Preface

New paradigms can popularize old technologies. A new "standalone" paradigm, the electronic desktop, popularized the personal computer. A new "connected" paradigm, the web browser, popularized the Internet. Another new paradigm, the mobile agent, may further popularize the Internet by giving people greater access to it with less effort.

Mobile Agent Paradigm

The mobile agent paradigm integrates a network of computers in a novel way designed to simplify the development of network applications. To an application developer the computers appear to form an electronic world of places occupied by agents. Each agent or place in the electronic world has the authority of an individual or an organization in the physical world. The authority can be established, for example, cryptographically.

A mobile agent can travel from one place to another subject to the destination place's approval. The source and destination places can be in the same computer or in different computers. In either case, the agent initiates the trip by executing a "go" instruction which takes as an argument the name or address of the destination place. The next instruction in the agent's program is executed in the destination place, rather than in the source place. Thus, in a sense, the mobile agent paradigm reduces networking to a program instruction.

A mobile agent can interact programmatically with the places it visits and, if the other agents approve, with the other agents it encounters in those places. An agent typically travels to obtain a service offered by an agent in a distant place. An agent might travel from a place in a personal computer to a "theater ticketing place" in a network server. Upon arrival, the agent might purchase theater tickets by interacting with a resident "theater ticketing agent". Thus agents can be instruments of electronic commerce.

Mobile Agent Advantages

The familiar remote procedure call (RPC) paradigm uses networks to carry messages —data— that request and confirm services. A client orchestrates the work of a server with a series of requests, sent from client to server, and responses, sent from server to client. To delete from a file server all files two weeks old might require one request to list the files and their modification dates and another to delete each sufficiently old file. Software on the client decides which files to delete. Deleting n files requires $2(n+1)$ messages.

The new mobile agent paradigm uses networks to carry objects —data and procedures— that are to be executed in the presence of service providers. A client orchestrates the work of a server by sending to the server an agent whose

procedure makes all of the required requests when it's executed. Deleting the old files —no matter how many— requires moving just one agent between computers. All of the orchestration, including the analysis that decides which files are old enough to delete, is done "on-site" at the server.

One advantage of mobile agents is performance. While two computers in an RPC network require ongoing communication for ongoing interaction, two computers in a mobile agent network can interact without the network's help once it has moved an agent that embodies the desired interaction from one computer to the other. The lower the network's throughput or the higher its latency or cost, the greater the performance advantage.

Another advantage of mobile agents is automation. A user can direct an agent to carry out a long sequence of tasks and then send the agent on its way. The tasks may require the agent to travel to many servers. The user's computer need be connected to the network only long enough to allow the agent to leave and perhaps later return. Thus mobile agent networks enable users to automate tasks that today they must perform interactively.

A third advantage of mobile agents is ease of software distribution. Agents enable applications to distribute themselves among the various computers on which they must execute. If installed on a client computer, an application can expand to encompass one or more servers. If installed on a server, an application can expand to encompass any number of client computers (in order to offer a service "door to door"). Thus a mobile agent network, like a personal computer, is an open platform for application developers.

Mobile Agent Applications

Users delegate to agents extended or complicated tasks they'd rather not perform themselves. Among the special talents of agents are watching, searching, and arranging.

Watching. In an investment application, a user's agent might monitor the stock market and notify the user when a specified stock reaches a specified price. The agent travels from a client computer to a stock server where it waits for the event to occur.

Searching. In a shopping application, a user's agent might determine the lowest price at which a specified product is sold. The agent travels from a client computer to a directory server and then to the commerce servers of merchants it selects from the directory.

Arranging. In an entertainment application, a user's agent might arrange a "night on the town" involving dinner and the theater, selecting a restaurant with a specified cuisine and price range and timing the reservation to allow for travel to the theater. The agent travels from a client computer to a restaurant reservation server and a theater ticketing server.

As in the last example, mobile agents can create new services by combining existing ones. In this way particularly, mobile agent networks are open platforms for developers.

The Technical Challenge

The mobile agent paradigm has problems as well as promise. Most of the problems have to do with safety and security. The mobile agent paradigm is most platform-like and so delivers the greatest value not when my agent visits my place, but when it visits yours. Before you can allow this, you must be sure that my agent, for example, can't access information you didn't intend for it to have, can't accidentally go into an infinite loop and so squander your resources, and can't deliberately interfere with the agents of other users.

In this book, you'll hear from some of the most prominent researchers in the mobile agent field. They'll tell you what they've accomplished and what remains to be accomplished. You'll gain a good understanding of the mobile agent paradigm —especially an understanding of how the paradigm can be made safe. Perhaps you'll be inspired to take on some of the remaining work yourself. We'd welcome your help.

Sunnyvale, California
March 1998

James E. White
Chief Technology Officer
General Magic, Inc.

Table of Contents

Introduction

Mobile code technologies are receiving a great deal of interest from both industry and academia. The ability to move computations across the nodes of a wide area network allows deployment of services and applications in a more flexible, dynamic, and customizable way with respect to the well-known client-server paradigm. Yet, the wide acceptance of the mobile code approach is hampered by the security issues that arise when executable content and associated execution state (often referred to as *agents* or *mobile computations*) are moved among different computational environments (called *places* or *sites*).

Mobile agent systems provide a computing infrastructure upon which distributed applications belonging to different and potentially untrusted users can execute concurrently. Moreover, the sites composing the infrastructure may be managed by different authorities (e.g., a university or a company) with different and possibly conflicting objectives and may communicate across untrusted communication infrastructures, e.g., the Internet. In this scenario several attacks are possible. Unauthorized people may eavesdrop network traffic to access or modify agents while the agents are in transit over the network. Agents may attack the sites supporting their execution to gain privileged or unrestricted access to resources and private information (e.g., the password file of a UNIX machine). Attacks may also try to misuse the services offered by a site to probe and exploit the vulnerabilities of other systems. The Internet worm showed the effectiveness of this approach ten years ago.

Some of these security concerns have been studied by the distributed systems community for a long time. However, mechanisms and technologies developed to secure communication and control access to resources must be adapted to take into account mobility. In addition, the mobile code approach introduces a new security issue: the protection of mobile agents from malicious sites. In fact, sites could tamper with agents' code or state in order to disclose private information, gain competitive advantage with respect to other sites, or "brainwash" agents so that they will attack other sites (or agents). This new security issue is particularly challenging since it is difficult—if not impossible—to protect an executing program from the interpreter responsible for its execution.

Research on mobile agent security has delivered a number of mechanisms that follow different approaches in addressing the aforementioned security issues. Some of these mechanisms have been embedded into mobile agent systems to provide secure platforms for network programming. This book presents a collection of invited papers from researchers that provided insights and novel ideas to secure the dangerous world of mobile agents.

Overview

The book is organized in four parts: (I) Foundations, (II) Security Mechanisms, (III) Mobile Code Systems, and (IV) Active Content and Security.

Part I contains chapters providing a discussion of security issues and theoretical work on mobile code security. The chapter by David Chess presents a comprehensive overview of the security issues involved in mobile code systems. The chapter by James Riordan and Bruce Schneier presents a cryptographic technique to generate secret keys in hostile environments. The chapter by Dennis Volpano and Geoffrey Smith describes how security influences the design of languages supporting code mobility. It is followed by a chapter by Tomas Sander and Christian Tschudin that discusses the use of encrypted function computation in mobile code systems.

Part II contains chapters describing mechanisms that address security problems associated with mobile code. The chapter by George Necula and Peter Lee describes the proof-carrying code approach, which associates a mobile agent with a formal proof of its behavior. The chapter by Fritz Hohl describes an approach to protect mobile agents against malicious sites that is based on code obfuscation. The chapter by Shimshon Berkovits, Joshua Guttman, and Vipin Swarup describes an architecture to achieve authentication and protection of mobile agents. The chapter by Giovanni Vigna describes cryptographic tracing, a mechanism to detect illegal tampering with code and state of a moving agent.

Part III contains descriptions of mobile code systems and how security mechanisms have been introduced into their architecture. The chapter by Robert Gray, George Cybenko, David Kotz, and Daniela Rus describes the security architecture of D'Agents, a multiple-language mobile agent system. The chapter by Günter Karjoth, Danny Lange, and Mitsuru Oshima describes how security issues have been tackled in the IBM Aglets system. The chapter by Li Gong and Roland Schemers describes new features to protect and sign Java objects.

Part IV describes approaches to the problem of managing executable content in the World Wide Web. The chapter by John Ousterhout, Jacob Levy, and Brent Welch describes Safe-Tcl, an extension of the Tcl language for safe interpretation of executable content coming from untrusted sources. The chapter by Flavio De Paoli, Andre Dos Santos, and Richard Kemmerer describes the vulnerabilities of existing "secure" WWW browsers.

Acknowledgments

We would like to thank the anonymous reviewers for their help and excellent reviewing. In addition, we thank Carlo Ghezzi, Arno Jacobsen, Ralph Keller, Gian Pietro Picco, Christian Tschudin, Jan Vitek, and Sharon Webb for their help in the editing of this volume.

Milano, February 1998 Giovanni Vigna