Volume 4, Issue 1, January 2014 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Security Perspective for Agent Based Computing

Manoj Sharma*, **Keshav Jindal, Dr. B.K. Sinha**
Assistant Professor
NITRA Technical Campus
Ghaizabad, India

Abstract: Security is an important issue for deployment of applications based on software agent technology. However, not all applications require the same set of countermeasures, nor can they depend entirely on the agent system to provide them. Instead, countermeasures are applied commensurate with the anticipated threat profile and intended security objectives for the application. Our focus here is specifically on technical mechanisms, as opposed to procedural or non-technical measures. Such countermeasures can be integrated directly into an agent system, or incorporated into the design of an agent to supplement the capabilities of an underlying agent system they typically face a more severe set of threats than do static agents and, therefore, demand more rigorous countermeasures The use of agents has several advantages and a few disadvantages, including added security issues. In this paper, we survey several experimental agent systems under development and describe their current security mechanisms. We then develop a general agent model and discuss general security issues in that model. . This paper gives an overview of the threats associated with software agent systems and including the strengths and weaknesses of the techniques involved. Emphasis is on mobile software agents, since

Keywords: Hypermedia, Software based applications

1. INTRODUCTION

Agents were developed as representatives of a user, which a program could "spawn" and delegate to perform a task independently of the program within a network. An agent can be thought of as a digital secretary with a "brain" to which one can assign tasks, then send it off to perform them. Because of this abstraction, they have a benefit of being intuitive for non-computer-literate people [1], due to the fact that people deal with agents all the time. For example, when many people plan a complicated vacation, they often do not contact the airlines, hotels, and car rental agencies directly. Instead, they interact with a travel agent who communicates with these companies on their be an agent is a process that can move from one machine to another by its own initiative. This is different from traditional process migration, where the transfer is initiated by the system. The ability for a process to move from machine to machine gives agents advantages over traditional processes, but it also leaves agent based systems open to additional problems. Advantage of agents is that the machine where an agent originates does not have to be "online" in order for the agent to complete its task. This is especially useful in the case of mobile computing or in locations with poor or intermittent network access, where an agent's originating machine might be disconnected from the network for extended periods of time. In these cases, work can be done by the agent in the network during periods when the machine is disconnected. For example, a system can be set up for mobile computers using "network sensing tools and a docking system that allows an agent to transparently move between mobile computers, regardless of when the computers connect to the network" [2]

Agents can also be used by a client to make a network service available that did not exist previously. For example, if a client wanted to be able to perform queries on data that is located in multiple different database systems, an agent can be sent to one database system, perform a query, and send the results to another agent on another machine. That agent uses those results to make a query at a completely different database system. Those results can be relayed to another agent for further processing or to the user. In this way, a client can define custom services on a network. This could be done from the client's site without using agents, but there would be increased network traffic (and a corresponding decrease in performance) because all intermediate query results would have to be sent from the remote site.

As we have seen above, agents can also be viewed as another approach to building distributed applications, by providing another abstraction level to distributed programming in reducing the gap between distributed and centralized programming. They also potentially provide good support to mobile users, because they allow a simple model for disconnected execution. Disadvantages to agents include extra required system services on the machines on which they can run and added security issues. In order to perform the tasks we have described, there are certain attributes that agents (and the systems they run on) must possess:

1. Machines on a network might have different architectures and will probably have different available resources. An agent must be flexible enough to run under these different types of environments.
2. Agents have the ability to move to another machine to continue execution, they must have some facility to save their code and state, send themselves to another machine, restore their code and state, and reinstate execution on the new machine.

3. Agents often have results to relay back to the parent program, they need methods to do this.

Agent systems need to worry about such things as potentially malicious agents in systems, potentially malicious systems under which agents are running

2. CURRENT AGENT SYSTEM

With the help of network services provided by Operating System agent based applications can be developed However, it is much more convenient to use an *Agent Support Environment* (ASE) to provide the functionality that is common to many agent-based applications. Such systems supply services like communication, execution, and migration, allowing the developer to focus on the application details, leaving the infrastructure details to the ASE.
Although all ASE's have the same goal, different systems provide dissimilar services and are based on different models of computation. In order to give the reader an idea of the kind of the services provided by the ASE's, this section briefly describes three systems that have been receiving attention: Agent Tcl and Telescript.

2.1 Agent Tcl

Agent Tcl is an ASE that is under development at Dartmouth College. Its goal is "to address the weaknesses of existing transportable-agent systems" [3]. An Agent Tcl agent can currently only be written in Tcl, but Java support is being added into the system. Agent Tcl was designed to be easily extended by adding new languages and network services. Its architecture has four levels. The agents themselves form the highest level. The lowest level consists of an API for each network service used to move agents and to provide inter-agent communication. Currently, only TCP/IP is supported. The second level is a server that runs at each network site that maintains running agents. The server provides the migration facilities, communication primitives, and non-volatile store. All other services (such as scheduling, group communication, and directory) are provided by specialized agents.

In Agent Tcl, the agent's state is implicitly transferred with the agent when it jumps from one machine to another. To make this transparent state transfer possible, Agent Tcl uses a modified version of the Tcl interpreter. The modification consists of

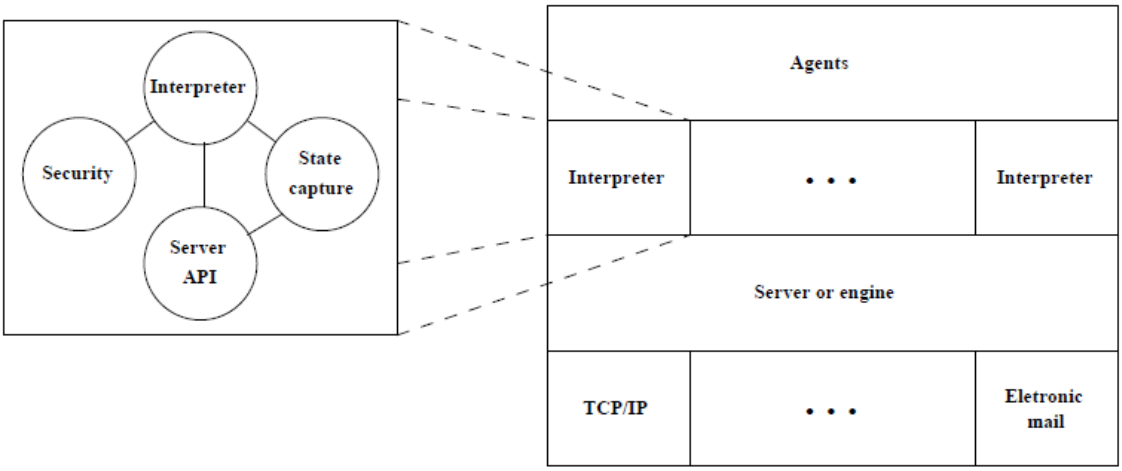

Figure 1. Agent Tcl Architecture

2.2 Telescript

It is object-oriented language that embeds facilities for building agent-based applications. Since it is a language-based solution, Telescript applications have to be written using only Telescript. No other language is supported. Telescript is commercially available by General Magic and is intended to be used as the basis for electronic marketplace networks. A Telescript program consists of a collection of *classes*. Classes have *proprieties*, which are either *operations* or *attributes*, and can be either *private* or *public*. The public proprieties of a class describe its *interface*. Classes are hierarchically organized by sub-classing, and a limited form of multiple inheritance is available. *Objects* are instances of classes.[4]
The three major concepts in the language are *agents*, *places*, and *go*. Both agents and places are processes. The difference between them is in their mobility: agents are mobile, can move about; places are stationary. Agents go to places, where they use places' services and/or interact with other agents that are in the same place. An agent always executes in the context of one or more enclosing place. Places provide a service API for agents to interact with. Places can be nested within other places. Once in the same Telescript place, two Telescript agents can *meet*. A meeting lets agents (in the same place) call each other's operations. Meetings motivate agents to migrate. They permit all interactions between two agents to be done in a local (rather then remote) fashion. One agent initiates the meeting using the instruction meet. The agent specified in the meet accepts or declines the invitation.
While in different places, two agents can communicate by exchanging messages through a connection. One agent requests the connection, the other accepts or rejects it. The initiating agent identifies the responding agent, the place the

latter occupies, and the quality of service required for the connection. Connections are a more traditional communication paradigm, but they can be useful for an agent-based application, especially when an agent needs some information from the user on who's behalf it is acting.

The Telescript language is interpreted rather than compiled. Telescript programs are interpreted by the Telescript engine, which provides all the functionality required by these programs, including the migration features. Therefore, the Telescript engine includes its own ASE. The Telescript engine is multitasked and isolates each agent or place in a separate Telescript process (not necessarily an OS process). Therefore, the purpose and progress of one agent have no influence, in general, over the purpose and progress of another.

3. SECURITY THREAT

Threats to the security agents generally fall into four comprehensive classes: disclosure of information, denial of service, corruption of information, and interference or nuisance. We use the components of an agent system to further delineate threats by identifying the possible source and target of an attack with respect to elements within that paradigm. It is important to note that many of the threats that are discussed have counterparts in classical client-server systems and have always existed in some form in the past (e.g., executing any code from an unknown source either downloaded from a network or supplied on floppy disk). Mobile agents simply offer a greater opportunity for abuse and misuse, broadening the scale of threats significantly. New threats arising from the mobile agent paradigm are due to the fact that, contrary to the usual situation in computer security where the owner of the application and the operator of the computer system are the same, the agent's owner and system's operator are different.

A number of models exist for describing agent systems , however, for discussing security issues it is sufficient to use a very simple one, consisting of only two main components: the agent and the agent platform. An agent comprises the code and state information needed to carry out some computation. Multiple agents cooperate with one another to carry out some application. [5]Mobility allows an agent to move or hop among agent platforms.The agent platform provides the computational environment in which an agent operates.

Four threat categories are identified: threats stemming from an agent attacking an agent platform, an agent platform attacking an agent, an agent attacking another agent on the agent platform, and other entities attacking the agent system. The cases of an agent attacking an agent on another agent platform and of an agent platform attacking another platform are covered within the last category, since these attacks are focused primarily on the communications capability of the platform to exploit potential vulnerabilities[6,7]. The last category also includes more conventional attacks against the underlying operating system of the agent platform.

3.1 Agent Against Agent Platform

The mobile agent paradigm requires an agent platform to accept and execute code developed elsewhere. An incoming agent has two main lines of attack. The first is to gain unauthorized access to information residing at the agent platform; the second is to use its authorized access in an unexpected and disruptive fashion. Unauthorized access may occur simply through a lack of adequate access control mechanisms at the platform or weak identification and authentication, which allows an agent to masquerade as one trusted by the platform. Once access is gained, information residing at the platform can be disclosed or altered. Besides confidential data, this information could include the instruction codes of the platform. Depending on the level of access, the agent may be able to completely shut down or terminate the agent platform. Even without gaining unauthorized access to resources, an agent can deny platform services to other agents by exhausting computational resources, if resource constraints are not established or not set tightly. Otherwise, the agent can merely interfere with the platform by issuing meaningless service requests wherever possible.

3.2 Agent Platform Against Agent

A receiving agent platform can easily isolate and capture an agent and may attack it by extracting information, corrupting or modifying its code or state, denying requested services, or simply reinitializing or terminating it completely. Extracting electronic cash directly from the agent is one simple example. An agent is very susceptible to the agent platform and may be corrupted merely by the platform responding falsely to requests for information or service, altering external communications, or delaying the agent until its task is no longer relevant. Extreme measures include the complete analysis and reversing engineering of the agent's design so that subtle changes can be introduced. Modification of the agent by the platform is a particularly insidious form of attack, since it can radically change the agent's behavior (e.g., turning a trusted agent into malicious one) or the accuracy of the computation (e.g., changing collected information to yield incorrect results).

3.3 Agent Against Other Agents

An agent can target another agent using several general approaches. These include actions to falsify transactions, eavesdrop upon conversations, or interfere with an agent's activity. For example, an attacking agent can respond incorrectly to direct requests it receives from a target or deny that a legitimate transaction occurred. An agent can gain information by serving as an intermediary to the target agent (e.g., through masquerade) or by using platform services to eavesdrop on intra-platform messages. If the agent platform has weak or no control mechanisms in place, an agent can access and modify another agent's data or code, or interfere with the agent by invoking its public methods (e.g., attempt buffer overflow, reset to initial state, etc.). Even with reasonable control mechanisms in place, an agent can attempt to send messages repeatedly to other agents in an attempt to deny them the ability to communicate.

3.4 Other Entities Against Agent System

Even assuming the locally active agents and the agent platform are well behaved, other entities both outside and inside the agent framework may attempt actions to disrupt, harm, or subvert the agent system. The obvious methods involve attacking the inter-agent and inter-platform communications through masquerade, (e.g., through forgery or replay) or intercept. For example, at a level of protocol below the agent-to-agent or platform-to-platform protocol, an entity may eavesdrop on messages in transit to and from a target agent or agent platform to gain information. An attacking entity may also intercept agents or messages in transit and modify their contents, substitute other contents, or simply replay the transmission dialogue at a later time in an attempt to disrupt thesynchronization or integrity of the agent framework. Denial of service attacks through available

4. A MODEL FOR AGENT BASED COMPUTING

After describing the current implementation of agent systems and discussing the way these systems tackle certain security problems. In this section, we will define a generalized ASE model by providing many of its features. This model will then be used to explore the main security problems in agent based systems based on a classification presented in [2].

4.1. The Model

An agent model that can be abstracted from the main descriptions in the previous sections is based on being able to build an ASE that is complete. This support system, as one can notice from the different implementations that were presented, needs to support the following main features, which define what an agent is by defining what its capabilities are.

4.1.1. Creation

This involves the ability of certain agents in creating other agents, to perform certain specified tasks. The resulting agents can be created to run localy or remotely.

4.1.2. Execution

Agents need to be able to execute in order to carry out their tasks. Execution is usually done through an interpreter that supports a runtime environment within which an agent can function. This environment is usually specified when the agent starts up in the machine.

4.1.3. Resource Access

This should implement ways in which an agent accesses certain local resources as well as resources being carried with the agent. What this implies is that there are two types of resources present within the execution environment, resulting in two very different security concerns. The types of resources being considered could include physical resources as well as logical resources and controlling access to those resources can be done using many different mechanisms including access control or capabilities. An example would be restricting access to the CPU and other resources through checking during the interpretation/execution phase.

4.1.4. Migration

This implements how an agent can move around from one machine to the next on its own initiative, while carrying out its task. This aspect of the architecture is what makes agents very interesting and what gives them unique properties. This migration process can be affected by the move being unrestricted (i.e. from any one machine to another) or restricted (i.e. from the home machine to the server machine and back). It also involves how and in what form the data is being carried around within the agent. This point refers to migrating agent resources (in addition to code migration). There is also the issue of implicit versus explicit transfer of an agent's state.

4.1.5. Communication

This takes care of the fact that agents need to interact with other agents to provide their services. There are two types of communication that could be available. One type would be local while the other is remote. Communication may also serve as a way to transfer objects (and agents) between other agents. This also includes the issue of creating synchronization primitives between agents that can help in organizing their efforts when obtaining a certain service in a cooperative manner.

4.1.6. Language Support

This involves the issue of interpreted versus compiled languages. It also involves support for just one specialized language versus the support of many different languages to be used in programming agents.

5. CONCLUSION

The area of agent based computing security is rapidly improving. The traditional orientation toward host-based security persists and, therefore, available protection mechanisms tend to focus on protecting the agent platform. Emphasis is beginning to move toward developing techniques that are oriented toward protecting the agent, a much more difficult problem. The initial efforts are encouraging and will hopefully continue. There are a number of agent-based application domains for which basic and conventional security techniques should prove adequate. However, applications are expected to require a more comprehensive set of mechanisms and a flexible framework in which to apply a subset of selected mechanisms to meet their needs. In many ways, the success of software agent technology in general, whether this evolutionary branch of computing successfully propagates forward or eventually dies out, remains to be seen.

REFERENCES

1. Dag Johansen, Robert van Renesse, and Fred Scheidner. *Operating system support for mobile agents*. In Proceedings of the 5th IEEE Workshop on Hot Topics in Operating Systems, 1995. http:// cs-tr.cs.cornell.edu/TR/CORNELLCS:TR94-1468.

2. Robert S. Gray et al. *Mobile Agents for Mobile Computing*. 1996. ftp://ftp.cs.dartmouth.edu/TR/TR96-285.ps.Z.
3. Robert S. Gray. *Agent Tcl: A Transportable Agent System*. In Proceedings of the CIKM Workshop on Intelligent Information Agents, Fourth International Conference on Information and Knowledge Management (CIKM 95), December 1995. http://www.cs.dartmouth edu/~agent/papers/cikm95.ps.Z.
4. General Magic. *Telescript Technology: Mobile Agents*. 1996. http:// www.genmagic.com/Telescript/Whitepapers/wp4/whitepaper- 4.html.
5. Agent Management, FIPA '97 Specification, part 1, version 2.0, Foundation for Intelligent Physical Agents, October 1998 <URL: http://www.fipa.org/spec/FIPA97.html>
6. Mobile Agent System Interoperability Facilities Specification, Object Management Group (OMG) Technical Committee (TC) Document orbos/97-10-05, November 1997. <URL: http://www.omg.org/techprocess/meetings/schedule/Technology_Adoptions.html#tbl_MOF_Specification>.
7. J. E. White, Mobile Agents, in J. M. Bradshaw (Ed.) Software Agents, AAAI/The MIT Press, 1997.

