Trust-Based Security in Pervasive Computing Environments

Lalana Kagal, Tim Finin, and Anupam Joshi
University of Maryland, Baltimore County

Traditionally, stand-alone computers and small networks rely on user authentication and access control to provide security. These physical methods use system-based controls to verify the identity of a person or process, explicitly enabling or restricting the ability to use, change, or view a computer resource.

However, these strategies are inadequate for the increased flexibility that distributed networks such as the Internet and pervasive computing environments require because such systems lack central control and their users are not all predetermined. Mobile users expect to access locally hosted resources and services anytime and anywhere, leading to serious security risks and access control problems.

We propose a solution based on trust management that involves developing a security policy, assigning credentials to entities, verifying that the credentials fulfill the policy, delegating trust to third parties, and reasoning about users' access rights. This architecture is generally applicable to distributed systems but geared toward pervasive computing environments.

PERVASIVE COMPUTING

Pervasive computing strives to simplify day-to-day life by providing mobile users with the means to carry out personal and business tasks via portable and embedded devices. These tasks range from the simple—switching on the lights in a conference room, checking e-mail, and organizing meetings—to the more complex—such as booking airline tickets, buying and selling stock, or managing bank accounts.

Pervasive computing environments require a security architecture based on trust rather than just user authentication and access control.

Pervasive computing environments of the near future will involve the interaction, coordination, and cooperation of numerous, casually accessible, and often invisible computing devices. As Figure 1 shows, these devices—whether carried on our person or located in our homes, businesses, and classrooms—will connect via wired and wireless links to one another as well as to the global networking infrastructure to provide more relevant information and integrated services.

The eBiquity Research Group (http://research.ebiquity.org) at the University of Maryland, Baltimore County, is designing pervasive computing systems composed of autonomous, intelligent, self-describing, and interacting components. SmartSpaces are instances of pervasive systems in which the domain is divided into a hierarchy of spaces with a controller managing the services in each space.

Centaurus is a framework for SmartSpaces that includes a message-based transport protocol designed to perform well in low-bandwidth networks and with resource-poor devices. We use this protocol in the Smart Office scenario, in which mobile users access computers, fax machines, printers, the lights, and even such mundane appliances as the coffee maker via handheld devices connected over short-range Bluetooth wireless links.

SECURITY CHALLENGES

Adding security to such open models presents challenges at many levels. How do you decide whether a person who does not work in an office but has access to it—for example, as a consultant or member of a partner firm—can use certain services?

We encountered several problems with providing security in environments using the Centaurus protocol. Having a central authority for a single building or even a group of rooms is infeasible because every possible access right will have to be specified for every user. Authenticating the identity certificate of a previously unknown user doesn't provide any access control information. Simple authentication and access control are only effective if the system knows in advance which users are going to access a Smart Room and what their access rights are.

Portable handheld and embedded devices have severely limited processing power, memory capacities, software support, and bandwidth characteristics. Also, hardware and software environments are becoming increasingly heterogeneous, a trend which will continue in the foreseeable future. Finally, security information in different domains is sub-

ject to inconsistent interpretations in such an open, distributed environment.

DISTRIBUTED TRUST

To satisfy the requirements of the pervasive computing model, we suggest adding *distributed trust* to the security infrastructure.

Dynamic rights

We view trust management as establishing trust relationships instead of its traditional meaning of quantifying trust. Our approach involves

- articulating policies for user authentication, access control, and delegation;
- assigning security credentials to individuals;
- allowing entities to modify access rights of other entities by delegating or deferring their access rights to third parties and revoking rights as well; and
- providing access control by checking if the initiators' credentials fulfill the policies.

Access rights are not static but change based on delegations and revocations. Users are assigned generic rights—based on their credentials, the security policy, and other users' delegations—that can be used to request access to other services. Appropriate users with these access rights can in turn delegate the requested right. Users can access a service only if they have the right to do so or if an authorized user has delegated that right to them; they can delegate all rights that they have the permission to delegate. Rights can likewise be revoked.

Rights can be associated with devices and agents as well as users: A software agent could have the right to use a certain service, or a service could have the right to use another service.

Models

Well-known distributed trust models include the simple public key infrastructure, Pretty Good Privacy, and Matt Blaze's PolicyMaker. SPKI is used for authentication and authorization but only includes a simple notion of delegation. In PGP, an entity is trusted when one or more trusted entities say that it can be trusted. Both of these schemes suffer from key distribution problems and do not deal with flexible or scalable access control.

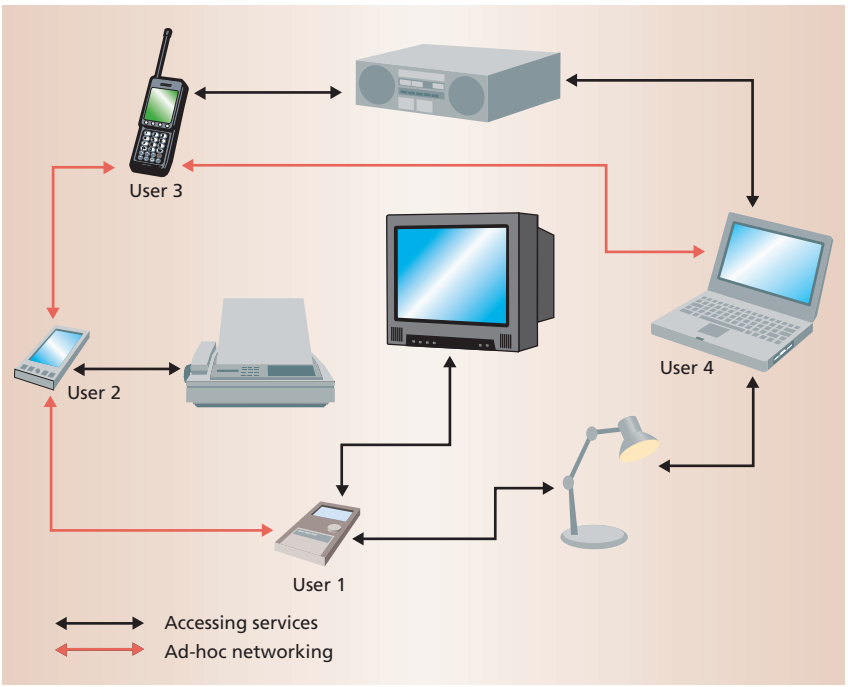

Figure 1. Pervasive computing. In the near future, mobile users will be able to access information and integrated services via hand-held devices.

Blaze, who coined the term *distributed trust management,* developed PolicyMaker, which binds public keys to access control without authentication. Although PolicyMaker is a powerful analytical tool, the nonprogrammers who are likely to develop policies may have difficulty expressing policies in this system. Also, it a query engine that answers questions about access rights to a given policy rather than a true security infrastructure.

Notions similar to delegation such as copy/copy propagation have been used even in operating systems, but they generally deal with a user domain in which all users are known in advance; such an assumption can't be made in a pervasive computing scenario.

TRUST ARCHITECTURE FOR PERVASIVE SYSTEMS

We have designed a policy-based framework that extends SPKI and role-based access control.

Distributed model

A security policy is a set of rules for authorization, access control, and trust in a certain domain; it can also contain information about some users' roles and the abilities associated with those roles. Each domain has security agents that enforce the policy.

The domain's services and users can additionally impose a local policy. Services register with a security agent in their space and rely on it to provide security.

A user is generally associated with a certain role in the system and assigned role-based axiomatic rights. This role can change based on the policy or user's actions.

Centaurus uses a distributed model in which hierarchically arranged security agents manage security and trust, and X.509 authentication certificates identify users and services. Authorized users can make delegations and revocations in the form of signed assertions. Security agents reason about these signed assertions and the appropriate security policies to provide access control to services in their domain.

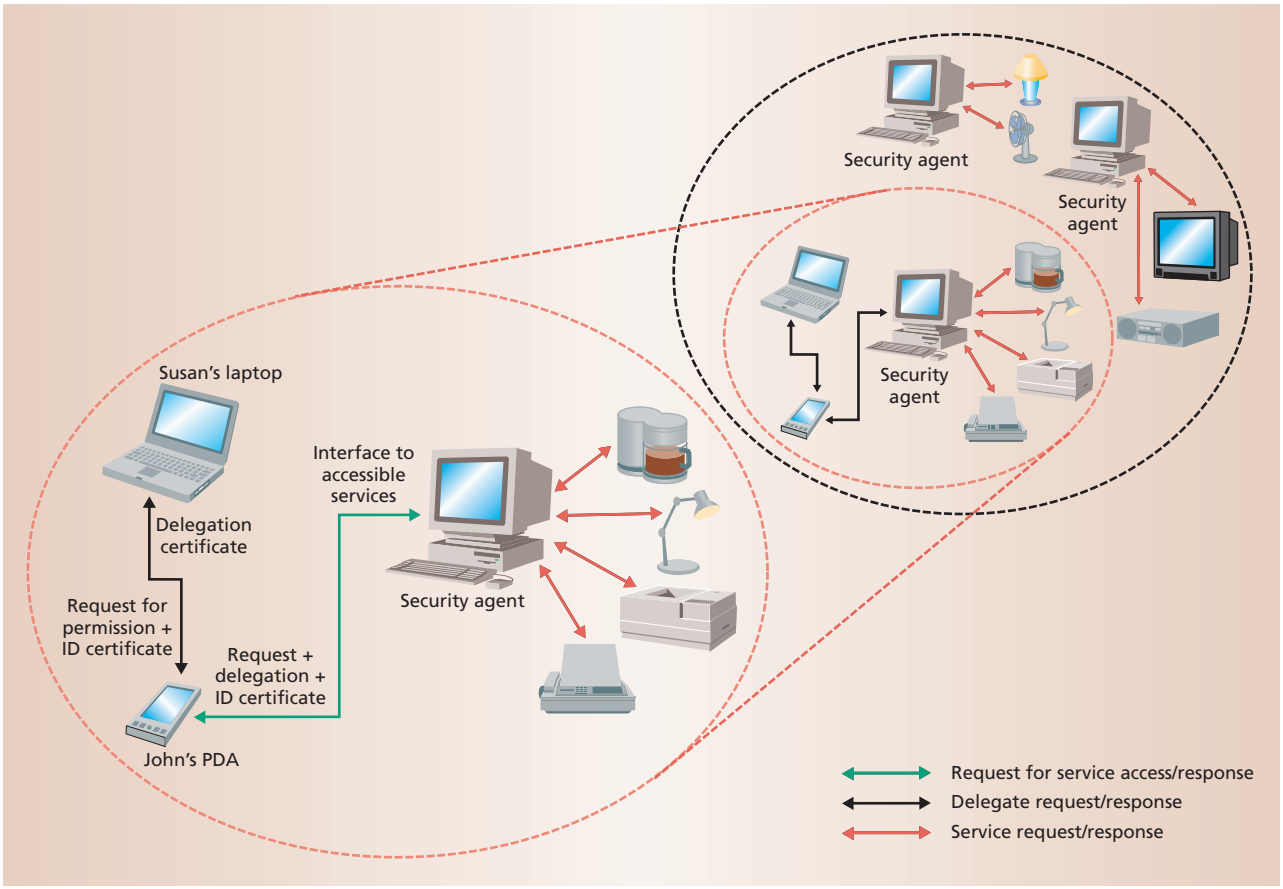

Figure 2. Trust in pervasive computing environments. John requests Susan for access to various services. Susan sends back a delegation certificate that John sends to the security agent. The security verifies the certificate and, because Susan is trusted, allows John to access the services.

Delegation chain

Centaurus views delegation as a right itself. Only users with the right to delegate a certain action can actually delegate that action, and the ability to delegate itself can be delegated. Users can constrain delegations by specifying whether delegated users can re-delegate the right and to whom they can delegate. Once users are given certain rights, they are responsible for the actions of the users to whom they subsequently delegate those rights and privileges.

This forms a delegation chain in which users only delegate to other users that they trust. If any user along this delegation chain fails to meet the requirements associated with a delegated right, the chain is broken. Following the failure, no user can perform the action associated with the right.

When users make requests to the security agent controlling the service, they attach their credentials—an ID certificate or a delegation certificate—to the request. Security agents may generate authorization certificates that users can employ as tickets to access a certain resource.

As Figure 2 shows, one user, John, can also ask another user, Susan, to delegate to him the right to access certain services. If Susan is satisfied with John's credentials, she will send back a signed statement containing the delegation, possibly with constraints attached—for example, one that limits access to a certain period or persons to whom John can re-delegate the right. The security agent is responsible for honoring the delegation, based on the delegator's and delegatee's credentials and the policies.

Ontologies

Our work is similar to role-based access control—an approach in which access decisions are based on the roles that individual users have as part of an organization, such as doctor, nurse, manager, or student—in that a user's access rights are computed from its properties.

Our approach, however, uses ontologies that include not just role hierarchies but any properties and constraints expressed in an XML-based language, including elements of both description logics and declarative rules. For example, a rule could specify that any user in a meeting room who is operating the projector during a presentation is probably the presenter and should thus be allowed to use the computer as well. In this way, rights can be assigned dynamically to users without creating a new role.

PERVASIVE COMPUTING SCENARIO

Consider the following example. John is an employee of one of the office's partners, but the security agent in the office doesn't understand his role in the organization, so it denies him access to the Smart Room services. John requests permission from Susan, one of the managers, to use the services. According to the office's security policy, Susan can delegate access rights to anyone she trusts. Therefore, she delegates to John the right to use the lights, coffee maker, and printer—but not the fax machine—for a short period of time.

Susan's laptop sends a short-lived signed delegation to John's handheld device. When John enters the Smart Room, the client on his handheld device sends his identity certificate and the delegation to the service manager. Because Susan is trusted and can delegate access rights, the delegation conforms to the policy and John now has access to the lights, coffee maker, and printer. Once the delegation expires, John must ask Susan for another delegation to access services in the room.

This scenario demonstrates the importance of trust over traditional security mechanisms in a pervasive computing environment. The system allows John, a foreign user, to access certain services without creating a new identity for him or insecurely opening up the system in any way.

We are working on integrating trust into the security infrastructure for Centaurus, which currently only provides authentication and access control for known users. We believe that trust will add a new dimension to pervasive computing, allowing greater flexibility in designing policies and providing more control over accessing services and information. We are also improving our trust architecture by extending Centaurus to include entitlements, prohibitions, and obligations and the ability to delegate them.

To protect the privacy of users who do not want the system to log their names and actions, we are replacing X.509 certificates with XML signatures (http://www.w3.org/signature) from a trusted authority that do not include the bearer's identity, but only a role or designation.

In our past work on distributed trust, we encoded actions, privileges, delegations, and security as horn clauses in Prolog. To develop an approach better suited to sharing information in an open environment, we are recasting this work in the DARPA Agent Markup Language. Built on XML and the Resource Description Framework, DAML provides a description logic language for defining and using ontologies on the Web in machine-readable form. In applying our framework, we are extending the initial ontology (http://daml.umbc.edu/ontologies/trust-ont.daml) by defining domain-specific classes for actions, roles, and privileges and creating appropriate instances. ✷

Acknowledgments

This research was supported in part by the IBM EECOMS program and by the DARPA DAML program under contract F30602-97-1-0215, NSF CCR0070802, 1159875433.

Lalana Kagal *is a PhD student in the Computer Science and Electrical Engineering Department at the University of Maryland, Baltimore County. Contact her at lkagal1@cs.umbc.edu.*

Tim Finin *is a professor in the Computer Science and Electrical Engineering Department at the University of Maryland, Baltimore County, and director of the Institute for Global Electronic Commerce. Contact him at finin@cs.umbc.edu.*

Anupam Joshi *is an associate professor in the Computer Science and Electrical Engineering Department at the University of Maryland, Baltimore County. Contact him at joshi@cs.umbc.edu.*

Editor: Upkar Varshney, Department of CIS, Georgia State University, Atlanta, GA 30002-4015; voice +1 404 463 9139; fax +1 404 651 3842; uvarshney@gsu.edu