniques currently available are not able to provide security in the application domain we consider in this paper. Security mechanisms used in some of the current agent platforms are designed for multi-user computer systems. These security mechanisms protect their local resources by rejecting migration requests from untrusted host. In a real world, for instance, individuals make some security assessment on a letter received based on its appearance. However, in a network world, security mechanisms are nowhere near the human capability. Agents' appearance will not provide a clue on how safe they are. We only know after malicious agents damage important data in PDAs.

This work was supported by the NSF through grants ANI-0083074 and ANI-9903427, by DARPA through Grant MDA972-99-1-0007, by AFOSR through Grant MURI F49620-00-1-0330, and by grants from the University of California MICRO Program, Hitachi, Hitachi America, Novell, NTT, NTT Docomo, Fujitsu, and NS Solutions.

Manuscript revised June 30, 2001.

[†] The author is with NS Solutions, Chuo-ku, Tokyo, 104-8280, Japan.

^{††} The author is with University of California Irvine, 92617 USA.

This paper describes a new security mechanism for mobile agent platforms in PDAs. Our new security mechanism is based on a PDAgentSecurityManager, an extended Java SecurityManager. Using an example application of a video rental shop, we have examined and confirmed that our security mechanism solves the concern of protecting data in an open network environment.

2. Mobile Agent and Agent Security

Security issues in a mobile agent environment may be divided into the following four groups [1].

- To protect local host from mobile agents,
- To protect mobile agents from other agents,
- To protect mobile agents from hosts,
- To protect a underlying network from malicious agents.

In this paper, we focus on the protection of local hosts from malicious mobile agents. Some of the existing security mechanisms for mobile agent systems are briefly summarized below.

- **Sandbox** -- Sandbox architecture is designed to execute a remote code in local Java Virtual Machine using Java 1.x [4]. In this security model, a remote code is not allowed to access any local resources.
- **Digital signature** -- Digital signature verifies that the mobile code has not been altered. In this technique, a local site administrator is required to register public keys from trusted sites to a local database before a local machine receives a code from a remote host.
- **Protect Domain** -- Java2 Virtual Machine creates a protection domain for program execution [3]. This domain restricts code access privileges based on who created the mobile code.
- **External Directory Service** -- Voyager security uses external directory service [7] to control access privileges. This security model also provides sandbox model when a mobile code is initialized.

3. Requirements and Design

In this section, we describe the design of our security mechanism for mobile agent platforms.

3.1. Requirements in a PDA Agent Platform

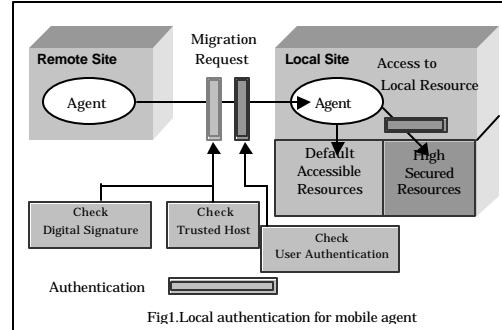
A PDA has limited resources and a simple interface (e.g., a small window and non-keyboard based input interface). Therefore, security mechanisms for PDA applications must be designed with a simple user interface. Other requirements are listed below.

- **Migration from non-registered host** -- In the PDA application considered in this paper, we assume that mobile agents containing advertise information migrate to a PDA. Such mobile agents may carry valuable information or junk information such as Spam mails. Existing security mechanisms requiring registered site information do not support all the functions required for providing security in such PDA applications. A desired security mechanism should allow a PDA mobile agent platform to consult its user first in deciding whether it accepts or denies a migration request from an unknown host. A desired security mechanism should also include functionality by which untrustworthy agents can be removed from the system.
- **Local authentication** -- Many existing mobile agent platforms use user information/profile registered in an external directory for access control. However, PDAs are primarily for personal use with a limited resource, and their network capability may be insufficient for communication with an external directory to conduct authentication in a timely manner. A desired PDA authorization mechanism may not involve an external directory and should be performed locally.
- **Flexible access control** -- Once a mobile agent migrates, a PDA user can evaluate the incoming agent and may change the access privilege that was originally given to the agent. A desired PDA security mechanism must support capability by which agent access privileges are dynamically changed.
- **Third person operation** -- Since PDAs are portable, PDAs may be stolen or misplaced, leaving sensitive personal information vulnerable to a third person. A desired security mechanism for a PDA agent platform should require local authorization to each security operations.
- **Easy operation** -- It is required to register a trusted server list and list of allowed agent actions to a local database or to a directory server when a system is initialized (or before platforms accept migrating agents) in current mobile agent platforms. **How is this related to "easy operation"? I don't understand.**

3.2. Design

We designed a new security mechanism based on the requirements described in the above subsection. Our security mechanism is based on PDAgentSecurityManager,

whose functionality is to control agent's access to local resources. It compares property of a mobile agent and user authentication with local security information to configure access control list. Fig 1. Illustrates the local authentication mechanism in our security mechanisms. Detailed mechanisms are described below.



- **Migration** -- PDAgentSecurityManager performs agent's access control to local resources. A property of agent or user authentication is compared with local security information to configure access control list on incoming mobile agents. (Security manager only checks newly migrating agents, not those which are already on a local node.) When a mobile agent platform receives a migration request from an unknown host or receives an unsigned migrating mobile agent, PDAgentSecurityManager signals a user. A PDA user then accepts or denies a migration request from an unknown host. In addition, PDA users are given capability to pre-set configuration to accept migration request from trusted hosts.
- **Authentication** -- PDAgentSecurityManager does not rely on external information. When a migrating mobile agent arrives at a host, PDAgentSecurityManager requests confirmation from the PDA user. PDAgentSecurityManager requires pre-defined password from the PDA user so that it can prohibit third person from giving permission.
- **Manageable Access Control List** -- A user can specify different access privileges to different mobile agents through PDAgentSecurityManager, which in turn creates an internal list containing resource names and permitted operation for different mobile agents. This resource list is associated with each protection domain. When a mobile agent accesses local resource, PDAgentSecurityManager checks whether the requested resource is in the list of allowed resources.
- **Dynamic Access Privilege Modification** -- A major complexity is that it is not known what resources an agent accesses before actually executing the agent. Thus, PDAgentSecurityManager provides a method to modify an access control list. A PDA user may call this method while running an agent. This method allows

modification of an access control list for its protection domain, not for other protection domains.

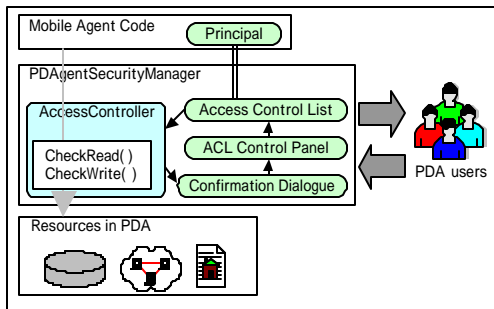


Fig 2. PDAgentSecurityManager architecture

- **Operation Monitoring** -- PDAgentSecurityManager monitors resource access by agents to detect illegal operation. When an illegal access to a resource occurs, a PDA user may deny such access, or he/she may modify the access control list so that such resource access becomes accepted.
- **User Interface** -- PDAgentSecurityManager provides uniform graphical user interface for user security operation. An interactive panel (i.e., a user interface) is called from PDAgentSecurityManager. (See Fig2.)

4. Implementation

4.1. Agent Platform

We implemented our new security mechanism (PDAgentSecurityManager) based on Voyager Object Request Broker, a popular mobile agent platform. We disabled the Voyager ORB's security mechanism (an extended Java 1.2 security mechanism) in our implementation. Our implementation effort concentrated on implementing security for local resources (i.e., files and directories), as we can apply techniques similar to existing ones to provide security for other resources.

4.2. Voyager ClassLoader

We implemented PDAgentSecurityManager on the Voyager ORB. The Voyager has its dedicated ClassLoader, which loads Java class files, instead of Java ClassLoader. The Voyager ClassLoader loads classes from local directories and remote Voyager ORBs [5]. In order to identify which host is sending a migration request, we used isForeign(), a function supported in Voyager ORB. This function detects where the code is from. If a code is from a remote host, PDAgentSecurityManager displays a dialogue box so that a PDA user can provide information on access privilege.

4.3. Password Authentication and Principle Object

PDAgentSecurityManager includes a password based local authentication. A password is stored in a PDA so that the authentication is performed locally within a PDA.

Since our PDAgentSecurityManager is extended the Java default security mechanism, we used a password-based authentication to identify the PDA user. If PDA users want to prohibit agent migration from specific hosts, they can reject such request by registering specific hosts in a Java 2 security policy file.

After authentication is done, a mobile agent can run in a local Java protection domain including dedicated principal object. This principal object contains information on where the agent comes from. It verifies the privileges given to an agent. After authentication, PDAgentSecurityManager monitors agent's access to local resources.

4.4. Local Policy File

In our implementation, a local policy file in PDA contains a default access privilege list for mobile agents. PDAgentSecurityManager reads a local policy file when a mobile agent accesses PDAgentSecurityManager to obtain migration permission. This file is a text file containing file/directory names and access categories. In our implantation, we focused on file and directories, and access categories implemented are read and write operations.



Fig 3. ACL modification panel

4.5. Access Control List Object (ACL)

Access Control List (ACL) is a list of objects containing pair-wise values, i.e., resource names and allowed actions. PDAgentSecurityManager creates the list when a mobile agent migrates to a local host. After PDAgentSecurityManager sets this ACL object list, a mobile agent can run in a local host. This object list is located in the PDAgentSecurityManager class so that only PDAgentSecurityManager can modify this object list. In order to provide users to modify the list, we have also implemented an interface (ACL modification panel) through which a user can modify this list

4.6. ACL modification panel

PDAgentSecurityManager provides uniform graphical user interface and a ACL modification panel to provide users with common interface to modify an access control list. This interface is included in the PDAgentSecurityManager

class. An ACL modification panel allows users to see the current ACL and to modify ACL associated with the current protection domain.

5. Our application

5.1. Scenario

We implemented a retail shop application to examine our security mechanism design and implementation. The application scenario implemented is described below. We assume a retail shop such as a bookstore or a video rental shop, where a mobile agent platform is deployed. When a customer carrying a PDA enters the shop, an agent migrates from a shop computer to customer's PDA and records customer actions and preference. (For instance, when a customer touches a product displayed in a shop, an agent in a PDA detects the action and records which product that the customer showed interests.) Then, user action and preference information are transmitted to a shop computer when the customer leaves the shop. Customers may receive electronically discount coupons in return for providing such information.

Our security mechanism is applied in the application scenario described above. When an agent on a shop computer issues a migration request to user's PDA, our security mechanism on a user PDA displays a dialogue box and asks a user to enter a password. If a user inputs an invalid password, PDAgentSecurityManager on a user PDA rejects a migration request from a shop computer. When a PDA user inputs a valid password, PDAgentSecurityManager further asks the user if he/she will accept the agent. Upon receiving user's permission, an agent migrates to a user PDA. Once an agent migrates to a user PDA, our security mechanism checks agent's resource access against a pre-defined access control list in the PDA to protect from illegal access. An agent in a PDA records customer action. (For instance, when a customer touches a product in a shop, embedded CPU in the product emits signal, and an agent records the signal.)

5.2. Component

To realize our scenario, we implemented the following components by using Voyager Object Request Broker. These components are built on a PC environment.

- **Shop computer (Agent platform)** -- A shop computer runs a mobile agent platform and dispatches an agent to customer's PDA. It also stores customer's profile and access history sent by an agent that has migrated to a customer PDA.
- **PDA** -- A PDA contains personal information such as a credit card number or Social Security number. In our implementation, a PDA runs Java VM as a mobile agent platform; thus, it can receive mobile agents from a shop computer. The PDA also runs our security

management system (PDAgentSecurityManager). When an agent migrates from a shop computer, PDAgentSecurityManager displays a dialogue box to allow incoming agent to access local resources.

- **Agent** -- An agent migrates from a shop computer to a PDA. An agent has two main functions. First, it receives a signal from products displayed in a shop. The signal includes product information, and the product information is recorded in an agent with a time stamp. Second, it communicates with a shop computer (after migrating to a customer PDA) and transmits user profile and an access log of a shop computer. To do this, an agent accesses a local user profile stored in the customer PDA.
- **Shop item** -- Shop products contain an embedded device that emits the product information. When a customer touches a product, its embedded device emits a signal to a PDA.

6. Implementation Lesson

We used Voyager Object Request Broker, a popular mobile agent platform, in our implementation. In our implementation, Voyager's original security mechanism was deactivated except for a function to identify a host where a mobile agent code migrated from.

Our PDAgentSecurityManager implements secure access to files and directories. We confirmed empirically that our implementation satisfies the security requirements for a mobile agent environment.

Our PDAgentSecurityManager may be applicable to other applications such as those using home computing devices or using built in computers in a car.

7. Conclusion

This paper explains requirements of security mechanisms for PDA applications and suggests a possible solution.

References

- [1] Ylitalo J., "Secure Platforms for Mobile Agents", Dept. of Computer Science, 2000, Helsinki University of Technology.
- [2] "Minstrel push system project", 1999, Technical University of Vienna. <http://www.infosys.tuwien.ac.at/Minstrel/>
- [3] Gong L., et. al., "Implementing Protection Domains in the Java Development Kit 1.2", 1998, JavaSoft, Sun Micro., Inc.
- [4] Charlie Lai Li Gong., et., al, "User Authentication and Authorization in the Java Platform", 1999, Sun Microsystems, Inc.
- [5] "Voyager Security 3.0 Developer Guide", ObjectSpace Inc, 1999.

Java, and all Java-related trademarks are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries