A SECURITY ARCHITECTURE FOR AGENT-BASED MOBILE SYSTEMS

N. Borselius[1], N. Hur[1], M. Kaprynski[2] and C.J. Mitchell[1]

[1]Royal Holloway, University of London [2]University of Strathclyde

ABSTRACT

Future mobile systems are expected to exploit the flexibility of agent-based software in a variety of ways. This will include agents providing both middleware and application-level functionality. Realising the full benefits of this innovative approach will require that security issues are properly addressed.

There are many security issues associated with agent-based systems; some of the most difficult to deal with arise when agents themselves can be mobile. There has been much recent interest in developing cryptographic protocols designed especially for securing mobile agents. However, this work has mainly been ad hoc in nature, i.e. it has not been developed in response to a thorough analysis of the security requirements for a particular agent application. This paper describes research intended to help rectify this gap by providing a detailed security architecture including a security model and a specification of security services provided within the model.

INTRODUCTION

In order for agent technology to be a viable solution for wide scale commercial applications, the associated security issues need to be properly addressed. Current work within the Mobile VCE Core 2 programme includes the development of a security model and architecture for agent-based mobile middleware and its management.

This paper describes results of this work, including a security model for an agent-based mobile system. This model is used as the basis of a security architecture for such systems. This in turn will be used as a framework to develop specific mechanisms and protocols to support security in this future mobile environment. These security techniques will provide security services designed to counter threats identified within the context of the security model.

The Mobile VCE model uses agent technology. Agents can exist on all kinds of hosts in the infrastructure, from the smallest devices (e.g. watch, PDA, or phone) to application servers and communications devices. While some agents will be able to move between platforms, others will always reside on the same platform. Whichever is the case, certain security functionality is required. The envisioned system will also include, and allow, devices not using agent technology.

FIPA (the Foundation for Intelligent Physical Agents) is a non-profit organisation aimed at producing standards for the interoperation of heterogeneous software agents. FIPA (1) has produced a high-level abstract architecture specification, which currently is in experimental stage. Security aspects are omitted from the FIPA model. The security model presented in this document is specified such that it will fit within the FIPA model when deployed in a telecommunications environment.

Contents of paper

This document contains two main sections. The first section specifies a detailed security model. The second section discusses some of the security services that need to be provided within the context of this model.

The security model is specified at four different levels of abstraction. This enables all types of interactions, including those within a device and those spanning the entire mobile system, to be modelled. The entities identified may sometimes coincide. However, it is important to ensure that different functionalities are separated within the security model, to ensure that it has sufficient generality. Note that our objective here is to model only those parts of security significance. The four levels of the model are as follows.

At the highest level are the **Involved parties**, including the *mobile device user*, the *mobile device owner*, and various *service providers*. At the next level down we consider the **Device structure**, including *agents*, an *agent execution environment*, a *subscription module* (as in a GSM SIM), and *communication services*. The next level of the model covers the **Agent execution environment**; this will include *agent communications services*, an *agent management entity*, and *agent security services*. Finally we consider the various parts of an **Agent** and how it interacts with its environment.

THE SECURITY MODEL

Involved parties

We first describe the high level entities that can be distinguished in the security model. The parties

can be thought of as distinct individuals or organisations. However, in practice one organisation can take the roles of more than one entity. It is also possible that not every party would be involved in a particular scenario.

Device user: also referred to as the user. The user is assumed to have physical control over the device, but may not necessarily be the same entity as the device owner.

Device provider: the manufacturer of the device. In order for the manufacturer to offer upgrades or additional services, the device provider will typically share a security context with the device. This security context will typically involve shared secrets and/or the provision of 'root' public keys.

Device owner: which might, for example, be the user or might be the employer of the user. Again there is a trust and possibly a cryptographic relationship with the device. The rationale for distinguishing the device owner from the device user is the fact that they might have different objectives. An employer might, for example, want to restrict the use of a device in order to protect itself from various threats, such as malicious code.

Service provider (SP): provides some kind of service, (e.g. transport service, information service, payment service, etc) including directory services and remote agent execution environments, to users and other SPs. A service provider may or may not have a pre-established contract with its clients.

'Home' service provider (HSP): i.e. an entity with which the device owner or user has a contractual relationship. This gives the provider of services to the device an identifiable entity from which he can extract payment (the HSP will then present a bill for all services provided to the device owner). Note that a device owner may have many HSPs.

Trust service provider (TSP): a special class of service provider providing trusted third party services, e.g. a CA (Certification Authority), an RA (Registration Authority), a timestamping service, an electronic notary, etc.

Agent provider: provides other parties with agents. The agent provider would typically be the developer of the agent. The agent provider can rely on its reputation or issue other guarantees concerning provided agents' behaviour. The agent provider and the agent owner can be different entities. One can envision a scenario where software developers provide (e.g. sell) agents to users. The users would then only need to provide the agent with certain credentials.

Agent owner: the entity on whose behalf an agent is executing. All parties can deploy agents to act on their behalf. These agents can execute on a device under the control of the agent owner as well as in other places within the infrastructure.

Device structure

We now describe the different parts of a device, including most importantly agents and the agent execution environment. It should be noted that devices and their resources can vary greatly, and depending on their purpose might not include all the components described here. The device described is a mobile one, but a similar structure can be assumed to exist within other devices in the infrastructure in which agents are executed.

A diagram of the device model is given in Figure 1. Only one agent is shown in the picture, but a device would typically have multiple agents executing in the agent execution environment.

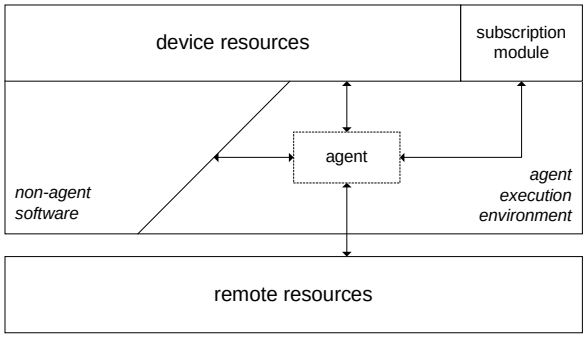

Figure 1 – Model for elements within a device

The elements of the device model are as follows.

Agent: executable code which is acting on behalf of its owner. All parties can use agents to represent themselves. An agent can execute in an environment that is under the control of its (i.e. the agent's) owner, or it may execute in an environment provided by another party. Agents may or may not be mobile. As illustrated in Figure 1, an agent can communicate with other entities over a network as well as with other functional blocks within the same device, including device resources, a subscription module, other agents, and other device software.

Agent execution environment: provides the resources needed for agents to execute and communicate with other agents as well as with other resources and entities. The agent execution environment, further described below, is regulated and controlled via mechanisms here referred to as agent control. Several agents can execute simultaneously within one environment.

Subscription module: a hardware device (e.g. smart card, USB token) which may interact with the device. An example of such a module is the GSM SIM. This module would typically be provided by an HSP, and share secret keys with a HSP and/or possess HSP-provided root public keys. Not every mobile device will be capable of directly interacting with such a module.

Remote resources: agents executing in a device can communicate with resources (e.g. agents and services) on other devices.

Non-agent software: software (applications and middleware functions) residing on the same device but not under the control of the agent platform control. Agents can communicate with such software just as such software would be able to access middleware services.

Device resources: refers to other resources residing in the device. Examples of such resources include a user interface, a hardware cryptographic processor, various cryptographic primitives that might be bound to the device (e.g. in the form of a cryptographic API), and communication resources.

Agent execution environment

The security functionality of the agent execution environment is now described. We only consider security functionality here, and a complete execution environment would be more complex. A diagram of the agent execution environment is given in Figure 2. It should be noted that, depending on the device on which the agent execution environment is residing, not all the elements of this model may exist. A device might, for example, not support the downloading of agents – in which case the agent mobility service would not exist. The complete agent execution environment will include the following elements.

Agent management and control: governs the security platform. This element is responsible for managing all agents executing on the platform including monitoring and controlling access to resources as well as communication between agents executing on the local platform.

Agent communications service: provides communications facilities to agents executing within the environment. This includes secure communication services.

Agent security service: includes security services provided by the environment to executing agents. For example, the environment may add a digital signature to data (signed with the private device signature key) at the request of an agent.

Agent mobility service: enables agents to send themselves (and associated stored state) to other devices. The service also includes functionality to assess received agents and any associated security information to decide if an agent shall be granted permission to execute on the platform. Agents requesting transfer to another platform will also be assessed for appropriate privileges here. If required, the agent mobility service can add platform specific information (e.g. agent trail) before transmission. The agent mobility service is responsible for setting up secure transmission channels when required for agent transfers.

Event logging service: logs security relevant events for storage in an audit trail. It may also provide security intrusion detection based on processing of recorded events.

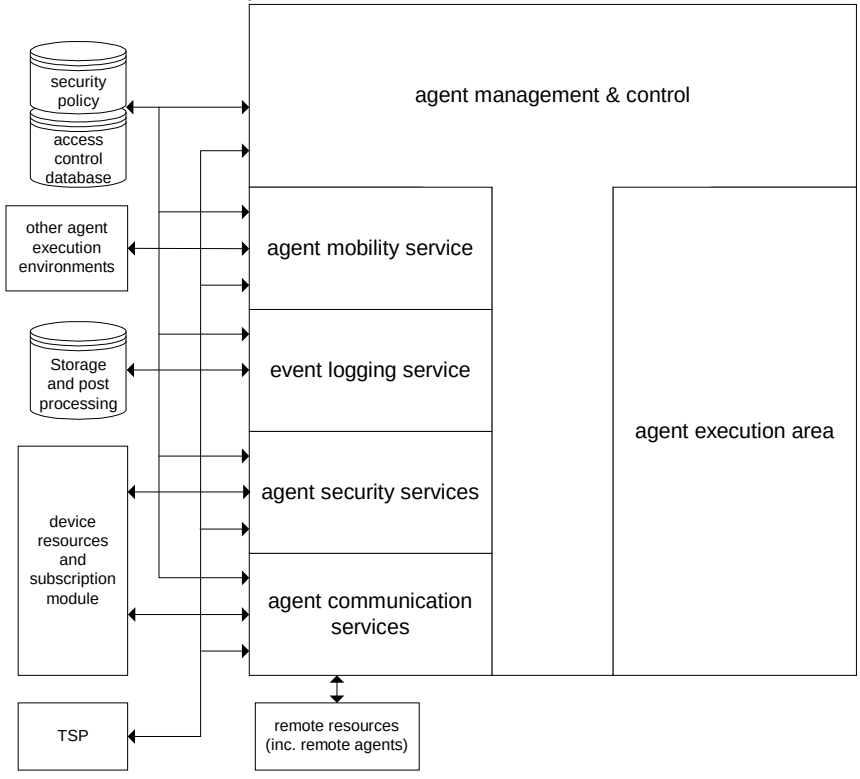

Figure 2 – Agent execution environment architecture

In addition to the described elements making up the execution environment, the following elements/functionality (which also appear in Figure 2) are part of the architecture.

The security policy and access control database regulate the behaviour of the security mechanisms. Information making up the security policy could include a rule base describing how, and when agents can be given access to the execution environment, and can interact with each other and their environment. Other examples include the specification of security related events for which log entries should be generated, and what controls should be implemented in order for an agent to start execution. The access control database contains information governing how various resources can be accessed by the various parties (this information could, for example, be in the form of an Access Control List (ACL) or a set of capabilities, or some combination of the two).

Remote systems can dispatch agents to the platform for execution. In the same manner, the agent execution environment can dispatch agents to execute in other environments.

Log storage and post processing manages and processes log data once generated.

Device resources and subscription module includes all kinds of resources (hardware and software) residing on the device.

Trust Service Provider (TSP) provides various trust services.

Remote resources are resources residing on other platforms with which agents can communicate, including other agents.

An agent

The various agent parts are likely to have different properties that need to be addressed via appropriate security mechanisms. The following distinctions between component parts of an agent can be made. Note that this agent model is designed for the purposes of security analysis only. As a result, important agent functionality may not be covered within this model.

Core executable part: executable information. This information is distinguished from other information to allow a user to obtain an agent from an independent party (agent provider).

Payloads: An agent is likely to have various kinds of payloads. Payloads can consist of non-executable data as well as executable information required by the agent to fulfil its task. Execution state, information supplied by the agent owner, and information collected at various hosts (for mobile agents), are all examples of payloads of an agent. In addition to this, an agent can obtain executable payloads to add agent functionality that is not part of the core executable part.

By separating agent parts in this way integrity verification values can be created where appropriate. The use of the above distinctions becomes particular apparent for mobile agents, but is also relevant for agents that are transferred to be executed on a platform not belonging to the agent provider. (We are here defining a mobile agent to be an agent that can move 'on its own initiative' and continue execution in the environment where it arrives.)

SECURITY SERVICES

Various classes of security services can be identified in the context of the security model. We focus on one such class, namely services to protect the execution platform. However, we also briefly review services to protect the agents.

Platform protection

We now describe security functionality addressing the protection of the execution environment. Note that agent execution environments will exist in various kinds of devices and the precise functionality, including security functionality, provided by the environment will also vary. Hence, the functionality described here may not be implemented in every device.

Logical Access Control. The platform needs to protect itself and its hosted agents against unauthorised access. Such functionality is often implemented in existing operating systems and execution environments. It can be implemented by using the sandbox concept, where executable code (e.g. an agent) would be able to do anything within the sandbox while any actions involving resources outside the sandbox are closely regulated and monitored. With this approach, the effort necessary to ensure the correctness of code received from outside the platform can be limited. However, in order to make full use of agents they need to be able to access resources outside the sandbox. Resources outside the sandbox include resources located on the same physical device as well as the ability to communicate with other devices/hosts/agents.

The execution environment has a security policy that regulates the requirements under which an access request will be granted. At this stage of the system design process it appears possible that an access control list (ACL) in combination with a capability-based scheme may be required for the provision of access control information. While an ACL is rather static in its nature, although dynamic changes to the list can be made, a capability scheme allows a subject to provide the required information at the point of an access request. A capability scheme based on public key

cryptography and a PKI will allow for the required delegation and transfer of rights between parties.

The agent management and control element is the main entity within the agent environment architecture enforcing access control. However access control is also part of the functionality of mobility, event logging, agent security, and agent communication services.

Authentication of foreign code. To provide flexibility a host needs to be able to receive, retrieve and execute agents. In fact, this applies to any downloadable code, and not only agents. In a mobile environment, with constant changes taking place, the ability to receive and execute software is likely to be very important.

As mentioned above, limited access can be given to an untrusted program in such a way that its behaviour can be regulated to prevent any potentially harmful behaviour. However, this is not enough to provide more powerful functionality. Applications will need to be given access to resources that, if misused, can result in unauthorised and potentially harmful actions.

Research on 'provably secure code' has been undertaken for several years. This research aims to verify that a piece of code is secure before it begins execution. However useful this would be, this is still very much an emerging area, and it is not clear how feasible it would be to restrict agents to those which have formal proofs of security. A more pragmatic approach is to trust a particular piece of software because one decides to trust the developer/supplier of the software. This technique is used in Java as well as in MExE (2). Using this technique we need ways of verifying that a particular piece of software does originate from a particular party. This can be done through cryptographic means.

When an agent arrives at the execution environment, various security checks are made by the mobility service. The following information associated with the agent can be verified and used by the mobility service in order to decide whether an agent should be granted execution rights: Agent owner, Agent provider, Required resources, Submitting host, Agent trail.

Platform communication. The platform will communicate with other entities in the infrastructure. For example, agents will be transferred between platforms and various trusted service providers will be contacted. Depending on the nature and sensitivity of the communication, various levels of protection are required.

Event logging. Unlike most security features which prevent security breaches, auditing enables follow-up when something goes wrong. The main purpose of an audit trail is to store information for later examination. Examples of applications for audit data include fraud detection, intrusion detection, and follow-up in case of failure or security breach. Audit information can also be used for real-time monitoring in order to take immediate actions in case of security violation.

The event logging service within the agent execution environment is responsible for generating audit trails. The security policy governs what is regarded as a security event to be logged. (Audit events can also be generated through the initiative of an agent.)

Once audit data is generated it needs to be stored and properly protected. Storage can be at the local platform but can also be at a trusted party or other remote site. If security of the platform is compromised it can be valuable to have transferred the audit data prior to the point of attack. This does, of course, involve network traffic, and hence is not always the best option.

Once stored, audit data can be analysed. The analysis can be automatic, e.g. by looking for known patterns or anomalies, or manual. The latter would apply particularly in the case of a security breach.

Agent protection

Analogously, security functionality is needed to protect agents executing in the agent execution environment. Issues to be addressed include: physical security, agent/platform authentication, agent mobility, agent communication, non-repudiation and event logging.

CONCLUSIONS AND ACKNOWLEDGEMENTS

In future work within Mobile VCE Core 2 we will develop specifications for security mechanisms and protocols to provide the security services specified in this security architecture.

The work reported in this paper has formed part of the Software Based Systems area of the Core 2 Research Programme of the Virtual Centre of Excellence in Mobile & Personal Communications, Mobile VCE, www.mobilevce.com, whose funding support, including that of the EPSRC, is gratefully acknowledged. More detailed technical reports on this research are available to Industrial Members of Mobile VCE.

REFERENCES

1. FIPA, *FIPA Abstract Architecture Specification.* Document number: XC00001J, 10/08/2001, Available online from www.fipa.org.
2. ETSI, *Mobile station application execution environment (MExE), Functional description*, Stage 2, 3GPP TS23.057 version 4.3. Release 4, October 2001.