

A Secure System Architecture for Software Agents: The Virtual Secretary Approach

Gunnar Hartvigsen*
Arne Helme**
Stig Johansen*

Broadcast Technical Report ????
Esprit Basic Research Project 6360
June 15, 1995

BROADCAST

Basic Research On Advanced Distributed Computing: from Algorithms to Systems

Esprit Basic Research Project 6360

BROADCAST will develop the principles for understanding, designing, and implementing large scale distributed computing systems (LSDCS), in three broad areas:

- **Fundamental concepts.** *Evaluate and design computational paradigms (such as ordering, causality, consensus); structuring models (groups and fragmented objects); and algorithms (especially for consistency).*
- **Systems Architecture.** *Develop the architecture of LSDCS, in the areas of: naming, identification, binding and locating objects in LSDCS; resource management (e.g. garbage collection); communication and group management. Solutions should scale and take into account fragmentation, and recent technological developments (disconnectable devices and 64-bit address spaces).*
- **Systems Engineering.** *Efficiently supporting the architecture, exploiting the concepts and algorithms developed earlier, as kernel and storage support for numerous fine-grain complex objects; and programming support tools for building distributed applications.*

The BROADCAST partners are: École Polytechnique Fédérale de Lausanne (EPFL, Lausanne, Switzerland), Université Joseph Fourier, Institut d'Informatique et de Mathématiques Appliquées de Grenoble (IMAG, Grenoble, France), Instituto de Engenharia de Sistemas e Computadores (INESC, Lisboa, Portugal), Institut National de Recherche en Informatique et Automatique (INRIA, Rocquencourt, France), Institut de Recherche en Informatique et Systèmes Aléatoires (IRISA, Rennes, France), Università di Bologna (Italy), University of Newcastle-upon-Tyne (United Kingdom), and Universiteit van Twente (the Netherlands).

For information, copies of the Broadcast Technical Reports, or to be put on the Broadcast mailing list, please contact: Broadcast Secretariat, Department of Computing Science, University of Newcastle-upon-Tyne, Claremont Road, Newcastle-upon-Tyne NE1 7RU, UK. Tel.: +44 (91) 222-7827. Fax: +44 (91) 222-8232. E-mail: nick.cook@newcastle.ac.uk.

The Broadcast Technical Reports Series

- 1 *SSP Chains: Robust, Distributed References Supporting Acyclic Garbage Collection*, by Marc Shapiro, Peter Dickman, and David Plainfossé, November 1992
- 2 *Consistent Global States of Distributed Systems: Fundamental Concepts and Mechanisms*, by Özalp Babaoğlu and Keith Marzullo, January 1993
- 3 *Understanding Non-Blocking Atomic Commitment*, by Özalp Babaoğlu and Sam Toueg, January 1993

Broadcast Technical Reports can be ordered in hardcopy from the Broadcast Secretariat. They are also available electronically: by anonymous FTP from server [broadcast.esprit.ec.org](ftp://broadcast.esprit.ec.org) in directory `projects/broadcast/reports`; or through the CaberNet Infrastructure AFS filesystem, in directory `/afs/research.ec.org/projects/broadcast/reports`.

A Secure System Architecture for Software Agents: The Virtual Secretary Approach

Gunnar Hartvigsen*[†] Arne Helme**[‡] Stig Johansen*

*Department of Computer Science, Inst. of Mathematical and Physical Sciences, University of Tromsø, N-9037 Tromsø, Norway. **Faculty of Computer Science/SPA, University of Twente, 7500 AE Enschede, The Netherlands.

Software agents in the form of worm-like programs represent a great potential for location independent processing and problem solving. Unfortunately, worm-like programs in the traditional sense are, when traversing a computer network, among the most uncontrollable software constructions. This paper presents the Virtual Secretary system architecture which is an attempt to utilize the potential processing and problem solving strength of software agents while eliminating most of the security risks related to such programs. This is done by the construction of an environment for software agents, including propagation mechanisms, mechanisms for authentication and control, and common user software. A key element in this approach is the propagation of the Virtual Secretary's user model (i.e., a non-executable initialization and control software). A necessary assumption is that an authorised Virtual Secretary body process is present at all possible target hosts in the network, or if not, a secure way to start a new Virtual Secretary body process at a remote host exists.

1 Introduction

Global internetworking has enabled many new services and opportunities, including the use of different kinds of software agents. As argued by Riecken [2], the term *intelligent agents* has become quite popular, both in the research and the commercial world. Announcements of products like Apple Newton's agent software and General Magic's messaging agent are evidence of the commercial interest in agent research and development. However, there is no consensus on the terminology. The agents have been denoted as intelligent agents, intelligent interfaces, adaptive interfaces, knowboths, softboths, personal agents, network agents, etc. One classification scheme used is whether the agent can propagate in a computer network or is tied to one specific computer. In the Virtual Secretary project, agents propagate in a global computer network. Currently, we focus on three secretarial tasks: local information filtering, global information filtering, and user environment personification.

The main problem with software agents in the form of worm-like programs is that unknown programs which propagate to a computer must be trusted. Since we (as the user) have no control over these programs they can be programmed by computer criminals, or there could be problematic bugs in them.

[†] On sabbatical leave at University of Twente, 1994-95. This work was supported by the Science Foundation of Norway (No. 100425/410).

[‡] This work was supported by Xerox EuroPARC, Cambridge, Olivetti Research Laboratory, Cambridge, Digital Equipment Corporation, and by the European Communities' ESPRIT Programme through BRA project 6360.

It is, of course, possible to accept only active programs from trusted users with whom we have an authentication agreement. Given all the possibilities for computer fraud the computer network provides, a user or system manager still takes a risk in letting active programs penetrate the system.

The ordinary way to deal with security problems is to adopt a *security policy* which is designed to ensure appropriate levels of security for the system's activities. A security policy is enforced through *security mechanisms*. This approach implies no additional precautions to meet new security challenges from software agents. The fundamental problem is that we do not have any guarantees for what might happen.

Another approach to the construction of software agents, taken by the Virtual Secretary project, is to exclude as many security risks as possible through the adoption of a different architectural approach to system design, especially the propagation scheme. In this approach an environment for software agents is created, including

- propagation mechanisms,
- authentication and control mechanisms,
- and common user software.

To reduce the security problem we have chosen to propagate passive code in the form of user models. This approach is based on the assumption that a Virtual Secretary body process exists on the target host, and that a description of this process is well known. A user model contains a description of the mission, the user (including the user's history, current tasks, preferences, etc.), administrative and control data, etc. The user model, or, if appropriate, part of the user model will initialize the body process on the remote host and continue the mission. The assumption of the existence of a body process will, to a certain degree, also eliminate the previous mentioned fundamental problem related to software agents.

This paper presents major security problems with software agents, outlines the Virtual Secretary system architecture and discusses the approach taken in the Virtual Secretary project to improve the security.

2 Security Problems with Software Agents

It is well known what risks worm-like programs can introduce on a network of computers. It is, perhaps, less understood what additional threats that are of concern for worm-like programs. A serious threat, for example, is that a worm-like program itself can be compromised by the host it enters. This host can therefore also impersonate the user that a worm-like program acts on behalf of. In the following some more important security related aspects of allowing worm-like programs, such as software agents, to be executed on a network of computers are described.

From a system oriented point of view the main security risk related to the use of worm-like programs is the amount of privileges that can be obtained when a new system is entered.

The security threats with the introduction of software agents are also similar to those related to the installation of a new program on a computer system. Unless explicit knowledge about the behaviour of a software agent is established (e.g., by only using certified software), it is in practice impossible to safeguard against unexpected behaviour. Also observe that in some environments (e.g., Unix-like environments) a program under execution obtains the rights of the user who invokes it.

Software agents have some of the same properties as batch oriented processes. A well-known security problem with such processing in a distributed environment is how to refresh and revoke the rights of long term jobs. Existing network authentication systems, such as Kerberos [3], issue session tokens with a lifetime (which is tailored to user logins) that eventually will time out. It is therefore highly probable that the use of such tokens will result in scenarios where timeouts occur before the life cycle of a software agent terminates. It is also not always the case that an accountable user will be available to refresh these rights. On the other hand, the use of session tokens with longer lifetimes introduces the problem of how to revoke the rights of a (misbehaving) software agent.

We distinguish between two different extremes of software agent architectures. The first is based on the traditional Unix model where an agent is a migrate-able process. In a Unix environment process migration is the mechanism most commonly used to propagate the state of worm-like programs to new machines. The other approach is to use a body process that can interpret information received over the network. The body process is a program that may already exist on a computer system and, when executed, is capable of and willing to receive instructions (subject to proper authorisation and access control). Obviously, the advantage of this approach over the former is that body processes may be certified in advance, or may in some other way be constructed to behave as a proper body for a worm-like program. In the rest of the discussion we will only consider the latter approach because it seems unreasonable to build a secure software agent architecture upon the traditional Unix model with process migration as the mechanism for state transfer. Note that the approach discussed is part of an environment for software agents.

A software agent must act on behalf of some *principal*¹ that can be held responsible for its actions. Let us for now assume that the user model of a software agent is defined by a sequence of statements where each statement needs to be authorised by the body process. Then it is important to establish what principal the software agent acts on behalf of since this will influence on what kind of access rights each individual statement will gain on a remote system. Depending on the policies in use, it is also likely that the principal a software agent acts on behalf of, will have different privileges in different administrative domains.

Security problems of the use of software agents can be divided into those that are related to network communication, and those that are related to execution on a remote and possibly hostile host:

- Issues related to the propagation of a software agent between different computer systems. This is a question of network security. Here we are concerned with:
 - Authentication of principals. It is of interest to know what principal a software agent acts on behalf of, what computer system it originated from and what system it tries to enter. Mutual authentication between all these principals may be important for some environments.
 - Integrity of communication, and of the state of a software agent.
 - Confidentiality. Protection against eavesdroppers.
 - Traffic analysis.
- Issues related to an agent's execution on a computing system. Carrying of execution in a protected environment. This is a question of access con-

¹ We use the term *principal* to denote the entity that will be held accountable for the resources used by the software agent. This can be a computer system, a user, or an organisational unit.

control decisions:

- Authentication of body process.
- Integrity of the state information.
- Access control decisions.

Thus, it may seem like proper verification of the identity of the principal a worm-like program acts on behalf of is fundamental to the realisation of secure worm-like programs.

Most likely a software agent must carry some kind of session token that proves its origin and identity. Its rights must be derived from this identity and the policy enforced on each individual host it enters. If this delegation token is part of the state information, it is a risk that possibly untrusted hosts may steal it and thus compromise whatever principal the software agent acts on behalf of.

Auditing and accounting mechanisms are also important. A Virtual Secretary carries a profile of the user it acts on behalf of. Knowledge about this user profile can be useful input for auditing purposes since deviations from an expected user profile can be audited. Such information can be valuable in order to detect attempts to violate the security of some of the computer systems or the worm-like programs.

To summarize, secure propagation of a software agent relies on the presence of mechanisms that can authenticate computer systems mutually (source and target systems), and mechanisms that can authenticate all principals that these agents act on behalf of. In addition, a software agent must trust its local computer system to be honest, i.e., to perform a proper authentication of a destination system. Authorised tasks on a remote system will be derived from the principal a software agent acts on behalf of. Cryptography can be used to provide secure communication between software agents on different computer systems.

3 The *Virtual Secretary*

The Virtual Secretary project focuses on the construction of an environment for worm-like programs which can conduct simple secretarial tasks. The first version of the Virtual Secretary concentrates on tasks that do not involve communication with other users' Virtual Secretaries. More specific, we focus on three secretarial tasks: Local information filtering (e-mail, news and diary); Global information filtering (file retrieval and World Wide Web); and, user environment personification. We consider the tasks above to be some of the most important and most frequent attended tasks this kind of secretary needs to take care of. The selection criteria of the tasks addressed is the tasks' need for and utilization of the user model.

To realize this kind of software we need a way to propagate the software to a remote host. In other words, we need to clone the Virtual Secretary and to emigrate the clone to the remote host. Traditionally, this has been done in a straight forward way with the transference of the whole process to the remote host. Unfortunately, this approach exposes the system for far more security risks than tolerable. This means that software based on this kind of platform hardly will be widely accepted. Therefore, a major goal in the Virtual Secretary project is to provide a secure propagation of worm-like programs. We want to secure both the owners (i.e., the users) as well as other users connected to the different hosts. This goal is expected to be reached through the construction of an environment for software agents including: propagation mechanisms, authentication and control mechanisms, and common user software.

The propagation mechanism is based on the use of user models, e.g., as in adaptive systems. In adaptive systems, the user model is the system's description of the user and the user's tasks. Finin [1] argues that "a user model is that knowledge about the user, either explicitly or implicitly encoded, which is used by the system to improve the interaction." The knowledge is instantiated in the system's user model, i.e., the representation of the user (see Figure 2). Williges [4] splits user models into two groups: *conceptual models* (cognitive process models, cognitive structure models, cognitive strategy models) which mainly focus on representation of cognitive processes, and *quantitative models* (behaviour models, ergonomic models, computer-based simulation models, static models) which deal with numerical representation of the user's execution. In the Virtual Secretary, we use an extended user model. To be able to transfer a task from one host to another, we need to include all user specific parts in the user model.

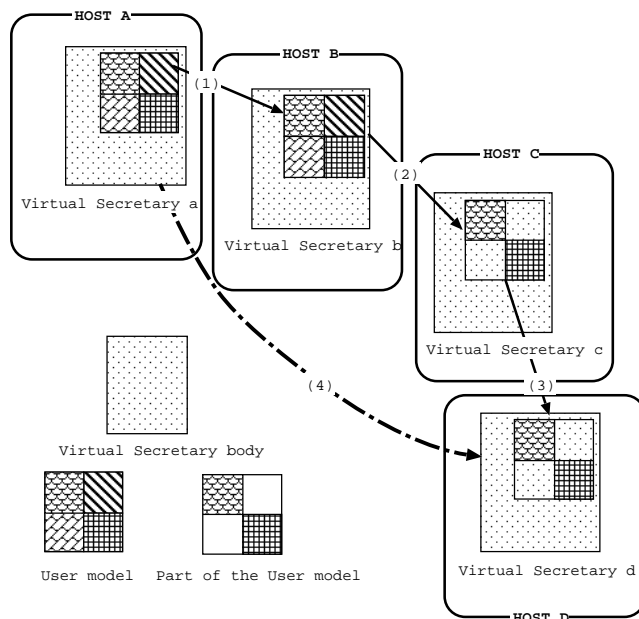


Figure 1. Propagation of the *Virtual Secretary* user model. Assumptions: (1) Propagation. (2) Mission to a host holding a *Virtual Secretary* body process. (3) Mission to a host not holding a *Virtual Secretary* body process. (4) Copy of a *Virtual Secretary* body process.

Figure 1 illustrates how the *Virtual Secretary* propagates to a remote host. In (1) we have that the hosts A and B both hold a *Virtual Secretary* body process (*Virtual Secretary a* and *b*). In addition, *Virtual Secretary a* holds a user model. Given that the user moves to host B only the user model needs to be propagated. In (2), we have a similar situation as for (1) except that now the *Virtual Secretary b* is sent on a mission to perform a specific task, e.g., information gathering. Now, only part of the user model might need to be propagated. Arrow (3) illustrates the situation that the *Virtual Secretary*, in order to complete its mission, needs to propagate to host D which does not hold a *Virtual Secretary* body. Since the allowance of this kind of operation heavily increases the risks, it should only be allowed after a thorough security control. If host C is a portable computer it might be more convenient to get the copy of the *Virtual Secretary* body from, e.g., host A (as shown in arrow (4)) and the user model from host C.

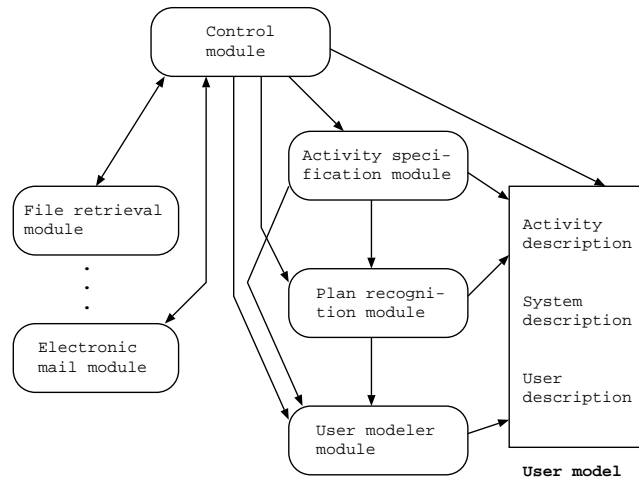


Figure 2. The *Virtual Secretary* system architecture.

As illustrated in Figure 2, a Virtual Secretary consists of the following modules:

Activity specification module This module offers manual specification of the job that is going to be conducted. E.g., when the user wants the Virtual Secretary to fetch information from a remote host he has to specify the task manually.

Plan recognition module The plan recognition module tries to recognize any pattern in the user's interaction with the Virtual Secretary. A recognized pattern will be restored for later use, e.g., in information gathering.

User modeler module This module uses information from the activity specification and plan recognition module to update the user module.

Control module The control module constitutes the central part of the Virtual Secretary. All input and output (I/O) are organized in this module. The control module calls other modules when needed.

Task module 1 .. n These modules conduct the offered services from the Virtual Secretary.

The plan recognition module will only briefly be addressed in the first version of the Virtual Secretary.

4 The *Virtual Secretary* Security Approach

The Virtual Secretary approach is based on the construction of an environment for software agents, including mechanisms for propagation, authentication and control, and common user software. A major element is the propagation of non-executable authenticated user models. In section two, we discussed security problems related to agent propagation. In the Virtual Secretary, we use body processes to enable certification in advance. The body processes are willing to accept requests over the network and can act as bodies of worm-like programs, because the requests can be propagated further.

We envision two levels of propagation/communication:

1. Public data to a restricted anonymous Virtual Secretary account on the target host.
2. Personal data to the Virtual Secretary's principal's account on the target host.

Task	Level 1	Level 2
Read News	*	*
Web search	*	*
Read diary	(*)	*
Update diary	(*)	*
Read shared files	*	*
Read user (private) files	-	*
Read user (private) mail	-	*

Table 1. Tasks available at Level 1 and Level 2. * denotes generally available. (*) means available only if explicitly stated by the system manager.

An external user can access data at two levels as illustrated in Table 1. Level 1 is public data where a general user can communicate with the global Virtual Secretary daemon (vised) and access cite-specific information from World Wide Web, News, and the files that are readable for the vise group. It can also communicate with the personal diary services through a well-defined interface if need be. Level 2 gives access to private data. Such access is granted through the personal Virtual Secretary daemon that handles each user's mail, diary, files, and user model. This level requires mechanisms for authentication of the user, because it handles personal information that the user wants to be able to access from remote cites.

5 Status and Open Problems

The Virtual Secretary project is an ongoing project at the University of Tromsø. Currently, we focus on the system architecture, including security issues.

We argue that our approach to the construction of software agents, i.e., the transference of passive code and not active programs, the use of authentication and control mechanisms, and the provision of common user software, exclude the major security problems related to worm-like programs. However, still several crucial security-related issues are not worked out.

The propagation scheme is based on the assumption that a body process already is present on the remote host. But, we cannot always expect that this would be the case. Therefore, we need a secure way to propagate a Virtual Secretary body to a host not holding a body process. This problem could partly be viewed as a system management problem.

A related management problem is the initialization scheme – should the network of body processes be established initially or should this gradually be established upon request? We have chosen to start a body process on all computers, but this choice needs further investigation.

Occasionally, we must expect to release new versions of the Software Secretary body process. This calls for a secure way to update, or alternatively invalidate, a process on a remote host. If the host is a battery powered mobile computer equipped with wireless communication, one probably does not want to update the worm automatically.

Given that new releases do not exist in all part of the network, e.g., on a host which has been disconnected for some time. Then a problem would be how to deal with earlier releases, especially if security bugs have been fixed in the new release.

More general problems include the question of lifetime (immortalisation) and access control. Immortalisation involves whether a Virtual Secretary has a limited lifetime? And, if not, may the lifetime vary, e.g., according to the

characteristic of the mission. Could we, e.g., assume that a target host more likely will be contacted again within a short period.

Access control problems include: Who can update a Virtual Secretary; Who can be the owner (principal) of a Virtual Secretary; and, How can a Virtual Secretary be identified? These are all questions addressed, but not thoroughly discussed.

6 Concluding Remarks

Although software agents have been at focus in several decays, the basic idea is still to have active programs (i.e., worms) which propagate through the network. Worm programs represent a powerful, although very frightening, piece of software. The fundamental problem is to secure that the process one let into ones host does not act in an unintended way.

To deal with security issues, we have constructed an environment for software agents, including software propagation, authentication and control mechanisms, and common user software. By propagating a non-executable initialization and control software in the form of user models we expect to exclude the major security problems related to worm-like programs. It must be added that our approach is based on a restrictive assumption that a body process exists on the target host or, if not, a secure way to copy a body process to the target host exists. Occasionally, this may exclude our approach. However, we argue that the Virtual Secretary system approach heavily increases the potential use of worm-like programs in network environments.

Acknowledgements

The authors would like to thank Sape Mullender for comments on drafts of this paper.

References

- [1] FININ, T. GUMS – a general user modelling shell. In *User Models in Dialog Systems*, W. Wahlster and A. Kobsa, Eds. Springer-Verlag, Berlin, 1989.
- [2] RIECKEN, D. Intelligent Agents. *Communications of the ACM* 37, 7 (July 1994), 18–21.
- [3] STEINER, J. G., NEUMANN, B. G., AND SCHILLER, J. I. Kerberos: An Authentication System for Open Network Systems. In *Proc. of the Winter 1988 Usenix Conference* (February 1988), pp. 191–201.
- [4] WILLIGES, R. The use of models in human-computer interface design. *Ergonomics*, 30 (1987), 491–502.

A Secure System Architecture for Software Agents: The Virtual Secretary Approach

Gunnar Hartvigsen*
Arne Helme**
Stig Johansen*

Broadcast Technical Report ???

Software agents in the form of worm-like programs represent a great potential for location independent processing and problem solving. Unfortunately, worm-like programs in the traditional sense are, when traversing a computer network, among the most uncontrollable software constructions. This paper presents the Virtual Secretary system architecture which is an attempt to utilize the potential processing and problem solving strength of software agents while eliminating most of the security risks related to such programs. This is done by the construction of an environment for software agents, including propagation mechanisms, mechanisms for authentication and control, and common user software. A key element in this approach is the propagation of the Virtual Secretary's user model (i.e., a non-executable initialization and control software). A necessary assumption is that an authorised Virtual Secretary body process is present at all possible target hosts in the network, or if not, a secure way to start a new Virtual Secretary body process at a remote host exists.