Secure Mobility in a World-Wide Object Environment

Michael Condict (contact person),
Dejan Milojicic, Franklin Reynolds, Don Bolinger

{**condict**,**dejan**,**fdr**,**bolinger**}**@osf.org**

Open Software Foundation Research Institute
11 Cambridge Center
Cambridge, MA 02142

The phenomenal expansion of the Web to this point has been based on capabilities that in some ways are actually rather limited: predominantly read-only access from fixed clients to content stored on fixed servers. The advent of portable, safe programming languages like Java has opened new possibilities for full, read-write interaction between mobile objects and a client or server environment, but a wide range of security issues (among others) still impede the easy, widespread use of such objects. Of course, the CORBA OMA specifies a security service for distributed objects, and an RFP is outstanding for a Mobile Agent Facility. However, the interaction between these two remains unclear.

As a testbed for applying mobile objects in a secure manner, we intend to build, as a direct extension of today's Web, a portable, distributed, object-oriented environment, called the Internet System Environment (ISE). It will be CORBA-compliant, and will be based on globally-named distributed objects, optionally persistent. It will contain a hierarchical name service integrated with hierarchical object caching. To promote more effective use of scarce and unpredictable network bandwidth, it will support easily designed and composed custom network protocols.

Its real innovation, however, will be twofold. Since the ISE must be easily deployed, it will be implemented as portable middleware in Java, so will run in any Java-enabled computer, though the fully-functional version may require extensions to the Java runtime and the language itself. And since we see secure mobility on the network as essential, support for mobile objects and agents (MOA), and for the enhanced security they need, will be integral to the system.

MOA serve many useful purposes in the ISE, especially if they are "active," i.e. contain their own threads of control. Aside from the obvious example of electronic interchanges with mobile agents that travel the Web acting on behalf of users, MOA left behind in the network can help overcome the problems of mobile users who are infrequently connected. Further, MOA can reduce network traffic and improve its real-time behavior, because they move towards the source of communication, localizing it. And active mobile objects can clone themselves for the optimal traversal of large networks in parallel.

Both network and local-node security problems arise in using MOA. The network problems concern the need for privacy, the need to authenticate the identity of clients doing RPC and of objects migrating between nodes, and the need to authorize their actions easily in a distributed environment. The ISE will solve these problems in part with standard methods such as public key cryptography and digital signatures. This will allow, for example, a mobile object to pass directly from one user to another user that doesn't trust the first, without the overhead of contacting the server for the object. In addition, we will apply our Adage framework for distributed authorization to implement a distributed trust model, and to organize subjects into groups and roles to act within it.

The local-node security problems consist of authorizing access to local resources (using Adage, and given that we can be sure about the source of RPC's and mobile objects), and preventing untrusted code from damaging the system or acting as a Trojan horse. Currently, the Java environment addresses problems of this sort by providing a safe, object-oriented environment for programs, and using run-time checking to determine that compiled programs continue to meet safety constraints. Java asserts that locally generated code is trusted, while remote code is untrusted and faces severe restrictions on its actions. Despite this, the system is vulnerable to a variety of attacks. There is no support for a "trusted path" between user and code, no comprehensive mediation of access to different name spaces, and no protection from overconsumption of resources (denial of service).

We will limit the risk posed by untrusted code by executing it within a Secure Application Execution Framework (SAEF). This is a framework that ensures that users know when they are communicating locally with untrusted code. The framework has the responsibility for mediating access of untrusted code to all name spaces and enforces limits on usage of resources such as CPU cycles, memory usage, disk storage and I/O bandwidth. The framework also denies untrusted objects access to sensitive system services and information such as user identity.

Active mobile objects pose significant security problems beyond those inherent in passive distributed objects. With the latter, cache coherency does not require taking back a modified object from any site unless the site had the authority to modify the object. But a mobile object may return to the server from an untrusted site, or migrate from one client to another who does not trust the first. When this happens, the object, even if allegedly unmodified, must be checked for validity. If it is an active mobile object, it must be treated like any code downloaded from an untrusted site, and confined to execute within a SAEF.

We will use existing distributed application support, like DCE, where possible. But the "intranet" environment DCE was designed for, in which the default behavior of hosts, users and applications over time is assumed to be orderly and stable, maps poorly onto a world-wide environment. We also aim for CORBA interoperability, but hope to gain simplicity and ease of use from our concentration on a single programming language that can serve as an IDL as well as an application language. By using Java specifically, we intend to promote easy, widespread deployment. The goal is to gain experience with mobile objects in a timely fashion, while retaining relevance to the CORBA community.